Tomi Kyöstilä

The effectiveness of evasion techniques against intrusion prevention systems

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 12.5.2014

Thesis supervisor:

Prof. Jukka Manner

Thesis advisor:

D.Sc. (Tech.) Mikko Särelä

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Tomi Kyöstilä

Title: The effectiveness of evasion techniques against intrusion prevention systems

Date: 12.5.2014 Language: English Number of pages:7+60

Department of Communications and Networking

Professorship: Networking Technology Code: S-38

Supervisor: Prof. Jukka Manner

Advisor: D.Sc. (Tech.) Mikko Särelä

Evasions and evasion combinations are used to masquerade attacks in order to avoid detection by security appliances. This thesis evaluates the effectiveness of these techniques against the state of the art intrusion prevention systems. In total, 11 intrusion prevention systems were studied, 10 commercial and 1 free solution.

Four experiments were conducted in this study. Each of the experiments contained a million attacks that were performed with randomized evasions and evasion combinations against each intrusion prevention system. The used attack stayed the same during a single experiment, but each attack was disguised with different evasion techniques. Standardized configurations were used in order to produce comparable results.

The results indicate that evasion techniques are effective against the majority of tested intrusion prevention systems. Even though some of the techniques are from the 1990s, they can be fine-tuned to fool most of the tested appliances. One evasion technique is not always enough to avoid detection, but combining multiple techniques increases the possibility to find a way to evade detection.

Keywords: Evasions, Evasion Combinations, Intrusion Prevention Systems, Network Security, IPS

AALTO-YLIOPISTO SÄHKOTEKNIIKAN KORKEAKOULU	DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Tomi Kyöstilä

Työn nimi: Evaasiotekniikoiden tehokkuus tunkeutumisenestojärjestelmiä vastaan

Päivämäärä: 12.5.2014 Kieli: Englanti Sivumäärä:7+60

Tietoliikenne- ja tietoverkkotekniikan laitos

Professuuri: Tietoverkkotekniikka Koodi: S-38

Valvoja: Prof. Jukka Manner

Ohjaaja: TkT Mikko Särelä

Evaasioita ja evaasiokombinaatiota käytetään naamioimaan hyökkäyksiä, jotta tietoturvalaitteet eivät havaitsisi niitä. Diplomityössä tutkitaan näiden tekniikoiden tehokkuutta uusimpia tunkeutumisenestojärjestelmiä vastaan. Yhteensä 11 tunkeutumisenestojärjestelmää tutkittiin, joista 10 on kaupallista ja yksi ilmainen.

Tutkimuksessa suoritettiin neljä koetta. Jokainen koe sisälsi miljoona hyökkäystä, jotka suoritettiin jokaista tunkeutumisenestojärjestelmää vastaan satunnaisin evaasioin ja evaasiokombinaatioin. Käytetty hyökkäys pysyi samana yksittäisen kokeen aikana, mutta jokainen hyökkäys oli naamioitu eri evaasiotekniikoin. Yhtenäistettyjä konfiguraatioita käytettiin, jotta saataisiin vertailukelpoisia tuloksia.

Tulokset osoittavat, että evaasiotekniikat ovat toimivia suurinta osaa testattuja tunkeutumisenestojärjestelmiä vastaan. Vaikka osa evaasiotekniikoista on peräisin 1990-luvulta, ne voidaan saada hienosäädettyä huijaamaan suurinta osaa testatuista laitteista. Yksi evaasiotekniikka ei ole aina riittävä, jotta voitaisiin välttää hyökkäyksen havainnointi. Monen eri tekniikan yhdistäminen lisää kuitenkin todennäköisyyttä löytää tapa kiertää havainnointi.

Avainsanat: Evaasiotekniikat, evaasiokombinaatiot, tunkeutumisenestojärjestelmät, tietoturva, IPS

Preface

This thesis was carried out at the Department of Communications and Networking at the Aalto University School of Electrical Engineering.

I would like to thank my supervisor, Professor Jukka Manner, for excellent guidance throughout the process of creating this thesis. All the conversations about implementations and experiments were extremely valuable and led to a better study. I would also like to thank him for giving me the opportunity to be a part of an interesting project at the Department which resulted in this thesis among other things.

I am grateful to my instructor, D.Sc. (Tech.) Mikko Särelä, who gave me important feedback during the writing process. His comments were crucial in order to improve this thesis. I would also like to express my gratitude to all the collaborators who helped me especially with the hardware.

I would like to give special thanks to my family for always being very supportive and encouraging.

Otaniemi, 12.5.2014

Tomi Kyöstilä

Contents

Abbreviations

AET	Advanced Evasion Techniques
ARP	Address Resolution Protocol
ARPA	Advanced Research Projects Agency
ASCII	American Standard Code for Information Interchange
CVE	Common Vulnerabilities and Exposures
DCE/RPC	Distributed Computing Environment / Remote Procedure Call
DDoS	Distributed Denial-of-Service
DARPA	Defense Advanced Research Projects Agency
DUT	Device Under Test
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
MSRPC	Microsoft Remote Procedure Call
NBA	Network Behavior Analysis
OSI	Open Systems Interconnection
OVF	Open Virtualization Format
phpBB	PHP Bulletin Board
RFC	Request for Comments
SMB	Server Message Block
TCP	Transmission Control Protocol
TTL	Time To Live

1 Introduction

Cybercrime is a huge issue, and regular Internet users are usually its victims. According to Norton, cybercrime cost $110 billion in 2012. Additionally, 66% of adults using the Internet have been victims of some kind of cybercrime in their lifetime, such as online scams, credit card frauds, hacking and malware attacks. [55] Over the years, various significant data thefts have been reported. One example is a card-processing company, Heartland Payment Systems, which was a victim of theft of over 130 million credit and debit card numbers. [3, 43]

As the Internet and web services gained popularity, it became more intriguing to attackers to try to exploit these services and servers running them. The more people online, the more possibilities it opens to attackers. From 2000 to 2010, the number of the Internet users increased from 360 million to over 2 billion [6].

Security was overlooked when the predecessor of the Internet, the ARPANET, was being developed. Its main goal was to share resources over the network. That is why protocol specifications focused on the operational behaviour. The environment was not considered hostile as it can be described today. Even though the Internet protocol suite has developed from its early versions, it still utilizes the core protocols that were adopted by the ARPANET in the 1980s. [39]

Appropriate security precautions are needed in order to protect hosts against different threats on the Internet. Intrusion prevention systems and firewalls are important aspects of network security, and these types of solutions have been widely deployed. The most important function of an intrusion prevention system is stopping malicious content from reaching its destination. Firewalls are used for limiting network access. A firewall analyses incoming and outgoing network traffic and filters the traffic according to its rules. [30, 69]

Even though intrusion prevention systems bring additional security to the network, they are not bulletproof. There are techniques, such as evasions, which can be used to try to outsmart them. Evasion techniques are used to masquerade attacks in order to avoid detection by security appliances. When an attack is equipped with evasions, it is actually masked to look like legal data from the point of view of a security appliance, such as an intrusion prevention system. That is why it might allow malicious data to reach its destination. Additionally, evasion techniques can be combined. The purpose of combining evasions is to provide means to evade detection, for example, if a single evasion technique is not adequate. [40, 49, 66, 80]

The severity of the threat that comes from evasions and evasion combinations is unclear. It has been reported that these techniques can be utilized to bypass intrusion prevention systems [4, 36]. However, the extent of the effectiveness of evasions and evasion combinations needs further studying.

The goal of this thesis is to evaluate if the state of the art intrusion prevention systems are vulnerable to different evasion techniques and evasion combinations. To achieve this, four million attacks were run against 10 commercial and 1 free intrusion prevention systems. Each attack was disguised with different evasion techniques. Because the magnitude of the experiments is so large, a reliable analysis can be conducted of the effectiveness of evasion techniques.

The results show that intrusion prevention systems are vulnerable to evasion techniques. Even old evasion techniques from the 1990s were still effective against the state of the art intrusion prevention systems. Only one intrusion prevention system was able to block all the attacks that were masked with carefully fine-tuned atomic evasions.

There are millions of evasion combinations that can be utilized in attacks. By using evasion combinations, almost all the tested intrusion prevention systems can be tricked into passing malicious data. So, one evasion technique is not always enough to avoid detection, but combining multiple techniques increases the possibility in finding a way to evade detection. Additionally, the results indicate that normalization and reassembly have not been properly implemented in some of the tested intrusion prevention systems because the number of successful attacks utilizing evasion techniques was so high.

This thesis consists of six chapters. The remainder of the thesis has been organized in the following way. Chapter 2 introduces to network security. It describes why there are vulnerabilities in software, lists multiple ways of attacking and how to protect against various threats. Chapter 3 discusses intrusion prevention systems and evasions. The main focus is on giving detailed information about how intrusion prevention systems function and detect and block attacks. Additionally, evasions are studied to give a thorough understanding about various evasion techniques. Chapter 4 concentrates on the actual study. It describes the environment where this study was conducted, the used methods and how the experiments were implemented. Chapter 5 lists the results of the experiments and analyses them. The last chapter summarizes the results and conclusions that were made in this thesis.

The references used throughout this thesis can be categorized according to the type of a reference. Mostly peer reviewed articles were used, but also books and technical specifications were common sources. Also, various product sheets about intrusion prevention systems and software were used. In limited cases, it is referred to security blogs or other web sites providing information that was not available in the previously specified literature.

2 Network Security

Network security is a crucial aspect in information technology. However, it was not the main concern when the development of the Internet and personal computers started. This chapter discusses various security threats and lists examples of them. First, a brief summary of the birth of the Internet is given. Then software vulnerabilities and the reasons of their existence are covered. The motives of attackers are discussed as well as various of types of attacks are studied. Finally, some security solutions are presented.

2.1 The Birth of the Internet

A key element in the history of the Internet is the development of the ARPANET. It was a packet switched network funded by the Advanced Research Projects Agency (ARPA). The agency, which was later renamed to DARPA, is an agency of the United States Department of Defense. In 1969, the first link in the ARPANET was established between the Stanford Research Institute and the University of California, Los Angeles. In the early 1980s, TCP/IP protocols were implemented into the ARPANET. Over the years, the ARPANET was expanded, and the concept of the Internet started to form. [39, 46]

The main goal of the ARPANET was to share the resources of large service machines over the network. That is why protocol specifications focused on the operational behaviour, and security aspects were not the main concern. The environment was not hostile as it can be described today when various attacks are being deployed over the Internet. [39]

Technology behind the Internet has evolved, but it still utilizes the core protocols that were adopted by the ARPANET in the 1980s. Over the years, flaws were found and fixed to some extent in the protocols and protocol implementations. Some of the flaws have been very serious threats to the network security. However, the protocols were effective from the beginning and thus early adopted in production environments. This led to the current situation where the global economy relies on them. [21, 39]

2.2 The Sources of Vulnerabilities

According to Internet Security Glossary, **vulnerability** is "a flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy." [71] Attackers try to exploit these vulnerabilities. Usually, it is done by using **exploits**. An exploit is a tool that actually takes advantage of the vulnerability [54]. Its purpose is usually malicious [25]. The exploit can be, for example, a small script or a dedicated application.

There are a few organizations, including CERT and MITRE, that inform about vulnerabilities. Also some vendors use security bulletins to provide information about vulnerabilities. There is a public database available, which is maintained by MITRE, and it lists publicly disclosed vulnerabilities. The database is called

Common Vulnerabilities and Exposures (CVE). Each vulnerability is identified by a CVE number. [78]

The poor design of software, protocols and algorithms as well as programming errors are common sources of vulnerabilities, which can be exploited. There are many programming errors, bugs, in any software. Programmers make mistakes such as logic errors and errors related to the used resources. An infinite loop is an example of the former and a buffer overflow of the latter. The buffer overflow happens when data is written to a buffer, but the boundary of the buffer is overrun, and adjacent memory is overwritten. Basically, it means that some memory fragments are modified even though they are supposed to stay untouched by the process. [27, 78]

According to McConnell, the industry average is 1-25 errors per 1000 lines of source code [50]. Depending on the size of the software, its source code can contain even millions of lines. The Linux kernel, for example, has been developed over 20 years. It has been growing during the years of development. Support for new hardware has been made greater, and new features have been added. The source code of the Linux kernel version 3.10 contains almost 17 million lines of code. [32]

One of the key design elements in the Internet protocol suite was the robustness principle [59, 60] that can be expressed as "be conservative in what you do, be liberal in what you accept from others". The correct way of following the robustness principle can be achieved by always sending well-formed data but accepting any data that is possible to interpret. It is guaranteed that the interoperation between different protocol implementations works easily. However, this may lead to different interpretations between network security appliances and hosts, which can be exploited.

One critical aspect is the quality of implementation. If a system is not configured correctly, it might be vulnerable to attacks that could have been avoided by a proper configuration. There are many reasons why faulty configurations are in use. The lack of experience and insufficient training can lead to situations where configurations are inadequate. Even experienced professionals make mistakes, so reviewing of configurations is needed. Human aspect cannot be underestimated. Laziness and carelessness are just some examples of behaviour that can lead to severe security threats. It is also possible that there are not enough employees and administrators do not have enough time to thoroughly configure every system, so corners are cut. [27]

Proper management and monitoring are needed to maintain the level of security. Security policies and measures need to be documented and monitored. Verifying that systems are operating as defined is a mandatory practice. [27]

2.3 Attackers

There are many types of online attackers. They also have various motives for their actions. According to Hyppönen, there are three main groups of attackers which are criminals, hacktivists and nation states. [43]

First on the list is online criminals. They are motivated by money. It is a known fact that cybercrime is a large scale problem [55], and there are many examples of cybercrime. In 2010, Albert Gonzalez was sentenced to 20 years in prison for his role in multiple data thefts. A card-processing company, Heartland Payment Systems, was a victim of theft of over 130 million credit and debit card numbers. Their system was initially breached by using an SQL injection attack. Gonzalez was also sentenced for his involvement in a theft of over 90 million credit and debit card information from TJX and other companies. [3, 90, 91]

Online criminals are well organized, and they can interact with each other. There are, for example, online forums dedicated to such purposes. In 2012, 24 people were arrested in an international operation that targeted carding crimes on an undercover carding forum. The term, carding, is used to refer to the trafficking of illegally obtained credit card information. This kind of credit card information can be acquired, for example, by gaining unauthorized access to the database of an online service. It was estimated that this particular operation prevented a loss of more than $205 million. It was also revealed that over 400 000 credit and debit cards were compromised. [9]

Hacktivists want to bring issues they feel important to public attention. Their motives are political rather than financial gain. Their way of operating varies. Hacktivists may launch, for example, distributed denial-of-service attacks against the organizations they are targeting to take their web services offline. This kind of an attack is covered in detail in the next section. After the United States Department of Justice shut down a popular file sharing site, Megaupload, as well as other file sharing sites, a hacktivist collective, Anonymous, started their attack against the parties involved in the operation. The websites of the FBI, U.S. Department of Justice and several entertainment industry websites, including Motion Picture Association of America, were taken offline momentarily in the attack. [41]

Last on the list is nation states and governments. Due to the nature of the operations, they are not public information. However, it has been documented that they exist. Stuxnet is an example of a complex piece of malicious software, malware, sanctioned by nation states. It targeted industrial control systems at the Natanz plant in Iran, and it is said that due to a programming error, it was able to spread beyond its initial target [67].

However, there are other attacker groups as well. They have their own unique characteristics, even though there are similarities with the previously mentioned groups. Some attackers operate out of curiosity. They want to try to obtain information they are not supposed to have access. Their motive is not to cause any severe damage. [78]

Industrial espionage is also happening online. In 2013, a Norwegian telecommunication company, Telenor, reported that a serious data breach happened against their top executives' computers. Emails, passwords and a high number of other files were stolen. Attackers were able to gain access to their network by using Trojans in email attachments. These emails appeared to come from trusted sources. After opening this kind of an email attachment, malicious software was covertly installed in the target system and a backdoor was opened for the attackers. [22]

2.4 Attacks

Attacks are attempts to bypass security services and violate target systems [71]. They can be classified in two categories. A **passive attack** tries to acquire useful information from a system without affecting it. That is why passive attacks are difficult to detect. However, encryption is a good way to prevent a data loss. Eavesdropping and traffic analysis are two examples of passive attacks. They are also used to gather information that can be used in active attacks. **Active attacks** affect the targeted network or system as the name suggests. That is why they are easier to detect, but they can also be fatal for the functionality of the system. [27]

The following paragraphs list some examples of different active attacks. It is also considered that attacks can spread because users are not cautious enough. The timeline of an attack is also studied.

As previously discussed, software contains bugs and vulnerabilities. Various software vulnerabilities are exploited in attacks against client-side software. Usually, these attacks are targeted against the software that is wide-spread and used very commonly. Web browsers, for example, are included in many systems, so they are intriguing targets for attackers as well as some of the most popular operating systems. Patching is an important process to prevent the unnecessary exposure of systems to the attacks against vulnerable software. [27, 78]

A **distributed denial-of-service** (DDoS) attack is a well-known way to cause harm. Its purpose is to prevent users from using the targeted service. In the distributed denial-of-service attack, the actual attack is deployed from many sources. It can be achieved, for example, by sending lots of packets, from multiple sources, which consume a key resource of the service. The DDoS attack will exhaust the resources of the service so that it will be unavailable for its users. DDoS attacks may also have great financial implications, for example, if a company cannot provide their service to their users reliably, some users might switch to another service. [52]

Web servers are targeted because they can have valuable information to attackers. Databases behind websites can contain confidential data, such as email addresses, passwords and credit card information. One way of attacking these databases is using **SQL injection** attacks. SQL is a language that is used to manage databases and query data. A website can take a user input from a web form, and it is passed to an SQL statement that is then executed. If the user input is not properly filtered, it can allow attackers to embed SQL statements in the input data that are then passed to the database. By crafting the user input to contain various SQL statements, attackers may gain restricted information. [78]

The actions of users can have serious consequences. Users may download and install software from untrusted sources. After installing the software, the system may seem to operate as previously, but it actually contains a **Trojan horse**. It is a malicious piece of software that covertly operates in the background. The downloaded software does what it promises, but it can also steal the user's data. [27]

Attacks are not restricted to be technical. Users are threats to the security of networks and systems as well as vulnerabilities to software. Users can be convinced to reveal confidential information to attackers. This is called **social engineering**. An employee can, for example, receive a call from a person who pretends to be from the IT support of the company. The attacker can use the actual names of people who are working in the company to be more convincing and gain the trust of the victim. The attacker can claim that there is a problem with the systems and ask the employee to help with the issue by performing certain tasks. The employee might get frustrated after failing to perform the tasks, which are actually impossible to perform, and start to feel some pressure when the person from the "IT support" starts complaining. Then the attacker may just ask the username and password of the employee so that they can finally resolve the problem together and continue to work with their actual tasks. [27]

Social engineering is used as a way of gaining access to systems by fooling the victim to reveal confidential information. There are many tricks that can be used when trying to gather confidential information from the victim. Basically, the technique is the same as it is with con artists who pretend to be somebody else. It is important that users know that they are not supposed to disclose their passwords to anyone. As previously mentioned, all the technical solutions can be hacked without the knowledge of their way of operating. There is no need to know if the systems are exposed to certain vulnerabilities if the users are willing to disclose crucial information quite easily. [27]

Phishing is also a form of social engineering. It means that attackers try to lure users into giving their personal data by pretending to be a trustworthy entity, such as a popular web site. Users may receive phishing emails with a link to a website that looks legitimate, but actually it is a fake. The sender of the email may also seem to be valid if it is not carefully studied. [78]

The timeline of an attack is shown in Figure 1. There can be two significant phases in the attack that are illustrated in the figure with large boxes and listed as A and B, respectively. In the first phase, the box A in the figure, the attack can take advantage of previously unknown vulnerability. Previously unknown means that no one has yet disclosed the vulnerability publicly. This type of an attack is called a **zero-day attack**.

As seen in Figure 1, after new vulnerability is found (the second small box in the figure) in software (1), an exploit is created (3). Before the vulnerability is public (5), a vendor might already have discovered it (4), but it is not necessarily the case. It is possible that the vendor discovers the vulnerability only after it has gained publicity. Zero-day attacks are dangerous because products are vulnerable until the vulnerability is publicly disclosed (5) and patches are implemented and deployed (6-7). Of course, there are not always zero-day attacks conducted against each vulnerability because the vulnerability is not discovered by attackers before it is made public. In such a case, there is only one significant attack phase that begins after the vulnerability is announced publicly. [24]

The second phase starts when the vulnerability becomes public, which is illustrated as a box B in the figure. The developers of the targeted product have to implement a patch to protect the product against that particular threat, and it usually takes some time to release it. Meanwhile, if there were zero-day attacks,

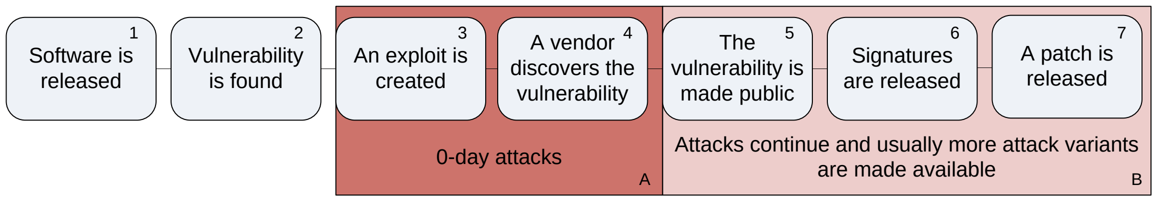
1 Software is released
2 Vulnerability is found
3 An exploit is created
4 A vendor discovers the vulnerability
5 The vulnerability is made public
6 Signatures are released
7 A patch is released
0-day attacks
A
Attacks continue and usually more attack variants are made available
B

Figure 1: The timeline of an attack

the same attacks tend to continue. They are no longer considered zero-day attacks, but they are still a serious threat if hosts are not protected. Usually, the number of attack variants also increases after the publication of the vulnerability. If zero-day attacks were not conducted, attackers still can try to exploit that particular vulnerability, since all of the hosts may not be patched or protected. Theoretically, after the patch is deployed to every host, the attack becomes pointless. It also means that the window of exposure is closed. [24]

In a recent study, Bilge and Dumitras stated that the duration of zero-day attacks varies from days to years, and they last 10 months on average. This means the duration between the first time an attack is conducted and the vulnerability is publicly disclosed. Although most of the zero-day attacks are targeted against few target systems, sometimes the magnitude can be larger. For example, the Conficker worm infected hundreds of thousands target systems before it was detected. [24]

Previously mentioned Stuxnet is an example of a targeted attack. It is a complex piece of malware, which exploited four different zero-day vulnerabilities. [37] It is considered a high-profile attack. [24]

2.5 Security Solutions

There is a need to protect hosts that are connected to the Internet. These hosts may be, for example, desktop computers or servers. Entire network segments including private company networks (intranet) need protection. That is why intrusion prevention systems (IPS) and firewalls were developed to improve the network security. These devices can be used to protect large networks or just a single host. [75]

An **intrusion prevention system** is used for identifying and logging malicious content in network traffic. Its most important function is to stop malicious content from reaching its destinations. [69] Section 3.1 discusses intrusion prevention systems in detail.

A **firewall** is a security system that limits network access. It can be software or a separate device. It analyses incoming and outgoing network traffic and filters the traffic according to its rules. Firewalls can also be deployed to the host itself, for example, operation systems can include a firewall. [30]

Firewalls have developed over time from packet filters to rather complex next-generation firewalls. There are four groups of firewalls that are briefly studied in the following paragraphs. All the firewall groups are presented in the chronological order of the development. [30, 87]

First firewall solutions were rather simple packet filters. A packet filter operates by dropping packets that match its rule sets. The rule sets include a combination of the source and destination addresses of a packet, a protocol that is used and port numbers. Packet filters do not keep track on the information about a connection state. They make decisions based on the current packet. Thus, it is not taken into consideration whether the packet is a part of an existing stream. Packet filters operate up to the network layer of OSI model, but they also operate at the transport layer because they need access to port numbers. [30]

Stateful firewalls operate as packet filters, but they also keep track on the state of connections. They operate at the transport layer of OSI model to be able to inspect connection states. Stateful firewalls determine if a packet is used to open a new connection, or if it is part of an existing connection. It is dropped if it is not a part of any connection, and it is not opening a new connection. [30]

Application layer firewalls operate, as the name suggests, up to the application layer of OSI model. An application layer firewall has the functionalities of a stateful filter, but it is also able to understand various application layer protocols, such as the HTTP protocol. It makes possible to block connections if the semantics of a protocol is not followed properly, or a disallowed protocol tries to bypass the application layer firewall using an allowed port. [30]

A next-generation firewall extends the functionalities of an application layer firewall by integrating the functionalities of an intrusion prevention system. Next-generation firewalls are application-aware solutions that are capable of performing deep packet inspection. This technique is studied in Section 3.2.3. [87]

A simplified example of the differences between firewalls and intrusion prevention systems is the way they operate. Firewalls block all network traffic except the network traffic that is explicitly allowed. Intrusion prevention systems permit all network traffic except the network traffic that is considered malicious and therefore explicitly disallowed.

Antivirus software is used to protect hosts from being infected by malware. It is also used to scan systems for the presence of malware. In case of an infection, antivirus software is responsible for removing the malicious content out of the system. Antivirus software provides protection against various threats, not just against viruses as implied by the term. There are also various commercial and free solutions available. [78]

2.6 Summary

Security was not the main concern when the Internet protocol suite was designed. The environment was not as hostile as it is nowadays. Also, the goal of the predecessor of the Internet, the ARPANET, was to share resources over the network, and thus security was initially overlooked.

Systems have weaknesses, vulnerabilities, in their design, implementation or operation and management. Their severities vary but they exist. Reasons for vulnerabilities can be, for example, the poor design of software or programming errors. Also, the quality of implementation has a significant role. Systems need to be properly configured to avoid unnecessary exposure.

Attackers exploit these vulnerabilities. There are three main groups of attackers that are criminals, hacktivists and nation states. Various types of attacks are implemented including DDoS attacks and SQL injections. A zero-day attack takes advantage of previously unknown vulnerability, which makes it difficult to detect such attacks.

There are various security solutions available to improve the network security. Intrusion prevention systems, firewalls and antivirus software were developed to protect hosts and networks from attackers.

3 Intrusion Prevention

This chapter studies intrusion prevention systems in detail and discusses how this type of devices operate and detect attacks. An overview of different evasion techniques is presented with a note how these techniques can be put together to create evasion combinations. Some tools are also listed that can be used to test various evasion techniques against intrusion prevention systems. Finally, research is presented that is relevant to this thesis.

3.1 Intrusion Prevention Systems

An intrusion prevention system is a network security appliance that is used for identifying and logging malicious content in network traffic and the most importantly stopping it from reaching its destination. Intrusion prevention systems work inline, so the network traffic goes through an IPS device. IPS devices are also capable of producing summarizing reports about the monitored threats. [69]

Intrusion prevention systems can be categorized as an extension to intrusion detection systems (IDS). An IDS is able to identify and log malicious content in network traffic, but it does not necessarily try to prevent any attacks. Intrusion detection systems are passive systems that get a copy of the network traffic which can then be monitored and logged. Thus, an IDS is used to monitor the network traffic. [69]

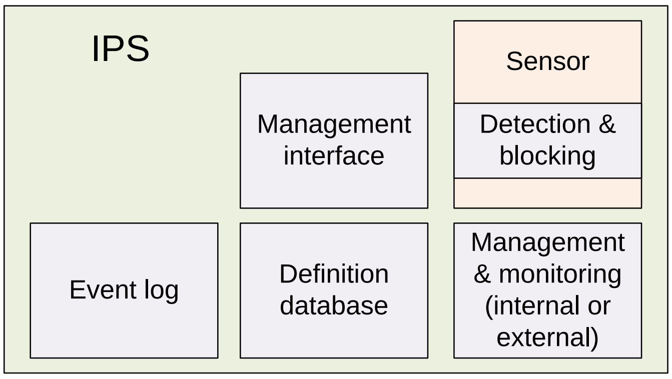

Figure 2: The components of an IPS

The components of an IPS can be seen in Figure 2. Intrusion prevention systems use sensors or agents to monitor and analyse the network traffic. They try to detect malicious content in the network traffic by using various detection methods. Devices contain, for example, a definition database of malicious content that is used in the detection process to determine if the data is legal or not. Since the network traffic goes through an IPS device, it is also able to prevent attacks by blocking them. Also, a log of the detected events is stored in a database. Devices can be administrated via a management interface. For example, updates can be installed via this interface. Some devices have an external management server, which receives information from the sensors. The management and monitoring capabilities can also be internal. Then the IPS is a stand-alone device. [69]

Intrusion detection systems have been commercially available from the 1990s [85]. Since there were appliances available, there were always techniques to outsmart them. As an illustration of the situation Fred Cohen published an article mentioning 50 ways to defeat intrusion detection systems in 1997 [31]. So, basically from the beginning of the industry, it has been seen that intrusion detection systems and intrusion prevention systems might be vulnerable to different tactics even though they bring additional security to the environment where they are deployed.

The process of mistakenly declaring clean network traffic as malicious (or malicious traffic as clean) is called having false positives (or false negatives), respectively [69]. Having lots of false positive results in a detection process can be potentially dangerous because it is very difficult to find an actual attack if the log is full of entries. For example, if an administrator has a reason to believe that the log is full of false positives, the administrator might allow all the previously blocked network traffic of that type. The decision might be based on the fact that the network traffic causing the alarms was created by certain software that is previously known to be valid. The problem is that the decision may open the door for attacks too. False positives can also cause inconvenience to the authorized users since their legal traffic may be blocked without a proper reason. This can happen, for example, if the quality of signatures or rules is somewhat questionable in intrusion prevention systems. The use of signatures is covered in Section 3.2.1.

A false negative can be dangerous because then the device passes malicious content through. This can happen due to many reasons, for instance, the device might not have up-to-date rules and signatures, it does not perform proper traffic normalization, or it lacks resources to perform thorough analysis.

False positives were a problem in the early years of intrusion detection [69]. Devices generated a large number of alerts that made it difficult to notice an actual attack from the log [53]. Administrators also needed to spend lots of time fine-tuning the system. IDS stimulator tools, for example, Stick [53], PCP [56] and Snot [53, 56], were developed to generate network traffic that triggers lots of alerts in the IDS systems. These tools used signatures from an open source system, Snort, to create the traffic in the way it would cause the alerts. [53, 56] In 2003, Gartner research company stated that "intrusion detection systems are a market failure". One of the reasons behind the argument was that devices generate too much false alarms. Devices also had problems with detecting attacks. [2] Nowadays intrusion prevention systems are improved from the early years, and false positives are not such an issue [69].

Intrusion prevention systems[1] are traditionally either network-based or host-

[1]From now on, unless stated otherwise, the term IPS is used throughout this thesis to unify the used terminology. The main operation and detection mechanisms between intrusion detection systems and intrusion prevention systems are the same, and the only big difference between the devices is that an IPS is also capable of preventing attacks. Also, the term IPS is used when referring to papers published before IPS devices were publicly available because the papers refer to detection mechanisms.

based, but there are other solutions available, such as wireless intrusion prevention systems and network behavior analysis systems (NBA). A **network-based** intrusion prevention system is commonly a stand-alone, possibly rack-mounted, appliance that is placed in the network to monitor and prevent attacks for certain network segments. This type of devices can be deployed, for example, near border firewalls or routers. A **host-based** intrusion prevention system is usually deployed to the host itself. It is responsible for monitoring and preventing attacks against that particular host. A **wireless** intrusion prevention system is used to monitor wireless network traffic. It analyses wireless protocols to detect malicious activity. It can also be used to detect rogue access points and rogue wireless networks. A **network behavior analysis** system tries to detect threats that cause uncommon network traffic flows, for example, distributed denial-of-service attacks. [69]

Nowadays virtualization is one of the key elements in information technology. To follow that trend, vendors have also published virtualized intrusion prevention systems [12, 15, 16]. Additionally, there are also other types of appliances, such as unified threat management solutions [89] and next-generation firewalls [87] that are capable of performing tasks which are traditionally assigned to intrusion prevention systems.

3.2 Detection Methods

Intrusion prevention systems can use multiple detection methods including signature-based detection, statistical anomaly-based detection and stateful protocol analysis. Sections 3.2.1 through 3.2.3 discuss more these methods. Intrusion prevention systems usually utilize multiple detection methods in order to perform as well as possible. Different detection methods can be used separately or in an integrated way. [69]

3.2.1 Signature-based Detection

A signature is a predefined pattern that corresponds to a threat. A pattern can be, for example, a specific piece of a string that describes uniquely the characteristics of an attack. A simplified example of the pattern can be a string containing the text "attack.exe", which is found in the attachment field of an email, where this text refers to a known filename of a virus. [69]

Intrusion prevention systems have a database that contains signatures against a variety of threats. In signature-based detection, an IPS tries to match the network traffic against these signatures to detect and prevent attacks. [69] The quality of the process depends on how well the signatures have been defined [80]. Also, the capability of reassembling and normalizing the traffic correctly is important [29].

The signatures should be based on vulnerability instead of an exploit. Not all the attacks are entirely new. Instead, they can be based on a previously known exploit. Exploit-based signatures are very ineffective against different variations of an exploit because the new variation of an attack does not match against the old signature. [23, 25, 82] For example, if the filename of the previously mentioned virus was changed from "attack.exe" to attack2.exe", it could not be detected by the same exploit-based rule [69]. Vulnerability-based signatures try to understand the vulnerability in order to be able to block all the attacks trying to exploit that particular vulnerability. In that way, only one properly made signature against vulnerability is enough to block all the different attack variations, even polymorphic variants, using that vulnerability. [25]

Signature-based detection is considered an effective method of detecting previously known threats [69]. However, it has some weaknesses. It cannot detect zero-day attacks because an IPS does not have the signatures of zero-day attacks until the vendor releases them and the signature database of the device is updated [85]. Sometimes it is considered that signature-based detection includes stateful protocol analysis, which is discussed in Section 3.2.3 [69].

3.2.2 Statistical Anomaly-based Detection

Statistical anomaly-based detection means that an IPS can detect incidents by comparing network traffic against the definitions of the network traffic which is considered normal. IPS devices are trained over the time to understand what kind of traffic is normal in that particular network or host. IPS devices have profiles that contain representations of normal behaviour of different types of instances, for example, network connections, hosts and applications. [69]

In this type of detection, an IPS analyses by using statistical methods whether the traffic falls within the correct thresholds according to the profiles it uses. The profiles can be either static or dynamic. A static profile remains the same after initializing it, but dynamic profiles develop over the time. The static profile can be a good choice if the nature of the network traffic is not going to change. [69]

Anomaly-based detection methods can be used to detect zero-day attacks, which is considered one of the advantages of this type of detection. When signature-based detection relies on proper signatures, anomaly-based detection can notice significant deviations in network traffic that might be caused by an attack. One of the problems with anomaly-based detection is that it may cause many false positives because legal activity may differ from the profile representations. For example, each time a new application is taken into use in the network, the network traffic might change just enough to cause false positives. [69]

3.2.3 Stateful Protocol Analysis

Stateful protocol analysis means that network traffic is compared in each protocol state against the definitions of the network traffic which is considered normal by vendors. The basis of stateful protocol analysis is the use of the universal, vendor-made, profiles of normal and valid protocol operation. The analysis is also stateful, so it is able to keep track of the protocol state. Thus, attackers cannot exploit different protocols by performing actions that are not allowed or valid in that particular protocol state. [69]

The protocol models that are used in stateful protocol analysis usually follow protocol definitions by software vendors and standardisation authorities, for example, Request for Comments (RFCs) from Internet Engineering Task Force. The models also need to take into consideration that protocol implementations may differ. Also, the use of proprietary protocols makes the detection process more difficult because vendors usually do not have all the details and semantics of a proprietary protocol. [69]

Stateful protocol inspection is an effective way to detect attacks, but it also consumes lots of resources. The analysing process is quite complex, and the heavier network load, the more demanding it is to keep on track of the increasing number of different connections and their states. Also, the term deep packet inspection is sometimes used to refer to stateful protocol analysis. [69]

3.3 Evasions

Evasions are techniques to masquerade attacks to avoid detection by security appliances. Basically, with the help of evasions, attackers try to mask the attacks so that security appliances are not able to detect and block them. [61]

There are a lot of techniques for evading IPS devices [28, 61, 62], and these different evasion techniques can also be combined to create new successful attacks. This process of combining evasion techniques was considerably studied in the 2000s. [40, 49, 66, 80] Evasion techniques are also known by names such as variations [49] and transformations [66].

Figure 3 illustrates a simplified situation of attacks with and without evasions. First, an attacker wants to exploit a vulnerable service but does not apply evasion techniques when trying to deliver the exploit to the target system. The IPS device is able to block the attack. Second, the attacker uses the same exploit, but this time evasions are also deployed. Now the IPS device just sees legal data and passes all the data to the vulnerable target system. In the target system, the data is interpreted correctly, and that is why the exploit can be run. Now the attacker can have, for example, a control over the vulnerable service or even the target system.

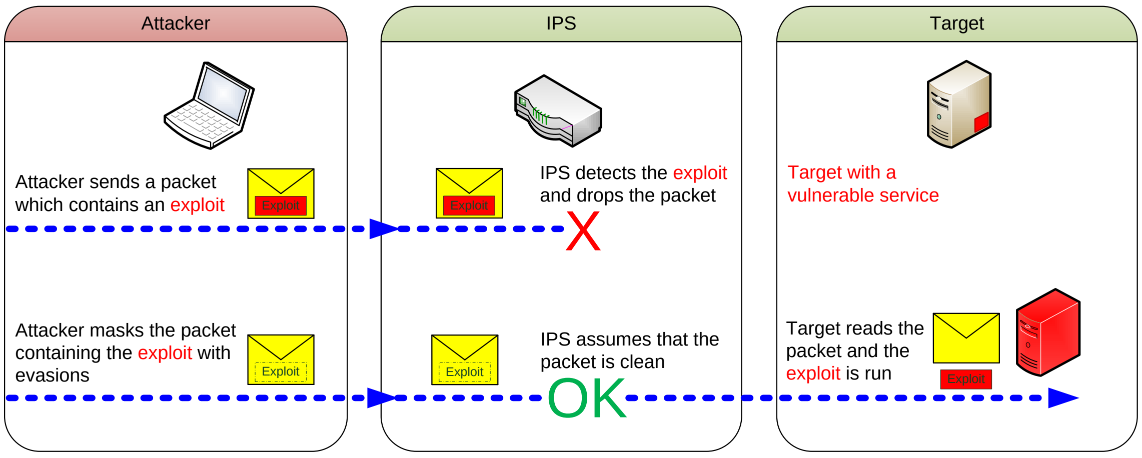
Attacker
IPS
Target
Attacker sends a packet which contains an exploit
Exploit
IPS detects the exploit and drops the packet
Exploit
X
Target with a vulnerable service
Attacker masks the packet containing the exploit with evasions
Exploit
IPS assumes that the packet is clean
Exploit
OK
Target reads the packet and the exploit is run
Exploit

Figure 3: An attack masked with evasions

The robustness principle makes it possible to exploit the fact that all the received traffic is always interpreted [10, 60]. The evasion techniques come in the picture because intrusion prevention systems and end-systems may not always interpret or process the incoming data similarly [61]. So, even though the robustness principle brings interoperability between devices, it also causes problems with different interpretations. Different protocols have numerous possible options and parameters that are not used very often, so it is possible that different devices end up having differently interpreted data steams. Attackers may also craft the network traffic without following the correct protocol semantics since, according to the robustness principle, the end-systems still have to try to interpret all the incoming traffic. [10, 60, 61, 68]

It should be noted that evasions are not restricted to uncommon protocol options. Much used protocol features can be used as well to try to bypass the IPS detection mechanisms. For example, TCP has mechanisms, such as flow control and congestion control, for ensuring reliability and a good performance through different networks. Even these aspects can be exploited [7]. Additionally, the capabilities of intrusion prevention systems are not unlimited, so it is possible that a device misses an attack because it lacks resources when reassembling the data [29].

Many of the evasive techniques can be categorized according to the way they operate. In Sections 3.3.1 through 3.3.4, a few techniques are listed that can be used for evading IPS devices or other security appliances. It is worth noticing that some of the categorized techniques may be slightly overlapping with each other.

3.3.1 Denial-of-Service and Timing Attacks

Denial-of-service attacks are used to overwhelm all the available resources of an IPS, such as the memory and CPU usage [65]. There are a few possible outcomes when the denial-of-service attacks succeed. Devices might start operating in a fail-closed state where they block all the traffic or go to a fail-open state where they pass all the traffic. [61] Resource exhaustion makes it possible that an attack goes through an IPS undetected [57, 73]. For example, the rule matching algorithms of the IPS can be exploited so that packet processing times slow down significantly. Then the device might be too busy to handle everything properly. [73] The use of denial-of-service attacks is a very visible way to attack since its primary purpose is to make the attacked resource unavailable [65]. It can be used to evade an IPS device, but other techniques provide stealth approaches. [57]

Timing attacks are managed by delaying the sending of packets and exploiting different timeouts between intrusion prevention systems and end-systems. It is possible that by delaying packets an IPS device might lose the track of the stream because the detection process takes too much time. In that case, when receiving the next new packet after timeout, the IPS device decides to pass the data. The data is clean itself, but when it is combined with the previously received data in the end-system, it is malicious. [26, 72]

3.3.2 Uncommon Protocol Fields

The use of uncommon protocol fields is a way for evading intrusion prevention systems [10]. As previously mentioned, IPS devices and end-systems do not always interpret or process all the incoming data similarly [61]. That is why it is possible to exploit the use of uncommon and rarely used protocol fields. Since some of the protocol fields are rarely used, they are not implemented consistently into every device, and the handling of this kind of data may be different [69]. Also, the robustness principle requires that all the devices at least try to interpret the incoming traffic [59, 60]. Different evasion techniques using uncommon protocol fields can be deployed, for example, with TCP [84**?**].

Attackers can also deliberately violate some of the protocol semantics [61]. This tactic is quite widely used since there are lots of attacks targeting a specific protocol [23, 28]. The more complex protocol, the more difficult it is to interpret correctly by any device, and therefore intrusion prevention systems are required to have a deep understanding about different protocols to be able to make right interpretations and decisions [69]. Server Message Block (SMB) protocol an example of a complex protocol that can be targeted with many different evasion techniques [7, 28, 36].

3.3.3 Packet Splitting Techniques

Packet splitting techniques are used to exploit the fact that IPS devices are not always able to reassemble data streams correctly [57, 61]. Due to limited buffer sizes intrusion prevention systems may not have enough resources to correctly reassemble all the incoming data to detect possible attacks if packets are split to, for example, small fragments or segments. Basically, in this case an IPS device tries to match the incoming data against attack signatures, but it is not able to find anything malicious or even suspicious since the data is not properly reassembled. [29]

Packet splitting techniques are also widely deployed with duplicate insertion techniques. Duplicate insertion means the use of duplicate or overlapping segments or fragments. [61] An illustration of an attack with duplicate fragments is shown in Figure 4 [61, 72]. In this example, the data, sent from the attacker, is split into five fragments. First, the attacker sends successfully fragments 1 to 4. In order to fool the intrusion prevention system, the attacker sends again fragments 3 and 4, but this time the payload differs from the previous data although all the header fields remain the same. Finally, the attacker sends the last fragment. In this example, the IPS device is not able to reassemble the data stream like the target system. The IPS device just ignores the latest fragments, which are duplicates, and reassembles the data by only using the original fragments. That is why it sees just some legal data. The end-system reassembles the data by using the latest fragments instead of the original fragments. Thus, the end-system reassembles totally different data steam, and the content is malicious. [61]

Like stated previously, it is possible to send overlapping fragments. It means that the IP header offset values are ambiguous so that, for example, a newly received fragment may actually partly or fully overwrite an earlier fragment. The key is, again, to reassemble the data correctly. These kinds of packet splitting techniques can be

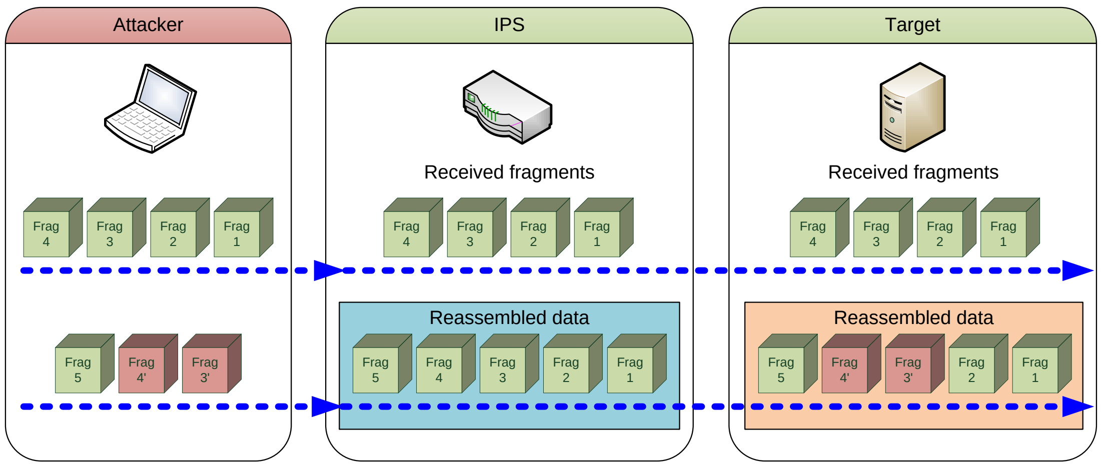

Figure 4: Evasion with duplicate fragments

applied to IP with the use of fragmentation. These techniques also work similarly with TCP when using segmentation. There are also lots of various techniques, that can be used, including sending fragments or segments out-of-order or splitting them to different sizes. [61] Different operation systems can also reassemble fragments differently, which can create challenges for intrusion prevention systems. They can favour, for example, the first received fragment for each offset in the packet or the last received fragment. [70]

Additionally, time to live (TTL) values can be used to evade the IPS detection mechanisms. With the help of duplicate segments and too small TTL values, it is possible that some of the segments get dropped before reaching the end-system, but the segments with long enough TTL values are able to reach it. [57, 61]

3.3.4 Obfuscation and Encryption

Obfuscation means that, for example, a piece of code is turned into looking entirely different, but it still functions the same way as before [36]. Usually different obfuscation methods are used to hide an attack payload, for example, a shellcode, in order to make it difficult for IPS devices to catch the attack [80]. In that case, an IPS device cannot any more rely on its own attack fingerprints because the new, obfuscated attack has an entirely different fingerprint [38]. It is also possible to use different encoding techniques to transform a payload to look different. This is usually done, for instance, when the true meaning of an HTTP request is wanted to hide. [63, 80]

Encrypting the data or the use of tunnels provides means for evading an IPS device [31]. Encryption is an important aspect to provide secure network communication between hosts. It is generally used in many good ways, but it also makes it difficult for intrusion prevention systems to inspect data streams and detect attacks since they cannot really do a proper analysis of the data. This can be achieved with the help of various tunneling technologies. Of course, the use of tunnels demands that an attacker is able to establish a secure connection, for example, an SSH tunnel, to the target system or network. [26]

3.3.5 Tools

Many tools have been released that can be used for testing security appliances against evasive techniques. These tools usually contain components that are shown in Figure 5. They are able to generate network traffic including certain attacks that are masked with a user-defined evasion technique or multiple evasion techniques. Some tools do not contain any attacks but just craft the network traffic with evasions, originating from the local host to the target host [74]. Tools also have a management interface, so users can select the attacks and evasions or possibly create automatized attacks. The following paragraphs list many tools that can be used to test IPS devices.

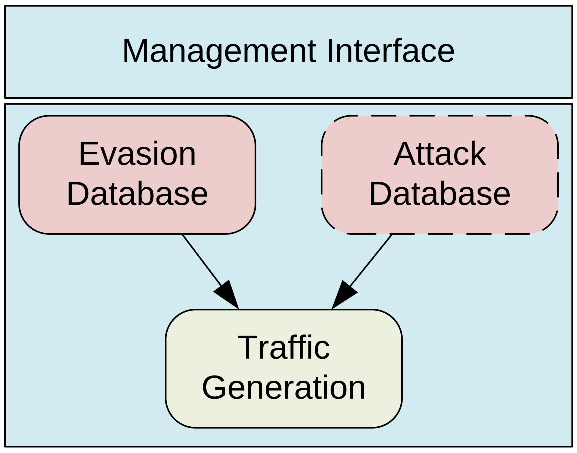

Figure 5: The common components of evasion tools

Fragroute is a testing tool, by Dug Song, which implemented IP and TCP protocol evasions, such as IP fragmentation and TCP segmentation [74]. It is also capable of combining user-specified evasions [66, 74]. In 1999, Rain Forest Puppy released a tool, **whisker**, that introduced application layer evasions using HTTP protocol [62]. A web scanner, **Nikto**, also uses whisker's libraries to include its functionalities [76]. A program called **SideStep**, which was implemented by Robert Graham, contained FTP evasion techniques that made it possible to exploit an FTP command stream by inserting Telnet control sequences into the stream [80]. **ADMmutate** was the first tool that utilized a polymorphic technique to mutate a shellcode for evading signature-based intrusion prevention systems. Afterwards other solutions of this type were also implemented [34]. A penetration test tool, **Metasploit**, also has capabilities to evade IPS detection [13].

There are many tools implemented combining different evasion techniques to create new modified attacks. **Thor**, released in 2002, is an automated tool supporting attack variations, which can be used to try to evade intrusion prevention systems [49]. In 2004, Vigna et al. [80] published a paper describing a more sophisticated framework, later named as **Sploit** [19], which is capable of generating mutant exploits. The framework used different evasion techniques at multiple layers (network, application and exploit) that were combined to create lots of variations of exploits to test the signatures of intrusion prevention systems. **AGENT** is an automated tool including transport and application layer evasions to combine new attacks. The paper describing the architecture of AGENT was also published in 2004. [66] In the same year, Gorton and Champion [40] continued Min G. Kang's work by doing some modifications to the previously released **Mendax** IDS evasion program. They concluded that even if intrusion prevention systems are able to catch all the attacks using only one evasion technique at the time, they still cannot always detect the attacks disguised with the same evasions when combining them together. In 2008, another solution called **idsprobe**, which is capable of combining multilayer evasions, was released [44].

In 2010, a security company called Stonesoft gained publicity by discovering Advanced Evasion Techniques (AETs) [5]. They were defined as "any evasive hacking techniques that allow an intruder to bypass security detection during a network-based attack" [1]. Later, in 2012, Stonesoft published a tool [7, 11], **Evader**, which was designed for testing network security devices against these techniques.

The idea behind AETs is to simultaneously execute different evasion techniques possibly at several OSI layers with different protocols [1]. By combining different evasion techniques and changing their parameters it is possible to create millions of different evasion combinations. Intrusion prevention systems are required to do a proper normalization in order to be able to detect attacks that are tried to mask with these techniques. [7, 11] The Evader tool was used throughout this study.

3.4 Evasion Research

Evasions have been a well-known issue in the network security industry for a long time [57, 61]. In 1998, Ptacek and Newsham published a seminal paper about evading network intrusion prevention systems with the help of many different techniques including IP fragmentation, overlapping IP fragments and TCP segmentation. Their main focus was on network and transport layer evasions, namely, TCP and IP evasions. The basis of these kinds of attacks relied on the fact that intrusion prevention systems and end-systems may not always process or interpret the incoming packets similarly [61].

In 1998, roughly when the paper of Ptacek and Newsham was published, Paxson also presented some of the same techniques in his pioneering paper [57, 61]. Even today some of the relatively old techniques presented in the studies of Paxson or Ptacek and Newsham are valid for evading IPS devices [29, 36].

It has been proposed that fragmentation can also cause problems in IPv6 [18]. Fragmentation is only allowed by source nodes in IPv6 and not by intermediary devices like in IPv4 [33]. It is also recommended that overlapping IPv6 fragments should be disallowed to prevent any foul play [45].

Evasion techniques are not restricted to network and transport layers, but also other protocols from different OSI layers are proven to be vulnerable [28]. In 2006 at Black Hat USA conference, Caswell and Moore [28] demonstrated that evasions are possible at every OSI layer. There are possible evasion techniques even at data link layer although it is not very practical to use them because local media access is needed. In the same conference Caswell and Moore presented already known evasion techniques, but also some new SMB and DCE/RPC protocol evasion techniques were studied [23, 28].

Evasions of application layer protocols, such as HTTP, are a well-documented topic. In 1999, Rain Forest Puppy published a pioneering paper concentrating on HTTP evasion techniques [62]. The idea behind these techniques was to mutate an HTTP request in a way that IPS devices do not understand the true meaning of the request, but the end-system still interprets it right. A classic example of this behaviour is an HTTP request that is encoded so that its string representation seems to be valid because it does not match against any known signatures of an IPS. For instance, a character 'i' can be encoded as '%69' in the request, and if the IPS is not able to decode it as 'i', the IPS fails to understand the request correctly.

Later, Rain Forest Puppy continued his work by stating the problems of UTF-8 Unicode encoding with requests [63]. Before that, mainly evasions of ASCII encoding had been under thorough investigation in the intrusion prevention systems. This work was extended by Daniel Roelker who evaluated two different types of HTTP evasion techniques. Those techniques were invalid protocol field decoding and invalid protocol parsing. The previously given example of encoding falls into the former category. The latter means that an IPS is not able to parse the request correctly which can happen, for example, when multiple requests are pipelined to a single request.

Usually, an attack can be divided into a few parts. First, you need to know the vulnerability to take advantage of and select an exploit against it. This way it is possible to enter the target system undetected. Second, you need to inject a shellcode to the attack payload. The shellcode is a piece of code carried in the attack payload that is run in the target system, for example, to gain remote access to the system. Third, you need to encode the payload, which can be done by using some shellcode encoding method. Encoding is done to avoid the signature-based detection of the shellcode. So, different obfuscation techniques can be used to try to hide the payload of the attack. Additionally, evasions can be used to mask the execution of the attack. [13, 61]

Obfuscation techniques at the exploit layer usually apply polymorphism or different encoding techniques [80]. Polymorphism is a technique to mutate a piece of code to look different each time it is run, but it still functions the way it did before mutating. With the help of polymorphism, attackers can be able to evade intrusion prevention systems' pattern matching techniques since they do not necessarily recognize the signatures after a mutation. Polymorphic engines are used to create a polymorphic shellcode. Even though polymorphic techniques were able to evade signature-based detection systems, anomaly-based systems detected these attacks because of the statistical anomalies they caused. Fogla et al. introduced polymorphic blending attacks in order to make these attacks look like normal traffic so that anomaly-based intrusion prevention systems would not catch them. [38]

Various countermeasures have been proposed against evasions. In 1999, Paxson and Handley [42] suggested that traffic normalization combined with reassembly could be used against the exploiting of protocol ambiguities. The purpose of traffic normalization is to eliminate potential protocol ambiguities so that the interpreted network traffic is seen in the same way by the intrusion prevention system and the target host. Several other studies about traffic normalization were also conducted [81, 83]. An entirely different approach [17, 79] was also suggested where reassembly does not occur. Then, the signatures of intrusion prevention systems are split into shorter strings, and those partial signatures are tried to match against the small segments of network traffic. The benefit of this technique is lower memory consumption. Besides these methods, some alternative techniques [64, 70, 77] were presented where an intrusion prevention system tries to interpret the network traffic in the same way as the target host with the help and information it receives from the target host or infers from the network traffic.

As discussed earlier, for example, HTTP requests can be encoded to look different. The process of decoding and normalizing these requests is an important part of detecting any misbehaviour. It has been also suggested that an IPS can work together with a web server so that the web server sends the requests back to the IPS in the format the server sees them [35].

The detection of a polymorphic shellcode can be problematic as previously discussed. However, there is a solution available that utilizes network-level emulation to overcome the issues [58]. A CPU emulator can be embedded in an intrusion prevention system where it executes all the potential instruction sequences in order to detect a specific behaviour that indicates the presence of a polymorphic shellcode. Bania [20] has addressed some limitations of this technique.

Since the security of intrusion prevention systems and the implications of different evasion techniques are topics that interest many parties, many different evaluations of intrusion prevention systems have been conducted. One well-known actor in evaluations was the Defense Advanced Research Projects Agency (DARPA) who funded two different evaluations of intrusion prevention systems in 1998 and 1999 [47, 48]. The later study included evasion techniques [47], but the DARPA evaluations were also criticized to be questionable regarding the test methodologies, which could have led to biased results [51]. Nowadays different independent organizations evaluate security devices, including intrusion prevention systems. NSS Labs and ICSA Labs, for example, are such evaluators.

Recently some studies were published about evasions against intrusion prevention systems, but they were restricted to a relatively limited number of evasion techniques or commercial IPS devices [29, 36]. However, there is a recent study available with a large number of tested commercial intrusion prevention systems, but the number of attacks, containing various evasions, ran against IPS devices is very limited [86].

3.5 Summary

This chapter studied intrusion prevention systems and their functionalities. An IPS is used for identifying and logging malicious content in network traffic. The most important functionality of the IPS is stopping malicious content from reaching its destination.

Evasions are techniques to masquerade attacks to avoid detection by security appliances. There are lots of different techniques that can be used for evading detection. Basically, with the help of evasions, attackers try to mask the attacks so that security appliances are not able to detect and block them. Different evasion techniques can also be combined to create new successful attacks. The process of combining evasion techniques was considerably studied in the 2000s.

4 Methods and Implementation

This chapter discusses the hardware and the software that were used in the experiments. The methods are also studied which were used to create reliable experiments as well as the actual implementation of the experiments is explained.

4.1 Environment

The environment of the study needs to match certain criteria in order to evaluate the effectiveness of evasion techniques against intrusion prevention systems. Sections 4.1.1 through 4.1.3 cover the used network topology, the intrusion prevention systems that were tested in the study and the software which was needed to perform the experiments.

4.1.1 Network Topology

The small testing network consisted of the attacker computer, the IPS device, which was under test, and the target computer, which ran vulnerable software. The network topology is shown in Figure 6. The attacker and the target computers were both connected to the intrusion prevention system, which is called the device under test (DUT). Every DUT operated as an inline layer-2 device. That is the reason why only the management port of the DUT required an IP address. The attacker and the target computers had fixed IP addresses. The target computer was virtualized, in order to be able to restore the initial state of the systems, when starting every test. Oracle VM VirtualBox was used as virtual machine software, and promiscuous mode was enabled.

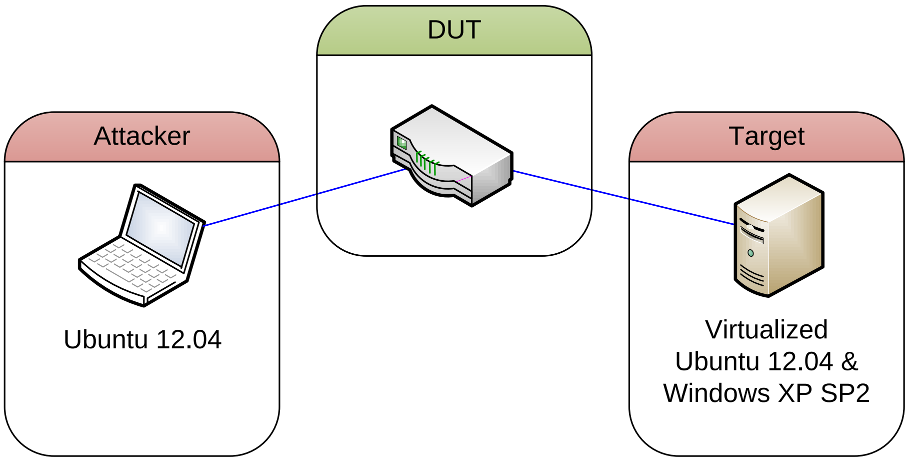

Figure 6: The used network topology

4.1.2 Hardware

In total, 11 intrusion prevention systems were tested, 10 commercial and 1 free and open source solution called Snort. The selected IPS devices contain, for example, all the leaders and challengers of the network intrusion prevention system market according to Gartner [88]. All the tested devices and their software versions are listed in Table 1.

Table 1: Tested IPS devices and their software versions

IPS	Software Version
Check Point UTM-1 270	634131022
Cisco IPS-4240	7.1.(5)E4
Fortigate FG800	V4.0, build0646
HP Tipping Point 110	2.5.5.6994
IBM Proventia GX4004	33.050
Juniper SSG 320M & IDP 75	IDP 5.1.139197
McAfee M-1450	7.5.3.11
Palo Alto PA-500	5.0.0
Snort	2.9.5.3, Snapshot-2950
SourceFire 3D Sensor 2100	4.10.2.7-708, VDB 169
Stonesoft StoneGate 1302	5.4.5, Update Package 527

4.1.3 Software

This study utilized Stonesoft Evader and Mongbat. Both of the tools are covered in the following paragraphs.

Stonesoft Evader

Stonesoft Evader[2] [7, 11] was used throughout this study to test the vulnerability of intrusion prevention systems against different evasion techniques and combinations. It is a vulnerability scanner, which was released by a security company called Stonesoft at Black Hat security conference in June 2012. Even though it contains exploits, it is not a penetration test tool. Its main purpose is to test if well-known exploits can be disguised, with the help of evasion techniques, in a way that an intrusion prevention system is not able to detect them. Evader has its own TCP/IP stack that makes it possible to generate network traffic which can possibly enter the target host undetected. Evader itself is a command line tool, but there is a graphical web interface available too.

Currently, the latest version of Evader contains three different exploits (using the vulnerabilities of CVE-2004-1315, CVE-2008-4250 and CVE-2012-0002) and 35

[2]Evader can be downloaded freely from Stonesoft's Evader web page.

different atomic evasions. The evasions can be applied to different, and also to the same, OSI layers simultaneously. In Evader there are 15 different evasions available for all the three exploits, using network and transport layers.

The exploit implementation using the vulnerability of CVE-2004-1315, which is called HTTP phpBB Highlight in Evader, has additionally 9 application layer evasions using HTTP protocol. The exploit using the CVE-2008-4250 vulnerability, which is called Conficker, has 11 evasions for MSRPC, NetBIOS and SMB protocols. For exploit using the CVE-2012-0002 vulnerability, there are not any additional atomic evasions available. In this study two different exploits were used, the Conficker exploit against Windows XP and HTTP phpBB Highlight against Ubuntu. These two exploits were used because the preference is to test the ability of the IPS devices to detect attacks using evasions at as many OSI layers as possible.

Evader supports payload obfuscation. In Evader obfuscation means that the used shellcode encoder is not static in every attack. IPS devices can detect used shellcode encoders, so Evader tries to create a new, randomized shellcode encoder for every attack. When obfuscation is disabled, Evader usually uses well-known shellcode encoders from Metasploit, for example, the Conficker exploit uses Metasploit based Fnstenv/mov shellcode encoder. Since Fnstenv/mov is a well-known shellcode encoder, it is possible for IPS devices to detect it from the stream by using signature-based detection. For example, Snort has a shellcode rule for it [14].

Mongbat

Mongbat [7] is a fuzz testing tool, which comes with the Evader package. Mongbat makes it possible to run multiple Evader instances in parallel. It also randomizes the used evasion techniques and the evasion parameters for every attack against the target host so that every attack is at least somewhat different. Attacks can use either one evasion technique at the time or combine multiple evasion techniques. Every attack also has its own parameters that can be adjusted, for instance, two attacks with the same evasion combinations are entirely different because the parameters of the evasions differ in the attacks.

Mongbat was the primary test tool in this study. It was used to run a million attacks, with a different set of evasion combinations and parameters, against each IPS device.

4.2 Methods

The methods used for creating the experiments are discussed in the following sections.

4.2.1 Statistically Significant Randomized Trials

A statistically significant number of attacks was performed with randomized evasions and evasion combinations against each IPS device under test. To achieve magnitude large enough, every experiment in this study contained a million attacks. The used exploit stayed the same during a single experiment, and only the evasions changed.

Obviously, the same set of evasions were run against each IPS device under test to get comparable results.

To get a broad knowledge of the effectiveness of various evasion techniques, two different exploits were used. Both of the exploits utilized different protocols. That is why it was possible to test more evasion techniques because each protocol has its own unique characteristics that protocol-specific evasions try to exploit. Thus, evasions were tested at as many OSI layers as possible within the limits of the testing tool.

35 different atomic evasions were utilized in this study. 15 of them can be used with both of the attacks. Besides those 15 evasions, 9 different evasions were only utilized in the attacks against Ubuntu and 11 in the attacks against Windows XP. The evasion techniques, which were used in this study, are listed in Appendix A with brief descriptions.

Reruns

Running of an attack can fail. In this context, an attack process is considered successful if the attack is terminated by an IPS, or it passes through the detection. The attack is considered failed if it was not able to run so that the IPS can decide whether to pass or block it. This can happen due to various reasons, for instance, when establishing of a TCP connection does not succeed because of a network error, or the target system is under a heavy load and not able to process all the requests. So, if running an attack failed instantly, the attack was tried to rerun later.

Obfuscation

Intrusion prevention systems cannot rely on catching an attack by using signature-based detection to just match the used shellcode encoder. Of course, the signature-based detection of shellcode encoders is a useful way to try to detect attacks, but it should not be the only method used. IPS devices should not need to know what is in the payload since they should be able to detect attacks based on the vulnerability which is being exploited.

From now on, throughout this thesis, obfuscation refers to the process of creating a randomized shellcode encoder for every attack. It is studied if the obfuscation of payloads affects the ability of IPS devices to detect attacks. That is why every experiment was performed with and without obfuscation.

4.2.2 Standardized IPS Configurations

Standardized configurations were needed since the goal is to perform objective experiments and examine the IPS devices in a reliable manner. Reliable results cannot be produced if the devices are configured to handle threats differently. This can only be achieved if the devices are configured to operate as similar to each other as possible. Of course, all the devices will always have their distinctive software and hardware properties and features, but with a proper, standardized configuration, the implications caused by the differences of the devices can be decreased.

There are lots of challenges for setting up a correct and the most accurate configuration and detection policy for each IPS device. For instance, different intrusion prevention systems do not behave similarly. They have their own unique way of interpreting the incoming network traffic and determining whether it contains some threats or not. Even the level of a threat's severity is not always considered the same in every IPS device. Additionally, all the same functionalities and services have not been implemented into every device.

Obviously, none of the device configuration interfaces are identical to each other, so it is not possible to have exactly the same configurations in every device. The lack of a common language between the different devices itself is not a problem, but it is worth noticing that the naming and implementation of certain functionalities may differ much between the devices which also means that it takes time to get familiar with the devices.

A major thing is that the devices usually need some modifications and adjustments to the default configuration and detection policy to match the requirements that are given from the environment the device is being used. Often an IPS device may contain a few default detection policies, which highlight some aspects that may be relevant to the end user. A good example of this behaviour is a policy that emphasizes connectivity over security since the customers may actually want to prioritize the accessibility of applications and other resources. Vendors usually have a detection policy in their products that aims to provide the highest level of security. However, selecting the most secure configuration template and detection policy, which are available by default, does not guarantee that it is actually effective enough in practice.

Due to these restrictions, it is important to standardize all the configurations and detection policies. If the standardisation is not performed properly, the results will not be comparable. That would invalidate the study. It means that the results would be biased since some of the device configurations and detection policies might be better than the others. In that case, those devices would also perform better in the experiments than the other devices even though the other devices might actually be capable to perform at the same level or even better.

Establishing a baseline, for the device configurations and detection policies, was needed to be able to evaluate if the state of the art intrusion prevention systems are vulnerable to different evasion techniques and combinations. The devices were configured to be as strict as possible, but still they were able to pass normal network traffic. Every device was configured to follow the guidelines presented in the next paragraph. In most cases, this was achieved by hardening the default configurations.

An IPS device has to:

- permit clean network traffic because an IPS device that terminates everything is useless.
- block the exploits that are used against it. In this study, two different exploits were in use.

- permit sending clean traffic using the same protocol as the exploit. It is not wanted that devices are configured to block an entire protocol in order to catch the exploit. As previously mentioned, all the configurations and detection policies were as strict as possible, but the configurations had to be usable and realistic, too.
- block the same exploits when different evasions and evasion combinations were used.

Since the devices were configured to perform in the experiments as well as possible and in the same way with each other within the limits of the capabilities of the devices, it is possible to get comparable results. It also makes possible to reliably evaluate the vulnerabilities of the devices to evasion techniques.

Still, of course, there is always a possibility of a misconfiguration or misbehaviour by the device even though the configurations were tried to make as good as possible. To minimize the risk of a configuration error, it was verified before each experiment that the device under test was functioning properly and not in any kind of error state. It was also verified that the device operated in the way that was specified in the configuration guidelines earlier.

4.3 Implementation

The following sections discuss the implementation of the experiments. They list the modifications needed to the software used throughout this study and explain how the environment was tested before running the experiments. The process of running the experiments is also covered.

4.3.1 Software Modifications

A small modification to software was needed in order to be able to perform all the experiments in this study. The output of Mongbat can be modified to show somewhat more information including the used Evader commands. Additionally, Mongbat always uses obfuscation by default, but with a few lines of modification to the source code, a new flag can be added to indicate whether to obfuscate or not. The modified version of software is available online[3].

4.3.2 Operating Systems

The attacker computer ran Ubuntu 12.04 LTS, Evader and the modified version of Mongbat. All the experiments were run against two different target operating systems, Ubuntu 12.04 LTS and Windows XP Service Pack 2.

Running a million attacks takes some time, but the process is considerably faster if the attacks are run simultaneously using multiple instances of Evader tool. In

[3]`http://isrg.blogs.glam.ac.uk/2012/09/05/stonesofts-evader-0-9-8-557-modifications-for-testing-aets`

Mongbat, 40 separate instances, called workers, were used throughout this study to speed up the process.

Setting a high number of workers is not enough because the target operation systems tend to reject some of the new incoming connections. It happens because the operation systems are not configured by default to handle such a high number of new connections in a short period of time. That is why some modifications to the target operating systems needed to be done. The following paragraphs list all the needed modifications to the target operation systems.

Ubuntu 12.04 LTS

The type of the Ubuntu installation was a 32-bit server. It can be downloaded, for example, from the official Ubuntu web page. In this study, a modified version of Ubuntu was used as a target system. It contained Evader, some vulnerable services and scripts to control those services. It is available online as an OVF template[4].

The /etc/sysctl.conf file, which has been dedicated to setting system variables, was modified to ensure connectivity to the target computer. Two changes were needed. The first step was to increase threshold values to avoid neighbour table overflow warnings. Thus, the ARP table of the server does not fill up immediately. The second step was to reduce the keepalive time and interval and the number of TCP keepalive messages to ensure the establishment of TCP connections. It is possible that the establishment of TCP connections to the target computer will constantly fail when running Mongbat with 40 workers. It happens because the server tries to maintain all the connections for a certain period of time. That is why it does not free resources right after each attack attempt, and eventually the server runs out of resources. These modifications were done in the following way:

```
net.ipv4.neigh.default.gc_thresh1 = 4096
net.ipv4.neigh.default.gc_thresh2 = 8192
net.ipv4.neigh.default.gc_thresh3 = 65536

net.ipv4.tcp_keepalive_time = 2
net.ipv4.tcp_keepalive_intvl = 1
net.ipv4.tcp_keepalive_probes = 1
```

Windows XP Service Pack 2

The unpatched version of Windows XP SP2 was used because it is vulnerable to Conficker. Some minor changes [8] were needed in the Windows registry to reduce the number of Event ID 2022 errors, which happen when there are not enough free network connections available for users. This can be achieved by using Registry Editor and going to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters` registry subkey and then adding two different values of data type REG_DWORD:

[4]`http://evader.stonesoft.com/evader/download`

```
Value Name: MaxFreeConnections
Value data: 4096 (decimal)
Value Name: MinFreeConnections
Value data: 256 (decimal)
```

4.3.3 Experiments

Four independent experiments with Mongbat were run against every DUT, two against Windows XP SP2 and two against Ubuntu 12.04 LTS. Both of the experiments against the same operation system were similar to each other except obfuscation was disabled in one test and enabled in the other. The Conficker exploit was used against Windows and the HTTP phpBB Highlight exploit was used against Ubuntu.

Initial Testing

Even though the configuration of the DUT was initially checked to be correct, the status of the DUT was rechecked before running every Mongbat test. This was achieved by performing a few tests. It was verified that the DUT was able to pass clean traffic by sending a non-malicious payload to the target host. If the target host replied, there was connectivity between the attacker and the target. The following command was run with Evader:

```
./evader --attack=conficker --if=eth0 --src_ip=10.1.0.10
--dst_ip=10.1.0.1 --verifydelay=1000 --clean
```

It was also needed to check that the DUT was able to block an actual attack. Basically, the same command was used as previously, but the flag "–clean" was excluded from the command. Since the used exploits are relatively old and well-known every DUT should be able to block them.

Additionally, it was needed to verify that the DUT was able to block an attack that uses obfuscation. Once again Evader was used:

```
./evader --attack=conficker --if=eth0 --src_ip=10.1.0.10
--dst_ip=10.1.0.1 --verifydelay=1000 --obfuscate
```

Of course, these tests were also performed when testing the environment using the HTTP phpBB Highlight exploit. Then naturally the attack flag was set to "http_phpbb_highlight".

4.3.4 Long Test Runs with Mongbat

It was noticed that after running the Conficker exploit successfully against Windows XP SP2 over two thousand times but less than three thousand times, the operation system goes into a state where it cannot receive any network connection. To avoid this from happening, a script was created to restore the virtualized Windows XP to its initial state every minute. When the virtual machine was being restored, the attacker cannot establish a TCP connection to the target and test evasion combinations' ability to fool the DUT. That is why rechecking of those failed attacks was needed after the test was finished. The Mongbat test run against Windows was performed by using the following command:

```
ruby mongbat.rb --mode=random --attack=conficker --iface=eth0
--victim=10.1.0.1 --validator=externals/conficker_validator.rb
--attacker=10.1.0.10 --ips_per_worker=1 --workers=40 --mask=16
--time=216000 --check_victim=false --randseed=1qaz2wsx3edc
--disable_obfuscate
```

Against Ubuntu, quite similar settings were used:

```
ruby mongbat.rb --mode=random --attack=http_phpbb_highlight
--iface=eth0 --victim=10.10.0.1
--validator=externals/http_phpbb_highlight_validator.rb
--attacker=10.10.0.10 --ips_per_worker=1000 --workers=40 --mask=16
--time=216000 --check_victim=false --randseed=1qaz2wsx3edc
--disable_obfuscate
```

Mongbat runs the same set of evasions in attacks if the seed value stays the same. When multiple workers are in use, Mongbat runs multiple Evader instances in parallel. If exactly the same Mongbat command is run twice, the log files produced by the modified version of Mongbat do not contain the attacks in the same order. Mongbat is always run for a certain amount of time. After finishing the runs, the first X attacks are probably not exactly the same because the workers might not execute the attacks in the same time in each run.

To avoid the need for processing the logs to get the results from attacks with the same set of evasions, a script was created that contains a list of exactly a million attacks with evasions and evasion combinations based on the initial Mongbat run. Basically, the list contains the used Evader commands. The script directly executes all of the million attacks using multiple simultaneously run Evader instances.

After running the million attacks, the script matches all the failed attacks from the log file and tries to rerun those attacks. Rerun is performed up to 20 times to make sure that all the attacks are executed properly. A failed attack means, for example, an attack that was not able to establish a TCP connection. These situations are logged by Evader with a message "300: TCP connection failed". The number of failed connections was usually significant in Windows XP because of the virtual machine restoring process. In Ubuntu, restoring was not needed, so the connection failure rate was also lower.

The script is considered hostile. It executes the million attacks and possibly tries to rerun some of them multiple times. That is why its source code is not included in this thesis. To run the same million attacks when obfuscating is enabled, the script only includes the flag "–obfuscate" in Evader commands.

4.4 Summary

The testing environment consisted of the attacker computer, the intrusion prevention system under test and the target computer, which ran vulnerable software. The tool that was used for attacking was Stonesoft Evader. Ten commercial intrusion prevention systems and one open source solution were tested in this study. Both Windows and Linux operating systems were used as target hosts in different experiments.

Each experiment in this study contained a million attacks in order to achieve magnitude large enough. All the attacks were performed with randomized evasions and evasion combinations against each intrusion prevention system. The used exploit stayed the same during a single experiment, and only the evasions changed.

5 Experiments and Analysis

Four experiments were conducted to test the vulnerability of intrusion prevention systems against evasion techniques. In each of the experiments, a million attacks were run against target operations systems. Two of the experiments were run against Windows and two against Ubuntu. The tested intrusion prevention systems were placed between the attacker and the target operation systems to block the attacks. Every attack was masked with different evasion techniques or evasion combinations. This chapter lists all the results from these experiments. Additionally, a thorough analysis is done about these results. The reliability of the results is also discussed.

There was limited time available to test the commercial IPS devices. Since Aalto University does not own the tested IPS devices, the experiments were dependent on the schedule of the third party, which provided the IPS devices. Due to these time limitations, few experiments were excluded from the final results.

With the McAfee IPS device, only 1 out of 4 experiments was successful. During the other experiments, which turned out to be unsuccessful, the device went to an error state after performing as expected for a certain period of time. During this erroneous state, the device terminated all the incoming connections. It happened when establishing TCP connections, so the terminations were not caused by detecting the actual attack. Due to the incomplete results, it was needed to exclude those experiments from this thesis.

The devices were tested during the summer of 2013. Because the process of having the commercial IPS devices tested was dependent on the schedule of the third party, it was not possible to test all the devices at once. When an IPS device was acquired for testing, software was always updated to the latest version available at that moment. However, there was an open source system, Snort, included in the experiments. The hardware used for Snort was provided by Aalto University. Since the hardware was available through the whole testing period, it was decided that it was tested last so that it was possible to use the latest software. There was no reason to use older software, since the other devices were not tested at the same time but during a few weeks.

It is argued that none of the tested IPS devices gained unfair benefit for the experiments because of the time they were tested. All the used exploits were old, from the year 2004 and 2008, and the used evasion techniques were not particularly new, since some of them were discovered in 1998 and the majority in the 2000s. So, the devices should have been able to block the attacks even though some of the devices were tested a few weeks earlier than the others.

5.1 Evasion Detection Rates

As explained in Section 4.2, a statistically significant number of attacks, namely a million, was performed with randomized evasions and evasion combinations against each IPS device under test. The used exploit stayed the same during a single experiment. These principles applied to all the experiments that were run against the target operation systems, Windows XP SP2 and Ubuntu 12.04.

The experiments were run with and without obfuscation. The set of evasions stayed the same in the experiments that were run against the same operation system. This means that the experiments targeting, for example, Windows XP utilized exactly the same evasions and evasion combinations with identical parameters when obfuscation was disabled or enabled. Thus, the same set of evasions were run against each IPS device under test.

The second and the third column of Table 2 list the results of Experiment A. In this experiment, a million attacks with various evasions and evasion combinations were run against each IPS device under test. The target host behind the intrusion prevention systems was running Windows XP SP2. The attacks exploited the vulnerability known as CVE-2008-4250. The obfuscation option provided by the Evader tool was disabled in this experiment.

Table 2: Evasive attacks with and without obfuscation against Windows

	Experiment A Obfuscation disabled		Experiment B Obfuscation enabled	
IPS	Successful evasions	Detection rate	Successful evasions	Detection rate
Check Point	40748	95.92%	40917	95.91%
Cisco	4397	99.56%	4499	99.55%
Fortigate	20705	97.93%	20949	97.91%
IBM	10970	98.90%	11665	98.83%
Juniper	84579	91.54%	86093	91.39%
Palo Alto	31155	96.88%	31373	96.86%
Snort	19485	98.05%	36983	96.30%
SourceFire	11726	98.83%	17543	98.25%
Stonesoft	0	100.00%	0	100.00%
Tipping Point	2891	99.71%	34582	96.54%

As shown in the results of Experiment A, almost all the tested IPS devices (9 out of 10) passed through some attacks. Detection rates between the devices varied from 91.54% to 100%. Six of the devices were capable of having over 98% detection rate.

The fourth and the fifth column of Table 2 list the results of Experiment B, which was similar to the previously presented Experiment A except the obfuscation option was enabled in the Evader tool. Again, almost all the tested IPS devices (9 out of 10) passed through some attacks as seen from the results of Experiment B. The overall detection rate stayed near the same. Four devices were capable of having over 98% detection rate in Experiment B.

Theoretically, the use of obfuscation makes it more difficult for intrusion prevention systems to detect the attacks because IPS devices cannot rely on detecting the used shellcode encoder. When comparing the results of Experiment B with Experiment A, where obfuscation was disabled, it can be noticed that the enabling of obfuscation had a very small effect on the majority of the tested devices. There was an exception though. When comparing the results of Tipping Point with and without obfuscation, it can be seen that obfuscation affected the results quite significantly. The number of successful attacks with obfuscation was ten times larger than without obfuscation. It means that the detection rate dropped from 99.71% to 96.54% with that particular device when obfuscation was enabled.

The second and the third column of Table 3 list the results of Experiment C. Once again a million attacks with various evasions and evasion combinations were run against each IPS device. The target host behind the intrusion prevention systems ran this time Ubuntu 12.04. The attacks exploited the vulnerability known as CVE-2004-1315. Obfuscation was disabled in this experiment.

Table 3: Evasive attacks with and without obfuscation against Ubuntu

	Experiment C Obfuscation disabled		Experiment D Obfuscation enabled	
IPS	Successful evasions	Detection rate	Successful evasions	Detection rate
Check Point	34350	96.56%	34688	96.53%
Cisco	63546	93.65%	189550	81.05%
Fortigate	1180	99.88%	1969	99.80%
IBM	37252	96.28%	257138	74.29%
Juniper	64648	93.54%	94595	90.54%
McAfee	5202	99.48%	-	-
Palo Alto	31464	96.85%	456109	55.39%
Snort	10623	98.94%	11985	98.8%
SourceFire	38557	96.14%	65835	93.42%
Stonesoft	0	100.00%	0	100.00%
Tipping Point	73650	92.64%	75380	92.46%

Almost all the tested IPS devices (10 out of 11) passed through some attacks. Detection rates between the devices varied from 92.64% to 100%. In Experiment C, four of the devices were capable of detecting over 98% of the attacks.

When comparing the results of Experiment C with Experiment A, it can be seen that in overall the detection rates were somewhat lower in half the cases. There are many different aspects that need to be noticed when comparing experiments A and C. They used entirely different exploits, and the used evasion techniques were not entirely the same.

The fourth and the fifth column of Table 3 list the results of Experiment D, which was similar to Experiment C except obfuscation was enabled. As seen from the results of Experiment D, almost all the tested IPS devices (9 out of 10) passed through some attacks. The variation of the results was relatively large. Detection rates between the devices varied from 55.39% to 100%. It was considerably larger variation when comparing with Experiment C. Some of the detection rates dropped significantly. Three of the devices were capable of having over 98% detection rate.

When comparing the results of experiments C and D, it is shown that the enabling of obfuscation affected many of the tested devices. For example, when comparing the results of Palo Alto with and without obfuscation, it can be seen that obfuscation had a huge impact on the results. The detection rate dropped from 99.81% to 55.39% when obfuscation was enabled.

Obfuscation

All the further analysis is conducted from the experiments that include obfuscation. As seen in Tables 2 and 3, obfuscation increases the possibility that an attack is capable of evading the detection of intrusion prevention systems. With obfuscation, the detection rates are equal or lower than the detection rates of the experiments without obfuscation. The result was expected because obfuscation can be considered as an additional mean to try to fool intrusion prevention systems.

By using a new, customized shellcode encoder in every attack, which means enabling obfuscation, intrusion prevention systems are not able to solely rely on detecting the shellcode encoders. As mentioned earlier, the test tool uses well-known shellcode encoders from Metasploit when obfuscation is disabled. That makes it possible that an IPS device just recognizes the shellcode encoder but does not actually detect the attack, which is masked with evasions.

Since this thesis studies how well different intrusion prevention systems can detect attacks, which are masqueraded with evasions, the analysis concentrates on experiments including obfuscation. The experiments indicate that obfuscation increases the possibility that an attack can go through an IPS device undetected.

5.2 Atomic Evasions

This study concentrates on evasion combinations, but it also considers the possibility that only one evasion technology can be enough to evade intrusion prevention systems. A single evasion technique is called an atomic evasion. There were 35 atomic evasions used in this study. 26 atomic evasions out of 35 can be utilized when attacking Windows. In the same way, 24 evasions were eligible against Linux. Evasions that utilize IP and TCP protocols are naturally eligible against both of the target systems. Thus, this study included 15 atomic evasions, which were utilized in each of the experiments.

The used tool and the exploits provided with it restrict the use of atomic evasions in a certain way. In this study, evasions utilizing HTTP were run against Linux. However, these evasions are not restricted to Linux but to the protocol. If an exploit against Windows used HTTP, the HTTP evasions would be eligible against Windows too.

Due to the nature of this thesis, the exact successful atomic evasions against each device under test (DUT) are not disclosed. The same policy is used throughout this study, so successful evasion combinations are not disclosed either. The reason is that this study only states the effectiveness of various evasion techniques, but its intention is not to become a reference guide in attacking some particular intrusion prevention system.

Every evasion has a dedicated set of parameters available that can only be applied to that particular evasion. Each of the parameters may have numerous values that can be assigned to the parameter. The evasion technique utilizing TCP segmentation, for example, has only one possible parameter, which is the segment size, but it allows integer values from 1 to 65535. It should be also noticed that other evasion techniques may be associated with more than one parameter.

It is possible to try to fine-tune atomic evasions to be successful. It is done by using different parameter combinations and by assigning different values to those parameters. Due to the high number of different parameters and parameter values, there are numerous ways of fine-tuning the atomic evasions.

Table 4 lists the number of successful atomic evasions against each intrusion prevention system. It contains evasion techniques that were used against Windows and Ubuntu operation systems. Table 4 clearly shows that just one evasion technique can still be enough to masquerade an attack. So, even though the atomic evasions, which were used in this study, are well-known and some of them from the 1990s [61], they were able to fool some of the tested intrusion prevention systems.

Table 4: The number of successful atomic evasions, which can be fine-tuned to be effective, when attacking against Windows and Ubuntu

IPS	Windows (out of 26)	Ubuntu (out of 24)
Check Point	4	1
Cisco	24	23
Fortigate	24	10
IBM	20	23
Juniper	22	23
Palo Alto	22	23
Snort	24	23
SourceFire	21	23
Stonesoft	0	0
Tipping Point	23	9

The variation of the results was large. There were devices that handled quite well various atomic evasions. On the other hand, there were many IPS devices struggling with atomic evasions. Some of the available atomic evasion techniques were not effective, so they were not capable of evading intrusion prevention systems if other evasion techniques were not included in the attack.

It can be seen from the results that only one evasion technique was enough in many cases. This applied to both of the target systems. It must also be taken into account that a single atomic evasion was considered successful if an attack was carried to the target system successfully at least once. Even though each of the atomic evasions can be assigned a huge number of different parameter values, it is possible that a successful combination of parameter values was found in this study only once.

The percentage of each of the successful atomic evasions against both of the target machines was generally quite low. There were cases with less than ten successful attacks and, like stated in the previous paragraph, sometimes it was found only one set of successful parameter values. However, larger percentages also existed.

Usually there was almost the same number of successful atomic evasions against Windows and Ubuntu when comparing the results between the DUT. However, Tipping Point seemed to be one of the exceptions. There were lots of successful evasions, namely 23, against Windows, but the number of successful evasions against Ubuntu was only 9. Fortigate had also quite similar results. These results can be explained easily. Both of the devices handled atomic evasions against HTTP protocol quite well, but the common evasion techniques, utilizing IP and TCP protocols, against Windows and Ubuntu target machines caused problems. Additionally, the atomic evasions with SMB, MSRPC, NetBIOS protocols sometimes caused problems, which explain the relatively high number of successful atomic evasions against Windows target machine.

Old atomic evasion techniques were still effective against the state of the art instruction prevention systems. Like previously stated, the first academic papers about evasions were published in the 1990s [61]. Over a decade after the discovery of these evasion techniques, they were able to fool the majority of the tested intrusion prevention systems.

These results regarding atomic evasions verify that a single atomic evasion can be enough to avoid detection. Dyrmose also showed in his study in 2013 that certain intrusion prevention systems are vulnerable when these techniques are used even though the scope of the study was significantly smaller. It only used a few evasion techniques with specific parameter values. [36]

The ineffectiveness of evasion techniques against each DUT was also studied. In Table 4, the number of successful atomic evasions is listed, so it can easily be determined that the other remaining evasions were not enough for masquerading attacks. If, for example, 5 atomic evasions are successful, the rest are ineffective. However, some of the remaining evasion techniques can be successful if they are combined with other evasions. That is why an atomic evasion was determined successful if it was capable of evading the detection when combining it with some other evasion techniques.

To be able to have a list that contains only evasion techniques that are not successful against a DUT, it is needed to further analyse evasion combinations. First, it is checked that there is not a successful attack available when using an atomic evasion or a combination utilizing that particular evasion. Then, the atomic evasion is ineffective.

It is also possible that an atomic evasion is used in an evasion combination, but it is not actually needed for evading. In that case, first it is verified that the atomic evasion is ineffective, when only that particular evasion is used in attacks. Then it is checked if the atomic evasion can be removed from the evasion combination. In other words, if there is an attack, which is masked with evasions A, B and C, it might be possible that only evasions A and B are necessary for evading and evasion C does not have any effect.

Table 5 lists the number of ineffective evasion techniques. As previously stated, the results take into account that some of the evasion combinations do not actually need to utilize all of their atomic evasions to be effective.

Table 5: The number of ineffective atomic evasions when attacking against Windows and Ubuntu

IPS	Windows (out of 26)	Ubuntu (out of 24)
Check Point	22	21
Cisco	2	1
Fortigate	2	13
IBM	6	1
Juniper	4	1
Palo Alto	4	1
Snort	2	1
SourceFire	5	1
Stonesoft	26	24
Tipping Point	3	15

There were evasion techniques available that seem to be ineffective against the majority of the tested IPS devices. However, the distribution of the used evasions was not the same. It means that some of the evasion techniques were used more often when comparing with the others. Generally, each of the evasion techniques was used thousands of times, but there was an exception. There was an evasion technique that was used in attacks against Windows machines, and it was utilized only about ten times in the attacks. This happened because Mongbat was used to create and launch a million attacks against the targeted machines. With the selected seed value, the tool itself did not frequently use that particular evasion. That was noticed after the testing of IPS devices had begun, so the test set of evasions stayed the same because it was not possible to run more attacks against each DUT at that point. Since that particular evasion was only used in few attacks, it is not possible to draw a conclusion of its effectiveness.

There were ineffective evasion techniques that were used in the attacks against Windows and Linux target machines. They seem to be ineffective against all the tested intrusion prevention systems. Additionally, there were evasion techniques that did not work against the majority of the devices but were still able to evade some of the IPS devices.

5.3 Evasion Combinations

As previously stated, it is possible to masquerade an attack by only using one evasion technique so that it is capable of reaching its target. However, one evasion technique is not always enough. Then the solution might be to combine multiple evasion techniques. Even a single atomic evasion technique can be used multiple times in the attack with different parameter values.

In this study, the average number of evasions used in successful attacks was below 2. The result shows that a relatively small number of different evasion techniques was needed to combine in order to avoid detection by intrusion prevention systems.

To be able to further analyse the results given in Section 5.1, the number of different evasion combinations is calculated. It means, for example, that if an attack is masked with evasion techniques X, Y and Z and it is not blocked by the DUT, these three evasion techniques form a successful evasion combination. Only the used evasion techniques are considered a significant factor, so all the possible evasion parameters are ignored in this study. Like stated previously, each evasion technique can have multiple parameters. These evasion-specific parameters can be assigned different values, and the set of possible values can be large.

If an attack is, for example, masked with an evasion technique X and the previously mentioned evasion technique, TCP segmentation, with a parameter value A and able to reach its destination, the two used evasion techniques are considered a successful evasion combination. That is why the number of evasion combinations is increased by one. If the attack is also successful with the same set of evasions but with different parameters, it is not considered a new, successful evasion combination any more. Thus, it does not increase the calculated number of evasion combinations.

Table 6 lists how many different evasion combinations were found that were capable of evading the detection. The number of successful evasion combinations is listed against Windows and Ubuntu. When the parameters and their values are ignored, there were 8278 possible combinations in this study that can be utilized against Windows and 11383 against Ubuntu. The results clearly indicate that some of these evasion combinations were effective.

Table 6: The number of evasion combinations, which can be fine-tuned to be effective, when attacking against Windows and Ubuntu

IPS	Windows	Ubuntu
Check Point	1496	1133
Cisco	519	4133
Fortigate	1445	412
IBM	913	5202
Juniper	2915	3639
Palo Alto	1338	5781
Snort	1950	1263
SourceFire	940	2804
Stonesoft	0	0
Tipping Point	1742	2934
Maximum	8278	11383

As the results indicated in Tables 2 and 3, there were more successful evasion combinations against Ubuntu than Windows. This is the expected result, as shown in Table 6, because the detection rates were worse in the attacks against Linux.

There were three IPS devices that had less successful evasion combinations against Linux than Windows. These results are also consistent with the results presented in Tables 2 and 3. The other intrusion preventions systems performed better in the attacks against Windows excluding the Stonesoft IPS, which performed equally well in both attacks.

Combining multiple evasion techniques increased the possibility to find a way to evade detection. As seen in Table 4, Checkpoint had only one atomic evasion that was capable of evading the detection when attacking Ubuntu. However, Table 6 lists that there are hundreds of successful evasion combinations that can be used in attacks. Even if the atomic evasion that was able to evade the detection itself, with certain parameter values, was excluded from the results, there were still successful evasion combinations available.

Additionally, it is worth noticing that even if a single evasion technique is successful with certain parameter values, it might be easier to find multiple successful evasion combinations than just use one single evasion technique. As previously stated, some of the atomic evasion techniques are able to fool intrusion prevention systems without the need for adding additional evasion techniques. Nevertheless, the finding of suitable values for the evasion parameters can be tricky. There were examples in this study that a single evasion technique was able to fool IPS devices only once. It means that all the other used values for the parameters were not effective.

The results of Experiments B and D are summarized in Tables 7 and 8. The tables list the number of successful attacks against Windows and Ubuntu. The columns list how often that particular evasion was used successfully in all of the attacks. However, the actual evasion technique is not disclosed. The last column lists the number of successful attacks against that particular DUT. Since each attack can contain a various number of different evasion techniques, the sum of the values in evasion columns is not equal to the total number of successful attacks. Some of the values are bold. That particular evasion technique had at least one set of parameter values available that was able to evade the detection without the help of other evasion techniques. So, if a single successful atomic evasion was detected, the value is bold.

An evasion technique was counted only once per an attack even if it was used twice or more in the attack. This means that the number of successful evasion usage increased only the same amount in each attack. In other words, using an evasion technique multiple times with different parameter values in an attack was counted as a single usage.

The tables disclose the volume of the successful attacks utilizing certain evasion techniques. Even though the distribution of the evasion techniques was varying, it still gives an indication how effective a certain evasion technique is when the result is compared with the other intrusion prevention systems. In the same manner the effectiveness of single atomic evasions can be compared between different intrusion

Table 7: The number of successful attacks against Windows when each column represents an evasion technique. Multiple evasion techniques were used in the attacks, so the total number of successful attacks is lower than the sum of the values in the evasion columns.

IPS	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	Total
Check Point	3102	3081	154	1	**7908**	65	1144	2	1330	**16963**	**11946**	235	1939	3570	138	140	140	137	359	1155	1538	126	**6776**	78	2756	1026	40917
Cisco	**351**	**448**	**18**	0	**71**	**17**	**425**	**20**	**297**	**225**	**141**	**31**	**318**	**501**	**23**	**18**	**23**	**19**	24	**194**	**454**	**2220**	**271**	**6**	**439**	**398**	4499
Fortigate	**2693**	**2526**	**126**	0	**357**	**197**	**1986**	**152**	**1713**	**1752**	**640**	**542**	**1725**	**2560**	**134**	**100**	**91**	**96**	38	**1088**	**2726**	**82**	**576**	**505**	**2036**	**8473**	20949
IBM	**675**	**852**	26	0	**120**	**27**	**9075**	**879**	**562**	**733**	**237**	**103**	**871**	**1078**	**39**	36	**34**	**33**	36	**431**	2217	**40**	**357**	14	**856**	**840**	11665
Juniper	**6787**	**6001**	**3829**	0	**869**	100	**31606**	**2948**	**4010**	**4426**	**1620**	**4540**	**3971**	**7383**	**260**	**267**	**279**	**302**	483	**2646**	**6906**	**270**	**8950**	146	**5655**	**40085**	86093
Palo Alto	**1527**	**3493**	93	0	**333**	25	**1759**	**35**	**1350**	**5111**	**589**	**122**	**1488**	**2847**	**99**	**66**	**109**	**120**	729	**848**	**15859**	**96**	**11276**	**63**	**2128**	**874**	31373
Snort	**3822**	**3658**	**149**	0	**484**	**115**	**4055**	**216**	**2457**	**5049**	**955**	**337**	**2694**	**4345**	**163**	**154**	**158**	**165**	207	**1868**	**4565**	**138**	**3207**	**147**	**3225**	**19944**	36983
SourceFire	**750**	**1307**	**39**	0	**186**	37	**2002**	**112**	**871**	**1305**	**318**	**142**	**887**	**1636**	**70**	**61**	63	**68**	129	**563**	**1565**	**37**	**1979**	29	**1241**	**16833**	17543
Stonesoft	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tipping Point	**2172**	**2858**	24	0	**361**	**115**	**2595**	**136**	**1935**	**8396**	**742**	**1889**	**17695**	**4629**	**1826**	**114**	**157**	**123**	273	**1210**	**1642**	**59**	**1836**	**76**	**2651**	**2962**	34582

Table 8: The number of successful attacks against Ubuntu when each column represents an evasion technique. Multiple evasion techniques were used in the attacks, so the total number of successful attacks is lower than the sum of the values in the evasion columns.

IPS	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Total
Check Point	136	1119	**34548**	1059	1047	1038	686	829	1177	3118	2989	3171	1086	1050	1020	1097	41	24	1931	478	1225	589	1029	659	34688
Cisco	**11360**	**10434**	**10356**	**9359**	**10084**	**10005**	**6231**	**5734**	**10442**	**17146**	**28019**	**25426**	**10579**	**10129**	**10511**	**10434**	574	**995**	**28275**	**5581**	**15342**	**5928**	**10464**	**8086**	189550
Fortigate	75	54	59	**351**	**589**	146	96	76	63	**195**	**178**	**174**	51	49	67	**77**	3	2	**592**	29	136	**1078**	**69**	**76**	1969
IBM	**6919**	**13513**	**13428**	**15592**	**17855**	**13336**	**8622**	**8965**	**13686**	**32775**	**37666**	**39569**	**13544**	**13058**	**13785**	**13547**	357	**2056**	**42115**	**7377**	**12355**	**7528**	**13717**	**13600**	257138
Juniper	**4040**	**4085**	**20184**	**24821**	**3799**	**3983**	**2303**	**1427**	**3210**	**11468**	**10338**	**11587**	**4040**	**3997**	**4003**	**4045**	195	**164**	**8965**	**1786**	**6550**	**2279**	**4037**	**10175**	94595
Palo Alto	**25309**	**25431**	**25433**	**21813**	**10435**	**24791**	**14215**	**6815**	**23117**	**61751**	**65815**	**74274**	**25442**	**24546**	**25477**	**25292**	876	**364**	**69487**	**11189**	**27077**	**14218**	**25356**	**18176**	456109
Snort	**413**	**402**	**422**	**7057**	**1431**	**397**	**235**	**206**	**577**	**3153**	**3206**	**1178**	**402**	**382**	**432**	**410**	20	**88**	**1044**	**171**	**591**	**237**	**409**	**498**	11985
SourceFire	**1282**	**2117**	**26472**	**14739**	**841**	**2530**	**1338**	**1498**	**20477**	**3839**	**5428**	**6059**	**2095**	**2045**	**2110**	**2115**	90	**80**	**3773**	**743**	**3333**	**1401**	**2056**	**6598**	65835
Stonesoft	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tipping Point	2285	2197	2025	**35053**	**1878**	2303	**1404**	1686	2465	**6192**	6080	**9080**	**24071**	2324	2391	2362	541	20	**4397**	920	**4855**	1315	2341	**12924**	75380

prevention systems by examining the bold values. However, the bold values only reveal that those evasion techniques can be fine-tuned to be successful without combining them with other evasions. It does not list how many successful attacks were found, utilizing only one evasion technique.

When studying closer the results of Table 7, it is seen that some of the evasion techniques were more effective against a group of tested intrusion prevention systems. SourceFire was fourth in the results, but it had problems with the evasion technique Z, which clearly weakened its results. Over 95% of the successful attacks utilized that particular evasion technique. Also Juniper and Snort had problems detecting the attacks that utilized the evasion technique Z. They allowed a higher number of attacks using that particular evasion technique than SourceFire. However, the percentage of successful attacks utilizing the evasion technique Z was not as high as it was with SourceFire where almost all the successful attacks utilized it.

The results show that there were various evasion techniques that were significantly more effective against one particular intrusion prevention system. For example, the evasion technique C was over 25 times more effective against Juniper than the other tested intrusion prevention systems. However, these results can be explained with the fact that Juniper passed the highest number of attacks, so it is possible that the evasion technique C was successfully combined with some other evasion techniques. The number of successful attacks utilizing the evasion techniques E, J and K were significantly higher against Check Point. Even though these results show that Check Point had problems with certain evasion techniques, the results with the other evasion techniques do not stand out. In fact, Check Point was one of the few tested intrusion prevention systems that were capable of stopping the majority of the attacks that were fine-tuned with atomic evasion techniques. As Juniper passed the highest number of attacks, it is not surprising to see that several evasion techniques, including G, H, I, L and T, were the most successful techniques in the attacks against Juniper.

There are many examples in the results showing that at least about half of the successful attacks utilized the same evasion technique. For example, this happened with SourceFire and Juniper when utilizing the previously mentioned evasion technique Z. Similar behaviour was found with Tipping Point and Palo Alto when they utilized the evasion techniques M and U, respectively.

As previously mentioned, Table 8 shows the results of the attacks against Ubuntu. Palo Alto passed the highest number of attacks, so the evasion columns of Palo Alto stand out. When comparing the results of a single evasion technique between the tested intrusion prevention systems, Palo Alto has the highest number of attacks 19 times out of 24. However, various evasion techniques also caused problems with the other intrusion prevention systems. For example, the evasion technique C was effective against Checkpoint since almost all the successful attacks utilized that particular evasion technique. Checkpoint also had the highest number of successful attacks utilizing that evasion technique among the tested intrusion prevention systems. Fortigate performed second best in overall, but the evasion technique V caused most of the problems since almost 55% of the successful attacks utilized that particular evasion technique. Snort was third in the results, but it also had difficulties with one evasion technique, namely D.

Tables 9 and 10 represent how well different evasions function as evasion combinations. The results count only the evasions that were combined with at least one other evasion technique. That is why the single atomic evasions were excluded. There are separate tables available for the results of the attacks against Windows and Ubuntu.

The columns list the success rates of evasions in percentages. Each success rate was calculated in proportion to the total number of the attempts to utilize that particular evasion in evasion combinations. So, unlike previously, the results in columns do not represent the successful evasion usage in all of the attacks including the whole set of a million attacks. The relative results can be compared with each other since the results take into consideration how often each evasion was tried to utilize in the attacks.

These tables give a strong indication of the evasion techniques that are the most probable techniques that can be successfully utilized to evade any of the tested intrusion prevention systems. There were clearly evasion techniques that were more successful than the others. Utilizing those evasion techniques by combining them with each other produces the most dangerous attacks. The results can also be interpreted in the other way around to see the most ineffective evasion techniques.

The results of Table 9 give an indication of the evasion techniques that can be successfully combined with other evasions. For example, the evasion technique H was successfully combined with some other evasions with the success rate over 49% against Juniper. As seen in Table 7, the highest number of successful attacks against Check Point utilized the evasion techniques E, J, K and W. When these techniques were combined with some other evasions, their effectiveness varied. For example, the evasion technique E was combined with other evasions with the success rate over 42%. However, the evasion technique W had only the success rate of 10%, which is a significantly weaker result. Thus, it can be seen that the evasion technique W is not the best choice when selecting the evasion combinations for attacks against Check Point.

The results of certain evasion techniques between different intrusion prevention systems may vary quite significantly. Some evasion techniques can cause more problems to a certain intrusion prevention system than to the others even if that IPS device actually performed better in overall. Thus, the weaknesses of different intrusion prevention systems were revealed against certain techniques. The results of Table 10 show that the evasion technique D was combined with other evasions with the success rate over 69% against Tipping Point and over 50% against Juniper. These results clearly show that this evasion technique can be successfully combined with other evasions when attacking many of the tested intrusion prevention systems. Additionally, the results of the evasion technique C stand out. As seen in Table 8, almost all the successful attacks against Check Point utilized the evasion technique C. The results of Table 10 show that even though that particular evasion technique was effective in the attacks against Check Point, it was not effective each time it was utilized since its success rate was over 61%.

Evasions are a real threat. The results show that evasion techniques can be

Table 9: The success rates of evasions against Windows when they are utilized as combinations, and each column represents an evasion technique

IPS	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Check Point	3.85	3.64	4.72	11.11	42.32	3.00	1.76	0.06	2.82	17.80	31.76	3.67	4.11	4.22	4.31	4.59	4.28	4.11	8.19	3.59	1.82	3.76	10.06	4.16	4.26	1.58
Cisco	0.28	0.38	0.34	0.00	0.57	0.55	0.47	0.46	0.44	0.39	0.53	0.36	0.46	0.40	0.47	0.43	0.40	0.45	0.55	0.43	0.32	33.68	0.57	0.27	0.46	0.36
Fortigate	2.20	2.01	3.03	0.00	2.68	8.39	2.16	3.23	2.56	2.79	2.19	7.58	2.57	2.01	2.94	2.10	1.59	1.83	0.87	2.27	2.06	1.82	1.23	14.10	2.02	7.18
IBM	0.80	0.97	0.80	0.00	1.25	1.20	7.91	14.99	1.15	1.51	1.18	1.56	1.75	1.24	1.09	1.18	0.98	0.93	0.82	1.29	1.90	1.19	0.94	0.75	1.28	1.25
Juniper	8.40	7.07	66.26	0.00	9.11	4.61	28.11	49.52	8.46	9.19	8.37	40.22	8.40	8.64	8.06	8.70	8.51	9.02	11.02	8.19	8.15	8.03	15.16	7.79	8.71	35.24
Palo Alto	1.87	3.40	2.85	0.00	3.48	1.15	2.69	1.02	2.79	6.42	3.01	1.87	3.13	3.34	3.06	2.10	3.30	3.54	16.63	2.59	10.44	2.63	15.31	3.26	3.26	1.31
Snort	3.45	3.18	3.62	0.00	3.84	4.24	4.96	5.51	3.88	7.06	3.80	4.24	4.21	3.80	4.03	3.84	3.92	3.57	4.72	4.49	3.95	3.05	7.77	4.86	3.70	19.20
SourceFire	0.89	1.49	1.16	0.00	1.87	1.71	3.02	3.32	1.79	2.63	1.58	2.18	1.82	1.87	2.12	1.90	1.93	1.95	2.94	1.68	1.79	1.07	5.47	1.55	1.87	15.93
Stonesoft	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Tipping Point	2.41	2.89	0.74	0.00	3.38	4.84	3.50	3.57	3.59	9.80	3.28	16.17	20.18	4.08	29.37	3.22	4.44	3.30	6.23	3.18	1.68	1.61	4.54	3.36	3.56	4.08

Table 10: The success rates of evasions against Ubuntu when they are utilized as combinations, and each column represents an evasion technique

IPS	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Check Point	0.44	3.66	61.60	3.87	3.43	3.39	3.67	3.59	3.81	3.67	3.43	3.67	3.57	3.70	3.31	3.59	2.58	0.16	2.42	1.57	4.11	3.43	3.38	2.15
Cisco	18.31	18.77	18.66	18.60	17.43	17.63	18.45	13.64	18.48	10.92	17.24	15.39	18.96	18.86	18.95	19.01	36.10	4.00	18.53	10.11	27.19	18.51	19.12	14.17
Fortigate	0.24	0.18	0.19	1.28	1.91	0.48	0.51	0.33	0.20	0.23	0.20	0.20	0.17	0.17	0.22	0.25	0.19	0.01	0.74	0.10	0.46	3.50	0.22	0.24
IBM	13.88	25.45	25.63	32.06	32.23	24.69	26.36	22.11	25.53	21.26	23.80	25.69	25.48	25.33	25.91	25.67	22.45	8.36	28.95	14.04	23.23	25.59	25.87	24.97
Juniper	9.40	9.79	37.63	50.40	8.68	9.05	8.90	4.85	7.89	9.38	8.44	9.56	9.54	9.99	9.38	9.58	12.26	0.69	7.88	4.31	13.44	9.67	9.46	19.30
Palo Alto	44.39	45.06	45.86	43.40	18.91	43.38	41.84	16.58	41.16	37.97	39.49	45.26	45.35	45.15	45.42	44.97	55.09	1.92	45.36	20.24	48.30	45.07	45.17	31.77
Snort	1.26	1.20	1.30	14.69	3.20	1.23	1.16	0.77	1.78	3.60	3.55	1.26	1.18	1.22	1.32	1.25	1.26	0.41	1.20	0.51	1.88	1.26	1.26	1.53
SourceFire	4.09	6.81	46.36	30.13	2.69	8.14	7.03	6.39	36.15	4.44	6.05	6.88	6.78	7.08	6.70	6.81	5.66	0.49	4,63	2.37	10.61	7.94	6.63	13.79
Stonesoft	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Tipping Point	7.39	7.19	6.68	69.68	4.62	7.52	7.51	7.30	7.99	7.28	6.97	9.00	43.85	8.19	7.75	7.73	34.03	0.13	5.50	3.02	12.13	7.66	7.69	24.71

used successfully to evade the detection of intrusion prevention systems. Even a single evasion technique can be enough to fool an intrusion prevention systems if it is equipped with suitable parameter values.

There are numerous of evasion combinations available, and it seems that intrusion prevention systems are beatable. Since there are lots of different evasions available, it is possible to find a successful evasion combination for an attack. The risk of a successful attack increases if an attacker has the possibility to test an intrusion prevention system and find a successful evasion combination before launching an actual attack.

Many of the tested intrusion prevention systems seem to have problems understanding the traffic that pass through them correctly. That is why it is difficult to detect all the attacks that are masked with evasions. That leads to a situation where various evasion techniques are effective against intrusion prevention systems.

As described in Section 3.4, there are countermeasures against evasions, such as traffic normalization and reassembly. The results of this study indicate that normalization and reassembly have not been properly implemented in some of the tested intrusion prevention systems. This can be stated because the number of successful attacks utilizing evasion techniques was so high. Reassembling TCP/IP traffic consumes resources significantly, and there is a need for intrusion prevention systems to perform at high speeds [79]. That is why the significance of reassembly may have been overlooked in the implementations of various intrusion prevention systems to ensure the highest performance.

Some of the tested intrusion prevention systems seem to have difficulties in correctly understanding all the semantics of various protocols. There were multiple examples in the results that show that an intrusion prevention system was able to detect attacks that were masked with evasions better when the attacks utilized certain protocols. This shows that understanding about different protocols can vary significantly in a single IPS device.

This study indicates that a free intrusion prevention system, Snort, performs as well as the majority of commercial solutions. There were not any results from the experiments that would imply otherwise. Snort was not last in any of the results, and its detection rate in the attacks against Ubuntu was third best.

The intrusion prevention system from Stonesoft was the only device that did not pass attacks which were masked with evasions in any of the experiments. It is fair to assume that Stonesoft has tested their product extensively against evasion techniques, which also explains the excellent results. They have released the evasion test tool, which was utilized in this study, that supports the notion of comprehensive testing.

Due to the nature of these findings, all the results of this study were disclosed to Finnish Communications Regulatory Authority[5]. It included a detailed list of successful evasion techniques and evasion combinations that were found against each of the tested intrusion prevention systems.

[5]`https://www.viestintavirasto.fi`

5.4 Reliability of the Results

The reliability of the results relies on the number of the attacks. A million attacks, masqueraded by randomly selected evasions and evasion combinations, guarantee that the test pool of evasion techniques is large. In previous studies [86] the magnitude of the results has not been as large as it is in this study. Since the same experiments, including the attacks with identical evasions and evasion combinations, were run against each DUT, the results are comparable.

As previously discussed, the configurations of intrusion prevention systems play an important role. The tested devices do not have the same capabilities, and they also have their own unique functionalities. That is why the configurations were standardized by following the guidelines that are presented in Section 4.2.2. The configurations were verified multiple times during the experiments by checking, for example, that the DUT was able to block an attack with and without obfuscation. It was also made sure that the devices were not in any kind of an error state, which could have an effect on the results.

The configurations of the tested IPS devices were usually hardened from the default configurations making them better against attacks. By using standardized configurations, the results of the experiments are comparable. The devices were configured to perform like each other that makes comparison possible although the possibility of a misconfiguration should not be overlooked. The used guidelines for standardized configurations should significantly decrease the risk of an error. The additional verifications of the proper behaviour of a DUT were also ways to ensure the reliability of the results.

5.5 Summary

This chapter listed the results of the experiments and analysed the findings. The results showed that almost all of the tested intrusion prevention systems were vulnerable to evasions and evasion combinations. It means that the threat that comes from the evasions is real.

The results showed that even some relatively old evasion techniques from the 1990s were successful against the state of the art intrusion prevention systems. Even though the techniques are old, they can be fine-tuned to fool most of the IPS devices. It was also seen from the results that combining multiple evasion techniques increased the possibility to find a way to evade detection. Additionally, the detection rates between the tested intrusion prevention systems varied significantly in Experiment D.

The results also indicated that normalization and reassembly have not been properly implemented in some of the tested intrusion prevention systems because the number of successful attacks utilizing evasion techniques was so high. Reassembling consumes resources significantly, and its significance may have been overlooked in some of the implementations.

6 Conclusions

Evasions are a real threat to the network security. The results from the experiments show that almost all the tested intrusion prevention systems were vulnerable to evasion techniques and evasion combinations.

The majority of the detection rates were over 95% in this study. When Windows was targeted in the attacks that were masked with various evasion techniques, 9 out of 10 devices dropped over 95 percent of the attacks. Two of them dropped over 99%. However, when targeting Ubuntu, only 4 out of 10 devices were able to accomplish 95% detection rates.

The results indicate that intrusion prevention systems are vulnerable to evasion techniques even though they were capable of detecting the majority of the attacks that were masked with evasions. Unfortunately, it is not enough because it leaves a possibility that some attacks go undetected when a suitable evasion technique or evasion combinations are in use. Successful attacks leave no logs to intrusion prevention systems. The fact also makes these attacks very serious.

Evasion techniques are not a new concept since the first academic papers about evasions were published in the 1990s. These old atomic evasion techniques were still effective against the state of the art intrusion prevention systems. Even though the techniques are old, they can be fine-tuned to fool most of the IPS devices. In certain cases, only one evasion technique was needed to utilize in an attack to avoid detection. The majority of these tested atomic evasion techniques can be fine-tuned to be effective against most of the tested intrusion prevention systems. Only one IPS device was able to block all the attacks that were masked with carefully fine-tuned atomic evasions.

There are millions of evasion combinations that can be utilized in attacks. By using evasion combinations, almost all the tested intrusion prevention systems can be tricked into passing malicious data. So, one evasion technique is not always enough to avoid detection, but combining multiple techniques increases the possibility in finding a way to evade detection.

The results indicate that normalization and reassembly have not been properly implemented in some of the tested intrusion prevention systems because the number of successful attacks utilizing evasion techniques was so high. Reassembling TCP/IP traffic consumes resources significantly. The significance of reassembly may have been overlooked in the implementations of various intrusion prevention systems to ensure the highest performance.

Generally, the default configurations of intrusion prevention systems were not strict enough to block the attacks that were masked with evasions. That is why most of the configurations were hardened. It is also worth noticing that even though a configuration process takes time, it is still beneficial to test configurations against attacks that are masked with evasions. Thus, it can be maximised that an intrusion prevention system can detect most attacks of this kind.

There are multiple ways to extend the work carried out in this thesis. This thesis does not take into account the parameter values of evasion techniques. It can be analysed if certain parameter values are more effective than the others when attacking various intrusion prevention systems. It is possible to study if the majority of the tested intrusion prevention systems are vulnerable to attacks that are masked with the same set of evasions and use the same parameter values.

The configurations of the tested intrusion prevention systems were hardened to improve the results. However, it is possible that default configurations are used, for example, in some corporations. That is why it can be tested how these devices operate with default configurations. It is also possible to compare the detection rates when using default configurations and hardened configurations to determine the impact of configuration changes.

The throughput performance was not studied in the scope of this thesis. The hardening of configurations improved the ability of intrusion prevention systems to detect attacks that were masked with evasions, but implications for performance are unknown.

References

[1] Stonesoft antievasion. Retrieved 27.06.2013 from `http://aet.stonesoft.com/stonesoft/`.

[2] Gartner information security hype cycle declares intrusion detection systems a market failure; money slated for intrusion detection should be invested in firewalls. Retrieved 27.06.2013 from `http://www.businesswire.com/news/home/20030611005056/en/Gartner-Information-Security-Hype-Cycle-Declares-Intrusion`, 2003.

[3] United states of america v. albert gonzalez, a/k/a segvec, a/k/a soupnazi, a/k/a j4guar17, hacker 1, and hacker 2. Retrieved 27.06.2013 from `http://www.wired.com/images_blogs/threatlevel/2009/08/gonzalez.pdf`, 2009.

[4] Cert-fi advisory on ids/ips device vulnerabilities that may circumvent protections. Tech. rep., Retrieved 27.06.2013 from `https://www.cert.fi/en/reports/2010/vulnerability385726.html`, 2010.

[5] Newly discovered threats pose serious risks for organizations worldwide. Retrieved 27.06.2013 from `http://www.stonesoft.com/en/company/press_and_media/releases/en/2010/18102010-2.html`, 2010.

[6] Department of defense strategy for operating in cyberspace. Retrieved 27.06.2013 from `http://www.defense.gov/news/d20110714cyber.pdf`, 2011.

[7] Evader users guide. Retrieved 27.06.2013 from `http://evader.stonesoft.com/assets/files/Evader_UsersGuide_20120905.pdf`, 2012.

[8] How to troubleshoot event id 2021 and event id 2022. Retrieved 27.06.2013 from `http://support.microsoft.com/kb/317249`, 2012.

[9] Manhattan u.s. attorney and fbi assistant director in charge announce 24 arrests in eight countries as part of international cyber crime takedown. Retrieved 27.06.2013 from `http://www.fbi.gov/newyork/press-releases/2012/manhattan-u.s.-attorney-and-fbi-assistant-director-in-charge-announce-24-arrests-in-eight-countries-as-part-of-international-cyber-crime-takedown`, 2012.

[10] Multilayer traffic normalization and data stream based inspection: Essential design principles of the stonesoft ips. Retrieved 27.06.2013 from `http://aet.stonesoft.com/assets/files/AET_Whitepaper2012-01-12_v3.pdf`, 2012.

[11] Stonesoft evader. Retrieved 27.06.2013 from `http://evader.stonesoft.com`, 2012.

[12] Ibm security network intrusion prevention system virtual appliance. Retrieved 27.06.2013 from `http://www-03.ibm.com/software/products/us/en/network-ips-virtual`, 2013.

[13] Metaploit user guide. Retrieved 27.06.2013 from https://community.rapid7.com/servlet/JiveServlet/previewBody/1563-102-13-4468/COM_UserGuide_4.6.pdf, 2013.

[14] Snort rules. Retrieved 27.06.2013 from https://www.snort.org/snort-rules/, 2013.

[15] Sourcefire virtual appliances and sourcefire virtual defense center. Retrieved 27.06.2013 from https://na8.salesforce.com/sfc/p/80000000dRH9KXPLJqkSwWBoW3e_vtLbnXOyiNg=, 2013.

[16] Stonesoft virtual ips. Retrieved 27.06.2013 from http://www.stonesoft.com/en/products/appliances/virtual-ips.html, 2013.

[17] Antichi, G., Ficara, D., Giordano, S., Procissi, G., and Vitucci, F. Counting bloom filters for pattern matching and anti-evasion at the wire speed. *Network, IEEE 23*, 1 (2009), 30–35.

[18] Atlasis, A. Attacking ipv6 implementation using fragmentation. *BlackHat Europe* (2012).

[19] Balzarotti, D. *Testing Network Intrusion Detection Systems.* PhD thesis, Politecnico di Milano, 2006.

[20] Bania, P. Evading network-level emulation. *arXiv:0906.1963* (2009).

[21] Bellovin, S. M. Security problems in the tcp/ip protocol suite. *ACM SIGCOMM Computer Communication Review 19*, 2 (1989), 32–48.

[22] Berglund, N. Telenor reports industrial espionage. Retrieved 27.06.2013 from http://www.newsinenglish.no/2013/03/17/telenor-reports-industrial-espionage/, 2013.

[23] Bidou, R. Ips shortcomings. *Black Hat USA* (2006).

[24] Bilge, L., and Dumitras, T. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security* (2012), ACM, pp. 833–844.

[25] Brumley, D., Newsome, J., Song, D., Wang, H., and Jha, S. Towards automatic generation of vulnerability-based signatures. In *Security and Privacy, 2006 IEEE Symposium on* (2006), IEEE, pp. 15–pp.

[26] Burns, D., and Adesina, O. Ccnp security ips 642-627 official cert guide: Network ips traffic analysis methods, evasion possibilities, and anti-evasive countermeasures. Retrieved 27.06.2013 from http://www.ciscopress.com/articles/article.asp?p=1728833&seqNum=3, 2011.

[27] Canavan, J. E. *Fundamentals of network security.* Artech House, 2001.

[28] Caswell, B., and Moore, H. Thermoptic camouflage: Total ids evasion. *Black Hat USA* (2006).

[29] Cheng, T.-H., Lin, Y.-D., Lai, Y.-C., and Lin, P.-C. Evasion techniques: Sneaking through your intrusion detection/prevention systems. *Communications Surveys & Tutorials, IEEE 14*, 4 (2012), 1011–1020.

[30] Cheswick, W. R., Bellovin, S. M., and Rubin, A. D. *Firewalls and Internet security: repelling the wily hacker*. Addison-Wesley Longman Publishing Co., Inc., 2003.

[31] Cohen, F. Managing network security-part 14: 50 ways to defeat your intrusion detection system. *Network Security 1997*, 12 (1997), 11–14.

[32] Corbet, J., Kroah-Hartman, G., and McPherson, A. Linux kernel development: How fast it is going, who is doing it, what they are doing, and who is sponsoring it. *The Linux Foundation* (2013).

[33] Deering, S., and Hinden, R. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), Dec. 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946.

[34] Detristan, T., Ulenspiegel, T., Malcom, Y., and Underduk, M. Polymorphic shellcode engine using spectrum analysis, 2003. Phrack.

[35] Dreger, H., Kreibich, C., Paxson, V., and Sommer, R. Enhancing the accuracy of network-based intrusion detection with host-based context. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2005, pp. 206–221.

[36] Dyrmose, M. Beating the ips. Tech. rep., SANS Institute, January 2013.

[37] Falliere, N., Murchu, L. O., and Chien, E. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response* (2011).

[38] Fogla, P., Sharif, M., Perdisci, R., Kolesnikov, O., and Lee, W. Polymorphic blending attacks. In *Proceedings of the 15th USENIX Security Symposium* (2006), pp. 241–256.

[39] Gont, F. Security Assessment of the Internet Protocol Version 4. RFC 6274 (Informational), July 2011. Internet Engineering Task Force.

[40] Gorton, A., and Champion, T. Combining evasion techniques to avoid network intrusion detection systems, 2004. Skaion.

[41] Halliday, J. Anonymous launches attacks in wake of megaupload closure. Retrieved 27.06.2013 from `http://www.theguardian.com/technology/2012/jan/20/anonymous-attacks-after-megauploads-closure`, 2012.

[42] Handley, M., Paxson, V., and Kreibich, C. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *Proc. USENIX Security Symposium* (2001), pp. 9–9.

[43] Hypponen, M. Three types of online attack. Retrieved 27.06.2013 from `http://www.ted.com/talks/mikko_hypponen_three_types_of_online_attack.html`, 2011.

[44] Juan, L., Kreibich, C., Lin, C.-H., and Paxson, V. A tool for offline and live testing of evasion resilience in network intrusion detection systems. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2008, pp. 267–278.

[45] Krishnan, S. Handling of Overlapping IPv6 Fragments. RFC 5722 (Proposed Standard), Dec. 2009. Internet Engineering Task Force.

[46] Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., Postel, J., Roberts, L. G., and Wolff, S. A brief history of the internet. *ACM SIGCOMM Computer Communication Review 39*, 5 (2009), 22–31.

[47] Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., and Das, K. Analysis and results of the 1999 darpa off-line intrusion detection evaluation. In *Recent Advances in Intrusion Detection* (2000), Springer, pp. 162–182.

[48] Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., Weber, D., Webster, S. E., Wyschogrod, D., Cunningham, R. K., et al. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings* (2000), vol. 2, IEEE, pp. 12–26.

[49] Marty, R. Thor: A tool to test intrusion detection systems by variations of attacks. Master's thesis, Swiss Federal Institute of Technology, 2002.

[50] McConnell, S. *Code complete.* O'Reilly Media, Inc., 2004.

[51] McHugh, J. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM transactions on Information and system Security 3*, 4 (2000), 262–294.

[52] Mirkovic, J., and Reiher, P. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review 34*, 2 (2004), 39–53.

[53] Mutz, D., Vigna, G., and Kemmerer, R. An experience developing an ids stimulator for the black-box testing of network intrusion detection systems.

In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual* (2003), IEEE, pp. 374–383.

[54] Newman, R. C. *Computer security: Protecting digital resources.* Jones & Bartlett Publishers, 2009.

[55] Palmer, A., and Merritt, M. 2012 norton cybercrime report. Tech. rep., Retrieved 27.06.2013 from `http://now-static.norton.com/now/en/pu/images/Promotions/2012/cybercrimeReport/2012_Norton_Cybercrime_Report_Master_FINAL_050912.pdf`, 2012.

[56] Patton, S., Yurcik, W., and Doss, D. An achilles heel in signature-based ids: Squealing false positives in snort. *Proceedings of RAID 2001* (2001). University of California-Davis.

[57] Paxson, V. Bro: a system for detecting network intruders in real-time. In *7th Annual USENIX Security Symposium* (1998).

[58] Polychronakis, M., Anagnostakis, K. G., and Markatos, E. P. Network–level polymorphic shellcode detection using emulation. In *Detection of Intrusions and Malware & Vulnerability Assessment.* Springer, 2006, pp. 54–73.

[59] Postel, J. DoD standard Internet Protocol. RFC 760, Jan. 1980. Internet Engineering Task Force.

[60] Postel, J. DoD standard Transmission Control Protocol. RFC 761, Jan. 1980. Internet Engineering Task Force.

[61] Ptacek, T. H., and Newsham, T. N. Insertion, evasion, and denial of service: Eluding network intrusion detection. Tech. rep., Secure Networks, Inc, 1998.

[62] Puppy, R. F. A look at whisker's anti-ids tactics. Retrieved 27.06.2013 from `http://www.ussrback.com/docs/papers/IDS/whiskerids.html`, 1999.

[63] Roelker, D. J. Http ids evasions revisited. *Sourcefire Inc* (2004).

[64] Rohrmair, G. T., and Lowe, G. Using csp to detect insertion and evasion possibilities within the intrusion detection area. In *Formal Aspects of Security.* Springer, 2003, pp. 205–220.

[65] Rowan, T. Intrusion prevention systems: superior security. *Network Security 2007*, 9 (2007), 11–15.

[66] Rubin, S., Jha, S., and Miller, B. P. Automatic generation and analysis of nids attacks. In *Computer Security Applications Conference, 2004. 20th Annual* (2004), IEEE, pp. 28–38.

[67] Sanger, D. E. Obama order sped up wave of cyberattacks against iran. *The New York Times 1* (2012).

[68] Sassaman, L., Patterson, M. L., and Bratus, S. A patch for postel's robustness principle. *Security & Privacy, IEEE 10*, 2 (2012), 87–91.

[69] Scarfone, K., and Mell, P. Guide to intrusion detection and prevention systems (idps). *NIST Special Publication 800*, 2007 (2007), 94.

[70] Shankar, U., and Paxson, V. Active mapping: Resisting nids evasion without altering traffic. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on* (2003), IEEE, pp. 44–61.

[71] Shirey, R. Internet Security Glossary, Version 2. RFC 4949 (Informational), Aug. 2007. Internet Engineering Task Force.

[72] Siddharth, S. Evading nids, revisited. Retrieved 27.06.2013 from `http://www.symantec.com/connect/articles/evading-nids-revisited`, 2005.

[73] Smith, R., Estan, C., and Jha, S. Backtracking algorithmic complexity attacks against a nids. In *Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual* (2006), IEEE, pp. 89–98.

[74] Song, D. Fragroute. Retrieved 27.06.2013 from `http://www.monkey.org/~dugsong/fragroute/`.

[75] Stallings, W. *Network Security Essentials: Applications and Standards, 4/e.* Prentice Hall, 2007.

[76] Sullo, C., and Lodge, D. Nikto - the manual. Retrieved 27.06.2013 from `http://cirt.net/nikto2-docs`, 2013.

[77] Taleck, G. Ambiguity resolution via passive os fingerprinting. In *Recent Advances in Intrusion Detection* (2003), Springer, pp. 192–206.

[78] Vacca, J. R. *Network and system security.* Elsevier, 2010.

[79] Varghese, G., Fingerhut, J. A., and Bonomi, F. Detecting evasion attacks at high speeds without reassembly. In *ACM SIGCOMM Computer Communication Review* (2006), vol. 36, ACM, pp. 327–338.

[80] Vigna, G., Robertson, W., and Balzarotti, D. Testing network-based intrusion detection signatures using mutant exploits. In *Proceedings of the 11th ACM conference on Computer and communications security* (2004), ACM, pp. 21–30.

[81] Vutukuru, M., Balakrishnan, H., and Paxson, V. Efficient and robust tcp stream normalization. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on* (2008), IEEE, pp. 96–110.

[82] Wang, H. J., Guo, C., Simon, D. R., and Zugenmaier, A. Shield: Vulnerability-driven network filters for preventing known vulnerability exploits. In *ACM SIGCOMM Computer Communication Review* (2004), vol. 34, ACM, pp. 193–204.

[83] Watson, D., Smart, M., Malan, G. R., and Jahanian, F. Protocol scrubbing: network security through transparent flow modification. *IEEE/ACM Transactions on Networking (TON) 12*, 2 (2004), 261–273.

[84] West, M., and McCann, S. TCP/IP Field Behavior. RFC 4413 (Informational), Mar. 2006. Internet Engineering Task Force.

[85] Whitman, M. E., and Mattord, H. J. *Principles of information security.* Course Technology Ptr, 2011.

[86] Xynos, K., Sutherland, I., and Blyth, A. Effectiveness of blocking evasions in intrusion prevention systems, 2013. University of South Wales.

[87] Young, G., and Pescatore, J. Defining the next-generation firewall. *Gartner Core RAS Research Note G 171540* (2009), 1–3.

[88] Young, G., and Pescatore, J. Magic quadrant for intrusion prevention systems. *Gartner Core RAS Research Note G 222572* (2012), 1–16.

[89] Young, G., and Pescatore, J. Magic quadrant for unified threat management. *Gartner Core RAS Research Note G 226111* (2012), 1–21.

[90] Zetter, K. Hacker sentenced to 20 years for breach of credit card processor. Retrieved 27.06.2013 from `http://www.wired.com/threatlevel/2010/03/heartland-sentencing/`, 2010.

[91] Zetter, K. Tjx hacker gets 20 years in prison. Retrieved 27.06.2013 from `http://www.wired.com/threatlevel/2010/03/tjx-sentencing/`, 2010.

A Evasions

Table A1 lists all the evasion techniques that were used in this study. Column C indicates the evasion techniques that were applicable when utilizing the Conficker attack. In the same way, column H lists the evasion techniques that were available and utilized with the HTTP phpBB Highlight attack.

The descriptions of the utilized evasion techniques were taken from Evader. The description of an evasion technique can be fetched, for example, by using a command ./evader –attack=conficker –evasion=ipv4_frag where the attack and evasion parameters are changed according to the evasion technique.

Table A1: All the evasions used in this study

Evasion	**Description**	**C**	**H**
IPv4 fragmentation	Fragment IPv4 packets to a given size.	X	X
IPv4 options	Send duplicate IPv4 packets with a changed payload and some IPv4 options.	X	X
MSRPC big-endian	Send MSRPC messages in the big-endian byte order.	X	-
MSRPC groupsends	Group MSRPC fragments to a single lower layer send.	X	-
MSRPC NDR modifications	Set NDR types, which are not related to endianness.	X	-
MSRPC request segmentation	Set the maximum number of bytes written in a single MSRPC fragment.	X	-
NetBIOS chaff	Send extra NetBIOS packets to break the packet flow.	X	-
NetBIOS initial chaff	Send chaff NetBIOS packets when establishing the NetBIOS connection.	X	-
SMB chaff	Send SMB messages to baffle inspection.	X	-
SMB decoy trees	Perform extra SMB writes before each normal SMB write.	X	-
SMB filename obfuscation	Obfuscate the tree name, which is used in the SMB NT Create AndX method.	X	-

Table A1 : *(continued)*

Evasion	**Description**	**C**	**H**
SMB write segmentation	Set the maximum number of bytes that are written in a single SMB write.	X	-
SMB WriteAndX padding	Insert extra padding between the WriteAndX header and payload.	X	-
TCP chaff	Send chaff TCP segments to baffle inspection.	X	X
TCP initial sequence number	Set the initial sequence number used by a TCP socket to 0xffffffff - n.	X	X
TCP timestamp option settings	Set initial values to TCP timestamps.	X	X
TCP nocwnd	Disable TCP congestion avoidance.	X	X
TCP no fast retrans	Disable TCP fast retransmit.	X	X
TCP segment order	Change the order of TCP segments.	X	X
TCP segment overlap	Make segments that are sent in a single send() to overlap.	X	X
TCP PAWS elimination	Send extra TCP segments that are eliminated by PAWS in the destination stack.	X	X
TCP receive window	Set the receive window of a TCP socket. This is used to force the other host to send small TCP segments.	X	X
TCP segmentation	Set the MTU of a TCP socket. This is used to create non-standard TCP segmentation.	X	X
TCP TIME-WAIT decoys	Open decoy connections from the same TCP source port before the actual attack.	X	X
TCP timestamp echo reply modifications	Modify the echo reply field of a TCP timestamp.	X	X
TCP urgent data	Set urgent data into TCP segments.	X	X

Table A1 : *(continued)*

Evasion	**Description**	**C**	**H**
HTTP header linear whitespace	Convert whitespaces in HTTP headers to linear whitespaces (LWS) with a given probability.	-	X
HTTP known user agent	Set the user agent to a widely known value.	-	X
HTTP request line separator	Set the separator, which is used in the request line.	-	X
HTTP request method	Set the request method. Possible values are, for example, GET, POST and HELLO.	-	X
HTTP request pipelined	Send a valid request pipelined before the exploit.	-	X
HTTP URL absolute	Change relative URIs into absolute URLs.	-	X
HTTP dummy paths	Add dummy paths into an URL.	-	X
HTTP URL encoding	Encode characters in URL.	-	X
HTTP request version	Set the request version number.	-	X