Systematic approach towards Analysis and Mitigation of Advanced Evasion Techniques

Master of Science in Technology thesis
University of Turku
Department of Future Technologies
Security of Networked Systems
December 2019
Jaspreet Singh Pannu

Supervisors:
Seppo Virtanen
Antti Hakkala

UNIVERSITY OF TURKU
Department of Future Technologies

JASPREET SINGH PANNU: Systematic approach towards Analysis and Mitigation of Advanced Evasion Techniques

Type of thesis, 59 p., 0 app. p.
Security of Networked Systems
December 2019

Advanced Evasion Techniques (AETs) can successfully evade most network security devices and execute attack on target system. This is still an occurring problem, even after 20 years since the disclosure of evasion techniques and how they can be used to bypass network security. Network security solutions, such as Intrusion Detection and Prevention Systems (IDPS) still struggle and are vulnerable to most of the evasions techniques mentioned identified in 1998.

In this thesis, a systematic analysis of advanced evasion techniques (AETs) is presented in the first two phases. Based on the results of the analysis, new mitigation methods against AETs are proposed in the third phase. Four experiments were executed in each of three different phases using advanced evasion techniques to masquerade the attack. The target of this analysis was to first recognize which combinations of evasions are most effective and which individual/ single evasion techniques are effective by itself. The final phase was to implement proposed mitigation methods and test the results.

The results from the analysis showed that 4-6 % of AETs, can successfully masquerade attacks and bypass one of the most modern and updated network security solution. Proposed mitigation methods are capable of normalizing traffic much better while improving the results significantly. In many cases 100 % attack techniques were mitigated and some particular techniques exploiting headers of protocol were also mitigated completely. Nonetheless, when evasion techniques are used in complex combinations, results become concerning and it is important to note that the danger from AETs may still persist.

Keywords: Network Security, AET, Advanced Evasion Techniques, Evasions, IPS, Intrusion Prevention System, IDPS, Intrusion Detection and Prevention System, Snort, Segmentation

Abbreviations and Acronyms

AET	Advanced Evasion Techniques
CNSS	Committee on National Security Systems of United States of America
CVE	Common Vulnerabilities and Exposures
(D)DoS	(Distributed) Denial of Service
DUT	Device Under Test
GDP	Gross Domestic Product
HTTP	Hyper Text Transfer Protocol
IDES	Intrusion Detection Expert System
IETF	Internet Engineering Task Force
IP	Internal Protocol
LTS	Long Term Support
MSRPC	Microsoft Remote Procedure Call
(N)IDPS	(Network) Intrusion Detection and Prevention System
(N)IDS	(Network) Intrusion Detection System
(N)IPS	(Network) Intrusion Prevention System

OSI	Open Systems Interconnection
phpBB	PHP Bulletin Board
RFC	Request For Comments
SMB	Server Message Block protocol
TCP	Transmission Control Protocol
TTL	Time To Live
TLS	Transport Layer Security

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The network of systems on the internet is increasing exponentially and so are the threats. On a global estimate, there are 4.48 billion active internet users [1]. According to the Symantec report, the overall endpoint attacks was observed to increase by 56 % from 2017 in 2018. Moreover, 1 in 10 URLs was found to be malicious in the year 2018. They further emphasized that the malicious attacks on small companies increased significantly in contrast to bigger organizations [2]. These malicious cyber attacks create huge loses to the global economy, from which organizations and bodies involved may not be able to recover from easily. According to the study from the Center for Strategic and International Studies (CSIS) with McAfee, an estimated $600 billion is lost to Cybercrime on a yearly basis. These estimated losses are calculated to be about 1 % of global GDP [2].

Cyber attacks are not only limited to active internet users, smartphones or computers. They can be targeted to any device that is connectable to the internet. According to statica, the number of IoT devices connected to the internet is around 26 billion and is estimated to increase by up to 75.4 billion by 2025 [3]. One of the major examples of how IoT devices can be used as a platform to execute the attack is by malware, such as Mirai. Mirai malware used IoT devices as a platform to execute a big DDoS attack [2].

All devices have unpredictable and various system configurations. The varying configurations make it impractical to searching and identifying one point of defense against every threat. This makes devices vulnerable to cyber attacks. The vulnerabilities of these devices to new attacks are making them prime targets to cyber crimes ranging from identity theft to cyber extortion, monetization of these threats. Several thousands of new variants of malware are developed on a daily basis which are capable of exploiting the vulnerabilities of devices. These new variants are still challenges for cyber security [4].

Various methods to tackle cyber attacks are being deployed to protect various OSI layer levels. One of the methods that is used to protect the network from potential threats and exploits is Network Intrusion Detection and Prevention System (NIDPS) along with Firewall. A firewall is used to limit the network access to traffic on both sides by restricting various IPs according to defined rules. Whereas the NIDPS detects threats and attacks by matching signatures or by analyzing anomaly activity over the network. When the Network Intrusion Detection System (NIDS) detects a threat, it logs it and sends an alarm to admin. Network Intrusion Prevention System (NIPS) along with all capabilities of the Intrusion Detection System is also capable of taking countermeasures against the threat. It is capable of dropping the threat packets and reports to the admin or firewall to block a particular IP or IP group. [5]

NIPS works on the analysis of live real-time in-line traffic by comprehensively inspecting live data. This comprehensive inspection is resource exhaustive and proves to be the network performance bottleneck. This exhaustion makes NIPS directly susceptible to DDoS attacks. Another issue that NIPS/NIDS faces is the interpretation of data which may appear differently than the target. This might result in bypassing the signature matching of NIPS and anomaly detection. [6]

Evasion techniques according to Newsham and Ptacek are "techniques that involve exploiting inconsistencies between the analyzer and an end system in order to slip packets past the analyzer" [6]. Evasion techniques can be customized to exploit any particular vulnerability or limitations of the NIPS, including its protocol analysis and segmentation. These techniques can be utilized to execute serious attacks against various targets by delivering malicious payload beyond the Intrusion Prevention System. Complex evasion or a combination of various evasion techniques can be described as advanced evasion techniques (AETs) which can be used as countless different combinations; hence, resulting in very serious implications for the security of the network. The major problem the NIPS encounters is the defense against these evasion techniques.

1.1 Statement of problem

Over the past 20 years from the first disclosure of evasion techniques, several studies have proven the effectiveness of AETs; however, these efforts are to no avail because the AETs are still capable of penetrating the modern and updated IDPS. This creates a need to create a method to counter the vulnerability of IDPS against AETs.

1.2 Aim of Thesis

The aim of my thesis project is to systematically analyze the AETs and propose mitigation methods against the AETs based on the results of the analysis from experiments carried out.

First, the analysis of AETs with several evasion attacks would be performed with the purpose of recognizing the kind of evasions that are more successful in evading IDPS, as well as protocols that are more prone to evasion attacks. The information obtained from the result analyzed would serve as the basis for the next stage of this thesis. Further, mitigation methods would be performed to prevent the success of AET based attacks.

1.3 Thesis structure

This thesis is divided into 8 chapters. After the introduction of the thesis, chapter 2 and 3 explains details of Intrusion detection and prevention system and Advanced evasions techniques. Further, the environment of the experiment and methodologies used for the experiment are explained in chapters 4 and 5.

Implementation of the experiment and evaluation of results are described in chapters 6 and 7. In these two chapters, implementation in different phases of the experiment is explained and how results retrieved from every stage are evaluated. All observations and results retrieved from the mitigation methods are explained in chapter 7.

Finally, we conclude with results from analysis about the effectiveness of Intrusion detection and prevention system against advanced evasion techniques. Also understanding the implications and possibilities of the mitigation methods proposed for the experiment.

Chapter 2

Intrusion Detection and Prevention System - IDPS

Network intrusion detection and prevention system is fundamental and a very crucial part of the network. It monitors and analyzes traffic to find and defend against any malicious attack. An Attack as defined by RFC4949 is “An intentional act by which an entity attempts to evade security services and violate the security policy of a system. That is, an actual assault on system security that derives from an intelligent threat.” [7]. Committee on National Security Systems (CNSS) of the United States of America defines attack as “Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself” [8].

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possibles incident in which an intruder gains or attempts to gain access to a system or system resource without having authorization [9].

In a computer security technology planning study, a United States Air Force paper written by James P Anderson published in October 1972 mentioned that “Air force has become increasingly aware of the problems of computer security. This problem has intruded upon virtually every aspect of US Air Force operation and administration." He highlighted that major problems of shared use of computer systems with different classification " the fact that there is growing requirement to provide shared use of computer systems containing information of different classification levels and need-to-know requirements in a user population not uniformly cleared or access-approved." [10].

In 1980 James P. Anderson published another study "Computer Security threat monitoring and surveillance". In this study, he disc "How to use accounting audit files to detect unauthorized access" is considered as the original idea behind automated Intrusion Detection. [11].

2.1 Intrusion Detection and Prevention System

2.1.1 Intrusion Detection System (IDS)

Intrusion Detection System is the software/system that automates the process of Intrusion Detection. In 1986, Dorothy Denning and Peter Neumann developed the first model of real-time intrusion detection systems. They developed a rule-based prototype system trained to detect already known activities. This prototype was named the Intrusion Detection Expert System (IDES) [12].

IDS sits on the side of the network or host system and receives a copy of the inline traffic stream to detect and report malicious activities.

It creates real-time alarms and reports them to the administrator. All types of IDS typically perform some particular functions that are enumerated below:

1. It produces regular reports about monitoring any particular malicious or suspicious event

2. It records the information either on the local system or centralized logging remote server

3. It provides information to system administration in form of alerts regarding any malicious or important events

2.1.2 Intrusion Prevention System (IPS)

IPS has all the capabilities of IDS and can also attempt to stop the incidents of intrusion [9]. IPS can be made to work as IDS with options for preventing and dropping or normalizing traffic disabled. It is also referred to as Intrusion Detection and Prevention System (IDPS) [9].

IPS works inline on the network, so all network traffic flows directly through this system. Due to the inline traffic flow, the system has capabilities to block or stop intrusions. IPS and IDS provides many similar capabilities whereby the network administrator is able to turn off prevention functionality in IPS making it function as IDS.[1]

IPS can prevent intrusion with several response techniques which are highlighted below;

- IPS stops attacks by terminating or blocking connection to target or host or vise versa
- IPS can change the security environment by altering the configuration of network devices like firewall, router or switches
- IPS can change the content of the attack file or packet by removing or replacing malicious content and forwarding remaining content to the destination

Along with the above-mentioned techniques, there is also another characteristic of IDPS that suggests it is impossible to provide completely accurate detection information. This characteristic may result in false positive or false negatives. False-positive occurs when IDS detects a normal or non-malicious abnormal activity and identify it as malicious. On

[1]From now on IDPS and IPS term would be used interchangeably to represent IDS/ IPS, since Intrusion Prevention System can be configured to work like Intrusion Detection System by disabling it and stopping it from taking action against events

the other hand, false negative occurs when IPS fails to detect real malicious activity or threat and passes it as normal traffic [9].

If the situation occurs that there are too many false-positive reports by the IDPS, there would consequently be false alarms and long reports generated without any real attack or malicious event. This can be problematic by making result analysis difficult, false assumption of a system fault which might result in allowance of undetected malicious events [13]. In 1990s, one of the major reasons IPS was less adapted as a network security solution is because it was believed that entire network traffic would become slow, the clean network traffic would be marked malicious and dropped due to excessive false positive. It is important to note that adversaries can take advantage of the situation by pushing extensive non-malicious data to create false positives and send malicious data also which might not be noticed [14].

Unlike false positives, false negatives signify bad detection policy or configuration issue. Moreover, if IDPS can not detect malicious events or attacks then there is no point in having the system on the network. Furthermore, false negatives generally happen due to the presence of outdated signatures, rules, or lack of adequate resources to perform the detection task. The number of false negatives can be reduced significantly by updating system, signature database and resolving IDPS configuration [15].

From the early year 2000, the market take on IPS was really low due to the fear of false positive. Organizations were ready to let anomalous activity into their networks rather than taking the chance of dropping a harmless network connection. In the year 2003, research vice president for Gartner stated that "Intrusion detection systems are a market failure, and vendors are now hyping intrusion prevention systems, which have also stalled," [14].

Over the period, evolution in the technology of IPS has taken the huge leap and its adoption has increased due to researches to improvement in the detection policies of IPS, as well as the reduction in false positives. Even though the concern of false-positive and alerts remains. In the year 2018, the Gartner Magic quadrant reported that IDS/ IPS offers the best detection, central prevention and response solution on a network [16].

2.2 Components of IDPS

The typical components of IDPS are the sensor, console, database server and management server as explained below [9].

- **Sensor**: This component monitors and analyzes the traffic of the network for any malicious activity.
- **Console**: This is a program or software that provides an interface of IDPS, its sensor, database for updating system or configuring the system. The console is typically installed on a system to provide an interface to the user or administrator.
- **Database server**: This component is a database that stores information of events produced by sensor or management server.
- **Management server**: This is a centralized device. This processes the information received from all the different sensor and analyze or manage them. The management server can analyze the traffic from consolidated data received from the various sensor. This analysis can recognize activity by correlation which might not be possible by a single sensor alone.

2.3 Different Types of IDPS

IDPS technologies exist in various types, depending on the kind of data and process they monitor. The four most common types of IDPS are enumerated below.

1. **Network-Based Intrusion Prevention System**: This is a type of IDPS that monitors network traffic entering or exiting a particular network or network segment [9].

It is usually deployed at the border of the network either along with or near to Firewall. This type of IDPS monitors and analyzes network, transport, and application layer data to identify and prevent any malicious activity [9].

2. **Host Based Intrusion Prevention system**: This type is deployed on a particular host, mostly critical systems. It monitors and analyzes network traffic, as well as processes, logs, and application activities for any abnormal and malicious activity [9].

3. **Wireless Intrusion Prevention System**: This is the type that monitors and analyze wireless traffic and networking protocol. This is deployed either in a particular organization's wireless range to monitor and detect any malicious activity within the network or it could also be used to monitor unauthorized wireless activity [9].

4. **Network Behavior Analysis (NBA)**: This is a type that monitors and examines network traffic to identify threats like certain malware flow, policy violation and unusual traffic that can cause DDoS attack [9].

2.4 Methodologies used by IDPS

IDPS uses many different event detection methods. IDPS uses more than one method for broad and more precise detection and prevention depending upon the requirements. Explained below are the most common IDPS methodologies.

2.4.1 Signature based Detection

Signature is a specific pattern, used to represent a textual or binary string that uniquely corresponds to the known vulnerabilities, exploits, threats and implementation of attacks. Signature-based detection is the process of comparing these patterns against data to identify possible attacks or known threats. An example of signature-based detection is a telnet attempt to login with username “root”, which is not a normal event; however, is a violation of security policy [9].

The signature-based detection system is vulnerable to many obfuscations or simple manipulation of payload or content attack to dodge the detection system. It has been recommended that signatures based on vulnerability are much more effective than signatures based on exploits [17]. The moment any new vulnerability is found then signature based on that would be much more effective in detecting any attack. These would be exploited generic, so any exploit trying to exploit specific vulnerability can be detected [17, 18].

IDPS maintains a separate database for all varieties of signatures. These IDPS compares network traffic against these various signatures to detect any possible attack. The success of detection depends on the capability of the processing of network traffic by IDPS. Moreover, it is recommended that IDPS shall be equipped with pre-processor to normalize the encoded strings and restore the original semantics of the attack. This is essential as otherwise, a simple mutation in data can evade detection [19].

Effectiveness of Signature-based detection by IDPS depends majorly on updated signatures. The signature database must be updated for any new vulnerabilities. This detection methodology is not effective on new vulnerabilities, new attack vectors and more especially zero-day attacks [20].

2.4.2 Anomaly Based Detection

The Anomaly-based detection method is based on monitoring and classifying system activity or behavior as normal or anomalous. It is the process of comparing system activity to the predefined behavior which is accepted as normal behavior to detect any abrupt deviation [9, 21]. An IDPS that uses anomaly-based detection has details of the network including users, connections, host and everything related, known as profiles. These profiles are generated over the period called the training period. Furthermore, the profiles can either be static or dynamic [9, 21]. Static profiles remain unchanged unless explicitly directed to generate a new profile by the administrator. On the other hand, dynamic profiles are updated regularly, hence changes according to the use of the system. This process defining profiles is very important as the efficiency of IDPS depends on how well normal behavior is defined. The IDPS engine must be able to cut through all the various protocols at all levels. The process of protocol analysis is expensive in terms of computational complexity. Nonetheless, it helps in understanding the traffic in detail and produces less false positives [9, 20, 21].

Another major benefit of this detection technique is that it can detect unknown threats, in contrast to signature-based technique [9, 21]. In the process of comparing the predefined normal behavior with the current behavior of the system. If any malicious activity is consuming system's processing or internet resources an alarm is raised, as this is unusual behavior and can be detected as an anomalous behavior [9, 21, 22].

The major drawback of this detection technique is that while still in the training period, it is susceptible to evasion attacks. If this malicious activity comes under normal behavior then the IDPS will be unable to raise any flag or alarm because the IDPS begins to consider the malicious event as normal behavior [23].

2.4.3 Stateful Protocol Analysis

According to Karen Scarfone "The 'stateful' in stateful protocol analysis insinuates the IDPS is capable of understanding and tracking the state of the network, transport, and application protocols that have a notion of state." [9]. The stateful protocol analysis is the process of comparing current system activity or event against a predefined profile that defines how a particular protocol shall work under normal scenario [9]. Unlike Anomaly-based detection where profiles are network or host-based, stateful protocol analysis relies on universal profiles prepared by specific vendors or standard bodies like Internet Engineering Task Force (IETF) Request for Comments (RFC) that specifies all details of how the protocol should be utilized in communication [9].

In stateful protocol analysis, protocols are checked in-details for individual commands and sometimes even there responses. Checks are made to track if data is more or less than what is defined in the protocol profile, if someone trying to send issue command out of order, or if the command is accessing resources without authentication and authorization [9]

Stateful protocol analysis very resources intensive because it maintains the state of (what sessions?) sessions for many concurrent sessions, and also the complexity of doing in-depth protocol analysis. Some vendors do not specify details of their protocols, which makes it difficult for analysis. Many conflicts arise on the implementation of various versions of a protocol and this implementation varies depending on the host operating system [23].

Although stateful protocol analysis performs in-depth analysis, it might be unable to detect new attacks that do not violate protocols, or does not perform any benign activity like that of Denial of Service attack (DoS) [23].

Chapter 3

Advanced Evasion Techniques

3.1 Advanced Evasion Techniques (AET)

Advanced evasion techniques refer to individuals or combination of many individual evasion techniques that can successfully execute an attack on a target system while avoiding any detection and dodging network security devices likes IDPS.

In 2010, the term AET was coined by Stonesoft Corporation which is now known as Forcepoint. AETs can be broadly defined as any technique or combination of techniques that increase the complexity of evasion and effectiveness to evade detection while exploiting the IDPS resources and increasing computational demand to detect attacks. According to Mark in whitepaper with International Computer Security Association (ICSA) Labs "These AETs prey upon protocol weaknesses and the permissive nature of network-based communication, exponentially increasing the number of evasions that can bypass even the most up-to-date IPS technologies" [24].

Ptacek and Newsham discussed evasion techniques in their paper of 1998. Evasion techniques are still effective against the modern network defense system even after 20 years. Moreover, tools have been developed over some time to exploit various layers of TCP/IP protocol and evade the detection of IDPS. Although, it is not possible to explain when or what level of complexity or sophistication of any evasion technique can be stated as an advanced evasion technique [25].

Forcepoint (formerly known as Stonesoft Corporation) developed a testing tool that uses AETs known as Evader in 2010. Evader is a deep inspection tool to test detection capabilities with AETs of a new combination of evasions. The Forcepoint's team successfully evaded all network industry's IDPS. Their research results in Finnish Transport and Communication Agency, National Cyber Security Centre (NCSC-FI's CERT) disclosing the first 23 new evasions and later another batch of 124 evasion techniques [24].

3.2 Evasions techniques

Evasion techniques are categorized according to the way they are used to evade the detection system. These techniques are not only limited to categories but emphasized due to extensive research is done and current systems still being vulnerable to the same techniques [19]. Below, the categorization is done according to different researches done on evasion techniques and evaluation of these techniques on IDPS by [19, 26, 27].

3.2.1 Packet splitting

IP fragmentation and TCP segmentation are forms of packet splitting. Fragmentation is an integral part of IP Protocol RFC791. It allows dividing bigger information to smaller Maximum Transmission Unit (MTU) to make it compatible with different network media size limitations. One message or the whole file can be fragmented into many smaller sized non-overlapping fragments [19, 28]. Similarly, TCP being a connection-oriented protocol uses segmentation to divide the big size packet to smaller ones and include a TCP header to maintain the connection. Furthermore, both IP and TCP protocols allow the fragments or packets to arrive in any random order instead of a proper order [6, 19, 26].

The fragmented data packets can arrive in any sequence, as it is the responsibility of the end machine to reassemble them [6]. For any network monitor like IDPS, this can be a complex and challenging task. Newsham and Ptacek mentioned in their 1998 paper, “an IDS that does not properly handle out-of-order fragments is vulnerable; an attacker can intentionally scramble fragment streams to elude the IDS” [6]. The system has to reassemble all the fragments to understand original content or detect the attack. An example of possible evasion in the scenario of fragmented data packet could be, if the signature of attack for detection is “/attack/exploit.php” and message is divided in fragments of “/atta”, another one “ck/exp”, as well as “loit.php” and detection system losses track on one of these packets [6, 19]. Another example of possible evasion could be if overlapping segments or fragment is sent, which is malicious and overwrite the last packet [6, 19]. It is important to note that if IDPS does not assemble the fragmented packet properly and misses one, or unable to replace it with a new packet, which target system is capable of handling. This will result in miss matching of signature, hence attack evades detection [26].

IDPS has to maintain a “per-connection state queue” to keep track of every connection in a large network. This is a resource exhaustive task, although IDPS system nowadays has ample memory to handle many states simultaneously; however, the system can run out of resources due to extensive network traffic [6, 29].

3.2.2 Denial of Service (DoS)

DoS attack occurs when an attack exhausts the computing resources of any system and makes it completely ineffective. The purpose of DoS attacks is to starve the system completely of its available resources by consuming more than it can provide [30]. The most common DoS attacks are “Ping to the death” that send oversized ICMP ping which can crash a system and other attacks like “Chargen” and “teardrop” that results in the same condition of target system [6, 31].

DoS attacks can disable IDPS by exhausting resources. When IDPS crashes or is disabled, it is designed to switch to a default mode. In the default mode, system switches to either “fail open mode” which completely opens the unchecked network traffic, or “fail closed mode” which completely closes network traffic for internal network or system [32, 33]. DoS attacks are not easy to defend. Researches to handle this DoS attacks has given many solutions. These solutions include rate-based IPS that has proven to handle traffic up to a very high rate, using data mining to detect DoS attack and also connection rate limiting solution [34].

DoS attacks are mostly classified as flood attacks or crafted packet attacks. The DoS attacks that originate simultaneously from several sources and locations are distinguished as Distributed Denial of Service (DDoS) attack [35, 36, 37]. These attacks create severe damages and are difficult to handle. DDoS again exploits the fundamentals of IP Protocol by using the fake source address to send attacks on the destination address which results in very high traffic [6, 19]. The sudden high traffic can result in complete resource exhaustion and shutting down of the system. Moreover, this DDoS traffic can reach more than 500Mbps and is capable of defeating almost all solutions. These attacks are executed by devices affected by different malware. The recent attack by devices infected by Mirai malware, traffic of DDoS attack was reported to be about one terabyte on French web host called OVH [31, 38].

3.2.3 Encryption

When payload or data stream is encrypted using complex algorithms, the data is transferred securely and decryption is done at the end system [39]. This is a very secure approach and protects data from any adversaries on the network. The adversaries could be MATE (Man At The End) or MITM (Man In The Middle) attack. Encryption makes data completely unrelated and unreadable. If complex encryption methods are used, then the security of data is significantly high. [39]

Similar techniques of protecting data can easily be used by an attacker to mask or encrypt the attack data. IDPS can not understand and interpret this data, making it nearly impossible for IDPS to detect if any malicious attempt is made. Encrypted tunnels are used for end to end encryption security [26]. IDPS has certain resources just to decrypt basic encrypted stream using XOR operations [6, 40]. It is important to note that too complex encryption makes IDPS unable to interpret data. The only way IDPS can interpret data is when encryption keys are shared with the system. Nonetheless, encryption and decryption operations are very resource exhaustive and can affect the performance of IDPS. An adversary can also use complex encryption algorithm to deprive IDPS of its resources and attempt possible DoS attack [6, 39, 40]

3.2.4 Time to Live Manipulation

In IP Protocol, Time-To-Live (TTL) is designed to prevent ever roaming packets over the internet. TTL is used as the number of hops that a packet can pass. The number of hops represents how many routers a packet can pass before reaching destination or being dropped [6, 19].

An attacker can exploit this value and can evade IDPS by manipulating hop counts or TTL of the packet. This can be achieved by either inserting segments or fragments with short TTL value into the stream. These short value segments will be part of the stream while being inspected by IDPS, although is expected to drop before the end of the system [6, 19]. The target system will get a malicious package as it gets different information than what was processed by IDPS. If the internal network information is known to the adversary, then it is easier to achieve success in the attack, as many hop count or TTL values are manipulated accordingly [6, 19]. The feasible solution for this technique could be that IDPS have a complete understanding of internal network topology and check TTL values in all packets. This information quite resources exhaustive; nonetheless, can protect the network from possible attack. [6, 40, 41]

3.2.5 Duplicate Insertion

Duplicate or overlapping segmentation occurs when the IP header offset value of the segment is similar to earlier segments. This new segment may overwrite already received segment or get completely ignored [19]. Duplicate segments can create confusion, hence can be utilized by malicious attackers. Different operating systems reassemble the segments of data is not very common ways and can handle the duplicate segments differently. If IDPS does not have information on network topology or end system details, then IDPS and victim might handle the duplicate segments inconsistently [6, 19].

As shown in figure 3.1, when duplicate segments used along with the TTL manipulation technique, attackers can leverage the ambiguity, which can evade IDPS detection. This can be done when an attacker intentionally fragments an attack and insert duplicate segments with different TTL values [26, 40]. Segments with smaller TTL values will never reach the end system. Meanwhile, IDPS would not know that the segment would not reach the end system without network topology information; however, would it will interpret all segments. This ambiguity can be resolved by configuring every system in the internal network about how to reassemble the segments and how to handle duplicate segments. Furthermore, configuring IDPS with internal network information includes complete network topology and configuration of all systems [19, 40].

3.2.6 Timing attack

Timing attack occurs when a source is sending packets very slowly without exceeding the threshold of the time window to correlate signatures [26]. This attack can be used to evade detection by IDPS. This evasion technique is effective against the detection system that uses a fixed time window to classify packets as one message and also loses the track of detection after the time window. This results in IDPS forwarding packets as non-malicious to the target, on the other hand, the target system all the packets together results in a malicious attack. [42]

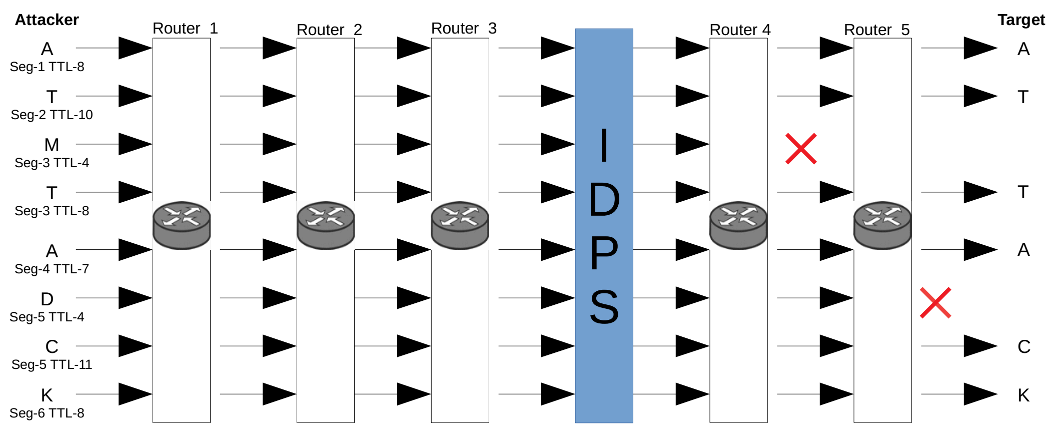

Figure 3.1: Attacking the target system with TTL Manipulation and Duplicate Insertion

This technique is used in many scanning tools like Nmap that delays the packets for scanning. It is capable of sending scanning pings so slowly and with such delay, most of IDPS might not consider its attack. Although, this is not a malicious attack against the target system, but can be made malicious after knowing the details of configuration [43].

3.2.7 Protocol violation

The concept of proper use of the protocol field is very understood, however, most protocols play a major role in defining how their fields will be utilized. IDPS needs to have a deep understanding of all the protocols [26, 33]. If it lacks understanding and semantic of protocol, then it becomes much more vulnerable to evasion, as IDPS would not be able to interpret the data. Some protocols are configured only on the end systems, and this kind of uncommon protocol produces inconsistencies between the system and IDPS. [6, 26, 33]

Even in case of extensively used protocols like TCP and IP, some fields are rarely or very inconsistently used and are called uncommon protocol fields [6, 26]. Due to the rare use of uncommon protocol fields, they are not implemented consistently on various devices. This inconsistency in implementation may result in a different interpretation of data streams, thus making them prone to be exploited in evading detection of IDPS systems [26, 35]. Many protocols are very complex in detail, and more complex the protocol, the more complex the processing is. An example of a complex protocol is Server Message Block (SMB), which is a transport layer protocol used for remote file access. In addition, SMB has been used in evasion by manipulating various fields of this protocol [26, 27, 41, 44].

3.2.8 Obfuscation

Obfuscation is the technique that transforms the content or data stream to completely different looking content, but the data still functions in the same way as to when it has not been transformed [6, 26]. This technique makes signature or fingerprint matching obsolete, as the transformed content will have totally different fingerprints. This technique can be used to transform any kind of content [26, 27].

IDPS needs to have the capability to somehow undo the obfuscation of data, or use a different approach to handle this evasion technique. An example of an application of obfuscation is to mutate program codes or shell-code [6, 26, 27].

Polymorphic techniques are used to generate dynamically changing signatures for attack instances in order to evade signature-based IDPS. Each time the code is executed, it mutates into a completely different code, hence results in a completely different signature [6, 45, 46]. Anomaly-based detection can detect abnormal content and attacks, even after the signature of the muted code has been changed. Furthermore, this makes it difficult to make the content normal for IDPS. An example of the advanced technique is the polymorphic bleeding attack, which has been performed to show the possibility of evading particularly the anomaly-based detection system [45, 46]. It is worthy of mention that the most used technique for shell-code mutation is polymorphic technique [6, 27].

3.3 Evasion Tools

The different types of evasion tools are enumerated in the following paragraphs.

1. **Fragroute** is a network packet fragmentation and firewall testing tool developed by Dug Song. It is the implementation of the attacks of “Insertion, evasion, and denial of service: Eluding network intrusion detection” paper of 1998. Fragroute can exploit TCP/IP protocols by implementing packet fragmentation and duplicate insertion to help evade detection by IDPS. It also has the capability to customize the attack by user arranged packets and automatically transform it into attack traffic to evade detection [6].

2. **Nikto** is a web scanning tool, that is capable of generating malicious URI requests and can be used to test web servers for potential security issues, such as programs and vulnerabilities. Nikto utilizes the support of LibWhisker to evade IDPS detection. Moreover, LibWhisker is part of Whisker, which is another automated web checking testing tool. It was written by Chris Sullo as proof of concept for HTTP specific evasion techniques. Whisker was one of the first web server testing tool built-in 1999 [47].

3. **ADMutate** was a shell-code mutation engine. ADMutate is an API designed to change the code structure and obfuscate the shell-code of attacks. Attacks remain effective even after being obfuscated. ADMutate used the polymorphic technique for the first time to generate different forms of signature of attack virtually evading any signature-based IDPS [48].

4. **Sploit** is another testing framework. It was developed from a sophisticated mutation framework published by Vigna et al. It allows tester and possibly attackers to develop advanced attacks and evasion techniques. Sploit was capable of using most techniques of Fragroute, Nikto, and ADMutate, as well as using different mu-

tation techniques at different layers. These different techniques are used to create variations of exploits capable of evading detection of IDPS [19, 49, 50].

5. **Metasploit by Rapid7** is the most popular tool used for penetration testing, exploit testing, searching security vulnerabilities and IDPS signature development. Its framework is emerging as an exploit development framework against target hosts. It also has variants of polymorphic shell-code encoders.

6. **Mendax** is a TCP de-synchronizer which is capable of injecting overlapping segments in a pseudo-random order. A modified version of Mendeax was used by Gorton and Champion to successfully evade IDPS [51].

7. **TWWWscan** was a vulnerability scanner that utilizes anti IDS url encoding. It was used to check up to 186 cgi vulnerabilities.

8. **Havij** is an automated evasion tool that can launch SQL injection attacks with attack strings. It can exploit SQL injection vulnerabilities of web pages while evading detection by mutating strings. Havij can manipulate white spaces with comment and encoding using hexadecimal and several other techniques.

9. **Forcepoint Evader** Contrary to general interpretation, forcepoint evader is not a penetration testing tool. It is however explained by Olli-Pekka Niemi and Antti Levomaki to be a deep inspection tool developed to test capabilities of IDPS [52]. Evader is the primary tool used in this thesis experiments.

Chapter 4

Environment for Experiment

In this chapter, technical environment established and used for the research experiment is explained. Details about the test environment including Hardware, Software, Device under Test (DUT) and Targets for the experiment are described in first section. In second section, network topology of experiment is described as how attacker and target machines are connected with DUT.

4.1 Environment Configuration

The environment for this experiment was to be kept secure from external network and especially from university network to avoid any chance of security compromise, and also to be configured according to certain criteria to evaluate results.

4.1.1 Hardware

The experiment was executed in a complete virtual environment. Oracle VM Virtual Box was used for this experiment. Due to nature of experiment, host system with ample resources was used to avoid any resource crunches.

System Configuration:

```
Operating System:  Ubuntu 18.04.3 LTS 64-bit

Virtual environment:  Oracle Virtual Box Ver 6.0.4 r128413

Processor:  Intel® Core™ i7-3770 CPU @ 3.4 GHz x 8

Memory:  32 GB RAM
```

4.1.2 Software

Forcepoint Evader

Forcepoint Evader is primary tool for this IDPS evasion experiment. Forcepoint Evader was publicly released with the name “Stonesoft Evader” in June 2012 in Black Hat security conference. Stonesoft is now Forcepoint so, name of the tool has been changed to “Forcepoint Evader”. This tool was testing tool and never claimed to be penetration tool. The main purpose of this tool is to prove that even advanced and modern IDPS can be evaded by exploiting techniques mentioned in a paper of 1998 [6].

Evader can be used as standalone command line application to send customized evasions or it can be used for fuzz testing against target by sending batch of evasions when used along with mongbat, which is another tool supplied in same package of evader. Interpretation of the results from evader is easy and self explanatory. Successful attacks using evasions results in stable shell of target or password or crashing target system. Logs of mongbat explains connection error or some other error that is encountered while attacking target system. Evader also provides option to obfuscated the payload of attack. Evader uses shell code encoder for obfuscation and it can create random shell code encoder to keep the obfuscations different for every attack. This feature has been used in this experiment, two cases of Obfuscated payload and two cases of payload without obfuscation has been used.

Current version of Evader from Forcepoint is not publicly available and was provided under strict conditions and only for this thesis research by Forcepoint. This version contained three exploits exploiting three well known vulnerabilities, CVE-2004-1315, CVE-2008-4250 and CVE-2012-0002. Evader contains 41 different atomic evasions to exploit these 3 vulnerabilities. Out of these 41 evasions, 29 atomic evasions can be used to exploit CVE-2008-4250, 30 evasions can be used to exploit CVE-2004-1315 and 18 evasions that can used to exploit CVE-2012-0002.

In this experiment, Conflicker and HTTP phpbb Highlight exploits are used. This selection of exploits is done on the bases to find evasions those works at multiple OSI layers. Conflicker exploits CVE-2008-4250 vulnerability, for this exploit evader can use 29 atomic evasions for IPv4, MSRPC, SMB, NETBIOS and TCP protocols to evade detection by DUT and attack target system. HTTP phpbb highlight exploits CVE-2004-1315 vulnerability. For this exploit, evader can use around 30 atomic evasions for IPv4, TCP, TLS and HTTP protocols to evade detection and attack target system. Conflicker exploit is used against Windows XP and HTTP phpbb highlight against Ubuntu (phpBB) targets.

Mongbat

Mongbat is a testing automation tool, this is impeccably part of this experiment. Mongbat comes as a part of Evader package. It is designed to randomize different evasion instances and evasion techniques and constantly create new combinations of evasions. Mongbat can run various parallel instances of Evader. Mongbat is capable of picking random IP address from the subnet mask and randomize ports to send command. This randomization is done to avoid any detection and blacklisting from IDPS. It can also be used to combine many evasions with specified parameters as individual attack. The primary use of the tool is to search the techniques that can exploit and also normalize traffic.

Mongbat can perform multiple attacks with chosen parameters, it control the traffic, sniff for any error while sending evasions commands and print the result of working evasion that have successfully evaded DUT and affected the target. It produces a report after every test, this report contains all the successful evasions. These evasions were later used further in this experiment to analyse the results.

4.1.3 Device under Test (DUT)

In this experiment, Snort (by Sourcefire now CISCO) was selected as only Device Under Test (DUT). The decision of testing only snort was based on the facts that Snort is most deployed IDPS worldwide. After first released in 1998 to till date, Snort is also considered as "the de-facto standard for IDPS". Details of Snort IDPS used:

```
Snort ver:  2.9.13 GRE

Rule Package:  snortrules-snapshot-29130.tar.gz
```

4.1.4 Selection of Target

Selection of targets was based on well integrated implementation of exploits by the tool used in this experiment. These vulnerabilities and exploits are well known and considering the time of release of these vulnerabilities, they shall be properly detected by any of the available IDPS.

PhpBB 2.x

In 2004, it was found that "viewtopic.php" was vulnerable to arbitrary code execution. In this vulnerability, viewtopic.php was improperly decoding URL to extract words and phrases. Remote attacker was able to insert special character into the result which was processed by PHP exec, this was possible by double encoding the highlight value. All the version of phpBB before 2.0.11 where vulnerable to it [53]. CVE-2004-1315 explains the details of the vulnerability and about its exploits.

In this experiment, Ubuntu Linux is used as target to test the evasions. Configuration of target machine:

```
Ubuntu 14.04.1 LTS

Apache HTTP Server version 2.0.64

MySQL 4.1.22

PHP 4.2.2

phpBB 2.0.10 (CVE-2004-1315)
```

Microsoft Windows RPC (Remote Code Execution)

Server Service was found to be vulnerable to arbitrary code execution by sending crafted RPC request packets. This vulnerability was found in Server service of Microsoft Windows 2000 SP4, XP SP2 and SP3, Server 2003 SP1 and SP2, Vista Gold and SP1, Server 2008, and 7 Pre-Beta. Server service of these operating system did not properly handled RPC request, this allowed remote attacker to exploit this vulnerability by sending crafted RPC request that can trigger overflow during path canonicalization and execute the exploit without any authentication [54]. CVE-2008-4250 explains details of this vulnerability and its exploits.

In this experiment Windows XP is used with following configuration:

```
Windows XP Professional Service Pack 2 (Unpatched)
```

4.2 Network Topology

The Network topology of this experiment is completely based on virtual environment using Virtual Machines on Oracle VM Virtual Box. Attacker, targets (Both Linux with phpBB and Windows XP SP2) and IDPS DUT are configured accordingly for this virtual environment. Attacker and target machines are connected with IDPS as DUT with local connection with promiscuous mode enabled. IDPS is configured as inline mode, hence all the traffic between attacker and target will be passing through it. Inline configuration is

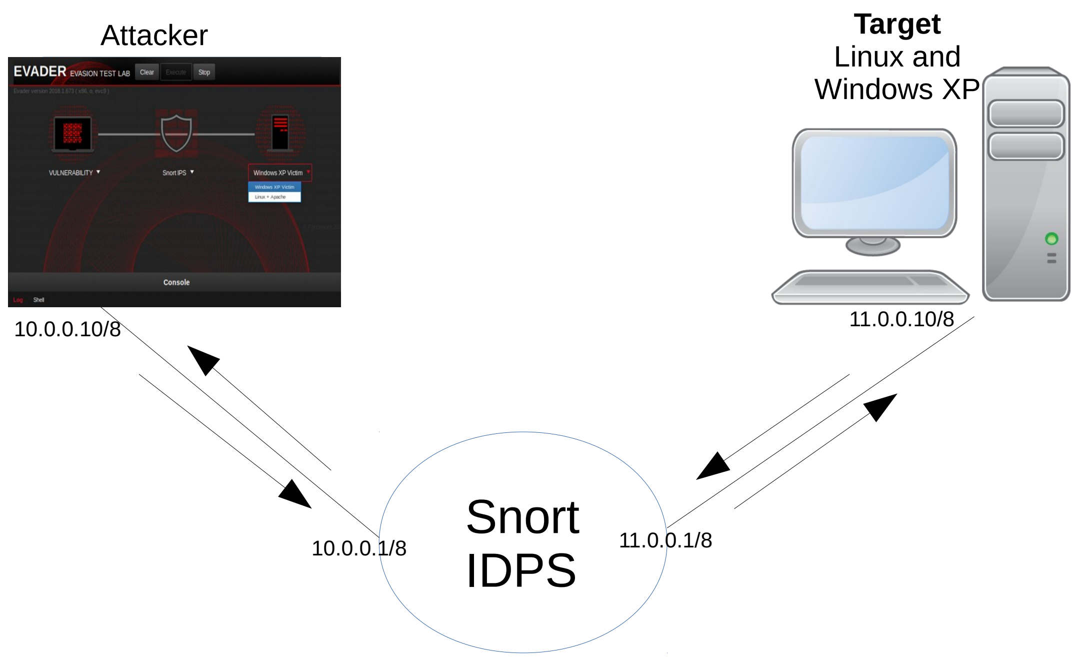

Figure 4.1: Network Topology used in Experiments

utmost important for this experiment as only then IDPS can decide to prevent any attack by dropping the packets from live traffic. Two Intel PRO/1000 MT Desktop (82540EM) network adapters were configured in snort IDPS to connect attacker and target machines and act as bridge for traffic.

Chapter 5

Methodologies for Experiment

5.1 Methodologies

5.1.1 Fuzzing

To recognize and find working AETs that can evade detection by IDPS DUT, a single test was done using both selected exploits. This experiment was done with huge number of attacks to gain significant amounts of advanced evasions.

To get broader information and significant number of successful evasions techniques, experiments using both of the exploits were performed separately. Both obfuscated payload and payload without obfuscation separately with each exploit experiment. Each exploit works on various protocols at different OSI layers. Due to working on many different protocols, it is possible to get better attacks sample against Windows XP and Ubuntu Linux target systems.

Attacks in first round of experiment were all randomized attacks, with combination of evasions created by Mongbat. This combination had evasions up to 5 different evasions in single attack. In total 41 evasions were used with different parameters, out of these 26 evasions were used against Linux (Ubuntu 14.04.1) and 30 evasions were used against Windows XP Service Pack 2. TCP and IPv4 protocol attack were common in both exploits but their implementation differed for each target system.

5.1.2 Segregation to Single evasions

Not all the evasions can evades the detection. In-fact very less evasions were able to evade detection of IDPS. These successful evasions were not single evasions but has many combinations of many evasions together. These combinations can be used in uncountable way by joining with each other endlessly. To analyze the successful evasion further we need to narrow down the scope of experiment, it was decided to divide these evasions to single evasions. This division to single evasions was attempted to recognize the single evasions those were able to evade the detection. These could be only evasions required in combination or could be part of necessary combination to evade the detection. If certain single evasion can evade detection by itself, then these single evasions is assumed to be the one that consistently evade detection of DUT.

After segregation of combinations, these single evasions were separated into 4 sets of evasions for each exploit and further separately for obfuscated payload and payload without obfuscation. These sets were later tested by creating special script to execute these single evasions using Evader, random IP addresses were again to avoid black listing. Results from this test provides another set of successful evasion which if capable of evading DUT once, is assumed to evade it every time, unless configuration of IDPS is changed and made more restricted.

5.1.3 Configuration change

In attempts to find possible solution to mitigate the problem, the configuration of DUT, Snort IDPS was changed. These changes were supposed to be using the options available in Snort and not any customization to normalization engine.

Details of the changes done on the Snort - DUT is provided in next chapter of implementation.

5.1.4 Multiple verification

The successful sets were verified multiple times before and after the change in configuration of IDPS. This was again in attempt to get error-less results as much possible. Multiple number of tests provided better result by removing any ambiguity and uncertainty from the outcome of the experiment.

5.1.5 Obfuscation

Quality of detection depends on how well IDPS can detect the attack. IDPS shall not depend on just simply matching of shell code using signature based detection. In reality, attackers seldom uses clear attacks and mostly uses various kind of techniques to hide the real attack to evade any detection. Attacker can use techniques like from encryption to NOP injection and various other techniques to masquerade real data behind attack. Even a decent IDPS in market ideally does not depend on only signature based detection, ideally they should be capable enough to detect which vulnerability is being exploited by the attack instead of just checking payload and matching signatures.

In this research experiment, we are obfuscating the payload that is used to attack the target machine. This is done to observe how well IDPS perform in detecting the attack when payload is obfuscated. We are using default obfuscation capability of Evader tool to obfuscate payload. Independent shell-code encoders are generated by Evader in its obfuscation techniques for each attack. This test is also important to recognize the evasions that can evade detection when payload is obfuscated and when it is without obfuscated payload. All the tests are done using both obfuscated payload and payload without obfuscation.

Chapter 6

Implementation

This Chapter explains different implementation phases of the experiment. All these tests were performed with Snort as DUT to highlight the vulnerabilities of network security devices. Even though many technologies are coming up everyday and systems are made more secure and more sensitive to any malicious intrusion attempt, nevertheless network security devices are yet vulnerable to attacks using the techniques revealed back in 1998. Proof of vulnerability to these old techniques was first presented by Stonesoft(Now Forcepoint) in 2012 that almost all IDPS were vulnerable to AETs. After more than 20 years of revelation of the techniques and 7 years after the proof of same, in 2019, world's most deployed IDPS is still vulnerable to these AETs and incapable in preventing from same attack techniques.

All the test cases and evasions involved in this experiment exploits well known and old vulnerabilities i.e CVE-2004-1315 and CVE-2008-4250 using Conflicker and Http phpbb highlight, which every IDPS available in the market is expected to recognize and prevent. These tests were performed against Windows and Linux target systems. Another reason of using these exploits was their default availability in Evader tool. Also underlying network stack that is used in these attacks are more complex and broader which is good for the experiment results of testing AETs, as multiple OSI layers can be used together to evade the detection.

Target of this experiment is more than just proving that AET can evade IDPS, but also which exact protocol based evasions are capable enough and most likely to do so. Moreover, attempt to mitigate same AETs. Tests for this experiment were performed systematically step by step to analyse the evasions for deeper understanding.

6.1 Experiment Phase 1

6.1.1 Setup and Execution

All the connections between machines were tested and checked before the experiment begins. Basic ping test and creation of bridge connection between target and attacker machine by inline DUT was configured and verified properly. To test if IDPS and target system are accepting and performing correctly, smaller tests with approximately 4000-5000 attacks were performed.

Evader was used to send individual attacks to test the connection and detection capability of DUT-Snort. Clean commands or commands without any malicious packets were sent to test the connectivity and response of target systems. Attacks with exploit towards both Ubuntu and Windows XP and also with and without obfuscated payload but without any evasions were tested. Attacks without evasions were performed to check the detection and prevention capability of Snort towards the well known exploits exploiting system vulnerabilities.

As the attacker, DUT and target machine all were configured and tested, next step was to perform the first phase of experiment. These tests were done in June 2019, with latest version of Snort available at that time. Target for first phase of this research experiment was to find as many working evasions as possible those were able to evade detection by IDPS.

Evader's web interface was used to execute this phase of experiment. Mongbat was used in back-end to send batch commands, randomize the evasions and their parameters.

Mongbat was responsible in creating randomized combinations of individual evasions to advanced evasions. 16 parallel threads were used, different random IP address were used for individual attack commands from the IP pool created for the experiment. Combination of minimum 1 and maximum 5 evasions was used to keep observations readable for next phase.

The number of attacks were not controlled but the execution time of experiment per operating system was fixed. On Windows XP and Ubuntu 14.04.1 LTS, the attack was executed with obfuscated payload and also without obfuscated payload. This was done to observe the effectiveness of detection by IDPS even when payload is obfuscated. `Conflicker` exploit was used to attack Windows XP SP2 and `http_phpbb_highlight` exploit was used to attack Ubuntu running vulnerable version of phpbb application. Sample of commands to attack these target systems:

Without Obfuscation

Mongbat command to attack Ubuntu 14.04.1 LTS Linux machine

```
ruby mongbat.rb -uid=webgui_8888
-attack=http_phpbb_highlight -if=enp0s8 -src_ip=12.0.0.10/8
-dst_ip=13.0.0.10 -gw=12.0.0.1 -mode=random -mask=8
-time=200000 -worker=16 -autoclose -payload=shell
-min_evasions=1 -max_evasions=5
-disable_payload_obfuscation -check_victim=true
-passthrough -verifydelay=1000
```

Monbat command to attack Windows XP SP2

```
ruby mongbat.rb -uid=webgui_8888 -attack=conflicker
-if=enp0s8 -src_ip=10.6.0.10/8 -dst_ip=11.0.0.10
-gw=10.0.0.1 -mode=random -mask=8 -time=200000 -worker=16
```

```
-autoclose -payload=calc -min_evasions=1 -max_evasions=5
-disable_payload_obfuscation -check_victim=true
-passthrough -verifydelay=1000
```

With Obfuscation

Mongbat command to attack Ubuntu 14.04.1 LTS Linux machine

```
ruby mongbat.rb -uid=webgui_8888
-attack=http_phpbb_highlight -if=enp0s8 -src_ip=12.0.0.10/8
-dst_ip=13.0.0.10 -gw=12.0.0.1 -mode=random -mask=8
-time=200000 -worker=16 -autoclose -payload=shell
-min_evasions=1 -max_evasions=5 -check_victim=true
-passthrough -verifydelay=1000
```

Monbat command to attack Windows XP SP2

```
ruby mongbat.rb -uid=webgui_8888 -attack=conflicker
-if=enp0s8 -src_ip=10.6.0.10/8 -dst_ip=11.0.0.10
-gw=10.0.0.1 -mode=random -mask=8 -time=200000 -worker=16
-autoclose -payload=calc -min_evasions=1 -max_evasions=5
-check_victim=true -passthrough -verifydelay=1000
```

6.1.2 Problems and Solutions during the experiment phase 1

While testing the Conflicker attacks against Windows XP SP2, one problem was found to be affecting whole experiment in a very critical manner. This same issue has been mentioned in research previously done on AET using conflicker. It was observed that after continuously attacking Windows XP target machine, it tends to stop accepting commands after a while. This behavior of operating system appears after many successful attacks. When an attack was successful, an instance of calculator application on target system was opened. When many of the instances were open, this problematic behavior was observed. This problem was to be resolved to continue the experiment.

The problem of not accepting further attack commands was persistent, hence a CMD script was written to kill any instance of calculator to keep system free and available for further attack commands. Nonetheless, the problem was yet reappearing again but this time only after significant amount of time. Another solution to resolve this issue was to schedule the operating system to reboot after a particular amount of time. So system was scheduled to reboot every 2 hours. Hence the attacks those were failed were not just due to performance of IDPS but also due to system rebooting. For the duration when system was rebooting, attacks failed with error "300: TCP connection failed"

6.2 Experiment Phase 2

After Phase 1 of experiment, we got many batches of the successful evasions. These evasions were logged in Mongbat log. Phase 1 of experiment had four tests with two case each against both target systems, hence 4 different log files were retrieved. We continued our experiment with those Mongbat logs of successful evasions. From this phase on, experiment phases were carried out using only Evader standalone application. Customized scripts were used to execute the attack commands in this phase of experiment and for analyzing the results.

Mongbat produced combination of evasions in previous phase using batch attack. Most of the successful evasions were combination of two to five evasions to evade detection by DUT. To analyze further, these successful evasions were segregated into the corresponding individual evasions. This segregation was done to test if all those single evasions can evade detection individually and which all gets detected and dropped by IDPS.

After segregating evasions into individual evasions, we got huge number of single evasions. These single evasions are subset of atomic evasions. 4 different lists of the individual evasions were created for both Windows XP and Ubuntu Linux target system. In each case of target systems, evasions were also separated accordingly if the payload used was with obfuscation or without obfuscation.

Experiment was continued by creating python script to create attack commands according to predefined syntax of evader and adding the single evasions. Separate scripts were written for different operating systems as the parameters like exploits, actions and others were different. It was also easier to execute this phase of the experiment with separate scripts to avoid any accidental or unwanted error. The script was made to log every evasion and the result, this was different from Mongbat log as here even failed evasions were logged.

One of the important point of this and further phases was, when customized scripts were used all the attacks were sent one by one and not multiple simultaneous attacks at same time. This attack with single thread was decided to test IDPS without any excessive load. In this condition, system shall work with its full efficiency as all the resources are free from any other processing.

Many evasions were same after being broken down into individual evasions and the script executed all evasions multiple times. Single evasions were executed multiple times to get better idea of which evasions are consistently evading detection by DUT and which evasions are inconsistent in doing same. The log from all 4 cases were processed and analyzed using another customized python script. This analysing script was capable of finding and separating successful and unsuccessful evasion from logs. It was also designed to create a statistical summary and description of successful and unsuccessful attempt.

6.3 Experiment Phase 3 - Attempt to Mitigate

The target of the experiment after detecting advanced evasions was to attempt to mitigate the issues. Mitigation was attempted while exploring available options in the IDPS DUT instead of creating whole new normalization engine or any other complex option.

First aim for this phase of experiment was to recognize most commonly exploited protocols by attacks. One of the solution to mitigate the issue of AETs was to make IDPS handle these protocols and check data more rigorously. This process of handling data rigorously can affect the overall performance of IDPS if not configured properly. In quit to find optimal mitigation solution, default options of Snort where explored.Many normalization preprocessor options where check and changed accordingly to handle AETs.

After optimizing and modifying the normalization options available in Snort, another test was done with all the successful individual evasions under the new configuration. Old scripts were used again to execute this phase of experiment. This test was performed multiple times to see the affect of changes on these individual evasions and their evading capabilities.

Tests of this phase of experiment included attacks against both Windows XP and Ubuntu Linux operating system. Also payload with obfuscation and without obfuscation were used against each target operating system. Nevertheless, this time top most unsuccessful evasions were also tested to confirm their failure again and evaluate any effect of changes done in Snort. Test were done multiple times to recheck results and outcome to get as better and error free results as possible.

6.3.1 Modification of snort to attempt mitigation

Snort includes normalization engine to drop the malicious data and is only used and installed if the snort is configured accordingly in the initial state of installation. This normalization is mostly utilized to normalize traffic by dropping unwanted traffic. Unwanted traffic can be extra data on header, flags or pointers depending upon requirements of user and hence configuration is selected by user. Snort package include **"README.normalize"** file in the bundle. This file explains all the options available in Snort to normalize the traffic. As DUT of this experiment only provides option to normalize TCP, IPv4, ICMP and IPv6 in this configuration file, and Evader exploits only IPv4 and TCP protocols directly out of all the options available. Due to this limited similarity of configuration, in this thesis research we will only change configuration related to these common protocol values.

Following are configuration change that was performed in DUT, Snort. We added these values in configuration file of Snort:

```
preprocessor normalize_ipv4:  trim
preprocessor normalize_tcp:  trim, urp
```

Change in configuration of Snort were supposed to target any extra data attached to the message in protocol header. **"trim"** in IPv4 remove any data attacked to the datagram and specially in IP header. The file mention this as "trim - truncate packets with excess payload to the datagram length specified in the IP header. The layer 2 header (e.g. ethernet), but don't truncate below minimum frame length."

Changed TCP options handled following:

- **`trim:`** `trims the data on SYN, from RST packet, to window and trim data to MSS`
- **`urp:`** `set the urgent pointer to the payload length if it is greater than the payload length.`

Chapter 7

Evaluation of results

In this chapter, we will evaluate the results obtained from the implementation of different phases of the experiment. Implementation of these phases are explained in Chapter 6.

In every phase of this experiment, four tests were performed on the selected target operating systems with well known vulnerabilities. Two tests were performed on each of the target operating systems, with direct payload and with obfuscated payload. The DUT was configured in the way to allow clean data and to drop the connection and packets containing malicious payload. All the experiments were performed with the latest version of DUT snort available at the time of commencing the experiment.

Interpretation of the results will be discussed along with observations from the experiments performed on the target system. Later limits and reliability of the experiment results are also discussed.

7.1 Successful combinations of evasions

Since test cases were performed using only one DUT, keeping the record of all the evasions used against the target systems was not necessary. According to the implementation explained in Section 6.1 of Chapter 6, only 2 exploits namely Conflicker and HTTP phpbb highlight were used foe this experiment. Conflicker exploits CVE-2008-4250 vul-

	Obfuscation	Attempted Attacks	Evaded	Evasion's Success rate
Windows XP SP2	Enabled	718093	45384	6.32%
	Disabled	922548	49586	5.37%
Ubuntu 14.04.1	Enabled	887839	40058	4.51%
	Disabled	1209301	54946	4.54%

Table 7.1: Successful evasions and their success rate

nerability of Windows XP SP2 whereas HTTP phpbb highlight exploits CVE-2004-1315 vulnerability of phpbb application running in Ubuntu 14.04.1 LTS operating system.

Mongbat along with Forcepoint evader were used in this first phase of experiment to produce and run different evasions. These evasions exploits different layer protocols. Single or variety of evasion combinations were used to attack the target system with malicious payload. Every test of this phase was performed for just 200000 seconds. Number of attacks varies according to the response of target systems or DUT systems. In this experiment the delay of IDPS was not taken into account. So overall delay of target system and DUT together were responsible for variation in number of attacks.

The set of evasions or combination of evasions created by mongbat and forcepoint evader to exploit protocols were similar when the payload was obfuscated or not obfuscated. Different evasions were created for different operating system depending on the protocols available to be exploited. Also mongbat did not created thousands of new evasions but they are same particular atomic evasions for particular protocols but with different parameters and their values. Even when payload is obfuscated, the evasions are same along with parameters those were used for payload without obfuscation. These variation in parameters makes the real difference if certain evasions or combination of evasions would be able to evade IDPS or not.

Table 7.1 shows the results from the first phase of experiment. Under "Evaded" in fourth column are the combination of evasions those successfully evaded DUT. These were successful in performing attack against the target system and retrieving the shell and opening an application on target system as proof of success. Mongbat was configured to use 16 threads to attack target machine simultaneously. The number of attacks attempted on target machines had big difference when obfuscation was enabled for attacks than when obfuscation was disabled.

In the case of target being Windows XP SP2, the number of attacks attempted without obfuscation were nearly more than 28% higher as compared to the test when obfuscation was used to masquerade the attack. Similarly in case of target being Ubuntu 14.04.1, the number of attacks attempted without obfuscation were more than 36% higher as compared to the test when payload was obfuscated. The reason behind the higher number of attacks in case of unobfuscated attack and lower number in case of obfuscated attacks were not checked in details. The reason might be due to more processing needed by IDPS to check for any malicious payload or due to greater use of fragmentation. Moreover, when fragmentation technique is combined with obfuscation technique, it exhaust processing resources of IDPS at higher rate.

The performance of IDPS also varied depending upon the target operating systems. Number of evasion combinations those were able to successfully evade detection were quite higher when the experiment was done on Windows XP SP2 operating system as compared to when similar experiment with Linux/Ubuntu 14.04.1 LTS was performed.

From third column of Table 7.1, we can observe the overall number of attempted attacks on Linux/Ubuntu 14.04.1 were much higher than when test was performed against Windows XP. The reason behind this possibly could be faster processing of packets by IDPS due to the absence of the issue that was interrupting in the case of Windows XP. This issues against windows XP was solved by rebooting the target but rebooting might have caused delay in the response.

One big difference in the results from cases between Ubuntu and windows XP targets is of evasion's success rate. From fifth column of Table 7.1, it can be noticed that success rate in test cases against ubuntu 14.04.1 much lower as compared to the similar cases but against windows XP. The difference in success rate is not only in test cases without obfuscation but also in test cases when obfuscation was enabled. The reason for this observation could be due to the fact that number of protocols and layers that could be exploited were higher in case of windows XP as compared to the case against Ubuntu. So the probability of new evasions combinations is much higher in case of windows target as compared to linux target.

7.2 Gaining successful single evasions

The results from first phase of experiment provided with enough evasions to test in this phase. These AETs were not in any particular order or any specific pattern. These were randomly selected evasions combined by mongbat for purpose of testing if they are able to evade the detection of IDPS.

To know if the success of these AETs depends on the precise combinations or just on individual evasion, we need to test all the unique single evasion found in the first experiment phase. If one particular evasion is able to evade the detection of IDPS then does the use of rest of the evasions in combinations becomes more of redundant? Unique advanced evasions discovered in last phase of experiment were formed from 26 or 30 atomic evasions with varying just the the parameters of each evasion. These variations might be quite small but still are capable of evading high end detection of IDPS. Changes of just the parameter of any particular atomic evasion can produce a new evasion, these parameters depends on the protocol being exploited by the evasion. Hence these variation themselves can create hundreds and thousands of different single evasions from one single variety or type of atomic evasion.

The AETs from previous experiment were in form of combinations of up-to 5 evasions in a single attack. Studying evasions with combination of each and every parameter is not

	Obfuscation	Successful Advanced Evasion	Single Evasions	Unique Evasions
Windows XP SP2	Enabled	45384	110748	37280
	Disabled	49586	121070	39150
Ubuntu14.04.1	Enabled	40058	97721	17039
	Disabled	54946	134541	21686

Table 7.2: Single evasions and Unique single evasions

only difficult but not feasible. Due to the fact that number of combinations can reach much more than just millions. Hence, testing only evasions with parameter chosen by Mongbat for the experiment in previous phase instead of trying evasions with new and every possible parameter.

It is assumed in this research that not all of the evasions are necessary in combination for any particular attack to be successful in evading IDPS. There are chances that only one evasion was enough to evade the detection and rest were more of redundant. It maybe possible that two or even all the evasions were necessary together.

Scope of this research experiment was kept limited to test single evasions from the AETs that we retrieved from first phase. This was decided due to fact that trying the combinations of two or more with all the evasions that got from experiment will go in many millions as the number of combinations are limitless. Due to the said fact, parameters of each evasion can be changed to make a new evasion and trying all the combinations with every single evasion is unrealistic.

Table 7.2 represents the number of these single evasions that was obtained from the first phase of experiment. Third column of the table represents the number of evasions those successfully evaded detection. Separate Python script was used to divide all the AETs to single evasions. Fourth column under "Single Evasion" represent the number of single evasions used to make combinations and evade IDPS. Some times some evasions are used to support other evasion as combination and they themselves does not evade any detection.

Out of all single evasions, fifth column represent the number of unique single evasions used in the experiment in each test cases. All duplicate evasions were removed. It can be noticed from the numbers that in case of Windows XP SP2 operating system, as protocols exploited by evader are higher and atomic evasions are higher. Therefore, number of unique evasions are found to be higher as compared to Ubuntu.

7.3 Testing single evasions

All the unique evasions from second phase of experiment were put on test using customized python script. This was done with target to observe which all evasions can actually evade detection even when those evasions are used alone.

Table 7.3 is representation of unique evasions those successfully evaded IDPS. Fifth column of table 7.3 lists the number of unique evasion those were successful individual evasions. These successful evasions were verified multiple times before attempting any mitigation solution to make sure that these were successful over multiple executions. We can observe that number of these evasions are much lower than overall single unique evasions used to perform this test. Furthermore, successful single evasions in case of Linux target system are much lower as compared to Windows XP target.

Lower number of successful unique evasions might be due to the fact that evasions targeting linux needs more complexity and also the fact that linux had limited exploitable protocols. Whereas, in case of windows target system, it had many more options of protocols to exploit and therefore number of single successful evasions are much higher. In case of linux target system, protocols exploited are limited to only TCP, IPv4 and HTTP. These are well known protocols and exploiting then needs higher level of manipulation of packets. In case of Windows target system, in addition to TCP and IPv4 protocols

	Obfuscation	Single Evasion	Unique Evasions	Successful single evasions out of unique evasions
Windows XP SP2	Enabled	110748	37280	945
	Disabled	121070	39150	790
Ubuntu14.04.1	Enabled	97721	17039	137
	Disabled	134541	21686	10

Table 7.3: Successful single unique evasions

MSRPC, Netbios and SMB adds complexity. This complexity is used by evader to produce more complex evasions against Windows XP target system.

7.3.1 Important observations

Most of the evasions those work against Windows XP using Conflicker are using segmentation. Segmentation is mostly successful against Windows only as the option to exploit other layers in same attack are more. In case of Linux the segmentation attack is not as successful as against Windows. Even the segmentation attacks which are successful against Linux are only fraction when compared to attacks against Windows. Most of the segmentation single attacks against Windows XP were same if payload is obfuscated or not. So segmentation clearly evades detection of malicious payload by DUT. Attacks where payload is sent over re transmitted sync packets were both successful and unsuccessful in both conditions when payload is obfuscated and not obfuscated. So, it is not certain that this technique of sending payload with Sync can be always successful.

DUT produced high amount of logs showing high level of segmentation being used and incapability of DUT to take any decision on them. These logs if processed using any machine learning algorithm or some data analysing tool might reveal presence of some malicious data. This solution is very important as most of the new IDPS solution uses these techniques. Still using these techniques in live traffic and detecting malicious traffic in real time is not easy but possible with much better results.

In case of attacks against Linux, most of the attacks were successful are only when payload is obfuscated. In attempts when obfuscation is not used, majority of cases where detected and dropped by DUT.

7.4 Mitigation Phase

Next target of the experiment after analysing the evasions was to mitigate the problem of evasions. We used default option available in DUT to make traffic filtering better by IDPS. No other modifications were done to default functionalities of DUT other than using it as inline Intrusion prevention system instead of just Intrusion detection system functionality.

To find mitigation solution, we needed to observe many characteristics from successful single evasions. Which protocols were exploited most frequently. How many successful evasions exploited those protocols, are there any options available in DUT by default to customize the settings accordingly to avoid or handle these attacks. Exploitation of protocols was being done by using basic and well known evasion techniques. These techniques includes techniques of adding data to sync, using extensively smaller fragmentation, Time to Live manipulation, and other techniques discussed already in above Chapters.

This attempt of mitigation of evasion was focused on the fact that the normalization of traffic is done by IDPS, overlooking certain parameters they are being exploited make it easy for evasion to evade IDPS.

We can not discuss the nitty-gritty of techniques that we found successfully evading detection. This is done as the target of this thesis and research of AETs is to show the vulnerabilities of world's most widely deployed or "de facto" IDPS with possibility of mitigating many of the issues. And not to make this as guide for intruders and also due to the restriction imposed by Forcepoint to use evader for this research related experiment purpose only. All the data presented is derived only from the experimental observations

	Obfuscation	Successful Single evasions out of found Evasions	After Mitigation attempt
Windows XP SP2	Enabled	945	918
	Disabled	790	754
Ubuntu14.04.1	Enabled	137	123
	Disabled	10	0

Table 7.4: Result after Mitigation attempt

and non of the data is modified or self designed.

There might be millions of more advanced evasion and combinations those might be successful if attempted and might even evade detection even after any countermeasures. Any assumed data and capability of AETs other than data from this thesis experiment and proved practically are just assumptions. The scope of these assumptions is so huge that it is impractical in-terms of its viability.

Most of the successful single evasions were found to be exploiting not just one protocol but at multiple OSI layers in same evasion while manipulating certain aspects of each protocol. Different parameter values of protocols were manipulated in attempted evasions. This manipulation of multiple protocols made it tricky for mitigation attempt to detect or prevent attack. Very small number of successful evasions were found to be exploiting just one protocol. It was also assumed that any of evasions that used single protocol to attempt attack can be mitigated by changing configuration settings of IDPS. AET which were exploiting one protocol using another protocol at same time in single evasion, any solution targeted to normalize single protocol might or might not mitigate the issue and hence will not necessarily stop every evasion. Still the mitigation attempt that was performed in this research found to be reducing the number of successful evasions in many cases and did not affected many others.

In test cases against Windows XP, 27 attacks with obfuscated payload and 36 without obfuscated payload were dropped. Here the most important thing observed is that with use of trim, any attack with payload on Sync packets were dropped. No warning or detection log was generated by DUT. This is important case as in normal scenario, no one shall use sync packets for data transfer unless with malicious intentions.

In test cases against Ubuntu Linux, 14 attacks with obfuscated payload and all the attacks without obfuscated payload were contained by DUT. This is a very important point as no attack was successful while using only single evasions. Even the attacks whose were successful with obfuscated payload, these attacks were mostly same atomic evasion using HTTP protocol with very less variation of parameter values. Most of the attacks were exploiting HTTP protocol based evasions were only successful with obfuscated payload. When same evasions were used as attack with unobfuscated payload, they failed to evade DUT. Good deobfuscation algorithm for normalization can easily counter these attacks.No attempt of mitigation against this protocol was performed in this thesis.

7.4.1 Further test result

To be sure of the results, just the case of attacks against Ubuntu Linux was studied with combination of attacks. It was observed that more than one or preferably even more than two evasions are needed to evade detection. When all the old evasions those were successful in phase one were test again with modified configuration of Snort, only `2329 evasions were successful out of 21686 unique evasions.`

This represent that even if IDPS can detect and drop malicious data using single technique of evasion. Still if combination of techniques are used then it is possible to evade the detection and attack the target even when mitigation methods are adopted.

Chapter 8

Conclusion

In this thesis, the effectiveness of advanced evasion techniques (AETs) based attacks in evading security on an updated IDPS was analysed, and based on the analysis, mitigation methods were proposed. It was observed that advanced and updated critical network security devices, such as IDPS are still quite vulnerable to threats based on evasion techniques that has been disclosed over two decades ago.

The results from the first phase of analysis showed that a low 4-6% of AETs were successful in evading the IDPS. Considering importance of security and complexity of networks, this percentage of successful evasions is very high. It is important to note that one evasion attack is enough to compromise the integrity of the whole network and make every system vulnerable.

In depth analysis from the second phase of analysis illustrated that IDPS are most vulnerable to segmentation/ fragmentation attacks. Furthermore, it was observed that evasion attacks proved to be more potent when protocols of various OSI layers were exploited together in the same evasion attack.

We proposed the methods to mitigate the problem by hardening the normalization parameters of our test device, which was Snort. Analysis results were tested multiple times during the second phase of experiment to assure reliability of results, and same was done after applying mitigation methods in third phase. The results obtained after adopting the mitigation methods highlighted that many evasions techniques exploiting some particular protocols were dropped completely; however, in other case evasion attacks for example, against linux were mitigated 100%. The adoption of mitigation methods removed many attack instances;however, it was found that the system was still vulnerable to segmentation/ fragmentation and obfuscation attacks. Details of the parameters of successful evasions are intentionally excluded due to security reasons.

The analysis of AETs was performed with target to recognize which particular evasion techniques are more efficient in evading updated network security devices and adopt mitigation methodologies based on the results of analysis to counter AETs. Any counter consequences of the mitigation methods were not tested further. The reliability of the mitigation methods were limited to particular evasions used in this experiment. Further research of the same set of advanced evasion techniques on commercial Intrusion detection and prevention systems can be performed to compare the performance of open-source verses commercial solutions.

References

[1] Statista. Global digital population as of October 2019. Retrieved on 2019-10-20 from https://www.statista.com/statistics/617136/digital-population-worldwide/.

[2] Symantec. Internet Security Threat Report. *Network Security*, 21, February 2019. Retrieved on 2019-10-20 from https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf.

[3] Statista. Number of connected devices worldwide from 2015 to 2025. 2019. Retrieved on 2019-10-20 from http://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[4] McAfee. Report: Economic Impact of Cybercrime – No Slowing Down. pages 1–28, February 2018. Retrieved on 2019-10-20 from https://www.mcafee.com/enterprise/en-us/assets/executive-summaries/es-economic-impact-cybercrime.pdf.

[5] A. Fuchsberger. Intrusion detection systems and intrusion prevention systems. In *Information Security Technical Report*, volume 10, pages 134–139. Elsevier J., 2005.

[6] T. Ptacek and T. Newsham. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Technical report, Secure Networks, Inc, Calgary, AB, Canada, January 1998. Retrieved on 2019-10-20 from https://apps.dtic.mil/docs/citations/ADA391565.

[7] R. Shirey. Internet security glossary, version 2. RFC 4949, August 2007.

[8] Committee on National Security Systems. Committee on National Security Systems (CNSS) Glossary. April 2015.

[9] K. Scarfone and P. Mell. Guide to intrusion detection and prevention systems (idps). *NIST Special Publication 800-94 IDS*, February 2007. Retrieved on 2019-10-20 from `https://csrc.nist.gov/publications/detail/sp/800-94/final`.

[10] J. P. Anderson. Computer Security Technology Planning Study. Technical report, James P. Anderson Co., Fort Washington, PA, October 1972.

[11] J. P. Anderson. Computer Security threat monitoring and surveillance. Technical report, James P. Anderson Co., Fort Washington, PA, April 1980.

[12] G. Bruneau. The History and Evolution of Intrusion Detection. Technical report, Sans Institute, October 2001. Retrieved on 2019-10-20 from `https://www.sans.org/reading-room/whitepapers/detection/paper/344`.

[13] R. Meyer. Challenges of Managing an Intrusion Detection System (IDS) in the Enterprise. Technical report, Sans Institute, March 2008. Retrieved on 2019-10-20 from `https://www.sans.org/reading-room/whitepapers/detection/paper/2128`.

[14] Gartner. Gartner Information Security Hype Cycle Declares Intrusion Detection Systems a Market Failure: Money Slated for Intrusion Detection Should Be Invested in Firewalls, June 2003. Retrieved on 2019-10-20 from `https://www.businesswire.com/news/home/20030611005056/en/Gartner-Information-Security-Hype-Cycle-Declares-Intrusion,2003`.

[15] B. Subba, S. Biswas, and S. Karmakar. False alarm reduction in signature-based ids: game theory approach. *Security and Communication Networks*, 9(18):4863–4881, November 2016.

[16] H. Adam, L. Craig, and N. Claudio. Magic Quadrant for Intrusion Detection and Prevention Systems, January 2017. Retrieved on 2019-10-20 from `https://www.gartner.com/doc/3571417/magic-quadrant-intrusion-detection-prevention`.

[17] D. Brumley, J. Newsome, D. Song, H. Wang, and S. Jha. Towards automatic generation of vulnerability-based signatures. *Proceedings - 2006 IEEE Symposium on Security and Privacy*, pages 2–16, 2006.

[18] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier. Shield: Vulnerability-driven network filters for preventing known vulnerability exploits. *Computer Communication Review*, 34(4):193–204, October 2004.

[19] T. H. Cheng, Y. D. Lin, Y. C. Lai, and P. C. Lin. Evasion techniques: Sneaking through your intrusion detection/prevention systems. *IEEE Communications Surveys and Tutorials*, 14(4):1011–1020, October 2011.

[20] N. M. Jacob and M. Y. Wanjala. A Review of Intrusion Detection Systems. *Global Journal of Computer Science and Technology*, 17:1–5, January 2018.

[21] V. Jyothsna, V. V. Rama Prasad, and K. Munivara Prasad. A Review of Anomaly based Intrusion Detection Systems. *International Journal of Computer Applications*, 28(7):26–35, August 2011.

[22] H. J. Liao, R. Lin, H. Chun, Y. C. Lin, and K. Y. Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, January 2013.

[23] Guide to Intrusion Detection and Prevention Systems (IDPS) (Draft). *Nist Special Publication*, 800-94:127, July 2012. Retrieved on 2019-10-20 from `https://csrc.nist.gov/publications/detail/sp/800-94/rev-1/draft`.

[24] M. Boltz, M. Jalava, J. Walsh, and I. Labs. New Methods and Combinatorics for Bypassing Intrusion Prevention Technologies. Technical report, Stonesoft Corporation, Helsinki, Finland, 2010.

[25] S. Gold. Advanced evasion techniques. *Network Security*, (1):16–19, January 2011.

[26] M. Särelä, T. Kyöstilä, T. Kiravuo, and J. Manner. Evaluating intrusion prevention systems with evasions. *International Journal of Communication Systems*, 30(16):1–15, June 2017.

[27] M. Dyrmose. Beating the IPS. Technical report, Sans Institute, March 2003. Retrieved on 2019-10-20 from `https://www.sans.org/reading-room/whitepapers/intrusion/paper/34137`.

[28] J. Postel. Internet Protocol. RFC 791, September 1981.

[29] S. Dharmapurikar and V. Paxson. Robust TCP Stream Reassembly in the Presence of Adversaries. In *Proceedings of the 14th Conference on USENIX Security Symposium - Volume 14*, pages 5–5. USENIX Association, August 2005.

[30] M. Ibrahim Salim and T. A. Razak. A study on IDS for preventing denial of service attack using outliers techniques. In *Proceedings of 2nd IEEE International Conference on Engineering and Technology, ICETECH 2016*, pages 768–775. IEEE, March 2016.

[31] A. Brindley. Denial of Service attacks and the emergence of Intrusion Prevention Systems. Technical report, Sans Institute, November 2002. Retrieved on 2019-10-20 from `https://www.sans.org/reading-room/whitepapers/firewalls/paper/818`.

[32] T. Kyöstilä. The effectiveness of evasion techniques against intrusion prevention systems. Master's thesis, Aalto University, Finland, May 2014.

[33] J. Marpaung, M. Sain, and H. Lee. Survey on malware evasion techniques: State of the art and challenges. *International Conference on Advanced Communication Technology, ICACT*, pages 744–749, February 2012.

[34] W. Buchanan, F. Flandrin, R. Macfarlane, and J. Graves. A methodology to evaluate rate-based intrusion prevention system against distributed denial-of-service (DDoS). In *Cyberforensics 2011*. University of Strathclyde, Glasgow, 2010.

[35] B. Hernacki, J. Bennett, and J. Hoagland. An overview of network evasion methods. *Information Security Technical Report*, 10(3):140–149, 2005.

[36] M. Alenezi and M. J. Reed. Methodologies for detecting DoS / DDoS attacks against network servers. *The Seventh International Conference on Systems and Networks Communications (ICSNC)*, pages 92–98, 2012.

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

[38] D. Bonderud. Leaked Mirai Malware Boosts IoT Insecurity Threat Level. Website: Retrieved on 2019-10-20 from `https://securityintelligence.com/news/leaked-mirai-malware-boosts-iot-insecurity-threat-level/`.

[39] S. Banescu and A. Pretschner. A Tutorial on Software Obfuscation. *Advances in Computers*, 108:283–353, 2018.

[40] P. Gibbs. Intrusion Detection Evasion Techniques and Case Studies. Technical report, Sans Institute, January 2017. Retrieved on 2019-10-20 from `https://www.sans.org/reading-room/whitepapers/detection/paper/37527`.

[41] V. Paxson. Bro: A System for Detecting Network Intruders in Real-time. In *Proceedings of the 7th USENIX Security Symposium*, volume 31, pages 2435–2463. Elsevier North-Holland, Inc., January 1999.

[42] D. Burns and O. Adesina . Network IPS Traffic Analysis Methods, Evasion Possibilities, and Anti-evasive Countermeasures. In *CCNP security IPS 642-627 official cert guide*. CISCO, July 2011.

[43] NMAP.ORG. Timing and Performance | Nmap Network Scanning, 2016. Retrieved on 2019-10-20 from `https://nmap.org/book/man-performance.html`.

[44] B. Caswell and H. D. Moore. Thermoptic camouflage: Total IDS evasion. *Black Hat USA*, pages 1–61, October 2006.

[45] P. Fogla and W. Lee. Evading Network Anomaly Detection Systems:Formal Reasoning and Practical Techniques. *Proceedings of the 13th ACM conference on Computer and communications security*, pages 59–68, 2006.

[46] P Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee. Polymorphic Blending Attacks. *15th USENIX Security Symposium*, pages 241–256, August 2006.

[47] C. Sullo. Nikto, 2001. Retrieved on 2019-09-20 from `https://cirt.net/Nikto2`.

[48] T. Detristan, T. Ulenspiegel, Y. Malcom, and M. Von Underduk. Polymorphic shellcode engine using spectrum analysis. In *Phrack Magazine Issue 0x3d, Phile 0x09*, 2003.

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

[50] D. Balzarotti. *Testing network intrusion detection systems*. PhD thesis, Politecnico di Milano, Italy, October 2006.

[51] A. Gorton and T. Champion. Combining evasion techniques to avoid network intrusion detection systems. Technical report, Skaion.inc, North Chelmsford, Massachusetts, USA, 2003.

[52] O. P. Niemi and A. Levomaki. Evading Deep Inspection for Fun and Shell. *Black Hat USA 2013*, pages 1–5, 2013.

[53] CVE-2004-1315. Available from MITRE, CVE-ID CVE-2004-1315, December 12 2004. Retrieved on 2019-09-20 from `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-1315`.

[54] CVE-2008-4250. Available from MITRE, CVE-ID CVE-2008-4250, September 25 2008. Retrieved on 2019-09-21 from `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4250`.