A Formalised Ontology for Network Attack Classification

Submitted in fulfilment
of the requirements of the degree of

Doctor of Philosophy

of Rhodes University

Renier Pelser van Heerden

Grahamstown, South Africa
April 2014

Abstract

One of the most popular attack vectors against computers are their network connections. Attacks on computers through their networks are commonplace and have various levels of complexity. This research formally describes network-based computer attacks in the form of a story, formally and within an ontology. The ontology categorises network attacks where attack scenarios are the focal class. This class consists of: Denial-of-Service, Industrial Espionage, Web Defacement, Unauthorised Data Access, Financial Theft, Industrial Sabotage, Cyber-Warfare, Resource Theft, System Compromise, and Runaway Malware. This ontology was developed by building a taxonomy and a temporal network attack model. Network attack instances (also know as individuals) are classified according to their respective attack scenarios, with the use of an automated reasoner within the ontology. The automated reasoner deductions are verified formally; and via the automated reasoner, a relaxed set of scenarios is determined, which is relevant in a near real-time environment. A prototype system (called Aeneas) was developed to classify network-based attacks. Aeneas integrates the sensors into a detection system that can classify network attacks in a near real-time environment. To verify the ontology and the prototype Aeneas, a virtual test bed was developed in which network-based attacks were generated to verify the detection system. Aeneas was able to detect incoming attacks and classify them according to their scenario. The novel part of this research is the attack scenarios that are described in the form of a story, as well as formally and in an ontology. The ontology is used in a novel way to determine to which class attack instances belong and how the network attack ontology is affected in a near real-time environment.

Acknowledgements

Firstly I would like to thank my wife Lizelle without whose support I would not have been able to complete this work. Lizelle supported me through many long nights and weekends. I would like to thank my supervisor Prof Barry Irwin for his leadership and patience in developing the ideas that are presented in this work. Along with Prof Irwin, my secondary supervisor and colleague Dr Louise Leenen's constant encouragement and feedback were essential during the research process.

I would like to thank my colleagues at the CSIR who contributed to the prototype development: Ivan Burke, Shaun Egan and Heloise Pieterse, and the CSIR management, Joey Jansen van Vuuren and Dr Jackie Phahlamohlaka, who supported this research. Finally my thanks to Jan Gutter and Aré van Schalkwyk for proofreading this thesis. This research was completed with funding support from a CSIR Parliamentary Grant.

ACM Computing Classification System Classification

- Security and privacy
  - Intrusion/anomaly detection and malware mitigation
    - Malware and its mitigation
    - Intrusion detection systems
  - Network security
  - Systems security
  - Formal methods and theory of security
    - Formal security models
    - Logic and verification
- Information systems
  - World Wide Web
    - Web data description languages
      - Web Ontology Language (OWL)
  - Information retrieval
    - Document representation
      - Ontologies
- Computing methodologies
  - Artificial intelligence
    - Knowledge representation and reasoning
      - Description logics
      - Ontology engineering

The 2012 ACM Computing Classification System was used[1].

[1] http://www.acm.org/about/class/2012

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF CODE LISTINGS

LIST OF ACRONYMS

ACK Acknowledge

AM Attack Mechanism

AS Attack Scenario

Aeneas Attack Estimation Network Evaluation Architecture System

API Application Program Interface

ARP Address Resolution Protocol

ARPANET Advanced Research Projects Agency Network

CAPEC Common Attack Pattern Enumeration and Classification

CD Compact Disk

CEE Common Event Expression

CERT Computer Emergency Response Team

CERT-In The Indian Computer Emergency Response Team

CGI Computer Generated Imagery

CIA Confidentiality, Integrity and Availability

CIA+ Confidentiality, Integrity and Availability Authentication

CIS Central Information Server

CORE Common Open Research Emulator

CVE Common Vulnerability and Exposure

CPU Central Processing Unit

CW Cyber-Warfare

DARPA Defense Advanced Research Projects Agency

DDoS Distributed Denial-of-Service

DMZ Demilitarised Zone

DNS Domain Name System

DoS Denial-of-Service

EQ Event Query

FBI Federal Bureau of Investigation

FIRE Fuzzy Intrusion Recognition Engine

FT Financial Theft

FTP File Transfer Protocol

GB Gigabyte

Gbps Gigabits per second

GHz Gigahertz

GloMoSim Global Mobile Information System Simulator

GUI Graphic User Interface

HIDS Host-based Intrusion Detection System

HTTP HyperText Transfer Protocol

Hz Hertz

IANA Internet Assigned Numbers Authority

ICMP Internet Control Message Protocol

IDES Intrusion Detection Expert System

IDS Intrusion Detection System

IE Industrial Espionage

IO Input and Output

IP Internet Protocol

IPv4 Internet Protocol version Four

IS Industrial Sabotage

LSASS Local Security Authority Subsystem Service

MAC Media Access Control

NIC Network Interface Card

NIDS Network-based Intrusion Detection System

NHS Network HTTP Simulator

NRL Naval Research Laboratory

NSA National Security Agency

OS Operating System

OSI Open Systems Interconnection

OWL Web Ontology Language

PETA People for the Ethical Treatment of Animals

PC Personal Computer

PLC Programmable Logic Control

RAID Redundant Array of Independent Disks

RINSE Real-time Immersion Network Simulation Environment for Network Security Exercises

SADT Structured Analysis and Design Technique

SC System Compromise

SCADA Supervisory Control and Data Acquisition

SCF Standard Communications Framework

SIEM Security Information and Event Management

SMTP Simple Mail Transfer Protocol

SQL Structured Query Language

SSH Secure Shell

SSL Secure Sockets Layer

SYN Synchronise

TB Terabyte

TCP Transmission Control Protocol

TIP Threat Identification Prototype

US United States

UDA Unauthorised Data Access

UDP User Datagram Protocol

UDON User-defined and Organised Network

URL Uniform Resource Locator

US CERT United States Computer Emergency Readiness Team

USB Universal Serial Bus

Velnet Virtual Environment for Learning Networking

W3C World Wide Web Consortium

WD Web Defacement

XML Extensible Markup Language

XSS Cross-site Scripting

Part I

Introduction

CHAPTER

ONE

INTRODUCTION

"... a science must deal with a subject and its properties."

Aristotle, 340 BC

Computers evolved from stand-alone systems, such as mainframes, to personal computers and, more recently, smart phones that are permanently connected to the Internet. In 1969, with the establishment of the United States (US) Advanced Research Projects Agency Network (ARPANET), computers started to communicate over long distances. ARPANET started a process of making networking an integral part of computing (Leiner, Cerf, Clark, Kahn, Kleinrock, Lynch, Postel, Roberts, and Wolff, 2009). The concept was already expressed in 1984 by John Gage[1] in his famous phrase: "the network is the computer" (Thornburg, 2009). With the emergence of the Internet, networking has grown to a level where computers can instantly communicate over the whole world. Networked computers have become standard for most companies and other technology using entities. Computers are connected via Intranets and connected to the rest of the world though the global Internet. A computer network is defined as a single network, including computers, servers, network equipment and other related devices, that form part of a single entity and that are connected to the Internet.

Attacks on computers have also evolved and the Internet is now used as an attack path as well as in attacks on the network infrastructure itself. Modern computer networks

[1] Chief Researcher and Vice President of the Science Office for Sun Inc. 1982-2008

connect to the Internet, and are thus rendered accessible to anyone in the world. The Internet presents four novel advantages to any attacker:

The first is that an aggressor can initiate a network-based attack with a high level of anonymity ensured. An aggressor can hide by forging the origin of an attack or use intermediates in an attack (Lee and Shields, 2001). The aggressor can thus potentially attack any computer-based network from any place, at any location in the world, thus making the source of the attack virtually untraceable (Baba and Matsuda, 2002; Choo, 2008).

The second is that the target of a computer-based attack is usually online, thus an attack cannot be prevented or mitigated by removing the target from the attack vector, i.e. taking it offline. Sorin (2008) states that website and server uptime are directly related to profit, and system outages may also lead to future losses from visitors who will not return to a site if it has been offline or inaccessible. In the movie *The Social Network*[2], the Mark Zuckerberg character states how important uptime is (Mezirch, 2009; Sorkin, 2010):

> "Okay, let me tell you the difference between Facebook and everyone else, we don't crash EVER! If those servers are down for even a day, our entire reputation is irreversibly destroyed! Users are fickle, Friendster has proved that. Even a few people leaving would reverberate through the entire userbase. The users are interconnected, that is the whole point. College kids are online because their friends are online, and if one domino goes, the other dominoes go, don't you get that."

This quote indicates how important uptime is for Facebook. Thus, removing a target's ability to communicate on the Internet is also a major attack methodology or goal.

The third advantage is that any computer network of significant size has a high probability of security holes existing in its design or implementation (Geer, 2007; Perrow, 2008). Although Shin and Williams (2008) only found weak evidence that software complexity is directly related to vulnerabilities, they still concluded that vulnerable code is more complex than other code. Harter, Kemerer, and Slaughter (2012) state that the size of the code is a key factor that influences the quality of software, and that increasing the complexity of software adds more flaws.

The fourth and final advantage is that many users of a computer network are not computer-security literate, and unaware of how their ignorance can be exploited. These users render

[2] http://www.imdb.com/title/tt1285016/

a whole network vulnerable (Ying, Dinglong, Haiyi, and Rau, 2007; Kritzinger and von Solms, 2010). Rescorla (2003, 2005) determined that administrators are on average slow to apply patches that fix security holes and postulated that it might not be economically worthwhile for security researchers to find and expose security holes (Rescorla, 2005):

> "...we cannot conclude that vulnerability finding and disclosure provides an increase in software security sufficient to offset the effort being invested."

Choo (2008) and Shahzad, Shafiq, and Liu (2012) states that organisations generally do not patch their systems with the same frequency as security vulnerabilities are made public and that laxness in security is often used to compromise systems. Arjun (2012) noted that people contributed to making networks vulnerable, and that if there is a choice between usability and security, people tend to choose usability.

1.1 Problem Statement

Attacks on computer networks have become so commonplace that it has become a fact of life (Choo, 2011). With the rise of complexity of computer networks, the attacks on them have also become more complex and varied. Attacks on a computer network can differ significantly. One of the popular attacks is a Denial-of-Service (DoS), which differs completely from defacing a website or stealing secrets from a networked computer. The goal of this research is to formally define network attacks.

The secondary goal of this research is to investigate how the definition of network attacks differs in the near real-time environments. Verizon[3] found that the average time for businesses and corporations to determine if they have been breached is seven months (Gliddon, 2012; Verizon RISK Team, 2012), with compromises typically being much longer.

The scope of this research will be limited to attacks on computer networks in the cybersphere, occurring through computer network connections, such as Universal Serial Bus (USB) flash drives. The target of the attacks are computer networks themselves and their component parts. Social engineering attacks such as Spam, Spear Phishing and physical attacks also fall outside the scope of this thesis, although social engineering is included in the taxonomy for completeness.

The specific research questions are:

[3] http://www.verizon.com/

- What are the different types of computer network attacks?
- How are such computer network attacks defined?
- What is the impact on attacks in a near real-time environment?

The answer to the first research question is to be evaluated by considering how well computer network attacks are classified. The second question will be addressed when networks attacks are formally described and modelled. The last question refers to how attacks are impacted when they are only evaluated in near real-time and the question can be validated by prototyping a network attack prediction system and empirically verifying the impact of attacks in near a real-time environment.

1.2 Research Method

The research questions raised in this thesis were addressed using four research methodologies: formal (analytical), experimental (empirical), build (engineering) and model (scientific). These methodologies are described by (Glass, 1995; Elio, Hoover, Nikolaidis, Salavatipour, Stewart, and Wong, 2005). **Models** were developed to enhance the understanding of all the factors that influence a network attack. One model described the temporal aspects of attacks, and the other model listed all the factors in the form of a taxonomy. The relationships between the classes of a taxonomy were **formally** specified in the form of an ontology in order to enable the verification of the software implementation. The model and formal description were verified by **building** a software artefact, in which the concepts were tested **experimentally**. The experiments proved that the conceptual framework is viable.

1.3 Document Structure

The body of this document consists of three parts that are structured as follows:

- **Part I** contains the introduction, a history of computing attacks and a literature study. This part supplies the background information on network attacks and investigates related work in this field.
  - **Chapter 2** investigates significant historical computer-based attacks. These attacks are discussed and a list of attack scenarios are constructed.

  - **Chapter 3** presents a literature study of related work. Models, taxonomies, ontologies and sensors related to network attacks are investigated.

- **Part II** contains the bulk of the work, namely the development of the temporal model, taxonomy, ontology and formal descriptions thereof.
  - **Chapter 4** develops a taxonomy that presents the classes of a computer network-based attack from both the point of view of an attacker and defender. A temporal model of network-based attacks is also presented.
  - **Chapter 5** presents an ontology that describes the relationships between the taxonomy classes formally. The ontology is presented in the form of a scenario, formally and via a software implementation. The *Denial-of-Service* scenario is presented in detail.
  - **Chapter 6** describes in detail the ontology representation of each of the remaining attack scenarios, in story form, formally and as an individual within the Protégé implementation.
- **Part III** describes the impact of the attack scenarios in near real-time. The impact is formally explored and a reduced set of scenarios are determined.
  - **Chapter 7** investigates the quantification and possible measurements of the classes on the taxonomy in near real-time. A determination of which attack scenarios can be quantified in near real-time is made. The *Denial-of-Service* and *Cyber-Warfare* scenarios quantification in near real-time are presented.
  - **Chapter 8** demonstrates how to identify attack scenarios by mapping sensors' outputs to the temporal attack model and attack scenarios. A prototype system called Aeneas is presented in this chapter. The Aeneas system classifies network attacks with respect to their related scenario and phase.
  - **Chapter 9** presents the environment in which the prototype system was verified. Two environments are presented: visualised systems with a firewall connected to the Internet and visualised systems within an Internet simulator. The validation of the prototype is presented. The Aeneas system is validated by performing a range of simple tests.

The document concludes with **Chapter 10**, which reflects on the research project and on possible future work. Supplemental information is supplied in the form of appendices. These appendices contain detailed information, and are referred to where applicable within the main text.

1.4 Research Output

This section lists the main research output of this thesis. Published conference and journal outputs arising from this research are:

1. **Grant, T., Burke, I., and van Heerden, R. P.** *Comparing models of offensive cyber operations.* In *Proceedings of the 7th International Conference on Information-Warfare & Security (ICIW 2012)*, pages 108–121. ACI, 2012

2. **van Heerden, R. P., Burke, I., and Irwin, B.** *Classifying network attack scenarios using an ontology.* In *Proceedings of the 7th International Conference on Information-Warfare & Security (ICIW 2012)*, pages 311–324. ACI, 2012b

3. **van Heerden, R. P., Pieterse, H., and Irwin, B.** *Mapping the most significant computer hacking events to a temporal computer attack model.* In *International Conference on Human Choice and Computers (HCC10): ICT Critical Infrastructures and Society*, pages 226–236. IFIP, Springer, 2012c

4. **van Heerden, R., Leenen, L., Irwin, B., and Burke, I.** *A computer network attack taxonomy and ontology. International Journal of Cyber Warfare and Terrorism*, 3:12–25, 2012a

5. **van Heerden, R., Leenen, L., and Irwin, B.** *Using an automated reasoner to classify computer network attacks.* In *5th Workshop on ICT Uses in Warfare and the Safeguarding of Peace.* November 2013a

6. **van Heerden, R., Pieterse, H., Burke, I., and Irwin, B.** *Developing a virtualised testbed environment in preparation for testing of network based attacks.* In *5th Workshop on ICT Uses in Warfare and the Safeguarding of Peace.* November 2013b

1.5 Document Conventions

In this document, some non-standard conventions are used. These include the term *near real-time* and the presentations of taxonomies. The main sub-concept is that the information presented in near real-time must be available timeously. For example, near real-time for a meteorologist is measured in hours and minutes, while near real-time for

a telecommunications systems would be in milliseconds. For the purpose of this thesis, near real-time is defined as within 60 seconds.

When referring to a website, the Uniform Resource Locator (URL) is listed as a footnote. This ensures that relevant information is available directly without obstructing the reader's flow.

Taxonomies, models and the ontology are presented in illustrations. By presenting them as figures, the reader can surmise the information visually, without having to read through long, detailed lists. The level of each class in the taxonomy is presented in a different colour and the various colours are associated with each level throughout this document.

Figure 1.1 illustrates five levels.

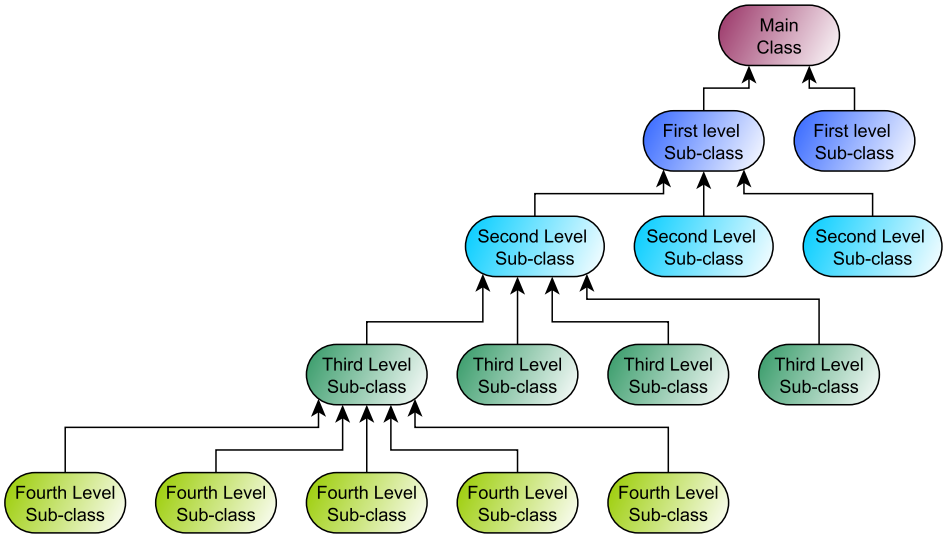


Figure 1.1: Taxonomy Class Level

When describing a logical set, the term *class* is used and individuals are members of these sets. For example, the *Denial-of-Service* scenario class is a set with *SCO Attack* and *SpamHaus Attack* individuals belonging to this set. These individuals are presented in Section 5.3.1. The description of the ontology as a story in Chapter 5 is not intended to produce grammatically perfect text, but rather a flexible skeleton which can be further developed. The first letters of the classes of the taxonomy and ontology are capitalised, whereas the first letters of the relationships are lower case.

The mathematical symbols used in this thesis are listed below:

$\exists$ *There exists at least one element*

$\ni$ *Such that*
$\in$ *Element of*
$\wedge$ *Logical And*
$\vee$ *Logical Or*
$\subseteq$ *Subset of*
$\times$ *Cartesian Product (Relation)*
$\circ$ *Composition*
$\equiv$ *Equivalent*
$\cup$ *Union*

Throughout the thesis, the DoS attack example is used to illustrate the principles that are developed in each chapter. The DoS attack is also formally presented in the main thesis in Section 5.4.4, whereas the other scenarios are presented formally in Chapter 6.

CHAPTER

TWO

SIGNIFICANT HISTORICAL COMPUTER PLATFORM ATTACKS

"You will never find a more wretched hive of scum and villainy."

Obi-Wan Kenobi – *Star Wars*

2.1 Introduction

This chapter presents a collection of the most significant attacks on computer platforms. This collection is the subjective view of the researcher, but is in part based on similar lists of other authors. It is a list of significant network attacks and it provides a large spectrum of different attacks, making it possible to identify network attack scenarios. The attack scenarios are used as a means of illustrating the Network Attack Taxonomy described in Chapter 4 and form the base of the ontology developed in Chapter 5.

This chapter begins with a literature study of similar lists compiled by other authors. These lists are also subjective, but provide a good starting point in compiling a list of the most significant network attacks. In Section 2.3, the criteria that were used to determine which computer attacks were significant are described. In Section 2.4 each significant computer attack is described in detail and in Section 2.5 attack scenarios are derived from the selection of attacks. In Section 2.6, the attacks are sorted according to four of the criteria mentioned in Section 2.3. Section 2.7 summarises the identified attacks.

2.2 Literature Survey

All lists that measure the significance of computer attacks are subjective. When one examines a collection of these lists, the most popular events can be extracted and used as a base for significant attacks. This base was augmented by computer attacks found to be significant by the researcher. In Appendix A, the complete lists of significant attacks are listed.

2.2.1 Malware Lists

Heater (2011), Miranda (2010) and Strickland (2008) made lists of significant malware. Heater listed malware that in his opinion best enabled insight into the development of malware, and concentrated on malware that targets novel aspects of computers, such as malware in shareware, email, databases, cellphones, social networks and the Internet. Heater also explicitly stated that his list is not definitive, but rather informative. Miranda examined the malware that had the biggest financial impact, and discovered that malware can cost millions in economic productivity. Strickland concentrated on malicious malware that can cripple computers or networks, and he noted that when the computer industry was in its infancy, systems were being sabotaged, but it took a few decades for hackers to start coding computer viruses. As early as 1949, John von Neumann developed a theory on the possibility of a self-replicating programme (Neumann and Burks, 1966).

2.2.2 Cyber-Attacks

Hall (2010), Julian (2011), Liddinton-Cox (2012) and (Tech Analyser, 2011) made lists of cyber-attacks. Hall listed seven major cyber-warfare events and noted that attacks are only reported after the fact, and that cyber-attacks can potentially target government, banking or military networks and affect vital data and funds, or can inflict physical damage. Liddinton-Cox listed hackers that had either achieved fame or were charged with felonies. The individuals Michael Calce (Mafiaboy), Kevin Mitnick and Adrian Lamo are personally listed, while the rest of his list concentrates on events rather than persons. *Tech Analyser* Online magazine explored some of the most notorious cyber-attacks (Tech Analyser, 2011). They examined how cyber-attacks have evolved from the initial basic spreading through floppy disks to the modern use of removable media or spreading malware via the Internet.

Julian (2011) listed ten of the costliest cyber-attacks. The claims of which attacks were the costliest cannot be verified, but the attacks themselves can all be considered significant. The list presented by Miranda differs from that of Julian, and both lists claim to present the most costly computer attacks. According to Leeson and Coyne (2005), financial institutions will most likely not report hacker-related incidents, fearing the effect this will have on their customers, as well as on stockholders' impressions of their security. Damage amounting to approximately $55 billion was caused by malware in 2002, according to Leeson and Coyne. It is impossible to measure the precise amount of damage done.

2.2.3 Cyber-Crime Lists

Marcus (2011), Buckland (2011), Poulsen (2009) and Armerding (2012) made lists of significant cyber-crimes. Marcus from McAfee discussed the last decade's (2000–2010) most serious cyber-related attacks, and found that cyber-crime has reached new levels of maturity, and that targeted attacks against governments and large organisations are becoming commonplace. The scope of attacks has also reached unprecedented levels. McAfee detected an average of 60 000 new pieces of malware each day in 2010, and malware directed at social media is one of the fastest-growing threats (Marcus, 2011).

Buckland from Microsoft published his list of the worst cyber-crimes of the 2000s. The crimes include malware creation, scams, hacking, credit card number theft, phishing and disclosure of secret information. Computer networks were used as tools or were the target of all the crimes. A similar list was developed by Poulsen (2009). Poulsen's list examined "the most ingenious, destructive or ground-breaking cyber-crimes of the first ten years of the new millennium". Only four entries appear on both the Buckland and Poulsen lists. Even the dates used to specify when some of the cyber-crimes occurred differ in the lists. These two lists indicate why such lists are considered to be subjective.

2.2.4 Data Leaks

Armerding (2012) listed 15 of the largest data breaches recorded, which demonstrates how diverse and widespread data breaches have become. Information from technology giants such as Google and even security companies such as RSA Security Inc. have been leaked. Since Armerding's list, two major data breaches have occurred: Bradley Manning gave over 260 000 sensitive diplomatic cables to Wikileaks[1] (Sweetman, 2011;

[1] http://wikileaks.org/

Jones, 2013) and Edward Snowden supplied the leaked information about the National Security Agency (NSA) worldwide covert data collection programme (Richelson, 2013).

2.3 Criteria and Timeline

Bases on the works described above, a timeline was developed of the major computer network attacks. These attacks are considered to be the most significant by the researcher as they had an impact in one of the following areas:

- first use of a particular attacking methodology
- first use of a new class of attack
- significant financial impact
- widespread geographical impact
- level of sophistication
- attacks considered to be significant by other authors
- famous hackers

Figure 2.1 illustrates the computer network attacks that the researcher considered to be the most significant. The attacks are divided into five categories, namely Infamous Hackers, Viruses and Trojans, Worms, Commercial Attacks and Cyber-War. Viruses were initially very significant. As the Internet became more available, viruses gave way to worms. In recent times, commercial attacks and cyber-warfare attacks have become more prominent. Figure 2.1 illustrates that the significant network attacks have moved from viruses in the 1980s and 1990s to worms in the early 2000s. From 2008, commercial attacks and cyber-warfare have become much more prevalent. These incidents are discussed further in Section 2.4.

2.4 Significant Computer Attacks Survey

The attacks listed in Section 2.2 that were chosen by three or more authors are listed in Table 2.1. The third column indicates the number of times a particular attack was chosen to be significant.

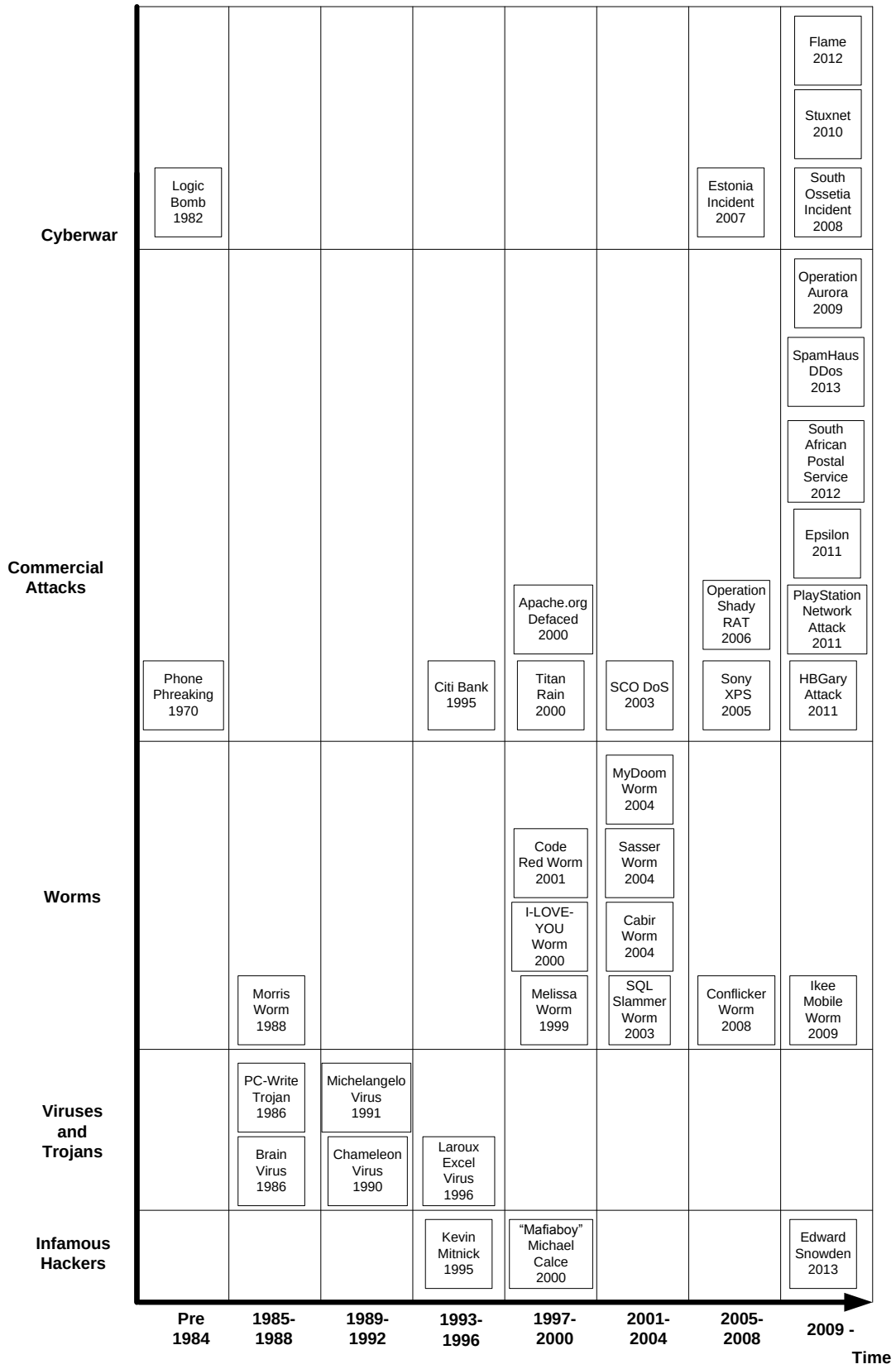


Figure 2.1: Timeline of Significant Attacks

Table 2.1: Attacks Most Frequently Listed

Date	**Attack**	#
2000	I-LOVE-YOU	6
2004	MyDoom	6
1988	Morris Worm	4
1999	Melissa Virus	4
2003	SQL Slammer	4
2008	Conficker	4
2000	Mafiaboy	3
2000	Titan rain	3
2004	Sasser Worm	3
2010	Stuxnet	3
2011	Epsilon	3
2011	Sony Playstation	3

In the following sections, the author will list significant computer network attacks. These attacks have been discussed previously in van Heerden, Pieterse, and Irwin (2012c).

2.4.1 Phone Phreaking, 1970

A toy whistle included in a box of Cap'n Crunch cereal became the simplest method to break into telephone systems (Rajagopalan, 2000; Robson, 2004). This little toy whistle produced a tone that was used to control telephone systems, and enable the user to make free long-distance calls (also known as phreaking). The whistle generated a frequency similar to the maintenance frequency used by the telephone systems. John Draper was one of the first hackers to abuse this simple method to make free long-distance calls (Levy, 1984; Massey, 2003). Phreaking became a popular hobby for college students, businessmen and anyone else who knew enough about electronics, and it led to the development of new methods that included war dialers, wiretapping and phreaking boxes (Rajagopalan, 2000). Joe the Whistler, a blind man who could whistle a perfect 2600 Hz tone, was used by phreakers to tune their boxes (Rajagopalan, 2000). For many years, US phone companies could do little to counteract phreaking, but as the systems changed from analogue to digital, and the 2600 Hertz (Hz) signal slowly phased out, phreaking techniques were successfully employed in fewer places (Lapsley, 2013). Phone phreaking is considered one of the first computer-based system hacks.

2.4.2 The Logic Bomb, 1982

In 1982 the Central Intelligence Agency discovered that Soviet spies had secretly acquired a gas pipeline controller built in Canada. The Central Intelligence Agency proceeded to plant a Trojan horse, which consisted of a logic bomb, in the software of the controllers (Goertzel, 2009). The controller software controlled the testing of the pipeline pressure gauges, and the logic bomb caused resetting of the gauges to read two-fold lower than the actual gas pressure in the pipelines. This resulted in one of the most monumental non-nuclear explosions ever seen from space (Safire, 2004). This event is one of the first known physical attacks perpetrated by means of hacking.

2.4.3 Brain Virus, 1986

The Brain Virus is considered the world's first computer virus. It was created by two brothers, Basit and Amjad-Farooq Alvi, in Lahore, Pakistan (Abou-Assaleh, Cercone, Keselj, and Sweidan, 2004). It was a boot sector virus since it only affected boot records (Spafford, Heaphy, and Ferbrache, 1989; Leyden, 2006). The Brain Virus marked the area where the virus code was hidden as having bad sectors. It occupied a part of the computer memory and infected any floppy disk that was accessed and hid itself from detection by hooking into the interrupt vector of the boot sector. When an attempt was made to read the infected sector, the virus simply showed the original sector (F-Secure, 2012). Thus the Brain virus was also the first "Stealth" virus that actively attempted to hide its presence.

2.4.4 PC-Write Trojan Horse, 1986

One of the first recorded Trojan Horse software developments, PC-Write Trojan, appeared in 1986. The Trojan pretended to be a Quicksoft PC-Write word processor version 2.72 (Wiggins, 2001; Wang, Chen, and Xu, 2011). When the application was started, the PC-Write Trojan also started. The Trojan then formatted the hard drive and deleted all stored data. It is well known that Quicksoft never published a PC-Writer version 2.72.

2.4.5 Morris Worm, 1988

On 2 November 1988, a Cornell graduate student, Robert Tappan Morris, unleashed one of the first computer worms into the wild (Orman, 2003). It started as a benign

experiment with a simple bug in a programme, but the worm replicated much faster than anticipated (Chen and Robert, 2004). By the following morning, it had infected over 6 000 hosts, nearly 10% of the Internet at the time (Spafford, 1989; Cass, 2001). Ultimately the worm became a victim of its own success as it could not determine whether a host had already been infected or not. As a result the worm distributed multiple copies of itself on a single host. The exponential increase in data load eventually tipped off the system administrators and the worm was discovered. The success and damage caused by the Morris worm led to the founding of the first Computer Emergency Response Team (CERT) at Carnegie Mellon University[2]. The Morris worm prompted the Defense Advanced Research Projects Agency (DARPA) to fund a computer emergency response team, now the CERT.

2.4.6 Chameleon Virus, 1990

Polymorphic computer viruses appeared in the early 1990s (Bania, 2009). The Chameleon Virus (also known as the 1260 Virus), created by Mark Washburn, was the first polymorphic virus. Polymorphic viruses modify themselves with every new infection. The Chameleon Virus consisted of a combination of the Vienna Virus and the Cascade Virus (Beaucamps, 2007b). Washburn extended the original Cascade Virus code and developed a decryptor with a mutable body. The creation of the polymorphic virus shocked the antiviral community since detection techniques used at the time relied on fixed signatures (Beaucamps, 2007a).

2.4.7 Michelangelo Virus, 1991

The Michelangelo Virus surfaced in 1991 (Baskerville, 1993; White, Swimmer, Pring, Arnold, Chess, and Morar, 1999). This particular virus was one of the first viruses to spread worldwide, and it received much media attention. The purpose of the virus was to strike on 6 March, with the effect of destroying millions of computer hard disks. Less than 20 000 computers were actually infected. The Michelangelo Virus contributed greatly to public awareness of computer viruses.

[2]http://www.cert.org/encyc_article/

2.4.8 Kevin Mitnick, 1995

The name Kevin Mitnick has become synonymous with hacking and computer crime (Shimomura and Markoff, 1995; Liddinton-Cox, 2012). As a young boy living in the San Fernando Valley, Kevin started developing his social engineering skills by obtaining free bus rides (Mitnick, Simon, and Wozniak, 2002). During the years that followed, Kevin evolved his skills from phone phreaking to hacking, and eventually mastered the art of social engineering. He soon became the most wanted cyber-criminal in the United States. The Federal Bureau of Investigation (FBI) arrested Kevin in 1995. Well-known criminal acts of Kevin include hacking into DEC, Motorola, Nokia, Sun, NEC, as well many other systems (Mitnick *et al.*, 2002). Kevin Mitnick became the poster boy for hacking and the role model for aspiring hackers (Greenberg, 2008).

2.4.9 Citi Bank, 1995

The attack on Citi Bank was one of the first major financial attacks (Hancock, 1995; Hesseldahl and Kharif, 2010). A Russian hacker (Vladimir Levin) programmed Citi Bank Systems to send $10 million to his own account. Hancock speculated that Levin gained access via stolen passwords. Vladimir Levin was eventually extradited to the United States where he served a three-year prison sentence, and paid $240 000 in damages. Citi Bank managed to recover most of the money (Kabay, 2003).

2.4.10 Laroux, 1996

In July 1996, the first Excel virus, called Laroux, was discovered (Davis, 1996; Haddox, 1996). This virus can be described as a macro virus that consisted of two other macros called "Auto_Open" and "Check_Files" which are stored in a hidden datasheet named "laroux". This virus replicates itself each time a new document is created. Documents that used to be read and write files have become executable, and thus a new agent to spread malware. Since documents are also now a danger, the scope of malware infections has increased from executable binaries only to all but the most basic documents.

2.4.11 Melissa, 1999

The Melissa Virus arrived in the early hours of 26 March 1999 in the form of a Word document (McNamara, 2009). David L. Smith was the alleged creator (Garber, 1999). The Word document, called list.doc, supposedly contained a list of passwords to adult-content websites. Upon opening the document, the virus turned off the security protocol and emailed copies of the infected document to other users of Microsoft Outlook. Melissa was responsible for serious disruptions in big organisations such as Intel, Lockheed-Martin and Microsoft. At the time, it was one of the most damaging computer viruses ever created, and was the first to use email methodology.

2.4.12 I-LOVE-YOU, 2000

The I-LOVE-YOU Worm first appeared on 4 May 4 2000 in the form of an email with the subject: I-LOVE-YOU (Ebel, Mielsch, and Bornholdt, 2002). It was created by a student named Onel de Guzman, and originated from Manila, Philippines. The worm code was written using Visual Basic and processed by the Microsoft WScript engine (Bishop, 2000). It targeted computers using Internet Explorer and Microsoft's Outlook application. Within a few hours, it had spread worldwide via email by making use of addresses in the Outlook address books of infected users. This worm exploited human curiosity in order to entice people into opening an untrusted email.

2.4.13 Mafiaboy, 2000

Michael Calce also known as "Mafiaboy", grabbed headlines in Canada when this high school student launched multiple Distributed Denial-of-Service (DDoS) attacks against major commercial websites. Yahoo, Amazon, Dell, eBay and CNN were some his targets (Groebel, Metze-Mangold, van der Peet, and Ward, 2001). Most of these websites were attacked using a well-known method called "Smurf attacks" (US-CERT, 1998). Smurf attacks involve using fake reply addresses (spoofing), and requesting that the replies be broadcast into a network, thus causing a denial of service though a spike in network traffic (Shearman, 1999). Michael Calce was considered an amateur by other hackers. Grabosky (2004); Genosko (2008) speculated that Michael's main motivation was to receive recognition within his hacker peer group.

2.4.14 Titan Rain, 2000-2008

In 2004, Shawn Carpenter discovered a series of "cyber-raids" carried out by alleged government-supported cells in China targeting the US (Kabay, 2008). Several sensitive computer networks were infiltrated, including Lockheed-Martin and Sandia. The FBI later named it "Titan Rain". The motives were mostly political and economic (Gandhi *et al.*, 2011). It is considered one of the most sophisticated state-sponsored computer attacks ever detected. The attackers searched military networks for single computers with vulnerabilities they would use at a later time to extract data (Thornburgh, 2005). The scale and ambition of this attack made it unique in its time.

2.4.15 Apache.org Defaced 2000

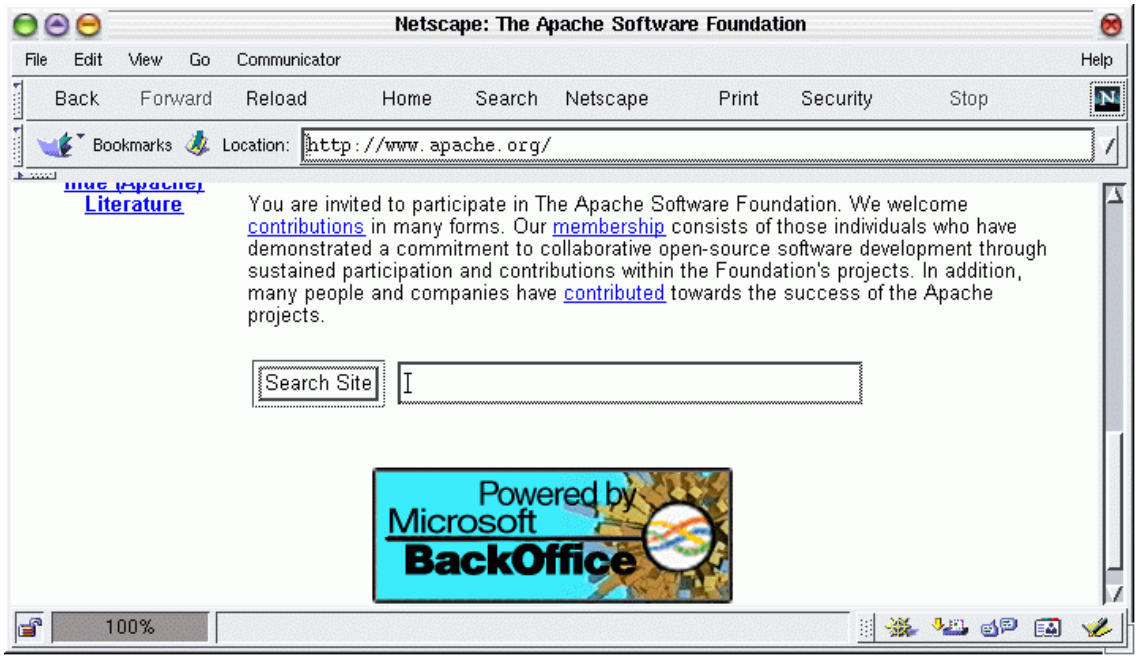


Figure 2.2: Apache.org Defaced (Dede, 2010)

In May 2000, the Apache.org website was defaced (Dede, 2010). Peter van Dijk and his accomplices modified the web page to include a banner "Powered by Microsoft BackOffice" (Apache is a Microsoft competitor in the web-hosting space). The hackers had discovered that their target was using a default configuration. Thus they were able to obtain root access (van Dijk, 2000). The Apache administration team resolved the problems quickly without any significant after effects. The defaced website is shown in Figure 2.2. This

attack is considered significant as it represents not only the defacement of a web page, but the defacement of the web page of the most popular web page software. From October 1996 to 2012, Apache had the top market share for web servers worldwide (Netcraft, 2012). If the attackers' motives were more sinister they could have explored the opportunity of installing malicious code into the web server code.

2.4.16 Code Red, 2001

The Code Red Worm appeared on 12 July 12 2001. It exploited a buffer-overflow vulnerability in Microsoft's IIS web servers (Moore *et al.*, 2003). Upon infection of a machine, it checked whether the date was between the first and the 19th of the month. If so, a random list of IP addresses was generated and each machine on the list was probed to infect as many other machines as possible. Proper propagation of the worm failed due to a code error in the random number generator (Zou, Gong, and Towsley, 2002). On 19 July, a second version of the Code Red Worm appeared, which infected computers at a rate of 200 hosts per minute (Orman, 2003) and infected more than 250 000 systems in just nine hours (Berghel, 2001). This new version shared no source code with the original, but used the same vulnerability and was called Code Red II (Dolak, 2001).

2.4.17 SQL Slammer, 2003

The SQL Slammer Worm consisted of a single User Datagram Protocol (UDP) packet that exploited a Structured Query Language (SQL) server vulnerability. This worm infected over 90% of vulnerable targets within ten minutes (Moore *et al.*, 2003; Zou, Gao, Gong, and Towsley, 2003). It caused significant network outages among financial and government institutions. One of the Slammer's novel features was its incredible scanning rate. The Slammer did not inflict damage with a malicious payload, but rather by overloading networks through the saturating of available bandwidth (Chen and Robert, 2004). In Figure 2.3 the geographical spread of the SQL Slammer worm is shown a mere 30 minutes after its release. The diameter of each circle represents the number of infected hosts on a logarithmic scale.

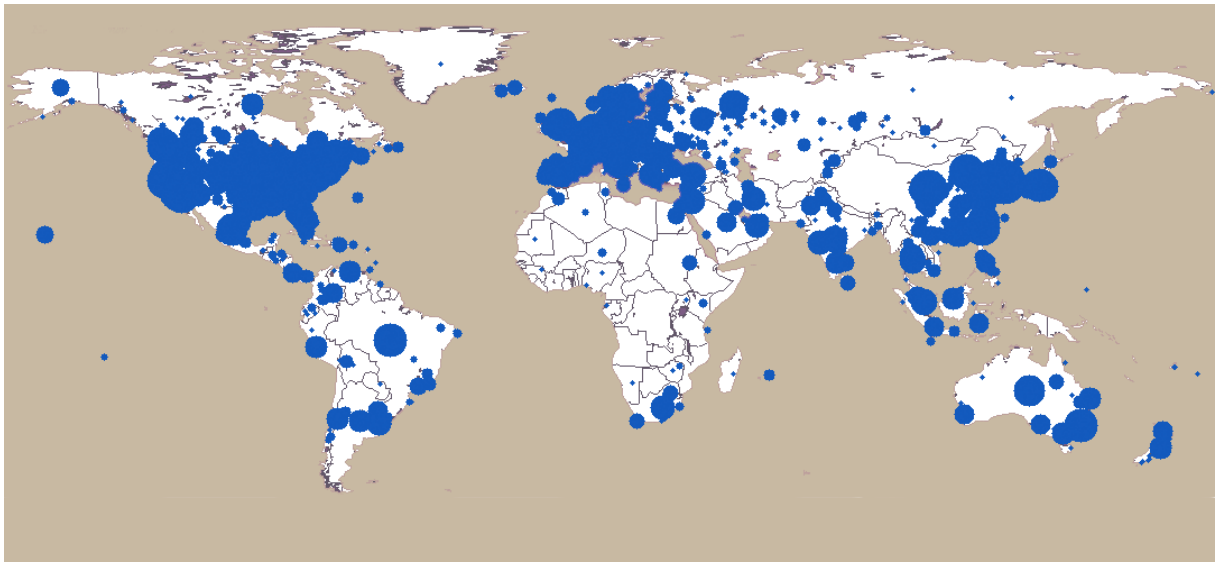


Figure 2.3: Geographical Spread of SQL Slammer by Moore *et al.* (2003)

2.4.18 SCO Denial-of-Service, 2003

An Unix company SCO fell victim to a distributed denial-of-service attack in May 2003 (Shankland, 2003). This attack used multiple computers to simultaneously request a magnitude of connections to the SCO web server. The SCO web server was unable to serve (respond in time) all the connections. Thus SCO's web presence was removed during the attack. A SCO representative stated that they had no indication of who was behind the attack, and nobody claimed official responsibility for it. In a similar attack during December 2003, the UCSD Network Telescope calculated that the SCO server had to respond to more than 700 million attack packets over a 32-hour period (Moore and Shannon, 2003). The motive for the attack was suspected to be related to the SCO lawsuit against IBM regarding copyright of the Linux code (Namuduri, 2006). This attack is considered significant as it is an example of hacktivism, where a website was attacked by non-state actors for political reasons.

2.4.19 Cabir Worm, 2004

The Cabir Worm was discovered by Symantec on 14 June 2004. It was the first worm to infect mobile devices (Sarwar, Ramadass, and Budiarto, 2007). It targeted mobile devices using the Symbian OS. Its creator lived in France and used the name Vallez (Gostev, 2006). Infection occurred via Bluetooth. The infection rate was significantly restricted due to the

short transmission distances of Bluetooth (Attewell, 2005). The worm did not succeed in creating major havoc on mobile devices, and caused little damage. However, with the rising popularity of smart phones it is expected that mobile devices will increasingly become the targets of malware.

2.4.20 MyDoom, 2004

27 January 2004 saw the arrival of the mass-mailing worm called MyDoom (Dübendorfer and Plattner, 2005). The worm spread via executable email attachments, and also set up a backdoor Trojan on infected computers. It used its own Simple Mail Transfer Protocol (SMTP) engine to send infected emails. During its active lifetime of 12 months it caused an increase in global email traffic estimated to be between 14% and 30% (Dübendorfer and Plattner, 2005). Public awareness, antivirus software and firewalls using SMTP filtering prevented it from growing rapidly. The MyDoom Worm had many variants, including the variant MyDoom.e, which attacked the SCO web page, and the variant MyDoom.f attacked Microsoft and RIAA websites (Germain, 2004).

2.4.21 Sasser Worm, 2004

The Sasser Worm spread through a Microsoft network vulnerability (MS04-011[3]). This vulnerability exploited a buffer overflow in the Local Security Authority Subsystem Service (LSASS). The Sasser Worm was allegedly authored by Swen Jaschan, a German high school student (TrendMicro, 2004). It randomly generated IP addresses and then attempted to connect on Transmission Control Protocol (TCP) port 445 and exploit more hosts. This worm was able to infect over half a million Windows users within the first few days of its release (Sanger, 2012).

2.4.22 Sony XCP, 2005

Sony BMG included digital rights management technologies in Compact Disks (CDs) released during 2005 (Halderman and Felten, 2006; Mulligan and Perzanowski, 2007). One such technology was XCP, a CD-based protection measure developed by First4Internet. The initial purpose of XCP was to place certain restrictions on the use of purchased CDs.

[3]http://technet.microsoft.com/en-us/security/bulletin/ms04-011/

In addition to the restrictions, XCP also created a number of security vulnerabilities for Windows users. Mark Russinovich was the first person to release information about these risks to the public on 31 October 2005 (Krebs, 2005). Sony's initial response was slow. By the end of 2005, millions of infected CDs were still available in retail stores before their eventual recall. This vulnerability was an example of where a large international corporation's (Sony) desire to protect its content led to damaging users' computers and the corporation's own reputation.

2.4.23 Operation Shady RAT, 2006-2010

Operation Shady RAT was a huge corporate spying network used to steal corporate secrets from over 72 international companies (Alperovitch, 2011). This included source code, email archives, exploration details for fossil fuels, legal contracts, design schematics, etc. McAfee was able to gain access to one of the command and control computers used in this attack and identified 72 compromised parties that were distributed all over the world (Figure 2.4). Although no definitive proof exists, it is currently suspected that the attack originated in China (Gross, 2011).

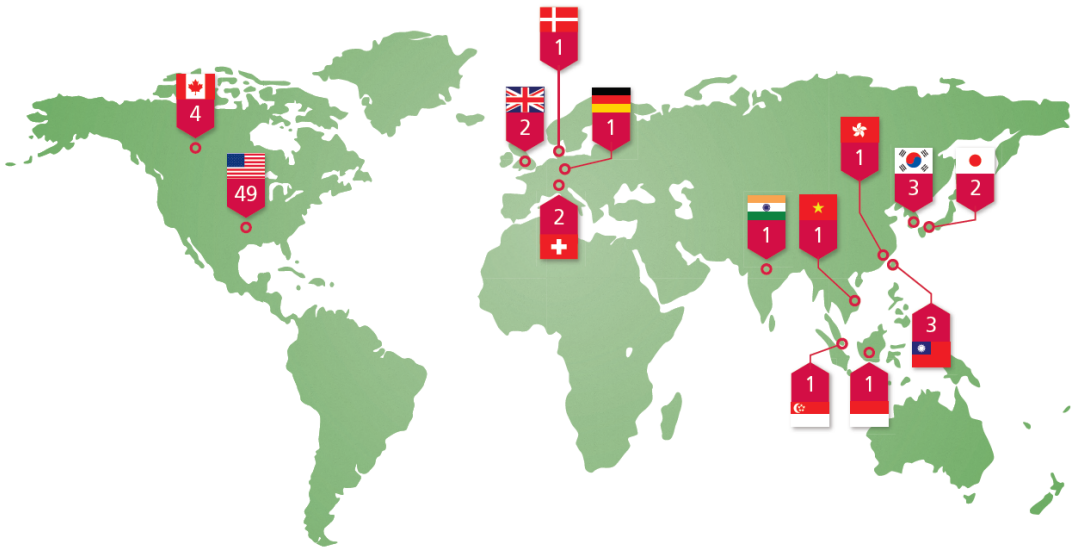


Figure 2.4: Operation Shady RAT Victims Geographical Locations (Alperovitch, 2011)

2.4.24 Estonia Incident, 2007

Early in 2007, a series of politically motivated cyber-attacks struck Estonia (Finn, 2007; Czosseck, Ottis, and Taliharm, 2011). The attacks included web defacements and DDoS attacks on well-known Estonia government agencies, banks and Internet service providers. The attacks followed the removal of a six-foot-tall bronze statue in Tallinn, which commemorated the dead of the Second World War (Davis, 2007). At the time of the attacks, Estonia was one of the leading nations in Europe with regard to information and communication technologies (Czosseck *et al.*, 2011). This can be considered an example of cyber-warfare and its potential effects (Clarke and Knake, 2011).

2.4.25 South Ossetia Incident, 2008

Websites in Georgia were hacked three days before the start of the Georgia Russia war. The websites of the Georgian Ministry of Foreign Affairs and its Parliament were replaced with images comparing the Georgian president Mikheil Saakashvili to Adolf Hitler, as shown in Figure 2.5 (Cluley, 2008).

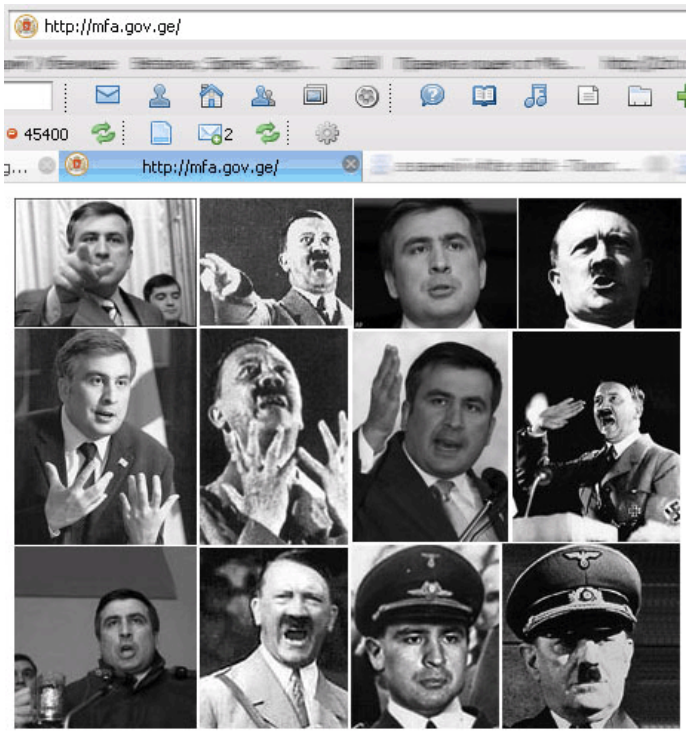


Figure 2.5: Defaced Georgian Parliamentary Website (Cluley, 2008)

Russian intelligence services also conducted DoS attacks on Georgian websites. Researchers concluded that the attacks were synchronised with Russian military operations, and that the malware was written or customized for the attack against Georgia (Rutherford, 2009). The Russian state-sponsored news agency, RIA Novosti was attacked in retaliation (Spiegelman, 2008).

2.4.26 Conficker Worm, 2008

The Conficker Worm was the first worm to penetrate cloud technology (Fitzgibbon and Wood, 2009; Sharma, 2011). It first appeared in November 2008 and quickly became one of the most infamous worms to date. It controlled over nine million computer systems and also controlled the world's largest cloud network at the time (Wattanajantra, 2009). As a result of the infrastructure of a cloud, the worm could propagate much faster, infect a broader range of hosts and cause greater damage. Conficker spread via autorun in removable storage devices and also spread via network shares (Porras, Saïdi, and Yegneswaran, 2009). Conficker has not been used as an attack weapon since, and it is speculated that it might have been a precursor to Stuxnet (Finkle, 2011).

2.4.27 Ikee Worm, 2009

The first worm to infect Apple's iPhones emerged in 2009. Ikee targeted jailbroken iPhones by exploiting default passwords (Porras, Saidi, and Yegneswaran, 2010). It did not cause serious damage to the infected iPhone, but simply changed the wallpaper to an image of the singer Rick Astley. After changing the wallpaper, it sought out other jail-broken iPhones to infect. The creator, a 21-year-old student called Ashley Towns, only developed the worm in order to raise concerns about certain security issues (Andersen, 2009). It did not contain any malicious content.

2.4.28 Operation Aurora 2009

Google disclosed in early 2010 that it fell under attack from China (Higgins, 2010; Drummond, 2010). The goal of the attacks was to extract source code from Google, Adobe and other significant technology companies (Zetter, 2010). The attackers used a 0-day exploit in Internet Explorer to steal intellectual property. Also, significantly, access to Chinese

Gmail accounts was sought. Attacks of this sophistication and scope was unknown at the time and required significant investment and time to execute (Zetter, 2012a). McAfee described the steps used by the attackers to steal data (McAfee, 2010):

1. *A targeted user received a link in email or instant message from a 'trusted' source.*
2. *The user clicked on the link, which caused them to visit a website hosted in Taiwan that also contained a malicious JavaScript payload.*
3. *The user's browser downloaded and executed the malicious JavaScript, which included a zero-day Internet Explorer exploit.*
4. *The exploit downloaded a binary disguised as an image from Taiwanese servers and executed the malicious payload.*
5. *The payload set up a back door and connected to command and control servers in Taiwan.*
6. *As a result, attackers had complete access to internal systems.*

The attack was traced back to two schools in China that have close ties with the Chinese military (Markoff and Barboza, 2010).

2.4.29 Stuxnet Worm, 2010

Stuxnet was one of the most complex threats ever analysed (Falliere, Murchu, and Chien, 2011). The primary purpose of Stuxnet was to target industrial control systems, such as gas pipelines and power plants, with the goal of reprogramming the Programmable Logic Controls (PLCs) systems to enable an attacker to control them. Stuxnet was also the first to exploit four zero-day vulnerabilities as well as compromise two digital certificates. As of 29 September 2010, Iran had the greatest number of infected computer systems. Stuxnet has shown that direct-attack attempts on critical infrastructure are no longer a myth, but a definite possibility. Stuxnet actions can be considered an act of war, but no one has officially claimed responsibility for it (Fidler, 2011), although Sanger (2012) was able to confirm that Stuxnet was a joint Israeli-American cyber-weapon.

2.4.30 Epsilon, 2011

In April 2011, customer information (names and emails) were stolen from Ameriprise Financial, Best Buy, Bookstone, Capital One, Citi, Disney Destinations, Home Shopping

Network, JPMorgan Chase, Marriott Rewards, US Bank, TiVo and Walgreens (Olivarez-Giles, 2011). Epsilon, a marketing email provider from which the email addresses were leaked stated that passwords and credit card details were not compromised (Halliday, 2011). The biggest danger resulting from this attack is a possibility that the stolen names and email addresses may be used in future Spear Phishing attacks (Kerber and Bartz, 2011).

2.4.31 PlayStation Network, 2011

Sony's PlayStation Network went offline on 20 April 2011 (Thomas, 2011). A few days later, Sony confessed that the network went offline due to an external intrusion. Sony also warned its users to watch out for possible identity theft as the hackers obtained sensitive information such as usernames, passwords, addresses and birth dates (Goodin, 2011). Sony blamed Anonymous for the attack, but Anonymous denied involvement (Cohen, 2011; Kaplan, 2011).

2.4.32 HBGary, 2011

In February 2011, a computer attack was launched on one of the leading computer security firms, HBGary Federal (Bright, 2011). The CEO of HBGary Federal, Aaron Barr, announced that he was going to unmask the well-known hacking group Anonymous. Anonymous responded swiftly and caused severe damage to the security firm. The attacks resulted in defacement of their website and deletion of vast amounts of data. In addition, a website owned by the owner of HBGary, Greg Hoglund, went offline and the user registration database was published on the Internet. Anonymous ultimately removed the links to the published emails after negotiations with Barr and Hoglund (Zetter, 2011). This attack is considered significant as it demonstrates the potentially negative impact of skilled hacker groups, and the inherent vulnerability of individuals.

2.4.33 South African PostBank, 2012

The South African PostBank was robbed of R42 million ($6.7 million) during a 72-hour operation that took place during the New Year holiday period (Swart and Afrika, 2012). Boy Meshack Thekiso a Postbank employee, used the computer of a colleague who was on

leave, which was linked to the PostBank server, to transfer money to multiple accounts. Large amounts of money were withdrawn from ATM's all over South Africa from these accounts. Two of the syndicate members, Motsoane and Masoleng, were arrested in February 2012 and sentenced to 15 years for their role in the theft (SAPA, 2012; Swart, 2012). Thekiso turned state witness and received a ten-year sentence.

2.4.34 Flame, 2012

Flame is a modular computer malware that attacks computers running the Microsoft Windows operating system. The Flame malware has a backdoor, a Trojan, and worm features which allow it to replicate in a local network (Gostev, 2012). The Flame software seems to especially target systems in Iran, Lebanon, Syria, Sudan, Israel and other Middle Eastern countries. In Figure 2.6, a map of the initially infected countries is shown.

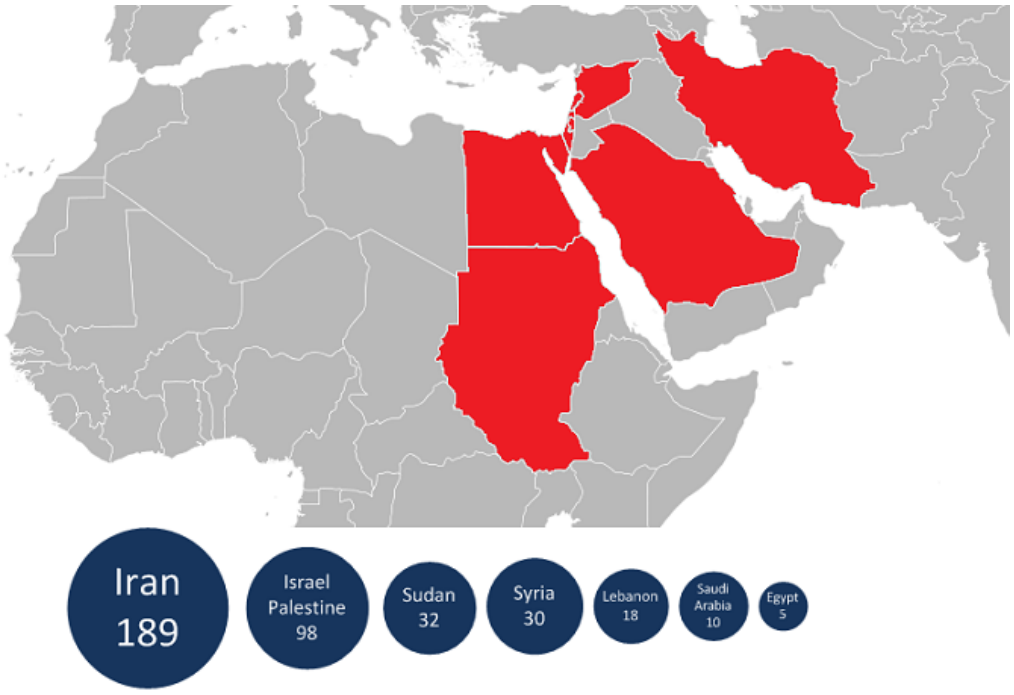


Figure 2.6: Flame-infected Countries 28 May 2002 (Gostev, 2012)

Flame is considered the most sophisticated and complex malware ever found (Sharma, 2012).

2.4.35 SpamHaus, 2013

On 16 March 2013, a DDoS attack was launched on the SpamHaus website (Hanford, 2013)[4]. The attack grew to 300 Gigabits per second (Gbps) of flood traffic, a level which threatened the overall Internet core infrastructure and Leyden (2013) claimed that it was the "Biggest DDoS attack in History". SpamHaus contracted CloudFare[5] to mitigate against the attack, and they were able to restore SpamHaus services (Prince, 2013). Figure 2.7 displays bandwidth across a number of the routers that CloudFare monitored in front of the SpamHaus web site. The green area represents incoming requests in-bound and the blue line represents outgoing data. The massive spike in incoming traffic presents the DDoS attack.

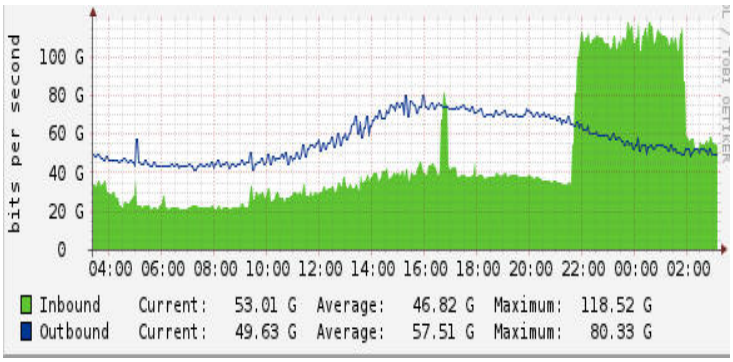


Figure 2.7: CloudFare Monitoring Traffic in front of SpamHaus (Prince, 2013)

Although they deny it, it has been speculated that a Dutch hosting company called Cyber-Bunker attached SpamHaus in retaliation for being placed on the SpamHaus anti-spam list (Markoff and Perlroth, 2013). A Dutch national was arrested in Spain during April 2013, was extradited to the Netherlands and faces charges with regard to the DDoS attack on SpamHaus. The DDoS attack on SCO in 2003 and SpamHaus in 2013 used the same methodology, and only differed in scale. In the ten years between the attacks, the Internet and related networks have grown to the extent that the amount of data required for a successful DDoS attack is significantly greater.

2.4.36 Edward Snowden, 2013

In June 2013 *The Guardian* and *Washington Post* newspapers revealed in June 2013 that the NSA accessed servers from large US tech companies to collect metadata (Greenwald,

[4]http://spamhaus.org/
[5]https://www.cloudflare.com/

2013). More revelations about NSA cyber-spying programme called PRISM was revealed in the following weeks. The source of the leaks were revealed to be Edward Snowden, a NSA contractor (Glenn Greenwald and Poitras, 2013). Snowden's motivation for revealing information about PRIMS and the NSA is that he wanted to prompt a national security debate (Farivar, 2013):
"The surveillance of whole populations, rather than individuals, threatens to be the greatest human rights challenge of our time..."

It has emerged that one of the techniques that Snowden used to collect all the material was by convincing "20 to 25" of his coworkers to hand over their login credentials and passwords (Geuss, 2013). Currently Snowden has been given temporary asylum in Russia while the US insists that he is a traitor (Walker, 2013). Catro (2013) estimates that Snowden's revelations will cost the US cloud computer industry up to $35 billion.

2.5 Attack Scenarios

Rahmad, Supangkat, Sembiring, and Surendro (2010) developed the concept of a "threat scenario". They reasoned that all threats can be classified as either the hijacking of uses, espionage, exceeded limits of operation, damage, modifications or loss of property. These scenarios were used as a starting point for development of the following attack scenarios:

- Denial-of-Service (SCO DoS, Mafiaboy, SpamHaus)
- Industrial Espionage (Titan Rain, Operation Aurora, Operation Shady RAT)
- Web Defacement (HB Gary Hack, Apache.org Defaced)
- Unauthorised Data Access (Kevin Mitnick, Playstation Network, Epsilon, Edward Snowden)
- Financial Theft (South African PostBank, Citi Bank)
- Industrial Sabotage (The Logic Bomb, Stuxnet Worm)
- Cyber-Warfare (South Ossetia Incident, Estonia Incident)
- Resource Theft (Phone Phreaking)
- System Compromise (Flame, MyDoom, Conficker, Code Red, Sony XCP)
- Runaway Malware (I-LOVE-YOU, Morris Worm, Mellissa, SQL Slammer, Brain Virus, PC-Write Trojan Horse, Chameleon Virus, Michelangelo Virus, Laroux, Cabir Worm, Sasser Worm, Ikee)

Each of the attack scenarios above are explored in more detail in Section 4.3. The *Denial-of-Service* scenario is used to describe attacks that target accessibility by overloading a

victim's capability to respond to a flood of interaction requests. *Industrial Espionage* refers to the theft of commercially valuable data such as trade secrets, system blueprints or sales numbers. *Unauthorised Data Access* refers to a situation where a person has access to a location that is hidden or contains sensitive or secret data. This can lead to the unauthorised entry of data into a file, reading a file, changing the contents of the file or for any other malicious purpose.

Web Defacement can be considered graffiti of the digital world. Websites are the public face of commercial and other entities in the digital world, and reputations are negatively affected by defacing them.

Computers can be used for direct financial gain by stealing money directly from banks, individuals or other institutions. Computer attacks with a sole financial goal are referred to as the *Financial Theft* scenario. Much malware attempts to control computers and networks. When control of a computer or network has been lost, the *System Compromise* is applied. *System Compromise* refers to the breaking or cracking into a single or multiple computers without authorisation. Such a compromise is often achieved by using stolen identification and/or passwords to achieve privilege escalation and then compromise the computer system.

Industrial Sabotage scenarios refer to where computers are used to attack industrial targets physically. The Logic Bomb and Stuxnet attacks resulted in physical damage to industrial equipment. The next step is to use computers directly in war. This was done in the South Ossetia Incident where computer attacks were launched in conjunction with military operations.

Much of the malware that caused the greatest damage and financial losses was software that was written with no other goal than to see how far it could spread. This *Runaway Malware* usually exploits some technical flaw that allows it to spread.

Resource Theft is the unauthorised usage of computer resources, such as bandwidth and disk space, to accomplish unofficial tasks. *Resource Theft* includes, but is not limited to, the following activities: using computing facilities and resources to interfere with the work of an employee, abusing computing facilities and resources to send unauthorised messages that are obscene, harassing or threatening and interfering with the normal operations of a company's computing system.

2.6 Significance

In this section, the significance of the attacks are sorted according to four criteria:

- first use of an attack methodology;
- first significant use of a new technology;
- attacks with significant financial impact; and
- sophisticated attacks.

The grouping of the attacks into these criteria is subjective and meant to be a definitive grouping.

In Figure 2.8, the attacks are listed that are significant because they were the first known instance of a new attack methodology. For example, the Laxroux Excel Virus was the first macro virus, and Ikee the first worm that attacked via mobile phones.

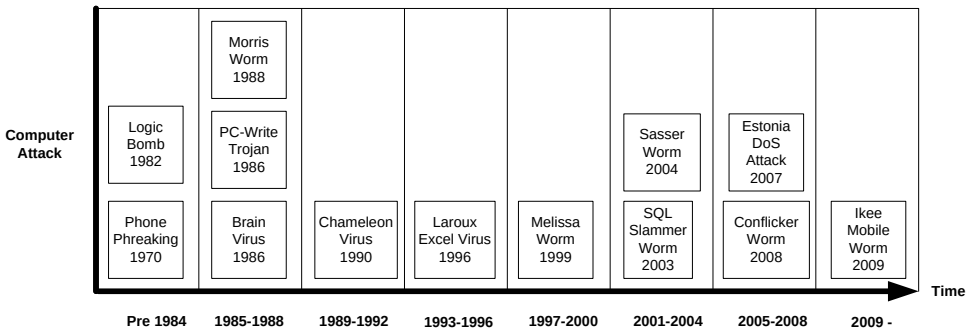


Figure 2.8: List of Computer Attacks with the First Significant Use of an Attack Methodology

The first significant use of a technology for computer-related attacks is shown is Figure 2.9. One of the best examples of an attack that used a new technology for the first time is the Sony XPS attack, which used digital rights management technology.

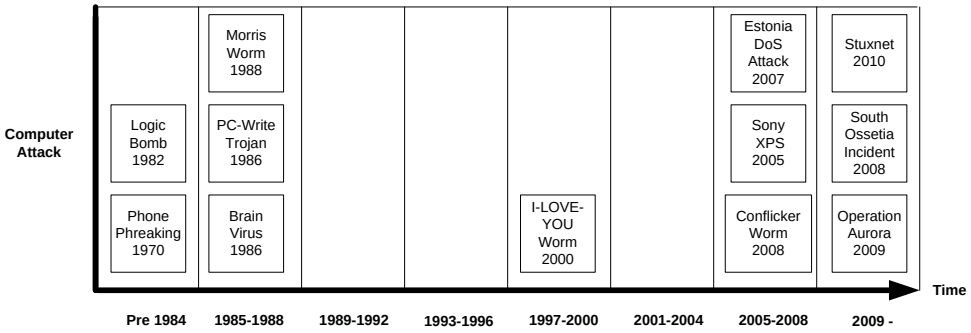


Figure 2.9: List of Computer Attacks with the First Use of a New Technology

Many of the computer-related attacks had a significant financial impact. Many of the attacks shown in Figure 2.10 regarding financial impact cannot be determined due to the difficulty in measuring confidence loss. For example, the Playstation Network Attack was estimated to have cost Sony over $170 million (Stone, 2012).

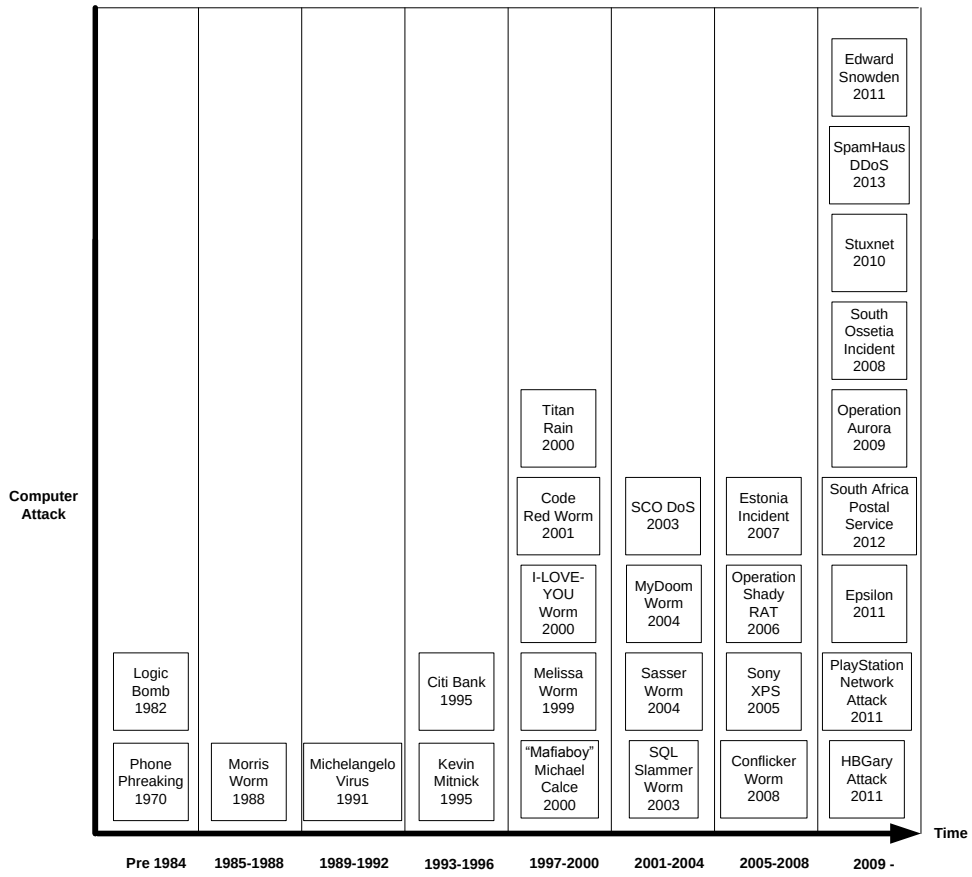


Figure 2.10: List of Computer Attacks with Significant Financial Impact

The final criterion is sophisticated attacks. One of the best examples is the Stuxnet attack, which used multiple zero-days and attacked a network that was not accessible via the Internet. Stuxnet is considered to be a cyber-weapon (Chen, 2010).

2.7 Summary

Computer viruses were the most significant computer attacks in the 1980s. Computer viruses used novel and unique methodologies to attack computers. Viruses started out as simple viruses such as the Brain Virus that simply copied itself through interrupt vectors. More complicated viruses are the polymorphic viruses such as the Chameleon Virus.

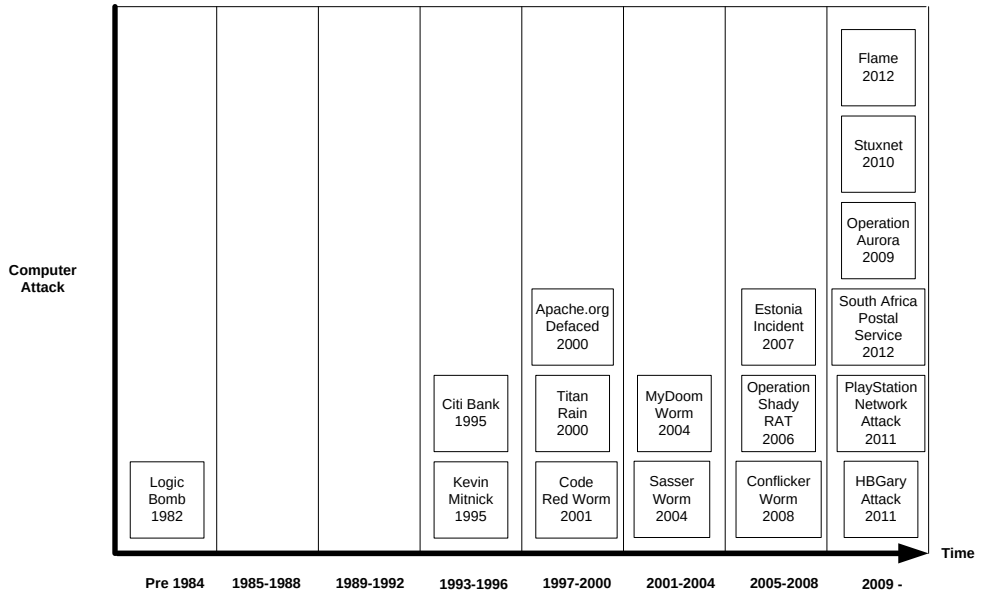


Figure 2.11: List of Computer Sophisticated Attacks

With the development of the Internet, the significant computer attacks became network-based attacks. During the 1990's and 2000's, worms became the most significant threat. These worms spread though server software vulnerabilities (such as the Morris Worm and SQL Slammer) or used social engineering methods such as the I-LOVE-YOU email worm.

With the commercial growth of the Internet, attacks on commercial entities became more sophisticated and significant. The scope of attacks on commercial entities ranged from politically motivated (SCO DoS) to large-scale industrial espionage. The step from industrial espionage to industrial sabotage reached a significant level with the Stuxnet-targeted attack. The Stuxnet attack proved that, since computer networks have become an integral part of a nation's infrastructure, they can be used as an attack vector.

This chapter examined recent history to identify computer attacks that have had the most impact or can be considered the most significant. These significant attacks were used to identify computer attack scenarios. These scenarios are used as the base for the computer attack ontology developed in Chapter 5. In the next chapter, related research on network attacks will be presented.

CHAPTER

THREE

RELATED RESEARCH

"Read at every wait; read at all hours; read within leisure; read in times of labour; read as one goes in; read as one goes out. The task of the educated mind is simply put: read to lead."

Marcus Tullius Cicero – 106 BC to 43 BC

3.1 Introduction

Attacks on computer networks can be considered an arms race. With every advance in protection/prevention, new attacks counter it and become more sophisticated (Zhou, Leckie, and Karunasekera, 2010; Kayacık, Zincir-Heywood, and Heywood, 2011). This chapter presents related work in the study of computer network attacks. In Section 3.2, a survey of network attack models is presented and in Section 3.3, a summary of a collection of network attack taxonomies is presented. Ontologies have been used in the field of computer attack detection, and such studies are presented in Section 3.4. Sensors that can be used to detect network-related attacks are discussed in Section 3.5.

Most models, taxonomies and ontologies are presented in the form of flow charts or class diagrams. The flowcharts simplify the presentation of the interaction within models and the the class diagrams enable a holistic view and comprehension. Where applicable, references are made to where these classes were used within the researcher's taxonomy and ontology.

3.2 Network Attack Models

In this subsection, models that represent attack on networks are discussed. The most basic attack process and slight variations on it are presented. This model was used to base the network attack model developed in Section 4.4. The variations on the basic model are presented in historical order as shown in Figure 3.2.

After the basic models, three complex models are presented. These complex models include the inputs and outputs of the attack models. The complex models are shown in Figures 3.3, 3.4 and 3.5.

3.2.1 Generalised Basic Attack Process

The most basic generalised attack process, as derived from research conducted by Cheswick (1992); Kurtz, McClure, and Scambray (1999); Boyd (2000); Schultze (2002); Teumim (2010), is shown in Figure 3.1.

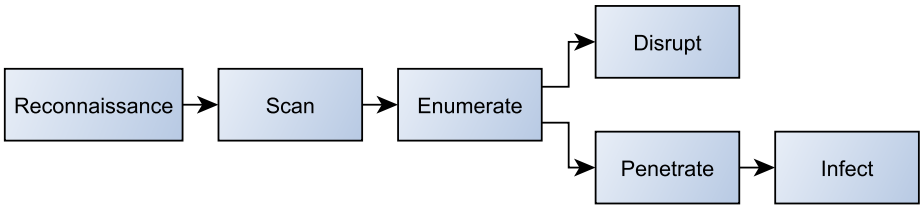


Figure 3.1: Generalised Basic Attack Process

According to Teumim (2010), and Knapp (2011) the attack process is equally valid for any target being attacked, be it Internet networks or industrial systems. Teumim defined the attack phases as follows:

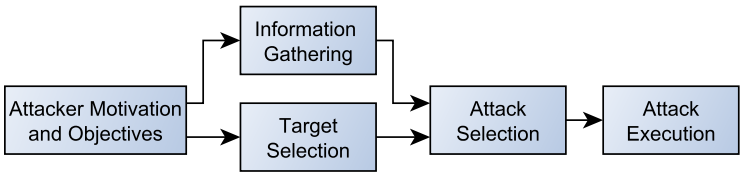
- Reconnaissance: This is the initial phase (also known as "foot printing") where an attacker obtains information about an organisation, such as its employees, Internet domain and other useful information.
- Scanning: This is the identification of network-related devices using Internet Control Message Protocol (ICMP) to determine live hosts, and open TCP and UDP ports to determine possible services.

- Enumeration: This is the identification of user accounts, network shares and other open information on the network. For example, if a user account name can be obtained, its password can be guessed or cracked by brute force.
- Disrupt: The goal of this activity is only to withhold some service from the computer user. This does not necessarily require further network or computer penetration.
- Penetrate: The goal of this phase is to overcome the system's defences, such as antivirus and privilege-level software.
- Infect: The goal of this phase is to become a persistent threat by attempting to stay hidden while performing any required tasks. Sterling (2010) listed additional steps required to maintain persistent infection:
  - constructing outbound connections (also known as backdoors) for command and control use
  - collecting user credentials and attempting to access more systems
  - escalating user privileges to a higher level
  - using legitimate services to hide outbound connections
  - utilising mutation and other polymorphic techniques to avoid detection
  - maintaining its hidden presence on the system by removing logs (evidence of infection)

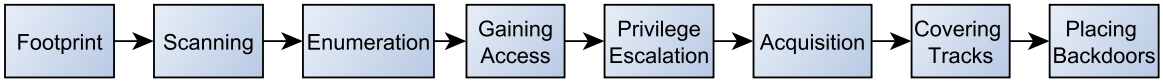
Formal variations on the basic network attack model are introduced in sections 3.2.2 to 3.2.6. These models all follow the same basic steps, with small variations in name conventions and steps. The models mostly follow a linear path, have between four and eight stages, and are presented in chronological form. In sections 3.2.7 to 3.2.9, more complicated attack models are presented. These models also require inputs and outputs for each stage.

3.2.2 Hansman and Hunt Model

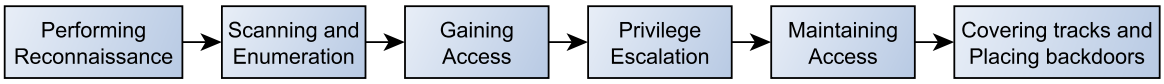
Hansman and Hunt (2003) described a computer attack as a process with several distinct phases occurring on a targeted computer or network. An attack follows four main phases, as demonstrated in Figure 3.2a. The Hansman and Hunt model denotes *Information Gathering* and *Target Selection* as phases that occur concurrently and distinctly. Most other models regard *Target Selection* as a phase that occurs before *Information Gathering*.



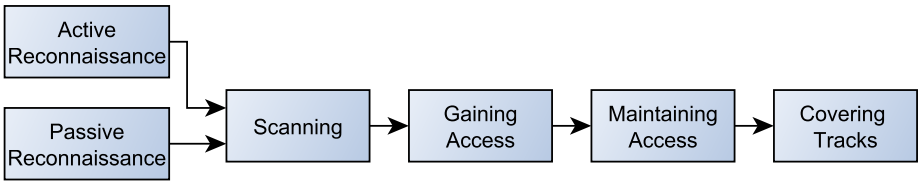
(a) Hansman and Hunt Model



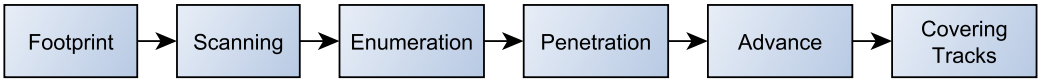
(b) Tutănescu and Sofron Model



(c) Gadge and Patil Model



(d) Sharan Model



(e) Nachenberg Model

Figure 3.2: Basic Attack Models

3.2.3 Tutănescu and Sofron Model

Tutănescu and Sofron (2003) presented a synthetic anatomy of network attacks, as shown in Figure 3.2b. Tutănescu and Sofron also stated that these attack phases are in continuous development and require constant updating. The *Footprint* and *Scanning* phase is similar to other *Reconnaissance* phases. This model is similar to the model of Grant and Kooter, which is also only limited to attacks that gain access to systems, and it does not represent the Denial-of-Service scenarios.

3.2.4 Gadge and Patil Model

The Gadge and Patil (2008) model consists of six basic steps: performing reconnaissance, scanning and enumeration, gaining access, escalation of privilege, maintaining access, and covering tracks and placing backdoors. Gadge and Patil defined reconnaissance as the phase where a hacker tries to find out as much as possible about the target. The researcher refers to this phase as *Target Identification*, but Jung, Paxson, Berger, and Balakrishnan (2004) refer to the Scanning and Enumeration phases as the *Reconnaissance* phase. The researcher chose to refer to scanning and other similar actives as reconnaissance. In Figure 3.2c, the Gadge and Patil model is displayed.

3.2.5 Sharan Model

Sharan (2010) described five stages of ethical hacking. These stages are shown in Figure 3.2d. The steps used by ethical hackers can also be used by hackers with more malicious intent. Sharan's steps are very similar to Teumim's model, except for differentiating between active and passive reconnaissance. Sharan uses slightly different terms, such as *Gaining Access* rather than the more popular *Enumeration* used by most authors.

3.2.6 Nachenberg Model

Nachenberg (2012) described hacking as a systematic, wearisome process. This process is methodical, and requires six main steps: footprinting, scanning, enumeration, penetration, advance and covering tracks. In the advance phase, the hacker beverages information gained to launch the next level of attacks, for example installing backdoors. Figure 3.2e displays Nachenberg's model.

3.2.7 Grant and Kooter Model

Grant and Kooter (2005) developed an attack model within a crime context. It was developed following an analysis of hackers' writings. The model is summarised in Figure 3.3, where it uses the similar phases as described previously and includes the inputs and outputs of each phase. This model is specific to hacking-type attacks, with phases such as *Penetration*, *Control*, *Embedding*, *Data Extraction* and *Attack Relay*. These phases can be used in denial-of-service-type scenarios. Grant, Venter, and Eloff (2007) refined the model into nine phases: footprinting, reconnaissance, vulnerability identification, penetration, control, embedding, data extraction, attack relay and attack dissemination.

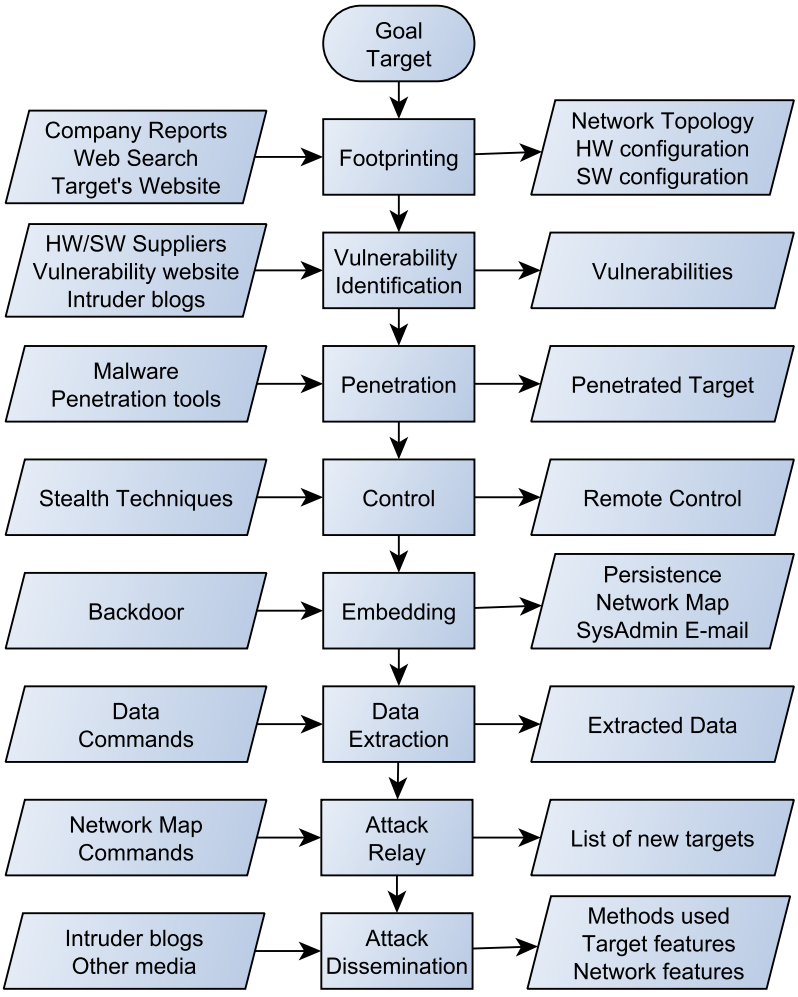


Figure 3.3: Grant and Kooter (2005) Model

3.2.8 Colarik and Janczowski Model

Colarik and Janczowski (2008) developed a model within a hacking terrorism context. Their model was developed by creating analogies based on hacker crime instances. The model is presented in Figure 3.4. The Colarik and Janczowski model groups the identification of vulnerabilities and selection of malware into a single *Penetration* phase. Unlike the Grant and Kooter model, this model differentiates between owning a system and disrupting a system.

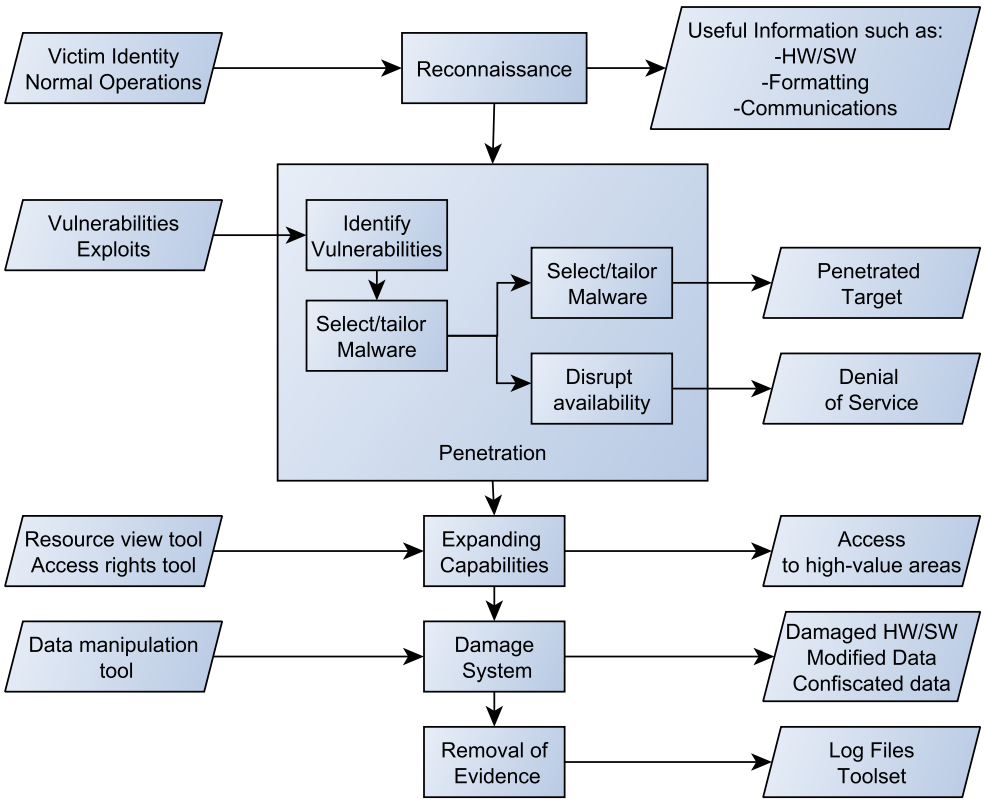


Figure 3.4: Colarik and Janczowski (2008) Model

3.2.9 Damballa Model

Damballa (2008) looked at case studies of lone hackers within the criminal context. His model is illustrated in Figure 3.5 and consists of only three basic steps: *Deliver Malware*, *Consolidation* and *Take Action*. This model does not use the *Reconnaissance* or similar phases.

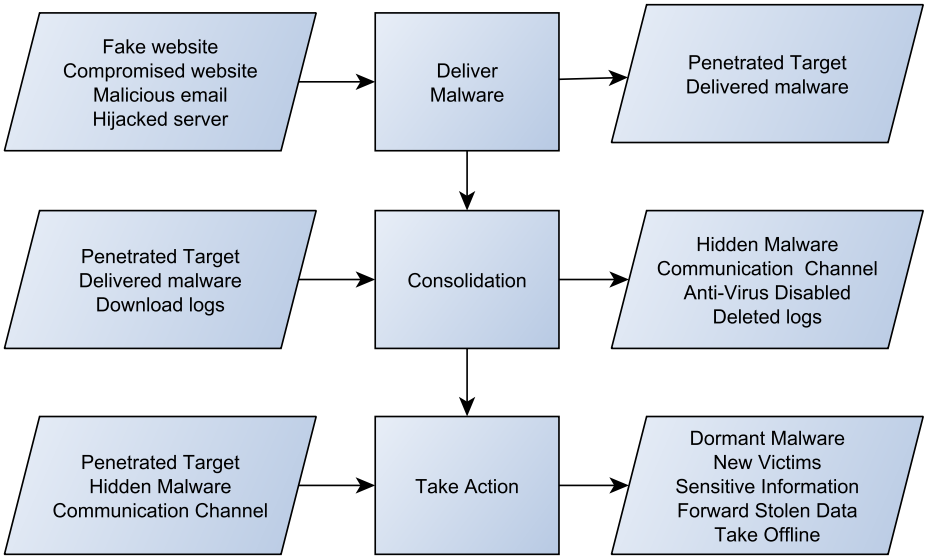


Figure 3.5: Damballa (2008) Model

3.2.10 Network Attack Models Overview

In this section, several network attack models are presented. All the models follow a similar path, with slight differences. These differences are in naming conventions or grouping of stages. For example, the terms *Footprint* and *Reconnaissance* could be used interchangeably. Most of the models start with some Footprint/Reconnaissance/Scanning, followed by Gaining Access/Enumeration/Escalation Privilege. After the system has been compromised, the attack Disrupts/Penetrates/Infects/Maintains Access of some sort and ends by Covering Tracks/Installing Backdoors. The attack model developed in Section 4.4 is based on the stages presented in this section.

3.3 Attack Taxonomies

In this section, the network attack and related taxonomies are explored. Rouse (2005) defines a taxonomy as:

> "the science of classification according to a predetermined system, with the resulting catalog used to provide a conceptual framework for discussion, analysis, or information retrieval."

The word taxonomy is derived from the Greek *taxis* (arrangement) and *nomos* (law). Hlava (2012) defines a taxonomy as a *knowledge organization system*. It is therefore a set of words that is used to define a field's vocabulary.

Taxonomies are presented as figures with their main class at the top. Some of the taxonomies were split to fit all the classes and sub-classes within a single page. The level of the classes and sub-classes can be differentiated by their shape, as illustrated in Section 1.5. The taxonomies are presented in a chronological sequence.

3.3.1 Taxonomy of Attack Techniques by Lindqvist and Jonsson (1997)

Lindqvist and Jonsson (1997) presented a classification of network intrusions. The classification was built on intrusion experiments. Lindqvist and Jonsson expanded on the nine computer misuse classes defined by Neumann and Parker (1989). In Figure 3.6, Lindqvist and Jonsson display the Computer Misuse and Computer Misuse Result classes.

The researcher's *Attack Goal*, *Effect* and *Sabotage* classes were influenced by Neumann and Parker's taxonomy, whose taxonomy is fully presented within Lindqvist and Jonsson's taxonomy. These can be seen in sections 4.2.5, 4.2.8, and 4.2.11. Their taxonomy *Computer Misuse Result* class does not present destruction, changing or disclosure of data on its disclosure. Their *Computer Misuse* class also caters for activities which fall outside the scope of computer network attacks, such as *Misuse from Inaction* and *Hardware Misuse*.

3.3.2 A Taxonomy of Network and Computer Attack Methodologies by Hansman and Hunt (2003)

In Figure 3.7, Hansman and Hunt present attack methodologies, and in Figure 3.8, their attack taxonomy. The attack methodologies are separated by the their mechanism (such as Buffer Overflows and Physical Attacks), their target (such as Web Application and Password Attacks) or effect of the attack (such as Denial-of-Service and Information Gathering Attacks). The complexity of attack mechanisms is shown as Figure 3.7 presents more than 30 unique attack mechanisms classes.

The *Attack Mechanism* class of Section 4.2.6 is based primarily on Hansman and Hunt's work. Furthermore, the *Target*, *Vulnerability* and *Attack Goal* classes were all influenced

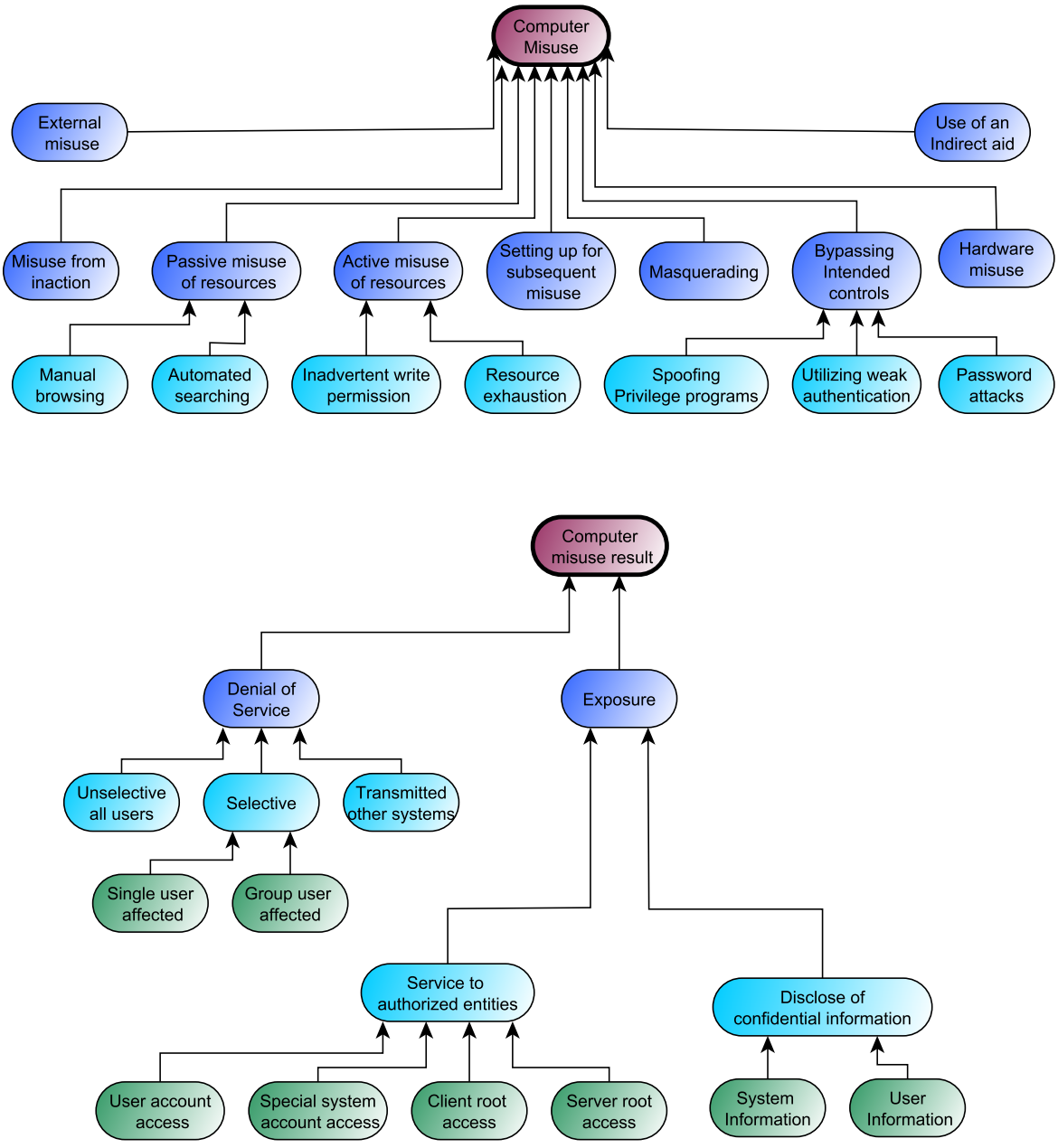


Figure 3.6: Classification of Computer Misuse and the Results of Computer Misuse after Lindqvist and Jonsson (1997)

by Hansman and Hunt's taxonomy, as shown in Figure 3.8. Hansman and Hunt's target class has three main distinctions — hardware, software and network — all referring to physical elements that are targeted. The *Target* class developed in Section 4.2.13 uses definition, but also adds an *Industrial Equipment* class. The three vulnerabilities; Configuration, Design and Implementation, are used in the *Vulnerability* class developed in Section 4.2.14. Some payloads shown in Figure 3.8, namely *Disclosures of Information* and *Corruption of Information*, correspond to the *Attack Goal* sub-classes: *Steal Secret* and *Change Data*.

The *Attack Methodology* (Figure 3.7) class of Hansman and Hunt only lists attack methods and does not group them according to a higher attack methodology goal. The human-based attack, such as social engineering and phishing, is also required. The *Network Attack Dimensions* (Figure 3.8) class only addresses the scope of the attack under the sub-class *Miscellaneous*, and not as a specific sub-class. Only the physical properties of an attack is mentioned, without investigating the origin of the attack.

3.3.3 Hacking? How They Do It, by CERT-In (2003)

The Indian Computer Emergency Response Team (CERT-In) conducted a crime and security survey of computer-related crime incidents (CERT-In, 2003). They made the following findings regarding hacking:

- The biggest financial loss was caused by theft of secret and proprietary information.
- The second biggest loss was due to denial of service attacks.
- Virus incidents and insider abuse were the most frequent types of incident.
- More than two-thirds of corporate companies are against hiring hackers that have been reformed.

CERT-In developed an attack methodology in their survey. In this methodology they examined the effect of hacking, the type of attacks, exploit types, vulnerabilities and hacking tools (Figure 3.9). The methodology was developed to enable administrators to maintain a constant watch over malicious code, and enable them to immediately update their security protection solution. Thus providing for rapid, timely patching.

Within the researcher's *Attack Mechanism* (as presented in Section 4.2.6), the *Malware* sub-class was derived from CERT-In's similar class. The *Popular Vulnerabilities*, *Attack Methods* and *Effect of Hacking* classes also influenced the researcher's *Attack Mechanism*

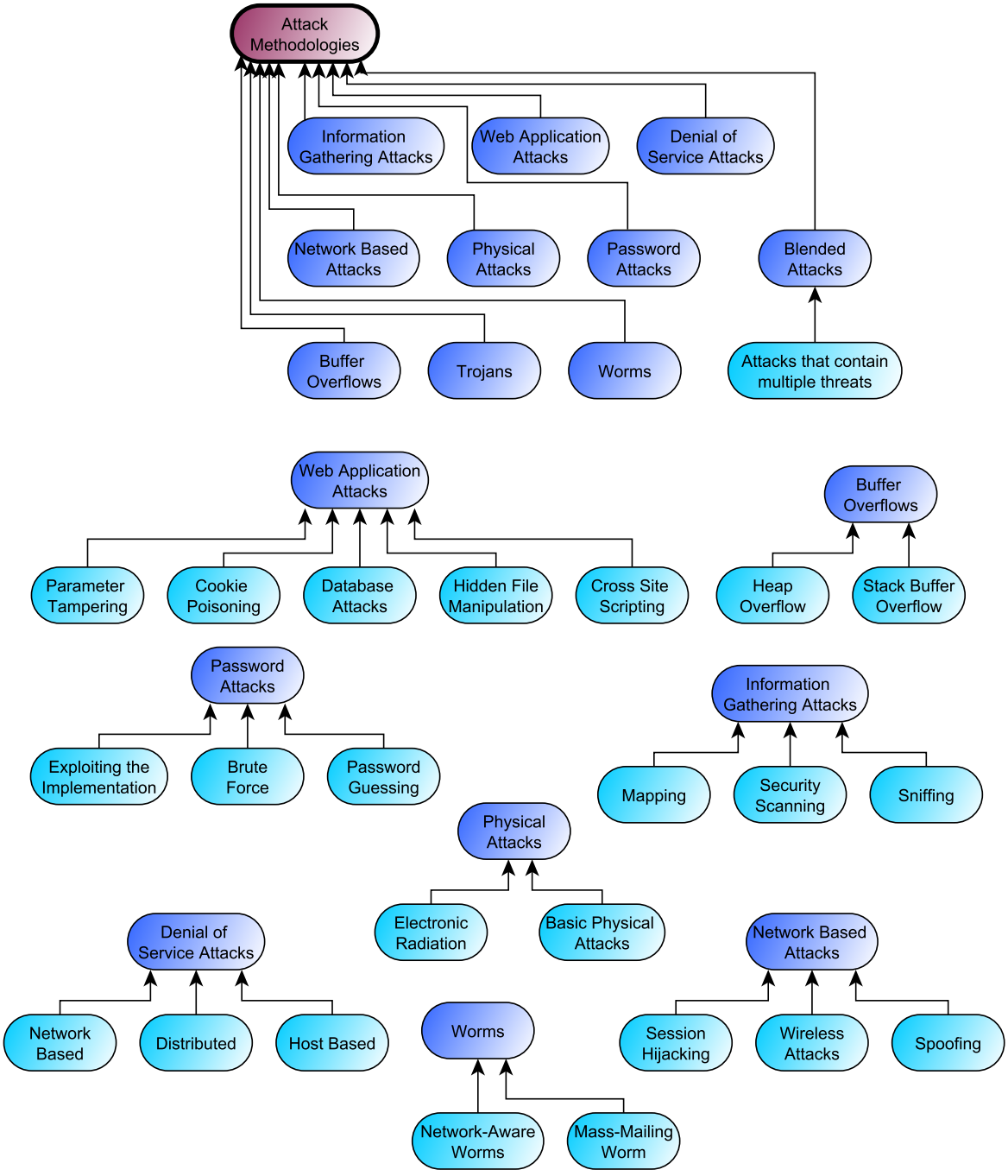


Figure 3.7: Attack Methodologies after Hansman and Hunt (2003)

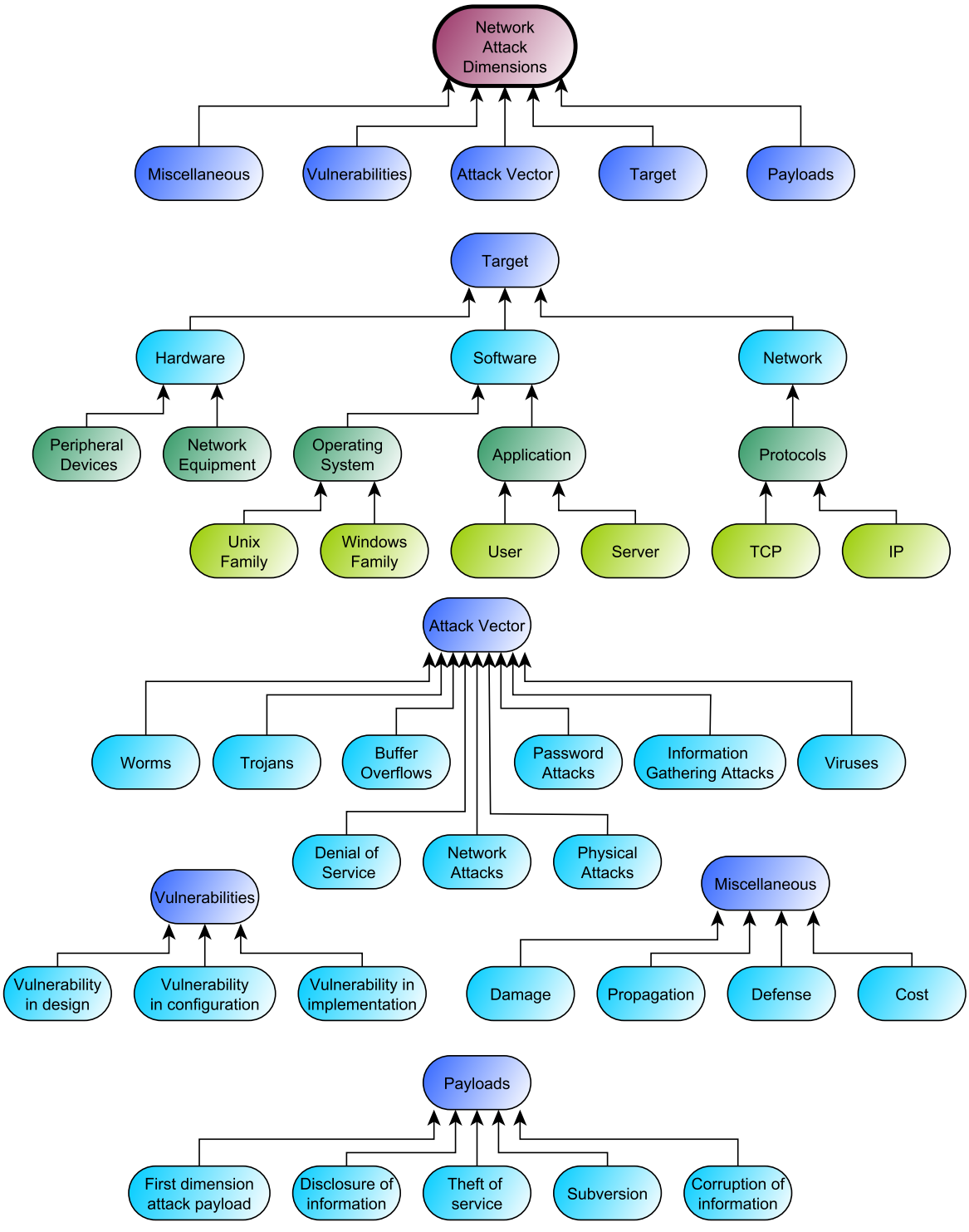


Figure 3.8: Hansman and Hunt (2003) Attack Taxonomy

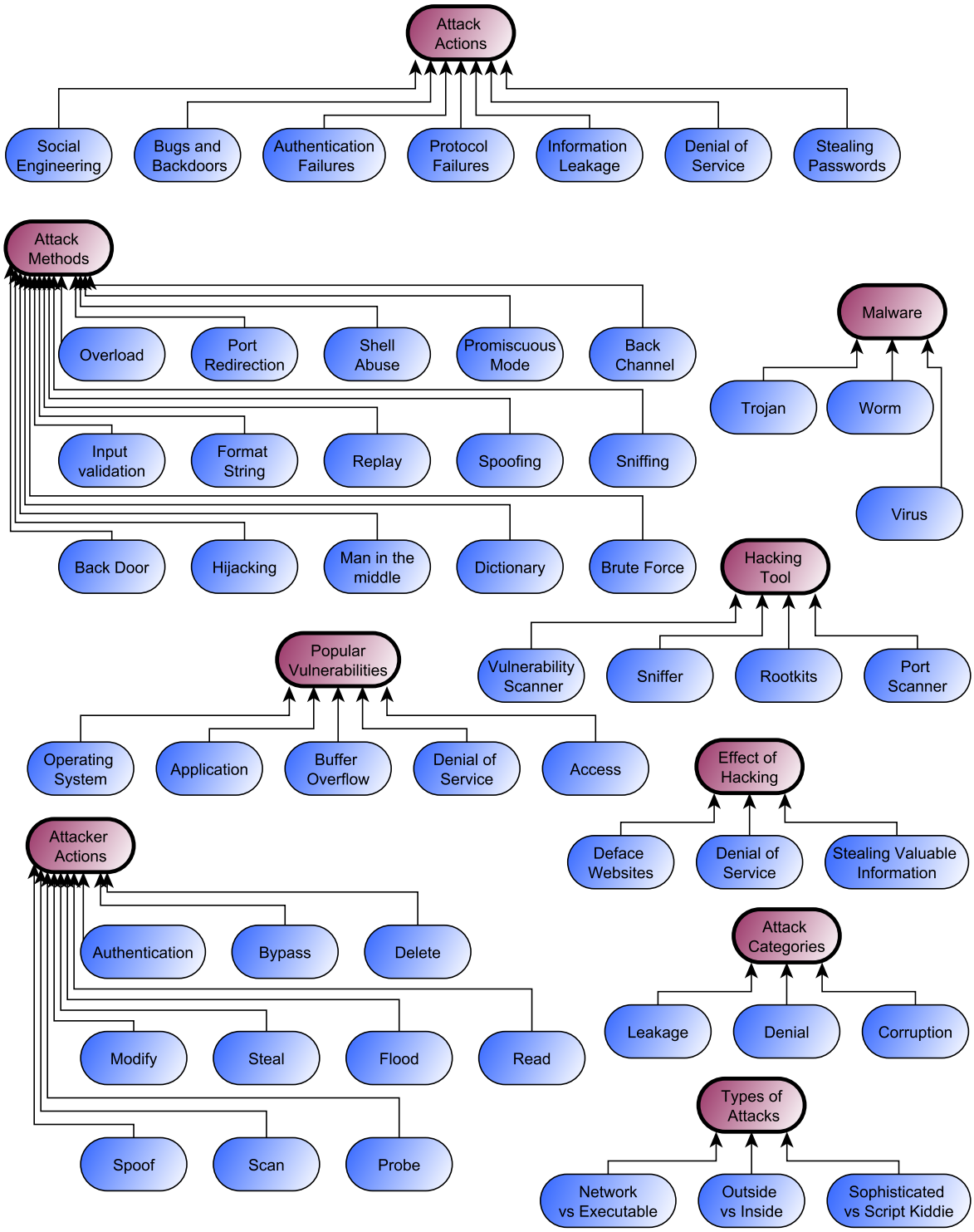


Figure 3.9: CERT-In (2003) Effect of Hacking, Malware, Popular Vulnerabilities, Attack Methods, Hacking Tools, Types of Attacks, Attack Actions, Attacker Actions and Attack Categories

class. CERT-In classes list the different attack methods in detail, which can be reduced to a simpler class, such as the first sub-class of the researcher's *Attack Mechanism* class, namely: Access Attack, Information Gathering and Data Manipulate. The critique holds for their *Popular Vulnerabilities* class which is oversimplified and does not present any relationship to the vulnerabilities. Their *Effect of Hacking* class is also too limited, without any references to other hacking effects, as listed by the researcher in Section 4.3.

3.3.4 Anatomy and Types of Attacks against Computer Networks by Tutănescu and Sofron (2003)

Tutănescu and Sofron (2003) described active and passive computer network attacks. These attack types are shown in Figure 3.10. The *Active Attacks* and *Passive Attacks* only

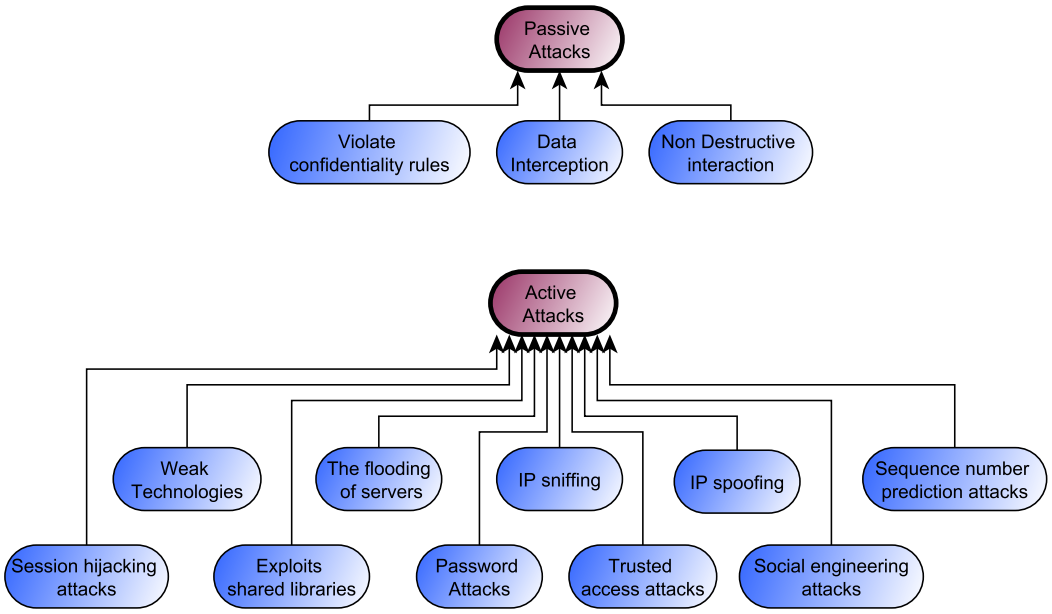


Figure 3.10: Active and Passive Attacks by Tutănescu and Sofron (2003)

describe a single property of attacks. This property does not add value to the researcher's taxonomy, and Tutănescu and Sofron's classification of *IP sniffing* could also fall within the passive class. Passive attacks formed part of the researcher's *Target Identification* (Section 4.2.10) sub-class.

3.3.5 An Ontology for Network Security Attacks by Simmonds *et al.* (2004)

Simmonds *et al.* (2004) defined an extensible ontology for network security from teaching a network security subject at the University of Technology in Sydney. Although Simmonds *et al.* refers to an ontology, the paper presents an extensive taxonomy that is presented in Figure 3.11. A map was developed to demonstrate the vulnerability relationships (Figure 3.12).

The Simmonds *et al.* *Actor*, *Asset*, *Outcome* and *Motive* classes directly influenced the author's *Actor*, *Asset*, *Effect* and *Motivation*. The researcher's classes are presented in sections 4.2.1, 4.2.4, 4.2.8 and 4.2.9. Their taxonomy used a *Fault* class to describe vulnerabilities. This class presents similar information to the *Vulnerability* class (Section 4.2.14), but differentiated according to structure, not methodology. Their *Attack On* class is a direct mapping to the traditional Confidentiality, Integrity and Availability Authentication (CIA+) mapping of security threats. The researcher's *Attack Goal* (Section 4.2.5) presented the same information and added a sub-class to present indirect attack goals. The CIA+ is an expansion of the popular Confidentiality, Integrity and Availability (CIA) triad (Anderson, 2003; Whitman and Mattord, 2011), as shown in Figure 3.13.

3.3.6 Taxonomies of Cyber-adversaries and Attacks by Meyers *et al.* (2009)

Meyers *et al.* (2009) proposed a taxonomy of cyber-adversaries based on the work done by Rogers (2006). This taxonomy was presented as a two-dimensional circumplex (Figure 3.14) image in which motivation is presented along the circumference and where the sophistication level increases with the radius. Each quadrant represents: Revenge, Financial, Notoriety and Curiosity.

Script kiddies, newbies and novices are adversaries with limited programming skills, who are new to hacking and rely mainly on prewritten tools. Hacktivists and political activists are motivated by political cause and not necessarily by personal gain. Cyberpunks, crashers and thugs are attention-seeking hackers with more programming skills than novices. Insiders and user malcontents are considered by many as the greatest risk (Rogers, 2006; Gellers, Brant, and B., 2008; Meyers *et al.*, 2009). Insiders, due to their specialised knowledge, can cause a very large amount of damage. Coders and writers are

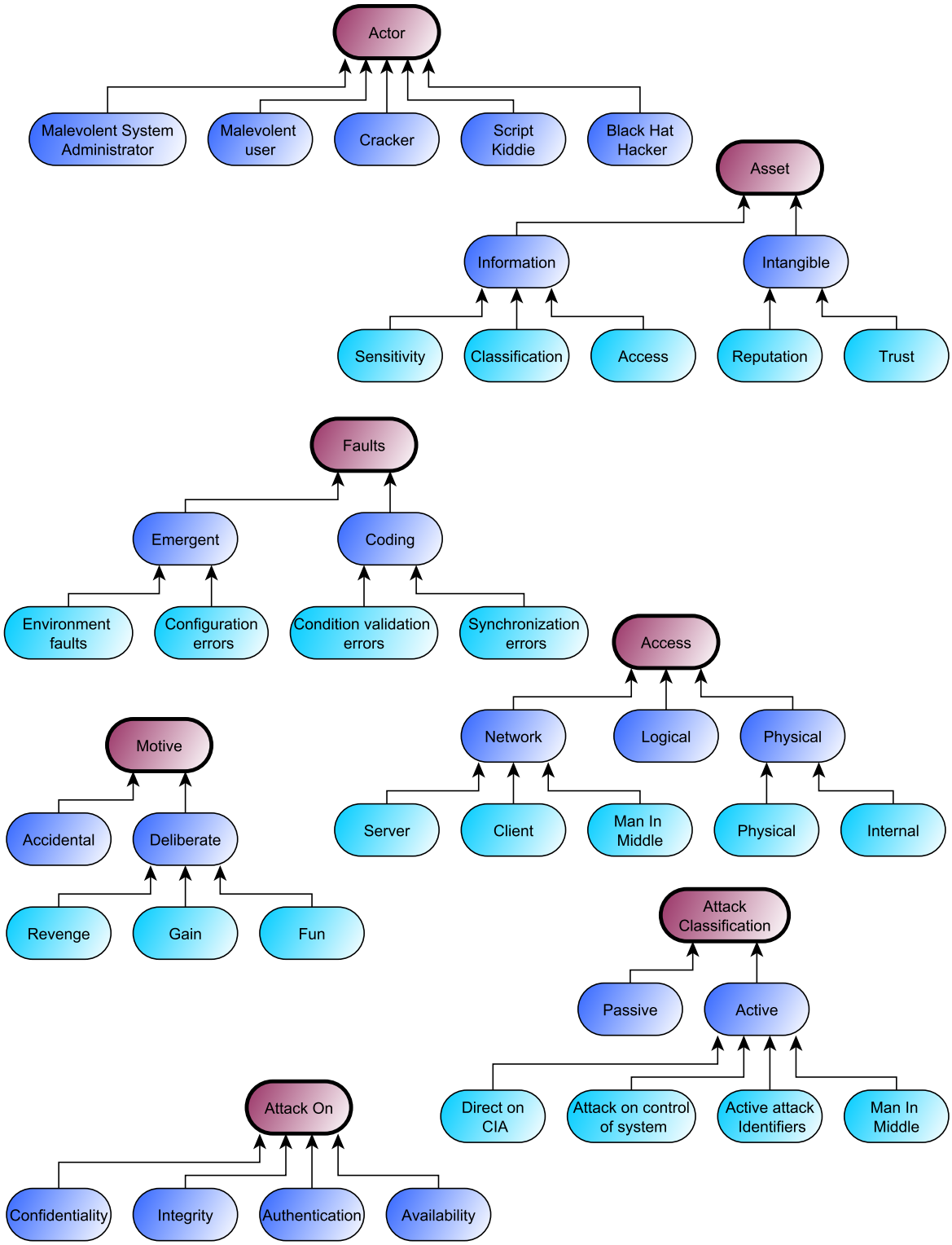


Figure 3.11: Network Security Attacks Taxonomy by Simmonds *et al.* (2004)

	Security Policy	Technology
Short-term	**Social Engineering:** Information fishing Trojan	**Logic Error:** Bugs OS vulnerabilities Network Protocol Design
Long-term	**Policy Oversight:** Poor Planning Poor Control	**Weakness:** Weak passwords Old encryption standards

Figure 3.12: Vulnerability Map after Simmonds *et al.* (2004)

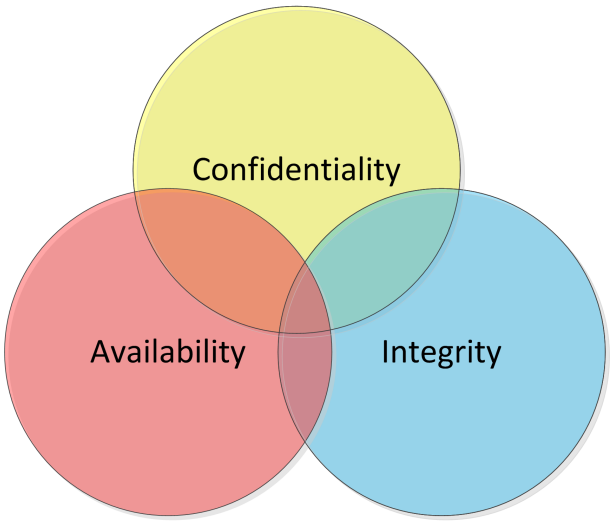


Figure 3.13: CIA Triad

primarily used to write code for use by the other groups. White hat hackers, old guard and sneakers are hackers without malicious intent, who have no regard for privacy or secrecy. Black hat hackers, professionals and elite are professional hackers who sell their skills to the highest bidder. These adversaries can be employed by organised crime syndicates. Cyber-terrorists are skilled hackers that engage in state-sponsored information warfare. Meyers *et al.*'s list of cyber-adversaries are contained in the *Actor*, *Aggressor* and *Motivation* classes (Section 4.2.1, 4.2.3, 4.2.9).

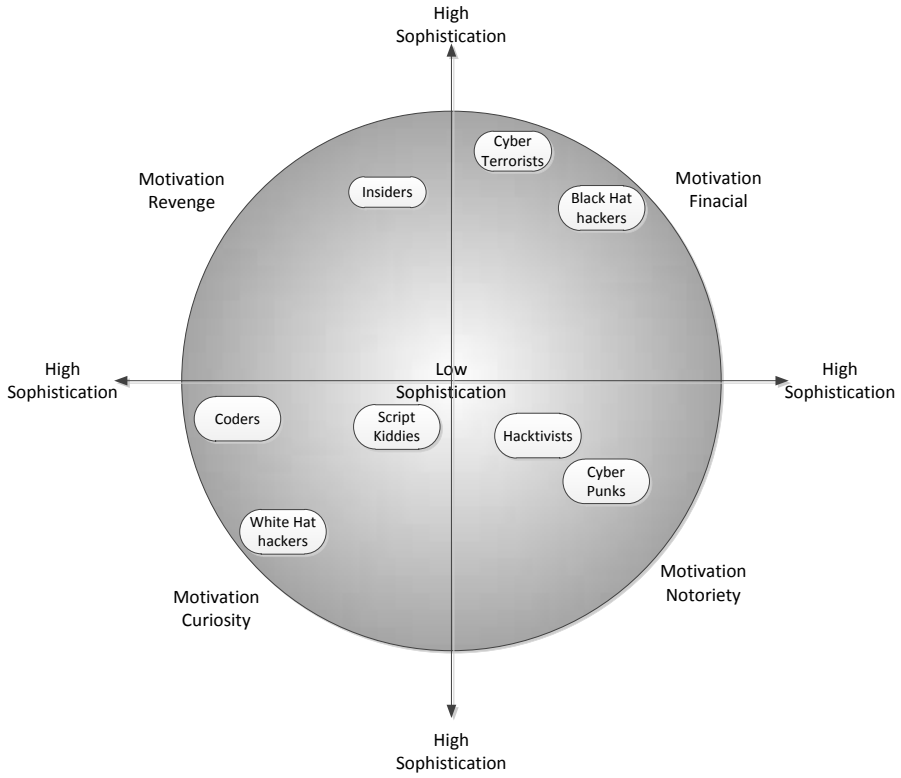


Figure 3.14: A Circumplex of Adversaries after Meyers *et al.* (2009)

3.3.7 Diversity in Network Attacker Motivation: A Literature Review by Rounds and Pendgraft (2009)

Rounds and Pendgraft (2009) investigated the diversity in network attacker motivations and compiled a list of possible hacker agents (Figure 3.15). The researcher's *Actor*, *Aggressor* and *Motivation* classes were influenced by their review of attacker motivation. These classes are presented in sections 4.2.1, 4.2.3 and 4.2.9. Rounds and Pendgraft's class does not separate between who is the attacker and who is sponsoring the attacker.

3.3.8 Dimension of Cyber-Attacks by Gandhi *et al.* (2011)

The goal of Gandhi *et al.* (2011)'s goal was to thoroughly understand a cyber-attack by studying the nature and the motivation behind it. The taxonomy developed by Gandhi *et al.* is shown in Figure 3.16. They noted that a hacker's motivation can be classified into three classes: political, sociocultural and economical. These classes can overlap, as shown in Figure 3.17.

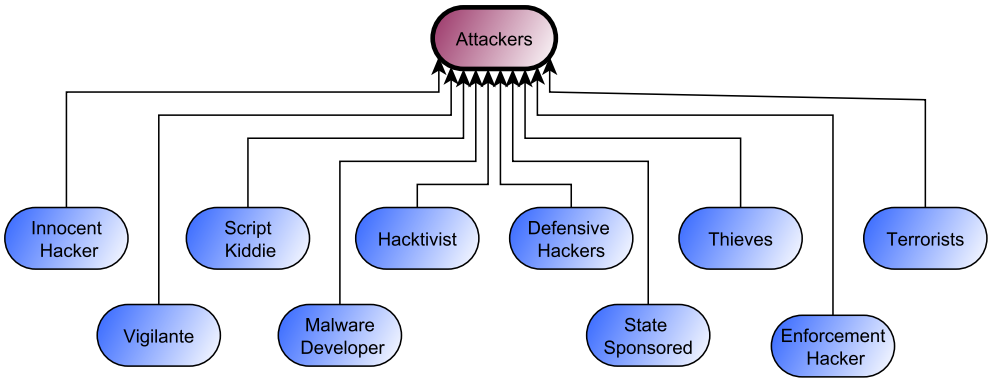


Figure 3.15: Hacker Agents after Rounds and Pendgraft (2009)

The *Aggressor* and *Actor* classes of Section 4.2.3 and 4.2.1 have similarities with the Gandhi *et al. Attack Agent* and *Attack Co-ordination* classes. For example, *Nation States*, *Script Kiddie*, *Hactivists*, *Cybervigilante*, *Cyber-Mafia, Organised Crime* can directly be mapped to the classes presented in sections 4.2.3 and 4.2.1.

Gandhi *et al.*'s *Attack Victim* class represents the main targets of cyber-attacks. These victims are presented as the scope of an attack in Section 4.2.12. The *Attack Consequences* class of Gandhi *et al.* Describes, with over 15 classes, what effects or consequences cyber-attacks have. These consequences are presented in Section 4.2.11 as the *Sabotage* class.

The *Attack Motive* presented by Gandhi *et al.* has three main classes: *Socio-cultural*, *Economic* and *Political*. Gandhi *et al.* state that motivation is not set, but could contain a combination of factors. How the factors can be combined is shown in Figure 3.17 as the *Attack Motive* class.

This taxonomy had only a simplistic method to vulnerabilities and attack mechanisms. The target and its properties were not presented in the same detail as the attacker, aggressor and attack agent, which was developed in detail by Gandhi *et al.*

3.3.9 The Scrap Value of a Hacked PC, revisited by Krebs (2012)

Krebs (2009, 2012); Cardenas, Radosavac, Grossklags, Chuang, and Hoofnagle (2010) compiled lists of all the methods in which a compromised PC can be used for financial gain. The goal of these lists are to demonstrate why someone would want to hack a PC (Figure 3.18). Krebs' classes are contained within the researcher's *Attack Goal* class. More specifically, the *Steal Data* and *Gain Control* sub-classes presented in Section 4.2.5 were

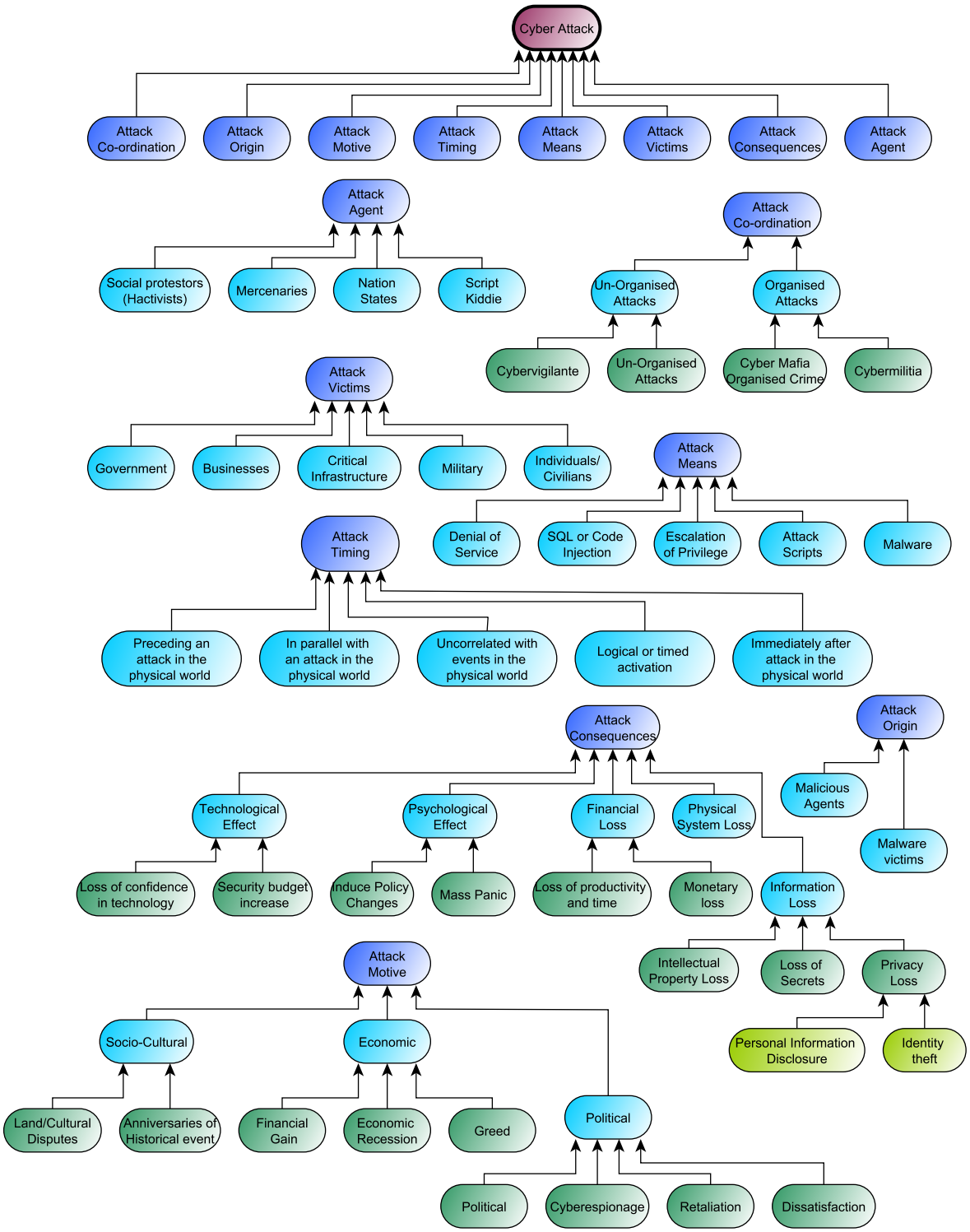


Figure 3.16: Dimensions of Cyber-Attacks after Gandhi *et al.* (2011)

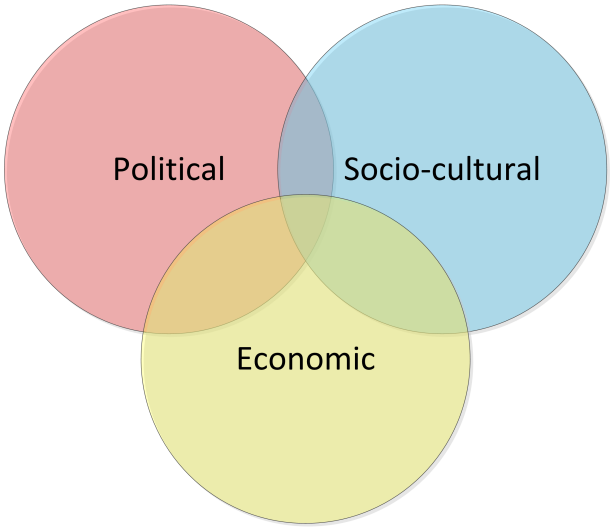


Figure 3.17: Motivation Classes after Gandhi *et al.* (2011)

influenced by Krebs's classes. Krebs's reasons for hacking a Personal Computer (PC) are extensive and can be used when analysing the reasons for attacking computers in detail.

3.3.10 Common Attack Pattern Enumeration and Classification

Common Attack Pattern Enumeration and Classification (CAPEC) is publicly available lists of common attack patterns and a classification taxonomy[1]. An *attack pattern* is defined as an abstraction of the mechanism employed by an attack. Each pattern defines a challenge that an attacker faces and provides a description of the technique(s) used by the attacker to overcome the problems faced in executing an attack. Recommended methods for mitigating an actual attack are also listed. The comprehensive CAPEC lists are community-developed and freely available[2]. The CAPEC lists' goal is to be as comprehensive as possible, and thus provides too much detail and information to be useful.

3.3.11 Taxonomies Overview

In this section, several network attack-related taxonomies are presented. Hansman and Hunt (2003), CERT-In (2003), Simmonds *et al.* (2004) and Gandhi *et al.* (2011) developed taxonomies that cover most aspects of network attacks. The motivations behind attacks

[1] http://capec.mitre.org/
[2] http://capec.mitre.org/data/index.html/

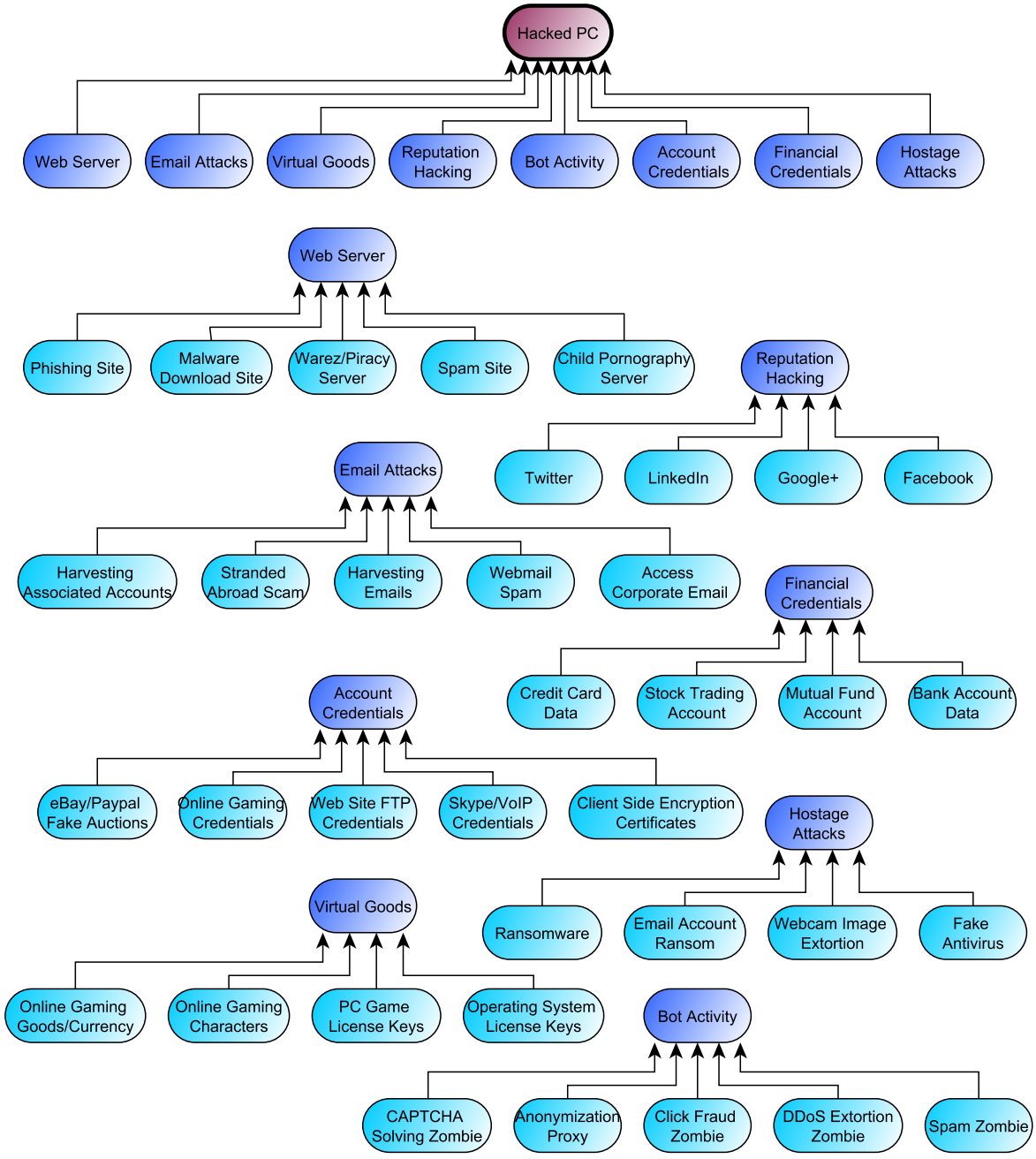


Figure 3.18: Reasons for Hacking a PC after Krebs (2012)

were probed by Rounds and Pendgraft (2009), whereas Krebs (2012) investigated all the different ways to make money from attacking a PC. The CAPEC lists provide a comprehensive enumeration of attack patterns. The taxonomies listed in this section contributed to the researcher's taxonomy, as presented in Chapter 4.

3.4 Ontologies used to Detect Computer-based Attacks

The use of ontologies in the study of network attacks is relatively new and not much has been published on this. In this section, four network attack-related ontologies are presented. Most of these ontologies have not used automated reasoners to infer information, such as those developed by the researcher in Chapter 5.

Gruber (1993) describes an ontology as: "a specification of a representational vocabulary for a shared domain of discourse – definitions of classes, relations, functions, and other objects...". Noy and McGuinness (2001) defined an ontology as: "... a common vocabulary for researchers who need to share information in a domain ... includes machine-interpretable definitions of basic concepts in the domain and relations among them" Grüninger and Fox (1995) state: "An ontology is a formal description of objects, properties of objects and relations among objects". They list the following motivations for developing an ontology:

- sharing a common understanding of the structure of information
- facilitating re-use of domain knowledge
- making domain assumptions clear
- separating domain knowledge from operational knowledge
- analysing domain knowledge

The research of Simmonds *et al.* (2004) into network security ontology concentrates mainly on the taxonomy and presents a very limited ontology. In sections 3.4.2 to 3.4.5, research into the use of network attacks is presented in chronological form.

3.4.1 RBAC Policy Engineering with Patterns by Rochaeli and Eckert (2005)

Rochaeli and Eckert (2005) proposed an ontological approach to construct the knowledge representation framework for computers and their vulnerabilities. A framework was de-

veloped in which security administrators can search for patterns that match the scenario-oriented risks and thus specify policies with the help of experts' knowledge. With this framework, security administrators can interpret their detected scenario, compare with patterns that match the scenario and assert instances of the scenario into the knowledge representation framework.

3.4.2 An Ontology-supported Outbound Intrusion Detection System by Mandujano (2005)

Mandujano (2005) developed an ontology to generate and identify attack programme signatures. Mandujano used Snort[3] to match signatures as input data for their ontology. This ontology is aimed at the packet-level intrusion detection and is therefore more suited to a sensor within the researcher's system.

3.4.3 An Ontology-based Intrusion Alerts Correlation System by Li and Tian (2010)

Li and Tian (2010) developed an ontology-based intrusion alerts correlation system. This system consisted of agents and sensors, where the sensors collate security information and agents process the information.

An automated reasoner was used to determine attack sessions and classes that could be used to determine risk. The attack classes have to be analysed offline by experts. Li and Tian's ontology cannot handle new types of attacks in real time as their knowledge base has to be updated for new attacks.

This is a design feature. Their ontology is shown in Figure 3.19. The researcher's taxonomy and ontology also have *Asset*, *Vulnerability* and *Attacker* classes and *hasAttacker* and *hasVulnerability* relationships. Li and Tian's *Address* class is presented by the author as *Actor Location*.

[3]http://www.snort.org/

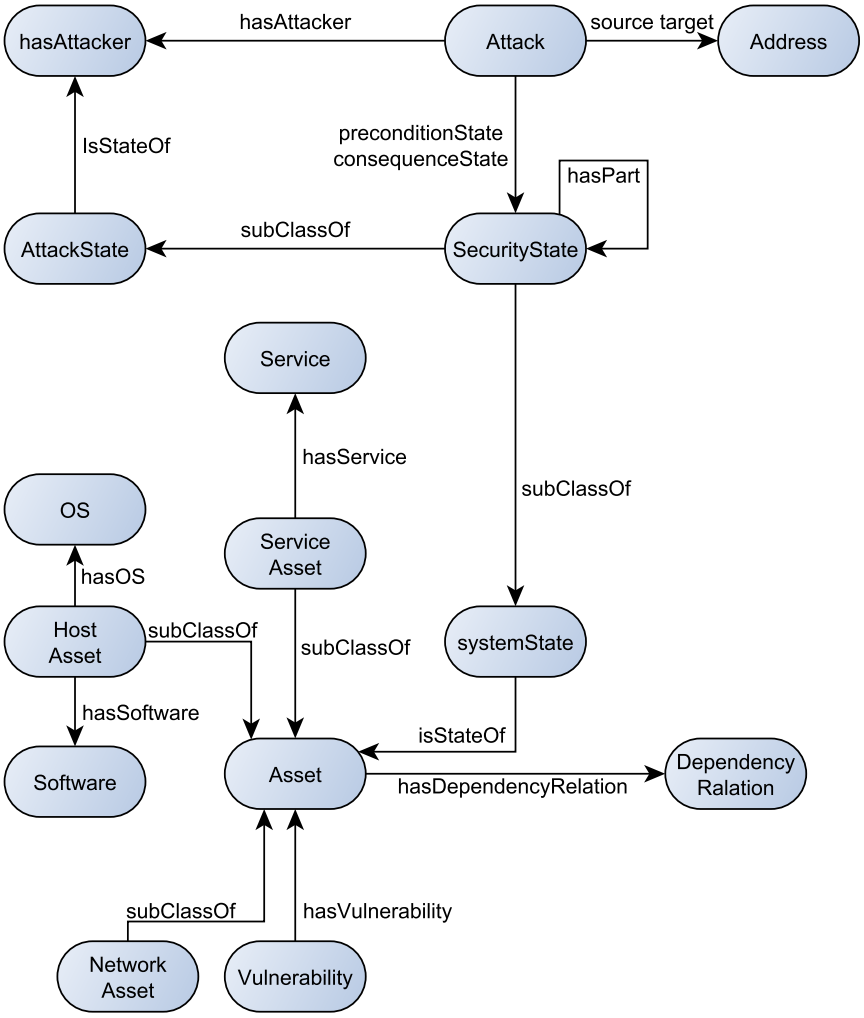


Figure 3.19: Alert Correlation Ontology (Li and Tian, 2010)

3.4.4 Ontology-based Distributed Intrusion Detection System by Abdoli and Kahani (2009)

Abdoli and Kahani (2009) developed a system that uses IDSagents and a special Master-Agent for intrusion detection. The MasterAgent contains the attack ontology. When an IDSagent detects an attack, a detection report is sent to the MasterAgent, which extracts the semantic relationships. Their system was able to reduce false negatives and false positives. Abdoli and Kahani's ontology is presented in Figure 3.20.

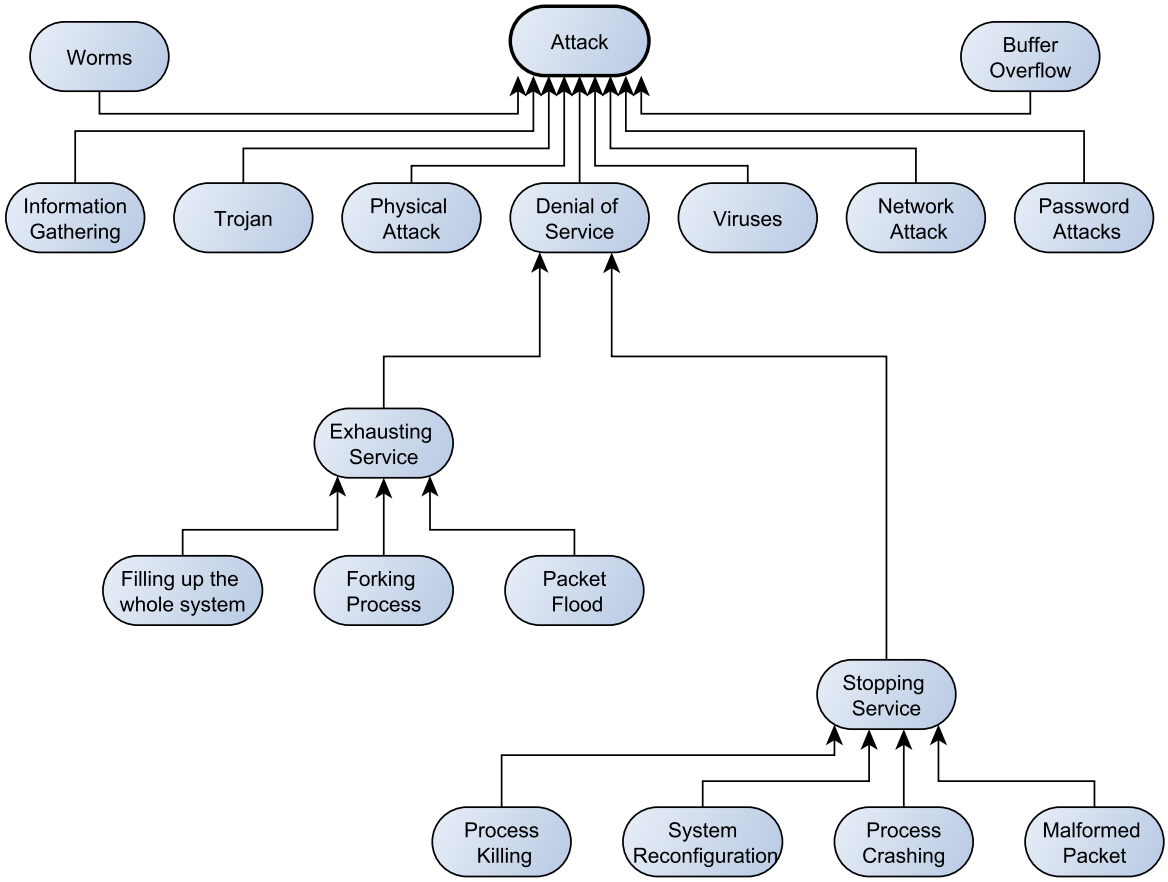


Figure 3.20: Abdoli and Kahani (2009)'s Attack Ontology

This ontology only presented attacks and their sub-classes, and relationships and other ontology properties were not developed.

3.4.5 An Ontology-based System to Identify Complex Network Attacks by Frye *et al.* (2012)

Frye *et al.* (2012) used an ontology to determine what constitutes a network attack. They developed an ontology with four main classes: Availability, Recon, GainAccess, and ViewChangeData, and a separate ontology that presents complex attacks. The complex ontology (Figure 3.21) with four main sub-classes is similar to four of the scenarios developed in Section 2.5. The relationships between the classes were only developed to a limited extent to the levels of sub-class "intersection", "union" or "contains".

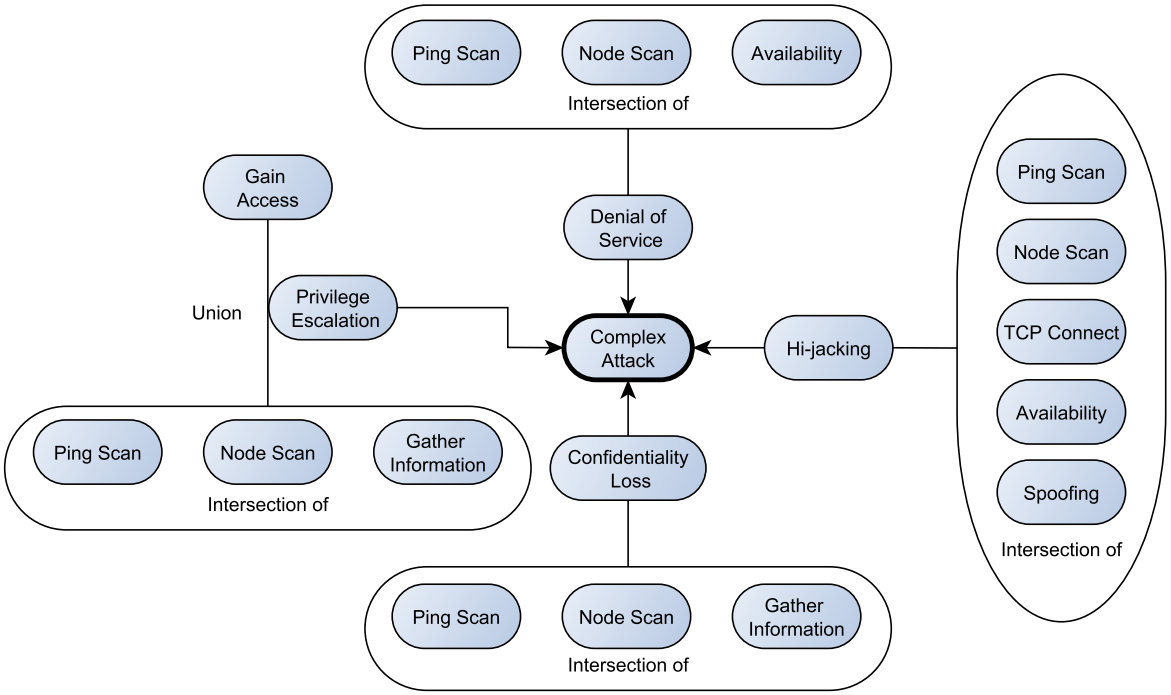


Figure 3.21: Complex Attack Ontology (Frye *et al.*, 2012)

3.4.6 Ontologies Overview

The use of ontologies to describe network attacks is still very limited and very few research papers have been published in this field of study. The ontologies vary significantly and thus no constant theme can be derived. For example, Frye *et al.* (2012) have only four main classes without specifying their relationships.

3.5 Network Attack Sensors

Network attacks are measured by integrating information from various sensors. The function and scope of these sensors are investigated in this section. The main sensors that are used to detect computer network attacks are Intrusion Detection Systems (IDSs). Mukherjee, Heberlein, and Levitt (1994) defined network intrusion as:

> "the problem of identifying individuals who are using a computer system without authorization (i.e., 'crackers') and those who have legitimate access to the system but are abusing their privileges (i.e., the 'insider threat')".

Mukherjee *et al.* (1994) also postulated:

> "IDS's are based on the belief that an intruder's behavior will be noticeably different from that of a legitimate user and that many unauthorized actions are detectable."

In this section, the type of sensors available and taxonomies of IDS are presented. Only two IDS research papers are presented in sections 3.5.3 and 3.5.4 due to the researcher's interest in the types of taxonomy of IDS, and not the details of their design or performance. In this subsection the characteristics and methods of sensors are presented to present a comprehensive picture on when and how sensors are used.

IDSs can either be host based or network-based (Anderson, 1980; Lunt and Jagannathan, 1988; Lunt, 1993; Mukherjee *et al.*, 1994; Kuwatly, Sraj, Al Masri, and Artail, 2004; Garcia-Teodoro, Diaz-Verdejo, Macia-Fernandez, and Vazquez, 2009). Network-based Intrusion Detection System (NIDS) monitor network traffic, by examining network packets. NIDS monitors the IP address, ports and the data segments of packets, whereas traditional firewalls only monitor the Internet Protocol (IP) addresses and ports of data packets (Kachirski and Guha, 2003). Host-based Intrusion Detection Systems (HIDSs) examine data that is held on individual systems, which are used to protect a single system or a single data source. Crosbie and Spafford (1995) and Balasubramaniyan, Garcia-Fernandez, Isacoff, Spafford, and Zamboni (1998) stated that for an IDS to be effective, it must have the following characteristics:

- It must run continually with minimal human supervision.
- It must be fault tolerant in the sense that it must be able to recover from system crashes, either accidental or caused by malicious activity.
- It must resist subversion. The IDS must be able to monitor itself and detect if it has been modified by an attacker.
- It must impose a minimal overhead on the system where it is running, so as to not interfere with its normal operation.
- It must be able to be configured according to the security policies of the system that is being monitored.
- It must be able to adapt to changes in system and user behaviour over time.

The Hybrid IDS system has been developed to use network- and host-based characteristics (Day, Flores, and Lallie, 2012; Aydin, Zaim, and Ceylan, 2009). Abraham and Thomas (2005) developed a Distributed IDS system that consists of multiple IDSs over a network, which all communicate with each other, or with a central server.

Mandujano (2005) determined that three main classes of intrusion detection agents exist:

- Sensors: Sensors are responsible for collecting data, and are divided into Traffic Sensors or Process Sensors, depending on the source of data.
- Correlators: Correlators receive data from sensors and examine the data for events.
- Reactors: Reactors are triggered after specific events, and are divided into Guards (that execute locally) and Tracers (that execute externally).

3.5.1 Anomaly and Misuse Detection

Stiawan, Idris, Ihsan, Hussain, and Abdullah (2011); Peddabachigari, Abraham, Grosan, and Thomas (2007); Wu and Banzhaf (2010) describe two main intrusion detection methods:

- anomaly detection; and
- misuse detection.

Misuse detection uses predefined signatures to search for matches for known intrusion behaviour or known malware. Anomaly detection looks for statistical differences between normal system behaviour and user behaviour. Behaviour that differs significantly from the statistical norm is classified as malware or an intrusion.

Aydin *et al.* (2009) discussed the advantages and disadvantages of the two types of detection systems. Anomaly detection can detect attacks even if previous information about the attack method is not available, but it has a high false positive rate. Anomaly detection systems also require a large training data set. Misuse detection systems provide a simple way of monitoring computer systems without the requirement of training data, but only previously characterised attacks can be identified.

Idika and Mathur (2007) made a survey of malware detection methods. They listed three main methods: anomaly, specification and signature methods. Each of these methods could then be split into a dynamic, static or hybrid approach to malware detection.

3.5.2 Threat Detection

Lunt and Jagannathan (1988) and Lunt (1993) developed an Intrusion Detection Expert System (IDES) that learns the behaviour patterns of users. Thus threats were detected by users or the computer system alternated suddenly from its usual behaviour. Two categorical measures were used:

- discrete measure: a function that uses finite measure of a user's behaviour, such as user time and location of login; and
- continuous measure, a function that changes during usage, such as average Central Processing Unit (CPU) use, and Input and Output (IO) activity.

Stiawan *et al.* (2011) investigated methods whereby threats can be identified. The following methods have been incorporated by other researchers:

- The use of Domain Name System (DNS) Blacklists to stop man-in-the-middle attacks have been investigated by Ramachandran, Dagon, and Feamster (2006).
- Internet Assigned Numbers Authority (IANA) port numbers can be used to identify botnet activity (Karasaridis, Rexroad, and Hoeflin, 2007).
- URL and IP Block Blacklists are used to prevent access to malicious Internet sites (Dietrich and Rossow, 2009).
- Common Vulnerability and Exposure (CVE) lists are used to identify network vulnerabilities (FIRST-Forum, 2007). Attack methodologies and the source of attackers can be determined from honeypot data (Provos, 2004).
- Intercepting and analysing traffic flows can be used to identify network intrusions (Sperotto, Schaffrath, Sadre, Morariu, Pras, and Stiller, 2010).
- Data stored in logs can indicate the presence of an attack and help the defender in managing the attack (Kent and Souppaya, 2006).
- Spam rules prevent unwanted email from spreading by stopping the email before it reaches its target (Madigan, 2005).
- Computer viruses are a significant risk to computer systems that require effective and timely response (Subramanya and Lakshminarasimhan, 2001).
- Computer security policy should be an integral part of securing any computer network (Sterne, 1991).
- IDS alerts indicate that an attack may be in progress and that a computer is vulnerable (Gula, 2011).
- Web Crawler data can be used to identify malware hosted on Internet sites (Moshchuk, Bragin, Gribble, and Levy, 2006).
- Regular Expression pattern matching can be used to detect attacks in raw throughput data (Vasiliadis, Polychronakis, Antonatos, Markatos, and Ioannidis, 2009).

3.5.3 Taxonomy for Intrusion Detection Systems by Debar *et al.* (2000)

Debar *et al.* (2000) constructed a taxonomy for intrusion detection systems. They build from the efficiency measurement defined by Porras and Valdes (1998). The measures are: Accuracy, Performance, Completeness, Fault Tolerance and Timeliness. Their five main concepts are: Detection Method, Behaviour on Detection, Audit Source Location, Detection Paradigm and Usage Frequency. These classes and the taxonomy sub-classes are:

- Accuracy: Reducing the occurrence of false positives. This refers to the classification of normal or legitimate traffic as malicious.
- Performance: The tempo or rate at which traffic can be analysed for malicious traffic.
- Completeness: The chance of detecting all attacks. This measure is difficult to measure and impossible to achieve. This measure can only be measured when an attack cannot be detected, which implies that the attack has been detected through some other method.
- Fault Tolerance: The intrusion detection system itself must be able to handle and defend the system against attacks. The detection systems itself must not add vulnerabilities to the system.
- Timeliness: The analysis and reporting of the intrusion detection system must occur timeously, to ensure that someone can act on the information it presents. If the detection or analysis of an attack is too slow, the relevance of identifying it may be lost.

Debar *et al.*'s intrusion detection concepts are shown in Figure 3.22.

3.5.4 Intrusion Detection Systems: A Survey and Taxonomy by Axelsson (2000)

Axelsson (2000) surveyed the field of intrusion detection and presented a taxonomy that described IDS systems with respect to their *System Characteristics* and *Detection Principles* (Figure 3.23). *Anomaly* detection principles depend on abnormalities in traffic rather than detecting known intrusions.

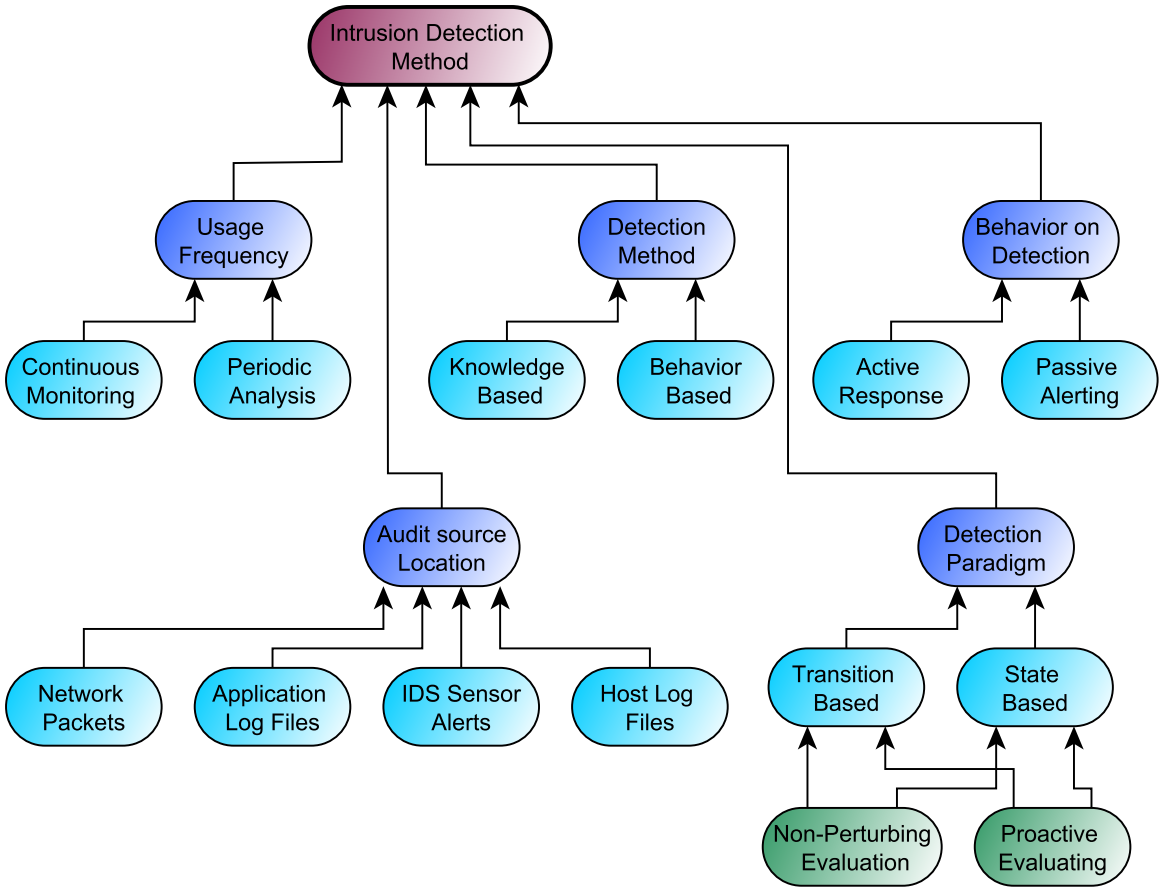


Figure 3.22: Intrusion Detection Concepts (Debar *et al.*, 2000)

Signature detection principles determine intrusions by comparing behaviour with a model of the intrusive process. These principles operate irrespective of user behaviour, by only looking for intrusion-like patterns. *Signature Inspired* detection principles use anomaly and signature principles to determine intrusive behaviour. *Programmed* methods require direct input (in the form of an algorithm or list) to detect what is a security violation. *Self-Learning* automatically learns suspicious behaviour after being trained through examples of normal and intrusive behaviour. *Time of detection* refers to real-time or postponed detection characteristics. *Granularity of Data-Processing* differentiates between continuously processing data or handling data in batches. The main *Source of Audit data* sources are either network data (for example multicast Ethernet streams) or host-based data (such as security, kernel, application, firewall, logs, etc.).

The *Response to Detected Intrusions* are passive or active. Passive responses notify the authority about the intrusion. Active responses can either try to thwart the attack by controlling the attacked system or neutralise the threat directly. The last option is con-

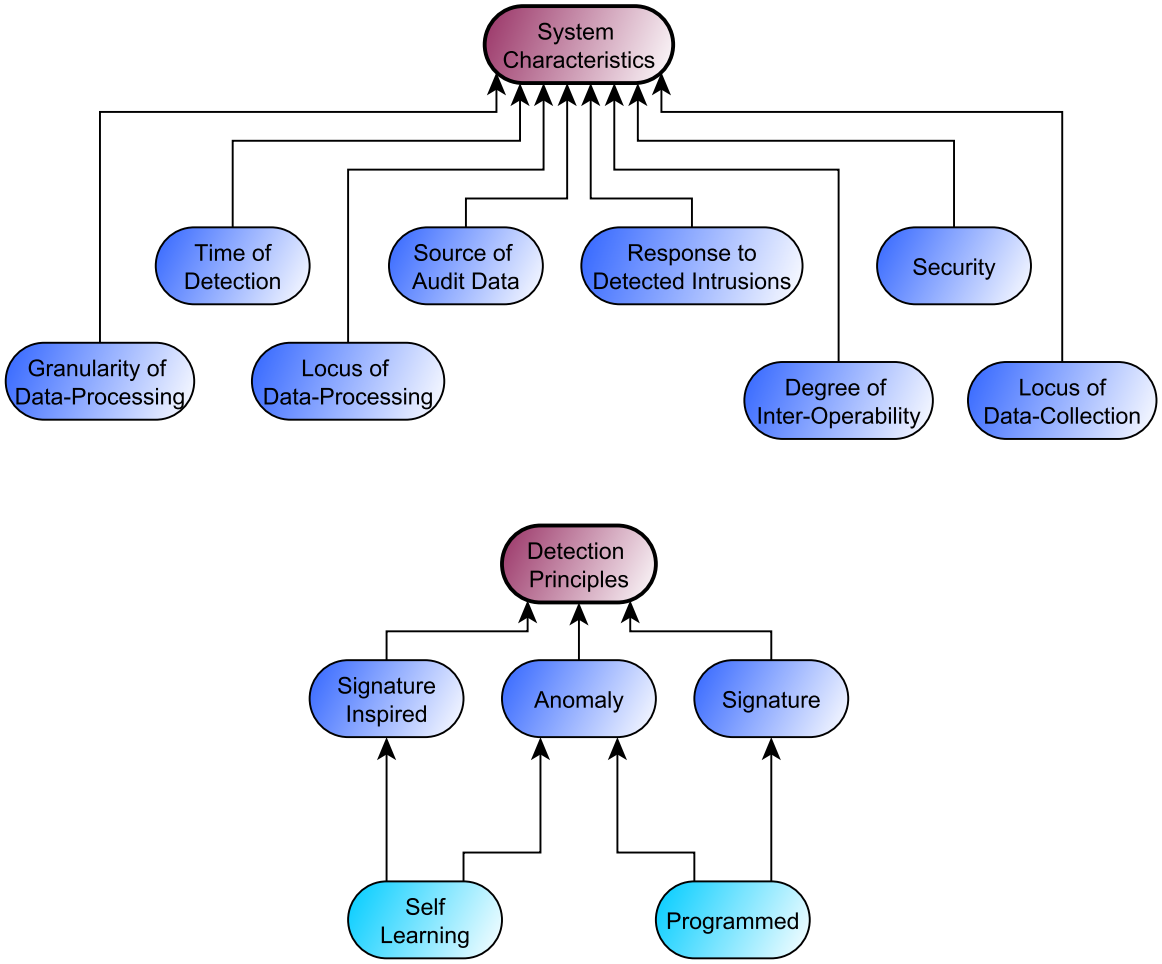


Figure 3.23: IDS System Characteristics and Detection Principles (Axelsson, 2000)

sidered a legal grey area and is fraught with legal dangers (Caltagirone and Frincke, 2005; Mansfield-Devine, 2009). *Locus of data processing* and *Locus of data collection* refer to the manner in which data is processed, namely: distributed or centralised.

The *Security* characteristic refers to the ability of the IDS system to withstand an attack on itself. The last characteristic found by Axelsson is *Degree of inter-operability*, whereby the IDS can co-operate in conjunction with other IDSs.

3.5.5 Network Telescope

A network telescope, also known as a darknet, Sinkhole, Internet Motion Sensor or Black Hole (Moore, Shannon, Voelker, and Savage, 2004; Harrop and Armitage, 2005; Bailey, Cooke, Jahanian, Myrick, and Sinha, 2006), is a network system that observes different

events taking place on a network. Network telescopes have been used to monitor the system for malicious Internet traffic (Irwin, 2011). The telescope observes traffic targeting the dark (unused) address space of the network. Since all traffic to these addresses is suspicious, information about possible network attacks can be obtained. The network telescope is an ideal sensor to detect threats because:

- only malicious traffic is captured; and
- the following types of malicious traffic can be detected:
  - random scanning malware (Moore *et al.*, 2004),
  - DDoS backscatter, (a DDoS attack using multiple spoofed addresses) (Moore, Shannon, Brown, Voelker, and Savage, 2006),
  - targeted scans.

3.5.6 Network Attack Sensors Overview

The best-known sensor to detect network attacks is an IDS. An IDS is typically either a host, a network or a combination of the two. These IDSs either detect anomalies or direct misuse. Threats can also be directly detected by learning users' patterns and detecting alternative behaviour, and indirectly by suspicious data though network telescopes.

3.6 Summary

This chapter introduced the academic base of the network attack model of Chapter 4, Section 4.4, the taxonomy of Chapter Section 4.2 and ontology of Chapter 5.3. Most of the network attack models presented in Section 3.2 have only a few basic steps. The network attack model developed in Section 3.2 uses very similar steps. Taxonomies that cover most of the aspects of network attacks are presented in Section 3.3. Only a limited number of ontologies are used to describe network attacks, and those presented in this chapter differ from each other. A comprehensive network attack ontology is presented in Chapter 5. IDS and network telescope-type systems are the main sensors that are available to detect network-based attacks. These sensors are either host or network based, or a combination of the two. In the next chapters, the attack model, taxonomy and ontology are developed in detail, based on the information provided in the presented literature study.

Part II

Theoretical

CHAPTER

FOUR

NETWORK ATTACK TAXONOMY

"You don't know the power of the Dark Side."

Darth Vader – Star Wars

4.1 Introduction

This chapter presents a taxonomy that describes network attacks and contains more than 15 main classes and more than 100 sub-classes. The researcher specifically developed it to address computer network-based attacks and it focuses on the classes that are required in order to specify such attacks. Several taxonomies have been developed by other researchers, as reviewed in Section 3.3. In this section, the taxonomies are presented in the form of diagrams, where each class and sub-class are represented. These taxonomies are either too wide or too focused for this research. The taxonomy developed by Hansman and Hunt (2003) provided the primary foundation for the author's taxonomy. According to Hansman and Hunt (2003), a taxonomy has the following basic requirements:

- Acceptability and usefulness: If the community accepts it, the taxonomy will be useful. This requirement can only be realised at a later time and is difficult and impractical to verify.

- Comprehensibility and unambiguousness: The taxonomy should be understood by novices as well as experts in the related field. No doubt should exist as to what a class or sub-class refers to. This taxonomy tries to enhance comprehensibility specifically by differentiating between attacker and defender. For example, the *Attack Mechanism* is not shortened to *Mechanism* to clearly differentiate between the mechanism used to attack and the mechanism under attack.
- Completeness: This requirement cannot be proven, or ever achieved. New technologies or too many subtle differences make it impossible to achieve completeness. The *Attack Mechanism* class has over 30 sub-classes, and these sub-classes can be further subdivided if required. For example, the *Virus* sub-class could be divided into *Boot Sector Virus*, *Polymorphic Virus* and *Macro Virus*.
- Determinism and repeatability: The class or sub-class that is used should be simple to determine. By clearly defining what each class and sub-class represents, individuals should find it simple to place. In most cases this could be easily achieved, except for classes that specify unclear concepts. For example, it is difficult to differentiate between the *Scope Size* class and *Medium Network* and *Large Network*.
- Existing terminology should be used to avoid confusion and to build on previous knowledge. Security terminology of previous taxonomies was used, where applicable.

The taxonomy developed in this chapter describes the attack from the point of view of the attacker (aggressor) and defender (target). Thus both sides of a network attack are described. The taxonomy presented in Section 4.2 was originally developed by the author and presented by van Heerden, Leenen, Irwin, and Burke (2012a), although some of the classes have since been updated. In Section 4.3, ten network attack scenarios are explored. These scenarios were originally developed by van Heerden, Burke, and Irwin (2012b). In Section 4.4, a temporal attack model is presented. Parts of this model were first presented by Grant, Burke, and van Heerden (2012). Section 4.5 concludes this chapter with a summary of the taxonomy.

4.2 Taxonomy of Network Attacks

The main classes in the author's Network Attack Taxonomy are shown in Figure 4.1. The classes are: *Actor*, *Actor Location*, *Aggressor*, *Asset*, *Attack Goal*, *Attack Mechanism*, *Attack Scenario*, *Automation Level*, *Effects*, *Motivation*, *Phase*, *Sabotage*, *Scope*, *Scope Size*, *Target* and *Vulnerability*. Each class has sub-classes that are described in the sections below.

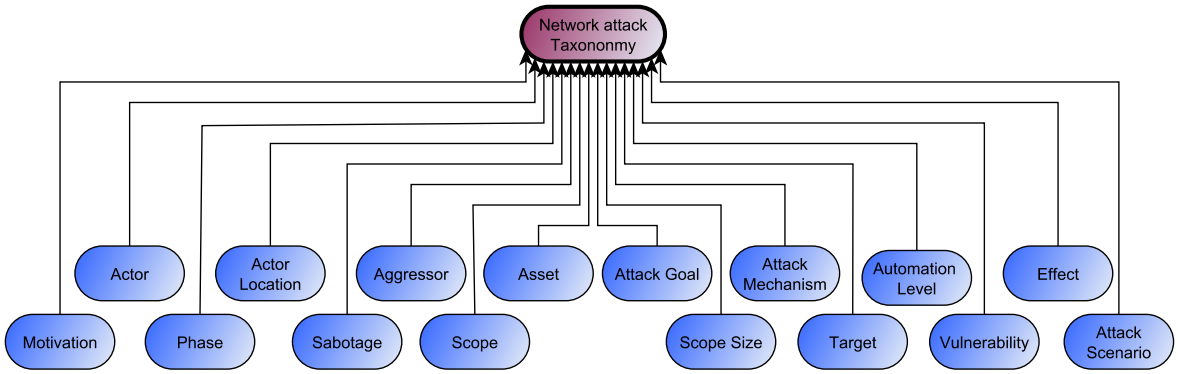


Figure 4.1: Network Attack Taxonomy

The main goal of this taxonomy is to present all the factors that define or differentiate a network-related attack. The secondary goal is to build an updated taxonomy that can be understood and accepted by researchers in the field of network attacks. The classes are presented alphabetically, except for the *Attack Scenario* class, which is presented at the end.

4.2.1 Actor Class

The *Actor* Class describes the entity that is performing the attack by coding malware, executing malicious scripts or abusing the system. The *Actor* class is presented in Figure 4.2. This class has four main classes and seven sub-classes. This class was primarily

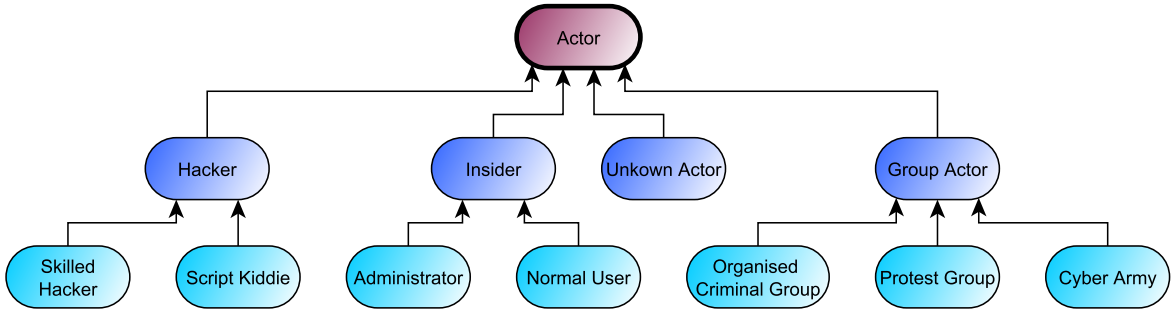


Figure 4.2: The *Actor* Class

derived from the work of Simmonds *et al.* (2004) and Rounds and Pendgraft (2009). The taxonomy is presented in Section 3.3.5 and Rounds and Pendgraft's taxonomy in Section 3.3.7. From Simmonds *et al.*, more detail was added to *Group Actor* to include *Organised Criminal Group*, *Protest Group* and *Cyber Army*.

The sub-class *Organised Criminal Group* refers to organisations that launch network and computer attacks for financial and other gain. For example, in Russia criminal organisations have recruited hackers to launch attacks on their behalf (Savona and Mignone, 2004). The *Organised Criminal Group* sub-class is not placed in the *Aggressor* class because the *Aggressor* class refers to criminal groups that perform their own attacks and not criminals who hire hackers. Choo (2008) stated that organised crime groups use the Internet for criminal activity.

Protest groups refers to groups that attack networks based on an ethical agenda. This also includes groups whose goals are driven by specific issues, and groups that use hacking to effect change or spread propaganda. Taylor (2001) referred to this practice as *Hacktivism*. The hacking group *Anonymous* is an example of a protest group that launched network attacks not as a criminal group, but rather as a protest group (Schwartz, 2012). The *Cyber Army* sub-class refers to military personnel who perform computer-based attacks as part of their normal duties. The concept of cyber-warriors and cyber-war had already became mainstream in the mid-1990s with *Time* magazine (Figure 4.3) referring to cyber-war and cyber-soldiers (Washington, 1995).

Figure 4.3: *Time* Magazine August 21, 1995 Cover Page

The *Insider* sub-class refers to a person who is a member of a target organisation or is in some trusted relationship with the target. Magklaras and Furnell (2001) defined three main insider groups: System masters, Advanced users and Application users. The

Advanced and Application users are classified as *Normal* users and System masters as *Administrators*. The distinction between Advanced and Application users was considered too vague for this taxonomy.

For this research it was decided to group *Hacker*, *Cracker* and *Malevolent user* in the researcher's *Hacker* sub-class, where Rounds and Pendgraft (2009) used a flat structure. Rounds and Pendgraft hacker agents are listed in Section 3.3.7. The hacker agents of Rounds and Pendgraft (2009) were used to verify the possible classes, although some of their classes were used by the *Aggressor* class. The *Hacker* sub-class was subdivided into Script Kiddie and Skilled Hacker. *Script Kiddie* refers to hackers that use freely available tools without any in-depth knowledge of their inner workings (Murry, 2004). (Spitzner, 2000, p. 1) defined a Script Kiddie as follows:

> "The script kiddie is someone looking for the easy kill. They are not out for specific information or targeting a specific company. Their goal is to gain root the easiest way possible. They do this by focusing on a small number of exploits, and then searching the entire Internet for that exploit. Sooner or later they find someone vulnerable."

4.2.2 Actor Location Class

This class refers to the country or state from where an attack is launched, and derives from the "location of attack" class developed by Undercoffer, Pinkston, Joshi, and Finin (2004). These researchers categorised the location of an attack as Remote, Local or Remote/Local. The only class not listed by Undercoffer *et al.* is one where the location is not known. The *Actor Location* sub-classes are shown in Figure 4.4.

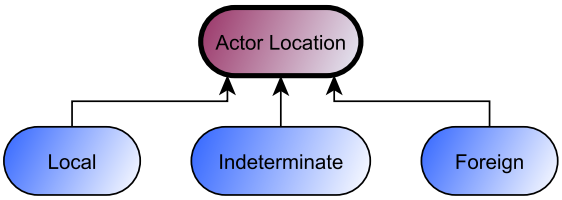


Figure 4.4: The *Actor Location* Class

The actor location can thus be outside the target's national borders, i.e. *Foreign*. *Foreign* refers to an *Actor* that is external to its own national borders. Lewis (2002) suggested that foreign militaries, criminals or terrorists can initiate cyber-attacks and thus constitute a cyber-threat. The second sub-class refers to an actor within the target's national borders.

Sometimes an actor location cannot be determined or spans different countries. In such cases, the *Indeterminate* sub-class is used. Although the location of an attacking computer can be determined, it does not necessarily correspond with the actor's physical location as the attack can be executed via the Internet (Xin, Dickerson, and Dickerson, 2003).

4.2.3 Aggressor Class

This class refers to the perpetrator of an attack, and differs from the *Actor* class in that it describes an association with an Actor, rather than a type of Actor. The sub-classes of the *Aggressor* class are shown in Figure 4.5.

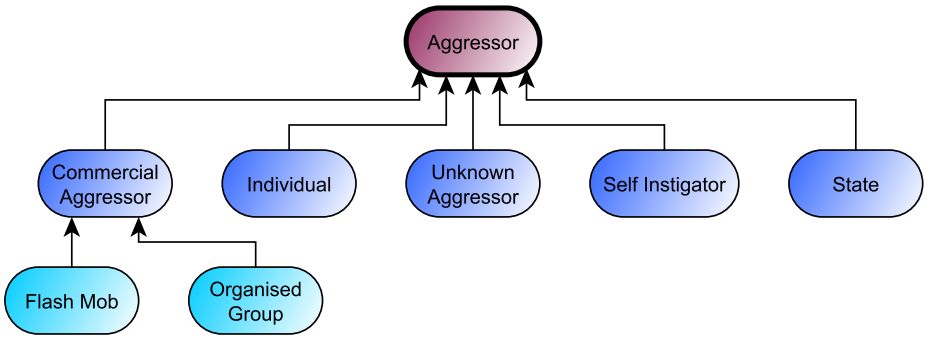


Figure 4.5: The *Aggressor* Class

State refers to a nation or state that sanctions an attack. Gandhi *et al.* defined a similar sub-class *Nation States*. Some researchers suggested that France, Russia, Japan, China, Germany, Israel and South Korea are actively engaged in economic espionage by means of the Internet and computer network attacks (Joyal, 1996; Kshetri, 2005; Brenner and Crescenzi, 2006; Burstein, 2009). *Commercial Aggressor* refers to a corporate entity, for example the *News of the World* British tabloid that authorised other entities to hack celebrities' cellphones (Myler and Wapping, 2011). *Commercial Aggressor* has the sub-classes *Organised Group Aggressor* and *Flash Mob*. *Organised Group Aggressor* refers to a perpetrator with commercial associations, for example People for the Ethical Treatment of Animals (PETA)[1]. *Flash Mob* refers to attackers that are not officially organised, and participants do not necessarily know each other. The SCO computer network was attacked in December 2003. Although no evidence exists, it is suspected that the attack was instigated following a lawsuit against IBM concerning IBM's use of Linux, and that open-source activists were the attackers (Argyraki and Cheriton, 2005). When the *Aggressor*

[1] http://www.peta.org/

and *Actor* are the same entity, the *Self Instigator* sub-class is used. This sub-class refers to lone hackers who are not motivated by an external party. The *Unknown Aggressor* sub-class is used when the identity of the perpetrator is unknown. For example, up to 2010, the instigators and perpetrators of the Conficker Worm attack have not been identified (Conficker Working Group, 2011).

4.2.4 Asset Class

This class refers to the device class that is under attack. This class distinguishes between different assets that can be attacked. Examples of assets are information stored as data, the system that uses computers, or the network infrastructure itself. The *Asset* class is shown in Figure 4.6.

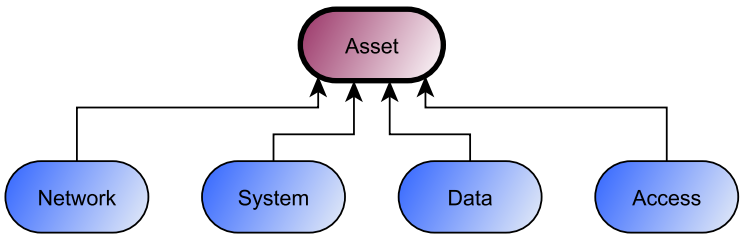


Figure 4.6: The *Asset* Class

Typically, the goal of Denial-of-Service attacks is to deny users access to their own computer resources or as described by Specht and Lee (2004):

> "DoS attack is an attack with the purpose of preventing legitimate users from using a specified network resource".

When an attack targets communication infrastructure, the affected asset is classified as *Network*. When attacks affect information, *Data* is the asset under attack. This can include changing data, stealing data and removing data. The *Access* sub-class refers to unauthorised access to the situation where computers or computer networks have been obtained.

Some attacks make use of computer networks. Two significant examples are the Logic Bomb (Section 2.4.2) and the Stuxnet Worm (Section 2.4.29), which affected physical assets outside the computer network. With the Logic Bomb, a pipeline was affected, and with Stuxnet centrifuges were affected. These attacks are classified so as to affect the *System* asset.

4.2.5 Attack Goal Class

This class refers to the purpose of the attack, and is subdivided as shown in Figure 4.7: *Steal Data*, *Change Data*, *Disrupt*, *Gain Control*, *Gain Resources* and *Spread*.

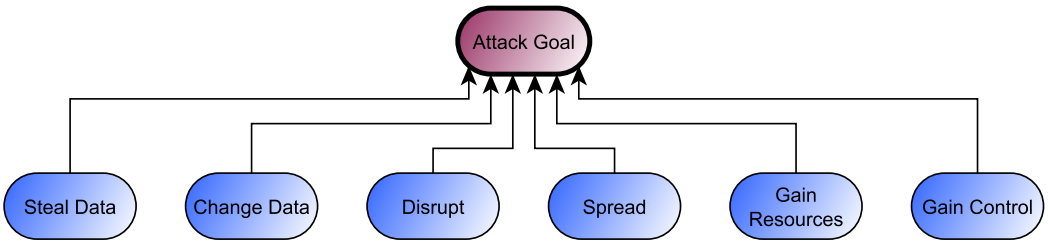


Figure 4.7: The *Attack Goal* Class

Steal Data, *Change Data*, *Disrupt* correspond with the traditional CIA+ information security principles and Simmonds *et al.* (2004) outcome classes. Examples of the *Stealing Data* sub-class are Titan Rain (Section 2.4.14) and Operation Aurora 2009 (Section 2.4.28). A hacker changed the grades of more than 60 current and former students from Santa Clara University (Zetter, 2012b). This attack falls under the *Change Data* sub-class as well as the attack on HP Gary (Section 2.4.32). Hansman and Hunt (2003) referred to this as *Corruption of Information*.

The *Gain Control* goal represents instances where the network under attack is used only as a staging post for attacks on a different network. When information is disclosed without permission (Hansman and Hunt, 2003), the *Steal Data* sub-class is used. The Conficker Worm's goal is to build a platform from where other attacks can be launched (Conficker Working Group, 2011), and thus can be classified as *Spread*. The *Gain Resources* goal represents the goal of obtaining computer resources such as processing power, bandwidth and disk memory. The *Gain Control* goal refers to gaining administrator rights to a system.

4.2.6 Attack Mechanism Class

This class represents the attack methodology, and is linked to vulnerability maps developed by Simmonds *et al.* (2004). Attack mechanisms have also been listed by Hansman and Hunt (2003). The sub-classes are presented in Figure 4.8, where *Attack Mechanism*

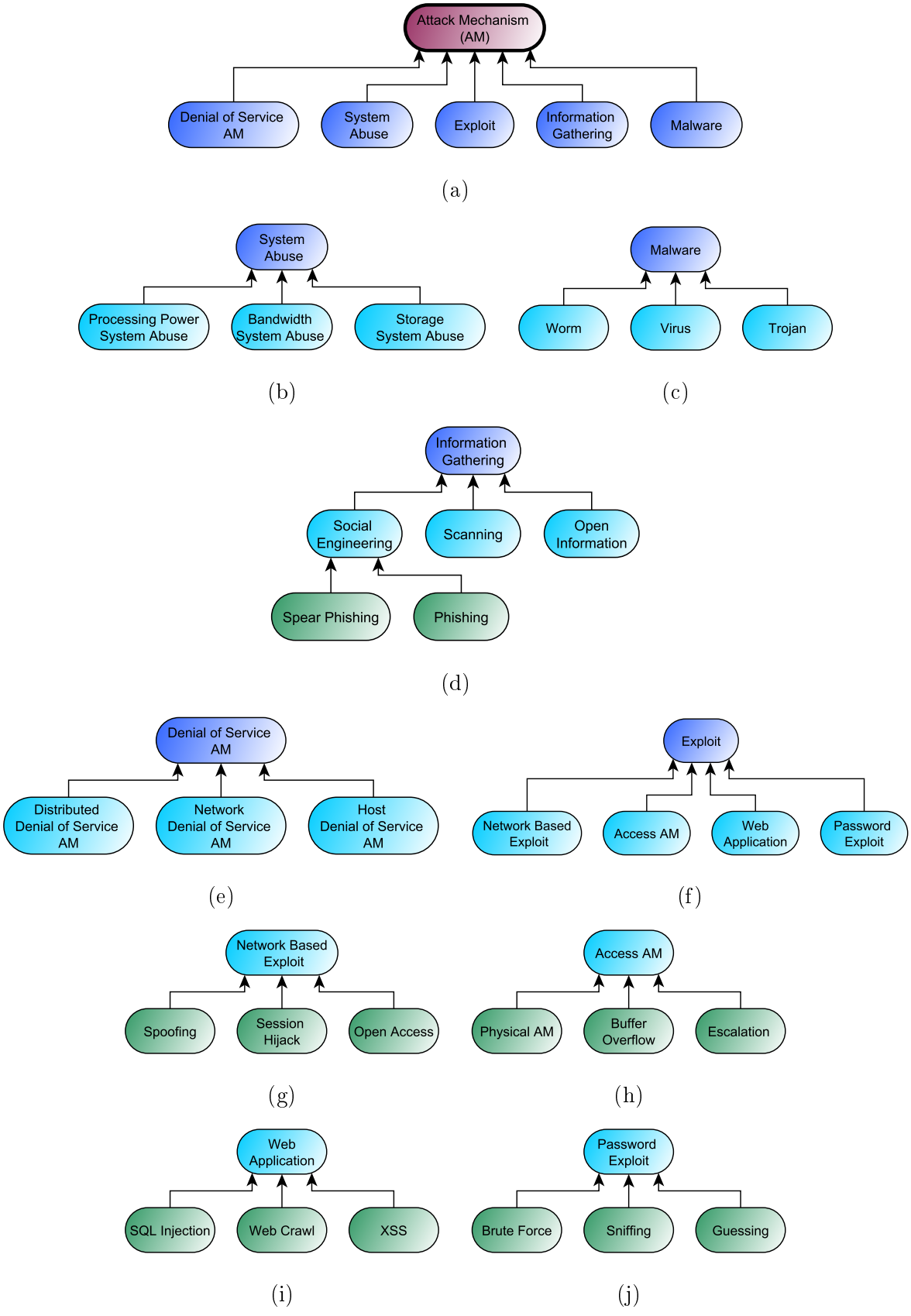


Figure 4.8: The Attack Mechanism (AM) Class

is shortened to *AM*. The first sub-classes are presented in Figure 4.8a: *Denial of Service AM*. *System Abuse*, *Exploit*, *Information Gathering*, and *Malware*.

The *System Abuse* sub-class refers to the abuse of computer resources (as shown in Figure 4.8b). Three main resources have been identified: *Processing Power System Abuse*, *Bandwidth System Abuse* and *Storage System Abuse*. Although Krebs (2012) identified over 30 abuses of computer systems, these can be summarised as abuse of processing power, disk space or bandwidth when the physical components of a system are considered.

Malware attacks can take the form of *Trojans*, *Viruses* or *Worms*, shown in Figure 4.8c. Vasudevan and Yerraballi (2006) defined malware as: "a generic term that encompasses viruses, trojans, spywares and other intrusive code." There is currently no clear scientific distinction between the different types of malware-based attack methodologies. Yampolskiy and Govindaraju (2007) listed the most acknowledged definitions:

- Virus: a self-replicating malicious programme which requires a careless user or external software to replicate itself
- Worm: a self-replicating programme that automatically spreads through vulnerabilities
- Trojan Horse: a malware programme posing as a legitimate programme

The *Information Gathering* sub-class refers to the mechanisms used to acquire information about a possible target, for example public information on websites. *Information Gathering* sub-classes are shown in Figure 4.8d. Search engines such as Google can be used to find *Open Information*, even though the target does not realise that the information is available. This technique is also referred to as Google Hacking (Long, 2007). Information required for an attack can also be totally in the open. Corporate websites and phone directories sometimes publicly list email addresses and telephone numbers that can be used for attacks. One of the most popular methods to scan computers is to identity which ports are open. Lee, Roedel, and Silenok (2003) found that port scans represented a measurable portion of Internet traffic. Some information that an attacker uses is publicly available. *Scanning* (also know as vulnerability scanning) refers to the process of probing computers with the goal of identifying which services are running, which operating system they are using or which applications are actively running with the goal of finding vulnerabilities. *Social Engineering* attack mechanisms refer to processes used to gain access to a target by misleading people into granting access or giving away confidential information by means of social interactions (Goodchild, 2010). Social engineering is defined by Rouse (2006, p. 1) as follows:

"Social engineering is a term that describes a non-technical kind of intrusion that relies heavily on human interaction and often involves tricking other people to break normal security procedures."

Spear Phishing refers to targeted social engineering-type email attacks (Jagatic, Johnson, Jakobsson, and Menczer, 2007). Boerio and McCracken (2012) differentiated between spear phishing and a general *Phishing* in that regular phishing targets a large amount of people, with a small chance of success per person, and spear phishing targets a small number of people with a large chance of success. Brody, Mulig, and Kimball (2007) state that spear phishing is harder to detect since spear messages appear to be legitimately sent from people known to have an established relationship with the target. Kevin Mitnick often used social engineering as his favourite method of attack, and even wrote two books on the subject (Mitnick *et al.*, 2002; Mitnick and Simon, 2005).

Denial of Service Attack Mechanisms refers to attacks that use valid communication methodologies in malicious methods or in great numbers to deny the correct users access (as shown in Figure 4.8e). These attacks can be on one of the following vectors (Lau, Rubin, Smith, and Trajkovic, 2000; Mirkovic and Reiher, 2004):

- *Host Denial of Service Attack Mechanisms*, an attack on a single hosts
- *Network Denial of Service Attack Mechanisms*, an attack that consumes all available bandwidth
- *Distributed Denial of Service Attack Mechanisms*, an attack that uses or targets multiple systems in the attacks

The *Exploit* attack mechanisms (Figure 4.8f) are used to present the methodologies that are used to attack vulnerabilities directly. The main sub-classes for this sub-class are:

- *Network-based Exploit* (Figure 4.8g);
- *Access Exploit* (Figure 4.8h);
- *Web Application Exploit* (Figure 4.8i); and,
- *Password Exploit* (Figure 4.8j).

Network-based Exploit refers to attacks that exploit vulnerabilities in the network. These attacks use and abuse data that flows on the network, using the following mechanisms:

- *Spoofing.* With a spoofing attack misleading context is created with the goal of hoaxing a victim into trusting malicious intended information (Felten, Balfanz, Dean, and Wallach, 1997). This technique is typically used to change IP source packets (Bremler-Barr and Levy, 2005).

- *Open Access.* This refers to attacks that defeated systems because no security existed, or that assumed that security by obscurity would be enough (Mercuri and Neumann, 2003; Hoepman and Jacobs, 2007).
- *Session Hijack.* With this attack mechanism the attacker gains access to a user's session by obtaining his session identification information (Kolšek, 2002).

Access Exploit attack mechanisms abuse access mechanisms to provide access on a false premise. This can be achieved by:

- *Buffer Overflow.* The Buffer Overflow method can be used when systems do not perform checks on the input limits. This oversight can be abused by overriding security measures via specialised crafted input. The Morris Worm (Section 2.4.5) used a buffer overflow vulnerability to propagate (Cowan, Pu, Maier, Walpole, Bakke, Beattie, Grier, Wagle, Zhang, and Hinton, 1998).
- *Physical Access.* This refers to manual methods of gaining access, for example physically removing the hard drive or breaking the access door to enter a secure server room.
- *Escalation.* This is also known as privilege escalation, where administration rights are obtained by attacking flaws in the operating system or application design (Govindavajhala and Appel, 2006).

Web Application Exploit attack mechanisms refer to methods used specifically on websites and web servers. Web servers are vulnerable to uniquely related attacks because of their interactive nature, and they are designed to interact and exchange data which enables them to use the following attack mechanisms:

- *SQL injection.* This uses common escape characters to execute user-defined database queries, thus bypassing authentications and other security measures (Mookhey and Burghate, 2004).
- *Cross-site scripting (XSS).* This is a methodology that enables attackers to inject client-side script into web pages. These pages can then be viewed by unsuspecting users (Mookhey and Burghate, 2004).
- *Web Crawl.* This is a process used by search engines or crawling software to collect information from web pages (Castillo, 2005).

Password Exploit attack mechanisms refer to methods used to obtain or bypass password protection, such as:

- *Brute force.* These are attacks that attempt to bypass security by trying each possible key sequentially (Cowan, Wagle, Pu, Beattie, and Walpole, 2000; Steffan and

Schumacher, 2002). This type of attack mechanism systematically uses all possible character key combinations to uncover username and password combinations. This process does not require skill and is thus referred to as brute force.

- *Sniffing.* This entails eavesdropping on communications to capture secrets such as passwords (Oppliger, 1998).
- *Guessing.* This entails overcoming password protection by guessing popular passwords (Gong, 1995).

4.2.7 Automation Level Class

This class describes the degree to which network attacks are automated. The sub-classes for the Automation Level class are shown in Figure 4.9.

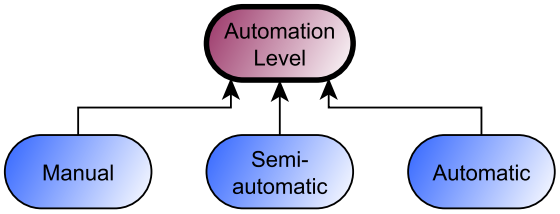


Figure 4.9: The Automation Level Class

The sub-classes were derived from Mirkovic and Reiher (2004)'s taxonomy. *Manual* refers to an attacker selecting the attack target and methodology by hand. *Automatic* refers to a system requiring minimum input from the attacker, even with regards to target selection. Mudge (2011) lists methods and tools that can be used to automate attacks. Many attacks are *Semi-automatic*, where a mixture of automation and manual methods are used, and some user interaction is required, but tools are used to execute attacks.

4.2.8 Effect Class

This class refers to the impact of an attack. Mirkovic and Reiher (2004) discussed the impact of different attacks. The sub-classes for the Effects class are shown in Figure 4.10.

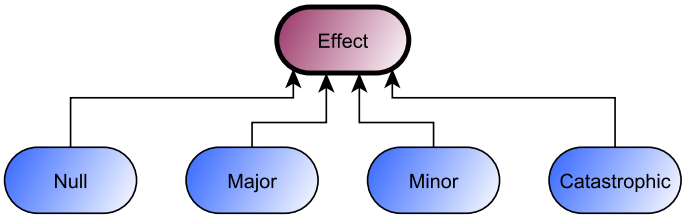


Figure 4.10: The Effect Class

Null refers to no effect on the target, *Minor* to recoverable damage and *Major* to non-recoverable damage. *Catastrophic* refers to damage of such a nature that the target ceases to operate as an entity, for example declaration of bankruptcy.

4.2.9 Motivation Class

This class refers to an attacker's motivation for an attack. Rounds and Pendgraft (2009) listed three kinds of possible motivations: political, socio-cultural and economical. These classes were also used by Gandhi *et al.* (2011). The sub-classes are shown in Figure 4.11:

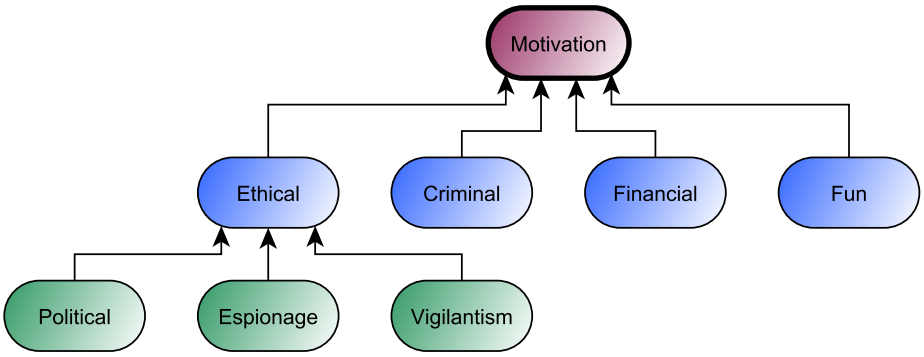


Figure 4.11: The *Motivation* Class

Financial refers to hacking for financial or other gain, such as stealing money or manipulating the stock market. The attack on SA Postbank (Section 2.4.33) was motivated by possible financial gain. The *Financial* motivation can be criminal in nature, but *Criminal* motivation refers in this case to criminal organisations that use network hacking to supplement their operations. For example, attacking law enforcement agencies' networks to disrupt investigations is motivated by criminal intent. The *Financial* and *Criminal* mutually are not multilaterally exclusive. *Fun* refers to hackers looking for a challenging

hack with no other evil intentions. Many of the most famous worms and viruses were not developed with any harm intended, but got out of the creators' control. *Ethical* motivation refers to motivation that has an ethical aspect. This ethical aspect can be national interest by spies, political reasoning or vigilantes.

4.2.10 Phase Class

The *Phase* class was derived from the temporal attack model in Section 4.4. Figure 4.12 presents the phase classes. Within the taxonomy, their temporal relationships are not presented, but only their definitions.

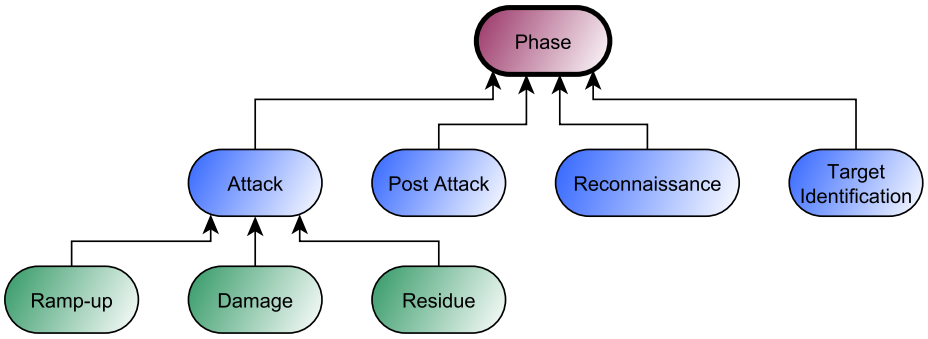


Figure 4.12: The *Phase* Class

Target Identification refers to the action of an attacker choosing a target. The target identification phase ends when a specific device or entity (an individual, company or state institution) has been identified.

Reconnaissance refers to the action of an attacker probing a target for a weakness. Probing consists of scanning, Google queries and other network-related activities. No computer or network system is changed or adversely affected. The goal is to identify avenues of attack whilst leaving network operations unaffected.

Attack refers to the action of compromising the target according to the CIA principles (confidentiality, integrity or availability), and has three sub-phases. The *Ramp-Up* sub-phase refers to the action of an attacker preparing to achieve a goal. The target may be affected, but not necessarily adversely. An example of the Ramp-Up phase is installation of a sniffer by an attacker on an unsuspecting user to harvest clear text passwords for later use so as to steal data. The *Damage* sub-phase refers to the action of the attacker

inflicting damage on the target. Damage may take the form of breached confidentiality, compromised integrity or disrupted service availability. Damage can be inflicted via data, physical means (computer-controlling hardware) or to the target's reputation. The *Residue* sub-phase refers to damage or artefacts of the attack that occur after the attack goal has been achieved, and occurs because the attacker loses control of some systems. For example, after the launch of a DDoS attack, zombie computers may still connect to the target for some days following the attack.

Post-Attack refers to actions undertaken by an attacker after the attack has occurred, and takes the form of inspections to verify if backdoors are still available, or scans to verify if security holes have been patched. The goal is not to inflict damage, but to verify the target's status.

4.2.11 Sabotage Class

This class refers to the type of loss the target experiences during and after the attack. The *Sabotage* class is inspired by Gandhi *et al.* (2011) *Attack Consequences* class. *Sabotage* differs from the *Attack Goal*, by referring to the damage of the target, not the attackers goal (even though it usually is the same). The sub-classes are shown in Figure 4.13.

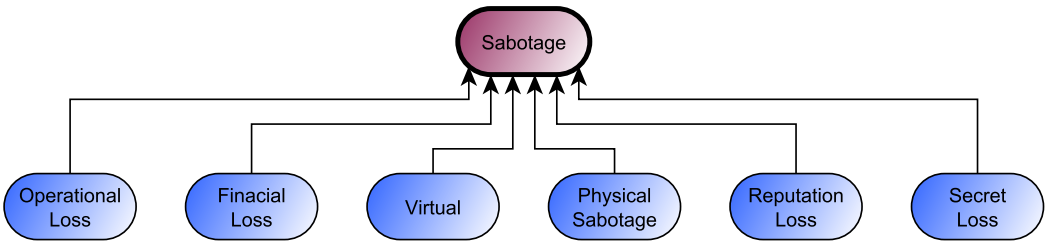
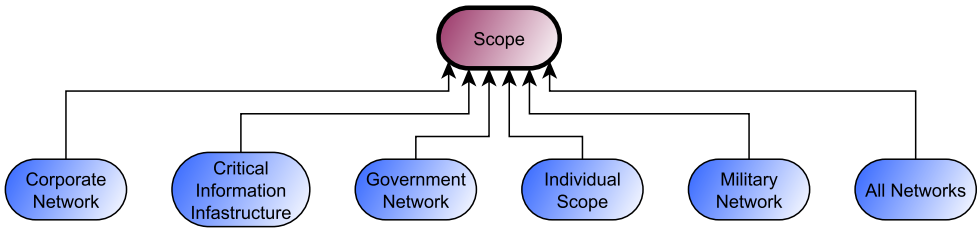
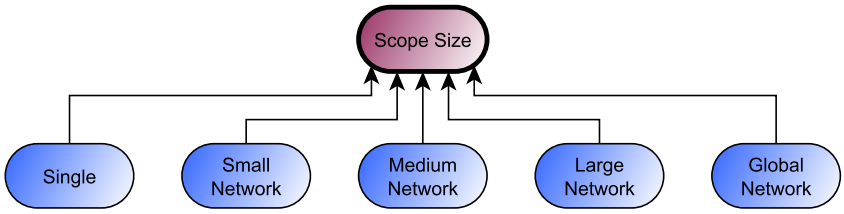


Figure 4.13: The *Sabotage* Class

Physical Sabotage refers to physical damage of a device; such as that caused to the Siberian pipeline by the original Logic Bomb. *Financial Loss* sabotage refers to monetary loss. *Virtual* sabotage occurs when computer resources are lost (such as processing, bandwidth or memory). *Reputational Loss* is not a measurable tangible loss, but may result in other related problems for a company later on. *Operational Loss* occurs when the system cannot perform its required function. *Secret Loss* refers to when secrets have been compromised.



(a) The *Scope* Class



(b) The *Scope Size* Class

Figure 4.14: *Scope* and *Scope Size* Classes

4.2.12 Scope and Scope Size Classes

This class refers to the type of entity that is targeted. The *Scope* class differs from the *Target* class in that it views the entity holistically, rather than looking at specific devices, and it is based on the Gandhi *et al.* (2011) *Attack Victim* class. The sub-classes for the *Scope* class are shown in Figure 4.14a. The *Corporate Network* sub-class refers to networks controlled by private companies. The *Government Network* sub-class refers to networks controlled by the government. *Individual Scope* is used when the target is a single person or computer. The *Military Network* sub-class refers to networks under the control of a military institution. *Critical Information Infrastructure* includes networks that are essential to a nation's economy by providing vital services.

The *Scope Size* class refers to the size of entity that is targeted. The sub-classes for this class are shown in Figure 4.14b. If the attacks affect a large portion of the Internet or multiple countries, the scope size is referred to as *Global Network*. *Large Network* represents large cooperates or significant government networks such as state departments. There are no hard definitions that separate small, medium and large networks and thus the separation is a subjective judgement. *Single* size is used to present attacks on a single person or single computer.

4.2.13 Target Class

This class refers to physical devices that are targeted by an attack, whereas the Hansman and Hunt (2003) taxonomy included the software operating systems in its taxonomy. Hansman and Hunt's level of detail is considered too fine for the researcher's taxonomy. The sub-classes are shown in Figure 4.15.

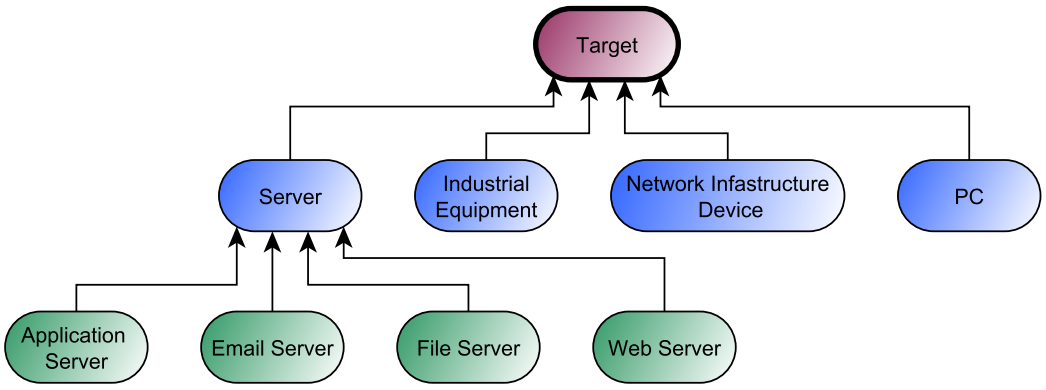


Figure 4.15: The *Target* Class

The *Personal Computer* sub-class refers to desktop PCs, laptops, tablets and similar devices with a single user. Internet-based attacks on smartphones also falls within this class. *Network Infrastructure Device* refers to devices such as routers and switches that only enable data flow, but can still be attacked. *Industrial Equipment* refers to computerised automation equipment used in industrial plants. This equipment is also referred to as Supervisory Control and Data Acquisition (SCADA) systems. A PC has a lot of useful information or other malicious uses that can be used by an attacker. Krebs (2009, 2012) compiled a list of all the methods in which the information on a compromised PC can be used for monetary gain (Section 3.3.9). *Server* subclass refers to computing devices that provide services to clients. These clients either run on the same computer or other computers via a network. *Server* has the following subclasses: *Web Server*, *File Server*, *Email Server* and *Application Server*. Any server that falls outside of an *Email Server*, a *File Server* or a *Web Server* can be classified as an *Application Server*.

4.2.14 Vulnerability Class

This class refers to the weaknesses exploited by the attacker. Simmonds *et al.* (2004) constructed a Vulnerability map, as shown in Section 3.3.5. Undercoffer *et al.* (2004)

listed the following vulnerabilities: input validation errors, buffer overflows, boundary condition errors and other malformed input. The vulnerability map of Simmonds *et al.* differentiates according to short and long terms. The researcher's *Vulnerability* does not have a temporal aspect, and no distinction should therefore be made between short- and long-term vulnerabilities. The first level of sub-classes for this class is the same as developed by Hansman and Hunt (2003). The sub-classes for the *Vulnerability* class are shown in Figure 4.16.

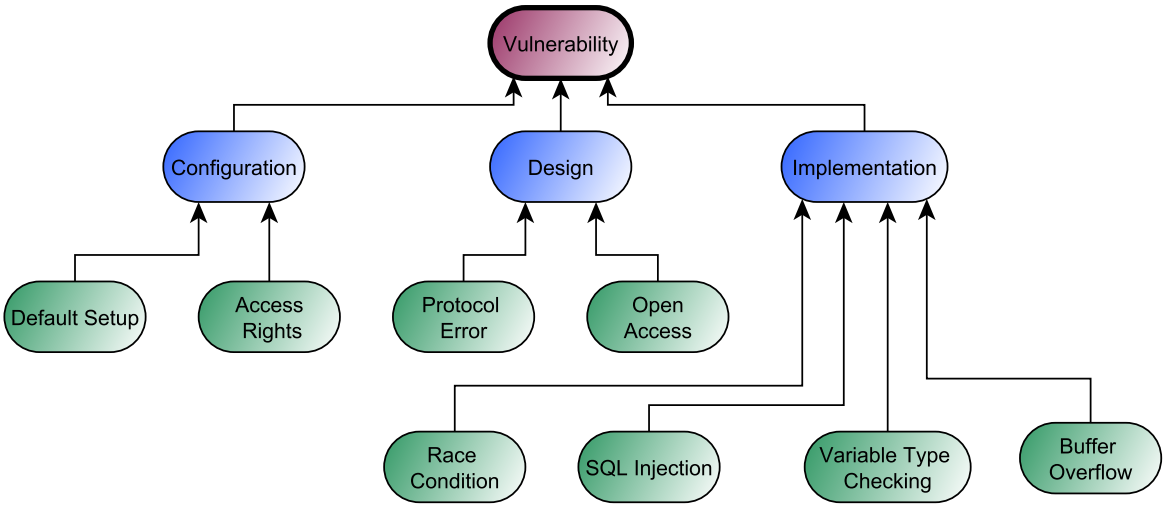


Figure 4.16: The *Vulnerability* Class

Configuration vulnerabilities describe instances where vulnerabilities are exposed by incorrect configuration of a device or software. Two types of incorrect configuration are listed, namely *Access Rights* and *Default Setup*. *Access Rights* refers to an instance where incorrect access rights have been allocated to normal users. For example, Citigroup was hacked by thieves that penetrated the bank's defences by first logging on to the site reserved for its credit card customers (Schartz and Dash, 2011). *Default set-up* refers to the use of default usernames and passwords to overcome the security of a system. This vulnerability is often caused by inexperienced or lazy users. Lancor and Workman (2007) described how Google can be used to hack systems by using default usernames and passwords.

Design vulnerabilities render a system insecure because of design errors. Design errors can be either in the protocol or in the access control. The "Ping-of-death" is an example of a protocol vulnerability (Karig and Lee, 2001).

Implementation vulnerabilities refer to vulnerabilities introduced by faulty coding or system construction, and have the following sub-classes:

- *Buffer Overflow* refers to the ability of injecting an attack code (Cowan *et al.*, 2000).
- *Race Condition* refers to the creation of a vulnerability in a programme due to a short opening for an attacker also known as a timed window vulnerability (Bishop and Dilger, 1996).
- An *SQL Injection* vulnerability enables an attacker to take advantage of flawed coding of websites. An attacker usually injects SQL commands into a website that then allows him access to a database (Razvan, 2009).
- *Variable Type Checking* is also known as format string vulnerability, where an attacker can abuse input variable strings to inject code or gain access (Shankar, Talwar, Foster, and Wagner, 2001).

4.3 Attack Scenarios

In Section 2.5, ten network attack scenarios were identified. The attacks listed in Chapter 2 (Figure 4.17) are classified according to these listed attack scenarios.

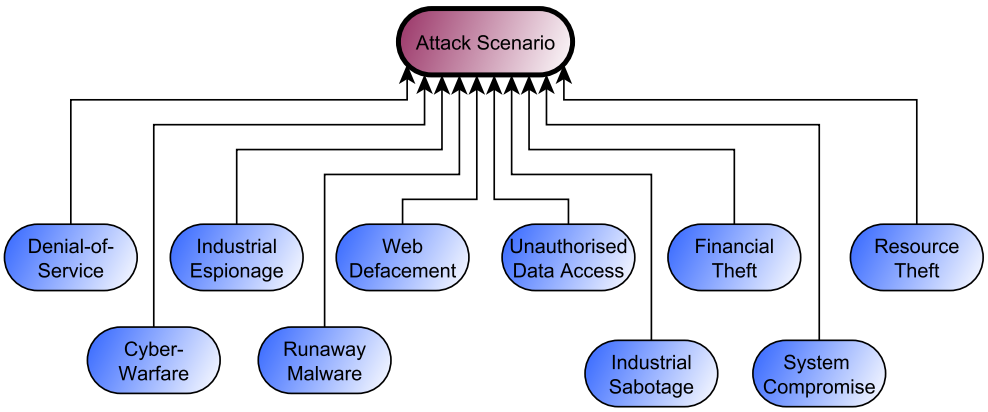


Figure 4.17: The *Attack Scenario* Class

McDowell (2009, p. 1) from the United States Computer Emergency Readiness Team (US CERT) defined a *Denial-of-Service* attack as: "attempts to prevent legitimate users from accessing information or services". Almeida and Mutina (2011) noted that they were able to archive 1 419 203 examples of web defacements. *Denial-of-Service* attacks can take many forms, such as Bandwidth Depletion Attacks (Amplification Attacks or

Flood Attacks) or Resource Depletion Attacks (Protocol Exploit Attacks or Malformed Packet attacks) (Specht and Lee, 2004). When multiple computers are used to attack a system and attempt to overwhelm it by sheer number of connections, it is referred to as a DDoS. For example, the DDoS attack on U.S. financial institutions in December 2012 peeked at 60 Gbps (Constantin, 2012). The Estonia Hack Attack, SCO Denial-of-Service and Mafiaboy exploits can be considered *Denial-of-Service* type attacks.

Industrial Espionage refers to network attacks of which the goal is to acquire commercial secrets such as source code, industrial processes, customer lists, etc. Titan Rain, Operation Aurora, and Operation Shady Rat can be considered Industrial Espionage-type attacks.

Web Defacement refers to vandalism of a public website. The motive for defacing websites could be for entertainment, looking for a challenge, patriotism, a political agenda or revenge (Balakrishnan and Sarma, 2004). Almeida and Mutina (2011) noted with concern that in 2010 they were able to archive 1 419 203 website defacements. The defacement of Apache.org and HB Gary Hack can be considered *Web Defacement* attacks.

Unauthorised Data Access refers to curious or malicious individuals, spies or anyone snooping around for secrets. Most of Kevin Mitnick's attacks focused on looking for secrets. The PlayStation hack can also be defined as snooping for secrets.

Financial Theft refers to stealing money via computers. Computer networks in banks and other financial institutions can be compromised and money can be transferred electronically to criminals. Individuals can also be targeted and attacked through web banking interfaces. The attacks on Citi Bank and SA Postbank are examples of network attacks with financial theft as the main goal.

Resource Theft refers to the act of controlling computers so that the collection of computer resources can be sold or used at a later date. For example, millions of "zombie" computers are for sale on the Internet black market (Markoff, 2007). MyDoom, Conficker and Code Red are examples of malware that attempted to amass computer resources. *Industrial Sabotage* refers to damaging industrial capability of commercial or state entities. The Logic Bomb and Stuxnet are examples of industrial sabotage.

The South Ossetia Incident and the Estonia Incident had elements of *Cyber-Warfare* even though no war was declared. Beidleman (2009, p. 10,12,13) defined cyber-war, cyber-attacks and cyber-space as follows:

> Cyber-war: when cyber-attacks reach the threshold of hostilities commonly recognized as war by the international community and defined by international law.

Cyber-attacks: a subset of cyber-operations employing the hostile use of computers and information technology infrastructure to achieve effects or objectives in or through cyber-space.
Cyber-space: global domain within the information environment consisting of the interdependent network of information technology infrastructures, including the Internet, telecommunications networks, computer systems, and embedded processors and controllers.

Some attacks result from software that escapes control and spreads further than initially intended. This malware is sometimes only written to prove a point or exploit a new vulnerability, but then goes out of control. Many of the famous viruses and worms of the 1980s and 1990s are examples of out-of-control software. The I-LOVE-YOU Worm, Morris Worm, Melissa, SQL Slammer, Brain Virus, PC-Write Trojan, Chameleon Virus, Michelangelo Virus, Laroux, Cabir Worm, Sony XPS and Ikee attacks listed in Chapter 2 fall within the *Runaway Malware* category.

When physical industrial equipment is targeted, rather than the information, network or services, the attack is referred to as *Industrial Sabotage*. Stuxnet, which targeted the centrifuges of Iran's nuclear programme, and the Logic Bomb attacks, which targeted the gas pipeline, fall within this category.

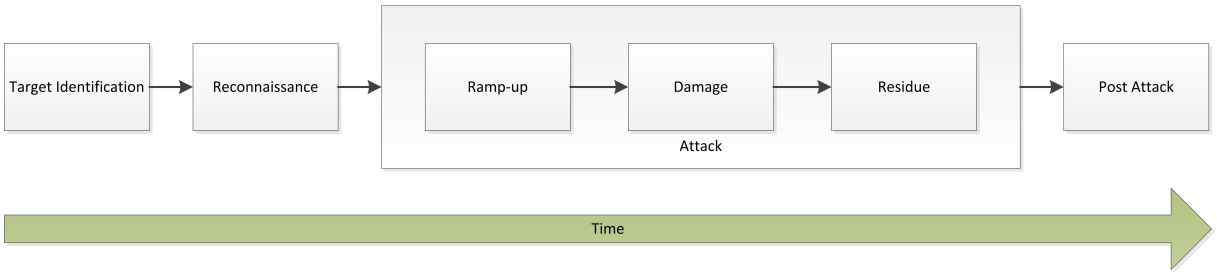
The *System Compromise* scenario refers to unauthorised personnel or hackers gaining user rights out of their scope. A system compromise attack refers to hackers breaking into a single or multiple computers without authorisation and taking control of such a system. Thus the computer system is considered compromised. Flame malware is an example of an attack with the goal of compromising systems.

4.4 Model of Network Attacks

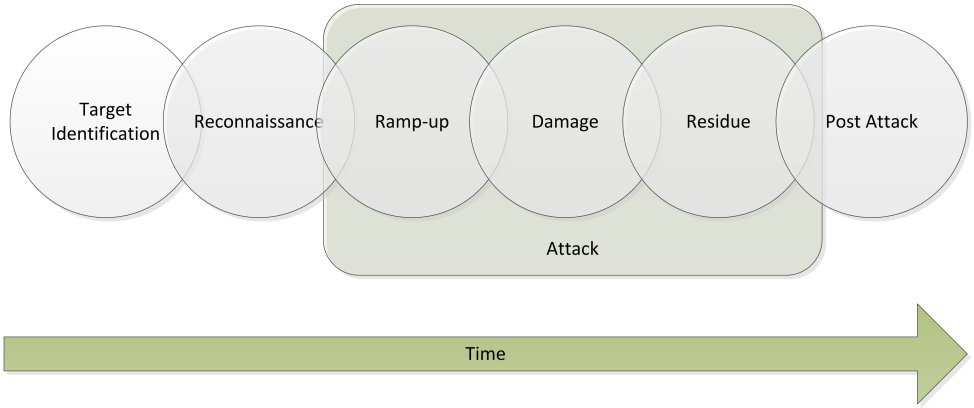
In this section, a temporal network attack model is described. This model is based on the models presented in Section 3.2 and by van Heerden *et al.* (2012c). This attack model consists of the multiple phases (also referred to as stages) of an attack. Four basic phases were identified: *Target Identification*, *Reconnaissance*, *Attack* and *Post-attack Reconnaissance*. The *Attack* phase was divided into three sub-phases: *Ramp-up*, *Damage* and *Residue*. Each phase is unique, but their temporal instances can overlap.

4.4.1 Network Attack Phase

The phases either follow each other discretely as shown in Figure 4.18a, or overlap temporally to some extent, as shown in Figure 4.18b.



(a) Discrete Attack Model



(b) Non-Discrete Attack Model

Figure 4.18: Network Attack Model

The *Target Identification* phase represents actions undertaken by an attacker in choosing a target. Identification of these actions falls outside the scope of threat identification, but forms part of the overall threat model.

The *Reconnaissance* phase represents actions undertaken by an attacker to identify potential weak spots. These actions are the earliest indications that a network will fall under attack, before any real damage has occurred. Popular reconnaissance actions include network mapping and scanning with tools such as Nmap[2], Nessus[3] (Feng, 2003; Fyodor, 1998; Deraison, 2005) and Zmap[4] (Durumeric, Wustrow, and Halderman, 2013).

Google and other search engines can also be used to identify potential weak spots. The

[2]http://www.nmap.org
[3]http://www.tenable.com/
[4]https://zmap.io/

Attack phase represents modification of the target system by the attacker. The system can be modified in terms of the following aspects:

- confidentiality;
- integrity; and
- availability.

These aspects are also known as the CIA triad (as shown in Section 3.3.5) . Confidentiality is the term used to prevent the disclosure of information to unauthorised individuals or systems. Integrity means that data cannot be modified undetectably. Availability refers to the availability of information when required by the system to serve its purpose. In computing, e-business and information security, it is necessary to ensure that data, transactions, communications and documents are genuine. It is also important that authentication validates the identities of both parties involved.

The *Attack* phase is subdivided into sub-phases. The first sub-phase is the *Ramp-up* phase. This sub-phase refers to the preparatory actions performed by an attacker before his/her final goal can be attained. The targeted computer network is modified in this phase, but only in preparation for some other goal. This phase typically includes the installation of backdoors and other malware.

The *Damage* sub-phase refers to actions undertaken by an attacker during the achievement of his/her final goal. In this sub-phase, the network is compromised in terms of the Information Security CIA principles. For example, when an attacker launches a DDoS attack on a network, the Damage sub-phase is entered as soon as the attack is launched. The process of installing DDoS attack software falls under the Ramp-up state.

The *Residue* sub-phase refers to unintended communications and actions by malware after an attack has been completed. For example, computers that have incorrect time settings may attack their target at a later date and/or time than when the original co-ordinated attack was planned. This is also noticed in DDoS attacks.

The *Post-Attack Reconnaissance* phase refers to scouting and other similar reconnaissance actions performed by an attacker after completion of the Attack phase. The attacker's goal in this phase is to verify the effects of his/her attack and to assess whether the same methodology can be used again in the future.

4.4.2 Structured Analysis and Design Technique Analysis

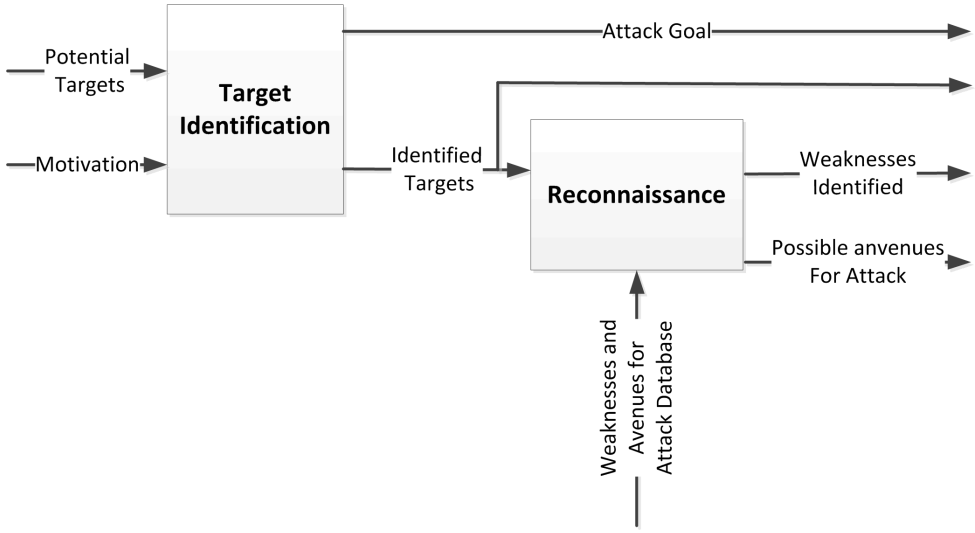
Structured Analysis and Design Technique (SADT) is used to specify the actions of systems in terms of functional processes (Marca and McGowan, 1987). SADT uses a graphical notation to symbolise the system as a group of boxes connected by arrows. The boxes represent processes and the arrows interfaces between the processes. Information is passed from each process concurrently to the next through the interface arrows. Arrows that enter into the left side of a box represent input data. Arrows that enter into the top of a box represent control inputs. Arrows that enter into the bottom of a box represent the mechanisms or resources required. Arrows are only allowed to exit a box from the right and represent data output. Only the input and output data is changed in a box; the control and resource inputs are not affected by these processes.

Each box (process) can be separated into sub-boxes (sub-processes). With the SADT notation, this was symbolised by enclosing a group of boxes within a larger box. The larger box inputs and outputs must be the same as the inputs and outputs of the smaller boxes that are not interconnected. These inputs and outputs are referred to as free inputs and free outputs. Marca and McGowan stated that SADT should topically have three to seven boxes (processes). Grant *et al.* (2012) used the SADT process to evaluate different Offensive Cyber Operations models. The attack model is presented as a SADT model in Figure 4.19.

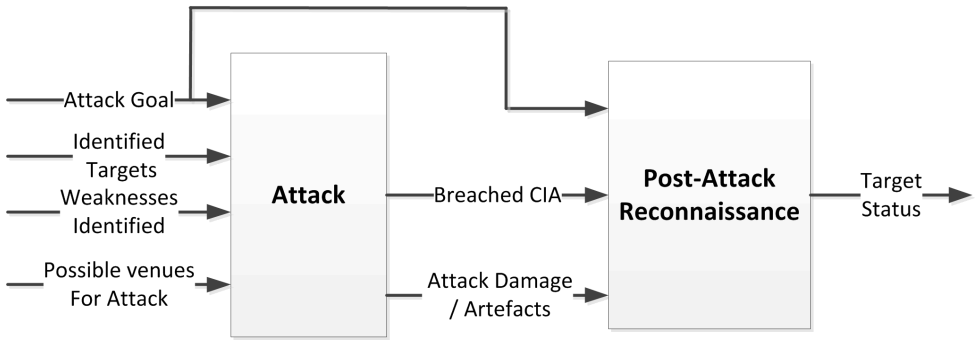
The model presented in Figure 4.19 consists of four main boxes: *Target Identification*, *Reconnaissance*, *Attack* and *Post-attack Reconnaissance*. These boxes are the same as the phases presented in Section 4.4. The *Target Identification* box has two inputs: potential targets that can be attacked, and the motivation for the attack. The output of this block is the attack goal and a list of targets.

The list of targets is the input for the *Reconnaissance* box. The mechanism which this box uses is a predefined list of weaknesses and attack avenues. The output of the *Reconnaissance* box identifies weaknesses and possible avenues for attack. Weakness refer to the state of the target and avenues for attack methodologies that the attacker can use.

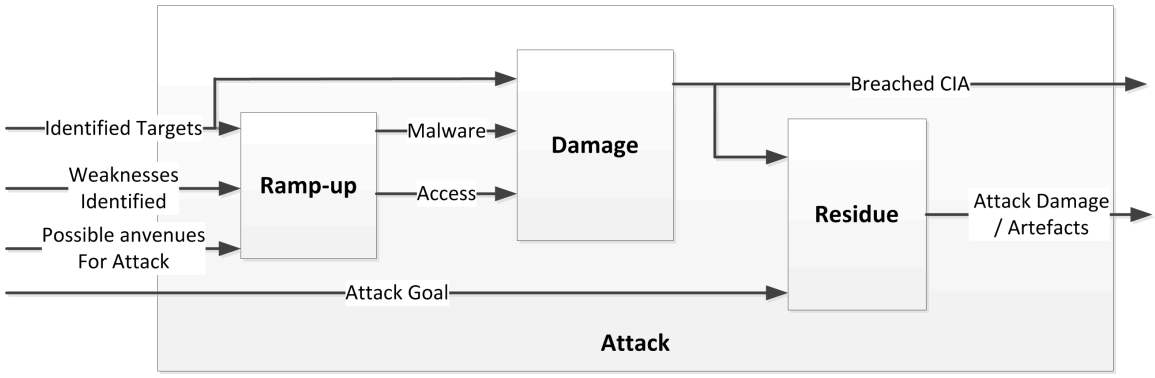
The *Attack* box uses four inputs: the attack goal and identified target outputs from the *Target Identification* box, weaknesses identified, and possible avenues for attack outputs from the *Reconnaissance* box. The output of the *Attack* box is the breached CIA and artefacts that the attack caused. These artefacts are unintentional behaviour of the targeted network because of the attack, for example when an attacker can lock out legitimate users by repeatable failed logins.



(a)



(b)



(c)

Figure 4.19: SADT Composition Attack Model

The *Post-attack Reconnaissance* box uses the attack goal, breached CIA and attack artefacts as inputs. This box has a single output: the status of the target. The status is an indication of how effective the attack was.

The *Attack* box is expanded to three more boxes: *Ramp-up*, *Damage* and *Residue*. The *Residue* box uses the identified targets, weaknesses and possible avenues for attacks as inputs. Its output is malware and access. This malware now directly attacks its target, or access to the required system or data has been achieved. The malware, access and identified targets are then used as the input for the *Damage* box. This box represents the place where the attacker goal is achieved. The *Damage* box output represents the breached CIA principles, which along with the attack goal represents the inputs for the *Residue* box. The *Residue* box represents effects that are not planned, but caused by the attack, and its output is attack-related artefacts.

4.5 Summary

In this chapter, a taxonomy was presented whereby the elements of a computer-based attack can be described. The taxonomy consists of the following classes: *Actor*, *Actor Location*, *Aggressor*, *Asset*, *Attack Goal*, *Attack Mechanism*, *Attack Scenario*, *Automation Level*, *Effect*, *Motivation*, *Phase*, *Sabotage*, *Scope*, *Scope Size*, *Target* and *Vulnerability*. The taxonomy presents both the view of the attacker and defender. Each class in this taxonomy has sub-classes that can be used to classify types of computer attacks in more detail.

The *Phase* class is used to build a temporal model of network attacks. This model consists of the following phases: *Target Identification Reconnaissance Attack* and *Post-attack Reconnaissance*. The *Attack* phase consists of *Ramp-up*, *Damage* and *Residue*. These phases are used in Chapter 8 when a network attack prediction system is developed.

The relationships between the classes are explored in the next chapter with the development of an ontology. An ontology builds on the knowledge of a taxonomy by defining and constraining the relationships between the classes. The ontology is formally described and the ontology editor Protégé and automated reasoner HermiT are introduced.

CHAPTER

FIVE

NETWORK ATTACK ONTOLOGY

"Deep in the human unconscious is a pervasive need for a logical universe that makes sense. But the real universe is always one step beyond logic."

Frank Herbert, *Dune*

5.1 Introduction

In this chapter, an ontology is presented whereby the classes that were introduced in the taxonomy described in Chapter 4 are used in the ontology. Each attack scenario in Section 4.3 is described in detail with an illustrative example. The ontology is formally described in Section 5.4. The same is done for the *Denial-of-Service* scenario.

Noy and McGuinness (2001) defined an ontology as a formal, explicit description of concepts of discourse classes, with the properties of each class describing various attributes of the concepts (slots) and their restrictions. Classes are the focal point of ontologies, and can be divided into sub-classes which represent more detailed concepts. The ontology presented in this chapter used the main classes in Chapter 4 as the focal point, and the sub-classes are used to present a more detailed picture. Noy and McGuinness (2001) further stated that developing an ontology requires:

- definition of classes;
- arrangement of classes in a taxonomy;
- description of the attributes of slots;
- definition of allowed values for attributes; and
- definition of events according to classes and slots.

Grüninger and Fox (1995) stated that an ontology is used to formally describe objects, their properties and relations, and stated:

"The development of ontologies is motivated by scenarios that arise in the applications. In particular, such scenarios may be presented by industrial partners as problems which they encounter in their enterprises. The motivating scenario often has the form of story problems or examples which are not adequately addressed by existing ontologies."

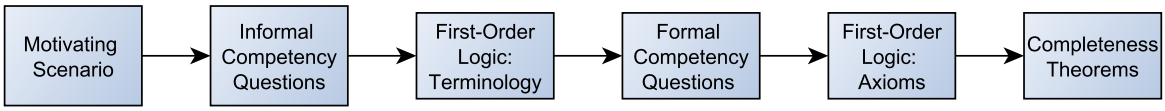


Figure 5.1: Design and Evaluation Procedure for Ontology by Grüninger and Fox (1995)

The network attack ontology presented in this chapter is presented as stories regarding the different types of scenarios. Grüninger and Fox (1995) noted that one of the first steps in verifying an ontology is providing scenarios from which the motivation of the ontology can be understood (Figure 5.1). In this chapter, the ontology is formally described and then implemented within Protégé. Grüninger and Fox noted that the first steps in specifying an ontology entails the identification of the objects in the domain, which was done as seen in the taxonomy in Chapter 4.

The relationships between objects are defined in this chapter. In Section 5.2, the Protégé ontology editor is presented. A story that describes network attacks utilising the taxonomy is presented in Section 5.3. A formal description of the ontology is presented in Section 5.4 to verify the ontology implementation. Network attack individuals are inferred from their respective scenarios in Section 5.5.

5.2 Protégé

Protégé is an editor that represents ontologies and their relationships. This editor was developed by Stanford University and is freely available[1]. Protégé is the most popular ontology editor and according to Cardoso (2007) has a 68% market share. The Protégé editor facilitates the building of ontologies via definitions of their relationships, properties and individuals (Akinbode and Longe, 2011; Malviya, Mishra, and Sahu, 2011). The Protégé editor enables a simple method for:

- definition of classes;
- relationships between classes;
- properties of these relationships; and
- class hierarchies.

To aid with the visualisation process, Protégé has additional plug-in packages such as OWLViz[2] and OntoGraf[3]. In Figure 5.2, the class editor of Protégé is shown, in Figure 5.3, an example of the visualisation tool OWLViz is shown and in Figure 5.4, an example of the visualisation tool OntoGraf is shown.

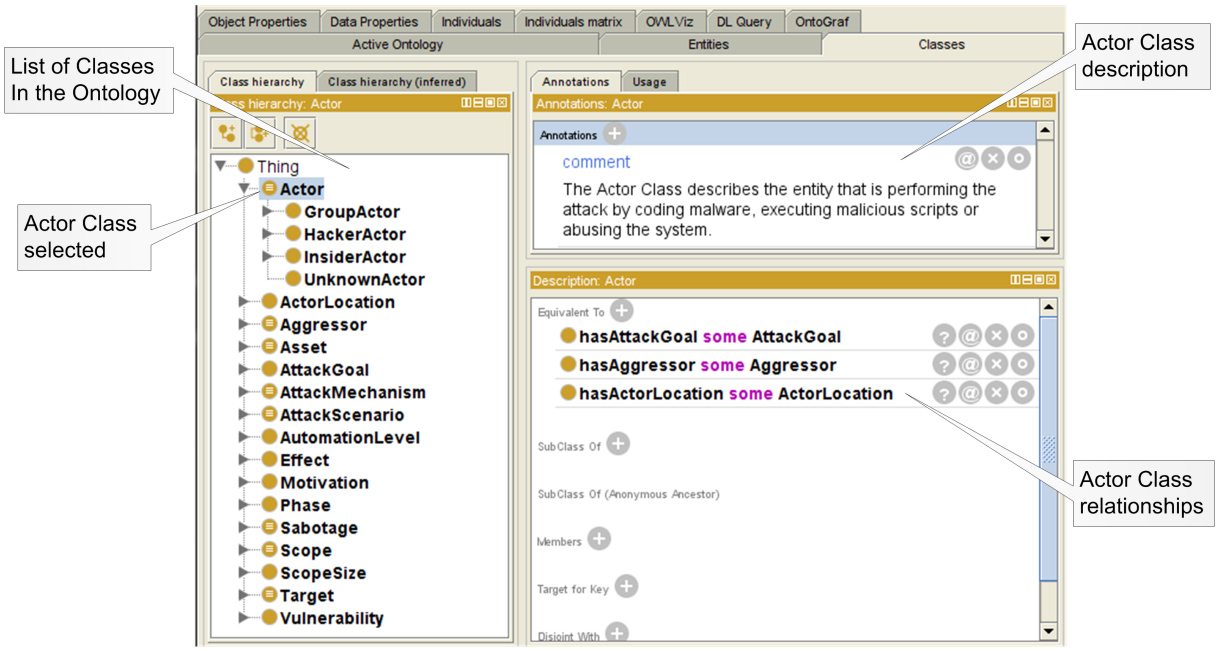


Figure 5.2: Example of Protégé Editor

[1] http://protege.stanford.edu/
[2] http://protegewiki.stanford.edu/wiki/OWLViz/
[3] http://protegewiki.stanford.edu/wiki/OntoGraf/

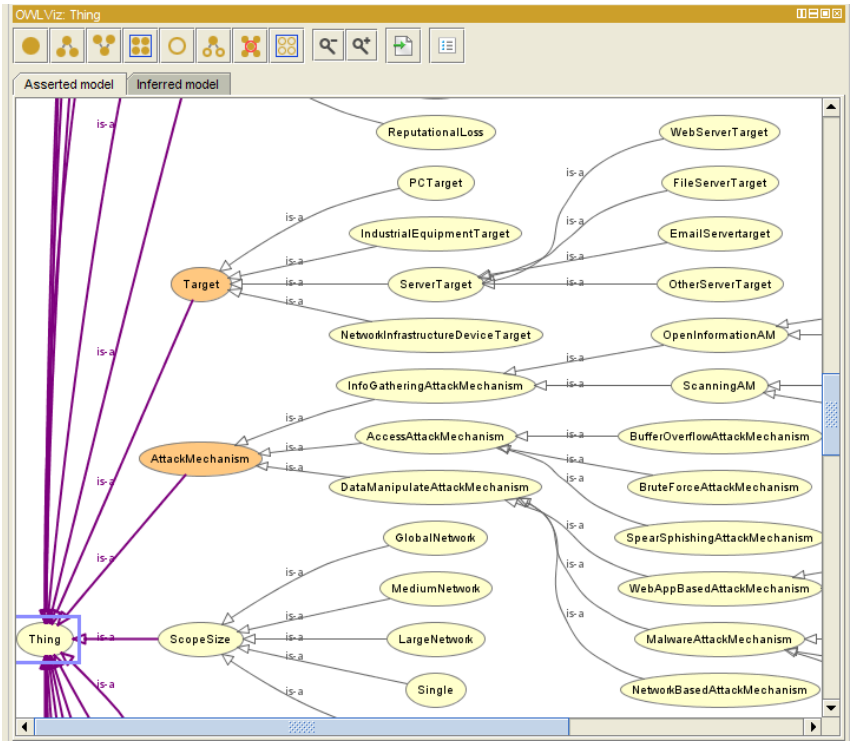


Figure 5.3: Example of OWLViz Visualisation Tool

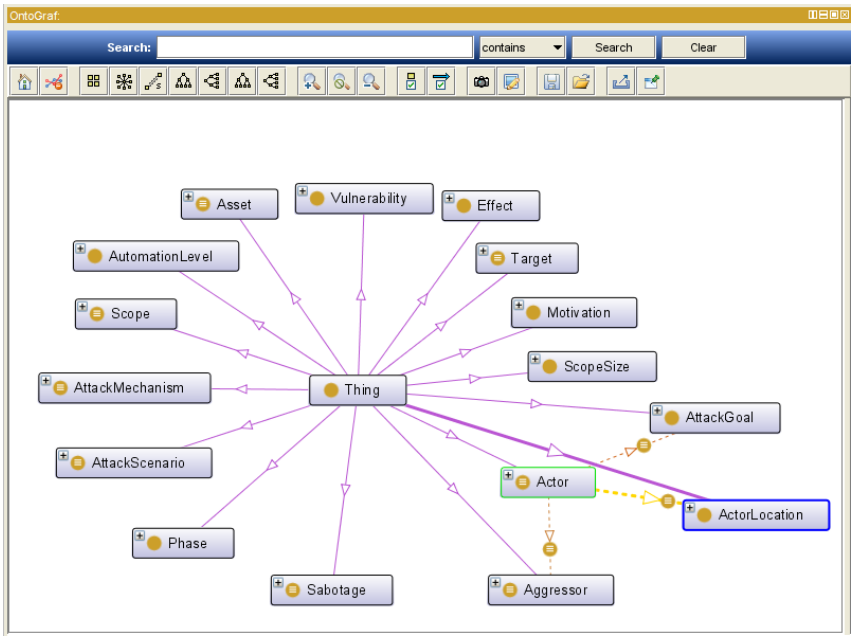


Figure 5.4: Example of OntoGraf Visualisation Tool

Protégé allows the user to store the ontology in Web Ontology Language (OWL) format. "The Semantic Web" was proposed by Berners-Lee, Hendler, and Lassila (2001) to link

data on the web so that the information can be re-used across multiple applications and lead to a common understanding. The OWL format is the accepted standard for implementing the "The Semantic Web" information in the form of ontologies (Antoniou and Van Harmelen, 2009; Akinbode and Longe, 2011). The official OWL overview is available at World Wide Web Consortium (W3C) standards website[4].

5.2.1 Automated Reasoner

One of Protégé's main features is its automated reasoners. Gong, Guo, Yu, Zhang, and Xue (2008) stated that the role of an automatic reasoner is to establish that the ontology is correct and consistent. The automatic reasoner has the ability to find contradictions regarding the ontology and thus ensure that the ontology is consistent. Bock, Haase, Ji, and Volz (2008) described some of the functionalities of an automated reasoner:

- ability to satisfy (verify if a class can have instances)
- subsumption (verify if a class is subsumed by another class)
- consistency (verify the consistency of individuals within the ontology)
- instance checking (verify the assertions within the ontology)
- retrieval problem (given a property and individual, determine all other individuals related to them)
- conjunctive queries (class selection, projection or renaming queries)

The researcher uses the HermiT[5] OWL automatic reasoner to answer the following questions (Shearer, Motik, and Horrocks, 2008):

- In which scenario does a specific network attack fall (subsumption relationships of individuals)?
- Can some attack scenario classifications be merged when scenarios in near real-time are considered (subsumption relationships of classes)?
- Which attack scenarios do network-based sensors indicate?

5.3 Network Attack Ontology

The Network Attack ontology maps all the classes of the Network Attack taxonomy into a single concept, with the *Attack Scenario* as the base class. This mapping is presented

[4]http://www.w3.org/TR/owl-features/
[5]http://www.hermit-reasoner.com/

in Figure 5.5 and as the following story (the classes are in bold and within brackets): *An **[Actor]** based at **[ActorLocation]** with the goal of **[AttackGoal]** is sponsored by **[Aggressor]** with a **[Motivation]** motivation. The attack effected **[ScopeSize]** **[Scope]** scope. A **[Target]** was attacked via **[Vulnerability]**. This attack effected **[Asset]** and resulted in **[Sabotage]** to **[Effect]** during each attack phase. During each phase the **[AttackMechanism]** was used, and was automated to **[AutomationLevel]** level.*

The story listed above is not intended to be grammatically perfect, but is used as a flexible skeleton. This skeleton is used to illustrate the scenarios and their individuals. Thus to maintain similar storylines, the grammatical presentation has to be flexible.

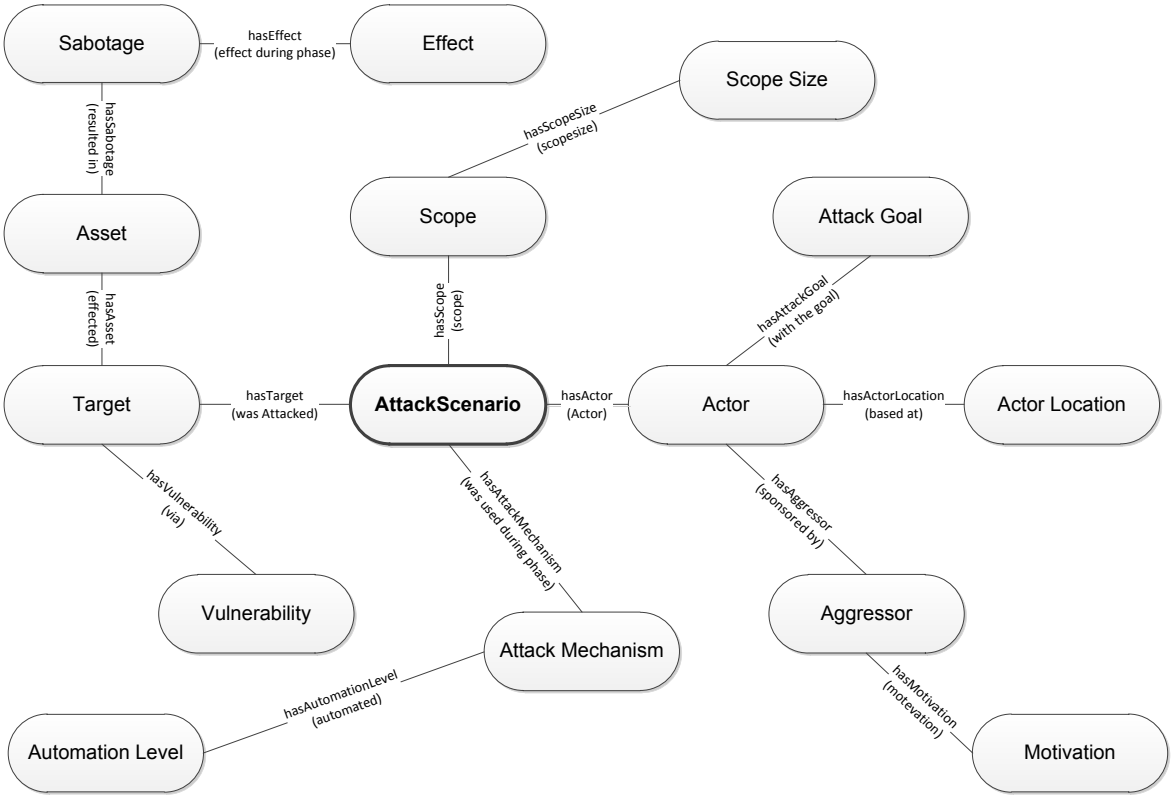


Figure 5.5: Network Attack Ontology

In Figure 5.5, the relationships between the formal classes are shown along with their use in the story. Each of the *Attack Scenarios* listed in Section 2.5 have unique constraints within the Network Attack ontology, and have a unique mapping.

5.3.1 Denial-of-Service Scenario

A story for each *Attack Scenario* can be constructed. Their stories differ where the classes are defined as sub-classes. The Denial-of-Service Attack Scenario story is as follows:
*An **Hacker** based at **[ActorLocation]** location with the goal of **Disrupt** sponsored by **[Aggressor]** with a **[Motivation]** motivation. The attack effected **[ScopeSize] [Scope]** scope. A **Network Infrastructure** was attacked via **[Vulnerability]**. This attack effected **Access** and resulted in **Operational Loss** to **Major** effect during the Damage attack phase and to **Null** effect during the Ramp-up attack phase. During the Ramp-up and Damage phase the **Denial-of-Service Mechanism** was used, and was automated to **Automatic** level.*

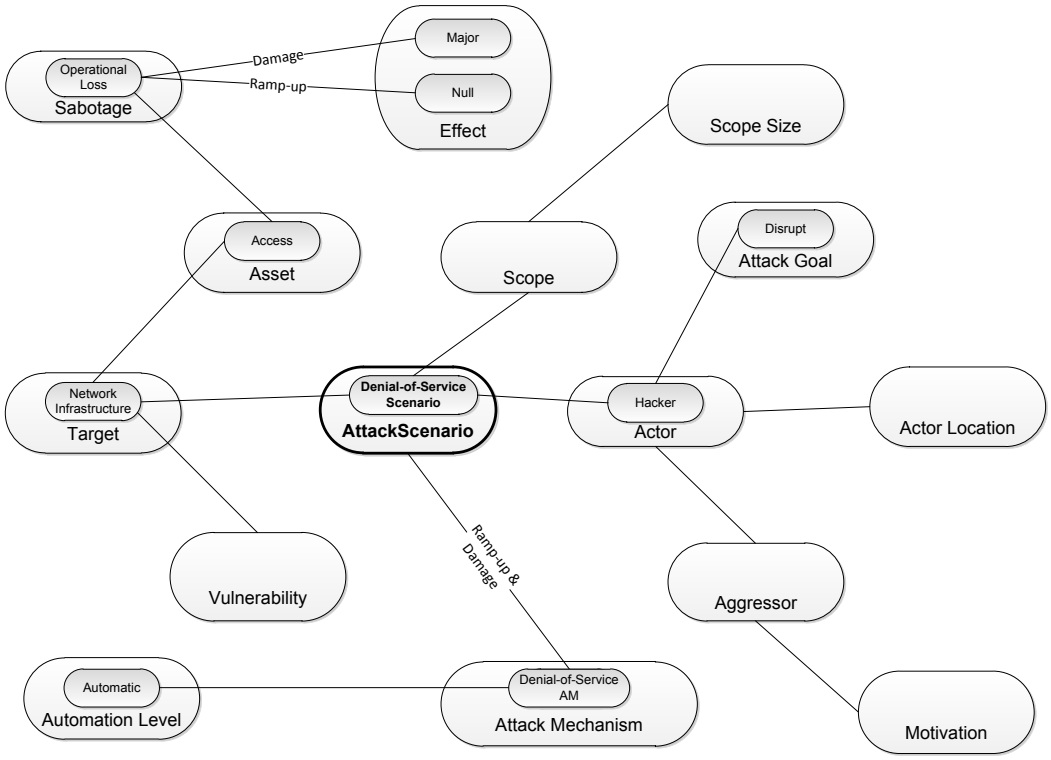


Figure 5.6: Denial-of-Service Attack Scenario

The *Denial-of-Service Attack Scenario* constrains are shown in Figure 5.6 and two individuals, the SCO, and Spamhaus DDoS attacks are shown in Figure 5.7 and Figure 5.8. In these figures, the sub-classes for each scenario are shown as sub-classes within the main classes and extra information that identifies individuals is shown in the rectangular blocks.

The attacks on the SCO network and SpamHaus (discussed in sections 2.4.18 and 2.4.35)

are examples of a Denial-of-Service attack scenario and their stories can be formatted in the same way:

SCO Attack:

Hackers *based at* ***Indeterminate*** *location with the goal of* ***Disrupting*** *sponsored by* ***Flash Mob*** *with a* ***Vigilantism*** *motivation. The attack effected* ***Medium Network Corporate (SCO)*** *scope. A* ***Network Infrastructure*** *was attacked via* ***Protocol Error*** *vulnerability. This attack effected* ***Access*** *and resulted in* ***Operational Loss*** *to* ***Major*** *during Damage phase and to* ***Null*** *effect during the Ramp-up attack phase. During the Ramp-up and Damage phase the* ***Denial-of-Service Mechanism*** *was used, and was automated to* ***Automatic*** *level.*

SpamHaus Attack:

Hackers *based at* ***Foreign (Netherlands)*** *location with the goal of* ***Disrupting*** *sponsored by* ***Commercial (CyberBunker)*** *with a* ***Vigilantism*** *motivation. The attack effected* ***Large Network Corporate (SpamHaus)*** *scope. A* ***Network Infrastructure*** *was attacked via* ***Protocol Error*** *vulnerability. This attack effected* ***Access*** *and resulted in* ***Operational Loss*** *to* ***Major*** *during Damage phase and to* ***Null*** *effect during the Ramp-up attack phase. During the Ramp-up and Damage phase the* ***Denial-of-Service Mechanism*** *was used, and was automated to* ***Automatic*** *level.*

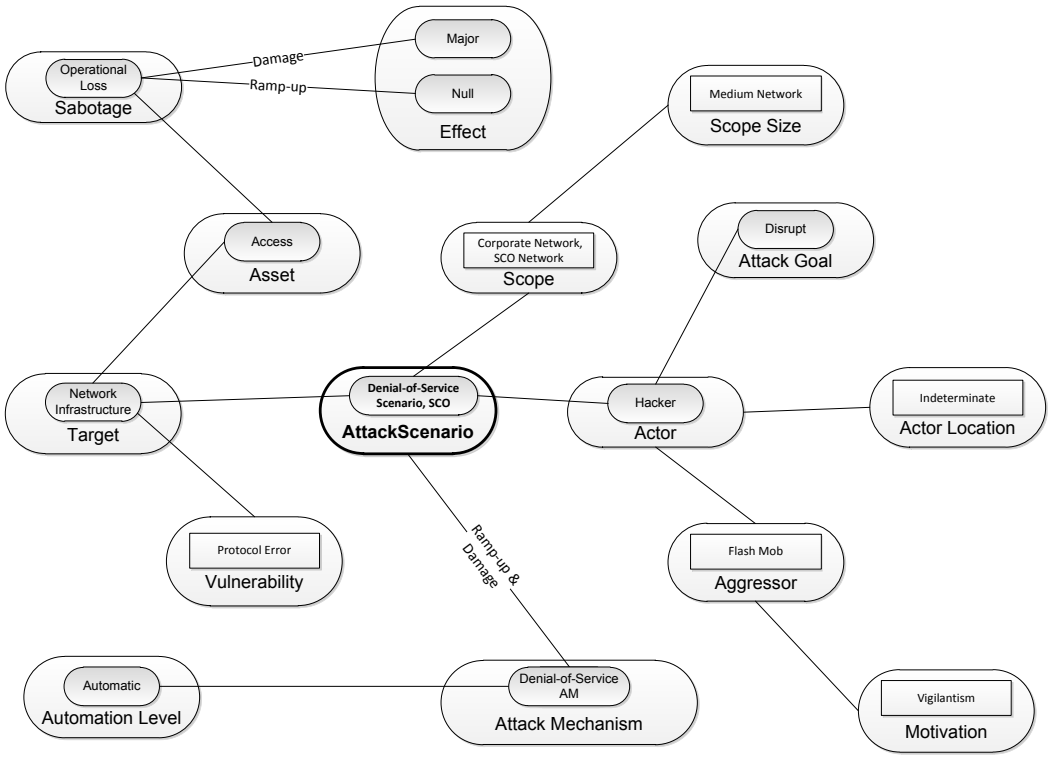


Figure 5.7: SCO Denial-of-Service Attack Scenario Example

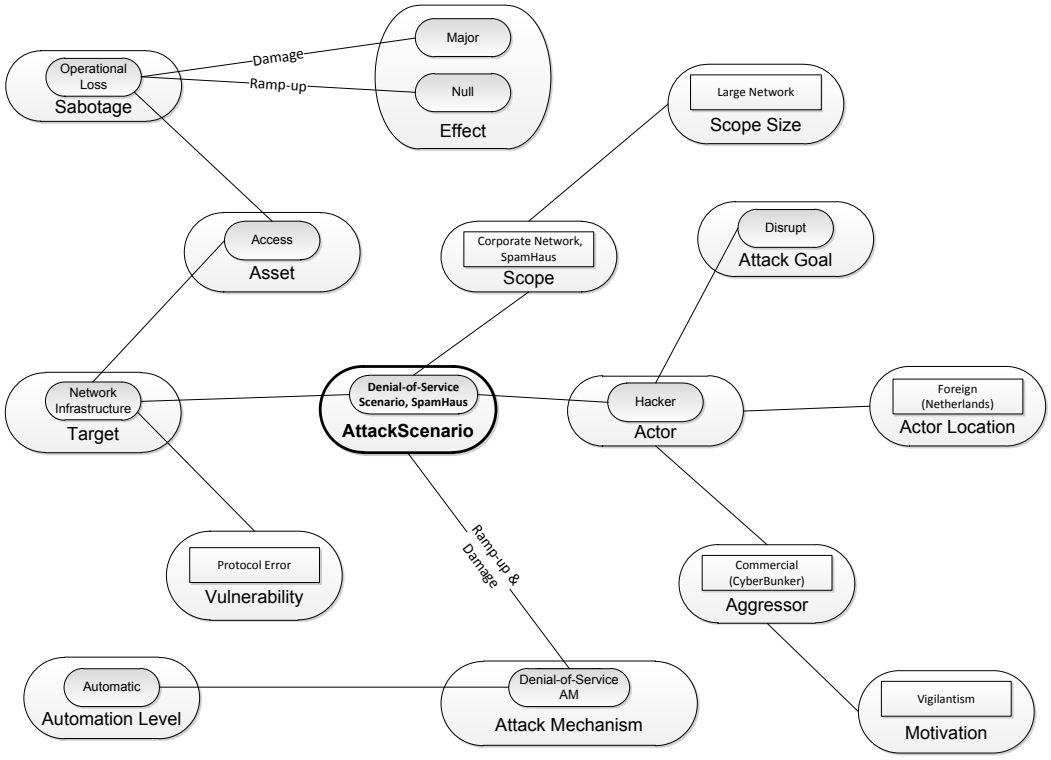


Figure 5.8: SpamHaus Denial-of-Service Attack Scenario Example

The rest of the scenarios, stories and examples are presented in Chapter 6.

5.4 Formal Description of Network Attack Ontology

The HermiT automated reasoner (Section 5.2.1) can infer from an ontology in which scenarios individuals fall and which scenarios can be merged. Thus to verify the ontology design, a formal description of ontology was developed.

An ontology can be defined as a 4-tuple according to Scharffe and de Bruijn (2005); Chaudhri, Farquhar, Fikes, Karp, and Rice (1998); Zhai, Chen, Yu, Liang, and Jiang (2009):

$O =< C, R, I, A >$

where

O is an ontology;

C is a set of concepts defined for the domain;

R is a set of binary semantic relations defined between concepts in C;

I is a set of instances where each instance can be one one or more classes linked by relations (Davies, Studer, and Warren, 2006) and
A is a set of axioms.

An axiom is a real fact or reasoning rule and a concept is considered to be a class in an ontology.

This definition assumes there is an implicit assumption of a set, D, which represents the domain of interest. It follows that:

$$C \subseteq D \tag{5.1}$$

$$R \subseteq D \times D \tag{5.2}$$

The network ontology is defined in Statement 5.3:

$$NA =< C_{NA}, R_{NA}, I_{NA}, A_{NA} > \tag{5.3}$$

where NA defines an ontology related to a network attack. The set of concepts (or base classes) C_{NA} is described in Section 5.4.1. Section 5.4.2 defines all the relations between the concepts, i.e. the set R_{NA}. An example of an individual is discussed in Section 2.4.18.

5.4.1 Network Attack Concepts

The subsets of the set C_{NA} are shown in Statement 5.4 and contain all the base classes of the taxonomy listed in Chapter 4.

$$\begin{aligned} &Actor, ActorLocation, Aggressor, \\ &Asset, AttackGoal, AttackMechanism, \\ &AttackScenario, AutomationLevel, \\ &Effect, Motivation, Sabotage, Scope, \\ &ScopeSize, Target, Vulnerability \subseteq C_{NA} \end{aligned} \tag{5.4}$$

The 15 subsets of C_{NA} are defined in the following statements: 5.5, 5.9, 5.10, 5.12, 5.13, 5.14, 5.24, 5.25, 5.26, 5.27, 5.29, 5.30, 5.31, 5.32 and 5.34. Some of these subsets are defined in more detail below. The class $Actor$ and its sub-classes as displayed in Figure

4.2 are presented in statements 5.5 to 5.8.

$$GroupActor, Hacker, \\ Insider, UnknownActor \subseteq Actor \tag{5.5}$$

$$OrganisedCriminalGroup, \\ ProtestGroup, CyberArmy \subseteq GroupActor \tag{5.6}$$

$$ScriptKiddie, SkilledHacker \subseteq Hacker \tag{5.7}$$

$$Administrator, NormalUser \subseteq Insider \tag{5.8}$$

The class *ActorLocation* and its sub-classes are displayed in Figure 4.4 and are presented in statement 5.9.

$$Indeterminate, Local, Foreign \subseteq ActorLocation \tag{5.9}$$

The class *Aggressor* and a sub-class *Commercial* are described in statements 5.10 and 5.11.

$$State, Commercial, Individual, SelfInstigator \\ UnknownAggressor \subseteq Aggressor \tag{5.10}$$

$$FlashMob, OrganisedGroup \subseteq Commercial \tag{5.11}$$

The classes *Asset*, *AttackGoal* and *AttackMechanism* are described in statements 5.12 to 5.14. Statements 5.15 to 5.23 give more detail regarding the sub-classes of *AttackMechanism*.

$$Access, Data, Network, System \subseteq Asset \tag{5.12}$$

$$GainControl, GainResources, SpreadAttack, \\ StealData, Disrupt, ChangeData \subseteq AttackGoal \tag{5.13}$$

$$\begin{aligned} &DenialOfService\,AttackMechanism,\\ &ExploitAttackMechanism,\\ &InformationGathering\,AttackMechanism,\\ &MalwareAttackMechanism,\\ &SystemAbuseAttackMechanism \subseteq AttackMechanism \end{aligned} \tag{5.14}$$

$$\begin{aligned} &SocialEngineering,\\ &Scanning,\\ &OpenInformation \subseteq InformationGathering\,AttackMechanism \end{aligned} \tag{5.15}$$

$$SpearPhishing, Spam \subseteq SocialEngineering \tag{5.16}$$

$$Worm, Virus, Trojan \subseteq MalwareAttackMechanism \tag{5.17}$$

$$\begin{aligned} &Distributed\,DenialOfService,\\ &Network\,DenialOfService,\\ &Host\,DenialOfService \subseteq DenialOfService\,AttackMechanism \end{aligned} \tag{5.18}$$

$$\begin{aligned} &NetworkBasedExploit, Access, WebApplication,\\ &PasswordExploit \subseteq Exploit \end{aligned} \tag{5.19}$$

$$BruteForce, Sniffing, Guessing \subseteq PasswordExploit \tag{5.20}$$

$$Physical, BufferOverflow, Escalation \subseteq Access \tag{5.21}$$

$$SQLInjection, WebCrawl, XSS \subseteq WebApplication \tag{5.22}$$

$$Spoofing, SessionHijack, OpenAccess \subseteq NetworkBasedExploit \tag{5.23}$$

Statements 5.24 to 5.27 describe *AttackScenario*, *Automation Level* , *Effect* and *Motivation*. Statement 5.28 describes a sub-class of *Motivation*, namely *Ethical*.

$$\begin{aligned} &DenialOfService, \\ &IndustrialEspionage, \\ &WebDefacement, \\ &SystemCompromise, \\ &FinancialTheft, \\ &UnauthorisedDataAccess \\ &IndustrialSabotage, \\ &CyberWarfare, \\ &ResourceTheft, \\ &RunawayMalware \subseteq AttackScenario \end{aligned} \tag{5.24}$$

$$Manual, SemiAutomatic, Automatic = \subseteq AutomationLevel \tag{5.25}$$

$$Null, Minor, Major, Catastrophic \subseteq Effect \tag{5.26}$$

$$Criminal, Financial, Fun, Ethical \subseteq Motivation \tag{5.27}$$

$$Espionage, Political, Vigilantism \subseteq Ethical \tag{5.28}$$

Statements 5.29 - 5.33 address the classes *Sabotage*, *Scope*, *ScopeSize* and *Target*.

$$\begin{aligned} &OperationalLoss, \\ &FinancialLoss, \\ &PhysicalSabotage, \\ &ReputationalLoss, \\ &SecretLoss, Virtual \subseteq Sabotage \end{aligned} \tag{5.29}$$

$$\begin{aligned} &CriticalInformationInfrastructure,\\ &Corporate, Government,\\ &IndividualScope, Military,\\ &AllNetworks \subseteq Scope \end{aligned} \tag{5.30}$$

$$Global, Large, Medium, Small, Single \subseteq ScopeSize \tag{5.31}$$

$$NetworkInfrastructure, PC, IndustrialEquipment, Server \subseteq Target \tag{5.32}$$

$$ApplicationServer, EmailServer, FileServer, WebServer \subseteq Server \tag{5.33}$$

The last subset of C_{NA}, *Vulnerability*, is described in Statement 5.34, and its sub-classes in statements 5.35 to 5.37.

$$Config, Design, Implementation \subseteq Vulnerability \tag{5.34}$$

$$AccessRights, DefaultSetup \subseteq Config \tag{5.35}$$

$$OpenAccess, ProtocolError \subseteq Design \tag{5.36}$$

$$\begin{aligned} &BufferOverflowVulnerability, RaceCondition,\\ &SQLInjectionVulnerability, VariableTypeChecking \subseteq Implementation \end{aligned} \tag{5.37}$$

5.4.2 Relations

One of the main benefits of an ontology is its capability to express the meaning of domain knowledge. Whilst a taxonomy provides a hierarchical classification of concepts in a domain, an ontology also represents the relationships between the concepts. In this section, the authors describe the relationships between the different classes in the ontology by means of mathematical relations. Statement 5.38 defines the set R_{NA}, whilst statements

5.39 to 5.57 define the elements of R_{NA}, i.e. the relations.

$$\begin{aligned} R_{NA} = \{&hasActor, hasActorLocation,\\ &hasAggressor, hasAsset, hasAttackGoal,\\ &hasAttackMechanism, hasAutomationLevel,\\ &hasEffect, hasMotivation, hasSabotage,\\ &hasScope, hasScopeSize, hasTarget\\ &hasVulnerability\} \end{aligned} \tag{5.38}$$

$$hasActor \subseteq AttackScenario \times Actor \tag{5.39}$$

$$hasActorLocation \subseteq Actor \times ActorLocation \tag{5.40}$$

$$hasAggressor \subseteq Actor \times Aggressor \tag{5.41}$$

$$hasAsset \subseteq Target \times Asset \tag{5.42}$$

$$hasAttackGoal \subseteq Actor \times AttackGoal \tag{5.43}$$

$$hasAttackMechanism \subseteq AttackScenario \times AttackMechanism \tag{5.44}$$

$$\begin{aligned} &hasAttackMechanismRecon,\\ &hasAttackMechanismRampup,\\ &hasAttackMechanismDamage\\ &\subseteq hasAttackMechanism \end{aligned} \tag{5.45}$$

$$\begin{aligned} hasAttackMechanismRecon &\subseteq AttackScenario \times AttackMechanism\\ hasAttackMechanismRampup &\subseteq AttackScenario \times AttackMechanism\\ hasAttackMechanismDamage &\subseteq AttackScenario \times AttackMechanism \end{aligned} \tag{5.46}$$

$$hasAutomationLevel \subseteq AttackMechanism \times AutomationLevel \tag{5.47}$$

$$hasEffect \subseteq Sabotage \times Effect \tag{5.48}$$

$$hasEffectRecon, hasEffectRampup, hasEffectDamage \subseteq hasEffect \tag{5.49}$$

$$\begin{aligned} hasEffectRecon &\subseteq Sabotage \times Effect\\ hasEffectRampup &\subseteq Sabotage \times Effect\\ hasEffectDamage &\subseteq Sabotage \times Effect \end{aligned} \tag{5.50}$$

$$hasMotivation \subseteq Aggressor \times Motivation \tag{5.51}$$

$$hasSabotage \subseteq Asset \times Sabotage \tag{5.52}$$

$$hasScope \subseteq AttackScenario \times Scope \tag{5.53}$$

$$hasScopeSize \subseteq Scope \times ScopeSize \tag{5.54}$$

$$hasTarget \subseteq AttackScenario \times Target \tag{5.55}$$

$$hasVulnerability \subseteq Target \times Vulnerability \tag{5.56}$$

Some of the relations are formed by composition. The compositions of two relations S and R are defined by $S \circ R$ below:

$$\begin{aligned} R &\subseteq X \times Y \\ S &\subseteq Y \times Z \\ S \circ R &= \{(x, z) \in X \times Z \mid \exists y \in Y : (x, y) \in R \wedge (y, z) \in S\} \end{aligned} \tag{5.57}$$

In statements 5.58 to 5.69, the composited relationships are presented (as shown in Figure 5.9):

$$hasChainActorActorLocation = hasActor \circ hasActorLocation \tag{5.58}$$

$$hasChainActorAggressor = hasActor \circ hasAggressor \tag{5.59}$$

$$hasChainActorAttackGoal = hasActor \circ hasAttackGoal \tag{5.60}$$

$$hasChainScopeScopeSize = hasTarget \circ hasScopeSize \tag{5.61}$$

$$hasChainTargetAsset = hasTarget \circ hasAsset \tag{5.62}$$

$$hasChainTargetAssetSabotage = hasTarget \circ hasAsset \circ hasSabotage \tag{5.63}$$

$$\begin{aligned} &hasChainTargetAssetSabotageEffect = \\ &hasTarget \circ hasAsset \circ hasSabotage \circ hasEffect \end{aligned} \tag{5.64}$$

$$\begin{aligned} &hasChainTargetAssetSabotageEffectRecon, \\ &hasChainTargetAssetSabotageEffectRampup, \\ &hasChainTargetAssetSabotageEffectDamage \\ &\subseteq hasChainTargetAssetSabotageEffect \end{aligned} \tag{5.65}$$

$$hasChainTargetVulnerability = hasTarget \circ hasVulnerability \tag{5.66}$$

$$\begin{aligned} &hasChainActorAggressorMotivation \\ &= hasActor \circ hasAggressor \circ hasMotivation \end{aligned} \tag{5.67}$$

$$\begin{aligned} &hasChainAMAutomationLevel \\ &= hasAttackMechanism \circ hasAutomationLevel \end{aligned} \tag{5.68}$$

$$\begin{aligned} &hasChainAMAutomationLevelRecon, \\ &hasChainAMAutomationLevelRampup, \\ &hasChainAMAutomationLevelDamage \\ &\subseteq hasChainAMAutomationLevel \end{aligned} \tag{5.69}$$

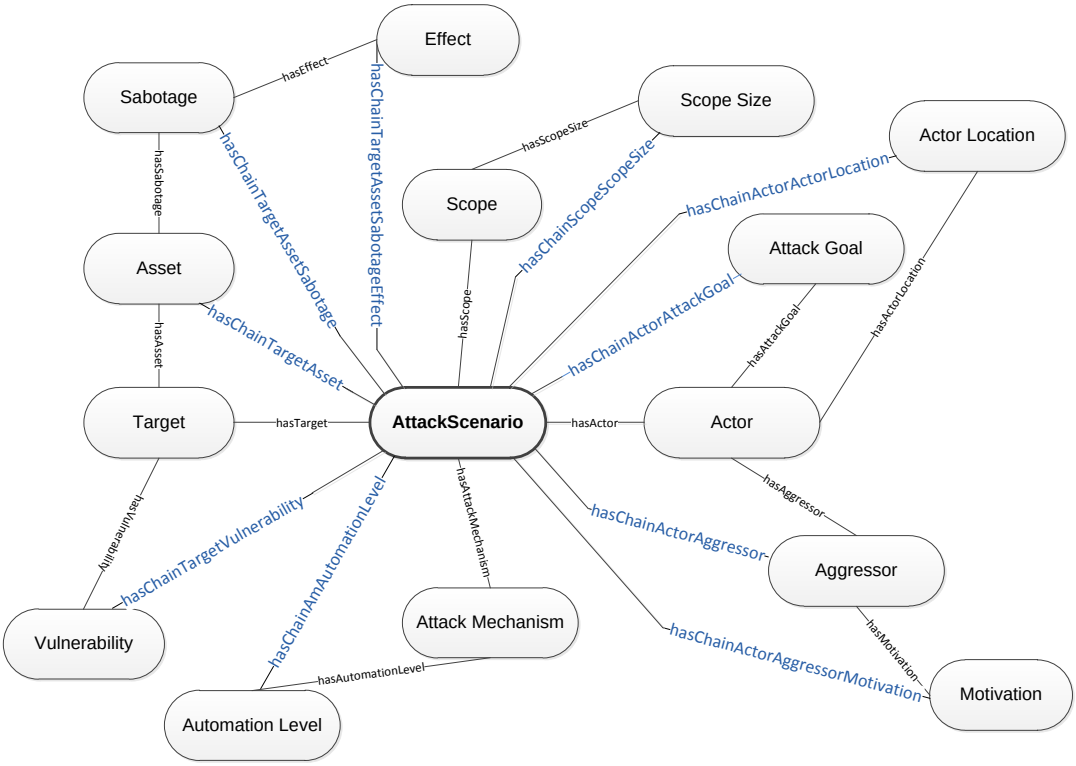


Figure 5.9: Composition Relationships

5.4.3 Constraints on Classes

In this section, the set Attack Scenario (AS) is described (refer to Figure 5.5). The symbol $\exists$ is the first-order existential quantifier: *there exists at least one element*. The symbol $\ni$ is used to express the words: *such that*. The symbol $\in$ represents to the classical set

theory operator: *element of*. The symbol $\wedge$ represents to the logical operator: *and*. The constrained definition of the set AS is presented in Statement 5.70.

$$\begin{aligned} AttackScenario = \{x | (\exists z \in Scope \ni (x,z) \in hasScope) \wedge \\ (\exists v \in Actor \ni (x,v) \in hasActor) \wedge \\ (\exists w \in AttackMechanism \ni (x,w) \in hasAttackMechanism) \wedge \\ (\exists u \in Target \ni (x,u) \in hasTarget)\} \end{aligned} \tag{5.70}$$

Statement 5.70 further constrains the *Attack Scenario* set such that for every element x of the set AS, as depicted in Figure 5.10. The following conjunction hold:

- At least one element exists, z, which is a member of the set $Scope$, and is such that the ordered pair (x, z) participates in the relation $hasScope$.
- At least one element exists, v, which is a member of the set $Actor$, and is such that the ordered pair (x, v) participates in the relation $hasActor$.
- At least one element exists, w, which is a member of the set $AttackMechanism$, and is such that the ordered pair (x, w) participates in the relation $hasAttackMechanism$.
- At least one element exists, u, which is a member of the set $Target$, and is such that the ordered pair (x, u) participates in the relation $hasTarget$.

Similarly, constraints for the sets $Scope$, $Actor$, $Aggressor$, $AttackMechanism$, $Target$, $Asset$ and $Sabotage$ are defined.

$$Scope = \{x | (\exists y \in ScopeSize \ni (x,y) \in hasScopeSize)\} \tag{5.71}$$

$$\begin{aligned} Actor = \{x | (\exists z \in AttackGoal \ni (x,z) \in hasAttackGoal) \wedge \\ (\exists v \in ActorLocation \ni (x,v) \in hasActorLocation) \wedge \\ (\exists w \in Aggressor \ni (x,w) \in hasAggressor)\} \end{aligned} \tag{5.72}$$

$$Aggressor = \{x | (\exists y \in Motivation \ni (x,y) \in hasMotivation)\} \tag{5.73}$$

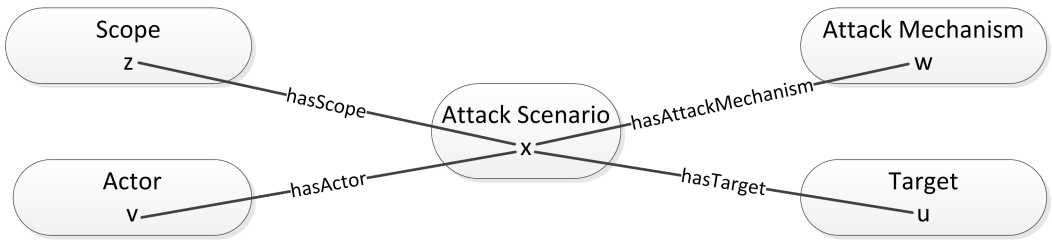


Figure 5.10: Statement 5.70

$$AttackMechanism = \{x|(\exists z \in AutomationLevel \ni (x,z) \in hasAutomationLevel)\} \tag{5.74}$$

$$\begin{aligned} Target = \{x|(\exists z \in Asset \ni (x,z) \in hasAsset) \wedge \\ (\exists y \in Vulnerability \ni (x,y) \in hasVulnerability)\} \end{aligned} \tag{5.75}$$

$$Asset = \{x|(\exists y \in Sabotage \ni (x,y) \in hasSabotage)\} \tag{5.76}$$

$$Sabotage = \{x|(\exists y \in Effect \ni (x,y) \in hasEffect)\} \tag{5.77}$$

5.4.4 Denial-of-Service Scenario Formal Definition

The goal of a Denial-of-Service (DoS) attack is to prevent or impair the legitimate use of computer networks (Houle and Weaver, 2001). One of the most frequent methods that DoS attacks use, is to flood a single network point with network traffic. This flood of traffic will then prevent normal network operations. DDoS attacks disrupt networks by flooding them with traffic from multiple sources. These sources can number in the millions.

The *Denial-of-Service* scenario set is defined in statements 5.78 to 5.84 (also refer to Figure 5.6). In Figure 5.6, the sub-classes that are specific to the *Denial-of-Service* scenario are displayed. This demonstrates which sub-classes are used when the *Denial-of-Service* attack scenario is presented. For example, only the *OperationalLoss* sub-class is used from *Sabotage* class. Note *Attack Mechanism* may be shortened to AM and *Denial-of-Service* is shortened to *DenialOfService* or *DoS*.

$$DoS \subseteq AttackScenario \tag{5.78}$$

$$HackerDoS \subseteq Hacker \subseteq Actor \tag{5.79}$$

$$DisruptDoS \subseteq Disrupt \subseteq AttackGoal \tag{5.80}$$

$$\begin{aligned} DenialOfService\,AM\_DoS \subseteq \\ DenialOfService\,AM \subseteq AttackMechanism \end{aligned} \tag{5.81}$$

$$DoS = \{x|(\exists v \in Hacker \ni (x,v) \in hasActor) \wedge (\exists w \in DenialOfService\,AM \ni (x,w) \in hasAttackMechanismRampup) \wedge (\exists y \in DenialOfService\,AM \ni (x,y) \in hasAttackMechanismDamage) \wedge (\exists u \in NetworkInfrastructure \ni (x,u) \in hasTarget)\} \tag{5.82}$$

$$NetworkInfrastructure\,DoS \subseteq NetworkInfrastructure \subseteq Target \tag{5.83}$$

$$OperationalLossDoS \subseteq OperationalLoss \subseteq Sabotage \tag{5.84}$$

$$NullDoS \subseteq Null \subseteq Effect \tag{5.85}$$

$$MajorDoS \subseteq Major \subseteq Effect \tag{5.86}$$

$$HackerDoS = \{x|\exists z \in Disrupt \ni (x,z) \in hasAttackGoal\} \tag{5.87}$$

$$DenialOfService\,AM\_DoS = \{x|\exists z \in Automatic \ni (x,z) \in hasAutomationLevel)\} \tag{5.88}$$

$$AutomaticDoS \subseteq Automatic \subseteq AutomationLevel \tag{5.89}$$

$$NetworkInfrastructureDoS = \{x|\exists z \in Access \ni (x,z) \in hasAsset)\} \tag{5.90}$$

$$AccessDoS \subseteq Access \subseteq Asset \tag{5.91}$$

$$AccessDoS = \{x|\exists y \in OperationalLoss \ni (x,y) \in hasSabotage\} \tag{5.92}$$

$$SabotageDoS = (\{x|\exists y \in Null \ni (x,y) \in hasSabotageRampup \wedge \exists z \in Major \ni (x,z) \in hasSabotageDamage)\} \tag{5.93}$$

5.5 Inferring Class Membership of Individuals

This section demonstrates how Protégé is able to infer to which class an individual belongs (van Heerden, Leenen, and Irwin, 2013a). The ontology classes and relationships were implemented in Protégé and information about attacks was used to populate the individual's properties. Two of the significant attacks discussed in Section 5.3.1, the SCO and SpamHaus attacks (discussed in sections 2.4.18 and 2.4.35), fit the description of the *Denial-of-Service* scenario. The SCO individuals were defined in Protégé as follows:

- is a member of the *Attack Scenario* class; and
- has at least one:

  - *Hacker Actor* defined by *hasActor* relationship;
  - *Denial-of-Service Attack Mechanism* defined by *hasAttackMechanismDamage* relationship;
  - *Denial-of-Service Attack Mechanism* defined by *hasAttackMechanismRampup* relationship;
  - *Automatic Level* defined by *hasChainAMAutomationLevel* relationship;
  - *Indeterminate Actor Location* defined by *hasChainActorActorLocation* relationship;
  - *Flash Mob Aggressor* defined by *hasChainActorAggressor* relationship;
  - *Vigilantism Motivation* defined by *hasChainActorAggressorMotivation* relationship;
  - *Disrupt Attack Goal* defined by *hasChainActorAttackGoal* relationship;
  - *Medium Network* defined by *hasChainScopeScopeSize* relationship;
  - *Access Asset* defined by *hasChainTargetAsset* relationship;
  - *Operational Loss* defined by *hasChainTargetAssetSabotage* relationship;
  - *Major Loss Effect* defined by *hasChainTargetAssetSabotageEffectDamage* relationship;
  - *Null Effect* defined by *hasChainTargetAssetSabotageEffectRampup* relationship;
  - *Protocol Error Vulnerability* defined by *hasChainTargetVulnerability* relationship;
  - *Corporate Network* defined by *hasScope* relationship;
  - *Network Infrastructure Device Target* defined by *hasTarget* relationship.

By setting an individual using the properties as above, the automated reasoner HermiT plug-in for Protégé was able to determine that the SCO attacks fall within the *Denial-of-Service* scenario. Protégé output is shown in Figure 5.11, with the automated reasoner-inferred class shown in yellow at the bottom.

The SpamHaus individual was defined in Protégé with the following characteristics:

- is a member of the *Attack Scenario* class; and
- has at least one:
  - *Hacker Actor* defined by *hasActor* relationship;
  - *Denial-of-Service Attack Mechanism* defined by *hasAttackMechanismDamage* relationship;
  - *Denial-of-Service Attack Mechanism* defined by *hasAttackMechanismRampup* relationship;
  - *Automatic Level* defined by *hasChainAMAutomationLevel* relationship;

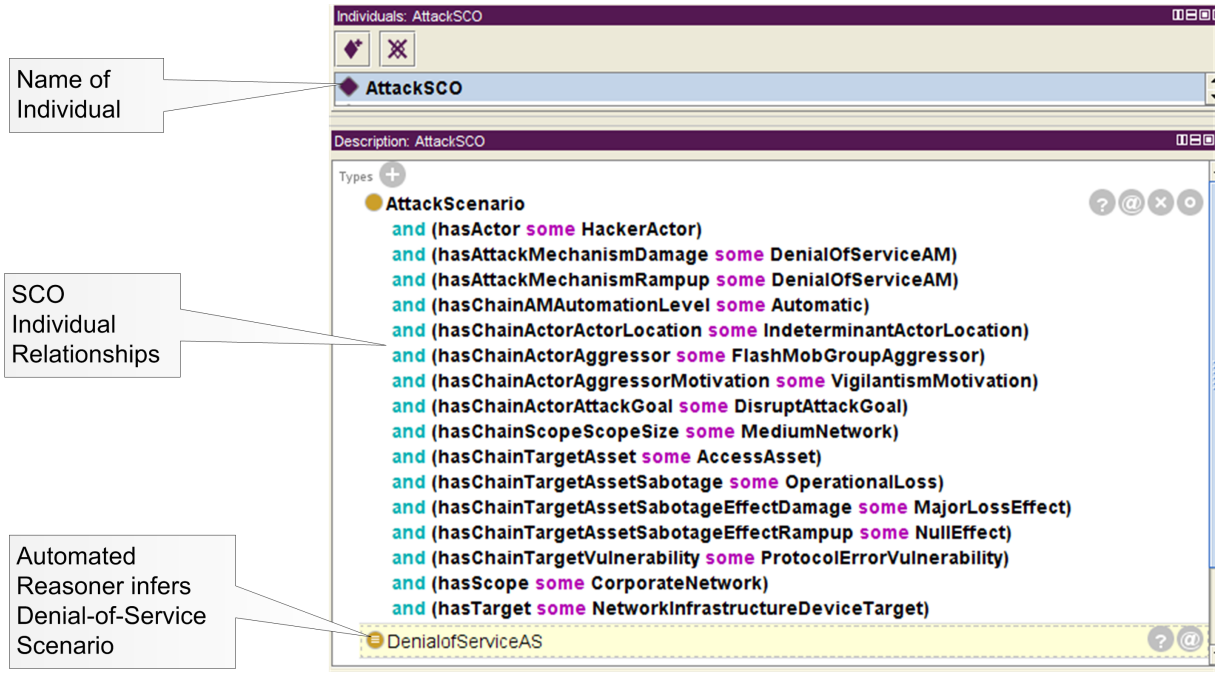


Figure 5.11: SCO Attack Inferred a *Denial-of-Service* Scenario

- *Foreign Actor Location* defined by *hasChainActorActorLocation* relationship;
- *Commercial Aggressor* defined by *hasChainActorAggressor* relationship;
- *Vigilantism Motivation* defined by *hasChainActorAggressorMotivation* relationship;
- *Disrupt Attack Goal* defined by *hasChainActorAttackGoal* relationship;
- *Large Network* defined by *hasChainScopeScopeSize* relationship;
- *Access Asset* defined by *hasChainTargetAsset* relationship;
- *Operational Loss* defined by *hasChainTargetAssetSabotage* relationship;
- *Major Loss Effect* defined by *hasChainTargetAssetSabotageEffectDamage* relationship;
- *Null Effect* defined by *hasChainTargetAssetSabotageEffectRampup* relationship;
- *Protocol Error Vulnerability* defined by *hasChainTargetVulnerability* relationship;
- *Corporate Network* defined by *hasScope* relationship;
- *Network Infrastructure Device Target* defined by *hasTarget* relationship.

By creating an individual using the properties as shown above, the automated reasoner HermiT plug-in for Protégé was able to determine that the SpamHaus attack belongs to the *Denial-of-Service* scenario. Figure 5.12 shows the entry for this individual in Protégé: the properties of the individual are shown in the first two blocks and the output of the automated reasoner-inferred class is highlighted in the bottom block, i.e. this individual

is a member of the Denial-of-Service Attack Scenario.

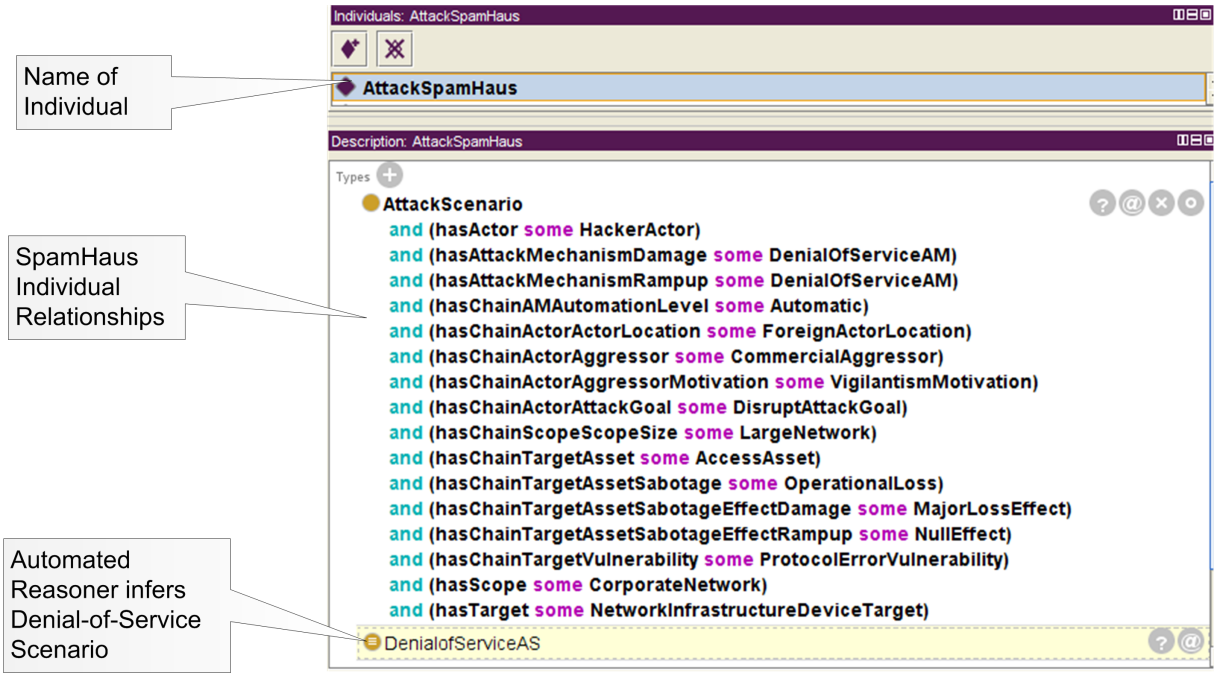


Figure 5.12: SpamHaus Attack Inferred a *Denial-of-Service* Scenario

Refer to Chapter 6 for examples of individuals of the other attack scenarios. In Chapter 6, the formal description and individuals for *Web Defacement*, *Unauthorised Data Access*, *Cyber-Warfare*, *Industrial Espionage*, *Financial Theft*, *Resource Theft*, *Industrial Sabotage* and *Runaway Malware* are presented.

5.6 Summary

In this chapter, the ontology that describes computer network attacks is presented. This ontology used the classes of the taxonomy and binds them with relationships that describe how the classes are related. The ontology is presented in story form, formally, and implemented within the Protégé editor[6]. Two attack individuals are represented within Protégé, and the HermiT automated reasoner is used to determine to which attack scenario they belong. Thus the ontology and automated reasoner can classify network attacks into their associated scenario. In the next chapter, the impact on the ontology in a near real-time environment is explored. The ontology relationships are relaxed to only include classes that have an impact in a near real-time environment.

[6]http://www.networkattackontology.com/Ontology/

CHAPTER

SIX

DETAILED ONTOLOGY

"Men rise from one ambition to another: first, they seek to secure themselves against attack, and then they attack others."

Niccolo Machiavelli – 1513

6.1 Introduction

The chapter describes in detail the ontologies for each of the attack scenarios, similar to how the Denial-of-Service scenario is described in Section 5.3.1. For each of the scenarios, their stories with an example is presented and detailed mathematical definitions are presented. In sections 6.2 to 6.10, the attack scenarios stories, formal descriptions and individuals are presented. A conclusion of this chapter is presented in Section 6.11.

6.2 Web Defacement

Web defacement can be considered graffiti of the digital world (Lewis, 2007). Websites are the public face of commercial and other entities in the digital world, and their reputations are negatively effected by defacing it. The web defacement scenario refers to attacks

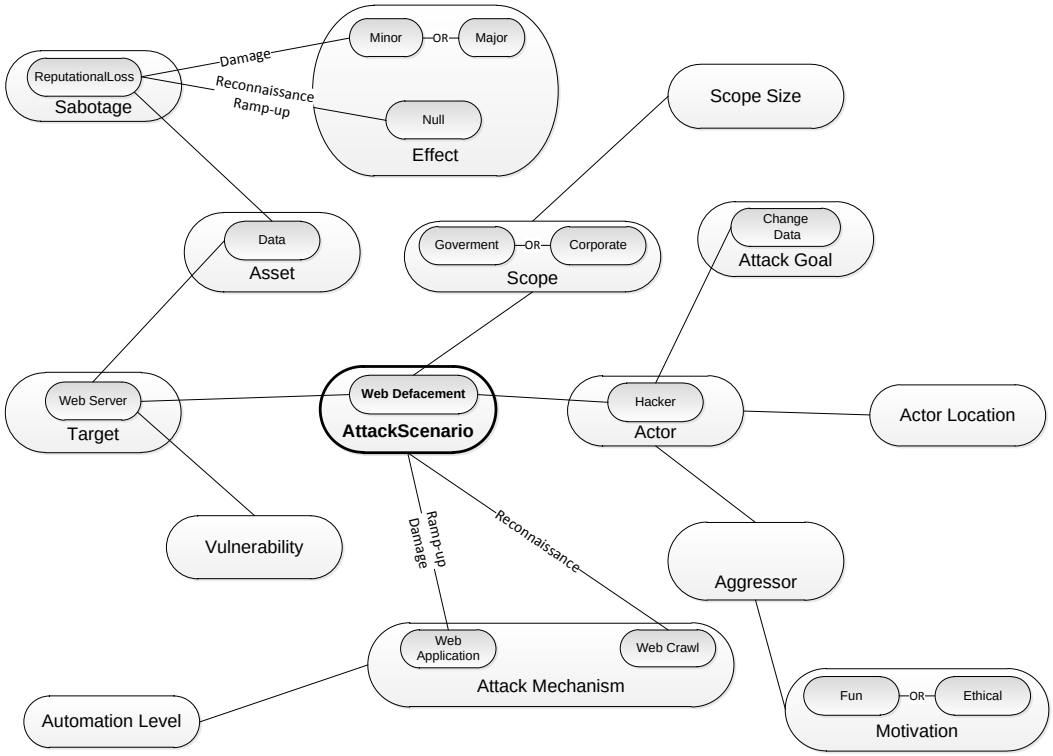


Figure 6.1: Web Defacement Attack Scenario

directed at a website's content. Web defacement is an attack on a website, which performs unauthorised changes to a specific web page. Such changes include altering the visual appearance, written content or overall message of the website into a form that is offensive and can potentially harm a company's reputation.

The story that describes the Web Defacement Scenario follows (Figure 6.1):
*A **Hacker** based at **[ActorLocation]** location with the goal of **Change Data** sponsored by **[Aggressor]** with a **Fun OR Ethical** motivation The attack effected **[ScopeSize] Corporate OR Government Network** scope. A **Server** was attacked via **[Vulnerability]**. This attack effected **Data** and resulted in **Reputation Loss** to **Minor OR Major** effect during the **Damage** attack phase. During the Reconnaissance phase **Web Crawl** was used, during the Ramp-up phase **Web Application** was used and during the Damage phase **Web Application** was used. The attack was automated to **[AutomationLevel]** level.*

The Apache.org (Section 2.4.15) is an attack that can be classified as a Web Defacement Attack Scenario (Figure 6.2):
*A **Hacker** based at **Foreign (Netherlands)** location with the goal of **Change Data** sponsored by **Self Instigator** with a **Fun** motivation The attack effected **Medium Cor-***

porate (Apache.org) *scope. A* ***Server*** *was attacked via* ***Configuration*** *vulnerability. This attack effected* ***Data*** *and resulted in* ***Reputation Loss*** *to* ***Minor*** *effect during the* ***Damage*** *attack phase. During the Reconnaissance phase* ***Web Crawl*** *was used, during the Ramp-up phase* ***Web Application*** *was used and during the Damage phase* ***Web Application*** *was used. The attack was automated to* ***Manual*** *level.*

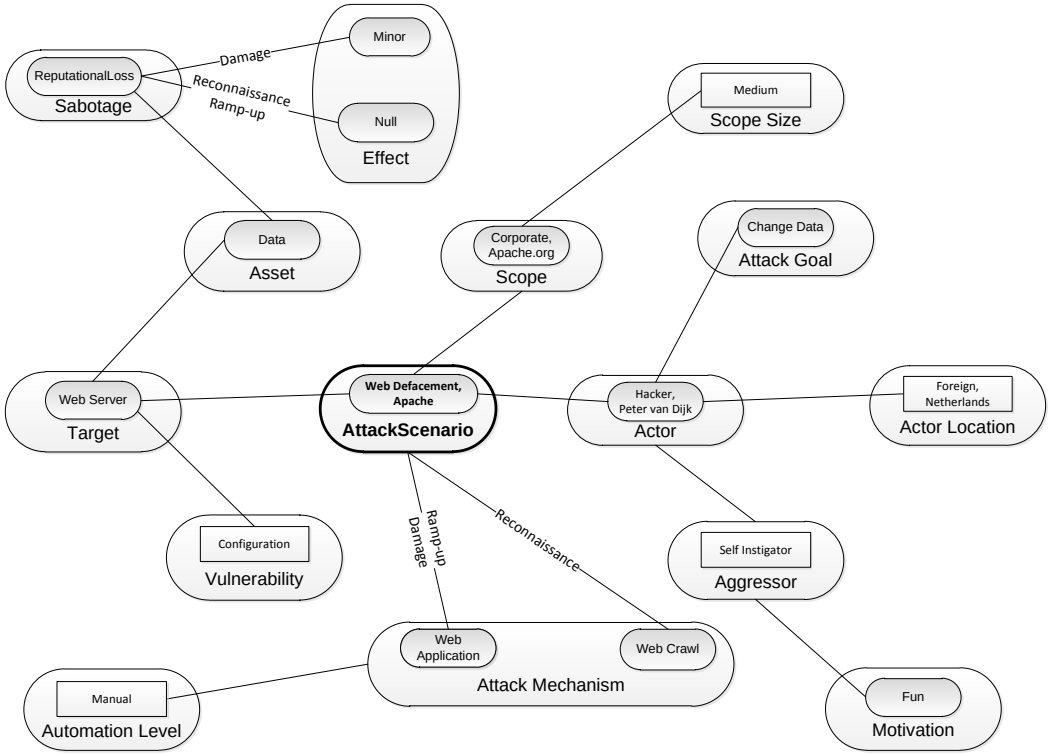


Figure 6.2: Apache.org Web Defacement Attack Scenario Example

6.2.1 Web Defacement Formal Description

The *Web Defacement* scenario set is defined in statements 6.2 to 6.12 (also refer to Figure 6.1). In Figure 6.1, the sub-classes that are specific to the *Web Defacement* scenario are displayed. This demonstrates which sub-classes are used when the Web Defacement attack scenario is presented. For example, only the *OperationalLoss* sub-class is used from the *Sabotage* class. The *Web Defacement* class name is shortened to *WD* in statements 6.1 to 6.12.

$$WebDefacement \subseteq AttackScenario \tag{6.1}$$

$$\begin{aligned} WD = \{x | (\exists v \in Hacker \ni (x,v) \in hasActor) \wedge \\ (\exists w \in WebCrawl \ni (x,w) \in hasAttackMechanismRecon) \wedge \\ (\exists y \in WebApplication \ni (x,y) \in hasAttackMechanismRampup) \wedge \\ (\exists s \in WebApplication \ni (x,s) \in hasAttackMechanismDamage) \wedge \\ (\exists t \in (Government \cup Corporate) \ni (x,t) \in hasScope) \wedge \\ (\exists u \in WebServer \ni (x,u) \in hasTarget)\} \end{aligned} \tag{6.2}$$

$$HackerWD \subseteq Hacker \subseteq Actor \tag{6.3}$$

$$\begin{aligned} HackerWD = \{x | \exists z \in ChangeData \ni \\ (x,z) \in hasAttackGoal\} \end{aligned} \tag{6.4}$$

$$WebServerWD \subseteq WebServer \subseteq Target \tag{6.5}$$

$$WebServerWD = \{x | \exists z \in Data \ni (x,z) \in hasAsset\} \tag{6.6}$$

$$DataWD \subseteq Data \subseteq Asset \tag{6.7}$$

$$DataWD = \{x | \exists y \in ReputationalLoss \ni (x,y) \in hasSabotage\} \tag{6.8}$$

$$ReputationalLossWD \subseteq ReputationalLoss \subseteq Sabotage \tag{6.9}$$

$$\begin{aligned} ReputationalLossWD = \{x | (\exists y \in Null \ni (x,y) \in hasSabotageRecon) \wedge \\ (\exists v \in Null \ni (x,v) \in hasSabotageRampup) \wedge \\ (\exists z \in (Minor \cup Major) \ni (x,z) \in hasSabotageDamage)\} \end{aligned} \tag{6.10}$$

$$AggressorWD \subseteq Aggressor \tag{6.11}$$

$$\begin{aligned} AggressorWD = \{x | (\exists z \in (Fun \cup Ethical) \ni \\ (x,z) \in hasMotivation)\} \end{aligned} \tag{6.12}$$

6.2.2 Web Defacement Individual

The Apache.org attack (Section 2.4.15) was inferred as part of the *Web Defacement* scenario. The Apache.org attack individual was defined in Protégé as follows:

- is a member of the *Attack Scenario* class; and
- has at least one:
  - *Hacker Actor* defined by *hasActor* relationship;

- *Web Application Attack Mechanism* defined by *hasAttackMechanismDamage* relationship;
- *Web Application Attack Mechanism* defined by *hasAttackMechanismRampup* relationship;
- *Web Crawl Attack Mechanism* defined by *hasAttackMechanismRecon* relationship;
- *Manual Automation Level* defined by *hasChainAMAutomationLevel* relationship;
- *Foreign Actor Location* defined by *hasChainActorActorLocation* relationship;
- *Self Instigator Aggressor* defined by *hasChainActorAggressor* relationship;
- *Fun Motivation* defined by *hasChainActorAggressorMotivation* relationship;
- *Change Data Attack Goal* defined by *hasChainActorAttackGoal* relationship;
- *Medium Network* defined by *hasChainScopeScopeSize* relationship;
- *Data Asset* defined by *hasChainTargetAsset* relationship;
- *Reputational Loss* defined by *hasChainTargetAssetSabotage* relationship;
- *Minor Loss Effect* defined by *hasChainTargetAssetSabotageEffectDamage* relationship;
- *Null Effect* defined by *hasChainTargetAssetSabotageEffectRampup* relationship;
- *Null Effect* defined by *hasChainTargetAssetSabotageEffectRecon* relationship;
- *Configuration Error Vulnerability* defined by *hasChainTargetVulnerability* relationship;
- *Corporate Network* defined by *hasScope* relationship;
- *Web Server Target* defined by *hasTarget* relationship.

By setting an individual using the properties as above, the automated reasoner HermiT plug-in for Protégé was able to determine that the Apache.org attacks fall within the *Web Defacement* scenario. Protégé output is shown in Figure 6.3, with the automated reasoner-inferred class shown in yellow at the bottom. Note, the terms Web Defacement and Web Defacing are interchangeable.

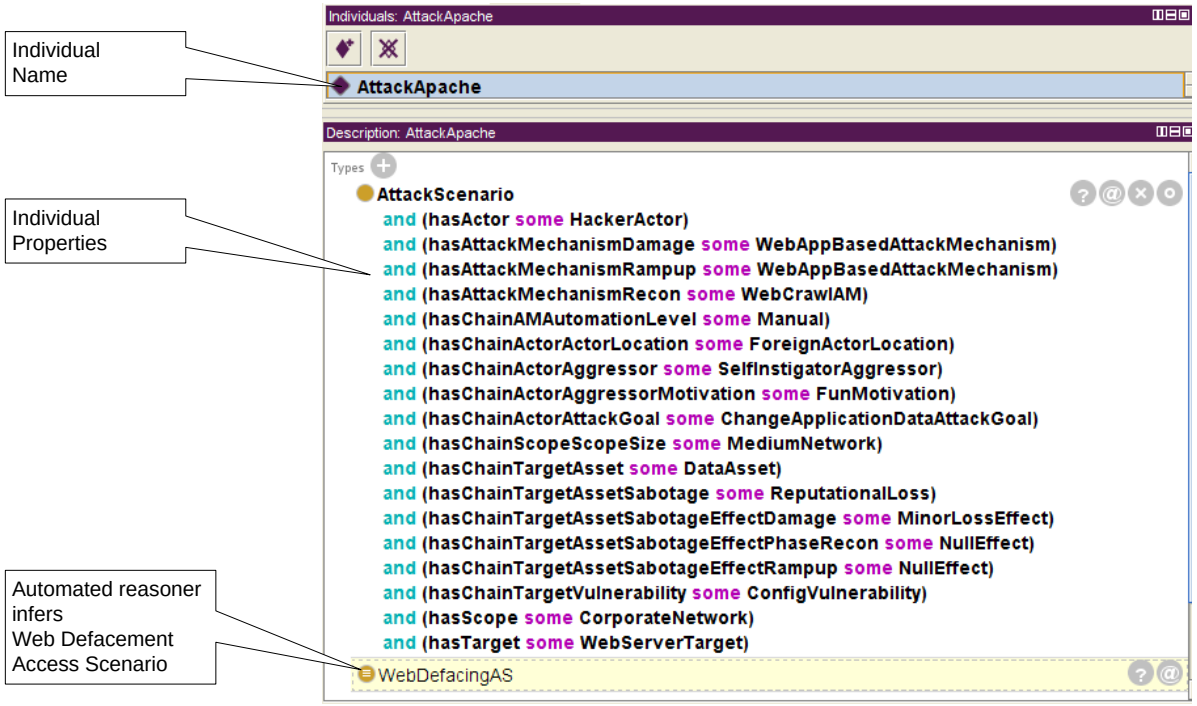


Figure 6.3: Apache.org Attack Inferred a *Web Defacement* Scenario

6.3 Unauthorised Data Access

This scenario refers to hackers/insiders gaining access data which they don't have permission to access. Unauthorised data access refers to a situation where a person has access to a location that is hidden or contains sensitive data. This can lead to the unauthorised entry of data into a file, reading a file, changing the contents of the file or for any other malicious purpose.

The story that describes the Unauthorised Data Access Scenario follows (Figure 6.4):
*A **Hacker** based at **[ActorLocation]** location with the goal of **Steal Data OR Change Data** sponsored by **[Aggressor]** with a **[Motivation]**. The attack effected **[ScopeSize] Corporate OR Government Network OR Military** scope. A **Server** was attacked via **[Vulnerability]**. This attack effected **Data** and resulted in **Secret Loss** to **Null** effect during the Reconnaissance attack phase, to **Null OR Minor** effect during the Ramp-up attack phase and to **Minor OR Major** effect during the Damage attack phase. During the Reconnaissance phase **Open Information** was used, during the Ramp-up phase **Scanning** was used and during the Damage phase **Exploit** was used. The attack was automated to **[AutomationLevel]** level.*

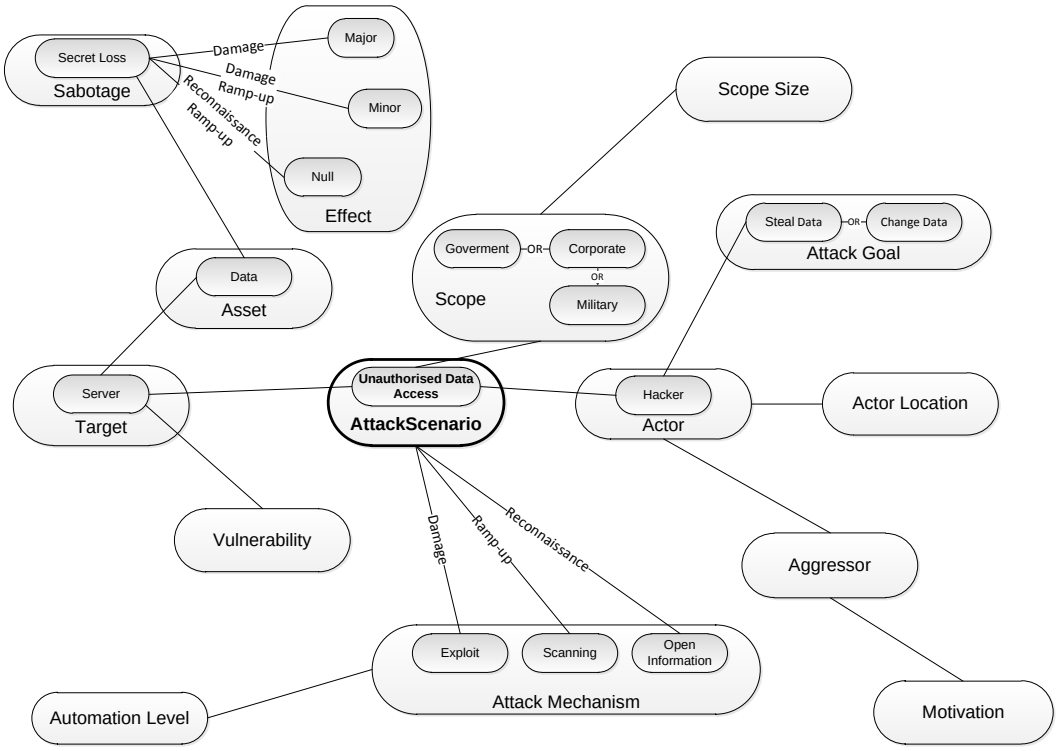


Figure 6.4: Unauthorised Data Access Attack Scenario

Kevin Mitnick (Section 2.4.8) gained unauthorised access to multiple classified computer systems (Figure 6.5):
*A **Hacker (Kevin Mitnick)** based at **Local (USA)** location with the goal of **Steal Data** sponsored by **Self Instigator** with a **Fun** motivation. The attack effected **Large Corporate** scope. A **Server** was attacked via **Access Rights** vulnerability. This attack effected **Data** and resulted in **Secret Loss** to **Null** effect during the Reconnaissance attack phase, to **Minor** effect during the Ramp-up attack phase and to **Minor** effect during the Damage attack phase. During the Reconnaissance phase **Open Information** was used, during the Ramp-up phase **Scanning** was used and during the Damage phase **Access Attack Mechanism** was used. The attack was automated to **Manual** level.*

6.3.1 Unauthorised Data Access Formal Description

The *Unauthorised Data Access (UDA)* scenario set is defined in statements 6.13 to 6.22 (also refer to Figure 6.4). In Figure 6.4, the sub-classes that are specific to the *Unauthorised Data Access* scenario are displayed. This demonstrates which sub-classes are used

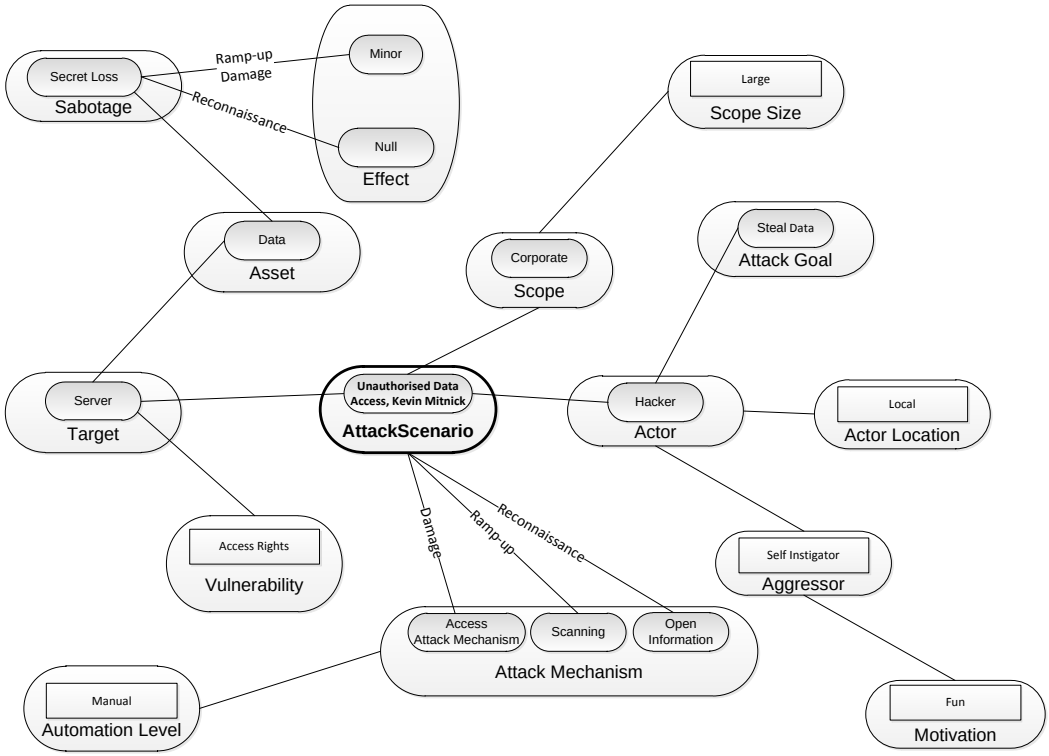


Figure 6.5: Kevin Mitnick *Unauthorised Data Access* Attack Scenario Example

when the Unauthorised Data Access attack scenario is presented.

$$UDA \subseteq AttackScenario \tag{6.13}$$

$$\begin{aligned} UDA = \{x | (\exists v \in Hacker \ni (x,v) \in hasActor) \wedge \\ (\exists w \in OpenInformation \ni (x,w) \in hasAttackMechanismRecon) \wedge \\ (\exists y \in Scanning \ni (x,y) \in hasAttackMechanismRampup) \wedge \\ (\exists s \in Exploit \ni (x,s) \in hasAttackMechanismDamage) \wedge \\ (\exists t \in (Government \cup Corporate \cup Military) \cup (x,t) \in hasScope) \wedge \\ (\exists u \in Server \ni (x,u) \in hasTarget)\} \end{aligned} \tag{6.14}$$

$$HackerUDA \subseteq Hacker \subseteq Actor \tag{6.15}$$

$$HackerUDA = \{x | \exists z \in (StealData \cup ChangeData) \ni (x,z) \in hasAttackGoal\} \tag{6.16}$$

$$Server\ UDA \subseteq Server \subseteq Target \tag{6.17}$$

$$Server\ UDA = \{x | \exists z \in Data \ni (x,z) \in hasAsset)\} \tag{6.18}$$

$$Data\ UDA \subseteq Data \subseteq Asset \tag{6.19}$$

$$Data\ UDA = \{x | \exists y \in SecretLoss \ni (x, y) \in hasSabotage\} \tag{6.20}$$

$$SecretLoss\ UDA \subseteq SecretLoss \subseteq Sabotage \tag{6.21}$$

$$\begin{aligned} SecretLoss\ UDA = \{x | (\exists y \in Null \ni (x, y) \in hasSabotageRecon \wedge \\ (\exists v \in (Null \cup Minor) \ni (x, v) \in hasSabotageRampup) \wedge \\ (\exists z \in (Minor \cup Major) \ni (x, z) \in hasSabotageDamage)\} \end{aligned} \tag{6.22}$$

6.3.2 Unauthorised Data Access Individual

The Kevin Mitnick attack individual (Section 2.4.8) was defined in Protégé as follows:

- is a member of the *Attack Scenario* class; and
- has at least one:
  - *Hacker Actor* defined by *hasActor* relationship;
  - *Access Attack Mechanism* defined by *hasAttackMechanismDamage* relationship;
  - *Scanning Attack Mechanism* defined by *hasAttackMechanismRampup* relationship;
  - *Open Information Attack Mechanism* defined by *hasAttackMechanismRecon* relationship;
  - *Manual Automation Level* defined by *hasChainAMAutomationLevel* relationship;
  - *Local Actor Location* defined by *hasChainActorActorLocation* relationship;
  - *Self Instigator Aggressor* defined by *hasChainActorAggressor* relationship;
  - *Fun Motivation* defined by *hasChainActorAggressorMotivation* relationship;
  - *Steal Data Attack Goal* defined by *hasChainActorAttackGoal* relationship;
  - *Large Network* defined by *hasChainScopeScopeSize* relationship;
  - *Data Asset* defined by *hasChainTargetAsset* relationship;
  - *Secret Loss* defined by *hasChainTargetAssetSabotage* relationship;
  - *Minor Loss Effect* defined by *hasChainTargetAssetSabotageEffectDamage* relationship;
  - *Minor Effect* defined by *hasChainTargetAssetSabotageEffectRampup* relationship;
  - *Null Effect* defined by *hasChainTargetAssetSabotageEffectRecon* relationship;

- *Access Rights Vulnerability* defined by *hasChainTargetVulnerability* relationship;
- *Corporate Network* defined by *hasScope* relationship;
- *Server Target* defined by *hasTarget* relationship.

By setting an individual using the properties as above, the automated reasoner HermiT plug-in for Protégé was able to determine that the Kevin Mitnick attacks fall within the *Unauthorised Data Access* scenario. Protégé output is shown in Figure 6.6, with the automated reasoner-inferred class shown in yellow at the bottom.

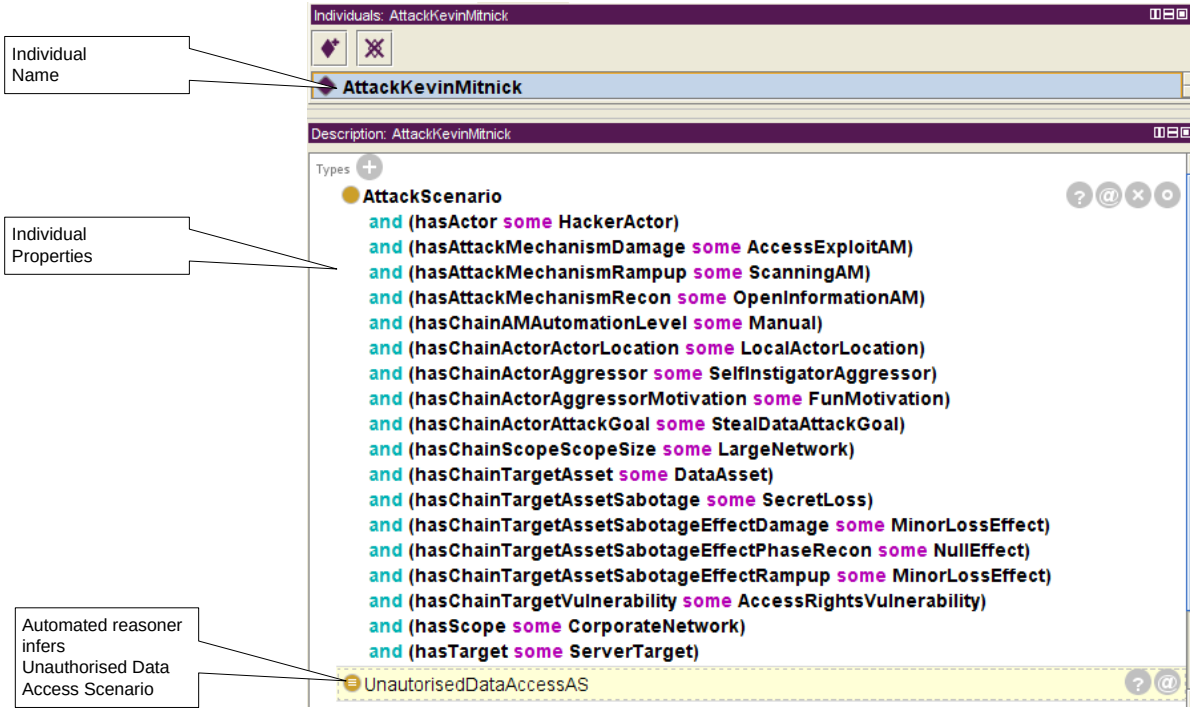


Figure 6.6: Kevin Mitnick Attacks Inferred to *Unauthorised Data Access* Scenario

6.4 Cyber-Warfare

Cyber-Warfare is the practical use of hacking and other information operations methodologies by a state against another state (Ophardt, 2010). The story that describes the Cyber-Warfare scenario follows (Figure 6.7):

*A **Cyber Army** based at **Foreign** location with the goal of **Disrupting** sponsored by **State** with a **Political** motivation, The attack effected **Large Corporate OR Military OR Government** scope. A **Network Infrastructure** was attacked via **[Vulnerability]**. This attack effected **Access** and resulted in **Operational Loss** to **Major OR***

Catastrophic *effect during the Damage attack phase and resulted in* ***Minor OR Null*** *effect during the Ramp-up attack phase. During the Ramp-up and Damage phase the* ***Denial-of-Service Mechanism*** *was used, and was automated to* ***[AutomationLevel]*** *level.*

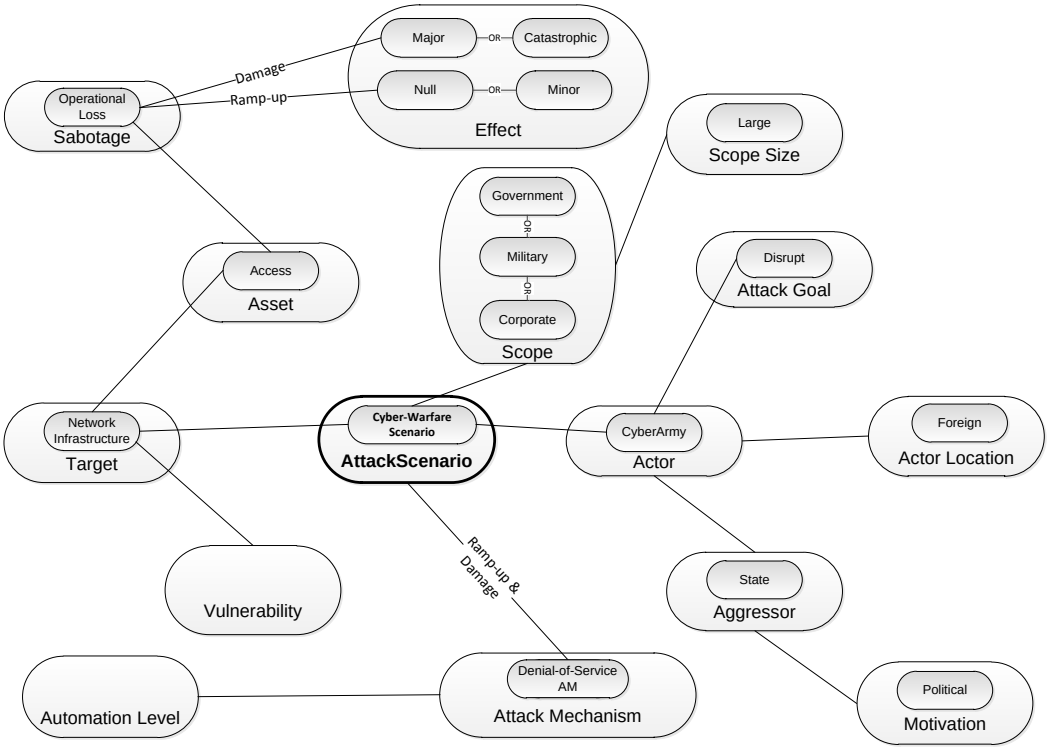


Figure 6.7: *Cyber-Warfare* Attack Scenario

The cyber-attack on Estonia (Section 2.4.24) is presented in Figure 6.8.
A ***Cyber Army*** *based at* ***Russia (Foreign)*** *with the goal of* ***Disrupting*** *sponsored by* ***State*** *with a* ***Political*** *motivation The attack effected* ***Large Corporate OR Government*** *scope. A* ***Network Infrastructure*** *was attacked via* ***[Vulnerability]****. This attack effected* ***Access*** *and resulted in* ***Operational Loss*** *to* ***Catastrophic*** *effect during the Damage attack phase and resulted in* ***Minor*** *effect during the Ramp-up attack phase. During the Ramp-up and Damage phase the* ***Denial-of-Service Mechanism*** *was used, and was automated to* ***[AutomationLevel]*** *level.*

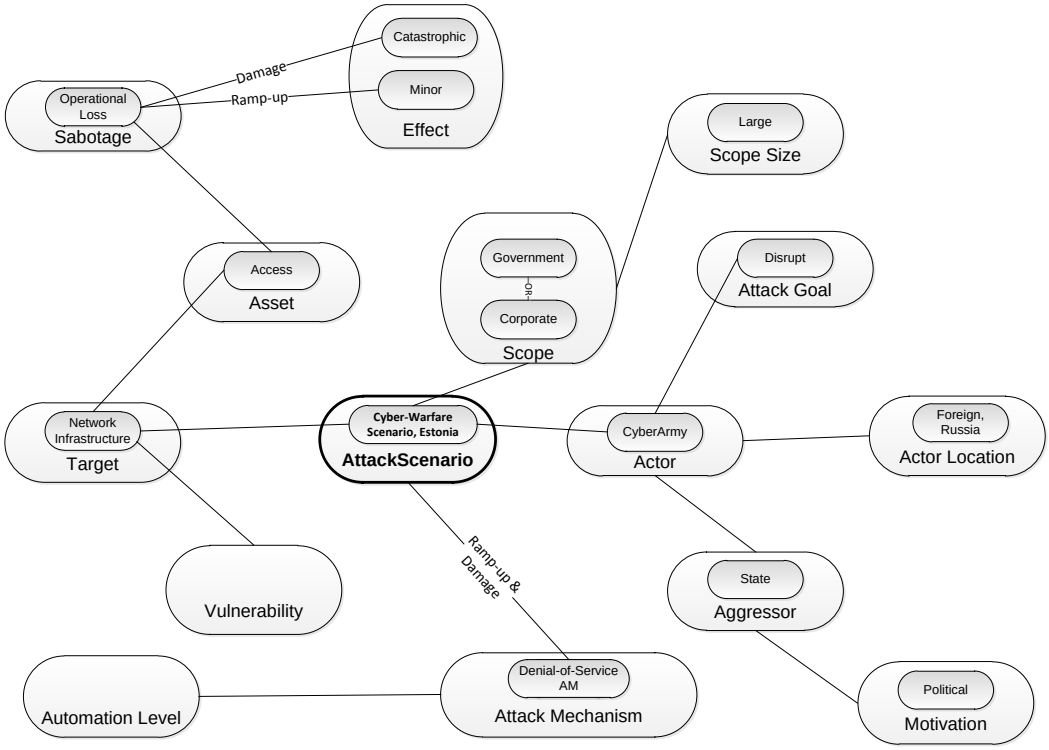


Figure 6.8: Estonia Cyber-Attack Scenario

6.4.1 Cyber-Warfare Formal Description

The *Cyber-Warfare (CW)* scenario set is defined in statements 6.23 to 6.36 (also refer to Figure 6.7). In Figure 6.7, the sub-classes that are specific to the *Cyber-Warfare* scenario are displayed. This demonstrates which sub-classes are used when the Cyber-Warfare (CW) attack scenario is presented.

$$CW \subseteq AttackScenario \tag{6.23}$$

$$\begin{aligned} CW = \{x | (\exists v \in CyberArmy \ni (x,v) \in hasActor) \wedge \\ (\exists w \in DenialOfService\,AM \ni (x,w) \in hasAttackMechanismRampup) \wedge \\ (\exists y \in DenialOfService\,AM \ni (x,y) \in hasAttackMechanismDamage) \wedge \\ (\exists s \in NetworkInfrastructure \ni (x,s) \in hasTarget) \wedge \\ (\exists u \in (Government \cup Military \cup Corporate) \ni (x,u) \in hasScope)\} \end{aligned} \tag{6.24}$$

$$CyberArmyCW \subseteq CyberArmy \subseteq Actor \tag{6.25}$$

$$CyberArmyCW = \{x|(\exists z \in Disrupt \ni (x,z) \in hasAttackGoal) \wedge (\exists y \in State \ni (x,y) \in hasAggressor) \wedge (\exists v \in Foreign \ni (x,v) \in hasActorLocation)\} \quad (6.26)$$

$$NetworkInfrastructureCW \subseteq NetworkInfrastructure \subseteq Target \quad (6.27)$$

$$NetworkInfrastructureCW = \{x|\exists z \in Access \ni (x,z) \in hasAsset)\} \quad (6.28)$$

$$AccessCW \subseteq Access \subseteq Asset \quad (6.29)$$

$$Access = \{x|\exists y \in OperationalLoss \ni (x,y) \in hasSabotage\} \quad (6.30)$$

$$OperationalLossCW \subseteq OperationalLoss \subseteq Sabotage \quad (6.31)$$

$$Sabotage = \{(x|\exists y \in (Null \cup Minor) \ni (x,y) \in hasSabotageRampup) \wedge \exists z \in (Major \cup Catastrophic) \ni (x,z) \in hasSabotageDamage)\} \quad (6.32)$$

$$StateCW \subseteq State \subseteq Aggressor \quad (6.33)$$

$$StateCW = \{x|\exists z \in Political \ni (x,z) \in hasMotivation\} \quad (6.34)$$

$$ScopeCW \subseteq (Corporate \cup Military \cup Government) \subseteq Scope \quad (6.35)$$

$$ScopeCW = \{x|\exists z \in Large \ni (x,z) \in hasScopeSize\} \quad (6.36)$$

6.4.2 Cyber-Warfare Individual

The attack on Estonia (Section 2.4.24) was inferred as part of the *Cyber-Warfare* scenario. The Estonia attack individual was defined in Protégé as follows:

- is a member of the *Attack Scenario* class; and
- has at least one:
  - *Cyber Army* defined by *hasActor* relationship;
  - *Denial-of-Service Attack Mechanism* defined by *hasAttackMechanismDamage* relationship;
  - *Denial-of-Service Attack Mechanism* defined by *hasAttackMechanismRampup* relationship;
  - *Foreign Actor Location* defined by *hasChainActorActorLocation* relationship;
  - *State Aggressor* defined by *hasChainActorAggressor* relationship;

- *Political Motivation* defined by *hasChainActorAggressorMotivation* relationship;
- *Disrupt Attack Goal* defined by *hasChainActorAttackGoal* relationship;
- *Large Network* defined by *hasChainScopeScopeSize* relationship;
- *Access Asset* defined by *hasChainTargetAsset* relationship;
- *Operational Loss* defined by *hasChainTargetAssetSabotage* relationship;
- *Minor Effect* defined by *hasChainTargetAssetSabotageEffectRampup* relationship;
- *Catastrophic Effect* defined by *hasChainTargetAssetSabotageEffectDamage* relationship;
- *Corporate Network OR Government Network* defined by *hasScope* relationship;
- *Network Infrastructure* defined by *hasTarget* relationship.

By setting an individual using the properties as above, the automated reasoner HermiT plug-in for Protégé was able to determine that the Apache.org attacks fall within the *Cyber-Warfare* scenario. Protégé output is shown in Figure 6.9, with the automated-reasoner inferred class shown in yellow at the bottom.

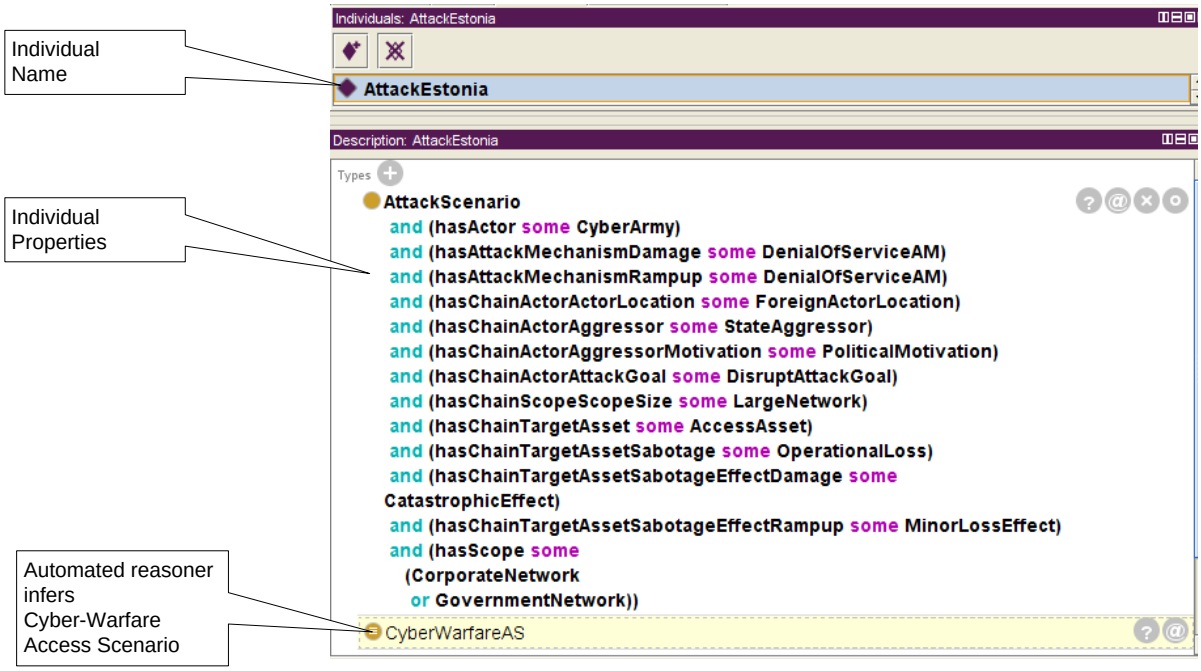


Figure 6.9: Estonia Attack Inferred a *Cyber-Warfare* Scenario

6.5 Industrial Espionage

Industrial Espionage refers to network attacks with the goal of stealing secret corporate information. This information can consist of various types of information such as defined by the FBI (1996). "...financial, business, scientific, technical, economic or engineering information, including patterns, plans, compilations, program devices, formulas, designs, prototypes, methods, techniques, processes, procedures, programs, or codes."
The story that describe the Industrial Espionage Scenario follows (Figure 6.10):
*A **HackerActor** based at **[ActorLocation]** location with the goal of **Stealing Data** sponsored by **State or Commercial Aggressor** with a **Espionage or Financial** motivation. The attack effected **[ScopeSize] Corporate** scope. A **FileServer Target** was attacked via **[Vulnerability]**. This attack effected **Data** and resulted in **Secret Loss** to **Null** effect during the Reconnaissance attack phase, to **Null OR Minor** effect during the Ramp-up attack phase and to **Minor OR Major** effect during the Damage attack phase. During the Reconnaissance phase **Open Information** was done, during the Ramp-up phase **Scanning** was used and during the Damage phase, **Exploit** was used. These mechanisms were automated to **[AutomationLevel]** level.*

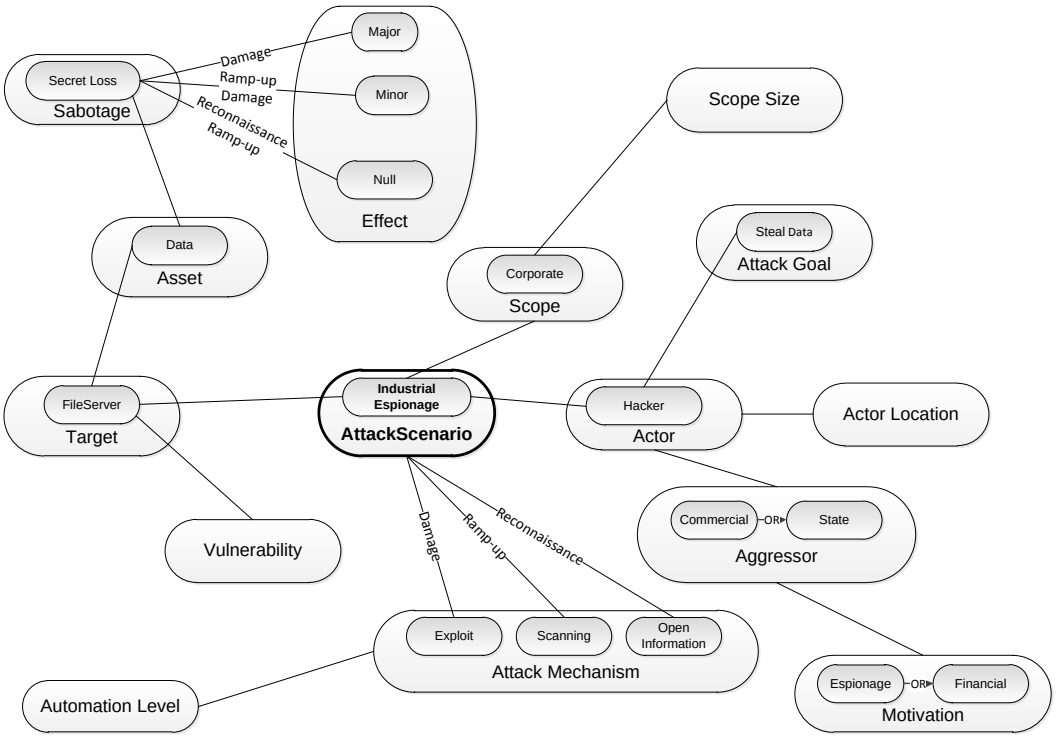


Figure 6.10: *Industrial Espionage* Attack Scenario

The Titan Rain (Section 2.4.14) network attack can be classified as an example of Industrial Espionage (Figure 6.11):
*A **HackerActor** based at **Foreign, China** location with the goal of **Stealing Data** sponsored by **State Aggressor** with a **Espionage** motivation The attack effected **Large Corporate** scope. A **FileServer Target** was attacked via **[Vulnerability]**. This attack effected **Data** and resulted in **Secret Loss** to **Null** effect during the Reconnaissance attack phase, to **Minor** effect during the Ramp-up attack phase and to **Major** effect during the Damage attack phase. During the Reconnaissance phase **Open Information** was done, during the Ramp-up phase **Scanning** was used and during the Damage phase, **Exploit** was used. These mechanisms was automated to **Manual** level.*

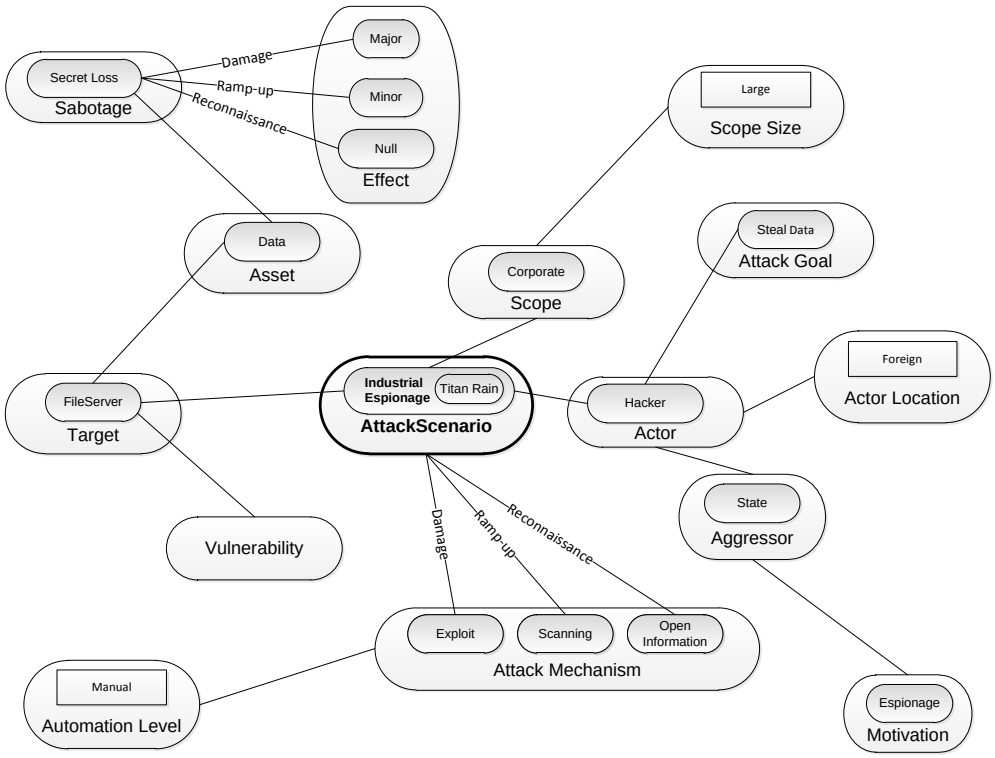


Figure 6.11: Titan Rain Industrial Espionage Attack Scenario Example

6.5.1 Industrial Espionage Formal Description

The *Industrial Espionage* scenario set is defined in statements 6.37 to 6.48 (also refer to Figure 6.10). In Figure 6.10, the sub-classes that are specific to the *Industrial Espionage* scenario are displayed. This demonstrates which sub-classes are used when the Industrial

Espionage (IE) attack scenario is presented.

$$IndustrialEspionage \subseteq AttackScenario \tag{6.37}$$

$$\begin{aligned} IndustrialEspionage = \{x | (\exists v \in Hacker \ni (x,v) \in hasActor) \wedge \\ (\exists w \in OpenInformation \ni (x,w) \in hasAttackMechanismRecon) \wedge \\ (\exists y \in Scanning \ni (x,y) \in hasAttackMechanismRampup) \wedge \\ (\exists s \in Exploit \ni (x,s) \in hasAttackMechanismDamage) \wedge \\ (\exists t \in Corporate \ni (x,t) \in hasScope) \wedge \\ (\exists u \in FileServer \ni (x,u) \in hasTarget)\} \end{aligned} \tag{6.38}$$

$$HackerIE \subseteq Hacker \subseteq Actor \tag{6.39}$$

$$\begin{aligned} HackerIE = \{x | (\exists z \in StealData \ni (x,z) \in hasAttackGoal) \wedge \\ (\exists y \in (Commercial \cup State) \ni (x,y) \in hasAttackGoal)\} \end{aligned} \tag{6.40}$$

$$FileServerIE \subseteq FileServer \subseteq Target \tag{6.41}$$

$$FileServerIE = \{x | \exists z \in Data \ni (x,z) \in hasAsset\} \tag{6.42}$$

$$DataIE \subseteq Data \subseteq Asset \tag{6.43}$$

$$DataIE = \{x | \exists y \in SecretLoss \ni (x,y) \in hasSabotage\} \tag{6.44}$$

$$SecretLossIE \subseteq SecretLoss \subseteq Sabotage \tag{6.45}$$

$$\begin{aligned} SecretLossIE = (\{x | (\exists y \in Null \ni (x,y) \in hasSabotageRecon) \wedge \\ (\exists v \in (Null \cup Minor) \ni (x,v) \in hasSabotageRampup) \wedge \\ (\exists z \in (Minor \cup Major) \ni (x,z) \in hasSabotageDamage)\} \end{aligned} \tag{6.46}$$

$$AggressorIE \subseteq (Commercial \cup State) \subseteq Aggressor \tag{6.47}$$

$$AggressorIE = \{x | \exists z \in (Espionage \cup Financial) \ni (x,z) \in hasMotivation\} \tag{6.48}$$

6.5.2 Industrial Espionage Individual

The Titan Rain (Section 2.4.14) was inferred as part of the *Industrial Espionage* scenario. The Titan Rain attack individual was defined in Protégé as follows:

- is a member of the *Attack Scenario* class; and

- has at least one:
  - *Hacker* defined by *hasActor* relationship;
  - *Exploit Attack Mechanism* defined by *hasAttackMechanismDamage* relationship;
  - *Scanning Attack Mechanism* defined by *hasAttackMechanismRampup* relationship;
  - *Open Information Mechanism* defined by *hasAttackMechanismRecon* relationship;
  - *Manual* defined by *hasChainAMAutomationLevel* relationship.
  - *Foreign Actor Location* defined by *hasChainActorActorLocation* relationship;
  - *State Aggressor* defined by *hasChainActorAggressor* relationship;
  - *Espionage Motivation* defined by *hasChainActorAggressorMotivation* relationship;
  - *Steal Data Attack Goal* defined by *hasChainActorAttackGoal* relationship;
  - *Large Network* defined by *hasChainScopeScopeSize* relationship;
  - *Data Asset* defined by *hasChainTargetAsset* relationship;
  - *Secret Loss* defined by *hasChainTargetAssetSabotage* relationship;
  - *Null Effect* defined by *hasChainTargetAssetSabotageEffectRecon* relationship;
  - *Minor Effect* defined by *hasChainTargetAssetSabotageEffectRamp* relationship;
  - *Major Effect* defined by *hasChainTargetAssetSabotageEffectDamage* relationship;
  - *Corporate Scope* defined by *hasScope* relationship;
  - *File Server* defined by *hasTarget* relationship.

By setting an individual using the properties as above, the automated reasoner HermiT plug-in for Protégé was able to determine that the Titan Rain attacks fall within the *Industrial Espionage* scenario. Protégé output is shown in Figure 6.12, with the automated reasoner-inferred class shown at the yellow at the bottom.

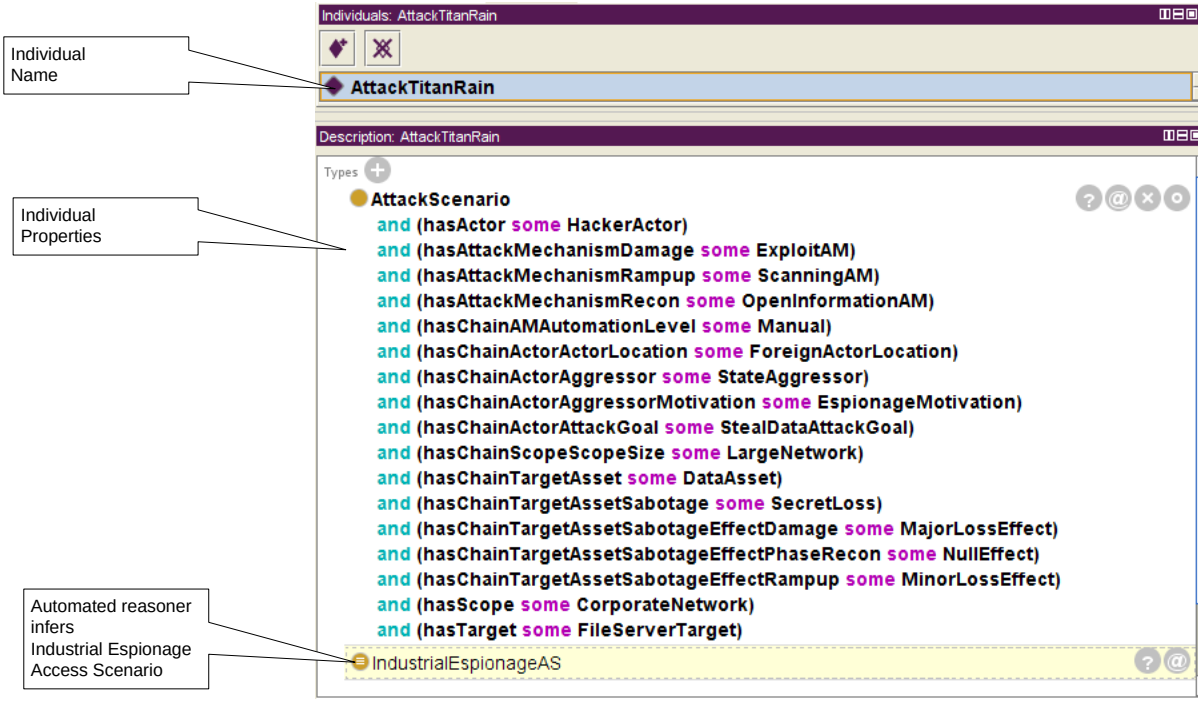


Figure 6.12: Titan Rain Attack Inferred an *Industrial Espionage* Scenario

6.6 Financial Theft

The *Financial Theft* scenario refers to the use of computer networks to steal money, or hacking directly for money. A famous bank robber, Willie Sutton, is credited with the following answer to why he robbed banks (Yoder, 1951):
"I rob banks because that's where the money is."
The same motivation holds true in the information age, where computer networks are used as the means to rob banks or other institutions for financial gain.

The story that describes the Financial Theft Scenario follows (Figure 6.13):
*An **Organised Criminal Group** based at **[ActorLocation]** location with the goal of **Changing Data** sponsored by **[Aggressor]** with a **Financial** motivation. The attack effected **[ScopeSize] Corporate** scope. A **ServerTarget** was attacked via **[Vulnerability]**. This attack effected **Data** and resulted in **Financial Loss** to **Null** effect during the Reconnaissance attack phase, to **Null OR Minor** effect during the Ramp-up attack phase and to **Minor OR Major** effect during the Damage attack phase. During the Reconnaissance phase **Open Information** was used, during the Ramp-up phase **Scanning** was used and during the Damage phase, **Exploit** was used. These mechanisms was*

automated to ***[AutomationLevel]*** *level.*

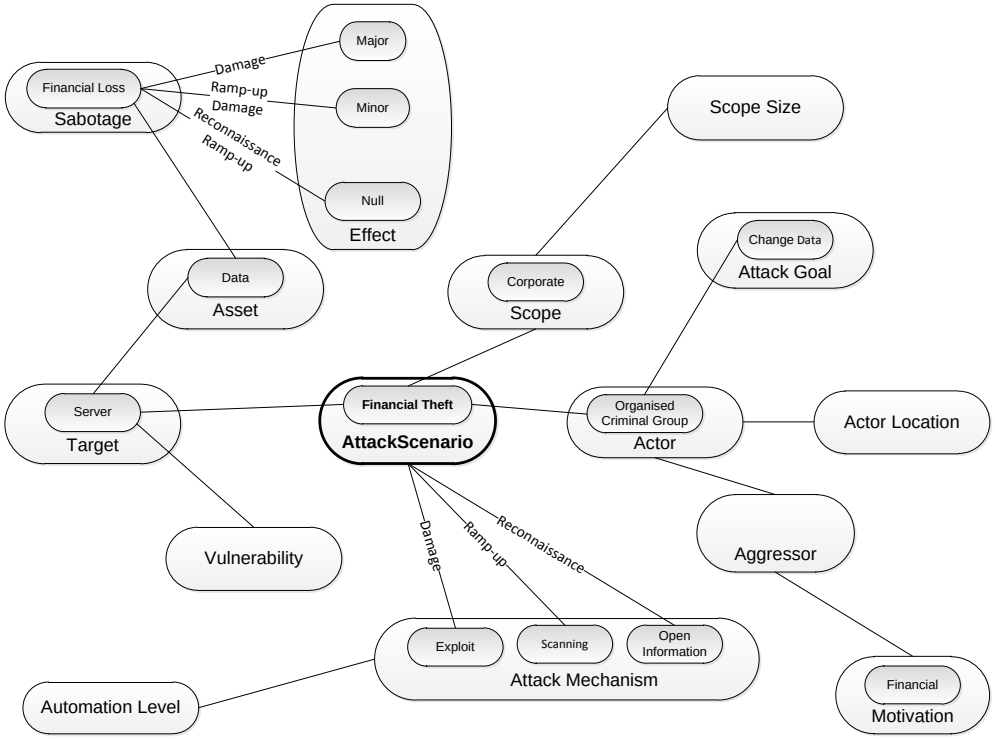


Figure 6.13: Financial Theft Attack Scenario

The PostBank (Section 2.4.33) network attack can be classified as an example of a Financial Theft Scenario (Figure 6.14):
An ***Organised Criminal Group*** *based at* ***Local*** *location with the goal of* ***Changing Data*** *sponsored by* ***Self Instigator*** *with a* ***Financial*** *motivation. The attack effected* ***Large Corporate*** *scope. A* ***ServerTarget*** *was attacked via* ***Access Rights Vulnerability****. This attack effected* ***Data*** *and resulted in* ***Financial Loss*** *and resulted in* ***Secret Loss*** *to* ***Null*** *effect during the Reconnaissance attack phase, to* ***Null*** *effect during the Ramp-up attack phase and to* ***Major*** *effect during the Damage attack phase. During the Reconnaissance phase* ***Open Information*** *was used, during the Ramp-up phase* ***Scanning*** *was used and during the Damage phase,* ***Physical Access Exploit*** *was used. These mechanisms was automated to* ***Manual*** *level.*

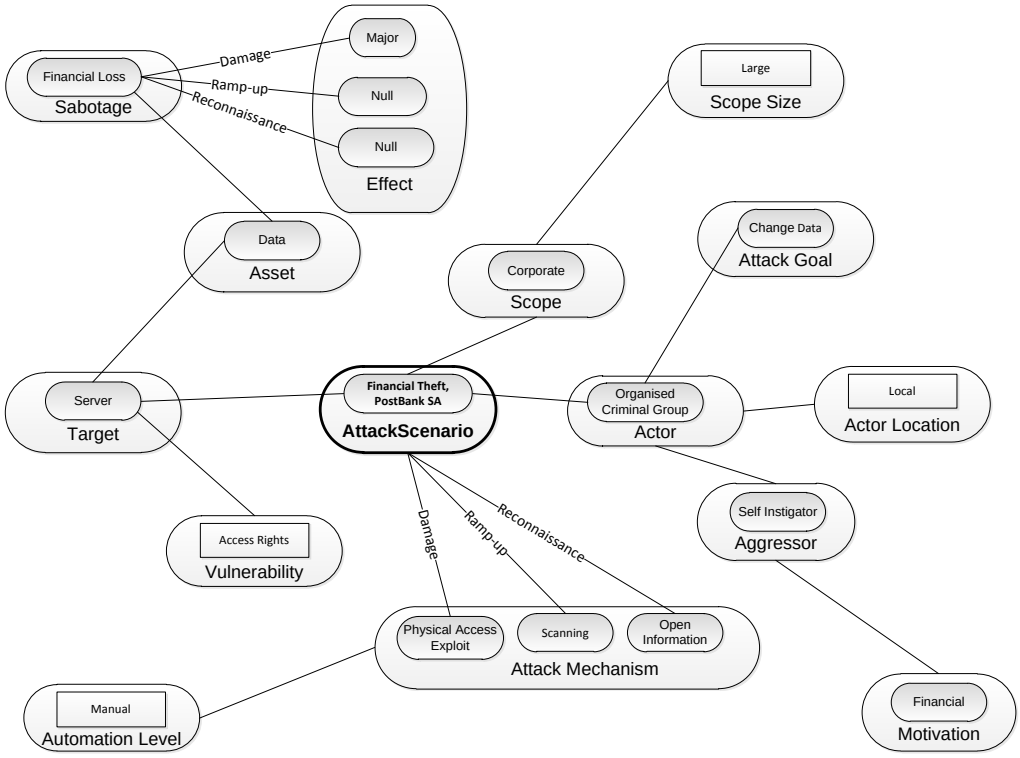


Figure 6.14: Post Bank SA Financial Theft Attack Scenario Example

6.6.1 Financial Theft Formal Description

The *Financial Theft* scenario set is defined in statements 6.49 to 6.60 (also refer to Figure 6.13). In Figure 6.13, the sub-classes that are specific to the *Financial Theft* scenario are displayed. This demonstrates which sub-classes are used when the Financial Theft (FT) attack scenario is presented.

$$FinancialTheft \subseteq AttackScenario \tag{6.49}$$

$$\begin{aligned} FinancialTheft = \{x | (\exists v \in OrganisedCriminalGroup \ni (x,v) \in hasActor) \wedge \\ (\exists w \in OpenInformation \ni (x,w) \in hasAttackMechanismRecon) \wedge \\ (\exists y \in Scanning \ni (x,y) \in hasAttackMechanismRampup) \wedge \\ (\exists s \in Exploit \ni (x,s) \in hasAttackMechanismDamage) \wedge \\ (\exists t \in Corporate \ni (x,t) \in hasScope) \wedge \\ (\exists u \in Server \ni (x,u) \in hasTarget)\} \end{aligned} \tag{6.50}$$

$$OrganisedCriminalGroupFT \subseteq OrganisedCriminalGroup \subseteq Actor \tag{6.51}$$

$$OrganisedCriminalGroupFT = \{x | \exists z \in ChangeData \ni (x, z) \in hasAttackGoal\} \tag{6.52}$$

$$ServerFT \subseteq Server \subseteq Target \tag{6.53}$$

$$ServerFT = \{x | \exists z \in Data \ni (x, z) \in hasAsset)\} \tag{6.54}$$

$$DataFT \subseteq Data \subseteq Asset \tag{6.55}$$

$$DataFT = \{x | \exists y \in FinancialLoss \ni (x, y) \in hasSabotage\} \tag{6.56}$$

$$FinancialLossFT \subseteq FinancialLoss \subseteq Sabotage \tag{6.57}$$

$$FinancialLossFT = (\{x | (\exists y \in Null \ni (x, y) \in hasSabotageRecon) \wedge (\exists v \in (Null \cup Minor) \ni (x, v) \in hasSabotageRampup) \wedge (\exists z \in (Minor \cup Major) \ni (x, z) \in hasSabotageDamage)\} \tag{6.58}$$

$$AggressorFT \subseteq Aggressor \tag{6.59}$$

$$AggressorFT = \{x | \exists z \in Financial \ni (x, z) \in hasMotivation\} \tag{6.60}$$

6.6.2 Financial Theft Individual

The PostBank SA (Section 2.4.33) was inferred as part of the *Financial Theft* scenario. The PostBank SA attack individual was defined in Protégé as follows:

- is a member of the *Attack Scenario* class; and
- has at least one:
  - *Organised Criminal Group* defined by *hasActor* relationship;
  - *Physical Exploit Attack Mechanism* defined by *hasAttackMechanismDamage* relationship;
  - *Scanning Attack Mechanism* defined by *hasAttackMechanismRampup* relationship;
  - *Open Information Mechanism* defined by *hasAttackMechanismRecon* relationship;
  - *Local Actor Location* defined by *hasChainActorActorLocation* relationship;
  - *Self Instigator Aggressor* defined by *hasChainActorAggressor* relationship;

- *Financial Motivation* defined by *hasChainActorAggressorMotivation* relationship;
- *Change Data Attack Goal* defined by *hasChainActorAttackGoal* relationship;
- *Large Network* defined by *hasChainScopeScopeSize* relationship;
- *Data Asset* defined by *hasChainTargetAsset* relationship;
- *Financial Loss* defined by *hasChainTargetAssetSabotage* relationship;
- *Null Effect* defined by *hasChainTargetAssetSabotageEffectRecon* relationship;
- *Null Effect* defined by *hasChainTargetAssetSabotageEffectRampup* relationship;
- *Major Effect* defined by *hasChainTargetAssetSabotageEffectDamage* relationship;
- *Manual* defined by *hasChainAMAutomationLevel* relationship.
- *Corporate Scope* defined by *hasScope* relationship;
- *Server* defined by *hasTarget* relationship.

By setting an individual using the properties as above, the automated reasoner HermiT plug-in for Protégé was able to determine that the PostBank SA attacks fall within the *Financial Theft* scenario. Protégé output is shown in Figure 6.15, with the automated reasoner-inferred class shown in yellow at the bottom.

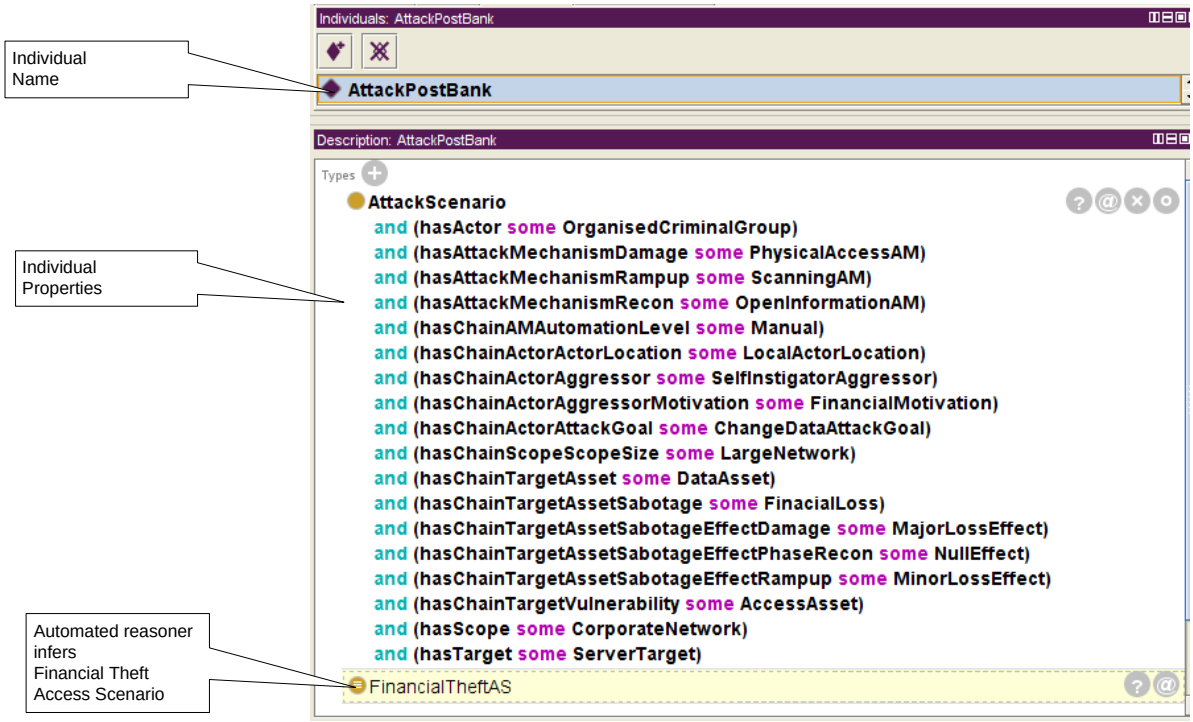


Figure 6.15: PostBank SA Attack Inferred a *Financial Theft* Scenario

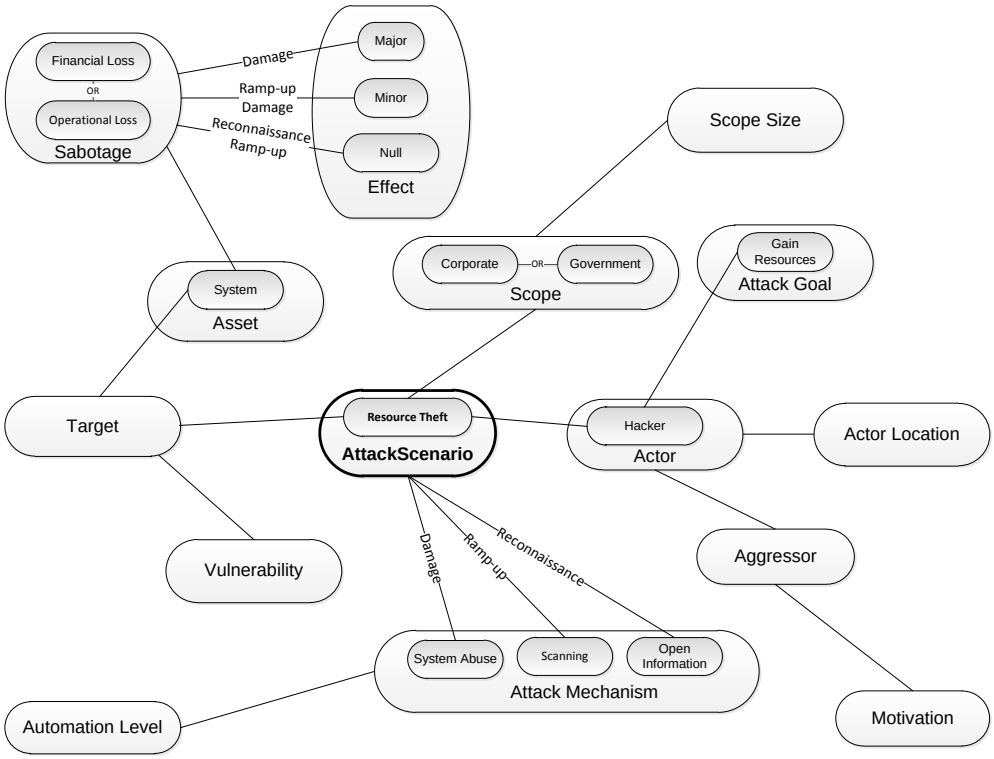


Figure 6.16: Resource Theft Attack Scenario

6.7 Resource Theft

The *Resource Theft* scenario refers to gaining control of computer resources. The computer resources can typically be used to attack other computers via DDoS. The main resources that are targeted are bandwidth, processing power or memory.

The story that describes the Resource Theft Scenario follows (Figure 6.16):
*A **Hacker** based at **[ActorLocation]** with the goal of **Gain Resources** is sponsored by **[Aggressor]** with a **[Motivation]** motivation The attack effected **[ScopeSize] Government OR Corporate network** scope. A **[Target]** was attacked via **[Vulnerability]**. This attack effected **System** and resulted in **Operational Loss OR Financial Loss** to **Minor OR Major** effect during Damage attack phase, to **Minor OR Null** effect during Ramp-up attack phase, and to **Null** effect during Reconnaissance attack phase. During the Reconnaissance phase **Open Information** was used, during the Ramp-up phase **Scanning** was used and during the Damage phase, **System Abuse** was used. These mechanisms were automated to **[AutomationLevel]** level.*

The Phone Phreaking (Section2.4.1) incident can be classified as an example of a Resource Theft scenario (Figure 6.17):

*A **Hacker** based at **Local** with the goal of **Gain Resources** is sponsored by **Self Instigator** with a **Financial** motivation The attack effected **Large Corporate network** scope. A **Industrial Equipment** was attacked via **Access Rights**. This attack effected **System** and resulted in **Financial Loss** to **Minor** effect during **Damage** attack phase. During the Reconnaissance phase **Open Information** was used, during the Ramp-up phase **Vulnerability Scanning** was used and during the Damage phase, **System Abuse** was used. These mechanisms was automated to **Manual** level.*

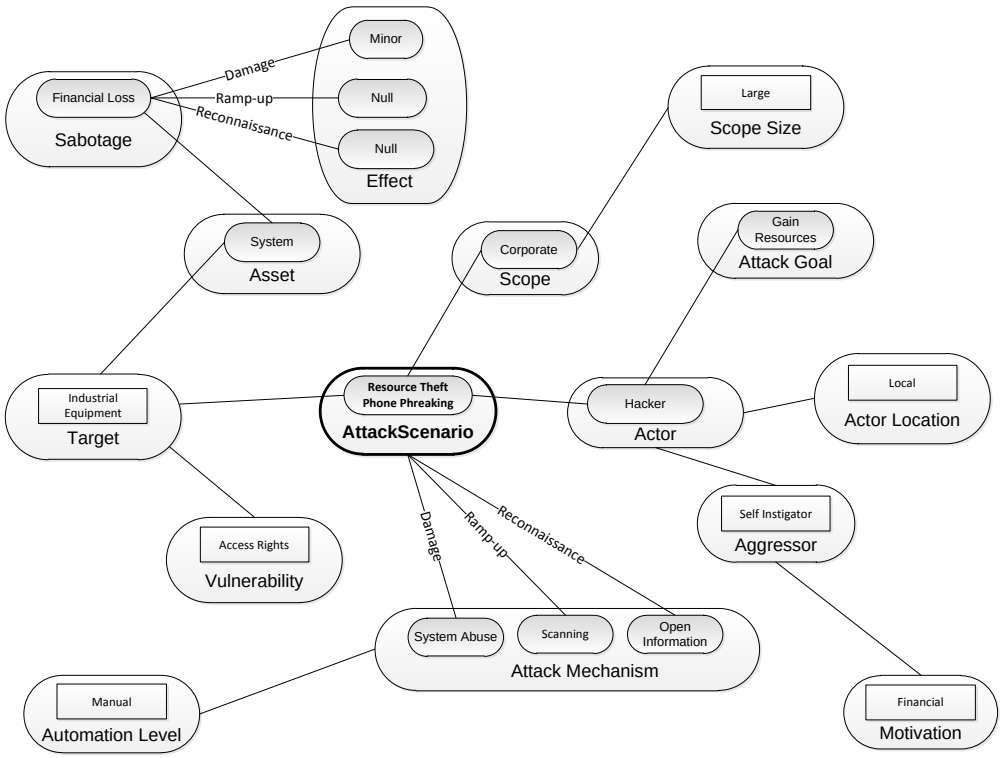


Figure 6.17: Phone Phreaking Resource Theft Attack Scenario Example

6.7.1 Resource Theft Formal Description

The *Resource Theft* scenario set is defined in statements 6.61 to 6.69 (also refer to Figure 6.16). In Figure 6.16, the sub-classes that are specific to the *Resource Theft* scenario are displayed. This demonstrates which sub-classes are used when the Resource Theft attack scenario is presented.

$$ResourceTheft \subseteq AttackScenario \tag{6.61}$$

$$\begin{aligned} ResourceTheft = \{x | (\exists v \in Hacker \ni (x,v) \in hasActor) \wedge \\ (\exists w \in OpenInformation \ni (x,w) \in hasAttackMechanismRecon) \wedge \\ (\exists y \in Scanning \ni (x,y) \in hasAttackMechanismRampup) \wedge \\ (\exists s \in SystemAbuse \ni (x,s) \in hasAttackMechanismDamage) \wedge \\ (\exists t \in (Corporate \cup Government) \ni (x,t) \in hasScope)\} \end{aligned} \tag{6.62}$$

$$HackerResourceTheft \subseteq Hacker \subseteq Actor \tag{6.63}$$

$$HackerResourceTheft = \{x | \exists z \in GainResources \ni (x,z) \in hasAttackGoal\} \tag{6.64}$$

$$Target = \{x | \exists z \in System \ni (x,z) \in hasAsset\} \tag{6.65}$$

$$SystemResourceTheft \subseteq System \subseteq Asset \tag{6.66}$$

$$\begin{aligned} SystemResourceTheft = \\ \{x | \exists y \in (FinancialLoss \cup OperationalLoss) \ni (x,y) \in hasSabotage\} \end{aligned} \tag{6.67}$$

$$Sabotage\ ResourceTheft \subseteq (FinancialLoss \cup OperationalLoss) \subseteq Sabotage \tag{6.68}$$

$$\begin{aligned} Sabotage\ ResourceTheft = \{x | (\exists y \in Null \ni (x,y) \in hasSabotageRecon) \wedge \\ (\exists v \in (Null \cup Minor) \ni (x,v) \in hasSabotageRampup) \wedge \\ (\exists z \in (Minor \cup Major) \ni (x,z) \in hasSabotageDamage)\} \end{aligned} \tag{6.69}$$

6.7.2 Resource Theft Individual

The Phone Phreaking (Section 2.4.1) was inferred as part of the *Resource Theft* scenario. The Phone Phreaking individual was defined in Protégé as follows:

- is a member of the *Attack Scenario* class; and
- has at least one:
  - *Hacker Actor* defined by *hasActor* relationship;
  - *System Abuse Attack Mechanism* defined by *hasAttackMechanismDamage* relationship;
  - *Scanning Attack Mechanism* defined by *hasAttackMechanismRampup* relationship;
  - *Open Information Mechanism* defined by *hasAttackMechanismRecon* relationship;
  - *Manual* defined by *hasChainAMAutomationLevel* relationship.
  - *Local Actor Location* defined by *hasChainActorActorLocation* relationship;

- *Self Instigator Aggressor* defined by *hasChainActorAggressor* relationship;
- *Financial Motivation* defined by *hasChainActorAggressorMotivation* relationship;
- *Gain Resources Attack Goal* defined by *hasChainActorAttackGoal* relationship;
- *Large Network* defined by *hasChainScopeScopeSize* relationship;
- *System Asset* defined by *hasChainTargetAsset* relationship;
- *Financial Loss* defined by *hasChainTargetAssetSabotage* relationship;
- *Null Effect* defined by *hasChainTargetAssetSabotageEffectRecon* relationship;
- *Null Effect* defined by *hasChainTargetAssetSabotageEffectRampup* relationship;
- *Minor Effect* defined by *hasChainTargetAssetSabotageEffectDamage* relationship;
- *Access Rights Vulnerability* defined by *hasChainTargetVulnerability* relationship;
- *Corporate Scope* defined by *hasScope* relationship;
- *Industrial Equipment* defined by *hasTarget* relationship.

By setting an individual with properties as above, the automated reasoner HermiT plug-in for Protégé was able to determine that the Phone Phreaking attacks fall within the *Resource Theft* scenario. Protégé output is shown in Figure 6.18, with the automated reasoner-inferred class shown in yellow at the bottom.

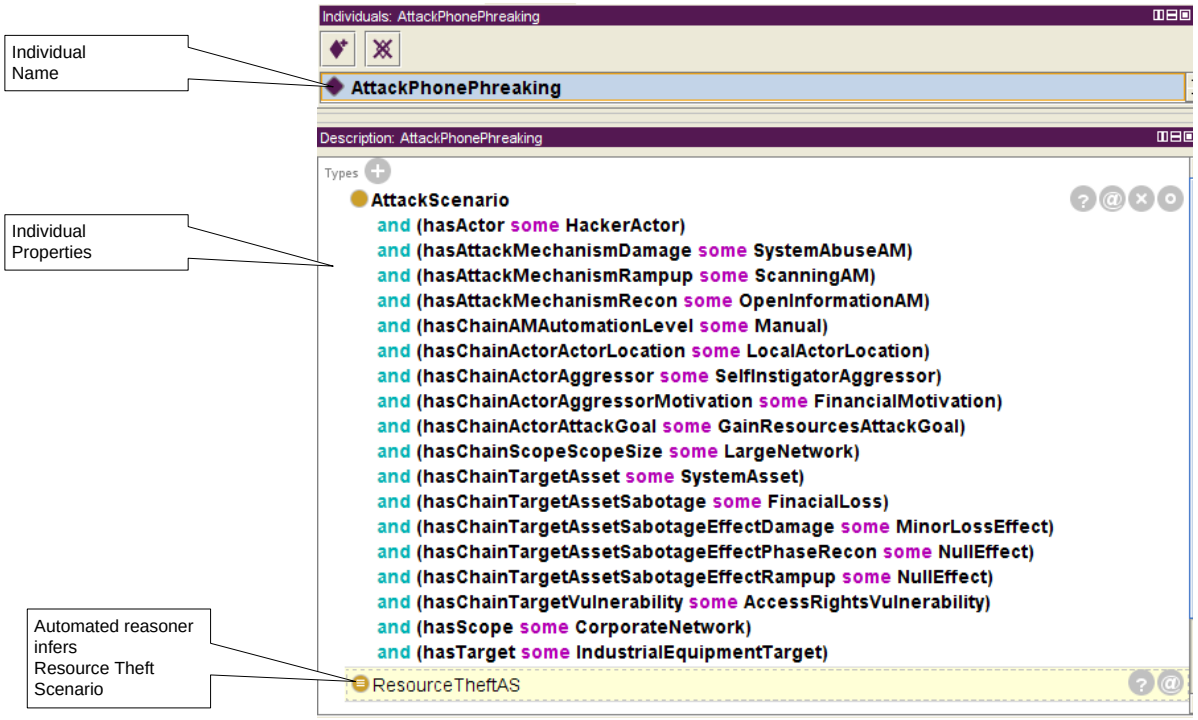


Figure 6.18: Phone Phreaking Attack Inferred a *Resource Theft* Scenario

6.8 Industrial Sabotage

The *Industrial Sabotage* scenario refers to using computer networks to damage an industrial complex or product.

The story that describes the Industrial Sabotage Scenario follows (Figure 6.19):
*A **Hacker** based at **[ActorLocation]** with the goal of **Disrupt** is sponsored by **Commercial OR State** with a **Criminal** motivation. The attack effected **[ScopeSize] [Scope]** scope. A **Industrial Equipment** was attacked via **[Vulnerability]**. This attack effected **System** and resulted in **Operational Loss** to **Catastrophic OR Major** effect during Damage attack phase, to **Null OR Minor** effect during Ramp-up attack phase, and to **Null** effect during Reconnaissance attack phase. During the Reconnaissance phase **Information Gathering** was used, during the Ramp-up phase **Scanning** was used and during the Damage phase, **Exploit** was used and was automated to **[AutomationLevel]** level.*

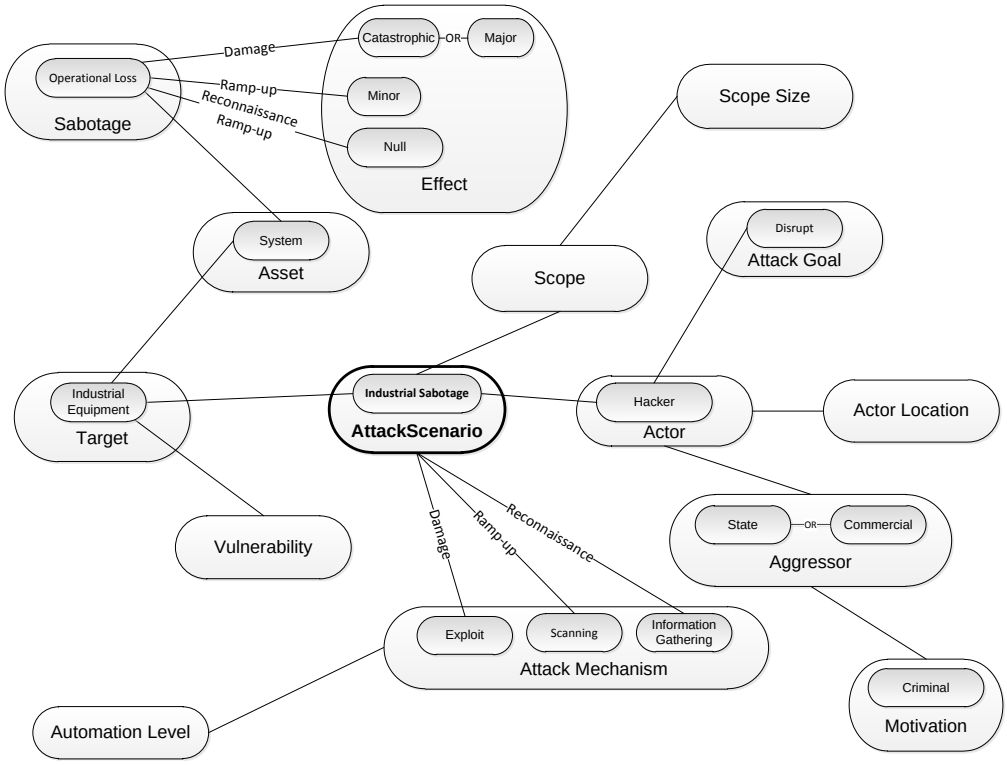


Figure 6.19: Industrial Sabotage Attack Scenario

The Stuxnet worm (Section 2.4.29) can be classified as an example of an Industrial Sabotage scenario (Figure 6.20):

*A **Hacker** based at **Foreign** with the goal of **Disrupt** is sponsored by **State** with a **Criminal** motivation. The attack effected **Large Government** scope. A **Industrial Equipment** was attacked via **Configuration** vulnerability. This attack effected **System** and resulted in **Operational Loss** to **Major Loss** effect during **Damage** attack phase, to **Null** effect during **Ramp-up** attack phase, and to **Null** effect during **Reconnaissance** attack phase. During the Reconnaissance phase Information Gathering was used, during the Ramp-up phase Scanning was used and during the Damage phase, Exploit was used. was automated to **Automatic** level.*

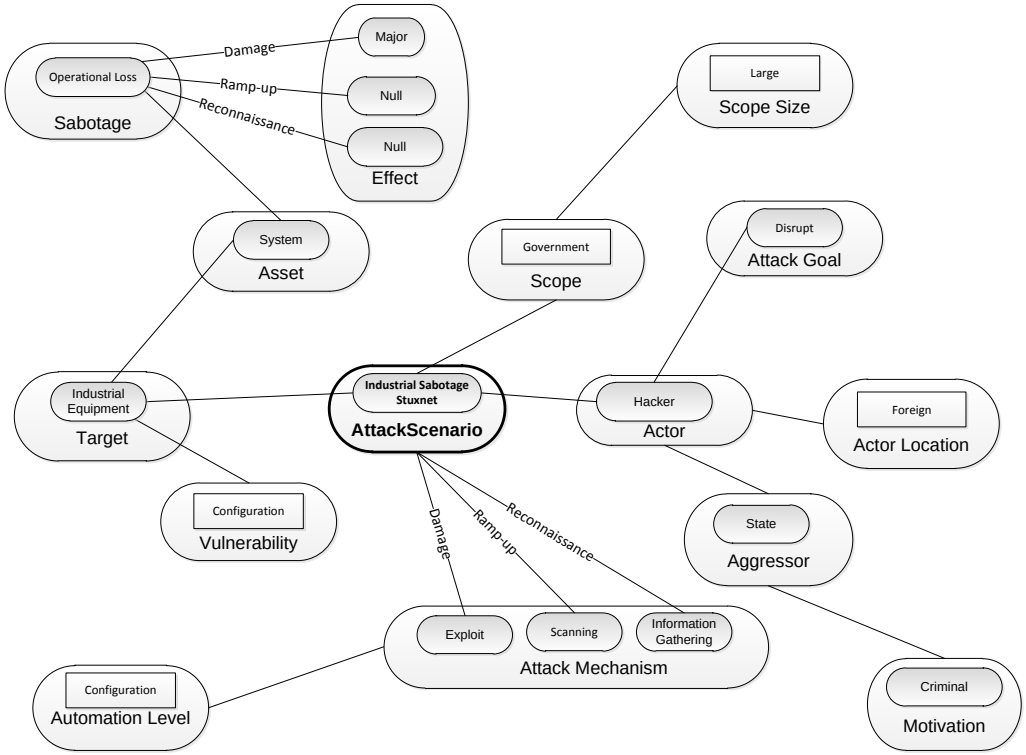


Figure 6.20: Stuxnet Industrial Sabotage Attack Scenario Example

6.8.1 Industrial Sabotage Formal Description

The *Resource Theft* scenario set is defined in statements 6.70 to 6.81 (also refer to Figure 6.19). In Figure 6.19, the sub-classes that are specific to the *Industrial Sabotage* scenario are displayed. This demonstrates which sub-classes are used when the Industrial Sabotage (IS) attack scenario is presented.

$$IndustrialSabotage \subseteq AttackScenario \tag{6.70}$$

$$IndustrialSabotage = \{x|(\exists v \in Hacker \ni (x,v) \in hasActor) \wedge (\exists w \in InformationGathering \ni (x,w) \in hasAttackMechanismRecon) \wedge (\exists y \in Scanning \ni (x,y) \in hasAttackMechanismRampup) \wedge (\exists s \in Exploit \ni (x,s) \in hasAttackMechanismDamage) \wedge (\exists t \in IndustrialEquipment) \ni (x,t) \in hasTarget)\} \quad (6.71)$$

$$HackerIS \subseteq Hacker \subseteq Actor \quad (6.72)$$

$$HackerIS = \{x|(\exists z \in Disrupt \ni (x,z) \in hasAttackGoal) \wedge (\exists y \in (State \cup Commercial) \ni (x,y) \in hasAggressor)\} \quad (6.73)$$

$$IndustrialEquipmentIS \subseteq IndustrialEquipment \subseteq Target \quad (6.74)$$

$$IndustrialEquipmentIS = \{x|\exists z \in System \ni (x,z) \in hasAsset\} \quad (6.75)$$

$$SystemIS \subseteq System \subseteq Asset \quad (6.76)$$

$$SystemIS = \{x|\exists y \in OperationalLoss \ni (x,y) \in hasSabotage\} \quad (6.77)$$

$$OperationalLossIS \subseteq OperationalLoss \subseteq Sabotage \quad (6.78)$$

$$OperationalLossIS = \{x|(\exists y \in Null \ni (x,y) \in hasSabotageRecon) \wedge (\exists v \in (Null \cup Minor) \ni (x,v) \in hasSabotageRampup) \wedge (\exists z \in (Catastrophic \cup Major) \ni (x,z) \in hasSabotageDamage)\} \quad (6.79)$$

$$AggressorIS \subseteq (State \cup Commercial) \subseteq Aggressor \quad (6.80)$$

$$AggressorIS = \{x|\exists z \in Criminal \ni (x,z) \in hasMotivation\} \quad (6.81)$$

6.8.2 Industrial Sabotage Individual

The Stuxnet (Section 2.4.29) was inferred as part of the *Industrial Sabotage* scenario. The Stuxnet individual was defined in Protégé as follows:

- is a member of the *Attack Scenario* class; and
- has at least one:
  - *Hacker Actor* defined by *hasActor* relationship;
  - *Exploit Attack Mechanism* defined by *hasAttackMechanismDamage* relationship;
  - *Vulnerability Scanning Attack Mechanism* defined by *hasAttackMechanismRampup* relationship;
  - *Information Gathering Mechanism* defined by *hasAttackMechanismRecon* relationship;
  - *Automatic* defined by *hasChainAMAutomationLevel* relationship.
  - *Foreign Actor Location* defined by *hasChainActorActorLocation* relationship;
  - *State Aggressor* defined by *hasChainActorAggressor* relationship;
  - *Political Motivation* defined by *hasChainActorAggressorMotivation* relationship;
  - *Disrupt Attack Goal* defined by *hasChainActorAttackGoal* relationship;
  - *Large Network* defined by *hasChainScopeScopeSize* relationship;
  - *System Asset* defined by *hasChainTargetAsset* relationship;
  - *Operational Loss* defined by *hasChainTargetAssetSabotage* relationship;
  - *Null Effect* defined by *hasChainTargetAssetSabotageEffectRecon* relationship;
  - *Null Effect* defined by *hasChainTargetAssetSabotageEffectRampup* relationship;
  - *Major Effect* defined by *hasChainTargetAssetSabotageEffectDamage* relationship;
  - *Config Vulnerability* defined by *hasChainTargetVulnerability* relationship;
  - *Government Scope* defined by *hasScope* relationship;
  - *Industrial Equipment* defined by *hasTarget* relationship.

By setting an individual with properties as above, the automated reasoner HermiT plug-in for Protégé was able to determine that the Stuxnet worm falls within the *Industrial Sabotage* scenario. Protégé output is shown in Figure 6.21, with the automated reasoner-inferred class shown in yellow at the bottom.

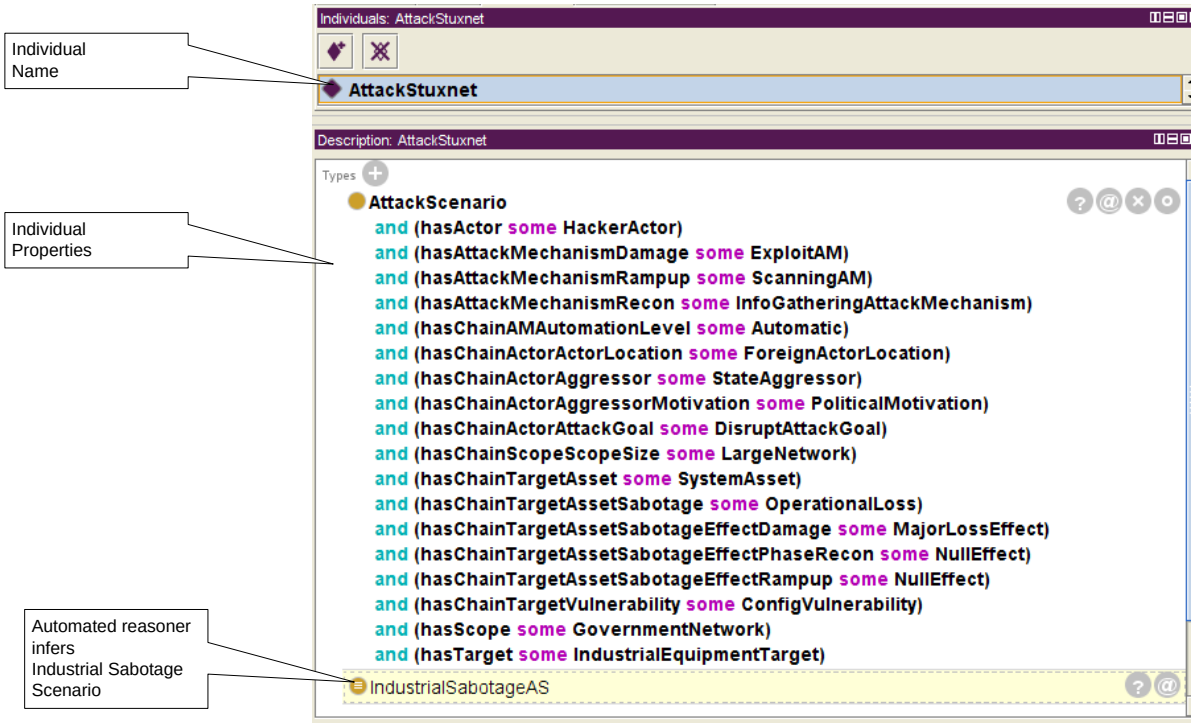


Figure 6.21: Stuxnet Attack Inferred an *Industrial Sabotage* Scenario

6.9 Runaway Malware

Runaway Malware is malicious software that has the ability to rapidly spread within a computer network. Such malware often comes in the form of viruses or computer worms, and has the potential ability to exploit weaknesses in a computer system that will allow the malware to spread to uninfected machines. The story that describes the Runaway Malware Scenario follows (Figure 6.22):
*A **Hacker** based at **[ActorLocation]** with the goal of **Spreading** is sponsored by **Self Instigator** with a **Fun** motivation. The attack effected **Global All networks** scope. A **PC OR Server** was attacked via **[Vulnerability]**. This attack effected **Network OR System** and resulted in **Operational Loss** to **Minor OR Null** effect during Ramp-up attack phase, and to **Minor OR Major** effect during Damage attack phase. During the Ramp-up phase **Malware** was used and during the Damage phase, **Denial-of-Service** was used, and was automated to **Automatic** level.*

The I-LOVE-YOU worm (Section 2.4.12) can be classified as an example of a Runaway Malware scenario (Figure 6.23):

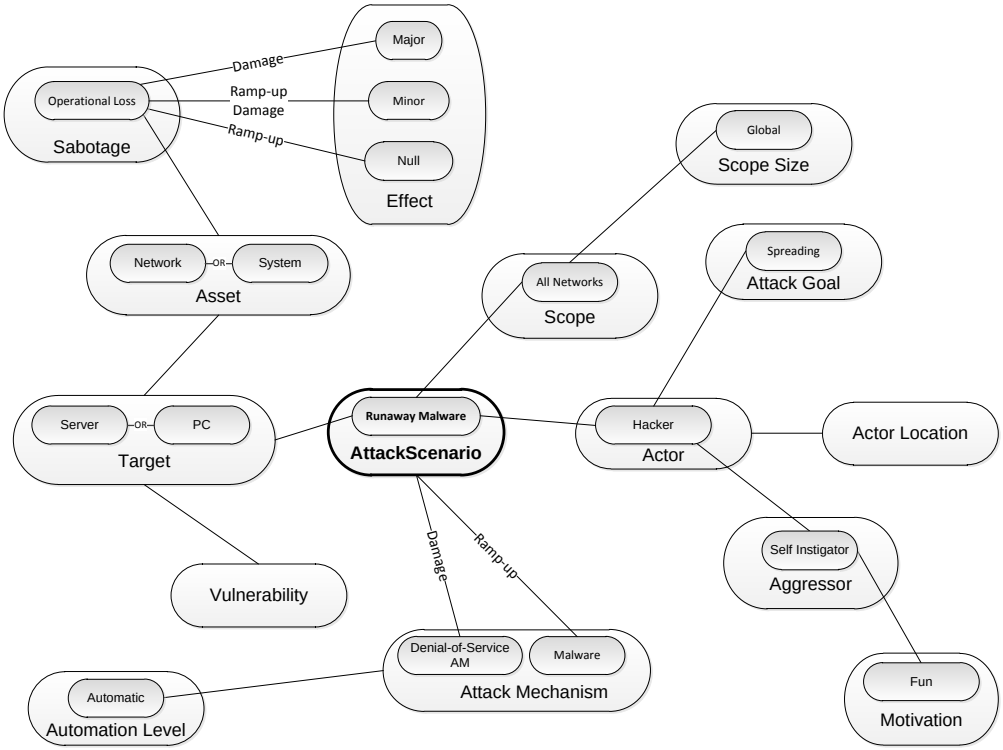


Figure 6.22: Runaway Malware Attack Scenario

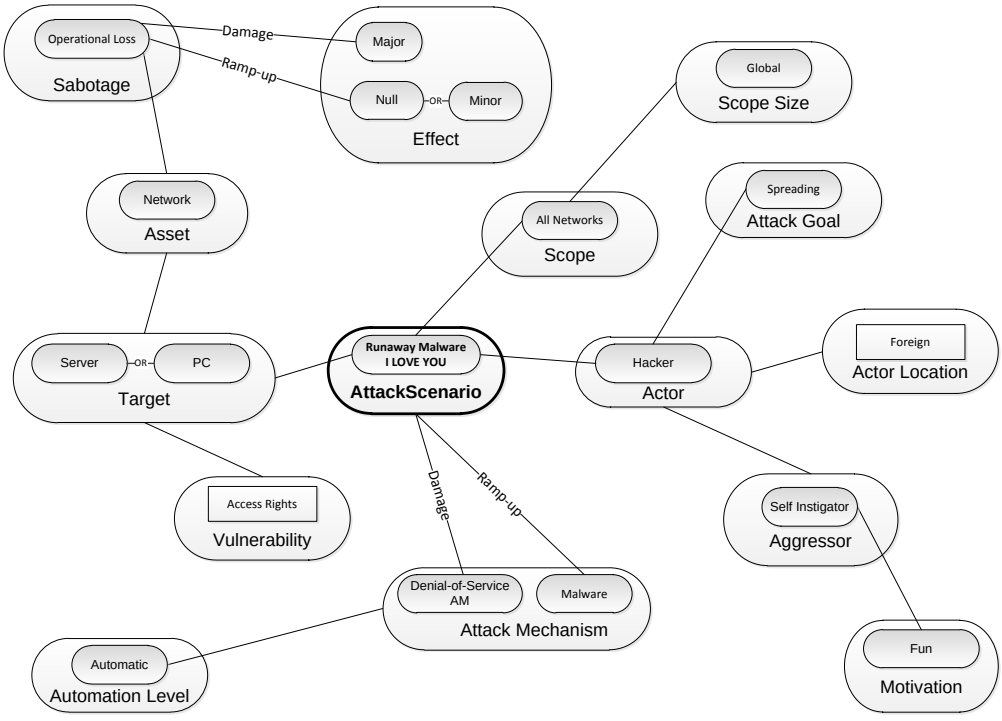


Figure 6.23: I LOVE YOU Worm Runaway Malware Attack Scenario Example

*A **Hacker** based at **Foreign** with the goal of **Spreading** is sponsored by **Self Instigator** with a **Fun** motivation. The attack effected **Global All networks** scope. A **PC OR Server** was attacked via **Access Rights Vulnerability**. This attack effected **Network** and resulted in **Operational Loss** to **Major** effect during Damage attack phase, to **Minor** effect during Ramp-up attack phase. During the Ramp-up phase **Malware** was used and during the Damage phase, **Denial-of-Service** was used, and was automated to **Automatic** level.*

6.9.1 Runaway Malware Formal Description

The *Runaway Malware* scenario set is defined in statements 6.82 to 6.97 (also refer to Figure 6.22). In Figure 6.22, the sub-classes that are specific to the *Runaway Malware* scenario are displayed. This demonstrates which sub-classes are used when the Runaway Malware attack scenario is presented.

$$RunawayMalware \subseteq AttackScenario \tag{6.82}$$

$$\begin{aligned} RunawayMalware = \{x | (\exists v \in Hacker \ni (x,v) \in hasActor) \wedge \\ (\exists y \in Malware \ni (x,y) \in hasAttackMechanismRampup) \wedge \\ (\exists s \in DenialOfService \ni (x,s) \in hasAttackMechanismDamage) \wedge \\ (\exists w \in AllNetworks \ni (x,w) \in hasScope) \wedge \\ (\exists t \in (Server \cup PC) \ni (x,t) \in hasTarget)\} \end{aligned} \tag{6.83}$$

$$HackerRunawayMalware \subseteq Hacker \subseteq Actor \tag{6.84}$$

$$\begin{aligned} HackerRunawayMalware = \{x | (\exists s \in SelfInstigator \ni (x,s) \in hasAggressor) \wedge \\ (\exists z \in Spreading \ni (x,z) \in hasAttackGoal)\} \end{aligned} \tag{6.85}$$

$$TargetRunawayMalware \subseteq (Server \cup PC) \subseteq Target \tag{6.86}$$

$$TargetRunawayMalware = \{x | \exists z \in (Network \cup System) \ni (x,z) \in hasAsset\} \tag{6.87}$$

$$AssetRunawayMalware \subseteq (Network \cup System) \subseteq Asset \tag{6.88}$$

$$AssetRunawayMalware = \{x | \exists y \in OperationalLoss \ni (x,y) \in hasSabotage\} \tag{6.89}$$

$$OperationalLossRunawayMalware \subseteq OperationalLoss \subseteq Sabotage \tag{6.90}$$

$$\begin{aligned} &OperationalLossRunawayMalware = \\ &\{x|(\exists y \in (Null \cup Minor) \ni (x,y) \in hasSabotageRampup) \wedge \\ &(\exists z \in (Major \cup Minor) \ni (x,z) \in hasSabotageDamage)\} \end{aligned} \tag{6.91}$$

$$SelfInstigatorRunawayMalware \subseteq SelfInstigator \subseteq Aggressor \tag{6.92}$$

$$SelfInstigatorRunawayMalware = \{x|\exists z \in Fun \ni (x,z) \in hasMotivation\} \tag{6.93}$$

$$AllNetworksRunawayMalware \subseteq AllNetworks \subseteq Scope \tag{6.94}$$

$$AllNetworksRunawayMalware = \{x|\exists z \in Global \ni (x,z) \in hasScopeSize\} \tag{6.95}$$

$$\begin{aligned} &AttackMechanismRunawayMalware \subseteq \\ &(DenialofServiceAM \cup Malware) \subseteq AttackMechanism \end{aligned} \tag{6.96}$$

$$\begin{aligned} &AttackMechanismRunawayMalware = \\ &\{x|\exists z \in Automatic \ni (x,z) \in hasAutomation\} \end{aligned} \tag{6.97}$$

6.9.2 Runaway Malware Individual

The I LOVE YOU (Section 2.4.12) was inferred as part of the *Runaway Malware* scenario. The I LOVE YOU individual was defined in Protégé as follows:

- is a member of the *Attack Scenario* class; and
- has at least one:
  - *Hacker Actor* defined by *hasActor* relationship;
  - *Denial-of-Service Attack Mechanism* defined by *hasAttackMechanismDamage* relationship;
  - *Malware Attack Mechanism* defined by *hasAttackMechanismRampup* relationship;
  - *Automatic* defined by *hasChainAMAutomationLevel* relationship;
  - *Foreign Actor Location* defined by *hasChainActorActorLocation* relationship;
  - *Self Instigator Aggressor* defined by *hasChainActorAggressor* relationship;
  - *Fun Motivation* defined by *hasChainActorAggressorMotivation* relationship;
  - *Spread Attack Goal* defined by *hasChainActorAttackGoal* relationship;
  - *Global Network* defined by *hasChainScopeScopeSize* relationship;
  - *Network Asset* defined by *hasChainTargetAsset* relationship;

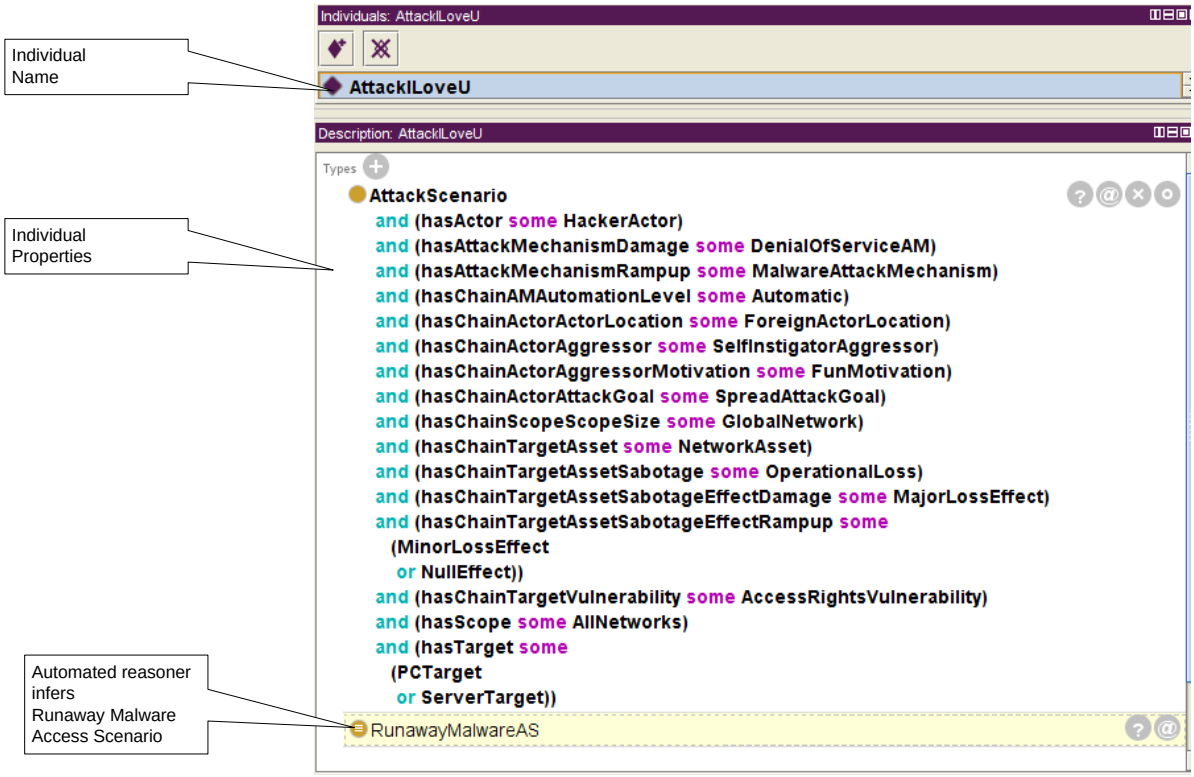


Figure 6.24: I LOVE YOU Inferred a *Runaway Malware* Scenario

- *Operational Loss* defined by *hasChainTargetAssetSabotage* relationship;
- *Minor OR Null Effect* defined by *hasChainTargetAssetSabotageEffectRampup* relationship;
- *Major Effect* defined by *hasChainTargetAssetSabotageEffectDamage* relationship;
- *All Networks Scope* defined by *hasScope* relationship;
- *PC OR Server Target* defined by *hasTarget* relationship.
- *Access Rights Vulnerability* defined by *hasChainTargetVulnerability* relationship;

By setting an individual with properties as above, the automated reasoner HermiT plug-in for Protégé was able to determine that the I LOVE YOU worm falls within the *Runaway Malware* scenario. Protégé output is shown in Figure 6.24, with the automated reasoner-inferred class shown in yellow at the bottom.

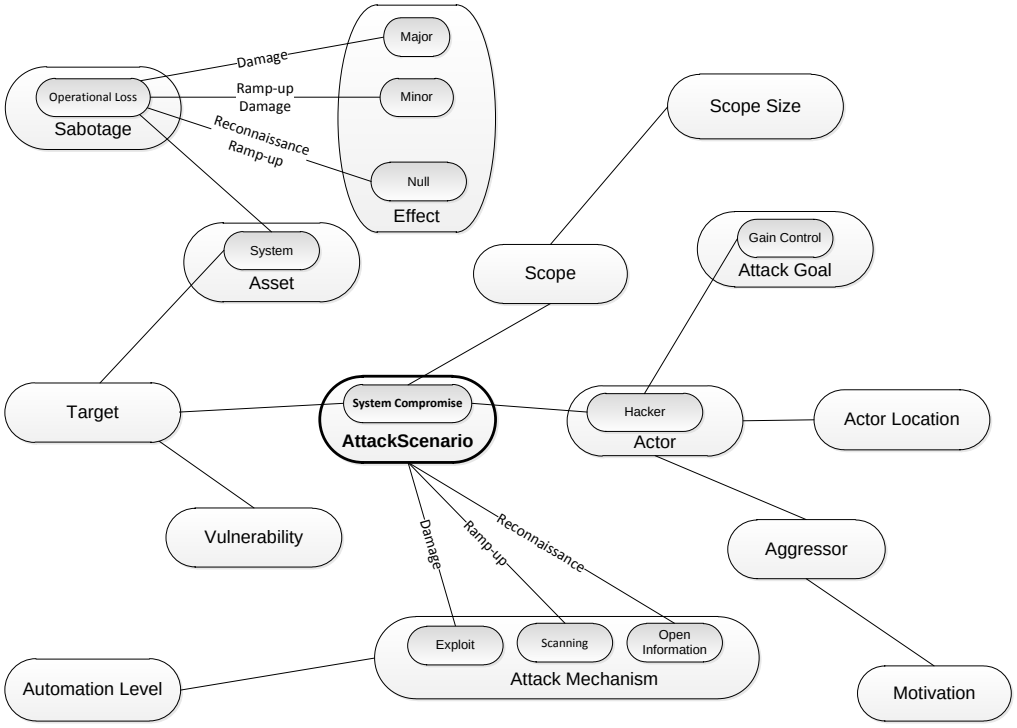


Figure 6.25: System Compromise Attack Scenario

6.10 System Compromise

The *System Compromise* scenario refers to unauthorised personnel or hackers gaining user rights out of their scope. A system compromise attack refers to breaking into a single or multiple computers without authorisation. Such a compromise is often achieved by using another individual's identification and/or password to achieve privilege escalation and then compromise the computer system The story that describes the *System Compromise* Scenario follows (Figure 6.25):
*A **Hacker** based at **[ActorLocation]** with the goal of **Gaining Control** is sponsored by **[Aggressor]** with a **[Motivation]** motivation. The attack effected **[ScopeSize] [Scope]** scope. A **[Target]** was attacked via **[Vulnerability]**. This attack effected **System** and resulted in **Operational Loss** to **Minor OR Major** effect during Damage attack phase, to **Minor OR Null** effect during Ramp-up attack phase and to **Null** effect during Reconnaissance attack phase. During the Reconnaissance phase **Open Information** was used, during the Ramp-up phase **Scanning** was used and during the Damage phase, **Exploit** was used. These mechanisms was automated to **[AutomationLevel]** level.*

The Flame malware (Section2.4.34) can be classified as an example of System Compro-

mise scenario (Figure 6.26):

*A **Hacker** based at **Foreign** location with the goal of **Gaining Control** is sponsored by **State** aggressor with a **Espionage** motivation. The attack effected **[ScopeSize] All Networks** scope. **PCs** were attacked via **Access Rights** vulnerability. This attack effected **System** and resulted in **Operational Loss** to **Major** effect during Damage attack phase, to **Minor** effect during Ramp-up attack phase and to **Null** effect during Reconnaissance attack phase. During the Reconnaissance phase **Open Information** was used, during the Ramp-up phase **Scanning** was used and during the Damage phase, **Exploit** was used. These mechanisms was automated to **Automatic** level.*

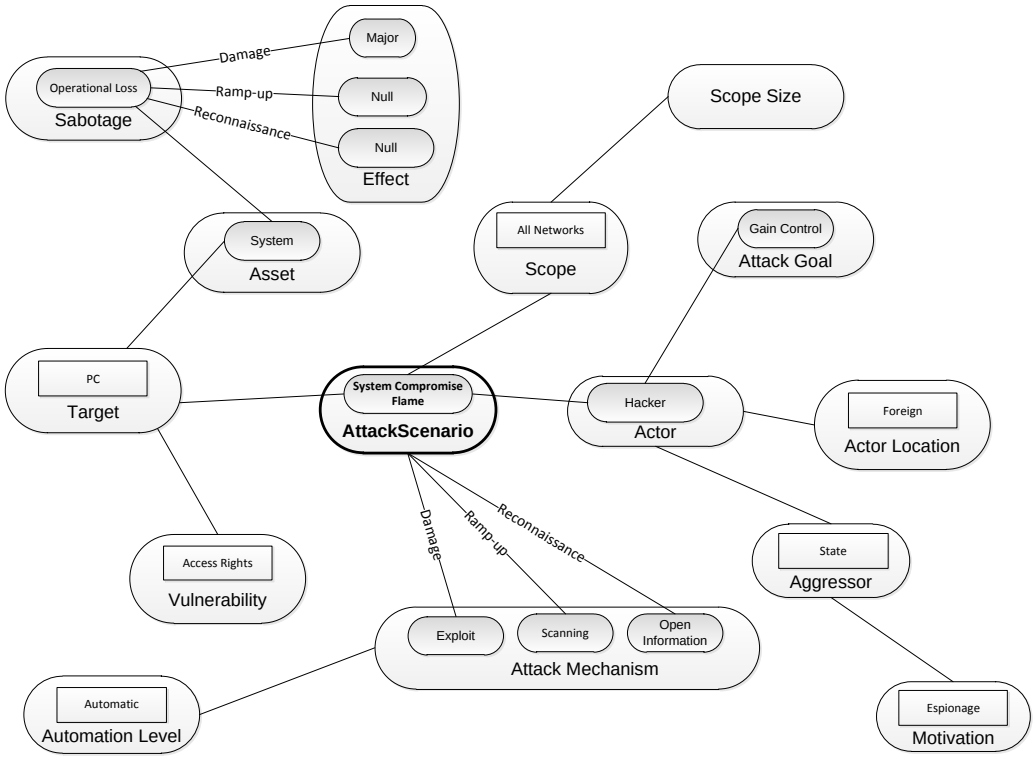


Figure 6.26: Flame System Compromise Attack Scenario Example

6.10.1 System Compromise Formal Description

The *System Compromise* scenario set is defined in statements 6.98 to 6.105 (also refer to Figure 6.25). In Figure 6.25, the sub-classes that are specific to the *System Compromise (SC)* scenario are displayed. This demonstrates which sub-classes are used when the Financial Theft attack scenario is presented.

$$SystemCompromise \subseteq AttackScenario \tag{6.98}$$

$$\begin{aligned} SystemCompromise = \{x | (\exists v \in Hacker \ni (x, v) \in hasActor) \wedge \\ (\exists w \in OpenInformation \ni (x, w) \in hasAttackMechanismRecon) \wedge \\ (\exists y \in Scanning \ni (x, y) \in hasAttackMechanismRampup) \wedge \\ (\exists s \in Exploit \ni (x, s) \in hasAttackMechanismDamage)\} \end{aligned} \tag{6.99}$$

$$HackerSC \subseteq Hacker \subseteq Actor \tag{6.100}$$

$$HackerSC = \{x | \exists z \in GainControl \ni (x, z) \in hasAttackGoal\} \tag{6.101}$$

$$SystemSC \subseteq System \subseteq Asset \tag{6.102}$$

$$SystemSC = \{x | \exists y \in OperationalLoss \ni (x, y) \in hasSabotage\} \tag{6.103}$$

$$OperationalLossSC \subseteq OperationalLoss \subseteq Sabotage \tag{6.104}$$

$$\begin{aligned} OperationalLossSC = \{x | (\exists y \in Null \ni (x, y) \in hasSabotageRecon) \wedge \\ (\exists w \in (Null \cup Minor) \ni (x, w) \in hasSabotageRampup) \wedge \\ (\exists z \in (Minor \cup Major) \ni (x, z) \in hasSabotageDamage)\} \end{aligned} \tag{6.105}$$

6.10.2 System Compromise Individual

The Flame (Section 2.4.34) was inferred as part of the *System Compromise* scenario. The Flame individual was defined in Protégé as follows:

- is a member of the *Attack Scenario* class; and
- has at least one:
  - *Hacker Actor* defined by *hasActor* relationship;
  - *Exploit Attack Mechanism* defined by *hasAttackMechanismDamage* relationship;
  - *Scanning Attack Mechanism* defined by *hasAttackMechanismRampup* relationship;
  - *Open Information Attack Mechanism* defined by *hasAttackMechanismRecon* relationship;
  - *Foreign Actor Location* defined by *hasChainActorActorLocation* relationship;
  - *State Aggressor* defined by *hasChainActorAggressor* relationship;

- *Espionage Motivation* defined by *hasChainActorAggressorMotivation* relationship;
- *Gain Control Attack Goal* defined by *hasChainActorAttackGoal* relationship;
- *System Asset* defined by *hasChainTargetAsset* relationship;
- *Operational Loss* defined by *hasChainTargetAssetSabotage* relationship;
- *Null Effect* defined by *hasChainTargetAssetSabotageEffectRecon* relationship;
- *Null Effect* defined by *hasChainTargetAssetSabotageEffectRampup* relationship;
- *Major Effect* defined by *hasChainTargetAssetSabotageEffectDamage* relationship;
- *Access Rights Vulnerability* defined by *hasChainTargetVulnerability* relationship;
- *Automatic* defined by *hasChainAMAutomationLevel* relationship;
- *All Networks Scope* defined by *hasScope* relationship;
- *PC Target* defined by *hasTarget* relationship.

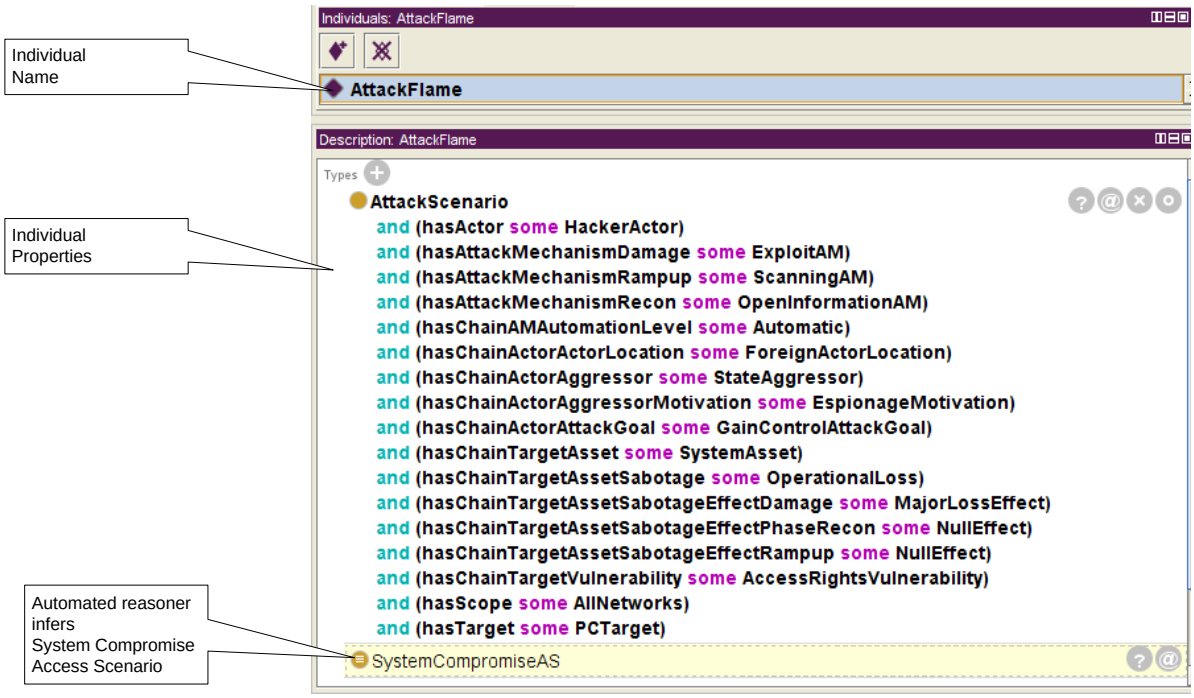


Figure 6.27: Flame Inferred a *System Compromise* Scenario

By setting an individual with properties as above, the automated reasoner HermiT plug-in for Protégé was able to determine that the Flame malware falls within the *System Compromise* scenario. Protégé output is shown in Figure 6.27, with the automated reasoner-inferred class shown in yellow at the bottom.

6.11 Conclusion

In this chapter, the stories and formalised descriptions of the remaining attack scenarios are presented. For each scenario an example individual is presented and it is demonstrated how the automated reasoner classifies the individual to its respected attack scenario class. The following attack scenarios are presented:

- *Web Defacement*,
- *Unauthorised Data Access*,
- *Cyber-Warfare*,
- *Industrial Espionage*,
- *Financial Theft*,
- *Resource Theft*,
- *Industrial Sabotage*,
- *Runaway Malware* and
- *System Compromise*.

This concludes the theoretical part of the thesis. In the next part, the implications of a near real-time environment are investigated and a prototype system that classifies network attacks is developed and tested.

Part III

Near Real-time

CHAPTER

SEVEN

EVALUATION OF NEAR REAL-TIME FITNESS

> "These blast points – too accurate for sand people. Only imperial storm troopers are so precise."
>
> Obi-Wan Kenobi – *Star Wars*

7.1 Introduction

In Part III, the researcher will investigate the effect of network attacks in a near real-time environment. The effect of a near real-time environment with the taxonomy developed in Chapter 4 is explored in this chapter. In the following chapter, a prototype system that identifies network attacks is presented. This part concludes with a validation chapter. In the validation chapter, two test beds are presented according to which the prototype system was verified.

The degree to which each class can be quantified or measured is determined by investigating the accuracy of various assessment methods, and then classifying the class as either defined, high, low or not quantifiable. For example, it may not be possible to determine the instigator of an attack (*Aggressor*), but only that the attack has been launched by a hacker (*Actor*). In addition, modeling malware depends on information reported by industry, media and academic papers. Thus the full functionality of malware may not

be known until much later than its initial discovery. Some classes can only be quantified with low confidence or not at all in (near real-time) time. The IP address of an attack can easily be faked (Bellovin, 1989), thus reducing the confidence regarding the information obtained from it, and thus determining the origin of an attack with low confidence. This determination itself is subjective. All the evaluations of the classes in this chapter are subjective, but due to the very basic grouping (high, low or not quantifiable) a subjective value can be used.

By relaxing the scenario definition to only classes that can be quantified in near real-time, some of the relaxed attack scenario classes were found to be equivalent. Thus the scenarios are reduced to the following:

- Denial-of-Service
- Web Defacement
- Resource Theft
- Unauthorised Data Access
- System Comprise
- Runaway Malware

Each of the classes and sub-classes can be quantified either directly or indirectly. They can also be defined by the configuration of the system under attack. Some classes cannot be quantified in a near real-time environment. This chapter describes the three levels of quantification wherein each class is placed with respect to measurement. Some classes are not measured, but defined by the nature of the attack. Thus the accuracy of the quantification is assigned. Three levels of accuracy are assigned: high, low or not quantifiable (Fenz and Neubauer, 2009; Fenz, Tjoa, and Hudec, 2009). Fenz and Neubauer state:

> "Since the threat probability or in influencing factors cannot be determined quantitatively, a qualitative rating is used in this approach. In contrast to a quantitative rating with which it is hardly possible to determine the occurrence of a certain threat with a 67% and not with a 68% chance, a qualitative rating (e.g. high, medium, and low)."

Since the accuracy of quantifying the classes can also be defined, the researcher also effectively uses three levels of qualitative ratings. In Section 7.2 the quantification of the classes within the taxonomy is presented. In Section 7.3 a relaxed attack scenarios are presented formally, and relaxed scenarios which are equivalent are shown.

7.2 Taxonomy Quantification

Each of the classes defined in Chapter 4 are investigated as to how they can be measured or quantified in near real-time. Some of the classes are quantified by definition and do not require any sensors to determine their value. These classes are referred to as "defined". For example, the target of an attack is not measured or quantified, but is rather "defined" by the attack. Some attacks are named after the target, such as in the cases of the SCO and SpamHaus attacks (sections 2.4.18 and 2.4.35). Some classes cannot be measured in a near real-time environment. The values of these classes only become apparent long after an attack and even then there is sometimes only speculation. For example, the *Aggressor* cannot be determined in a near real-time environment and for some attacks, the real power behind the attack is never determined or proven.

7.2.1 Actor Quantification

The *Group Actor* sub-class and its sub-classes *Organised Criminal Group*, *Protest Group* and *Cyber Army* can be quantified by their IP addresses. An IP address can be used to find the physical location of a *Group Actor*. Free and subscription geolocation databases exist, which claim to be capable of identifying the physical location of any IP address worldwide. Thus looking up IP addresses is considered a direct quantification. The group that owns/rents that location can be determined from the IP location. By using the IP, the *Group Actor* can be determined indirectly. Shavitt and Zilberman (2011) studied the accuracy of geolocation databases and found that the results of most databases are similar and that the accuracy cannot be trusted. Errors included wrongful estimation of distances and incorrect identification of country. IP addresses can be spoofed, and intermediate computers located anywhere in the world can be used for attack. For these reasons, using IP to locate the Group Actor is assigned a low accuracy.

The *Hacker* sub-class and its sub-classes *Script Kiddie* and *Skilled Hacker* can be quantified by looking at the pattern of an attack. One of the first detailed documented determinations that a system was hacked by a skilled hacker was done by Stoll (1989), where Stoll determined that the hacker was extremely skilled by printing out all the keystrokes of the attack. Script Kiddies use standardised tools of which the characteristics (or fingerprint) are static and can be identified. For example, the pattern of standard Nmap scans can easily be identified (Staniford, Hoagland, and McAlerney, 2002; Ezzeldin, 2008). By using an elaborate honeypot, the skill level of a *Hacker Actor* can also be determined

(Ramsbrock, Berthier, and Cukier, 2007; Aliyev, 2010; Kibret, 2011). Script Kiddies will attack the honeypot directly with standard tools such as Metasploit[1] with all the possible exploits (Sigholm, 2013), whereas skilled hackers will use more subtle techniques and only targeted exploits and try to hide their origin (Yung, 2002). The skill level of hackers can also be deduced by the consequences of their attacks. Meyers *et al.* (2009) stated that skilled hackers are rare and dangerous, and that information about them is rare. If the attack was successful in web defacement or a secure server was compromised, it can be assumed that a skilled hacker was involved. Tripwire[2] and other host-based IDSs can alert system administrators to compromises, although they cannot prevent attacks. They notify administrators that some secure data has been accessed or modified. Thus the Hacker Actor can be measured indirectly, and the accuracy is low.

Insider threats can be detected by internally orientated honeypots or telescopes (Spitzner, 2003; Myers, Grimaila, and Mills, 2009; Maybury, Chase, Cheikes, Brackney, Matzner, Hetherington, Wood, Sibley, Marin, and Longstaff, 2005). These insider honeypots work according to the same principle as externally orientated honeypots, but reside within a network and are not accessible from outside. Externally orientated honeypots are connected to external networks and capture traffic from attackers from outside the scope of the defender's network. Insider honeypots can detect *Normal User*, but not *Administrator*. Administrators have access to most of the network. No network can be made safe against its own administrators, thus administrators fall within the immeasurable group, whereas normal users can be measured directly. When such honeypots are triggered, the odds of it being an insider is low due to possible false positives or attackers masquerading as insiders. All the sub-classes of the *Actor* class have low accuracy, thus in summary, the *Actor* class accuracy is defined as low.

7.2.2 Actor Location Quantification

The *Actor Location* class and its sub-classes can be measured in a similar way to the *Group Actor* sub-class by means of IP location. Only a single look-up in a geolocation database is required and it is therefore considered to be directly measurable. The values of the geolocation database are also considered unreliable, with Poese, Uhlig, Kaafar, Donnet, and Gueye (2011) stating that these geolocation databases are accurate at a country level, but not at a city level. (Hunter and Irwin, 2011) developed a framework to track malware

[1] http://www.metasploit.com/
[2] http://www.tripwire.org/

via their IP address. An alternative method to find the location of IP addresses is to use latency measurements (Katz-Bassett, John, Krishnamurthy, Wetherall, Anderson, and Chawathe, 2006). Katz-Bassett *et al.* were able to achieve a medium error of 67 km in optimal circumstances. The same accuracy problems as stated for the *Group Actor* apply to the *Actor Location* sub-classes. Thus the accuracy is considered to be low.

7.2.3 Aggressor, Motivation, Effect and Sabotage Quantification

The *Aggressor* cannot be quantified in near real-time. In most cases, the aggressor is only determined months after an attack. For example, it took a few months before the aggressor behind the Stuxnet attack was confirmed (Sanger, 2012). The aggressor and people behind most viruses are difficult if not impossible to find (Shiffman and Gupta, 2013). The *Aggressor* class and its sub-classes are not considered to be quantifiable. The same is true regarding the motivation of an aggressor, which can also not be determined in near real-time. The type of sabotage caused by an attack can only be calculated after the full impact of the attack is known, and cannot thus be measured in near real time. The effects of an attack can only be quantified with a full investigation into the compromised systems and assessments of the damage done. Thus the *Aggressor*, *Effect*, *Motivation* and *Sabotage* class and its sub-classes cannot be measured in near real-time.

7.2.4 Asset Quantification

The *Access* and *System* sub-classes of the *Asset* class can be measured with automated testing scripts. These testing scripts simulate human requests at a very basic level and can thus indicate when access to the system or the system functionally have been altered. Stout (2001) stated that automated testing is critical to a quality website and his statement holds true for all servers. The scripts directly measure access and the system's functionality, and the accuracy of these quantifications are regarded to be as high.

The *Data* sub-class of the *Asset* class can be quantified by host-based IDSs. These sensors are capable of determining alterations to data. Typically, two main aspects of the data can be measured, namely unauthorised access or unauthorised manipulation of the data (Lunt, 1993). These quantifications are direct and occur in the Application layer. Although the possibility of false alarms exists (Tjhai, Papadaki, Furnell, and Clarke, 2008), these quantifications are considered to be very accurate.

The *Network* sub-class of the *Asset* class can be quantified indirectly by considering the networking performance of devices or testing whether systems in the network can communicate (Hariri, Qu, Dharmagadda, Ramkishore, and Raghavendra, 2003). Communication errors, hardware breakdown or system misconfiguration can be possible reasons for disruption of communication. The accuracy of quantifying an attack on the network is regarded as high. All the sub-classes of the *Asset* class are very accurate and thus, in summary, the accuracy of the *Asset* class is defined as high.

7.2.5 Attack Goal Determination

The *Attack Goal* can be determined indirectly by ascertaining which asset is under attack. Similar to the *Data* sub-class of the *Asset* class, the *Destroy Data*, *Steal Data*, *Gain Control*, *Spread* and *Change Data* sub-classes can be determined by host-based IDSs (Lunt, 1993). The *Disrupt* sub-class can be determined indirectly by looking at the type of attack that is launched on a honeypot or similarly to the *Network* sub-class of the *Asset* class, by monitoring the network performance (Lunt, 1993; Kuwatly *et al.*, 2004). The accuracy of determining the goal is considered to be high.

The *Gain Resources* sub-class of the *Attack Goal* class can be determined by intercepting communications that do not fit the normal profile. Strayer, Lapsely, Walsh, and Livadas (2008) developed a system that identifies networks that support malicious traffic[3]. Thus malicious traffic bound for addresses listed in their system can be null-routed. The Finding Rogue Network project has since been discontinued, but similar work is done commercially by Lastline[4]. It can be determined if local systems are being used as a springboard for attacks on others. Since this determination depends on the accuracy of the identification of malicious networks, and the possibility of misconfigured networks looking like botnets, the determination of *Gain Control* is not considered to be very accurate.

Since five out of the six sub-classes are of high accuracy,, the *Attack Goal* class accuracy is defined as high.

7.2.6 Attack Mechanism Determination

The *Information Gathering* sub-class of the *Attack Mechanism* class can be indirectly measured by detecting scans. These scans can be detected by interpreting access logs or

[3]http://maliciousnetworks.org/
[4]http://www.lastline.com/

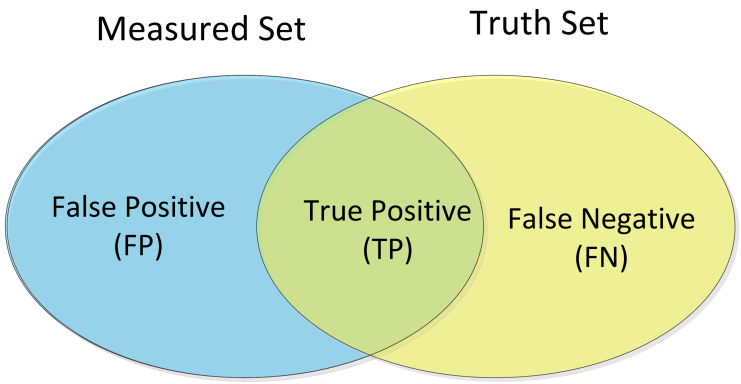


Figure 7.1: The Difference between False Negative and False Positive

analysing network traffic (Lee *et al.*, 2003; Bhuyan, Bhattacharyya, and Kalita, 2011). Port scanning and vulnerability scan determination have a high accuracy rate.

The *Brute Force*, *Escalation*, *Spoofing*, *Session Hijack* and *Buffer Overflow* sub-classes can be identified by network-based IDSs and by looking at access logs. These are directly identified by matching known methods to observed events. The accuracy of identifying these attacks mechanisms is high.

The *Spear Phishing* and *Social Engineering* sub-classes can be identified by specially crafted traps that lure such attackers to a fake target (Harley and Lee, 2007; Merritt, 2011). Due to the difficulty of detecting social engineering attacks, detection of such attack mechanisms have low accuracy.

The *Network Based* sub-class can be identified indirectly by intercepting strange communications or by monitoring the amount of traffic on the system (Heberlein, Dias, Levitt, Mukherjee, Wood, and Wolber, 1990). Although it is difficult to distinguish between attacks and innocent network anomalies, it is simple to detect and it is thus highly accurate.

The *Malware* attack mechanism can be identified either on the Open Systems Interconnection (OSI) Application layer with Antivirus software, or in the OSI Network layer with IDS software (Christodorescu and Jha, 2004). Malware can be identified directly and the accuracy of the identification is high with a low false positive rate. False positive refers to when a classifier incorrectly classified an item as harmful (Owen, 2010). Malware that is not detectable is also a concern (Christodorescu and Jha, 2004). False negative refers to malware that was not detected. In Figure 7.1, the difference between False Negative and False Positive is shown. The detection of *Malware* is highly accurate.

If a sub-system is abused, it can be measured simply by looking at systems logs. The processing utilisation and disk usage can be measured directly on systems. A firewall and

some advanced routers can measure the network throughput, thus identifying bandwidth abuse. The detection of *System Abuse* is highly accurate.

A *Web Application* such as a *SQL Injection* or a *Web Crawl* attack mechanism can be detected directly with specially crafted traps or logging of unusual web behaviour (Rietta, 2006; Fu, Lu, Peltsverger, Chen, Qian, and Tao, 2007; Manmadhan and Manesh, 2012). Error messages can also be used to detect *SQL Injection* attacks (Ciampa, Visaggio, and Di Penta, 2010). Win and Htun (2013) used SQL normal queries to identify safe request and then identity attacks by restricting the allowed queries. Misuse of Web applications have a high accuracy level.

XSS Web Application attack mechanisms can be detected indirectly by comparing posted URLs to blacklisted sites (Jim, Swamy, and Hicks, 2007), by identifying typical Cross-site Scripting (XSS) coding patterns (Mookhey and Burghate, 2004; Scholte, Robertson, Balzarotti, and Kirda, 2012). The detection and prevention of XSS attacks are difficult because of incomplete implementations, inherent limitations, the complexity of development frameworks and the requirement for run-time compatibility (Rao, 2012). The efficiency of this detection method is determined by the quality of the blacklist, and the accuracy level is low.

Denial-of-Service attack mechanisms can mostly be detected by filtering incoming network traffic (Karig and Lee, 2001; Argyraki and Cheriton, 2005). Mirkovic and Reiher (2004) present a taxonomy in defences that can be used against DoS attacks, which includes: system security, protocol security, resource accounting, resource multiplication, pattern matching, anomaly detection, filtering, automated reconfiguring, rate limiting and agent identification. The accuracy of detecting DoS attack mechanisms is high.

Since most of the sub-classes of the *Attack Mechanism* class are of high accuracy, the accuracy of the *Attack Mechanism* class is high.

7.2.7 Automation Level Quantification

The *Automatic* sub-class of the *Automation Level* class can be indirectly quantified by observing the scanning pattern and other features with honeypots and other scan detection sensors. Kuwatly *et al.* (2004) and Staniford *et al.* (2002) were able to detect Nmap[5] scans by training their detection systems to recognise Nmap-specific scan characteristics.

[5] http://nmap.org/

Similarly, it should be possible to detect automated tools by their specific behaviour. A lack of automation can point to the *Manual* or *Semi-automatic* automation level. The accuracy level of these quantifications is low since the difference between automation and the other modes is difficult to determine and thus difficult to quantify.

7.2.8 Phase Classification

The *Phases* of an attack are set by the attack scenario. Thus the phase of an attack is considered to be defined rather than quantifiable. The process of determining which phase of an attack is currently happing is discussed in Section 8.4.1, and is an outcome of the research presented in this thesis.

7.2.9 Scope and Scope Size Measurement

The target scope and the scope size are defined by the entity under attack. These classes represent physical attributes of the target, which cannot be measured or quantified, and should rather be considered so as to be defined.

7.2.10 Target Monitoring

The *Target* class and its sub-classes can be monitored indirectly by observing which systems are not performing as expected. The *Network Infrastructure* sub-class can be observed by monitoring network performance in the Network layer. Attacks that affect the *PC* sub-class can be observed using anti-virus software. The *Server* sub-class can be monitored by using heart-beat sensors or data integrity sensors (Bhide, Elnozahy, and Morgan, 1991). *Industrial Equipment* is monitored directly via its control software (Yang, Usynin, and Hines, 2006). Industrial equipment can monitor communications in the Physical, Network and Application layers. Even though system problems or other errors can also lead to system failures, monitoring these classes is considered to be highly accurate.

7.2.11 Vulnerability Identification

The *Vulnerability* class and its sub-classes can be identified directly using a combination of IDSs and honeypots (Gula, 2011). Although IDSs can have false positives (incorrectly

Table 7.1: Summary of the Measurement Taxonomy

Class	Quantification	Accuracy
Actor	Indirect	Low
Actor Location	Direct	Low
Aggressor	Not Quantifiable	N/A
Asset	Direct	High
Attack Goal	Indirect	High
Attack Mechanism	Indirect	High
Automation Level	Indirect	Low
Effect	Not Quantifiable	N/A
Motivation	Not Quantifiable	N/A
Sabotage	Not Quantifiable	N/A
Scope	Defined	N/A
Scope Size	Defined	N/A
Target	Indirect	High
Vulnerability	Direct	High

identify attacks), their accuracy is considered to be high.

7.2.12 Quantification Summary

In Table 7.1, a summary of the required quantification is shown. This table lists all the classes with respect to quantification methodology and accuracy. Only five of the classes are considered quantified or measurable for high accuracy: *Asset, Target, Vulnerability, Attack Mechanism* and *Attack Goal*. Three classes are considered to have low accuracy: *Automation Level, Actor* and *Actor Location*. Four classes cannot be measured or quantified in a near real-time environment: *Sabotage, Effect, Aggressor* and *Motivation*. The remaining two classes are defined: *Scope* and *Scope Size*.

7.3 Attack Scenarios Quantification

Not all the attack scenarios that were identified in Chapter 2 and formally described in Chapter 5 and Chapter 6 can be identified in near real-time. Effectively, only the *Attack Mechanism, Asset, Target* and *Vulnerability* classes can be quantified to a high accuracy level in a near real-time environment. In Figure 7.2, the impact of the quantification options on the ontology is shown. As shown in Figure 7.2; only a subset of the classes

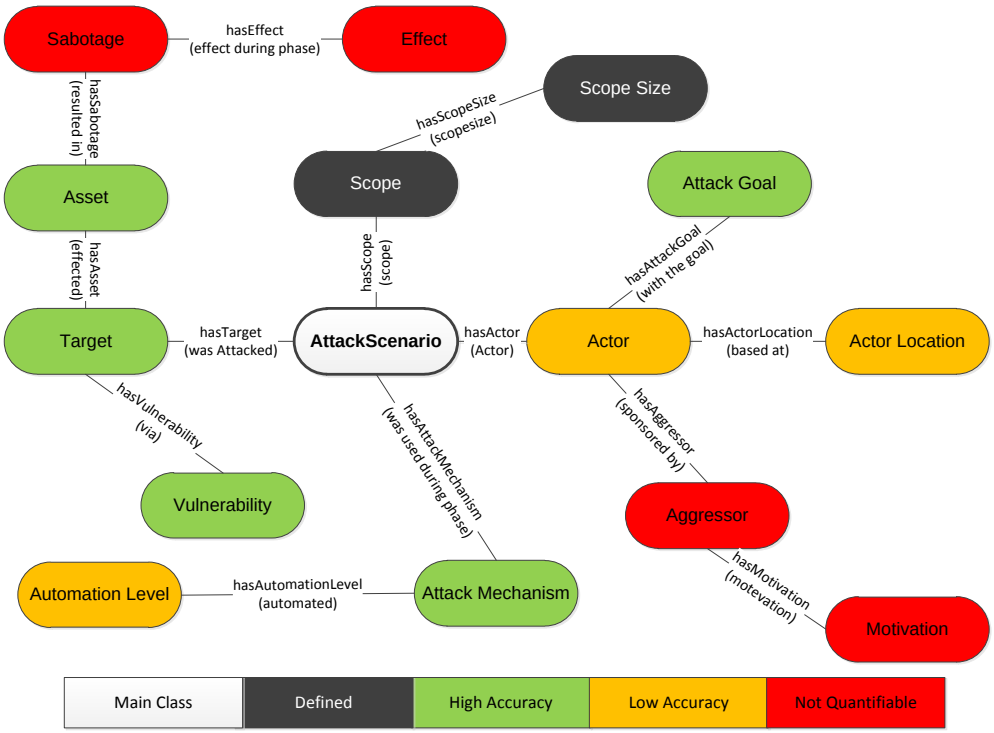


Figure 7.2: Impact of Quantification on the Ontology

are of significance in a near real-time environment. This environment is referred to as: relaxed. A relaxed environment only used classes and relationships that are quantifiable and measurable in near real-time. Thus a scenario will be a subset of the relaxed scenario. In the following sub-sections the relaxed *Denial-of-Service* and *Cyber-Warfare* are explored in detail. These were selected as they demonstrate how two different scenarios' relaxed versions can be equivalent.

7.3.1 Relaxed Denial-of-Service and Cyber-Warfare Scenarios Formal Descriptions

In this sub-section, the formal description of the *Denial-of-Service* and *Cyber-Warfare* scenarios of sections 5.4.4 and 6.2.1 are described with only the classes that have a high quantification accuracy. Attack scenarios with only the near real-time classes are considered relaxed attack scenarios. These relaxed scenarios are used to describe scenarios with only classes that have high accuracy in a near real-time environment. Thus the conditions for the relaxed scenarios are less strict than the corresponding scenarios.

This relaxed definition of the *Denial-of-Service* scenario, *DoSRelaxed*, can formally be

written as:

$$DoS \subseteq DoSRelaxed \subseteq AttackScenario \tag{7.1}$$

$$\begin{aligned} DoSRelaxed = \{x | (\exists v \in AttackGoal \ni (x,v) \in hasChainActorAttackGoal) \wedge \\ (\exists w \in DenialofServiceAM \ni (x,w) \in hasAttackMechanismRampup) \wedge \\ (\exists y \in DenialofServiceAM \ni (x,y) \in hasAttackMechanismDamage) \wedge \\ (\exists u \in NetworkInfrastructure \ni (x,u) \in hasTarget)\} \end{aligned} \tag{7.2}$$

$$NetworkInfrastructure\,DoSRelaxed \subseteq NetworkInfrastructure \subseteq Target \tag{7.3}$$

$$NetworkInfrastructure\,DoSRelaxed = \{x | \exists z \in Access \ni (x,z) \in hasAsset\} \tag{7.4}$$

$$AccessDoSRelaxed \subseteq Access \subseteq Asset \tag{7.5}$$

$$hasChainActorAttackGoal = hasActor \circ hasAttackGoal \tag{7.6}$$

The relaxed definition of the *Cyber-Warfare* scenario, *CyberWarfare Relaxed*, can formally be written as:

$$CyberWarfare \subseteq CyberWarfare\,Relaxed \subseteq AttackScenario \tag{7.7}$$

$$\begin{aligned} CyberWarfare\,Relaxed = \\ \{x | (\exists v \in AttackGoal \ni (x,v) \in hasChainActorAttackGoal) \wedge \\ (\exists w \in DenialofServiceAM \ni (x,w) \in hasAttackMechanismRampup) \wedge \\ (\exists y \in DenialofServiceAM \ni (x,y) \in hasAttackMechanismDamage) \wedge \\ (\exists u \in NetworkInfrastructure \ni (x,u) \in hasTarget)\} \end{aligned} \tag{7.8}$$

$$\begin{aligned} NetworkInfrastructure\,CyberWarfare\,Relaxed \subseteq \\ NetworkInfrastructure \subseteq Target \end{aligned} \tag{7.9}$$

$$\begin{aligned} NetworkInfrastructure\,CyberWarfare\,Relaxed = \\ \{x | \exists z \in Access \ni (x,z) \in hasAsset\} \end{aligned} \tag{7.10}$$

7.3.2 Inferring *Cyber-Warfare* and *Denial-of-Service* Scenarios

The defining statements for the relaxed *Denial-of-Service* and relaxed *Cyber-Warfare* are the same, thus in a near real time environment these scenarios can be merged into a

single scenario. This deduction is supported by the automatic reasoner. Within the Protégé editor, the HermiT reasoner was able to infer that the relaxed *Cyber-Warfare* and *Denial-of-Service* scenarios are the same as shown in Figure 7.5. In Figure 7.5, the automatic reasoner results are shown highlighted, indicating that the *Relaxed Cyber-Warfare* and *Relaxed Denial-of-Service* scenarios are the same. The two scenarios are shown together in Figure 7.3. In Figure 7.3, the classes that are not used in a near real-time environment are greyed out, and the classes that can be quantified in near real-time are left open. Effectively only the *Attack Mechanism, Asset, Target* and *Vulnerability* classes are used in the relaxed near-real time environment. The subset-relationships were given in statements 7.2 and 7.7:

$$CyberWarfare \subseteq CyberWarfare\,Relaxed \tag{7.11}$$

$$DoS \subseteq DoSRelaxed \tag{7.12}$$

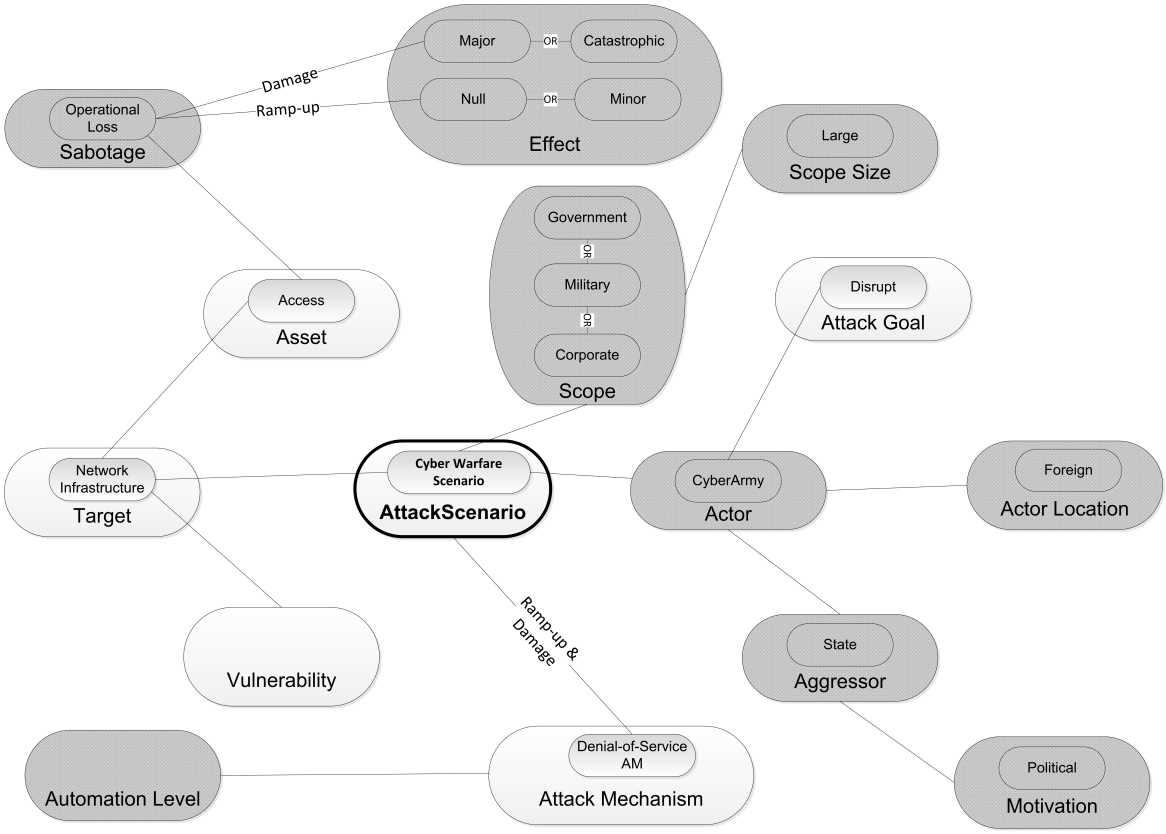
In Figure 7.3a, *Access* is the sub-class of Asset for *Cyber-Warfare* scenario and in Figure 7.3b, *Access* is the sub-class of Asset for *Denial-of-Service* scenario. Similarly, both scenarios' attack goal is *Disrupt* and their target is *Network Infrastructure*. It follows that the requirements for *DoS Relaxed* and *Cyber-Warfare Relaxed* are exactly the same, thus:

$$DoSRelaxed \equiv CyberWarfare\,Relaxed \tag{7.13}$$

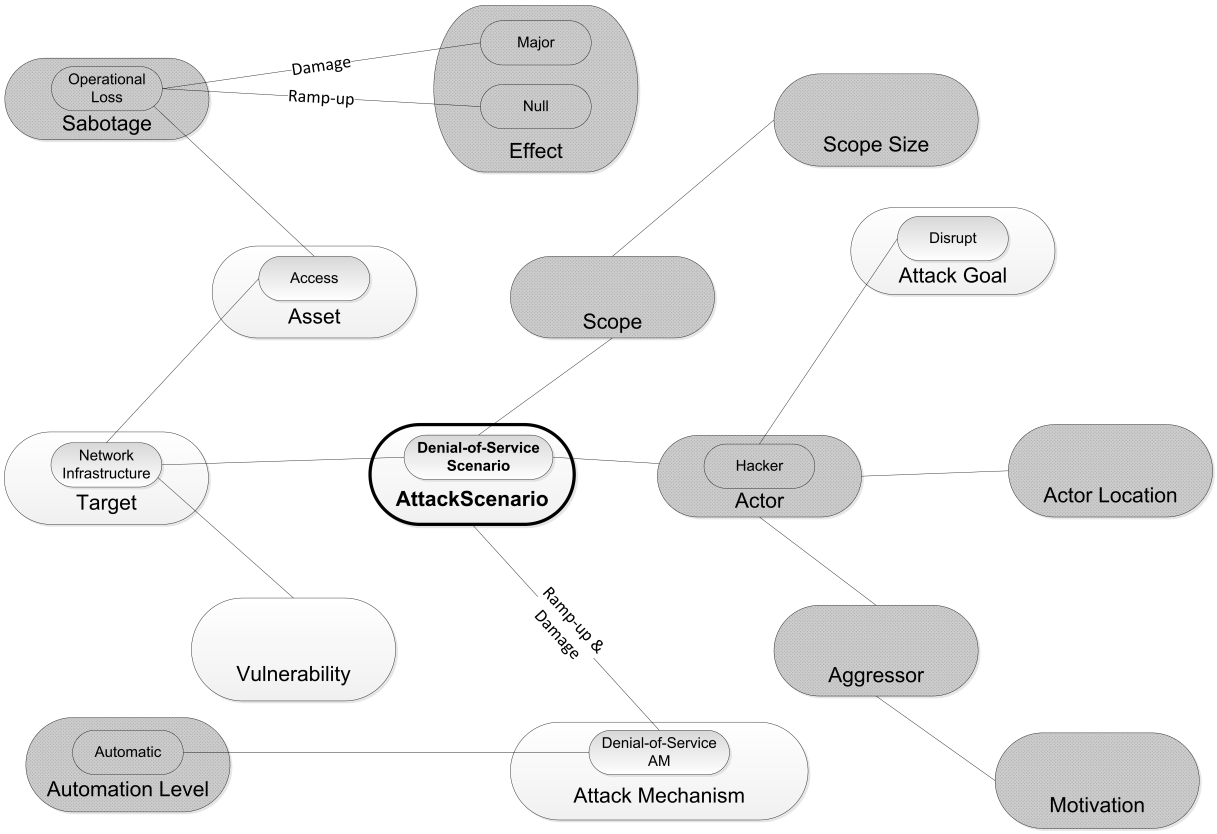
Figure 7.4 displays the relaxed *Cyber-Warfare* and *Denial-or-Service* subsets. In Figure 7.4a, the two scenario classes *Cyber-Warfare* and *Denial-of-Service* are shown as a sub-class of the *Attack Scenario*. These two classes are disjointed (thus they do not overlap). Each of the sub-classes have individuals — South Ossetia an individual of the *Cyber-Warfare* sub-class and SCO an individual of the *Denial-of-Service* sub-class.

The relaxed sub-classes are shown in Figure 7.4b. The *Cyber-Warfare* is contained within the *Relaxed Cyber-Warfare* class. Similarly, the *Relaxed Denial-of-Service* fully contains the *Denial-of-Service* class. The relaxed classes are thus more encompassing than other sub-classes. An individual to one of the sub-classes will also be an individual to the related relaxed class. Thus the SCO individual belongs to the *Relaxed Denial-of-Service* as well as the *Denial-of-Service* sub-classes. Similarly, the South Ossetia individual belongs to both the *Relaxed Cyber-Warfare* and *Cyber-Warfare* classes.

The two relaxed sub-classes were shown to be equivalent in Statement 7.13. This is shown in figures 7.4c and 7.4d. In Figure 7.4d, it is shown that the two sub-classes *Cyber-Warfare*

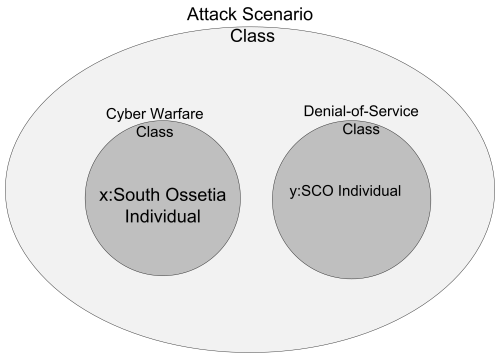


(a) *Relaxed Cyber-Warfare* Scenario

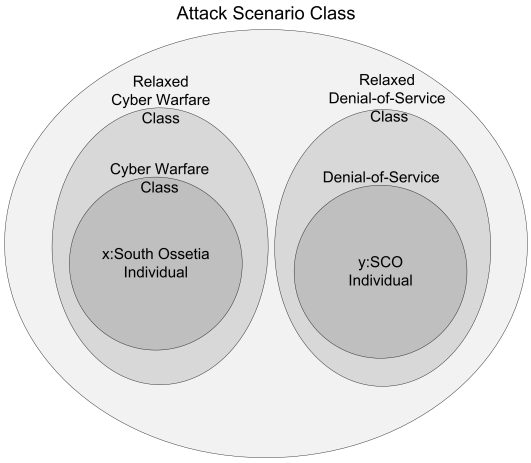


(b) *Relaxed Denial-of-Service* Scenario

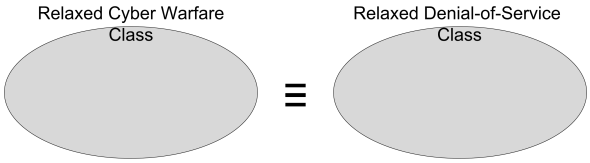
Figure 7.3: Relaxed *Cyber-Warfare* and *Denial-of-Service* Scenarios



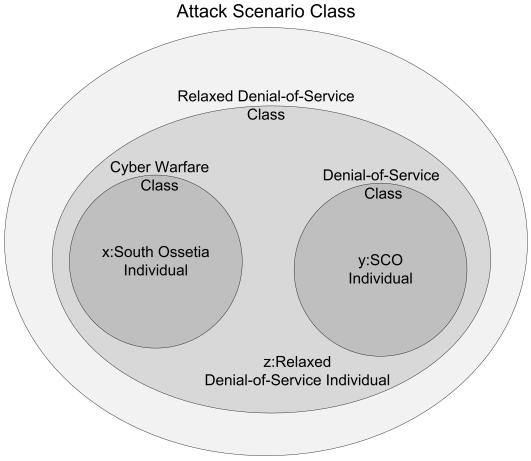
(a) *Cyber-Warfare* and *Denial-of-Service* Subsets



(b) Relaxed and Original Subsets



(c) Relaxed *Cyber-Warfare* and *Denial-of-Service* are Equivalent



(d) Relaxed *Denial-of-Service* Subset

Figure 7.4: Relaxed *Cyber-Warfare* and *Denial-of-Service* Subset Visually Presented

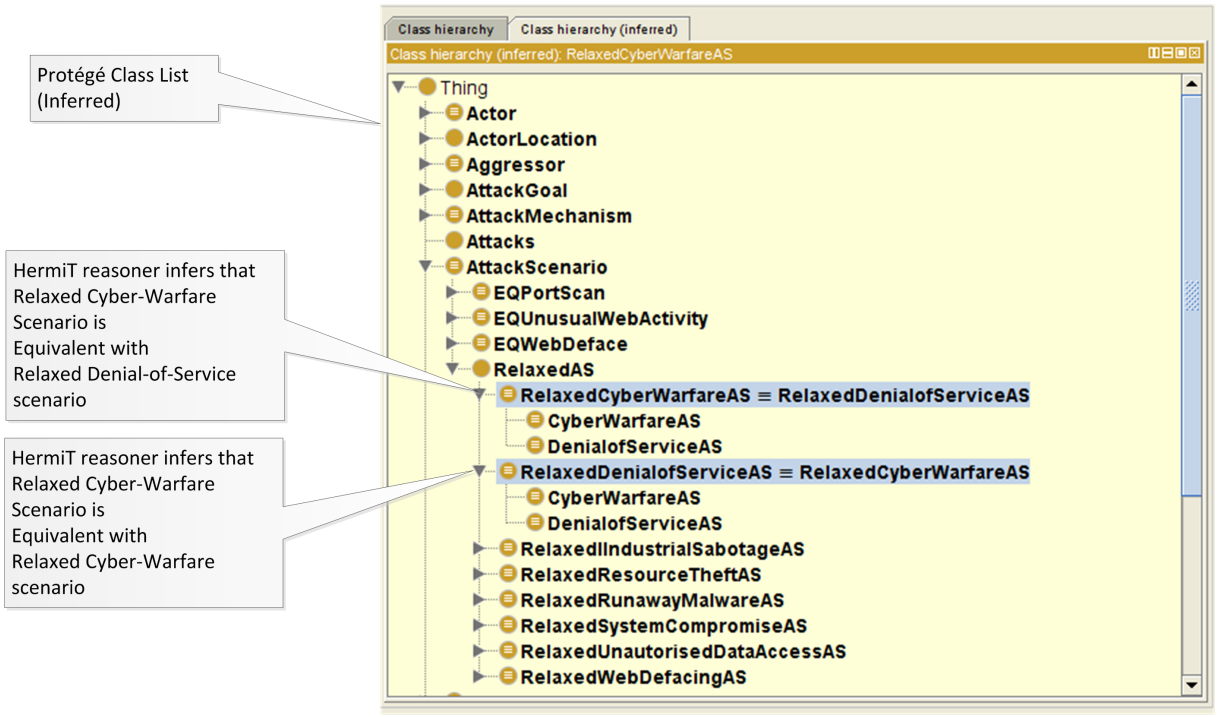


Figure 7.5: Protégé and HermiT Inferring the Relaxed *Cyber-Warfare* and *Denial-of-Service* Scenarios

and *Denial-of-Service* fall within the single *Relaxed Denial-of-Service* sub-class. Since the two relaxed classes are equivalent and the sub-classes are contained within them, they can be shown as a single sub-class that contains the respective sub-classes.

In Figure 7.4c an individual z is shown. Any individuals that are detected in the *Relaxed Denial-of-Service* sub-class can either later be shown to be within the *Cyber-Warfare* or *Denial-of-Service* sub-class. It is also possible that the individual z is not a member of either sub-class. Individuals detected in the near real-time environment would fall within the relaxed sub-class. For example, in the initial states South Ossetia and SCO attacks, it would be classified within the *Relaxed Denial-of-Service* sub-class.

The automated reasoner HermiT showed that the *Relaxed Cyber-Warfare* and *Relaxed Denial-of-Service* sub-classes are equivalent. In Figure 7.5, Protégé shows that the *Cyber-Warfare* and *Denial-of-Service* classes are subsets of both *Relaxed Cyber-Warfare* and *Relaxed Denial-of-Service* sub-classes. The relaxed classes are highlighted in blue. Protégé and the automated reasoner also automatically deduced the findings presented in statements 7.11, 7.12 and 7.13.

The relaxed scenarios are shown as a diagram in Figure 7.3. In Figure 7.3a, the *Cyber-Warfare* scenario is shown with all the sub-classes greyed out that cannot be accurately

determined. Similarly, in Figure 7.3b the *Denial-of-Service* is shown. Through inductive reasoning it is clear that the non-greyed-out sub-classes are the same for *Cyber-Warfare* and *Denial-of-Service* scenarios.

7.3.3 Inferring *Unauthorised Data Access*, *Industrial Espionage* and *Financial Theft* Scenarios

Since the automated reasoner deduction was shown to be correct in detail in Section 7.3.2 its deductions are trusted. The HermiT reasoner is used to determine which of the remaining relaxed scenarios are equivalent or sub-set of each other. Within the Protégé editor, the HermiT reasoner was able to infer that the *Relaxed Industrial Espionage* and *Relaxed Financial Theft* scenario are subsets of the *Relaxed Unauthorised Data Access* scenario. Since statements 7.14, 7.15 and 7.16 is defined as follows:

$$UnauthorisedDataAccess \subseteq UnauthorisedDataAccessRelaxed \tag{7.14}$$

$$IndustrialEspionage \subseteq IndustrialEspionageRelaxed \tag{7.15}$$

$$FinancialTheft \subseteq FinancialTheftRelaxed \tag{7.16}$$

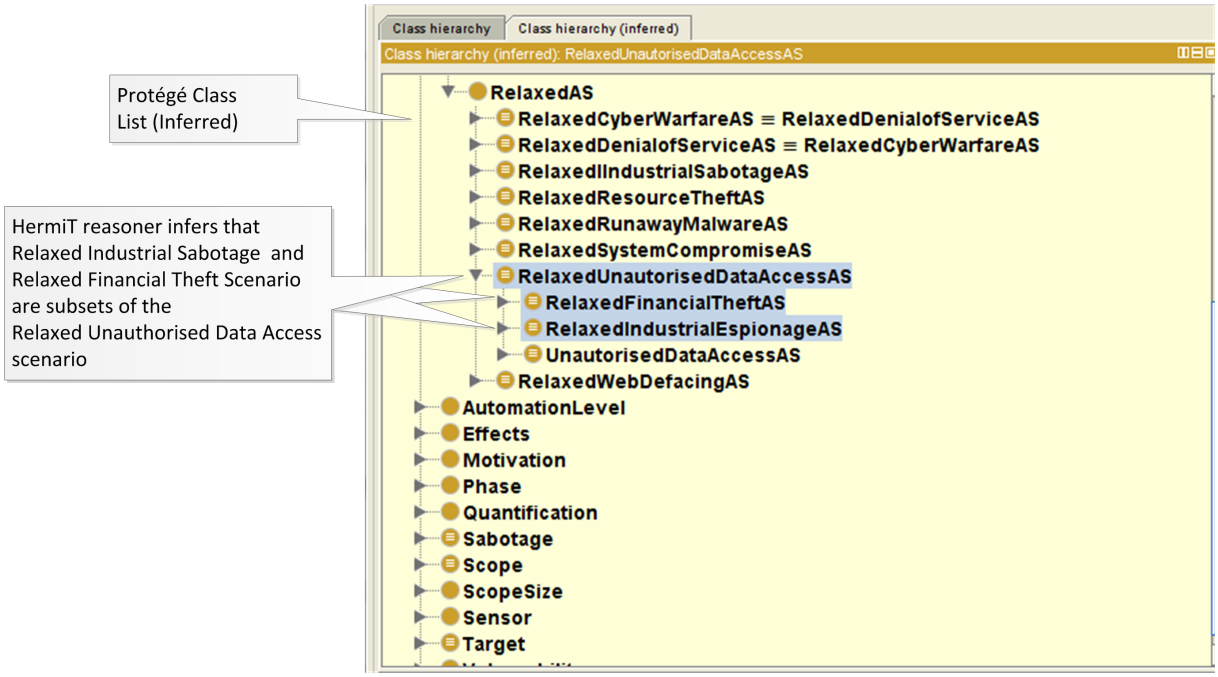


Figure 7.6: *Relaxed Unauthorised Data Access*, *Industrial Espionage* and *Financial Theft* Scenarios

It follows that the requirements for *FinancialTheftRelaxed* and *IndustrialEspionageRelaxed* is subsets of *UnauthorisedDataAccessRelaxed* scenario, and in Figure 7.6 displays how the HermiT automated reasoner was able to infer the following holds:

$$FinancialTheftRelaxed \subseteq UnauthorisedDataAccessRelaxed \tag{7.17}$$

$$IndustrialEspionageRelaxed \subseteq UnauthorisedDataAccessRelaxed \tag{7.18}$$

7.4 Summary

This chapter investigated how each class can be quantified in near real-time, and how this influences the different attack scenarios. The only classes that can be quantified to a high accuracy level were: *Attack Mechanism*, *Target*, *Asset*, *Attack Goal* and *Vulnerability*.

By relaxing the definitions of each scenario to only include these classes, some of the scenarios could be collapsed. The *Relaxed Web Defacement* and *Relaxed Denial-of-Service* scenarios are equivalent, and the *Relaxed Industrial Espionage* and *Relaxed Financial Theft* scenarios are sub-classes of the relaxed *Relaxed Unauthorised Data Access* class. These six scenarios are practically measurable in near real-time. Thus in a near real-time environment (as defined in Section 1.5), only the following scenarios can be measured to a degree of certainty (as shown in Figure 7.7):

- Denial-of-Service (and Cyber-Warfare)
- Web Defacement
- Resource Theft
- Unauthorised Data Access (and Industrial Espionage, Financial Theft)
- System Compromise
- Runaway Malware

Although the relaxed scenarios are simpler than the original scenarios, they represent how network attacks behave in a near real-time environment. The most significant deduction from these relaxed scenarios is that some of these scenarios are equivalent. For example, it was shown that the *Cyber-Warfare* and *Denial-of-Service* scenarios are not distinguishable in near real-time. This finding was proven formally, intuitively and deduced via the HermiT automated reasoner. The formal description proved that the automated reasoner can be trusted, and thus used for the remaining deductions.

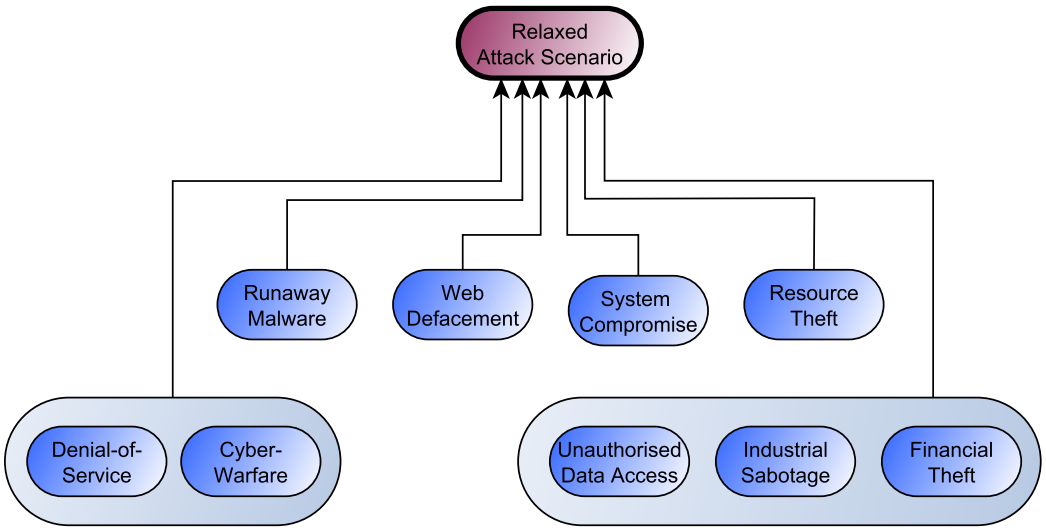


Figure 7.7: Relaxed Attack Scenarios

The goal of detecting network-based attacks is to mitigate against them. By understanding the type (scenario) of attack, correct mitigation actions can be taken. For example, the action taken against a Denial-of-Service attack differs significantly from a Web Defacement attack. Mitigation of an attack can only be effective if it is done at the start of the attack, or before significant damage has been done.

Thus in the next chapters, a prototype system was developed that only concentrates on the scenarios that can be detected in near real-time. In the next part, the prototype developed is presented. The goal of the prototype is just to verify that network attacks can be classified according to their scenario and phase, and not act as a comprehensive and complete system.

CHAPTER

EIGHT

ATTACK ESTIMATION NETWORK EVALUATION ARCHITECTURE SYSTEM

"I fear the Greeks even when bearing gifts."

Virgil, *The Aeneid*

8.1 Introduction

In the previous chapter, six attack scenarios were identified that can be detected in near-real time. These scenarios are: *Denial-of-Service, Web Defacement, Resource Theft, Unauthorised Data Access, System Compromise* and *Runaway Malware*. In Section 4.4, a temporal attack model was presented. The attack model has the following phases: Target Identification, Reconnaissance, Attack (Ramp-up, Damage, Residue) and Post Attack. This chapter demonstrates how to identify attack scenarios by mapping sensor outputs to the temporal attack model and attack scenarios. A prototype system called Aeneas is presented in this chapter.

When a computer network is under attack, mitigating actions depend on which type of attack has taken place. To take mitigating action after the attack has taken place may protect against the next attack, but it has no use during the attack. Some attack

scenarios can only be realistically determined long after an attack has reached its damage phase, such as the *Cyber-Warfare* scenario. During the attack, a *Cyber-Warfare* scenario is indistinguishable from a *Denial-of-Service* scenario. Thus the Aeneas system only tries to detect the relaxed scenarios, which were identified in Chapter 7. The Aeneas was developed specifically to provide early warning of network attacks and relevant information about the attacks.

In Section 8.2, the rationale of the prototype is explored and a high-level description of the sub-systems is shown. In sections 8.3 to 8.4.2, the main components of the Aeneas system are presented. Three sensors that provide the input data for Aeneas are presented in Section 8.5. The remaining sensors are available in Appendix C.

8.2 Design Rationale

The Aeneas demonstrates how to classify network attacks according to the most probable attack scenario in near real-time. Network attacks are classified according to their related scenario and phase. Traditional network sensors such as an IDS and a Network Telescope do not provide direct information related to the attack scenarios or temporal phases of an attack. The Aeneas uses the information of sensors by means of Event Query (EQ) to determine if a scenario-related event has taken place. The Reconnaissance, Ramp-up and Damage temporal phases map the EQs to each attack scenario. The Target Identification phase is not mapped because it takes place outside the scope of the targeted network or is indistinguishable from normal activity. The Residue and Post Attack phases are not used: Attacks that fall within these phases are rather classified as new attacks.

The Aeneas consists of four main parts: Sensors, Central Information Server (CIS), Database and Graphic User Interface (GUI), as shown in Figure 8.1.

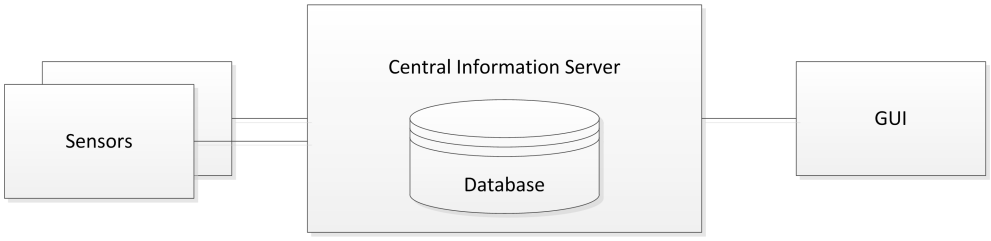


Figure 8.1: The Aeneas Prototype

The sensors provide raw information, such as network activity, unauthorised access or malicious activity detected. The sensors include systems that directly detect attacks and

systems that monitor network or user activity. They can be very simple devices that report data received directly or they can have a high level of sophistication, such as Network or Host IDSs.

The CIS has two main roles: to collate all the sensor data and to determine the state of the attacks by looking at which events have been triggered. If an event has been triggered, the CIS determines the state of an attack as follows (Figure 8.2):

- Sensors collect data from an attack. They are software based and run independently from the rest of the Aeneas system. The role of these sensors is to collect raw data and send it through with minimal processing. A sensor would typically filter data to remove the most noise before sending it to the CIS.
- The sensor data is sent to the CIS. The data is transmitted through a network via the TCP/IP protocol. The format of the sensor data is described in Section 8.5.
- The CIS determines which Event Query has been triggered and in which phase it is. By mapping sensors and their outputs to respective phases and scenarios, the CIS reports back the type and stage of detected attacks. The mapping of an Event Query and its phases is shown via two examples in Section 8.4.1.

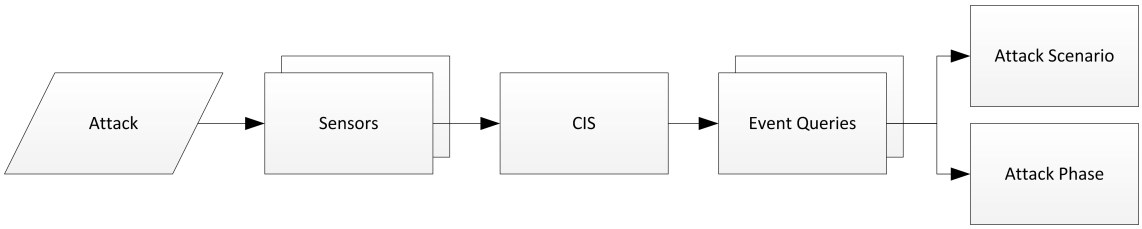


Figure 8.2: Aeneas Prototype Process

8.3 Central Information Server

The CIS has three distinct activities: collecting the data from the sensors, executing the Event Query and determining in which scenario and phase the attack has been detected. The Aeneas uses a web server to collect all the sensor data. An Apache[1] server was selected as Aeneas web server. The data is copied directly into a database. The overall design of the Threat Identification Prototype is shown in Figure 8.3.

[1] http://www.apache.org/

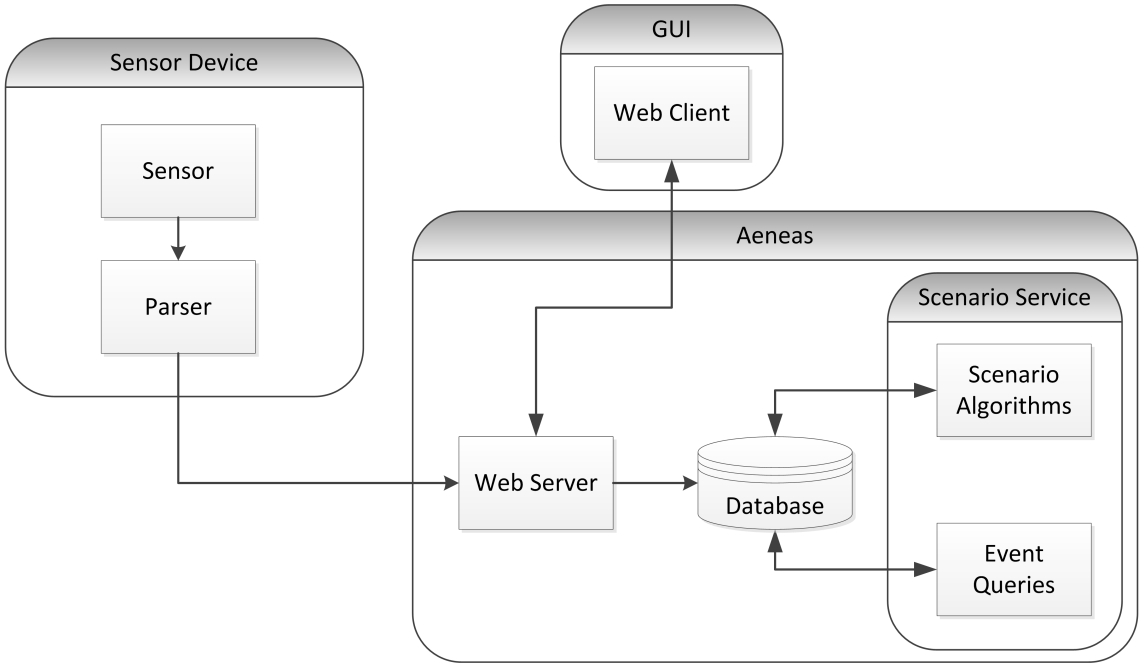


Figure 8.3: Attack Estimation Network Evaluation Architecture System

All communication from the sensors only flow from the sensors to the database: the CIS does not communicate back to any sensor. The CIS has a service that executes every five seconds. This service determines which Event Query has been triggered and which scenarios and phase combinations have been detected. Each Event Query's results are mapped to a scenario (or scenarios) and a phase, as shown in Figure 8.4.

8.4 Scenario Algorithm, Event Queries and Database

In Figure 8.4, all the EQs are shown in relation to the attack scenario and temporal phases. Each of the blocks in Figure 8.4 represent an Event Query. Each Event Query uses data from one or more sensors to identify if an attack is in the corresponding state. To determine if a specific scenario and phase have been triggered, their corresponding Event Query must be set. The ontology reasoner is used to determine which scenarios are detected by each Event Query.

8.4.1 Event Queries Example

In this section, two Event Querys will be explored in detail, namely *Unusual Web Activity* and *Port Scan*. The *Unusual Web Activity* and *Port Scan* Event Queries take place during

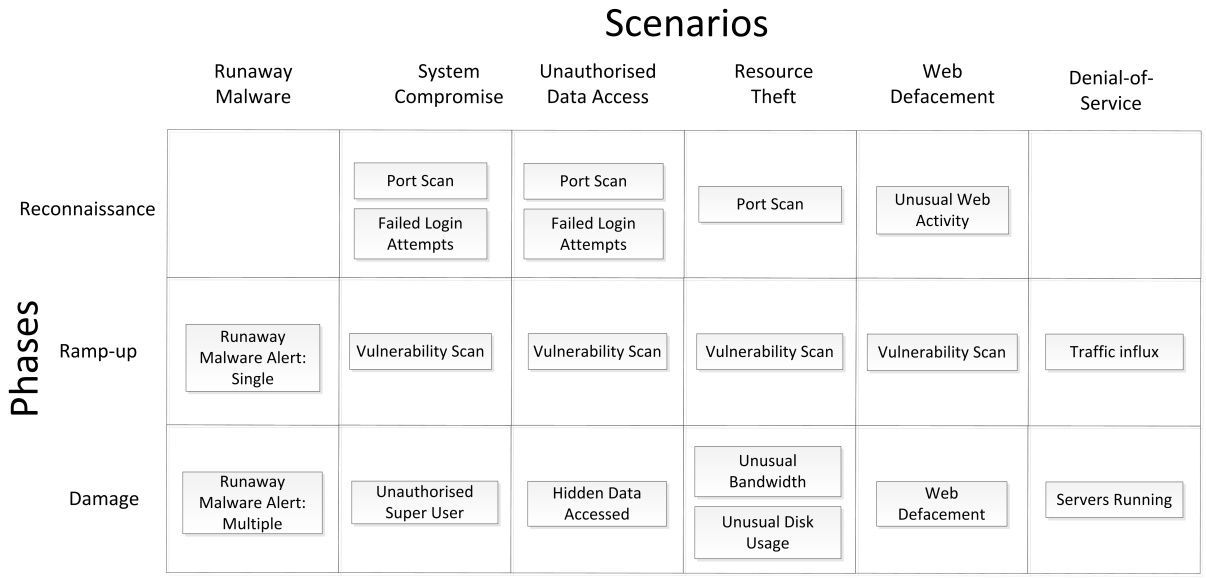


Figure 8.4: Event Queries Mapped to Attack Scenarios and Attack Phases

Listing 1 Unusual Web Activity Question

if Non-human-like activity on the website has been detected. **then**
Yes, at time T, non-human activity has been detected within a given confidence level.
else
No, at time T, no non-human activity has been detected.
end if

the *Reconnaissance* phase.

The *Unusual Web Activity* uses the Crawler Detector sensor, which detects non-human-type web page activity. This sensor is described in detail in Section 8.5.3. This Event Query (Listing 1) uses only the web-crawler sensor and is triggered directly if the web-crawler sensor detects any activity. The *Unusual Web Activity* basically asks the following question:

The same question rephrased in relation to the ontology in Chapter 5 asks the questions as follows:

> *For which* ***Attack Scenario****, during the reconnaissance phase has an* ***Attack Mechanism****, namely* ***Web Crawler****, been launched against the* ***Web Server Target****?*

Thus, for the ontology the question is asked which scenario has the limitations that the *Open Information* sub-class of the *Attack Scenario* class is specified and that the *Web Server* sub-class of *Target* class is specified. The automated reasoner (refer to Section 5.2.1) determines that only the *Web Defacement* scenario is a match. The result of the Protégé query is shown in Figure 8.5.

Listing 2 Port-scans Question

if Network port scans been detected **then**
Yes, at time T, port scans have been detected from the following IPs on these IPs and ports.
else
No, at time T, no port scans have been detected.
end if

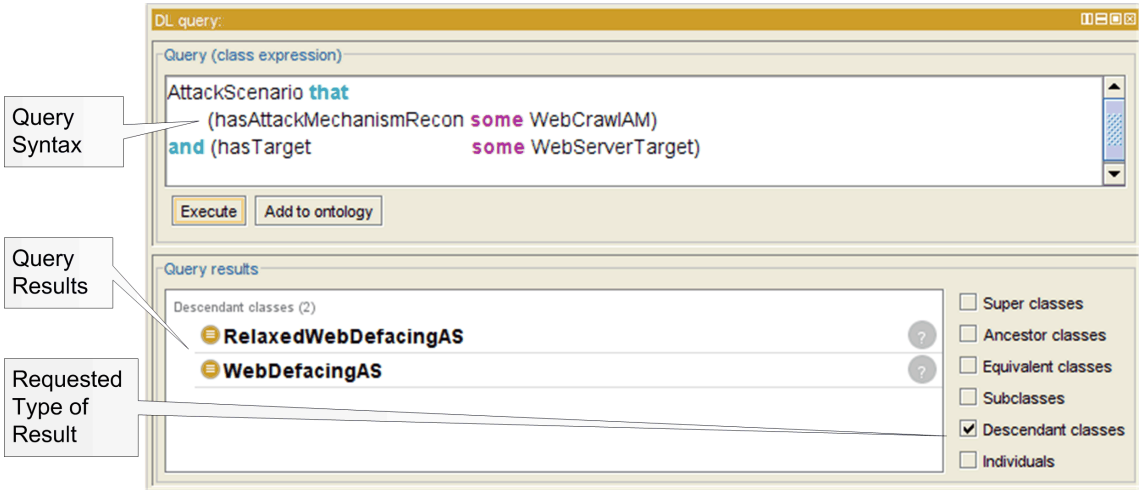


Figure 8.5: Web Crawler Scan Query Result

In the top textbox, the query that is used, is shown. The result of the query is two descendant attack scenario classes: *Web Defacement* and Relaxed Web Defacement. Dependant classes are classes that the automated reasoner determined which comply with the query.

The *Port Scan* Event Query uses a Network Telescope and a Snort IDS to detect port scans. Port scans are defined as follows(Lee *et al.*, 2003):
"... (They) consist of sending a message to a port and listening for an answer. The received response indicates the port status and can be helpful in determining a host's operating system and other information relevant to launching a future attack [p1]."

Port scans are detected when the Network Telescope detects scan-like activities. The Network Telescope sensor is described in detail in Section 8.5.1. The *Port Scan* basically asks the following question (also shown in Listing 2):

The same question rephrased in relation to the ontology (Chapter 5) asks the question as follows:

Which *Attack Scenario*, during the reconnaissance phase has an *Attack Mechanism*, namely *Open Information?*

Listing 3 Traffic Influx Event Query

```
Traffic Influx Event Query Start
Load Traffic Influx Settings
Retrieve Bandwidth Sensor Data
Calculate Bandwidth used
Calculate Bandwidth Threshold
if Bandwidth exceeds Bandwidth Threshold then
   Traffic Influx Event Query detected
end if
Traffic Influx Event Query End
```

Using Protégé, the query resulted in identifying that *System Compromise*, *Unauthorised Data Accesses* and *Resource Theft* (*Industrial Sabotage* falls outside the scope of this system and *Industrial Espionage* is the same as *Unauthorised Data Accesses*). The result of the Protégé query is shown in Figure 8.6. The reasoner determined that a port scan is an indication of a *System Compromise*, *Unauthorised Data Access* or *Resource Theft* scenario.

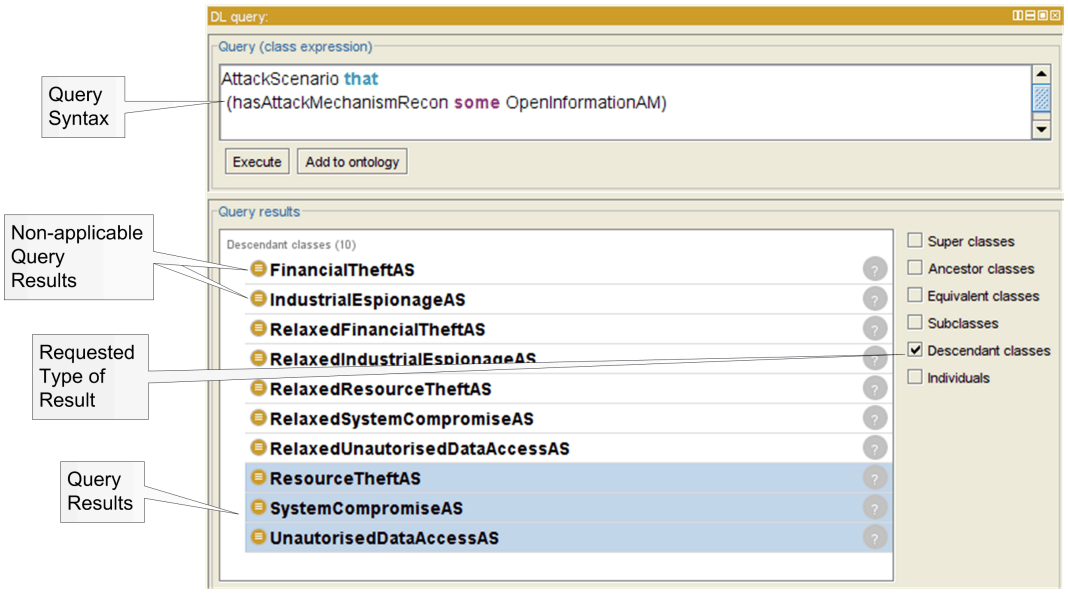


Figure 8.6: Protégé Port Scan Query Result

For example, to determine if a *Denial-of-Service* scenario is in the *Ramp-up* phase, the *Traffic Influx* Event Query will have to be triggered. The algorithm that describes the *Traffic Influx* Event Query is shown in Listing 3. The algorithm that is used to determine the state of the DOS scenario is shown in Listing 5. The *Traffic Influx* Event Query determines if the traffic volume has increased significantly and that the traffic is above a specified threshold. The *Traffic Influx* Event Query uses the SYN and bandwidth sensors.

Listing 4 Servers Running Event Query

```
Servers Running Event Query Start
Servers Running Settings
Retrieve IsAlive Sensor Data
Calculate the number of the servers are communicating
if Servers down greater than set threshold OR IsAlive Sensor stopped communicating
then
  Servers Running Event Query detected
end if
Servers Running Event Query End
```

Listing 5 DoS Algorithm

```
DoS Scenario Start
Execute Traffic Influx Event
Execute Server Running Event
if Traffic Influx is true AND Server Running is false then
  DoS Scenario detected
  DoS Scenario inRamp-up phase
else if Server Running is true then
  DoS Scenario detected
  DoS Scenario inDamage phase
end if
DoS Scenario End
```

To determine if a *Denial-of-Service* attack has reached the Damage phase, the *Servers Running* Event Query will have to be triggered. The algorithm that describes the *Servers Running* Event Query is shown in Listing 4. The algorithm that is used to determine the state of the DOS scenario is shown in Listing 5. The *Servers Running* Event Query determines if the servers have stopped communicating. When the servers cannot communicate anymore, the IsAlive sensor would communicate this fact, or if the sensor itself stopped communicating, the *Denial-of-Service* attack has reached *Damage* phase. (Refer to Appendix C.2 for detail about the IsAlive sensor.)

The algorithm that set the *Denial-of-Service* is shown in Listing 5. The algorithm checks if the *Traffic Influx* and *Server Running* events have been triggered and then sets the detected scenario accordingly.

For the rest of the scenarios, similar algorithms are executed to determine their state. These algorithms are presented in Appendix B. The process of executing the Event Query and determining which scenario and phase of the attack have been detected are shown in Listing 6.

Listing 6 Threat Identification Prototype Process

```
while true do
  for all Scenarios do
    for all Scenario's Events do
      Load Event settings
      Calculate Event information
      Execute Event logic
      Set Event state
    end for
    Execute Scenario logic
    if Scenario detected then
      Set Scenario flag
    end if
    Maintain database
  end for
  Delay 10 Seconds
end while
```

8.4.2 Database

The database of the Aeneas has to keep the relevant information for each sensor, the results of the EQs and some of the system settings. A simple database design was used to simplify the prototype. The database only has to store incoming sensor data and handle data requests from the Scenario Service. The output of the Scenario Service is also stored in the database. The database has three tables:

- LiveEventTable
- EventDetection
- Scenarios

The LiveEventTable collects all the data directly from the sensors. This table is constantly receiving new data from all of the sensors, and thus grows significantly in time. The EventDetection table stores the data of each detected event. Scenarios table store the detected scenarios and their phases. The database was implemented on MySQL[2].

8.5 Sensors

The sensors are used by the Event Query to determine if an attack scenario has been triggered. These sensors do not have to be directly related to security. Some sensors

[2]http://www.oracle.com/us/products/mysql/overview/index.html/

use security-related software and use IDSs such as Snort and Bro, while others are basic applications whose outputs can be used to determine a network state.

The sensors were developed by parsing application log files in the researcher's Standard Communications Framework (SCF). The SCF was custom-developed for the Aeneas system and is used because of its simplicity. Since the Aeneas is only proof of concept, other frameworks such as Common Event Expression (CEE)[3], Splunk[4] and similar Security Information and Event Management (SIEM) products were considered to be cumbersome. The Mosaic Security research group has compiled a list of over 80 SIEM products[5]. The SCF uses an Extensible Markup Language (XML) format and has the following data fields (also shown in Figure 8.7):

- Time
- Source IP
- Destination IP
- Source Port
- Destination Port
- Message

```
- <Event>
-     <Sensor> sensorType="[Sensor Name]">
-         <Data>
              <DataElement column="LogEventTime" varType="String">[Epoc Time]</DataElement>
              <DataElement column="SourceIP" varType="String">[IP]</DataElement>
              <DataElement column="DestinationIP" varType="String">[IP]</DataElement>
              <DataElement column="SourcePort" varType="String">[Port]</DataElement>
              <DataElement column="DestinationPort" varType="String">[Port]</DataElement>
              <DataElement column="Message" varType="String">[Event Message]</DataElement>
          </Data>
      </Sensor>
  </Event>
```

Figure 8.7: XML Schema

Dickerson and Dickerson (2000) used a similar scheme for their Fuzzy Intrusion Recognition Engine (FIRE). The FIRE system used TCP control bits, packet length and did not have fields to represent the message.

[3] http://cee.mitre.org/about/
[4] http://www.splunk.com/
[5] http://mosaicsecurity.com/categories/85-log-management-security-information-and-event-management/

8.5.1 Network Telescope Sensor

The Network Telescope's main advantage is to identify probing and other Reconnaissance activities (refer to Section 3.5.5). The Network Telescope sensor was developed by adapting multiple open-source Unix network tools and coding the required functionality. Figure 8.8 depicts how the Network Telescope sensor is constructed. The Farpd[6] software replies to any Address Resolution Protocol (ARP) request for an IP address. It matches the specified destination with the hardware Media Access Control (MAC) address of the specified destination, but only after determining if another host has not already claimed it.

Figure 8.8: The Network Telescope

Tshark[7] is a command-line network protocol analyser that is used to capture data packets. Thus, as shown in Figure 8.8, the Network Telescope consists of Farpd, Tshark and an IP filter. In the Network Telescope text file, the date, event type and source and destination IP address, source and destination port are stored. The date is represented in "%b %d %Y %H:%M:%S.%F" time format (refer to Appendix D). An example of raw data from the Network Telescope is shown below:

```
Feb 23, 2013 03:01:24.12531800:10.0.1.13;10.0.1.141;2414;3001;0x06;eth:ip:tcp
Feb 23, 2013 03:01:24.37774400:10.0.1.25;10.0.1.144;35663;3004;0x06;eth:ip:tcp
Feb 23, 2013 03:01:24.46749500:10.0.1.7;10.0.1.229;2411;3008;0x06;eth:ip:tcp
Feb 23, 2013 03:01:24.46750900:10.0.1.7;10.0.1.243;2049;3002;0x06;eth:ip:tcp
```

8.5.2 Honeypot and IDS Sensor

This sensor uses a combination of an IDS and a honeypot. The honeypot lures the attacks while the IDS classifies them. This sensor uses Honeyd[8] as the honeypot and Snort[9] for an IDS. Anagnostakis, Sidiroglou, Akritidis, Xinidis, Markatos, and Keromytis (2005)

[6] http://manpages.ubuntu.com/manpages/hardy/man8/farpd.8.html/
[7] http://www.wireshark.org/docs/man-pages/tshark.html/
[8] http://www.honeyd.org/
[9] http://www.snort.org/

introduced "Shadow Honeypots" that combined features from honeypots and anomaly detection systems. The work of Anagnostakis *et al.* inspired this sensor. Honeyd[10] was chosen as the Honeypod sensor. Honeyd is an ideal sensor for the following reasons:

- Honeyd is freely available (open-source software).
- Honeyd enables a safe FTP and Telnet simulation without endangering the host operating system.
- The FTP, Telnet and SSH simulations are configurable.

Snort is an open-source network IDS developed by Sourcefire[11]. Combining the benefits of signature, protocol, and anomaly-based inspection, Snort is one of the most widely deployed IDS technologies worldwide (Gandhi and Srivatsa, 2008). With millions of downloads and nearly 400 000 registered users, Snort has become the de-facto standard for an IDS. Snort is highly configurable and has many specialised rule sets. These rule sets can be used to detect a specific attack or attack phase. Third-party rule sets such as those developed by Bleeding Snort[12] can also be used.

The Honeyd system generates a fake target that is monitored by Snort for suspicious activity. An attacker will usually try to attack the simplest target first. The target generated by Honeyd therefore lures an attacker into the open before an attack on the real network is attempted. Figure 8.9 depicts how the Honeyd and Snort combination sensor is constructed.

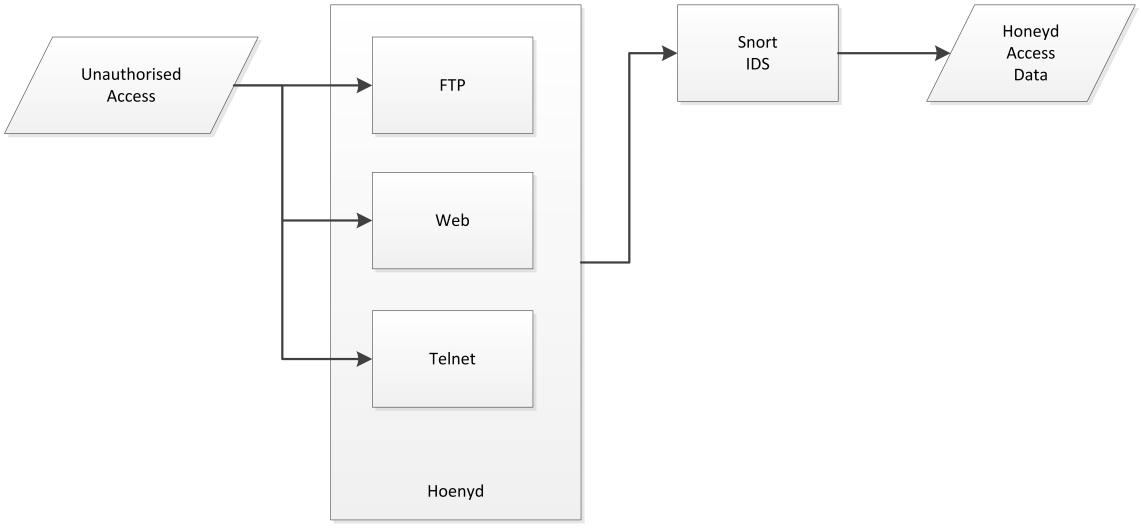


Figure 8.9: The Honey Snort Sensor

[10]http://www.honeyd.org/
[11]http://www.sourcefire.com/
[12]http://www.bleedingsnort.com/

Additionally, the Farpd[13] application is used to direct traffic into the honeypot. The sensor uses the text log file that the Snort IDS generates in response to any intrusions detected at the honeypot.

In the Snort raw text file, the date, source IP, source port, destination IP, destination port and event type are used. The date is represented in "%m/%d/%y-%H:%M:%S" time format (refer to Appendix D). The source and destination IPs are represented in Internet Protocol version Four (IPv4) human readable format. The source and destination ports are represented by an integer value. The event type is presented by a text string that indicated a Snort-generated warning. The raw output of the sensor is shown below:

```
02/25/13-13:20:25.487090 ,10.0.1.4,10.0.1.162,,,,
        "(portscan) TCP Portsweep",templogfile (END)
```

8.5.3 Crawler Detector Sensor

The Crawler Detector sensor uses a custom script to detect if any files have been accessed via a web crawler. These scripts are placed in locations that no human user would need to access. The scripts detect dictionary attacks, web vulnerability scans and crawlers that specifically target contact information. The output of these scripts are logged to a single log file. This text file is shown as follows:

```
1361772842        10.0.2.8        80        Brute Force crawler Detected
1361772925        10.0.2.8        80        Vulnerability Scan: Nikto Detected
```

Stored in the log file is the time, source IP address, source port, destination IP address, destination port and a description of the crawl event type. The time is stored in Epoch time format and the IP addresses are in IPv4 format. The web crawler sensor is classified as a host-type sensor and indicates when hidden web pages have been accessed. The algorithm used to parse to the web crawler monitor's output is shown in Listing 7.

In Appendix C, the remaining sensors are discussed. Some of the sensors were built on existing security applications such as Tripwire and Bro IDS. Other sensors used Linux log files from which security information could be deduced. The Apache web server and firewall were also used as sensors. Custom applications were also developed as sensors.

[13] http://manpages.ubuntu.com/manpages/hardy/man8/farpd.8.html/

Listing 7 Crawler Detector Sensor Algorithm

```
Crawler Detector Sensor Start
if Hidden Web Site Accessed then
  Read in Web Defacement Raw Output
  Parse Time, Event Type and Destination IP
  Parse Crawler Raw Output
  Send Parse Output to Aeneas Server
end if
Wait 30 Seconds
Crawler Detector Sensor Start
```

8.6 Summary

In this chapter, the Aeneas prototype was introduced. The Aeneas prototype consists of sensors, a Central Information Server and a GUI. The Central Information Server determines if an attack is in progress by means of an Event Query and sensor data. Each Event Query was mapped to an attack scenario and phase, thus when an Event Query is triggered, a specific attack scenario is detected in its phase. The sensor data, Event Query statuses, attack scenario and phase detected were stored and maintained in a database. Three sensors were presented in detail: the Network Telescope, Honeypot and IDS and Crawler Detector sensors. The remaining sensors are presented in Appendix C.

In the next chapters, Aeneas is validated with some empirical experimentation. The environment in which the Aeneas prototype was tested is presented in the next chapter. This environment must emulate a topical network and network attacks without endangering other networks. The Aeneas is validated by verifying each of the Event Queries.

CHAPTER

NINE

EMPIRICAL VALIDATION

"It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong."

Richard P. Feynman

9.1 Introduction

This chapter describes how the Aeneas system was verified. The goal is not to conduct statistically significant or performance experiments, but rather to look for corrective testing to verify that the principles explored in the previous chapters are valid. The Event Query defined in Section 8.4.1 and Appendix B were tested empirically.

Two main methodologies are used to test the effects of computer attacks and defence mechanisms. One approach is to reproduce the effects through simulations. This approach is useful when simulating large networks (Zeng, Bagrodia, and Gerla, 1998; Baumgart, Heep, and Krause, 2007). The second approach is to build a test bed on which real operating systems and applications are installed and used. Although this approach can only be used on a smaller scale, it presents a more accurate platform for testing. The challenge in building a test bed is to construct an environment in which tests can be

repeated from a baseline, and it should also prevent any malware from infecting other systems.

Two test beds were developed. The first test bed implemented virtual computers on an ESXi server, connected them via a firewall to the Internet and simulated user traffic via randomised traffic scripts. The goal of this test bed was to recreate a realistic network in which to test the Aeneas system using a real operating system and hardware. The second test bed used the Common Open Research Emulator (CORE) tool to simulate the network environment. The goal of the second testbed is to test how the Aeneas would perform in a stressful environment such as a DDoS attack. The second test bed used external tools such as the CORE Emulator and BreakingPoint system to generate a test environment and test data.

Within the two test environments three kinds of sensors of Event Query combinations were verified: Interrupt binary sensors, Continuous polling sensors and Interrupt information sensors. Each of the Event Queries is verified via an empirical test described in this chapter. In Section 9.2, other available test beds are presented. The design constraints of the test beds are discussed in Section 9.3, and the implementation of the test beds are presented in Section 9.4. The performance of the test beds is presented in sections 9.5 and 9.6. Three types of validation for the Aeneas system are discussed in sections 9.7, 9.8, 9.9 and 9.10. The chapter is concluded in Section 9.11.

9.2 Test beds

The test bed section of this chapter is based on work published by van Heerden, Pieterse, Burke, and Irwin (2013b). Six other test bed environments are presented, which have shortcomings that prevented their use to validate the Aeneas system.

9.2.1 Global Mobile Information System Simulator

The Global Mobile Information System Simulator (GloMoSim) is a scalable simulation library that uses parallel execution to effectively reduce the simulation time for large communication networks (Zeng *et al.*, 1998; Bajaj, Takai, Ahuja, Tang, Bagrodia, and Gerla, 1999). This simulation environment can simulate large-scale networks linked by a standard communication structure. This structure includes multicast, asymmetric communications, multi-hop wireless communications and traditional Internet protocols. GloMoSim

supports performance prediction of large-scale network models via parallel execution. Although GloMoSim provides an adequate library for testing purposes, it was not used due to the complexity associated with it.

9.2.2 User-defined and Organised Network

The User-defined and Organised Network (UDON) architecture provide Application Program Interfaces (APIs) to control virtual test bed network resources (Horib, Yamamoto, and Sekiya, 2012). This design allows the test bed to define a virtual experimental network topology, and has the ability to modify specific properties and test scenario scripts. UDON focuses on the topology of the simulated network and provides its users with the ability to modify their experimental network. Since UDON does not support simulation of larger quantities of network traffic it is not an optimal solution as a test bed for Aeneas.

9.2.3 NetSim

NetSim is a simulation architecture that supports distributed cyber-exercises[1]. NetSim consists of a collection of applications that include the simulation engine, web servers, Computer Generated Imagery (CGI) applications, MySQL databases and Perl scripts. Normal and attack conditions can be simulated. The devices represented in the simulation include workstations, routers, servers, and firewalls. NetSim provides the required functionality, but as it is aimed at being used as a cyber-exercise it was not used to test the Aeneas system.

9.2.4 Network HTTP Simulator

The Network HTTP Simulator (NHS) was developed by the University of Patras, Greece, and measures the load of HyperText Transfer Protocol (HTTP) traffic (Aravantinos, Bouras, and Ganos, 2002). It also acts as a network-stressing tool to evaluate HTTP requests and responses. The operation of NHS consists of three phases. The first phase provides the user with the ability to select simulation scenarios. The second phase creates HTTP traffic. The third phase presents the results of the simulation. NHS offers a stable platform for network simulation, but due to its singular focus on HTTP, it was not selected as an environment to test the Aeneas system.

[1] http://www.iseesystems.com/softwares/NetSimWizard.aspx/

9.2.5 Virtual Environment for Learning Networking

The Virtual Environment for Learning Networking (Velnet) provides a secure learning environment for the teaching of computer networking (Kneale, De Horta, and Box, 2004). Velnet consist of a host machine, a host operating system, VMware-based virtualisation and a virtual network. This simulation environment creates a platform aimed at teaching network architecture to students and was not designed to act as a viable attack platform to test the Aeneas system.

9.2.6 Real-time Immersion Network Simulation Environment for Network Security Exercises

The Real-time Immersion Network Simulation Environment for Network Security Exercises (RINSE) simulator supports large-scale network security preparedness and training exercises (Liljenstam, Liu, Nicol, Yuan, Yan, and Grier, 2006). The goal of RINSE is to simulate large-scale, real-time human/machine-in-the-loop network architectures with the focus on security exercises and training. The RINSE architecture consists of the following components: the network simulator, database, manager, server and network viewers. The large-scale architecture presented by RINSE is not viable to test the Aeneas system as the scope is too large.

9.3 Test Bed Design Considerations

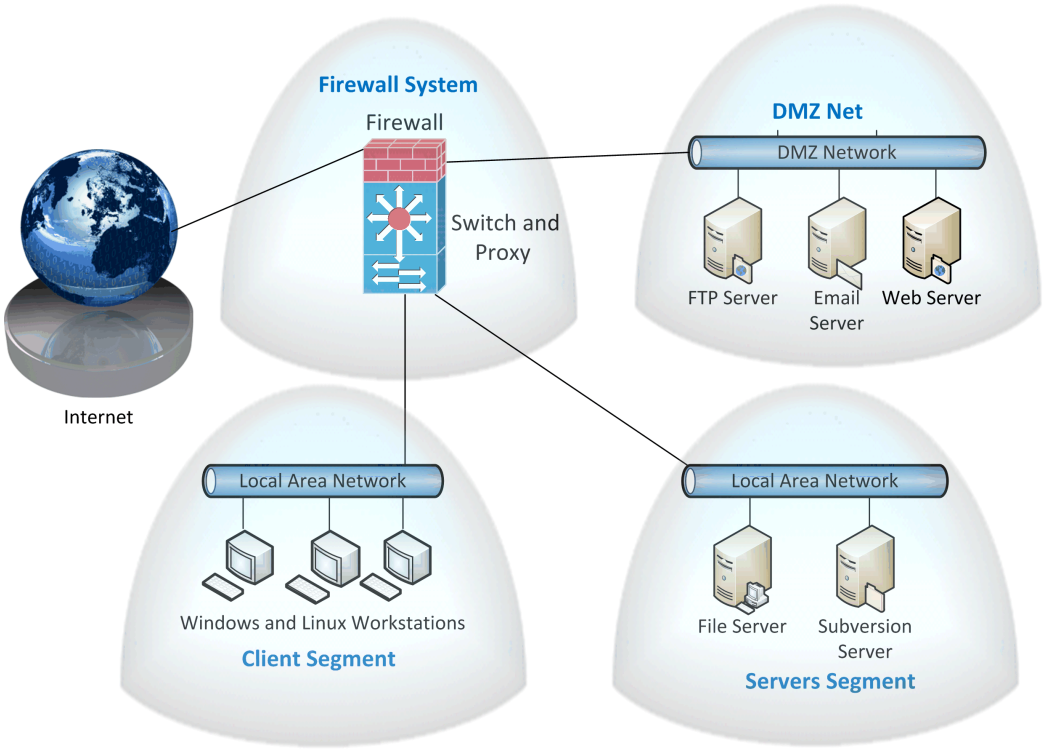
In order to test the effects of network attack scenarios, a stable re-usable platform is required. Some of the significant problems with test environments are as follows:

- Trade-off between complexity and size, also referred to as fidelity (Rosenblum, Herrod, Witchel, and Gupta, 1995). Fidelity represents how real the simulation environment is. A high-fidelity simulation emulates all the communications and computer interactions within a network, but the simulations are hardware- and processing power-intensive. Thus, for high-fidelity the number of systems emulated is low. Low-fidelity emulates the minimum network communications, but at a bigger scale. Thus with similar processing power, many more systems and their interactions can be emulated, but not with the same detail as with a high-fidelity simulation.

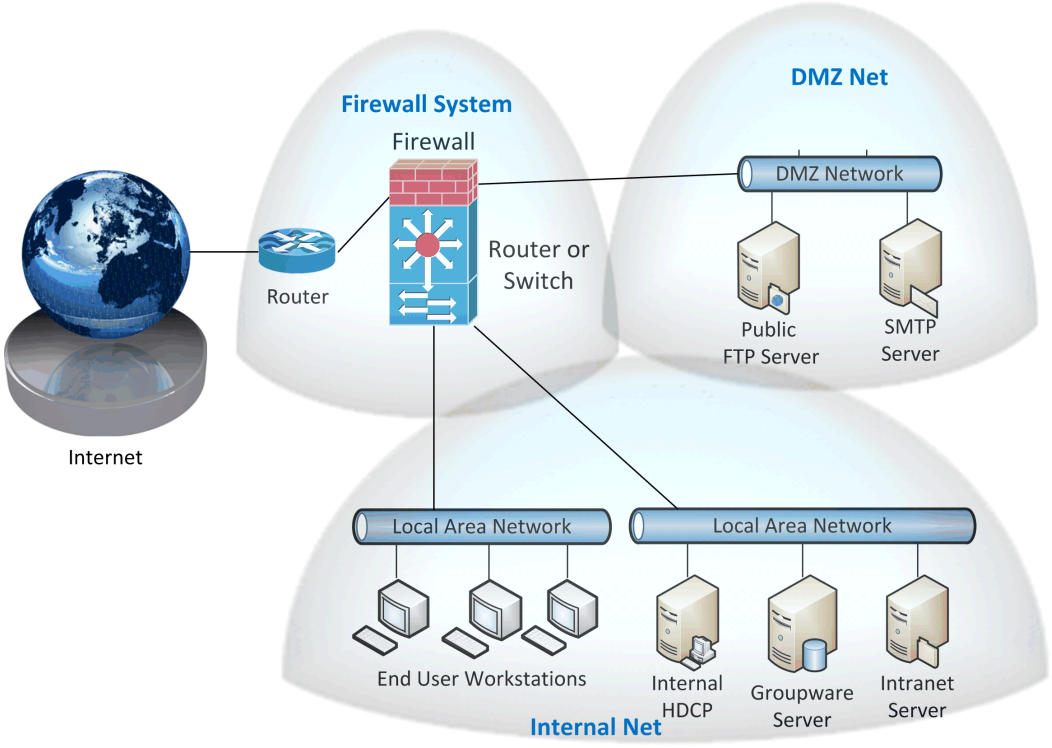
- Preventing malware from escaping and influencing the test equipment (Benzel, Braden, Kim, Neuman, Joseph, Sklower, Ostrenga, and Schwab, 2006; Cavallaro, Saxena, and Sekar, 2008). The test bed may require a connection to the Internet or Intranet to emulate such communications. While connecting to other networks, the test bed itself should not be an infector or spreading agent of malware. If a new virus/worm's influence and behaviour must be tested, the test bed must ensure that the malware does not spread to surrounding networks. Apart from the damage it can cause, the test bed tests should also not be influenced by malware from outside their domain.
- The ability to reset and redo tests from a standard baseline (Nilsson, Offutt, and Mellin, 2006; Vigna, Robertson, and Balzarotti, 2004; Kornexl, Paxson, Dreger, Feldmann, and Sommer, 2005). To ensure that a previous test does not influence the current tests, the test bed must have the ability to restore itself to a clean state (baseline). For example, a test that determines what the acceptable background network activity is must not be influenced by botnet communications from a previous test.

To test the Aeneas system, a test bed was developed that represents the environment of a small software development company. This pseudo company consists of 30 odd computer users, half of which use Unix, and the other half the Windows Operating Systems (OSs). The company has three network segments. The first segment is used by the users. The second segment is a Demilitarised Zone (DMZ) that is accessible from the Internet. The third segment is used to host the internal servers, such as a Subversion repository and File server. The company also has web, email and File Transfer Protocol (FTP) servers within the DMZ. All the segments are separated physically by means of a switch and firewall. The architecture used for the test bed is shown in Figure 9.1a. This design was based on the most common DMZ architecture (Bauer, 2001). Bauer architecture differs by separating the DMZ physically from the internal network. This is a more secure method, but as the goal of the test bed is to test for network attacks, the architecture used in Figure 9.1a was chosen over the architecture shown in Figure 9.1b.

The test bed only includes appropriate components for the purpose of validation testing. The size of the network was chosen as a compromise between complexity and the magnitude of the network. The size of the text network was constrained by the availability of hardware, but could still present an environment to test the various scenarios.



(a) Implemented Test Bed Architecture



(b) Architecture After Bauer (2001)

Figure 9.1: Test Bed Architecture

9.4 Test Bed Implementation A: ESXi and Firewall

The test bed was implemented with the following hardware:

- VMware ESXi Server[2]
- Lucidview Firewall[3]
- Network Switch
- Administrator Computer

In Figure 9.2, the physical setup is shown. The workstations and servers are implemented within the ESXi 5 server. The ESXi server has the following hardware and software:

- Memory Capacity: 192 GB
- CPU Cores: 8 CPUs x 2.659 GHz
- Processor type: Intel (R) Xeon (R) CPU E5640 @ 2.67 GHz
- Storage: 2x1 TB, RAID 1 (Mirror)
- ESXi 5.0 Operating System

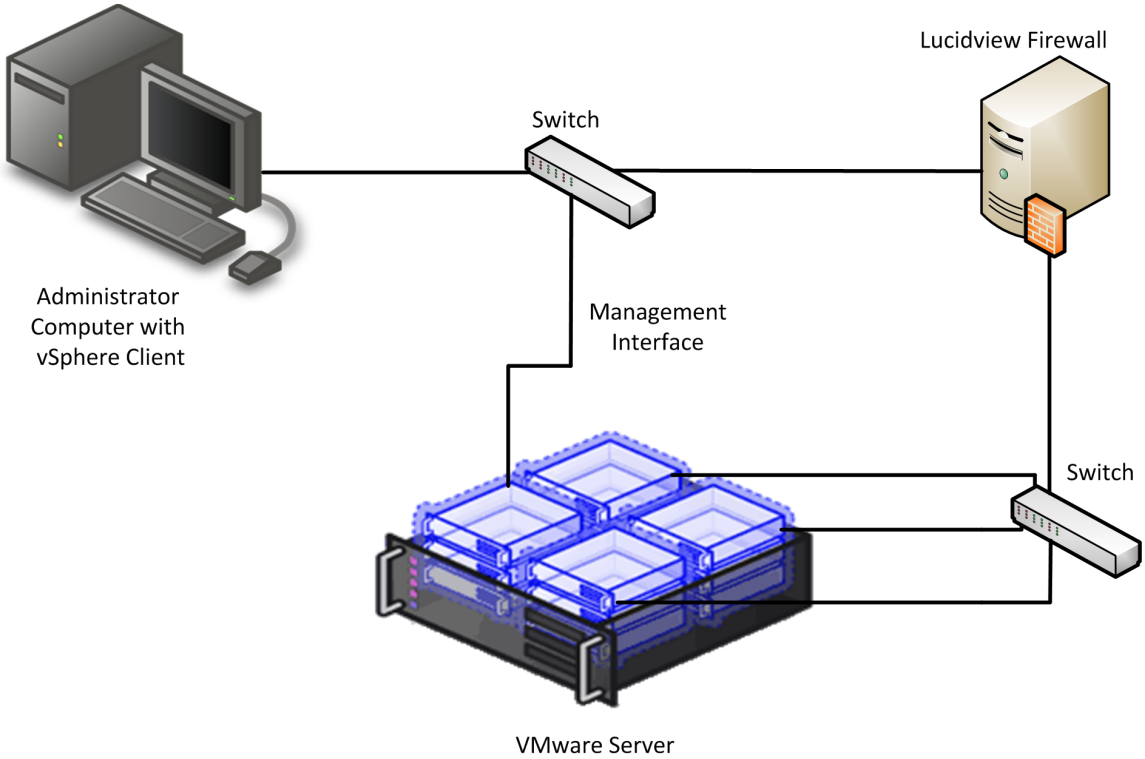


Figure 9.2: Physical Implementation of the Test Bed

[2]http://www.vmware.com/files/pdf/VMware-ESX-and-VMware-ESXi-DS-EN.pdf/
[3]http://www.lucidview.net/products-and-services/lucidview-guardian/

The ESXi server is a production-level hypervisor which abstracts computer hardware such as storage, the processor, and network and memory resources into multiple virtual computers that can each run their own unmodified operating system. The vSphere client is used to monitor and control the ESXi server's virtual computers. Through the vSphere client, virtual computers can be viewed and controlled. Each virtual machine can be restarted from a saved snapshot. Thus all the computers and servers can be restored to a base state. The ESXi can be programmed though a script to restore all computers to a base snapshot. Thus the process of restoring the system to an initial state is trivial. Snapshots can also be used to store computer and server states after an experiment. These snapshots can be exported for offline external analysis.

Each of the network segments (as shown in Figure 9.1a) are logically separated on a separate network interface in the ESXi server. The Server, Client and DMZ segments are all virtualised computers within the ESXi server. Each segment uses its own network interface and therefore all traffic between different segments will have to pass through a physical switch. When the workstations communicate with the servers, physical data packets are generated, which flow through a real switch between the virtualised computers.

The Lucidview firewall has the following main functions:

- It only allows access from the Internet to the DMZ.
- It acts as an Internet proxy for all the virtualised computers and servers.
- It protects the Internet from malware tested on the test bed.
- It logs the traffic flow between the test bed and the Internet.

The Lucidview firewall can display the amount of traffic according to preset definitions.

9.4.1 Simulated Network Traffic

The network developed in this section has a significant shortcoming: It does not have any user traffic. The only traffic visible on the network is generated by the operating system, such as update requests, Microsoft-related DNS queries, etc. No two networks are the same with respect to their network traffic (Hassan, Garcia, and Brun, 2005). The traffic has temporal, protocol, bandwidth and destination properties that need to be simulated.

A corporate computer network is used mostly during working hours with different protocols taking precedence at different times. For example, web browsing may be popular during lunch, and email in the mornings. The temporal aspect of the network traffic

is dependent on the network users or company culture. Thus the simulation should be adjustable to represent different usage temporal patterns. Network usage has a certain aspect of randomness. No two users will reply to their email at the exact same time, and browse the same websites at the same time.

To compensate for the lack of traffic, a custom script was developed that generates network traffic. This script generates traffic by performing a network request that is similar to users on the network. Traffic is generated by browsing the web, downloading FTP data, sharing files and using a repository, which is similar to human users of a network. In the following section, the traffic-generating script is presented in more detail.

9.4.2 Network Traffic Type

To simulate network traffic, the following constraints have to be addressed:

- Network Traffic Type (Protocol) (Rizzo, 1997). Traffic simulation has to handle diverse applications deployed on the Internet. These diverse applications can be simplified by the protocol used. Although there are more than 20 000 protocols used with TCP/IP networks (Touch, Lear, Mankin, Ono, Stiemerling, Eggert, Melnikov, and Eddy, 2013), the most Internet-popular ones are HTTP connections (Smith, Campos, Jeffay, and Ott, 2001).
- Temporal Variations (Deng, 1996; Kornexl *et al.*, 2005; Hernández-Campos, Karaliopoulos, Papadopouli, and Shen, 2006). Human temporal behaviour influences the type of traffic flow on the network at different times. For example, during working hours, a company's network should be more active than at night. During working hours, the behaviour should differ according to other patterns, such as relaying to email during the morning, or saving source code to the repository in the afternoon.
- Data Destination (Barford and Crovella, 1998). Data does not distribute randomly, but rather according to users' requirements. For example, in South Africa some sites that receive the most Internet traffic are: news24.co.za, bidorby.co.za, iol.co.za, gumtree.co.za and mybroadband.co.za (Enikeev, 2013).

Each network traffic type is dependent on what the network is used for and what the users require from the network. For the setup, four network traffic types were identified to represent the majority of network traffic for a small software development company (Barford and Crovella, 1998):

Listing 8 Network Simulation Algorithm

```
Start Protocol Specific Simulation
Select Destination (IP, Popularity)
Determine Pseudo Time Interval (Temporal map)
if Time Interval has been reached then
  Simulate Traffic
end if
Return to Destination Selection
```

- web browsing (Aravantinos *et al.*, 2002)
- email (Zou, Towsley, and Gong, 2007)
- subversion[4]
- FTP
- Intranet file access

Web browsing would typically be more active during lunch and after working hours, and email at the start of the working day (Karlson, Meyers, Jacobs, Johns, and Kane, 2009). Subversion is a revision control and software-versioning system. Many revision control and software-versioning applications are used today, but a single small company would typically standardise to a single system. Intranet file access should peak at the start and end of a working day. These assumptions give the test bed a more realistic flavour, but are not meant to represent definitive usage.

9.4.3 Traffic Algorithm

The algorithm in Listing 8 describes how the time interval between network activity is calculated. The algorithm requires input in the form of a protocol required, destination address, the popularity of the address and the temporal map.

The traffic algorithm is used for each required protocol that has to be simulated. Multiple destinations for the traffic can be set, and with each destination a popularity can be assigned. For example, some websites are visited more often and should therefore be more popular with the simulator. This popularity is demonstrated with the web traffic example in Section 9.4.5. The interval between simulated traffic is set with a temporal map. Each protocol temporal map differs according to its specific behaviour pattern. In Section 9.4.4, an example of the temporal map for web browsing is shown.

[4] http://subversion.apache.org/

9.4.4 Temporal Map

The temporal map describes the tempo at which network connections are made. When the temporal map has a higher value, more frequent network connections should be made. Thus the temporal map allows the simulation of different network temporal patterns according to the time of day, and time intervals differ depending on the time of the day. The pseudo time interval is determined by the value assigned for the hour interval in the temporal map. In Figure 9.3, the temporal map for web browsing is shown.

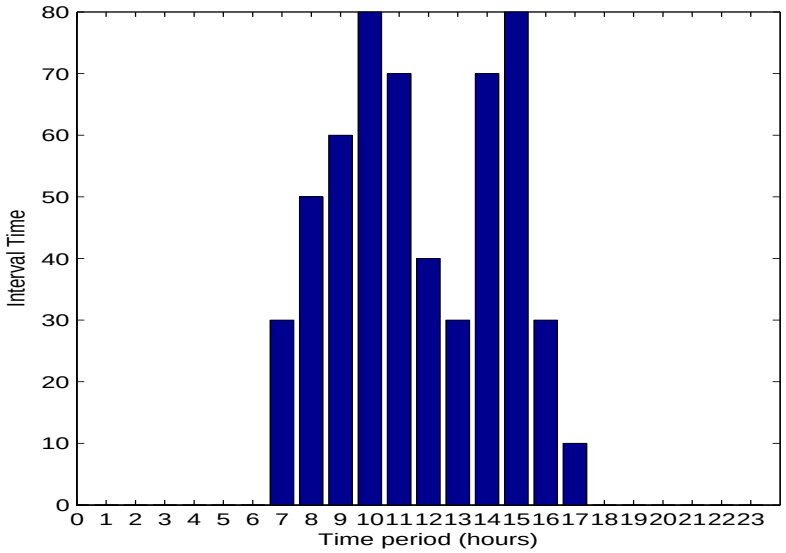


Figure 9.3: Web Traffic Temporal Map

From 20:00 to 07:00 no web traffic is expected on the network, and thus the interval value is presented as a zero. The web traffic peaks at 10:00 and 16:00, representing active browsing during tea-time and late afternoon. The pseudo random time interval is determined by using the appropriate temporal map time value, adding a random value (between -10 and 10) to it and dividing by 60 (minutes in an hour). For example, the temporal map value at 08:00 is 50 with a random number of 10 the pseudo time interval will be 1 minute. Thus after a minute, a request will be made to the web page.

9.4.5 Web Traffic Example

To simulate a reasonable representation of websites that are accessed, a list of websites and their relative probabilities were used: each time a website destination is requested,

Table 9.1: Most Visited Websites (eBizmda.com)

Rank	Domain	Unique Hits per Month	Popularity
1	Facebook.com	9,753,424	20.57
2	Twitter.com	6,471,809	13.65
3	Google.com	6,315,679	13.32
4	Youtube.com	5,213,892	11.00
5	Wordpress.org	4,010,710	8.46
6	Adobe.com	3,883,600	8.19
7	Blogspot.com	3,384,289	7.14
8	Godaddy.com	3,040,779	6.41
9	Wikipedia.org	2,987,226	6.30
10	Wordpress.com	2,349,888	4.96

a website is selected according to popularity. The list and popularity of websites were obtained at eBizmda.com (April 2013)[5]. The top 10 websites visited in 2013 (April) according to Moz.com are listed in Table 9.1.

For each of the selected protocols, a temporal map was made. Thus each protocol has its own temporal variations. Web browsing, FTP and email traffic destinations were selected according to lists. The lists contain the destination and the frequency of choosing the destination. The FTP and email lists are similar to the list shown in Table 9.1, but with arbitrary addresses constructed by the researcher. For each new request, a new destination was selected randomly according to the frequencies. For example, for each new web browsing request, Facebook will have a 20.57% probability, and YouTube will have an 11% probability to be selected. Subversion and Intranet File Access do not have multiple destinations: a single repository server and single file server were used.

9.5 Test Bed A Performance

The performance of the test bed was evaluated using simulated network traffic in the virtualised test bed. The simulation consisted of 30 corporate workstations, of which 15 used Windows and 15 used Unix. Each workstation simulated web, email, subversion and file access traffic types. Each traffic type has its own temporal map.

The ESXi server CPU load increased from an average of 47% load without simulation running to 55% load with the simulations running. Thus the CPU load by average only

[5]http://www.moz.org/top500/

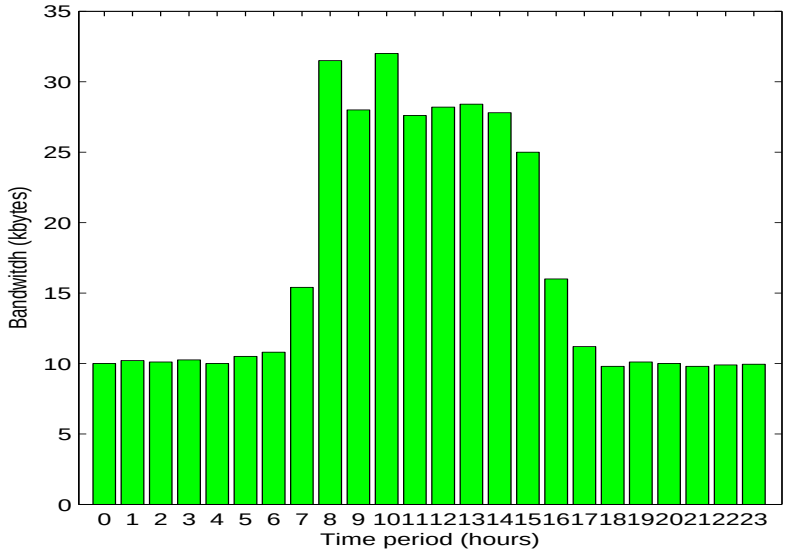


Figure 9.4: Firewall Traffic Incoming without Traffic Simulation

increases 8% when the workstations generate traffic. The ESXi memory use without simulation traffic averages 34 gigabyte usage and with the simulation 60 gigabyte usage. Thus the simulation has a significant effect on memory usage. 60 gigabyte is still significantly less than the system's 192 gigabyte capacity.

9.5.1 Firewall Data

The Lucidview firewall has the ability to log all incoming and outgoing traffic. The firewall has the ability to separate traffic according to its predefined groups. For evaluating the test bed performance, the experiment focused on the total bandwidth and not the individual bandwidth usage as the firewall logging settings were not aligned to the exact protocols that were simulated. In figures 9.4 and 9.5, the incoming and outgoing bandwidth into the test bed are shown.

In Figure 9.4, the maximum traffic is 32 kilobytes per second. Although no traffic is simulated, the traffic usage still has a form similar to the temporal maps. This shape is most influenced by other entities that try to scan or communicate with the test bed from the Internet. In Figure 9.5, the maximum traffic is 26 kilobytes per second. The outgoing traffic does not correspond to temporal maps, but rather is flatline, except for an outlier at 08:00. This represents the base state communications from the Windows and Unix workstations.

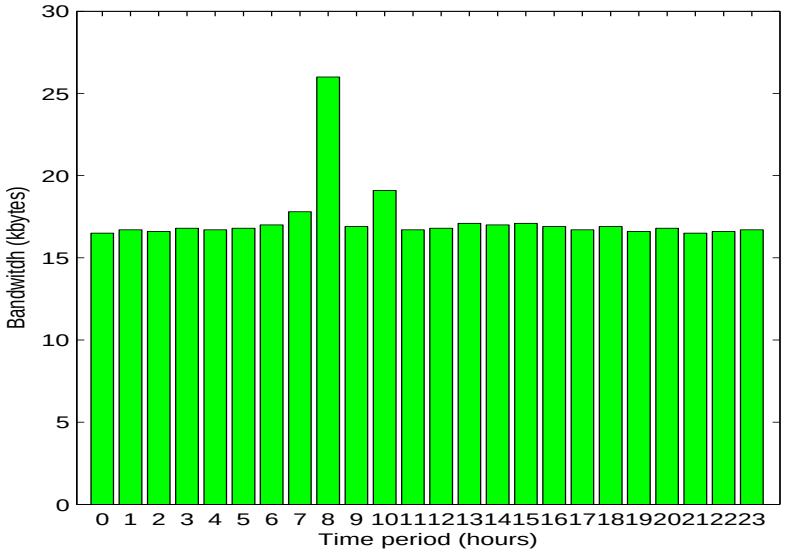


Figure 9.5: Firewall Traffic Outgoing without Traffic Simulation

In figures 9.6 and 9.7, the incoming and outgoing traffic through the firewall with the simulated traffic are shown. The red and blue bars show the simulated traffic at different dates. This illustrates how the overall pattern of the traffic is the same, but it still has subtle differences.

The incoming and outgoing traffic peaks at 350 kilobytes per second. The traffic shape is as expected, with usage during office hours much higher than outside office hours, and with a dip in usage during lunchtime. The dark blue bars have a higher maximum bandwidth usage, and although the shape differs slightly it maintained the expected shape. Thus the network simulation is successful in generating pseudo traffic.

9.6 Test Bed Implementation B: ESXi and Core Emulator

Test Bed B differs from Test Bed A by using external hardware and software to generate and shape traffic. With this test bed, the BreakingPoint system is used to generate attack-type traffic. The BreakingPoint system is designed to be used to qualify network hardware by generating increasing levels of traffic. This feature is used by the researcher to emulate DDoS and related attacks.

The BreakingPoint system can simulate millions of users' interaction and traffic up to

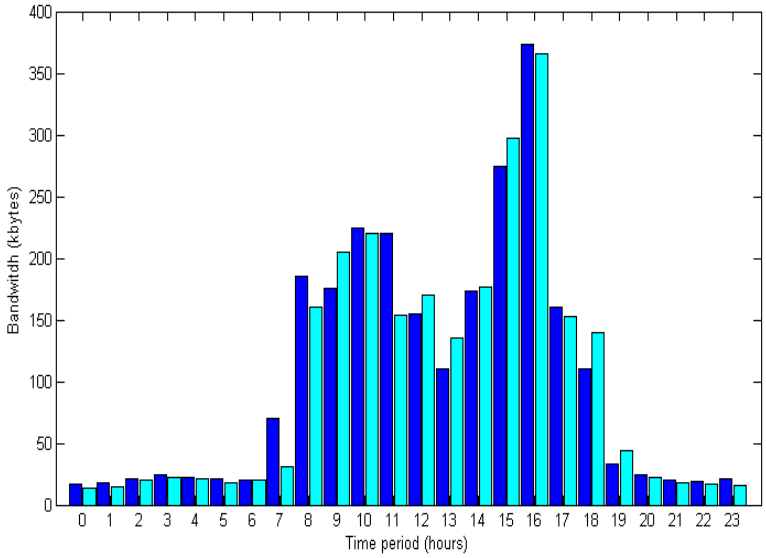


Figure 9.6: Firewall Traffic Incoming with Traffic Simulation

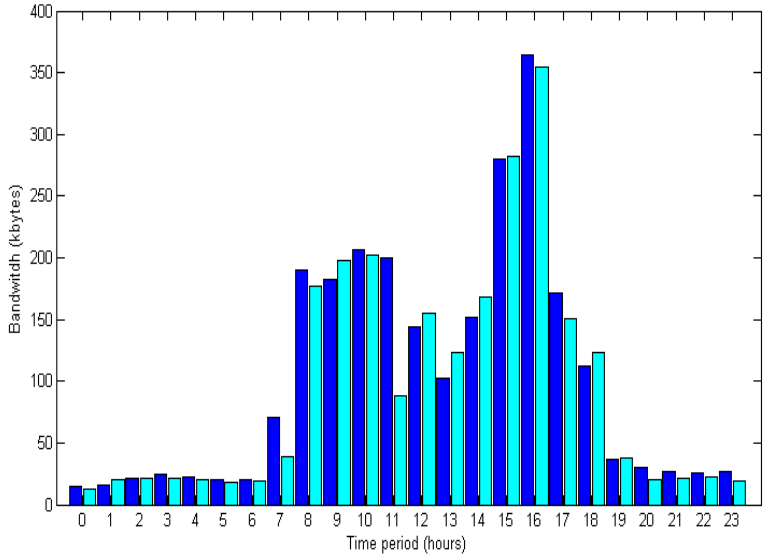


Figure 9.7: Firewall Traffic Outgoing with Traffic Simulation

1 gigabyte line speed on eight physical network ports. The BreakingPoint[6] system has the following features:

- It generates up to 4 Gbps of stateful application traffic.
- It conducts 7.5 million concurrent flows.
- It generates 2 100 SSL transactions per second.
- It has eight gigabyte Ethernet network ports.

The test bed is implemented within the CORE simulation environment. The CORE emulation environment allows for the emulation of the effect of network devices such as routers and switches in a controlled repeatable environment. CORE can also emulate workstations that respond to basic network communications such as ICMP requests. The CORE emulation environment is used due to its ease of integration and its availability to the researcher.

Test Bed B was implemented with the following hardware:

- VMware ESXi server
- CORE server
- Breaking Point server

In Figure 9.8, the physical setup for Test Bed B is shown. The ESXi server is similar to the setup of Test Bed A. The physical CORE server has the following specifications:

- memory capacity: 32 GB
- CPU cores: 8 CPUs x 2.4 GHz
- Processor type: Intel® Xeon® E5-2400 @ 2.4 GHz
- Storage: 2x1 TB, RAID 1 (Mirror)
- Broadcom 5729 Quad Port 1 GB Network Interface Card (NIC)
- Ubuntu 13.04 operating system

The CORE emulator software is used on these servers. This software has been developed by the Boeing Research and Technology division (also known as Boeing Phantom Works) (Ahrenholz, Danilov, Henderson, and Kim, 2008). The CORE emulator is freely available from the Naval Research Laboratory (NRL)[7]. This emulator is a tool that enables network emulations on a single or multiple computers. These simulations can also be connected to live networks, such as the Aeneas system for this test bed.

[6] http://www.ixiacom.com/pdfs/datasheets/ds-4-port-storm.pdf/
[7] http://downloads.pf.itd.nrl.navy.mil/core/

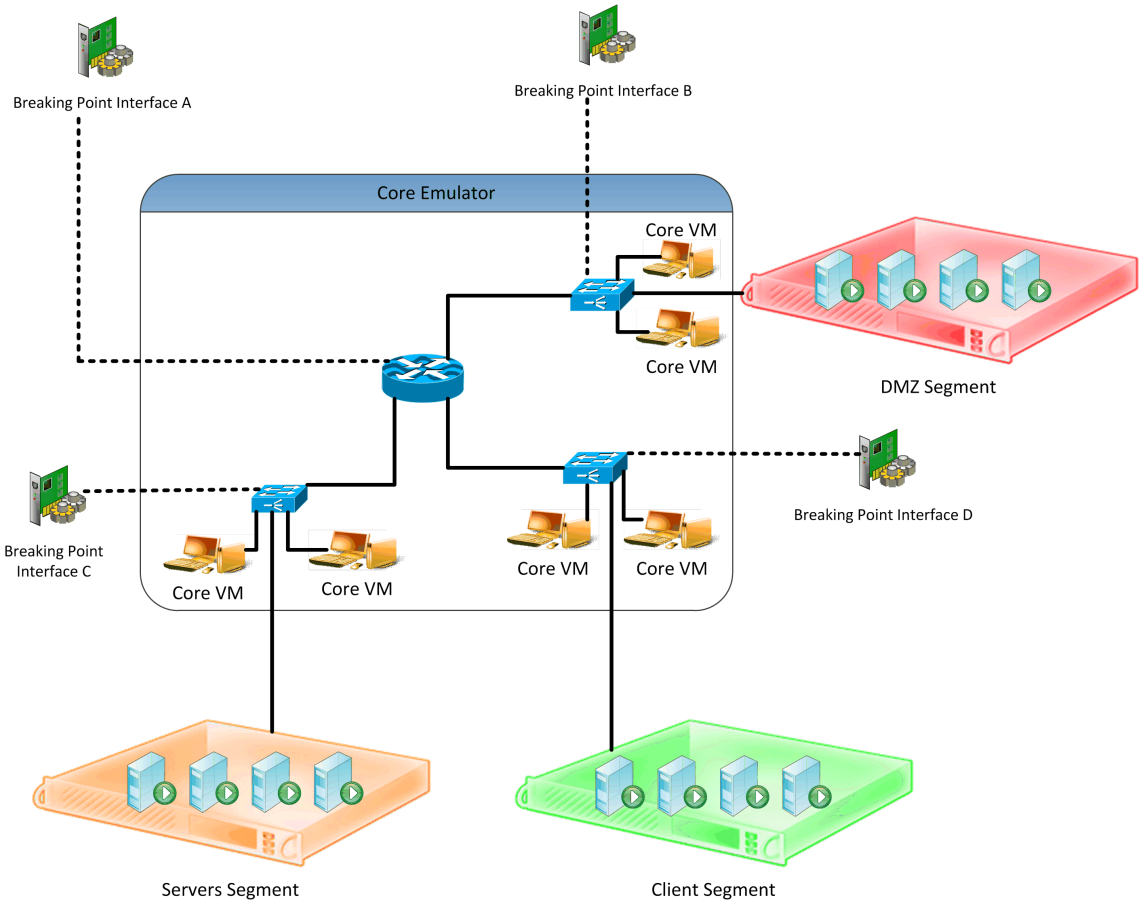


Figure 9.8: Core Emulator and ESXi Test Bed

The CORE server emulates a router, three switches, Core virtual computers and their connections, as shown in Figure 9.8 under the blue label. The ESXi server with its three segments (servers, workstations and DMZ) are connected via their own network interface to the CORE emulator. Thus, for network traffic to flow between the computers from different segments of the ESXi server, the data will have to flow though the CORE emulator.

The CORE emulator display with BreakingPoint connected to Interface C is shown in Figure 9.9. Within the CORE display, new network connections and network devices can be emulated. The setup in Figure 9.9 was used for Test Bed B experimentations, with the BreakingPoint interface moved to where it was required.

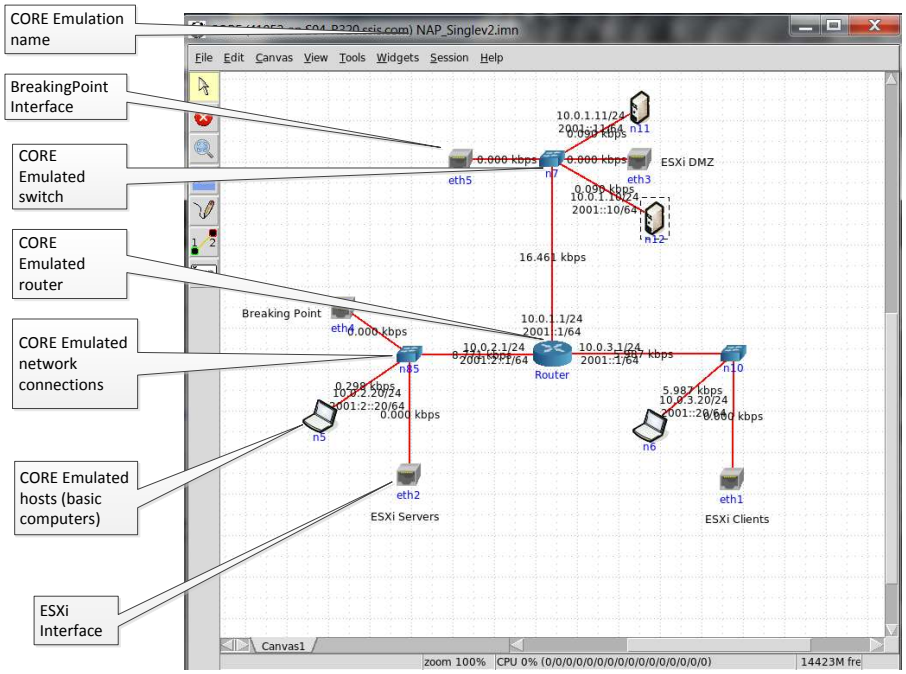


Figure 9.9: Core Emulator Display

Test Bed B can be set up to launch the network attacks from four possible connection points. These connection points are shown as BreakingPoint Interface A to D, in Figure 9.8. The server segment within the ESXi server has production servers and sensors. This segment is shown in orange in Figure 9.8. The user workstations are within the client segment of the ESXi server (green segment in Figure 9.8). The email, web servers and related sensors of the DMZ segment are hosted by the ESXi server DMZ segment and are shown in red in Figure 9.8.

9.6.1 Traffic Simulation

User traffic is simulated similar to Section 9.4.1, without the temporal map. One shortcoming is that the simulated Internet traffic will not be viable, since Test Bed B is not connected to the Internet, but the traffic generation abilities of BreakingPoint compensates for it.

The BreakingPoint can simulate typical user traffic or specialised network attacks. In Figure 9.10, the main setup screens for BreakingPoint are shown. With BreakingPoint, the user can set a source and destination IP range, the transmission tempo and type of network data profile. The network data profile can be set up from the application to

data OSI. In Figure 9.10a, the network setup for the BreakingPoint system is shown. It generates data from the client in Figure 9.10a and sends them via the "Device Under Test" to the server. The "Device Under Test" is the test bed. If a device has the same IP as the server, it will try to respond to the traffic generated by BreakingPoint. For Figure 9.10a, BreakingPoint will attempt to transmit data from the client side, using the 10.0.3.100/24 source IP addresses and the server-side destination IP address of 10.0.1.8. Thus the test bed uses 10.0.1.8 as its source IP address, BreakingPoint will generate traffic from the client side on the BreakingPoint interfaces to the test bed.

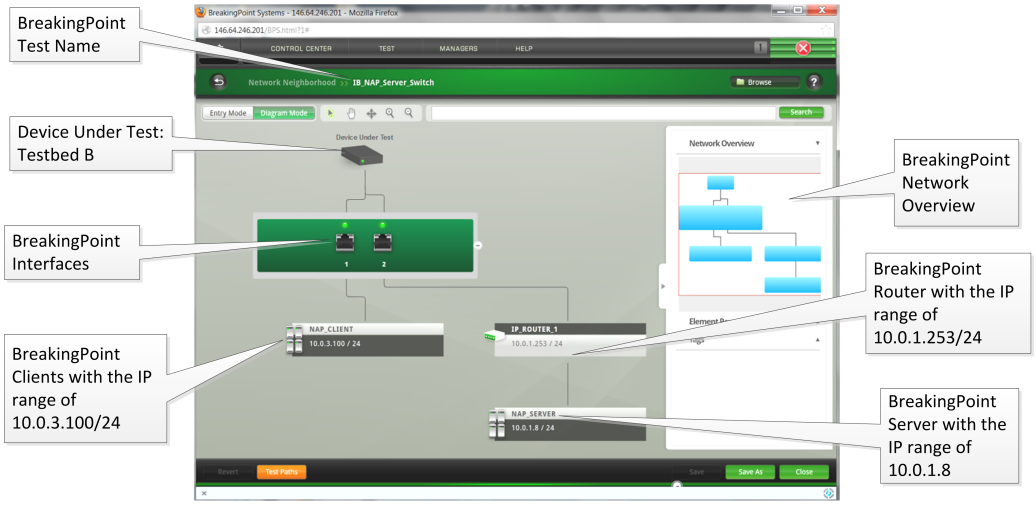
In Figure 9.10b, the type of test and its parameters are shown. The test type is "Session-Sender", which refers to test attempts to create new network connections (also referred to as network session). In this setup, the maximum number of sessions per second (flows) is set to 900 000. Each flow is a single network request (also known as a Synchronise (SYN) request). Within BreakingPoint, every detail of the emulated data can be set. For the use of BreakingPoint in Test Bed B, only the ability to generate a DDoS attack was used and the only parameter set was the number of sessions requested per second.

In Figure 9.10c, the temporal profile of the test is shown. The top part of 9.10c displays with a graph how the number of sessions per second will increase and decrease according to a pattern set in the bottom part of the Figure. In this test, the tests increase the frequency of the attack every six seconds until a maximum of 900 000 flows is reached at 60 seconds. After 120 seconds, the test is gradually slowed down, until it stops at 150 seconds.

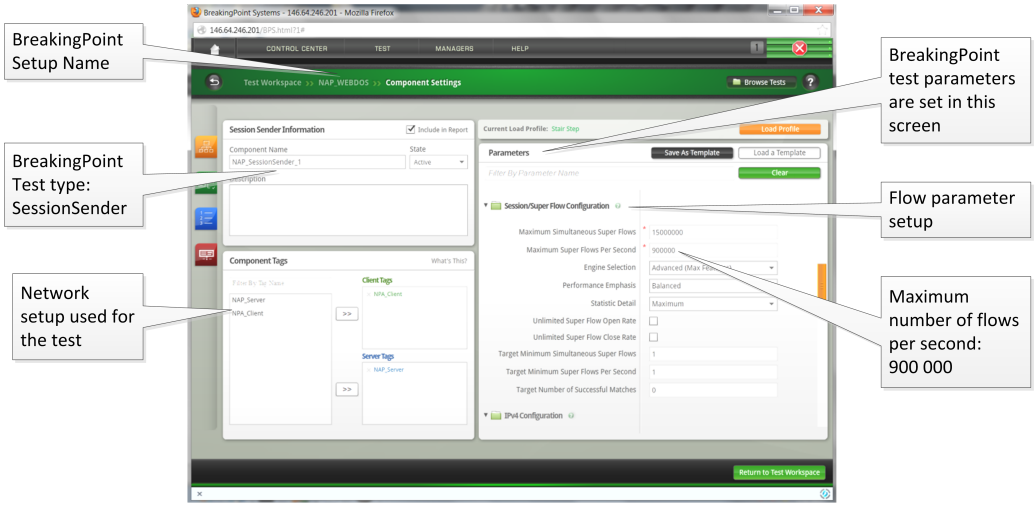
9.7 Validation

The aim of this research is not to try and build a prediction system, but to validate the ontology and attack scenarios. In this section, empirical experimentation to validate the Aeneas system is presented. Each of the Event Queries is validated in sections 9.8 to 9.10.2 by presenting single examples or test cases only. The Event Query used three main types of sensors:

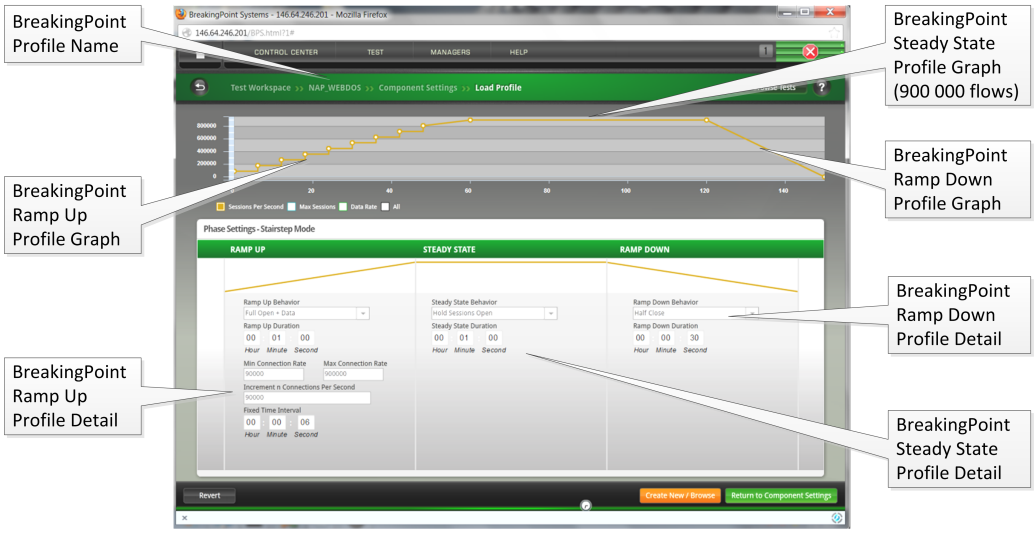
- Interrupt binary sensors. These are sensors that only send data if an event has been triggered. They usually map directly to an Event Query and to a specific phase and scenario.
- Continuous polling sensors. These sensors continuously provide data, and by integrating their output, Event Query could be triggered.



(a) BreakingPoint Network Neighbourhood Diagram Screen



(b) BreakingPoint Session Test Setup Screen



(c) BreakingPoint Session Test Profile Screen

Figure 9.10: Breaking Point Screens

- Interrupt information sensors. These sensors only provide data after an event has been triggered, but the data still has to be integrated by an Event Query before phase and scenario can be triggered.

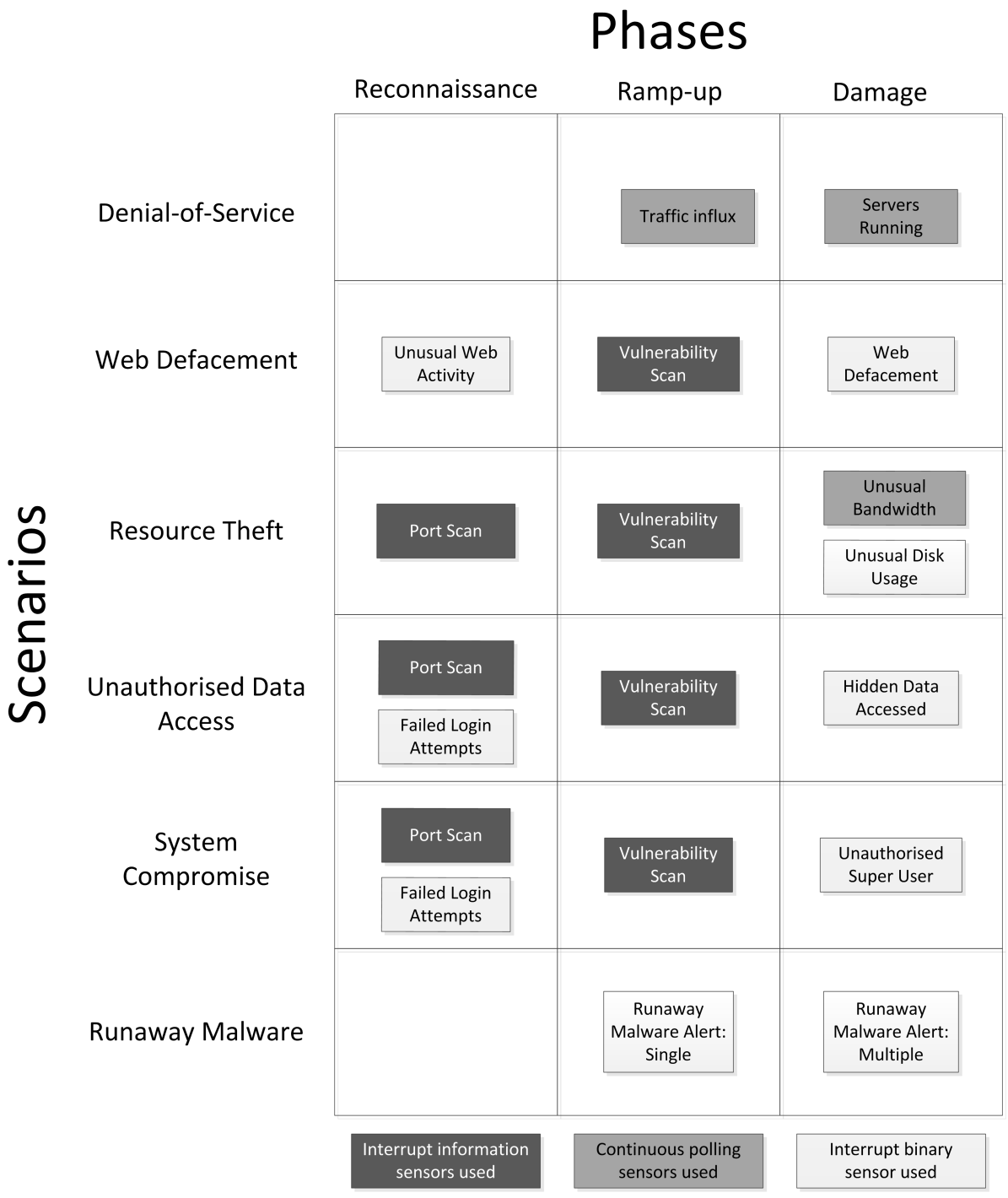


Figure 9.11: Three Types of Sensors Used with Event Queries

Each Event Query sensor type is shown in Figure 9.11. The experiments discussed in this chapter were not intended to be an exhaustive test of the Aeneas system, but rather a small sample set of tests designed to prove that the prototype works in principle. Each of the Event Queries shown in Figure 9.11 are verified only by one test, thus proving that the mapping of sensor and Event Queries to scenario and phase works. The sensors and the Event Queries can be optimised and expanded significantly in the future to build an extensible and robust system.

The experiments were chosen to present realistic attacks on the testbeds. The attacks used in the experiments range from manual hacking (as used in Section B.9), automated tools (Section 9.8.1) to crafted malware (Section 9.8.6). By generating examples of attacks, the Aeneas system could be experimentally verified, and thus by inference the network attack ontology as well.

9.8 Event Queries that use Interrupt Binary Sensors

Some of the sensors only send data if an event has been triggered. These sensors usually map directly to an Event Query and to a specific phase and scenario. Thus if an interrupt binary sensor is triggered, the corresponding Event Query should be triggered directly.

In this section, sensors, Event Queries, phases and scenarios that are directly related were tested. In these tests, interrupt sensors were tested on Test Bed A and Test Bed B. Both test beds delivered the same successful results.

9.8.1 Unusual Web Activity

By scanning the test bed website with a web crawler, the Unusual Web Activity Event Query is triggered. This Event Query is triggered directly by the Unusual Web Activity sensor. The Web Crawler Security[8] software was used to scan the test bed website. This test resulted in a successful indication that a *Web Defacement* attack scenario is in the *Reconnaissance* phase.

[8] http://sourceforge.net/p/webcrawler-py/wiki/Home/

9.8.2 Failed Login Attempt

By using incorrect usernames and passwords while trying to log into one of the production servers in the test bed, the *Failed Login Attempts* Event Query was triggered. This Event Query is triggered directly by the failed login sensor. The attempted login was done from a workstation with Secure Shell (SSH). The test resulted in a successful indication that the *System Compromise* and *Unauthorised Data Access* attack scenarios are in the *Reconnaissance* phase. Since it is not possible to determine at this stage of the attack which of the two scenarios are happening, both were expected to trigger.

9.8.3 Unauthorised Super User

By gaining remote access to an administration server, the *Unauthorised Super User* Event Query was triggered. This Event Query is triggered when an administration user (also known as root) logs into the administration server. This server should only be logged into from its own terminal (also known as localhost). Any remote logins indicated that the system has been compromised. The remote login was done from a workstation computer with SSH using a correct username and password. The test resulted in a successful indication that the *System Compromise* attack scenario has entered the *Damage* phase.

9.8.4 Hidden Data Accessed

The production server has a hidden directory. By accessing this hidden directory, the *Hidden Data Accessed* Event Query was triggered. This Event Query is triggered after access has been obtained to a directory that should never have been accessed, and thus the data is already compromised. The hidden data was accessed from a workstation with SSH using a correct username and password. The test resulted in a successful indication that the *Unauthorised Data Accessed* attack scenario has entered the *Damage* phase.

9.8.5 Web Defacement

When any part of the web page (apart from forums) was changed, the *Web Defacement* Event Query was triggered. The web page is changed by manually editing the website Index.html page. This event is triggered independently from the methodology used, for

example if the page was altered via SQL Injection, the Web Defacement sensor would still be triggered. The test resulted in a successful indication that the *Web Defacement* attack scenario has entered the *Damage* phase.

9.8.6 Runaway Malware

The *Runaway Malware* EQ is triggered by the *Network Telescope* sensor. The *Runaway Malware: Single* gets triggered when the *Network Telescope* sensor is triggered from a single source, and the *Runaway Malware: Multiple* is triggered when the *Network Telescope* sensor is triggered from multiple sources. The test resulted in a successful indication that the *Runaway Malware* attack scenario has entered the *Ramp-up* and *Damage* phases.

9.9 Event Queries that Use Continuous Polling Sensors

Some sensors provide continuous data, and by integrating their output, Event Query could be triggered. Such sensors typically monitor a resource such as bandwidth. The data from the sensor must also be interpreted to determine if the corresponding Event Query is triggered. The Event Query that uses continuous polling sensors was tested on Test Bed B.

9.9.1 Traffic Influx

The *Traffic Influx* EQ uses two sensors to determine if the traffic has increased to an out-of-norm level. The Bandwidth and Connections sensors are used to determine if the amount of traffic has increased to a level at which the *Traffic Influx* Event Query should be triggered. The Bandwidth sensor measures the amount of network traffic, and the Connections sensor measures the amount of SYN packets. These sensors are placed within the Servers and DMZ sections of the test bed and are able to monitor all incoming traffic because the ESXi interfaces were set to promiscuous mode.

BreakingPoint was used to generate a DDoS attack called SYN Flood (Trammell and Manning, 2009). SYN flood attacks used a flaw in the TCP protocol (Bellovin, 1989; Lau *et al.*, 2000). The TCP protocol uses a three-way handshake: SYN, SYN/ACK and ACK. This attack overwhelms a target with SYN packets without acknowledging the SYN/ACK

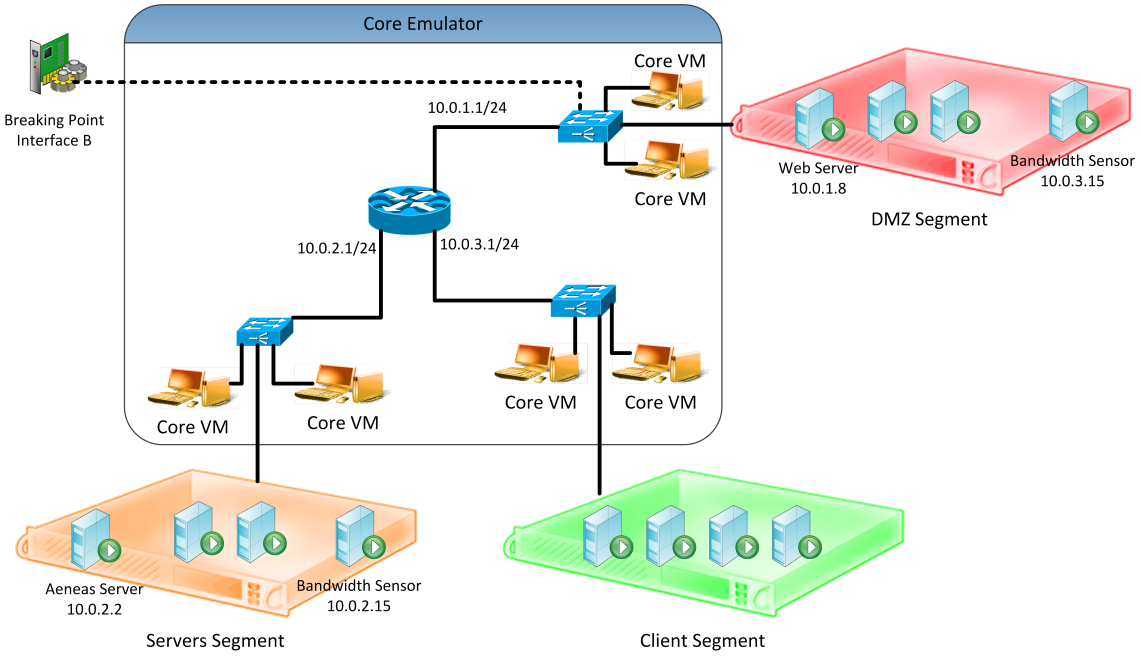


Figure 9.12: Session Attack Experiment 1

packets. Thus in effect the target network resources keep waiting for the ACK packets at a rate slower than incoming SYNs packets, in effect preventing any other legitimate network connections.

The BreakingPoint system was set up to send a maximum of 900 000 SYN requests per second. BreakingPoint setup is shown in Chapter 9, figures 9.10a, 9.10b and 9.10c. The sessions per second are increased stepwise every six seconds by 90 000 connections until a maximum of 900 000 connections. After two minutes, the attack gradually stopped.

Two experiments were conducted. In the first experiment, BreakingPoint was connected to the DMZ segment and the web server in the same segment was attacked. This represents a SYN attack from the Internet. The experimental setup is shown in Figure 9.12 and the CORE emulator display is shown in Figure 9.13. The thicker red lines represent high bandwidth of the SYN flood attack.

In Figure 9.14, the bandwidth and SYN packets bandwidth detected are shown with the first experiment. In this attack, a SYN flood attack is launched directly into the DMZ. The Event Query (EQ) is set to trigger if the number of SYN connections per five-second interval is more than 500. As shown in Figure 9.14, the bandwidth and SYN packets in the servers segments are nearly constant. This proves that the SYN flood attack was not routed to the servers segment. The DMZ segment bandwidth and SYN packets spike significantly after 20 seconds, which corresponds to the time at which the BreakingPoint

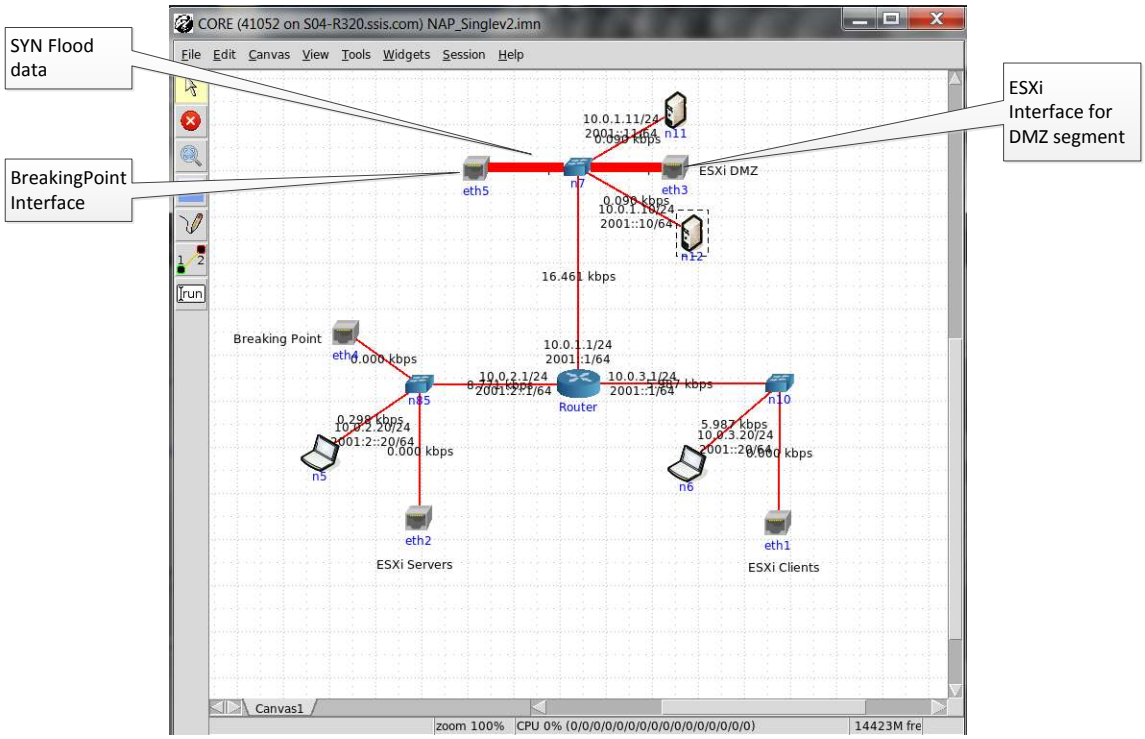


Figure 9.13: CORE Emulator Display of SYN Attack Direct DMZ

starts with its flood attack. After 40 seconds, the DMZ sensors start to fail, with only the bandwidth sensor recovering after 140 seconds. The SYN flood attack stopped the sensors from communicating to the Aeneas server. With this experiment, the *Ramp-up* phase of *Denial-of-Serive* scenario was triggered. For the detection to work, the scenario must be triggered before the sensors go offline due to the severity of the attack. The same scenario could also be triggered by increasing the bandwidth usage significantly in a short time.

In the second test, the network used setup as shown in Figure 9.15 and the CORE emulator display is shown in Figure 9.16. BreakingPoint was used to simulate a DoS attack. The Session Breaking Point attack was used. This attack simulates a DoS attack by opening an increasing number of network sessions, until the network is saturated. The attack is modelled to start slowly and to increase until a maximum session is reached. The tempo of new sessions is set within BreakingPoint and is shown in Figure 9.10c.

In Figure 9.17, the bandwidth sensors within the DMZ and servers segments are shown as red and blue lines. The bandwidth stays relatively low, but after 95 seconds, the communication between the web server (10.0.1.8) and the Aeneas server (10.0.2.2) fails. The second experiment demonstrates a limitation in the system in that the network communication failed before a *Traffic Influx* Event Query (EQ) has been detected. In

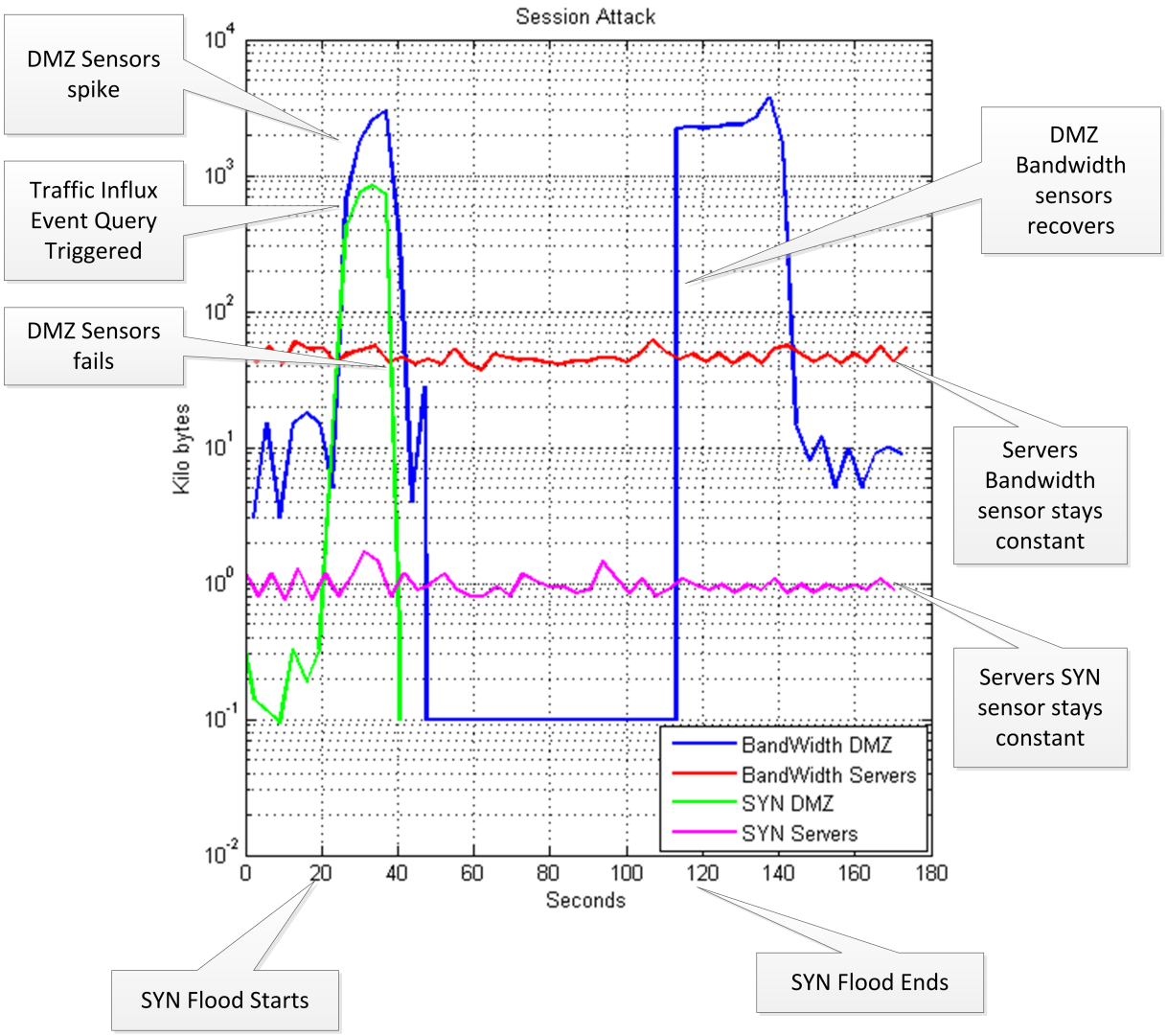


Figure 9.14: Bandwidth Measured with the Bandwidth and SYN Sensors

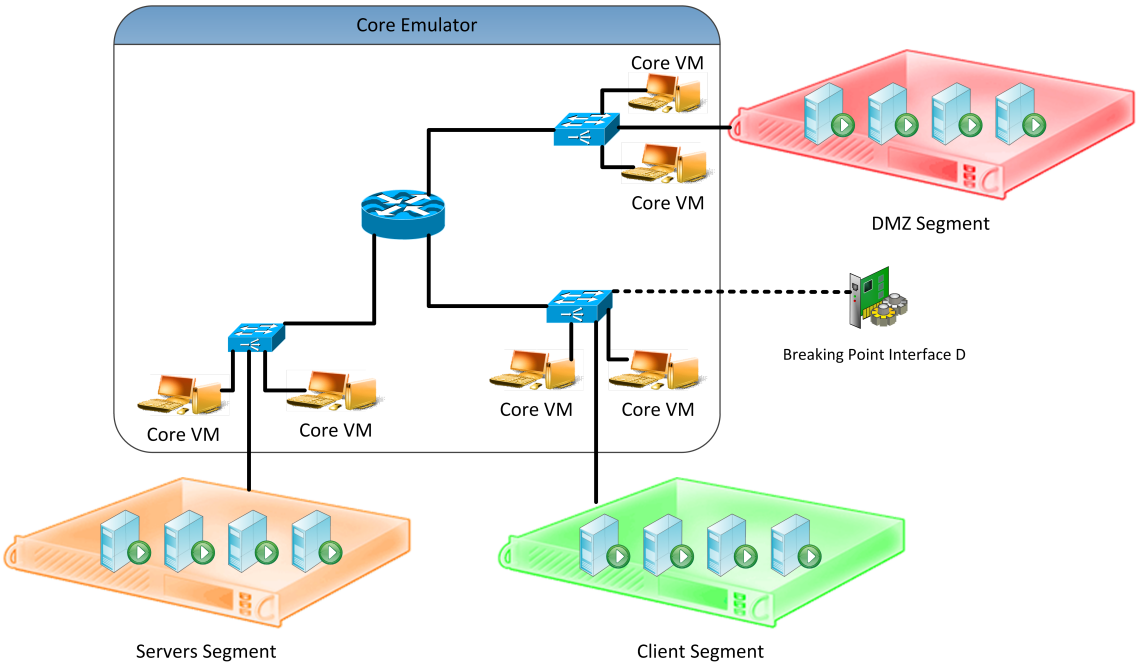


Figure 9.15: Session Attack Experiment 2

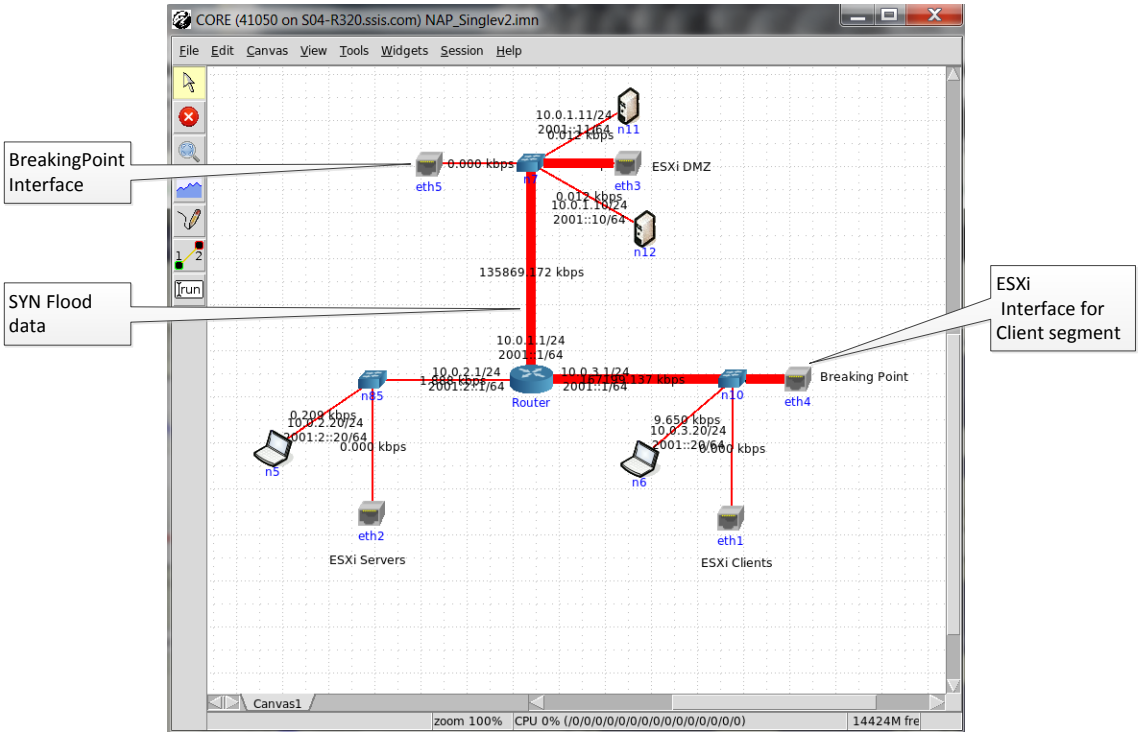


Figure 9.16: CORE Emulator Display of SYN Attack via Client Segment

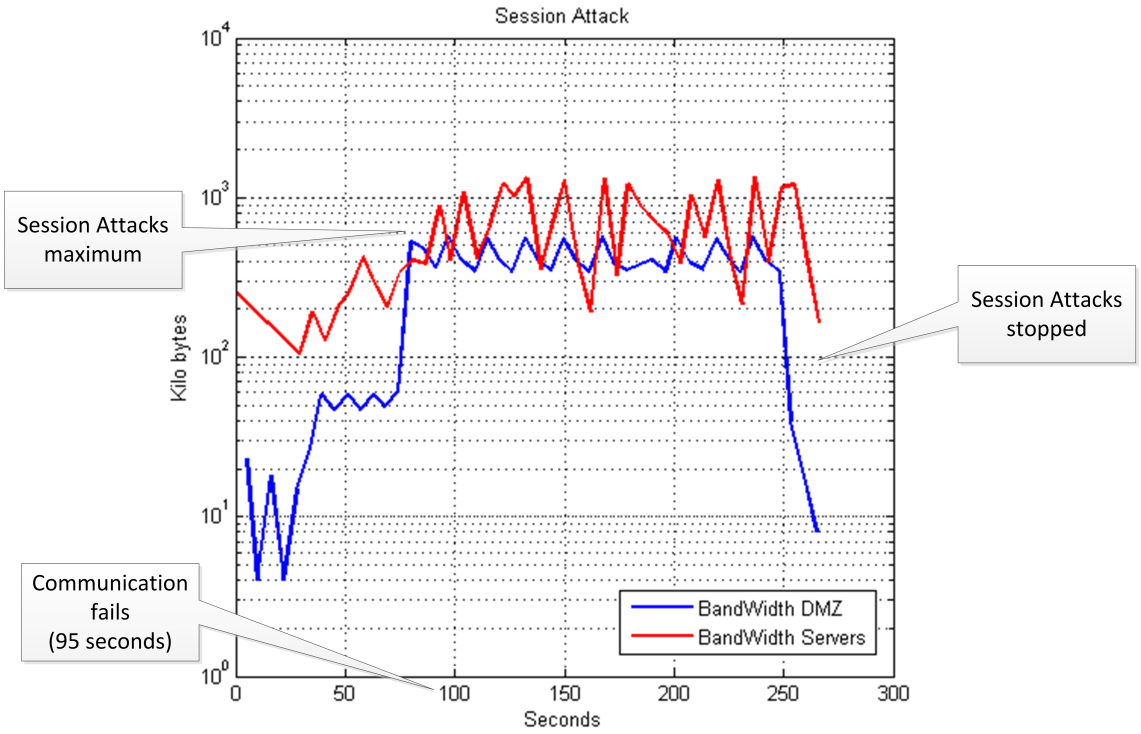


Figure 9.17: Bandwidth Measurement During Session Attack

this case, the *Servers Running* Event Query (EQ) is triggered (Section 9.9.2) and the session attack was able to trigger the *Denial-of-Service* attack scenario *Damage* phase.

9.9.2 Servers Running

As soon as one or more servers cannot be contacted (via ICMP), the *Servers Running* Event Query is triggered. The IsAlive sensor is used to determine if the servers are still communicating on the network. When one or more servers were removed from the network, the test resulted in a successful indication that the *Denial-of-Service* attack scenario has entered the *Damage* phase.

9.9.3 Unusual Bandwidth

This *Unusual Bandwidth* Event Query (EQ) used the *Bandwidth* sensor to determine if the network load is high during non-working hours. If the sensor reports bandwidth used beyond a threshold during non-working hours, the *Unusual Bandwidth* Event Query (EQ)

is triggered. The test resulted in an indication that the *Resource Theft* scenario is in the *Damage* phase.

9.10 Event Queries that use Interrupt Information Sensor

In this section, Event Query, phases and scenarios that use interrupt sensor information Event Queries were tested. These events use sensors that only communicate when data is available, and the data is to be analysed before the Event Query can be triggered.

9.10.1 Port Scan

The *Port Scan* Event Query uses the *Snort Honeyd* sensor to determine if the network is being scanned. The popular port scan tool, Nmap[9] was used to scan the test bed. The *Snort Honeyd* sensor was able to detect a port scan in normal and stealth scan mode. The test resulted in a successful indication that the *System Compromise*, *Resource Theft* and *Unauthorised Data Access* attack scenarios are in the *Reconnaissance* phase.

9.10.2 Vulnerability Scan

In this test, a vulnerability scan was launched against a SSH server (IP of 10.0.2.4). The SSH vulnerability scan was launched from a computer with the Backtrack 5 R3[10] operating system using the Metasploit[11] framework. Once the vulnerability scan has started, the Snort-Honeyd sensor starts to send data to the Aeneas server. Only after 32 036 messages does the sensor determine that a vulnerability scan is in progress. After the Snort-Honeyd sensor sent the key message, the *System Compromise*, *Unauthorised Data Access* and *Resource Theft* scenarios' *Ramp-up* phases were detected.

The same *Vulnerability Scan* was also triggered by scanning the *Snort and Honeyd* sensor with the popular vulnerability scanner Nessus[12]. Since Nessus also performs a port scan, it also triggered the *Post Scan* Event Query.

[9] http://nmap.org/
[10] http://www.backtrack-linux.org/
[11] http://www.metasploit.com/
[12] http://www.tenable.com/products/nessus/

9.11 Summary

Network attacks do not occur in a vacuum, thus two test environments (test beds) are presented in which network traffic is generated. On a sterile network it would be a trivial task to detect anomalies as any spike in traffic would indicate an attack. The simulated traffic described in this chapter simulates fairly accurate traffic volumes of a small-scale corporation without the use of complex or expensive packet simulation tools. In the first test bed, real network traffic is generated by software application within a collection of virtual computers. The second test bed used a combination of the CORE emulator and BreakingPoint systems to emulate network attacks.

The Aeneas system presented in Chapter 8 is tested by empirically verifying each of the Event Queries. The Aeneas system was able to classify attacks according to their respective scenario and phase. As expected, some attacks, such as a port scan, could not be narrowed down to a single scenario. The prototype was only tested with a limited set of attacks as the goal was not to test the system exhaustively, but to prove that the prediction methodology is viable. The Aeneas system proved that the methodology is viable and that network attacks can be classified in a near real-time environment.

CHAPTER

TEN

CONCLUSIONS

"Men in general are quick to believe that which they wish to be true."

Julius Caesar

10.1 Introduction

The primary focus of this thesis was the development, formalisation and validation of an ontology on network attacks. The ontology was developed by firstly constructing a taxonomy. This taxonomy was constructed by building from existing taxonomies by other researchers and by studying significant historical attacks (van Heerden *et al.*, 2012c). Along with the taxonomy, a temporal model was developed to describe network-based attacks. From studying significant historical attacks, ten main network attack scenarios were identified (van Heerden *et al.*, 2012b). These attack scenarios form the base of the ontology, and the other classes of the taxonomy are linked via their relationships.

The ontology was developed formally, thus ensuring the Protégé implementation and related automated reasoner deductions were correct. By utilising an automated reasoner, the attack scenarios were reduced to ones that are viable in a near real-time environment. The taxonomy, ontology and related work resulted in the publication of several international and local papers. These papers are listed in Section 1.4.

During the course of this study, a network attack prediction system Attack Estimation Network Evaluation Architecture System (Aeneas) was developed. Aeneas mapped sensor output to network attack scenarios and the network attack temporal model. The system was able to detect attacks and classify them according to the reduced set of attack scenarios and the network attack model. The viability of Aeneas was proved with empirical experimentation.

10.2 Research Review

This section recapitulates the research presented in this thesis.

Chapter 1 presented a summary of the research question and outlined the research method and research processes followed.

Chapter 2 investigated significant historical computer-based attacks. From this collection of significant attacks, the following attack scenarios were derived: Denial-of-Service, Industrial Espionage, Web Defacement, Unauthorised Data Access, Financial Theft, Industrial Sabotage, Cyber-Warfare, Resource Theft, System Compromise, and Runaway Malware.

Chapter 3 presented a literature study of models, taxonomies, ontologies and sensors related to network attacks. The models were presented as flow charts, the taxonomies and ontologies as class diagrams and the sensor types were listed.

Chapter 4 developed a taxonomy that presented the classes of a computer network-based attacks from both the point of view of an attacker and a defender. These classes were: Actor, Actor Location, Aggressor, Asset, Attack Goal, Attack Mechanism, Automation Level, Effects, Motivation, Phase, Sabotage, Scope, Target, and Vulnerability. The Phase class was further developed into a temporal network attack model. The model consisted of five main phases: Target Identification, Reconnaissance, Attack and Post-Attack Reconnaissance. The Attack Phase was divided into three sub-phases: Ramp-up, Damage and Residue.

Chapter 5 and Chapter 6 presented an ontology that describes the relationships between the taxonomy classes formally. The Attack Scenario class formed the base of the ontology. The ontology was presented in story form, formally and via a software implementation.

Chapter 7 investigated the quantification and possible measurements of the classes on the taxonomy in near real-time. By relaxing the scenario definition to only classes that can be quantified in near real-time, some of the relaxed attack scenario classes were found to be equivalent. The attack scenario was reduced to: Denial-of-Service, Web Defacement, Resource Theft, Unauthorised Data Access, System Comprise, Runaway Malware.

Chapter 8 demonstrated how to identify attack scenarios by mapping sensors' outputs to the temporal attack model and attack scenarios. A prototype system called Aeneas was presented.

Chapter 9 presented the environment in which the Aeneas system was verified. Two environments were presented: virtualised systems with a firewall connected to the Internet, and virtualised systems within an Internet simulator. Empirical experiments were performed that verify the prototype. Each Event Query was verified to report the correct phase and scenario. The Event Queries were grouped in three main types: Interrupt binary sensors, Continuous polling sensors and Interrupt information sensors.

10.3 Research Goals Achieved

The goals of this research were to formally define network attacks and to investigate how the detection of network attacks differs in near real-time environment.

The first goal were achieved by defining attack scenarios in a taxonomy. The relationships between the classes in the ontology were formally defined into an ontology. Along with the scenarios, a temporal model was constructed that describe attack phases. Not all of the scenarios and phases are relevant in a near real-time environment. The ontology was evaluated in the near real-time environment and the scenarios were reduced to the ones that are relevant in near real-time. The ontology was implemented within the Protégé ontology, and the HermiT-automated reasoner was used to infer which scenarios can be identified in near real-time. Attack individuals were classified according to their scenario in the ontology, utilising Protégé and the HermiT-automated reasoner.

Thus the research questions from Section 1.1 are answered:

- Different types of computer network attacks were presented in a taxonomy as attack scenarios (Chapter 4).

- Computer network attacks are defined in an ontology (Chapter 5 and 6).
- A relaxed set of computer attack scenarios that are viable in near real-time are defined in Chapter 7.

A prototype system Aeneas was developed to identify network attacks. This Aeneas system used various network-related sensors and collated their information into a central server. The sensors' data was mapped to scenarios and phases via Event Queries. The automated reasoner mapped the relationship between the network sensors and the Event Query. The Event Queries were polled to check the status of incoming network attacks. The Aeneas system was tested on two test beds, and was able to identify and classify network attacks successfully.

10.4 Future Work

The struggle between those who attack networks and those who defend is never ending. This struggle can also be considered an arms race, where the weapons (tools) used by both sides are in continuous development. When these weapons and tools become redundant, future work will entail updating the taxonomy, ontology and Aeneas.

The taxonomy developed in Chapter 4 can be expanded and updated to keep up to date with new and future developments. Since the attack and defence struggle is an eternal arms race, the taxonomy should be updated regularly. For example, some of the sub-classes of the *Attack Mechanism* class can be expanded to include the mobile computer environment. The ontology that links the taxonomy classes should also be updated when new technologies become available. The ontology and automated reasoner infer binary results — either an individual is a member of a scenario or not.

The prototype system also only reports on an attack that is in progress, but there is no confidence level. The system can be expanded to rather incorporate probabilistic results, and thus infer pseudo probabilities to which scenario and attack individuals belong.

The prototype would then also report a confidence level of detected attacks. The Aeneas prototype can be further developed into a viable network attack prediction system. This will require extensive testing on live networks so as to reduce the false positives. A significant challenge would be to verify/train the system and increase the true positive rate, and this can only be done in the case of real attacks. The approach of using BreakingPoint may be the only viable method to test via realistic attacks.

REFERENCES

Abdoli, F. and Kahani, M. *Ontology-based distributed intrusion detection system.* In *The 14th International Computer Conference (CSICC 2009)*, pages 65–70. IEEE, 2009.

Abou-Assaleh, T., Cercone, N., Keselj, V., and Sweidan, R. *Detection of new malicious code using n-grams signatures.* In *Proceedings of Second Annual Conference on Privacy, Security and Trust*, pages 193–196. 2004.

Abraham, A. and Thomas, J. *Distributed intrusion detection systems: a computational intelligence approach. Applications of Information Systems to Homeland Security and Defense. USA: Idea Group Inc. Publishers*, 5:105–135, 2005.

Ahrenholz, J., Danilov, C., Henderson, T. R., and Kim, J. H. *Core: A real-time network emulator.* In *Military Communications Conference (MILCON 2008)*, pages 1–7. IEEE, 2008.

Akinbode, K. and Longe, O. *Constructing ontologies in owl using protégé 4. African Journal of Computing and ICT*, 4(3):23–26, 2011.

Aliyev, V. *Using honeypots to study skill level of attackers based on the exploited vulnerabilities in the network.* Master's thesis, Department of Computer Science and Engineering, Chalmers University of Technology, 2010.

Almeida, M. and Mutina, B. *Defacements statistics 2010: Almost 1,5 million websites defaced, what's happening?* Online, June 2011. Accessed 2012/09/22.
URL `http://www.zone-h.org/news/id/4737`

Alperovitch, D. *Revealed: operation shady rat.* Technical report, McAfee, 2011. Accessed 2013/01/01.
URL `http://www.mcafee.com/us/resources/white-papers/wp-operation-shady-rat.pdf`

Anagnostakis, K., Sidiroglou, S., Akritidis, P., Xinidis, K., Markatos, E., and Keromytis, A. *Detecting targeted attacks using shadow honeypots.* In *Proceedings of the 14th USENIX security symposium*, volume 144, pages 129–144. 2005.

Andersen, B. *Australian admits creating first iphone virus.* Online, May 2009. Accessed 2012/12/23.
URL `http://www.abc.net.au/news/stories/2009/11/09/2737673.htm`

Anderson, J. M. *Why we need a new definition of information security. Computers & Security*, 22(4):308–313, 2003.

Anderson, J. P. *Computer security threat monitoring and surveillance.* Technical Report 215 546-4706, James P. Anderson Company, Fort Washington, Pennsylvania, April 1980. Accessed 2013/06/17.
URL `http://csrc.nist.gov/publications/history/ande80.pdf`

Antoniou, G. and Van Harmelen, F. *Web ontology language: Owl.* In *Handbook on ontologies*, pages 91–110. Springer, 2009.

Aravantinos, E., Bouras, C., and Ganos, P. *Http traffic simulation and evaluation for multiple users in intranet network.* In *EUROMEDIA – WEBTEC*, pages 63–67. 2002.

Argyraki, K. and Cheriton, D. *Active internet traffic filtering: real-time response to denial-of-service attacks.* In *Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 10–10. USENIX Association, 2005.

Arjun, K. *Key factors impacting on response time of software vendors in releasing patches for software vulnerabilities.* Ph.D. thesis, University of Southern Queensland, 2012.

Armerding, T. *The 15 worst data security breaches of the 21st century.* Online, February 2012. Accessed 2012/12/02.
URL `http://www.csoonline.com/article/700263/the-15-worst-data-security-breaches-of-the-21st-century`

Attewell, J. *Mobile technologies and learning. London: Learning and Skills Development Agency*, 2:1–19, 2005.

Axelsson, S. *Intrusion detection systems: A survey and taxonomy.* Technical Report 99-15, Dept. of Computer Engineering, Chalmers University of Technology, Sweden,

2000. Accessed 2012/12/24.
URL http://neuro.bstu.by/ai/To-dom/My_research/Paper-0-again/For-research/D-mining/Anomaly-D/Intrusion-detection/taxonomy.pdf

Aydin, M., Zaim, A., and Ceylan, K. *A hybrid intrusion detection system design for computer network security. Computers & Electrical Engineering*, 35(3):517–526, 2009.

Baba, T. and Matsuda, S. *Tracing network attacks to their sources. IEEE Internet Computing*, 6(2):20–26, 2002.

Bailey, M., Cooke, E., Jahanian, F., Myrick, A., and Sinha, S. *Practical darknet measurement.* In *40th Annual Conference on Information Sciences and Systems*, pages 1496–1501. IEEE, 2006.

Bajaj, L., Takai, M., Ahuja, R., Tang, K., Bagrodia, R., and Gerla, M. *Glomosim: A scalable network simulation environment. UCLA Computer Science Department Technical Report*, 990027:213, 1999.

Balakrishnan, N. and Sarma, M. *A perspective on the social cognition of hacker groups and the multi dimensional aspects of web defacements: A fused analysis.* Technical report, Supercomputer Education and Research Center (SERC), 2004. Accesed 2012/12/23.
URL http://www.dli.gov.in/data/HACKING_INFORMATION/

Balasubramaniyan, J., Garcia-Fernandez, J., Isacoff, D., Spafford, E., and Zamboni, D. *An architecture for intrusion detection using autonomous agents.* In *14th Proceedings of Annual Computer Security Applications Conference*, pages 13–24. IEEE, 1998.

Bania, P. *Evading network-level emulation.* Technical report, Packetstormsecurity.net, 2009.
URL http://dl.packetstormsecurity.net/papers/bypass/pbania{-}evading{-}nemu2009.pdf

Barford, P. and Crovella, M. *Generating representative web workloads for network and server performance evaluation. ACM SIGMETRICS Performance Evaluation Review*, 26(1):151–160, 1998.

Baskerville, R. *Information systems security design methods: implications for information systems development. ACM Computing Surveys (CSUR)*, 25(4):375–414, 1993.

Bauer, M. *Paranoid penguin: Designing and using DMZ networks to protect internet servers. Linux Journal*, 2001:16, 2001.
URL `http://www.linuxjournal.com/article/4415`

Baumgart, I., Heep, B., and Krause, S. *Oversim: a flexible overlay network simulation framework.* In *IEEE Global Internet Symposium*, pages 79–84. IEEE, 2007.

Beaucamps, P. *Advanced metamorphic techniques in computer viruses.* In *International Conference on Computer, Electrical, and Systems Science, and Engineering.* 2007a.

Beaucamps, P. *Advanced polymorphic techniques. International Journal of Computer Science*, 2(3):194–205, 2007b.

Beidleman, S. *Defining and deterring cyber war.* Technical report, DTIC document, 2009. Accessed 2012/12/23.
URL `http://www.dtic.mil/dtic/tr/fulltext/u2/a500795.pdf`

Bellovin, S. M. *Security problems in the TCP/IP protocol suite. ACM SIGCOMM Computer Communication Review*, 19(2):32–48, 1989.

Benzel, T., Braden, R., Kim, D., Neuman, C., Joseph, A., Sklower, K., Ostrenga, R., and Schwab, S. *Experience with deter: a testbed for security research.* In *2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM)*. IEEE, 2006.

Berghel, H. *The code red worm. Communications of the ACM*, 44(12):15–19, 2001.

Berners-Lee, T., Hendler, J., and Lassila, O. *The semantic web. Scientific American*, 284(5):28–37, 2001.

Bhide, A., Elnozahy, E. N., and Morgan, S. P. *A highly available network file server.* In *Proceedings of the 1991 USENIX Winter Conference*, pages 199–205. Citeseer, 1991.

Bhuyan, M. H., Bhattacharyya, D., and Kalita, J. *Surveying port scans and their detection methodologies. The Computer Journal*, 54(10):1565–1581, 2011.

Bishop, M. *Analysis of the iloveyou worm.* Online, May 2000. Accessed 2012/12/23.
URL `http://nob.cs.ucdavis.edu/classes/ecs155-2005-04/handouts/iloveyou.pdf`

Bishop, M. and Dilger, M. *Checking for race conditions in file accesses. Computing Systems*, 2(2):131–152, 1996.

Bock, J., Haase, P., Ji, Q., and Volz, R. *Benchmarking owl reasoners.* In *Proceedings of the ARea2008 Workshop.* Tenerife, Spain, 2008.

Boerio, J. and McCracken, S. *The emerging cyber threat landscape: 2012 and beyond.* Technical report, Information Technology Information Sharing and Analysis Center, 2012. Accessed 2012/12/23.
URL `https://www.it-isac.org/files_n/GFIRST_WhitePaper.pdf`

Boyd, I. *The fundamentals of computer hacking.* Technical report, SANS Institute, 2000. Accessed 2012/12/23.
URL `http://www.sans.org/reading_room/whitepapers/hackers/fundamentals-computer-hacking_956`

Bremler-Barr, A. and Levy, H. *Spoofing prevention method.* In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, pages 536–547. IEEE, 2005.

Brenner, S. and Crescenzi, A. *State-sponsored crime: The futility of the economic espionage act. Houston Journal of International Law*, 28:389, 2006.

Bright, P. *Anonymous speaks: the inside story of the hbgary hack.* Online, 2011. Accesed 2012/12/23.
URL `http://arstechnica.com/tech-policy/news/2011/02/anonymous-speaks-the-inside-story-of-the-hbgary-hack.ars`

Brody, R. G., Mulig, E., and Kimball, V. *Phishing, pharming and identity theft. Academy of Accounting and Financial Studies Journal*, 11(3):43–57, 2007.

Buckland, J. *10 worst cybercrimes of the decade.* Online, 2011. Accessed 2012/12/02.
URL `http://tech.ca.msn.com/photogallery.aspx?cp-documentid=27611570`

Burstein, A. *Trade secrecy as an instrument of national security–rethinking the foundations of economic espionage. Arizona State Law Journal*, 41:933–1167, 2009.

Caltagirone, S. and Frincke, D. *The response continuum.* In *Proceedings from the Sixth Annual Information Assurance Workshop (IAW'05)*, pages 258–265. IEEE, 2005.

Cardenas, A., Radosavac, S., Grossklags, J., Chuang, J., and Hoofnagle, C. *An economic map of cybercrime.* In *37th Research Conference on Communication, Information and Internet Policy (TPRC), George Mason University Law School, Arlington, VA.* September 2010.

Cardoso, J. *The semantic web vision: Where are we? IEEE Intelligent Systems*, 22(5):84–88, 2007.

Cass, S. *Anatomy of malice [computer viruses]. IEEE Spectrum*, 38(11):56–60, 2001.

Castillo, C. *Effective web crawling.* Ph.D. thesis, University of Chile, November 2005.

Catro, D. *How much will prism cost the u.s. cloud computing industry?* Technical report, The Information Technology and Innovation Foundation, 2013. Accessed 2013/10/17. URL `http://www2.itif.org/2013-cloud-computing-costs.pdf`

Cavallaro, L., Saxena, P., and Sekar, R. *On the limits of information flow techniques for malware analysis and containment.* In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 143–163. Springer, 2008.

CERT-In. *Hacking – how they do it?* Security Guideline CISG-2003-03, CERT-In, 2003. Accessed 2013/01/08.
URL `http://ittripura.gov.in/Policies/Guidelines/Security/Hacking.pdf`

Chaudhri, V., Farquhar, A., Fikes, R., Karp, P., and Rice, J. *Okbc: A programmatic foundation for knowledge base interoperability.* In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 600–607. AAAI Press, 1998.

Chen, T. and Robert, J. *Worm epidemics in high-speed networks. Computer*, 37(6):48–53, 2004.

Chen, T. M. *Stuxnet, the real start of cyber warfare? IEEE Network*, 24(6):2–3, 2010.

Cheswick, B. *An evening with berferd in which a cracker is lured, endured, and studied.* In *Proceedings Winter USENIX Conference, San Francisco*. 1992.

Choo, K. K. R. *Organised crime groups in cyberspace: a typology. Trends in Organized Crime*, 11(3):270–295, 2008.

Choo, K.-K. R. *The cyber threat landscape: Challenges and future research directions. Computers & Security*, 30(8):719–731, 2011.

Christodorescu, M. and Jha, S. *Testing malware detectors. ACM SIGSOFT Software Engineering Notes*, 29(4):34–44, 2004.

Ciampa, A., Visaggio, C. A., and Di Penta, M. *A heuristic-based approach for detecting sql-injection vulnerabilities in web applications.* In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems*, pages 43–49. ACM, 2010.

Clarke, R. A. and Knake, R. K. *Cyber war.* HarperCollins, 2011.

Cluley, G. *Conflict between russia and georgia turns to cyber warfare.* Online, August 2008. Accessed 2012/09/05.
URL `http://nakedsecurity.sophos.com/2008/08/12/conflict-between-russia-and-georgia-turns-to-cyber-warfare/`

Cohen, P. *Sony implicates 'anonymous' in Playstation network attack.* Online, May 2011. Accessed 2012/09/20.
URL `http://www.zdnet.com/blog/gamification/sony-implicates-anonymous-in-playstation-network-attack/369`

Colarik, A. and Janczowski, L. *Introduction to cyber warfare and cyber terrorism.* In *Cyber Warfare and Cyber Terrorism. Hershey: Information Science Reference.* 2008.

Conficker Working Group. *Conficker working group: Lessons learned.* Technical Report FA8750-08-2-0141, Tech. rep., Conficker Working Group–confickerworkinggroup.org, 2011. Accesed 2012/08/28.
URL `http://www.confickerworkinggroup.org/wiki/uploads/Conficker_Working_Group_Lessons_Learned_17_June_2010_final.pdf`

Constantin, L. *Ddos attacks against US banks peaked at 60 gbps.* Online, December 2012. Accessed 2012/12/21.
URL `http://www.cio.com/article/723889/DDoS_Attacks_Against_US_Banks_Peaked_At_60_Gbps`

Cowan, C., Pu, C., Maier, D., Walpole, J., Bakke, P., Beattie, S., Grier, A., Wagle, P., Zhang, Q., and Hinton, H. *Stackguard: Automatic adaptive detection and prevention of buffer-overflow attacks.* In *Proceedings of the 7th USENIX Security Symposium*, pages 346–355. 1998.

Cowan, C., Wagle, P., Pu, C., Beattie, S., and Walpole, J. *Buffer overflows: Attacks and defenses for the vulnerability of the decade.* In *Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX'00)*, pages 119–129. IEEE, Published by the IEEE Computer Society, 2000.

Crosbie, M. and Spafford, G. *Active defense of a computer system using autonomous agents.* Computer science technical reports, Purdue University, Department of Computer Science, 1995. Accessed 2012/12/24.
URL `http://www.cs.purdue.edu/research/technical_reports/1995/TR\%2095-008.pdf`

Czosseck, C., Ottis, R., and Taliharm, A.-M. *Estonia after the 2007 cyber attacks: Legal, strategic and organisational changes in cyber security. International Journal of Cyber Warfare and Terrorism*, 1(1):24–34, 2011.

Damballa. *Anatomy of a targeted attack.* Technical report, Damballa, 2008. Accessed 2012/12/24.
URL `http://www.damballa.com/downloads/r_pubs/WP_Anatomy_of_a_Targeted_Attack.pdf`

Davies, J., Studer, R., and Warren, P. *Semantic web technologies.* John Wiley and Sons, 2006.

Davis, J. *Vendors out to get* Excel *virus.* Online, July 1996. Accessed 2012/12/24.
URL `http://news.cnet.com/Vendors-out-to-get-Excel-virus/2100-1023_3-218868.html`

Davis, J. *Hackers take down the most wired country in Europe.* Online, August 2007. Accessed 2012/12/24.
URL `http://www.wired.com/politics/security/magazine/15-09/ff_estonia?currentPage=all`

Day, D., Flores, D., and Lallie, H. *Condor: A hybrid ids to offer improved intrusion detection.* In *11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 931–936. IEEE, 2012.

Debar, H., Dacier, M., and Wespi, A. *A revised taxonomy for intrusion-detection systems. Annals of Telecommunications*, 55(7):361–378, 2000.

Dede, D. *Apache.org defaced – security archive case study.* Online, March 2010. Accessed 2012/08/20.
URL `http://blog.sucuri.net/2010/03/apache-org-defaced-security-archive-case-study.html`

Deng, S. *Empirical model of www document arrivals at access link.* In *International Conference on Communications, Conference Record, Converging Technologies for Tomorrow's Applications.*, pages 1797–1802. IEEE, 1996.

Deraison, R. *The nessus attack scripting language reference guide.* Technical report, Tenable Network Security, Inc, January 2005. Accessed 2012/12/24.
URL `http://www.dn-systems.org/boss/doc/nasl_guide-20050103.pdf`

Dickerson, J. E. and Dickerson, J. A. *Fuzzy network profiling for intrusion detection.* In *19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS)*, pages 301–306. IEEE, 2000.

Dietrich, C. J. and Rossow, C. *Empirical research of ip blacklists.* In *ISSE 2008 Securing Electronic Business Processes*, pages 163–171. Springer, 2009.

Dolak, J. C. *The code red worm.* Technical report, SANS Institute InfoSec Reading Room, 2001. Accessed 2012/12/24.
URL `http://www.sans.org/reading_room/whitepapers/malicious/code-red-worm_85`

Drummond, D. *A new approach to China.* Online, January 2010. Accessed 2012/09/21.
URL `http://googleblog.blogspot.com/2010/01/new-approach-to-china.html`

Dübendorfer, T. and Plattner, B. *Host behaviour based early detection of worm outbreaks in internet backbones.* In *Proceedings of 14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WET ICE)*, pages 166–171. IEEE, 2005.

Durumeric, Z., Wustrow, E., and Halderman, J. A. *Zmap: Fast internet-wide scanning and its security applications.* In *22nd USENIX Security Symposium.* 2013.

Ebel, H., Mielsch, L., and Bornholdt, S. *Scale-free topology of e-mail networks. Physical review. E, Statistical, nonlinear, and soft matter physics*, 66(3):35103, 2002.

Elio, R., Hoover, J., Nikolaidis, I., Salavatipour, M., Stewart, L., and Wong, K. *About computing science research methodology.* Technical report, University of Alberta, 2005.
URL `http://webdocs.cs.ualberta.ca/~c603/readings/research-methods.pdf`

Enikeev, R. *South Africa's internet, mapped.* Online, July 2013. Accessed 2013/08/03.
URL `http://mybroadband.co.za/news/internet/83189-south-africas-internet-mapped.html`

Ezzeldin, H. *Nmap detection and countermeasures.* Online, March 2008. Accesed 2012/09/05.
URL `http://haymanezzeldin.blogspot.com/`

F-Secure. *Virus:boot/brain*. Online, May 2012. Accessed 2012/12/31.
URL `http://www.f-secure.com/v-descs/brain.shtml`

Falliere, N., Murchu, L., and Chien, E. *W32. stuxnet dossier*. Technical report, Symantec Corp, 2011. Accessed 2012/12/24.
URL `http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf`

Farivar, C. *Snowden says his "sole intention" was to prompt national security debate*. Online, September 2013. Accessed 2013/11/18.
URL `http://arstechnica.com/tech-policy/2013/09/snowden-says-his-sole-intention-was-to-prompt-national-security-debate/`

FBI. *Economic espionage*. Online, 1996. Accessed 2014/04/01.
URL `http://www.fbi.gov/about-us/investigate/counterintelligence/economic-espionage`

Felten, E. W., Balfanz, D., Dean, D., and Wallach, D. S. *Web spoofing: An internet con game. Software World*, 28(2):6–8, 1997.

Feng, W. *The case for tcp/ip puzzles. ACM SIGCOMM Computer Communication Review*, 33(4):322–327, 2003.

Fenz, S. and Neubauer, T. *How to determine threat probabilities using ontologies and bayesian networks*. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, page 69. ACM, 2009.

Fenz, S., Tjoa, A., and Hudec, M. *Ontology-based generation of bayesian networks*. In *International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'09)*, pages 712–717. IEEE, 2009.

Fidler, D. P. *Was stuxnet an act of war? Decoding a cyberattack. IEEE Security & Privacy*, 9(4):56–59, 2011.

Finkle, J. *Insight: Did conficker help sabotage Iran program?* Online, December 2011. Accessed 2012/09/19.
URL `http://www.reuters.com/article/2011/12/02/us-cybersecurity-iran-idUSTRE7B10AP2011120`

Finn, P. *Cyber assaults on estonia typify a new battle tactic.* Online, May 2007. Accessed 2012/12/24.
URL `http://msl1.mit.edu/furdlog/docs/washpost/2007-05-19_washpost_estonia_cyberattacked.pdf`

FIRST-Forum. *A complete guide to the common vulnerability scoring system version 2.0.* Technical report, FIRST-Forum of Incident Response and Security Teams, 2007. Accessed 2013/06/17.
URL `http://www.first.org/cvss/cvss-guide.pdf`

Fitzgibbon, N. and Wood, M. *Conficker. c: A technical analysis.* Technical report, SophosLabs, Sophon Inc, 2009. Accessed 2012/12/24.
URL `http://www.sophos.com/sophos/docs/eng/marketing_material/conficker-analysis.pdf`

Frye, L., Cheng, L., and Heflin, J. *An ontology-based system to identify complex network attacks.* In *International Conference on Communications (ICC)*, pages 6683–6688. IEEE, 2012.

Fu, X., Lu, X., Peltsverger, B., Chen, S., Qian, K., and Tao, L. *A static analysis framework for detecting sql injection vulnerabilities.* In *31st Annual International Conference on Computer Software and Applications (COMPSAC)*, volume 1, pages 87–96. IEEE, 2007.

Fyodor, R. *Remote os detection via tcp/ip stack fingerprinting.* Online, October 1998. Accessed 2012/11/19.
URL `http://nmap.org/nmap-fingerprinting-article.txt`

Gadge, J. and Patil, A. A. *Port scan detection.* In *16th International Conference on Networks (ICON)*, pages 1–6. IEEE, 2008.

Gandhi, M. and Srivatsa, S. *Detecting and preventing attacks using network intrusion detection systems. International Journal of Computer Science and Security*, 2(1):49–58, 2008.

Gandhi, R., Sharma, A., Mahoney, W., Sousan, W., Zhu, Q., and Laplante, P. *Dimensions of cyber-attacks: Cultural, social, economic, and political. Technology and Society Magazine, IEEE*, 30(1):28–38, 2011.

Garber, L. *Melissa virus creates a new type of threat. IEEE Computer*, 32(6):16–19, 1999.

Garcia-Teodoro, P., Diaz-Verdejo, J., Macia-Fernandez, G., and Vazquez, E. *Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security*, 28(1-2):18–28, 2009.

Geer, D. E. *A witness testimony in the hearing, Wednesday 25 April 07, entitled addressing the nation's cybersecurity challenges: Reducing vulnerabilities requires strategic investment and immediate action.* Technical report, Subcommittee on Emerging Threats, Cybersecurity, and Science and Technology, 2007. Accessed 2012/12/31.
URL `http://chsdemocrats.house.gov/SiteDocuments/20070425145243-10189.pdf`

Gellers, M., Brant, D., and B., G. *Building a secure workforce, guard against insider threat.* Technical Report 100108, Deloitte Federal Consulting Services, 2008. Accessed 2013/06/20.
URL `http://www.deloitte.com/assets/Dcom-UnitedStates/Local\%20Assets/Documents/us_ps_insiderthreat_100108.pdf`

Genosko, G. *The case of 'mafiaboy' and the rhetorical limits of hacktivism.* Online, 2008. Accessed 2012/12/31.
URL `http://nine.fibreculturejournal.org/fcj-057/`

Germain, J. M. *Mydoom: A wrap-up on the world's most vicious worm.* Online, March 2004. Accessed 2012/11/07.
URL `http://www.technewsworld.com/story/33068.html`

Geuss, M. *Snowden may have persuaded 20 to 25 nsa colleagues to give up their passwords.* Online, November 2013. Accessed 2013/11/17.
URL `http://arstechnica.com/tech-policy/2013/11/snowden-may-have-persuaded-20-to-25-nsa-colleagues-to-give-up-their-passwords/`

Glass, R. L. *A structure-based critique of contemporary computing research. Journal of Systems and Software*, 28(1):3–7, 1995.

Glenn Greenwald, E. M. and Poitras, L. *Edward snowden: the whistleblower behind the nsa surveillance revelations.* Online, June 2013. Accessed 2013/10/15.
URL `http://www.theguardian.com/world/2013/jun/09/edward-snowden-nsa-whistleblower-surveillance`

Gliddon, J. *Businesses take 7 months to detect intruders.* Online, October 2012. Accessed 2012/12/31.

URL `http://www.itnews.com.au/News/319549,businesses-take-seven-months-to-detect-intruders.aspx`

Goertzel, K. M. *Software survivability: Where safety and security converge. CrossTalk*, 34(6):15–19, 2009.

Gong, H., Guo, J., Yu, Z., Zhang, Y., and Xue, Z. *Research on the building and reasoning of travel ontology.* In *International Symposium on Intelligent Information Technology Application Workshops (IITAW'08)*, pages 94–97. IEEE, 2008.

Gong, L. *Optimal authentification protocols resistant to password guessing attacks.* In *Proceedings of Computer Security Foundations Workshop*, pages 24–29. IEEE, 1995.

Goodchild, J. *Social engineering: The basics.* Online, January 2010. Accessed 2012/08/28.
URL `http://www.csoonline.com/article/514063/social-engineering-the-basics`

Goodin, D. *User data stolen in sony PlayStation hack attack.* Online, April 2011. Accessed 2012/12/31.
URL `http://www.theregister.co.uk/2011/04/26/sony_playstation_network_security_breach/`

Gostev, A. *Mobile malware evolution: An overview, part 1.* Online, September 2006. Accessed 2012/12/31.
URL `http://www.securelist.com/en/analysis?pubid=200119916`

Gostev, A. *The flame: Questions and answers.* Online, May 2012. Accessed on 2012/09/20.
URL `https://www.securelist.com/en/blog/208193522/The_Flame_Questions_and_Answers`

Govindavajhala, S. and Appel, A. W. *Windows access control demystified.* Technical report, Princeton University, 2006.

Grabosky, P. *The global dimension of cybercrime. Global Crime*, 6(1):146–157, 2004.

Grant, T., Burke, I., and van Heerden, R. P. *Comparing models of offensive cyber operations.* In *Proceedings of the 7th International Conference on Information-Warfare & Security (ICIW 2012)*, pages 108–121. ACI, 2012.

Grant, T. and Kooter, B. *Comparing ooda and other models as operational view architecture.* In *Proceedings of the 10th International Command & Control Research & Technology Symposium (ICCRTS'05)*, 196. US DoD, 2005.

Grant, T., Venter, H., and Eloff, J. *Simulating adversarial interactions between intruders and system administrators using ooda-rr.* In *Proceedings of the 2007 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, pages 46–55. ACM, 2007.

Greenberg, A. *Reformed hacker looks back.* Online, August 2008. Accessed 2014/04/01.
URL `http://www.forbes.com/2008/08/21/mitnick-hackers-security-tech-security-cx\ag\0821mitnick.html`

Greenwald, G. *Nsa collecting phone records of millions of verizon customers daily.* Online, June 2013. Accessed 2013/10/16.
URL `http://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order`

Groebel, J., Metze-Mangold, V., van der Peet, J., and Ward, D. *Twilight Zones in Cyberspace: Crimes, Risk, Surveillance and User-Driven Dynamics.* Stabsabteilung der Friedrich-Ebert-Stiftung, 2001, 46 pages.

Gross, M. J. *Exclusive: Operation shady rat-unprecedented cyber-espionage campaign and intellectual-property bonanza.* Onine, August 2011. Accessed 2012/09/21.
URL `http://www.vanityfair.com/culture/features/2011/09/operation-shady-rat-201109`

Gruber, T. R. *A translation approach to portable ontology specifications. Knowledge acquisition*, 5(2):199–220, 1993.

Grüninger, M. and Fox, M. *Methodology for the design and evaluation of ontologies.* In *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI)*, volume 95. 1995.

Gula, R. *Correlating ids alerts with vulnerability information.* Technical Report Revision 4, Tenable Network Security, May 2011.
URL `http://www.tenable.com/sites/drupal.dmz.tenablesecurity.com/files/uploads/documents/whitepapers/va-ids_0.pdf`

Haddox, A. *First Excel macro virus discovered, detected and repaired.* Online, Sept 1996. Accessed 2012/12/31.
URL `http://www.symantec.com/avcenter/reference/newsletter/sarcanuv1i1.html`

Halderman, J. A. and Felten, E. W. *Lessons from the sony cd drm episode.* In *Proceedings of the 15th USENIX Security Symposium*. 2006.

Hall, K. *The 7 worst cyberattacks in history (that we know about).* Online, September 2010. Accessed 2012/12/02.
URL `http://dvice.com/archives/2010/09/7-of-the-most-d.php`

Halliday, J. *Epsilon email hack: millions of customers' details stolen.* Online, April 2011. Accessed 2012/09/20.
URL `http://www.guardian.co.uk/technology/2011/apr/04/epsilon-email-hack`

Hancock, B. *Recent history of known network breaches. Network Security*, 1995(11):6 – 9, 1995. ISSN 1353-4858. doi: 10.1016/1353-4858(96)89764-9.

Hanford, S. *Chronology of a ddos: Spamhaus.* Online, March 2013. Accessed 2013/05/19.
URL `http://blogs.cisco.com/security/chronology-of-a-ddos-spamhaus/`

Hansman, S. and Hunt, R. *A Taxonomy of Network and Computer Attack Methodologies.* Master's thesis, Department of Computer Science and Software Engineering University of Canterbury, 2003.
URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.77.1606`

Hariri, S., Qu, G., Dharmagadda, T., Ramkishore, M., and Raghavendra, C. S. *Impact analysis of faults and attacks in large-scale networks. IEEE Security & Privacy*, 1(5):49–54, 2003.

Harley, D. and Lee, A. *A pretty kettle of phish.* Technical report, ESET ThreatCenter, June 2007. Accessed 2012/12/31.
URL `http://www.nod32.com.sh/download/whitepapers/Phishing\%28June2007\%29Online.pdf`

Harrop, W. and Armitage, G. *Greynets: a definition and evaluation of sparsely populated darknets.* In *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, pages 171–172. ACM, 2005.

Harter, D. E., Kemerer, C. F., and Slaughter, S. A. *Does software process improvement reduce the severity of defects? a longitudinal field study. IEEE Transactions on Software Engineering*, 38(4):810–827, 2012.

Hassan, H., Garcia, J., and Brun, O. *Generic modeling of multimedia traffic sources.* In *IEEE International Workshop on Heterogeneous Wireless Networks: Resource Management and QoS (HWN)*. IEEE, 2005.

Heater, B. *Malware: A brief timeline.* Online, March 2011. Accessed 2012/12/02.
URL `http://www.pcmag.com/slideshow/story/261678/malware-a-brief-timeline`

Heberlein, L. T., Dias, G. V., Levitt, K. N., Mukherjee, B., Wood, J., and Wolber, D. *A network security monitor.* In *Proceedings of Computer Society Symposium on Research in Security and Privacy*, pages 296–304. IEEE, 1990.

Hernández-Campos, F., Karaliopoulos, M., Papadopouli, M., and Shen, H. *Spatio-temporal modeling of traffic workload in a campus wlan.* In *Proceedings of the 2nd annual international workshop on Wireless Internet*, page 1. ACM, 2006.

Hesseldahl, A. and Kharif, O. *Cyber crime and information warfare: A 30-year history.* Online, October 2010. Accesed on 2012/09/05.
URL `http://images.businessweek.com/ss/10/10/1014_cyber_attacks/1.htm`

Higgins, K. J. *'Aurora' attacks still under way, investigators closing in on malware creators.* Online, February 2010. Accessed on 2012/08/14.
URL `http://www.darkreading.com/security/news/222700786`

Hlava, M. *What is a taxonomy?* Online, March 2012. Accessed 2013/01/01.
URL `http://www.kmworld.com/Articles/Editorial/What-Is-../What-is-a-Taxonomy-81159.aspx`

Hoepman, J.-H. and Jacobs, B. *Increased security through open source. Communications of the ACM*, 50(1):79–83, 2007.

Horib, K., Yamamoto, S., and Sekiya, Y. *Udon: User defined and organized network.* In *Internet Conference*. Toyama, Japan, 2012.

Houle, K. and Weaver, G. *Trends in denial of service attack technology.* Technical report, CERT Coordination Center, October 2001. Accessed 2013/01/01.
URL `www.cert.org/archive/pdf/DoS_trends.pdf`

Hunter, S. O. and Irwin, B. *Tartarus: A honeypot based malware tracking and mitigation framework.* In *Information Security South Africa*. 2011.

Idika, N. and Mathur, A. *A survey of malware detection techniques.* Technical report, Purdue University, 2007. Accessed 2013/01/01.
URL `http://www.serc.net/system/files/SERC-TR-286.pdf`

Irwin, B. *A framework for the application of network telescope sensors in a global IP network.* Ph.D. thesis, Rhodes University, 2011.

Jagatic, T. N., Johnson, N. A., Jakobsson, M., and Menczer, F. *Social phishing. Communications of the ACM*, 50(10):94–100, 2007.

Jim, T., Swamy, N., and Hicks, M. *Defeating script injection attacks with browser-enforced embedded policies.* In *Proceedings of the 16th international conference on World Wide Web*, pages 601–610. ACM, 2007.

Jones, S. L. *The passion of Bradley Manning: The story behind the wikileaks whistle-blower. Journal of Military Ethics*, 12(2):195–196, 2013.

Joyal, P. *Industrial espionage today and information wars of tomorrow.* In *19th National Information Systems Security Conference*, pages 139–151. 1996.

Julian. *10 most costly cyber attacks in history.* Online, August 2011. Accessed 2012/10/30.
URL `http://www.businesspundit.com/10-most-costly-cyber-attacks-in-history/`

Jung, J., Paxson, V., Berger, A. W., and Balakrishnan, H. *Fast portscan detection using sequential hypothesis testing.* In *Proceedings of Symposium on Security and Privacy*, pages 211–225. IEEE, 2004.

Kabay, M. *Industrial espionage.* In *Network World Fusion Security Newsletter*. School of Graduate Studies / Norwich University, 2008.

Kabay, M. E. *Crime, Use of Computers in.* Academic Press (Amsterdam), 2003, 15 pages.

Kachirski, O. and Guha, R. *Effective intrusion detection using multiple sensors in wireless ad hoc networks.* In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pages 8–16. IEEE, 2003.

Kaplan, D. *Anonymous spokesman on Sony hack: "it wasn't us".* Online, May 2011. Accessed 2012/09/12.
URL `http://www.scmagazine.com/anonymous-spokesman-on-sony-hack-it-wasnt-us/article/202134/`

Karasaridis, A., Rexroad, B., and Hoeflin, D. *Wide-scale botnet detection and characterization.* In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. Cambridge, MA, 2007.

Karig, D. and Lee, R. *Remote denial of service attacks and countermeasures.* Technical Report CE-L2001-002, Princeton University Department of Electrical Engineering, October 2001. Accessed 2013/01/01.
URL `https://www.princeton.edu/~rblee/ELE572Papers/karig01DoS.pdf`

Karlson, A. K., Meyers, B. R., Jacobs, A., Johns, P., and Kane, S. K. *Working overtime: Patterns of smartphone and pc usage in the day of an information worker.* In *Pervasive Computing*, pages 398–405. Springer, 2009.

Katz-Bassett, E., John, J. P., Krishnamurthy, A., Wetherall, D., Anderson, T., and Chawathe, Y. *Towards ip geolocation using delay and topology measurements.* In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, pages 71–84. ACM, 2006.

Kayacık, H. G., Zincir-Heywood, A. N., and Heywood, M. I. *Can a good offense be a good defense? Vulnerability testing of anomaly detectors through an artificial arms race. Applied Soft Computing*, 11(7):4366–4383, 2011.

Kent, K. and Souppaya, M. *Guide to computer security log management.* Technical Report SP800-92, National Institute of Standards and Technology, 2006.
URL `http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf`

Kerber, R. and Bartz, D. *Analysis: Epsilon hacking shows new "spear-phishing" risks.* Online, April 2011. Accessed 2012/09/20.
URL `http://www.reuters.com/article/2011/04/04/us-hackers-epsilon-idUSTRE7336DZ20110404`

Kibret, W. E. *Analyzing Network Security from a Defense in Depth Perspective.* Master's thesis, Department of Informatics University of Oslo, 2011.

Kim, G. H. and Spafford, E. H. *The design and implementation of tripwire: A file system integrity checker.* In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 18–29. ACM, 1994.

Knapp, E. *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems.* Syngress, 2011.

Kneale, B., De Horta, A. Y., and Box, I. *Velnet: virtual environment for learning networking.* In *Proceedings of the Sixth Australasian Conference on Computing Education*, pages 161–168. Australian Computer Society, Inc., 2004.

Kolšek, M. *Session fixation vulnerability in web-based applications.* Technical Report 1, Acros Security, December 2002.
URL `http://www.acrossecurity.com/papers/session_fixation.pdf`

Kornexl, S., Paxson, V., Dreger, H., Feldmann, A., and Sommer, R. *Building a time machine for efficient recording and retrieval of high-volume network traffic.* In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 23–23. USENIX Association, 2005.

Krebs, B. *Study of sony anti-piracy software triggers uproar.* Online, November 2005. Accessed 2014/01/01.
URL `http://msl1.mit.edu/furdlog/docs/washpost/2005-11-03_washpost_sony_drm.pdf`

Krebs, B. *The scrap value of a hacked pc.* Online, May 2009. Accessed 2012/11/07.
URL `http://voices.washingtonpost.com/securityfix/2009/05/the_scrap_value_of_a_hacked_pc.html`

Krebs, B. *The scrap value of a hacked pc, revisited.* Online, October 2012. Accessed 2012/11/07.
URL `http://krebsonsecurity.com/2012/10/the-scrap-value-of-a-hacked-pc-revisited/`

Kritzinger, E. and von Solms, S. H. *Cyber security for home users: A new way of protection through awareness enforcement. Computers & Security*, 29(8):840–847, 2010.

Kshetri, N. *Pattern of global cyber war and crime: A conceptual framework. Journal of International Management*, 11(4):541–562, 2005.

Kurtz, G., McClure, S., and Scambray, J. *Hacking Exposed: Network Security Secrets and Solutions.* Computing Mcgraw-Hill, 1999.

Kuwatly, I., Sraj, M., Al Masri, Z., and Artail, H. *A dynamic honeypot design for intrusion detection.* In *International Conference on Pervasive Services (ICPS)*, pages 95–104. IEEE, 2004.

Lancor, L. and Workman, R. *Using google hacking to enhance defense strategies. ACM SIGCSE Bulletin*, 39(1):491–495, 2007.

Lapsley, P. *Phreaking out ma bell. IEEE Spectrum*, 50(2):30–35, 2013.

Lau, F., Rubin, S. H., Smith, M. H., and Trajkovic, L. *Distributed denial of service attacks.* In *International Conference on Systems, Man, and Cybernetics*, pages 2275–2280. IEEE, 2000.

Lee, C., Roedel, C., and Silenok, E. *Detection and characterization of port scan attacks.* Technical report, Univeristy of California, Department of Computer Science and Engineering, 2003. Accessed 2013/01/01.
URL `http://www.cs.ucsd.edu/users/clbailey/PortScans.pdf`

Lee, S. and Shields, C. *Tracing the source of network attack: A technical, legal and societal problem.* In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, pages 239–246. IEEE, 2001.

Leeson, P. and Coyne, C. *The economics of computer hacking. The Journal of Law, Economics and Policy*, 1(1.78):511, 2005.

Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., Postel, J., Roberts, L. G., and Wolff, S. *A brief history of the internet. ACM SIGCOMM Computer Communication Review*, 39(5):22–31, 2009.

Levy, S. *Hackers: Heroes of the computer revolution*, volume 4. Doubleday, 1984.

Lewis, J. A. *Assessing the risks of cyber terrorism, cyber war and other cyber threats.* Technical report, Center for Strategic and International Studies, 2002. Accessed 2013/01/08.
URL `http://csis.org/files/media/csis/pubs/021101_risks_of_cyberterror.pdf`

Lewis, L. *A facelift for defacement: Graffiti as new media.* In *Society for Information Technology & Teacher Education International Conference*, pages 2687–2691. 2007.

Leyden, J. *PC virus celebrates 20th birthday.* Online, January 2006. Accessed 2013/01/03.
URL `http://www.theregister.co.uk/2006/01/19/pc_virus_at_20/`

Leyden, J. *Biggest ddos attack in history hammers spamhaus.* Online, March 2013. Accessed 2013/05/18.
URL `http://www.theregister.co.uk/2013/03/27/spamhaus_ddos_megaflood/`

Li, W. and Tian, S. *An ontology-based intrusion alerts correlation system. Expert Systems with Applications*, 37(10):7138–7146, 2010.

Liddinton-Cox, A. *Top 10 hacks of all time.* Online, May 2012. Accesed 2012/12/02. URL `http://www.businessspectator.com.au/bs.nsf/Article/hacks-IT-internet-security-email-Google-Sony-pd20110505-GJVTZ`

Liljenstam, M., Liu, J., Nicol, D. M., Yuan, Y., Yan, G., and Grier, C. *Rinse: The real-time immersive network simulation environment for network security exercises (extended version). Simulation*, 82(1):43–59, 2006.

Lindqvist, U. and Jonsson, E. *How to systematically classify computer security intrusions.* In *Proceedings of Symposium on Security and Privacy*, pages 154–163. IEEE, 1997.

Long, J. *Google hacking for penetration testers.* Syngress Publishing, 2007.

Lunt, T. F. *A survey of intrusion detection techniques. Computers & Security*, 12(4):405–418, 1993.

Lunt, T. F. and Jagannathan, R. *A prototype real-time intrusion-detection expert system.* In *Proceedings of Symposium on Security and Privacy*, pages 59–66. IEEE, 1988.

Madigan, D. *Statistics and the war on spam. Statistics: A Guide to the Unknown*, pages 135–147, 2005.

Magklaras, G. and Furnell, S. *Insider threat prediction tool: Evaluating the probability of IT misuse. Computers & Security*, 21(1):62–73, 2001.

Malviya, N., Mishra, N., and Sahu, S. *Developing university ontology using protégé owl tool: Process and reasoning. International Journal of Scientific & Engineering Research*, 2(9):1, 2011.

Mandujano, S. *An ontology-supported outbound intrusion detection system.* In *Proceedings of the 10th Conference on Artificial Intelligence and Applications.* Taiwanese Association for Artificial Intelligence, 2005.

Manmadhan, S. and Manesh, T. *A method of detecting sql injection attack to secure web applications. International Journal of Distributed and Parallel Systems*, 3:1–8, 2012.

Mansfield-Devine, S. *Hacking the hackers. Computer Fraud & Security*, 2009(6):10–13, 2009.

Marca, D. and McGowan, C. *SADT: structured analysis and design technique.* McGraw-Hill, Inc., 1987.

Marcus, D. *A good decade for cybercrime.* Technical report, McAfee, January 2011. Accessed 2013/01/01.
URL `http://www.mcafee.com/us/resources/reports/rp-good-decade-for-cybercrime.pdf`

Markoff, J. *Attack of the zombie computers is growing threat. New York Times*, 7:1–4, 2007.

Markoff, J. and Barboza, D. *2 China schools said to be tied to online attacks.* Online, Febraury 2010. Accessed 2012/09/21.
URL `http://www.nytimes.com/2010/02/19/technology/19china.html?_r=0`

Markoff, J. and Perlroth, N. *Firm is accused of sending spam, and fight jams internet.* Online, March 2013. Accessed 2013/05/16.
URL `http://www.nytimes.com/2013/03/27/technology/internet/online-dispute-becomes-internet-snarling-attack.html?`

Massey, D. *Phone phreaking – the telecommunications underground.* Online, 2003. Accessed 2013/10/17.
URL `http://www.telephonetribute.com/phonephreaking.html`

Maybury, M., Chase, P., Cheikes, B., Brackney, D., Matzner, S., Hetherington, T., Wood, B., Sibley, C., Marin, J., and Longstaff, T. *Analysis and detection of malicious insiders.* Technical report, DTIC document, 2005. Accessed 2013/01/01.
URL `http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA456356`

McAfee. *Protecting your critical assets lessons learned from "operation aurora".* Technical report, McAfee Labs and McAfee Foundstone Professional Services, 2010. Accessed 2013/01/01.
URL `http://www.mcafee.com/us/resources/white-papers/wp-protecting-critical-assets.pdf`

McDowell, M. *Understanding denial-of-service attacks.* Online, November 2009. Accessed 2012/09/22.
URL `http://www.us-cert.gov/cas/tips/ST04-015.html`

McNamara, P. *Melissa virus turning 10 ... (age of the stripper unknown).* Online, March 2009. Accessed 2013/01/01.
URL `http://www.networkworld.com/community/node/40059`

Mercuri, R. T. and Neumann, P. G. *Security by obscurity. Communications of the ACM*, 46(11):160, 2003.

Merritt, D. *Spear Phishing Attack Detection.* Master's thesis, Air Force Institute of Technology, March 2011. Accessed 2013/01/01.
URL `http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA540272`

Meyers, C., Powers, S., and Faissol, D. *Taxonomies of cyber adversaries and attacks: a survey of incidents and approaches.* Technical Report LLNL-TR-4190, Lawrence Livermore National Laboratory, April 2009. Accessed 2013/10/01.
URL `https://e-reports-ext.llnl.gov/pdf/379498.pdf`

Mezirch, B. *The Accidental Billionaires: The Founding of Facebook: A Tale of Sex, Money, Genius and Betrayal.* Random House, 2009.

Miranda, M. *The 12 costliest computer viruses ever.* Online, August 2010. Accesed 2012/12/02.
URL `http://blog.insure.com/2010/08/03/the-12-costliest-computer-viruses-ever/`

Mirkovic, J. and Reiher, P. *A taxonomy of ddos attack and ddos defense mechanisms. ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

Mitnick, K. and Simon, W. *The Art of Intrusion: The real stories behind the exploits of hackers, intruders and deceivers.* Wiley, 2005.

Mitnick, K. D., Simon, W. L., and Wozniak, S. *The art of deception.* Wiley Indianapolis, 2002.

Mookhey, K. and Burghate, N. *Detection of sql injection and cross-site scripting attacks.* Technical Report INFOCUS 1768, Symantec, 2004. Accessed 2013/01/08.
URL `http://www.symantec.com/connect/articles/detection-sql-injection-and-cross-site-scripting-attacks`

Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., and Weaver, N. *Inside the slammer worm. IEEE Security & Privacy*, 1(4):33–39, 2003.

Moore, D. and Shannon, C. *Sco offline from denial-of-service attack.* Online, December 2003. Accessed 2012/09/02.
URL `http://www.caida.org/research/security/sco-dos/`

Moore, D., Shannon, C., Brown, D., Voelker, G., and Savage, S. *Inferring internet denial-of-service activity. ACM Transactions on Computer Systems (TOCS)*, 24(2):115–139, 2006.

Moore, D., Shannon, C., Voelker, G. M., and Savage, S. *Network telescopes: Technical report.* Technical report, Department of Computer Science and Engineering, University of California, San Diego, 2004.
URL `http://ants.iis.sinica.edu.tw/3bkmj9ltewxtsrrvnoknfdxrm3zfwrr/17/tr-2004-04.pdf`

Moshchuk, A., Bragin, T., Gribble, S. D., and Levy, H. M. *A crawler-based study of spyware on the web.* In *Proceedings of the 2006 Network and Distributed System Security Symposium*, pages 17–33. 2006.

Mudge, R. *Live-fire security testing with armitage and metasploit.* Online, July 2011. Accessed 2013/01/01.
URL `http://www.linuxjournal.com/article/10973`

Mukherjee, B., Heberlein, L., and Levitt, K. *Network intrusion detection. IEEE Network*, 8(3):26–41, 1994.

Mulligan, D. K. and Perzanowski, A. K. *The magnificence of the disaster: Reconstructing the Sony BMG rootkit incident. Berkeley Technology Law Journal*, 22(3):1157–1232, 2007.

Murry, P. *Network security, the bad, the good, and the quiz.* Technical report, Ohio State University, May 2004. Accessed 2013/01/01.
URL `https://kb.osu.edu/dspace/bitstream/handle/1811/5934/securityhandoutmurray.pdf`

Myers, J., Grimaila, M., and Mills, R. *Towards insider threat detection using web server logs.* In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, pages 54–58. ACM, 2009.

Myler, C. and Wapping, L. *News of the world.* Online, 2011. Accessed 2013/01/01.
URL `http://medbib.com/News_of_the_World`

Nachenberg, C. *Hacking*. Online, 2012. Accessed 2012/12/23.
URL `http://www.bookrags.com/research/hacking-csci-03/`

Namuduri, S. *Distributed Denial of Service Attacks (DDoS)-Consequences and Future*. Master's thesis, Linköping, November 2006.

Netcraft. *November 2012 web server survey*. Online, November 2012. Accessed 2013/01/08.
URL `http://news.netcraft.com/archives/category/web-server-survey/`

Neumann, J. and Burks, A. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.

Neumann, R. and Parker, C. *A summary of computer misuse techniques*. In *Processings of the 12th National Computer Security Conference*, pages 396–407. 1989.

Nilsson, R., Offutt, J., and Mellin, J. *Test case generation for mutation-based testing of timeliness. Electronic Notes in Theoretical Computer Science*, 164(4):97–114, 2006. ISSN 1571-0661.

Noy, N. and McGuinness, D. *Ontology development 101: A guide to creating your first ontology*. Technical Report KSL-01-05, SMI-2001-0880, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics technical report, March 2001. Accessed 2013/01/01.
URL `http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf`

Olivarez-Giles, N. *Epsilon hacking exposes customers of best buy, capital one, citi, jpmorgan chase and others*. Online, April 2011. Accessed 2012/09/20.
URL `http://latimesblogs.latimes.com/technology/2011/04/epsilon-cutsomer-files-email-addresses-breached-including-best-buy-jpmorgan-chase-us-bank-capital-on.html`

Ophardt, J. A. *Cyber warfare and the crime of aggression: The need for individual accountability on tomorrow's battlefield. Duke Law & Technology Review*, 3:1–27, 2010.

Oppliger, R. *Security at the internet layer. Computer*, 31(9):43–47, 1998.

Orman, H. *The morris worm: a fifteen-year perspective. IEEE Security & Privacy*, 1(5):35–43, 2003.

Owen, D. *What is a false positive and why are false positives a problem?* Online, May 2010. Accessed 2012/11/21.
URL `http://www.sans.org/security-resources/idfaq/false_positive.php`

Peddabachigari, S., Abraham, A., Grosan, C., and Thomas, J. *Modeling intrusion detection system using hybrid intelligent systems. Journal of Network and Computer Applications*, 30(1):114–132, 2007.

Perrow, C. *Complexity, catastrophe, and modularity. Sociological Inquiry*, 78(2):162–173, 2008.

Poese, I., Uhlig, S., Kaafar, M. A., Donnet, B., and Gueye, B. *IP geolocation databases: unreliable? ACM SIGCOMM Computer Communication Review*, 41(2):53–56, 2011.

Porras, P., Saïdi, H., and Yegneswaran, V. *A foray into Conficker's logic and rendezvous points.* In *Proceedings of the 2nd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More*, pages 7–16. USENIX Association, 2009.

Porras, P., Saidi, H., and Yegneswaran, V. *An analysis of the ikee.b iphone botnet.* In *Security and Privacy in Mobile Information and Communication Systems*, pages 141–152. Springer, 2010.

Porras, P. and Valdes, A. *Live traffic analysis of TCP/IP gateways.* In *Networks and Distributed Systems Security Symposium*. 1998.

Poulsen, K. *The decade's 10 most dastardly cybercrimes.* Online, December 2009. Accesed 2012/12/02.
URL `http://www.wired.com/threatlevel/2009/12/ye_cybercrimes/`

Prince, M. *The ddos that knocked spamhaus offline (and how cloudflare mitigated it).* Online, March 2013. Accessed 2013/06/15.
URL `http://www.outlookseries.com/A0989/Security/3856_DDoS_Knocked_Spamhaus_Offline_CloudFlare_Mitigated.htm`

Provos, N. *A virtual honeypot framework.* In *Proceedings of the 13th USENIX Security Symposium*. 2004.

Rahmad, B., Supangkat, S., Sembiring, J., and Surendro, K. *Threat scenario dependency-based model of information security risk analysis. International Journal of Computer Science and Network Security*, 10(8):93, 2010.

Rajagopalan, S. *A study of security problems associated with the telephone network.* Technical report, Department of Electrical and Computer Engineering, 2000. Accessed 2013/01/01.
URL `http://www.artofhacking.com/Tucops/Phreak/GENERAL/R2.PDF`

Ramachandran, A., Dagon, D., and Feamster, N. *Can dns-based blacklists keep up with bots.* In *Conference on Email and Anti-spam*. 2006.

Ramsbrock, D., Berthier, R., and Cukier, M. *Profiling attacker behavior following ssh compromises.* In *37th Annual International Conference on Dependable Systems and Networks*, pages 119–124. IEEE/IFIP, 2007.

Rao, T. *Defending against web vulnerabilities and cross-site scripting. Journal of Global Research in Computer Science*, 3(5):61–64, 2012.

Razvan, R. *Over the sql injection hacking method.* In *Proceedings of the 3rd International Conference on Communications and Information Technology*, pages 116–118. World Scientific and Engineering Academy and Society (WSEAS), 2009.

Rescorla, E. *Security holes... who cares?* In *Proceedings of the 12th USENIX Security Symposium*, pages 75–90. 2003.

Rescorla, E. *Is finding security holes a good idea? IEEE Security & Privacy*, 3(1):14–19, 2005.

Richelson, J. T. *The snowden affair.* Technical report, National Security Archive, 2013.
URL `http://www2.gwu.edu/~nsarchiv/NSAEBB/NSAEBB436/`

Rietta, F. S. *Application layer intrusion detection for sql injection.* In *Proceedings of the 44th Annual Southeast Regional Conference*, pages 531–536. ACM, 2006.

Rizzo, L. *Dummynet: a simple approach to the evaluation of network protocols. ACM SIGCOMM Computer Communication Review*, 27(1):31–41, 1997.

Robson, G. *The origins of phreaking.* Online, April 2004. Accessed 2013/01/01.
URL `http://www.robson.org/gary/a-blacklisted1.php`

Rochaeli, T. and Eckert, C. *Rbac policy engineering with patterns.* In *W9: The Semantic Web and Policy Workshop (SWPW)*, pages 148–153. 2005.

Rogers, M. K. *A two-dimensional circumplex approach to the development of a hacker taxonomy. Digital Investigation*, 3(2):97–102, 2006.

Rosenblum, M., Herrod, S. A., Witchel, E., and Gupta, A. *Complete computer system simulation: The simos approach. IEEE Parallel & Distributed Technology: Systems & Applications*, 3(4):34–43, 1995.

Rounds, M. and Pendgraft, N. *Diversity in network attacker motivation: A literature review.* In *2009 International Conference on Computational Science and Engineering*, pages 319–323. IEEE, 2009.

Rouse, M. *Definition: Taxonomy.* Online, 2005. Accessed 2013/01/01.
URL `http://searchcio-midmarket.techtarget.com/definition/taxonomy`

Rouse, M. *Defintion: Social engineering.* Online, October 2006. Accessed 2012/09/12.
URL `http://searchsecurity.techtarget.com/definition/social-engineering`

Rutherford, M. *Report: Russian mob aided cyberattacks on georgia.* Online, August 2009. Accessed 2012/09/05.
URL `http://news.cnet.com/8301-13639_3-10312708-42.html`

Safire, W. *The farewell dossier.* Online, February 2004. Accessed 2013/01/01.
URL `http://www.nytimes.com/2004/02/02/opinion/the-farewell-dossier.html`

Sanger, D. *Obama order sped up wave of cyberattacks against iran.* Online, June 2012. Accessed 2012/08/24.
URL `http://www.nytimes.com/2012/06/01/world/middleeast/obama-ordered-wave-of-cyberattacks-against-iran.html`

SAPA. *Man sentenced for post bank cyber heist.* Online, March 2012. Accessed 2013/01/01.
URL `http://www.techcentral.co.za/man-sentenced-for-post-bank-cyber-heist/30448/`

Sarwar, U., Ramadass, S., and Budiarto, R. *Dawn of the mobile malware: Reviewing mobile worms.* In *Sciences of Electronic, Technologies of Information and Telecommunications*. IEEE, March 2007.

Savona, E. and Mignone, M. *The fox and the hunters: How IC technologies change the crime race. European Journal on Criminal Policy and Research*, 10(1):3–26, 2004.

Scharffe, F. and de Bruijn, J. *A language to specify mappings between ontologies.* In *Proceedings of the Internet Based Systems (SITIS05)*. IEEE, 2005.

Schartz, N. and Dash, E. *Thieves found citigroup site an easy entry.* Online, June 2011. Accessed 2013/07/21.
URL `http://www.nytimes.com/2011/06/14/technology/14security.html`

Scholte, T., Robertson, W., Balzarotti, D., and Kirda, E. *An empirical analysis of input validation mechanisms in web applications and languages.* In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1419–1426. ACM, 2012.

Schultze, E. *Thinking like a hacker.* Technical report, Shavlik Technologies, March 2002. Accessed 2013/01/01.
URL `http://pdf.textfiles.com/security/thinkhacker.pdf`

Schwartz, M. J. *Who is anonymous: 10 key facts.* Online, July 2012. Accessed 2012/08/28.
URL `http://www.informationweek.com/security/attacks/who-is-anonymous-10-key-facts/232600322`

Shahzad, M., Shafiq, M. Z., and Liu, A. X. *A large scale exploratory analysis of software vulnerability life cycles.* In *34th International Conference on Software Engineering (ICSE)*, pages 771–781. IEEE, 2012.

Shankar, U., Talwar, K., Foster, J. S., and Wagner, D. *Detecting format string vulnerabilities with type qualifiers.* In *USENIX Security Symposium*, pages 201–220. 2001.

Shankland, S. *Net attack crushes sco web site.* Online, May 2003. Accessed 2012/09/02.
URL `http://news.cnet.com/Net-attack-crushes-SCO-Web-site/2100-1002_3-999584.html`

Sharan, R. *The five stages of ethical hacking.* Online, December 2010. Accessed 2013/01/01.
URL `http://hack-o-crack.blogspot.com/2010/12/five-stages-of-ethical-hacking.html`

Sharma, A. *The Flame – most powerful malware till date.* Online, May 2012. Accessed 2012/09.20.
URL `http://www.symantec.com/connect/blogs/flame-most-powerful-malware-till-date`

Sharma, V. *An analytical survey of recent worm attacks. International Journal of Computer Science and Network Security*, 11(11):99–103, November 2011.

Shavitt, Y. and Zilberman, N. *A geolocation databases study. IEEE Journal on Selected Areas in Communications*, 29(10):2044–2056, 2011.

Shearer, R., Motik, B., and Horrocks, I. *Hermit: A highly-efficient owl reasoner.* In *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2008)*, pages 26–27. 2008.

Shearman, A. *Hack the planet.* Online, 1999. Accessed 2012/11/03.
URL `http://www.thehackademy.net/madchat/reseau/syn.smurf/dos.htm`

Shiffman, G. and Gupta, R. *Crowdsourcing cyber security: a property rights view of exclusion and theft on the information commons. International Journal of the Commons*, 7(1):93–112, February 2013.

Shimomura, T. and Markoff, J. *Takedown: The Pursuit and Capture of Kevin Mitnick, America's Most Wanted Computer Outlaws – by the Man Who Did It.* Hyperion Press, 1st edition, 1995. ISBN 0786862106.

Shin, Y. and Williams, L. *Is complexity really the enemy of software security?* In *Proceedings of the 4th ACM Workshop on Quality of Protection*, pages 47–50. ACM, 2008.

Sigholm, J. *Non-state actors in cyberspace operations. Journal of Military Studies*, 4(1):1–37, 2013.

Simmonds, A., Sandilands, P., and van Ekert, L. *An ontology for network security attacks. Applied Computing*, (2):317–323, 2004.

Smith, F. D., Campos, F. H., Jeffay, K., and Ott, D. *What tcp/ip protocol headers can tell us about the web. ACM SIGMETRICS Performance Evaluation Review*, 29(1):245–256, 2001.

Sorin, P. *Web server monitoring. Annals of University of Craiova-Economic Sciences Series*, 2(36):710–715, 2008.

Sorkin, A. *The social network.* Screenplay, 2010. Accessed 2013/07/27.
URL `http://flash.sonypictures.com/video/movies/thesocialnetwork/awards/thesocialnetwork_screenplay.pdf`

Spafford, E. *The internet worm program: an analysis. ACM SIGCOMM Computer Communication Review*, 19(1):17–57, 1989.

Spafford, E., Heaphy, K., and Ferbrache, D. *A computer virus primer.* Technical Report CSD-TR-935, Purdue University, November 1989.
URL `http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1794\&context=cstech`

Specht, S. and Lee, R. *Distributed denial of service: Taxonomies of attacks, tools, and countermeasures.* In *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems*, pages 543–550. 2004.

Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., and Stiller, B. *An overview of IP flow-based intrusion detection. IEEE Communications Surveys & Tutorials*, 12(3):343–356, 2010.

Spiegelman, I. *Cyber terrorists attack Russian news agency.* Online, August 2008. Accessed 2012/09/05.
URL `http://gawker.com/5035278/cyber-terrorists-attack-russian-news-agency`

Spitzner, L. *Know your enemy.* Online, July 2000. Accessed 2012/11/18.
URL `http://www.firstnetsecurity.com/library/misc/knowyourenemy1.pdf`

Spitzner, L. *Honeypots: Catching the insider threat.* In *Proceedings of the 19th Annual Computer Security Applications Conference*, pages 170–179. IEEE, 2003.

Staniford, S., Hoagland, J., and McAlerney, J. *Practical automated detection of stealthy portscans. Journal of Computer Security*, 10(1/2):105–136, 2002.

Steffan, J. and Schumacher, M. *Collaborative attack modeling.* In *Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 253–259. ACM, 2002.

Sterling, B. *The advanced persistent threat attack.* Online, January 2010. Accessed 2013/01/02.
URL `http://www.wired.com/beyond_the_beyond/2010/01/the-advanced-persistent-threat-attack/`

Sterne, D. F. *On the buzzwordsecurity policy.* In *Proceedings of Computer Society Symposium on Research in Security and Privacy*, pages 219–230. IEEE, 1991.

Stiawan, D., Idris, M., Ihsan, Z., Hussain, K., and Abdullah, A. *Heterogeneous parameters for accuracy threat. Journal of Theoretical and Applied Information Technology*, 33(2):142–154, 2011.

Stoll, C. *The cuckoo's egg. Tracking a spy through a maze of computer espionage*, volume 1. Doubleday, 1989.

Stone, H. L. *Political strategy for cyber security. Intersect: The Stanford Journal of Science, Technology and Society*, 5:1–15, 2012.

Stout, G. *Testing a website: Best practices.* Technical report, Reveregroup, 2001. Accessed 2013/01/02.
URL `http://home.comcast.net/~glennastout/test_meth/TestWebsite_Stout.pdf`

Strayer, W., Lapsely, D., Walsh, R., and Livadas, C. *Botnet detection based on network behavior. Botnet Detection*, 36:1–24, 2008.

Strickland, J. *10 worst computer viruses of all time.* Online, 2008. Accessed 2012/12/02.
URL `http://computer.howstuffworks.com/worst-computer-viruses.htm`

Subramanya, S. and Lakshminarasimhan, N. *Computer viruses. IEEE Potentials*, 20(4):16–19, 2001.

Swart, W. *'Inside man' who sank R30m heist.* Online, September 2012. Accessed 2012/12/04.
URL `http://www.timeslive.co.za/sundaytimes/2012/09/23/inside-man-who-sank-r30m-heist`

Swart, W. and Afrika, M. *It was a happy new year's day for gang who pulled off R42m Postbank heist.* Online, January 2012.
URL `http://www.timeslive.co.za/local/2012/01/15/it-was-a-happy-new-year-s-day-for-gang-who-pulled-off...r42m-postbank-heist`

Sweetman, D. *Bradley manning, collateral murder, truth, and power.* Technical report, The School for Conflict Analysis and Resolution, 2011.
URL `http://scar.gmu.edu/magazine-article/bradley-manning-collateral-murder-truth-and-power`

Taylor, P. A. *Editorial: Hacktivism. The Semiotic Review of Books*, 12(1):1–4, 2001.

Tech Analyser. *Best known cyber-attacks of all times.* Online, Nov 2011. Accessed 2012/12/02.
URL `http://tech-analyser.blogspot.com/2011/11/most-recent-cyber-attacks-were-laid.html`

Teumim, D. *Industrial Network Security.* International Society of Automation, 2010.

Thomas, K. *Playstation network hacked.* Online, April 2011. Accessed 2013/01/07.
URL `http://www.pcworld.com/article/226128/`

Thornburg, D. D. *The network is the computer: The changing direction of classroom computing.* Technical report, Thornburg Center for Space Exploration, 2009.
URL `http://www.tcse-k12.org/pages/network.pdf`

Thornburgh, N. *Inside the Chinese hack attack.* Online, August 2005. Accessed 2013/06/19.
URL `http://www.time.com/time/nation/article/0,8599,1098371,00.html`

Tjhai, G., Papadaki, M., Furnell, S., and Clarke, N. *Investigating the problem of ids false alarms: An experimental study using snort.* In *Proceedings of the IFIP TC 11 23rd International Information Security Conference*, pages 253–267. IFIP, 2008.

Touch, J., Lear, E., Mankin, A., Ono, K., Stiemerling, M., Eggert, L., Melnikov, A., and Eddy, W. *Service name and transport protocol port number registry.* Technical report, Internet Assigned Numbers Autority, August 2013.
URL `http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml`

Trammell, D. and Manning, T. *Simulating distributed denial of service with breakingpoint.* Technical report, BreakingPoint Labs, April 2009.
URL `http://druid.caughq.org/papers/Simulating-Distributed-Denial-of-Service-with-BreakingPoint.pdf`

TrendMicro. *The Sasser event: History and implications.* Technical report, Trendlabs Research, 2004. Accessed 2013/01/07.
URL `http://www.antivirus2u.com/trend/WP02SasserEvent040812US.pdf`

Tutănescu, I. and Sofron, E. *Anatomy and types of attacks against computer networks.* In *2nd RoEduNet International Conference*, pages 265–270. 2003.

Undercoffer, J., Pinkston, J., Joshi, A., and Finin, T. *A target-centric ontology for intrusion detection.* In *18th International Joint Conference on Artificial Intelligence*, pages 9–15. 2004.

US-CERT. *Smurf ip denial-of-service attacks.* Technical Report CA-1998-01, Software Engineering Institute Carnegie Mellon, 1998.
URL `http://www.cert.org/advisories/CA-1998-01.html`

van Dijk, P. *How we defaced www.apache.org.* Online, May 2000. Accessed 2012/08/20. URL `http://www.dataloss.net/papers/how.defaced.apache.org.txt`

van Heerden, R., Leenen, L., and Irwin, B. *Using an automated reasoner to classify computer network attacks.* In *5th Workshop on ICT Uses in Warfare and the Safeguarding of Peace*. November 2013a.

van Heerden, R., Leenen, L., Irwin, B., and Burke, I. *A computer network attack taxonomy and ontology. International Journal of Cyber Warfare and Terrorism*, 3:12–25, 2012a.

van Heerden, R., Pieterse, H., Burke, I., and Irwin, B. *Developing a virtualised testbed environment in preparation for testing of network based attacks.* In *5th Workshop on ICT Uses in Warfare and the Safeguarding of Peace*. November 2013b.

van Heerden, R. P., Burke, I., and Irwin, B. *Classifying network attack scenarios using an ontology.* In *Proceedings of the 7th International Conference on Information-Warfare & Security (ICIW 2012)*, pages 311–324. ACI, 2012b.

van Heerden, R. P., Pieterse, H., and Irwin, B. *Mapping the most significant computer hacking events to a temporal computer attack model.* In *International Conference on Human Choice and Computers (HCC10): ICT Critical Infrastructures and Society*, pages 226–236. IFIP, Springer, 2012c.

Vasiliadis, G., Polychronakis, M., Antonatos, S., Markatos, E. P., and Ioannidis, S. *Regular expression matching on graphics hardware for intrusion detection.* In *Recent Advances in Intrusion Detection*, pages 265–283. Springer, 2009.

Vasudevan, A. and Yerraballi, R. *Spike: Engineering malware analysis tools using unobtrusive binary-instrumentation.* In *Proceedings of the 29th Australasian Computer Science Conference*, pages 311–320. Australian Computer Society, Inc., 2006.

Verizon RISK Team. *2012 data breach investigations report.* Technical report, Australian Federal Police, Dutch National High Tech Crime Unit, Irish Reporting and Information Security Service, Police Central e-Crime Unit, and United States Secret Service, 2012. Accessed 2013/01/07.
URL `http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2012_en_xg.pdf`

Vigna, G., Robertson, W., and Balzarotti, D. *Testing network-based intrusion detection signatures using mutant exploits.* In *Proceedings of the 11th ACM conference on Computer and Communications Security*, pages 21–30. ACM, 2004.

Walker, S. *Edward snowden: first photo appears since russian asylum granted.* Online, October 2013. Accessed 2013/10/16.
URL `http://www.theguardian.com/world/2013/oct/10/edward-snowden-first-photo-russian-asylum`

Wang, K., Chen, X., and Xu, Y. *A brief study of trojan.* Technical report, Uppsala Unversitet, 2011. Accessed 2013/01/07.
URL `https://www.it.uu.se/edu/course/homepage/sakdat/vt09/pm/programme/trojans.pdf`

Washington, D. *Onward cyber soldiers. Time Magazine*, 21:1–5, 1995. Accessed 2013/01/07.
URL `http://www.time.com/time/magazine/article/0,9171,983318,00.html`

Wattanajantra, A. *Windows worm could create the 'world's biggest botnet'.* Online, January 2009. Accessed 2012/09/19.
URL `http://www.itpro.co.uk/609562/windows-worm-could-create-the-world-s-biggest-botnet`

White, S., Swimmer, M., Pring, E., Arnold, W., Chess, D., and Morar, J. *Anatomy of a commercial-grade immune system.* Technical report, IBM Research, 1999. Accessed 2013/01/08.
URL `http://www.research.ibm.com/antivirus/SciPapers/White/Anatomy/anatomy.html`

Whitman, M. and Mattord, H. *Principles of information security.* Course Technology Ptr, 2011, 8-9 pages.

Wiggins, G. *Living with malware.* Technical report, Sans Institute, 2001. Accessed 2013/01/08.
URL `http://www.sans.org/reading_room/whitepapers/malicious/living-malware_48`

Win, W. and Htun, H. H. *A simple and efficient framework for detection of sql injection attack. International Journal of Computer & Communication Engineering Research*, 1(2):26–30, 2013.

Wu, S. and Banzhaf, W. *The use of computational intelligence in intrusion detection systems: A review. Applied Soft Computing*, 10(1):1–35, 2010.

Xin, J., Dickerson, J., and Dickerson, J. *Fuzzy feature extraction and visualization for intrusion detection.* In *The 12th IEEE International Conference on Fuzzy Systems (FUZZ'03)*, pages 1249–1254. IEEE, 2003.

Yampolskiy, R. V. and Govindaraju, V. *Computer security: a survey of methods and systems. Journal of Computer Science*, 3(7):478–486, 2007.

Yang, D., Usynin, A., and Hines, J. W. *Anomaly-based intrusion detection for scada systems.* In *5th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (NPIC&HMIT 05)*, pages 12–16. 2006.

Ying, L., Dinglong, H., Haiyi, Z., and Rau, P. *Users' perception of mobile information security. Hacker Journals White Papers*, pages 1–5, 2007.

Yoder, R. M. *Someday they'll get Slick Willie Sutton. The Saturday Evening Post*, 223(30):17, January 1951.

Yung, K. H. *Detecting long connection chains of interactive terminal sessions.* In *Recent Advances in Intrusion Detection*, pages 1–16. Springer, 2002.

Zeng, X., Bagrodia, R., and Gerla, M. *Glomosim: a library for parallel simulation of large-scale wireless networks.* In *Proceedings of the Twelfth Workshop on Parallel and Distributed Simulation (PADS 98)*, pages 154–161. IEEE, 1998.

Zetter, K. *Google hack attack was ultra sophisticated, new details show.* Online, January 2010. Accessed 2012/08/12.
URL `http://www.wired.com/threatlevel/2010/01/operation-aurora/`

Zetter, K. *Anonymous hacks security firm.* Online, February 2011. Accessed 2013/01/08.
URL `http://www.wired.com/threatlevel/2011/02/anonymous-hacks-hbgary`

Zetter, K. *Google hackers had ability to alter source code.* Online, March 2012a. Accessed 2012/09/21.
URL `http://www.wired.com/threatlevel/2010/03/source-code-hacks/`

Zetter, K. *Hacker schools university in grade change caper.* Online, Novemeber 2012b. Accessed 2012/08/28.
URL `http://www.wired.com/threatlevel/2011/11/santa-clara-university-hacked/`

Zhai, J., Chen, Y., Yu, Y., Liang, Y., and Jiang, J. *Fuzzy semantic retrieval for traffic information based on fuzzy ontology and rdf on the semantic web. Journal of Software*, 4(7):758–765, 2009.

Zhou, C. V., Leckie, C., and Karunasekera, S. *A survey of coordinated attacks and collaborative intrusion detection. Computers & Security*, 29(1):124 –140, 2010.

Zou, C. C., Gao, L., Gong, W., and Towsley, D. *Monitoring and early warning for internet worms.* In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 190–199. 2003.

Zou, C. C., Gong, W., and Towsley, D. *Code red worm propagation modeling and analysis.* In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 138–147. ACM, 2002.

Zou, C. C., Towsley, D., and Gong, W. *Modeling and simulation study of the propagation and defense of internet e-mail worms. IEEE Transactions on Dependable and Secure Computing*, 4(2):105–118, 2007.

Appendices

APPENDIX

A

DETAIL OF LITERATURE SURVEY OF SIGNIFICANT COMPUTER-BASED ATTACKS

In this Appendix, the detail of the literature survey of signifcant computer based-attacks are shown in Table A.1 to Table A.12.

Table A.1: Malware – a Brief Timeline (Heater, 2011)

Date	**Malware**	**Brief Description**
1971	Creeper Virus	Regarded to be one of the first viruses.
1982	Eric Cloner	One of the first self-replicating virus's that affected personal computers. This virus was aimed at Apple II personal computers.
1986	Brain Virus	This virus is considered the first self-replicating virus for MS-DOS. This virus used floppy disks to transfer itself between computers.
1986	PC-Write Trojan	A trojan that was hidden in shareware.
1988	Morris Worm	The Morris worm was one of the first Internet-distributed worms.
1991	Michelangelo Virus	This virus erased data on a predefined day of the year (March 6).
1999	Melissa Virus	This virus is notable as the first mass-mailed virus
2000	I-LOVE-YOU	This worm spread itself to all available contacts in an e-mail client address book.
2003	SQL Slammer	In its time, it became the fastest spreading worm ever seen. It used database flaws to spread.
2005	Commwarrior-A	This virus targeted Symbian cell phones.
2005	Koobface	This virus used social networks to spread.
2008	Conficker	This complex worm used multiple infection vectors and inflicted significant damage.
2010	Stuxnet	A worm that targeted Iran's nuclear program.

Table A.2: Ten Most Costly Cyber-attacks in History (Julian, 2011)

Date	**Cyber Attacks**	**Brief Description**
2011	Citigroup	Citigroup customers' information were stolen.
2000	Titan rain	Titan rain is the FBI's name for an extensive cyber spying campaign for US military secrets.
2008	Heartland Payment Systems	Credit card numbers were stolen from Heartland servers. An estimated $140 million loss.
2007	Hannaford Bros	Over 4 million credit card numbers were stolen from the Hannaford Bros store servers. Estimated $252 million loss
2007	TJX	Over 45 million credit card numbers were stolen from the TJX retail company. An estimated $250 million loss.
2004	Sven Jaschan	Sven distributed a virus that damaged systems all over the world and caused damage of appoximate $500 million.
2000	Michael Calce	Michael, also known as Mafiaboy, disabled corporate company's networks (such as Dell, CNN, Amazon and Ebay) and caused an estimated damage of $1.2 billion.
2011	Sony Playstation	Over 100 million Sony Playstation accounts were breached and credit card information stolen. The estimated damage was between $1 billion and $2 billion.
2011	Epsilon	E-mail addresses of large corporations such as Best Buy and JP Morgan Chace were stolen when Epsilon's e-mail-handling servers was hacked. The damage is estimated to be between $225 million and $4 billion.
1982	The Logic Bomb	The CIA blew up a Siberian gas pipeline by inserting malicious code into the computer systems controlling the gas pipeline.

Table A.3: The 12 Costliest Computer Viruses Ever (Miranda, 2010)

Date	**Virus**	**Brief Description**
1988	Morris Worm	The Morris worm was one of the first Internet-distributed worms.
2003	Blaster Worm	This worm targeted Microsoft's windowsupdate.com site and caused Windows PCs to crash as soon as they connected to a network.
2004	Sasser Worm	The Sasser worm made use of a buffer overflow in the component LSASS (Local Security Authority Subsystem Service.
2001	Nimda	A Virus that infected though several vectors and thus caused significant damage.
2003	SQL Slammer	In its time, it became the fastest spreading worm ever seen. It used database flaws to spread.
2001	SirCam	A file-based virus that attacked computers after itself was opened.
1999	Melissa	A virus embedded in a Microsoft Word document. When opened, the macro in the document mass-e-mailed itself to the first 50 entries in the user's address book.
2001	Code Red	The worm exploited a vulnerability in Internet Information Services (IIS) from Microsoft, and spread itself using the buffer overflow technique.
2008	Conficker	This complex worm used multiple infection vectors and inflicted significant damage.
2000	I-LOVE-YOU	This worm spread itself to all available contacts in an e-mail client address book.
2003	SoBig	A virus that spread through e-mail attachments.
2004	MyDoom	A worm that infected computers and sent spam e-mails. This worm slowed down the Internet by 10%.

Table A.4: The Seven Worst Cyber-attacks in History (That We Know About) (Hall, 2010)

Date	Cyber Warfare	Brief Description
2000	Titan rain	Titan rain is the FBI's name for an extensive cyber spying campaign for US military secrets.
1999	Moonlight Maze	Hackers compromised American computer systems (Pentagon, NASA, the Department of Energy and others).
2007	The Estonian Cyberwar	Estonian computer networks were flooded and disabled after the Bronze Soldier of Tallinn statue was removed and Russian sentiments angered.
2008	Presidential-level Espionage	Foreign sources successfully used computer attacks against the computers used by Obama and McCain during their presidential campaigns.
2007	China's "750,000 American zombies"	Paul Strassmann (US information security official) stated that there were 735,598 compromised computers "infested by Chinese zombies." (Hall, 2010).
1982	The Logic Bomb	The CIA blew up a Siberian gas pipeline by inserting malicious code into the computer systems controlling the gas pipeline.
2008	The Most Serious Breach	A flash drive was inserted into a military laptop in the Middle East, creating a digital staging post, from which data were transferred to servers under foreign control.

Table A.5: The Decade's Biggest Cyber Crime Attacks Exploits (Marcus, 2011)

Date	Exploits	Brief Description
2004	MyDoom	A worm that infected computers and sent spam e-mails. This worm slowed down the Internet by 10%.
2000	I-LOVE-YOU	his worm spread itself to all available contacts in an e-mail client address book.
2007	Conficker	This complex worm used multiple infection vectors and inflicted significant damage (Other researchers states that Conficker only surfaced in 2008).
2010	Stuxnet	A worm that targeted Iran's nuclear program.
2007	Zeus Botnet	An information stealing botnet that has been used to steal Internet Banking details and other identity information. Over 700 variants of the Zeus botnet have been found.

Table A.6: The Decade's Biggest Cyber-crime Attacks Scams (Marcus, 2011)

Scams	**Brief Description**
Scareware	Fake anti-virus software that is used to induce unsuspecting users into installing malware.
Phishing Scams.	The art of tricking users into freely giving away personal information by spoofing legitimate information requests.
Phony Websites	Fake websites that simulate e-commerce sites such as banking or auction websites.
Online Dating Scams	Fake personal relationships are created in order to steal information or extract cash.
419 Scam	An e-mail message that requests help in moving money from a foreign country. The scammer asks for an up-front transfer or other related fees that are then never returned.

Table A.7: Ten Worst Cyber-crimes of the Decade (Buckland, 2011)

Date	**Cybercrimes**	**Brief Description**
2000	I-LOVE-YOU	This worm spread itself to all available contacts in an e-mail client address book.
2000	Michael Calce	Michael, also known as Mafiaboy, disabled corporate companies (such as Dell, CNN, Amazon and Ebay) and caused an estimated damage of $1.2 billion.
-	419 Scam	An e-mail message that requests help in moving money from a foreign country. The scammer asks for an up-front transfer or other related fees that are then never returned.
2004	MyDoom	A worm that infected computers and sent spam e-mails. This worm slowed down the Internet by 10%.
2005	Operation Get Rich or Die Trying	Albert Gonzalez and his conspirators stole credit card information from major global retailers.
2006	The L.A. traffic signal attack	Los Angeles' city traffic signals were changed by a malicious engineer during a strike. That caused traffic to be gridlocked for days.
2006	John Dillinger returns	Credit card thefts were being attributed to alias named "John Dillinger", a famous Great Depression-era bank robber. Polish and Romanian criminals were arrested for selling credit card information.
2008	Conficker	This complex worm used multiple infection vectors and inflicted significant damage.
2009	Facebook Profile Spy	By promising to let users know who views their facebook profiles, this software hacked 500 000 Facebook accounts and sent out fake "Help! I've been robbed!" phishing spam.
2006	WikiLeaks	Leaked official documents are posted to WikiLeaks.

Table A.8: The Decade's Ten Most Dastardly Cyber-crimes (Poulsen, 2009)

Date	**Cybercrimes**	**Brief Description**
2000	Michael Calce	Michael, aka Mafiaboy, disabled corporate companies (such as Dell, CNN, Amazon and Ebay) and caused an estimated damage of $1.2 billion by attacking their web services.
2002	California Payroll Database Breach	A California server housing the state government's payroll database was compromised with names, Social Security numbers and salary information for 265,000 state workers.
2003	Sasser Worm	The Sasser worm made use of a buffer overflow in the component LSASS (Local Security Authority Subsystem Service).
2004	Foonet	A small ISP hosted in Ohio (US) that was the first blackhat (hackers for hire) hosting company.
2005	Operation Get Rich or Die Trying	Albert Gonzalez and his conspirators stole credit card information from major global retailers.
2006	The L.A. traffic signal attack	Los Angeles' city traffic signals were changed by a malicious engineer during a strike. That caused traffic to be gridlocked for days.
2006	Max Vision	Max Vision hacked other hackers collections' of stolen credit card and he opened his own site own site, CardersMarket to sell the numbers.
2008	RBS Worldpay Heist	Payment processor RBS Worldpay had been hacked and $9.5 million dollars were stolen by cashiers to slam the accounts with repeated rapid-fire withdrawals.
2009	Conficker	This complex worm used multiple infection vectors and inflicted significant damage.
2009	Money Mules	Small businesses that use on-line banking are targeted by trojan horses software that steal credentials and initiate wire transfers from their accounts.

Table A.9: Fifteen Worst Data Breaches (Armerding, 2012)

Date	**Data Breach**	**Brief Description**
2008	Heartland Payment Systems	134 million credit card numbers were stolen. SQL injection was used to install spyware.
2006	TJX	94 million credit cards exposed because the corporate network was not protected by a firewall.
2011	Epsilon	Names and e-mails of millions of customers of retail stores plus several financial institutions were compromised.
2011	RSA	An estimated 40 million employee records were stolen as well as information on the company's SecurID authentication tokens.
2010	Stuxnet	A worm that targeted Iran's nuclear program.
2006	Department of Veterans Affairs	A database with personal information for over 26 million USA veterans, active-duty military personnel and their spouses were stolen
2011	Sony Playstation	Over 100 million Sony Playstation accounts was breached and credit card information stolen. The estimated damage was between $1 billion and $2 billion.
2011	ESTsoft	Personal information of 35 million South Koreans were stolen from ESTsoft database.
2010	Gawker Media	1.3 Million e-mail addresses and passwords were stolen from popular blogs and the source code from Gawkerś content management system.
2009	Google, ect	The Chinese government launched a massive and unprecedented attack on Google, Yahoo, and several computer and communications companies. A security hole in an older version of Internet Explorer was used.
2010	VeriSign	The incidents only became public after mandatory Security Exchange fillings listed the network breaches.
2005	CardSystems	40 million credit card accounts numbers were stolen.
2006	AOL	More than 20 million private web-page inquiries were exposed.
2007	Monster.com	Personal information of 1.3 million job seekers were stolen. The information was then used in a phishing scam.
2007	Fidelity National Information Services	Information from 3.2 million private customers were stolen by an employee.

Table A.10: Top Ten Hacks of All Time Liddinton-Cox (2012)

Date	**Hack**	**Brief Description**
2011	Sony Playstation	Over 100 million Sony Playstation accounts were breached and credit card information stolen. The estimated damage was between $1 billion and $2 billion.
2011	Epsilon	e-mail addresses of large corporations such as Best Buy and JP Morgan Chace were stolen when Epsilon's e-mail-handling servers was hacked. The damage is estimated to be between $225 million and $4 billion.
2000	Titan rain	Titan rain is the FBI's name for an extensive cyber spying campaign for US military secrets by unknown actors.
2009	Google, ect	The Chinese government launched a massive and unprecedented attack on Google, Yahoo, and several computer and communications companies. A security hole in an older version of Internet Explorer was used.
2004	MyDoom	A worm that infected computers and sent spam e-mails. This worm slowed down the Internet by 10%.
2003	Adrian Lamo	Adrian hacked Microsoft, Yahoo, Bank of America, Cingular and Citigroup.
1988	Morris Worm	The Morris worm was one of the first Internet-distributed worms.
2006	WikiLeaks	Leaked official documents are posted to WikiLeaks.
1995	Kevin Mitnick	Famous hacker that was a very skilled social engineer.
2000	Michael Calce	Michael, aka Mafiaboy, disabled corporate companies (such as Dell, CNN, Amazon and Ebay) and caused an estimated damage of $1.2 billion.

Table A.11: Best Known Cyber-attacks of All Time (Tech Analyser, 2011)

Date	Cyber Attack	Brief Description
1971	Creeper	Software that spread through ARPANET and displayed "I'M THE CREEPER CATCH ME IF YOU CAN".
1982	Eric Cloner	A virus that spread through floppy disks and displayed a poem.
1986	Brain Virus	This virus is considered the first self-replicating virus for MS-DOS. This virus used floppy disks to transfer itself between computers.
1988	Morris Worm	The Morris worm was one of the first Internet-distributed worms.
1991	Michelangelo Virus	This virus erased data on a predefined day of the year (March 6).
1995	Macro	Macro virus was written in order to prove that macro viruses can spread
1999	Melissa Virus	This virus is notable as the first mass-mailed virus
2000	I-LOVE-YOU	This worm spread itself to all available contacts in an e-mail client address book.
2001	Code Red	The worm exploited a vulnerability in Internet Information Services (IIS) from Microsoft, and spread itself using the buffer overflow technique.
2001	Nimda	The Nimda worm used five different infection vectors to spread.
2003	SQL Slammer	In its time, it became the fastest spreading worm ever seen. It used Database flaws to spread.
2003	Blaster Worm	This worm targeted Microsoft's windowsupdate.com site and caused Windows PCs to crash as soon as they connected to a network
2004	MyDoom	A worm that infected computers and sent spam e-mails. This worm slowed down the Internet by 10%.
2004	Sasser Worm	The Sasser worm made use of a buffer overflow in the component LSASS (Local Security Authority Subsystem Service
2007	Storm Botnet Worm	The Storm Botnet used a Trojan horse that spread through e-mail spam, and gathered infected computers into a remotely controlled network of zombied (hijacked) computers.
2009	July 2009 cyber attacks	North Korea's telecommunications ministry spread malicious code that copies data to an encrypted file, and then overwrites the original files.
-	Autorun	This virus use flash drives as its source of propagation.

Table A.12: Ten Worst Computer Viruses of All Time (Strickland, 2008)

Date	**Malware**	**Brief Description**
1999	Melissa Virus	This virus is notable as the first mass-mailed virus
2000	I-LOVE-YOU	This worm spread itself to all available contacts in an e-mail client address book.
2001	The Klez Virus	e-mail virus that spoof the "From" field in e-mail headers.
2001	Code Red	The worm exploited a vulnerability in Internet Information Services (IIS) from Microsoft, and spread itself using the buffer overflow technique.
2001	Nimda	The Nimda worm used five different infection vectors to spread.
2003	SQL Slammer	In its time, it became the fastest spreading worm ever seen. It used Database flaws to spread.
2004	MyDoom	A worm that infected computers and sent spam e-mails. This worm slowed down the Internet by 10%.
2004	Sasser Worm	The Sasser worm made use of a buffer overflow in the component LSASS (Local Security Authority Subsystem Service.
2006	Leap-A	A virus that targeted Mac Computers (made by Apple Computers).
2007	Storm Botnet Worm	The Storm Botnet used a Trojan horse that spread through e-mail spam, and gathered infected computers into a remotely controlled network of zombied (hijacked) computers.

APPENDIX

B

EVENT QUERIES

In this Appendix, Event Queries are described and mapped to their respective *Attack Scenarios*. Protégé with the HermiT reasoner (Section 5.2) are used to determine which scenario is detected by which Event Query. The *Traffic Influx* and *Unusual Web Activity Port Scan* are presented in Section 8.4.1.

B.1 Traffic Influx

The *Traffic Influx* Event Query determines if the amount of incoming traffic has increased. The *Firewall* Sensor (Section C.3) is used to determine the amount of incoming traffic. This Event Query is triggered when the amount of incoming traffic exceeded a specified threshold and the amount of data increased significantly.

The algorithm used for the *Traffic Influx* Event Query is shown in Listing 9 (similar to the algorithm shown at for Listing 3).

The HermiT automatic reasoner used the following question to determine which scenario is detected by the *Traffic Influx* Event Query:

- has at least one:
  - *Disrupt Attack Goal* defined by *hasChainActorAttackGoal* relationship

Listing 9 Traffic Influx Event Query

```
Traffic Influx Event Query Start
Load Traffic Influx Settings
Retrieve Bandwidth Sensor Data
Calculate Bandwidth used
Calculate Bandwidth Threshold
if Bandwidth exceeds Bandwidth Threshold then
   Traffic Influx Event Query detected
end if
Traffic Influx Event Query End
```

- *Access Asset* defined by *hasChainTargetAsset* relationship
- *Network Infrastructure Device Target* defined by *hasTarget* relationship
- *Denial of Service Attack Mechanism* defined by the *hasAttackMechanismRampup* relationship

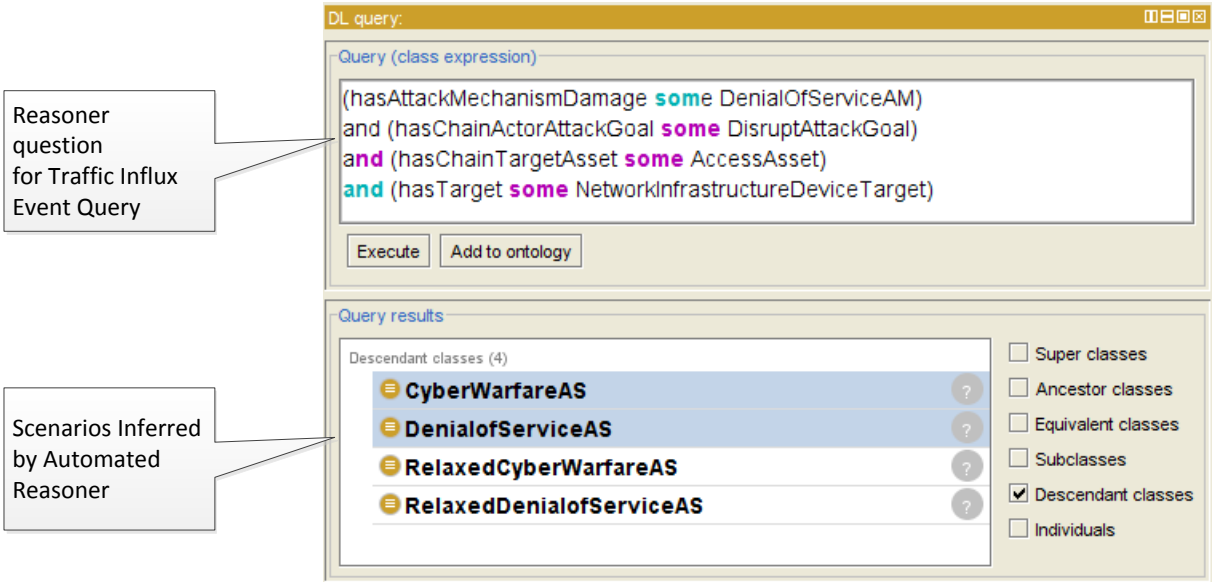


Figure B.1: HermiT automatic reasoner Traffic Influx Event Query result

The HermiT automatic reasoner with the *Traffic Influx* Event Query calculated the the following scenario:

- Cyber-Warfare
- Denial-of-Service
- Relaxed Cyber-Warfare
- Relaxed Denial-of-Service

The *Cyber-Warfare* and *Denial-of-Service* scenario have been proven equivalent for near-

Listing 10 Servers Running Event Query

```
Servers Running Event Query Start
Servers Running Settings
Retrieve IsAlive Sensor Data
Calculate the number of the servers that are Communicating
if Servers down greater than set threshold OR IsAlive Sensor stopped communicating
then
  Servers Running Event Query detected
end if
Servers Running Event Query End
```

real time environment (Section 7.3.2), thus this Event Query will be classified within Denial-of-Service. The phase is defined as *Ramp-up* because the *Attach Mechanism* in the query is defined as *Ramp-up*. In Figure B.1, the result of the *Traffic Influx* Event Query in Protégé is shown.

B.2 Servers Running

The *Servers Running* Event Query determines if one can communicate with the servers through the network. The *IsAlive* Sensor (Section C.2) is used to test the communications ability of the servers. This Event Query is triggered when the number of servers that cannot communicate falls below a specified threshold.

The algorithm used for the *Servers Running* Event Query is shown in Listing 10 (similar to the algorithm shown at for Listing 10).

The HermiT automatic reasoner used the following query to determine which scenario is detected by the *Servers Running* Event Query:

- has at least one:
  - *Disrupt Attack Goal* defined by *hasChainActorAttackGoal* relationship
  - *Access Asset* defined by *hasChainTargetAsset* relationship
  - *Network Infrastructure Device Target* defined by *hasTarget* relationship
  - *Denial-of-Service Attack Mechanism* defined by the *hasAttackMechanismDamage* relationship

The HermiT automatic reasoner with the *Servers Running* Event Query calculated the the following scenario:

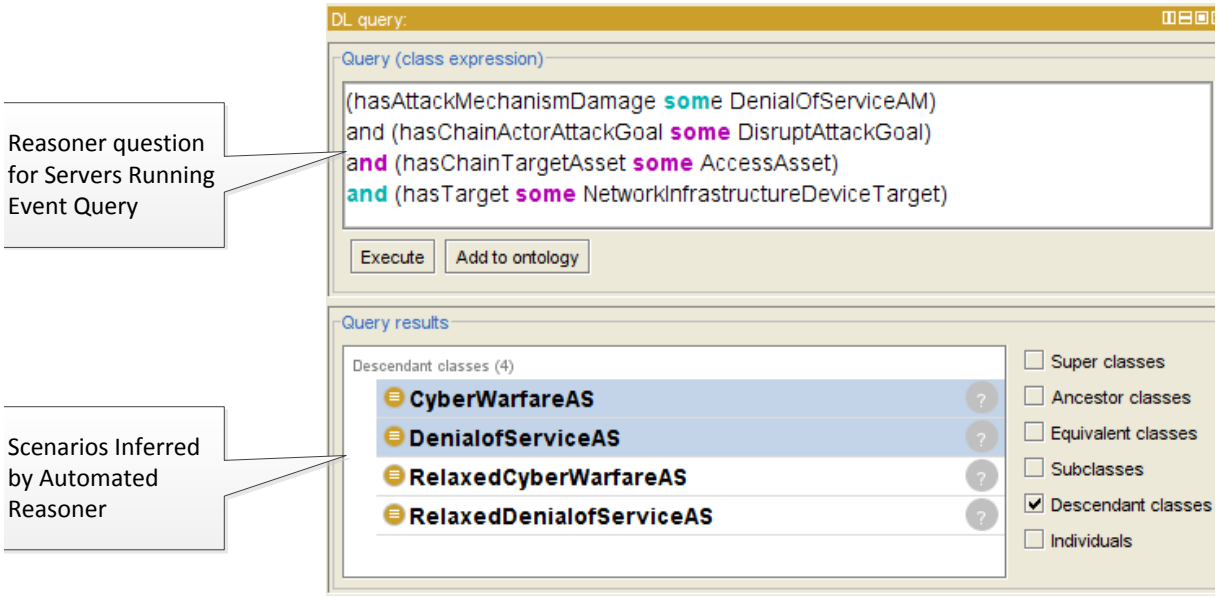


Figure B.2: HermiT automatic reasoner Servers Running Event Query result

- Cyber-Warfare
- Denial-of-Service
- Relaxed Cyber-Warfare
- Relaxed Denial-of-Service

The *Cyber-Warfare* and *Denial-of-Service* scenarios have been proven equivalent for the near real-time environment (Section 7.3.2), thus this Event Query will be classified within Denial-of-Service. The phase is defined as *Damage* because the *Attach Mechanism* in the query is defined as *Damage*. In Figure B.2, the result of the *Servers Running* Event Query in Protégé is shown.

B.3 Unusual Web Activity

The *Unusual Web Activity* Event Query looks for non-human web-crawling type activities. The *Crawler Detector* sensor (section 8.5.3) detects if a website has been accessed by a non-human looking for vulnerabilities. This is typically done with a web crawler. A web crawler is software that systematically scans and harvests information of a web site. The algorithm used for the *Unusual Web Activity* Event Query is shown in Listing 11.

The HermiT automatic reasoner used the following query to determine which scenario is detected by the *Unusual Web Activity* Event Query:

Listing 11 Servers Running Event Query

```
Unusual Web Activity Event Query Start
Load Unusual Web Activity Settings
if Crawler detected unusual activity then
   Unusual Web Activity Event Query detected
end if
Unusual Web Activity Event Query End
```

- has at least one:
  - *Web Server Target* defined by *hasTarget* relationship
  - *Web Crawl Attack Mechanism* defined by the *hasAttackMechanismRec* relationship

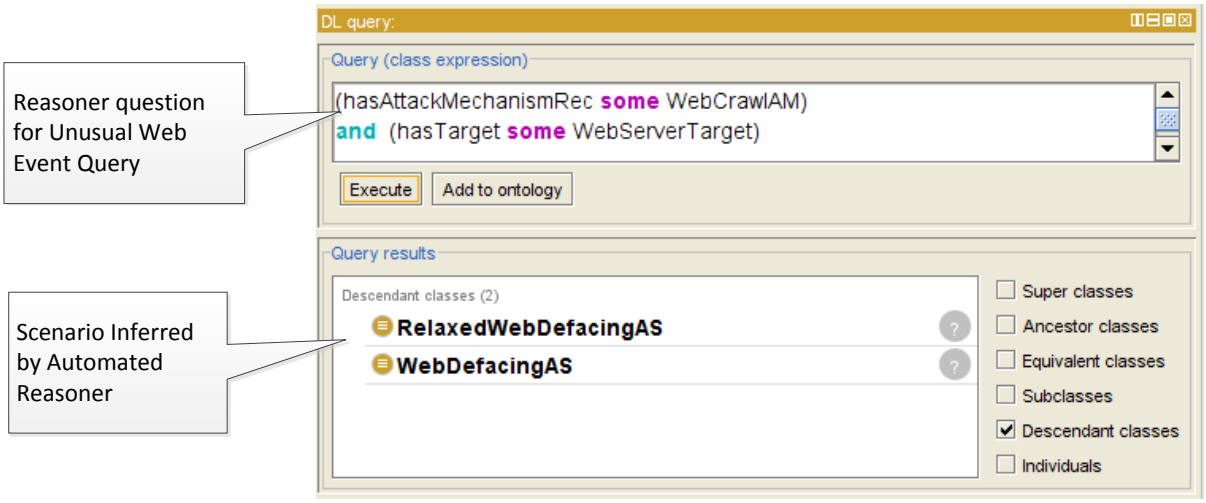


Figure B.3: HermiT automatic reasoner Unusual Web Activity Event Query result

The HermiT automatic reasoner with the *Unusual Web Activity* Event Query calculated the following scenario:

- Web Defacement
- Relaxed Web Defacement

The phase is defined as *Reconnaissance* because the *Attach Mechanism* in the query is defined as *Reconnaissance*. In Figure B.3, the result of the *Unusual Web Activity* Event Query in Protégé is shown.

Listing 12 Web Defacement Event Query

```
Web Defacement Event Query Start
Load Web Defacement Settings
Retrieve Deface Sensor Data
Determine if the website has been defaced
if Website was defaced then
   Web Defacement Event Query detected
end if
Web Defacement Event Query End
```

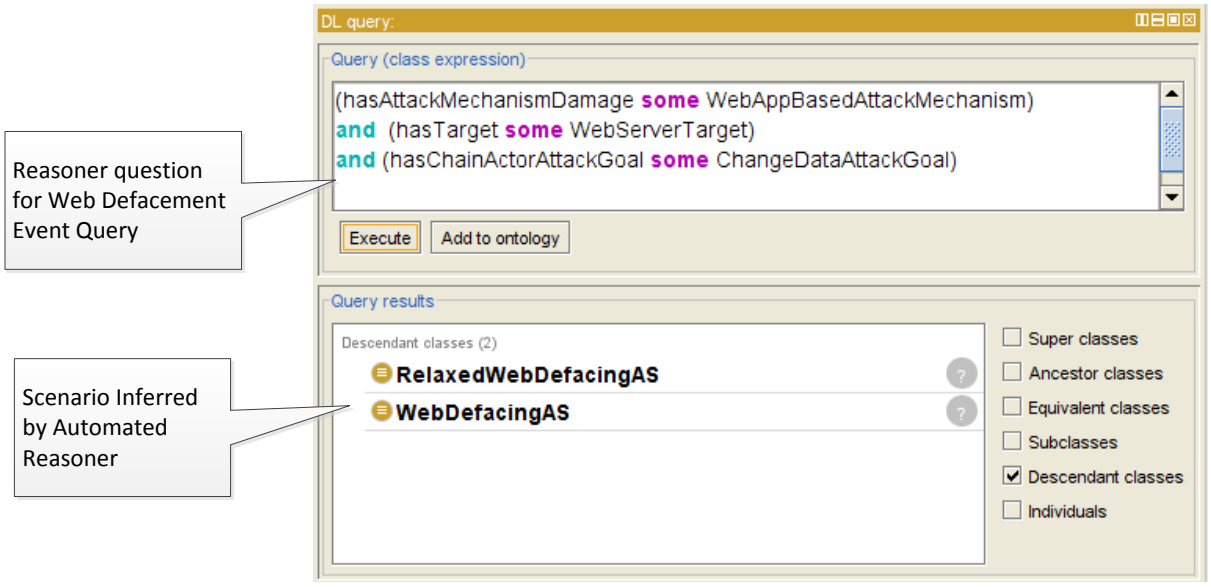


Figure B.4: HermiT automatic reasoner Web Defacement Event Query result

B.4 Web Defacement

The *Web Defacement* Event Query is triggered if the website has changed without approval. The *Deface* Sensor (Section C.4) is used to detect a vulnerability scan on the web server. This Event Query triggers as soon as a vulnerability scan on the web server is detected. The algorithm used for the *Web Defacement* Event Query is shown in Listing 12.

The HermiT automatic reasoner used the following query to determine which scenario is detected by the *Web Defacement* Event Query:

- has at least one:
  - *Web Server Target* defined by *hasTarget* relationship
  - *Change Data* defined by *hasChainActorAttackGoal* relationship

Listing 13 Failed Login Attempts Event Query

```
Failed Login Attempts Event Query Start
Load Failed Login Attempts Settings
Retrieve Failed Logins Sensor Data
Determine an Unsuccessful Occurred
if Unsuccessful Login then
  Failed Login Attempts Event Query detected
end if
Failed Login Attempts Event Query End
```

  - *Web Crawl Attack Mechanism* defined by the *hasAttackMechanismDamage* relationship

The HermiT automatic reasoner with the *Web Defacement* Event Query calculated the the following scenario:

- Web Defacement
- Relaxed Web Defacement

The phase is defined as *Damage* because the *Attach Mechanism* in the query is defined as *Damage*. In Figure B.4, the result of the *Web Defacement* Event Query in Protégé is shown.

B.5 Failed Login Attempts

The *Failed Login* Event Query determines if incorrect login attempts have been made. This can be an indication that an attacker is trying to login, but was unsuccessful. The *Failed Login Attempts* sensor (Section C.8) is used to detect unsuccessful logins. The algorithm used for the *Failed Login Attempts* Event Query is shown in Listing 13.

The HermiT automatic reasoner used the following query to determine which scenario is detected by the *Failed Login Attempts* Event Query:

- has at least one:
  - *Change Data* ***or*** *Steal Data* ***or*** *Gain Control* defined by *hasChainActorAttackGoal* relationship
  - *Open Information Attack Mechanism* defined by the *hasAttackMechanismDamage* relationship

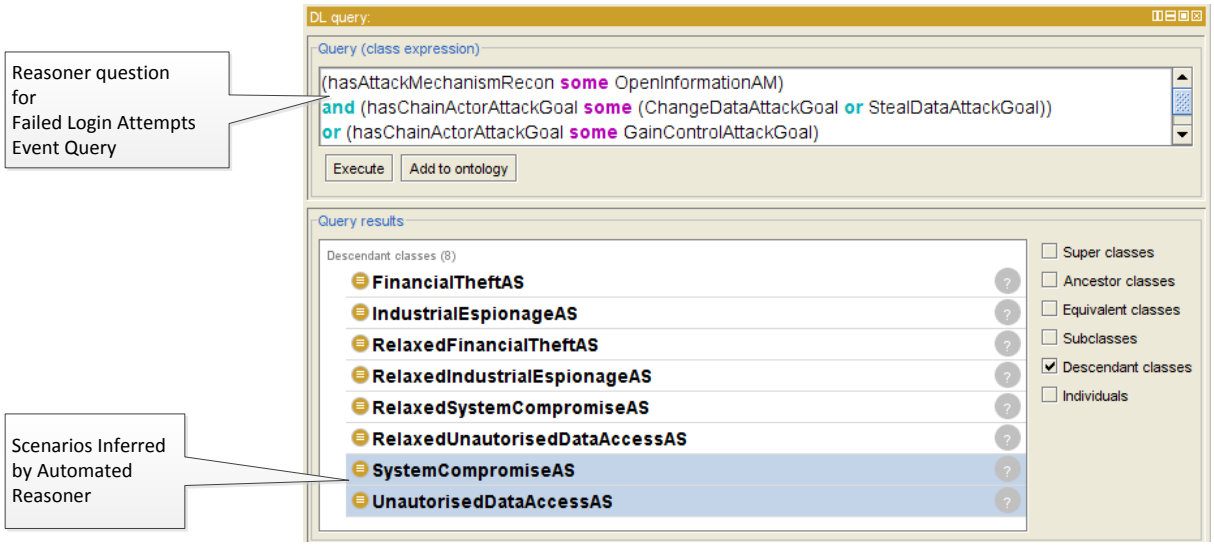


Figure B.5: HermiT automatic reasoner Failed Login Attempts Event Query result

The HermiT automatic reasoner with the *Failed Login Attemps* Event Query calculated the following scenario:

- System Compromise
- Unauthorised Data Access

The phase is defined as *Reconnaissance* because the *Attach Mechanism* in the query is defined as *Reconnaissance*. In Figure B.5, the result of the *Failed Login Attempts* Event Query in Protégé is shown.

B.6 Runaway Malware: Single and Multiple

The *Runaway Malware: Single* and *Runaway Malware: Multiple* Event Query detects self-spreading malware. The malware is still limited to a single host, *Runaway Malware: Single* is set, if the malware is detected from multiple hosts, and *Runaway Malware: Multiple* is set. The *Network Telescope Sensor* (Section 8.5.1) is used to detected self-spreading malware.

As soon as the network telescope detects attempted communications from an infected host the Event Query is triggered. The algorithm used for the *Runaway Malware: Single* and *Runaway Malware: Multiple* EQs is shown in Listing 14.

The HermiT automatic reasoner used the following query to determine which scenario is detected by the *Runaway Malware* Event Query:

Listing 14 Runaway Malware: Single Event Query

```
Runaway Malware Event Query Start
LoadRunaway Malware: Single Settings
if Network Telescope Sensor Detects Malware then
  if Malware From a Single Host then
    Runaway Malware: Single Event Query detected
  end if
  if Malware From Multiple Hosts then
    Runaway Malware: Multiple Event Query detected
  end if
end if
Runaway Malware Event Query End
```

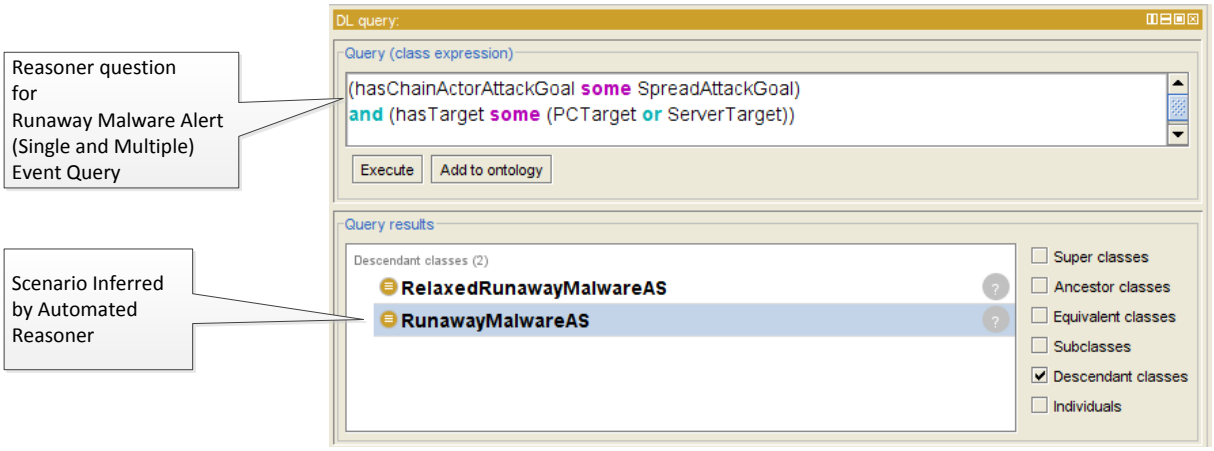


Figure B.6: HermiT automatic reasoner Runaway Malware Event Query result

Listing 15 Unusual Bandwidth Event Query

```
Unusual Bandwidth Query Start
Load Failed Unusual Bandwidth Settings
Retrieve Firewall Bandwidth Monitor Sensor Data
Determine if an Unusual Amount of Bandwidth has been used
if Unusual Amount of Bandwidth then
  Unusual Bandwidth Event Query detected
end if
Unusual Bandwidth Event Query End
```

- has at least one:
  - *Spread* defined by *hasChainActorAttackGoal* relationship;
  - *PC* ***or*** *Server Target* defined by the *hasTarget* relationship;

If a host source of malware is detected, the phase is defined as *Reconnaissance* and the Event Query *Runaway Malware Alert: Single* is triggered. If multiple most sources of malware are detected the phase is defined as *Ramp-up* and the Event Query *Runaway Malware Alert: Multiple* is triggered. In Figure B.6, the result of the *Runaway Malware* Event Queries in Protégé is shown.

B.7 Unusual Bandwidth

The *Unusual Bandwidth* Event Query is triggered if the amount of bandwidth used is not within usual parameters. This can occur when an attacker uses network resources to launch DDoS attacks or hosts a warez site remotely.

The *Unusual Bandwidth* sensor (Section C.3) is used to detect unsuccessful logins. The algorithm used for the *Unusual Bandwidth* Event Query is shown in Listing 15.

The HermiT automatic reasoner used the following query to determine which scenario is detected by the *Unusual Bandwidth* Event Query:

- has at least one:
  - *Gain Resources Control* defined by *hasChainActorAttackGoal* relationship
  - *System Abuse Attack Mechanism* defined by the *hasAttackMechanismDamage* relationship

The HermiT automatic reasoner with the *Unusual Bandwidth* Event Query calculated the the following scenario:

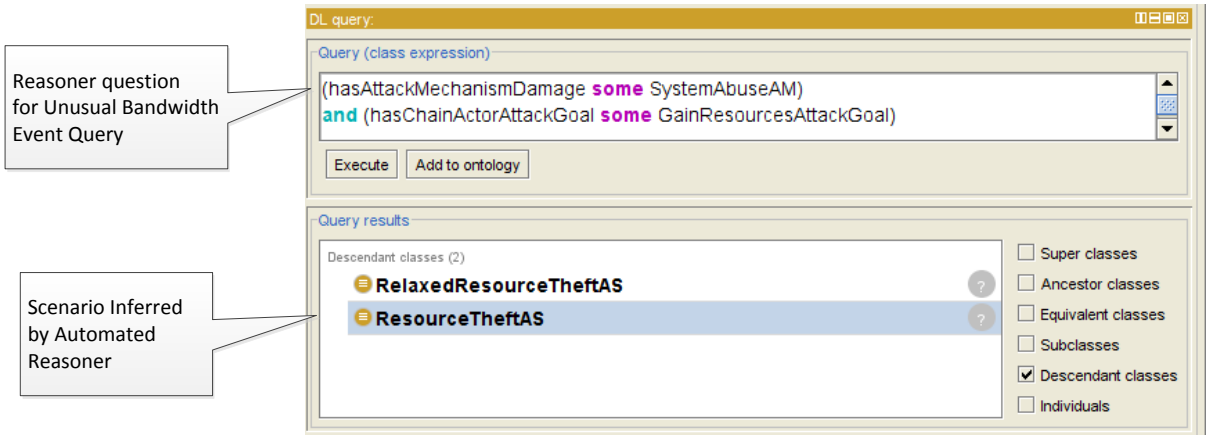


Figure B.7: HermiT automatic reasoner Unusual Bandwidth Event Query result

Listing 16 Unusual Disk Usage Event Query

```
Unusual Disk Usage Query Start
Load Failed Unusual Disk Usage Settings
Retrieve Unusual Disk Usage Sensor Data
Determine if Disk has been Unauthorised
if Unauthorised Disk Usage then
   Unusual Disk Usage Event Query detected
end if
Unusual Disk Usage Event Query End
```

- Resource Theft

The phase is defined as *Damage* because the *Attach Mechanism* in the query is defined as *Damage*. In Figure B.7, the result of the *Unusual Bandwidth* Event Query in Protégé is shown.

B.8 Unusual Disk Usage

The *Unusual Disk Usage* Event Query is triggered if the disk space is being used by an unauthorised actor. This can typically occur when an attacker hosts warez or multmedia on corporate data storage. The *Unusual Disk Usage* sensor (Section C.9) is used to detect unsuccessful logins. The algorithm used for the *Unusual Disk Usage* Event Query is shown in Listing 16.

The HermiT automatic reasoner used the following query to determine which scenario is detected by the *Unusual Disk Usage* Event Query:

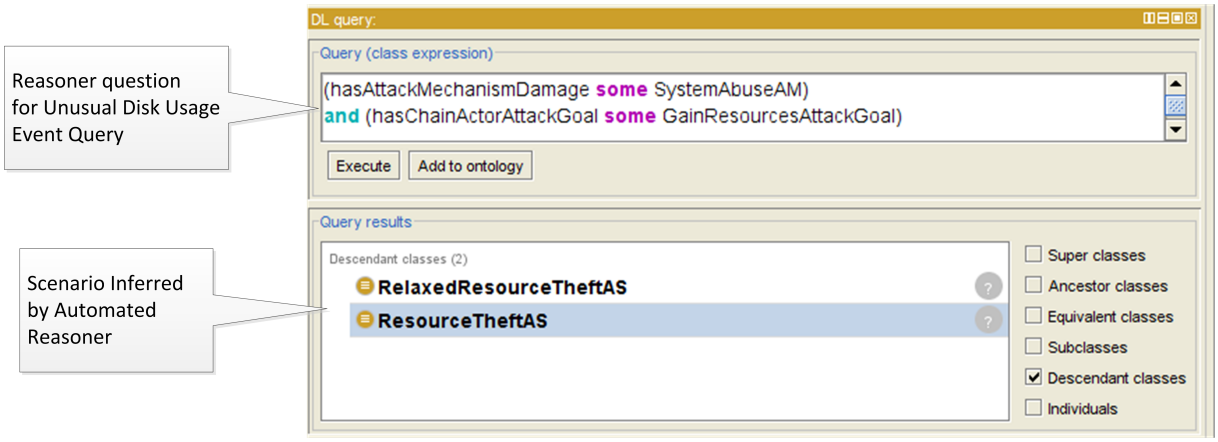


Figure B.8: HermiT automatic reasoner Unusual Disk Usage Event Query result

- has at least one:
  - *Gain Resources Control* defined by *hasChainActorAttackGoal* relationship
  - *System Abuse Attack Mechanism* defined by the *hasAttackMechanismDamage* relationship

The HermiT automatic reasoner with the *Unusual Disk Usage* Event Query calculated the following scenario:

- Resource Theft

The phase is defined as *Damage* because the *Attach Mechanism* in the query is defined as *Damage*. In Figure B.8, the result of the *Unusual Disk Usage* Event Query in Protégé is shown.

B.9 Hidden Data Accessed

The *Hidden Data Accessed* Event Query is set when data that should never be seen is accessed. The *Tripwire Access* sensor (Section C.1) is used to detect unsuccessful logins. The algorithm used for the *Hidden Data Accessed* Event Query is shown in Listing 17.

The HermiT automatic reasoner used the following query to determine which scenario is detected by the *Hidden Data Accessed* Event Query:

- has at least one:
  - *Change Data* ***or*** *Steal Data* defined by *hasChainActorAttackGoal* relationship

Listing 17 Hidden Data Accessed Event Query

```
Hidden Data Accessed Usage Query Start
Load Hidden Data Accessed Settings
Retrieve Tripwire Access Sensor Data
Determine if TripWire Sensor has been Triggered
if TripWire was Triggered then
   Hidden Data Accessed Event Query detected
end if
Hidden Data Accessed Event Query End
```

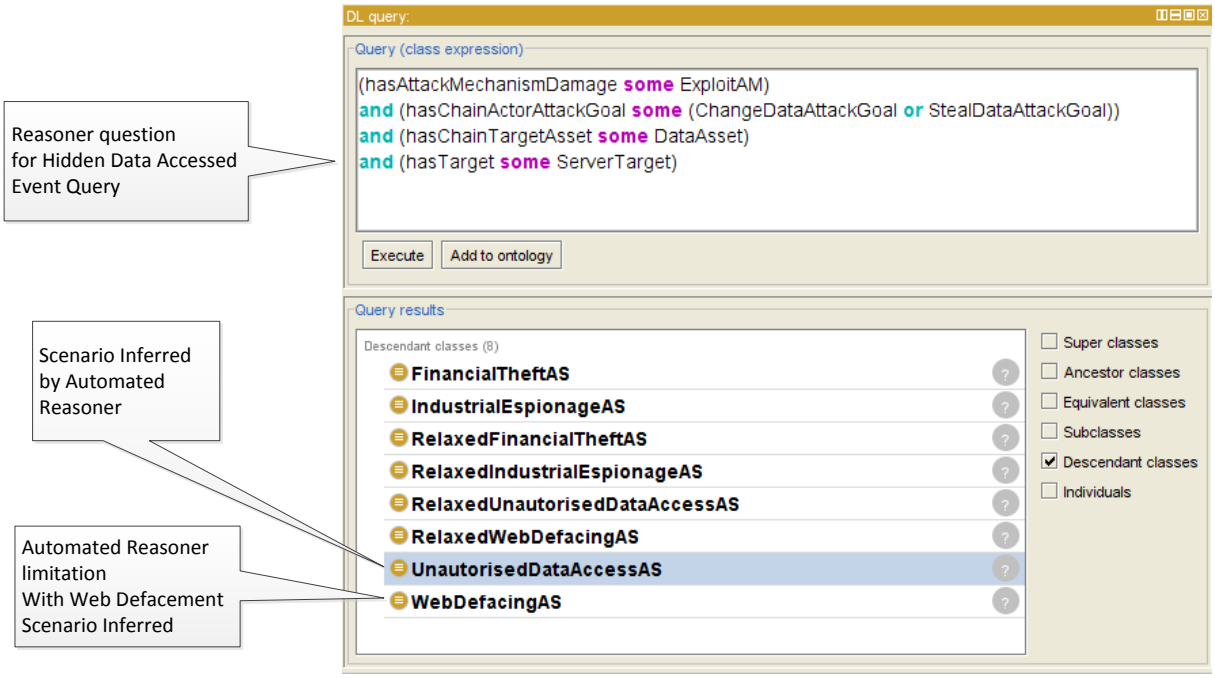


Figure B.9: HermiT automatic reasoner Hidden Data Accessed Event Query result

Listing 18 Unauthorised Super User Event Query

```
Unauthorised Super User Query Start
Load Unauthorised Super User Settings
Retrieve Root Login Sensor Data
Determine if Server has been Remotely Accessed
if Server has been Remotely Accessed then
  Unauthorised Super User Event Query detected
end if
Unauthorised Super User Event Query End
```

- *Exploit Attack Mechanism* defined by the *hasAttackMechanismDamage* relationship
- *Data Asset* defined by the *hasChainTargetAsset* relationship;
- *Server Target* defined by the *hasTarget* relationship;

The HermiT automatic reasoner with the *Hidden Data Accessed* Event Query calculated the following scenario:

- Unauthorised Data Access
- Web Defacement

The phase is defined as *Damage* because the *Attach Mechanism* in the query is defined as *Damage*. In Figure B.9, the result of the *Unusual Bandwidth* Event Query in Protégé is shown. The *Web Defacement* attack scenario is identified because a *WebServer* is a subset of the *Server* class. The automated reasoner has the limitation that its query cannot search match a class without a specified sub-class. This problem can potentially be corrected by moving the *Web Server* class out from within *Server* class.

B.10 Unauthorised Super User

The *Unauthorised Super User* Event Query is triggered when a server administrator account is accessed remotely. The server should only be accessed locally. The *Root Login* sensor (Section C.9) is used to detect unsuccessful logins. The algorithm used for the *Unusual Disk Usage* Event Query is shown in Listing 18.

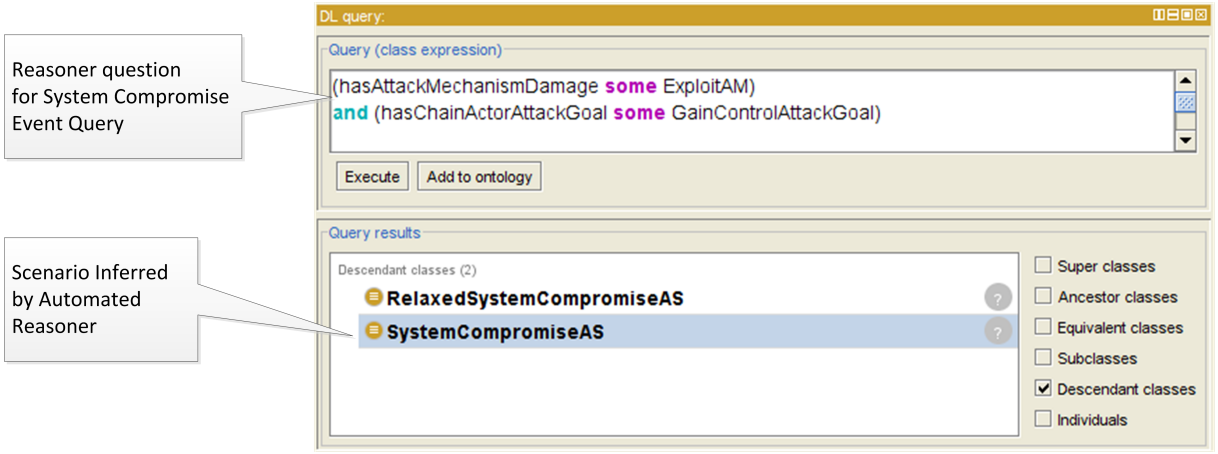


Figure B.10: HermiT automatic reasoner Unauthorised Super User Event Query result

The HermiT automatic reasoner used the following query to determine which scenario is detected by the *Unauthorised Super User* Event Query:

- has at least one:
  - *Gain Resources Control* defined by *hasChainActorAttackGoal* relationship
  - *System Abuse Attack Mechanism* defined by the *hasAttackMechanismDamage* relationship

The HermiT automatic reasoner with the *Unusual Disk Usage* Event Query calculated the following scenario:

- System Compromised

The phase is defined as *Damage* because the *Attach Mechanism* in the query is defined as *Damage*. In Figure B.10, the result of the *Unusual Bandwidth* Event Query in Protégé is shown.

APPENDIX

C

SENSORS

In sections 8.5.2 and 8.5.3, *Honeypot and IDS* and *Crawler Detector* sensors are presented. In this appendix, the remaining sensors that are used by the Aeneas are presented. The goal of these sensors is to validate the Aeneas system, and not to present any improvements to the sensors. In sections C.1 to C.10, the remaining sensors are presented.

C.1 Tripwire Access Sensor

Tripwire is a software tool that is used to monitor a designated set of files and directories for any change (Kim and Spafford, 1994). This tool was first made available in November 1992 and is used as a sensor because of its long track record of being secure and automatable. Kim and Spafford developed it to mitigate break-in activity on the Internet and to be used as a tool to find a backdoor left by hackers.

The Tripwire monitor sensor uses the Tripwire programme[1] to detect when hidden files have been modified in any way. Tripwire logs any change event, including:

- creation of new files;
- deletion of files;
- change of file content; and

[1] http://sourceforge.net/p/tripwire/discussion/

Listing 19 Tripwire Sensor Algorithm

```
Tripwire Sensor Start
if Hidden data Accessed then
   Read Tripwire Log File
   Parse Tripwire Log
   Send Parse Log to Aeneas Server
end if
Wait 1 Second
Return to Tripwire Sensor Start
```

- change of file permissions.

The output is parsed by a script which extracts the relevant data and outputs it to a text file. An example of the log follows:

```
2013:02:22:10:48:45,Tripped,10.0.3.6
```

Stored in the raw data file is the time of the logged event, the event type, as well as the destination IP address. The time is stored in the format of Year:Month:Day:Hour:Minute:Second and the IP address is in IPv4 format. The Tripwire sensor is a host-type sensor that logs that hidden data has been accessed. The algorithm used to parse to the Tripwire monitor's output is shown in Listing 19.

C.2 Is Alive Sensor

The goal of the Is Alive sensor is to determine if servers in a network are still communicating. This sensor verifies the connectivity of networked devices from a static list using ICMP (also known as a ping request). A custom script was developed to determine which of the servers are not responding and to log which severs stopped communicating. The output of these scripts are logged to a single log file called alive.log. An example of the alive.log follows:

```
2013:02:17:10:31:12 10.0.2.8
2013:02:17:10:31:12 10.0.2.100
```

Stored in the log file is the time and the server that has been reported as being down. The time is stored in the format Year:Month:Day:Hour:Minute:Second. IP addresses are

Listing 20 IsAlive Sensor Algorithm

```
IsAlive Sensor Start
Read List of Servers
for all List of Servers do
   Verify Server is Communicating
end for
Calculate Number of Servers Communicating
Send Number of Servers Communicating to Aeneas
Wait 30 Seconds
Return to IsAlive Sensor Start
```

stored in IPv4 format. The IsAlive sensor is a network-type sensor and indicates when servers no longer respond to remote ping requests. The algorithm used to parse to the IsAlive sensor's output is shown in Listing 20.

C.3 Firewall Bandwidth Monitor Sensor

The Firewall Bandwidth sensor uses data collected by the Lucidview[2] firewall. The main feature of this sensor is its ability to log the bandwidth usage over a period of one minute. These logs of bandwidth usage are then stored in a text file as follows:

```
1899.8955078125         1360323780
652.0068359375          1360323840
942.8291015625          1360323931
996.6462484375          1360323992
```

The following data is stored in the Firewall Bandwidth sensor raw text file: bandwidth usage (in kilobytes) per minute intervals, and the time in Epoch format. The firewall has the ability to store source and destination IPs as well as the source and destination ports. This data may be used if need be in the future.

The Bandwidth sensor is a firewall-type sensor which counts all the raw data retrieved from the Lucidview firewall. The algorithm used to parse to the Firewall Bandwidth sensor's output is shown in Listing 21.

[2]http://www.lucidview.net/products-and-services/lucidview-guardian/

Listing 21 Firewall Sensor Algorithm

```
Firewall Sensor Start
Request Bandwidth use from the Firewall
Calculate Bandwidth
Send Bandwidth to Aeneas
Wait 30 Seconds
Return to Firewall Sensor Sensor Start
```

C.4 Web Defacement Sensor

The Web Defacement sensor detects if any of the static files of a website have been changed. This sensor retrieves the website with the wget[3] utility and them compares the static parts of the website to verify that no part has changed. Any user content pages are ignored. The output of this sensor is logged to a single log file. This text file is shown below:

```
1361777839        10.0.2.8        127.0.0.1        80        Website defaced
1361777978        10.0.2.8        127.0.0.1        80        Website DoS
```

Stored in the log file is the time, source IP address, source port, destination IP address, destination port and a description of the crawl event type. The time is stored in Epoch time format and the IP addresses are in IPv4 format. The Web Defacement sensor is a host-type sensor and indicates when the website has been defaced. If the wget utility can't access the page, it will report that the website cannot be accessed. The algorithm used to parse to the web defacement monitor's output is shown in Listing 22.

C.5 Bro Connections Sensor

The Bro Connections sensor uses the Bro[4] IDS. One of the features of this IDS is to log all connections that are visible from itself. These connections are stored in text file as follows (some fields were committed below):

```
1358857747.681339  10.0.3.12   46956  10.0.3.217      80   tcp http
1358857712.125418  10.0.3.10   40853  10.0.3.217      80   tcp
1358857723.964073  10.0.2.100  138    10.0.255.255    138  udp
1358857736.807653  10.0.3.42   8      10.0.3.10       8    icmp
```

[3] http://www.gnu.org/software/wget/
[4] http://www.bro.org/

Listing 22 Web Defacement Sensor Algorithm

```
Web Defacement Sensor Start
if Static Web Page Not Accessible then
  Set DoS Detected
end if
if Static Page Altered then
  Set Web Defacement Event
end if
Parse Time, Event Type and Destination IP
Parse Web Defacement Raw Output
Send Parse Output to Aeneas Server
Wait 30 Seconds
Return to Web Defacement Sensor Start
```

Listing 23 Bro Connections Sensor Algorithm

```
Bro Connections Sensor Start
Measure Conventions with Bro IDS
Parse Time,Number of Connections, Source IP, Source Port, Destination IP and Destination Port
Send Parse Data to Aeneas Server
Wait 30 Seconds
Return to Web Bro Connections Sensor Start
```

In the Bro Connection raw text file, the time, source IP, source port, destination IP, destination port and event type are logged. The date is represented in Epoch time. The source and destination IPs are represented in IPv4 human readable format. The source and destination ports are represented by an integer value. The event type is represented as the protocol type.

The Bro Connection sensor is a a network IDS-type sensor that indicates the network load and number of connections available. The Bro Connection sensor algorithm is shown in Listing 23.

C.6 Root Login Sensor

This sensor was custom written for Aeneas. The sensor monitors the .bashrc file that executes every time a user logs in, and it has been modified to execute the rootsensor bash script. This script checks if the current executing user is a root user. This sensor is used to detect if hackers gained root (administrator) access to a server. Logs of root user logins are stored in a text file as follows:

Listing 24 Root Login Sensor Algorithm

```
Root Login Sensor Start
while .bashrc Changed do
  Read .bashrc File
  if Root User Detected then
    Parse Time, Source IP, Event Type
    Send Parse Data to Aeneas Server
  end if
end while
Return Root Login Defacement Sensor Start
```

```
2013:02:22:09:14:49,Root Detected,10.0.4.30
```

The date, event type and destination IP address are stored in the Root Logins raw text file. The time is stored in human readable time in the following format: Year:Month:Day:Hour:Minute:Second. The destination IP is stored in human readable IPv4. The Root Login sensor is classified as a host-type sensor that indicates all root user logins on a particular host. In Listing 24, the algorithm used by the Root Login sensor is shown.

C.7 SSH Login Sensor

Similar to the Root Login, this sensor is also custom written for Aeneas. This script executes every second and uses the linux "who"[5] command to determine if any users are connected via SSH. This sensor determines if someone has logged into a server remotely. An example of logs of SSH logins are stored in a text file as follows:

```
2013:02:22:10:02:55, SSH Login,10.0.4.30,10.0.4.31
```

In the SSH raw text file, the date, event type, source IP address and destination IP address are stored. The date is stored in a similar way to the Root Login sensor and is in the following format: Year:Month:Day:Hour:Minute:Second. The source and destination IP addresses are in IPv4 format. The SSH Login sensor is host-type sensor and indicates all successful SSH logins on the host. The algorithm used by the SSH Login sensor is shown in Listing 25.

[5] http://linux.about.com/library/cmd/blcmdl1_who.htm/

Listing 25 SSH Login Sensor Algorithm

```
SSH Login Sensor Start
Execute "who" command
if SSH User Detected then
  Parse Time, Source IP, Destination IP, Event Type
  Send Parse Data to Aeneas Server
end if
Wait 1 Second
Return to SSH Login Defacement Sensor Start
```

Listing 26 Failed Login Sensor Algorithm

```
Failed Login Sensor Start
Read /var/log/auth.log
if Failed User Detected then
  Parse Time, Source IP, Destination IP, Event Type
  Send Parse Data to Aeneas Server
end if
Wait 1 Second
Return to Failed Login Defacement Sensor Start
```

C.8 Failed Login Sensor

The Failed Login sensor is a script that checks the /var/log/auth.log file for "authentication failure" entries. This sensor determines if failed attempts have been made to log into a server. The raw data for the Failed Login sensor is shown below:

```
2013:02:22:10:02:57, Failed Login,10.0.4.30,10.0.4.31
```

The raw data is stored in a similar way to the SSH Login sensor entries. The date has the following format: Year:Month:Day:Hour:Minute:Second. The source and destination IP addresses are in IPv4 format. The Failed Login sensor is classified as a host-type sensor and indicates all unsuccessful logins on the host. The algorithm used is shown in Listing 26.

C.9 Unusual Disk Usage Sensor

This sensor uses the "du"[6] application to check if the disk usage of a certain directory has changed. Similar to Tripwire, the output is parsed by a script and then the relevant

[6] http://linux.about.com/library/cmd/blcmdl1_du.htm/

Listing 27 Unusual Disk Usage Sensor Algorithm

```
Unusual Disk Usage Sensor Start
Execute "du" command
if Directory Usage has changed then
  Parse Time, Source IP, Event Type
  Send Parse Data to Aeneas Server
end if
Wait 1 Second
Return to Unusual Disk Usage Defacement Sensor Start
```

data is placed into a text file, as shown as follows:

```
2013:02:22:10:02:52, Storage Usage,10.0.3.6
```

This raw data file contains the time (formatted as Year:Month:Day:Hour:Minute:Second), the event type and the IPv4 destination IP address. This sensor is classified as a host-type sensor and indicates if resources in terms of disk usage have been used without authorisation. The algorithm used is shown in Listing 27.

C.10 Bandwidth and SYN Sensor

The Bandwidth and SYN sensor uses custom-written software to determine the bandwidth usage detected and the number of SYN packets visible on an interface. The Bandwidth sensor is similar to the Firewall Bandwidth Monitor sensor (Appendix C.3), except for not requiring a firewall. The bandwidth is reported in kilobytes. The SYN sensor is similar to the Bro Connection sensor (Appendix C.5), except only the number of TCP connections are detected. ICMP, UDP and other connections that do not use SYN packets are not counted.

The output of this sensor is logged to two log files. One file stores bandwidth, the other the number of SYN connections. Examples of the logged files are shown as follows:

```
1374250912,321,Bandwidth DMZ
1374250912,14,SYN DMZ
```

The algorithm for the Bandwidth and SYN sensor as shown in Listing 28.

Listing 28 Bandwidth and SYN Sensor Algorithm

```
Bandwidth and SYN Sensor Start
Clear Bandwidth and number of SYN connections
while 30 Seconds passed do
  Calculate Total Bandwidth usage
  Calculate number of SYN connections
end while
Parse Time, Source IP, Event Type
Send Bandwidth u sage number of SYN connections to Aeneas Server
Return to Bandwidth and SYN Sensor Start
```

C.11 Summary

The sensors used by the Aeneas are presented in this appendix. These sensors were developed to validate the principles and not to be unique or new sensors.

APPENDIX

D

TIME FORMATS

Formats the given timestamp according to the given format. The format string is used as a template to format the date and is copied character by character except for the following special characters, which are replaced by the corresponding value[1].

% w - abbreviated weekday (Mon, Tue, ...)
% W - full weekday (Monday, Tuesday, ...)
% b - abbreviated month (Jan, Feb, ...)
% B - full month (January, February, ...)
% d - zero-padded day of month (01 .. 31)
% e - day of month (1 .. 31)
% f - space-padded day of month (1 .. 31)
% m - zero-padded month (01 .. 12)
% n - month (1 .. 12)
% o - space-padded month (1 .. 12)
% y - year without century (70)
% Y - year with century (1970)
% H - hour (00 .. 23)
% h - hour (00 .. 12)
% a - am/pm
% A - AM/PM

[1] http://www.gnu.org/software/octave/doc/interpreter/Timing-Utilities.html/

% M - minute (00 .. 59)
% S - second (00 .. 59)
% s - seconds and microseconds (equivalent to %S.%F)
% i - millisecond (000 .. 999)
% c - centisecond (0 .. 9)
% F - fractional seconds/microseconds (000000 - 999999)
% z - time zone differential in ISO 8601 format (Z or +NN.NN)
% Z - time zone differential in RFC format (GMT or +NNNN)
%% - percent sign
% E - Epoc Time