Fighting an Unfair Battle: Unconventional Defenses against Advanced Persistent Threats

Nikolaos Virvilis Kollitiris

October 2015

Information Security and Critical Infrastructure Protection (INFOSEC) Laboratory
Department of Informatics

Fighting an Unfair Battle: Unconventional Defenses against Advanced Persistent Threats

Nikolaos Virvilis Kollitiris

A dissertation submitted for the partial fulfillment of a Ph.D. degree

October 2015

Supervising Committee:

1. **Dimitris Gritzalis**, Professor, Dept. of Informatics, Athens University of Economics & Business, Greece (Chair)
2. **Ioannis Stamatiou**, Associate Professor, Dept. of Business Administration, University of Patras, Greece
3. **Costas Lambrinoudakis**, Associate Professor, Dept. of Digital Systems, University of Piraeus, Greece

Examination Committee:

1. **Dimitris Gritzalis**, Professor, Dept. of Informatics, Athens University of Economics & Business, Greece (Chair)
2. **Theodoros Apostolopoulos**, Professor, Dept. of Informatics, Athens University of Economics & Business, Greece
3. **Vasilis Katos,** Professor, Dept. of Computing, Bournemouth University, United Kingdom
4. **Ioannis Mavridis**, Associate Professor, Dept. of Applied Informatics, University of Macedonia, Greece
5. **Ioannis Stamatiou**, Associate Professor, Dept. of Business Administration, University of Patras, Greece
6. **Costas Lambrinoudakis**, Associate Professor, Dept. of Digital Systems, University of Piraeus, Greece
7. **Giannis Marias**, Assistant Professor, Dept. of Informatics, Athens University of Economics & Business, Greece

Fighting an Unfair Battle: Unconventional Defenses against Advanced Persistent Threats

Nikolaos Virvilis Kollitiris
Information Security and Critical Infrastructure Protection (INFOSEC) Laboratory
Department of Informatics
Athens University of Economics and Business
76 Patission Ave., Athens GR-10434, Greece

Disclaimer

The views and opinions expressed in this thesis are those of the author and do not in any way represent the views, official policy or position of the Athens University of Economics and Business or his employer.

"Η έγκριση διδακτορικής διατριβής υπό του Τμήματος Πληροφορικής του Οικονομικού Πανεπιστημίου Αθηνών δεν υποδηλοί αποδοχή των γνωμών του συγγραφέως" (Ν. 5343/ 1932, άρ. 202).

Acknowledgements

First and foremost, I would like to thank my Ph.D. supervisor Prof. Dimitris Gritzalis. We met more than a decade ago, while I was on my third year of undergraduate studies. I had selected Information Security as an optional module without too much thought, as my plan was to do an MSc in Computer Networking. However, after a handful of lectures I decided that Information Security was the field that I would like to pursue a career in. Professor, thank you for your support and inspiration during my undergraduate degree, my postgraduate studies, but most importantly your support and patience during the more than five years of my academic research.

I would also like to thank Prof. Theodoros Apostolopoulos and Prof. George Polyzos for their support and inspiration during my undergraduate studies and Associate Prof. Konstantinos Markantonakis (Royal Holloway) who guided me in my preliminary steps in research by supervising my MSc thesis.

Secondly, I would like to thank Dr. Vasilis Tsoumas, Dr. Stelios Dritsas, Dr. Marianthi Theocharidou and Dr. Giannis Soupionis. They were the ones that helped me the most on my first research steps, either by co-authoring, reviewing or tutoring. I am grateful for their support and professionalism but mainly for their friendship.

I am grateful that I shared my research journey with Dr. Alexios Mylonas and Miltiadis Kandias. I also want to thank George Stergiopoulos, Nikolaos Tsalis and Vasilis Stavrou for their support, friendship and cooperation.

Furthermore, I would like to thank Prof. Panos Constantopoulos for offering me a position at the IS Lab as a systems and network administrator during the first few years of my academic research.

From the other side of the Atlantic, I would like to thank Dr. Vasilios Kemerlis for his support, guidance and friendship since 2001 and also Yannis Mallios for his advice, ideas and help.

I would like to express my appreciation and gratitude to Dr. Antonios Atlasis, Dr. Ioannis Dimou and Ms. Athanasia Botsou for reviewing the first drafts of this thesis and their valuable comments. I am also thankful to Giannis Kolovos, Fanis Drosos, Fotis & Thomas Ailianos, George Fekkas, Michalis Foukarakis, Dionisis Antonopoulos, Spyros Papageorgiou, and Monogioudis and for their friendship, support, ideas and guidance during these years.

Finally, I would like to thank my parents for their love and support since the beginning of my life. I may not forget the things that they did wrong, but I will never forget the things they did right.

Dedication

To all of you who have helped me shape my life.
Family, friends, colleagues, acquaintances and random people.
Our paths have crossed for a reason.
Thank you for been there.

Abstract

The number and complexity of cyber-attacks has been increasing steadily in recent years. The major players in today's cyber conflicts are well organized and heavily funded teams with specific goals and objectives, working for or supported by a nation-state. A commonly used term to describe such teams/groups is Advanced Persistent Threat (APT). APT target the communication and information systems of government, military and industrial organizations and are willing to use vast amounts of money, time and expertise to reach their goals.

A clear indication of the level of sophistication of APT is their impressive arsenal. The complexity and capabilities of recently discovered malware used to facilitate such attacks are remarkable: Stuxnet, Duqu, Flame, Red October, MiniDuke and more recently Regin are examples of highly sophisticated malware, the development of which required skillful individuals – in some cases (e.g. Stuxnet) with expertise in multiple technology fields – as well as substantial financial resources.

In addition, serious insider attacks have occurred that resulted in the publication of several thousand classified documents, highlighting the fact that even in sensitive institutions, the effectiveness of the existing security safeguards is insufficient.

Advances in attacker sophistication have not been matched by similar defensive advances. The concept of keeping the internal, trusted network separated from the external, untrusted one (i.e. boundary protection) has become obsolete. The use of blacklists or signatures for attack detection is practically useless against sophisticated attackers. The security industry, having spent decades developing security products such as anti-malware solutions and intrusion-detection/prevention systems, refuses to admit the shortcomings of these products.

It is not uncommon for security companies to advertise that their products can detect and stop APT, even though the same products have been unable to detect such attacks for several years. Furthermore, C-level executives fail to understand the need for more robust security mechanisms, as they believe that by following vendor recommendations and making significant investments in traditional security solutions, they will keep their organization secure. However reality has proven them wrong, over and over again.

In order to defend against such sophisticated adversaries, it is necessary to redesign our defenses and develop technologies focused more on detection than prevention.

The purpose of this thesis is to offer a comprehensive view of the APT problem by analyzing the most common techniques, tools and attack paths that attackers are using, and highlighting the shortcomings of current security solutions. The use of deception techniques for attack detection is one of the integral focal points of this thesis. Based on this concept, a novel APT detection model is proposed, implemented and evaluated.

The evaluation results highlight the significant efficacy of the model in detecting sophisticated attacks, with a very low false positive rate.

(This page is intentionally left blank)

Table of Contents

Disclaimer ..... **v**

List of Figures ..... **xvi**

List of Tables ..... **xvii**

List of Acronyms ..... **xviii**

Chapter 1: Introduction **1**

1.1 Research motivation 1

1.2 Research statement and approach 3

1.3 Contributions 4

1.4 Dissertation outline 5

Chapter 2: Background **8**

2.1 Advanced Persistent Threat (APT) 8

2.2 The failure of traditional security technologies and architectures 9

2.2.1 Technology Limitations 9

2.2.1.1 Intrusion Detection/Prevention Systems (IDS/IPS) 10

2.2.1.2 End Point Protection (Anti-Malware) 10

2.2.1.3 Full packet capture 11

2.2.1.4 SIEM 11

2.2.2 Proprietary technology and lack of integration 12

2.3 Multiple Dimensions of the APT 12

2.3.1 External Attacks 12

2.3.2 Internal Attacks (Insider threat) 13

2.3.3 Indirect attacks 13

2.4 Increasing growth of cyber-operations 14

2.4.1.1 China 14

2.4.1.2 Iran 15

2.4.1.3 Israel 15
2.4.1.4 Russia 16
2.4.1.5 Syria 16
2.4.1.6 United Kingdom 16
2.4.1.7 United States 17
2.5 (Most of the) Industry in denial 17
2.6 Related Work 17
Chapter 3: External Attacks 22
3.1 Attacker's objectives 22
3.1.1 Information gathering 22
3.1.2 Gaining foothold and further compromise 23
3.1.3 Exfiltration 24
3.2 Technical review of major APT malware 24
3.2.1 Stuxnet 24
3.2.2 Duqu 26
3.2.3 Flame 27
3.2.4 Red October 28
3.2.5 MiniDuke 29
3.2.6 Regin 30
3.3 Common characteristics 32
3.4 Limiting the impact 34
3.5 Indirect attacks 35
3.5.1 Supply chain attacks 35
3.5.2 Attacks against third party providers 36
Chapter 4: Internal Attacks 38
4.1 The insider threat as a subset of APT 38
4.1.1 Major insider attacks 39
4.1.2 Manning's disclosures 39
4.1.3 Snowden's disclosures 40

4.1.4 Public Opinion 41
4.2 An insider threat prediction model 41
4.2.1 Component: User Taxonomy 42
4.2.2 Component: Psychological Profiling 43
4.2.3 Component: Real Time Usage Profiling 44
4.2.4 Component: Decision Manager 45
4.3 Requirements and limitations 48
Chapter 5: Indirect Attacks: Mobile devices 51
5.1 The exponential growth of mobile devices 51
5.2 Background 52
5.2.1 Phishing Attacks 52
5.2.2 Malware attacks 54
5.3 Methodology 55
5.3.1 Test methodology 55
5.3.2 Phishing Tests 56
5.3.3 Malware Tests 58
5.4 Experimental Results 58
5.4.1 Protection against phishing sites 58
5.4.1.1 iOS browsers 58
5.4.1.2 Android browsers 59
5.4.1.3 Desktop browsers 60
5.4.1.4 Comparison with previous evaluation 61
5.4.2 Protection against malicious sites 62
5.4.2.1 iOS Browsers 62
5.4.2.2 Android Browsers 62
5.4.2.3 Desktop browsers 63
5.5 Secure Proxy 63
5.5.1 Architecture 64
5.5.2 Secure proxy evaluation 64

5.5.3 URL-only and hash-based analysis 66
5.5.4 File-based analysis 67
5.5.5 Performance evaluation 68
5.6 Discussion 68
5.6.1 Limitations 68
5.6.2 Protection of desktop and mobile browsers 70
5.6.3 Proposed countermeasure 71
5.7 Conclusion 73
Chapter 6: Redefining the Security Architecture 76
6.1 Introduction 76
6.2 Unconventional defenses: The use of deception 76
6.2.1 Phase 1: Attack planning 77
6.2.1.1 DNS honey tokens 78
6.2.1.2 Web server honey tokens 79
6.2.1.3 Social network avatars 79
6.2.2 Phase 2: Attack execution 80
6.2.2.1 Network layer deception 81
6.2.2.2 Application layer deception 81
6.3 A novel APT detection model 83
6.3.1 Anomaly-based attack indicators 83
6.3.2 Deception-based attack indicators 84
6.3.3 Data collection 84
6.3.3.1 Collection of attack indicators: Log file analysis 84
6.3.3.2 Collection of attack indicators: Network traffic Analysis 86
6.3.4 Data analysis (Correlation engine) 88
6.3.4.1 Correlation window 88
6.3.4.2 Threat Rating, Average Threat Rating 90
6.3.5 Alerting level - thresholds 92
6.4 Model Evaluation 93

6.4.1 External Attack Scenario 94
6.4.2 Internal Attack Scenario 96
6.4.3 Proof of Concept Implementation 98
6.4.4 Evaluation Results 100
6.4.4.1 External Scenario Results 100
6.4.4.2 Internal Scenario Results 103
6.4.4.2.1 Comparison with Snort 107
6.5 Discussion 111
6.5.1 Technical challenges 111
6.5.2 Business challenges 113
Chapter 7: Conclusions 116
7.1 Summary 116
7.2 Publications 118
7.4 Future Work 120
7.5 Concluding remarks 120
Appendix 122
Mobile Statistics 122
Source Code 126
Snippet 1 – External Scenario 126
Snippet 2 – Internal Scenario, honeyd configuration 127
Snippet 3 – Internal Scenario, LogParser.py 128
Snippet 4 – Internal Scenario, NetworkParser.py 133
Snippet 5 – Internal Scenario, Correlation.py 139
References 142

List of Figures

Figure 1 - Spear-phishing attack 23
Figure 2 - Insider threat prediction model 42
Figure 3 - Laboratory setup 56
Figure 4 - Comparison of anti-phishing protection (Q1 2014 – Q2 2014) 61
Figure 5 - Proposed architecture 65
Figure 6 - Detection percentage of hashes 67
Figure 7 - Detection percentage of executables 68
Figure 8 - Phishing URL detection percentage 70
Figure 9 - Malicious URL detection percentage 71
Figure 10 - APT attack life cycle 77
Figure 11 - Common sources of information gathering 80
Figure 12 - Windows audit policy 86
Figure 13 - Windows audit policy for File System access 86
Figure 14 - Sample architecture with proposed network monitoring points 88
Figure 15 - Inputs to correlation engine 89
Figure 16 - ATR calculation process 92
Figure 17 - External attack architecture 95
Figure 18 - Internal network architecture 97
Figure 19 - Data sources for PoC implementation 100
Figure 20 - Snort alerts based on signature id 102
Figure 21 - Triggered attack indicators per team 107

List of Tables

Table 1 - Malware comparison 32
Table 2 - Motive score 46
Table 3 - Opportunity score 47
Table 4 - Capability score 47
Table 5 - Overall threat score 48
Table 6 - Browser availability on tested platforms 55
Table 7 - Phishing protection statistics on iOS 59
Table 8 - Phishing protection statistics on Android 60
Table 9 - Phishing protection statistics on Windows 61
Table 10 - Malware protection statistics on iOS 62
Table 11 - Malware protection statistics on Android 62
Table 12 - Malware protection statistics on Windows 63
Table 13 - Percentage of blacklisted malicious sites 66
Table 14 - Typical NetFlow record (NFDUMP) 87
Table 15 - Indicator threat rating 91
Table 16 - Server and Client Segment: VM type and services 96
Table 17 - Parsed Windows event logs 99
Table 18 - Triggered attack indicators for external scenario 101
Table 19 - Detection results 102
Table 20 - Triggered attack indicators for the internal scenario 104
Table 21 - Triggered Snort rules 108
Table 22 - Comparison of alerts 110
Table 23 - Desktop browser popularity 122
Table 24 - Browser popularity on Android 122
Table 25 - Default CIF feeds 122
Table 26 - Percentage of URLs that were blacklisted 123
Table 27 - Percentage of false negatives 123
Table 28 - Percentage of URLs that were manually verified as non-phishing 123
Table 29 - Malicious file detection based on the hash of the samples 124
Table 30 - Malicious file detection based on file analysis (submission of the file) 124
Table 31 - VirusTotal AV Engines 125
Table 32 - VirusTotal URL reputation providers 125

List of Acronyms

APT	Advanced Persistent Threat
AD	Active Directory
ATR	Aggregated Threat Rating
BRO	A Network Analysis Framework
BYOD	Bring Your Own Device
C&C	Command and Control
DDOS	Distributed denial of Service Attack
FTP	File transfer protocol
HTTP	Hyper Text Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICS	Industrial Control Systems
NIDS	Network Intrusion Detection System
NIPS	Network Intrusion Prevention System
OPSEC	Operations Security
Pentest	Penetration Testing
PoC	Proof of Concept
RAT	Remote Administration Tool
RDP	Remote Desktop Protocol
SNORT	Open Source IDS
SSH	Secure Shell
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TR	Threat Rating

(This page is intentionally left blank)

Chapter 1: Introduction

1.1 Research motivation

The Cyberwar era has begun. In fact, it began decades ago. Contrary to the common belief that Stuxnet (Langner 2011) was the first cyber-attack that caused physical destruction, similar attacks date as far back as 1982. Thomas Reed, a former Air Force secretary, wrote in his book (Reed 2007) that the trans-Siberian pipeline explosion in June 1982 was the result of a logic bomb that the CIA had planted on the Russian SCADA systems that controlled the pipeline. Although there is no concrete evidence to support Reed's statement, in a collection of intelligence documents known as "The Farewell Dossier" (Weiss 1996), it is stated that the an effort to sabotage multiple Soviet projects. The leaks included software to control SCADA CIA intentionally leaked modified (i.e. rogue) software and hardware designs to the Soviets in systems, which thus supports Reed's statement.

The biggest known cyber-attack in the 1990s was given the code name "Moonlight Maze". The attack targeted numerous military networks, NASA and universities throughout the United States. Russian involvement in this attack was suspected but was never proven (Healey & Grindal 2013).

In 2004, a series of sophisticated attacks that the FBI later called "Titan Rain" were detected. The attacks had started in 2003, targeting multiple sensitive US networks, including numerous US defense contractors. China was the main suspect behind these cyber-attacks, but the Chinese government denied any involvement (Bodmer et al. 2012).

In 2006, the term "advanced persistent threat" (APT) was introduced for the first time, by the US Air Force, to describe highly sophisticated attacks.

In April 2007 a series of cyber-attacks against multiple Estonian organizations, including financial institutions, ministries, the Estonian parliament and media, took place. It is believed that the cyber-attacks came as a response to the Estonian decision to relocate the "Bronze Soldier of Tallinn", a Russian World War II memorial. This move created significant tension between Estonia and Russia, which resulted in a number of severe riots but most importantly, to a wave of massive Distributed Denial of Service (DDoS) attacks against Estonia, over a period of three weeks. Russia was openly accused of orchestrating this attack, however Russian officials refused to admit any involvement (Clarke & Knake 2012).

In 2008, a USB stick that had been infected by a foreign intelligence agency was left in the parking lot of a US Department of Defense facility at a base in the Middle East. The worm spread on both unclassified and classified networks, "*establishing what amounted to a digital*

beachhead, from which data could be transferred to servers under foreign control", as William Lynn states. The attack was *"the most significant breach of U.S. military computers ever"* (Lynn 2010).

In the same year, a massive cyber-espionage operation was conducted during the presidential campaigns of Barack Obama and John McCain. According to officials, the goal of the attack was to gain as much information as possible for both campaigns, including emails and position papers. According to Dennis Blair, who served as President Obama's Director of National Intelligence, the attackers *"were looking for positions on China, surprises that might be rolled out by campaigns against China."* (Isikoff 2013).

Furthermore, on 5 August 2008, an explosion at "valve station 30" of the BP pipeline in Refahiye, Turkey, caused a massive fire and billions of dollars in lost revenue. The Kurdistan Workers' Party (PKK) claimed credit for the attack, however reports from intelligence services challenge that claim, based on its level of sophistication. The pipeline was one of the most secure in existence, monitored in real time with advanced sensors and cameras and it even had a satellite connection as a backup mechanism, in case the primary connectivity was disrupted. According to some reports (Robertson & Riley 2014), there is footage showing two men with laptop computers walking near the pipeline a few days before the explosion. The attackers managed to gain unauthorized access to the control infrastructure and changed the pressure in the pipe to critical levels, while blocking the sensors from raising alerts.

Google, on 12 January 2010, publicly disclosed on a blog post that it had suffered *"a highly sophisticated and targeted attack on our corporate infrastructure originating from China that resulted in the theft of intellectual property from Google."* (Drummond 2010). The attacks had started in mid-2009 and affected dozens of large organizations including Yahoo, Adobe and Symantec. The attack was given the name "Operation Aurora" by McAfee (McAfee Labs 2010), based on an analysis of the malicious binary files related to the attack. The attackers used spear-phishing emails that included links to a malicious website. The attack exploited a zero-day vulnerability on Internet Explorer, which resulted in remote code execution and subsequently the installation of a remote administration tool (RAT). Apart from the theft of intellectual property, the attackers were interested in a database containing the Google accounts for which a court-ordered wiretap had been activated. It is therefore believed that the attackers were trying to learn if there was an active investigation against undercover Chinese operatives in the US (Schwartz 2013).

In June 2010, Stuxnet was discovered. It was a computer worm designed to target specialized industrial control systems (ICS), and more specifically, the Siemens systems that were used to control arrays of centrifuges in the Natanz uranium enrichment plant in Iran. Stuxnet caused the destruction of hundreds of centrifuges and as a result slowed down the

Iranian nuclear program by four years (Lemos 2011). The complexity of the attack is a strong indication that it was planned by (at least one) resourceful nation (Chen & Abu-Nimeh 2011).

In 2011, McAfee published a report on "Operation Shady RAT", a targeted cyber-attack that lasted 5 years and affected 71 organizations. Multiple US federal and state government organizations were targeted, as well as industry, media and international sports organizations. According to McAfee (Alperovitch 2011), the attackers sent phishing emails to their targets, attaching malicious Microsoft Office and Adobe PDF documents, which when executed, infected the systems with a remote administration tool.

In subsequent years, several cases of advanced malware used in targeted attacks against sensitive organizations were detected: Flame, Duqu (Bencsáth et al. 2012b), Red October (Kaspersky 2013b), (Virvilis & Gritzalis 2013) and most recently Regin (Symantec 2014b).

The level of sophistication of these attacks, but most importantly the resources required for orchestrating them, leaves no room for doubt: cyberspace has become the fifth domain of warfare (Economist 2010), where multiple nations are investing vast amounts of resources in developing offensive and defensive capabilities.

Organizations face the always present threat of insider threat, a clear example of which is Edward Snowden, who exfiltrated 50,000 to 200,000 classified documents belonging to the US National Security Agency (Hosenball 2013). This incident took place shortly before Chelsea (Bradley) Manning was convicted and sentenced to 35 years in prison in connection with the largest data leak in US history (Denver 2012).

1.2 Research statement and approach

Although the cyber-security industry is booming, with global spending of more than 71 billion USD in 2014 and an increase of almost 8% over 2013 (Gartner 2014a), the majority of Information Security professionals are deeply concerned about the effectiveness of existing defenses against sophisticated attackers (ISACA 2014). In 2013, there was an astonishing 62% increase in the number of security breaches compared to the year before (ISACA 2014). Although spending for security solutions has increased, most of the organizations have not increased training budgets for their cyber-security professionals (ISACA 2014). Furthermore, there is a global shortage of skilled security professionals (Libicki et al. 2014), which directly affects a nation's (or an organization's) ability to defend.

It is becoming very clear that traditional security solutions and architectures are ineffective against sophisticated attackers. High-speed networks, mobile users, BYOD (bring your own device), complex web platforms and the proliferation of social networks and cloud services have created new challenges. The new challenges are very different from the challenges of the 1980s and early 1990s, when the majority of the security solutions that are in

use today were designed. Although security vendors are constantly trying to improve their products in order to keep up with new threats, such solutions keep failing over and over again. The root cause lies in their design: traditional security solutions are focused on attack prevention, which, especially against sophisticated attackers, will eventually fail (Cole 2012), (Bejtlich 2013).

Sophisticated attacks can manifest themselves in different ways (i.e. via different attack paths):

- They can originate over the Internet (external), where attackers manage to compromise their target's infrastructure remotely.
- They can originate internally (insider attacks). The insiders can be working on their own or as part of a larger team/group (e.g. the APT group can use extortion to force an employee of the organization to perform an action on their behalf).
- They can be indirect attacks e.g. attacks against a service provider in order to facilitate further attacks against the real target, such as the attack against RSA (RSA 2011)**,** or attacks against mobile devices that will eventually be connected to the targeted network (e.g. smart phones, tablets, etc.).

Because multiple APT attack paths are possible, it is not realistic to focus on a single defensive technology (e.g. research on intrusion detection), as even if a perfect detection system existed – something that has been proven to be impossible (Cohen 1987) – the attackers would just follow a different path to achieve their goal.

This thesis focuses separately on each of the attack paths, presenting realistic countermeasures. In addition, proposes and evaluates a novel APT detection model which combines deception and anomaly-based attack indicators. The model improves significantly the possibility of early detection against sophisticated attacks, regardless of the attack path chosen.

1.3 Contributions

In summary, this thesis makes the following contributions:

- *A detailed review of APT attack paths:* The main APT attack paths (external, internal and indirect) are reviewed, and the unique characteristics of each are presented in detail, along with countermeasures for limiting our exposure.

- *A technical review of popular APT malware:* A technical review of Stuxnet, Duqu, Flame, Red October, Mini Duke and Regin is presented, highlighting the

common characteristics and techniques of such advanced malware. Recognizing the characteristics can lead to better understand of why our existing security safeguards have failed to detect such threats.

- *A description of the serious security concerns affecting mobile platform users:* As a result of the proliferation of smartphones/tablets, their continuously expanding capabilities and their increasing use in business environments, such devices are expected to be a major target for APT in the future. Our experiments reveal that the level of security offered on mobile platforms against web based attacks is significantly lower than that for desktop platforms.

- *A sophisticated APT detection model:* A sophisticated model for APT detection which correlates deception and anomaly-based attack indicators, is proposed. The model has been implemented and evaluated; the results highlight its superior effectiveness in attack detection and very low false positive rate, compared to traditional security solutions.

1.4 Dissertation outline

The rest of the dissertation is organized as follows: Chapter 2 defines APT and highlights the main reasons that the security industry has failed to address such attacks. The most common cyber-security technologies are presented, focusing on their main shortcomings and ways that they can be evaded. The major players in the cyber "arena" are presented with a summary of their defensive and offensive capabilities, as derived from publicly available information.

Chapter 3 discusses external APT attacks, analyzing the most common steps that attackers follow when targeting an infrastructure remotely. A technical review of advanced APT malware is presented, with emphasis on the common techniques and methods used by the malware authors, followed by a review of security countermeasures, which, if they had been implemented correctly, would have limited the impact of these attacks.

Chapter 4 presents the insider threat, with special attention paid to Snowden's and Manning's attacks. This chapter proposes an insider threat detection model which makes use of psychometric tests to assess a user's predisposition to malicious acts. Parts of it have been incorporated into the proposed APT detection model.

Chapter 5 focuses on indirect attacks and more specifically on the risks introduced by the use of smartphones in sensitive environments. The continuously increasing use of smartphones for both personal and business use, the sensitive data that is accessed, processed and stored on these devices, and the limited or in some cases non-existent security measures, have made them

a prime target for attackers. In this chapter, the security countermeasures available on the Android and iOS devices are reviewed, with focus on the protection offered against phishing and malware attacks.

Chapter 6 proposes a novel APT detection model. A detailed analysis, a proof of concept implementation, and the evaluation of the model against two different attack scenarios used to simulate external and internal APT attacks, are presented. The results demonstrate the substantial effectiveness of the proposed model in the detection of malicious actions.

Finally, conclusions, publications and recommendations for future work are presented in Chapter 7.

(This page is intentionally left blank)

Chapter 2: Background

This chapter defines Advanced Persistent Threat (APT) and highlights the major shortcomings of traditional cyber-security technologies in addressing such attacks. It presents the main APT attack paths and gives historical examples of such incidents. Additionally, it presents an overview of what is being done currently by industry and academia regarding the detection of APT.

2.1 Advanced Persistent Threat (APT)

The term Advanced Persistent Threat (APT), coined by the US Air Force in 2006 (Bejtlich 2010), is not strictly defined and loosely covers threats with a number of common characteristics. The definition of APT given by the National Institute of Standards and Technology (Ross 2012) is as follows:

> *"An adversary with* ***sophisticated levels of expertise and significant resources****, allowing it through the use of* ***multiple different attack vectors*** *(e.g. cyber, physical, and deception) to generate opportunities to achieve its objectives, which are typically to establish and extend its presence within the information technology infrastructure of organizations for purposes of continually exfiltrating information and/or to undermine or impede critical aspects of a mission, program, or organization, or place itself in a position to do so in the future; moreover, the advanced persistent threat* ***pursues its objectives repeatedly over an extended period of time, adapting to a defender's efforts to resist it****, and with determination to maintain the level of interaction needed to execute its objectives."*

The term APT is frequently misused, often as an easy excuse for a security incident suffered by an organization. By placing the blame on APT the organization can claim that although they had been following best practices (which is not usually the case), the incident was inevitable, owing to the advanced offensive capabilities of the attacker. Additionally, the difference between an APT attack and an opportunistic (i.e. less sophisticated) attack is not always clear. The use of sophisticated techniques, novel attack methods and custom malware are indications of APT. However, attacks have occurred that used far less sophisticated tools and techniques yet are still considered APT, such as the attack against RSA (RSA 2011), in which the attackers used a freely available remote administration tool (RAT). But their end goal (stealing data that would allow them to gain access to organizations that use RSA products for secure authentication) was a clear indication of a well-organized, sophisticated attack. Thus, it is important to recognize that attackers will not spend resources in developing advanced tools

if they can achieve their goals by using publicly available ones. This, apart from the obvious benefits (cost/time savings), also makes attribution more difficult, as the attackers are using tools to which anyone can have access.

2.2 The failure of traditional security technologies and architectures

2.2.1 Technology Limitations

One of the most concerning facts regarding APT, is that certain attacks have been ongoing for several months (Bencsáth et al. 2012b), (Virvilis et al. 2013), targeting sensitive organizations with substantial cyber-security budgets. Nevertheless, these organizations have failed to detect the compromise and the majority of them were notified about the incident from a third party (usually law enforcement agencies).

APT actors possess unique characteristics that differentiate them from opportunistic attackers:

- APT take advantage of zero-day (unknown) vulnerabilities and develop their own tools and techniques. In some cases the tools are only used once, against a specific target. In contrast, opportunistic attackers make use of existing attack tools and vulnerabilities.
- They focus on a specific target and are willing to spend a substantial amount of resources and explore all possible attack paths until they are able to subvert its defenses. In contrast, opportunistic attackers will move on to an easier target after a small number of unsuccessful exploitation attempts against the initial target.
- Based on the analysis of the major APT incidents, it is evident that some perpetrators are state-sponsored and have significant enabling capabilities (intelligence collection, manufacturing, covert physical access) for cyber-attacks. On the other hand, opportunistic attackers have limited resources and thus, their attacks tend to be less sophisticated.
- APT are highly selective. Only a small and very carefully selected number of victims are targeted, usually in non-technical departments of an organization, as they are less likely to identify and report the attack. In contrast, opportunistic attackers usually spread their attacks against a broader audience, hoping for ‘quick wins’.

These characteristics make it challenging for current cyber-security solutions to detect and mitigate APT attacks. Furthermore, current solutions suffer from a number of shortcomings which limit their effectiveness, and are discussed below.

2.2.1.1 Intrusion Detection/Prevention Systems (IDS/IPS)

There are two main detection strategies used by network and host-based intrusion detection (or prevention) systems (NIDS/HIDS):

1. **Signature-based,** which is still the most common strategy and focuses on the identification of known-bad patterns (Wu et al. 2008).
2. **Anomaly-based**, which uses heuristic or statistical analysis to determine whether an observed activity could be an indication of a malicious action (Maxion & Tan 2000).

Signature detection, similarly to all blacklist approaches, cannot detect attacks for which a signature has not yet been created (such as zero-day exploits). As this shortcoming has been known for decades (Cohen 1987), the research community has focused on the use of anomaly-based detection systems.

However, the effectiveness of anomaly-based detection systems has also been challenged: Sommer and Paxson describe anomaly detection as flawed in its basic assumptions (Sommer & Paxson 2010). Research relies on the assumption that anomaly detection is suitable for finding new types of attacks, however it is known that machine-learning techniques are best suited to finding events similar to ones seen previously. Therefore, these approaches show promising detection efficacy for specific (training) data sets, but are subject to serious operational limitations when used in operational environments (Sommer & Paxson 2010).

Regardless of the detection strategy, a major challenge for current NIDS/NIPSs is the limited time window for which the connection state can be maintained (for TCP connections). Port scanning is a practical example of this weakness: a quick port scan against a host will trigger an alert from virtually any NIDS/NIPS. However, if the scan is spread over a period of several minutes and thus outside the detection/correlation window of the network intrusion prevention system, the attack will pass undetected by the majority of those systems.

2.2.1.2 End Point Protection (Anti-Malware)

End point protection products face the same limitations as NIDS/HIDS, as their detection method is mainly signature-based, with only a few products using behavioral/heuristic analysis, to complement their detection rates. To make things worse, it is trivial for attackers to test a wide range of antivirus products and modify their malware accordingly, to evade detection, either by download trial version of these products or using free online services (VirusTotal 2014).

The research community has highlighted for many years the limited effectiveness of such products: Fred Cohen proved almost thirty years ago, that *"Precise detection (of a computer*

virus) is undecidable" (Cohen 1987). However, industry has been unwilling to admit this fact and continues to present their products as panaceas. One of the few occasions on which industry publicly admitted the shortcomings of such solutions occurred in 2014, when Symantec's senior vice president for information security stated publicly that "*antivirus is dead*", as its effectiveness in detecting modern threats is less than 50% (Yadron 2014). Although this statement was not a surprise for security experts, it served as an alert for higher management.

2.2.1.3 Full packet capture

Full packet capture (FPC) devices are specialized devices for capturing and archiving network traffic. They are mainly used by network analysts, to inspect captured traffic after an incident. Although they offer the most complete view of the network traffic at any given time and support in-depth analysis, FPCs have important shortcomings:

- They are very expensive.
- Limited analysis options are typically provided by the capture system itself, requiring the use of external tools for low-level traffic inspection.
- They offer very limited (if any) integration with other systems (e.g. NIDS/NIPS, SIEM).
- Even with massive storage capacity, archiving traffic for more than a few days on high speed networks is unrealistic.

2.2.1.4 SIEM

Security incident and event management (SIEM) systems collect events from a wide range of sources (e.g., IDS/IPS, antivirus, event logs) and apply statistical correlation to identify potential attacks. However, their efficacy in detecting sophisticated attacks is limited (Kotenko & Skormin 2010), (Geftic 2013). The main challenges that such systems face are:

- There is a limited time window during which these systems will correlate events - usually a few minutes. Events spread over a larger time period will not be correlated, and as a result, a carefully orchestrated attack may be undetected or presented as a series of seemingly unrelated events.
- The correlation is performed centrally and is therefore limited by the available resources.

2.2.2 Proprietary technology and lack of integration

In addition to the weaknesses and shortcomings of the aforementioned security solutions, perhaps an even more significant factor contributing to the difficulty of detecting sophisticated attacks is the lack of efficient integration among security solutions. The solutions work as "black boxes" and tend to offer (limited) integration only if they come from the same vendor. If the solutions do not themselves offer integration, the only possible option for integration among them is generally through a SIEM system, which suffers from the aforementioned shortcomings. Furthermore, the systems tend to be static, rely on their own (usually proprietary) rules and configuration language, and have their own individual knowledge banks of attack information, with which users must become familiar. Also, owing to the proprietary nature of these devices and the lack of open standards, an analyst who wishes to write custom rules for detecting specific incidents must do so using a different language for each system (e.g. Snort-compatible signature for the NIPS, new correlation rule based on SIEM-specific correlation language) (Kotenko & Skormin 2010).

2.3 Multiple Dimensions of the APT

As previously mentioned, APT have clearly defined goals. In contrast with opportunistic attackers, they focus on a specific target and will use all the tools in their arsenal to achieve their goal. APT attacks can manifest via three different attack paths: external, internal and indirect. Additionally, it is not uncommon for APT actors to use multiple attack paths at the same time.

2.3.1 External Attacks

The majority of known APT attacks fall into this category, for example the attacks against Google and RSA (Drummond 2010), (RSA 2011). In these attacks, the perpetrators try to compromise their target's infrastructure remotely (i.e. over the Internet). This is usually achieved using social engineering techniques, such as sending a cleverly crafted email to a limited number of users working at the targeted infrastructure (spear-phishing attack). It is not uncommon for the attackers to perform extensive information gathering beforehand to increase their chance of success. This can be achieved by exploiting social networks in order to find potential targets, their interests, expertise and position in the organization, or through intelligence agencies (for state-sponsored attacks). Regardless of the way that the APT plan their attacks, the more information they have in advance, the more likely it is that their initial exploitation attempt will succeed.

Common targets for such attacks are developers, because they usually have elevated privileges on their systems and their systems are not hardened (e.g. a workstation with a software restriction policy in place would be the least ideal developing environment). Less technical departments, such as Human Resources (HR), are also promising targets for attackers, as they are used to receiving documents via email (e.g. MS Word files, PDFs), which can be exploited to trigger malicious code execution. Additionally, the limited technical expertise of these users will make it less likely that they will detect an attack against them. External attacks are discussed in detail in Chapter 3.

2.3.2 Internal Attacks (Insider threat)

Insider attacks are attacks originating from within the organization. These are either attacks planned and conducted by a malicious employee (insider), or attacks in which the insider acts as an accomplice for a larger attack group. Edward Snowden's and Chelsea (Bradley) Manning's actions belong in the first category, while the double agent who infected the Natanz nuclear facility with Stuxnet (Kelly 2012) belongs to the second category.

Such attacks are very effective and tend to have severe consequences as the insiders have either authorized access to the internal resources or are in a position to gain access with less effort than an external attacker. Furthermore, the insiders may be better informed about the security safeguards in place, and thus be able to evade them more effectively. Insider attacks are presented in depth in Chapter 4.

2.3.3 Indirect attacks

The term "Indirect attacks" refers to attacks which have as an end goal the exploitation of a specific target but instead of attacking directly the target's infrastructure, the perpetrators target third party providers and services which are used by their target. A successful exploitation of those providers/services can allow them easier access to their target or target's data (e.g. by exploiting the trust relationship between the third party provider and the target).

One example is the attack against the Dutch Certification Authority (CA) Diginotar, in which the attackers managed to create more than five hundred fake certificates. The generated certificates (until they got blacklisted) allowed the attackers to impersonate HTTPS web sites, including multiple Google services, Yahoo and the TOR project. As Diginotar's CA certificate was pre-installed on all major web browsers, the victims did not receive a warning when accessing those services, with the exception of Google Chrome browser due to certificate pinning (OWASP 2014).

In a similar attack against RSA, the attackers obtained information that would allow them to gain access to organizations that used RSA's products for secure authentication. It is believed that a number of US defense contractors were the real targets of that attack (RSA 2011).

Finally, the attack against Google (Drummond 2010), apart from the theft of intellectual property, had a secondary objective: to detect whether or not any of the Gmail accounts of undercover Chinese operatives in the US were being monitored (Schwartz 2013).

The more individuals and organizations use third-party services/providers either for generic tasks i.e. web-based email, Cloud storage (Virvilis et al. 2011a), (Virvilis et al. 2011b), (Agudo et al. 2011) or security related tasks (Marianthi Theoharidou et al. 2013), (Pitropakis et al. 2013), the higher will be the gain for the attackers if they manage to exploit those services. As a result, we should expect that APT will continue to target them.

Alternatively, attackers can focus on systems which have access to the internal network of their target but are easier to compromise than other internal systems. Mobile devices and especially smartphones/tablets with mobile Operating Systems (e.g. iOS, Android) tend to be much easier targets than workstations/laptops with desktop Operating Systems (Mylonas et al. 2012). Regardless of the limited effectiveness (or even complete absence) of robust security mechanisms on mobile devices, the significant benefits offered by such devices make them very appealing to users, who care more about usability than security (Mylonas, Gritzalis, et al. 2013). As a result, the number of tasks performed using such devices is increasing, and this includes accessing and storing potentially sensitive information.

Apart from the risk of information disclosure, smartphones/tables, when used for business purposes, allow access to corporate resources (e.g. access to business email or VPN services). Thus, a compromised device can be the ideal attack path for gaining unauthorized access to a corporate infrastructure, and because such devices generally are subject to limited auditing and lack of security solutions, detection can be very challenging. Hence, it should be no surprise if such devices become the next major APT target, a risk that is discussed in detail in Chapter 5.

2.4 Increasing growth of cyber-operations

Cyber operations (both offensive and defensive) are a high-priority topic for most of the developed nations and are discussed in detail in sections 2.4.1.1 through 2.4.1.7.

2.4.1.1 China

China is one of the most advanced nations in the cyberspace arena. It has been accused publicly of being responsible for a wide number of cyber-attacks, focused mainly on

information gathering and exfiltration of intellectual property. In recent years these accusations became more direct, with Mandiant publishing a detailed report exposing one of China's cyber-attack units, named (by Mandiant) "APT1". The report focuses on the targets of the APT1 group, their tools and methodology, their physical location and even the actual identities of the members of the team (Mandiant 2013).

Similarly, in the Pentagon's 2003 annual report to Congress, the Chinese government and military are accused openly of targeting *"numerous computer systems around the world, including those owned by the US government"* (DoD 2013). However, the Chinese government has denied the accusations.

China's significant cyber capabilities should not come as a surprise, as China's long-term plan is to achieve electronic dominance by 2050 (Andreasson 2011). Furthermore, one of the future goals of China's military is to efficiently combine cyber-attacks with traditional kinetic strikes and thus gain significant strategic advantage during a conflict (Kernel 2009).

2.4.1.2 Iran

Iran is another player in the cyber defense arena. It has suffered major cyber-attacks, but has also been identified as the source of attacks against other nations. In 2010 Stuxnet, which was then declared the "world's most complex malware" (a title that it has since lost), caused severe damage to the Iranian nuclear program (Langner 2011).

On the offensive side, an Iranian group called "Cutting Sword of Justice" has attacked the Saudi Arabian national oil company Aramco, causing significant damage (Goman & Barnes 2012). A large number of Distributed Denial of Service attacks against multiple US institutions have also been attributed to Iran. Other attacks include the compromise of a Dutch Certification Authority (CA) (Galperin et al. 2011), which was used to issue fraudulent certificates for major organizations and thus allow the attackers to perform hard-to-detect man-in-the-middle attacks, and attacks against Twitter (Geers et al. 2014).

2.4.1.3 Israel

Israel has been presumed to be the nation (or at least one for the nations) behind Stuxnet attack (Broad et al. 2011), (Chen & Abu-Nimeh 2011) however, there has been no concrete proof to support this claim. The recent Discovery of Duqu 2.0 (Kaspersky 2015), (Bencsáth et al. 2015), a new version of the infamous Duqu malware which has been created by the same team which developed Stuxnet, put Israel once more on the spotlight. Duqu 2.0 has been active for the last few months, targeting individuals involved in the negotiations regarding Iran's Nuclear program (Bertrand & Kelley 2015), (Gibbs 2015).

Israel has also been the target of cyber-attacks from multiple groups, including Hezbollah and Anonymous (Moskowitz 2015), (TrendMicro 2015), (Moore 2015).

2.4.1.4 Russia

Russia's cyber defense plans include the creation of a unit that will specialize in cyber-attacks and will be responsible for the defense of Russian armed forces. The unit is planned to go into operation by 2017 (Fogarty 2014).

On the offensive side, Russia has been accused multiple times of being the source of cyber-attacks, but in most cases there was a lack of hard evidence (Healey & Grindal 2013). However, FireEye in a recent report accuses Russia directly and exposes a Russian group that they believe is responsible for a large number of cyber-attacks since 2007. The *modus operandi* of the group and the information they seek are clear indications that they have connections (if not working directly) with the government (FireEye 2014).

2.4.1.5 Syria

The Syrian Electronic Army, a group loyal to Syrian President Bashar al-Assad, has conducted a large number of cyber-attacks in recent years. It usually targets media that are considered hostile to the Syrian Government. One of the most damaging attacks of the group, based on a hoax, was achieved by compromising the Twitter account of "The Associated Press", and then tweeting that there had been explosions in the White House that resulted in injury to Barack Obama. The panic that followed caused a 3-minute dip in the stock market of several billion USD (Geers et al. 2014).

Other targets of the Syrian Electronic Army include online services such as Truecaller and Viber, information from which can be invaluable for Syrian intelligence agencies.

2.4.1.6 United Kingdom

The UK has a strong presence in the cyber arena. The UK government is continually investing in cyber security in an effort to augment the country's cyber defense capabilities (Gov.uk 2014a). A clear cyber strategy has been defined, with the main goals of making the UK one of the world's most secure places to do business, increasing resilience against cyber-attacks and further expanding cyber security knowledge and skills (Gov.uk 2014b). The UK Government Communications Headquarters has published several quizzes in recent years (GCHQ 2014) in an effort to find and hire new skilled individuals.

On the offensive side, based on Snowden's disclosures, GCHQ has develop sophisticated attack and monitoring tools, which are being used by the agency for collecting intelligence information (Leyden 2014).

2.4.1.7 United States

The US is one of the top players in the cyber arena. US military command has recognized cyberspace as the fifth domain of warfare (Economist 2010) and in 2009 created the US Cyber Command, an Armed Forces command, focused on cyberspace operations. US short-term plans include increasing the Cyber Command staff numbers to 6000 by 2016, which will make it one of the largest cyber forces in the world (Nakashima 2014).

The US was heavily criticized following Snowden's disclosures regarding the NSA's aggressive cyber operations in gathering intelligence (Gellman & Markon 2013), (Greenwald et al. 2013). The seriousness of the disclosures led to the proposal to Congress of an amendment intended to limit the NSA's ability to collect information about US citizens without a warrant (Timm 2014).

2.5 (Most of the) Industry in denial

As already mentioned, current security solutions are not able to address sophisticated attacks. The vast majority of vendors refuse to admit this publicly. Some of them even state that their solutions detect and stop APT, ignoring the fact that their solutions have failed to detect such attacks over and over again (Bencsáth et al. 2012b), (Virvilis et al. 2013).

This attitude is somewhat to be expected, as any statement that brings into question the effectiveness of current security solutions will inevitably affect vendor revenue. However, lying about the problem creates a false sense of security, and organizations continue to spend vast amounts of money on security solutions, believing that this is the way to address the APT issue. Additionally, organizations that believe the "APT stopper" label will usually not invest in additional countermeasures that could help them detect sophisticated attacks, as they believe that they are adequately protected.

2.6 Related Work

Almost three decades ago, Butler Lampson described how, in the absence of total isolation, it is impossible to safeguard data from unauthorized access and programs from unauthorized execution (Lampson 1973). Later, Fred Cohen claimed that *"Precise detection (of a computer virus) is undecidable"*. Paul Helman demonstrated that the intrusion detection problem is NP-Hard (Helman et al. 1992), which means that it is a decision problem that cannot

be resolved in polynomial time in any known way, although it is possible to compute approximations to the solution.

The most widely used approach in intrusion detection is signature-based detection, the shortcomings of which have been discussed extensively in the literature. A simple testing methodology using known attack patterns revealed substantial limitations in intrusion detection systems (Berthier et al. 2010), (Modi et al. 2013), while detection rates have been shown to change drastically using small variations of known attack patterns (Brown et al. 2002).

For many years, the research community has been focusing on alternative attack detection methods (behavioral/statistical) (Denning 1987). Machine learning techniques have been successfully used in certain domains, yet despite extensive academic research efforts, such systems have had limited success in the field of intrusion detection (Tsai et al. 2009), (Sommer & Paxson 2010). Although some of these systems show promising results in controlled environments (detection rates higher than 90 percent and false-positive rates lower than 0.33 percent), they achieve unsatisfactory results in real environments (Hadžiosmanović et al. 2012). Furthermore, behavioral/statistical analysis may fail to detect known attacks that could easily be detected with a signature-based intrusion detection system (IDS), if these attacks do not differ significantly from what the system establishes to be normal behavior (Brown et al. 2002).

In the last few years, developers of IDSs have tried to take advantage of cloud-based approaches to optimize their detection capabilities. Cloud-based approaches allow for the collection of information from a very large pool of users and data analysis at a central point. There has been research on the development of fully distributed IDSs to address the limitations of central and hierarchical systems (Snapp et al. 1991), (Locasto et al. 2005), (Dash et al. 2006), however only small-scale implementations have been developed. Although centralized detection may enable quicker responses to emerging threats (e.g. a new fast-spreading worm), it offers limited benefits against APT, because in targeted attacks, the number of infections (usually a handful) is too low to raise an alarm. In addition centralized detection introduces important privacy issues (M. Theoharidou et al. 2013). Similarly, early warning systems (EWS) (Kijewski 2004), which are used extensively for the detection of emerging threats (e.g. worms, botnets), face significant limitations in identifying threats that affect only a very small number of individuals. EWSs and Network IDS/IPSs also face scalability issues, owing to the continuous increase of data traffic.

Regarding insider threat detection, several models have been proposed in the literature. One of them uses multiple but difficult to quantify indicators, such as personality traits and verbal behavior, to predict insider attacks (Schultz 2002). Another model uses the attributes user knowledge, privileges and skills as metrics (Wood 2000). Hidden Markov Models have also been used to infer divergence between the activity patterns of a user and a set of activity

models that are in place (Thompson 2004). Psychological attributes of an insider, such as introversion and depression, have been identified and dealt with (Shaw et al. 1998). The connection between intent and user action has also been investigated through experiments (Caputo et al. 2009). Finally, user sophistication (computer skills) has been used as a metric for the detection of insiders or as a component of a more general insider detection model (Magklaras & Furnell 2005).

The aforementioned models focus on both the prevention and the detection of the insider threat, and draw upon other disciplines, such as psychology. The potential role of criminology theories has also been examined (Theoharidou et al. 2005), and best practices for the prevention and detection of insider threat have been proposed (Cappelli et al. 2009). Additionally, insider detection models using psychometric tests to assess a user's predisposition to malicious acts have been proposed (Kandias et al. 2010), (Kandias, Virvilis, et al. 2013), as well as models which use data from social media (Gritzalis et al. 2014), (Kandias, Galbogini, et al. 2013), (Kandias, Stavrou, et al. 2013). Finally, the use of business process monitoring tools for insider threat detection has also been investigated (Stavrou et al. 2014).

The technical approaches rely mainly on the detection of insider activity. In (Caputo et al. 2009), the system detects violations of an already existing set of policies. Another system detects anomalous user search behavior by applying machine-learning algorithms to analyze collected search events (Bowen et al. 2013).

Despite the amount of research being done on intrusion and insider threat detection, and the special focus that has been given on the protection of critical infrastructures (Soupionis & Benoist 2014), (Soupionis et al. 2014), (Stergiopoulos et al. 2015), (Kokolakis et al. 1998), (Kotzanikolaou et al. 2013a), (Lekkas & Gritzalis 2007b), (Lekkas & Gritzalis 2007a), (Polemi et al. 2013), (Theoharidou et al. 2011), (Gritzalis et al. 2013), (Kotzanikolaou et al. 2013b), (Gymnopoulos et al. 2003), (Dritsas et al. 2005), very little of it has been focused on APT. As mentioned, sophisticated attackers will use all available means to fulfil their objective and thus, even if a foolproof intrusion detection or insider threat detection system were developed, the attackers would just choose a different attack path.

Research on deception techniques (e.g. honeypots, honeynets, honeyfiles), has resulted in a number of useful models than can increase the detection efficacy even against sophisticated attacks. Honeypots and honeynets have been proposed for attack detection on both Internet facing and internal systems (Lance Spitzner 2003), (Thonnard & Dacier 2008) (Nazario 2009), (Mairh et al. 2011), for the detection of botnets (Wang et al. 2010) and attacks on wireless networks (Prathapani et al. 2009), (Bowen et al. 2010). In addition, honeyfiles have been proposed for the detection of unauthorized access to resources (Voris et al. 2013), (Bowen et al. 2013) while honeywords – a type of honeyuser – for the detection of compromised credentials (Juels & Rivest 2013).

However, most of the aforementioned deception techniques, do not take into account the *modus operandi* of sophisticated attackers and thus, their efficacy against them is questionable. For example in (Bowen et al. 2013), the honeyfiles (decoys) include code which is executed when the document is opened and reports back to a monitoring server. This approach will not work if the attackers exfiltrate the file and open it from a location where the monitoring server cannot be accessed (e.g. offline system).

Finally, (Wang et al. 2013) have proposed a model similar to the one presented in this thesis, however their model is only based on a small number of deception techniques. In contrast, our work proposes a comprehensive APT detection model that combines multiple types of deception and anomaly-based attack indicators and allows their correlation over a wide period of time.

(This page is intentionally left blank)

Chapter 3: External Attacks

3.1 Attacker's objectives

External attacks, which are the most common type of APT attack, originate from outside the targeted organization's perimeter. The main benefit for the attackers is that they can perform their actions over the Internet from a convenient, physically remote location. These attacks entail very little risk for the attackers, as it is very difficult to attribute them to a specific individual, even if investigation results in the identification of the real source IP addresses that the attackers were using. Furthermore, such identification is unlikely, not only for APT but also for less skilled attackers, since hiding the originating IP address is a trivial task (e.g. it is possible to use anonymous proxies, TOR (TOR 2015), public hotspots, etc.).

In the very rare cases in which the attackers have been identified owing to poor OPSEC (Mandiant 2013), even if the targeted organization presses charges against them, it is very unlikely that they will face any penalties, especially if they are working for a state-sponsored group.

Although there are many ways to orchestrate an attack against a remote infrastructure, most of them follow a set of generic steps, which are presented below.

3.1.1 Information gathering

The initial step of an APT attack is the preparation phase, in which perpetrators gather as much information as possible about their target. Identification of the operating system(s), third-party software and publicly accessible services (e.g. web servers, mail servers) of the organization is crucial for planning a successful attack. Information related to the security solutions in use (intrusion detection/prevention systems, anti-malware solutions data leakage prevention, etc.) is also important, as it allows the attackers to test their tools and techniques in advance, and make sure that they are not detected/blocked.

An additional element of the preparation phase is collection of information about employees, their positions in the organization, their skills and their connections with other employees. Using such information APT can create highly targeted spear-phishing campaigns. For example, if an attacker has identified an employee working in the human resources (HR) department as well as his supervisor, she can send a spoofed email from the email address of the supervisor to the employee, asking him to review an attached file (e.g. a curriculum vitae). The attachment can be a malicious Word or PDF document that when opened, will execute the attacker's payload. The fact that the email originates from a person known to the victim substantially increases the likelihood of it being accepted as legitimate.

3.1.2 Gaining foothold and further compromise

After the attacker's payload has been executed, the compromised system(s) will usually connect back to a Command and Control (C&C) Server, where it can be accessed and controlled by the attackers (see Figure 1). After gaining a foothold on the network, the next step is to exploit other systems to use as backup entry points, in case the initial exploited system(s) is no longer accessible (e.g. in case the user reported suspicious behavior and the system was taken offline for analysis).

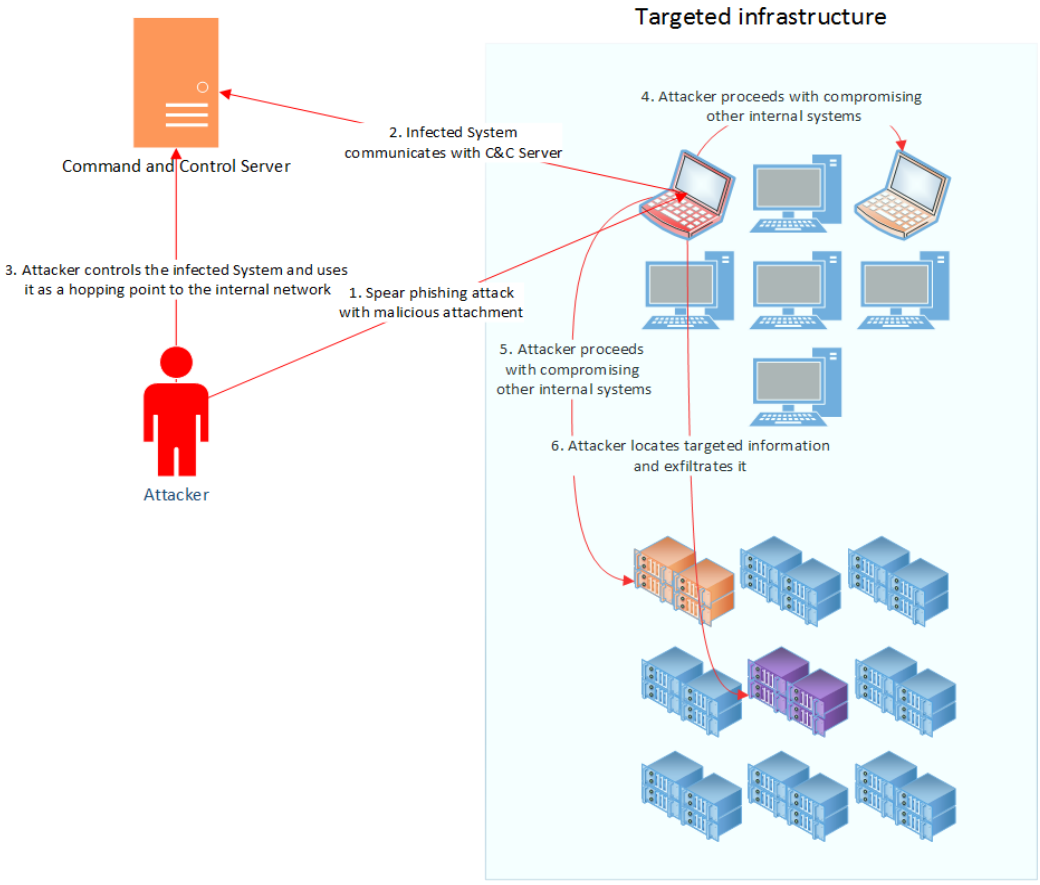

Figure 1 - Spear-phishing attack

When the attackers are satisfied with their ability to connect back to the compromised infrastructure, they will focus on locating the information that they are interested in.

As best practices mandate, networks should be segmented and communication between segments should be strictly controlled and as mentioned earlier, non-technical users are the usual targets, as they are less likely to spot an intrusion attempt. However, on a properly segmented network such users will have only limited access to production servers. Thus, attackers will have to compromise other users and systems and hop between networks in order to be able to reach the targeted information.

3.1.3 Exfiltration

The final step of the APT attack is the exfiltration of the targeted information. Depending on the size of the exfiltrated information and the egress filtering performed by the organization, different exfiltration techniques can be used. As presented in section 3.2 the majority of the APT malware tries to obfuscate its network connections and make them look like legitimate user traffic. Using SSL/TLS over known ports (e.g. connections destined to port 443/TCP), HTTP traffic with additional headers, use of steganography and DNS tunneling, are the most common techniques.

3.2 Technical review of major APT malware

In the last few years we have witnessed an impressive change in the complexity of malware. Stuxnet, Duqu, Flame, Red October, MiniDuke and Regin are examples of highly sophisticated malware, the development of which required skillful individuals - in some cases with expertise in multiple technology fields (e.g. Stuxnet) - as well as significant financial resources (Fisher 2012).

There are two main reasons that differentiate the aforementioned malware from other in-the-wild malware samples: a. they were used against sensitive state, military and industrial organizations in targeted attacks and in some cases they have been developed for attacking a specific target (e.g. Stuxnet), and b. they make use of advanced exploitation and evasion techniques, the majority of which were unknown to the security community (i.e. zero-day).

3.2.1 Stuxnet

Stuxnet's earliest sample dates back to June 2009[1]. Stuxnet's speculated purpose was to sabotage the Iranian Nuclear Program and more specifically Natanz uranium enrichment plant, something which has succeeded by causing physical damage to the infrastructure and as a result slowing down the program by four years (Lemos 2011). Stuxnet interfered with Industrial Control Systems (ICS), which were configured by Programmable Logic Controllers (PLC), and more specifically Windows systems using Siemens Step-7 software. After infecting such systems, Stuxnet would reprogram the PLC to make the centrifuges operate at speeds outside acceptable limits, causing their malfunction and eventually destruction. Stuxnet was completely autonomous - a "fire and forget weapon" in military jargon and as a result, there was very limited window for programming or logical errors from the attacker's side.

[1] This refers to Stuxnet v1. Previous version of Stuxnet (Stuxnet v0.5) were discovered later, and had been active since 2005.

In order to develop such a complex threat, the team behind Stuxnet should have at minimum: a. access to a test environment with all relevant ICS, PLC, centrifuges and supporting equipment - ideally an exact replica of the Natanz infrastructure, b. experts for installing, configuring and operating such specialized equipment, and c. a team of highly skilled programmers, and security researchers for the development of zero-day exploits used for infection of the targets (or access to a private exploit repository). Based on these requirements, it is logical to assume that the team behind Stuxnet was state-sponsored (Chen & Abu-Nimeh 2011). An interesting development on Stuxnet's case happened in 2013, where an older sample was identified, named as Stuxnet 0.5 (McDonald et al. 2013). Preliminary research revealed that it had been active since 2005, supporting a subset of Stuxnet's functionality and following a different technique for controlling the centrifuges, which was abandoned due to limited success in damaging them.

Initial infection and propagation

The initial infection method has not been identified, but taking into account that such sensitive infrastructures are usually not connected to the Internet (and most of the time not even connected to a network at all), it could be due to the use of an infected removable drive (e.g. USB stick). This is a realistic assumption, as Stuxnet actually infected removable drives. Stuxnet was able to spread, using multiple propagation methods: Once an infected USB drive was connected to a Windows system, Stuxnet would auto-execute requiring no user interaction, making use of a zero-day vulnerability (MS10-046, although older versions of Stuxnet used a modified autorun.inf technique). Also, it would try to exploit any network accessible Windows systems using the Windows Server Service (MS08-067) or Print Spooler Zero-Day (MS10-061) vulnerabilities and perform privilege escalation using MS10-073 and MS10-092.

Apart from MS08-067, the rest were unknown to the security community (zero-day). This means that Stuxnet's team had either developed those exploits internally, or had access to a private exploit repository. Other propagation methods include copying itself to accessible network shares and the infection of WinCC database server using a hardcoded database password.

Finally, Stuxnet infected Step 7 project files, which - if copied in another system and opened - would infect it. When Stuxnet managed to gain access to its target systems - a Windows System with Siemens Step-7 PLC - it would re-program the PLC, thus making the centrifuges operate outside their limits and eventually destroy them (Langner 2011).

Command and Control Servers

Stuxnet was mainly a "fire and forget" malware. However, three C&C Servers have been identified where the malware was trying to connect to in order to send basic information about the infected system.

Rootkit Functionality

Stuxnet included rootkit code to hide its binaries on Windows systems and also modified PLC code to present "acceptable" values to the monitoring software, although the centrifuges were working above the operational limits. It also used of two compromised digital certificates to sign its drivers in an additional effort to evade detection (Chen & Abu-Nimeh 2011).

Evasion Techniques

It would scan for known endpoint security products and based on product name and version it would inject its payload accordingly, to evade detection.

Encryption

Stuxnet was using XOR encryption with a static key (0xFF) to decrypt parts of its payload and a 32-byte fixed key to encode the data it sent to the C&C server.

3.2.2 Duqu

Duqu was detected in September 2011. However, it is believed that it has been active since February 2010 (Chien et al. 2012). It has significant similarities with Stuxnet, which have led researchers to believe that both threats were developed by the same team, with a different objective (Bencsáth et al. 2012a): Instead of sabotage, Duqu's objective was espionage. Duqu was a targeted malware and according to estimations infected no more than 50 targets worldwide. After initial infection, Duqu remained active for 36 days before self-destructing, although attackers could command it to persist for as long as needed. It included a key logging component which was used to collect sensitive information, such us passwords, which attackers could use to gain access to other systems on the network. Similarly with Stuxnet, it was a modular malware, which made use of compromised certificates to sign its components and thus be harder to detect (Bencsáth et al. 2012a).

Initial infection and propagation

Microsoft Word files which exploited the zero day True Type font parsing vulnerability (CVE-2011-3402) were used as the initial attack vector. The malware did not replicate on its own. Nevertheless, attackers could use an infected system as a stepping stone, for manually exploiting and infecting other systems.

Command and Control Servers

A small number of C&C Servers running CentOS Linux were identified. The malware connected to these servers over ports 80/TCP and 443/TCP, and used a custom C&C protocol. More specifically, for port 443/TCP a proprietary encrypted protocol was used. For traffic destined to port 80/ TCP, Duqu used steganography, by encoding and attaching the transferred data to JPEG image files (Chien et al. 2012), (Bencsáth et al. 2012b).

Rootkit Functionality

Similarly to Stuxnet, Duqu was using a rootkit module to hide its files.

Evasion Techniques

Duqu, having a similar list as Stuxnet, would scan for known security products and based on the product and version it would inject its payload accordingly, to evade detection.

Encryption

Duqu used AES-CBC for the decryption of executable code received from the C&C server. Additionally, it used XOR to encrypt the data captured by the key logger module and to encrypt its configuration file (Chien et al. 2012).

3.2.3 Flame

Flame was first detected in May 2012. However, it is believed that it had been already active for 5-8 years. Flame was incidentally discovered while researching for another malware infection (Bencsáth et al. 2012b). One of the most interesting aspects of this malware was its abnormal size, which was almost 20 megabytes. Based on the analysis of its code, it is unlikely that Flame was developed by the same team that developed Stuxnet and Duqu. However, considering the use of common exploits and code, it could be the case that these teams were collaborating and had shared source code for at least one module (Kaspersky 2012). Flame, like Duqu, was a targeted information stealing malware, however it was significantly more widespread than the latter as it infected thousands Windows system, mainly in the Middle East. It had a key logging module, took screenshots, intercepted email messages, used the internal microphone of the computer to record conversations and captured information about Bluetooth devices in close proximity.

Initial infection and propagation

Flame's initial infection method is unknown. It did not replicate on its own, but attackers could command it to infect other hosts using a variety of techniques: Apart from infecting USB devices, it exploited two zero-day vulnerabilities (same as Stuxnet): Print Spooler (MS10-061) and Windows Shell (MS10-046). However, the most impressive propagation technique was the

impersonation of a Windows Update Server (WSUS). As Windows software updates are digitally signed, the attackers had to perform a complex cryptanalytic attack (chosen prefix collision attack for the MD5 algorithm) against Microsoft's Terminal Services licensing certificate authority. This attack allowed the creation of valid digital signatures for Flame modules. Such advanced cryptanalytic attack required a team of skilled cryptanalysts. According to estimations, it has cost between 200K and 2M USD (Fisher 2012), another indication that such attacks are likely to be state-funded.

Command and Control Servers

Flame used more than 80 domains as C&C Servers, mostly Ubuntu Linux Servers. The communication was performed over HTTP, HTTPS and SSH protocols.

Rootkit Functionality

Flame's rootkit functionality enabled it to hide its network connections.

Evasion Techniques

Flame included an extended list of more than 100 security products and changed its infection method accordingly to evade them. Its binaries were using the .ocx extension as it is often not scanned by antivirus engines in real time.

Encryption

Flame made extensive use of encryption, using substitution ciphers, XOR encryption, and RC4 algorithm to encrypt its configuration, modules and captured data.

3.2.4 Red October

Red October was discovered in October 2012. It is believed that it has been active since May 2007, targeting diplomatic, governmental and scientific agencies (Kaspersky 2013b). Red October does not seem to have common characteristics with any of the aforementioned malware samples. It had a minimalistic architecture, with one main component responsible for connecting to the C&C Servers. When commanded to do so by the attackers, it downloaded and executed specific modules (at least 1000 different modules have been identified) enabling it to perform a wide range of tasks (Kaspersky 2013b). This small footprint was one of the main reasons it managed to evade detection for several years. Its modules allowed: Stealing of information from Nokia phones and iPhones, SNMP brute forcing, and recovery of deleted files on removable drives (e.g. usb hard drives/sticks).

Moreover, it included key logging/screen capturing functionality and intercepted outlook's email messages, calendar and contacts list. As a robust persistence mechanism, Red October installed a plugin for Office and Adobe reader applications. This plugin parsed each

opened Office or PDF file and tried to identify embedded commands (injected by the attackers) to execute. Using this technique, even if the C&C servers were taken down, the attackers could email specially crafted documents to their victims and regain control of their systems.

Initial infection and propagation

Targeted emails containing malicious Word and Excel documents which exploited known vulnerabilities (CVE-2009-3129, CVE-2010-3333 and CVE-2012-0158) were used for infecting the victims. Each sample was uniquely build for the specific target and each e-mail was also tailor-made to increase the probability of been opened by the victim. Exploitation of a Java vulnerability (CVE-2011-3544) was another infection method.

Command and Control Servers

More than 60 C&C domains were identified. However, only three hardcoded domains were included in each custom build of the malware. Additionally, none of these domains were the actual C&C servers, but acted as proxies in order to hide the real C&C infrastructure.

Rootkit Functionality

No rootkit component has been identified.

Evasion Techniques

The minimalistic architecture of the malware, having a basic component responsible for downloading encrypted modules and executing them in memory, allowed it to remain undetected without having to perform additional evasion techniques.

Encryption

Red October used a custom packer with XOR encryption. The same algorithm was also used for encrypting exfiltrated data.

3.2.5 MiniDuke

MiniDuke was discovered on the 27th February of 2013, but earlier samples have been identified and date back in June 2011 (Tivadar et al. 2013). It targeted government bodies in 23 countries, mainly in Europe.

Initial infection and propagation

MiniDuke spread over email, using malicious, well-crafted PDF files. The PDF files contained code which triggered a vulnerability in Adobe Reader versions 9, 10 and 11, bypassing the sandbox and executing the malware's payload on the victim's system.

Command and Control Servers

The malware used a sophisticated, layered technique for locating the C&C servers. It connected to specific Twitter accounts controlled by the attackers, which contained the encrypted URL's of the C&C servers. If Twitter was not accessible (e.g. twitter accounts had been blocked) the malware would use Google Search to find the encrypted C&C servers by searching for specific strings. Upon locating the servers, the malware would receive the second stage, obfuscated as GIF images, which in turn would download a larger payload (the third stage), which was the one that allowed the attackers to control the infected system. A large number of C&C servers have been identified, however all of them have been legitimate web sites which were compromised by the attackers (Raiu et al. 2013), (Tivadar et al. 2013).

Rootkit Functionality

No rootkit functionality has been identified.

Evasion Techniques

MiniDuke's first stage, written completely in assembly language, contained a list of security related processes (debuggers, disassemblers, file system monitoring software, etc.). Upon detection of any of these processes, it would remain at an idle state, not performing any malicious actions. Newer samples would also wait for user interaction (e.g. mouse movement) before decrypting and executing the payload to thwart automated malware analysis. Finally, after initial execution the malware would generate a new copy of itself - that was encrypted using a key derived from the computer's hardware configuration - and replace the original executable. As a result, the malware would only be able to decrypt correctly under this particular system, making analysis of the sample on a different system, challenging.

Encryption

MiniDuke made heavy use of encryption. Each payload (after the initial infection) was encrypted with a key generated from the CPU, Drive and Computer name of the victim. Additional layers of XOR and ROL obfuscation were also used.

3.2.6 Regin

Regin was discovered in November 2014, however it is believed that it has been active for several years, with some samples having compilation dates from 2003 (Kaspersky 2014). It has been used against a number of government organizations, telecommunication providers and industry and its main focus was information gathering. The victim organizations were geographically diverse as there have been infections in Russia, Saudi Arabia, Mexico, India and Iran and other nations. Regin used a complex six-stage architecture making extensive use

of encryption. Its very complex architecture is an indication that a skillful team has been involved in its development (Symantec 2014b).

Among other things, it was capable of capturing network traffic, stealing passwords, recovering of detected files and keystroke logging. One of the most interesting features was the ability to interact with GSM BSCs (Base Station Controllers), a feature which has been used against (at least one) GSM provider (Kaspersky 2014).

Initial infection and propagation

The initial infection point is not known, however spear-phishing attacks and zero day browser exploits are the most likely scenarios.

Command and Control Servers

A small number of command and control servers have been identified. The malware supported multiple communication protocols including transmitting data over ICMP ping packets, raw UDP & TCP and HTTP/HTTPS connections, where information was encoded in cookie data. Even more interesting is the fact that Regin supported peer-to-peer communication, which allowed infected machines to route information through other infected machines, making network detection more challenging.

Rootkit Functionality

Regin had rootkit functionality which was used to hide all its components but its first stage.

Evasion Techniques

Regin used a six-stage architecture and encryption in an effort to evade detection. All stages but the first one were encrypted. Each stage was responsible for decrypting and executing the next one. Regin was modular, allowing its operators to include different modules in each sample, depending on their needs.

Encryption

Regin used a variant of RC5 algorithm.

Table 1 - Malware comparison

	Stuxnet	Duqu	Flame	Red October	Mini Duke	Regin
Active since	June 2009 (2005)	Nov. 2010	May 2012 (2006)	May 2007	June 2011	Possibly 2003
Detected	June 2010	Sept. 2011	May 2012	Oct. 2012	Feb. 2013	Nov. 2014
PE Type	DLL		OCX	EXE	EXE	SYS/DLL
Initial infection	Unknown	MS Word	Unknown	MS Excel / Word, Java	PDF	Unknown
Replication	Removable drives, network	Manual replication only				
Rootkit module	Yes		No			Yes
Key logging	No	Yes			No	Yes
Evasion	Yes			No	Yes	Yes
Encryption	XOR	XOR, AES-CBC	XOR, RC4, Substitution	XOR	Unique per victim, XOR, ROL	RC5 Variant, XOR
Target	Sabotage	Information gathering				

3.3 Common characteristics

From the aforementioned analysis common malware characteristics are evident, such as similar techniques, attack paths and functionality. Focusing on these characteristics, the potential reasons that could have enabled the malware to evade detection, are discussed:

A. Targeted operating system and architecture

All samples were targeting 32-bit versions of Windows, with the exception of Regin, for which 64-bit samples have also been discovered. The additional security mechanisms that are available on 64-bit versions of Windows (Windows vista and newer) complicate significantly the exploitation process, especially when malware tries to gain kernel-level access (Field 2006). These countermeasures forced Regin creators to change significantly the architecture of the malware for 64-bit systems (Symantec 2014b). However, based on the fact that attackers in some cases had access to valid certificates (e.g. Stuxnet, Duqu), they could have used them to sign the 64-bit components of their malware, and thus be allowed to execute in kernel space. Hence, that the main reason that most of malware targeted 32-bit systems was most likely that the majority of the victims were using this architecture.

B. Initial attack vectors

Duqu and Red October both used malicious Word and Excel documents for infecting their targets, while MiniDuke exploited Adobe's PDF Reader. For Stuxnet and Flame the initial

infection method has not been identified, however infection through removable drives or spear phishing attacks are the most likely scenarios. Microsoft since the release of Office 2010 Suite, has introduced "protected view" (Microsoft 2013), a sandbox feature for opening office documents received from untrusted sources - such as downloaded from the Internet, network share or received as an email attachment. By default, all new files are opened in protected mode thus, any potentially malicious payload would have to face the additional barrier of escaping this security feature.

Based on the analysis of the malware samples, none of the Office vulnerabilities used by them was able to escape protected view. Thus, we have to assume that victims who got infected were either running outdated versions of MS Office, had disabled this security feature or had been tricked into opening the document in unprotected mode. This was not the case for MiniDuke, which was able to exploit all versions of Adobe Reader and evade its sandbox.

C. Command execution and escalation of privileges

All malware made use of exploits for remote code execution or escalation of privileges, the vast majority of which were unknown to the security community (zero-day). Exploitation of zero-day vulnerabilities against the Operating System is probably the most challenging problem to address. As long as these vulnerabilities remain unknown to the vendor, all affected systems could be exploited. However, one interesting aspect is that even when security patches addressing these vulnerabilities had been released, victims continued to get infected. This highlights the lack of proper patch management, even in sensitive environments.

D. Network Access

All malware communicated over ports 80/TCP, 443/TCP or 22/TCP, as egress traffic destined to these ports is frequently allowed to pass through network access control mechanisms. The communication protocols were HTTP, HTTPS, SSH and ICMP, however, additional layers of encryption/obfuscation and compression were also used. Malware's success to communicate back to the C&C infrastructure highlights the fact that most of the victims had very relaxed Internet access restrictions in place (if any) - a worrying finding, taking into account the sensitive nature of the targeted organizations.

E. Network IDS/IPS and endpoint antivirus products

Stuxnet, Flame, Duqu and MiniDuke were designed to detect and evade Antivirus Software using multiple techniques. Moreover, Flame, Duqu, Red October, MiniDuke and Regin encrypted or obfuscated their network traffic, to and from the C&C servers, so as to "pass under the radar" of Network Intrusion Detection Systems (NIDS). As a result, the level of protection offered by both antivirus and network security products against advanced threats has received strong criticism, as they failed to detect such threats - even when they had been active for several months/years (Schneier 2012).

F. Use of Encryption / Obfuscation.

All samples relied strongly on XOR "encryption" to deter detection and complicate malware analysis (packing), as well as for protecting the configuration file(s) and network traffic. Additionally, Duqu and Flame also used AES and RC4 algorithms, while Regin used a variation of RC5. The use of encryption in a program cannot be categorized as malicious on its own, however it is a useful indication during behavioral analysis of potentially malicious files.

G. Exploitation of digital signatures.

Stuxnet and Duqu binaries were digitally signed using compromised digital certificates. Thus, these samples would have managed to infect hardened systems where only digitally signed binaries were allowed to execute. Similarly, Flame exploited the collision resistance of the MD5 hash function to perform the same task.

Based on these findings, allowing execution of binaries based on the existence of valid digital signatures cannot be considered an effective defense on its own. This is particularly important for Antivirus and HIPS solutions, which tend to avoid real-time analysis of signed binaries for performance reasons.

3.4 Limiting the impact

Due to the complexity of those threats and the multiple attack paths, a wide range of security countermeasures and hardening procedures are necessary in order to reach an acceptable level of security. This section focuses on common countermeasures which can be implemented by most organizations with minimal cost, resulting to an increased level of protection against malware attacks.

A. Patch Management: Patch management, for both Operating System and third party software, is one of the most effective defenses. Although it will have no effect against zero-day vulnerabilities, it will stop further exploitation of new systems, after the relevant vulnerabilities exploited by the malware have been patched.

B. Network Segregation: Strong internal network access controls and monitoring are also crucial. In the majority of cases, multiple systems had to be exploited until the objective of an attack was met (e.g. data exfiltration or sabotage). One of the most effective techniques that could severely block the ability of the malware to spread internally is the isolation between the systems. This could be achieved by the use of network or host based firewalls, configured to allow workstations to only connect to specific systems, based on their role/business requirements.

C. Network Whitelisting: As all malware had to connect back to a C&C server for receiving commands or exfiltrating data (with the exception of Stuxnet), strict Internet access

policies and granular traffic inspection of both incoming and outgoing data, should be implemented. Instead of following a blacklist approach which can be evaded, a whitelist approach should be preferred. Depending on the risk appetite of the organization, the whitelist could even include the most popular (non-work related) websites that users are visiting, to limit the negative impact on user experience. This countermeasure would block any connection attempts to C&C servers, unless the attackers were able to exploit any of whitelisted domains and set their C&C infrastructure there.

D. Dynamic content execution: Focusing on the client side exploitation mechanisms, it is evident that the majority of end-user exploitation techniques (e.g. Malicious Office, PDF documents) required dynamic content execution for triggering the vulnerabilities. Content protection mechanisms at the network ingress points could filter dynamic content in incoming traffic (e.g. JavaScript from PDF files, macro code from Office files etc.), thus protecting against a wide range of vulnerabilities (Lagadec 2006).

3.5 Indirect attacks

3.5.1 Supply chain attacks

Supply chain attacks can be considered indirect, external attacks. The malicious actors by compromising the supply chain are able to make unauthorized changes to hardware or software components, which will be used in the targeted infrastructure.

Interest in supply chain security is not something new, in a matter of fact, significant effort has been made since 2001 on addressing such security issues (Miller 2013), (Williams et al. 2008), (Lee & Whang 2005), (Kandias et al. 2011). Most of the technology products nowadays are produced and assembled in Eastern countries, due to the lower labor cost. In addition, it is not uncommon for the manufacturers of these products, to have multiple subcontractors which in turn have their own subcontractors, making tracing and auditing of the supply chain challenging. Even if assembly of a product is made under strict control in a trusted location, use of any components in those products coming from third party providers, introduces a potential risk.

Until recently, supply chain attacks although possible, were mostly regarded as a potential, but not an eminent threat. However, based on Snowden's disclosures things have changed dramatically (Spiegel 2013), (Schneier 2013). According to the disclosures, NSA had been intercepting equipment while been shipped from the supplier to the purchaser, modifying it in such a way that allowed them remote access to it in the future. The list of products was vast (Spiegel 2013), ranging from firewalls and routers of major US manufactures, to personal computers, servers and even hard drives and mobile phones.

A number of countermeasures, such as side-channel fingerprinting (e.g. analysis of power consumption) or use of X-Rays, have been proposed for the detection of modified hardware and low level software (Edwards 2013), (Guin et al. 2014). However, such techniques are mostly effective for testing very specific components (e.g. a single chip) and not a complete system. A slight code modification in the authorization function of a network device would be very challenging - if not impossible - to detect with such techniques.

As a result, supply chain attacks pose a major detection challenge and the research community needs to focus upon this complex threat.

3.5.2 Attacks against third party providers

Another kind of indirect external attack is the exploitation of third party providers, which are used by the targeted infrastructure. These can be Cloud Service providers or any other type of provider, which the target is interacting and exchanging data with. The attack against RSA (RSA 2011), was a clear example of an indirect external attack, as the attackers goal was to exfiltrate information which would allow them to gain access to companies/organizations which were using RSA security products for authentication.

From the defender's perspective, there are limited options for preventing such indirect attacks, mainly due to the fact that they have no control over the provider's infrastructure. However, depending on the type of outsourced data or service, the defenders may be able to implement additional measures that will limit the impact in case of an attack against the provider. Cloud storage services, which are becoming continuously more popular due to the vast storage space they offer at very affordable prices, are a good example. Virtually all services support encryption during data transmission, but only a subset of those support data encryption at rest, and even fewer encrypt the data on the client side (Virvilis et al. 2011b). The latter, is the most secure option, as even if the cloud provider gets compromised and the attackers gain access to the stored data, encryption will safeguard data confidentiality.

Depending on the type, volume of data and security requirements, organizations can either choose a provider which supports data encryption on the client side, or use third party software which acts as a transparent encryption layer (Cloudfogger 2015), (Boxcryptor 2015), for providers that do not offer such functionality. Finally, a Cloud provider agnostic secure storage protocol has been proposed (Virvilis et al. 2011a), which offers similar functionality with the aforementioned commercial solutions, but is based on an open architecture.

(This page is intentionally left blank)

Chapter 4: Internal Attacks

4.1 The insider threat as a subset of APT

Information systems face several security threats, a number of which may originate from internal actors (i.e. insiders). The motives of insiders vary and can be based on revenge or can be financial, ethical or political (Gellman & Markon 2013).

The definition of a malicious insider based on Silowash et al. (Silowash et al. 2012) is:

> *"... a current or former employee, contractor, or business partner who meets the following criteria:*
>
> - *has or had authorized access to the organization's network, system, or data*
> - *has intentionally exceeded or intentionally used that access in a manner that negatively affected the confidentiality, integrity, or availability of the organization's information or information systems."*

APT and insiders tend to share specific characteristics that markedly differentiate them from traditional (i.e. opportunistic) attackers. Firstly, their attacks require detailed planning and are frequently spread over a long period of time, in an effort to evade detection. Secondly, both groups are willing to explore all possible attack paths for reaching their goals, including social engineering and use of deception (Hudson 2013).

The main difference between the two groups is that malicious insiders have by definition authorized access to the infrastructure and potentially to the servers storing sensitive information (e.g. file servers, database servers). Furthermore, they may be aware of existing security controls. This is very likely if the insider holds a privileged position, e.g. an administrator is expected to have knowledge of the deployed security mechanisms and potentially has the access rights to control them, while a less privileged user would not. Nevertheless, experience has shown that sophisticated attackers (e.g. APT) have also managed to reach their goals while evading detection without prior knowledge of the infrastructure (Zetter 2010).

APT and insider threats are usually considered to be two different threat groups in the literature. However, given the fact that APT groups that have used insiders to perform malicious actions on their behalf (Kelley 2012), it is our belief that insiders should be regarded as a subset of APT.

Robust models have been proposed for the detection of insider threats (Kandias et al. 2010), however they are based on the assumption that the malicious insider will perform the

entire attack life-cycle on his own (information gathering, exploitation, exfiltration). Yet, in the Stuxnet case (Kelley 2012), a malicious insider was used only to deliver the payload, while the rest of the exploitation was performed in an automated way. This attack strategy, which combines APT with the insider element, poses a serious challenge for such models. Taking into consideration the substantial resources available to APT groups (Fisher 2012), similar attacks should be expected in the future.

4.1.1 Major insider attacks

In discussions of insider attacks, two names are often mentioned first: Chelsea Manning and Edward Snowden. Both Manning and Snowden leaked thousands of classified documents to the public, causing a substantial amount of frustration to the United States and their allies. Their *modus operandi*, technical skills and access levels differed significantly, as did their goals and objectives.

4.1.2 Manning's disclosures

Chelsea Manning (born Bradley Manning) is a discharged US Army soldier convicted of leaking several thousand classified documents to the public (Denver 2012). Manning joined the Army in 2007 and in October 2009 was sent to Iraq as an Intelligence Analyst, where she had access to a vast amount of classified information. In early January 2010 she downloaded classified information about the Iraq and Afghanistan wars and leaked it to the WikiLeaks web site. Manning was arrested shortly after and prosecuted. She was sentenced in August 2013 to 35 years of imprisonment and was dishonorably discharged from the Army. One day later, she announced that she is a female and asked to be referred to with her new name (Chelsea).

WikiLeaks published a number of the leaked documents, including videos of airstrikes that had cost the lives of civilians, including two Reuters employees (Barnes 2013). Although some of the disclosures were "problematic" for US diplomacy, in the end the damage was less serious than initially anticipated (Strobel 2013).

According to Manning, her motives were humanitarian and political. In her court statement she states that release of these documents *"could spark a domestic debate on the role of the military and our foreign policy in general as it related to Iraq and Afghanistan"*, and that *"... the detailed analysis of the data ... might cause society to reevaluate the need or even the desire to engage in counterterrorism and counterinsurgency operations that ignore the complex dynamics of the people living in the affected environment every day."* (Manning 2013).

4.1.3 Snowden's disclosures

Edward Snowden is a former NSA contractor. In early June 2013 he disclosed thousands of classified documents, exposing intrusive global surveillance programs, compromised cryptosystems and multiple attack tools and techniques that the NSA was using to perform mass surveillance both on US soil and abroad (Hosenball 2013). Shortly after, on 14 June, he was charged with theft of government property and violation of the espionage act and had his passport revoked. On 23 June, he flew to Russia, were he was granted asylum.

Snowden released a wealth of information regarding NSA operations and tools, including information about:

- Telecom operators that were sharing their customer phone records with the NSA (Greenwald 2013)
- The PRISM program, which allowed the NSA to access the user data of a number of Internet giants including Google, Yahoo, Apple and Facebook (Greenwald & MacAskill 2013).
- The NSA spying on a large number of world leaders, including German Chancellor Angela Merkel (Poitras et al. 2014)
- Multiple efforts of the NSA and GCHQ to cryptanalyze or weaken encryption mechanisms, standards and tools (Ball et al. 2013)
- NSA and GCHQ monitoring of Google's and Yahoo's data-center links, which allowed them access to "bulk access" of data (Gellman & Soltani 2013)

Snowden's disclosures fueled worldwide debates over privacy and mass surveillance. An NSA reform bill was proposed in US Congress (Timm 2014), however it was later dropped. Industry's response was much more aggressive: Google's chief legal officer, David Drummond, said in a statement: *"We have long been concerned about the possibility of this kind of snooping, which is why we have continued to extend encryption across more and more Google services and links, especially the links in the slide. We do not provide any government, including the U.S. government, with access to our systems. We are outraged at the lengths to which the government seems to have gone to intercept data from our private fiber networks, and it underscores the need for urgent reform"*. Since then, Google has started encrypting the traffic between its own data centers, something that is expected to hinder NSA's mass collection capabilities (Google 2014). Yahoo and Microsoft have also followed with similar statements and actions (Mayer 2014), (Timberg 2013).

Germany, after the disclosures, started investigating the possibility of restricting its national Internet traffic to remaining always within its borders, in an effort to defend against

eavesdropping, and similar efforts are being made by other European Union member nations (Birnbaum 2013).

4.1.4 Public Opinion

Both Snowden and Manning have been accused as traitors by the U.S. Government (Carrie 2014), (Pilkington 2013). However public option differs, as according to a study 67% of the Canadians and 60% of the British are considering Snowden a hero (Freeman & Edwards-Levy 2013). Snowden has been voted as “The Guardian's person of the year” for 2013, has been awarded the first place on the 2013 list of leading Global Thinkers (Foreignpolicy 2013) and was the “Time’s person of the year” (2013) runner up, while in 2014 he was included in the Times list of most Influential People in the world (Times 2014). Manning, has been awarded the “Whistleblowerpreis” by the German Section of the International Association of Lawyers against Nuclear Arms and the Federation of German Scientists as well as the "People's Choice Award", “the Sean MacBride Peace Prize” and “the Sam Adams Award” (Sam Adams Award 2014).

Regardless on the different views and the motives of the individuals, the grave impact of a successful insider attack has been made crystal-clear. The insider threat is a multi-dimensional problem, which cannot be addressed solely with technological countermeasures, especially when the perpetrators have privileged access to the infrastructure (e.g. like Snowden). As a response to this challenge, in the following section a novel insider threat detection model is proposed.

4.2 An insider threat prediction model

This section presents a prediction model which combines a user taxonomy with approaches from psychology and monitoring techniques (e.g. system call analysis, IDS and honeypots). The proposed model (Figure 2) collects two types of information about a user. The first type is user characteristics, collected by the Psychological Profiling component. The model also analyses data from the IT components of the information systems in order to collect usage information about the user. This is the role of the Real Time Usage Profiling component. All the information collected serve as input in the Decision Manager, which assesses whether a user is potentially dangerous or not.

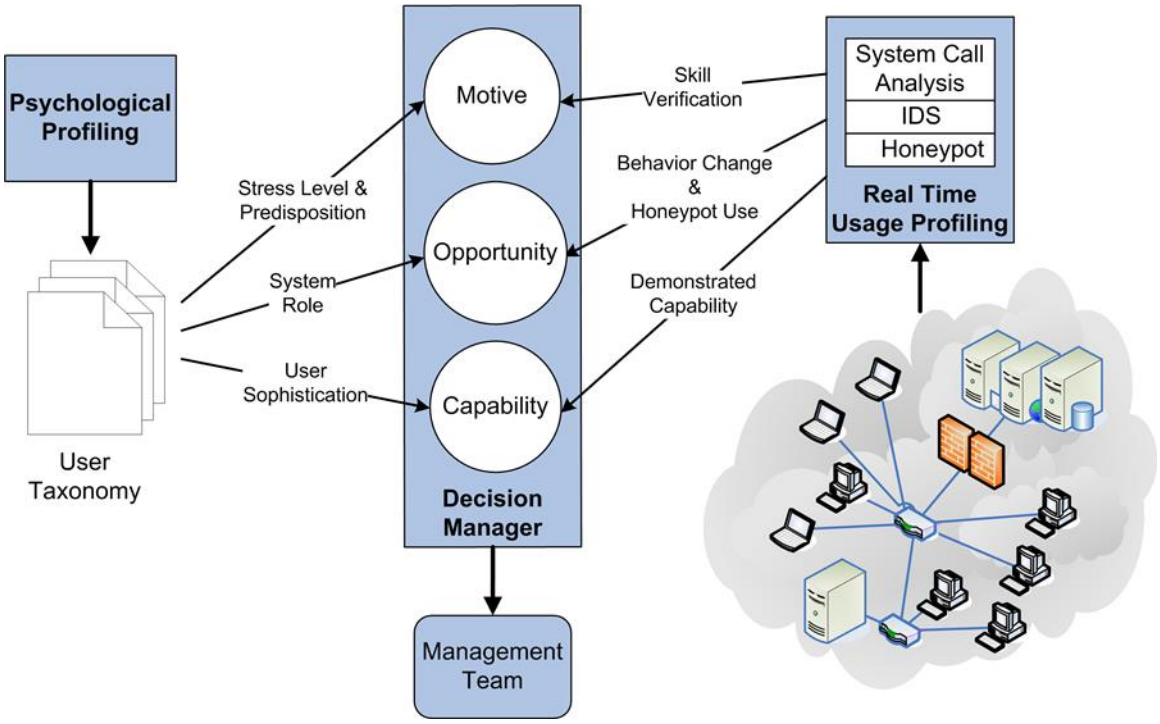

Figure 2 - Insider threat prediction model

When an organization uses monitoring techniques and psychological tests as in this model, two conflicting requirements emerge: the security of the organization (assuring business continuity and profit) vs. the employees' privacy (Mitrou & Karyda 2006). This issue is highly dependent on the organization's context and legislation. Thus, the above conflicting requirements should be weighed and the model should be developed in a way that fully complies with the organization security policy, its culture and the legal framework.

4.2.1 Component: User Taxonomy

The user taxonomy is the first building block of the threat prediction model. More specifically, each user is categorized on four dimensions:

System Role {Novice, Advanced, Administrator}. This dimension determines the access rights of a user. This value is not expected to change often and is defined in advance by the management team. Although an administrator is expected to receive a high user sophistication score (see below), this is a different category than this one. For example, a department manager can be an advanced user, in terms of access rights, but his computer skills may not be advanced.

Sophistication {Low, Medium, High}. The user's capabilities are examined under the prism of three further dimensions: "range of knowledge", "depth of knowledge" and "skill" (Magklaras & Furnell 2005). User sophistication is assessed using the formula:

$F_{sophistication} = F_{breadth} + F_{appscore} + F_{resutil}$, where $F_{breadth}$ indicates how many different applications has the user utilized, $F_{appscore}$ indicates user's sophistication regarding the type of applications he invokes and $F_{resutil}$ that represents the arithmetic sum of three computational resource consumption indicators (CPU, RAM and simultaneous applications running). Magklaras and Furnell proposed this technique as a method to measure the actual computer skills of a user (Magklaras & Furnell 2005). This technique is used in the model in order to verify the accuracy of the stated "computer skills" of a user, in parallel with the psychological profiling. Later, when usage information is also available (by the system call analysis module), these values can be updated.

The following two dimensions are assessed during the psychological profiling of the user:

Predisposition {Low, Medium, High}. This refers to the tendency of a user to demonstrate malevolent behavior.

Stress Level {Low, Medium, High}. It measures the current degree of personal and professional stress that a user experiences.

4.2.2 Component: Psychological Profiling

This component draws upon Industrial and Organizational Psychology. In specific, it applies techniques of the Social Learning Theory (Theoharidou et al. 2005). The profiling contains three stages: the first determines a user's sophistication, the second assesses predisposition to malicious behavior and the third assesses stress level. The results of the test are used to populate the user taxonomy and also as part of the decision process ("motive" and "capability" dimension).

User Sophistication. The questionnaire includes questions regarding the computer skills of a user. These questions require from the user to evaluate her knowledge on computer usage, in terms of operating systems in use, techniques implemented, familiarization with specific technologies, etc. These questions, although technical, are used in the beginning of the interview, in order to make the user familiar with the process, as well as to encourage his cooperation. These questions allow the management team to initialize the user sophistication attribute in the taxonomy. This is later updated and verified by the technique of Magklaras and Furnell (Magklaras & Furnell 2005).

Predisposition. It utilizes the CCISLQ questionnaire (Rogers 2001) in order to measure the following parameters for each user: (1) Demonstration of delinquent behavior in the

past, (2) Imitation and ability to reproduce ideas, (3) Level of influence from family and friendly environment, (4) Differential association, (5) Perception of punishment and balance of punishment and rewards, (6) Moral disengagement, (7) Sense of collective responsibility, and (8) Blaming or devaluation of a victim. Depending on the answers given, each user can be categorized as: "Low", "Medium" or "High".

Stress level. Regardless of predisposition, a user has to experience something stressful to trigger the above tendency (Heuer & Herbig 2001). This parameter is assessed by a psychometric test which evaluates both personal and professional stress (Puleo 2006). The factors examined include personal stressful triggers (e.g. death of spouse, financial difficulties, etc.) or triggers from the work environment. Such tests already exist and they can be customized, so as to embody characteristics of various organizations (e.g. military employees are expected to experience higher stress levels than others). The selected test is based on the multidimensional Rasch model (Rasch 1993). The output represents a snapshot of the user's current state of stress as "Low", "Medium", or "High".

4.2.3 Component: Real Time Usage Profiling

This component monitors the user interaction with the technical components of an information system. User behavior is monitored in real time. Usage information is collected from networks, operating systems, databases and applications. The modules of the model are as follows:

System calls analysis. The idea of using system call analysis to detect potentially malicious actions is not new (Liu et al. 2005). There have been attempts to detect attacks using n-grams (Forrest et al. 1996), (Hofmeyr et al. 1998), frequency analysis (Liao & Vemuri 2002), etc. For example, it has been demonstrated that the average user presents predictable behavior regarding daily file usage (Nguyen et al. 2003). Also, it appears that users have a concrete behavior when accessing specific files for a standard number of times daily (Nguyen et al. 2003). System call analysis can be used to create behavioral patterns that trigger an alarm when violated with a low false positive rate. This behavior pattern can be reinforced by application execution analysis, which is a variation of system call analysis, and searches for usage patterns of applications (Nguyen et al. 2003).

The previous module is used as part of the decision model measuring several parameters. It examines whether a user's behavior has changed or not, which is the "change of behavior" parameter (dimension "opportunity"). It also examines whether the user is demonstrating behavior which exceeds his defined skills. This is determined by two parameters, i.e., the Boolean parameter "skills verification" (dimension "motive") and the scale parameter

"demonstrated capability" (dimension "capability"). It can also provide data for the assessment of the "user sophistication" attribute in the taxonomy.

Intrusion Detection System. The output of an IDS can be used in order to decide whether a user has modified his behavior, which affects the following parameters of the decision module (see Section 3.4): "behavior change", "skills verification", and "demonstrated capability" attributes. Like above, it can also provide data for the assessment of the "user sophistication" taxonomy attribute.

Honeypot. Honeypots are used to attract malicious users. Their advantages include the collection of a small amount of data, the low percentage of false positives/negatives, flexibility and adaptability (Lance Spitzner 2003). A honeynet is a network of (usually) virtual computers, which wait to be targeted. Any interaction with these systems is an indication of an attack, as users are not expected to connect to them. The model uses them, so as to evaluate whether a user is trying to exploit an opportunity to attack the information infrastructure (dimension opportunity), or not.

4.2.4 Component: Decision Manager

The previous components include a number of heterogeneous techniques. When these components are properly combined with the user taxonomy, they can assess potential insider behavior. The model adopts Wood's assumption that each threat requires: (a) motive, (b) opportunity, and (c) capability (Wood 2000). Each factor receives an assessment of the following form: (1-2) low, (3-4) medium, and (5-6) high. Assessment of these factors is described below.

Factor: Motive. The last dimension of the first stage of the model is the motive of the user to launch an attack (see Table 2). The motive of a user M_i is assessed using three parameters: (a) predisposition to malicious behavior P_i, (b) current stress level S_i, and (c) skill verification V_i.

$$M_i = f(P_i, S_i, V_i)$$

At first, the user's predisposition to malicious behavior is accessed through the test mentioned in section 4.2.2. This parameter is important as it indicates the user's tendency to misbehavior however, users of "High" predisposition are not considered as an *a priori* insider threat. The second parameter is the stress level that the user experiences before he decides to commence an attack. A high level of stress can be an enabler for a user to start an attack, as it helps him overcome his moral inhibitions (Heuer & Herbig 2001). The last parameter is the verification of skills, which the user has declared during the psychometric tests.

The model considers unexpected the fact that a user was proven to have considerably higher skills than he initially declared.

Table 2 - Motive score

Skill Verification	Stress Level	Predisposition		
		Low	Medium	High
False	Low	1	2	3
	Medium	2	3	4
	High	3	4	5
True	Low	2	3	4
	Medium	3	4	5
	High	4	5	6

Factor: Opportunity. A malevolent user usually requires an opportunity in order to launch an attack. The opportunity level O_i of a user i, depends on three parameters: change of work behavior B_i, system role R_i, and honeypot use H_i.

$$O_i = f(B_i,R_i,H_i)$$

Any change in the user behavior during the interaction with the information system may theoretically indicate that the user is in the process of finding a possible target in the system, or that he is trying to exploit one. The second parameter is the user role in the system, which can be “novice”, “advanced” or “administrator”. The last parameter is the potential user interaction with the honeypots; a user is not expected to access a honeypot for a legitimate purpose. However, any user may accidentally access a honeypot (even an administrator), assuming that they are not aware of their existence. The opportunity factor can be assessed through on a scoring table (see Table 3).

Factor: Capability. The skills of a user are already defined by the user sophistication attribute in the user taxonomy S_i, as a numerical value. However, the IDS and the call analysis module may indicate that the user has the ability to use considerably more advanced skills than the ones assessed. This is indicated by the parameter “Demonstrated Capability” D_i, which is measured by these two modules as “Low”, “Medium”, “High” and “Very High”. These parameters may seem similar but, if a user demonstrates advanced skills e.g. use of an automated exploitation tool, this does not mean *per se* that his user sophistication is advanced. The capability factor can be assessed by using a scoring table like Table 4.

$$C_i = f(D_i,S_i)$$

Table 3 - Opportunity score

Behavior Change	Honeypot Use	System Role		
		Novice	Advanced	Administrator
False	False	1	2	3
	True	2	3	4
True	False	3	4	5
	True	4	5	6

Table 4 - Capability score

Demonstrated User Capability	Sophistication		
	Low	Medium	High
Low	1	2	3
Medium	2	3	4
High	3	4	5
Very High	4	5	6

Decision algorithm. After the above mentioned factors have been assessed, the model has a component that can decide whether the user arouses suspicions and should be closely monitored. Every organization that uses this model must set a scoring system for the dimensions mentioned above. This system cannot be universal, because every organization has different security needs, demands, staff, and philosophy. As a result, every organization adopting this model should study several parameters before deciding which scoring system to use. Some of these parameters are the average and the fluctuation of the results, the required strictness, possibly a risk analysis, etc.

Herein a simple scoring system is proposed, so as to demonstrate the role of the Decision Manager. As already mentioned, every dimension classifies the users into three categories: low (1), medium (2), high (3). Hence, if user i is assessed as user with “high” motivation (M_i= 3), “medium” opportunity, (O_i= 2), and “high” capability (C_i= 3), then his threat score T_i equals to 8 points.

$$T_i = M_i + O_i + C_i$$

After assessing each user’s final score, a scoring system is used to map the user into a category indicating how potentially dangerous he can be. Each organization has to choose the score intervals that classifies each user to a category. For example, four intervals can be used (3, 4), (5, 6), (7, 8), and (9), which map the user into the categories: “harmless”, “medium risk”, “dangerous”, and “very dangerous”, respectively (Table 5).

Table 5 - Overall threat score

Motive	Opportunity	Capability		
		Low	Medium	High
Low	Low	3	4	5
	Medium	4	5	6
	High	5	6	7
Medium	Low	4	5	6
	Medium	5	6	7
	High	6	7	8
High	Low	5	6	7
	Medium	6	7	8
	High	7	8	9

Note: Low =1, Medium = 2, High = 3

These results are then examined and evaluated by the management team. It is important to mention that the model determines an estimated value of the potential danger a user may pose to the organization. However, it does not receive any automated decision regarding the particular user access rights.

4.3 Requirements and limitations

The proposed model has a number of requirements which have to be met in order to be effective: The monitoring points for the IDS sensor(s) have to be correctly selected and as a minimum, should cover all traffic between the workstations and servers.

Sensitive applications that may be targeted by insiders should have logging capabilities, so that all user actions can be logged and analyzed for potentially malicious behavior. Ideally, the logging of user actions has to be in (semi) real time. The storage space for the log file should be taken into consideration, as - depending on the application - the size can increase rapidly, making storage and analysis cumbersome. This is particularly important for databases, as the database logging mechanism is frequently disabled, due to the performance overhead.

Additionally, all sensors should be able to communicate directly and in a secure way with the Decision Manager, which will analyze all reported data and decide if a particular user action/behavior has to be further analyzed. A considerable level of expertise is also required from the network and system administrators, as monitoring the aforementioned resources requires manual reconfiguration of the operating systems and network devices. Performance issues need also to be taken into account.

A clever insider might change his behavior slowly, trying to fool the Decision System and prevent it from identifying his attack as abnormal behavior. Furthermore, in case the insider has authorized access to specific resources she can access a small part of them each day, simulating a normal user behavior, knowing that accessing all information at the same time would raise an alarm. His attack will succeed and probably pass undetected, but he will need much more time to conduct it. Another important limitation is, that in case the insider has administrative rights, he can disable the sensors/logging mechanisms and thus avoid detection.

Regarding psychological profiling, when the model is applied in different organizations, it is likely that its statistical parts may vary significantly. For example, the user computer skills in a software house can differ significantly from another organization. This could be addressed, if every organization adopts different statistical constants according to its own needs and characteristics. This also applies to the algorithm for the classification of users in categories. Every organization can develop its customized system of classification. Furthermore, the model should study each user's stress level throughout time, in comparison with the results of the user's predisposition to malicious behavior and his behavior in the information infrastructure.

In addition, as already mentioned, when an organization applies monitoring techniques and psychological tests as in this model, compliance with the legal requirements is imperative.

Finally, the model assumes that the whole attack life cycle, will be performed by a single individual. As discussed, there has been at least one sophisticated attack where an insider was used in order to perform a very specific action (e.g. infecting a system), while the rest of the steps were done in an automated fashion. Due to this limitation, the current insider detection models including the proposed one, will be ineffective against such attacks. As a result, different detection strategies like the ones presented in the APT detection model in Chapter 7, need to be investigated.

(This page is intentionally left blank)

Chapter 5: Indirect Attacks: Mobile devices

5.1 The exponential growth of mobile devices

Mobile devices and more specifically smartphones and tablets, have become an integral part of our life. Apart from facilitating communication, smartphones are frequently used for a number of sensitive tasks which include online banking and shopping, acting as two-factor authentication tokens for online services, and wireless payments. Furthermore, the use of smartphones in business under the Bring Your Own Device (BYOD) trend is continuously increasing, even in sensitive environments, with iOS and Android devices getting accredited for use in the US Dept. of Defense (Taborek & Capaccio 2013).

Smartphones are now being the primary device used to access the web (Gartner 2014b). However, while browsing the web users might visit rogue web sites, namely sites that serve malicious software (malware) and/or host phishing scams. It is worth clarifying that users might be exposed to such threats not only when they visit nefarious web sites, such as adult websites, ones hosting pirated software or gambling sites. Benign sites (e.g. social media websites, search engines, news sites, etc.) have been misused in the past to deliver phishing/malware attacks after being compromised (i.e., watering hole attacks) (Cisco 2013). As a result, the likelihood that users will be exposed to such threats, should not be neglected. In fact, Symantec states that 38% of mobile users have experienced mobile cybercrime in past 12 months (Symantec 2014a).

The malware threat is constantly increasing, with Kaspersky Labs reporting that over 3 billion attacks were detected in 2013, with a total of 1.8 million malicious or potentially unwanted programs used in these attacks (Kaspersky 2013a). McAfee Labs Threat Report highlighted that there is a 167 percent growth in mobile malware between 2013 and 2014 (McAfee 2014b). The lack of security awareness of mobile users is well known (Mylonas, Kastania, et al. 2013) and this has been confirmed by a recent report that reveals that 57% percent of adults are unaware of the existence of security solutions for mobile devices (Symantec 2014a). This discovery is very worrying if we take into account the sensitive information stored on smartphones as well as their use to access business resources (e.g. accessing corporate emails, files etc.).

Most modern operating systems (e.g. Windows 7 and newer, Mac OS X 10.9 and newer) include built-in security features such as a firewall, malware detection software, automated security updates and basic auditing. Thus, even the less security conscious users are enjoying a basic level of protection. Unfortunately, this is not the case for smartphone platforms. Especially for the Android platform malicious applications have found their way even on the official marketplace (Zhou et al. 2012), causing data loss, exfiltration of information

and phone charges etc. The availability of endpoint protection software is limited and its effectiveness questionable. The same is true for the auditing/logging features of mobile platforms. As a result, it is logical to assume that smartphones will be the next APT target, as they offer an easy way to gain access to personal and business data, with significantly less risk of detection for the attackers.

Phishing is one of the most popular and profitable attacks as almost 450.000 attacks happened in 2013 with an estimated loss of over $5.9 billion (RSA 2014). In addition, one in every 392 emails contained a phishing attack in 2013 (Symantec 2014a) and to make things worse, 80% of business users were unable to detect phishing attacks effectively (McAfee 2014a). If we take into account the increased exchange of emails on mobile devices (Gartner 2012), users are very likely to access phishing pages on such a device.

Although the majority of phishing attacks are widespread and focus on financial gain, targeted phishing attacks also exist. These attacks are known as spear-phishing and have been used in a large number of sophisticated attacks against government, military and financial institutions. The analysis of multiple security incidents has revealed that attackers used targeted phishing attacks in order to gain access to the internal network of their target (Drummond 2010), (RSA 2011), (Virvilis et al. 2013).

Web browsers are the first (and unfortunately in some cases the last) line of defense on mobile platforms against such attacks. Although it is unrealistic to expect that targeted phishing/malware attacks will be detected by the web browser or any third party security solution (e.g. Antivirus engine), what is evident from our analysis (see 5.5.4) is that the available security mechanisms on smartphones are failing to address even the most basic, non-targeted web threats. Furthermore, their detection efficacy differs significantly when compared to similar mechanisms available on Desktop environments.

In the next paragraphs, the most popular browsers on Android and iOS are tested, and their efficacy in blocking phishing and malicious URLs is compared with popular Desktop browsers on Windows. Furthermore, statistics are created for all the malicious files which were downloaded during the tests (when visiting a malicious web site), revealing the disappointing efficacy of Antivirus engines, even against non-targeted malware.

5.2 Background

5.2.1 Phishing Attacks

The main browser defense against phishing/malware sites is based on blacklists, which are used by browsers to identify if a requested URL has been reported as malicious. The most popular blacklist is Google's Safe Browsing (Google 2013), which protects from both phishing

and malicious web sites. Safe Browsing is currently used by Chrome, Firefox and Apple Safari browsers. Internet Explorer uses SmartScreen, Microsoft's proprietary blacklist (Microsoft 2011). Other browsers use their own proprietary lists and/or aggregate data from third parties. For instance, Opera uses phishing blacklists from Netcraft (Netcraft 2014) and PhishTank (Phishtank 2014) and a malware blacklist from TRUSTe (Abrams et al. 2013).

A number of approaches has been proposed by the research community to protect users from phishing attacks, which vary from user awareness surveys to experiments of the effectiveness of current security mechanisms and proposals of novel ones. More specifically, the work in (Banu & Banu 2013), (Rosiello et al. 2007) and (RaniSahu & Dubey 2014) focuses on phishing with regards to its properties, characteristics, attack types, and available counter-measures. Authors in (RaniSahu & Dubey 2014), (Jansson & von Solms 2013) present a survey on user training methods, as well as their effectiveness against phishing attacks.

Literature has also focused on visual indicators that protect users from phishing. An overview of the warning indicators is presented in (Bian 2013). Also, (Darwish & Bataineh 2012) has surveyed users' interaction regarding security indicators in web browsers. A study on the effectiveness of browser security warnings was published in (Akhawe & Felt 2013), focusing on Chrome and Firefox browsers. A similar study in (Egelman & Schechter 2013) analyzed the impact on the users' decision based on the choice of background color in the warning and the text descriptions that were presented to them.

The authors in (Sheng et al. 2009) focused on the effectiveness of phishing blacklists, and more specifically on their update speed and coverage. They found that less than 20% of phishing sites were detected at the beginning of their test. Similarly, in (Kirda & Kruegel 2005) the authors proposed the use of 'Anti-Phish', an anti-phishing extension for Firefox. Zhang et al. (J. Zhang et al. 2011) used a text classifier, an image classifier, and a fusion algorithm in order to defend against known properties of phishing attacks.

The authors in (Comparatives 2012) analyzed 300 phishing URLs and measured the detection effectiveness of desktop browsers. Opera browser offered the highest level of protection by blocking 94.2% of the phishing sites. Mazher et al. (Mazher et al. 2013) tested the effectiveness of anti-phishing add-ons for Internet Explorer, Chrome, and Firefox, finding that Chrome outscored the other browsers. Finally, (Abrams et al. 2013) examined the time required for Firefox, Chrome, Opera, IE, and Safari to block a malicious site (from the creation of the malicious domain until the time it was blocked by the browsers).

The authors in (Vidas et al. 2013) investigated the viability of QRishing (i.e. QR-code-initiated phishing attacks). Similarly, (Xu & Zhu 2012) examined how notification customization may allow an installed Trojan application to launch phishing attacks or anonymously post spam messages. Work by (Mylonas, Tsalis, et al. 2013) revealed that security controls, which are typically found on desktop browsers, are not provided by their smartphone counterparts.

Finally, (Virvilis, Tsalis, et al. 2014) highlighted significant differences in the effectiveness of phishing blacklists of Android, iOS, and desktop browsers.

5.2.2 Malware attacks

Although the majority of browsers rely solely on blacklists to detect malicious URLs, on Windows Internet Explorer and Chrome perform further analysis to detect malicious downloads. Both browsers analyze the metadata of the downloaded file (i.e. digital signature, hash, file's popularity, etc.) to warn users about potential risks (Rajab et al. 2013), (Microsoft 2011), (Colvin 2011). Furthermore, non-browser specific models have been proposed for the detection of malicious domains through DNS monitoring (Antonakakis et al. 2011), while (Bilge et al. 2011) discusses large-scale, passive DNS analysis techniques to detect domains that are involved in malicious activity. A zero-day anti-malware solution is proposed in (Shahzad et al. 2013), which uses a combination of whitelists and blacklists. The authors discuss that such whitelists do not need signature updates and provide protection against sophisticated zero-day malware attacks by enforcing software restriction policies, which allow only legitimate executables, processes and applications to be executed.

In addition, a number of models have been suggested in the literature focusing on malware detection, namely: (a) the AMICO project (Vadrevu et al. 2013), which detects malware downloads in live web traffic using statistical analysis to identify characteristics of malware distribution campaigns, (b) the ZOZZLE (Curtsinger et al. 2011), which detects and prevents JavaScript malware from been deployed in the browser, and (c) the EFFORT system (Shin et al. 2012), which focuses on the detection of malware serving bots. Furthermore, multiple models have been proposed that rely on machine learning techniques: (a) for malware detection (Kolter & Maloof 2006), (Roberto Perdisci et al. 2008), (Antonakakis et al. 2011), (R Perdisci et al. 2008) and (b) for detection of drive-by downloads (Caballero et al. 2011), (Cova et al. 2010), (Lu et al. 2010), (Provos et al. 2007), (Mavrommatis & Monrose 2008), (H. Zhang et al. 2011). Finally, models focusing on malware and attack propagation models have also been proposed (Komninos et al. 2007), (Kammas et al. 2008).

The industry offers a large number of content filtering solutions, ranging from software-based solutions, to Cloud services and hardware appliances. Multiple software solutions exist, such as McAfee's Site Advisor and Symantec's Safe Web (Mcafee 2014), (Symantec 2014c). They are usually offered for free or at low cost. However, they require the installation of third-party software/browser extensions and are browser and platform dependent, with limited support for mobile platforms. Commercial content filtering appliances and Cloud Services (e.g. OpenDNS (Opendns 2014)) are popular in enterprises and are usually platform

and browser agnostic. However, their effectiveness is hard to measure due to the use of proprietary technologies. Also, their significant cost limits their use.

Lastly, a number of online services exists offering file analysis. One of the most popular is VirusTotal (VirusTotal 2014), which utilizes a large number of popular antivirus engines to analyze (suspicious) files (c.f. Table 31 in the Appendix for a list of AV engines). Users can upload and analyze their files, or query the service for files that have already been analyzed by searching their hash. VirusTotal also supports URL scans, querying a URL against a large number of blacklists (c.f. Table 32 in the Appendix).

Table 6 - Browser availability on tested platforms

	iOS 7.1.1	Android 4.0.4	Windows 7
Safari Mobile	X		
Chrome Mobile	X	X	
Opera Mini	X	X	
Browser†		X	
Firefox Mobile		X	
Opera Mobile		X	
Chrome			X
Firefox			X
Internet Explorer			X
Opera			X

5.3 Methodology

5.3.1 Test methodology

The experiments were conducted in June and July 2014 and focused on the most popular browsers on the desktop and smartphone platforms (c.f. Appendix, Tables 9-10), namely:

- **Desktop browsers**. Internet Explorer 11, Chrome v35, Firefox v29, and Opera v22, which were installed on a Windows 7 64-bit system.
- **Mobile browsers**. Safari Mobile (built-in on iOS 7.1.1), Chrome Mobile v35, Opera Mini v7.0, "Browser" or "Internet" (i.e. the default browser for Android 4.0.4), Firefox Mobile v30, and Opera Mobile v22, which were installed on an iPhone 5S (iOS v7.1.1) and a Sony Xperia Tipo (Android v4.0.4). Although some of the desktop browsers have mobile counterparts, their availability on the two smartphone platforms is heterogeneous, as shown in Table 6.

Our effort in the smartphone platform focused on iOS and Android, as these are the two most popular operating systems, comprising almost 90% of the global smartphone market share (Bradley 2013).

To evaluate the protection that is offered to users against rogue web sites, 1400 phishing and 1400 malicious URLs were accessed. The ones for which browsers raised alerts, were recorded. As technology that can be used to fully automate this process is not currently available, a security savvy user manually verified if a web page that had not been reported by the browser, was indeed a rogue or a benign web page. Since this was a cumbersome task, the architecture presented in Figure 3, was set up.

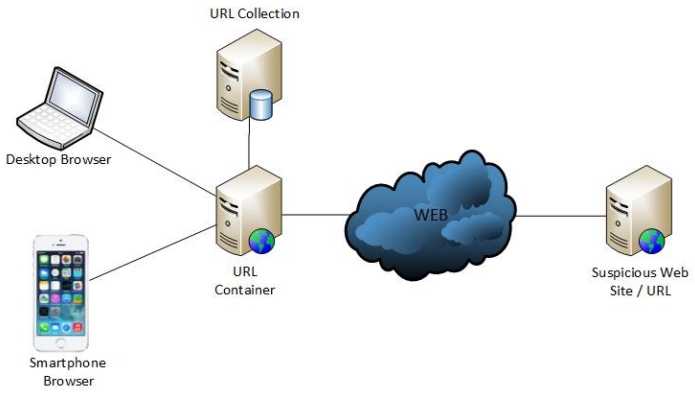

Figure 3 - Laboratory setup

The URL Collection included the URLs that were used in the evaluation. The URL Container parsed daily the URL Collection and selected the URLs that had been reported to Phishtank in the last 24 hours. Then, two HTML files were created, one for phishing URLs and one for malicious URLs, which were formatted as the Snippet below. Finally, a researcher used each browser installed on the test devices, to access each HTML file and collect the number of: a) blocked URLs, b) false negatives, and c) non-phishing/malicious URLs.

5.3.2 Phishing Tests

To evaluate the anti-phishing protection that is offered by the aforementioned web browsers, 1400 phishing URLs were collected from PhishTank (Phishtank 2014). PhishTank was selected as it is a popular online service that lists phishing URLs, which are verified by an active community. PhishTank publishes every day a list of verified phishing URLs, i.e. ones that have been confirmed as fraudulent and online. However, as the state of a phishing URL is dynamic, a confirmed URL might be cleaned or be taken down shortly after its submission.

```
<!DOCTYPE html>
<html>
<body>
<script>
      window.open("http://testurl1.com");
      window.open("http://testurl2.com");
      …
      window.open("http://testurln.com");
</script>
</body>
</html>
```

Snippet - HTML content

Although part of the experiments could have been automated (e.g. when the request returned an HTTP Error Code or the browser raised a warning), manual review was required in order to measure the false negatives (actual phishing URLs, which were not blocked by the browser). In specific, each URL that was not blocked by the browser was examined and classified it as (a) benign or not responsive/non accessible site (i.e. inactive phishing site) or (b) false negative (i.e. an active phishing site that was not blocked by the browser).

Thus, in the end of the experiment every URL in the phishing collection was manually classified as:

- ***a) Blacklisted***: The phishing URL was blocked by the web browser, i.e. the user received a warning indicating a phishing site.
- ***b) False Negative***: An active phishing URL that was manually verified as fraudulent, but was not blocked by the browser.
- ***c) Non-Phishing/Timeout/Error***: The phishing URL had been suspended/taken down/cleaned when it was accessed.

It should be noted that the data set included only verified phishing URLs (i.e. a human operator from PhishTank has manually verified them as fraudulent), as the main objective was to examine the efficacy of browsers' anti-phishing blacklists. Therefore, the false positive rate of all browsers, which is out of the scope of this work, was zero. Finally, the results are compared with our previous work in (Virvilis, Tsalis, et al. 2014).

5.3.3 Malware Tests

An online, well-known service that reports on a daily basis verified malware-hosting websites, offering strong community support (comparable to PhishTank), is not currently available. Therefore, to gather the malicious URL collection the open source Collective Intelligence Framework (CIF) (CIF 2014) was used. CIF allows the collection and analysis of malicious threat information from a large number of trusted sources (c.f. Table 25 in the Appendix), which is used for incident response and intrusion detection and mitigation.

Similarly to the anti-phishing experiment, our tests included manual browsing to 1400 URLs that hosted malicious software. Every URL was categorized as follows:

a) ***Blacklisted***: The browser blocked either the URL or the file that was downloaded (or issued a warning that the file could be potentially dangerous).

b) ***False Negative***: A URL that was not blocked by the browser and triggered the download of a potentially malicious file, without any further alert being raised by the browser.

c) ***Non-Malicious/Timeout/Error***: The URL had either been cleaned or suspended/taken down when it was accessed. This category also included the URLs that did not trigger a download.

Similarly, the dataset for this test included only verified malicious URLs as this work examines the efficacy of browser blacklists in blocking such attacks. As a result, the false positive rate of all browsers, which is out of the scope of this work, was zero.

5.4 Experimental Results

5.4.1 Protection against phishing sites

5.4.1.1 iOS browsers

Mobile Safari, which is the pre-installed iOS web browser, uses Google's Safe Browsing to provide anti-phishing protection. The evaluation revealed that the implementation of this anti-phishing control suffers from a significant design weakness, as the Safe Browsing blacklist is only updated when the iOS device is synchronized with iTunes. Considering that some iOS users may not synchronize their devices frequently, they may end up with an outdated blacklist. Furthermore, the list is updated only once per day. Thus, any phishing site that has been created in the meantime - even if it has been reported to Safe Browsing list - will not be blocked. As a result, iOS users receive considerably limited protection against phishing attacks.

In fact, during the experiments Safari Mobile did not block any phishing URL when this synchronization step was skipped. Therefore, the iOS test device was synced daily, right before starting our evaluation.

Chrome Mobile offers phishing protection since January 2014. However, this option is not enabled by default, but requires the user to enable the "Reduce Data Usage" option, which uses Google's servers as a proxy to fetch the requested URL. When this option is enabled, the contents of the web page are downloaded and compressed and the URL is checked against the Safe Browsing list. This feature is privacy intrusive - as all traffic is transferred through Google's servers - and does not work for SSL/TLS pages or in Incognito mode (private browsing). This browser has been excluded from the evaluation as: a) it is less likely that normal users (i.e. not security and savvy ones) would enable security controls, as smartphone users tend to be oblivious about their security and privacy (Mylonas, Kastania, et al. 2013) and (b) the control is not 'easily configurable' (Mylonas, Gritzalis, et al. 2013), i.e. the label of the control is not intuitive and confusing even for security savvy users. Finally, Opera Mini did not support phishing protection.

Table 7 - Phishing protection statistics on iOS

URL iOS Browser	Blacklisted	False negatives	Non-phishing
Safari Mobile	542	370	488
Chrome Mobile	N/A	N/A	N/A
Opera Mini	N/A	N/A	N/A

5.4.1.2 Android browsers

Android users also receive limited protection against phishing attacks, as the default Android browser (known as "Browser" or "Internet") does not offer phishing protection. The same holds true for Chrome Mobile and Opera Mini. It must be noted that these are the most popular browsers on Android, according to the number of downloads on Google Play (c.f. Table 24 in the Appendix).

On the other hand, Firefox Mobile and Opera Mobile offer anti-phishing protection. The results suggest that both browsers offer similar protection with their desktop counterparts. Specifically, Firefox and Opera on Windows blocked 86.7% and 77.9% of the phishing URLs and Firefox Mobile and Opera Mobile blocked 85.4% and 75.9%, respectively (c.f. Figure 8) Nevertheless, if one considers that: (a) not all users feel the need and/or are capable to install a third-party browser on their devices (Mylonas, Kastania, et al. 2013) and (b) the pre-installed

browser offers no anti-phishing protection, then a large number of Android users is not protected from phishing attacks.

Table 8 - Phishing protection statistics on Android

URL Android Browser	Blacklisted	False negatives	Non-phishing
Firefox Mobile	1196	48	156
Opera Mobile	1062	110	228
Chrome Mobile	N/A†	N/A	N/A
Opera Mini	N/A	N/A	N/A
Android Browser††	N/A	N/A	N/A

†† 'Browser' (or Internet) is the pre-installed Android browser

† N/A: Browser does not support anti-phishing mechanisms

5.4.1.3 Desktop browsers

The analysis revealed that all desktop browsers offered anti-phishing protection using either Safe Browsing list (Chrome and Firefox) or their own proprietary blacklists (Opera and Internet Explorer). The most phishing URLs were blocked by Chrome and Firefox. Although their results are similar - which is expected as they use the same blacklist - Chrome outperforms Firefox, as in our experiments it blocked roughly 5% more phishing sites and has a lower false negative rate.

During our experiments another issue was encountered with the synchronization of blacklists in Firefox, which was also raised by (Abrams et al. 2013). Specifically, each day Safe Browsing blacklist in Firefox was not updated, unless Firefox was executed for a few minutes before our evaluation, which resulted to a large number of false negatives. This stems from the way the Safe Browsing protocol updates its local database (Sobrier 2014). Interestingly, this problem did not appear in Chrome or any smartphone browsers that use Safe Browsing. To avoid this synchronization issue, Firefox was executed for at least 10 minutes prior testing, to allow the browser to update its blacklist.

Opera blocked roughly 10% less phishing sites than Firefox and had slightly more false negatives. Finally, Internet Explorer offered the lowest level of protection among the desktop browsers, having the smallest percentage of blocked URLs (less than 50%).

Table 9 - Phishing protection statistics on Windows

URL Browser	Black-listed	False negatives	Non-phishing
Firefox	1215	83	102
Chrome	1302	18	80
Opera	1090	118	192
Internet Explorer	678	138	584

5.4.1.4 Comparison with previous evaluation

Herein, the anti-phishing protection that the popular desktop and mobile browsers offer in Q2 2014, are compared with the results of our previous work in (Virvilis, Tsalis, et al. 2014), which was conducted in Q1 2014. Since the phishing 'ecosystem' is dynamic, i.e. phishing sites are short-lived with an average life expectancy of 23 hours (Abrams et al. 2013), our aim is to examine how this dynamic nature is reflected in the browsers' anti-phishing protection over this period of time.

The results are summarized in Figure 4 (c.f. Table 26 to Table 28 in the Appendix for detailed results). The browsers blocked fewer phishing URLs in Q2 with the exception of Firefox Mobile on Android. Safari Mobile's (iOS) detection dropped almost by half. This stresses again the problematic implementation of the Safe Browsing protocol on iOS. Furthermore, the analysis showed a small decrease in the performance of the desktop versions of Firefox and Opera, with respect to the blacklisted URLs and false negatives. Finally, Internet Explorer blacklisted less URLs and was prone to more false negatives, and Opera Mobile had more false negatives.

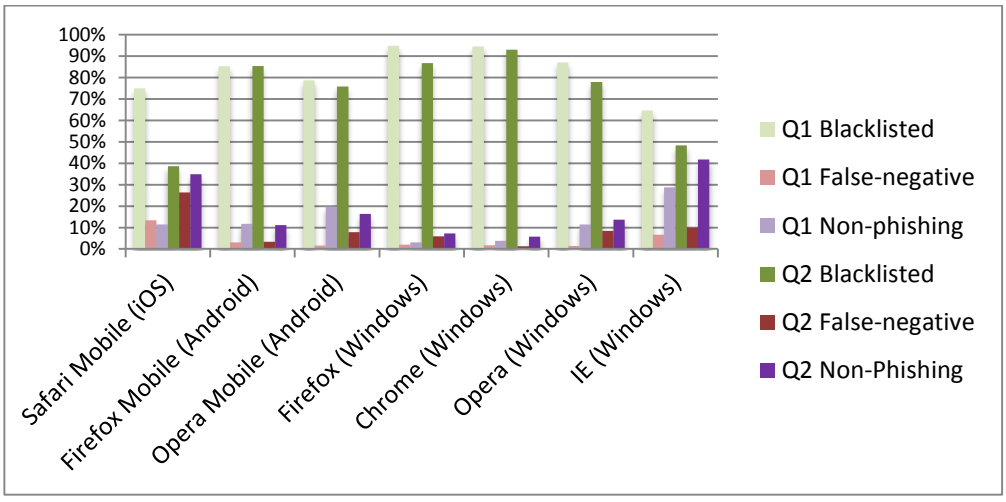

Figure 4 - Comparison of anti-phishing protection (Q1 2014 – Q2 2014)

5.4.2 Protection against malicious sites

5.4.2.1 iOS Browsers

The results revealed that none of the iOS browsers offered any protection against malicious sites, leaving their users exposed to this threat. In the case of Mobile Safari this was rather surprising, as the browser uses Safe Browsing to provide anti-phishing protection, but it does not provide detection of malicious sites. Opera Mini did not utilize any blacklist for the detection malicious sites and neither did Chrome Mobile (not enabled by default and excluded due to the shortcomings that were mentioned previously).

Table 10 - Malware protection statistics on iOS

URL iOS Browser	Blacklisted	False negatives	Non-malware
Safari Mobile (iOS)	N/A	N/A	N/A
Chrome Mobile	N/A	N/A	N/A
Opera Mini	N/A	N/A	N/A

N/A: Browser does not support anti-malware mechanisms

5.4.2.2 Android Browsers

The results suggest that Android users are also unprotected against malicious sites. This finding is very worrying if one considers the increasing number of attacks against Android and the exponential growth of Android malware (Kaspersky 2013a), (Zhou & Jiang 2012), (Zhou et al. 2012). Specifically, the pre-installed web browser ("Browser" or "Internet"), Chrome Mobile and Opera Mini offered no protection against malicious sites. As summarized in Table 11, only Firefox Mobile and Opera Mobile utilized malware blacklists. Nonetheless, our results suggest that the level of offered protection was very limited, as they blocked only 10-12% of the malicious URLs.

Table 11 - Malware protection statistics on Android

URL Android Browser	Blacklisted	False negatives	Non-malware
Firefox Mobile (Android)	139	641	620
Opera Mobile (Android)	166	683	551
Chrome Mobile	N/A	N/A	N/A
Opera Mini	N/A	N/A	N/A
Browser†	N/A	N/A	N/A

† 'Browser' (or Internet in newer versions) is the pre-installed browser on Android
N/A: Browser does not support anti-malware mechanisms

5.4.2.3 Desktop browsers

The results suggest that popular desktop browsers offer poor protection against malicious sites. Internet Explorer (IE) blocked the most malicious sites and was the least prone browser to false negatives, which confirms findings from similar research conducted by the industry (Abrams et al. 2013). Nevertheless, even though IE outperformed all the other browsers, it only blocked ~41% of the malicious URLs and had a 30% of false negatives. This highlights that the application reputation mechanism that is used by IE does offer an extra line of defense against malware, but is far from perfect.

Chrome had more false negatives than IE, even though it offers a similar mechanism against malicious downloads. Opera ranked third in terms of blocking malicious sites, but had 58% of false negatives, the highest in the experiments. Firefox offered the poorest protection, blocking only 5% of the malicious URLs and having a ~43% of false negatives.

During the experiments Firefox and Opera blocked malicious sites only by examining their URLs, without analyzing the downloaded files. Newer versions of Firefox now analyze the downloaded files using the same technology as Chrome (Mozilla 2014).

Table 12 - Malware protection statistics on Windows

URL Browser	Blacklisted	False negatives	Non-malware
Firefox	70	729	601
Chrome	280	552	568
Opera	180	816	404
Internet Explorer	573	420	407

5.5 Secure Proxy

The results uncovered three problems that must be addressed to protect users from rogue sites:

a. The limited effectiveness of blacklists against malicious sites and phishing sites.
b. The limited effectiveness of reputation based mechanisms (e.g. in Internet Explorer and Chrome) to block malicious downloads.
c. The unavailability of the relevant security controls in popular mobile browsers.

In this context, *Secure Proxy* is proposed and implemented, as a proof-of-concept security mechanism, which proves the efficacy of aggregating multiple data sources in the detection of rogue sites. Currently, and due to the heterogeneity and restrictions of smartphones and their security models, a different security control might be incompatible (e.g. due to the

limitations enforced by iOS) or infeasible to be implemented due to resource restrictions (e.g. on older Android smartphones). The proposed proxy is browser and platform agnostic (i.e. does not require the installation of third party software) and can protect the user, regardless of the browser she is using. Finally, and in contrast with the commercial and closed source content filtering solutions, we are proposing an open architecture which can be built with a fraction of the cost.

5.5.1 Architecture

A secure forward HTTP proxy has been implemented, which uses VirusTotal's public API to analyze the requested URLs and downloaded files. VirusTotal was selected due to (a) its popularity, (b) the number of AV engines and blacklist providers that it offers, and (c) the availability of free API. Nevertheless, any other similar service could be used.

The proxy queries VirusTotal for each requested URL to identify if the URL is blacklisted by any of the blacklist providers. If the URL is blacklisted, the request is blocked and a warning message is returned to the user. Otherwise, the proxy returns the HTTP response (i.e. page contents) to the browser. If a download is triggered, the proxy calculates the SHA-256 hash of the file and queries VirusTotal. Once more, if the hash is known and is reported as malicious by any AV vendor, then the download will be blocked and a warning will be raised. If the hash is unknown (i.e. the file has not been analyzed beforehand), then the proxy uploads the contents of the file for analysis and allows the user to download the file if it not flagged as malicious by the AV engines. These steps are summarized in Figure 5.

5.5.2 Secure proxy evaluation

The evaluation of *Secure Proxy* focused on the detection of malicious sites, as the majority of browsers provide only weak protection (or hardly any) against them. The proxy was also tested with the complete collection of phishing URLs from PhishTank. However, this test was only a verification of the correct operation of *Secure Proxy*, as PhishTank is one of the anti-phishing providers that is used by VirusTotal.

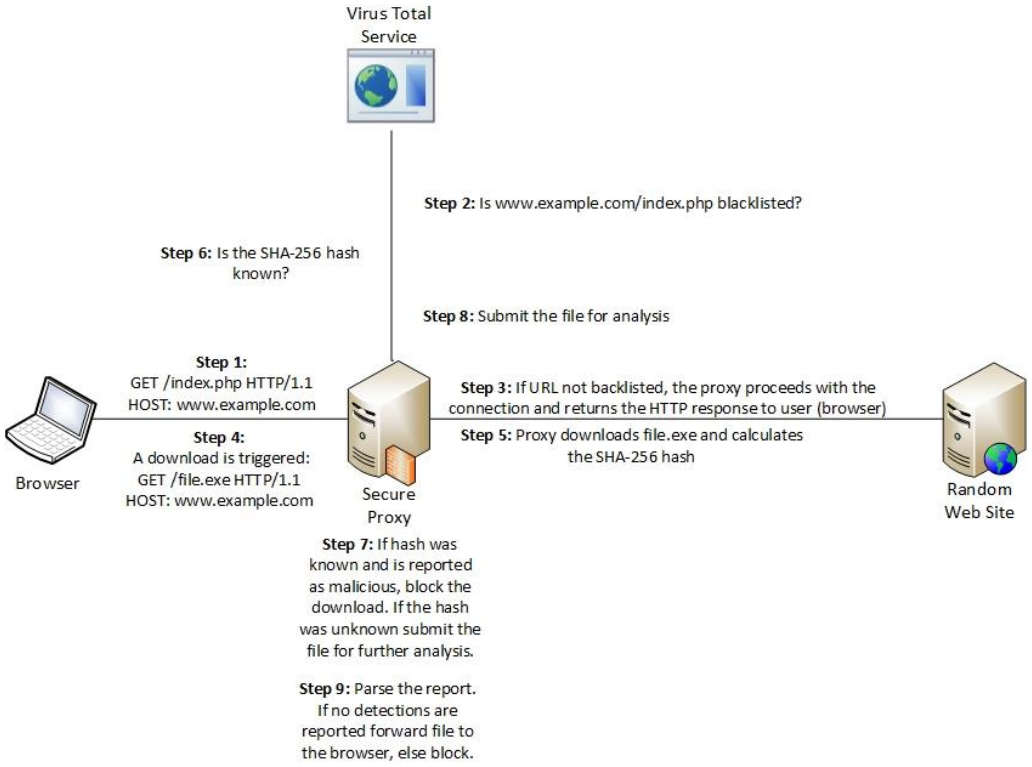

Figure 5 - Proposed architecture

To evaluate *Secure Proxy* the same list of malicious URLs that the browsers were tested against on a daily basis was accessed, and the requests were redirected through the *Security Proxy*. A script that simulated the web requests was used instead of a browser, to make sure that no browser specific countermeasure would interfere with the results, as well as to automate this process. The number of blocked URLs were collected and compared with the results of the browser that achieved the highest blocking rate in our evaluation. *Secure Proxy* was configured to block downloaded files (either based on the hash or the actual file analysis), when the number of detections reported by VirusTotal were at least one. This parameter is configurable and it can be configured to block a request with a different blocking threshold, according to the existing security policy or risk appetite.

Statistics were also collected regarding: (a) the number of URLs that were blocked due to URL-only analysis, (b) the number of downloads that were blocked due to hash analysis, and (c) the number of downloads that were uploaded and blocked by VirusTotal.

Table 13 - Percentage of blacklisted malicious sites

Secure Proxy vs. Internet Explorer (n=1400)

Secure proxy	Best browser (Internet Explorer)
53.2%	40.9%

5.5.3 URL-only and hash-based analysis

As summarized in Table 13, the use of multiple blacklists enabled *Secure Proxy* to block almost half of the malicious URLs - thus outperforming Internet Explorer by 12.3% - which was the browser that blocked the most malicious URLs. This finding highlights that while the aggregation of multiple blacklists provides higher protection than any individual browser, it still fails to detect almost half of the malicious URLs (i.e. 46.8% false negatives).

Browsing to these URLs triggered the download of 460 unique files (based on their SHA-256 hash), all of which were PE executables. *Secure Proxy* downloaded these files and queried VirusTotal for their hashes. The results revealed that 57.3% of the submitted hashes (i.e. 264 out of 460) were unknown to VirusTotal, meaning that the files had not been submitted for analysis beforehand. The detection rate of the rest of the files is summarized in Figure 6 (for detailed results see Table 15 in the Appendix).

The number of AV engines that are available during the analysis of a file in VirusTotal ranges between 49-54 antivirus engines. Figure 6 summarizes the results based on the detection ratio for each file. This ratio was calculated with z/n, where z represents the number of antivirus engines that flagged the file (hash) as malicious and n is the number of antivirus engines that were used. The results indicated that the detection rate of approximately half of the malicious files was in the range of 6-38% of the antivirus engines. Moreover, only the 27.4% of the malicious files were detected by the majority of the antivirus engines.

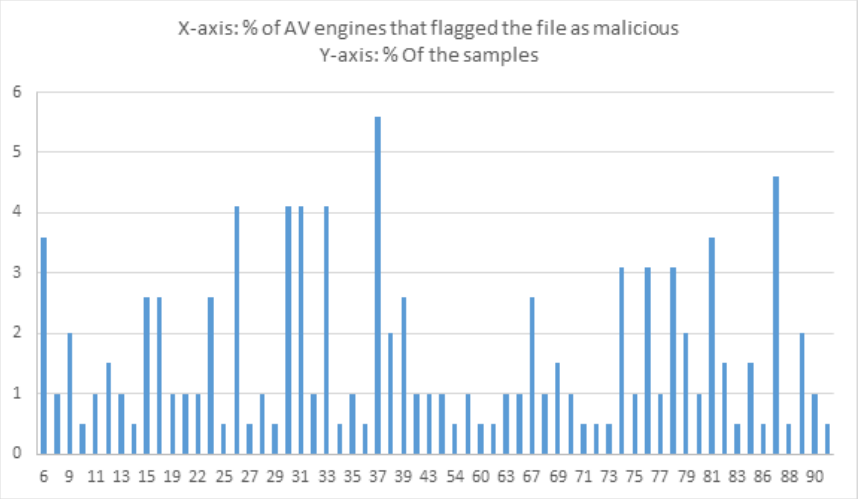

Figure 6 - Detection percentage of hashes
(X-axis lists the percentage of AV engines that flagged the file as malicious,
Y-axis lists the percentage of the samples).

5.5.4 File-based analysis

During the experiments, *Secure Proxy* uploaded 264 (57.3%) of the downloaded files to VirusTotal for analysis, as their hashes were unknown. All of them were reported as malicious and the majority of them (~71.0%) were detected by 46-50% of the antivirus engines (c.f. Figure 7, and Table 30 in the Appendix). This suggests that with the absence of file-based analysis from *Secure Proxy*, there is ~50% likelihood that malicious files will not be detected from the user's AV (assuming that the user is using only one AV program).

Finally, it should be highlighted that the abovementioned detection rates refer to desktop antivirus engines. Mobile antivirus applications are more constrained due to the sandboxed environment of mobile platforms and also lack the advanced features that are available at desktop antivirus engines (i.e. advanced heuristic analysis).

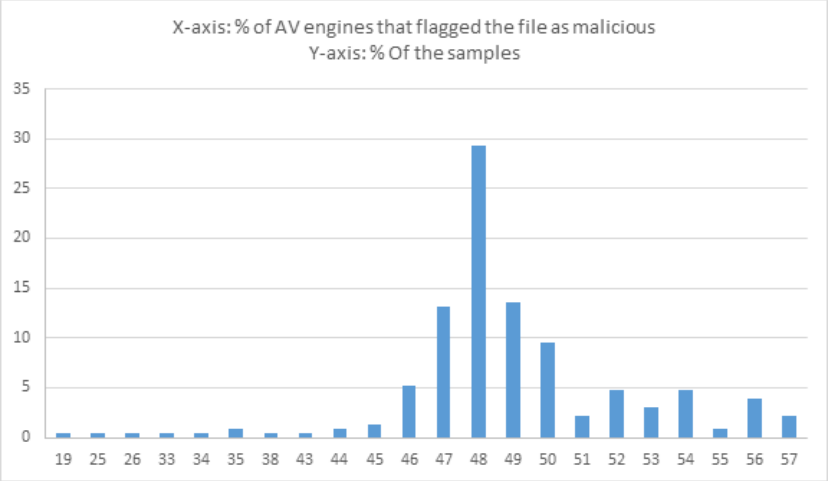

Figure 7 - Detection percentage of executables
(X-axis lists the percentage of AV engines that flagged the file as malicious,
Y-axis lists the percentage of the samples).

5.5.5 Performance evaluation

As mentioned beforehand, *Secure Proxy* performs three types of queries to VirusTotal, which incur delays: (a) URL Query to identify if the URL is reported as rogue by any of the blacklist providers, (b) Hash Query to identify if the hash of the file is known to be malicious, and (c) File Query in which the actual file is uploaded to VirusTotal for analysis.

The analysis revealed that for URL Queries the proxy received a response from VirusTotal and allowed or blocked the request on average in 648ms. The average Hash Query time for each triggered download was 516ms. The longest delay occurs when the hash is unknown as the file needs to be uploaded for further analysis (File Query). The delay depends on various parameters, e.g. the size of the file, the network speed, and the load on the VirusTotal service - with the latter often being the most time consuming parameter. In the evaluation, the average size of the collected malicious executables was 848KB and our Internet connection was a 20Mbit (2048Kbit upload) ADSL line. On average, the file queries were completed in under 41sec, including the time required to upload a file and get the detection report.

5.6 Discussion

5.6.1 Limitations

The corpus was limited to 2800 rogue URLs (1400 phishing and 1400 malicious URLs), due to the significant manual effort that was required to test different browsers on Windows, iOS and Android. This introduced a potential bias in the results, which describe the level of protection in the period that the tests took place, namely June-July 2014. However,

even though the results are not generalizable, it is our belief that they provide adequate indications about the level of protection offered to the users for two reasons: a) the findings for desktop browsers are in accordance with the results in (Abrams et al. 2013) and b) the results of this work regarding the effectiveness of anti-phishing protection are similar to our previous evaluation of a much larger data set of 5651 URLs (Virvilis, Tsalis, et al. 2014).

This work focuses only on the popular desktop browsers (Windows) and their smartphone counterparts that are available on iOS and Android. While there are other browsers that were not examined, such as Safari on Mac OS X and Internet Explorer Mobile on Windows Phone, the current results are considered as representative. This holds true as Windows is the most popular operating system for desktops and laptops, as well as Android and iOS users constitute the 94% of the smartphone users (78.4% and 15.6% respectively) (Gartner 2013). In addition, as iPads and Android tablets use a similar operating system (iOS, Android), and in most cases the exact same browser versions, the findings are considered to reflect the protection that is offered on a larger user base.

In this work *Secure Proxy* was proposed, as a countermeasure against rogue sites. It is worth noting that the implementation is a proof-of-concept one, highlighting the benefits that the aggregation of multiple blacklists and AV engines offers against rogue websites. The evaluation regards issues such as privacy or performance as out of scope of this work. However, as discussed in section 5.6.2, both issues can be addressed in a real-world implementation.

The implementation of *Secure Proxy* is based on the public version of VirusTotal's API, which introduces limitations. Firstly, VirusTotal is a service which was not designed to support semi-real time queries, as the ones used by the proposed control. A dedicated service optimized for such use, such as CloudAV (Oberheide et al. 2008), might achieve better performance and as it can be hosted locally, it avoids privacy concerns. Also, *Secure Proxy* - similarly to VirusTotal - does not weight differently the responses from the various antivirus engines or URL blacklists. It can be extended to assign different weights to these responses according to organization's security policy.

Finally, the results are affected by the dynamic nature of the web ecosystem. This is due to the dynamic nature of the threats and the new evasion techniques that attackers create. This is reflected on the comparison of the anti-phishing protection that is offered by the examined browsers in Q1 and Q2 (2014). Moreover, browsers add to the complexity of the evaluation due to their frequent updates, which might include new security controls (e.g. analysis of downloaded files is now supported in newer versions of Firefox).

5.6.2 Protection of desktop and mobile browsers

Overall, the results revealed that desktop browsers performed better in comparison to their smartphone counterparts, both against phishing and malicious sites. This is a worrisome finding if we consider the proliferation of smartphones, as well as the increased web browsing with these devices. One could argue that this is expected, as smartphones lack the processing capabilities of desktops and laptops. Nonetheless, this is only partly true today, as most smartphones have similar resources as a 3-4 year old laptop (e.g. dual core CPU, 1-2 GB or RAM, etc.). In addition, work by (Mylonas, Gritzalis, et al. 2013) has shown that the unavailability of important security controls - such as blacklists for phishing and malicious sites in which this paper focuses - does not stem from the (API) restrictions that are imposed from the smartphones operating system (i.e. sandbox profile). The reason that this happens is still unclear; however, it falls out of the scope of this work.

More specifically, the results revealed that only a subset of the mobile browsers offer anti-phishing protection and thus, their users are not protected from such attacks. This is particularly true for Android users, where the pre-installed browser does not offer anti-phishing protection. On iOS, the pre-installed browser offers anti-phishing protection, but its effectiveness is questionable (c.f. Section 5.4.2.1). On the contrary, all desktop browsers provided anti-phishing protection, even though their effectiveness was significantly different and blacklist synchronization issues were identified for Firefox on Windows. Figure 8 summarizes the results of the anti-phishing experiments.

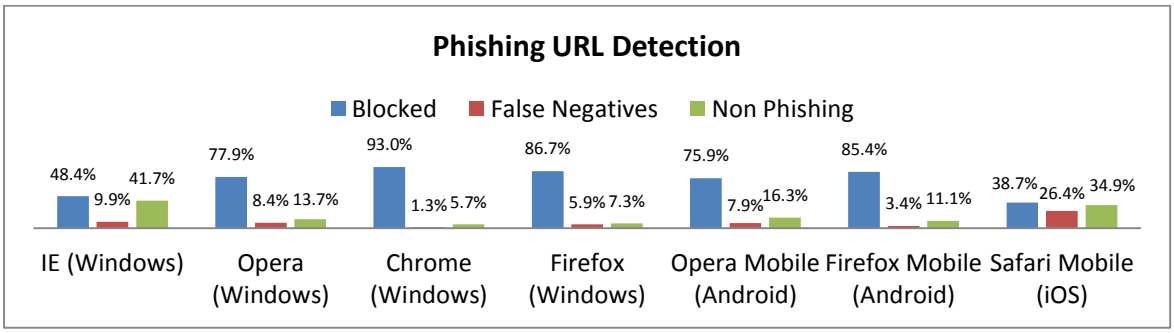

Figure 8 - Phishing URL detection percentage

Contrary to anti-phishing protection, the results suggest that both desktop and mobile browsers offer very limited protection against malicious URLs (Figure 9). While Internet Explorer and Chrome, which used application reputation mechanisms, blocked more malicious URLs/resources than other browsers, their detection rate was still low. Moreover, only a subset of the browsers on iOS and Android offer any protection against malicious URLs - the browsers that did not block phishing URLs also did not block malicious URLs.

Interestingly, Safari Mobile did not block malicious URLs, even though it uses Safe Browsing to offer anti-phishing protection. Apple may have assumed that using such a blacklist would not increase the level of protection for iOS users, based on the fact that iOS devices only execute code signed by Apple. This assumption seems flawed as: (a) a significant percentage of users jailbreak their iOS devices, thus the device can also execute unsigned code (Love 2013), (b) if iOS users do not receive any warning when visiting a malicious site, they can unwillingly put other users at risk by forwarding/sharing the URL, and (c) files that are downloaded on an iPhone might be synchronized to a computer, resulting to its infection.

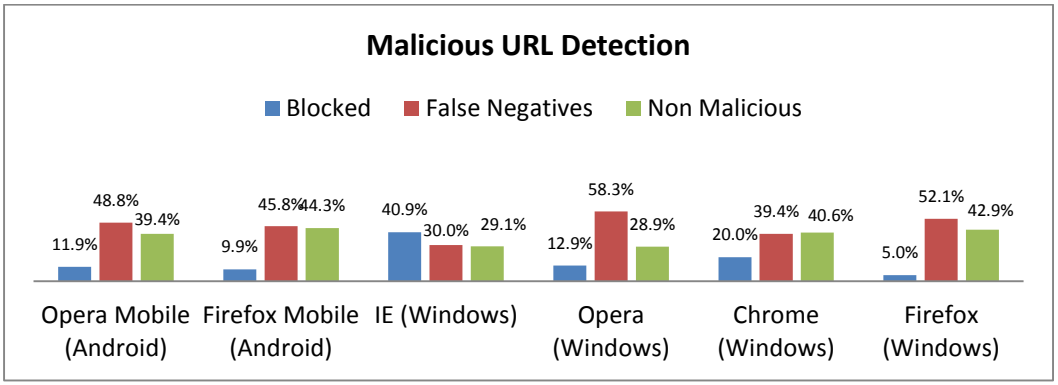

Figure 9 - Malicious URL detection percentage

5.6.3 Proposed countermeasure

To raise the bar of protection that is offered to users against rogue sites *Secure Proxy* was proposed, implemented as a proof of concept and evaluated. *Secure Proxy* is a HTTP forward proxy that uses multiple anti-phishing and anti-malware blacklists. Our work suggests that such browser agnostic architectures are currently the only solution for protecting normal (i.e. not security savvy) smartphone users, as they are less likely to install third-party browsers or security products on their devices (Mylonas, Kastania, et al. 2013).

The evaluation showed that *Secure Proxy* blocks more rogue URLs and is less prone to false negatives than any individual browser. Our work focused on reducing false negatives, i.e. active phishing or malicious URLs that were not blocked by the browsers, as they could result in a successful attack. *Secure Proxy* however, may be prone to false positives as some sites may have erroneously been added to a blacklist without proper verification. In this case, a URL might be blocked until it is removed from all blacklists causing a temporal nuisance to users. Nonetheless, popular blacklists (e.g. Safe Browsing) allow site administrators to request their site's removal from a blacklist when it has been cleaned, which would reduce the inaccessibility time. Furthermore, *Secure Proxy* can be configured, based on the user's or organization's risk appetite, to block a URL if the number of blacklists that identify it as

malicious exceeds a threshold, thus reducing potential false positives. However, specifying this threshold falls outside the scope this work.

Similarly to Internet Explorer's and Chrome's metadata analysis of the downloaded files, *Secure Proxy* offers hash-based analysis with the aggregation of multiple AV engines. In specific, it queries the file's hash on VirusTotal and blocks its download if it is reported as malicious by at least one AV engine – similarly to URL-only analysis this threshold is configurable. The results prove that hash-based analysis adds an extra layer of protection against malicious sites and - as URL-only analysis - does not introduce significant delays.

The benefit of online queries is that the URL or hash will always be checked against an up-to-date blacklist, avoiding any blacklist synchronization issues. In addition, using a browser agnostic countermeasure, such as *Secure Proxy*, enables the end user to use a browser that does not offer any built-in countermeasures and still be protected. Finally, it also allows devices with limited resources (e.g. smartphones) to avoid resource intensive operations and thus reduces energy consumption.

The inherent drawback of any online, centralized architecture is user privacy. Each visited URL is submitted to a third party for analysis, thus exposing the users' browsing history/profiles. Even though this falls outside the scope of this work, it can be mitigated by maintaining a local blacklist/whitelist, thus avoiding the need for a central architecture (e.g. as in our case, a proxy server) - similarly to the way Safe Browsing protocol works. The browsers that implement the protocol, keep a local database of reported URLs, which is updated frequently while the browser is running. As a result, all lookups use the local database, thus avoiding unnecessary delays and privacy issues. Based on the fact that VirusTotal is owned by Google, it seems fairly easy to include the aggregated results from all blacklist providers to a single list (e.g. imported into Safe Browsing list). Still, this will require the browser vendors to implement/adopt the Safe Browsing protocol and make sure that they avoid any synchronization issues. Another potential solution could be to host a service similar to VirusTotal locally e.g. CloudAV (Oberheide et al. 2008), which would also address the privacy issues.

The tests have also revealed the significant benefit of aggregating multiple AV engines for the detection of malicious files. *Secure Proxy* uploads for further analysis all downloaded files for which a hash-only query returned no results. This analysis introduces delays (41 sec on average in our experiments), which may not be acceptable for some users. However, *Secure Proxy* can be configured to upload these files according to a policy, e.g. by default deny access to these files, move the files in a sandbox or ask the user to decide whether the files will be submitted for further analysis. The last option however assumes users' security awareness, which might not be the case for all users, as they are known to click through security messages

(Akhawe & Felt 2013), (Egelman & Schechter 2013), (Mylonas, Kastania, et al. 2013), (Mylonas, Gritzalis, et al. 2013).

Similarly to other instances in the security domain, there is a tradeoff between security and usability. A combination of a whitelist and reputation based system will further limit the number of files that have to be submitted for analysis - as only files that are not included in the whitelist and are not blocked by the reputation system have to be analyzed. Nevertheless, the benefits of such detailed analysis are significant. This holds true as the results suggest that even if an AV engine is in use, it will fail to detect all malicious files (in our experiments on average an AV detected only half of the malicious files). On the contrary, *Secure Proxy* offered better anti-malware protection due to the aggregation of multiple AV engines and blocked all the malicious files in the corpus.

Finally, it should be noted that the focus of *Secure Proxy* is to allow mobile users to achieve a comparable level of protection with the one offered to desktop users, regardless of the limitations imposed by the mobile platform or browsers. However, as this countermeasure is based on the aggregation of Antivirus engines, the shortcomings of which have been raised both in this chapter as well as in the bibliography (Cole 2012), (Schneier 2012), it should not be considered effective against targeted attacks. For the latter, different detection techniques such as the ones presented in the next chapter, have to be investigated.

5.7 Conclusion

It is evident that even basic security mechanisms which have been available on desktop browsers for years, are not present (or are significantly less effective) on mobile platforms. Furthermore, there is a significant gap in the number of security solutions available for desktops and mobile platforms, as well as major differences in robustness, effectiveness and list of features.

On mobile platforms, the security products are severely affected by the limitations imposed by the sandbox environment. In contrast, on desktop environments they are able to interact with the operating system at a low level (i.e. kernel space) and thus, have a much higher flexibility. Furthermore, they can perform resource intensive tasks (e.g. execution of the suspect file(s) in an emulated environment), that are currently not possible on mobile platforms due to architecture and performance constrains.

Hence, even if smartphone users are technically capable and willing (or forced by policy) to install third party browsers or security solutions on their devices, they will still be significantly less protected than desktop users.

These findings, along with the continuously increasing popularity of smartphone devices in sensitive environments, has turned them into a high value target for attackers. Thus, it is

logical to assume that smartphones will be one of the prime APT targets in the years to come, and consequently any robust defense planning should take these devices into consideration.

(This page is intentionally left blank)

Chapter 6: Redefining the Security Architecture

6.1 Introduction

The limited effectiveness of the existing cyber security solutions against sophisticated attackers has been criticized harshly by the security community (Schneier 2012), (Bejtlich 2013). Taking into account the severe security incidents that have been published in recent years, it is more than evident that existing solutions - even the ones considered to be the state of the art - are unable to address sophisticated attacks (Cole 2012), (Bejtlich 2013). As a result, a radical change in the way we defend our assets is required.

The use of deception, is a very promising approach for attack detection, regardless of the attacker's skills and capabilities. This chapter begins with a presentation of both existing and novel deception techniques, and their potential uses for attack detection. Furthermore, it proposes a robust APT detection model, a proof of concept implementation of it, and its evaluation against two realistic scenarios. The model combines anomaly detection with multiple deception-based attack indicators and correlates both current and historical events over a wide period of time. It achieves high detection efficacy and a very low false positive rate, even against sophisticated attacks designed to evade traditional security solutions.

6.2 Unconventional defenses: The use of deception

Prevention of sophisticated attacks by focusing solely on technical countermeasures is unrealistic, due to the technical and resource superiority of the attackers. Defenders will always have to find and fix all vulnerabilities and attack paths that could enable the attackers to compromise the infrastructure, while the latter only need to exploit one weakness to succeed. The use of zero-day exploits and robust malware makes the defender's task very challenging, even for large organizations with a substantial cyber security budget. Furthermore as already mentioned there is a global shortage of skilled security professionals (Libicki et al. 2014), and thus the assumption that every organization will be able to establish a skilled cyber security team, is not realistic.

Use of deception techniques can be invaluable for the detection of malicious actions, regardless of the level of sophistication. Examples of common deception techniques include honeypots (physical or virtual systems, simulating real production systems) and honey tokens (fake files or records which are likely to be accessed by the attackers).

After unauthorized access has been achieved, the attackers will need to locate and exfiltrate the information they are interested in. Doing so, is a complex and time consuming task (especially for a large infrastructure), as the attackers will have to hop between network

segments, compromise domain accounts and gain access to systems (e.g. file servers) which might contain the targeted information. Similar actions should be expected for database servers and all other systems than could potentially host what the attackers are after. As a result, the introduction of deception-based countermeasures in the form of honeypots, honey files, honey tokens and fake accounts (e.g. honey accounts) can significantly raise the possibility of detection, as the attackers are likely to access these resources while trying to find the information they are interested in. For example, this could be the inclusion of honey files (e.g. word documents) on a file server, as the attackers are very likely to access them among all other files on the server. This statement is also supported from our evaluation results (see section 6.4.4.2).

The APT attack life-cycle (Figure 10) is discussed in the following paragraphs. For each stage of the life-cycle, deception techniques that can increase the possibility of detection are proposed. It consists of the following stages: a. information gathering, b. attack preparation, c. initial exploitation, d. internal access, e. further exploitation (e.g. escalation of privileges, network reconnaissance and identification of targeted information) and finally f. exfiltration of data. These stages can be grouped into two generic phases: attack planning (steps a, b) and attack execution (steps c to f).

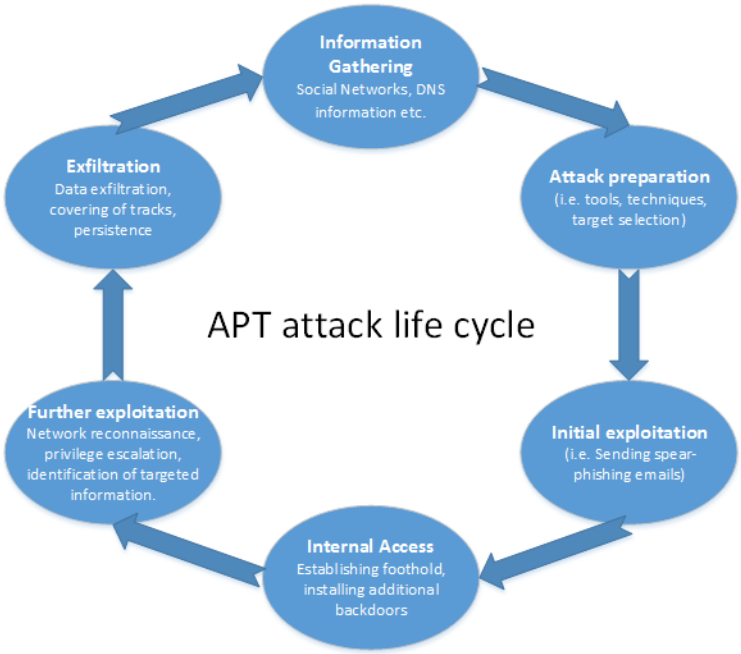

Figure 10 - APT attack life cycle

6.2.1 Phase 1: Attack planning

This phase focuses on actions that the attackers usually perform while planning their attack (e.g. information gathering). The attackers will try to gather as much information as

possible for their target, from multiple sources such as: social networks, bulletin boards, published vacancies, search engine queries (e.g. using Google dorks to find email addresses or domains registered to the organization), DNS zone transfers/brute-forcing, server/service fingerprinting, WHOIS lookups etc.

It is critical for the defenders to be able to detect any information gathering attempt against their infrastructure as this is a strong indication of an eminent attack. Early detection will allow them to raise the readiness level and be better prepared (e.g. reminding the employees that opening email attachments from unknown sources is dangerous, that they should report immediately any suspicious behavior, etc.). Sections 6.2.1.1 to 6.2.1.3 propose deception-based attack indicators that could be used for the detection of information gathering attempts.

6.2.1.1 DNS honey tokens

As the attackers are expected to try to identify the Internet-facing systems/services belonging to their target, a straight-forward approach that the defenders could try is the deployment of honeypots across the unused public IP range of the organization. Based on the fact that these honeypots will not be publicly listed, a connection attempt towards them could be due to one of the following reasons: (a) human error (mistyping an IP address/domain name), (b) mass scanning of the IP address space used either for legitimate purposes (i.e. research) or due to malicious actions (e.g. automated attacks, worms) or (c) an information gathering attempt trying to identify all publicly accessible systems and services of the organization.

However, due to the second reason, honeypots are expected to generate a substantial amount of noise (alerts) (Gebauer 2012). Furthermore, it is difficult to differentiate between an automated, non-targeted interaction with a honeypot and a targeted one. As a result, Internet facing honeypots are not the preferred option for the detection of information gathering attempts.

Another, simpler to implement technique with a significantly limited number of false positives, is the insertion of fake DNS records (a type of honey token) to the authoritative DNS server(s) of the monitored domain.

Attackers as part of their information-gathering process, are likely to attempt a DNS zone transfer (Edge et al. 2010) or to “brute force” the target’s DNS servers for common subdomain names, trying to identify interesting resources. By creating a small number of fake DNS records on the authoritative DNS servers of the organization and monitoring for requests for those records, defenders can receive an early warning of DNS-related information-gathering attempts against their infrastructure.

6.2.1.2 Web server honey tokens

The public web servers of an organization are another fruitful source of information for attackers. Three kinds of honey tokens could be used for the detection of malicious web-site visitors:

- Addition of fake entries in *robots.txt* file
- Use of invisible links in web pages
- Inclusion of honey-token(s) in HTML comments

robots.txt file (Hendler & Berners-Lee 2010) is a simple text file located in the root folder of the web server, which legitimate bots (e.g. Google bot) parse in order to identify which resources on the web server should not be indexed. The file is one of the first places that attackers (and automated web vulnerability scanning tools) look for potentially sensitive resources. By including non-existing but potentially interesting resources such as "/admin" or "/login" in the *robots.txt* file and monitoring for access requests to them, administrators could be alerted for visitors with malicious intents.

The inclusion of invisible links (e.g. links with the same color as the background color of the web page) pointing to non-existing resources (but potentially interesting from the attacker's perspective i.e. *http://www.example.com/secret*), can serve a similar purpose. Although these links will be invisible to legitimate visitors, they will be accessed by web crawling tools that attackers are likely to use.

Another deception mechanism particularly useful for web sites that support authentication, is the inclusion of honey tokens in HTML comments (e.g. fake accounts). Legitimate users have no need to review the source code of a web page, however attackers frequently do, trying to identify vulnerabilities. The inclusion of an HTML comment similar to the following in a login page, is very likely to tempt the attacker to use it:

<!--test account: admin, pass: passworD123. Remove before production!-->

A login attempt with these credentials, is a clear indication of malicious activity with a zero false positive rate.

6.2.1.3 Social network avatars

Social networks are an invaluable source of information for attackers.

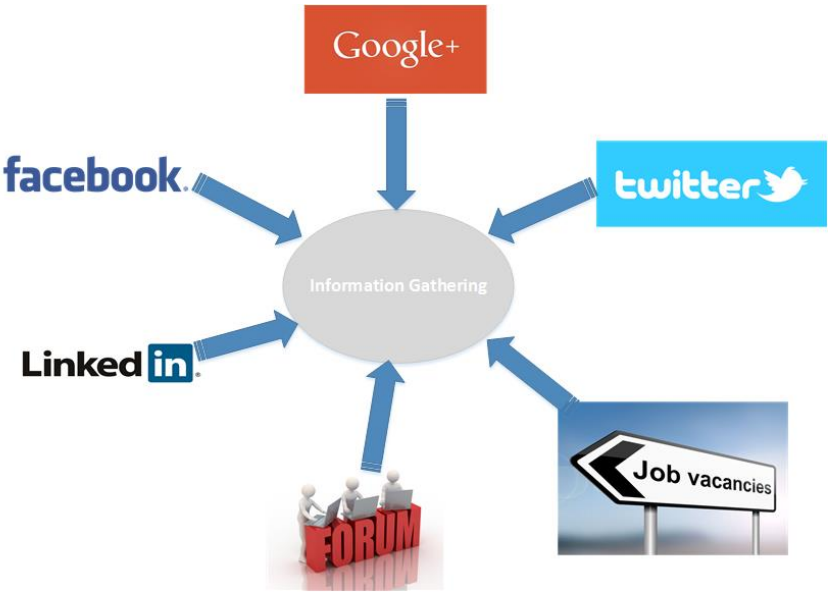

Figure 11 - Common sources of information gathering

A proposed solution for the detection of information gathering attempts on social networks is the creation of avatars (fake personas). The avatars should appear as realistic as possible, having connections with people from both inside and outside the organization. In addition, these avatars could have active - but limited and very closely monitored - accounts in the organization (e.g. active directory accounts), as well as valid email addresses. Interaction with the avatars should be regularly monitored (friend requests, private messages, emails, etc.), as it is a potential indicator of malice.

Of course, there is always the possibility of benign interaction e.g. individuals interested in applying for a position in the organization may contact one of the avatars, trying to get more information related to the position or the company and thus creating a false alert. However, this would be a rare case.

6.2.2 Phase 2: Attack execution

The second phase of the APT life-cycle is the attack execution. The attackers after having gained foothold on the network (e.g. by exploiting a client side vulnerability), will usually proceed with the following steps: Firstly, they will compromise additional systems and use them as alternative entry points into the network in case the initial one(s) are detected and quarantined. Following, they will move laterally, try to escalate their privileges and will explore the internal network looking for systems that may host the information they are seeking, or that can help them access it. In order to address this attack phase, the deception-based attack indicators in sections 6.2.2.1 and 6.2.2.2 could be implemented.

6.2.2.1 Network layer deception

As previously mentioned, in a medium to large organization in which hundreds or even thousands of systems are active, identifying where the targeted information is located is not a trivial task. Use of darknets and honeynets can be invaluable in detecting malicious actions, as attackers are very likely to access them among other network segments/systems, and thus raise alert(s).

A darknet also known as a "black hole" or "Internet sink" is a portion of routed but unallocated IP space, meaning that no workstations/servers or other network devices are located in that segment. Access to such regions of the network may occur by mistake (e.g. a user mistyping an IP address), however multiple connection attempts should be considered suspicious, as this is an indication of network mapping activity.

Honeynets (L. Spitzner 2003) are used for monitoring larger and/or more diverse networks in which one honeypot may not be sufficient. Defenders can use honeynets to create multiple fake systems in the same IP range(s) as legitimate workstations/servers. An attacker who gains access to a specific network segment is very likely to access these fake systems among with the real ones. Interaction with such systems should be very closely monitored as it is a strong indication of an active attack.

6.2.2.2 Application layer deception

The same techniques used for detecting malicious activity on external web servers can also be used for protecting internal ones. Furthermore, additional application specific attack indicators can be implemented, to further increase coverage:

Database server honey tokens

Use of honey tokens (also known as honey records) on databases (e.g. fake information that is likely to lure the attacker to access it) can be used to highlight malicious activity. For example, a number of fake patient records can be introduced in a hospital's patient database. Similarly, fake credit card records, passwords or any other kind of information that could be considered valuable for the attackers could be used. Attempts to access those records should be considered highly suspicious, as there is no legitimate need for users to access them. This countermeasure is also very effective against insiders even if they are aware of the use of deception techniques, as long as they do not know which specific records are the honey tokens (Virvilis & Serrano 2014).

Honey files

In (Bowen et al. 2009) and (Voris et al. 2013) a number of strategies for creating decoys (honey files) have been proposed, focusing either on the generation of perfectly believable decoys or the modification of legitimate files to include alerting functionality. Although the practical use of perfectly believable decoys has been challenged, use of legitimate files is likely to create multiple false positives.

To overcome these challenges decoy files with potentially interesting file names (e.g. *passwords.docx, new_investments.pdf,* etc.) could be created. The contents of the files should be fake (i.e. no value for the attacker), but at the same time realistic enough, especially if an action is expected from the attacker's side. For example, the contents for *passwords.docx* decoy file could be a list of fake usernames and passwords. Any authentication attempts from these accounts should raise an immediate alert.

Honey files should be spread across the file servers of the organization and/or even workstations, however the latter will increase the number of false positive alerts (Ben Salem & Stolfo 2011). Access to these files can be monitored by:

- Enabling file system auditing (Melber 2013), which is the method used by the proposed model.
- Inclusion of code which when executed will report back to a monitoring server. This can be achieved by using JavaScript for PDF files or embedding remote images in Word documents, which the application will try to access when the document is opened (Bowen et al. 2013). This approach is more useful for the detection of malicious insider users; APT actors are likely to exfiltrate the files from the infrastructure before they open them, and thus connection attempts towards the monitoring server will fail.

Honey accounts

Use of honey accounts is an additional way of detecting attackers, as any interaction (e.g. authentication attempts) with these accounts is a clear indication of an active attack, while the false positive rate is practically zero. Honey accounts can either be real, active accounts (i.e. domain users) or fake ones. In both cases, they should be very closely monitored. The access rights for real/active honey accounts need to be selected carefully, as even if an active honey account gets compromised, it should not offer any benefit to the attacker.

Honey accounts could be combined with the aforementioned example of placing honey files on file servers, where a file with fake credentials could be created. An attacker with access to this file is very likely to use the credentials in an effort to gain further access to the network. The evaluation results of the proposed model support this statement (see 6.4.4.2).

Finally, a similar approach is the creation of fake authentication tokens in memory, which will be listed along the real authentication tokens when attackers attempt to do Pass-the-Hash attacks (Ewaida 2010). Such tokens can be easily created using built-in Windows tools (Baggett 2015).

6.3 A novel APT detection model

6.3.1 Anomaly-based attack indicators

Regardless of the skill, techniques and tools that the attackers use when interacting with a system or network, their actions will inevitably generate some noise (anomalies). The anomaly-based indicators used by the proposed model are listed below. It should be noted that depending on the characteristics of the infrastructure that needs to be protected, policy requirements and risk appetite, the list of indicators can be enriched or reduced accordingly.

- Multiple, successful logins from an account to multiple systems
- Successful logins during irregular business hours and/or non-working days
- Multiple login failures for one or more accounts
- A large number of network connections originating from a system
- Network connections between workstations
- Network connections from a workstation to multiple servers
- Network connections originating from server(s), towards workstation(s) or the Internet
- Protocol anomalies e.g. large number of failed DNS queries which could be an indication of malware using a domain generation algorithm
- TCP connections which last for several minutes (as the majority of TCP connections are short lived)
- TCP connections for which the transmitted traffic from the internal system is significantly more than the received traffic (this is an indication of uploading data)
- Periodical connections to specific domain names/IP addresses
- High volume of traffic between two systems (especially when one of the systems is outside of the organization e.g. Internet)
- Connections to countries where the organization does not do business with
- High resource utilization (CPU, memory, disk usage etc.)
- Access to multiple files (e.g. on a file server) from a single account within a small time period
- Multiple or large (multiple records) database queries from a single account within a small time period

It should be noted that these indicators won't raise alerts when triggered in isolation; each indicator is assigned a threat value (i.e. a weight) and alerts will only be raised if this value exceeds a predefined threshold (See section 6.3.4 for more details).
The aforementioned indicators can be collected from two main sources:

1. System and network device logs
2. Network traffic metadata e.g. IPFIX (Claise et al. 2013) / NetFlow (Claise 2004)

The most common log types that can be used for the collection of these statistics are discussed in section 6.3.3.1.

6.3.2 Deception-based attack indicators

In contrast with the anomaly-based attacked indicators which can be prone to false positives, when deception-based indicators are triggered the possibility of a true positive (an actual attack) is much more likely. This makes them ideal for attack detection on Internet facing systems, where anomaly-based indicators would create significant noise. The proposed model monitors for the following deception-based attack indicators:

- Authentication attempts using honey-accounts
- Access of honey files
- Access of honey records
- Network traffic destined to darknets (preferably for monitoring internal networks)
- Connection/Interaction with honeypots (preferably for monitoring internal networks)

Potentially some of the above indicators may also generate false positives, as a user may mistype an IP address and thus connect to honeypot or a darknet. However, if multiple deception indicators are triggered and especially if these are accompanied with anomaly-based attack indicators, then the probability of a true positive (i.e. actual attack) is very high.

6.3.3 Data collection

6.3.3.1 Collection of attack indicators: Log file analysis

Most modern Desktop/Server Operating Systems support robust auditing mechanisms. The following log files can be used to gather the required information for the calculation of the anomaly-based attack indicators:

- *Authentication logs*, which include the date and time, the user (username) who tried to log in to a system or service, the location from which the authentication was initiated (e.g. locally or from a remote system) and the result of the authentication attempt (success, failure).
- *System logs*, which contain information regarding the status of the system, the services that are running, the resource utilization etc.
- *File system auditing*, which includes granted and denied access requests to files and folders. This is an invaluable source of information for the detection of abnormal access patterns (e.g. an authorized user who accesses several documents in a short period of time).
- *Database auditing*, which logs database queries and can be used to detect anomalies, like a user accessing a large number of database records in a short period of time or accessing records that she shouldn't have need to.
- *Application specific logs*, e.g. DNS server logs, which include the successful and failed DNS queries that have been performed.

Despite the fact that the aforementioned auditing features are available in most OS's, the format and the storage mechanism of the logs varies significantly. Ideally, all relevant logs should be collected to a central repository for processing, to simplify analysis. Forwarding the logs to a central system has the additional benefit of safeguarding them, as it is a common practice for attackers to delete local log files when they compromise a system to hide their traces.

The means for collecting the log files differ based on the log source and the version of the operating system. Most of the operating systems have built-in support for forwarding the logs to another system, however a number of alternative solutions also exist (commercial and open source). These support additional features such as secure transmission of the logs, log buffering in case the collection server is not available, normalization of the logs in a common format for easier querying and indexing, and more.

Unfortunately, on most operating systems (particularly Windows) only very basic auditing is enabled by default, especially with regards to authentication and file system access. A detailed auditing policy needs to be enabled in order to be able to extract the required information for the attack indicators. The proposed audit policy configuration for Windows environments is presented in Figure 12 and Figure 13 (Windows Server 2008).

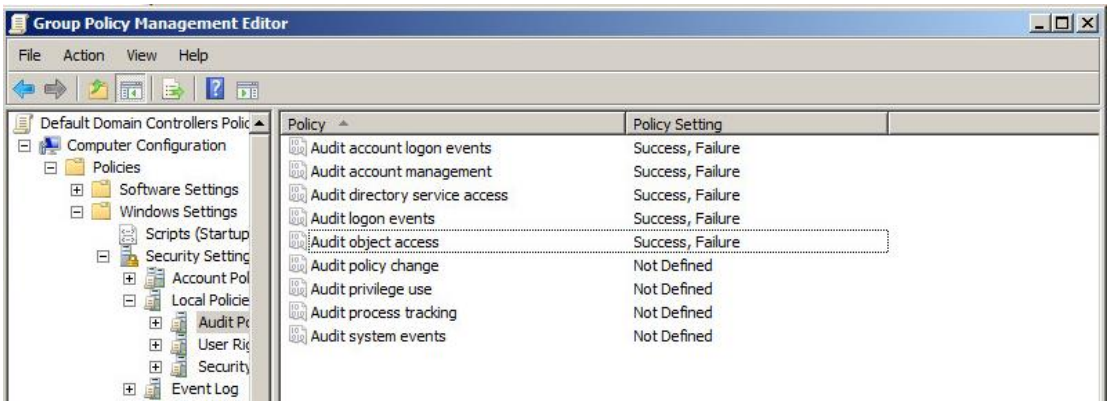

Figure 12 - Windows audit policy

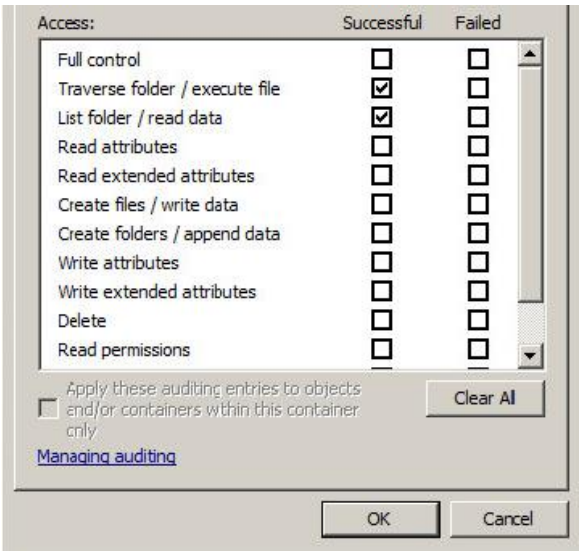

Figure 13 - Windows audit policy for File System access

It should be noted that enabling detailed logging has hardly any noticeable performance impact on modern workstations/servers, however it tends to affect database performance (database auditing) under heavy load. Nevertheless, the benefits significantly outweigh the performance hit.

Ideally, logs from all workstations, servers (including database servers) and network devices (routers, firewalls, switches) should be collected. If this is not feasible, then as a minimum, logs from the Domain Controller(s), DNS server(s), File server(s), Email Server(s) and Database Server(s) should be collected.

6.3.3.2 Collection of attack indicators: Network traffic Analysis

Collection of network traffic can be achieved in multiple ways, which depend on the size of the organization, the network architecture, the networking technologies in use and the

available budget. As a minimum, statistics (metadata) related to network connections have to be collected. These should include: a. the source IP address and port, b. the destination IP address and port, c. the time the connection was established and d. the number of packets and bytes transferred. The above can be easily achieved by collecting IPFIX/NetFlow data from router(s) (see Table 14), as most commercial routers offer this capability. If not, software based solutions can be used, such as "the Bro Network Security Monitor" (Bro 2015).

Table 14 - Typical NetFlow record (NFDUMP)

Date/Time	Duration	Proto	Src Addr & Port	Dst Addr & Port	Pkt	Bytes	Flows
2015-02-07	0.001	UDP	172.16.10.1:9001	172.16.10.41:443	1	46	1
2015-02-07	63.541	TCP	172.16.10.52:4368	172.16.10.51:9837	62	3512	1

Depending on the solution used for generating these connection metadata, it may also be possible to collect additional information. The first few bytes after the establishment of a TCP connection and the last few bytes before its termination, will offer additional insight on the protocols used and the transmitted data. Although more storage space will be required compared to pure IPFIX/NetFlow records, in most of the cases the storage requirements will be achievable even for small organizations.

The solution offering the greatest visibility is the full packet capture (FPC) of all network traffic. Unfortunately, commercial FPC solutions are very expensive and although open source implementations exist, the cost of storing traffic even for a duration of a few days on high throughput networks is prohibitive for most organizations. Furthermore, FPC solutions suffer from a number of shortcomings (Virvilis, Serrano, et al. 2014), as discussed in section 2.2.1.3.

As the goal of the proposed model is to be as widely deployable as possible with a minimal cost, only connection metadata are used. The location of the collection points will depend on the network architecture but the general rule is the more visibility, the better (assuming proper de-duplication is in place). A sample architecture and collection points are show in Figure 14. As a bare minimum there should be collection points on the border router(s) and the core switch(es), especially the ones between workstations and servers.

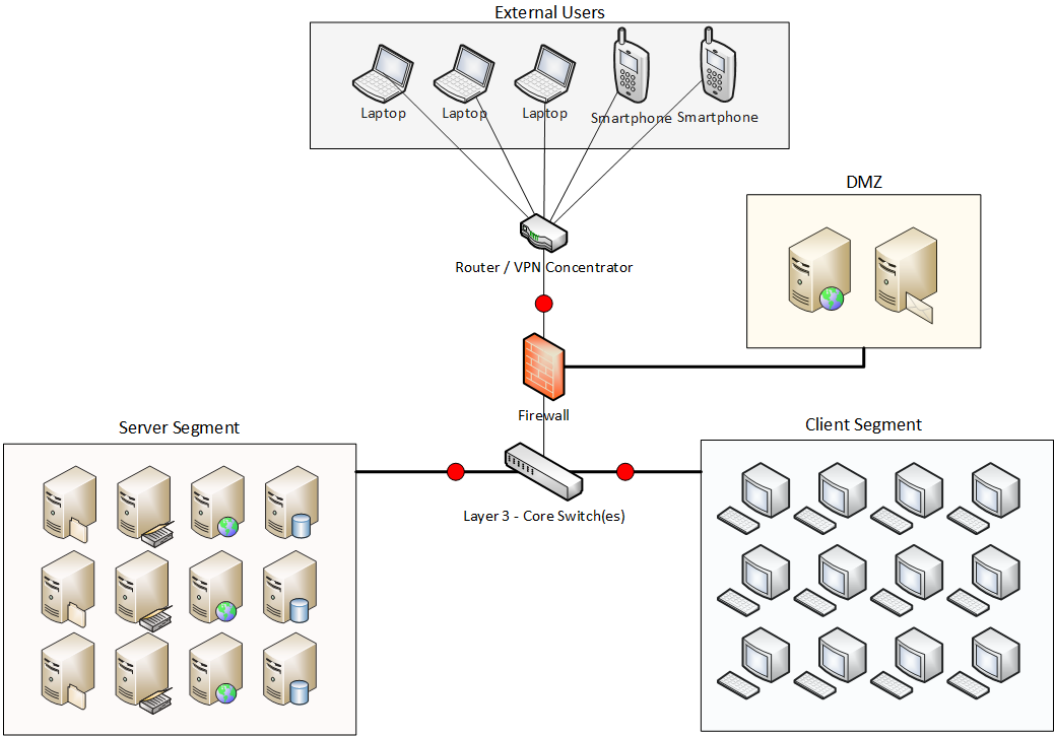

Figure 14 - Sample architecture with proposed network monitoring points

6.3.4 Data analysis (Correlation engine)

6.3.4.1 Correlation window

The correlation engine is responsible for the analysis of the collected data (e.g. logs, network traffic metadata) and the identification of anomalies (attack indicators). The aforementioned indicators (see 6.3.1, 6.3.2) consist the basis of the correlation engine and for each one of them a correlation window is proposed (see Table 15). For the indicators with a short correlation window (i.e. minutes), all relevant captured data between the current time minus *Nm* (minutes) will be analyzed. For the indicators with a correlation window of a few hours, all relevant captured data between the current time minus *Nh* (hours) will be analyzed, while for indicators with a large correlation window (several days), all relevant captured data between the current time minus *Nd* (days) will be analyzed.

The default *N* values (based on which this model has been evaluated are: *Nm*=30, *Nh*=24, *Nd*=15 however, they can be fine-tuned depending on the unique characteristics of the monitored infrastructure (e.g. open or closed network, classification, number of users, allowed protocols/services etc.).

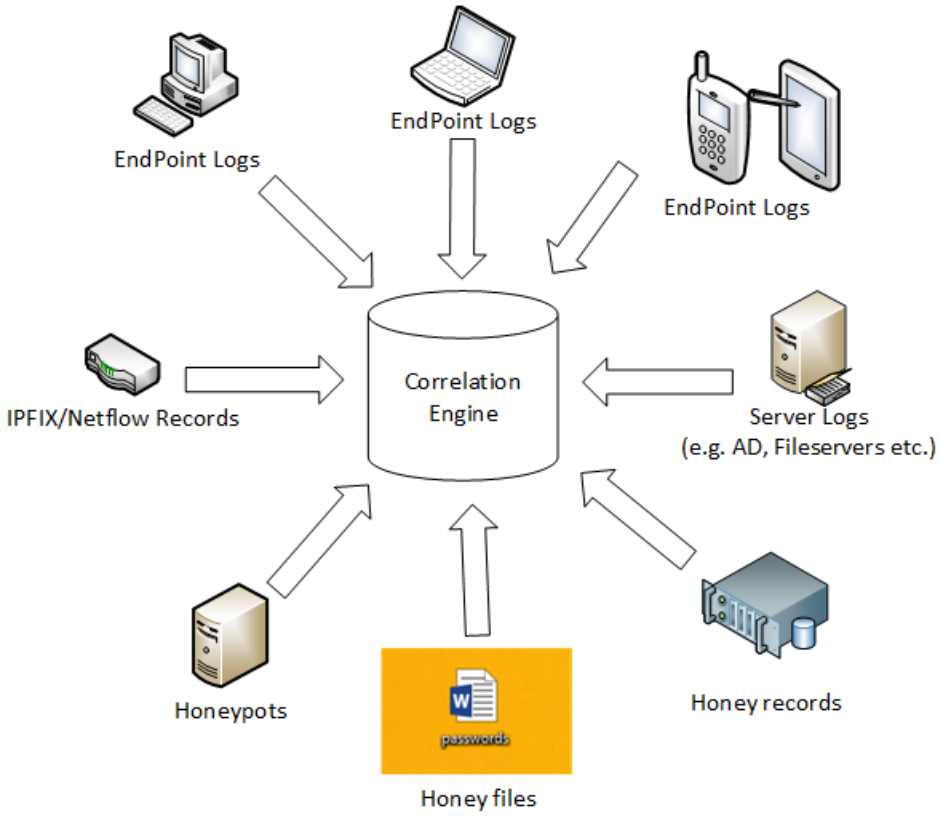

Figure 15 - Inputs to correlation engine

As shown in Table 15, for the calculation of the *"Multiple, successful logins from an account to multiple systems"* indicator, there should be correlation of all relevant data (e.g. authentication logs) that have been created in the last *Nh* hours. A smaller correlation window could also be used (e.g. successful authentications in the last *Nm* minutes) but is not recommended, as it would enable the attackers to avoid triggering the indicator by logging in to only 1-2 systems per *Nm* minutes.

On the other hand analysis of *"Successful logins during irregular business hours and/or non-working days"* should have a short correlation window. This indicator reports on irregular authentication/login times (off-hours, weekends etc.) and thus there is limited value in analyzing historical data.

"Multiple login failures for one or more accounts" which is a characteristic of brute-force attacks used to guess valid credentials for a system/service, should have a correlation window of a few hours (e.g. *Nh* hours). A smaller correlation window would allow the attackers to evade detection by reducing their brute-forcing rate. Although it is still possible that they may decide to reduce the rate even further (e.g. limit it to a handful attempts per day) in order to evade triggering the indicator, this would limit dramatically the effectiveness of their attack. Finally, a longer correlation window is

expected to create false positives as users might mistype their passwords, and thus should be avoided.

Connection statistics group (all network related indicators) is used to identify protocol anomalies which could be the results of malicious actions. These indicators should also have correlation windows of *Nh* hours as a lower value would allow easy evasion, while a higher would create false positives.

"High resource utilization" should have a small correlation window as the focus of this indicator is to detect sudden usage spikes.

Indicators related to file system access and database queries (excluding access to honey files / honey tokens) should have a medium correlation depth (*Nh* hours). It is possible that an attacker might try to access only a handful files/records per day to evade detection, however performing multi-day correlation will inevitably create false positives (e.g. it is expected that benign users will access a small number of files every day, as part of their day-to-day activities).

Finally, all deception based indicators due to their low false positive rate and their significant impact should be correlated over a long window (*Nd* days).

6.3.4.2 Threat Rating, Average Threat Rating

As not all of the aforementioned attack indicators have the same severity or are prone to false positives, each indicator has been assigned a "Threat Rating" (TR) with values: **Low**, **Medium**, **High** or **Critical** (see Table 15). The indicators that have a higher likelihood of triggering false positive alerts have received a lower threat rating. On the other hand, indicators that when triggered there it is a high probability of malicious activity (true positive) have received a higher threat rating.

For each IP address/domain name or username which triggers an attack indicator, a value called Aggregated Threat Rating (*ATR*) is calculated. Each of the Threat Rating values (Low to Critical) is assigned a different weight, which is used for the calculations of the ATR: **Low = 1, Medium = 3, High = 6, Critical = 9.**

If multiple indicators are triggered from the same origin (e.g. same IP address or username) then, the Aggregated Threat Rating is equal to the sum of the independent Threat Ratings *(TR)*:

$$ATR = TR_1 + TR_2 + \ldots + TR_n$$

The higher the value of the *ATR*, the higher the probability of an actual attack.

Table 15 - Indicator threat rating

Type[1]	Indicator	Threat rating	Correlation Window (*Nm, Nh, Nd*)
A	Multiple, successful logins from an account to multiple systems	M	*Nh*
A	Successful logins during irregular business hours and/or non-working days	M	*Nm*
A	Multiple login failures for one or more accounts	M	*Nh*
A	A large number of network connections originating from a system	L	*Nh*
A	Network connections between workstations	L	*Nh*
A	Network connections from a workstation to multiple servers	L	*Nh*
A	Network connections originating from server(s), towards workstation(s) or the Internet	L	*Nh*
A	Protocol anomalies	L	*Nh*
A	TCP connections which last for several minutes	L	*Nh*
A	TCP connections for which the transmitted traffic from the internal system is significantly more than the received traffic	L	*Nh*
A	Periodical connections	L	*Nd*
A	High volume of traffic between two systems	L	*Nh*
A	Connections to countries where the organization does not do business with	M	*Nm*
A	High resource utilization	L	*Nm*
A	Access to multiple files from a single account within a small time period	H	*Nh*
A	Multiple or large (multiple records) database queries from a single account within a small time period	H	*Nh*
D	Network traffic destined to unused IP ranges	H	*Nd*
D	Network traffic or interaction with honeypots	H	*Nd*
D	Authentication attempt using a honey-account	C	*Nd*
D	Communication attempts (e.g. emails) towards a honey-account	H	*Nd*
D	Access of honey files (e.g. on a file server)	H	*Nd*
D	Access of honey records (e.g. database records, DNS records)	H	*Nd*

[1]A: Anomaly indicators, D: Deception indicators

For example, if network connections between workstations are detected (*TR=Low*) and the originating IP of these connections also connects to multiple servers (*TR=Low*), followed by high volume of traffic between that particular IP and one of the servers (*TR=Low*), then the *ATR* for this IP will be 3 (3 indicators with low TR (weight 1)). On the other hand, an authentication attempt with a honey user account (*TR=C*), followed by an interaction with one of the honeypots (*TR=H*), will have an aggregated threat rating of 15 (*9 + 6*). If the *ATR* exceeds a predefined level (see next section) an alert will be generated. The process is depicted in Figure 16.

The duration for which the *ATR* will be valid for the IP Address/Domain name or a user account, depends on the correlation window of the threat indicator that was triggered. If more than one threat indicators were triggered, the *ATR* will be valid for the duration mandated by the indicator with the higher correlation window. For example, if a particular IP address receives an *ATR* of 1 because it initiated a large number of connections (*Nh* hours) and one of these connections was to the IP address of a honeypot (*Nd* days), the *ATR* for that IP address will be active for *Nd* days.

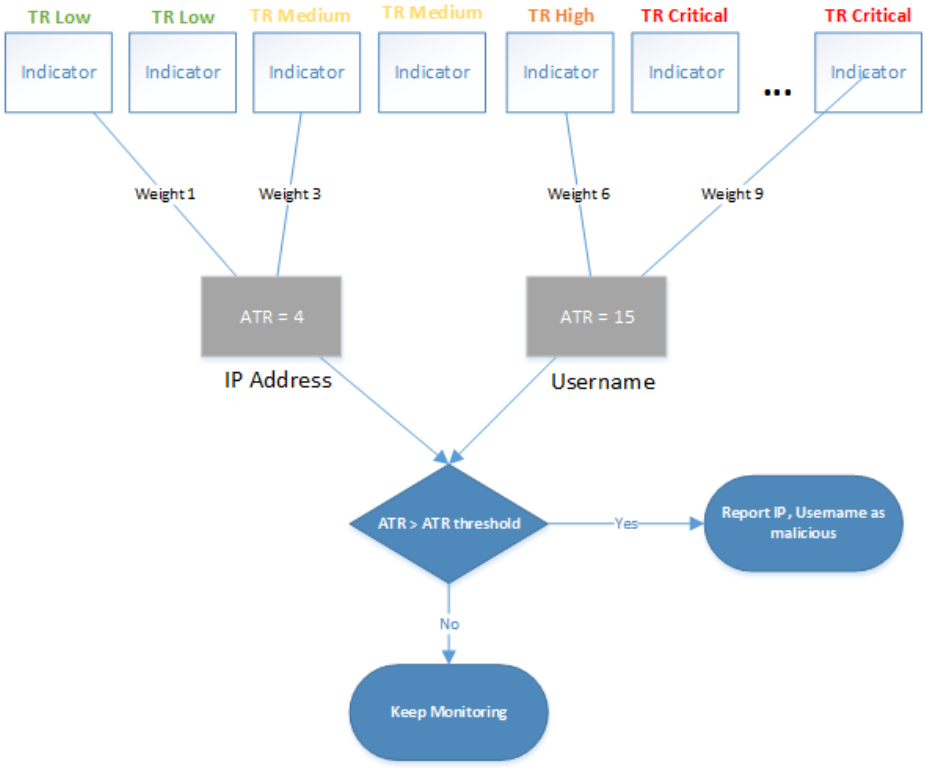

Figure 16 - ATR calculation process

6.3.5 Alerting level - thresholds

Depending on the characteristics of the infrastructure that needs to be protected and the risk appetite of the organization, the correlation window(s) and the threat rating might need to

be fine-tuned. The threat rating of the attack indicators in a closed classified network with a limited number of users, a handful of services and only specific network protocols, can be much stricter than for a public network with Internet facing systems.

The proposed model is flexible and allows the fine-tuning of: a. the attack indicators (i.e. which ones to use) and their threat rating, b. the correlation window for each indicator and c. the *ATR* threshold, which if exceeded the IP address or username will be reported as malicious.

The proposed *ATR* threshold for an unclassified, Internet connected network is equal to the value of the Critical Threat Rating (9). At this level there is only one indicator ("*Authentication attempt using a honey account"*) that could trigger an immediate alert. The reason is that the possibility of an actual attack when this indicator is triggered is very high, while the false positive rate is virtually zero. For the rest of the indicators, at least two have to be triggered within the respective correlation window(s), for an IP address/username to be reported as malicious.

Finally, for some of the attack indicators, indicatory-specific thresholds have to be defined. For example, for the "*Multiple, successful logins from an account to multiple systems*" indicator, a threshold of 5 would mean that if more than 5 successful logins are detected from the same account to different systems within Nh hours, then the attack indicator would be triggered. These thresholds depend on multiple parameters such as the way users authenticate to systems (e.g. single/two factor authentication, the use of single sign-on etc.), as well as the risk appetite of the organization. A higher threshold would reduce potential false-positive triggers but at the same time could potentially allow malicious actions to pass undetected. As a result, these values need to be fine-tuned by taking into account the unique characteristics of the protected infrastructure. The threshold values which were used for the evaluation of the PoC implementation can be found in the Appendix (Snippets 4 & 5).

6.4 Model Evaluation

The evaluation of an APT detection model is challenging as the *modus operandi* of APT groups needs to be simulated realistically. In order to address this challenge, four expert penetration testing teams (two military and two from the industry) were asked to participate in two attack scenarios. For each scenario the teams were asked to achieve specific objectives while trying to be as silent/stealthy as possible.

The first scenario was simulating an external attack (i.e. an attack originating from the Internet). The teams were given a domain name and were asked to find and submit a specific value (the flag), thus proving that they had achieved their goal(s).

The second scenario was simulating an internal attack, where the attackers had managed to establish foothold in an internal network with several workstations and servers. The teams

were given remote access (via RDP) to one of the internal workstations and were asked to locate and exfiltrate specific information from the infrastructure, without being told on which system(s) the information was located.

The aforementioned scenarios are described in detail in sections 6.4.1 and 6.4.2.

6.4.1 External Attack Scenario

For the external attack scenario a new domain name was registered: *virvilis.com*. The A record for the *www.virvilis.com* subdomain was pointing to a Linux server hosting the following services: a. web server on port 80/TCP, b. an SSH server on port 2222/TCP and c. a custom application on port 31337/TCP (see Snippet 1 in the Appendix). A second Linux Server was acting as the authoritative DNS server for the *virvilis.com* domain and its only Internet accessible service was DNS at port 53/UDP. The architecture is depicted in Figure 17.

The web server hosted a very basic 10 page PHP application, which was developed for the purposes of this scenario. In addition, a python application was developed, which listened on port 31337/TCP and asked for the "flag" (e.g. the proof that the team managed to complete the challenge). A correct submission of the flag was concluding the challenge. There were two valid values for the flag: The first one was a word available on the owner's LinkedIn profile (see below), while the second one could be accessed by exploiting a command injection vulnerability which was intentionally coded in the web application.

The following deception indicators were used by our model, to detect malicious actions against the infrastructure:

- Four subdomains were configured on the DNS server out of which two were honey tokens
- Three *robots.txt* entries on the web server, one of which was a honey token
- One honey token at the login page of the web application (*base64 encoded(username:password))*, as an HTML comment
- One invisible link in the admin page

In addition:

- For the web site owner (a fake persona), a LinkedIn profile was created with realistic information
- An email address was created for the fake persona, and was monitored for interaction (e.g. phishing emails)

As previously mentioned, only deception indicators were used for monitoring the infrastructure due to their very low false positive rate, which makes them ideal for Internet facing systems.

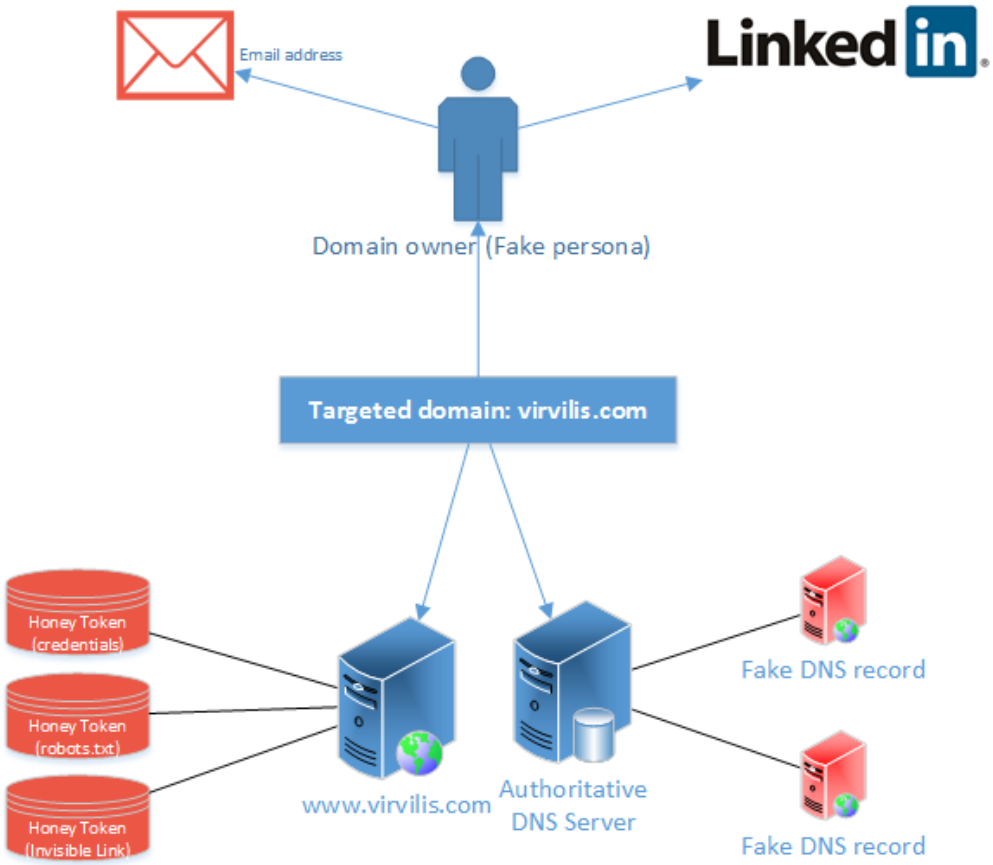

Figure 17 - External attack architecture

Furthermore, the open source Intrusion Detection System Snort (Snort 2015) was monitoring all traffic to and from the servers (Snort 2.9.7 with the community detection rules). Snort was not used to feed any data into the model; its purpose instead was to measure and compare the efficacy of a "traditional" security solution with the proposed model.

Two of the teams were informed about the use of deception countermeasures in advance, but no specific details were revealed to them. The other two teams were not given any information regarding the implemented defense mechanisms. The teams were asked at the end of the challenge to send the IP address(es) that they had used, in order to validate the alerts that were triggered by the model.

Finally, as the infrastructure was Internet accessible anyone could connect and interact with it. To increase realism, five individuals (not part of the attack teams) were asked to browse regularly to *www.virvilis.com* so as to simulate normal user activity on the web server and thus, identify if the deception indicators would create any false positive alerts.

6.4.2 Internal Attack Scenario

For the internal attack scenario the following architecture was created and hosted on a single ESXi Server:

- Three Class C networks, two of which were production networks (server and client segment), while the third one was used as a darknet
- 20 hosts in total, 15 of which were actual Virtual Machines (workstations / servers), while 5 were simulated systems acting as honeypots (using honeyd)
- All Windows hosts were members of the same Active Directory domain
- Linux hosts were stand-alone (i.e. there was not centralized authentication)
- Multiple, commonly used services, were accessible on each VM (see Table 16)

Table 16 - Server and Client Segment: VM type and services

IP address	Type	OS	Main Services / Purpose
172.16.10.1	VM	Windows Server	Active Directory, RDP
172.16.10.5	VM	Windows Server	File Server, RDP
172.16.10.6	VM	Windows Server	MSSQL Server, RDP
172.16.10.10	VM	Windows Server	Application Server, RDP
172.16.10.100	VM	Linux Server	SSH, HTTP
172.16.10.101	VM	Linux Server	Hosted our detection model
172.16.10.102	VM	Linux Server	SSH, MYSQL
172.16.10.105	VM	Linux Server	FTP, SSH
172.16.10.110	VM	Linux Server	SSH, POSTGRESQL
172.16.10.20	Honeypot	Windows Server	RDP
172.16.10.120	Honeypot	Linux Server	TELNET, SSH, HTTP
172.16.10.2	Honeypot	Emulated Network Attached Storage (NAS)	SMB2, HTTP
172.16.11.1	VM	Windows	Workstation (Client)
172.16.11.2	VM	Windows	Workstation (Client)
172.16.11.3	VM	Windows	Workstation (Client)
172.16.11.4	VM	Windows	Workstation (Client)
172.16.11.5	VM	Windows	Workstation (Client)
172.16.11.6	Honeypot	Windows	Workstation (Client)
172.16.11.7	Honeypot	Windows	Workstation (Client)
172.16.10.254 172.16.11.254	Router	Router	SSH

The network architecture was as follows:

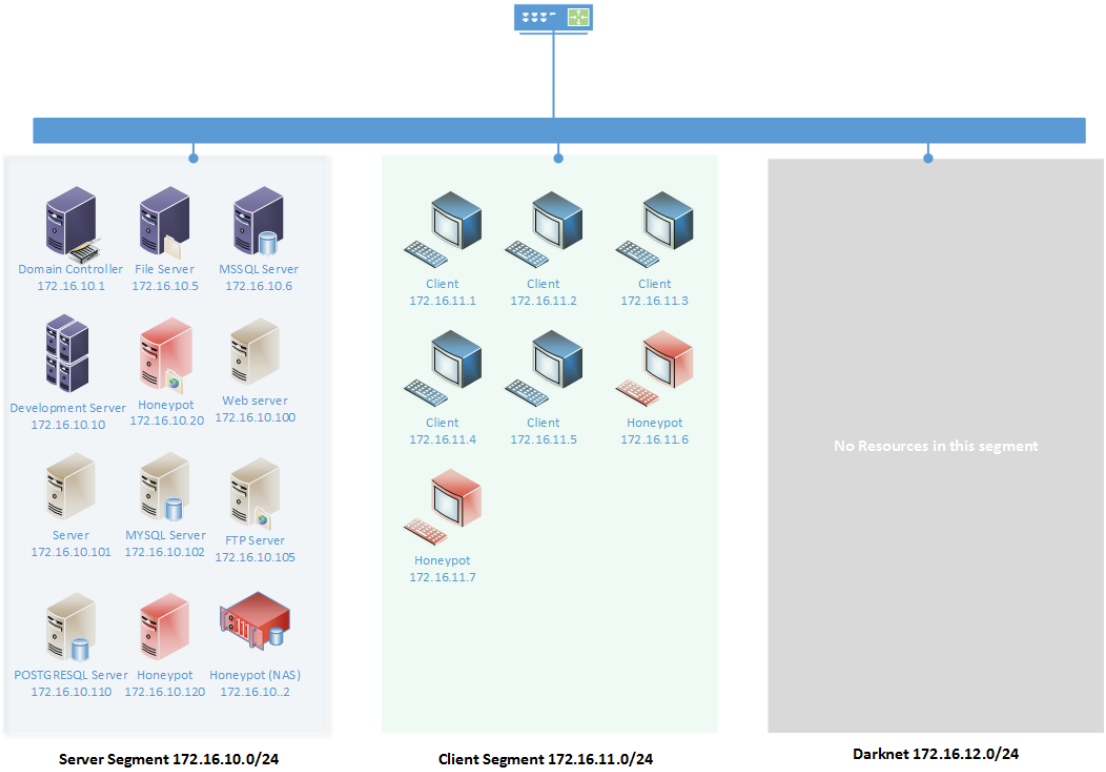

Figure 18 - Internal network architecture

The teams were given remote desktop access (RDP) to one of the client VMs (workstations) in the client segment (different VM per team). This was done to simulate a compromised endpoint controlled via a remote administration tool, which is very common for APT attacks.

They were asked to find information related to a software patent (2 flags in total) and exfiltrate it, while taking great care to avoid detection. One flag was located on the file server, while the latter on the MySQL database. The MySQL Service was accessible over the network, but required authentication. Once more, two of the teams were informed about the use of deception techniques but no specific details were given. The other two teams did not receive any information related to the implemented security measures.

A proof of concept implementation (PoC) of (a subset) of the proposed model was monitoring the virtualized infrastructure. More specifically, the following deception indicators were implemented and monitored:

- 2 honey files on the fileserver (out of 150 files in total). One of the files contained 4 honey accounts (username & password combinations)

- 2 honey authentication tokens on the File Server and MSSQL server (to detect pass-the-hash attacks)
- 5 honeypots, 3 on the server segment and 2 on the client segment, emulating different operating systems/services
- A darknet (172.16.12.0/24)

In addition, the following anomaly-based attack indicators were monitored:

- Multiple, successful logins from an account to multiple systems
- Successful logins during irregular business hours and/or non-working days
- Multiple login failures for one or more accounts
- A large number of network connections originating from a system
- Network connections between workstations
- Network connections from a workstation to multiple servers
- Network connections originating from server(s) towards external IP addresses
- TCP connections which last for several minutes
- TCP connections for which the transmitted traffic from the internal system is significantly more than the received traffic (i.e. uploading of data)

Similarly to the previous scenario, Snort 2.9.7 with the community rules was monitoring all internal traffic. More details regarding the proof of concept implementation can be found in the next section.

6.4.3 Proof of Concept Implementation

A subset of the proposed model was built and evaluated. The model consisted of a SQLite database and three modules: The first module was responsible for parsing the Windows binary XML log files (.evtx) and Linux log files (syslog format) (Snippet 3 in the Appendix). The second module was responsible for the analysis of network traffic (Snippet 4 in the Appendix). The results from both modules were stored in the database and were analyzed by the correlation engine (third module), every hour (Snipped 5 in the Appendix).

The model was set up on an Ubuntu Linux 14.04 LTS VM, with 2 GB RAM, 1 vCPU and 20 GB of disk space. Apart from the model's components, the following applications were also installed on the VM: The Bro network security monitor (Bro 2015) (for the creation of network traffic metadata) and the honeyd service (for the creation of honeypots). The port group on the virtual switch was configured to allow the network interface of this VM (monitored by Bro) to receive all traffic generated in the virtualized infrastructure (e.g. acting as a SPAN port). For the rest of this thesis this VM will be referred to as the "Monitor VM".

More specifically, for the internal scenario the log parser parsed the *"Security.evtx"* log files on the Domain Controller and the File Server VMs, and collected events related to user authentication and file system access (see Table 17). Furthermore, it also parsed the authentication log file (auth.log) of Linux VMs to gather logon events. The windows logs were exported to a shared folder every 60 mins on each of the servers, using the wevtutil command line tool (*wevtutil epl Security c:\fslogs\Security.evtx /ow:True*), via a scheduled task. A domain user account was created specifically for the model and it was used to mount the shared folders on both servers from the Monitor VM (the folders were only accessible from this account). For Linux systems, SCP was used to copy the authentication log file (auth.log) from each VM to the Monitor VM every 60 mins.

Table 17 - Parsed Windows event logs

Event ID	Purpose
4624	Successful logon events for logon types: 2 (Interactive), 3 (Network) and 10 (remote interactive)
4771	Failed logon event (Kerberos pre-authentication)
4625	Failed logon event (This will also capture failed "runas" commands)
4663	File system access events

The Bro network security monitor (Bro 2015) was used to generate network connection metadata, which were then parsed by the network module. More specifically, for each network connection the source and destination IP and ports, the duration of the connection and the amount of transmitted data were extracted from the connection log file (*conn.log)* of Bro. This also included traffic to and from the honeypots. The architecture of the PoC implementation is depicted in Figure 19.

The output of both modules (e.g. logon events, file system access, unusual network patterns etc.) was stored in the database. The database schema can be found in the Appendix (Snippet 3). The correlation engine parsed the data in the database and reported the IP addresses/usernames for which the ATR was above the alert threshold, based on the limits and threat ratings discussed in sections 6.3.4.1 and 6.3.4.2.

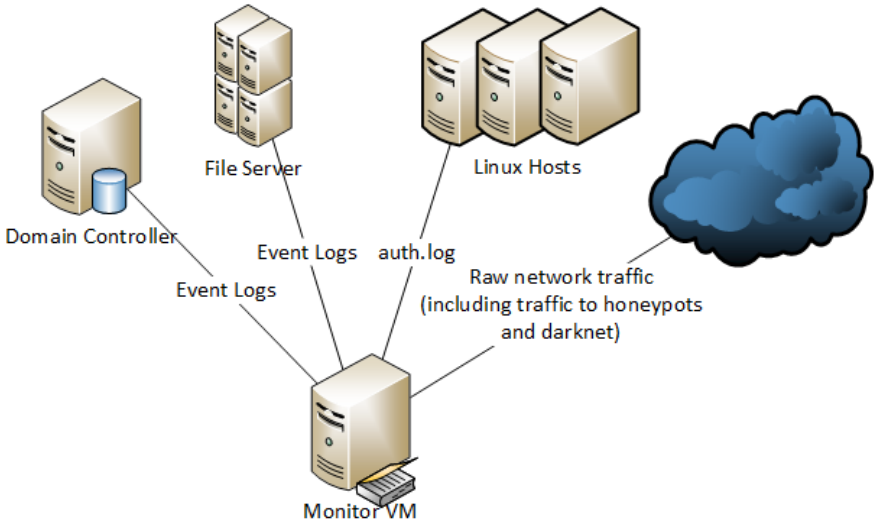

Figure 19 - Data sources for PoC implementation

6.4.4 Evaluation Results

6.4.4.1 External Scenario Results

The total duration of the external scenario was two weeks. During this time, the web server received 8941 requests from 413 unique IP addresses. As presented in Table 18 the first three teams, triggered a wide number of deception indicators and as a result their actions were detected by the model. The fourth team did not complete in the challenge and did not submit the flag.

Table 18 - Triggered attack indicators for external scenario

Team Name	Informed for the use of deception techniques	DNS honey tokens	Robots.txt honey tokens	Login page honey token	Invisible Link	Used Fake Persona Information	Flag	ATR
A	Yes	2/**2**	1/**1**	1/**1**	1/**1**	Yes	Yes	30
B	Yes	1/**2**	1/**1**	1/**1**	1/**1**	No	Yes	24
C	No	2/**2**	1/**1**	1/**1**	1/**1**	No	Yes	30
D	No	0/**4**	0/**1**	0/**1**	0/**1**	No	No	0
Other	No	2/**2**	0/**1**	0/**1**	0/**1**	No	No	12

For example, team A's IP address received an ATR of 30 which was calculated as follows: 5 triggered honey tokens multiplied by 6, where 6 is the weight for an indicator with high threat rating.

One interesting finding was that the first two teams although they had been informed about the use of deception techniques, they triggered similar number of indicators with the team that had not been informed (e.g. team C). Thus, it is evident that deception indicators - when they are realistic enough - are hard to be avoided by attackers.

Team D did not achieve its goal (i.e. did not complete the challenge due to lack of time). Their actions were limited to basic exploration of the targeted systems (i.e. browsing to the web application) and as no malicious actions were performed, the system correctly did not report any alerts.

One team used the information available on the fake persona's LinkedIn profile page, based on which they created a wordlist of possible flag values which they tried. They also used this list to brute force the SSH service on the webserver. None of the teams had any interaction with the fake persona's email address. Based on the interviews at the end of the challenge, the teams said that they would only reside to client side attacks if there was no other way to achieve their objective. Once more, this highlights the fact that when external deception-based indicators are triggered (e.g. due to information gathering attempts), an attack is eminent. The defenders can use this early warning to raise the readiness level (e.g. asking the users to report immediately any unexpected behavior etc.).

The line “Other”, represents the triggered indicators from IP addresses which did not belong to the teams. A total of 16 IPs triggered DNS honey tokens. The IPs were sequential, from two different networks, both of which belonged to Google Inc. These were benign DNS queries used by Google bot to index new web sites.

As mentioned, an open source Intrusion Detection System was also monitoring all traffic to and from the infrastructure. While the challenge was active, Snort generated 386 alerts (8 medium severity and 378 low severity ones) for 54 unique IPs (see Figure 20). None of the alerts included the IP addresses of any of the participating teams and all of them were false positives, highlighting once more the questionable effectiveness of traditional security solutions. The number of reported alerts is presented in Table 19.

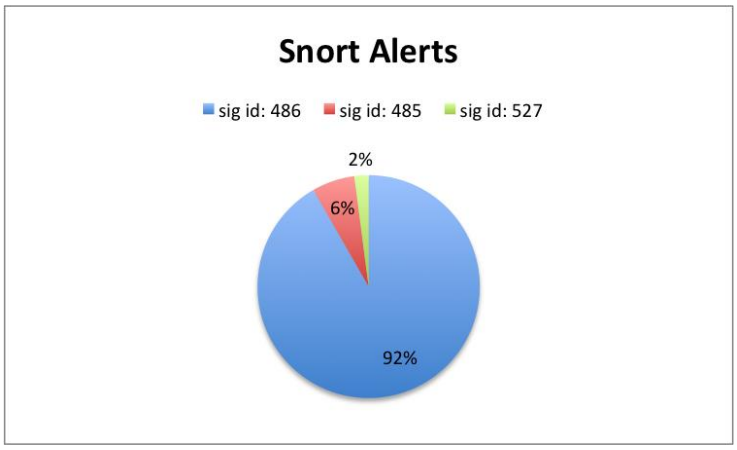

Figure 20 - Snort alerts based on signature id

Table 19 - Detection results

Detection Mechanism	Number of True Positive alerts	Number of False Positive alerts
Proposed model	3/3	16
Snort IDS	0/3	397

It should be noted that neither Snort nor the proposed model had been fine-tuned in order to reduce the number of false positives. All Snort alerts were related to ICMP error messages or bad traffic (packets with the same source and destination IP address). Thus, with minimum fine-tuning the number of false positives would be significantly reduced. Similarly, the model could be configured not to report triggered DNS honey tokens from Google’s IP ranges (and other major search engines) and thus, could achieve a zero false positive rate.

Finally, it should be noted that although the teams refrained from changing their IP address during the challenge which made the analysis easier, in actual APT attacks the perpetrators are likely to try and hide their real IP address and/or use multiple addresses, to reduce the possibility of being detected. Use of anonymous proxies, compromised hosts (acting as a hopping points) or an anonymizing network, will hide the real IP address of the attacker(s), however this does not affect the effectiveness of the model. Attribution of cyber-attacks is a very challenging problem and is out of the scope of this work. Our focus instead, is to provide an early warning of an active attack to the defenders. Thus, even if the true origin of the attack (IP address) is spoofed, the defenders will be aware that someone has started a targeted attack against their infrastructure, which will allow them to raise the readiness level.

6.4.4.2 Internal Scenario Results

The overall duration of the internal scenario was three weeks. Some teams managed to gain access to the targeted information (the flags) using multiple strategies, however all of the teams were detected at the early stages of their attack process. As the purpose of the scenario was to evaluate the effectiveness of the proposed model in detecting malicious actions, the teams were allowed to continue (regardless that they had been detected) until they completed the challenge (or decided to stop), while their actions were monitored and additional alerts were raised. At the end of the challenge a short interview with the teams was conducted to gather feedback and confirm the triggered alerts. Statistics for each team are presented in Table 20.

Table 20 - Triggered attack indicators for the internal scenario

	Team A	Team B	Team C	Team D
Honey Files	2/2	2/2	2/2 [4]	0/2
Honey Accounts	4/4	1/4 [2]	0/4	0/2
Honey Authentication Tokes	0/2	0/2	0/2	0/2
Honeypots	0/5	5/5	0/5	3/5
Darknet [3]	0/1	1/1	0/1	0/1
Anomaly-based indicators	None	1. Connections to multiple workstations 2. Connection to multiple Servers 3. Connections to external systems 4. TCP connections which last for several minutes	1. Successful logins during irregular business hours and/or non-working days	1. TCP connections which last for several minutes
1st Flag	Yes	Yes	Yes	No
2nd Flag	No	Yes	No	No
ATR Username (final value)	12	12	18	0
ATR IP (final value)	36	67	3	20
Time of initial alert[1]:	0h 32m	0h 4m	0h 21m	0h 54m
Total duration of attack	2 days	3 days	2 days	1 day

[1] Time of first generated alert (i.e. ATR has exceeded the alert threshold). Time started when the team accessed the workstation for the first time. As the logs were analyzed every 60 mins, the alerts were reported back with a delay (in the worst case scenario, a delay of 59 mins).

[2] Team only used one of the four honey accounts but attempted to authenticate 3 times with it, thus triggering 3 events.

[3] Connections to the darknet were manually monitored via tcpdump, as Bro would only record established connections between endpoints and not connection attempts.

[4] User accessed twice one of the honey files, thus three attempts were recorded.

More specifically, the model raised the first alert for team A 32 minutes after they started the challenge. The reason for this alert was that the team had accessed the two honey files which were located on the file server. This resulted to an ATR of 12 (higher than the alert

threshold) for the username of the logged on user on the workstation controlled by team A. Another alert followed 26 minutes later, as four authentication attempts were made with honey accounts (the accounts were listed in one of the two honey files). Authentication attempts with honey accounts have a Critical Threat Rating (9) and as a result the model reported the IP address of the workstation used by the team with an ATR of 36 (4 times 9).

For Team B, multiple alerts were raised within the first 5 minutes the team started the challenge: Similarly to the first team, team B also accessed the two honey files on the file server (resulting to an ATR of 12 for the username of the logged on user), and performed three authentication attempts using one honey account. The authentication attempts resulted to an ATR of 27 for the IP address of the workstation. No further alerts were raised on that day. On the second day the team while trying to explore the internal network using a port scanning tool, accessed all 5 honeypots and the darknet, resulting to an ATR of 30. Anomaly-based attack indicators were also triggered (as a result of the port scan) due to multiple connections between the workstation used by the team and other workstations, multiple connections from the workstation to servers and multiple connections to external systems (for downloading the tools that were used by the team). As a result, the ATR of the IP used by the team at the end of day two was 66. Finally, the team concluded the challenge on day 3, after gaining access to the MySQL server which contained the second flag. The database was accessible from the network and team successfully guessed the password after a few tries. The only additional indicator that was raised on that day (as failed authentication attempts to the database were not monitored), was the high duration of the TCP connection between the workstation and the MySQL server.

The first alert for team C was triggered within 21 minutes, once more due to accessing the two honey files on the file server. As one of the files was accessed twice, the ATR value for the username of the logged on user was 18 (e.g. the attack indicator triggered 3 times). Other indicators included the successful authentication during non-working days, as the team kept playing during the weekend (ATR=3). The team did not complete the challenge and gave up after finding the first flag, due to time constraints.

Team D triggered the first alert after 54 minutes, as they accessed 3 honeypots. This team followed a different attack strategy than the other teams: They downloaded and used a custom remote administration tool on the workstation, and all further system and network reconnaissance actions were performed via this tool. Their main objective behind this decision was to limit their footprint on the workstation. For the same reason they avoided browsing to the shared folder on the file server, and as a result they did not access the first flag (or any other file on the file server). Their network reconnaissance actions were subtle and limited only to the server segment, as the team expected that the targeted information would be hosted there. However, they got detected when connections were made to the honeypots located on that segment. Finally, the team focused on the exploitation of the MSSQL server as they assumed

that the targeted information might be stored there, but did not attempt to access neither the MySQL server nor the file server, where the information was actually located.

By reviewing the attack indicators which were triggered during the whole duration of the challenge, it is evident that some indicators are more effective in detecting malicious actions than others. For example 3 out of 4 teams accessed the honey files on the file server among multiple other files. In addition, two of the teams performed at least one authentication attempt with a honey account listed in one of the honey files, as they believed that these accounts would allow them to access other systems. As it is highly unlikely that a benign user would try to authenticate with credentials she had found in a random file (i.e. the honey file), such indicator apart from been very effective in detecting malicious users has practically a zero false positive rate.

No authentication attempts were made with the honey authentication tokens located on the File Server and MSSQL server. In order to gain access to the authentication tokens in memory, access to the system with administrative privileges was required. None of the teams gained such access (mainly due to the fact that it was not required for solving the challenge). Nevertheless, use of fake authentication tokens is considered an effective way of detecting pass-the-hash attacks and has a zero false positive rate.

The honeypots and the darknet were also very effective in detecting network reconnaissance actions and this was particularly true for teams B and D, as their network scanning attempts were quickly detected.

Anomaly-based attack indicators were also triggered, mainly due to the network reconnaissance activities. More specifically, multiple connections between the workstation used by team B and other workstations, multiple connections between the workstation and multiple servers (teams B & D), multiple connections to external systems (team B) and long-lasting TCP connections (teams B & D), were reported. In addition, an indicator regarding authentication attempts during non-business hours/days was triggered by team C, as they connected during the weekend.

The anomaly-based indicators that were not triggered by any of the teams were related to multiple authentication attempts and connections which transmitted a large amount of data. The teams were aware that multiple authentication attempts (e.g. account brute forcing) would be noisy and thus avoided them, while the second indicator did not trigger as none of the teams exfiltrated a large amount of data. Although teams B & D performed a handful of authentication attempts against the telnet service on one of the honeypots, this counted as an interaction with the honeypot (deception indicator) and not as an anomaly-based one.

It should be highlighted that although teams A & B had been informed about the use of deception-based attack indicators, they did not manage to evade them. Actually based on the

feedback received by Team B, knowing that such indicators were in place made their task more challenging as it slowed them down significantly in the fear that every action they made could trigger an alert.

Finally regarding accuracy, no false positive alerts were raised by the model during the challenge. There were a few anomaly-based attack indicators that triggered multiple times (long lasting TCP connections and multiple connections to external systems) which were caused mainly due to the automatic installation of software updates (security patches) from the Windows Operating Systems. However, the ATR of these systems never exceeded the alert threshold and thus no alerts were raised.

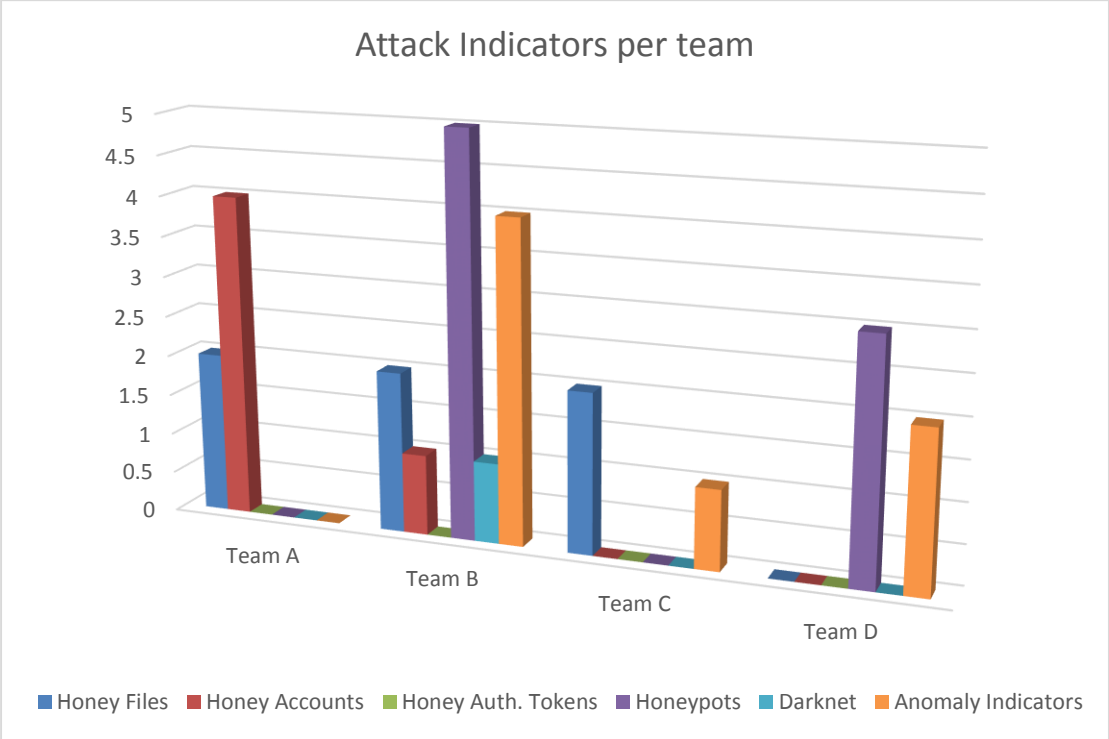

Figure 21 - Triggered attack indicators per team

6.4.4.2.1 Comparison with Snort

Snort raised 2930 alerts during the whole duration of the challenge. The description of the rules that were triggered and the number of times a rule got triggered, are presented in Table 21.

Table 21 - Triggered Snort rules

Signature ID	Number of alerts	Description
538	1485	This event is generated when an attempt is made to gain access to private resources using Samba.
527	171	This event is generated when traffic on the network is using the same source and destination IP address.
408	148	This event is generated when a network host generates an ICMP Echo Reply in response to an ICMP Echo Request message.
466	25	This event is generated when an ICMP echo request is made from a host running the L3 "Retriever 1.5" security scanner.
409	11	This event is generated when a network host generates an ICMP Echo Reply with an invalid or undefined ICMP Code.
486	10	This event is generated when an ICMP destination unreachable (Communication with Destination Host is Administratively Prohibited) datagram is detected on the network.
404	7	This event is generated when An ICMP Protocol Unreachable datagram is detected on the network.
718	7	This event is generated when an attempted telnet login fails from a remote user.
472	4	This event is generated when a network host generates an ICMP Redirect for Host datagram.
485	4	This event is generated when a router was unable to forward a packet due to filtering and used the Internet Control Message Protocol to alert involved hosts.
1042	2	This event is generated when an attempt is made to craft a URL containing the text 'Translate: f' in an attempt to view file source code.
1251	2	This event is generated when an unsuccessful telnet login attempt was detected.
1411	2	This event is generated when an SNMP connection over UDP using the default 'public' community is made.
1417	2	This event is generated when an SNMP-Trap connection over UDP to an SNMP daemon is made.
1418	2	This event is generated when an SNMP-Trap connection over TCP to an SNMP daemon is made.
1420	2	This event is generated when an SNMP-Trap connection over TCP to an SNMP daemon is made.
1421	2	This event is generated when an attempt is made to attack a device using SNMP v1.
1448	2	This event is generated when a request is sent to the Microsoft Terminal Server port.
2925	2	This event is generated when an image fitting the profile of a web bug has been detected in network traffic.
366	2	ping is a standard networking utility that determines if a target host is up. This rule indicates that the ping originated from a host running Unix.
372	2	This event is generated when an ICMP echo request is made from a Windows host running Delphi software.
384	2	This event is generated when an generic ICMP echo request is made.
402	2	This event is generated when an ICMP Port Unreachable message was detected.
491	1	This event is generated when a failed attempt to login to an FTP server is detected.
553	1	The event is generated when an attempt is made to log on to an FTP server with the username of "anonymous".

Similarly to the external attack scenario, neither Snort nor the proposed model had been fine-tuned in order to reduce the false positive alerts. The rational behind this decision was that fine-tuning is a very subjective process, which depends on the skill and risk appetite of the person/team responsible for performing this action. By reviewing the triggered Snort alerts it is evident that a large number of them were due to "noisy" rules, which have no real value for attack detection. For example, rule 538 was triggered every time a samba connection was established, which was very common in our scenario as there were multiple network shares; such connections were normal as all the workstations had a mapped netword drive pointing to a shared folder on the file server (and thus the large number of alerts for this rule). In a production environment such rule whould have to either be disabled or modified in a way that would trigger only for samba connections from systems other than the workstations. In any case, as the teams were using a compromized workstation to access the file server, this rule would not be able to detect any malicious actions. The multiple ICMP alerts were also due to "noisy" rules. Finally, there are some alerts which are clear false positives (verified by analyzing network traffic and host information), such as rule 466.

From all triggered rules, there are only a handful that could – under conditions – help in the detection of a subset of the malicious actions performed by the teams. These are the rules with signature ids: 718, 1042, 1251, 491 and 553.

Rules 718 and 1251 were triggered due to the actions of teams B & D, as they performed a small number of failed authentication attempts to the telnet service of one of the honeypots. Snort reported these as medium severity alerts, as it is quite common for users to mistype their passwords and actually, in a real-world enviroment it is very likely that these rules would be configured to trigger only after several consequent failed authentication attempts, to limit false positives. In comparison, the model reported the IP addresses used by the teams as clearly malicious, due to the fact that they connected to the honeypot(s).

Rule 1042 (medium severity) was triggered by teams C & D as they connected to the IIS server hosted on the Application Server, and attempted to view the source code of the main page. In a real environemnt it is common for developers to review the source code of a web page and thus, it is not clear if this rule would have actually helped in the detection of malicious actions or would be considered a false positive alert.

Finally, rules 491 (medium severity) and 553 (low severity) were triggered after a single failed FTP authentication attempt was performed by team D. Once more, in a real environment it is highly unlikely that the IDS will be configured to alert on single failed authentication attempts, due to the vast amount of alerts that will be raised.

Based on the above results, it is evident that Snort was only able to detect a very small subset of the malicious actions performed by teams B,C,D, while no alerts were raised for team A. No high or critical severity alerts were reported, but only low-medium severity ones, which

are likely to be ignored by the system administrator/security team. Finally, as all of the triggered rules were "noisy" ones, in an operational enviroment it is likely that most of them would be disabled or have a higher trigger threshold and thus, the detection efficacy would be even lower. A direct comparison between the model and Snort is not fair, as the former monitors both network traffic and OS events, while the latter only network traffic. However, even if one considers only the network-related alerts, it is evident that the effectiveness of the proposed model in detecting malicious actions is much better than Snort's. The results are summarized in Table 22.

Table 22 - Comparison of alerts

Team	Snort Alerts	Proposed model Alerts
A	(Nothing detected by Snort)	Access to honeyfiles
		Authentication attempts to other systems using honey accounts
B	Failed authentication attempt to telnet server	Access to honeyfiles
		Authentication attempts to other systems using honey accounts
	Failed authentication to the FTP server	Connections to honeypots
		Connections to darknet
		Multiple (4) anomaly based attack indicators
C	Access of the IIS server	Access to honeyfiles
		One anomaly based attack indicator
D	Failed authentication attempts	Connections to honeypots
	Access of the IIS server	Multiple (2) anomaly based attack indicators

Note: Critical & High severity alerts are depicted in red. Medium in yellow and Low in green.

6.5 Discussion

6.5.1 Technical challenges

As discussed, the correct placement of the traffic collection points and the deception indicators is critical for the effectiveness of the model. Honeypots have to be placed in the same network segments as operational servers/workstations and must be configured as realistically as possible (i.e. same services, OS versions & configuration as production systems). For example on a Windows-only network, Linux honeypots will be of limited use and could even stand out as potential honeypots. Even minor details have to be taken into account e.g. if all Ethernet network cards have MAC addresses from a specific vendor, then the honeypots should also have MAC addresses from the same vendor.

Similarly, honey accounts, honey files and honey tokens, have to be deployed/set-up in a way that will look realistic and of value to an attacker. In order for these requirements to be fulfilled, some effort is needed from the administrator(s) and/or security team of the organization. Initial fine-tuning is required to optimize the detection rate based on the unique characteristics of the protected infrastructure. Specific user accounts and IP addresses might have to be whitelisted to avoid false alerts; for example if the infrastructure is regularly scanned by a network vulnerability analysis tool, the IP addresses of the honeypots will need to be either excluded from the range that the tool will scan, or the IP address of the scanning machine will need to be whitelisted in the model. Similarly, if there is a third party backup solution that archives (thus accesses) at regular intervals all files on the file server(s), the honeyfiles will have to either be excluded for the backup process or the user account (service account) used to launch this process, will need to be whitelisted in the model.

Another challenge is that small and medium enterprises do not always have adequate in-house knowledge to support the installation of such a model. Furthermore, it is common (and unfortunate) that a lot of organizations prefer to invest in “black boxes”, which can be installed with minimal effort and be “forgotten” (i.e. require minimal interaction and maintenance). Thus, it is unlikely that these organizations would consider solutions that require additional involvement from their side.

Maintenance is also a challenge as changes in the network topology or configuration of systems have to be reviewed, to make sure that they won’t affect the deployed attack indicators and as a result limit the efficacy of the model. However, this is also a challenge for traditional security solutions (e.g. NIDS/NIPS, DLP etc.).

Another, obvious requirement is the need for monitoring the infrastructure. However, if the organization has already an established incident management team, the overhead of receiving input (alerts) from this model in addition to the ones generated by their traditional

monitoring systems (e.g. IDS), is negligible – and can even be beneficial – as it could help validate the alerts raised by the latter.

The model's evaluation introduces potential bias due to the fact that the APT groups had to be simulated by penetration testing experts. Although the skillset of these experts is believed to be at the same level as most sophisticated attackers, there are two important parameters to consider: Firstly, APT make frequent use of zero-day exploits for gaining access to systems. Thus, there is always the possibility that the attackers could have used such exploits to compromise the system(s), without performing any reconnaissance/information gathering activities. In that case, the effectiveness of the deception indicators for the external scenario would be limited. However, for the internal scenario no significant changes in the detection efficacy should be expected, as the attackers would still have to explore the internal network to locate and exfiltrate the targeted information. Hence, the possibility of triggering one or more attack indicators would still be high. Secondly, as already mentioned most APT attacks are prolonged and the attackers can spend several months exploiting their target before they find and exfiltrate the information they are interested in. It was not possible to test the efficacy of the model against prolonged attacks, as none of the teams could commit to such a lengthy task. This limited availability was also the reason that some teams did not get access to the MySQL database (second flag), as the straight forward (quick) way of doing so required guessing the database password (i.e. brute forcing), which they considered an action that no sophisticated attacker would do, as it would be easily detected. However, even if the malicious actions were spread over a wide period of time, this would affect mainly the efficacy of the anomaly-based attack indicators. As the deception-based ones have much higher threat ratings and some of them raise immediate alerts when triggered (e.g. authentication attempts with honey accounts/authentication tokens), prolonging the attack would not allow the attacker's actions to pass undetected; the overall ATR value for the offending IP or username could be lower but it would still be reported as malicious.

In addition, due to resource constraints the size of the second (internal) scenario resembles a very basic infrastructure of a small organization. Although the results were very positive, implementation and evaluation of the model in a real environment with a large number of systems and users is needed, in order to test the real-world efficacy of it and more specifically the false positive rate.

When it comes to insider threats (working either on their own or as part of an APT group), the more comprehensive the knowledge of the insider about the deployed security measures is (including the proposed model) the higher will be the likelihood of evasion. An administrator who is allowed to create offline backups of files as part of her job duties, can trivially create additional copies of sensitive information on external media and thus exfiltrate them without

using the network. This is not a limitation of the model, as it is related to physical security. However, even if the malicious insider does not have such rights, if she is aware of all the attack indicators, their type and which parts of the network/infrastructure they are protecting, she can try to evade them. For example, if the insider wants to exfiltrate files from a file server to which she has authorized access to, she might refrain from accessing all the files at once as this will trigger (at least) one of the aforementioned indicators. Instead, she may download a small number of files per day, an action which will not generate any alerts as it is expected user behavior. Inevitably, this will slow her down forcing her to spread her attack over a wide period of time. This could be considered a weakness of the model, but firstly it should be noted that this is a common problem of anomaly-based detection systems and secondly, only a very small number of individuals (should) have complete and in depth knowledge of all implemented security countermeasures. Nevertheless, section 7.3 proposes a potential solution to this problem.

The model does not address supply chain attacks, meaning that it has no way of identifying existing compromised software or hardware components. However, malicious actions can still be detected indirectly. For example, if the attackers have access to the infrastructure via hardware or firmware implants, network connections from/to the compromised systems would still be visible to the model and thus, could be potentially highlighted as abnormal.

Finally, the proposed model should not be considered as panacea against all attacks (regardless of the level of sophistication) and under no circumstances should it be the only line of defense for an organization. Defense in depth, implementation of security best practices, training and awareness and use of traditional security solutions, are all necessary components for achieving a robust security posture. The model does not replace current security solutions (e.g. AV, IDS etc.). Instead it complements them, by increasing significantly the possibility of detecting attacks designed to evade traditional security solutions.

6.5.2 Business challenges

One of the main shortcomings that affect negatively the security posture of organizations nowadays, is the misinterpretation of the risk from the management. Unfortunately most of the executives perceive security as an unfortunate expense that needs to be paid. The effectiveness of a security program is hard to measure and thus, the return on investment (ROI) is not easily justifiable. The lack of (detected) security incidents has a dual meaning: either no one is attacking the organization or the security countermeasures that are in place are mitigating all attacks. As a result, it is common for organizations not to invest in cyber security (or do the absolutely minimum) until a major incident happens.

To make things worse, even when the management understands the real need for security, it tends to have an inaccurate view regarding the effectiveness of the security mechanisms that have been implemented. The main culprit for this is the security industry itself. The vast majority of security products are presented as panacea from the vendors. Some of the vendors even state that their products detect and block APT, something that has been challenged harshly by the security community (Cole 2012; Bejtlich 2013; Schneier 2012). Due to unrealistic promises and the significant cost of these solutions, the management has an incorrect view of the organization's risk exposure. This not only makes risk management harder but hinders the organization's efforts to achieve a robust security posture. Even if the administrators or security team is willing to invest time to implement solutions/models such as the one presented in this thesis, it is hard to convince the management to support the effort.

(This page is intentionally left blank)

Chapter 7: Conclusions

7.1 Summary

The security industry has been criticized severely in recent years because traditional security solutions - even those considered to be state-of-the-art - have failed over and over again to address sophisticated attacks. Advanced malware, such as Stuxnet, Duqu, Flame, Red October, MiniDuke and Regin, have managed to evade detection – in some cases for several years – while allowing the operators to exfiltrate Terabytes of sensitive data or cause physical destruction.

This dissertation begins with an overview of the major cyber-attacks that occurred from the early 1980s to the present, identifies the main players in the cyberwar arena, and presents an overview of the most popular cyber security technologies, highlighting the main shortcomings that have enabled attackers to evade them for years. Furthermore, the three APT attack paths are presented: a) external attacks (including supply chain compromise), b) internal attacks, which include the insider threat, and c) indirect attacks. Each attack path is presented in detail, its main characteristics are highlighted and realistic countermeasures are proposed to limit the exposure.

For the external attacks an in-depth, technical review of the major APT malware used to compromise several sensitive targets in the last years is presented (Chapter 3). Multiple common characteristics are identified including similar code, exploits, programming techniques and methods. Based on these results, the main reasons that the malware (and consequently the malware authors) were able to evade detection for several years are presented. The chapter concludes with a number of technical countermeasures for limiting the impact of malware attacks (regardless of its sophistication level), that can be easily implemented in a wide range of environments.

Internal attacks are discussed in Chapter 4. Two major "insider threat" incidents from two threat actors with very different skillsets but seemingly similar motives are presented, highlighting the severe impact that such attacks can have. Furthermore, the basic components of a robust insider threat-prediction model are presented; they combine technical attack indicators and psychological profiling of users as a means to detect malicious actions.

Attacks against mobile platforms (e.g. smartphones and tablets), although they could be categorized as external attacks, are presented separately in detail in Chapter 5. The rationale for this decision is that smartphones and tablets are expected to be one of the prime targets in the years to come. There are three reasons for this: a) owing to their small size and high mobility

such devices can be stolen easily, b) as a result of limited security countermeasures on mobile platforms, exploitation of such devices is much easier compared to laptops/desktops, and c) smartphones/tablets used for work almost always have remote access to the internal resources (e.g. email, file server) of the organization and thus are a potential entry point.

The focus of the work reported here was to measure and compare the effectiveness of the security countermeasures available on mobile and desktop platforms against phishing and malware attacks, as both are frequently used by APT actors. It was determined that the level of protection offered on mobile platforms – even against non-targeted phishing and malware attacks – was very low, and differed significantly from the protection offered by desktop platforms. Therefore, additional countermeasures, user training and comprehensive monitoring of such devices are required, to safeguard them from being an ideal attacker entry point to sensitive environments.

Based on the aforementioned challenges, it is evident that a significant change in the way we protect our networks is needed. More specifically, the focus needs to shift from prevention – which has been the dominant defensive approach for several decades – to detection, as the former is not realistic against sophisticated attackers.

To address this challenging problem, this thesis proposes a unique APT detection model (Chapter 6). The model uses data collected from multiple sources, including network traffic, security audit logs and usage statistics, as well as data from multiple deception attack indicators, such as honeypots, honey (bait) files, honey records and honey accounts/authentication tokens.

Traditional security solutions are focused on real-time detection, which significantly limits the effectiveness of the correlation; the proposed model offers a much broader correlation window by including historical data in the analysis process. Secondly, regardless of the attacker's skill, methods and tools, his interaction with the targeted infrastructure will inevitably generate (possibly subtle) indicators (e.g. increased resource utilization, abnormal login times, etc.). Such indicators are invaluable for the detection of stealthy attacks, but are almost always ignored by existing security solutions. In contrast, the proposed model focuses on the identification of these subtle indicators and includes them in the decision process.

This comprehensive approach allows the detection of sophisticated attacks that evade traditional security solutions and at the same time reduces the number of false positive alerts. Although delayed detection of an incident is not ideal, it should be noted that in the vast majority of APT attacks, attackers have spent several weeks (and in some cases years) exploiting an infrastructure, trying to locate and exfiltrate the information of interest. Thus, even if there is a detection delay of a few hours, the defenders should be able to stop an attack that would otherwise pass undetected by traditional security solutions.

The proposed model was evaluated in the following scenarios:

a. An attack against an Internet-facing architecture, which simulated the information-gathering attempts and malicious actions that attackers frequently perform while trying to compromise a target and/or gain foothold on a network.

b. An internal attack, simulating the attacker's actions on an internal network, while trying to locate and exfiltrate the data of interest.

The evaluation was performed by four expert penetration testing teams, which were asked to achieve the objectives of each scenario while taking every possible precaution to avoid detection. For both scenarios, the proposed model successfully detected all malicious actions with a minimum delay, generating a negligible number of false positive alerts. The robust effectiveness does not come without a price, however. The model requires skilled individuals to configure it and set it up, taking into account the network architecture, the operating system(s), applications and services, as well as the way users interact with those. Furthermore, the defenders need to have the required technical skills to limit or block an active attack when detected by the model. This can be challenging for small organizations that do not have a dedicated security team. Nevertheless, considering the inability of current security solutions to address sophisticated threats, the benefits of adopting such a model as a complementary mechanism for increasing detection robustness are significant.

7.2 Publications

Our research work related to this PhD thesis has been published in peer-reviewed journals, conferences and book chapters, namely:

- **Publications in peer-reviewed, academic journals:**

1. Virvilis N., Mylonas A., Tsalis N., Gritzalis D., "Security Busters: Web Browser security vs. rogue sites", Computers & Security, 2015 (to appear)
2. Virvilis N., Serrano O., Dandurand L., "Big Data analytics for sophisticated attack detection", ISACA Journal, Vol. 3, 2014

- **Publications in peer-reviewed, international conferences:**

1. Virvilis N., Tsalis N., Mylonas A., Gritzalis D., "Mobile devices: A phisher's paradise", in Proc. of the 11th International Conference on Security and Cryptography (SECRYPT-2014), pp. 79-87, ScitePress, Austria, August 2014

2. Virvilis N., Vanautgaerden B., Serrano O. S. (2014, June). “Changing the game: The art of deceiving sophisticated attackers.” in Proc. of the 6th International Conference on Cyber Conflict (CYCON-014), pp. 87-97, Estonia, June 2014
3. Virvilis N., Gritzalis D., “Trusted Computing vs. Advanced Persistent Threats: Can a defender win this game?”, in Proc. of 10th IEEE International Conference on Autonomic and Trusted Computing (ATC-2013), pp. 396-403, IEEE Press, Italy, December 2013
4. Virvilis N., Gritzalis D., “The Big Four - What we did wrong in Advanced Persistent Threat detection?”, in Proc. of the 8th International Conference on Availability, Reliability and Security (ARES-2013), pp. 248-254, IEEE, Germany, September 2013
5. Kandias M., Virvilis N., Gritzalis D., "The Insider Threat in Cloud Computing", in Proc. of the 6th International Conference on Critical Infrastructure Security (CRITIS-2011), pp. 93-103, Springer (LNCS 6983), 2013
6. Virvilis N., Dritsas S., Gritzalis D., “Secure Cloud Storage: Available Infrastructure and Architecture Review and Evaluation”, in Proc. of the 8th International Conference on Trust, Privacy & Security in Digital Business (TRUSTBUS-2011), Furnell S., et al. (Eds.), pp 74-85, LNCS-6863, Springer, France, August 2011
7. Virvilis N., Dritsas S., Gritzalis D., “A cloud provider-agnostic secure storage protocol”, in Proc. of the 5th International Conference on Critical Information Infrastructure Security (CRITIS-2010), Wolthusen S., et al. (Eds.), pp. 104-115, LNCS-6712, Springer, Greece, September 2010
8. Kandias M., Mylonas A., Virvilis N., Theoharidou M., Gritzalis D., “An Insider Threat Prediction Model”, in Proc. of the 7th International Conference on Trust, Pri-vacy, and Security in Digital Business (TrustBus-2010), pp. 26-37, Lopez J., et al. (Eds.), LNCS-6264, Springer, Spain, August 2010

- **Publications in peer-reviewed book chapters:**

1. Tsalis N., Virvilis N., Mylonas A., Apostolopoulos A., Gritzalis D., “Browser Blacklists: A utopia of phishing protection”, in Security and Cryptography, M. Obaidad and A. Holzinger (Eds.), Lecture Notes (CCIS), Springer, 2015 (to appear)

7.4 Future Work

Future work should focus on the likely benefits of introducing psychological profiling, as presented in Chapter 4, to complement the proposed APT detection model. This could potentially allow the detection of skillful insiders, who, owing to their thorough knowledge of the infrastructure, are aware of all the attack indicators (including the deception-based ones) and thus are able to evade them.

On the more practical side, future work should investigate ways to automate the model's deployment, and especially the deployment of deception-based indicators. Finally, additional focus should be given to mobile platforms by collecting OS level statistics and implementing deception indicators in order to increase visibility.

7.5 Concluding remarks

APT is a complex, multi-dimensional problem. In contrast with opportunistic attackers, APT are highly motivated and skillful. The use of zero-day exploits and sophisticated tools – which are often developed for attacking a specific target – enable them to evade detection by traditional security methods. In most of the cases, they are working for, or supported by, a government or a powerful organization, which gives them access to significant financial and intelligence resources as well as immunity from legal actions against them.

Our current security solutions, even those that are considered state-of-the-art, have failed over and over again to address such sophisticated attacks. The prevention mentality on which the security industry has been based since the 1980s and the notion of a secure external network and a trusted internal one are outdated and unrealistic for application against modern threats.

A significant change in the way we defend our networks is required, and entails accepting the fact that sophisticated attackers will eventually subvert our preventive defenses. We have to invest in developing robust detection techniques. Deception is an invaluable tool for detecting malicious actions in an infrastructure, on both the network and operating system level. Thus, it is a key component of the proposed APT detection model.

The model offers robust detection of sophisticated attacks, by allowing in-depth correlation of multiple deception and anomaly-based attack indicators. It has been evaluated in realistic scenarios, which demonstrated its significant superiority against traditional security technologies; however, the purpose of this model is not to replace the existing technologies but to add an additional defensive layer against attacks that manage to evade them.

Finally, this thesis is offered as influence towards a change in the way security professionals, administrators and users defend their assets and as inspiration for further research that will allow stronger defenses to be built.

(This page is intentionally left blank)

Appendix

Mobile Statistics

Table 23 - Desktop browser popularity

Source: http://gs.statcounter.com/ (June-July 2014)

Browser	Use percentage
Chrome	46.03%
Internet Explorer	25.87%
Firefox	20.04%
Safari	4.93%
Opera	1.3%
Other	1.84%

Table 24 - Browser popularity on Android

Based on the number of installs from Google Play (as of Jul 2014)

Browser	Million installs
Opera Mini	100-500
Chrome Mobile	500-1000
Firefox Mobile	50-100
Opera Mobile	50-100
Android Browser	In all browsers

Table 25 - Default CIF feeds

http://aper.svn.sourceforge.net/svnroot/aper/phishing_reply_addresses
http://data.phishtank.com/data/online-valid.json.gz
http://malc0de.com/rss
http://mirror3.malwaredomains.com/files/bulk_registrars.zip
http://mirror3.malwaredomains.com/files/domains.zip
http://mirror3.malwaredomains.com/files/url_shorteners.zip
http://reputation.alienvault.com/reputation.data
http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
https://feodotracker.abuse.ch/blocklist/?download=badips
https://feodotracker.abuse.ch/blocklist/?download=domainblocklist
https://feodotracker.abuse.ch/blocklist/?download=ipblocklist
https://spyeyetracker.abuse.ch/blocklist.php?download=domainblocklist
https://spyeyetracker.abuse.ch/blocklist.php?download=ipblocklist
https://spyeyetracker.abuse.ch/monitor.php?rssfeed=binaryurls
https://spyeyetracker.abuse.ch/monitor.php?rssfeed=configurls
https://spyeyetracker.abuse.ch/monitor.php?rssfeed=dropurls
https://zeustracker.abuse.ch/blocklist.php?download=domainblocklist
https://zeustracker.abuse.ch/blocklist.php?download=ipblocklist
https://zeustracker.abuse.ch/monitor.php?urlfeed=binaries
https://zeustracker.abuse.ch/monitor.php?urlfeed=configs
https://zeustracker.abuse.ch/monitor.php?urlfeed=dropzones
http://www.malwaredomainlist.com/updatescsv.php
http://www.mirc.com/servers.ini
http://www.spamhaus.org/drop/drop.lasso
http://www.spamhaus.org/drop/edrop.txt
http://dragonresearchgroup.org/insight/sshpwauth.txt
http://dragonresearchgroup.org/insight/vncprobe.txt
http://www.openbl.org/lists/date_all.txt

Table 26 - Percentage of URLs that were blacklisted

Browser	Blacklisted	
	Results Q2 2014 (n=1400)	**Results Q1 2014 (n=5651)**
Safari Mobile (iOS)	38.7%	75%
Firefox Mobile (Android)	85.4%	85.3%
Opera Mobile (Android)	75.9%	78.7%
Firefox (Windows)	86.7%	94.9%
Chrome (Windows)	93%	94.5%
Opera (Windows)	77.9%	87.1%
IE (Windows)	48.4%	64.6%

Table 27 - Percentage of false negatives

Browser	False negatives	
	Results Q2 2014	**Results Q1 2014**
Safari Mobile (iOS)	26.4%	13.3%
Firefox Mobile (Android)	3.4%	3%
Opera Mobile (Android)	7.9%	1.5%
Firefox (Windows)	5.9%	2%
Chrome (Windows)	1.3%	1.7%
Opera (Windows)	8.4%	1.4%
IE (Windows)	9.9%	6.7%

Table 28 - Percentage of URLs that were manually verified as non-phishing

Browser	Non-phishing	
	Results Q2 2014	**Results Q1 2014**
Safari Mobile (iOS)	34.9%	11.5%
Firefox Mobile (Android)	11.1%	11.7%
Opera Mobile (Android)	16.3%	19.8%
Firefox (Windows)	7.3%	3%
Chrome (Windows)	5.7%	3.8%
Opera (Windows)	13.7%	11.5%
IE (Windows)	41.7%	28.7%

Table 29 - Malicious file detection based on the hash of the samples

AV Engine Detection %	% of Malware	Cumulative Percent	AV Engine Detection %	% of Malware	Cumulative Percent
6	3.6	3.6	44	1	57.1
7	1	4.6	54	0.5	57.7
9	2	6.6	56	1	58.7
10	0.5	7.1	60	0.5	59.2
11	1	8.2	61	0.5	59.7
12	1.5	9.7	63	1	60.7
13	1	10.7	65	1	61.7
14	0.5	11.2	67	2.6	64.3
15	2.6	13.8	68	1	65.3
17	2.6	16.3	69	1.5	66.8
19	1	17.3	70	1	67.9
20	1	18.4	71	0.5	68.4
22	1	19.4	72	0.5	68.9
24	2.6	21.9	73	0.5	69.4
25	0.5	22.4	74	3.1	72.4
26	4.1	26.5	75	1	73.5
27	0.5	27	76	3.1	76.5
28	1	28.1	77	1	77.6
29	0.5	28.6	78	3.1	80.6
30	4.1	32.7	79	2	82.7
31	4.1	36.7	80	1	83.7
32	1	37.8	81	3.6	87.2
33	4.1	41.8	82	1.5	88.8
34	0.5	42.3	83	0.5	89.3
35	1	43.4	85	1.5	90.8
36	0.5	43.9	86	0.5	91.3
37	5.6	49.5	87	4.6	95.9
38	2	51.5	88	0.5	96.4
39	2.6	54.1	89	2	98.5
40	1	55.1	90	1	99.5
43	1	56.1	91	0.5	100

Table 30 - Malicious file detection based on file analysis (submission of the file)

AV Engine Detection %	% of Malware	Cumulative Percent	AV Engine Detection %	% of Malware	Cumulative Percent
19	0.4	0.4	48	29.3	53.7
25	0.4	0.9	49	13.5	67.2
26	0.4	1.3	50	9.6	76.9
33	0.4	1.7	51	2.2	79
34	0.4	2.2	52	4.8	83.8
35	0.9	3.1	53	3.1	86.9
38	0.4	3.5	54	4.8	91.7
43	0.4	3.9	55	0.9	92.6
44	0.9	4.8	56	3.9	96.5
45	1.3	6.1	57	2.2	98.7
46	5.2	11.4	61	0.9	99.6
47	13.1	24.5	62	0.4	100

Table 31 - VirusTotal AV Engines

AVG	DrWeb	NANO-Antivirus
AVware	ESET-NOD32	Norman
Ad-Aware	Emsisoft	Panda
AegisLab	F-Prot	Qihoo-360
Agnitum	F-Secure	Rising
AhnLab-V3	Fortinet	SUPERAntiSpyware
AntiVir	GData	Sophos
Antiy-AVL	Ikarus	Symantec
Avast	Jiangmin	Tencent
Baidu-International	K7AntiVirus	TheHacker
BitDefender	K7GW	TotalDefense
Bkav	Kaspersky	TrendMicro
ByteHero	Kingsoft	VBA32
CAT-QuickHeal	Malwarebytes	VIPRE
CMC	McAfee	ViRobot
ClamAV	McAfee-GW-Edition	Zillya
Commtouch	MicroWorld-eScan	Zoner
Comodo	Microsoft	nProtect

Table 32 - VirusTotal URL reputation providers

ADMINUSLabs	Kaspersky	SpyEyeTracker
AegisLab	Malc0de	StopBadware
AlienVault	Malekal	Sucuri
Antiy-AVL	Malware	Tencent
AutoShun	MalwareDomainList	ThreatHive
Avira	MalwarePatrol	Trustwave
BitDefender	Malwarebytes	URLQuery
C-SIRT	Malwared	VX
CLEAN	Netcraft	Web
CRDF	OpenPhish	Websense
Comodo	Opera	Webutation
CyberCrime	PalevoTracker	Wepawet
Dr.Web	ParetoLogic	Yandex
ESET	Phishtank	ZCloudsec
Emsisoft	Quttera	ZDB
Fortinet	Rising	ZeusTracker
FraudSense	SCUMWARE.org	malwares.com
G-Data	SecureBrain	zvelo
Google	Sophos	
K7AntiVirus	Spam404	

Source Code

Snippet 1 – External Scenario

```
#!/usr/bin/env python
"""
Listens on TCP port 31337 and upon connection ask for the “flag”.
The submitted value is checked against two hashes. One is the word available on the fake persona’s LinkedIn page, while the second can be accessed by exploiting a command injection vulnerability on the web app.
"""
import socket, hashlib, datetime

hashes=["51db1cd987251beffb9c666761a9b062","9724f396135aa0bfe71d172bd76254a9"]

host = ''
port = 31337
backlog = 5
size = 1024
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((host,port))
s.listen(backlog)
while 1:
    client, address = s.accept()
    client.send("\nFlag:")
    data = client.recv(size)
    if data:
        h=hashlib.md5(data.strip())
        if h.hexdigest() in hashes:
            client.send("\n***You did it! End of  CTF1***\n")
            client.send("\nPlease email us with a brief explanation on how you did it, when (date/time) your started and your IP address. \n\n")
            localtime = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            f=open('/home/ubuntu/winners.txt','a')
            f.write(localtime)
            f.write(str(address))
            if h.hexdigest()== hashes[0]:
                f.write(" LinkedIn hash!\n")
            else:
                f.write(" Command Injection hash!\n")
            f.write("****\n")
            f.close()
        else:
            client.send("No, try again!\n")
    client.close()
```

Snippet 2 – Internal Scenario, honeyd configuration

```
create default
set default default tcp action block
set default default udp action block
set default default icmp action allow

create windows2008
set windows2008 personality "Microsoft Windows Server 2008 R2"
set windows2008 default tcp action reset
add windows2008 tcp port 135 open
add windows2008 tcp port 139 open
add windows2008 tcp port 445 open
add windows2008 tcp port 80 "sh /usr/share/honeyd/scripts/win32/web.sh"
add windows2008 tcp port 3389 open

create windows7
set windows7 personality "Microsoft Windows 7 or Windows Server 2008 R2"
set windows7 default tcp action reset
add windows7 tcp port 135 open
add windows7 tcp port 139 open
add windows7 tcp port 445 open

create linux
set linux personality "Linux 3.0 - 3.1"
set linux default tcp action reset
add linux tcp port 80 "sh /usr/share/honeyd/scripts/linux/httpd $ipsrc $sport $ip dst $dport"
add linux tcp port 22 "sh /usr/share/honeyd/scripts/linux/ssh.sh $ipsrc $sport $ipdst $dporta"
add linux tcp port 23 "sh /usr/share/honeyd/scripts/linux/telnetd.sh  $ipsrc $sport $ipdst $dport"

create nas
set nas personality "Netgear ReadyNAS 3200 NAS device (Linux 2.6)"
set nas default tcp action reset
add nas tcp port 445 open
add nas tcp port 80 open

set linux ethernet "00:11:32:28:84:f7"
set nas ethernet "00:00:24:b2:b1:74"
set windows2008 ethernet "00:00:25:f0:c9:11"
set windows7 ethernet "00:00:25:f0:c1:11"
set windows7 ethernet "00:00:25:f0:c1:12"

bind 172.16.10.2 nas
bind 172.16.10.20 windows2008
bind 172.16.10.120 linux
bind 172.16.11.6 windows7
bind 172.16.11.7 windows7
```

Snippet 3 – Internal Scenario, LogParser.py

```
import mmap, os, time, sys, sqlite3
import datetime
import contextlib
import argparse
import subprocess
'''
Required: https://github.com/williballenthin/python-evtx.git
Copy evtxdump.py to the same folder as this
'''
import xml.etree.ElementTree as ET
XMLdata=""

#-------- configuration----------
data_age_in_mins = 120
honeyusers=["hightechadmin","backupuser"]
honeyfiles=["C:\Shares\\For review\\old_patent.doc", "C:\\Shares\\backups\\old.zip"]
whitelistedAccounts=["FILESERVER$"] #Accounts for which access to honeyfiles won't be reported. (e.g. For a backup process which accesses all the files)
LinuxAuthLog = '/var/log/auth.log'
#-------- configuration----------

NowEpoch = (datetime.datetime.today() - datetime.datetime(1970,1,1)).total_seconds() #This will import to the db all data captured in the last data_age_in_mins minutes.
FormatTime = '%Y-%m-%d %H:%M:%S.%f'

previous_record = "" # Windows creates 4 logon events when a user successfully authenticates. This quick trick keeps only the first

def CreateDB():
  conn = sqlite3.connect("core.db")
  c = conn.cursor()
  c.execute("CREATE TABLE logins (username TEXT, systemname TEXT, ip TEXT, timeepoc TEXT, authgood INTEGER, weekends INTEGER, honeyaccount INTEGER)")
  c.execute("CREATE TABLE honeyfiles (username TEXT, systemname TEXT, timeepoc TEXT, honeyfile TEXT)")
  c.execute("CREATE TABLE suspectIP (ip TEXT, timeepoc TEXT, numberofoffences INTEGER, honeypotconnection INTEGER)")
  c.close()

def WriteRecordsLoginDB(username, systemname, ip, timeepoc, authgood, weekends, honeyaccount):
  global conn, c
  params = (username, systemname, ip, timeepoc, authgood, weekends, honeyaccount)
```

```
  c = conn.cursor()
  c.execute("INSERT INTO logins(username, systemname, ip, timeepoc, authgood, weekends, honeyaccount) VALUES (?,?,?,?,?,?,?)", params)
  conn.commit()
  c.close()

def WriteRecordsFilesDB(username, systemname, timeepoc, honeyfile):
  global conn, c
  params = (username, systemname, timeepoc, honeyfile)
  c = conn.cursor()
  c.execute("INSERT INTO honeyfiles(username, systemname, timeepoc, honeyfile) VALUES (?,?,?,?)", params)
  conn.commit()
  c.close()

def ParseLinuxLogs(file):
    LinuxFormatTime = '%Y %b %d %H:%M:%S'
    year= datetime.date.today().year #linux SSH logs don't include year
    AuthFromIp={} # Holds the number of failed auth attempts per IP
    if os.path.exists(file):
        #for failed SSH authentication attempts
        proc = subprocess.Popen(["grep", "Invalid user", file], stdout=subprocess.PIPE) #get list of failed SSH authentications
        data=proc.stdout.readlines()
        for line in data:
            record=line.split()
            eventdate = datetime.datetime.strptime(str(year) + " " + " ".join(record[:3]),LinuxFormatTime)
            eventdateEpoch = (eventdate - datetime.datetime(1970,1,1)).total_seconds()
            if eventdate.weekday() > 4:
                weekends=1 #weekday() > 4 is Saturday or Sunday
            else:
                weekends=0
            if record[7] in honeyusers:
                honeyaccount=1
                print "SEVERE: IP:", record[9], "tried to connect with honeyaccount:", record[7], "on", eventdate
            else:
                honeyaccount=0
            WriteRecordsLoginDB(record[7], record[3], record[9], eventdateEpoch, 0, weekends, honeyaccount)

        #Failed password for user from 10.0.1.51 port 54337 ssh2
        proc = subprocess.Popen(["grep", "Failed password for", file], stdout=subprocess.PIPE)
        data=proc.stdout.readlines()
        for line in data:
            record=line.split()
            eventdate = datetime.datetime.strptime(str(year) + " " + " ".join(record[:3]),LinuxFormatTime)
            eventdateEpoch = (eventdate - datetime.datetime(1970,1,1)).total_seconds()
```

```
            if eventdate.weekday() > 4:
                weekends=1 #weekday() > 4 is Saturday or Sunday
            else:
                weekends=0
            if record[8] in honeyusers:
                honeyaccount=1
                print "SEVERE: IP:", record[10], "tried to connect with honeyaccount:", record[8], "on", eventdate
            else:
                honeyaccount=0
            WriteRecordsLoginDB(record[8], record[3], record[10], eventdateEpoch, 0, weekends, honeyaccount)

    else:
        print "Warning. Linux auth log could not be parsed."
        return 1

def ParseWindowsLogs(file):
  global XMLdata, previous_record, previous_epoch
  with open(file) as f:
    XMLdata = f.read()

  root = ET.fromstring(XMLdata)
  for child in root:
    try:
      if child[0][1].text=="4624": # successful logon event 4624
        if (child[1][8].text == "3" or child[1][8].text == "2" or child[1][8].text == "10") and not child[1][5].text.endswith("$") and not child[1][5].text=="ANONYMOUS LOGON" :  #logon type 3(network) or 2 (interactive) or 10 RemoteInteractive (remote desktop). Second part ignores computer and anonymous logins
          eventdate = datetime.datetime.strptime(child[0][7].get("SystemTime"), FormatTime)
          eventdateEpoch = (eventdate - datetime.datetime(1970,1,1)).total_seconds()
          if (previous_record != child[1][5].text + child[0][12].text + str(int(eventdateEpoch))): #if you see succesful auth from the same user to the same machine in the same second, ignore...
            if eventdate.weekday() > 4:
              weekends=1 #weekday() > 4 is Saturday or Sunday
            else:
              weekends=0
            if (NowEpoch - eventdateEpoch) < (data_age_in_mins * 3600): # if not too old
              if child[1][5].text in honeyusers:
                honeyaccount=1
                print "SEVERE: IP", child[1][18].text, "tried to connect with honeyaccount:", child[1][5].text, "on", eventdate
              else:
                honeyaccount=0
              previous_record = child[1][5].text + child[0][12].text + str(int(eventdateEpoch))
```

```
            WriteRecordsLoginDB(child[1][5].text, child[0][12].text, child[1][18].text, eventdateEpoch, 1, weekends, honeyaccount)

      elif child[0][1].text=="4771": #failed logon event 4771
       if child[0][4].text=="14339" and not child[1][0].text.endswith("$"): #kerberos auth task. Second ignores computer accounts
          eventdate = datetime.datetime.strptime(child[0][7].get("SystemTime"), FormatTime)
          eventdateEpoch = (eventdate - datetime.datetime(1970,1,1)).total_seconds()
          if child[1][0].text in honeyusers:
           honeyaccount=1
           print "SEVERE: IP", child[1][6].text[7:], "workstation", child[0][12].text, "tried to connect with honeyaccount:", child[1][0].text, "on", eventdate
          else:
           honeyaccount=0
          WriteRecordsLoginDB(child[1][0].text, child[0][12].text, child[1][6].text[7:], eventdateEpoch, 0, 0, honeyaccount)
      elif child[0][1].text=="4625": #failed logon event 4625 (to catch runas commands)
          if child[1][5].text: #if we have a username value
           eventdate = datetime.datetime.strptime(child[0][7].get("SystemTime"), FormatTime)
           eventdateEpoch = (eventdate - datetime.datetime(1970,1,1)).total_seconds()
           if child[1][5].text in honeyusers:
              honeyaccount=1
              print "SEVERE: IP", child[1][19].text, "workstation", child[1][13].text, "tried to connect with honeyaccount:", child[1][5].text, "on", eventdate #have added the workstation part, as for failed auth attempts for non existing users the ip is captured as '-'
           else:
              honeyaccount=0
          WriteRecordsLoginDB(child[1][5].text, child[1][13].text, child[1][19].text, eventdateEpoch, 0, 0, honeyaccount)
      elif child[0][1].text=="4663": #filesystem access
          if not (child[1][1].text in whitelistedAccounts) and (child[1][6].text in honeyfiles):#if the useraccount is not whitelisted (e.g. Antivirus/Backup service)
           eventdate = datetime.datetime.strptime(child[0][7].get("SystemTime"), FormatTime) # warning: Windows will log it in UTC, regardless of the time configured on the windows system
           eventdateEpoch = (eventdate - datetime.datetime(1970,1,1)).total_seconds()
           WriteRecordsFilesDB(child[1][1].text, child[0][12].text, eventdateEpoch, child[1][6].text)
           print "SEVERE: User:", child[1][1].text, "accessed honeyfile", child[1][6].text, "on", eventdate
    except IndexError:
      pass

def main():
    global XMLdata, conn, c
    parser = argparse.ArgumentParser(
        description="Dump a binary EVTX file into XML.")
    parser.add_argument("evtx", type=str,
```

```
                help="Path to the Windows EVTX event log file")
    args = parser.parse_args()

    if os.path.exists(args.evtx):
      xmlfile = time.strftime("%Y%m%d%H%M%S")
      xmlfile = xmlfile + ".xml"
      command='python evtxdump.py ' + args.evtx + ' > ' + xmlfile
      os.system(command) #XML file created
      #Prepare the DB
      if not os.path.exists('core.db'): CreateDB()
      conn = sqlite3.connect('core.db')
      ParseWindowsLogs(xmlfile)
      ParseLinuxLogs(LinuxAuthLog)
      #clean up
      os.remove(xmlfile)

    else:
      print args.evtx, "does not exist. Exiting\n"


if __name__ == "__main__":
    main()
```

Snippet 4 – Internal Scenario, NetworkParser.py

```
from netaddr import IPNetwork, IPAddress
import os, sys, time, datetime, sqlite3, commands

#-------- configuration----------

CLIENT_NET='172.16.11.0/24' #Client network range
SERVER_NET='172.16.10.0/24' #Server network range
#Alert Threshold
w2wlimit=5 #connections between workstations
w2slimit=5 #connections between a single workstation and multiple servers
s2elimit=1 #connections originating from server to an external network
tcptimelimit=600 #alert on TCP connections that last more than 600 seconds (10 min)
tcpsizelimit=2000000000 #alert on an internal system uploaded more than 2 GB 2000000000
BroConnLog='/nsm/bro/logs/current/conn.log'
honeydconnlog='/var/log/honeypot/connectionsLoopback.log'
honeypotIP=["172.16.10.2, 172.16.10.20, 172.16.10.120, 172.16.11.6, 172.16.11.7"]
path_brocut='/nsm/bro/bin/bro-cut'
#-------- configuration----------

w2w={} #holds the connections a workstation has initiated to other workstations
w2s={} #holds the connections a workstation has initiated to servers
s2e={} #holds the connections a server has initiated to an external network
conn2honeypot={}
conndur={} # holds the TCP connections that have lasted more than tcptimelimit
connsize={} # holds the TCP connections that have transferred more than tcpsizelimit
suspects={}

def writetoDB():
    '''This function writes the results in the DB to be parsed by the correlation module'''
    honeypotconnection=0
    if not os.path.exists('core.db'):
        print "Error! Database file not found. Exiting"
        sys.exit(1)
    NowEpoch = (datetime.datetime.today() - datetime.datetime(1970,1,1)).total_seconds()
    conn = sqlite3.connect('core.db')
    c = conn.cursor()
    for ip in suspects:
        params = (ip, NowEpoch, str(int(suspects[ip])), honeypotconnection)
        c.execute("INSERT INTO suspectIP(ip, timeepoc, numberofoffences, honeypotconnection) VALUES
(?,?,?,?)", params)
    conn.commit()
    # add honeypot connections to the DB
    honeypotconnection=1
```

```
    for ip in conn2honeypot:
        NumberOfHoneypots=len(conn2honeypot[ip].split())
        params = (ip, NowEpoch, NumberOfHoneypots, honeypotconnection) #NumberOfHoneypots holds the number of honeypots the ip has connected to
        c.execute("INSERT INTO suspectIP(ip, timeepoc, numberofoffences, honeypotconnection) VALUES (?,?,?,?)", params)
    conn.commit()
    c.close()

def ReportIP(ip):
    '''Goes through the list of malicious IPs and calculates the number of times an IP has triggered an alert'''
    if ip in suspects:
        suspects[ip] = suspects[ip] + 1
    else:
        suspects[ip] = 1



def honeypots():
    ips=" ".join(honeypotIP).replace(",", '|')
    badguys={}
    threshold=2 # alert where more than X connections to the honeys are done by the same IP
    cmd="grep -E " + "\"" + ips + "\"" + " " + honeydconnlog
    (status,output)=commands.getstatusoutput('grep -E "10.0.1.40|10.0.1.41|10.0.1.42" /var/log/honeypot/connectionsLoopback.log')
    s=output.split('\n')

    for line in s:
        if len(line.split()) == 0:
            print "No events, exiting..."
            sys.exit(0)
        bad=line.split()[3] #field with sounce IP
        if len(bad) > 15 or ';' in bad or '|' in bad or '&' in bad : #lame command injection check
            print "Error parsing IP... Exiting"
            sys.exit(1)
        if bad in badguys:
            badguys[bad]= badguys[bad] + 1
        else:
            badguys[bad]=1
    print badguys

def analyzeresults():
    '''This function parses the anomalous connections and checks to see if they have exceeded the alert threshold. If yes they are reported as malicious'''
    for ip in w2w:
```

```
        if len(w2w[ip]) > w2wlimit:
            ReportIP(ip)
    for ip in w2s:
        if len(w2s[ip]) > w2slimit:
            ReportIP(ip)
    for ip in s2e:
        if len(s2e[ip]) > s2elimit:
            ReportIP(ip)
    for ip in conndur:
        ReportIP(ip)
    for ip in connsize:
        ReportIP(ip)

def printresults():
    '''This Function prints the analysis results'''
    print "\n"
    print "Connection between workstations"
    print "-" * 50
    for ip in w2w:
        print "*", ip, "Connected to", str(len(w2w[ip].split())), "workstations", "*"
        print "*", ip, "Connected to workstations", w2w[ip], "*"
    print "-" * 50

    print "\n"
    print "Connections from workstations to servers"
    print "-" * 50
    for ip in w2s:
      #print ip, "Connected to", str(len(w2s[ip].split())), "servers", "*"
      print "*", ip, "Connected to servers", w2s[ip], "*"
    print "-" * 50

    print "\n"
    print "Connection from servers to external IP addresses"
    print "-" * 50
    for ip in s2e:
      #print ip, "Connected to", str(len(s2e[ip].split())), "External IPs", "*"
      print "*", ip, "Connected to External IP:", s2e[ip], "*"
    print "-" * 50

    print "\n"
    print "TCP connections that lasted more than", str(tcptimelimit), "seconds"
    print "-" * 50
    for ip in conndur:
      print "From", ip, "to", conndur[ip]
    print "-" * 50
```

```
  print "\n"
  print "Connections which uploaded more than", str(tcpsizelimit), "bytes"
  print "-" * 50
  for ip in connsize:
    print "From", ip, "to", connsize[ip]
  print "-" * 50

  print "\n"
  print "Connections to honeypots"
  print "-" * 50
  for ip in conn2honeypot:
    print ip, "Connected to honeypot", conn2honeypot[ip]
  print "-" * 50
  print "\n"


def networkstats(file):
  '''This function parses bro conn.logs and identifies anomalies based on our proposed attack indicators. The anomalous connections are stored and processed by analyzeresults function'''
  with open(file) as f:  # zcat conn.log | bro-cut -d id.orig_h id.orig_p id.resp_p id.resp_h duration orig_bytes resp_bytes
    data=f.readlines()

  honeypotlist=honeypotIP[0].replace(" ","").split(",") #convert to a form ['ip1', 'ip2', 'ip3']

  for connection in range(len(data)):
    SRC=data[connection].split()[0]
    DST=data[connection].split()[3]
    #connection between workstations
    if IPAddress(SRC) in IPNetwork(CLIENT_NET) and IPAddress(DST) in IPNetwork(CLIENT_NET):
      if SRC in w2w: #if already reported
        if not DST in w2w[SRC]:
          w2w[SRC] = w2w[SRC]+ " " + DST
      else:
        w2w[SRC] = DST
    #connection from workstation to servers
    elif IPAddress(SRC) in IPNetwork(CLIENT_NET) and IPAddress(DST) in IPNetwork(SERVER_NET):
      if SRC in w2s: #if already reported
        if not DST in w2s[SRC]:
          w2s[SRC] = w2s[SRC]+ " " + DST
      else:
        w2s[SRC] = DST
    #connections originating from servers to an external network
```

```
    elif IPAddress(SRC) in IPNetwork(SERVER_NET): #if its a server address
     if (IPAddress(DST) not in IPNetwork(SERVER_NET)) and (IPAddress(DST) not in IPNetwork(CLIENT_NET)): # and the DST is not an internal segment
       #print "*"*8, DST
       if SRC in s2e: #if already reported
        if not DST in s2e[SRC]:
          s2e[SRC] = s2e[SRC]+ " " + DST
       else:
        s2e[SRC] = DST

    if data[connection].split()[4] != "-":
     if float(data[connection].split()[4]) > tcptimelimit: #if the TCP connection lasted more than the alert limit
       if SRC in conndur:
        if not DST in conndur[SRC]:
          conndur[SRC] = conndur[SRC]+ " " + DST


       else:
        conndur[SRC] = DST
    if IPAddress(SRC) in IPNetwork(CLIENT_NET) or IPAddress(SRC) in IPNetwork(SERVER_NET): # if the source is an internal system
     if (IPAddress(DST) not in IPNetwork(SERVER_NET)) and (IPAddress(DST) not in IPNetwork(CLIENT_NET)): #if data is uploaded from an internal system to an external one
       if data[connection].split()[5] != "-":
        if float(data[connection].split()[5]) > tcpsizelimit: #if the TCP connection uploaded more than tcpsizelimit
          if SRC in connsize:
            if not DST in connsize[SRC]:
              connsize[SRC] = connsize[SRC]+ " " + DST
          else:
            connsize[SRC] = DST
    #Check for Honeypot Access (basic only checks for connection attempts. Even if an IP connects to multiple ports on a honeypot only, this will count as a single offence. For more info honeydlogs needs to be parsed.
    if DST in honeypotlist:
       if SRC in conn2honeypot:
          if DST not in conn2honeypot[SRC]:# Add only if we haven't seen this connection before
             conn2honeypot[SRC] = conn2honeypot[SRC] + " " + DST
             print "SEVERE: IP", SRC, "connected to honeypot", DST
       else:
          conn2honeypot[SRC] = DST
          print "SEVERE: IP", SRC, "connected to honeypot", DST


def main():
   if os.path.exists(BroConnLog):
      outputfile = time.strftime("%Y%m%d%H%M%S")
      outputfile = "Bro" + outputfile + ".txt"
```

```
        command="cat " + BroConnLog + " |" + path_brocut + " -d id.orig_h id.orig_p id.resp_p id.resp_h duration
orig_bytes resp_bytes > " + outputfile
        os.system(command)
        networkstats(outputfile)
        printresults()
        analyzeresults()
        writetoDB()
        #clean up temp files
        os.remove(outputfile)
    else:
        print "Warning! Network statistics not available."

if __name__ == "__main__":
    main()
```

Snippet 5 – Internal Scenario, Correlation.py

```
import os, sys, sqlite3, operator

#-------- configuration----------
SuccessfulLoginLimit = 5 #if more than 5 successful authentication attempts in 24h from the same user alert
FailedLoginLimit = 5 #if more than 5 failed authentication in 24h from the same IP

TRLow=1      #The TR weight for TR=L
TRMedium=3   #The TR weight for TR=M
TRHigh=6     #The TR weight for TR=H
TRCritical=9 #The TR weight for TR=C
#-------- configuration----------

MalUser={}
MalIP={}

def ReportIP(ip, TR):
   print "Reported IP", ip, "with TR", TR
   if ip in MalIP: #if this ip has already been reported as malicious
      MalIP[ip] = MalIP[ip] + TR
   else:
      MalIP[ip] = TR

def ReportUser(username, TR):
   print "Reported Username", username, "with TR", TR
   if username in MalUser: #if this username has already been reported as malicious
      MalUser[username] = MalUser[username] + TR
   else:
      MalUser[username] = TR

def ATRCalc():
   conn = sqlite3.connect('core.db')
   c = conn.cursor()

   #Multiple, successful logins from an authorized account in the last 24h to different systems.
   c.execute("SELECT username, count(DISTINCT ip) FROM logins WHERE (SELECT strftime('%s','now') -
timeepoc < 86400) AND authgood = 1 GROUP BY username") #change to <!!!
   data = c.fetchall()
   for record in data:
      username=record[0]
      n=record[1] #number of successful authentications to different systems
      if n > SuccessfulLoginLimit:
         ReportUser(username,TRMedium) # we don't care about the exact number. If it is above the limit it will get
reported
```

```
    #Multiple login failures from one or more ips in the last 24h.
    c.execute("SELECT ip, count(ip) FROM logins WHERE (SELECT strftime('%s','now') - timeepoc < 86400)
AND authgood = 0 AND honeyaccount = 0 AND ip != '-' GROUP BY username")#change to <!!!
    data = c.fetchall()
    for record in data:
        ip=record[0]
        n=record[1] #number of failed auth attempts
        if n > FailedLoginLimit:
            ReportIP(ip,TRMedium)

    #Succesful authentications during irregular business hours and/or non-working days
    c.execute("SELECT DISTINCT username FROM logins WHERE weekends = 1 GROUP BY username")
    data = c.fetchall()
    for username in data:
        ReportUser(username[0],TRMedium)

    #Access to honey files
    c.execute("SELECT username, count(username) FROM honeyfiles GROUP BY username")
    data = c.fetchall()
    for record in data:
        username = record[0]
        n = record[1] #number of honeyfiles access by user
        TR = TRHigh * n # n times she has accessed honeyfiles times the defined high weight of this action
        ReportUser(username,TR)

    #Login attempts with a honeyaccount
    c.execute("SELECT ip, count(ip) FROM logins WHERE honeyaccount=1 GROUP BY ip")
    data = c.fetchall()
    for record in data:
        ip = record[0]
        n = record[1]
        TR = n * TRCritical
        ReportIP(ip, TR)

    #Get the alerts from network traffic analysis (excluding connections to honeypots)
    c.execute("SELECT ip, sum(numberofoffences) FROM suspectIP WHERE honeypotconnection = 0  GROUP BY
ip")
    data = c.fetchall()
    for record in data:
        ip =record[0]
        n = record[1]
        TR = TRLow * n
        ReportIP(ip,TR)
```

```
    #Get the IPs which have interacted with honeypots
    c.execute("SELECT ip, sum(numberofoffences) FROM suspectIP WHERE honeypotconnection = 1  GROUP BY
ip")
    data = c.fetchall()
    for record in data:
        ip =record[0]
        n = record[1]
        TR = TRHigh * n
        ReportIP(ip,TR)

def main():
    if not os.path.exists('core.db'):
        print "Error! Database file not found. Exiting"
        sys.exit(1)
    ATRCalc() # Calculate the ATR for offending usernames and IPs
    sorted_MalIP = sorted(MalIP.items(), key=operator.itemgetter(1), reverse=True) #Sorting based on largest ATR
    sorted_MalUser = sorted(MalUser.items(), key=operator.itemgetter(1), reverse=True)
    print "*" * 29
    print "*"*10, "Results", "*"*10
    print "*" * 29
    print "\nOffending IP(s):"
    print "-"*29
    for ip in sorted_MalIP:
        print ip[0], "with an ATR of:", ip[1]
    print "-"*29
    print "\nOffending Users(s):"
    print "-"*29
    for username in sorted_MalUser:
        print username[0], "with an ATR of:", username[1]
    print "-"*29

if __name__ == "__main__":
    main
```

References

1. Abrams, R., Barrera, O. & Pathak, J., 2013. *Browser Security Comparative Analysis*, Available at: https://www.nsslabs.com/reports/browser-security-comparative-analysis-phishing-protection.
2. Agudo, I., Nuñez, D., Giammatteo, G., Rizomiliotis, P. & Lambrinoudakis, C., 2011. Cryptography goes to the cloud. In *Secure and Trust Computing, Data Management, and Applications*. Springer, pp. 190–197.
3. Akhawe, D. & Felt, A.P., 2013. Alice in Warningland: A Large-Scale Field Study of Browser Security Warning Effectiveness. In *Usenix Security*. pp. 257–272.
4. Alperovitch, D., 2011. *Revealed: operation shady RAT*, McAfee. Available at: http://www.mcafee.com/us/resources/white-papers/wp-operation-shady-rat.pdf.
5. Andreasson, K.J., 2011. *Cybersecurity: Public Sector Threats and Responses*, CRC Press.
6. Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou II, N. & Dagon, D., 2011. Detecting Malware Domains at the Upper {DNS} Hierarchy. In *{USENIX} Security Symposium*. p. 16.
7. Baggett, M., 2015. Detecting Mimikatz Use On Your Network. Available at: https://isc.sans.edu/forums/diary/Detecting+Mimikatz+Use+On+Your+Network/19311/.
8. Ball, J., Borger, J. & Greenwald, G., 2013. Revealed: how US and UK spy agencies defeat internet privacy and security. *The Guardian*. Available at: http://www.theguardian.com/world/2013/sep/05/nsa-gchq-encryption-codes-security.
9. Banu, M.N. & Banu, S.M., 2013. A Comprehensive Study of Phishing Attacks. *International Journal of Computer Science and Information Technologies*, 4(6), pp.783–786.
10. Barnes, J., 2013. What Bradley Manning Leaked. Available at: http://blogs.wsj.com/washwire/2013/08/21/what-bradley-manning-leaked/ [Accessed January 20, 2014].
11. Bejtlich, R., 2013. *The Practice of Network Security Monitoring: Understanding Incident Detection and Response* 1st ed., No Starch Press.
12. Bejtlich, R., 2010. Understanding the advanced persistent threat. Available at: http://searchsecurity.techtarget.com/magazineContent/Understanding-the-advanced-persistent-threat.
13. Bencsáth, B. et al., 2015. *Duqu 2.0: A comparison to Duqu*, Available at: http://www.crysys.hu/duqu2/duqu2.pdf.
14. Bencsáth, B., Pék, G., Buttyán, L. & Félegyházi, M., 2012a. Duqu: Analysis, detection, and lessons learned. In *ACM European Workshop on System Security (EuroSec)*.
15. Bencsáth, B., Pék, G., Buttyán, L. & Félegyházi, M., 2012b. The cousins of stuxnet: Duqu, flame, and gauss. *Future Internet*, 4(4), pp.971–1003.
16. Berthier, R., Sanders, W.H. & Khurana, H., 2010. Intrusion detection for advanced metering infrastructures: Requirements and architectural directions. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. pp. 350–355.
17. Bertrand, N. & Kelley, M., 2015. "Complete takeover": Israel unleashed one of the world's most sophisticated cyberweapons on the Iran talks. *BusinessInsider*. Available at: http://www.businessinsider.com/israel-spying-on-the-iran-talks-2015-6.
18. Bian, R.M., 2013. *Alice in Battlefield: An Evaluation of the Effectiveness of Various {UI} Phishing Warnings*, Available at: https://www.cs.auckland.ac.nz/courses/compsci725s2c/archive/termpapers/725mbian13.pdf.
19. Bilge, L., Kirda, E., Kruegel, C. & Balduzzi, M., 2011. {EXPOSURE}: Finding Malicious Domains Using Passive {DNS} Analysis. In *NDSS*.
20. Birnbaum, M., 2013. Germany looks at keeping its Internet, e-mail traffic inside its borders. *The Washington Post*. Available at: http://www.washingtonpost.com/world/europe/germany-looks-at-keeping-its-internet-e-mail-traffic-inside-its-borders/2013/10/31/981104fe-424f-11e3-a751-f032898f2dbc_story.html.

21. Bodmer, S., Kilger, M., Carpenter, G. & Jones, J., 2012. *Reverse Deception: Organized Cyber Threat Counter-Exploitation*, McGraw Hill Professional.
22. Bowen, B., Ben Salem, M., Hershkop, S., Keromytis, A. & Stolfo, S., 2013. Designing Host and Network Sensors to Mitigate the Insider Threat. *Journal of Security & Privacy IEEE*, 7, pp.22–29.
23. Bowen, B.M., Salem, M. Ben, Hershkop, S., Keromytis, A.D. & Stolfo, S., 2009. Designing host and network sensors to mitigate the insider threat. *IEEE Security & Privacy*, 7(6), pp.22–29.
24. Bowen, B.M., Salem, M. Ben, Keromytis, A.D. & Stolfo, S.J., 2010. Monitoring technologies for mitigating insider threats. In *Insider Threats in Cyber Security*. Springer, pp. 197–217.
25. Boxcryptor, 2015. Boxcryptor. Available at: https://www.boxcryptor.com.
26. Bradley, T., 2013. *Android Dominates Market Share, But Apple Makes All The Money*, Available at: http://goo.gl/B0ylqi.
27. Bro, 2015. The Bro Network Security Monitor. Available at: https://www.bro.org.
28. Broad, W.J., Markoff, J. & Sanger, D.E., 2011. Israel tests on worm called crucial in Iran nuclear delay. *New York Times*, 15, p.2011.
29. Brown, D.J., Suckow, B. & Wang, T., 2002. A Survey of Intrusion Detection Systems. *Department of Computer Science, University of California, San Diego*.
30. Caballero, J., Grier, C., Kreibich, C. & Paxson, V., 2011. Measuring Pay-per-Install: The Commoditization of Malware Distribution. In *{USENIX} Security Symposium*.
31. Cappelli, D., Moore, A.P., Trzeciak, R.F. & Shimeall, T., 2009. *Common Sense Guide to Prevention and Detection of Insider Threats 3rd Edition*, Available at: http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=50275.
32. Caputo, D., Maloof, M. & Stephens, G., 2009. Detecting insider theft of trade secrets. *IEEE Security and Privacy*, 7(6), pp.14–21.
33. Carrie, D., 2014. Kerry: Snowden a "Coward" and "Traitor." *NBC News*. Available at: http://www.nbcnews.com/politics/first-read/kerry-snowden-coward-traitor-n116366.
34. Chen, T. & Abu-Nimeh, S., 2011. Lessons from stuxnet. *Computer*, 44(4), pp.91–93. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5742014 [Accessed November 26, 2014].
35. Chien, E., OMurchu, L. & Falliere, N., 2012. W32. Duqu: the precursor to the next stuxnet. In *Proc. of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*.
36. CIF, 2014. *Collective Intelligence Framework*, Available at: https://code.google.com/p/collective-intelligence-framework/.
37. Cisco, 2013. *Cisco Annual Security Report*, Available at: http://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2013_ASR.pdf.
38. Claise, B., 2004. Cisco systems NetFlow services export version 9.
39. Claise, B., Trammell, B. & Aitken, P., 2013. Specification of the IP Flow Information Export (IPFIX) protocol for the exchange of flow information. *draft-ietf-ipfix-protocol-rfc5101bis-08 (work in progress)*.
40. Clarke, R. & Knake, R., 2012. *Cyber War: The Next Threat to National Security and What to Do About It*, Ecco.
41. Cloudfogger, 2015. cloudfogger. Available at: https://www.cloudfogger.com/.
42. Cohen, F., 1987. Computer viruses: theory and experiments. *Computers & security*, 6(1), pp.22–35.
43. Cole, E., 2012. *Advanced Persistent Threat: Understanding the Danger and How to Protect Your Organization* 1st ed., Syngress.
44. Colvin, R., 2011. *{SmartScreen} Application Reputation Building Reputation*, Available at: http://goo.gl/ChWEW.
45. Comparatives, A. V, 2012. *Anti-Phishing protection of popular web browsers*, Available at: http://www.av-comparatives.org/images/docs/avc_phi_browser_201212_en.pdf.

46. Cova, M., Leita, C., Thonnard, O., Keromytis, A.D. & Dacier, M., 2010. An analysis of rogue {AV} campaigns. In *Recent Advances in Intrusion Detection*. Springer, pp. 442–463.

47. Curtsinger, C., Livshits, B., Zorn, B.G. & Seifert, C., 2011. {ZOZZLE}: Fast and Precise In-Browser {JavaScript} Malware Detection. In *{USENIX} Security Symposium*. pp. 33–48.

48. Darwish, A. & Bataineh, E., 2012. Eye tracking analysis of browser security indicators. In *2012 International Conference on Computer Systems and Industrial Informatics ({ICCSII})*. pp. 1–6.

49. Dash, D. et al., 2006. When gossip is good: Distributed probabilistic inference for detection of slow network intrusions. In *Proceedings of the national conference on Artificial Intelligence*. p. 1115.

50. Denning, D.E., 1987. An intrusion-detection model. *Software Engineering, IEEE Transactions on*, (2), pp.222–232.

51. Denver, N., 2012. *Private: Bradley Manning, WikiLeaks, and the Biggest Exposure of Official Secrets in American History*, Chicago Review Press.

52. DoD, 2013. *Annual Report to Congress*, Available at: http://www.defense.gov/pubs/2013_china_report_final.pdf.

53. Dritsas, S. et al., 2005. Employing ontologies for the development of security critical applications. In *Challenges of Expanding Internet: E-Commerce, E-Business, and E-Government*. Springer, pp. 187–201.

54. Drummond, D., 2010. Official Google Blog: A new approach to China. Available at: http://googleblog.blogspot.gr/2010/01/new-approach-to-china.html#!/2010/01/new-approach-to-china.html [Accessed November 25, 2014].

55. Economist, 2010. War in the fifth domain. Available at: http://www.economist.com/node/16478792.

56. Edge, C., Barker, W., Hunter, B. & Sullivan, G., 2010. Network Scanning, Intrusion Detection, and Intrusion Prevention Tools. In *Enterprise Mac Security*. Springer, pp. 485–504.

57. Edwards, N., 2013. Hardware intrusion detection for supply-chain threats to critical infrastructure embedded systems.

58. Egelman, S. & Schechter, S., 2013. The Importance of Being Earnest [in Security Warnings]. In *Financial Cryptography and Data Security*. Springer, pp. 52–59.

59. Ewaida, B., 2010. *Pass-the-hash attacks: Tools and Mitigation*, Available at: https://www.sans.org/reading-room/whitepapers/testing/pass-the-hash-attacks-tools-mitigation-33283.

60. Field, S., 2006. An Introduction to Kernel Patch Protection - Windows Vista Security - Site Home - MSDN Blogs. Available at: http://blogs.msdn.com/b/windowsvistasecurity/archive/2006/08/11/695993.aspx [Accessed November 26, 2014].

61. FireEye, 2014. *APT28: A window into Russia's Cyber Espionage Operations?*, Available at: https://www.fireeye.com/blog/threat-research/2014/10/apt28-a-window-into-russias-cyber-espionage-operations.html.

62. Fisher, D., 2012. *What have we learned: FLAME malware*,

63. Fogarty, K., 2014. Russian Cyberwar Force Intensifies 'net Arms Race. Available at: http://news.dice.com/2014/02/01/russian-cyberwar-force-intensifies-net-arms-race/.

64. Foreignpolicy, 2013. The Leading Global Thinkers of 2013. Available at: http://2013-global-thinkers.foreignpolicy.com/.

65. Forrest, S., Hofmeyr, S.A., Somayaji, A. & Longstaff, T.A., 1996. A sense of self for unix processes. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*. pp. 120–128.

66. Freeman, S. & Edwards-Levy, A., 2013. Americans Still Can't Decide Whether Edward Snowden Is A "Traitor" Or A "Hero," Poll Finds. *Huffington Post*. Available at: http://www.huffingtonpost.com/2013/10/30/edward-snowden-poll_n_4175089.html.

67. Galperin, E., Schoen, S. & Eckersley, P., 2011. A Post Mortem on the Iranian DigiNotar Attack. Available at: https://www.eff.org/deeplinks/2011/09/post-mortem-iranian-diginotar-attack [Accessed April 12, 2014].

68. Gartner, 2013. *Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013*, Available at: https://www.gartner.com/newsroom/id/2665715.

69. Gartner, 2014a. Gartner Says Worldwide Information Security Spending Will Grow Almost 8 Percent in 2014 as Organizations Become More Threat-Aware. Available at: https://www.gartner.com/newsroom/id/2828722 [Accessed November 29, 2014].

70. Gartner, 2012. Gartner Survey Highlights Top Five Daily Activities on Media Tablets. Available at: https://www.gartner.com/newsroom/id/2070515.

71. Gartner, 2014b. *Top 10 Strategic Technology Trends For 2014*, Available at: http://goo.gl/AdfeBN.

72. GCHQ, 2014. GCHQ. Available at: http://www.gchq.gov.uk/Pages/homepage.aspx.

73. Gebauer, M., 2012. Warfare with Malware: NATO Faced with Rising Flood of Cyberattacks. *Spiegel*.

74. Geers, K., Kindlund, D., Moran, N. & Rachwald, R., 2014. *WORLD WAR C : Understanding Nation-State Motives Behind Today's Advanced Cyber Attacks*,

75. Geftic, S., 2013. From SIEM to Security Analytics: The Path Forward. Available at: https://blogs.rsa.com/siem-security-analytics-path-forward/.

76. Gellman, B. & Markon, J., 2013. Edward Snowden says motive behind leaks was to expose "surveillance state" - The Washington Post. Available at: http://www.washingtonpost.com/politics/edward-snowden-says-motive-behind-leaks-was-to-expose-surveillance-state/2013/06/09/aa3f0804-d13b-11e2-a73e-826d299ff459_story.html [Accessed November 25, 2014].

77. Gellman, B. & Soltani, A., 2013. NSA infiltrates links to Yahoo, Google data centers worldwide, Snowden documents say. *The Washington Post*. Available at: http://www.washingtonpost.com/world/national-security/nsa-infiltrates-links-to-yahoo-google-data-centers-worldwide-snowden-documents-say/2013/10/30/e51d661e-4166-11e3-8b74-d89d714ca4dd_story.html.

78. Gibbs, S., 2015. Duqu 2.0: computer virus "linked to Israel" found at Iran nuclear talks venue. *The Guardian*. Available at: http://www.theguardian.com/technology/2015/jun/11/duqu-20-computer-virus-with-traces-of-israeli-code-was-used-to-hack-iran-talks.

79. Goman, S. & Barnes, J., 2012. Iran Blamed for Cyberattacks. Available at: http://www.wsj.com/articles/SB10000872396390444657804578052931555576700.

80. Google, 2013. *Safe Browsing API*, Available at: https://developers.google.com/safe-browsing/.

81. Google, 2014. Staying at the forefront of email security and reliability: HTTPS-only and 99.978 percent availability. Available at: http://googleblog.blogspot.gr/2014/03/staying-at-forefront-of-email-security.html.

82. Gov.uk, 2014a. Cyber defence funding worth £2 million available to suppliers. Available at: https://www.gov.uk/government/news/cyber-defence-funding-worth-2-million-available-to-suppliers.

83. Gov.uk, 2014b. *The UK Cyber Security Strategy*, Available at: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/386093/The_UK_Cyber_Security_Strategy_Report_on_Progress_and_Forward_Plans_-_De___.pdf.

84. Greenwald, G., 2013. NSA collecting phone records of millions of Verizon customers daily. *The Guardian*. Available at: http://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order.

85. Greenwald, G. & MacAskill, E., 2013. NSA Prism program taps in to user data of Apple, Google and others. *The Guardian*. Available at: http://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data.

86. Greenwald, G., MacAskill, E. & Poitras, L., 2013. Edward Snowden: the whistleblower behind the NSA surveillance revelations. *The Guardian*, 9.

87. Gritzalis, D. et al., 2013. The Sphinx enigma in critical VoIP infrastructures: Human or botnet? In *Proc. of the 4th International Conference on Information, Intelligence, Systems and Applications (IISA-2013)*. pp. 1–6.

88. Gritzalis, D., Stavrou, V., Kandias, M. & Stergiopoulos, G., 2014. Insider Threat: Enhancing BPM through Social Media. In *Proc. of the 6th IFIP International Conference on New Technologies, Mobility and Security (NMTS-2014)*. UAE: Springer.

89. Guin, U. et al., 2014. Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain. *Proceedings of the IEEE*, 102(8), pp.1207–1228.

90. Gymnopoulos, L., Dritsas, S., Gritzalis, S. & Lambrinoudakis, C., 2003. GRID security review. In *Proc. of the MMM-ACNS-2003 2nd International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security (MMM '03)*. Springer.

91. Hadžiosmanović, D., Simionato, L., Bolzoni, D., Zambon, E. & Etalle, S., 2012. N-gram against the machine: On the feasibility of the n-gram network analysis for binary protocols. In *Research in Attacks, Intrusions, and Defenses*. Springer, pp. 354–373.

92. Healey, J. & Grindal, K., 2013. *A Fierce Domain: Conflict in Cyberspace, 1986 to 2012*, Cyber Conflict Studies Association.

93. Helman, P., Liepins, G. & Richards, W., 1992. Foundations of intrusion detection [computer security]. In *Computer Security Foundations Workshop V, 1992. Proceedings.* pp. 114–120.

94. Hendler, J. & Berners-Lee, T., 2010. From the Semantic Web to social machines: A research challenge for AI on the World Wide Web. *Artificial Intelligence*, 174(2), pp.156–161.

95. Heuer, R.J. & Herbig, K., 2001. The insider espionage threat. *Research on Mitigating the Insider Threat to Information Systems*, 2.

96. Hofmeyr, S.A., Forrest, S. & Somayaji, A., 1998. Intrusion detection using sequences of system calls. *Journal of computer security*, 6(3), pp.151–180.

97. Hosenball, M., 2013. NSA chief says Snowden leaked up to 200,000 secret documents | Reuters. Available at: http://www.reuters.com/article/2013/11/14/us-usa-security-nsa-idUSBRE9AD19B20131114 [Accessed November 25, 2014].

98. Hudson, J., 2013. Deciphering How Edward Snowden Breached the NSA – Venafi. Available at: https://www.venafi.com/blog/post/deciphering-how-edward-snowden-breached-the-nsa/ [Accessed November 25, 2014].

99. ISACA, 2014. *The Growing Cybersecurity Skills Crisis*, Available at: http://www.isaca.org/cyber/Documents/Cybersecurity-Report_pre_Eng_0414.pdf.

100. Isikoff, M., 2013. Chinese hacked Obama, McCain campaigns, took internal documents, officials say - Investigations. Available at: http://investigations.nbcnews.com/_news/2013/06/06/18807056-chinese-hacked-obama-mccain-campaigns-took-internal-documents-officials-say [Accessed December 2, 2014].

101. Jansson, K. & von Solms, R., 2013. Phishing for phishing awareness. *Behaviour & Information Technology*, 32(6), pp.584–593.

102. Juels, A. & Rivest, R.L., 2013. Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. pp. 145–160.

103. Kammas, P., Komninos, T. & Stamatiou, Y.C., 2008. A queuing theory based model for studying intrusion evolution and elimination in computer networks. In *Information Assurance and Security, 2008. ISIAS'08. Fourth International Conference on*. pp. 167–171.

104. Kandias, M., Galbogini, K., Mitrou, L. & Gritzalis, D., 2013. Insiders trapped in the mirror reveal themselves in social media. In *Proc. of the 7th International Conference on Network and System Security (NSS 2013)*. Springer, pp. 220–235.

105. Kandias, M., Mylonas, A., Theoharidou, M. & Gritzalis, D., 2011. Exploitation of auctions for outsourcing security-critical projects. In *Proc. of the 16th IEEE Symposium on Computers and Communications (ISCC '11)*. pp. 646–651.

106. Kandias, M., Mylonas, A., Virvilis, N., Theoharidou, M. & Gritzalis, D., 2010. An insider threat prediction model. In *Proc. of the 7th International Conference on Trust, Privacy, and Security in Digital Business (TrustBus-2010)*. Springer, pp. 26–37.

107. Kandias, M., Stavrou, V., Bozovic, N. & Gritzalis, D., 2013. Proactive insider threat detection through social media: The YouTube case. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. pp. 261–266.

108. Kandias, M., Virvilis, N. & Gritzalis, D., 2013. The insider threat in Cloud computing. In *Proc. of the 6th International Conference on Critical Infrastructure Security (CRITIS-2011)*. Springer, pp. 93–103.

109. Kaspersky, 2013a. *Kaspersky Security Bulletin 2013. Overall Statistics for 2013*, Available at: http://goo.gl/30qqal.

110. Kaspersky, 2013b. "Red October" Diplomatic Cyber Attacks Investigation - Securelist. Available at: http://securelist.com/analysis/publications/36740/red-october-diplomatic-cyber-attacks-investigation/ [Accessed November 26, 2014].

111. Kaspersky, 2012. Resource 207: Kaspersky Lab. Available at: http://www.kaspersky.com/about/news/virus/2012/Resource_207_Kaspersky_Lab_Research_Proves_that_Stuxnet_and_Flame_Developers_are_Connected [Accessed November 26, 2014].

112. Kaspersky, 2015. *The Duqu 2.0*, Available at: https://securelist.com/files/2015/06/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf.

113. Kaspersky, 2014. The Regin Platform Nation-state ownage of GSM networks. Available at: http://securelist.com/files/2014/11/Kaspersky_Lab_whitepaper_Regin_platform_eng.pdf.

114. Kelley, M., 2012. *The Stuxnet Virus at Iran's Nuclear Facility was Planted by an Iranian Double Agent*,

115. Kelly, M., 2012. The Stuxnet Virus At Iran's Nuclear Facility Was Planted By An Iranian Double Agent. Available at: http://www.businessinsider.com/stuxnet-virus-planted-by-iranian-double-agent-2012-4 [Accessed November 25, 2014].

116. Kijewski, P., 2004. ARAKIS-An early warning and attack identification system. In *Proc. of the 16th Annual First Conference, Dudapest, Hungary*.

117. Kirda, E. & Kruegel, C., 2005. Protecting users against phishing attacks with antiphish. In *Computer Software and Applications Conference, 2005. {COMPSAC} 2005. 29th Annual International*. IEEE, pp. 517–524.

118. Kokolakis, S., Gritzalis, D. & Katsikas, S., 1998. Generic security policies for healthcare information systems. *Health informatics journal*, 4(3-4), pp.184–195.

119. Kolter, J.Z. & Maloof, M.A., 2006. Learning to detect and classify malicious executables in the wild. *The Journal of Machine Learning Research*, 7, pp.2721–2744.

120. Komninos, T., Spirakis, P., Stamatiou, Y.C. & Vavitsas, G., 2007. A worm propagation model based on scale free network structures and people's email acquaintance profiles. *International Journal of Computer Science and Network Security*, 7(2), pp.308–315.

121. Kotenko, I. & Skormin, V., 2010. *Computer Network Security: 5th International Conference, on Mathematical Methods, Models, and Architectures for Computer Network Security, MMM-ACNS 2010, St. Petersburg, Russia, September 8-10, 2010, Proceedings*, Springer Science & Business Media.

122. Kotzanikolaou, P., Theoharidou, M. & Gritzalis, D., 2013a. Assessing n-order dependencies between critical infrastructures. *International Journal of Critical Infrastructures*, 9(1-2), pp.93–110.

123. Kotzanikolaou, P., Theoharidou, M. & Gritzalis, D., 2013b. Cascading effects of common-cause failures in critical infrastructures. In *Proc. of the 7th IFIP International Conference on Critical Infrastructure Protection (CIP-2013)*. Springer, pp. 171–182.

124. Krekel, B., 2009. *Capability of the People's Republic of China to Conduct Cyber Warfare and Computer Network Exploitation*, Available at: http://www2.gwu.edu/~nsarchiv/NSAEBB/NSAEBB424/docs/Cyber-030.pdf.

125. Lagadec, P., 2006. Diode r{é}seau et ExeFilter: 2 projets pour des interconnexions s{é}curis{é}es. *Proc. of SSTIC06*, pp.1–15.

126. Lampson, B.W., 1973. A note on the confinement problem. *Communications of the ACM*, 16(10), pp.613–615.

127. Langner, R., 2011. Stuxnet: Dissecting a cyberwarfare weapon. *Security & Privacy, IEEE*, 9(3), pp.49–51.

128. Lee, H.L. & Whang, S., 2005. Higher supply chain security with lower cost: Lessons from total quality management. *International Journal of production economics*, 96(3), pp.289–300.

129. Lekkas, D. & Gritzalis, D., 2007a. e-Passports as a means towards the first world-wide Public Key Infrastructure. In *Proc. of the 4th European PKI Workshop (EuroPKI '07)*. Springer, pp. 34–48.

130. Lekkas, D. & Gritzalis, D., 2007b. Long-term verifiability of the electronic healthcare records authenticity. *international journal of medical informatics*, 76(5), pp.442–448.

131. Lemos, R., 2011. Stuxnet attack more effective than bombs | InfoWorld. Available at: http://www.infoworld.com/article/2625351/malware/stuxnet-attack-more-effective-than-bombs.html [Accessed November 26, 2014].

132. Leyden, J., 2014. GCHQ: We can't track crims any more thanks to Snowden. *The register*. Available at: http://www.theregister.co.uk/2014/12/23/gchq_criminal_tracking_post_snowden/.

133. Liao, Y. & Vemuri, V.R., 2002. Using Text Categorization Techniques for Intrusion Detection. In *USENIX Security Symposium*. pp. 51–59.

134. Libicki, M., Sentry, D. & Pollak, J., 2014. *An Examination of the Cybersecurity Labor Market*, Available at: http://www.rand.org/content/dam/rand/pubs/research_reports/RR400/RR430/RAND_RR430.pdf .

135. Liu, A., Martin, C., Hetherington, T. & Matzner, S., 2005. A comparison of system call feature for insider threat detection. In *Proc. of the 6th Annual IEEE Systems, Man & Cybernetics, Information Assurance Workshop*. pp. 341–347.

136. Locasto, M.E., Parekh, J.J., Keromytis, A.D. & Stolfo, S.J., 2005. Towards collaborative security and p2p intrusion detection. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*. pp. 333–339.

137. Love, D., 2013. *The Latest Jailbreak Statistics Are Jaw-Dropping*, Available at: http://www.businessinsider.com/jailbreak-statistics-2013-3.

138. Lu, L., Yegneswaran, V., Porras, P. & Lee, W., 2010. Blade: an attack-agnostic approach for preventing drive-by malware infections. In *Proceedings of the 17th {ACM} conference on Computer and communications security*. ACM, pp. 440–450.

139. Lynn, W.J., 2010. Defending a New Domain: The Pentagon's Cyberstrategy. *Foreign Affairs*, pp.97–108.

140. Magklaras, G.B. & Furnell, S.M., 2005. A preliminary model of end user sophistication for insider threat prediction in IT systems. *Computers & Security*, 24(5), pp.371–380.

141. Mairh, A., Barik, D., Verma, K. & Jena, D., 2011. Honeypot in network security: a survey. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*. pp. 600–605.

142. Mandiant, 2013. *Exposing One of China's Cyber Espionage Units*, Mandiant.

143. Manning, B., 2013. Bradley Manning's statement taking responsibility for releasing documents to WikiLeaks. Available at: http://www.chelseamanning.org/news/bradley-mannings-statement-taking-responsibility-for-releasing-documents-to-wikileaks.

144. Mavrommatis, N. & Monrose, M., 2008. All your iframes point to us. *17th USENIX Security Symposium*. Available at: https://www.usenix.org/legacy/event/sec08/tech/full_papers/provos/provos.pdf [Accessed January 22, 2015].

145. Maxion, R.A. & Tan, K.M.C., 2000. Benchmarking anomaly-based detection systems. In *Dependable Systems and Networks, 2000. DSN 2000. Proceedings International Conference on*. pp. 623–630.

146. Mayer, M., 2014. Our Commitment to Protecting Your Information. Available at: http://yahoo.tumblr.com/post/67373852814/our-commitment-to-protecting-your-information.

147. Mazher, N., Ashraf, I. & Altaf, A., 2013. Which web browser work best for detecting phishing. In *Information & Communication Technologies (ICICT), 2013 5th International Conference on*. pp. 1–5.

148. McAfee, 2014a. McAfee Labs Report Highlights Success of Phishing Attacks with 80 Percent of Business Users Unable to Detect Scams. Available at: http://www.mcafee.com/us/about/news/2014/q3/20140904-01.aspx.

149. McAfee, 2014b. *McAfee Labs Threats report*, Available at: http://www.mcafee.com/mx/resources/reports/rp-quarterly-threat-q1-2014.pdf.

150. Mcafee, 2014. *Site Advisor*, Available at: https://www.siteadvisor.com/.

151. McAfee Labs, 2010. *Protecting Your Critical Assets - Lessons Learned from "Operation Aurora,"* Available at: http://www.wired.com/images_blogs/threatlevel/2010/03/operationaurora_wp_0310_fnl.pdf.

152. McDonald, G., Murchu, L.O., Doherty, S. & Chien, E., 2013. Stuxnet 0.5: The missing link. *Symantec Report*.

153. Melber, D., 2013. Securing and Auditing High Risk Files on Windows Servers. Available at: http://www.windowsecurity.com/articles-tutorials/windows_server_2008_security/securing-auditing-high-risk-files-windows-servers.html [Accessed November 25, 2014].

154. Microsoft, 2011. SmartScreen Filter - Microsoft Windows. *windows.microsoft.com*. Available at: http://windows.microsoft.com/en-us/internet-explorer/products/ie-9/features/smartscreen-filter [Accessed November 15, 2014].

155. Microsoft, 2013. What is Protected View? Available at: https://support.office.com/en-us/article/What-is-Protected-View-d6f09ac7-e6b9-4495-8e43-2bbcdbcb6653?ui=en-US&rs=en-US&ad=US [Accessed November 26, 2014].

156. Miller, J.F., 2013. *Supply Chain Attack Framework and Attack Patterns*,

157. Mitrou, L. & Karyda, M., 2006. Employees' privacy vs. employers' security: Can they be balanced? *Telematics and Informatics*, 23(3), pp.164–178.

158. Modi, C. et al., 2013. A survey of intrusion detection techniques in Cloud. *Journal of Network and Computer Applications*, 36(1), pp.42–57.

159. Moore, J., 2015. Anonymous's "Electronic Holocaust" Against Israel Falls Flat. Available at: http://europe.newsweek.com/anonymous-electronic-holocaust-against-israel-has-limited-success-320176.

160. Moskowitz, J., 2015. Cyberattack tied to Hezbollah ups the ante for Israel's digital defenses. Available at: http://www.csmonitor.com/World/Passcode/2015/0601/Cyberattack-tied-to-Hezbollah-ups-the-ante-for-Israel-s-digital-defenses.

161. Mozilla, 2014. Mozilla Support. Available at: https://support.mozilla.org/en-US/kb/how-does-phishing-and-malware-protection-work#w_how-does-phishing-and-malware-protection-work-in-firefox.

162. Mylonas, A., Dritsas, S., Tsoumas, B. & Gritzalis, D., 2012. On the feasibility of malware attacks in smartphone platforms. In *E-Business and Telecommunications*. Springer, pp. 217–232.

163. Mylonas, A., Gritzalis, D., Tsoumas, B. & Apostolopoulos, T., 2013. A qualitative metrics vector for the awareness of smartphone security users. In *Trust, Privacy, and Security in Digital Business*. Springer, pp. 173–184.

164. Mylonas, A., Kastania, A. & Gritzalis, D., 2013. Delegate the smartphone user? Security awareness in smartphone platforms. *Computers & Security*, 34, pp.47–66.

165. Mylonas, A., Tsalis, N. & Gritzalis, D., 2013. Evaluating the manageability of web browsers controls. In *9th International Workshop on Security and Trust Management (STM-2013)*. United Kingdom: Springer, pp. 82–98.

166. Nakashima, E., 2014. U.S. cyberwarfare force to grow significantly, defense secretary says. Available at: http://www.washingtonpost.com/world/national-security/us-cyberwarfare-force-to-grow-significantly-defense-secretary-says/2014/03/28/0a1fa074-b680-11e3-b84e-897d3d12b816_story.html.

167. Nazario, J., 2009. Phoneyc: A virtual client honeypot. In *Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*. p. 6.

168. Netcraft, 2014. *Phishing Site Feed*, Available at: http://www.netcraft.com/anti-phishing/phishing-site-feed/.

169. Nguyen, N.T., Reiher, P.L. & Kuenning, G.H., 2003. Detecting Insider Threats by Monitoring System Call Activity. In *IAW*. pp. 45–52.

170. Oberheide, J., Cooke, E. & Jahanian, F., 2008. {CloudAV}: N-Version Antivirus in the Network Cloud. In *{USENIX} Security Symposium*. pp. 91–106.

171. Opendns, 2014. *OpenDNS*, Available at: http://www.opendns.com/.

172. OWASP, 2014. *Certificate and Public Key Pinning*, Available at: http://www.netcraft.com/anti-phishing/phishing-site-feed/.

173. Perdisci, R., Lanzi, A. & Lee, W., 2008. Classification of packed executables for accurate computer virus detection. *Pattern Recognition Letters*, 29(14), pp.1941–1946.

174. Perdisci, R., Lanzi, A. & Lee, W., 2008. McBoost: Boosting scalability in malware collection and analysis using statistical classification of executables. *Computer Security Applications …*. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4721567 [Accessed January 22, 2015].

175. Phishtank, 2014. *Join the fight against phishing*, Available at: https://www.phishtank.com/.

176. Pilkington, E., 2013. Bradley Manning a traitor who set out to harm US, prosecutors conclude. *The Guardian*. Available at: http://www.theguardian.com/world/2013/jul/25/bradley-manning-traitor-wikileaks-prosecution.

177. Pitropakis, N., Darra, E., Vrakas, N. & Lambrinoudakis, C., 2013. It's All in the Cloud: Reviewing Cloud Security. In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*. pp. 355–362.

178. Poitras, L., Rosenbach, M. & Holger, S., 2014. "A" for Angela: GCHQ and NSA Targeted Private German Companies and Merkel. *Spiegel*. Available at: http://www.spiegel.de/international/germany/gchq-and-nsa-targeted-private-german-companies-a-961444.html.

179. Polemi, D. et al., 2013. S-port: Collaborative security management of port information systems. In *Information, Intelligence, Systems and Applications (IISA), 2013 Fourth International Conference on*. pp. 1–6.

180. Prathapani, A., Santhanam, L. & Agrawal, D.P., 2009. Intelligent honeypot agent for blackhole attack detection in wireless mesh networks. In *Mobile Adhoc and Sensor Systems, 2009. MASS'09. IEEE 6th International Conference on*. pp. 753–758.

181. Provos, N. et al., 2007. The ghost in the browser analysis of web-based malware. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. p. 4.

182. Puleo, A.J., 2006. *Mitigating insider threat using human behavior influence models*,

183. Raiu, C., Soumenkov, I., Baumgartner, K. & Kamluk, V., 2013. *The MiniDuke Mystery: PDF 0-day Government Spy Assembler 0x29A Micro Backdoor*, Available at: https://kasperskycontenthub.com/wp-content/uploads/sites/43/vlpdfs/themysteryofthepdf0-dayassemblermicrobackdoor.pdf.

184. Rajab, M.A., Ballard, L., Lutz, N., Mavrommatis, P. & Provos, N., 2013. {CAMP}: Content-Agnostic Malware Protection. In *NDSS*. Citeseer.

185. RaniSahu, K. & Dubey, J., 2014. A Survey on Phishing Attacks. *International Journal of Computer Applications*, 88(10), pp.42–45.

186. Rasch, G., 1993. *Probabilistic models for some intelligence and attainment tests.*, ERIC.

187. Reed, T., 2007. *At the abyss: an insider's history of the Cold War*, Random House LLC.

188. Robertson, J. & Riley, M., 2014. Mysterious '08 Turkey Pipeline Blast Opened New Cyberwar Era. *Bloomberg*. Available at: http://www.bloomberg.com/news/2014-12-10/mysterious-08-turkey-pipeline-blast-opened-new-cyberwar.html [Accessed December 15, 2014].

189. Rogers, M.K., 2001. *A social learning theory and moral disengagement analysis of criminal computer behavior: An exploratory study*. University of Manitoba.

190. Rosiello, A.P.E., Kirda, E., Kruegel, C. & Ferrandi, F., 2007. A layout-similarity-based approach for detecting phishing pages. In *Security and Privacy in Communications Networks and the Workshops, 2007. {SecureComm} 2007. Third International Conference on*. IEEE, pp. 454–463.

191. Ross, R.S., 2012. Guide for Conducting Risk Assessments (NIST SP-800-30rev1). *The National Institute of Standards and Technology (NIST), Gaithersburg*.

192. RSA, 2011. Anatomy of an Attack - Speaking of Security - The RSA Blog and Podcast. Available at: https://blogs.rsa.com/anatomy-of-an-attack/ [Accessed November 25, 2014].

193. RSA, 2014. *RSA Online fraud report*, Available at: http://www.emc.com/collateral/fraud-report/rsa-online-fraud-report-012014.pdf.

194. Ben Salem, M. & Stolfo, S., 2011. Decoy Document Deployment for Effective Masquerade Attack Detection. In T. Holz & H. Bos, eds. *Detection of Intrusions and Malware, and Vulnerability Assessment*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 35–54.

195. Sam Adams Award, 2014. Chelsea Manning. Available at: http://samadamsaward.ch/chelsea-manning/.

196. Schneier, B., 2013. More about the NSA's Tailored Access Operations Unit. Available at: https://www.schneier.com/blog/archives/2013/12/more_about_the.html.

197. Schneier, B., 2012. Schneier on Security: The Failure of Anti-Virus Companies to Catch Military Malware. Available at: https://www.schneier.com/blog/archives/2012/06/the_failure_of_3.html [Accessed November 26, 2014].

198. Schultz, E.E., 2002. A framework for understanding and predicting insider attacks. *Computers & Security*, 21(6), pp.526–531.

199. Schwartz, M., 2013. Google Aurora Hack Was Chinese Counterespionage Operation. Available at: http://www.darkreading.com/attacks-and-breaches/google-aurora-hack-was-chinese-counterespionage-operation/d/d-id/1110060?

200. Shahzad, A., Hussain, M. & Khan, M.N.A., 2013. Protecting from Zero-Day Malware Attacks. *Middle-East Journal of Scientific Research*, 17(4), pp.455–464.

201. Shaw, E., Ruby, K. & Post, J., 1998. The insider threat to information systems: The psychology of the dangerous insider. *Security Awareness Bulletin*, 2(98), pp.1–10.

202. Sheng, S. et al., 2009. An empirical analysis of phishing blacklists. In *Sixth Conference on Email and Anti-Spam ({CEAS})*.

203. Shin, S., Xu, Z. & Gu, G., 2012. {EFFORT}: Efficient and effective bot malware detection. In *{INFOCOM}, 2012 Proceedings {IEEE}*. IEEE, pp. 2846–2850.

204. Silowash, G.J. et al., 2012. Common Sense Guide to Mitigating Insider Threats (4th Edition). *Software Engineering Institute*, (677).

205. Snapp, S.R. et al., 1991. DIDS (distributed intrusion detection system)-motivation, architecture, and an early prototype. In *Proceedings of the 14th national computer security conference*. pp. 167–176.

206. Snort, 2015. Snort. Available at: https://www.snort.org/.

207. Sobrier, J., 2014. *Google Safe Browsing v2 {API}: Implementation notes*, Available at: https://www.zscaler.com/research/Google.

208. Sommer, R. & Paxson, V., 2010. Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy (SP), 2010 IEEE Symposium on*. pp. 305–316.

209. Soupionis, Y. & Benoist, T., 2014. Cyber attacks in Power Grid ICT systems leading to financial disturbance. In *9th International Conference on Critical Information Infrastructures Security (CRITIS-2014)*. Cyprus: Springer.

210. Soupionis, Y., Ntalampiras, S. & Giannopoulos, G., 2014. Faults and Cyber Attacks Detection in Critical Infrastructures. In *9th International Conference on Critical Information Infrastructures Security (CRITIS-2014)*. Cyprus: Springer.

211. Spiegel, 2013. Inside TAO: Documents Reveal Top NSA Hacking Unit. Available at: http://www.spiegel.de/international/world/the-nsa-uses-powerful-toolbox-in-effort-to-spy-on-global-networks-a-940969.html.

212. Spitzner, L., 2003. Honeypots: Catching the insider threat. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. pp. 170–179.

213. Spitzner, L., 2003. The Honeynet Project: Trapping the Hackers. *Security Privacy, IEEE*, 1(2), pp.15–23.

214. Stavrou, V., Kandias, M., Karoulas, G. & Gritzalis, D., 2014. Business Process Modeling for Insider threat monitoring and handling. In *Proc. of 11th International Conference on Trust, Privacy & Security in Digital Business (TRUSTBUS-2014)*. Germany: Springer, pp. 119–131.

215. Stergiopoulos, G., Theoharidou, M. & Gritzalis, D., 2015. Using logical error detection in Remote-Terminal Units to predict initiating events of Critical Infrastructures failures. In *Proc. of the 3rd International Conference on Human Aspects of Information Security, Privacy and Trust (HCI-2015)*. USA: Springer.

216. Strobel, W., 2013. Analysis: Manning damage has fallen well short of worst U.S. fears. *Reuters*. Available at: http://www.reuters.com/article/2013/07/31/us-usa-wikileaks-manning-damage-analysis-idUSBRE96U00420130731.

217. Symantec, 2014a. *2014 Internet Security Threat Report*, Available at: http://www.symantec.com/security_response/publications/threatreport.jsp.

218. Symantec, 2014b. *Regin: Top-tier espionage tool enables stealthy surveillance*, Available at: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/regin-analysis.pdf.

219. Symantec, 2014c. *Safe Web*, Available at: https://safeweb.norton.com.

220. Taborek, N. & Capaccio, T., 2013. Apple Mobile Devices Cleared for Use on U.S. Military Networks. *Bloomberg*. Available at: http://www.bloomberg.com/news/2013-05-17/apple-mobile-devices-cleared-for-use-on-u-s-military-networks.html [Accessed November 15, 2014].

221. Theoharidou, M., Kokolakis, S., Karyda, M. & Kiountouzis, E., 2005. The insider threat to information systems and the effectiveness of ISO17799. *Computers & Security*, 24(6), pp.472–484.

222. Theoharidou, M., Kotzanikolaou, P. & Gritzalis, D., 2011. Risk assessment methodology for interdependent critical infrastructures. *International Journal of Risk Assessment and Management*, 15(2-3), pp.128–148.

223. Theoharidou, M., Papanikolaou, N., Pearson, S. & Gritzalis, D., 2013. Privacy risks, security and accountability in the Cloud. In *Proc. of the 5th IEEE Conference on Cloud Computing Technology and Science (CloudCom-2013)*. United Kingdom: IEEE Press, pp. 177–184.

224. Theoharidou, M., Tsalis, N. & Gritzalis, D., 2013. In cloud we trust: risk-assessment-as-a-service. In *Proc. of the 7th IFIP International Conference on Trust Management (IFIP TM-2013)*. Spain: Springer, pp. 100–110.

225. Thompson, P., 2004. Weak models for insider threat detection. In *Defense and Security*. pp. 40–48.

226. Thonnard, O. & Dacier, M., 2008. A framework for attack patterns' discovery in honeynet data. *digital investigation*, 5, pp.S128–S139.

227. Timberg, C., 2013. Microsoft to encrypt data in its services in bid to prevent snooping. *The Washington Post*. Available at: http://www.washingtonpost.com/business/technology/microsoft-to-encrypt-data-in-its-services-in-bid-to-prevent-snooping/2013/12/04/f91f7b02-5d2c-11e3-bc56-c6ca94801fac_story.html.

228. Times, 2014. Time: The 100 most influential people in the world. *Time magazine*. Available at: http://time.com/time100-2014/.

229. Timm, T., 2014. Congress wants NSA reform after all. Obama and the Senate need to pass it. *The Guardian*. Available at: http://www.theguardian.com/commentisfree/2014/jun/20/congress-obama-nsa-reform-obama-senate.

230. Tivadar, M., Balazs, B. & Istrate, C., 2013. *A Closer Look at MiniDukele*, Available at: http://labs.bitdefender.com/wp-content/uploads/downloads/2013/04/MiniDuke_Paper_Final.pdf.

231. TOR, 2015. TOR. Available at: https://tor.eff.org/.

232. TrendMicro, 2015. *Operation Arid Viper*, Available at: http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-operation-arid-viper.pdf.

233. Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y. & Lin, W.-Y., 2009. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10), pp.11994–12000.

234. Vadrevu, P., Rahbarinia, B., Perdisci, R., Li, K. & Antonakakis, M., 2013. Measuring and Detecting Malware Downloads in Live Network Traffic. In *Computer Security–{ESORICS} 2013*. Springer, pp. 556–573.

235. Vidas, T. et al., 2013. QRishing: The Susceptibility of Smartphone Users to QR Code Phishing Attacks. In *Financial Cryptography and Data Security*. Springer, pp. 52–69.

236. VirusTotal, 2014. *VirusTotal*, Available at: https://www.virustotal.com/.

237. Virvilis, N., Dritsas, S. & Gritzalis, D., 2011a. A cloud provider-agnostic secure storage protocol. In *Proc. of the 5th International Conference on Critical Information Infrastructure Security (CRITIS-2010)*. Springer, pp. 104–115.

238. Virvilis, N., Dritsas, S. & Gritzalis, D., 2011b. Secure cloud storage: Available infrastructures and architectures review and evaluation. In *Proc. of the 8th International Conference on Trust, Privacy & Security in Digital Business (TRUSTBUS-2011)*. Springer, pp. 74–85.

239. Virvilis, N. & Gritzalis, D., 2013. The big four-What we did wrong in advanced Persistent Threat detection? In *Proc. of the 8th International Conference on Availability, Reliability and Security (ARES-2013)*. IEEE, pp. 248–254.

240. Virvilis, N., Gritzalis, D. & Apostolopoulos, T., 2013. Trusted Computing vs. Advanced Persistent Threats: Can a defender win this game? In *Proc. of 10th IEEE International Conference on Autonomic and Trusted Computing (ATC-2013)*. IEEE, pp. 396–403.

241. Virvilis, N. & Serrano, O., 2014. Changing the game: The art of deceiving sophisticated attackers. In *Proc. of the 6th International Conference on Cyber Conflict (CYCON-014)*. Estonia: IEEE, pp. 87–97.

242. Virvilis, N., Serrano, O. & Dandurand, L., 2014. Big Data analytics for sophisticated attack detection. *ISACA*, 3, pp.22–25.

243. Virvilis, N., Tsalis, N., Mylonas, A. & Gritzalis, D., 2014. Mobile devices: A phisher's paradise. In *Proc. of the 11th International Conference on Security and Cryptography (SECRYPT-2014)*. pp. 79–87.

244. Voris, J.A., Jermyn, J., Keromytis, A.D. & Stolfo, S.J., 2013. Bait and Snitch: Defending Computer Systems with Decoys.

245. Wang, P., Wu, L., Cunningham, R. & Zou, C.C., 2010. Honeypot detection in advanced botnet attacks. *International Journal of Information and Computer Security*, 4(1), pp.30–51.

246. Wang, W. et al., 2013. Detecting targeted attacks by multilayer deception. *Journal of Cyber Security and Mobility*, 2(2), pp.175–199.

247. Weiss, G.W., 1996. *The Farewell Dossier*,

248. Williams, Z., Lueg, J.E. & LeMay, S.A., 2008. Supply chain security: an overview and research agenda. *International Journal of Logistics Management, The*, 19(2), pp.254–281.

249. Wood, B., 2000. An insider threat model for adversary simulation. *SRI International, Research on Mitigating the Insider Threat to Information Systems*, 2, pp.1–3.

250. Wu, H., Schwab, S. & Peckham, R.L., 2008. Signature based network intrusion detection system and method.

251. Xu, Z. & Zhu, S., 2012. Abusing Notification Services on Smartphones for Phishing and Spamming. In *WOOT*. pp. 1–11.

252. Yadron, D., 2014. Symantec Develops New Attack on Cyberhacking. *The Wall Street Journal*. Available at: http://www.wsj.com/articles/SB10001424052702303417104579542140235850578.

253. Zetter, K., 2010. Google Hack Attack Was Ultra Sophisticated, New Details Show. Available at: http://www.wired.com/2010/01/operation-aurora/ [Accessed November 25, 2014].

254. Zhang, H., Liu, G., Chow, T.W.S. & Liu, W., 2011. Textual and visual content-based anti-phishing: a Bayesian approach. *Neural Networks, {IEEE} Transactions on*, 22(10), pp.1532–1546.

255. Zhang, J., Seifert, C., Stokes, J.W. & Lee, W., 2011. Arrow: Generating signatures to detect drive-by downloads. In *Proceedings of the 20th international conference on World wide web*. ACM, pp. 187–196.

256. Zhou, Y. & Jiang, X., 2012. Dissecting android malware: Characterization and evolution. In *Security and Privacy ({SP}), 2012 {IEEE} Symposium on*. IEEE, pp. 95–109.

257. Zhou, Y., Wang, Z., Zhou, W. & Jiang, X., 2012. Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets. In *19th Annual Symposium on Network and Distributed System Security (NDSS)*.