ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

Cécile Lüssi

Signature-based Extrusion Detection

Master Thesis MA-2008-06
March 2008 to August 2008

Tutor: David Schweikert
ETH-Tutor: Bernhard Tellenbach
Supervisor: Prof. Dr. Bernhard Plattner

Abstract

Today, the comprehensive protection of a company computer network is extremely important. Intrusion detection systems, an important building block of any sound defense infrastructure, usually do not play the role they should: Alerting the administrators in case of a severe security breach. The reason lies in the huge amount of false alarms, which makes the system not practicable. The big challenge is to differentiate between an attempted attack and a successful attack.

In this thesis, we investigate if and how the outbound traffic of an infected host can be used to detect intrusions. Based on my findings, we propose and implement a proof-of-concept extension to the open source intrusion detection system Snort. This extension establishes a layer which refines the escalated alerts using behaviour-based correlation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The protection of a company network is and has always been important. This thesis focuses on network-based intrusion detection systems (NIDS) which monitors all network traffic passing on the segment, where the sensor is installed and reacting to signature-based activity or suspicious anomaly [3].
Most companies heavily depend on a well-working and secure computing infrastructure. Any incident could be critical to their doing business. The increasing usage of interactive internet applications in the area of e-business and e-commerce induces a rise in risks and possibilities of misuse [7]. The following core objectives have to be met in order to protect the network [1, 2, 4]:

- *Confidentiality*
  Confidentiality means that secret information remains undisclosed.
  Possible countermeasure: encryption

- *Integrity*
  Integrity means that information is protected against unexpected modification.
  Possible countermeasure: Message Authentication Code (MAC)

- *Availability*
  Availability means that it is guaranteed that a resource is available when needed.
  Possible countermeasure: firewall blocking denial of service attacks

- *Authentication*
  Authentication means that the identity of a party is confirmed.
  Possible countermeasure: digital certificates

- *Non-repudiation*
  Non-repudiation means that the denial of integrity and authenticity of information is not possible.
  Possible countermeasure: digital signatures

In order to meet all of these requirements, it is essential to protect a system as good as possible against intruders. If, however, an intrusion occurs despite of defense, this incident should at least be detected: This is the task of intrusion detection systems (IDS).

1.1 The Different Types of IDS's

An IDS observes the network assets with the goal to detect misuse and anomalous behaviour. This concept has been known for almost 30 years. Beginning in 1980, with [11] the first concept of intrusion detection was born [12]. In the last 30 years the development of IDS's has followed different tracks. The different advantages and disadvantages of the various types resulted in the large choice amongst IDS's.

- **Network-based versus host-based IDS**
  A network-based IDS (NIDS) monitors the network traffic of a particular network. A host-based IDS (HIDS) monitors the operating system, applications, and the host specific network traffic. They reside, at least partially, on a host. But some IDS's are of a hybrid type and implement parts of both approaches. See Table 1.1 for advantages and disadvantages.
  This thesis is only about NIDS's. In the following chapter of this thesis the term IDS is used instead of NIDS.

Advantages	Disadvantages
network-based	
– no impact on the end system – invisible configuration (stealth mode) – detection of distributed attacks (e.g. SYN[1] flooding, denial of service (DoS))	– high requirements on computing performance to scan every packet – detection of attacks that manifest themselves in the network traffic – cannot be used if encrypted communication is allowed (unless a workaround like an SSL proxy is available)
host-based	
– monitors the actual reaction of the host – access to host-specific information e.g. integrity checker, process or system call monitoring – monitoring on all protocol layers – encryption is no hindrance (except on the application layer)	– installation on every single host – adaptation to the different platforms and operating systems – performance requirements on every supervised host – no detection of distributed attacks on multiple targets

Table 1.1: Network-based versus host-based IDS [7]

- **Intrusion prevention system (IPS) versus IDS**
  An IPS, also known as active IDS, investigates the traffic inline. This means that the packets are analysed continuously and the reaction to an attack is in real-time. The IPS blocks traffic independently without human interaction. It aims not only at detecting, but also at preventing an attack. An IPS can be seen as an extended firewall, which does not inspect the packet headers alone (e.g. IP's and ports), but also other properties such as protocol flow or the content of a packet. In contrast, a passive IDS does not act by itself but does only raise an alarm in case of a supposed attack. The handling of this alarm needs human interaction.
  See Table 1.2 for a list of advantages and disadvantages.

- **Signature-based versus behaviour-based IDS**
  A signature-based or so-called misuse detection system searches for known malicious patterns in the payload. A pure signature-based IDS uses only single events for the analysis. A behaviour-based IDS, also known as an anomaly detection system, analyses in the first instance the traffic data. The goal is to distinguish between normal and abnormal traffic examining the fundamental behaviour of a system. See Table 1.3 for advantages and disadvantages.
  In this thesis, we combine these two approaches. We implement a signature-based IDS using Snort and refine the escalated events using behaviour-based correlation.

[1] SYN is the first packet to be sent during the TCP handshake

Advantages	Disadvantages
IDS	
– false-positives raise only an alert and do not block the system	– intrusion results in a detection instead of a prevention
IPS	
– it does not only detect but also prevent an attack	– it is very error-prone and a false-positive has serious consequences (blocking of useful traffic, possibility of DoS attacks) – bad performance and reliability of the monitored network

Table 1.2: IPS versus IDS

Advantages	Disadvantages
signature-based	
– detects known attacks reliably	– detects only known attacks – needs a regular update of the rules – often no differentiation between an attack attempt and a successful attack
behaviour-based	
– possible to detect unknown attacks	– needs to be trained and tuned carefully, otherwise it tends to false-positives

Table 1.3: Signature-based versus behaviour-based IDS

- **IDS versus extrusion detection system (EDS)**
  Most of the current used IDS's focus on the intrusion from outside of the network into the monitored network. Such a detection of attacks creates a lot of false alarms. A new approach called extrusion detection is focusing on the traffic, whose source address is inside of the monitored network.
  See Table 1.4 for advantanges and disadvantages of intrusion and extrusion detection.

Advantages	Disadvantages
IDS	
– detection during the infection	– detection of unsuccessful attacks without a real intrusion
EDS	
– detection denotes a successful intrusion and not only an attempted attack	– not detectable before the host is infected

Table 1.4: Intrusion versus extrusion detection

The extrusion detection technique is a promising approach because the behaviour of an infected system and the generated traffic due to this infection is often conspicuous. Such an extrusion is a clear indication of a occurred intrusion, because an extrusion only happens as a result of a successful attack. The differentiation between an attempted and a successful attack is the ultimate goal of this thesis. Therefore, we will analyse and test the effectivity of the extrusion detection approach.

1.2 Pros and Cons of an IDS

The opinions about the benefit of an IDS differ. In the follow, we discuss some pros and some cons of operating an IDS.
An IDS is an important tool for the defense of a network against attacks. Since 1980 the sophistication of attacks has increased enormously. By contrast, the necessary intruder knowledge to create malware is decreasing, due to the fact that multiple tools on the internet are available, which support an easy creation of malware. So it is possible for almost anybody to create their own worm. Figure 1.1 shows this development of the increasing sophistication of attacks and the decreasing intruder knowledge very well. The result of these changes is a huge amount of sophisticated attacks, against which a network needs to be defended. This is only possible with a multilayer defense strategy containing firewalls, content filtering, vulnerability and virus scanners as well as IDS's.

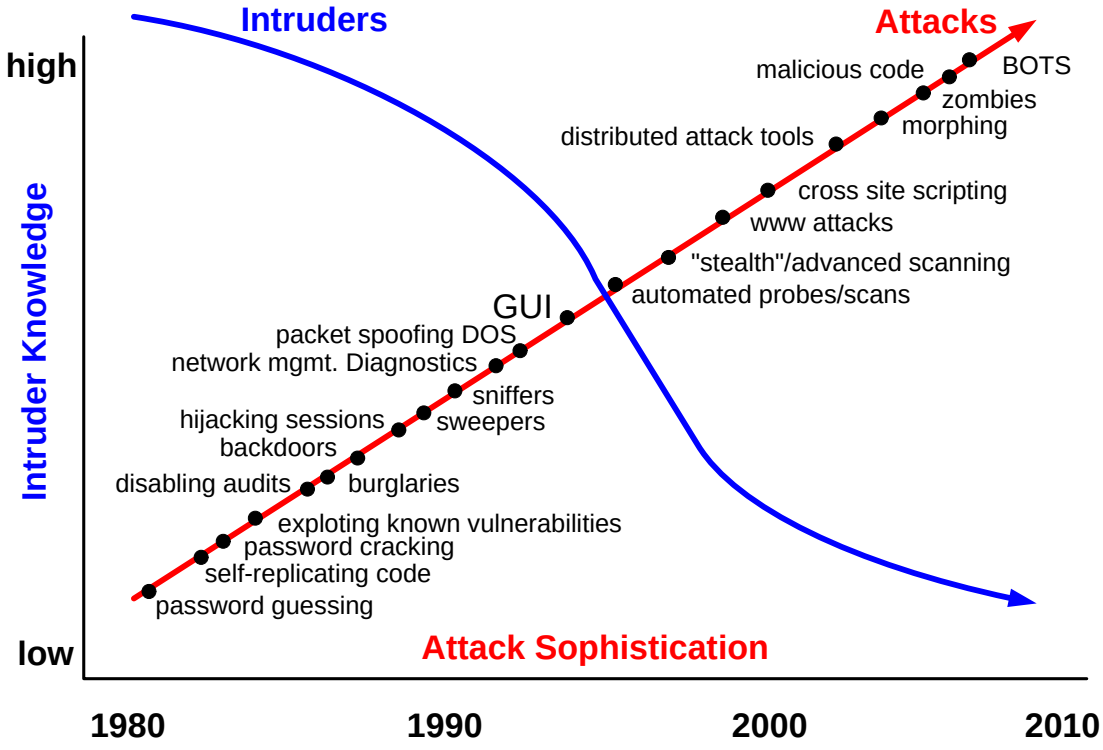


Figure 1.1: The increasing sophistication of attacks and the decreasing knowledge of the intruder [1, 62])

An other important reason to operate an IDS is that every company has to comply with legal requirements. Hence, there exists a need for action to achieve the IT-compliance. In case of an incident (e.g. a violation of the data protection act) the company has to prove that appropriate measures had been taken. If it is not possible to adduce evidence, the company has to pay a penalty. Maintaining an IDS is a significant argument in case of an accusation.

In reality, most large companies have an IDS, but only few of this systems are actually used.

There is definitely a need for an IDS, but multiple challenges have to be improved for a useful real-world solution. Some important and unsolved challenges of current IDS's are: [1]

1. False alarms and false-negatives:
   Figure 1.2 gives an overview of the different types of alerts and false alarms. *False alarms* can be categorized into two different groups namely false-positives and unsuccessful attacks. A *false-positive* is an alert which is not related to a real attack attempt. An attempted attack can result in a successful intrusion (the red area) and in no intrusion (the green area), in case of an *unsuccessful attack*. The goal is to escalate successful attacks since only in this case human intervention is necessary. An attack without an intrusion is mainly used for statistical reasons or forensic analysis. Therefore, an unsuccessful, attempted attack is valued as a false alarm. An IDS with a lot of false-positives and unsuccessful attacks will lose the interest of the system administrator and a true alert might not be taken seriously. A *false-negative* is a successful attack, which was not detected. In this case an attack compromises a network without beeing noticed.

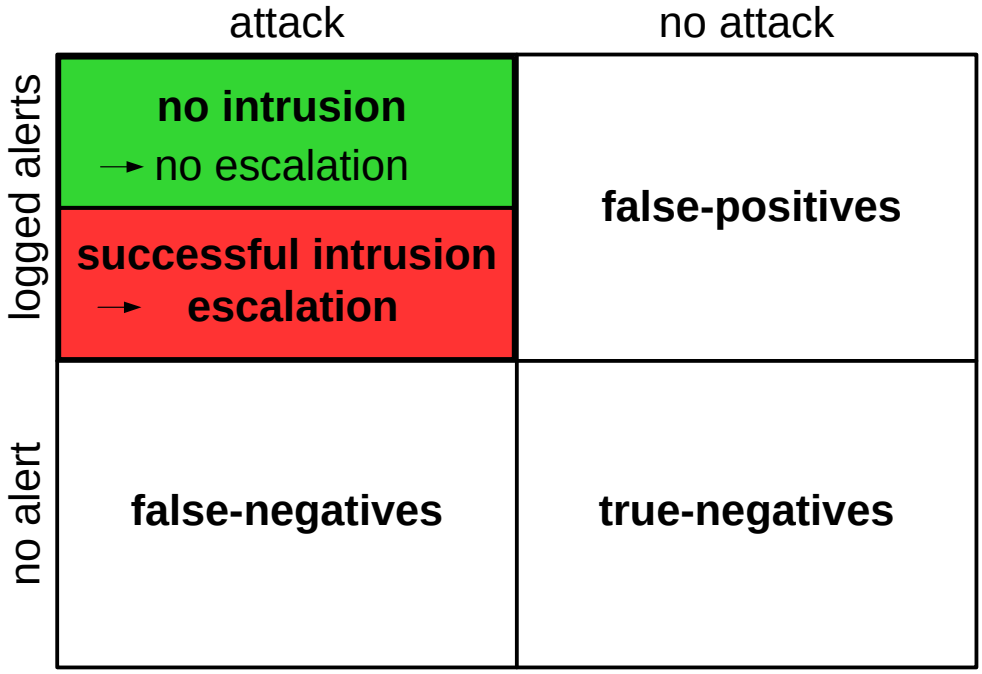


Figure 1.2: Overview of alerts and false alerts

2. Amount of data:
   The data of a single host might be insufficient to properly identify an intrusion. An ideal network-based IDS would correlate data from as many hosts as possible to detect attacks. That is a large amount of data. It is essential for an intelligent and logic system to correlate the data in a good manner and to extract the relevant information. Intrusion detection is a needle-in-the-haystack type of problem, where the normal activity data is the haystack and the anomaly is the needle, which has to be found [1].

3. Encryption:
   Encryption is more and more used for reasons of privacy. Encryption has an impact on the effectivity of content filtering and virus scanning systems. A content inspection is only possible with unencrypted data. The consequences for the IDS depend on the type of the encryption. Basically, one can choose among the following approaches of encryption:

   (a) Transmission of the encrypted data over a common, insecure channel (e.g. PGP encryption of emails).

(b) Transmission of unencrypted data over a secured, encrypted channel (e.g. IPsec on the network layer).

Data encryption (approach a) affects a network-based IDS marginally. Only the content of the data is uncontrollable. If the communication channel is encrypted (approach b), the disturbance depends on the protocol layer of the encryption. Therefore, packets over an SSL-encrypted channel still contain unencrypted information about the transport layer and all layers below, and an attack might be detected only by analysing that information. Furthermore, it is possible to install an SSL-proxy, which operates as a man-in-the-middle. This SSL-proxy will decrypt the traffic for the analysis of the IDS and then encrypt the traffic again.
Hence, an attack on the IP layer (e.g. an IP spoofing attack) is still detectable for the IDS. Every encrypted message is surrounded by a plaintext package. Analysis of the data flow can also indicate an anomaly. Examples are an abnormal rush of traffic or a communication between unauthorized systems [7].

4. Security of an IDS:
Security tools have to be secure themselves; otherwise, they are just another point of attack. An IDS must be designed in such a way that it cannot be compromised with traffic on the networks that it monitors. This goal is hard to achieve with today's software and hardware architecture, but this aspect is important for designing a protection system [1].

1.3 Problem Statement

From the afore mentioned problem fields, the problem of false alarms appears to be the most pressing one. Therefore, this study aims at differentiating between an attack without an intrusion and an attack with a successful intrusion (see Figure 1.2). There are different approaches in order to be able to distinguish between a successful and an unsuccessful intrusion:

- Smart IDS approach:
A smart IDS contains a software component, which leverages knowledge such as e.g. the network architecture or the installed software packages of each participant along with their version or patch level. The alerts of the IDS will be filtered based on this additional knowledge. As the complete system state is known, it can be deduced for which attacks the host is vulnerable. The challenge using a smart IDS is certainly to get information of the system state. Following two possibilities are used to acquire knowledge:

  – External mechanism such as network scanning:
  A network scanning tool scans the monitored system and collects knowledge about the system state. The open source project Qualys IDS Correlation Daemon (QUIDScor) implements such a tool for correlating IDS events with vulnerabilities detected by QualysGuard. QUIDScor eliminates a high percentage of false alerts by filtering for absent vulnerabilities and inactive services [9]. A weak point of this approach is that the reliability of this technique depends on the accuracy of the scan. This is conflicting with the security principle "Promote privacy." This principle means that services e.g. have to avoid to give information about themselves, which helps the attacker to figure out how to break through [8, 10]. This does not imply "security by obscurity." The security of the system is not based on the obscurity property. But there is no reason to give unnecessarily information away.

  – Internal mechanism such as a host agent:
  This approach deploys agents, which allow analysing the whole system. These agents run on the monitored host and list all installed software with version and patch level. This information is part of the security information management tool and filters the alert without a successful intrusion. The problem of this approach is the security of the IDS itself (see section 4). The agent has to communicate the system state to the IDS. This implies that the IDS must be addressable. But the advantage of a network-based IDS with stealth mode configuration is that it has no IP, therefore, it is

invisible [7] within the monitored network. Attention should also be paid to the transmission of the system state from the protectable system to the IDS. This connection has to be confidential and authenticated.

- Correlation:
  Correlation is the logic of evaluating sensors of different types (e.g. host-based and network-based) or asynchronous sensors. Hence the goal would be to correlate different attack traces of multiple sensors and to yield a reliable result concerning intrusion. The difficulty with this approach is to handle the amount of data (see Chapter 1.2 Challenge 2) and to deliver an appropriate logic respectively. Correlation can be used together with a signature-based or a behaviour-based detection system. In this thesis, correlation is used in combination with a signature-based IDS. Another possibility of correlation would be to correlate logged events from different systems and sources such as syslog, firewall, IDS or netflow data.

- Analysis of the outbound traffic:
  The analysis of the outbound traffic could be useful to detect successful attacks, because a host will react to an infection in some way. This reaction is detectable through monitoring the outbound as well as the intra network traffic of a network and can be taken as a serious indication of a successful attack. The Figure 1.3 visualises the three terms, which will be used in the further chapters:

  – Inbound: Traffic from outside into the monitored network.
  – Outbound: Traffic from the monitored network outwards.
  – Intra: Traffic between two hosts within the monitored network.

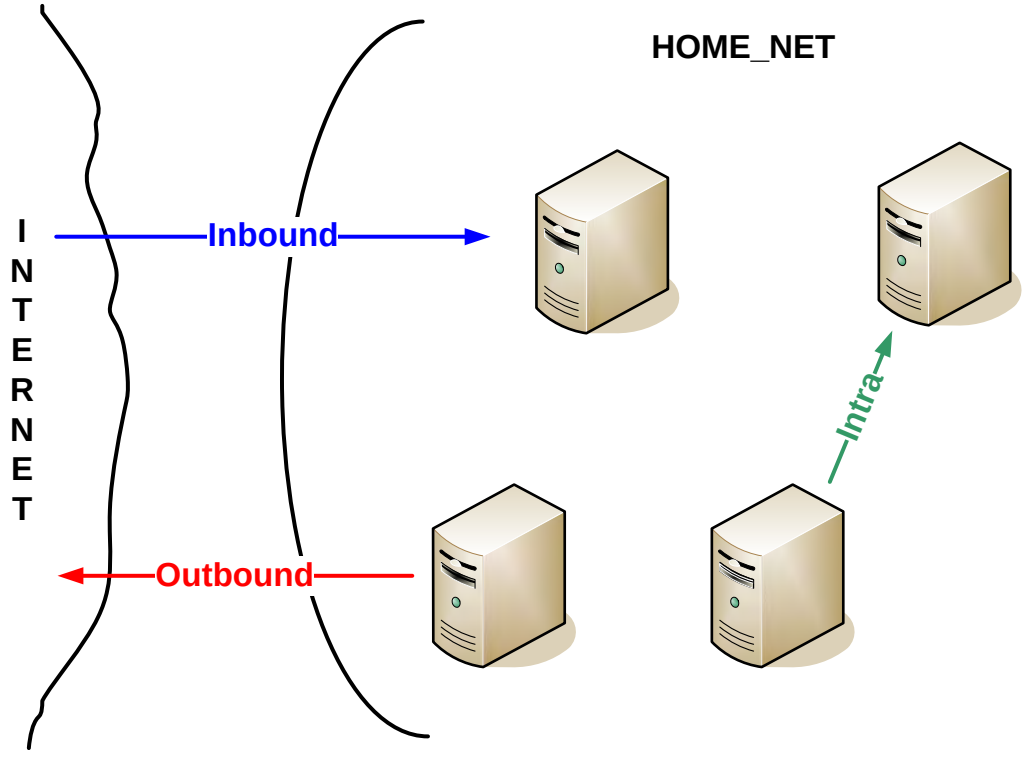


Figure 1.3: Differentiation of inbound, outbound and intra traffic

To sum up, the goal of this thesis is to work out a network-based IDS to detect successful attacks rather than unsuccessful ones. To reach this goal, we will use the approach of correlation and focus on the outbound and intra traffic.

1.4 The Task

The task of this master thesis (C) is to analyse the practical viability of an outbound traffic based intrusion detection (also known as “extrusion detection”) system. Particular attention is paid to the alarms which are caused by an attack without a following, successful intrusion. This is the case if the attacked system is not vulnerable to this particular attack, for example if the attacked system is already patched or does not use the exploited application.
More specifically, the following points are of interest and should be analysed:

- Analysis of the outbound traffic of five different popular worms. The purpose is to figure out if a detection of the worm is possible using only the outbound traffic.
- Behaviour-based systems: Is signature-based extrusion detection an alternative to behaviour-based extrusion detection (flow analysis) or something that could complement it? Does it make sense to use both and if not, which one is the most effective?
- False-negatives: How reliable is a system that only looks at outgoing traffic? How many of the most popular attacks can be detected without analysing the inbound traffic? Which kind of attacks can be detected and which kind cannot?
- False-positives: Is such a system really better when it comes to not triggering failed intrusion attempts? When do false-positives still occur?
- Signatures: Is it possible to implement such a system with currently available signatures (Sourcefire, Bleeding Edge Snort, ...)? What is the overhead of managing signatures on your own?
- A proof-of-concept prototype implementation should be built and tested using the Snort engine. Real live traffic should be analysed with it, or, if that is not practicable, network traces should be used for testing the prototype.

1.5 Related Work

A relevant study for this thesis is the BotHunter paper [40]. BotHunter is a passive network monitoring system driven by Snort. It correlates the inbound intrusion alarms with the outbound communication patterns that are highly indicative of successful local host infection. The experimental results using BotHunter are presented in a virtual and live testing environment. BotHunter focuses on botnet detection and its traffic. In this thesis, we will choose a broader approach and examine different classes of malware. We focus on all outbound traffic generated by a malicious source.
A further relevant topic for this thesis is the reduction of false-positives and the detection of successful attacks. The diploma thesis on “Smart Intrusion Detection Systems” [34] designed and implemented a concept, which reduced the number of false-positives of an IDS. This implementation investigates alerts generated by a conventional IDS and finds out if a raised alarm is justified. In particular the implementation is able to determine if an attacked host is vulnerable against the observed attack or not. The severity level extension, which makes it possible to combine different alerts is very interesting. The smart IDS approach needs additional knowledge about the monitored network. The gathering of such knowledge is sophisticated and hard to implement in a real world system. Hence, for this thesis we are looking for other solutions without information about the monitored systems.
Another important topic is the outbound traffic. The diploma thesis [35] about scan detection concentrates on a behaviour-based analysis on the outbound traffic. This means the scan behaviour of an infected host is analysed in detail. By contrast, in this thesis the signature-based approach is used as well.
Paper [36] focuses on the outbound traffic with the intention to guarantee that the host will not be used as an attack launcher or intrusion relayer to compromise other systems. Therefore, the intended goal using the outbound traffic are different for the mentioned paper and the present work. Paper [36] focuses on the prevention of further propagation of malware. In this thesis, on the other hand, the outbound traffic is analysed to get a clear indication about a successful attack.

In addition, there are some interesting works about correlation. The paper [41] about alert verification differentiates between asset correlation and alert correlation. For this thesis, alert correlation is relevant. The work about alert correlation [42] describes different types of it. One implemented possibility is to escalate particular alerts, only when a certain sequence is given. This interesting feature is accounted for in this thesis.

1.6 Overview

Chapter 2 presents some general definitions and a short passage on Snort signatures. Chapter 3 is about extrusion detection and presents a classification of malicious outbound traffic. Using this classification and the defined methodology, five different worms have been analysed. Chapter 4 describes the architecture and correlation techniques of the prototype implementation. Chapter 5 compares a conventional IDS with an EDS and evaluates the implemented prototype. Chapter 6 contains a summary of the key contributions of this thesis, the conclusion and the outlook.

Chapter 2

Malware and Snort Signatures

2.1 Terms and Definitions

Malware is a relevant topic with respect to IDS. In the last few years, the topic of malware has become increasingly popular. Therefore, multiple distinct definitions exist of the different terms. In this part, we will take a closer look at malware and present a clear definition, which we will apply to in the following chapters of this thesis. Certainly, a lot of malware applies to multiple categories. A lot of new malware is able to propagate and to infect new hosts in different ways. These properties make the detection clearly harder.

Malware is a newly created compound word and evolves from “malicious” and “software”. Malware denotes all software which attempts to intrude in a system without the consent of its user. Hence, malware is a generic term for worms, computer viruses, trojan horses, backdoors, rootkits, spywares, dishonest adwares et al. Malware can be categorised by different criteria. A differentiation by the propagation behaviour distinguishes between the following three categories:

- Computer worm:
  A computer worm is a self-replicating program. A worm does not need a host program. In contrast to a virus, it is an autonomous entity. A worm exploits software vulnerabilities to intrude a system [13].

- Computer virus:
  A computer virus is a self-replicating program. A virus needs a host program for the propagation. Hence, the virus copies itself and infects existing programs. If one of these infected programs is copied to an other system in some way (e.g. file sharing, P2P or by downloading an attachment from an email), the new system is infected too and so on [13].

- Trojan horse:
  A trojan has no self-replicating ability. Normally, a trojan is part of a supposedly useful program and the user downloads the program without knowledge of the hidden and unintentional functionality. A trojan is often used to install a backdoor.

Another possibility is to differentiate by the intention:

- Backdoor:
  A backdoor in a computer system is a possibility to avoid the normal authentication mechanism which protects the computer against unauthorized remote access [16].

- Rootkit:
  A rootkit is a collection of software tools to hide an intrusion into a system. In particular, this means that processes, files and future logins of the intruder are hidden and the installed malware is protected from the antivirus detection [15].

- Spyware:
  Spyware sends personal data without the knowledge or the consent of the computer user to the software producer or a third party [17].

- Adware:
  Adware is software which automatically downloads banner ads and adverts pop-ups. Adware is a borderline case between malware and normal software and is also known as greyware [18].

- Botnet:
  A botnet is a group of software bots. A bot (derived from robot) is an infected host controlled by a master. Using this bots, a botnet master is able to contact, control and command the compromised hosts (zombies) of a botnet. A botnet is able to provide different services to the operator such as e.g. [14]:

  – Gain access to locally stored data (e.g. theft of personal data like credit card numbers)
  – Proxying of network traffic to hide the original source of an attack
  – Sending of large amounts of emails known as spam
  – Execution of distributed denial of service attacks (DDoS)

As already noted, malware consists of several components e.g. a backdoor trojan for expanding a botnet. In this case, the trojan horse installs the backdoor and the bot master is able to connect to the infected host through the backdoor. Another example is spyware in combination with a rootkit. The rootkit protects the spyware from being detected and the spyware collects the data calmly.

2.2 Snort Signatures

Snort with its signatures plays an important role in this thesis. This section presents an overview of the characteristics of Snort signatures, as well as the possibilities and limitations of using them.
One of the main problems using a signature-based IDS is certainly the maintenance of the signature database. The signatures have to be updated very regularly and the generation of new signatures for the detection of new attacks has to be efficient and, if possible, real-time. This is particularly important for the handling of zero-day attacks [60]. As soon as a new signature is available, the database should be updated, otherwise, the system becomes needlessly vulnerable. Therefore, the update of the ruleset has to be done automatically. This automatic update makes specific adjustments to the particular environment impossible. The updated ruleset overwrites the old one.
It is not recommended for a single enterprise to generate the signatures on their own and to hold their own signature data base as the effort for the maintenance and the generation of new signatures is enormous.
Snort.org maintains a big signature database. The Vulnerability Research Team (VRT) is constantly creating new certified rules for this database. These new rules are made available to all subscribers in real-time. All registered users get the signatures 30 days later. It is free of charge to be a registered user, but as a subscriber with a business, annual costs of around $500 (for one to five sensors) have to be paid [54]. Another source for signature updates beside Snort is Emerging Threats. Emerging Threats is a collection point for a number of open source security projects. The primary project is the contribution and maintenance of the only open Snort Signature Ruleset. The rules are free to use by any Snort user [59].

Here we present the basic structure of a Snort rule:

```
alert any_protocol HOME_NET any_src_port -> EXTERNAL_NET any_dst_port
```

Explanation of the Snort rule:
Raise an alert for **any_protocol** e.g. TCP or UDP.
The source IP address belongs to the defined address space of the **HOME_NET** on the source port **any_src_port**.
The destination IP belongs to the defined address space of the **EXTERNAL_NET** on the destination port **any_dst_port**.

The definiton of the HOME_NET and the EXTERNAL_NET has to be set according to the networks where the sensor is located.

Generally, there is the conflict of deciding between a specific rule, which is precise, and a generic rule, which is more general. Both decisions have their pros and cons (see Table 2.1). The goal is to combine the advantages of the two approaches.

Advantages	Disadvantages
generic	
• fewer signatures as a result of the aggregation • might also detect new versions of a metamorphic worm	• a lot of false-positives
specific	
• few false-positives	• large amount of signatures • a little change in the attack results in concealment

Table 2.1: Generic versus specific signatures

Every network packet can be divided in two parts: the payload and the header information (e.g. protocol, ports, source and destination address). This distinction is also made for the generation of rules. There are keywords for payload detections and non-payload detection rules. To get an idea of the available options, see the Tables 2.3 and 2.2.

Keyword	**Description**
fragoffset	The fragoffset keyword allows to compare the IP fragment offset field against a decimal value.
ttl	The ttl keyword is used to ckeck the IP time-to-live value.
tos	The tos keyword is used to check the IP TOS field for a specific value.
id	The id keyword is used to check if a specific IP option is present.
fragbits	The fragbits keyword is used to check if fragmentation and reserved bits are set in the IP header.
dsize	The dsize keyword is used to test the packet payload size.
flags	The flags keyword allows rules to track states across transport protocol sessions.
flow	The flow keyword allows rules to apply only to certain directions of the traffic flow.
flowbits	The flowbits keyword allows rules to track states across transport protocol sessions.
seq	The seq keyword is used to check for a specific TCP sequence number.
ack	The ack keyword is used to check for a specific TCP acknowledge number.
window	The window keyword is used to check for a specific TCP window size.
itype	The itype keyword is used to check for a specific ICMP type value.
ip_proto	The ip_proto keyword allows rules checks against the IP protocol header.
sameip	The ip_proto keyword allows rules to check if the source ip is the same as the destination IP.

Table 2.2: Non-payload Detection Rule Options (source: [54])

Keyword	Description
content	The content keyword allows the user to set rules that search for specific content in the packet payload and trigger response based on that data.
nocase	The nocase keyword allows the rule writer to specify that the Snort should look for the specific pattern, ignoring case.
rawbytes	The rawbytes keyword allows rules to look at the raw packet data, ignoring any decoding that was done by preprocessors.
depth	The depth keyword allows the rule writer to specify how far into a packet Snort should search for the specified pattern.
offset	The offset keyword allows the rule writer to specify where to start searching for a pattern within a packet.
distance	The distance keyword allows the rule writer to specify how far into a packet Snort should ignore before starting to search for the specified pattern.
within	The within keyword is a content modifier that makes sure that at most N bytes are between pattern matches using the Content.
pcre	The pcre keyword allows rules to be written using perl compatible regular expressions.
byte_test	Test a byte field against a specific value (with operator).
ftpbounce	The ftpbounce keyword detects FTP bounce attacks.
regex	The regex keyword has been superceded by PCRE.
content-list	The content-list keyword is broken and should not be used.

Table 2.3: Non-payload Detection Rule Options (source: [54])

A rule typically combines the two variants, the payload-based and the non-payload-based rule options. An analysis of the non-payload traffic is always possible because this information is available in unencrypted form. New malware often use encrypted communication. Encoded traffic alone is no hindrance for a payload-based detection since the content patterns are still recognizable. Whenever a real encryption is used and not just an encoding, a decryption algorithm is necessary; otherwise, an analysis of the payload is not possible. But there is still a lot of unencrypted malicious traffic in the wild.

Chapter 3

Extrusion Detection

The mission of an IDS is comparable with a smoke detector in a building. The normal case is that there is no fire because there are many precautions: The building material is fire-proof, it is not allowed to smoke, or ashtrays are available. The concept in a network using an IDS is the same. There are firewalls, security awareness of all users, physical safety measures, and other arrangements to secure a company network. The IDS is only used in case of an incident, if in spite of all measurements an intrusion occurs. Only in such a case an escalation of an IDS is needed and human intervention is necessary to prevent further fraudulent use and to recover the system. To perform this task an IDS constantly observes the traffic. The bigger the network is, the better the performance and the capacity of the IDS management system has to be. For big networks the traffic amount is huge. A statistical survey [53] about the university network (ETH Zürich) for the year 2005, shows an average of nine million alerts per day and, today, this value is most likely considerably bigger. The greatest part of this incoming traffic is uninteresting. It is a fact that the amount of alerts generated by outbound traffic is much smaller. And it is a goal of this thesis to show that an intrusion is also detectable by focusing on the extrusion traffic.

3.1 A Classification of Malicious Outbound Traffic

This chapter contains a classification of the malicious traffic leaving a host. This traffic is always a consequence of an infection and, therefore, a reliable indicator of a successful attack. The different classes of traffic are presented in a chronological order with respect to the appearance after an infection. Figure 3.1 shows the chronological chain of malicious outbound traffic. Changes between download, command and control (C&C), propagation and commanded traffic are possible. These traffic classes typically occur repeatedly and in various orders. The following sections present the different steps in more detail.

3.1.1 Attack Response Traffic

Attack response traffic is the immediate response to an attack. It might be an error message from the infected host. An example of attack response traffic is the following traffic captured by a Snort rule which detects if an attacker performs the shell command “id” and gets the response “uid=0(root).” This action is a verification of the attacker to check if the intrusion was successful. This is the entire rule:

```
alert ip any any -> any any msg:"ATTACK-RESPONSES id check returned root";
content:"uid=0|28|root|29|"; classtype:bad-unknown; sid:498; rev:6; tag:session,5,
    packets;)
```

3.1.2 Malware Download or Command and Control (C&C) Traffic

A lot of malware needs further downloads after an attack to complete the intrusion. A botnet normally uses the C&C channel for these downloads and later updates. The download traffic

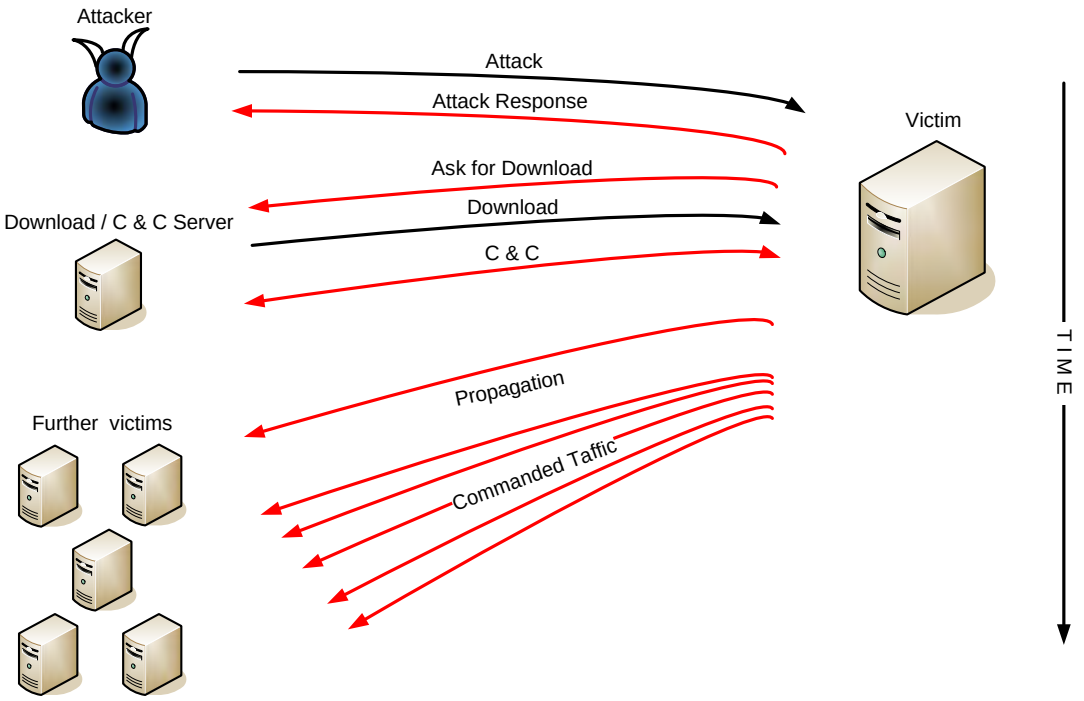


Figure 3.1: Overview of the different presented traffic classifications

might be detectable in general. C&C traffic is the communication channel between an attacker and its victim. A common example is the communication between a bot master and its bots. This communication tends to be hidden and encrypted. Some years ago, the Internet Relay Chat (IRC) protocol was quite popular, but today more sophisticated techniques are used. Here is a summary of the known techniques:

- Push style: (see Figure 3.2)
  Push style means that the request contains the command of the C&C server to the bot. A popular example is the Internet Relay Chat (IRC) protocol. The original purpose of IRC was group communication in discussion forums (one-to-many), but a one-to-one communication is also possible [37]. IRC is not a very common protocol in enterprise environment and, therefore, blocking all IRC traffic to prevent an IRC bot infection is often a valid option.

- Pull style: (see Figure 3.3)
  A C&C communication channel implements the pull style if the bot asks the C&C Server about the current command periodically (e.g. every five seconds) and gets the command as the protocol response. The most popular example is using the Hypertext Transfer Protocol (HTTP) communication channel. HTTP is suitable for a C&C channel, because the protocol is very popular,it is usually not feasible to block all HTTP traffic. If the communication is not encrypted, it might be possible to detect the C&C traffic by analysing the payload.

- Peer-to-peer (P2P):
  In a peer-to-peer network, there is no centralized coordination point, and every node acts simultaneously as client and as server. Therefore, there is no single point of failure and it is harder than in a IRC botnet to disrupt the botnet [38].

The C&C channel is also used to complete the infection after a penetration. There is more and more sophisticated malware and especially botnets, which update themselves regularly to hide from malware detection tools and adapt to the current environment. This metamorphic property

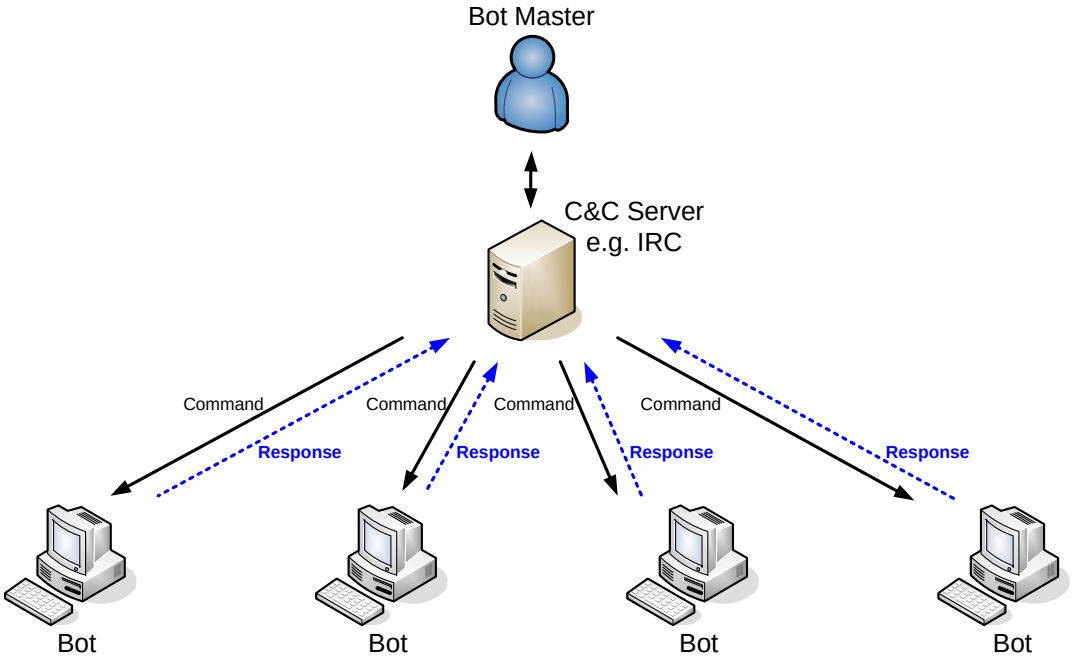


Figure 3.2: The process of a C&C protocol using the push style

makes the detection a lot more difficult.
Another advanced problem for the detection of C&C channels is a covert channel. «A covert channel is a parasitic communication channel that draws bandwith from another channel in order to transmit information without the authorization of knowledge of the latter channel's designer, owner, or operator» [63]. Covert channels exist based on the TCP and the UDP layer. Examples for TCP-based protocols allowing a covert channel are HTTP, HTTPS and MSN. An example for a UDP-based protocol is DNS.

The detection of covert channels goes beyond the scope of this master thesis. Basically, the detection needs a behaviour-based approach since signature-based detection is not possible. This is because the used channels are covert and nobody knows of their existence. Therefore, no signatures are available. Still, the change in traffic behaviour is eventually detectable. Detection are possible methods, which compare the current traffic and detect patterns like a very long trial for a communication channel followed by very short request and response packets. Another possibility for a detection is the analysis of protocol anomalies. But this approach assumes that a covert channel needs illegitimate traffic; otherwise, the communication channel is not detectable [39].

3.1.3 Propagation Traffic

One of the main missions of a virus or worm is to replicate itself and infect further hosts. There are different types of propagations, but always the same two generic steps:

1. determination of further victims
2. actual attack of these victims

Here are the different propagation types used for worm propagation:

- Network scanning:
  The infected host scans the network for attackable IP addresses and will attack all found IP addresses exploiting a vulnerability.

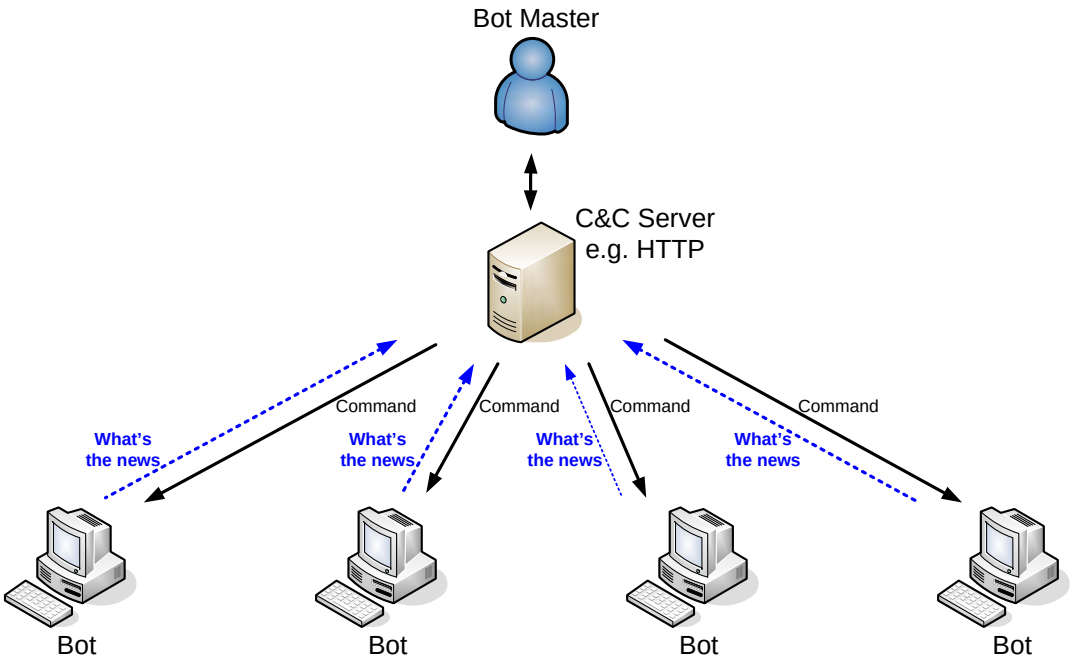


Figure 3.3: The process of a C&C protocol using the pull style

- Email address scanning:
  The infected host scans the hard drive for contacts like email addresses and uses its own SMTP engine to send emails to the found email addresses containing malicious attachments or links to infected web sites.
- Instant Messenger (IM) contacts scanning:
  The infected host uses the online contact lists and connects to the target machines via the IM server.
- P2P Network:
  A worm using an existing P2P overlay network to spread out.
- Websites using cross site scripting:
  If a client makes a request to an infected server, it gets back a malicious response. This response initiates a malicious request from the client to a server, the new victim. This request infects the server with the worm executing a cross site scripting attack on the server side.

Scanning activity on the network generates abnormal traffic and is detectable by a behaviour-based IDS. There are many different network scanning mechanisms such as random scanning, localized scanning and so on. For more details see [35]. The scanning activity for new contacts is not detectable with an IDS because it functions inside a host and does not generate network traffic.

3.1.4 Commanded Traffic

Multiple reasons motivate hackers to create malware. Besides reasons like need for recognition or adventure, financial reasons are supposed to be the most important ones nowadays. The commanded traffic often generates this financial profit. There are three main categories of this profitable, commanded traffic:

- Spam: The infected host is used to send spam.

- DoS: The infected host is used to participate in a (maybe distributed) denial of service attack
- Sniffer: The infected host sends information to its master. This information can consist of passwords, data of a keylogger, surfing behaviour data, credit card information and so on.

3.2 Analysis of Malware

In the following, we study several recent malware samples and analyse them with regard to the outgoing traffic they produce. We base our analysis on related work on the following five malware samples:

- A) Beagle
- B) Netsky
- C) Mytob
- D) The Storm Worm
- E) Kraken

One of the difficulties created by malware is the huge number of different types and versions. Therefore, it was a difficult choice out of many different botnets, viruses and worms. Crucial for the decision was, on the one hand, the novelty and, on the other hand, the current pervasiveness of the particular malware. We choose the botnets Storm and Kraken because they are quite advanced in the techniques they use and show the trend of malware. We choose Beagle, Netsky and Mytob because all these worms are durable and still very active. A statistic of a very active mail server of a client of Open Systems supports this assumption (see Table 3.1).

Table 3.1: Statistic of an active mail server measured during 23 days from July 18th, 2008

virus name	absolute occurences since 23 days	percentage occurences
Troj/Agent-HFU	479	41.5 %
W32/Netsky-P	174	15.1 %
W32/MyDoom-N	89	7.7 %
W32/Mytob-AK	70	6.1 %
W32/Beagle-QW	61	5.3 %

The statistic of Kaspersky Lab “Virus Top 20 for April 2008” shows a very similar picture (see Figure 3.4).

3.2.1 Methodology

The goal is to analyse the usability of the outbound traffic of a network-based IDS for the detection of successful attacks. We use the following methodology for the analysis and apply it to the five malware samples.

Table 3.2: Methodology to analyse the detection of malicious outbound traffic

	behaviour-based	signature-based	correlation of multiple events	
			outbound & intra	all traffic
Attack Response				
Download				
Command and Control				
Propagation				
Commanded				

Position	Change in position	Name	Proactive Detection Flag	Percentage
1.	0	Email-Worm.Win32.NetSky.q	Trojan.generic	40.58
2.	+1	Email-Worm.Win32.NetSky.d	Trojan.generic	8.18
3.	+6	Email-Worm.Win32.NetSky.y	Trojan.generic	7.62
4.	+3	Email-Worm.Win32.Bagle.gt	Trojan.generic	6.64
5.	+1	Email-Worm.Win32.Scano.gen	Trojan.generic	6.47
6.	+2	Email-Worm.Win32.NetSky.aa	Trojan.generic	5.81
7.	New!	Trojan-Downloader.Win32.Agent.ica	downloader	3.08
8.	-5	Email-Worm.Win32.Nyxem.e	Trojan.generic	3.01
9.	New!	Net-Worm.Win32.Mytob.x	Worm.P2P.generic	2.94
10.	New!	Net-Worm.Win32.Mytob.r	Worm.P2P.generic	2.68
11.	-1	Email-Worm.Win32.Bagle.gen	Trojan.generic	1.73
12.	+3	Email-Worm.Win32.Scano.bn	Trojan.generic	1.19
13.	-2	Email-Worm.Win32.Mydoom.l	Worm.P2P.generic	1.07
14.	New!	Net-Worm.Win32.Mytob.bk	Worm.P2P.generic	0.91
15.	-13	Email-Worm.Win32.Mydoom.m	Trojan.generic	0.89
16.	+1	Email-Worm.Win32.NetSky.c	Trojan.generic	0.70
17.	Return	Net-Worm.Win32.Mytob.c	Trojan.generic	0.69
18.	0	Email-Worm.Win32.NetSky.t	Trojan.generic	0.62
19.	New!	Email-Worm.Win32.Bagle.dx	Trojan.generic	0.47
20.	New!	Email-Worm.Win32.NetSky.ac	Trojan.generic	0.47
Other Malicious Programs				4.06

Figure 3.4: Statistic from Kasperky Lab: Virus Top 20 for April 2008

The different classes of the outbound traffic discussed in the previous section.
In the horizontal line, the Table 3.2 distinguishes four categories.

- behaviour-based:
  This means that the particular outbound traffic, e.g. the propagation traffic, is detectable using a behaviour-based IDS. That is the case if the current traffic is abnormal and therefore distinguishable from the normal appearance. A pure behaviour-based system uses no payload for the analysis; it uses only the behaviour of the traffic and the packet headers.
- signature-based (single event):
  This means that the outbound traffic is detectable based on a single signature. A signature is generated using the packet headers as well as the payload of a packet.
- correlation of multiple events (hybrid):
  This means that the detection system uses multiple events and correlates these events using a time-dependent and behaviour-based strategy.
  - outbound and intra traffic:
    This means that for the detection it is sufficient to correlate outbound and intra traffic events. Therefore, the traffic amount which has to be scannend is much smaller.
  - all traffic:
    This means that for the detection we need a powerful IDS, which correlates inbound, outbound and intra traffic events. It is important to note that the attacks detectable using only outbound traffic are a subset of all attacks, which are detectable using in- and outbound traffic.

3.2.2 Five Recent Worms

A) Beagle

The first variant of the Beagle worm also known as Bagle, appeared in January 2004. But, despite its age, this mass-mailing worm is still active and passes anti-virus engines. This is pos-

sible because of its polymorphic technique. Symantec security response noticed 18 different variants of Beagle during the first six month (January until July 2004) of infection.
The similarities of the different versions are the following: Beagle is a mass-mailing worm and contains its own SMTP engine to send emails to spread out. The infection arrives normally as a malicious attachment, but some versions use a malicious HTML email to exploit the "Microsoft Internet Explorer object type validation vulnerability" (Common Vulnerability Number CVE: CAN-2003-0532). A few versions also propagate by copying themselves to particular shared folders. The email address for the propagation collects the worm by scanning the infected host and the sender address is spoofed.
Most versions of the worm install a backdoor. Furthermore, the worm executes several actions such as changing registry keys, or terminating particular processes (e.g. antivirus software related processes).
Beagle contains an interesting feature, namely the fact that it stops functioning and removes itself, if the date is past a particular point in time. Beagle is not a stealthy worm. We think most of the versions can be easily detected after a worm analysis. The difficulty of Beagle is its polymorphic property, if one version is analysed, there is already a new version circulating. Sophisticated versions of Beagle additionally contain rootkit functionality such as lowering security settings [31].
Unfortunately, there is no information available about the most active version W32/Bagle.QW. We choose W32/Beagle.B [30] reported on the Symantec security response website for the detailed analysis of the outbound traffic:

- **Attack Response**: There is no detectable attack response traffic known.
- **Download**: If an attacker sends a particularly formatted data message to the backdoor port, the worm allows to download an arbitrary file to the folder %Windir%.
- **C&C Communication**: Beagle.B opens a backdoor on TCP port 8866. Furthermore, Beagle.B sends HTTP GET requests every 10'000 seconds to one of the following Web sites on TCP port 80: "www.strato.de/1.php," "www.strato.de/2.php," "www.47df.de/wbboard/1.php" and "www.intern.games-ring.de/2.php." This GET request notifies the connected Web server of the IP, an ID number and port number where the infected host is listening.
- **Propagation Traffic**: Beagle.B propagates itself using its own SMTP engine, searches for email addresses on the compromised host and sends emails, containing partly random generated messages, to the found addresses.
- **Commanded Traffic**: There is a backdoor on port 8866 and, therefore, the attacker has several possibilities, but there is no particular commanded traffic known, which could be detected.

Table 3.3: Analysis of the Beagle Worm

	behaviour-based	signature-based	hybrid	
			outbound & intra	all traffic
Attack Response				
Download	√	√	√	√
Command and Control	√	√	√	√
Propagation	√		√	√
Commanded				

An example signature from Emerging Threats [52]:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg: "ET WORM Beagle User
    Agent Detected"; flow: to_server,established; dsize: < 150; content: User-Agent
    \: beagle_beagle; nocase; classtype: trojan-activity; reference: url,
    securityresponse.symantec.com/avcenter/venc/data/w32.beagle.i@mm.html; sid
    :2001269; rev:12;)
```

This signature detects the User Agent of Beagle, which always contains the string “User-Agent\: beagle_beagle”. In general, the difficulty to detect Beagle is its high polymorphic technique. The most versions of Beagle need their own signatures to be detected.

B) Netsky

The first variant of the Netsky worm appeared on February 16, 2004 [48]. Netsky is a vintage mass-mailing worm and affects most Windows Systems (e.g. Windows 2000, XP, NT).
Netsky creates a mutex to ensure that only one instance of the worm is allowed to execute in memory. Then, it makes some changes so that it runs when Windows is restarted and modifies or deletes some registry key values. The next step is to gather email addresses from files with defined file extensions (e.g. .php, .txt, .msg). Furthermore, Netsky searches the drives C through to Z for shared folders and copies itself to these folders using file names like “angels.pif” or “porno.scr”.
After these steps, Netsky uses its own SMTP engine and sends itself to the found email addresses. The email contents are various depending on the version, e.g. the mail simulates a successful auction notification and the attachment suggests more information about the auctioned product. But in reality, opening this attachment results in a Netsky worm infection [48]. Only the newer versions of Netsky install a backdoor on TCP port 6789. There was no backdoor in the 18 first versions. For the detail analysis of the outbound traffic we choose the version Netsky.S:

- **Attack Response**: There is no detectable attack response traffic known.
- **Download**: Netsky.S listen on port 6789. An attacker can send an executable file to an infected host. The worm will save this file as <RandomFilename>.exe and then will execute that file.
- **C&C Communication**: Netsky.S installs a backdoor on the TCP port 6789 but there is no particular commanded traffic known.
- **Propagation Traffic**: Netsky uses emails for its propagation. The content of the emails varies and, therefore, it is difficult to define a reliable signature. But there are some signatures for the different Netsky versions. A good indication for a Netsky infection is definitely the increasing email traffic since Netsky sends an email to every found email address. Therefore, the detection correlating multiple events might be better tuned.
- **Commanded Traffic**: Netsky.S will try to perform a DoS attack against one of the Web sites “www.cracks.am,” “www.emule.de,” “www.kazaa.com,” “freemule.net,” and “www.keygen.us,” but only if the system date is between April 14, 2004 to April 23, 2004.

Table 3.4: Analysis of Netsky

	behaviour-based	signature-based	hybrid	
			outbound & intra	all traffic
Attack Response				
Download	√	√	√	√
Command and Control				
Propagation	√		√	√
Commanded	√	√	√	√

An example signature from Emerging Threats [51]:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 25 (msg: "BLEEDING-EDGE Virus Netsky.P Worm
     detected"; flow: established; content: "
    AAAAAAYAAAAA4fug4AtAnNIbgBTM0hV2luZG93cyBQcm9ncmFtDQokUEUA"; nocase; classtype:
    misc-activity; reference:url, vil.nai.com/vil/content/v_101119.htm; sid:
    2001566; rev:10;)
```

This signature detects emails sent from Netsky using its own SMTP engine.

C) Mytob

W32.Mytob@mm is a mass-mailing worm and first appeared on February 26, 2006. Mytob spreads in several ways. It can exploit several vulnerabilities (e.g the LSASS Windows vulnerability [61] or the DCOM RPC Vulnerability [64]), or it spreads, using its own SMTP engine, MSN or Windows Messenger. Furthermore, it copies itself to writeable network shares that are protected with weak passwords.
Mytob has several functionalities on a compromised computer. It can copy itself to other files and can change some Windows registry keys for establishing a C&C IRC channel to receive commands from the attacker. Moreover, there are functions to install a rootkit hiding the worm process and prevent access to certain security-related web sites [27].

- **Attack Response**: There is no detectable attack response traffic known.
- **Download**: Mytob does not need an additional download.
- **C&C Communication**: The command and control traffic uses the IRC protocol. This is quite conspicious and detectable because this protocol is normally rarely used in enterprise environments.
- **Propagation Traffic**: Mytob uses for its propagation emails or the LSASS vulnerabillity. The content of the emails is miscellaneous and, therefore, it is difficult to define a reliable signature. But there exist some signatures for the different Mytob versions. A good indication for a Mytob infection is definitely the increasing email traffic because Mytob sends an email to every found email address. Therefore, the detection correlating multiple events might be promising. The propagation via the LSASS vulnerability is detectable by a signatre. In this case, Mytob sends requests to TCP port 445 of selected IP addresses. If the remote computer responds, Mytob will exploit the vulnerability and launch itself on this new victim machine [65].
- **Commanded Traffic**: A Mytob infected host can be used as a spam bot. The increase in sending emails is detectable. Furthermore, it may perform a denial of service attacks on specific servers.

Table 3.5: Analysis of Mytob

	behaviour-based	signature-based	hybrid	
			outbound & intra	all traffic
Attack Response				
Download				
Command and Control	√	√	√	√
Propagation	√	√	√	√
Commanded	√	√	√	√

An example signature from Emerging Threat [51]:

```
alert TCP $HOME_NET any -> $EXTERNAL_NET 25 (msg: "BLEEDING-EDGE VIRUS Mytob.GC -
    outbound" ; flow: established; content: "K1ryJsALgAAAC4AAB"; reference: url,www.
    norman.com/Virus/Virus_descriptions/23458/en?show=default; classtype: trojan-
    activity; sid: 2002049; rev:5;)
```

This signature detects emails sent from Mytob using its own SMTP engine.

D) The Storm Worm

The Storm Worm first appeared on January 19, 2007 [23]. Storm is a backdoor trojan and is also known as Trojan.Peacomm, Nuwar, Tibs or Zhelatin. The arranged backdoor is a control channel which allows its infected hosts to operate as a botnet.
Storm is a P2P-based botnet. This means that there is no central coordination server as in traditional botnets.
Another meaningful property of Storm is its high metamorphic behaviour. The attackers behind

Storm quite frequently change their tactics and communication protocols and move to new attack vectors. The older versions of Storm used the Overnet [20] P2P network based on the Distributed Hash Table (DHT) Kademlia. Newer versions (since October 2007) use their own P2P network, which is from now on called Stormnet as in paper [43]. The Stormnet is similar to the Overnet with the difference that the Stormnet is XOR encoded with a 40 byte long key. The key is always the same and known [44]. The assumption is that the use of multiple keys would allow to subdivide the Storm botnet and sell the several parts. Furthermore, in the Stormnet we need not differentiate between bots and benign peers because only bots participate in this network [43, 20, 44, 19].

- **Attack Response**: There is no detectable attack response traffic known.
- **Download**:
- **C&C Communication**: Storm Worm infected bots get their commands using a two-tiers organized architecture. The first-tier is the P2P network which is the Overnet or else the Stormnet. This overlay P2P network is used to find the second-tier computers that send the actual commands and updates.
- **Propagation Traffic**: Storm propagates itself via emails using sophisticated social engineering (e.g. Stormy April Fool's Day email, which was distributed exactly on the first of April [21]). Storm scans the hard drive and the cache of the Internet Explorer to find the email addresses of the next victims. The email contains a malicious, executable attachment or a link to a web site. The linked web site attempts to convince the user to download something like a greeting card, but in reality the user would download the Storm worm. Furthermore, these web sites try to launch a browser exploit. It is remarkable that only old exploits are used because the target of infection is the unexperienced user, who does not patch his computer regularly.
  This email propagation is detectable because a non-SMTP-Server is suddenly sending SMTP mails to a wide range of external SMTP servers.
- **Commanded Traffic**: Storm is primarily a spam-botnet. The spam traffic can be detected in the same way as the propagation traffic because it is a non-SMTP-Server, which suddenly sends SMTP traffic. Storm sometimes executes DoS attacks like SYN or ICMP flooding attacks. This is also conspicuous behaviour and is detectable.

Table 3.6: Analysis of the Storm Worm

	behaviour-based	signature-based	hybrid	
			outbound & intra	all traffic
Attack Response				
Download				
Command and Control	√		√	√
Propagation	√		√	√
Commanded	√	√	√	√

An example signature from Emerging Threats [50]:

```
alert icmp any any -> any any (msg: "ET TROJAN Storm Worm ICMP DDOS Traffic" ; itype
    :8; icode:0; dsize:32; content: "abcdefghijklmnopqr|00 00|" ; depth:22;
    threshold: type both, track by_src, count 1, seconds 60; classtype: trojan-
    activity; sid:2007618; rev:5;)
```

This signature detects an ICMP DDOS attack executed by Storm.

E) Kraken

Kraken first appeared in the late 2007 [47]. Kraken is one of the largest known botnets, the estimation of infected hosts is over 400,000. Kraken exists in multiple versions and the propagation technique is very similar to the technique of Storm. Both use social engineering to spread out. Damballa, a computer security company, specialised in botnet analysis, investigated Kraken in

depth.
Old versions up to and including v315 used UDP port 447 for the C&C Communication. The infected host first sends its hardware and software specification to the C&C Server over this channel. The response of the C&C server contains spam templates and IP addresses. These IP addresses are used to obtain new versions of the bot over a TCP connection on port 447. The port 447 is rarely used and an important indication for a detection. All traffic is encrypted using Kraken's algorithm. The new version of Kraken v316 first appeared in the middle of April 2008 [45]. This new version uses HTTP for the C&C communication. The infected hosts still use UDP to contact the C&C server, but no longer the port 447. Therefore, a detection is much harder. The payload and encryption algorithm of the HTTP C&C packets is still the same as in the older versions of Kraken [32, 33, 45].
All traffic is encrypted, so the detection is more difficult. Kraken uses a definite data structure for its packets. This packet contains the keys (key0, key4, key8), as well as information like the version number of Kraken or the type of the traffic. It is possible to analyse this client packet and to decrypt the payload using an algorithm like the Kraken Decryptor Source Code [46]. This algorithm is already implemented as a plug-in for WireShark. It should also be possible to implement a Snort preprocessor for the decryption of the Kraken traffic. The decrypted traffic contains identifiable signatures for a reliable detection [45]. Kraken v315 is used as baseline for the following evaluation, because there is not yet enough related work material available for newer versions. This version always used the UDP port 447, which makes a reliable detection possible.

- **Attack Response**: There is no detectable attack response traffic known.
- **C&C Communication**: Kraken uses a P2P-based architecture for its C&C communication. The entire C&C traffic except the traffic data (e.g. protocol, port, ip's...) is encrypted, but, due to the fact that Kraken always uses the UDP port 447, a detection is possible using this unencrypted traffic data. If an infected host connects to the network, it exchanges some information like version number and machine specific details like operating system, hostname or average online time with other peers. Then a bot-update over TCP with destination port 447 (for versions up to and including v315) takes place if there is a newer version of Kraken available. The size of this update is between 100KB and 200KB. After the download, the bot starts the update by itself.
- **Propagation Traffic**: The propagation technique of Kraken is based on social engineering and very similar to the propagation of Storm. Both worms use emails to spread. They lure the victim to click on a link or to open an attachment contained in this email.
- **Commanded Traffic**: When a bot gets connected to the network, it tries to connect to mx.google.ch 3 times using TCP destination port 25 (SMTP). This looks like a connection quality test. If the test is successful, the bot will send spam emails at a later date. The bot gets the spam templates, spam payloads and mx server addresses from other peers. When the information has been received, the bot starts spamming. After the termination of the spam stack, it gets new updates and a new spam stack from other peers.

Table 3.7: Analysis of the Kraken

	behaviour-based	signature-based	hybrid	
			outbound & intra	all traffic
Attack Response				
Download	√	√	√	√
Command and Control	√	√	√	√
Propagation	√		√	√
Commanded	√		√	√

An example signature from Emerging Threats [49]:

```
alert udp $HOME_NET 1024: -> $EXTERNAL_NET 447 (msg: "ET TROJAN Possible Bobax/
    Kraken/Oderoor UDP 447 CnC? Channel Outbound"; threshold:type threshold, track
    by_src, count 5, senconds 60; classtype: trojan-activity;reference: url,doc.
    emergingthreats.net/bin/view/Main/OdeRoor; sid:2008109; rev:2;)
```

This signature detects Kraken up to and including version v315 because in these versions the C&C traffic is managed over a connection using UDP on port 447. UDP port 447 is reserved for ddm-dfm Distributed File Management and is very rarely used. A rule based on the property of the port 447 is possible. The C&C traffic of the new version v316 is stealthier, it uses: HTTP on port 80 or 443, encrypted data, random UDP destination ports with random size packet payload and communication on TCP destination port 443. Its C&C communication is difficult to detect. The detection might be possible due to the sent spam emails, the commanded traffic.

3.3 Conclusion

Most items of new malware are botnets, which infect new systems to enlarge the botnet. The motivation of the attackers producing malware has changed. Some years ago, ethical reasons like revenge and power enforcement were prevalent. Today, financial reasons are much more important. Owning a botnet is very lucrative. There is a real market where bot masters offer their botnets and spammers buy these botnets for a lot of money.
Malware has to be more and more complex to remain undetected by current antivirus tools. The competition between malware developers and antivirus industry pushes the development of malware. In the previous analysis, we found interesting trends in the current development of malware:

- **Rise of the diversity of worm features:**
  The chronological order of the analysed worms in the previous Chapter 3.1 shows the rise of the diversity of worm skills very well. One can see this development in the propagation mechanisms. Sophisticated worms have multiple techniques for spreading built-in. They use social engineering techniques by sending mass-mails with malicious attachments or malicious links, copy themselves to files in shared directories or exploit vulnerabilities. Another example is the establishment of a C&C channel. Bots of current versions like Kraken have different ways to connect to their bot master, if one alternative does not work, the bot tries another.

- **Metamorphic code:**
  Metamorphic code is mainly a problem of using signature-based detection. It is not realistic to generate a new signature for every automatically generated subversion of a metamorphic worm. The signature management would be unclear. Normally, a metamorphic worm always uses the same vulnerability, but the exploit code is metamorphic. Therefore, a better solution would generate a signature, which detects the used exploit of a vulnerability and not a specific part of the exploit code.

- **Encryption:**
  If the payload is securely encrypted, an important measuring instrument for the detection is missing. The header data and behaviour-based properties have to be employed for a successful detection. Storm and Kraken use an encrypted channel for the C&C communication. But both encryptions are breakable and, therefore, a detection using the encrypted payload might be possible. However, in case of using Snort IDS, it might be necessary to implement a preprocessor for the decryption. It would be possible to implement an unbreakable encryption system using a public key infrastructure. In this cases an IDS cannot detect the malicious traffic only for a HIDS a detection would still be possible. But a proper encryption of malicious traffic has rarely been used until now.

- **Uncommon protocol used:**
  The first versions of botnets used the IRC protocol for the C&C communication. IRC is a uncommon protocol in enterprise environments and, therefore, it is easy to disable the bot communication by just blocking the IRC traffic. Newer versions use HTTP or HTTPS for the C&C communication. A communication using such a common protocol is much harder to detect, and blocking the entire traffic is not possible. Kraken is a good example to illustrate this development because the version v315 always uses the protocol UDP/TCP over the port 447. The Snort rule uses this property. Now, the new version v316 of Kraken uses either HTTP/HTTPS or UDP/TCP over different port numbers. Therefore, the most important property for the detection of Kraken is no more existing. If the used protocol and

port numbers are no longer helpful for the detection, the payload or suspicious data like unusual packet size, source or destination addresses can be used.

- **No central point of control:**
  An important property for a botnet is robustness. That is why they try to avoid a central point of control, which makes it very hard to disrupt a botnet. Storm and Kraken use a P2P-based control structure and, consequently, it is not sufficient to break one single server to disturb the botnet.

- **Covert channels:**
  One of the most subtle techniques is to establish a covert channel. Nobody knows of or guesses at this traffic. Therefore, there is no detection, interruption or blocking.

Which IDS techniques cope with these current trends?
On the basis of this analysis an IDS using the signature-based and the behaviour-based approach is reasonable. This hybrid approach in combination with a consideration of the outbound traffic might be a good system to detect successful attacks. The following items are some helpful properties of an IDS:

- correlation of different events for the reliability of an alert
- using behaviour-based (no-payload) analysis if the payload is encrypted or common protocol are used
- using signature-based analysis if uncommon protocols are used

Chapter 4

Prototype Development

The theoretical analysis has shown that the analysed malware is detectable monitoring the outbound traffic. The assumption is to reduce the alerts caused by unsuccessful attacks, because an unsuccessful attempted attack does not influence the outbound or the intra traffic. Therefore, the consideration of the traffic direction might be useful. Most of the Snort rules alert on the direction definition “EXTERNAL_NET -> HOME_NET,” that is the direction definition of a conventional inbound traffic rule. The prototype defines the EXTERNAL_NET as ANY, which means the EXTERNAL_NET includes all IP’s no matter the IP is inside or outside the monitored network. Using this definition every in- and outbound rule is also a rule to detect intra traffic. For example the rule “EXTERNAL_NET -> HOME_NET” means “ANY -> HOME_NET” and, therefore detects inbound as well as intra traffic.
The theoretical analysis showed that a purely signature-based approach is weaker than a hybrid approach which correlates different outbound and intra events. A proof-of-concept prototype will be implemented to test these assumptions. In the following, we will explain the underlaying architecture of the prototype.

4.1 Setup of Test Environment

Figure 4.1 presents an overview of the monitored test network. This network can be reached directly from the Internet because there is no firewall in-between. The monitored network consists of different independent servers. A detailed description of the IDS sensor can be found in Chapter 4.1.3.

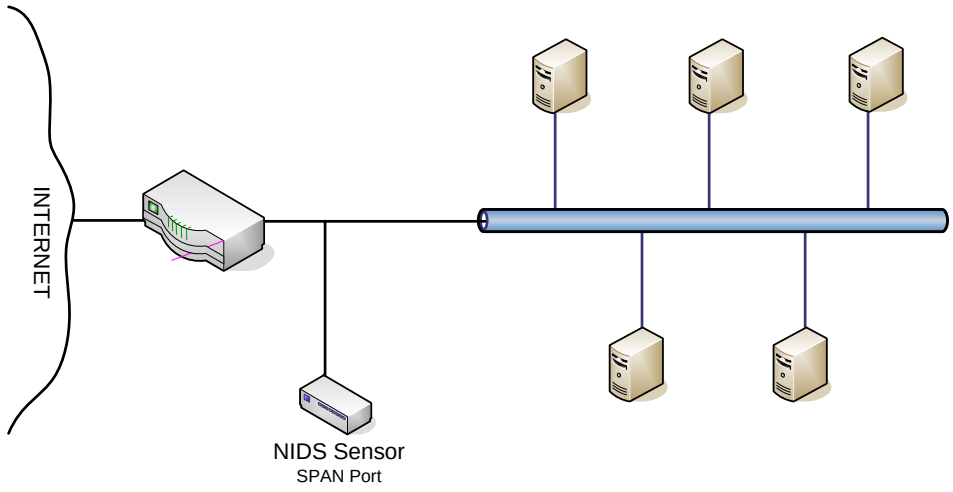


Figure 4.1: Overview of the prototype development architecture

4.1.1 The Router

The connection between the monitored network and the Internet is a router. The traffic in the monitored network is unfiltered because there is no protecting firewall. This is quite an unrealistic scenario since a network with an IDS component is normally well protected using different layers of defense. For testing reasons, however, it is suitable to have a lot of events. An event is a raised alert caused by a matching signature. Most of these events are false-positives and should not lead to an escalation. An escalation is an alert demanding further human interaction.

4.1.2 The Monitored Network

The monitored network, called HOME_NET, consists of around 20 different servers. Most of these servers use Unix-like operating systems.

4.1.3 The Intrusion Detection System

For the IDS we use the open source implementation Snort. A Postgres database is employed for the event storage and the further evaluation of alerts. Additionally, we implemented a set of logical operators which allow a flexible way to specify escalation rules. See Figure 4.2 for an overview.

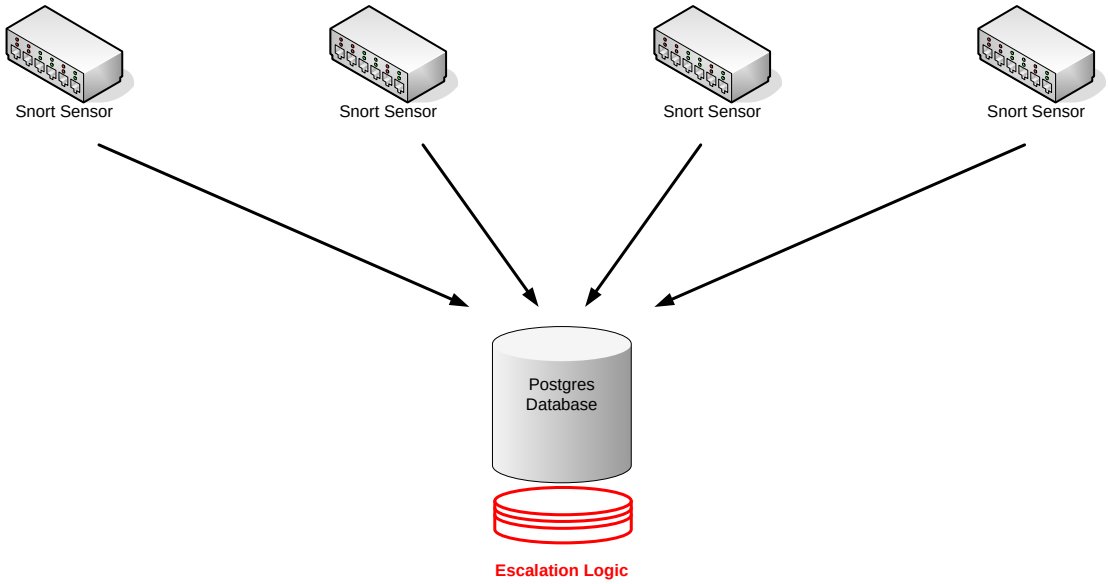


Figure 4.2: Overview of the different components of the IDS

4.2 Escalation Algorithm

In the following, we describe the basic escalation algorithm. The input of the algorithm is the logged events of the Postgres database. The algorithm correlates and filters these alerts. In the best case the algorithm would only output alerts, which detect successful attacks and need human intervention. Therefore, the idea is to implement an additional layer, which inspects the numerous logged events of the database and raises only an alert if it is necessary. The basic logic of the algorithm is to operate as an EDS, which only monitors the outbound and intra traffic. This means in normal case the algorithm ignores all inbound events and escalates all outbound

and intra events. Further, we define a table escalation_rules (Table 4.1) which is use to define the special cases. We use the concept of black and whitelisting to specify the special cases, which we call “escalation rules.” Blacklisting is used to ignore irrelevant outbound and intra events, because the normal case is escalating every outbound and intra event. Whitelisting is used to activate the escalation of an inbound event, because in normal case all inbound events are ignored. This black- and whitelisting can be specified based on several properties which are explain in detail later. in detail.

4.3 The Database Extension

The database is a part of the IDS. One goal of this thesis is to extend the database scheme to enable a correlation of different events and to make a difference between inbound, outbound and intra traffic. We have already visualised these terms in Figure 1.3. Basically, we need three additional tables: A table to store the escalation rules (escalation_rules), a table to store particular content patterns (content_patterns) and a table which stores correlated signatures (signature_correlation). In the following sections, we will present the use and benefits of these three tables.

4.3.1 The Table escalation_rules

id	class_id	sig_id	dir	pid_eq_dir	pid_op_dir	weight	action	count	dt

Table 4.1: Escalation_rules

This table contains all special escalation rules. The normal case is the escalation of all outbound and intra traffic events and no escalation of all inbound traffic events. Therefore this table defines the rules, which are handled different from that normal case. This escalation rules are specified using the following attributes:

- **id** is automatically generated, unique and the primary key of the table.
- **class_id** defines special conditions for a particular signature classification. Signature classification is an option to group and distinguish rules according to their causing threat. Some examples for signature classifications are: “policy-violation,” “trojan-activity” or “misc-activity.”
- **sig_id** defines special conditions for a particular signature.
- **pattern_eq_dir** and **pattern_op_dir** references to a pattern in the content_pattern Table 4.2. Snort offers an option “tagged packets,” which allows to log all packets from the same session. Therefore, all this logged packets can be examined, if they contain a particular pattern. The attributes pattern_eq_dir and pattern_op_dir allow a differentiation, which packets have to be analysed; the packets of the same direction like the logged attack or the packets of the opposite direction. The idea that this content analysis of following packets allows a more reliable statement concerning the effectiveness of the attack.
- **pattern_op_dir** references to a pattern in the content_pattern table. All packets from the same session and opposite direction like the initial event will be examined.
- **weight** depends on the accuracy of the rule. An exact rule gets a high weight. The most general rule, that is the escalation of the outbound traffic, gets the weight 0.
- **action** is of the type boolean and defines the action if the rule applies (false: no escalation; true: escalation).
- **count** and **d_t** are coupled and are used to make a statement about the frequency of occurrence of a particular event. This condition is applied if a particular alert occurs more than “count” -times during the time “d_t.”

It is possible that several rules apply for the same event. In this case, the rule with the highest weight determines the action. That is to say the decision for an escalation or no escalation comes with the matching rule having the highest weight.

4.3.2 The Table content_pattern

id	msg_offset	msg_lenght	string

Table 4.2: Content_pattern

The attributes pattern_eq_dir and pattern_op_dir reference the table content_pattern. The table content_pattern contains specifications about the data payload following the escalated event. A content pattern is defined using the following attributes:

- **id** is automatically generated, unique and the primary key of the table.
- **msg_offset** and **msg_lenght** define the part of the payload, which will be analysed.
- **string** defines text, which will be searched for in the payload using “substring matching.”

4.3.3 The Table signature_correlation

id	rid_ref	dir	sig	d_t	count	comment

Table 4.3: Signature_correlation

The table signature_correlation allows correlating multiple signatures. For every rule of the escalation_rules table, it is possible to define correlated signatures. The table signature_correlation contains conditions related to signatures which have to be fulfilled for an escalation or for a prevention of an escalation. To specify the properties of the entries following attributes are available:

- **id** is automatically generated, unique and the primary key of the table.
- **rid_ref** is a reference to the id of the table escalation_rules. An escalation rule matches an event, if the corresponding correlated signatures of this table match as well.
- **dir** specifies the direction of the packet containing the signature.
- **sig** specifies the signature number.
- **d_t** and **count** are coupled and used to make a statement about the frequency of occurrence of a particular event. This condition is applied if a particular alert occurs more than “count” -times during the time “d_t.” The attributes “count” and “d_t” always have to be assigned to a value unequal to NULL.

4.4 The Escalation Query

The information contained in the described tables has to be combined to determine if an escalation of an alert is necessary. A Query using several functions analyses every alert. Following three functions were implemented:

- **check_frequency**
  The function check_frequency checks if a particular signature exceeds a fixed limit in a fixed period of time.

- **check_pattern**
  The function check_pattern checks if a defined pattern of the table content_pattern is found in particular packets. These packets follow an alert and are of the same session like the packet, which rose the alert.

- **check_sig_correlation**
  The function check_sig_correlation check the occurrence of a signature combination and their frequency.

These functions help to implement the escalation algorithm written in Perl. The PostgreSQL Query is a big part of this Perl script. The idea is to periodically (e.g. every ten minutes) execute this script using a daemon or a cron job and to analyse the events, which occurred since the last execution of the script.

4.5 The Escalated Correlation Rules

In this section, we will present some escalation rules to get an idea of the possibilities of the implemented correlation tool.

4.5.1 Successful Web Application Attacks

The signature class web application attack contains signatures for attacks such as injection flaws (particularly SQL injection), or cross site scripting. All these attacks have in common that just few of them are successful. We analyse the traffic after the escalated web application attack to differentiate between a successful and an unsuccessful attack. A replay of an attempted attack containing the string “401 Unauthorized”, “403 Forbidden” or “404 Not Found,” is an indicator for an unsuccessful attack and would remove alerts for attacks that were likely unsuccessful. If this is true for every single attack has to be proved in detail, but it is at least an indication. Therefore, following two entries are necessary for a correct filtering of the events:

id	msg_offset	msg_lenght	string
x	0	25	401 Unauthorized|403 Forbidden|404 Not Found

Table 4.4: Entry in the table content_pattern for the filtering of successful web application attack

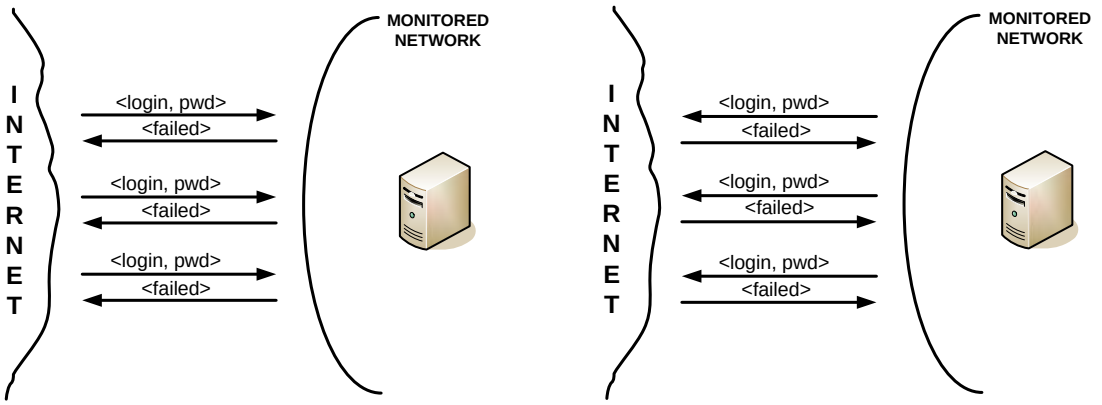
4.5.2 Login Attempts

A login brute force attack from the internet is a common scenario. Normally, an intrusion into a c protected network is not possible, due to a powerful authentication mechanism. More interesting is a login brute force attack launched from the internal network. This is a strong indication of a worm infection. In this case, the brute force attack is the commanded traffic (see Chapter 3.1.4) of the worm or frequently it is a matter of a botnet. Therefore, in the second case it is important to escalate, but in the first case the event has to be ignored. Hence, the ability to value events in consideration of the traffic direction is necessary.

The Snort rules contain the following signature:

```
alert tcp $HOME_NET 21 -> $EXTERNAL_NET any (msg: "INFO FTP Bad login" ; flow:
    from_server,established; content: "530"; pcre: " /^530\s+(Login|User)/smi"
    classtype:bad-unknown; sid:491; rev:8; tag:session,5,packets;)
```

Normally, this signature only alerts in case of the harmless brute force attack from the outside of the network into the HOME_NET. Naturally, the filtered event is not the inbound <login, pwd> attempt, but the outbound <failed> response.
In our case, we define the EXTERNAL_NET as ANY and, therefore, we also get an event for the opposite direction. In a second step, we filter the events again using their directions and only escalate the login brute force attacks from the inside of the network.



A harmless brute force attack from the internet to the monitored network

An alarming brute force attack from the inside of the monitored network

Figure 4.3: Harmless versus alarming login attempts

4.6 Conclusion

The alert system of the implemented prototype is based on the outbound and intra traffic. The definition of the EXTERNAL_NET as ANY allows also to log potential inbound events. The speciality of the prototype is an additional layer which refines the escalation of the logged alerts. For this refinement we can use the following six features to specify the different escalation rules:

- **signature:**
  For which signature is the escalation rule applicable.
- **direction:**
  For which direction is the escalation rule applicable.
- **signature classification:**
  For wich signature classification is the escalation rule applicable.
- **content pattern in traffic following an alerting packet:**
  What has the content in following traffic packets pattern to be (or not to be) so that the escalation rule is applicable.
- **frequency:**
  For which frequency is the escalation rule applicable.
- **correlated signatures:**
  For which correlated signature and with which frequency is the escalation rule applicable.

These features give the opportunity to tune the alerts according to the monitored network. It is easy to implement additional escalation rules according to the explained features but requires a detailed analysis of potential anomalies.

Chapter 5

Evaluation and Results

In the first part of this chapter, we will compare a conventional IDS with an EDS. In the second part, we will evaluate the implemented prototype with its correlation rules. Therefore, we will differentiate between the following three systems:

- **IDS:** Snort including all rules
- **EDS:** Snort including the outbound rules
- **Prototype:** Snort and an extension to refine the alerts using correlation

For the evaluation of an IDS in general we differentiate between the failures of the system using following structure:

1. **No alert occurs but an alert is required** (e.g. false-negative, packet loss)
   Such a failure has different causes. It is important that the performance of the IDS scales with the existent traffic load. Traffic overload provokes packet loss and, therefore, this loss might inhibit a necessary alert. Furthermore, if it is a signature-based IDS, a particular difficulty is the high polymorphism of current malware and zero-day attacks. The cause for missed alarms using a behaviour-based system is bad fine tuning. Indeed, it frequently is very difficult to differentiate between normal and abnormal traffic.
2. **An alert occurs but no alert is required** (e.g. false-positive, unsuccessful attack)
   Using a signature-based system, imprecise signatures normally trigger the false-positives or attacks, which are not successful. With a behaviour-based system the problem of the right tuning is again the weak point.

For the evaluation, we choose an approach based on a reference system. The reference system is always the conventional IDS. The test systems are the EDS, which only monitors the outbound traffic, but uses correlation, and the prototype, which uses some kind of correlation (see Chapter 4.6 for details). The goal in this evaluation is to show that the test system is better than the reference system concerning false alarms and that the test system is at least almost as good as the reference system concerning false-negatives. All systems, the IDS, the EDS and the prototype use the same Snort ruleset, which contains around 9200 rules, of which 7700 are inbound rules and 1500 are outbound rules. The tool Oinkmaster is used to disable rules which are irrelevant to the monitored system.
In the following chapters, we will first of all evaluate the general approach of extrusion detection and, in a second part, we will evaluate the developed prototype.

5.1 General comparison between an IDS and an EDS

5.1.1 The Methodology of the Evaluation

An IDS monitoring the entire traffic represents the reference system for the evaluation of the general extrusion detection approach. The analysis of the reliability of an EDS consists of an analysis of the false-negatives and an analysis of the false alarms (this means false-positives and unsuccessful attacks), which we evaluate using the following methodology:

- **false-negatives:** Using capture files
- **false-positives and unsuccessful attacks:** Using the alerts monitoring the test network and evaluate these alerts in a quantitative, as well as in a qualitative manner. We have measured the quality of the alerts using the scale visualised in Table 5.1.

false-positive No attack, caused by imprecise rules	fp
forensic information No attack, but might be recognition for a planned attack such as scanning	for
unsuccessful attack An attempted attack, but most likely without a performed intrusion	unsuc
might be successful attack An attempted attack where it is ambiguous if the attack was succcessful	
successful attack An attempted and, first of all, a successful attack	

Table 5.1: Alert classification

5.1.2 General Conditions

In the following, we will compare the reliability of an IDS, the reference system, with the reliability of an EDS, which only monitors the outbound traffic. First of all, we will analyse the behaviour relating to false-negatives and, in the second part, we will analyse the false alarms.

The False-Negatives

For this evaluation, we have used dump files of malicious traffic. The general problem of testing an EDS is the need of the complete traffic dumps. It is not enough to replay only the attack; we also need the responses and the reaction of the infected host. In order to be capable to differentiate between successful and unsuccessful attacks, we need the traffic, which follows the successful intrusion.
It is also important to get the very first packet of the connection establishment, because Snort only analyses packets with a correct preceding establishment protocol (e.g. correct and complete TCP handshake).
The test environment 4.1 consists of almost no malicious traffic. Therefore, we input dump files of the libpcap format to Snort to test the presence of false-negatives. The config-file and some database entries have to be adapted to these test files. In case of a self-propagating worm, we choose the HOME_NET in such a way, that both source and destination address belongs to this subnet. In this way, the input dump file simulates an internal propagation attempt. The infected host in the HOME_NET is affecting other hosts in the HOME_NET. This approach is reasonable since a worm-infected host will normally also spread in the same subnet. These propagation attempts on a host of the HOME_NET are actual indications of an infection.
We have used dump files from the website openPacket.org [55] and dump files from the project LOBSTER [56].
OpenPacket.org provides a centralized repository of network traffic traces. These traces are used by researchers, analysts and other members of the digital security community.
We have used the following traces from OpenPacket and have detected the attacks contained in them successfully:

- **Kraken**
  The C&C communication of the botnets is detected by means of the following signature of the type “trojan-activity:”

```
alert tcp $HOME_NET 1024: -> $EXTERNAL_NET 447 (msg: "ET TROJAN Bobax/Kraken/
    Oderoor TCP 447 CnC? Channel Initial Packet Outbound"; flow:established;
    dsize:24; threshold:type threshold, track by_src, count 2, seconds 360;
    classtype:trojan-activity; reference:url, doc.emergingthreats.net/bin/view/
    Main/OdeRoor; sid:2008103; rev:2;)
```

- **Web Server Folder Traversal**
  A vulnerability of the Microsoft Internet Information Web Server (IIS 4.0 and 5.0) allows remote attackers to change files as if they were logged on locally. This vulnerability is used by different worms. One example is the Nimda worm.
  The trace file escalated six outbound alerts of the following two signatures of the type attack-response:

```
alert tcp $HTTP_SERVERS $HTTP_PORTS -> $EXTERNAL_NET any (msg: "ATTACK-
    RESPONSES file copied ok"; flow:established; content: "1 file|28|s|29|
    copied"; nocase; metadata:policy balanced-ips drop, policy security-ips
    drop; reference:bugtraq,1806; reference:cve,2000-0884; classtype:bad-
    unknown; sid:497; rev:14; tag:session,5,packets;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "ATTACK-RESPONSES directory
    listing"; flow:established; content: "Volume Serial Number"; classtype:bad-
    unknown; sid:1292; rev:9; tag:session,5,packets;)
```

- **SQL Slammer** The SQL Slammer escalates one intra alert. This alert is reliable because it simulates worm propagation inside the monitored network. The following signature triggered the escalation:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 1434 (msg: "SQL Worm propagation
    attempt"; flow:to_server; content: "|04|"; depth:1; content: "|81 F1 03 01
    04 9B 81 F1 01|" ; content: "sock"; content: "send"; metadata:policy
    balanced-ips drop, policy security-ips drop; reference:bugtraq,5310;
    reference:bugtraq,5311; reference:cve,2002-0649; reference:nessus,11214;
    reference:url,vil.nai.com/vil/content/v_99992.htm; classtype:misc-attack;
    sid:2003; rev:12; tag:session,5,packets;)
```

  Remarkably, this alert is only interesting when the direction of the packet is outbound or intra. An inbound alert is mostly a false-positive because it is an unsuccessful attempted attack.

Microsoft Windows Server Service Remote Code Execution is the only malicious traffic capture which raised an inbound, but no outbound alarm.
The escalated inbound signature is:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg: "NETBIOS SMB-DS srvsvc
    NetrPathCanonicalize WriteAndX little endian overflow attempt"; flow:established
    ,to_server; flowbits:isset,dce.bind.srvsvc; content: "|00|"; depth:1; content: "
    |FF|SMB/"; within:5; distance:3; byte_test:1,!&,128,6,relative; pcre: "/^.{27}/
    sR"; byte_jump:2,23,relative,from_beginning,little; pcre:"/^.{4}/sR"; content: "
    |05|"; within:1; byte_test:1,&,16,3,relative; content: "|00|"; within:1;
    distance:1; content: "|1F 00|"; within:2; distance:19; pcre: "/^.{4}(\x00\x00\
    x00\x00|.{12})/sR"; byte_jump:4,-4,little,multiplier 2,relative,align; byte_test
    :4,>,256,0,little,relative; metadata:policy balanced-ips drop, policy
    connectivity-ips drop, policy security-ips drop; reference:bugtraq,19409;
    reference:cve,2006-3439; reference:url,www.microsoft.com/technet/security/
    bulletin/MS06-040.mspx; classtype:attempted-admin; sid:7250; rev:8; tag:session
    ,5,packets;)
```

The reason why no outbound escalation occurs is simple. The signature escalates on inbound attack attempts. There is no testing of the reaction of the host concerning a successful infection. Therefore, this signature is susceptible to alarms without a successful intrusion. A signature, which escalates on the reaction of an infected host, would be reasonable.
OpenPackets.org contains further malicious traffic captures of TCP-scans and UDP-scans. The used ruleset contains no active rule to detect TCP- or UDP-scanning. The detection of inbound scanning indicates upcoming attacks, but it is no indicator for a successful intrusion. It might be more interesting to detect outbound scanning traffic, which could originate from an infected host within the monitored network in terms of propagation or commanded traffic.

LOBSTER is the short form for Large Scale Monitoring of Broadband Internet Infrastructure. This project uses the passive network monitoring approach. It aims at detecting attackers trying

to penetrate legitimate computers [57]. LOBSTER offers some further capture files. All these files exploit buffer overflow vulnerabilities. Some of these vulnerabilities are typically used by worms. For such attacks, we have defined both source and destination addresses as inside the HOME_NET and assumed that a host would spread further in case of an infection. Hence, the capture file simulates a propagation of the worm within the monitored network, and a detection of the corresponding malicious traffic is possible. Some vulnerabilities are probably not used by worms, therefore, it is unknown if the particular attack is detectable using an EDS. The capture file might not complete be and detectable outbound traffic is missing.
The name of the malware, which generated the capture file, is not always known. Therefore, we differentiate the capture files according to the signature, which escalated.
These are typically vulnerabilities used by worms, a successful attack will generate propagation traffic on the infected host and, therefore, we defined the source and destination address as inside the monitored network:

- **Netbios SMB-DS Buffer Overflow Attempt (Mytob)**
  The vulnerability is a stack-based buffer overflow in the Plug and Play (PnP) service for Microsoft Windows 2000 and Windows XP Service Pack 1. This vulnerability allows remote attackers to execute arbitrary codes and has been exploited by the Mytob worm [58].
  The following signature detects the malicious traffic:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg: "NETBIOS SMB-DS umpnpmgr
    PNP_QueryResConfList unicode little endian attempt"; flow:established,
    to_server; flowbits:isset,dce.bind.umpnpmgr; content: "|00|"; depth:1;
    content: "|FF|SMB%"; within:5; distance:3; byte_test:1,&,128,6,relative;
    pcre: "/^.{27}/sR"; content: "&|00|"; within:2; distance:29; byte_jump
    :2,-6,relative,from_beginning,little; pcre:"/^.{4}/sR"; content:"|05|";
    byte_test:1,&,16,3,relative; content:"|00|"; within:1; distance:1; content:
    "6|00|"; within:2; distance:19; metadata:policy balanced-ips drop, policy
    connectivity-ips drop, policy security-ips drop; reference:bugtraq,14513;
    reference:cve,2005-1983; reference:url,www.microsoft.com/technet/security/
    bulletin/ms05-039.mspx; classtype:protocol-command-decode; sid:3999; rev:5;
     tag:session,5,packets;)
```

- **Netbios DCOM RPC Buffer Overflow Attempt (Welchia/Blaster)**
  The vulnerability is a buffer overflow vulnerability in a certain Distributed Component Object Model (DCOM) interfaces for RPC in some Microsoft Windows Servers (NT 4.0, 2000, XP, 2003). The vulnerability allows attackers to execute an arbitrary code using a malformed message. The worms Blaster and the Welchia both exploit this vulnerability [58].
  The following signature detects the malicious traffic:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 135 (msg: "NETBIOS DCERPC NCACN-IP-TCP
     ISystemActivator RemoteCreateInstance little endian attempt"; flow:
    established,to_server; flowbits:isset,dce.bind.ISystemActivator; content: "
    |05|"; byte_test:1,&,16,3,relative; content:"|00|"; within:1; distance:1;
    content: "|04 00|"; within:2; distance:19; content: "|01 10 08 00 CC CC CC
    CC|"; distance:0; content: "|5C 00 5C 00|"; distance:0; byte_test
    :4,>,256,-8,relative,little; metadata:policy balanced-ips drop, policy
    connectivity-ips drop, policy security-ips drop; reference:bugtraq,8205;
    reference:cve,2003-0352; reference:cve,2004-0116; reference:url,www.
    microsoft.com/technet/security/bulletin/MS03-026.asp; classtype:protocol-
    command-decode; sid:9601; rev:5; tag:session,5,packets;)
```

- **NETBIOS SMB-DS lsass Buffer Overflow Attempt (Sasser)**
  The vulnerability is a stack-based buffer overflow in certain Active Directory service functions. This vulnerability allows remote attackers to execute an arbitrary code and has been exploited by the Sasser worm [58].
  The following signature detects the malicious traffic:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg: "NETBIOS SMB-DS lsass
    DsRolerUpgradeDownlevelServer unicode little endian overflow attempt"; flow
    :established,to_server; flowbits:isset,dce.bind.lsass; content:"|00|";
    depth:1; content:"|FF|SMB%"; within:5; distance:3; byte_test:1,&,128,6,
    relative; pcre: "/^.{27}/sR"; content:"&|00|"; within:2; distance:29;
    byte_jump:2,-6,relative,from_beginning,little; pcre: "/^.{4}/sR"; content:
    "|05|"; byte_test:1,&,16,3,relative; content: "|00|"; within:1; distance:1;
     content: "|09 00|"; within:2; distance:19; byte_test:4,>,256,0,little,
    relative; metadata:policy balanced-ips drop, policy connectivity-ips drop,
    policy security-ips drop; reference:bugtraq,10108; reference:cve,2003-0533;
     reference:nessus,12205; reference:url,www.microsoft.com/technet/security/
    bulletin/MS04-011.mspx; classtype:attempted-admin; sid:5219; rev:4; tag:
    session,5,packets;)
```

The IDS but not the EDS could detect the malicious traffic which exploits the listed vulnerabilities:

- **Web-FrontPage Buffer Overflow Attempt**
  The vulnerability is a stack-based buffer overflow in Symantec Antivirus 10.1 and Client Security 3.1. This vulnerability allows remote attackers to execute an arbitrary code via unknown attack vectors [58].

- **WEB-MISC Chunked-Encoding transfer attempt**
  The vulnerability is a buffer overflow in the chunked encoding transfer mechanism in Internet Information Server (IIS 4.0 and 5.0) Active Server Pages. This vulnerability allows attackers to execute an arbitrary code or to cause denial of service [58].

- **Exploit Virusscan Overflow Attempt**
  The vulnerability is a stack-based buffer overflow in Symantec Antivirus 10.1 and Client Security 3.1. This vulnerability allows remote attackers to execute an arbitrary code via unknown attack vectors [58].

- **Specific-Threats ASN.1 Buffer Overflow Attempt**
  The vulnerability is a heap-based buffer overflow in the BERecBitString function in Microsoft ASN.1 library. This vulnerability allows remote attackers to execute an arbitrary code [58].

- **IMAP Login Buffer Overflow Attempt**
  The vulnerability is a buffer overflow in the login function in the IMAP server (imapd) in Ipswitch IMail 5.0 and earlier versions. This vulnerability allows remote attackers to execute an arbitrary code and cause denial of service using either a long user name or a long password [58].
  The following signature detects the malicious traffic:

There is one capture file containing an attack on TCP port 1025 which is not detected, neither by using the EDS nor using the IDS. Our Snort ruleset includes no signature for this attack and, therefore, the detection is not possible.

The False-Positives and Unsuccessful Attacks

This section presents the comparison between the false-positives caused by an inbound alert and the false-positives caused by an outbound alert. The test net consists of about 20 independent servers and most of them use a Unix-like operating system. The Figure 4.1 in Chapter 4 presents an overview of the test environment. The testing phase executed over 20 days from Monday, July 21, until Tuesday, August 11.

Quantitative Evaluation

The Figure 5.1 presents a quantitative evaluation of all alerts. The qualitative analysis of the alerts, in the next section, will show that we have found no assumed successful attacks during the test phase. The hosts in the test environment are well patched and most of them of a Unix-like operating system. There exist less malware for Unix-like operating systems than for the Microsoft Windows operating system (MS Windows), and therefore, systems using the MS Windows operating system are more susceptible. Therefore, in this quantitative analysis we value all alerts as false alarms. The intention is to compare the numbers of alerts of an IDS with the number of alerts of an EDS. The simple way is to count all alerts of the IDS and all alerts of the EDS per day. But it turned out that such statistics are not representative in case of attempted brute force attacks because some signatures occur hundreds of times within a few seconds. Such occurrences would falsify the statistics. Therefore, we count one particular signature only once a day and in this way, we avoid this falsification.
August, 4 is conspicuous in the statistic. The reason for only this small number of alerts is not known, but it might be a failure of the IDS operation because there is no alert for 24 hours after the system was rebooted on 5:30 am.

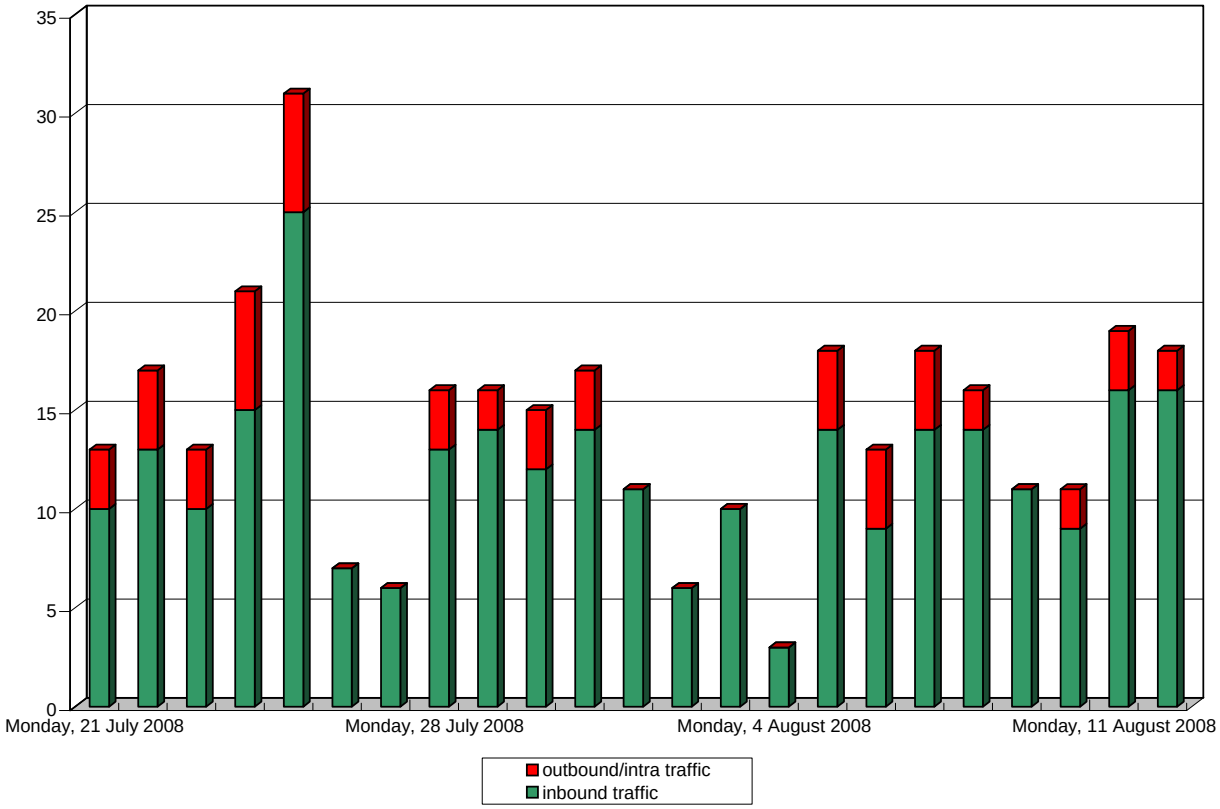


Figure 5.1: The number of different signatures per day, divided in inbound and outbound alerts

Figure 5.1 shows a remarkable reduction of false alarms by 80 percent using an EDS instead of an IDS. This result has to be handled with care. An analysis of active rules of the ruleset shows the following image: There are a lot more inbound rules than outbound rules, therefore, more inbound false alarms have to be expected. Table 5.2 shows the numbers of total active rules as well as the number of active in- and outbound rules. Special rules are those, which do not have a specified direction. For example some rules detect traffic direction independently. A short analysis of the different outbound alerts shows that, except for one alert, all alerts are of the signature class “policy-violation.” That is the reason why we include the numbers concerning “policy-violation.” There are proportionally a lot of outbound “policy-violation” rules. In the next section of the qualitative evaluation, we will discuss the significance of “policy-violation” rules.

	total rules	inbound rule	outbound rules	special rules
all rule classes	9256	7718	1485	53
policy-violation rules	118	34	80	4

Table 5.2: The active rules of the Snort ruleset

Qualitative Evaluation
The evaluation of the alerts is done using the terms presented in Table 5.1:
Table 5.3 and 5.4 analyse the different alerts reported during the testing phase. The attribute "no" of the tables represents the number of days on which the corresponding signature occurs. The qualitative evaluation contains no alerts of the signature classification "policy-violation." Each company has to define their own policy. If some actions like P2P or MSN chatting are permitted, the corresponding rule has to be deleted from the ruleset. If the action is forbidden due to company policy, the malefactor has to be traced and reproved. Furthermore, a "policy-violation" does normally not threaten the network security, but it can be the source of big capacity exhaustion and, therefore, an interesting information for companies.
The analysis of the outbound alerts was easy since there is only one alert (see Table 5.3).

no	signature	rating
9	INFO FTP Bad login	unsuc

Table 5.3: Outbound and intra alerts

The escalated alert "INFO FTP Bad login" is an unsuccessful brute force attack, as we have discussed in section 4.5.2 in detail. This inbound brute force attack escalates because the replay <failed> to the brute force attack raises the alert.
There are many more inbound alerts than outbound alerts. We have analysed every one of these alerts, to determine whenever it is a successful attack, an unsuccessful attack or a false-positive. The rating of the alerts is difficult and often ambiguous. Only a few alerts could be definitely rated. Rated is the most likely possibility. Sometimes, an alert was raised several times and the evaluation of these alerts is different. In this case, two ratings (e.g. unsuc/fp) are marked.
We have not found a single successful inbound attack. But there are some uncertain alerts rated as "unknown" ( ). The analysis of the alert is only effective if enough information is available. There are different sources from which we can gain additional information: The rule escalating the alert, the signature database of snort.org, the payload of the suspect packet, the reverse lookup of the participating IP's and so on. However this information is sometimes not available.
It seems to be very plausible that all these alerts were raised as a result of unsuccessful attempted attack. A real infection is improbable and, normally, such attacks are not successful.

5.1.3 Results

The Table 5.5 sums up the results of the false-negatives analysis of the previous chapter. Only a part of the capture files are detected by the IDS and the EDS. Both systems do not detect the attacks "TCP-scan and UDP-scan" and the "Attack on TCP Port 1025." This means the test system and the reference system react in the same way. The reason for no alert of "TCP-scan and UDP-scan" and "Attack on TCP Port 1025" is the missing rule in the Snort ruleset. The rule for detecting "TCP-scan and UDP-scan" produces a lot of inbound false alarms. Inbound host scanning is common but normally harmless. More suspicious is host scanning caused by a internal host. This could be a strong indicator of an infection.
Some alerts raised only an IDS alert but no EDS alert. The reason lies in the signature, which detects inbound attack attempts. The capture file contains no traffic of the reaction to the intrusion. Possibly, that the EDS would also detect the attack by means of this missing traffic.

no	signature	rating
20	IMAP SSLv3 invalid Client_Hello attempt	unsuc
17	WEB-MISC DELETE attempt	unsuc
14	WEB-CGI calendar access	fp
10	WEB-PHP remote include path	unsuc
10	WEB-MISC IBM Lotus Domino Web Server Accept-Language header buffer overflow attempt	unsuc
9	SQL probe response overflow attempt	fp
8	WEB-CGI finger access	unsuc
7	SMTP possible BDAT DoS attempt	unsuc
6	WEB-CLIENT IE JPEG heap overflow single packet attempt	unsuc
6	WEB-MISC SSLv2 openssl get shared ciphers overflow attempt	unsuc
6	SMTP Content-Transfer-Encoding overflow attempt	unsuc
6	ICMP PING NMAP	for
6	WEB-FRONTPAGE /_vti_bin/ access	fp
5	WEB-CLIENT Adobe BMP image handler buffer overflow attempt	unsuc/fp
5	ICMP Source Quench	for
5	SMTP MAIL overflow attempt	fp
5	WEB-MISC intranet access	unsuc
5	DNS SPOOF query response with TTL of 1 min. and no authority	unsuc
5	SMTP SEND overflow attempt	fp
4	RPC portmap NFS request UDP	unsuc
4	WEB-MISC SSLv3 invalid data version attempt	unsuc
4	WEB-CGI redirect access	unsuc
4	EXPLOIT Microsoft Exchange ical/vcal malformed property	unsuc
4	DNS named version attempt	for
3	SMTP PCT Client_Hello overflow attempt	unsuc
3	SMTP TLS SSLv3 invalid data version attempt	unsuc
3	WEB-IIS view source via translate header	unsuc
3	SQL generic sql update injection attempt	fp
3	SMTP Content-Type overflow attempt	fp
2	WEB-PHP admin.php access	unsuc
2	SMTP x-unix-mode executable mail attachment	unsuc
2	WEB-CGI guestbook.cgi access	unsuc
2	WEB-MISC http directory traversal	unsuc
1	WEB-MISC backup access	unsuc
1	ICMP PING CyberKit 2.2 Windows	for
1	SQL ping attempt	for
1	SMTP SSLv2 openssl get shared ciphers overflow attempt	unsuc
1	WEB-PHP modules.php access	unsuc
1	BAD-TRAFFIC udp port 0 traffic	unsuc
1	WEB-MISC Phorecast remote code execution attempt	unsuc
1	SQL generic sql insert injection atttempt	unsuc
1	SMTP MS Windows Mail UNC navigation remote command execution	unsuc
1	WEB-PHP IGeneric Free Shopping Cart page.php access	unsuc
1	BACKDOOR only 1 rat runtime detection - icmp request	for
1	SPYWARE-PUT Trackware casalemedia runtime detection	unsuc
1	SMTP chameleon overflow	unsuc
1	WEB-CLIENT QuickTime Object ActiveX CLSID access	unsuc

Table 5.4: Inbound alerts

The results of the false alarm analysis show a reduction of false-positives and unsuccessful attacks using an EDS instead of an IDS. We also would like to stress the fact that the ruleset contains many more inbound rules and, therefore, more inbound false-positives are to be expected. A closer look at the outbound alerts shows that all, except for one outbound alert, be of the signature class “policy-violation,” which normally does not threaten the security of a network. An overview of the rated alerts of the different system one can find in the Table 5.6.

Attack	IDS	EDS
Kraken	√	√
Web Server Folder Traversal	√	√
SQL Slammer	√	√
MS Windows Server Service Remote Code Execution	√	
TCP-scan and UDP-scan		
Netbios SMB-DS Buffer Overflow Attempt (Mytob)	√	√
Netbios DCOM RPC Buffer Overflow (Welchia/Blaster)	√	√
Netbios SMB-DS lsass Buffer Overflow Attempt (Sasser)	√	√
Web-FrontPage Buffer Overflow Attempt	√	
Web-Misc Chunked-Encoding Transfer Attempt	√	
Exploit Virusscan Overflow Attempt	√	
Specific-Threats ASN.1 Buffer Overflow Attempt	√	
IMAP Login Buffer Overflow Attempt	√	
Attack on TCP Port 1025		

Table 5.5: Results of the false-negative analysis of the EDS

	fp	for	unsuc		
IDS	8	7	33	0	0
EDS	0	0	1	0	0

Table 5.6: Overview of the rated alerts

5.1.4 Conclusion

Summing up the reliability concerning false-negatives of a system monitoring only the outbound traffic, we can say that it is comparable to an IDS monitoring the entire traffic. There are some false-negatives, but a closer look shows that the reason for “no alert” is caused by a rule which simply detects an attack attempt and not a successful attack. Therefore, this failed alert is not significant. All other successful attacks are normally detected. The reduction of the number of false-positives is remarkable, especially in the qualitative analysis, where we have omitted alerts of the signature classification “policy-violation” for well-funded reasons. These outbound traffic alerts could be managed in a good way, not like the huge amount of alerts generated of the IDS monitoring the entire traffic.

5.2 The Prototype and an EDS in Comparison with an IDS

5.2.1 The Methodology of the Evaluation

An IDS monitoring the entire traffic has been used as a reference systems. The prototype and again an EDS are evaluated using real-life traffic. The goal of the prototype and the EDS is to reduce the false alarms without increasing the false-negatives in reference to the IDS. The evaluation includes a qualitative analysis of the raised alerts of the three systems (IDS, EDS and the prototype). For this analysis we have used the scale visualised in the Table 5.1.

5.2.2 General Conditions

The developed prototype is evaluated using real-life traffic. The monitored network is a part of the ETH Zürich university network. Figure 5.2 shows the test architecture. The IDS prototype gets access to the traffic of the ETH subnetwork using a SPAN port and there is also a remote access to manage the IDS.

The monitored network consists of different workstations, which are assumedly well patched and which are connected to the ETH local area network (LAN). Furthermore, for this evaluation we have ignored all alerts caused by a “policy-violation” rule. All these alerts do not really threaten

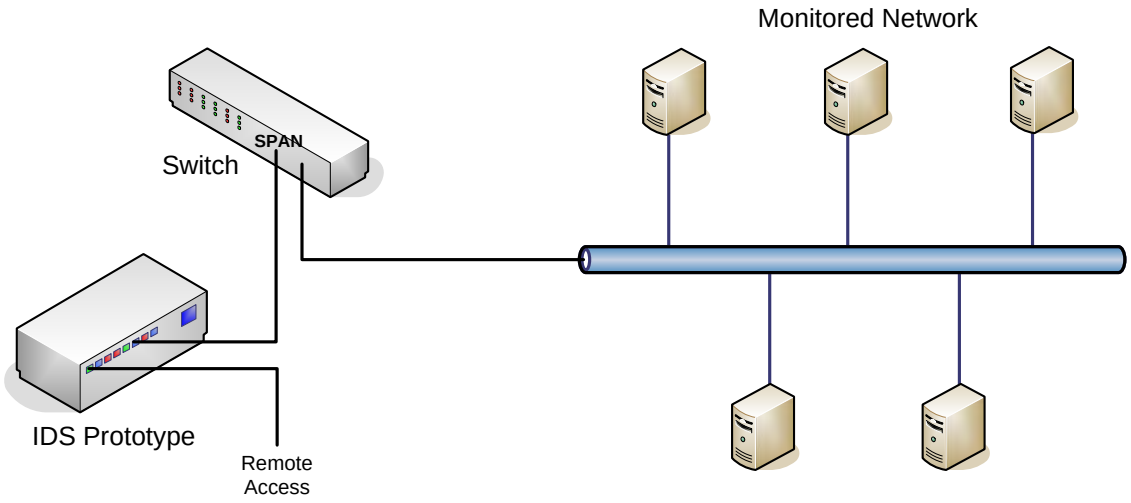


Figure 5.2: The prototype test architecture

a host. However, for a company it is important to define and achieve a policy and, therefore, to track possible policy violations. All three systems (IDS, EDS and the prototype) analyse exactly the same traffic. Tables 5.8, 5.7 and 5.7 contain the alerts of the particular system.
Every alert is listed once only per table, independently of the number of occurences.
A precise analysis of every alert is necessary to make a statement about the reliability of the raised alarms. And often not even an in-depth analysis was successful. We classify the analysed alerts in four different categories according to the Table 5.1.

Alert Comparison of the different IDS's

Table 5.7 shows the alerts of the prototype, Table 5.8 the alerts of the conventional IDS and Table 5.9 the alerts using the EDS.

signature	rating
SPYWARE-PUT Trackware funwebproducts mywebsearchtoolbar-funtools runtime detection	unsuc
SPYWARE-PUT Hijacker side find 1.0 runtime detection - hijacks search engine	
WEB-PHP Opt-X header.php remote file include attempt	unsuc
BACKDOOR exception 1.0 runtime detection - intial connection server-to-client	
VIRUS Possible Sober virus set one NTP time check attempt	unsuc
INFO psyBNC access	
WEB-PHP Opt-X header.php remote file include attempt	unsuc
WEB-PHP phpBB viewtopic double URL encoding attempt	unsuc

Table 5.7: Alerts of the prototype

signature	rating
MISC MS Terminal server request	unsuc
ICMP PING NMAP	for
DNS SPOOF query response with TTL of 1 min. and no authority	unsuc
WEB-PHP remote include path	unsuc
VIRUS Possible Sober virus set one NTP time check attempt	unsuc
ICMP redirect host	for
SQL probe response overflow attempt	unsuc
WEB-CLIENT IE JPEG heap overflow single packet attempt	unsuc
SCAN myscan	for
ICMP Source Quench	for
WEB-MISC IBM Lotus Domino Web Server Accept-Language header buffer overflow attempt	unsuc
ICMP Destination Unreachable Communication with Destination Network is Administratively Prohibited	for
WEB-CLIENT Adobe BMP image handler buffer overflow attempt	unsuc
WEB-IIS view source via translate header	fp
WEB-MISC SSLv3 invalid data version attempt	unsuc
WEB-CLIENT winhelp clsid attempt	unsuc
SPYWARE-PUT Hijacker side find 1.0 runtime detection - hijacks search engine	
WEB-MISC backup access	unsuc
MYSQL yaSSL SSLv2 Client Hello Message Challenge Buffer Overflow attempt	unsuc
MYSQL yaSSL SSLv2 Client Hello Message Session ID Buffer Overflow attempt	unsuc
WEB-MISC Compaq Insight directory traversal	unsuc
INFO psyBNC access	
MYSQL yaSSL SSLv2 Client Hello Message Cipher Length Buffer Overflow attempt	unsuc
SPYWARE-PUT Trackware funwebproducts mywebsearchtoolbar-funtools runtime detection	fp/unsuc
MYSQL 4.0 root login attempt	unsuc
WEB-PHP xmlrpc.php post attempt	unsuc
WEB-PHP Opt-X header.php remote file include attempt	unsuc
WEB-MISC Compaq web-based management agent denial of service attempt	unsuc
BACKDOOR exception 1.0 runtime detection - intial connection server-to-client	
INFO FTP Bad login	unsuc
EXPLOIT RealVNC server authentication bypass attempt	unsuc
WEB-PHP phpBB viewtopic double URL encoding attempt	unsuc
WEB-CLIENT Excel malformed FBI record	unsuc
WEB-MISC cross site scripting attempt	fp
SPYWARE-PUT Trackware casalemedia runtime detection	unsuc
WEB-CLIENT Windows Media Player 7+ ActiveX Object Access	unsuc
WEB-PHP admin.php access	unsuc
WEB-CGI wrap access	unsuc
WEB-CLIENT Shell.Explorer ActiveX Object Access	unsuc

Table 5.8: Alerts of the conventional IDS

signature	rating
VIRUS Possible Sober virus set one NTP time check attempt	unsuc
SPYWARE-PUT Hijacker side find 1.0 runtime detection - hijacks search engine	
INFO FTP Bad login	unsuc
BACKDOOR exception 1.0 runtime detection - intial connection server-to-client	
INFO psyBNC access	

Table 5.9: Alerts of the EDS

5.2.3 Results

An overview of the rated alerts of the different system one can find in the Table 5.10. All systems detect the single attack assumed to be a real intrusion. It is an adware called Adware.FindFm.

The infected host contacts the IP of a typical adware server, which is reported in [66]. The EDS raised the fewest alerts. It contains only two alerts, which are probably false-positives. All other alerts need to have human intervention and access to the possibly infected host. The prototype has some more alerts than the EDS but could reduce the false-positive “INFO FTP Bad login” of the EDS. The additional alerts of the prototype are caused by the escalation rule for web application attacks. This rule produces needless alerts of unsuccessful attacks. Therefore, this rule is in this form not applicable and has to be refined. The IDS has the most alerts, but it has also the biggest ruleset. Anyway, the extraction of the single true alert out of more then 20 alerts is difficult.

	fp	**for**	**unsuc**		
IDS	3	5	29	2	1
EDS	0	0	2	2	1
Prototype	0	0	5	2	1

Table 5.10: Overview of the rated alerts

5.2.4 Conclusion

The evaluation of the prototype showed that the escalation ruleset has to be tuned in a better way. However, we think the prototype provides a good tool for tuning an IDS. The EDS has only few false-positives; if it really detects successful attacks, an EDS would be a better real-world solution than an IDS, because the raised alerts are manageable. The problem of this evaluation is the missing malware in the monitored network. If there is no malware, there is no possibility of the detection. Therefore, the prototype and the EDS could not be sufficiently tested for false-negatives.

5.3 Conclusion of the Evaluation

The evaluation of an EDS, which is using the outbound traffic, and the evaluation of the prototype showed that to focus on the outbound traffic reduces the alerts remarkably. Fortunately, but in this case unfortunately, a network including many of infected hosts is rare. Therefore, representative testing is difficult. With the utmost probability, all reported alerts are false-positives or unsuccessful attacks (except for the one detected adware). Therefore, a system with no alert would be the best, which is easy to achieve by simple ignoring all alerts. An evaluation of an IDS is only representative if both sides, the false alarms and the false-negatives, are evaluated carefully. A network containing infected hosts is necessary for such a representative evaluation; otherwise, an evaluation with well-founded results and conclusions is not possible.

Chapter 6

Summary, Conclusion and Outlook

6.1 Summary and Conclusion

In order to make an IDS suitable for the real world, we need to reduce its number of false alarms to a manageable amount. It goes without saying that an IDS also has to detect successful attacks. Therefore, this thesis focuses on the detection of successful attacks with a performed intrusion since all other alerts are uninteresting for a production system. Having this goal in sight, we have analysed the “malicious outbound traffic” of infected hosts. Our classification of “malicious outbound traffic” includes the following four traffic categories: attack response traffic, command and control traffic, propagation traffic and commanded traffic.
Applying this classification, we have analysed five different recent worms. The analysis confirms the assumption that all these worms generate at least one type of “malicious outbound traffic” as a result of an occurred infection. Therefore, a detection using an EDS should be possible.
Based on the open source project Snort, we have developed a prototype which monitors first and foremost the outbound traffic and should allow to detect successful attack more easily. In a second step, we extended this system with the application of a behaviour-based approach, the correlation of different events and properties. The prototype implements features which can be used to tune an IDS. Some examples are the correlation of single signatures, the checking of content from different packets or the frequency of the occurrence of particular signatures.
On the one hand, we have evaluated an EDS which only monitors the outbound traffic and, on the other hand, the developed prototype. In order to evaluate the EDS, we have replayed captured files. The evaluation showed a remarkable reduction of the false-positives by 80 percentages in comparison to an IDS which monitors the whole traffic. This result has to be handled with care since there are many more active inbound rules than active outbound rules in the used ruleset. The qualitative evaluation of the raised alerts has showed that most of the outbound alerts are of the signature classification “policy-violation” and, therefore, it depends on the policy of a company to benchmark the validity of such an alert.
We also got some false-negative alerts. All of them were based on the missing reaction of the infected host. This means the dump file contains indeed a successful attack, but no response traffic to the intrusion. In this case, the detection failed in spite of a successful attack.

For the evaluation of the prototype, real-life traffic of a university network has been used. A comparison with the prototype, a conventional IDS and an EDS has been made, analysing the raised alerts from the different systems. The reduction of the false-positives of the EDS and the prototype in contrast to the conventional IDS is remarkable. The alerts of the prototype and the EDS are quite similar as was expected. Both use the outbound traffic as a baseline for the alerting.

6.2 Outlook

The developed prototype uses a hybrid approach and contains behaviour-based aspects. A good tuning of the ruleset and the escalation rules (see chapter 4.3.1) are very important for a useful IDS. Currently, the tuning is done manually. The actuation including the practicable use

of the prototype is pending. Questions about the generation and maintenance of the escalation rule have to be clarified.

An advanced study of anomaly detection might be useful to find additional possibilities to define attributes or tables. One possibility might be to change the attributes “dt” (time difference) and “count.” Currently, the prototype computes the frequency of some alerts and raises an alert if this frequency is too high. It might be useful to replace these absolute values using relative ones (e.g. a mean instead of any value). This relative value would change according to the development of the network and, therefore, not only detect a high value but a real anomaly.
This anomaly-based approach might be helpful for a detection of sophisticated threats like covert channels.

The prototype implementation is based on an approach using escalation rules with different weights. The applied rule with the highest weight determines the treatment of the corresponding alert. A scoring system would be another approach. Every alert would collect points corresponding to particular properties, which we defined in the escalation_rule table. An alert would be escalated, if a calculated point is exceeded. It might be quite interesting to implement this approach and compare it to the prototype.

There was no time for a performance analysis. It would be interesting to test the performance of the Snort IDS in general. The measurement of dropped packets is important for the reliability of the IDS. Dropped packets will not be analysed from the IDS and, therefore, a dropped attack will remain undetected.
The IDS enabled the feature “tagged packets”. This feature records a certain amount of packets after particular alerts. A test of the performance using these tagged packets might also be interesting. The rule performance and the preprocessor performance of an IDS can be tested as well.

Appendix A

The Escalation Rules

Figure A.1: The implemented escalation rules

id	class_id	sig_id	dir	pid_eq_dir	pid_op_dir	action	weight	count	dt	hostorsubnet	netmask	comment
1			0			0	0					no escalation of inbound traffic
2			1			1	0					escalation of outbound traffic
3			2			1	0					escalation of intra traffic
5		153	0			1	100	10	00:00:01			escalation of inbound FTP Bad login
4		153	1			0	10					no escalation of FTP Bad login brute force attacks
6		153	2			0	10					no escalation of a single intra FTP bad login
7		153	2			1	100	10	00:00:01			escalation of FTP Bad login brute force attacks intra
8	13		0		1	0	100					no escalation of unsuccessful web app attack
12	12		1			0	10					no escalation of policy rules
13	13		0			1	10					web app attack without failed answer