Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Candid Wüest

Desktop Firewalls and Intrusion Detection

Diploma Thesis DA-2003.22
Winter Term 2002/2003

Tutors:
Diego Zamboni (IBM Rüschlikon Res.Lab),
Marc Rennhard

Supervisor:
Prof. Dr. Bernhard Plattner

28.2.2002

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Abstract

The use of desktop firewalls has become more and more popular these days. The goal of this thesis is to analyze the use of desktop firewalls in detail, to get a better understanding of their capabilities, strengths and weaknesses. Because they are installed on end-user machines together with other applications, the question arises whether they can add security or whether they open new security holes on the controlled machine. An interesting idea is to analyze the cooperation of multiple instances of desktop firewalls with intrusion detection systems (IDSs). Therefore, rules for detecting attacks with the log files gathered will be established and a generic desktop firewall log file format will be introduced that enables to correlate the corresponding events from the different log files.

Zusammenfassung

Das Benutzen von Desktop Firewalls ist heutzutage immer mehr in Mode gekommen. Ziel dieser Arbeit ist es, Desktop Firewalls genauer zu untersuchen und zu verstehen, wo ihre Möglichkeiten, Stärken und Schwächen liegen. Da sie auf den Computern der Endbenutzern zusammen mit anderer Software installiert werden, soll auch die Frage beantwortet werden, ob sie ein Gewinn an Sicherheit für das System darstellen oder ob sie neue Sicherheitslöcher öffnen. Spezielles Augenmerk wird darauf gerichtet, ob sie beim Zusammenspiel mit Intrusion Detection Systemen (IDSs) und anderen Instanzen von Desktop Firewalls einen Mehrwert an Information generieren können. Dazu werden auch Methoden zum Detektieren von Attacken mit den Desktop Firewall Logdateien eingeführt und ein generisches Logdatenformat wird vorgestellt, welches ermöglicht die zugehörigen Logereignisse zu korrelieren.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

It has become common to have an Internet connection at every location. Often it is running during day and night. Even at home, many people have a PC that is connected to the Internet 24 hours a day. During these long time periods, the machines are exposed to the threats of the Internet. The longer a machine is connected, the more likely it is to be affected by malicious activity. New technology like wireless networks increases even more the risk of getting attacked, as they introduce new possibilities of getting attacked. With all the recent events of computer viruses and worms spreading through the Internet and causing damage, users slowly but surely realize that there is danger in the Internet and that they need some sort of protection against it [7].

In small companies the level of security is often not that high and even in big enterprises with good security policies, it is not unusual that employees take their laptops from their office back home and connect them to their local Internet connections. At this moment, all the data that was protected trough the enterprise security measures before is exposed to a possible attacker. After a successful break-in into this machine, it can, when brought back into the corporation's network, infect other machines on the Intranet. Therefore, people have started using desktop firewalls to protect themselves against attackers. Having an anti-virus product installed is nothing unusual today, but having a desktop firewall running is still not common.

In contrast to the traditional model where firewalls are recommended to be installed on a dedicated and well-controlled machine, desktop firewalls are installed on end-user machines, together with a wide variety of tools and applications. Therefore the question arises whether they provide additional security or whether they are also vulnerable to new attacks and decrease the overall security. To provide more security the log files of desktop firewalls should be analyzed and considered when monitoring a network for attacks. There has been no effort yet in the direction of correlating the logged events of desktop firewalls, and also good detailed tests on the capabilities of desktop firewalls are rare [1]. The available test results are normally consumer oriented and do not go into much detail [4].

This thesis is not intended to be another comparison study of desktop firewalls regarding their ease of use or how nice they look. Rather it should as a first phase provide information about the concept of desktop firewalls in general and whether using them to protect user machines makes sense from a security point of view. In a second phase, the thesis provides mechanisms to correlate the generated events, making it possible to integrate the desktop firewalls into a security framework. Until now, the logged data is nearly always discarded and not further used in any way. Only if there occurred a problem that needs further investigation this log files might be analyzed. Thinking of all the possibly valuable logged information that is unused is enough motivation to take a closer look at the further use of this information.

1.2 Related work

By the time of writing this thesis, I was not aware of any work that is directly related to the idea of this topic. Most of the work on correlating log files targets only network firewalls and not desktop firewalls themselves [12]. We can find some log file analyzers that are able to process desktop firewall log files. A good example for this is Symantec's *Deep Sight* tool, also used at Security Focus, which is able to look through some desktop firewalls log files, for example Zonelabs Zonealarm [2]. *Dshield* is another tool that lets users submit their log files for collective examination [6]. Normally they are limited to a few widely-used desktop firewalls and do not support many different versions of them. Another problem with these tools is that the users will often not get a detailed analysis of the attacks on their home network. The information is just added to a pool and analyzed over all the events of all the submitters.

If we already have a security infrastructure, then we probably want to be able to further process the events with all their relevant information on our own. Submitting data to others might also not be compliant with security policies. Tools like *Dshield* cover more or less just the idea of sending the log files to a third party, which will then generate statistics, such as the ports scanned most frequently, but they will not provide a detailed report of the attacks that have been run against the machines. In addition we are not able to integrate the results easily into the correlation engine for the decision process.

There are a few short articles about the topic of this thesis, pointing out the idea of using desktop firewalls as cheap sensors in an intrusion detection network. One of them by Marcus J. Ranum [1], but it does not go into details on how it would be implemented and what could be gained out of using desktop firewalls as intrusion detection sensors. It only mentions that it would probably make sense to do so.

On the other side, there have been plenty of examinations of desktop firewalls and nearly every PC user magazine has published a review of a desktop firewall examination. The problem with those articles is that they are mainly dedicated to help the end-user chose a product that is easy to manage. This means the focus lies seldom on the core technology and the real capabilities of the desktop firewalls but rather on how nice the graphical user interface (GUI) looks or how fast the support hotline is. These kind of tests are valuable for the consumer but still missing are tests that analyze the concept behind the curtain. One of the more technical examinations of desktop firewalls can be found at Boran's website [3]. Some websites have done generic tests of desktop firewalls with so called "leak test tools" to see if it is possible to send out information from a protected system. Many of them are very successful and claim that desktop firewalls are unsafe and useless. Unfortunately some of them used attacks, from which the desktop firewall was never meant to protect, and thus failed. The charges of failing to protect from certain attacks are investigated later in Chapter 5. Therefore, some of the applied techniques have been taken into account in the examination part of this thesis. The mentioned websites can be found at PC flank [5] or in the article from G. Bahadur [4].

1.3 Desktop firewalls

1.3.1 Desktop firewall technology

For quite a long time people in the IT security field have been using network firewalls to secure their networks. Network firewalls can be a hardware device or a software program running on a dedicated machine. In either case, it must have at least two network interfaces, one for the network it is intended to protect, and one for the network it is exposed to. A network firewall sits at the junction point or gateway between the two networks, usually a private network and a public network such as the Internet. The earliest computer firewalls were simple routers. All traffic that is passing the network firewall is inspected and if it fulfills defined criteria it will be allowed or otherwise blocked [17]. But for a normal computer user at home, a network firewall is usually too complex or too expensive. This is often the reason, why the common home users would not use a network firewall even if they wanted to protect themselves. This has led to the idea of desktop firewalls.

Desktop firewalls, sometimes also referred to as personal firewalls or distributed firewalls, target end users and do not try to replace network firewalls. They intend to be easy to setup and maintain. Most desktop firewalls are preconfigured so that the user is not bothered with difficult rule programming, and if later an unknown event occurs, a wizard will help in creating the necessary rules.

As the name implies, desktop firewalls are host-based applications and run on the users machine. Therefore they are only able to see traffic that is directed to the monitored machine. A desktop firewall is usually a stateless packet filter. Stateless means, the filter does not track the state of a connection, it decides for each packet on an individual basis to either permit it or block it [17]. On the other side, statefull filters do accept packets only if the are possible in the actual connection state. Therefore, a stateless packet filter is an application that can accept packets, examine their headers independent and then decide whether to forward them, depending on whether they meet the rules defined by the user. They work on the level of IP packets. Disadvantage of stateless inspection is that packets which would not be possible in a certain state of communication still may pass the desktop firewall. Desktop firewalls typically do not implement packet content filter. This means they do not inspect the payload of the packet to see if it consists of malicious content. Only the headers of the packets are inspected by desktop firewalls.

1.3.2 Reasons for desktop firewalls

The main purpose of a desktop firewall is to monitor and eventually block incoming and outgoing network traffic on a machine according to the wishes of the user. This is normally done by applying some filter rules to the packets. So far the behavior does not differ much from a standard network firewall, except that network firewalls filter the traffic dedicated to multiple machines and desktop firewalls check only the packets targeting one local machine. Desktop firewalls can additionally check the name of the application that opens the connection. This enables them to filter certain applications in the rules. This means that filtering rules not only may contain IP addresses but also may refer to trusted or untrusted application names, offering a finer tuning of the rules.

The purpose of a desktop firewall can be classified in two categories. First, it should protect from attacks that come from outside and are targeting the monitored machine. This includes attacks such as port scans, misuse of open daemons like network shares, and denial of service attacks. The second category are the attacks or threats originating from the inside of the system, such as a trojan horse server that tries to connect home or an *ad-ware* tool that wants to send some personal information back to its vendor. This should be blocked or at least monitored and alerted by the desktop firewall. Figure 1.1 shows the two concepts.

There are a lot of reasons why we should use a desktop firewall solution, even if we are considering it for a machine that is located behind a full-featured network firewall.

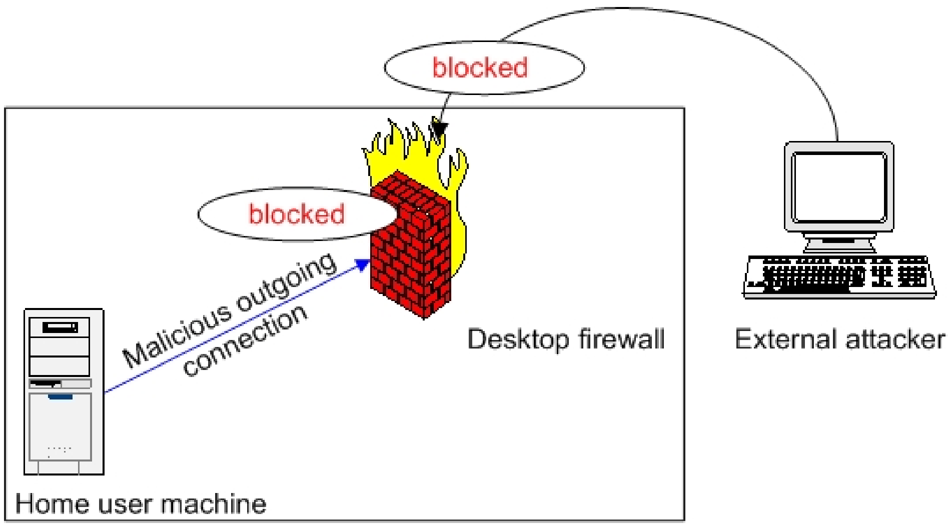


Figure 1.1: Filter schema of a desktop firewall

The first reason is employing a second line of defense to prevent attacks and to secure the system. Given the fact that desktop firewalls have a different configuration and also different vulnerabilities than network-based firewalls, an attacker would have to breach two security systems. Searching for two different exploits that work together decreases the chance of a successful attack. Furthermore, those external firewalls which normally are placed on the network border will not help if the attacker is coming from inside the network. According to several surveys most attacks come from the local network [13]. Even if we fully trust our co-workers, we can never be sure that their machines have not been broken into and are used against us.

Another benefit of personal firewalls is that it can make use of the local context. Furthermore, it has access to data such as the name of the application that opens a connection and not only the to IP address of the source machine. Taking this additional information into account, it is possible to make decisions on subtle distinctions. For example, assume that some *ad-ware* has managed to install itself on the machine and is trying to send some personal data back to a marketing company by HTTP transmission. The majority of external firewall products are configured to permit this traffic as it looks like legitimate HTTP traffic. On the other hand, a properly configured desktop firewall would at least alert us before sending the packets and tell us the name of the offending application. However not only with minor threats like *ad-ware* a desktop firewall is useful, in fact also with computer worms they help. As recent cases show, computer worms spread so fast that most users will not have a chance to update their anti-virus signatures to catch them. For example the computer worm SQL.Slammer (also known as Hellkern or Sapphire) doubled the number of infected machines every 9 seconds in the first minutes of its outbreak [8]. After 11 minutes it was spread all over the Internet. Desktop firewalls do a fairly good job in preventing those computer worms from spreading in the network as they block outgoing connections to the network for unknown applications. So if the worm would try to send itself out from an infected machine, it would be blocked by the desktop firewall and would be prevented from infecting other machines. Although in this scenario the infection of the first machine was not prevented, the danger is still stemmed.

Many home users think they are safe at home. They often argue: "Why should someone be interested in my data, it is nothing valuable and if I loose some of the data, I done really care." Even when agreeing with these points, it still makes sense to install a desktop firewall. Especially home users with a permanent Internet connection such as DSL users are more and more the target of attackers. They are not after secret data but they can use those systems as stepping stones to obfuscate their true origin while attacking other systems. For example these systems could be used as sources for denial of service flood attacks.

Most of the above mentioned scenarios could also be detected using a host-based intrusion detection system (IDS). The difference is that an IDS is normally used for detecting attacks and not for preventing or protecting against them. An IDS can tell if a trojan horse is trying to call home from the machine, but it may not block the traffic as it is not intended to do so. This means that IDSs are not a replacement for desktop firewalls. Still most desktop firewall vendors have started to include features to report known attacks that have been detected, similar to intrusion detection systems. So they present the user with additional information about the recorded events. Another added feature is content filtering, which enables the desktop firewall to control the information that is stored in cookies, blocking advertisements pop-ups or controlling the execution of active content like ActiveX in web pages on an individual basis for the user.

Finally, as this report will show, another reason for using desktop firewalls is to use them as an additional information source that could help in getting a better view of the overall security state. The log files provide information that can be used in the process of correlating alarms and responding to attacks.

1.4 Approach

The approach followed in this thesis is to first think of possible attack scenarios on different levels targeting the desktop firewalls. For these reasons a small test network is set up, with all the desktop firewall products to be tested running in parallel on identically configured machines. By running the chosen attack set we can verify if the desktop firewalls are able to protect against those specific attacks. This experiment gives an overview of the capabilities of desktop firewalls. The generated log files are collected for further inquiries. Analyzing the logged events leads to a generic event log format for desktop firewalls. Three Perl scripts are developed in this thesis which translate the log files into the generic event log format. In order to find correlation rules for these alarms the generated log files are analyzed. To verify the made conclusions a real world experiment is made with multiple desktop firewalls running in a network.

1.5 Outline

Chapter 2 introduces the chosen products and their features, including the logging capabilities.

The testbed and the preconditions that where made for the testing are explained in Chapter 3.

An overview of the attack scenarios used is given in Chapter 4, introducing the impacts of each attack.

Chapter 5 provides the results of the experiments and some direct conclusions.

In Chapter 6 the generic event log format is introduced with the corresponding mappings. Furthermore, it explains the correlation of log events with the generic event log format illustrating it with a real world experiment, as well as discussing the idea of cooperation of desktop firewalls.

The conclusions gained in this thesis are explained in Chapter 7.

An outlook for possible future work is given in Chapter 8.

Chapter 2

Specification of desktop firewalls

In this chapter, the products chosen for testing are introduced and their features are explained.

2.1 Selected products

Most Unix systems or Linux derivatives have already full featured firewalls included that has little in common with desktop firewalls on Windows systems. Therefore I decided to chose only Windows based desktop firewalls for my test series. The non-Windows desktop firewall would probably distort the result of the tests because they are implemented like normal network firewalls and lack some of the special features that are common among the other desktop firewalls. As the purpose of this thesis is to reflect the real user situation, products from the whole range of the market segment are chosen. The selected desktop firewall are the following:

Name:	Zonealarm
Version:	3.1.395
Vendor:	Zonelabs
Website:	`http://www.zonelabs.com`
Type:	freeware
Supported OS:	Win98/ME/NT/2K/XP

Name:	Symantec Desktop Firewall
Version:	2.01
Vendor:	Symantec
Website:	`http://www.symantec.com`
Type:	commercial
Supported OS:	Win95/98/ME/NT/2K/XP

Name:	Sygate Personal Firewall
Version:	5.0.1150
Vendor:	Sygate
Website:	http://soho.sygate.com
Type:	free for personal use
Supported OS:	Win95/98/ME/NT/2K/XP

They where chosen, because they represent a good overview of what can be found on the market, regarding the small office and home user market. If anywhere in this thesis one of the names above appears without further indication of version, then the specific versions, which are listed above, are assumed.

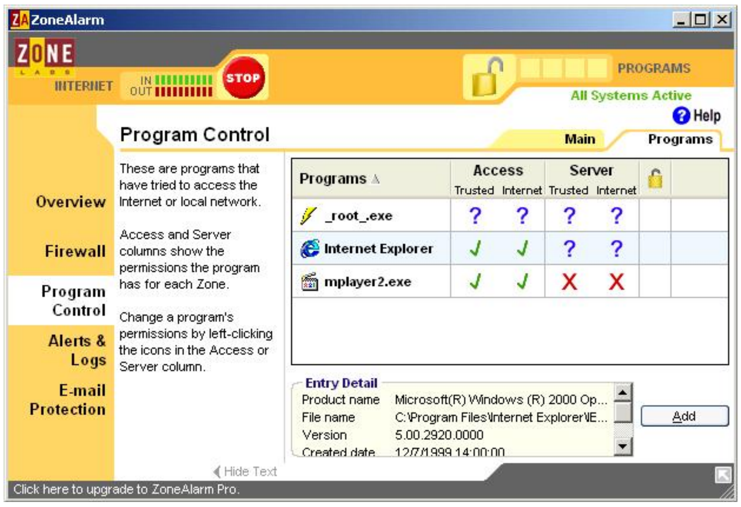
2.2 Features of desktop firewalls

In this Section some of the interesting features of the desktop firewalls chosen will be explained. The focus will be set on the logging and rule options, as they play the main rule in this thesis, but also other helpful features will be mentioned. This list is not meant to be complete, as some features like "live update" are not explained here.

2.2.1 Zonealarm

Zonelabs's desktop firewall provides two different zones of trust, the *Internet zone* and the *trusted zone*. For each zone the user can add IP addresses or networks. Both zones may have different security settings. The provided security can be one of three levels, either high, medium or low. In the program control settings we can define for each application the rights in the Internet zone and the rights in the trusted zone. Further we can specify if the application should be able to act as a server, making it possible for it to wait for incoming connections. The possible options are: allow, block or ask, as shown in Figure 2.1

A feature that is not related to the main desktop firewall but still interesting is "mailsafe". It is a basic protection from visual basic script viruses in emails. The idea is to move the script to a safe location and make sure, that it does not execute automatically. This is achieved by monitoring the email traffic. As an additional feature a panic button is implemented, that switches the desktop firewall into the "block all" mode when pressed. This can be useful if the user notices some abnormal behavior of the system. There is no possibility to add advanced custom rules, which use filtering options based on port numbers. This means each application will have the rights to use all ports, when allowed to communicate with the network.



Program Control

These are programs that have tried to access the Internet or local network.

Access and Server columns show the permissions the program has for each Zone.

Change a program's permissions by left-clicking the icons in the Access or Server column.

Programs	Access		Server		
	Trusted	Internet	Trusted	Internet	
root.exe	?	?	?	?	
Internet Explorer	✓	✓	?	?	
mplayer2.exe	✓	✓	X	X	

Entry Detail

Product name	Microsoft(R) Windows (R) 2000 Op...
File name	C:\Program Files\Internet Explorer\IE...
Version	5.00.2920.0000
Created date	12/7/1999 14:00:00

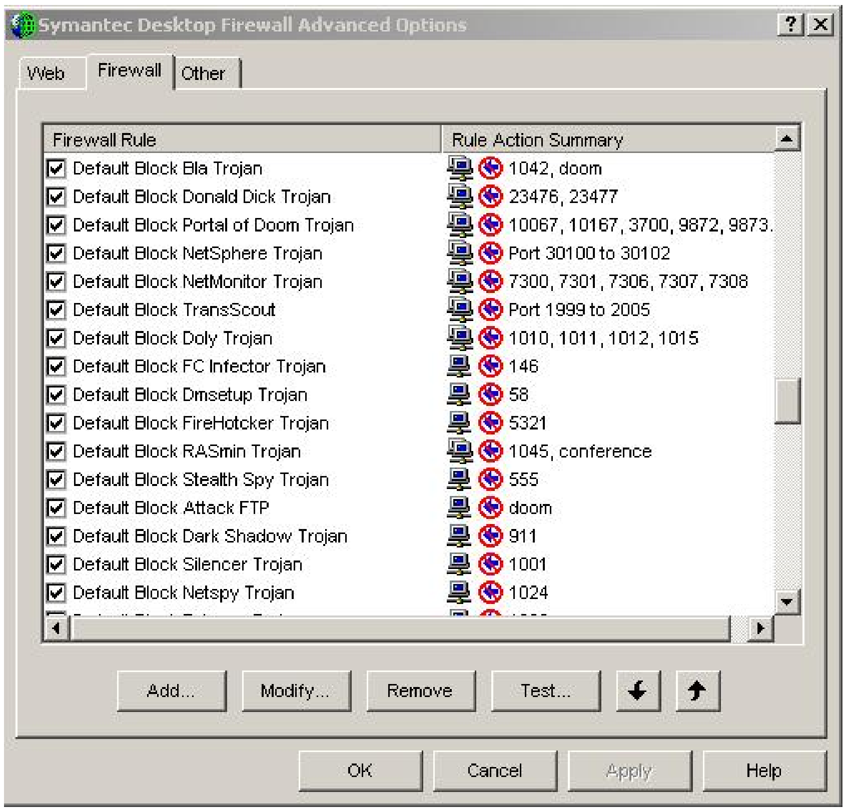
Figure 2.1: Zonealarm's application rule editor

2.2.2 Symantec desktop firewall

The Symantec desktop firewall has two main categories, security and privacy. In security the user has the option to select one of three predefined levels (high, medium or low). These three levels are settings for the internal options. They are split in three topics: desktop firewall, Java applets and ActiveX. For each of these groups we can once more choose one of three protection levels. For desktop firewall this means, option *high:* block everything until user allows it, *medium:* block known malicious applications or *none:* allow all traffic. The other two groups can be set to either block, prompt each time or allow the traffic.

Additionally there are two more options. One for enabling alerts which will be displayed in the task bar icon, and one for silently blocking unused ports. The latter means, that traffic which is directed to an unused port and not matched by any filtering rule will be blocked, as if there was a rule to do so.

The second category is privacy. A slider allows the user to set one of three predefined levels of security; high, medium or minimal. These are bound to the internal settings for cookie blocking (block, prompt or allow) and for confidential information blocking (block, prompt or allow). Confidential information can be any information that we enter in a custom field, for example the last 4 digits of the credit card number. If enabled, the desktop firewall attempts to make sure that this information does not leave the machine. Symantec desktop firewall has further the ability to block IGMP traffic and also to block fragmented IP packets. Both are often used in denial of service or nuke attacks, which try to crash the computer with intentionally badly crafted packets.


Symantec Desktop Firewall Advanced Options

Web | Firewall | Other

Firewall Rule	Rule Action Summary
✓ Default Block Bla Trojan	1042, doom
✓ Default Block Donald Dick Trojan	23476, 23477
✓ Default Block Portal of Doom Trojan	10067, 10167, 3700, 9872, 9873.
✓ Default Block NetSphere Trojan	Port 30100 to 30102
✓ Default Block NetMonitor Trojan	7300, 7301, 7306, 7307, 7308
✓ Default Block TransScout	Port 1999 to 2005
✓ Default Block Doly Trojan	1010, 1011, 1012, 1015
✓ Default Block FC Infector Trojan	146
✓ Default Block Dmsetup Trojan	58
✓ Default Block FireHotcker Trojan	5321
✓ Default Block RASmin Trojan	1045, conference
✓ Default Block Stealth Spy Trojan	555
✓ Default Block Attack FTP	doom
✓ Default Block Dark Shadow Trojan	911
✓ Default Block Silencer Trojan	1001
✓ Default Block Netspy Trojan	1024

Figure 2.2: Symantec's rule editor

Symantec desktop firewall offers the possibility to enable an automatic firewall rule creation wizard. This wizard tool has a internal database of known applications and corresponding default rules. If the desktop firewall notices a new application that wants to access the network, the automatic firewall rule creating wizard will look up the name in the database and check if it is a known application. When it is a known application, it will create the corresponding rules automatically, without asking us. This can lead to problems if we want to have non-standard rules for standard applications like web browsers.

In the advanced option section we are able to set some more privacy rules on a per domain name basis. This allows to block user-agent strings, cookies, referrer and email names from being submitted to websites.

Connections can be filtered by setting up rules in the rule editor shown in Figure 2.2. These rules can match incoming or outgoing packets and filter them according to criteria like: protocol, application name, port numbers or the IP addresses used.

The action on a filtered packet can be to allow it, block it or ignore it. Ignoring means, that the packet will be logged but not blocked. For special events an alarm flag can be raised, which will be displayed in the tray icon. A complete list of all the options in the rule editor can be found in the Appendix C.1.

2.2.3 Sygate personal firewall

Sygate personal firewall offers three different modes of operating levels, block all, allow all and normal.

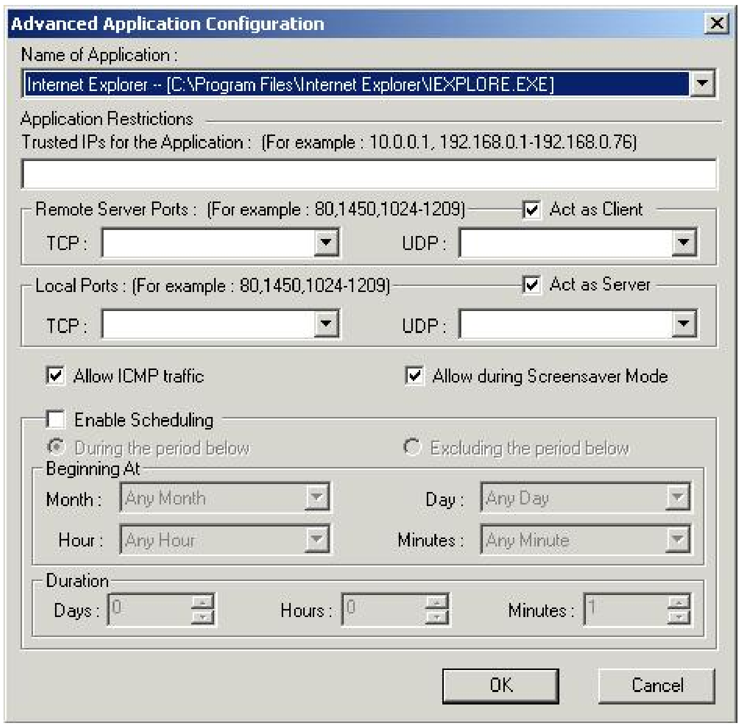
- **Block All**, means that all transmission incoming and outgoing are prevented.
- **Allow All**, means that all traffic will be allowed from and to the protected machine. Still it will write a log event if a rule was matched.
- **Normal**, means that custom defined filtering rules will be applied.

There is an application list, that remembers all the rights that we have given the applications, on a per-application basis. It has the following fields:

- **FileName:** The name of the application.
- **Version:** The version (build number) of the application.
- **Access:** The access status that was applied to this program (either Block, Ask, or Allow)
- **Path:** The location of the application.

For all applications it is also possible to create an advanced rule. Those rules can filter the traffic of the applications with additional fields like, IP addresses or port numbers. Each application can be granted the right to send or receive traffic during screensaver mode or during specified time periods. The detailed options for the advance application rules are listed in the Appendix C.2.

Further it is possible to define advanced rules for connections, as shown in Figure 2.3. Each rule can be set to filter packets according to the protocol, the IP addresses, the MAC addresses or the port numbers they use. All rules can be binded to multiple applications. The action on a filtered packet can be to allow it or to block it. A rule can be set to apply only during a specified time period or during screensaver mode. The detailed options for the advance rule editor are listed in the Appendix C.2.



Advanced Application Configuration

Name of Application : Internet Explorer -- [C:\Program Files\Internet Explorer\IEXPLORE.EXE]

Application Restrictions

Trusted IPs for the Application : (For example : 10.0.0.1, 192.168.0.1-192.168.0.76)

Remote Server Ports : (For example : 80,1450,1024-1209) [✓ Act as Client]

TCP : UDP :

Local Ports : (For example : 80,1450,1024-1209) [✓ Act as Server]

TCP : UDP :

[✓ Allow ICMP traffic] [✓ Allow during Screensaver Mode]

☐ Enable Scheduling

[✓ During the period below] Excluding the period below

Beginning At

Month : Any Month Day : Any Day

Hour : Any Hour Minutes : Any Minute

Duration

Days : 0 Hours : 0 Minutes : 1

Figure 2.3: Sygate's rule editor

Instead of logging to a file the personal firewall can inform about unusual events by sending an email. This can be done immediately when the alarm occurs, or regularly on a time interval basis.

Sygate personal firewall provides a driver-level protection. This feature blocks protocol stacks from accessing the network unless the user allows it. If some application installs its own protocol stack and tries to bypass the personal firewall, it will be detected.

The feature called "DLL authentication" provides a method to determine which DLLs are associated with an application and mark them in an internal table. All DLLs not marked will be blocked from accessing the network.

2.3 Logging capabilities

2.3.1 Zonealarm

Zonealarm provides the possibility of logging into a plain text file, in a straightforward format. It is not in real time because it goes through an internal buffer before getting flushed, but it is nearly real time because the delay is less than a second. Each event is reported on a separated line, containing the fields shown in Table 2.1, separated by commas. Table 2.2 shows a sample fragment of a Zonealarm log file.

Field name:	Example:	Content format:
Type	FWIN	FWIN | FWOUT | FWLOOP | LOCK | PE | ACCESS | FWROUTE | MS
Date	2002/11/26	YYYY/MM/DD
Time	11:38:02 +1 GMT	HH:MM:SS +n GMT
Source	192.168.0.66:3244	IP address:Port
Destination	192.168.0.10:80	IP address:Port
Transport	TCP(flags:s)	TCP(flags:x)|UDP|ICMP|IGMP|N/A

Table 2.1: Zonealarm log file format

```
FWIN,2002/11/26,13:21:22 +1:00 GMT,10.10.50.42:63348,10.10.50.2:17978,TCP (flags:S)

FWIN,2002/11/26,14:07:22 +1:00 GMT,192.168.0.9:0,10.10.50.2:0,ICMP (type:8/subtype:0)

FWIN,2002/11/26,14:22:00 +1:00 GMT,10.10.50.4:0,10.10.50.2:0,IGMP

FWIN,2002/11/25,13:41:36 +1:00 GMT,192.168.0.9:0,10.10.50.2:0,IGMP (type:2/subtype:31)

FWIN,2002/11/26,15:31:46 +1:00 GMT,192.168.0.9:7617,10.10.50.2:42,UDP

ACCESS,2002/11/28,14:05:58 +1:00 GMT,,N/A,N/A

ACCESS,2002/11/28,14:05:58 +1:00 GMT,flood.exe was temporarily blocked from connecting
to the Internet (10.10.50.3).,N/A,N/A

PE,2002/11/28,14:09:06 +1:00 GMT,flood.exe,10.10.50.3:0,N/A

PE,2002/11/28,10:10:46 +1:00 GMT,Internet Explorer,192.168.0.9:80,N/A

FWOUT,2002/11/28,14:09:08 +1:00 GMT,10.10.50.2:96,10.10.50.3:66,TCP (flags:S)

FWOUT,2002/11/28,14:09:36 +1:00 GMT,10.10.50.2:0,10.10.50.3:0,IGMP

FWOUT,2002/11/25,14:37:40 +1:00 GMT,10.10.50.2:53,10.10.50.2:53,UDP

FWROUTE,2002/11/28,14:09:58 +1:00 GMT,0.0.0.0:96,10.10.50.3:66,UDP

FWROUTE,2002/11/28,14:11:28 +1:00 GMT,0.0.0.0:0,10.10.50.3:0,ICMP (type:8/subtype:0)
```

Table 2.2: Example Zonealarm log

2.3.2 Symantec desktop firewall

Symantec desktop firewall logs six different things in separate log files, which will be explained later. The internal representation is encoded in hexadecimal and uses a proprietary format. Unfortunately Symantec does not provide any explanation about the format of these files. There exists an option to export the log files to plain text, but there is some information discarded from the original log file, like the sub type of the protocol. The following sample extracts are all taken from the exported log files and therefore are in plain text. The six different log files are explained in the next sub chapters.

Content blocking log

Stored in the file *iamtdi.log*. This file stores information about blocked ActiveX or Java applets. The user has the option to enable this filtering in three different levels: low, medium and high. This feature is not directly relevant for the experiments of this thesis, because the events do not contain information about intrusion attempts. Therefore this log file was not further inspected.

Connections log

Stored in the file *iamtcp.log*. This log file reports all incoming and outgoing connections including ports, time stamp and number of bytes sent. Especially the last information could be interesting, for example, for checking if an attack was successful or not. This logging is not influenced by the filter rules.

Table 2.3 shows an example of Symantec's desktop firewall connection log.

```
1/28/2003 18:00:48 Connection:  192.168.0.33: http  from  192.168.0.2: 1802,
537 bytes sent,  1380 bytes received,  3:35.268 elapsed time

1/28/2003 18:00:12 Connection:  192.168.0.66: http  from  192.168.0.2: radius,
3162 bytes sent,  8352 bytes received,  21.451 elapsed time

1/28/2003 17:59:54 Connection:  192.168.0.66: http  from  192.168.0.63: radacct,
503 bytes sent,  221 bytes received,  0.650 elapsed time
```

Table 2.3: Example Symantec connection log

Firewall log

Stored in the file *iamfw.log*. This is the main desktop firewall log file. It records all the incoming and outgoing connections as specified with the filter rules. Each event is reported on multiple lines. As observed the number of lines can vary from three to six. As Table 2.4 shows, the event format does not follow a strict formatting rule. This fact makes it a bit elaborate to parse the events into the generic event log format which is introduced in Chapter 6.

```
11/25/2002 14:39:05 Rule "Default Outbound ICMP" permitted (10.10.50.4,systat).  Details:
Outbound ICMP request
Local address is (10.10.50.1)
Remote address is (10.10.50.4)
Message type is "Time Exceeded for a Datagram"
Process name is "N/A"

11/26/2002 14:14:30 Blocked inbound IGMP packet.  Details:
Remote address (10.10.50.4)
Local address (10.10.50.1)

11/27/2002 16:39:53 Rule "block all" blocked (10.10.50.1,441).  Details:
Inbound TCP connection
Local address,service is (10.10.50.1,441)
Remote address,service is (10.10.50.4,44614)
Process name is "N/A"

1/27/2003 14:08:45 Rule "Eudora HTTP" blocked (192.168.0.7,http).  Details:
Outbound TCP connection
Local address,service is (0.0.0.0,2683)
Remote address,service is (192.168.0.7,http)
Process name is "C:\Program Files\Qualcomm\Eudora\Eudora.exe"

11/26/2002 15:24:28 Rule "block all" blocked (10.10.50.1,nameserver).  Details:
Inbound UDP packet
Local address,service is (10.10.50.1,nameserver)
Remote address,service is (192.168.0.2,15897)
Process name is "N/A"
```

Table 2.4: Example Symantec firewall log

Privacy log

Stored in the file *iampriv.log*. This file will record all privacy-related events. For example the sending of cookies or of the browser's user-agent identifier. Because this data is not directly related to intrusion attacks, this log file will not be further analyzed.

System log

Stored in the file *iamsys.log*. The system log records all operational messages, such as starting and stopping of the service. In the scope of the experiments of this thesis, these alarms where not further analyzed, as they have no direct impact. In future work it would make sense to include this log file for completeness. In some scenarios it could be of interest to have this information. For example, when an attack has successfully shutdown the desktop firewall, even if the attacker should not be able to do so.

Web history log

Stored in the file *iamwebh.log*. Similar to a web browsers history file this log file records all the visited URLs with time and date. Unless we want to keep track of people who visit blocked web pages, this file will not be of interest for the further experiments and therefore is not analyzed.

2.3.3 Sygate personal firewall

Sygate logs events into four different log files: system log, security log, traffic log and packet log. The log files itself are encoded in hexadecimal, but there is an option to export it into plain text messages from the log view console. Additionally there is a debug log file named *debug.log* in the same folder, which contains information like, when the GUI was started or which driver where loaded with the desktop firewall. As the name implies, it is just for debugging reason and was not used in the experiments of this thesis.

All of these log files can be displayed in two different modes, *local view* or *source view*. The only difference is, that they call two fields remote host & local IP in local mode and destination host & source IP in the other mode. Actually I could not figure out why this feature was implemented, since the information stays the same. It is just that these two fields are swapped. A normal end user might get irritated and confused by this option. To simplify matters I have made the tests always using the source view mode.

System log

Stored in the file *syslog.log*. The system log records all operational changes, such as the starting and stopping of services, detection of network applications, software configuration modifications, and software execution errors. The system log is especially useful for troubleshooting but is not used for correlation aspect. In future work it would make sense to include this log file for completeness. In some scenarios it could make sense to have this kind of information, like when the service started and stopped. For the scope of this thesis, this aspect was not included. Table 2.5 shows an extract of an example Sygate system log file.

```
*************** Windows Version info ***************
Operating System: Windows 2000 (5.0.2195 Service Pack 2)
*************** Network  info ***************
No.0  "Local Area Connection"  00-04-ac-44-ab-ba  "Intel 8255x-based PCI Ethernet
Adapter (10/100)" 10.10.50.3

96 01/22/2003 16:00:38 Information 12070202 Start Sygate Personal Firewall...
97 01/22/2003 16:00:38 Information 12070202 Sygate Personal Firewall has been started.
98 01/22/2003 16:00:38 Information 12070305 Security level has been changed to Normal
99 01/22/2003 16:00:54 Information 12070305 New Option Settings is applied
100 01/22/2003 16:01:14 Information 12070305 New Advance rule has been applied
101 01/22/2003 16:01:20 Information 12070204 Stopping Sygate Personal Firewall....
102 01/22/2003 16:01:24 Information 12070204 Sygate Personal Firewall is stopped
103 01/22/2003 17:02:08 Information 12070201 Sygate Personal Firewall 5.0.1150
```

Table 2.5: Example Sygate system log

Security log

Stored in the file *seclog.log*. The security log records potentially threatening activities directed toward the machine, for example port scanning or denial of service attacks. The security log is like a simple IDS console that lists events that have been generated from the traffic log. Table 2.6 shows an extract of an example Sygate security log file, and Table 2.7 shows the description of the fields in the security log file.

Traffic log

Stored in the file *tralog.log*. The traffic log records every packet information of traffic that enters or leaves a port on the monitored machine.

Table 2.8 shows the format of the traffic log entries, which can be verified by looking at the example traffic log files in Table 2.9.

The first time field (*Time*) is the time of when the alarm was generated, the second time field (*Begin Time*) is when the attack which is reported has started, and the third (*End Time*) is when the attack ended. This means that the time marks behave always like (Begin Time) $\leq$ (End Time) $\leq$ (Time).

```
113 11/28/2002 10:04:56 Executable File Change Denied Major Outgoing TCP 10.10.50.4
0.0.0.0 C:\Program Files\Internet Explorer\IEXPLORE.EXE 1 11/28/2002 10:04:52
11/28/2002 10:04:52

114 11/28/2002 10:05:19 Executable File Change Accepted Information Outgoing
TCP 10.10.50.4 0.0.0.0 C:\Program Files\Internet Explorer\IEXPLORE.EXE 1
11/28/2002 10:05:15 11/28/2002 10:05:15

56 11/26/2002 14:02:59 Denial of Service Major Incoming ICMP 192.168.0.8 10.10.50.3 1
11/26/2002 14:02:52 11/26/2002 14:02:52

58 11/26/2002 14:45:34 Denial of Service Major Incoming Unknown 10.10.50.4 10.10.50.3 7
11/26/2002 14:45:32 11/26/2002 14:45:34

105 11/27/2002 15:55:24 Port Scan Minor Incoming TCP 10.10.50.4 10.10.50.3 6
11/27/2002 15:55:18 11/27/2002 15:55:19
```

Table 2.6: Example Sygate security log

Filed name:	**Description:**
Event number	Events consecutively numbered.
Time	The date and time when the event was logged.
Information	Message of the alarm.
Severity	Severity of the alarm (Critical, Major, Minor or Information.
Direction	Direction from the context of the User.
Protocol	Type of protocol (TCP,UDP or ICMP).
Destinationn	IP address of the machine being attacked.
Source	IP address of the machine the traffic was coming from.
Dest. Port or ICMP	The destination port number or the type of the ICMP traffic that was sent.
Src. Port or ICMP	The source port number or the type of the ICMP traffic that was sent.
Count	Number of events of this type that where logged.
Application	The name of the application that was involved.
Begin Time	The time that the attack attempt began.
End Time	The time that the attack attempt ended.

Table 2.7: Sygate security log schema

Packet log

Stored in the file *(rawlog.log)*. The packet log captures every packet of data that enters or leaves the computer. It can be though of as a network sniffer that is not

Field name:	Description:
Event number	Events consecutively numbered.
Time	The date and time when the event was logged.
Action	Action taken by the desktop firewall (Blocked or Allowed)
Protocol	The type of protocol used in the attempt (TCP, UDP, ICMP or Unknown)
Direction	Direction from the context of the User (Incoming or Outgoing)
Source	IP address of the machine the traffic was sent from
Destination	IP address of the machine attacked
Application	Name of the application that was involved
Count	Number of events of this type that where logged
Begin Time	The time that the attack attempt began
End Time	The time that the attack attempt ended

Table 2.8: Sygate traffic log schema

in promiscuous mode. This logging option is turned off by default, because it may consume a lot of disk space.

Unfortunately the exported packet log does not contain the raw packet, but instead just the information that is already provided in the traffic log, like source and destination IP address for example. To extract the packet content we would have to write our own parser for their format, which is not publicly available. Once having the packet information from the raw log files, we would have to create matching rules for all the possible attack events. This is just the raw packet information, without any pattern matching rules applied. To check if this packets are malicious,

```
56 11/25/2002 14:22:54 Blocked UDP Incoming 127.0.0.1 53 10.10.50.3 53 2
11/25/2002 14:22:32 11/25/2002 14:22:50 Block_all

57 11/25/2002 14:22:54 Allowed UDP Incoming 10.10.50.4 138 10.10.50.255 138
C:\WINNT\System32\ntoskrnl.exe 1 11/25/2002 14:22:52 11/25/2002 14:22:52
GUI%GUICONFIG#SRULE@NBENABLEYOU#ALLOW-UDP

30545 11/26/2002 14:01:10 Blocked ICMP Incoming 192.168.0.9 0 10.10.50.3 8 1
11/26/2002 14:00:51 11/26/2002 14:00:51 Block_all

1460410 11/28/2002 14:08:22 0.0.0.0 0 0.0.0.0 0 Outgoing Blocked
C:\sygate_DFW\exploits\flood\flood.exe
```

Table 2.9: Example Sygate traffic log

different filtering rules must be applied to the packet content. This is exactly the job that an IDS already can do. Reinventing the wheel does not make sense and would have exceeded the scope of this thesis. Therefore this log file, even though it is one of the best sources for information, was not further analyzed in this thesis, because in the exported format it provides the same information as the traffic log file.

Table 2.10 shows the constitution schema of the exported packet log file format. Some examples of the exported packet log file are shown in Table 2.11. As mentioned, these messages do not include the full packet because the full packet is only recorded in the internal representation in the *rawlog.log* file.

Field name:	**Description:**
Event number	Events consecutively numbered
Time	The date and time when the event was logged.
Source	IP address of the machine the traffic was sent from
SrcPort	The source port number
Destination	IP address of the machine attacked
DestPort	The destination port number
Direction	Direction from the context of the User (Incoming or Outgoing)
Action	Action taken by the desktop firewall (Blocked or Allowed)
Application	Name of the application that was involved

Table 2.10: Sygate packet log schema

```
1019023 01/22/2003 15:41:01 10.10.50.4 80 10.10.50.3 1038 Outgoing Allowed
C:\WINNT\system32\telnet.exe

1019034 01/22/2003 15:44:00 10.10.50.4 80 10.10.50.3 1040 Incoming Allowed
C:\Program Files\Internet Explorer\IEXPLORE.EXE

1019035 01/22/2003 15:44:07 10.10.50.2 138 10.10.50.255 138 Incoming Blocked
C:\WINNT\System32\ntoskrnl.exe
```

Table 2.11: Example Sygate packet log

Chapter 3

Test environment

This chapter explains the setup of the testbed used for running the attacks against the desktop firewalls to be tested and introduces the assumptions made and the goals of the experiments.

3.1 Goals of testing

In the Internet we can find some websites that provide firewall tests specific for personal firewalls [3]. All the found tests simply do a port scan of an IP address and report the open ports. This will only show misconfiguration problems, but as the presumption in this thesis is that desktop firewalls are well configured to their best possible level, we do not have to care about these problems.

Having open ports is not dangerous *per se*, as long as the service is meant to be accessible from the Internet. All personal firewalls will in the default installation block the incoming traffic and ask the user for each application if it should be allowed or not. So I have my doubts that these testing sites really help in checking the security of desktop firewalls. Somehow they make the end users think that they have a secure system, probably leaving other open holes undiscovered.

Therefore the idea of this thesis was to focus on design faults and holes in the concept of desktop firewalls, that could not be fixed by reconfiguring the desktop firewall. Some of the problems might be patched by the vendor in later versions, some might never get fixed, because they address difficulties that the desktop firewalls have not been intended to solve. But once analyzed and discovered, those loopholes can be avoided or secured with other mechanisms. So the focus was really set to find attacks that cover the whole range of possible scenarios, which will be discussed in Chapter 4.

A second goal of the testing process is of course to generate log files which later can be used to generate the correlation schema. For this purpose the desktop firewalls have been configured to log all the important information. With the help of a small batch script the generated log files have been moved to separate folders for each attack, to make later analysis easier.

3.2 Assumption

The emphasis of this thesis is the concept of desktop firewalls, therefore issues related to weak operating systems or attacks that are only possible because of a flawed configuration of the desktop firewall itself, are not covered.

Therefore it is assumed that the underlying operating system has been fully patched with all the necessary updates and is running stable. The desktop firewalls are assumed to run with a reasonable rule set, that results in the maximum of security that could possibly be achieved. For this the default configuration of each desktop firewall is adapted, according to its possibilities. All outgoing traffic is blocked except for the Internet explorer, which is granted access permissions for destination TCP port 80, 8080 and 443. These represent the standard web server port, an additional web server port 8080 and the SSL connection port 443. Incoming communications are only accepted for a telnet server on TCP port 21. This set up was chosen to represent real world conditions and not just block all traffic by default, because simply blocking all traffic is not a reasonable option. The logging process is set up to the possible maximum of details, so that no information is lost in this step.

3.3 Testbed

For testing purposes, four machines are set up in a small network, as shown in Figure 3.1. All of them are IBM PL300 desktop machines with Pentium II 350 MHz processors and 128 MB of RAM. Three machines are identically installed with Windows 2000 and service pack 2. On each system, one of the desktop firewalls to be tested is installed. The fourth machine is used as an attack machine and has a dual boot system for Windows 2000 with service pack 2 and RedHat Linux 8.0. All the machines are connected to an isolated 100 Mbit network through a Netgear HUB. On all Windows systems an administrator account and a restricted user is created.

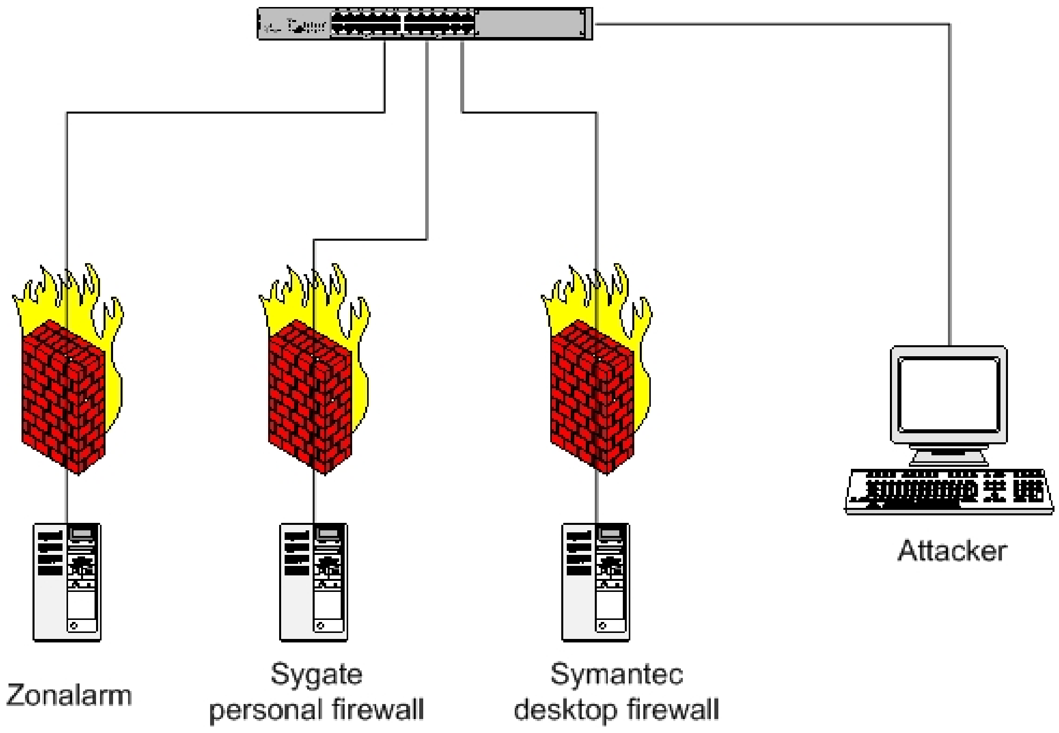


Figure 3.1: Testbed set up

Chapter 4

Attacks

In this chapter, the attack scenarios used will be explained in detail, showing the impact that they can have.

4.1 Description of attacks used

Since the idea of this thesis was to check the strengths and weaknesses of the desktop firewalls and not of the operating system, the focus of the attacks was targets the desktop firewall itself.

The attack set includes many ideas which require modifications on the local machine to work. For justifying this, we can think of a virus or a trojan horse that was executed on the target system and can now make the desired modifications. As normal anti virus tools mostly do their detection based on signatures, it is obvious that a new malware application will not be detected. This fact brings us to the conclusion that expecting some malware running unnoticed on our machine is not such a devious idea. Therefore, assuming that local modification have taken place for some attacks is not so absurd.

All used attacks, that I have not programmed myself, can be found for free in the Internet. I did not include them in this thesis paper, as it is against Swiss federal law to provide source code or detailed help about working exploits. I decided not to include the tools, because this work is not about the attack tools itself. Rather it is about showing that it would be possible, and illustrating the problems. A technical reader will be able to perform the same tests by using similar tools or program his own for this purpose.

Figure 4.1 shows the used attack tree from where the different scenarios were derived.

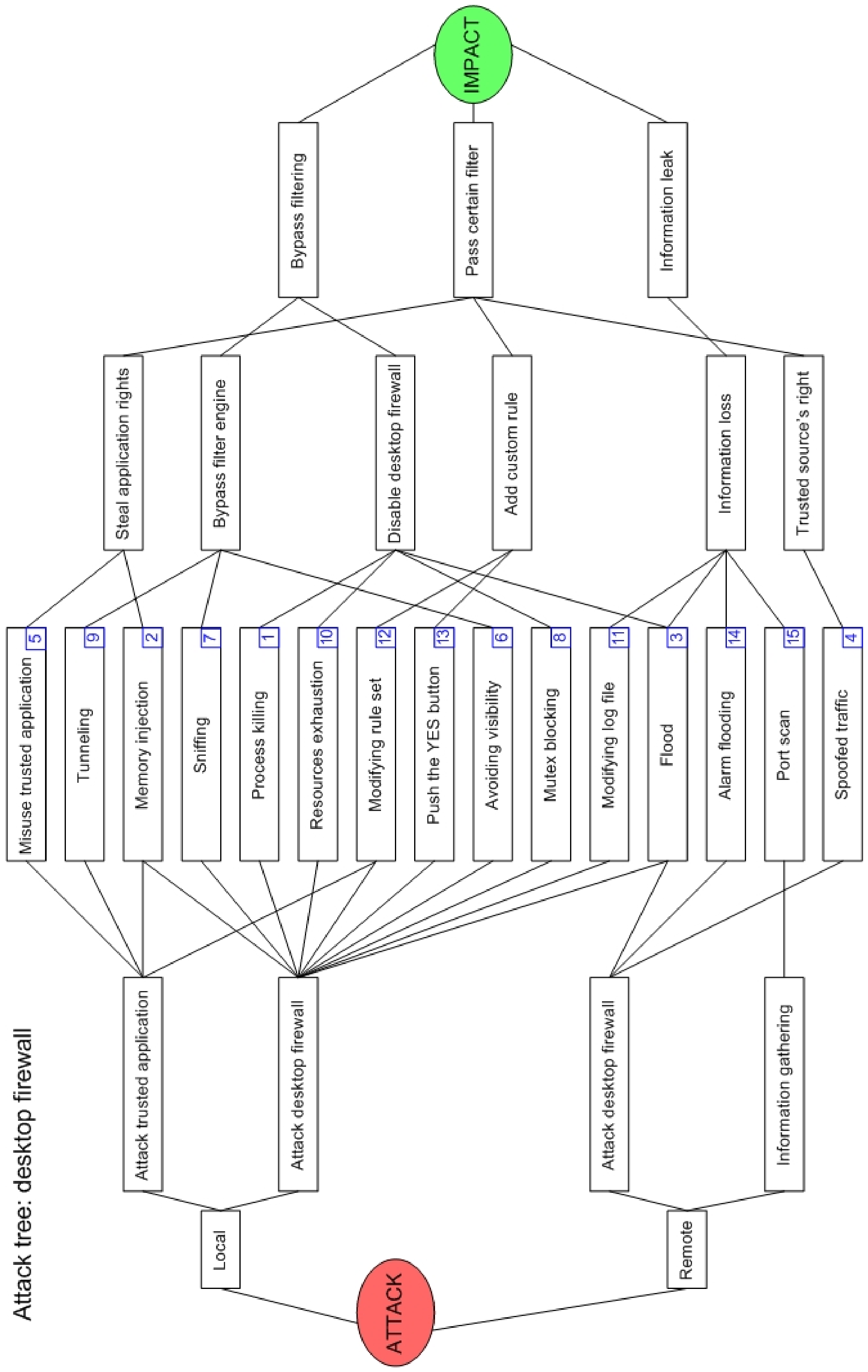


Figure 4.1: Attack tree: desktop firewall

We describe now the different groups of attacks, which will be used to test the desktop firewalls. This should explain the ideas of the attacks and also the impact that they could have.

4.1.1 Process killing

Description:	Locally try to stop the running desktop firewall process.
Required access:	Local
Idea:	Stop the desktop firewall process.
Impact:	Disabling the desktop firewall.
Leads to:	Bypass incoming and outgoing filter.
Variations:	Send termination message as administrator or as normal user.
Attack tree ref:	1

4.1.2 Memory injection 1

Description:	Inject a process into the memory space of the desktop firewall.
Required access:	Local
Idea:	Pretend to be part of the desktop firewall process.
Impact:	Use access rights of desktop firewall.
Leads to:	Bypass incoming and outgoing filter.
Variations:	Have a DLL loaded into the desktop firewalls memory space. Allocate memory and have a function executed in the desktop firewalls working memory.
Attack tree ref:	2

4.1.3 Information gathering

Description:	Do a port scan to gather information about the protected system.
Required access:	Remote
Idea:	Gain as much information as possible about the system for further attacks.
Impact:	Information leak.
Leads to:	Possible specific succeeding attack.
Variations:	Use special stealth scan techniques, like XMAS scan.
Attack tree ref:	13

4.1.4 Memory injection 2

Description:	Inject a process in the memory space of a trusted application.
Required access:	Local
Idea:	Pretend to be part of the trusted application.
Impact:	Filter of trusted application will be applied to traffic.
Leads to:	Bypass incoming and outgoing filter.
Variations:	Have a DLL loaded into this memory space. Allocate memory and have a function executed in the applications working memory.
Attack tree ref:	2

4.1.5 More info button

Description:	Try to catch the information that is sent out, when a user clicks on the *more info* button and is redirected to the vendors page.
Required access:	Remote
Idea:	Some desktop firewall send the version number and IP addresses to the website for getting additional informations.
Impact:	Information leak.
Leads to:	Possible specific succeeding attack.
Variations:	-
Attack tree ref:	14

4.1.6 Incoming flood

Description:	Send a huge amount of traffic to the desktop firewall.
Required access:	Remote
Idea:	Use all resources to temporary or permanently disable the desktop firewall.
Impact:	Disabling the desktop firewall.
Leads to:	Bypass incoming and outgoing filter.
Variations:	Use different protocols like TCP, UDP, ICMP or IGMP for flooding. Use special crafted packets like SYN or FYN packets. Vary the load of traffic. Use spoofed random source addresses. Use the same target and source address for the packets.
Attack tree ref:	3

4.1.7 Outgoing flood

Description:	Send a huge amount of traffic from the machine the desktop firewall is running on.
Required access:	Local
Idea:	Use all resources to temporary or permanently disable the desktop firewall.
Impact:	Disabling the desktop firewall.
Leads to:	Bypass incoming and outgoing filter.
Variations:	Use different protocols like TCP, UDP, ICMP or IGMP for flooding. Use special crafted packets like SYN or FYN packets. Vary the load of traffic. Use random target IP addresses and random spoofed source IP addresses. Use the same target and source address for the packets.
Attack tree ref:	3

4.1.8 Spoofed packets

Description:	Send special packets with 127.0.0.1 as source IP address.
Required access:	Remote
Idea:	Pretend traffic to come from trusted loopback device.
Impact:	Filter of trusted source will be applied to traffic.
Leads to:	Bypass incoming filter.
Variations:	Use different protocols like TCP, UDP, ICMP or IGMP. Use special crafted packets like SYN or FYN packets.
Attack tree ref:	4

4.1.9 Replacing a binary

Description:	Replace a trusted application with a malicious tool which has the same name and path.
Required access:	Local
Idea:	Impersonate the trusted application.
Impact:	Filter of trusted application will be applied to traffic.
Leads to:	Bypass incoming and outgoing filter.
Variations:	Replace the hash value of the trusted application which is stored by the desktop firewall for detection of misuse.
Attack tree ref:	5

4.1.10 Sniffing

Description:	Use a packet sniffer to receive traffic before it gets discarded by the desktop firewall.
Required access:	Local
Idea:	Receive traffic before it gets blocked later.
Impact:	No incoming filter will be applied to traffic.
Leads to:	Bypass incoming filter.
Variations:	Use different kind of network sniffers.
Attack tree ref:	7

4.1.11 Mutex blocking

Description:	Block the semaphore of the desktop firewall.
Required access:	Local
Idea:	Prevent desktop firewall from loading.
Impact:	Disabling the desktop firewall.
Leads to:	Bypass incoming and outgoing filter.
Variations:	-
Attack tree ref:	8

4.1.12 Tunneling

Description:	Use allowed protocols for communication, by hiding the real traffic in new packets.
Required access:	Local
Idea:	Repack traffic in a different protocol and use corresponding ports to send and receive traffic.
Impact:	Filter of trusted application will be applied to traffic.
Leads to:	Bypass incoming and outgoing filter.
Variations:	Use different protocols, for example HTTP, SMTP or ICMP to hide packets.
Attack tree ref:	9

4.1.13 Different IP stack

Description:	Use a different IP stack to send and receive packets.
Required access:	Local
Idea:	Desktop firewall does not see traffic.
Impact:	No filtering will be applied to traffic.
Leads to:	Bypass incoming and outgoing filter.
Variations:	Use windows VXDs to install a new network driver. Install a layered service provider to send traffic.
Attack tree ref:	6

4.1.14 Avoiding visibility

Description:	Make the process invisible for the desktop firewall.
Required access:	Local
Idea:	Use a kernel patch to hide the applications from API calls.
Impact:	No filtering will be applied.
Leads to:	Bypass outgoing filter.
Variations:	Use different *root-kits* for windows systems.
Attack tree ref:	6

4.1.15 Resource exhaustion

Description:	Consume all available resources on the protected machine.
Required access:	Local
Idea:	Consume all available resources to probably crash the desktop firewall, or at least temporary disable it.
Impact:	Disabling the desktop firewall.
Leads to:	Bypass incoming and outgoing filter.
Variations:	Try different resources like CPU time or memory.
Attack tree ref:	10

4.1.16 Modifying log file

Description:	Modify the log file of the desktop firewall.
Required access:	Local
Idea:	Remove traces after a successful attack.
Impact:	Fake log files events.
Leads to:	Misinformation.
Variations:	Add alarms of attack that have not taken place. Delete the complete log file.
Attack tree ref:	5

4.1.17 Modifying rule set

Description:	Modify the rule set of the desktop firewall.
Required access:	Local
Idea:	Edit the rule set, so that all traffic is allowed.
Impact:	Add custom rule.
Leads to:	Bypass incoming and outgoing filter.
Variations:	Add new specific rules or edit existing rules.
Attack tree ref:	15

4.1.18 Push the yes button

Description:	Write an application that automatically pushes the “yes” button of the rule creation wizard when a new rule should be created.
Required access:	Local
Idea:	Automatically create a rule for the malicious application.
Impact:	Add custom rule.
Leads to:	Bypass incoming and outgoing filter.
Variations:	Use wizard to create an *allow_all* rule.
Attack tree ref:	7

4.1.19 Alarm flood

Description:	Try to over flood the desktop firewall with alarm events.
Required access:	Remote
Idea:	To many alarms might overwrite older alarms in the log file.
Impact:	Information loss.
Leads to:	Unnoticed attack.
Variations:	Use different log file sizes.
Attack tree ref:	11

4.1.20 Misuse trusted application

Description:	Remote control a trusted application, having it communicating to the network. Use COM objects to open a browser window that is hidden. Let it access a special web page to exchange information.
Required access:	Local
Idea:	Pretend to be trusted application.
Impact:	Filter of trusted application will be applied to traffic.
Leads to:	Bypass outgoing filter.
Variations:	-
Attack tree ref:	12

Chapter 5

Results

This chapter presents the results found while testing the desktop firewalls, and explains occurred design problems of the desktop firewalls.

5.1 Results of attacks

5.1.1 Process killing

Description of attack: As a restricted user, try to terminate the running desktop firewall process. When successful, this will disable the desktop firewall and stop its services and thus offer full access to the Internet. If not successful try with administrator rights.
Tool used: Process Xplorer from Sysinternals [11].

Zonealarm:

Selecting the process *Zonealarm.exe* and sending a termination message will end the desktop firewall processes including the TrueVector service *VSmon.exe* which does the filtering. Although during the experiment sometimes the TrueVector service did not terminate correctly. Therefore it seems better to first stop the *VSmon.exe* and then stop *Zonealarm.exe* separately. Zonealarm will notice that TrueVector service is no longer running and will ask the user if it should be restarted, but if we kill zonealarm too, this does not matter.

When the desktop firewall was in *BLOCK_ALL* mode, also known as panic mode, then after killing the process no further network connection is possible. It seems that in this case a low level driver is injected in the network stack to prevent all communications. By terminating the process the protection in the network driver will not be removed, as it is not the process itself which blocks the traffic. It should be investigated what exactly gets installed by the process, to be able to see if it can be removed by an attacker.

Symantec Desktop Firewall:

Logged in as restricted user it is possible to kill the process named *IAMAPP.EXE*. Unfortunately this will only terminate the tray icon. Examining the start up shows that the first process starts another tool called *NISSERV.EXE*. This process is responsible for the traffic filtering. Therefore killing this process is what we want. During the experiment it was not possible to kill it as a restricted user, but it went to a state where it did not respond and did not perform filtering anymore. With administrator rights terminating the desktop firewall process was no problem.

If this is not an option, then one idea would be to remove the registry key at HKEY_LOCAL_MACHINE/SOFTWARE/Microsoft/Windows/CurrentVersion/Run → IAMAPP disabling the autostart of the desktop firewall. Next time the system restarts the desktop firewall will not be loaded.

Sygate Personal Firewall:

Even when being logged on as administrator, it is not possible to terminate the *SCM.EXE* process of the desktop firewall directly. It includes several threads that constantly check for termination messages and try to block them or restart the service. This is a clever idea. Thus it is not possible to terminate the process directly, but indirect it is possible. If we list all open handles of the desktop firewall process, we can see one that is named \Default and is of type *Desktop*. If we close this handle, then soon a Dr. Watson message will appear and tell that the process named *smc.exe* has generated errors and will stop now. Even when we do not click on the "*OK*" button of this message, after 15 seconds (on a idle system), the process will terminate and leave the system unprotected. There is an option to set a password for starting and stopping the desktop firewall service, but this feature will not prevent from the above explained attack, as the password prompt will not appear.

General conclusion

For all three test products, it is possible to terminate the desktop firewall process. This can be implemented by a computer virus or a trojan horse program, giving it full access to the Internet after the shutdown of the desktop firewall. Of course it is not easy to design a process that can not be stopped by a malicious application executed by the user. Since one of the requirements of desktop firewalls is, that the user is able disable the desktop firewall when needed. This leads to the problem, that an attacker could theoretically use the desktop firewall configuration utility and remove all unwanted functions ending up with an application, that shuts down the desktop firewall process. Therefore, I am not aware of a reliable method of ensuring that the desktop firewall process is not stoppable, unless we remove the right of the normal user to start and stop the desktop firewall.

Counter measure

Sygate's personal firewall is on the right way, but should improve its service some more, so that it will be impossible, at least for a normal restricted user, to shut down the firewall.

5.1.2 Memory injection

Description of attack: Load a malicious program into the memory space of a trusted application, by allocating memory space for a function in its working memory space and starting a thread there.
Tool used: backstealth

Unfortunately this version of backstealth was not compatible with the desktop firewall versions of our test candidates, since it was designed for older versions of desktop firewalls. As the time line for this thesis was short, it did not allow to code a custom memory injection for the three desktop firewalls. This means the test with this tool could not be performed. According to the information available, I assume that it would be possible to successfully insert code in a trusted application. Generic code examples for this exist in different programming languages on the Internet. For example on MADshi's website source code for inserting own functions into applications can be found [15]. Researches showed that some trojan horses like Assasin, already uses similar techniques to bypass desktop firewalls.

5.1.3 SYN flood 1, total random

Description of the attack: Flood the target machine with TCP packets, which have the SYN flag set. The packets are crafted with random source IP addresses, random source ports and random destination ports.
Tool used: HGOD_flood.exe

Zonealarm:

Reports each packet as a different event. No special event message is generated. No denial of service behavior occurs during the attack.

Symantec Desktop Firewall:

Reports each packet as a different event. No special event message is generated. No denial of service behavior occurs during the attack.

Sygate Personal Firewall:

Reports each packet as a different event. No special event message is generated. The CPU utilization goes up to 100% during the attack.

General conclusion

This kind of attack can not be reduced to one single event message, as the events are different. Each packet has a different source IP address, and can not be distinguished from non related traffic. Anyhow, no denial of service behavior should occur.

Counter measure

Make sure that no denial of service behavior occurs, by observing memory and CPU usage. Maybe also block packets on a lower layer. SYN floods should be detected and reported with an alarm event.

5.1.4 SYN flood 2, random ports

Description of the attack: Flood the target machine with TCP packets, which have the SYN flag set. The packets are crafted with static source IP address, random source ports and random destination ports.
Tool used: HGOD_flood.exe

Zonealarm:

Reports each packet as a different event. No special event message is generated. No denial of service behavior occurs during the attack.

Symantec Desktop Firewall:

Reports each packet as a different event. No special event message is generated. No denial of service behavior occurs during the attack.

Sygate Personal Firewall:

Reports each packet as a different event. In the system log a port scan is reported several times from the same IP address. No denial of service behavior occurs during the attack.

General conclusion

This attack should get reported as a port scan or as a denial of service attack. There is no good argument for one or the other alarm, as it could be both of them. Therefore, reporting it as a port scan is not wrong, as for the desktop firewall it looks like a port scan.

Counter measure

There is no real need to do anything against this kind of attack, as the traffic gets blocked at the desktop firewall. No response will be sent back to the attacker.

5.1.5 SYN flood 3, static

Description of the attack: Flood the target machine with TCP packets, which have the SYN flag set. The packets are crafted with static source IP address, static source port and static destination port.
Tool used: HGOD_flood.exe

Zonealarm:

Reports all packets in one event message with an increasing count token. The summarized count was very small, so it must have dropped some packets. No special event message is generated. No denial of service behavior occurs during the attack.

Symantec Desktop Firewall:

Reports each packet as a different event. The count of all event messages was very small, so it must have dropped some packets. No special event message is generated. No denial of service behavior occurs during the attack.

Sygate Personal Firewall:

Reports all packets in one event message with an increasing count token. After an irregular amount of time a new event message will be written, as long as the attack is going on. No special event message is generated. No denial of service behavior occurs during the attack. Some packet seem to be droped during the attack.

General conclusion

This attack should be reported as a denial of service attack. Of course it could be a port scan as mentioned in section 5.1.4, but after monitoring a few identical packets always coming from the same source port, targeting the same destination port, a port scan can be excluded. This behavior would not make sense for a port scan, however it is regular for a denial of service attack.

Counter measure

There is no real need to do anything against this kind of attack, as the traffic gets blocked at the desktop firewall. No response will be sent back to the attacker. However the attack should get reported with a special event message of denial of service attack.

5.1.6 SYN flood 4, heavy

Description of the attack: Flood the target machine with TCP packets, which have the SYN flag set. The packets are crafted with random source IP addresses, random source ports and random destination ports. About 300 KB of network load per second was generated, which is a lot for the test network.
Tool used: HGOD_flood.exe

Zonealarm:

After a short time CPU utilization goes up to 100% and stays there until the end of the attack. Switching the **BLOCK_ALL** button on, will not protect the system from this denial of service attack.

Symantec Desktop Firewall:

After a short time CPU utilization goes up to 100% and stays there until the end of the attack.

Sygate Personal Firewall:

After a short time CPU utilization goes up to 100% and stays there until the end of the attack. Switching the **BLOCK_ALL** button on, will not protect the system from the denial of service attack. The desktop firewall needs more then 60 seconds to recover from the attack. During testing the desktop firewall still used up to 100% of CPU time for 80 seconds after an attack of 15 second. After an attack of 30 seconds, the desktop firewall needed 100 seconds to recover.

General conclusion

For all desktop firewalls it was possible to make a remote denial of service attack. This makes it possible to temporary block a protected target machine.

Counter measure

A denial of service attack should not be possible. Dropping the packets should be done as early as possible in the stack. It is a difficult issue to deal with as at some point of network load, there will be no possibility for the receiving network card to work properly anymore on all network packets. This behavior would also occur if it there is no desktop firewall running on the target machine. Still the desktop firewall should prevent from such attacks.

5.1.7 ICMP flood

Description of the attack: Flood target machine with ICMP packets, type echo request, with spoofed source IP addresses.
Tool used: HGOD_flood.exe

Zonealarm:

Reports each packet as a different event. No special event message is generated.

Symantec Desktop Firewall:

Reports each packet as a different event. No special event message is generated.

Sygate Personal Firewall:

Reports each packet as a different event. Reports a denial of service attack on ICMP protocol in the system log. Regardless of the spoofed source addresses, the desktop firewall reports only a few source IP addresses.

General conclusion

This kind of attack can not be reduced to one single event message, as the events are different. Each packet has a different source IP address, and can not be distinguished from non related traffic.

Counter measure

None needed.

5.1.8 IGMP flood 1

Description of the attack: Flood target machine with Internet Group Management Protocol (IGMP) packets. The packet size was set to 1480 bytes which means no fragments. No spoofed source IP address was used.
Tool used: HGOD_flood.exe

Zonealarm:

Reports one message for all packets, with an increasing counter. No denial of service behavior occurs.

Symantec Desktop Firewall:

Reports each packet as a different event. No denial of service behavior occurs. Symantec desktop firewall has an option to block IGMP traffic. I was not able to see any difference with this option enabled. Maybe this feature only blocks the badly crafted packet used for a *kiss of death* attack [20] and not all IGMP traffic in general.

Sygate Personal Firewall:

Sygate's personal firewall does not support the IGMP protocol. Therefore no Packets are seen and no events are generated. During attack 100% of CPU time is used. After attack was stopped it recovered back to normal. If the raw packet log is enabled, then the packets will be logged there, but no events in the traffic log are generated.

General conclusion

All desktop firewall should be able to see and log IGMP traffic. No denial of service behavior should occur.

Counter measure

Implement IGMP filter into the desktop firewall.

5.1.9 IGMP flood 2

Description of the attack:
Flood target machine with IGMP traffic. The packet size was set first set to 1481 bytes and then to 65000 bytes to have multiple fragmented packets. No spoofed source IP address was used.
Tool used: HGOD_flood.exe

Zonealarm:

Reports one event for all packets, with an increasing counter. No denial of service behavior occurs.

Symantec Desktop Firewall:

Reports each packet as a different event. No denial of service behavior occurs.

Sygate Personal Firewall:

After a short time the CPU load goes up to 100%. It generates an event in the system log for a denial of service attack with unknown protocol. If we have a look at the raw packet log, we notice that half of the packets are blocked and half of them are permitted. Because a payload of 1481 bytes generates two fragmented packets. When the packet size is set to 65000 bytes which will generate 44 fragmented packets, then 43 out of 44 packets are unblocked as they don't have the *offset* bit set to 0 and the *more fragments* bit set to 0. The system log will report a denial of service attack. It seems, that only the fragmented packets get reported to the denial of service detection engine.

When pushing the **BLOCK_ALL** button during the attack, the CPU load rarely goes back to less then 80% most often it remains at 100%. Pressing this button before the attack starts, will limit the CPU usage of the desktop firewall to 50% and keep it at this level during the attack. This is of course not what normaly will happen, because we do not know in advance when an attack will be launched.

General conclusion

The same conclusions as for the first IGMP flood attack in section 5.1.8.

5.1.10 UDP flood

Description of the attack: Flood target machine with UDP packets, with random source IP addresses and random source ports, using a packet size of 1000 bytes.
Tool used: HGOD_flood.exe

Zonealarm:

Reports each event in a separate message. No denial of service behavior occurs.

Symantec Desktop Firewall:

Reports each event in a separate message. No denial of service behavior occurs.

Sygate Personal Firewall:

Reports all packets in one message with increasing counter. No denial of service alarm is created, although the CPU utilization was 100% during the attack.

General conclusion

No denial of service behavior should occur.

Counter measure

Change the implementation of the filtering engine, so that no denial of service behavior occurs.

5.1.11 Modifying rule set

Description of the attack: Locate where the desktop firewall stores the rule set. Modify it, so that it is possible to add custom rules, which will not be detected as suspicious rules. Thus giving the malicious application the privileges that it needs to communicate to the Internet.

Zonealarm:

Zonealarm saves the rules for the applications in a file called *IAMDB.RDB*, which can be found at `C:\WINNT\internetlogs\` on a Windows 2000 system. The file is machine independent. This means that we can replace this file and the file *BACKUP.RDB* which is a copy of the rule file, with a modified version from another machine. While the desktop firewall process is running it has this file opened in a non-shared mode. As discussed in section 5.1.1, it is possible to terminate the desktop firewall process and thus end the lock on the rule file, making it possible even for a restricted user to replace the rule set.

Symantec Desktop Firewall:

Symantec desktop firewall does store all the firewall rules in plain text in the registry. The corresponding registry key is:
`\HKEY_LOCAL_MACHINE\SOFTWARE\Symantec\IAM\FirewallObjects\IPFilterRules\Rule1`
There we can find all the rule related options, like direction or protocol. It is simple for an attacker to insert a rule at top of the ordering, that will allow all traffic through the desktop firewall.

Sygate Personal Firewall:

Sygate's personal firewall stores the rule set in a file called *STDDEF.DAT* in the default installation folder of the desktop firewall. The file is machine independent. This means that we can replace this file with a modified version from another machine. While the desktop firewall process is running it has this file opened in a non-shared mode. As discussed in section 5.1.1 it is possible to terminate the desktop firewall process and thus end the lock on the rule file, making it possible even for a restricted user to replace the rules.

General conclusion

The rule file is the brain of the desktop firewall. If we are able to modify those entries, then the use of the whole desktop firewall is at danger. In the experiments it was possible to add custom rules for all tested desktop firewalls. This method can be implemented by a trojan horse to add an *allow_all* rule, thus being able to communicate with the Internet without restrictions.

Counter measure

The rule set can be encrypted and protected with a message authentication code (MAC) but all the data especially the key that is needed to decrypt it, would have to be stored local on the same machine. Otherwise, the firewall would not be able to read the rules after a reboot of the machine. However assuming that the secret key is stored locally makes it possible for another application to read it and further more use it to decrypt the rule data.

If it is opened by the desktop firewall in a non-shared mode by blocking the ability of another process to access this file, the question arises if it is stored somewhere in memory where it could be manipulated? Even if this is not the case, it is possible to terminate the current desktop firewall process and thus also stop the exclusive access mode of the rule file, leaving the file unprotected as we have seen in Section 5.1.1.

Another possibility is that the first time when the user installs the desktop firewall it asks for a password. With this password, the rule set will be signed using a asymmetric encryption schema. Each time the desktop firewall starts, it can use the public key to check the signature of the rule file to verify if the file has been altered. If the user wants to modify some rule, he supplies his private key and the rule file gets updated. This could work, if there was not the problem, that a malicious user could simply replace the rule set and provide a corresponding public key for it, so that the verification method would accept it. The problem in this setup is, that we can not relay on any data, as it has to be stored local and therefore can be altered by a malicious attacker. The public key is not authentic as it can be replaced. Moving the verification process to an other machine is not practical.

The rule files could be installed with administrator rights, having a small service running, which accepts only calls from the user interface for modifying requests. In this way, it would be possible to protect the rule file from direct altering, as the normal user has not the rights to modify this file. But the problem is, that the normal user wants to be able to change their rule configuration. Having a possibility for the normal user to change the rules, always makes it possible for a malicious application to use the same methods. For example, we can think of a modified version of the users interface, that automatically installs an *allow_all* rule when started. For the skeptics, think of the following scenario. A malicious application makes a screenshot of the actual user desktop. Then it displays this picture as an image on top of everything and disables the keyboard. For a normal user this will look like his machine has frozen. Behind this image, the tool simulates mouse clicks and key strokes to the desktop firewall. This way the tool can create an *allow_all* rule like the normal user would have done it. In the end the screenshot picture will be removed and the user did not notice that a new rule was installed.

Sygate is using another good approach. They write the rules from memory back to the file when the desktop firewall is shutdown. This eliminates the problem of file manipulations during normal use. Unfortunately this method is only safe when it is not possible to manipulate the rules in memory and when it is not possible to shutdown the desktop firewall process illegitimate. The latter has been proven to be possible in section 5.1.1.

Unfortunately I can not provide a stable working solution for this problem. In my opinion it has to be solved using an administrator process that handles the rule file.

5.1.12 Replacing the binary

Description of the attack: Replace a trusted application with a malicious one, that pretends to be the trusted application. For the test `C:\Program Files\Internet Explorer\IEXPLORE.EXE` is replaced by the `telnet.exe` application, by renaming and overwriting it. After this an outgoing connection to TCP port 80 is initiated.

Zonealarm:

The desktop firewall does notice the change of the binary and asks if it should update its rule base. Zonealarm saves the rules for the applications in a file called *IAMDB.RDB*, which can be found at `C:\WINNT\internetlogs\` on a Windows 2000 system. The file is machine independent. This means that we can replace this file and the file *BACKUP.RDB* which is a copy of the rule file, with a modified version from another machine. While the desktop firewall process is running it has this file opened in a non-shared mode. As discussed in section 5.1.1 it is possible to terminate the desktop firewall process and thus end the lock on the rule file. Making it possible even for a restricted user to replace the rules.

Symantec Desktop Firewall:

It does notice the change of the binary and tells that there exist some rules for telnet, but that the file has changed its path since last time, and the desktop firewall needs to create a new rule. It recommends using the automated rule creating wizard, which looks at the filename and then grant even more access to the modified telnet then we wanted. For example FTP, Gopher and SSL ports were allowed. The recommendation wizard should be disabled, as it would lead an unexperienced user to chose the wrong thing. It is a valuable feature for normal configuration, but can easy be misused for a malicious purpose.

All the signature of the applications are saved in the registry in plain text and without any further integrity checks. Therefore we can easy overwrite Internet explorer's signature hash with the new hash for the malicious application. The hash is stored in a registry key located at:
`HKLM\Software\symantec\IAM\FirewallObjects\Applications`
`\Internet Explorer\ApplicationSignature1 Reg_Binary`
This hash is machine independent. We do not have to know how it is generated, as we can simply generate one on another machine and overwrite the original one with the new one.

Sygate Personal Firewall:

I replaced *Internet explorer* with a renamed *telnet.exe* and started it. The replacement of the binary is reported by the desktop firewall as an event and information is displayed asking the user, if the changes are expected. In the application list it did change the name from Internet explorer to telnet, but the network connections are still blocked for the renamed telnet.

The question araises where Sygate does store the hashes of the applications. Nothing was found in the registry. So I assumed that they are stored in a file. With the help of file access monitor tools, I was able to trace the write commands back to the file called *stdState.dat*, which is located in the home directory of the Sygate installation. Sygate stores the rules for each application in this file. The file is encrypted, so that we can not change it. However during the experiment it was possible to replace this file, with a *stdState.dat* file from another machine. If the replacement is done while the desktop firewall is running then nothing happens, as at shutdown Sygate writes back the rules that it had been loaded into memory. But if the firewall was not running, then it will accept the new file at next start up and present all the rules, which we have configured on the remote machine. This makes it possible to replace the application rule file.

General conclusion

The rule file and hash values are the brain of the desktop firewall. When we are able to modify those entries then the use of the whole desktop firewall is at danger.

In the experiments it was possible to replace the application fingerprints for all tested desktop firewalls. This method can be implemented by a trojan horse to add itself to the trusted applications, which enables communication with the Internet without restrictions.

Counter measure

For counter measurements see the attack about modifying the rule set in Section 5.1.11.

5.1.13 Port scan 1

Description of the attack: Perform a port scan on the target machine with nmap to find out which ports are open. This is often the first thing an attacker does, to find out more about the target. Nmap was used to scan with the option `-sT -p 440-450 -v -P0 -T Aggressive.` This is a TCP connect scan witch is the basic variant of a port scan, using full TCP handshakes. It is not very stealth but still widely used. Testing all TCP ports from 440 to 450 with no delay between each scan. This port range was chosen with no special intenion.
Tool used: nmap

Zonealarm:

Each packet was reported in an individual message. Nmap reported all ports as filtered.

Symantec Desktop Firewall:

Each packet was reported in an individual message. Nmap reported all ports as filtered.

Sygate Personal Firewall:

Each packet gets logged. The system log reports 3 port scans events with a count of 6,4 and 10. Nmap reported all ports as filtered.

General conclusion

All three desktop firewalls do not react in a fully stealth way, but they all react in the same way. No port was reported as open or blocked.

Counter measure

There is no need for a counter measure, as no information leaked out.

5.1.14 Port scan 2

Description of the attack:
Perform a port scan on the target machine with nmap to find out which ports are open. Use nmap to scan with the option `-sT -p 440-444 -v -P0 -T sneaky.` This is the same TCP connection scan as used in *port scan 1*, but this time with a large delay of 15 seconds between the single scans. Slowing down the scan can bypass some detection engines, as they watch for a specific number of connections in a fixed time interval.
Tool used: nmap

Zonealarm:

Similar to the fast scan result, it does report all the packets separately.

Symantec Desktop Firewall:

Similar to the fast scan result, it does report all the packets separately.

Sygate Personal Firewall:

Similar to the fast scan result, it does report all the packets separately. System log reports two port scan events, each with a count of one.

General conclusion

All ports are reported filtered. The attacker did not gain information out of this attack.

Counter measure

There is no need for a counter measure, as no information leaked out.

5.1.15 Port scan 3

Description of the attack: Perform a port scan on the target machine with nmap to find out which ports are open. Use nmap to scan with the option `-sS -p 440-444 -v -P0 -T sneaky.` This time only SYN packets are sent out. This technique is often referred to as "half-open" scan, because we do not complete the full three way TCP handshake. After getting a response from the target system, we immediately close the connection request, by sending a RST packet.
Tool used: nmap

Zonealarm:

Each packet was reported in an individual message. Nmap reported all ports as filtered.

Symantec Desktop Firewall:

Each packet was reported in an individual message. Nmap reported all ports as filtered.

Sygate Personal Firewall:

Each packet was reported in an individual message. Nmap reported all ports as filtered. System log reports 4 port scan events.

General conclusion

All ports are reported filtered. The attacker did not gain information out of this attack.

Counter measure

There is no need for a counter measure, as no information leaked out.

5.1.16 Port scan 4

Description of the attack: Perform a port scan on the target machine with nmap to find out, which ports are open. Use nmap to scan with the option `-sX -p 440-444 -v -P0 -T normal.` In this scan a Xmas scan is used. This means a special packet with the FIN, URG, and PUSH flags set is sent. As the SYN flag is not set this packet will bypass stateless filters. Depending on the answer we can guess if the port was open or not. This method does not work every time, as it is OS dependent.
Tool used: nmap

Zonealarm:

No event messages are generated. Nmap reports all ports open, which is wrong.

Symantec Desktop Firewall:

No event message are generated. Nmap reports all ports closed.

Sygate Personal Firewall:

No events are generated in the traffic or system log. The raw packet log does report the packets. Nmap reports all ports open.

General conclusion

This behavior makes it possible to guess which desktop firewall is used at the target machine. This information could later be used to launch an exploit that targets a specific version of desktop firewall.

Counter measure

This problem could only be solved by implementing a statefull packet inspection firewall.

5.1.17 Mutex blocking

Description of the attack: Some of the desktop firewall use a mutex process to check if there is already an instance of itself loaded. A mutex is a program object, that allows multiple program threads to share the same resource, such as file access, but not simultaneously. When a program is started, a mutex is created with a unique name identifier. After this point, any thread that needs the resource must lock the mutex from other threads while it is using the resource. The mutex is set to unlock when the data is no longer needed or the routine is finished.

By stopping the desktop firewall process and blocking its mutex with a malicious process, it is possible to prevent the desktop firewall from loading and so from protecting the network.
Tool used: zonemutex from Diamond Computer Systems labs

Zonealarm:

The tool kills the running desktop firewall process. Then it will set a mutex with the name `Zone Alarm Mutex` preventing the real zonealarm from reloading. As long as this tool blocks the mutex the machine will be unprotected. A malicious application could even implement the tray icon to pretend that the original desktop firewall is working normally.

Symantec Desktop Firewall & Sygate Personal Firewall:

Unfortunately there was no ready made attack for these desktop firewalls available. As the time schedule of this thesis is short set, I decided not to invest time to make an attack for these desktop firewalls. Therefore, this test could not be done for Symantec's desktop firewall & Sygate's personal firewall.

General conclusion

This problem is somehow related to the process killing issue. Because if the firewall is already running we have to kill the process before we can block the mutex. Still it is very serious that a single call to the mutex API can block the desktop firewall from loading.

Counter measure

Prevent other process from imitating the desktop firewall mutex. This can probably be achieved by using cryptographic algorithms to protect the mutex. Still it is not trivial. We can think of the idea that a malicious attacker takes the original desktop firewall application and extracts the creation part of the mutex from this application. With this it would be possible to create a blocking mutex, regardless of what is used to protect the mutex.

5.1.18 Inject DLL

Description of attack: Make a DLL that will be automatically loaded with a trusted application. Then it can use the rights of the trusted applications to start connections to the network. For this test, a tool was used which creates a DLL that is loaded with Internet explorer.
Tool used: firehole

Zonealarm:

Zonealarm reports nothing unusual. The packets were granted the same rights as the trusted parent application.

Symantec Desktop Firewall:

Symantec reports nothing unusual. The packets were granted the same rights as the trusted parent application.

Sygate Personal Firewall:

Reports a changed DLL and asks if it should be blocked or permitted. This is done comparing the DLL to an internal list with names and fingerprints. As discussed in section 5.1.12 it would be possible to replace this internal list, so that the new DLL can be loaded without an alarm.

General conclusion

This idea is similar to the idea of injecting a thread in the process memory of a trusted application.

Counter measure

The desktop firewall should also check all the loaded DLLs of an application and keep track of them. Sygate's desktop firewall does this already. Of course this goes back to the problem of where and how to store the signatures that are checked against the DLLs, as mentioned in section 5.1.12.

5.1.19 URL encoding

Description of attack: Use the given web browser to send a hidden string encoded in an URL. For example opening the URL
`http://www.attackerssite.com\input.php?passwd=secret&IP=192.168.0.7`
The Browser window used to send this information might be set to invisible or to be out of the users desktop screen, so that it is not suspicious.
Tools like tooleaky use this method.

General conclusion

This method will always work, as long as web traffic is allowed. If we allow normal web traffic to pass the desktop firewall, then this method can not be blocked. It is impossible to distinguish between malicious and harmless web traffic. As all simple mail forms use the same methods of HTTP posts for communication as well. This really leads to a problem, as no users will give up the possibility to browse the Internet, just for security reasons.

Counter measure

Blocking the web browser from the Internet is not practicable and therefore not an option. Even if the desktop firewall does statefull packet inspection, this method would still be undetected, as the traffic is legal HTTP traffic. One thing that would help, is if we block requests from other applications to open an URL in the default browser. But this is one of the features the user probably is not willing to give up. Using a web proxy would not eliminate the problem. The final conclusion is, that there is no practicable solution to prevent this attack.

5.1.20 LSP

Description of attack: Install a layered service provider (LSP) [16] that will listen for all the incoming traffic. If the LSP gets installed under the desktop firewall in the protocol chain, it might be possible to read packets before they are rejected in the chain by the desktop firewall. When implemented to filter out the packets of the attacker these special packet would never reach the desktop firewall. This would make a two way communication possible without the desktop firewall even

knowing what was going on.
Tool used: AWP LSP

Zonealarm:

LSP can be installed with no problem, packets that are blocked do not show up in the LSP.

Symantec Desktop Firewall:

LSP can be installed with no problem, packets that are blocked do not show up in the LSP.

Sygate Personal Firewall:

LSP can be installed with no problem, packets that are blocked do not show up in the LSP.

General conclusion

The LSP was not installed deep enough in the chain, especially not under the desktop firewall driver or they used other methods to get the network card events. The desktop firewall was still acting normal and discarding all the packets before the LSP was able to view them. Therefore this attack was not successful.

Counter measure

None needed, but make sure that it is not possible to install a LSP below the the desktop firewall driver.

5.1.21 Outgoing flood

Description of attack: Try simulating a malicious application, which wants to attack another target from the protected machine. Therefore a flooding tool is used to attack an other machine. The protocol is varied with TCP packets with SYN flag set, IGMP traffic and UDP packets.
Tool used: HGOD_flood.exe

Zonealarm:

Zonealarm detects the tool and asks to block or allow its traffic.

Symantec Desktop Firewall:

No event is created, no packet is seen.

Sygate Personal Firewall:

Sygate detectes the tool and asks to block or allow its traffic. All packets can be found in the raw packets log.

General conclusion

This would be the usual scenario that gets created if a denial of service attack would be started from this machine. For example when a computer worm tries to spread [8].

Counter measure

Outgoing floods should be noticed and blocked, if unwanted or not authorized. Therefore a statefull inspection filter is needed, as flood tools may send special packets.

5.1.22 Using 100% CPU and MEM

Description of attack: With the help of a JavaScript file it is easy to generate 100% CPU load and 100% memory usage. The idea behind this is, that it might be possible that some of the events will not get logged by the desktop firewall, because of resource problems. If the desktop firewall was not well programmed it could also be possible that the process will crash.

Zonealarm:

The desktop firewall did not crash and reported normally, no events are skipped during tests.

Symantec Desktop Firewall:

The desktop firewall did not crash and reported normally, no events are skipped during tests.

Sygate Personal Firewall:

The desktop firewall did not crash and reported normally, no events are skipped during tests.

General conclusion

Intensely using resources of the target system seems not to affect the desktop firewall. Of course it renders the system itself unusable.

Counter measure

None needed.

5.1.23 Special packet check

Description of attack: Send special crafted packets to the target system, to see what kind of packets can be detected. Focus will be held on all variations of flags that can be set in a TCP packet. If it is possible to send packets that are not logged, it could be possible to set up a communication channel with a trojan horse using these special packets.
Tool used: packetcrafter

Zonealarm:

Type of packet:	**Flags set:**	**Detected:**
IP	-	no
TCP	-	yes
TCP	URG	yes
TCP	ACK	yes
TCP	PSH	yes
TCP	RST	yes
TCP	SYN	yes
TCP	FIN	yes
TCP	all except PSH	yes
UDP	-	yes

Table 5.1: Zonealarm special packet test

Table 5.1 shows the result of the special packet test for Zonealarm. When source and destination IP address are set to local address (127.0.0.1), then the packet will be treated as outgoing packet.

Symantec Desktop Firewall:

The results of the special packet test for Symantecs desktop firewall are shown in 5.2. Only packets that have the SYN flag set will be detected. When source and destination IP addresses are set to local address (127.0.0.1), then the packet will be handled as incoming traffic.

Type of packet:	**Flags set:**	**Detected:**
IP	-	no
TCP	-	no
TCP	URG	no
TCP	ACK	no
TCP	PSH	no
TCP	RST	no
TCP	SYN	yes
TCP	FIN	no
TCP	all except PSH	yes
UDP	-	yes

Table 5.2: Symantec special packet test

Type of packet:	**Flags set:**	**Detected:**
IP	-	no
TCP	-	yes
TCP	URG	yes
TCP	ACK	yes
TCP	PSH	yes
TCP	RST	yes
TCP	SYN	yes
TCP	FIN	yes
TCP	SYN & ACK	no
UDP	-	yes

Table 5.3: Sygate special packet test

Sygate Personal Firewall:

Table 5.3 shows the results of the test with special packets for Sygates personal firewall. All packets get logged in raw packet log. When source and destination IP address are set to local address (127.0.0.1), then the packet will be handled as incoming traffic.

General conclusion

A malicious tool like a trojan horse could use special packets to communicate through the desktop firewall, as not all of them detect all sorts of packets.

Counter measure

Desktop firewalls should contain a statefull packet inspection filter which is able to detect all these packets, and check if they belong to a valid connection.

5.1.24 Modifying log file

Description of attack: Try to modify the log file of the desktop firewall. With this idea an attacker could delete all possible traces of a successful break in. Even placing wrong information to blame other people could be possible. This could lead to problems, when the data in the log file is held as true data.

Zonealarm:

Zonealarm stores the log in a plain text file which can be altered easily or even deleted while the process is running. The application console itself has a buffer on its own, that will remain filled even if the program is restarted. This means altering the log file will not be reflected in the console. Still all inserted events will propagate to the centralized security console if one is used.

Symantec Desktop Firewall:

While the desktop firewall is running, the log file can not be deleted. But there is a registry flag at
`HKEY_LOCAL_MACHINE\SOFTWARE\Symantec\IAM\Logs\Firewall` when switching this off, the logging process will be disabled. Of course when the firewall is not running then the log files can be altered. At the next reload all new events will also appear in the console view.

Sygate Personal Firewall:

While the desktop firewall is running, the log file can not be deleted. Of course when the firewall is not running then the log files can be altered. At the next reload all new events will also appear in the console view.

General conclusion

By terminating the desktop firewall process it is possible to inject alarms or delete the whole log file.

Counter measure

Protect the log file from altering during running process. There is no practical way to protect the log file from modifications, when the process is shutdown. Encrypting the file is not an option, as it should be possible to extract the data by third party tools, so that they could send the events to a centralized correlation engine. Even more important is the argument that the key for decryption will be stored locally too and can be extracted from the desktop firewall. As the desktop firewall is not running, there is also no way of protecting this key from malicious tools.

5.1.25 Push the yes button

Description of attack: Make a small application that rests in the users memory and monitors the names of the appearing windows. It waits until it detects a window from the desktop firewall wizard, which asks the user if he wants to block or allow a certain traffic. At this moment, the application simulates pressing the "*yes*" button, for example by sending the keystrokes for the shortcut. If this is done fast enough the user will not notice it. This method enables us to create rules for new applications that allow them to communicate with the Internet.

General conclusion

This is a simple local attack that does not attack the desktop firewall itself. Rather it makes use of a feature that all desktop firewalls have implemented and enabled by default. Therefore this attack works on all versions of desktop firewalls.

Counter measure

This new added rule could be seen in the rule set, but a common rule set has more then 100 rules, making it not easy to find a newly added one. A normal user will seldom look at his rule set, at least not as long as all his applications still work. Disabling this feature is not an option, because it is the most helpful feature for a new user. With it, the user does not have to find out himself, what communication a new installed application needs. Thus there is no protection from this attack.

5.1.26 Avoiding visibility

Description of attack: There exist several *rootkits* for windows. With such tools it is possible to hide running applications from the system. This is done by patching kernel functions, or catching API calls in the system.

General conclusion

The used rootkits were able to successfully hide the process from the task manager and from the explorer. But they are not able to prevent the desktop firewalls from seeing and blocking the traffic. I think that it will be not easy to manipulate the network driver to filter out certain traffic, before the desktop firewall does see it. On the other hand supplying a new network stack could solve this problem. This would then be the same idea as discussed in section 5.1.20.

Counter measure

None needed.

5.1.27 Tunneling

Description of attack: Tunneling is an expression for repacking packets in other protocols. The idea is to use an allowed protocol such as HTTP and fill those packets with the original traffic. For this method often protocols like HTTP, SMTP or ICMP are used.

General conclusion

This method works fine with network firewalls, as they can not see the difference between the tunneled packets and the original packets.

This method will not work with desktop firewalls, as they will see the different application names and then ask to block them. For a desktop firewall it makes a difference which application is sending the traffic.

Counter measure

None needed.

5.1.28 Trojan horse port

Description of attack: Simulate a trojan horse server program that tries to make a listening server for incoming connections.

General conclusion

Incoming connection to trojan horse servers are blocked as normal incoming traffic, and are mapped with the rule name to the name of the trojan horse which is normaly using this port. A trojan horses can be configured to run at any port, therefore it makes no sense to map this ports to a trojan horse name.

From the security point of view, this scenario is not a problem for the desktop firewalls, as they will detect the trojan horse server when it tries to access the Internet. Then the desktop firewall will ask to block this application, preventing the trojan horse from accepting connections.

Counter measure

None needed.

5.2 Conclusions

Desktop firewalls do a fairly job in protecting against incoming attacks. They can not prevent a major denial of service attack but this is normal. Still the experiments have shown that they could improve the prevention of denial of service attacks to a better level. One thing that should be kept in mind is that it is dangerous to implement automatic response systems into desktop firewalls. React on alarms that have been raised by the desktop firewall is ok as long as we do not trust the source of the attack. Most of the attacks used during the experiment can be run with fake source addresses, making it impossible and probably fatal to block it. Setting the system into stealth mode, and thus not respond to any unwanted packet is a nice feature, but not required. Open ports on the local system will always be reported as open ports in a port scan, and these are the only ones that the attacker is interested in. Of course services that should not be accessible from the network should be blocked, but that is a matter of configuration.

On the other hand desktop firewalls are vulnerable to different attacks coming from the system they are installed on. This problems can not be solved. We can not prevent actions which the normal user is able to perform too. For example, as long as an user is able to start and stop the service of the firewall on its own, a malicious application executed by the user will have the same possibilities. The same applies to the rule set. As long as the user is able to modify the rule set it will be possible for a malicious application to modify it too. This can be solved by running the desktop firewall with administrator rights, like most anti-virus scanner already do. However this takes away the comfort for the user to add new rules on the fly. This really is the main problem of desktop firewalls, they run on a system where they can be attacked from the inside. Therefore it is not possible to make anything that a malicious user with administrator rights can not change.

Desktop firewalls should implement statefull packet inspection filter, otherwise the possibility for a hidden communication channel using special packets is given. This would also eliminate some outgoing denial of service attacks which are not yet detected because of this problem. Of course desktop firewalls should also monitor all the used protocols, especially IGMP.

Chapter 6

Correlation & Cooperation

In this chapter, the generic event log format is defined and some examples for cooperation of desktop firewalls are given. Then the correlation of log events is discussed and illustrated with a real world experiment.

6.1 Generic event log format

If we want to compare and correlate events from desktop firewalls with other desktop firewalls or other security related tools, it is inevitable to map the different log file formats into one generic format. This format should contain all the necessary information. Having a consistent format will make it easier to write matching rules for alarms. Unfortunately there does not exist something like a generic log format, that is already accepted and used by every vendor. Therefore we have to provide tools which convert the original event format into the new generic log format. As part of this thesis I developed three Perl scripts that can be used to translate the log files of the three tested desktop firewalls into the generic log event format. These tools can be found on the attached CD-ROM or on the Internet at the URL `http://www.trojan.ch/DFW/`.

6.1.1 Definition

Information, which all desktop firewalls have in common should be included in the generic event log format. In addition fields that are not present in the log files but still needed should be added by the parser script, for example a trust field. After analyzing the log files and comparing them with IDS and network firewall log files, we came up with a generic event log format, containing the fields shown in Table 6.1, and described next:

- **SensorIP:** (mandatory)
  This field specifies the IP address of the machine that the desktop firewall

Description:	**Field name:**	**Value:**
IP address of sensor	sensorIP	IP address [4 octets]
Type of sensor	type	[ZOA3|SDF2|SPF5]
Time when event occurred	time	Unix timestamp
Source IP address	srcIP	IP address [4 octets]
Source host name	srcName	[STRING]
Source port number	srcPort	[Integer]
Destination IP address	destIP	IP address [4 octets]
Destination host name	destName	[STRING]
Destination port number	destPort	[Integer]
Action that was performed	action	[blocked|allowed|ignored|info|N/A]
Direction	direction	[incoming|outgoing|N/A]
Protocol which was used	protocol	[TCP|UDP|ICMP|IGMP|N/A]
Flags or type of the packet	flag	[SYN|FIN|ECHO_REPLY|...]
Application which was involved	application	[STRING]
Level of trust in this event	trust	[0-9]
Signature name	signature	[STRING]
Entire original message	msg	[STRING]

Table 6.1: Generic event log format

is running on. This information could also be calculated by looking at the source address, destination address and the direction, but this method would fail in some cases. For example when the packet was sent to the broadcast address. In additional, this is an information that is frequently needed and it makes sense to have it as a self-contained field. It will be set by the parser script, which was developed in this thesis.

- **Type:** (mandatory)
This field defines the actual type and version of the desktop firewall that was used. This can be used to filter out events from a specific vendor. For example, if it is known that those signatures do not match a certain attack very well. Or this information could be used to write correlation rule, which take advantage of the peculiarities of a specific type of desktop firewall. The products used in this thesis have been labeled **ZOA3** for Zonelab's Zonalarm 3, **SDF2** for Symantec's desktop firewall 2 and **SPF5** for Sygate's personal firewall 5. The parser script will fill in this information.

- **Time:** (mandatory)
  This is simply a common Unix timestamp. Make sure that all monitored machines are in the same timezone.

- **SrcIP:** (optional)
  This is the IP address of the origin, where the packet came from. It might be empty if the host name was given instead.

- **SrcName:** (optional)
  This field contains the host name of the machine that the traffic was sent from. It does not always make sense to convert this name into an IP address, as the corresponding IP address might have changed till the conversion takes place or the local DNS resolution might have been altered.

- **SrcPort:** (optional)
  This field contains the source port number, from where the traffic was sent.

- **DestIP:** (optional)
  This is the IP address of the target, where the packet was directed to.

- **DestName:** (optional)
  This field contains the host name of the target machine that the traffic was sent to. It does not always make sense to convert this name into an IP address, as the corresponding IP address might have changed till the conversion takes place or the local DNS resolution might have been altered.

- **DestPort:** (optional)
  This field contains the destination port number of the target machine, where the traffic was sent to.

- **Action:** (mandatory)
  The field action may contain one of five different things:

  1. **blocked**, which means that the traffic was blocked and discarded at the desktop firewall. This information can be useful to verify wheter an attack was successful or not.
  2. **allowed**, which means that the traffic was not blocked and did pass the desktop firewall. This information can be used to verify whether an attack was successful or not.
  3. **ignored**, which means that the traffic was logged but not blocked.
  4. **info**, which means that this event is just an information event, like *"new rule added for telnet.exe"*. When this parameter is set, then the information of the event can be found in the "msg" field.
  5. **N/A**, which means that none of the above things matched or that there was an error while parsing this event line in the original log file.

- **Direction:** (optional)
  The direction, incoming or outgoing, is included in the generic format as a field of itself. It could be calculated from the source and destination IP addresses, but they could be spoofed addresses. Some of the desktop firewall for example Zonealarm, seem to do it like this and fail when we send a packet that has source and destination IP address set to the desktop firewall's address. It will then mark this packet as outgoing traffic, because the source IP address is the local address. This is an information that we probably often need for filtering, and it therefore makes sense to have it in a self-contained field.

- **Protocol:** (optional)
  This field contains the protocol of the packet that was monitored, for example TCP or UDP. It is optional as not all events contain it. For example informational messages do not include a protocol specification.

- **Flag:** (optional)
  In this field additional information of the used protocol will be placed if available. This can be "SYN" if for example the syn flag was set in the packet.

- **Application:** (optional)
  This field contains the name of the application that was sending or receiving packets on the monitored system.

- **Trust:** (mandatory)
  This field was additionally introduced and can not be found in the desktop firewall log files itself. Its value is a number between 0 and 9, expressing the level of trust that we have in this alarm message. A value of 0 means we do not trust this alarm and we think that it is incorrect, while 9 means that we trust this alarm and that we think that it is true. This parameter should help eliminate false positives by rating events and maybe ignoring an alarm which has a low trustiness during correlation. In this thesis there is only one rule that modifies this level of trust. Per default a trust level of 5 is assigned to all the parsed messages. The only rule used to modify is, rating alarms that result from outbound connections from the protected machine with *default_trust + 1*. Inbound events will have an alarm with a trust level of *default_trust - 1*. The reason for this is that we have more control over the local events and we can provide more information on those events. Remote events can easily be faked to make us believe whatever the attacker wants us to believe. For example a port scan targeting the protected machine can come from a spoofed source IP address such that it looks like it is coming from someone else. Modifying local events is much harder.

- **Signature:** (optional)
  This field contains the name of the signature that was matched for this alarm. Keep in mind, that the user could have the right to change the rules and therefore also to rename it to something that does not correspond with the

real event. But when controlled, this information can be useful in filtering the events later.

- **Msg:** (mandatory)
  This field contains the whole event message as it appeared in the original log file. This field was added especially for the information messages. If an event has set the action to information, then we can find here the specific message string. As there are too many different information messages, it would not make sense to add new fields for each of them. Also it can be used as a fall back if some information got lost in the parsing script.

Some of the fields might stay empty after parsing the different event logs, because not all of the information might be present in the original event message. The event numbers have been removed, as they might not be consistent and are not of interest. The correlation center or central console can and most probably will add their own numbering to the events.

6.1.2 Exporting

The following are the actual export mappings from the specific desktop firewalls into the defined generic format. As mentioned at the beginning of this chapter, during this thesis tools were developed for translating the events, which can be found on the attached CD-ROM, or on the Internet at the URL `http://www.trojan.ch/DFW/`. They are Perl scripts which use regular expressions to match the different event messages and apply the information found to the generic event log format schema. Once generated, they are written to a plain text file and can be further analyzed.

Zonealarm log file mapping

A standard line in a Zonealarm log file looks like this:

```
FWIN,2002/11/25,14:35:52 +1:00 GMT,10.10.10.7:53,10.10.50.2:53,TCP (flags:S)
```

This would be mapped into the generic format as shown in Table 6.2.

Symantec log file mapping

A standard event in Symantec's desktop firewall does look like this:

```
11/27/2002 15:50:48 Rule "Implicit block rule" blocked (10.10.50.1,446).
Details:  Inbound TCP connection
Local address,service is (10.10.50.1,446)
Remote address,service is (10.10.50.4,1684)
Process name is "N/A"
```

FWIN,	→ direction
2002/11/25, 14:35:52 +1:00 GMT,	→ time
10.10.10.7:53,	→ srcIPaddr + srcPort
10.10.50.2:53,	→ destIPaddr + destPort
TCP (flags:S)	→ protocol + flag
complete string	→msg
depending on rule	trust
set by script	sensorIP
set by script	type
set by script	action

Table 6.2: Zonealarm translation table

12/27/2002 15:50:48	→ time
Rule “Implicit block rule”	→ signature
blocked	→ action
(10.10.50.1,446). Details:	*ignored because redundant*
Inbound TCP connection	→ direction + protocol
Local address,service is	*ignored*
(10.10.50.1,446)	→ destIPaddr + destPort
Remote address,service is	*ignored*
(10.10.50.4,1684)	→ srcIPaddr + srcPort
Process name is “N/A”	→ application
complete string	→ msg
depending on rule	trust
set by script	sensorIP
set by script	type

Table 6.3: Symantec desktop firewall translation table

A corresponding mapping into the generic log event format can be done as shown in Table 6.3. Unfortunately the port number of known service ports are automatically mapped to the corresponding service names. This means, that for example port number 80 will be replaced with the string "http" in the log file. This fact is quite annoying, because it makes it harder to compare the logs. A switch to disable this option could not be found. For further analysis this has to be remapped to the original port number. Another installed feature is the resolving of the IP addresses into the host names, which can also not be turned off in the configuration.

Event messages like the ones shown in Table 6.4, which are initialization informations, will be mapped into information messages with the action parameter set to "info".

```
1/23/2003 9:19:14 Inbound IGMP packets are being blocked.
1/23/2003 9:19:14 Inbound IP fragments are being blocked
1/23/2003 9:19:14 NDIS filtering is enabled
1/23/2003 9:19:14 Interactive learning mode is enabled
1/23/2003 9:19:14 Firewall configuration updated: 149 rules
```

Table 6.4: Symantec desktop firewall sample information events

Sygate traffic log file mapping

A typical event in Sygate's personal firewall traffic log file looks like this:
`32104 11/27/2002 15:55:24 Blocked TCP Incoming 10.10.50.4 1897 10.10.50.3 446 1 11/27/2002 15:55:11 11/27/2002 15:55:11 Block_all_unknown` Table 6.5 shows how the mapping into the generic event log format can be done for the traffic log file.

Sygate system log file mapping

The events in the system log file are on a higher level. They are already correlated and look like the events of an internal IDS. This makes them different than the three log files that we already have mapped. Still this information is of great value for us, as they compress the events into one alarm. A typical event in the system log looks like:

```
103 11/27/2002 15:55:14 Port Scan Minor Incoming TCP 10.10.50.4 10.10.50.3 10
11/27/2002 15:55:10 11/27/2002 15:55:13
```

Table 6.6 shows how the mapping into the generic event log format can be done for the system log file.

32104	*ignored*
11/27/2002 15:55:24	→ time
Blocked TCP Incoming	action + protocol + direction
10.10.50.4	→ srcIPaddr
1897	→ srcPort
10.10.50.3	→ destIPaddr
446	→ destPort
1	*ignored*
11/27/2002 15:55:11	*ignored*
11/27/2002 15:55:11	*ignored*
Block_all_unknown	signature
complete string	msg
depending on rule	trust
set by script	sensorIP
set by script	type

Table 6.5: Sygate traffic log translation table

103	*ignored*
11/27/2002 15:55:14	→ time
Port Scan Minor	→ action
Incoming	→ direction
TCP	→ protocol
10.10.50.4	→ srcIPaddr
10.10.50.3	→ destIPaddr
10	*ignored*
11/27/2002 15:55:10	*ignored*
11/27/2002 15:55:13	*ignored*
complete string	msg
depending on rule	trust
set by script	sensorIP
set by script	type

Table 6.6: Sygate system log translation table

6.2 Collaboration

6.2.1 Multiple instances of the same desktop firewall

Having multiple instances of the same type of desktop firewall installed on different machines might be the common case in a company. This is also the same set up that was tested in a real world experiment in Section 6.4. This setups are simpler to maintain because we only have one type and can create a specific rule set using all the provided features. The advantage of this setup against a single installed desktop firewall is, that we have access to the same information on different machines. With this, we can see if a detected attack was only targeting a single machine, or the whole subnet. This makes it possible for us to detect attacks that have made only a few events on each machine, but a lot of events in total over all machines.

A disadvantage is, that if the chosen desktop firewall has a vulnerability, all the machines in the network are affected in the same way. An attacker could render the complete network unusable by exploiting such a vulnerability.

6.2.2 Different desktop firewalls

Having different types of desktop firewalls installed in a subnet can have an advantage over the scenario of a monoculture of desktop firewalls discussed in Section 6.2.1. One advantage is that most likely not all instances are affected by the same vulnerabilities. This makes it harder for an attacker to take over the complete subnetwork. A disadvantage is that it is more costly to maintain different rule sets for the different types of desktop firewalls. For this reason it would make sense to create a generic firewall rule format as we will discuss in Chapter 8. The detection of attacks which target multiple hosts is of course also possible with this setup.

6.2.3 Desktop firewalls and network firewalls

In a setup where we have desktop firewalls and network firewalls running in the same subnet, we do not gain much additional information out of it. But still we can check with the desktop firewall if the network firewall rules do filter out what they should. For example we set the network firewall to block all telnet traffic from the Internet into the Intranet. When the desktop firewall then reports incoming telnet traffic from IP addresses outside the Intranet, we know that something is going wrong. Either the network firewall has a unwanted hole, or some internal machine is faking traffic. So we can further investigate the suspicious event.

6.2.4 Desktop firewalls and intrusion detection systems

This setup is probably one of the best that we can have. A network-based intrusion detection system (NIDS) will identify the attacks by searching for patterns in the traffic. This makes it possible to exactly identify the type of attack. Having then the information from the desktop firewall log files available for correlation, enables us to confirm suspicious events. For example, if an IDS sensor reports a Nimda computer worm, that has tried to access one of the monitored machines, then the desktop firewall of the involved machine should also report an event on the given port. If one of these two messages is missing some special investigation must be made. It could be that an attacker is injecting alarm events into the IDS to overload it. This way the desktop firewall would not report anything. Or it could be that an attacker has compromised the IDS sensor machine and is blocking the messages.

6.3 Correlation

6.3.1 Internet worms

The most frequently seen computer threats nowadays are probably computer worms [13]. Often, they come in by email messages and once activated, try to spread themselves using different methods, such as sending themselves to other victims by email messages, looking for open network shares or searching for vulnerable machines.

A desktop firewall could help in this case very much,, as it would prevent the worm from spreading in the network by blocking its network access. Furthermore it would log the name of the application as which the worm is running. At the console a rule could be set up, so that specific application names get reported. This would work if the used application name of the worm is known and does not vary much. But even with a unknown application name it can work. When the suspicious application continually is trying to access the Internet, the desktop firewall will prevent it and generate an event each time. A simple rule could catch those events when a threshold is reached and rise an alarm. As long as the user does not grant access rights to this application it would work.

Nevertheless the infection of a machine protected by a desktop firewall can not be prevented if the malware comes in by some legitimate ways, such as being received as an attachment by email. In this case only an anti-virus scanner could help in an early detection.

6.3.2 Port scans

A port scan on its own is nothing dangerous and not uncommon nowadays. Still it is often the first step of an attack and should be monitored. With a desktop firewall we are able to monitor our machine for port scans. If we have multiple machines with desktop firewalls installed then we are also able to keep track of IP scans. This are simply port scans that are repeated for multiple IP addresses in a sequence. If only a few destination ports are checked it often is difficult to say if it was a worm or a port scan, but often a port scan will scan more then five ports, which a worm seldom does.

Port scans can easily be detected as we can monitore all the requests from the remote machine to the target system. We just need to analyze those events. Not all events will be of the state *blocked*, as some of the scanned ports might have been in use and therefore were open. This means we have to inspect all the log events regardless of there state. For this purpose we have to enable logging also for accepted packets, and this could lead to much data in the log file.

The source IP address of a port scan will always be the same if the attacker is interested in the result. Well at least this is what people usually think. But since the introduction of blind scans like idlscan, it is possible to have a remote machine scanned without sending any packets directly to it [9]. Combining this techniques with several different relay hosts makes it possible to have a target system scanned from different IP addresses, not involving the real source IP address in any of the logged messages. This makes it difficult to identify the real source of a port scan which is going on. Actually in this case it is impossible for us to identify the real attacker. In some situations it can even be that we reveal much more information to the attacker than we normally would. For example, a desktop firewall which allows the user to specify a trusted IP address and let traffic from it pass the desktop firewall unblocked, can be tricked into revealing information about the running services. If the attacker knows that a machine is a trusted machine, he can misuse this machine for performing a stealth idlscan against the desktop firewall. Since this is a trusted host the desktop firewall will not block the packets. This could lead to the real status of the ports leaking out to the attacker, assuming that the trusted remote machine, can be used for the idlscan. This opens the possibility of a distributed port scan from multiple sources, such that different sources are used to check for open ports on one target machine, even web based services like *hexillion* could be used for that [10].

A sophisticated port scanner, will not search the port range in a linear way. This, together with the fact not all attackers being interested in the same ports, indicate that it makes no sense to watch for a linear series of ports as a signature for port scans.

These thoughts lead to the following criteria:

> Indications for a port scan are that within a short period of time (first threshold) an amount of different ports (second threshold) get contacted.

The thresholds have to be adapted to the environment that the system is exposed to. Regarding the problems discussed above, it does not make sense to say that all port scan events will come from the same source, as they obviously don't have to. Still for reasons of completeness all available data should be sent to the engine that analyzes the event. It might be that the attacker did forget to hide its real IP address.

As we never will be able to tell if we have the real source IP address of the attacker, we should be careful with automated blocking features. If an attacker knows that a system will block each attack for one hour from accessing the target machine, it is possible for him to use this against the protected system. For example, the attacker can fake a port scan so that it looks like it is coming from the DNS server of the victim. The desktop firewall will then automatically block the DNS for one hour, and thus block the protected machine form using the DNS server.

6.3.3 Denial of service attacks

A denial of service (DoS) attack normally tries to flood a target machine with lots of bogus packets so that it will stop responding to legitimate wanted traffic. As the attacker is not interested in any response from the target, this is considered a one way attack, no feedback will be sent back to the attacker. This fact explains, why often spoofed packets are used during DoS attacks.

It is very difficult to distinguish between a port scan and a denial of service attack against TCP services, like a random SYN flood attack, because they simply result in the same messages. A SYN flood might result in more packets but if someone is scanning all the TCP ports this can also generate a lot of traffic. Therefore we are not able to define exactly what the cause is, but we detect that we are attacked.

6.4 Real world experiment

To have a real world test environment I did the following experiment. I searched for some Windows user within IBM Zurich Research Lab that would be willing to participate in my test. I asked them to have the Symantec desktop firewall running during a whole week and send me back the generated log files and their rule set. The activated rules are important too, as the people are all able to generate new rules or deactivate default rules. This could lead to the problem that events will not get logged on certain machines, because the rule set was modified not to do so.

The purpose of the whole experiment was to see if the correlation ideas introduced in Chapter 6 would really make sense and could work in a real world scenario.

After a week I was presented with the gathered logs and rule configuration files of 12 people situated on the same subnet within IBM. After parsing the log files into the generic event format, with the Perl scripts developed in this thesis, I ended up having around 6000 alarms that I could work with. For simpler processing of the data, I loaded them into a Excel table. Now I was able to perform the tasks that normally the security console would do, based on defined rules. I analyzed the data by hand to see if the methods worked or if they needed adaption.

By searching for events with the same source IP address and the same destination port, I started looking for suspicious alarms, filtering all the outgoing connections so that I would only see incoming events with those criteria. In addition I set a time window that filters events that have occurred within a window of one minute. The so found groups of events contained a lot of very suspicious events, such as the ones shown in Table 6.7. The IP addresses have been anonymized and unnecessary fields have been removed, for better readability.

time:	srcIP:	srcPort:	destIP:	destPort:
16:53:11	10.10.10.66	2703	10.10.10.134	80
16:53:14	10.10.10.66	2703	10.10.10.134	80
16:53:16	10.10.10.66	2728	10.10.10.159	80
16:53:16	10.10.10.66	2728	10.10.10.159	80
16:53:16	10.10.10.66	2771	10.10.10.222	80
16:53:19	10.10.10.66	2771	10.10.10.222	80

Table 6.7: Example IP scan log

Table 6.7 shows that the machine with IP address 10.10.10.66 scanned several machines in the network on TCP port 80. If we calculate the differences between the different source ports and the corresponding destination IP addresses, then we realized that they match. This can be verified in the following calculation:

$$2728 - 2703 = 25 = 159 - 134$$
$$2771 - 2728 = 43 = 222 - 159$$

This fact indicates that the attacker probably did a scan in sequential order of at least the IP address range from 10.10.10.134 to 10.10.10.222. Scanning the IP address 10.10.10.135 with source port 2704, IP address 10.10.10.136 with source port 2705 and so on. The problem is, with only having access to the desktop firewall log files shown in Table 6.7, it is not possible to tell if it was a port scan or a computer worm that searched for vulnerable web servers. I guess that it was an ordinary port scan, as most newer computer worms search in random order rather than in sequential order for new targets. But we cannot be sure.

The second search pattern that I used, was looking for events with same destination IP address and same destination port. Once again sorting them in time and looking at a small time window. An extract of the alarms found can be seen in Table 6.8.

time:	srcIP:	srcPort:	destIP:	destPort:
10:12:11	10.10.10.66	1511	10.10.10.134	80
10:12:14	10.10.10.66	1511	10.10.10.134	80
14:53:42	10.10.10.88	1254	10.10.10.134	80
14:53:45	10.10.10.88	1254	10.10.10.134	80
17:23:31	10.10.10.22	1397	10.10.10.134	80
17:23:34	10.10.10.22	1397	10.10.10.134	80

Table 6.8: Example computer worm scan log

In the whole aggregated data I found about 40 times a conspicuous set of events that contained 2 alarm messages in a time offset of 3 seconds using a source port number in the range of 1100 up to 1600. They all targeted the same destination port which is the web server TCP port 80. Similar series could be found for the other destination IP addresses, each time following the same pattern.

This leads to the assumption that this events could be generated by a computer worm that is randomly checking IP addresses for running web servers. Unfortunately, as mentioned before, I had no sniffer log of the analyzed time period, so I was not able to check whether it was a computer worm or not. For coming up with an exact result I would need other sources of information where the full packet is logged. But still we can identify those addresses as suspicious and even malicious. So we could take further steps to check those machines.

I also observed similar patterns with the grouping of 3 events instead of 2, those are probably related to a different computer worm. There was no misconfiguration in the network that could be observed in the analyzed log files.

By having a look at the rule sets, I found out, that most users did have the same default rules still enabled. In addition there were certain rules applied for specific applications, that where obviously used on those machines. This information could also be used to detect the use of disallowed software on those machines.

6.5 Conclusion

Log files of desktop firewalls are useful to correlate attack events. To be of even more value they should be extended in different directions. The logging feature of desktop firewalls should be modified in a way which allows easy further analysis on a centralized correlation engine. This could be achieved by sending the events as syslog messages or by writing them to a file in an easy parseble format. Another point is the implementation of stateful packet inspection filtering. With this feature it would be possible to add more useful information to the event message, for example the request string if it was a connection attempt on a web server. Furthermore it would enable us to set up precise rules which also look at the payload of the packet, making an even more finer rule tuning possible.

As long as the above mentioned points are not solved, desktop firewalls collaborating with intrusion detection system, seems to be the desired set up in my opinion. This combines the power of good attack identification from an intrusion detection system with the ability of protecting from unknown attacks of a desktop firewall. If then the generated event messages are analyzed in a central place and correlated, most attacks can be detected and prevented.

Chapter 7

Conclusions

Nowadays, when many people have a permanent Internet connection and more and more people are trying to act as attackers and harm others, the danger from the Internet grows. Of course, the security measures of the defender evolve too, and newer products are introduced to protect the systems. Desktop firewalls are one kind of the new products that fill a gap in the chain of protection tools. While mainly targeting end-users, they also have an application in companies on the desktop machines of the employees.

This thesis consists of two parts. The first part shows that there is a huge variety of attacks that target desktop firewalls, both from local or remote sources. Desktop firewalls do a fairly job in protecting against incoming attacks from remote machines, such as denial of service attacks. However some of the local attacks can not be prevented with the current definition of a desktop firewall. As long as the user is able to modify the rule set while running the desktop firewall, it will always be possible for a malicious application executed by the user to perform the same modifications as the user. This could be solved by running the desktop firewall process with administrator rights, like most anti-virus scanners already do. However this would remove the power that the desktop firewall is configurable by the user on the fly. This is the main problem of desktop firewalls, they run on a system where they can be attacked from the inside. Therefore, it is not possible to make anything that a malicious user with administrator rights can not change. This applies to log files, rule sets, and start up behavior of desktop firewalls. On the other hand, some of the attacks can and should be prevented by desktop firewalls, such as denial of service attacks with unmonitored protocols like IGMP. Therefore statefull packet inspection filter should be implemented in desktop firewalls.

The second part of the thesis reveals that it is helpful to use desktop firewalls as an additional information source. In general we can think of desktop firewalls as a cheap way of getting additional host-based sensor log files from distributed systems. Many companies have already desktop firewalls installed but do not use the generated log files. With the help of the tools developed in this thesis, it is possible to export those log files to a generic event format which is introduced in this thesis so that the events then can be integrated into an existing security framework. By using a common log file adapter it is possible to send the events to a correlation engine, for example, to the IBM Tivoli Risk Manager console for further analysis. At this centralized point the rules for correlation with other desktop firewall logs or with IDSs can be added, so that no information gets lost. The provided information can really support the decision of an IDS. By correlating the log events it is possible to increase the level of certainty to eliminate false positives and prioritize true positives. It has to be mentioned that the data of desktop firewall logs alone does not provide enough information for an exact identification of the attacks. For example, when we get attacked on the TCP port 80, we know that there was an attack but we can not tell for sure if it was a computer worm, a port scan or just a user that by accident typed our IP address in his browser. For clearing this situation we have to correlate the events with packet logs or with IDS logs, making it possible to analyze the payload of the packets. However this feature can be integrated in future versions of desktop firewalls.

This usefulness is also visible in the real world experiment, where multiple identical installations of desktop firewalls in a network were tested during one week. On the basis of the generated log files we were able to identify and classify different attacks during this time period. This includes computer worms which tried to spread in this network, *ad-ware* trying to send information back to its vendor and attackers port scanning the network for possible weaknesses.

Chapter 8

Future Work

This work could be extended in different directions.

One potential immediate step is to look at the correlation engine where the generated events come together and check, how this new information source influences the quality of the detection, and whether it decreases the number of false positives.

Analyzing the information logs of desktop firewalls such as starting time of desktop firewall, which where not processed in this thesis could be of interest as well, maybe more for general system health checking or as a method of checking if the systems where rebooted or modified in an unauthorized fashion. At least it is data that is rarely used and freely available and can give information about the monitored system.

Another idea is to check if the problems mentioned in the conclusions, such as better protection from incoming denial of service attacks or prevention from changes in the local rule set and log files, were solved in upcoming versions.

In addition it could be of interest to create a concept of an ideal desktop firewall, and maybe even implement it. As we see a huge demand for good, reliable, stable but also easy to use desktop firewalls. This ideal desktop firewall should solve the problems with the issue that the desktop firewall can be attacked from the system it is running on. In addition it should address features like outgoing spoofing protection, rule set altering prevention and denial of service protection.

An unanswered question is also if it is possible to detect which version of a desktop firewall is installed on a target machine and which rules are enabled. This information could lead to more sophisticated attacks exploiting specific vulnerabilities of the desktop firewall following the first attack.

Another topic is to check if it is possible to write firewall rules in a generic format and then translate them into the specific format of different firewall applications and install them in their rule set. The idea behind this is to have one single rule file and apply it to all different kind of firewalls that are installed in an environment.

Chapter 9

Acknowledgments

First of all I want to thank Prof. Dr. Bernhard Plattner, my supervisor from ETH Zurich for once again supporting me and my project. It is amazing how he always supports new ideas with his friendly and positive attitude.

I would also like to thank Marc Rennhard, my tutor at ETH Zurich for managing the coordination of ETH with IBM Zurich Research Laboratory, and also for the friendly talks we always had when I met him. As always he has been very helpful and understanding.

Also I would like to thank Dr. Andreas Wespi, manager of the Global Security Analysis Lab group (GSAL) at the IBM Zurich Research Laboratory. He always found some spare time to answer my questions and to come up with new ideas.

My special gratitude goes to my advisor Dr. Diego Zamboni at the IBM Zurich Research Laboratory. He always cared about my concerns and helped me in all situations with a good advice. I thank him for being a supporter and a good friend to me and for taking the time for proof-reading my thesis.

I thank the whole GSAL team for always supporting me with good advice and helpful discussions and my co-workers, especially Frederik Schaller, for shortening the work time with interesting discussions.

Candid Wüest
Rüschlikon, February 2003

Appendix A

Assignment

Diploma Thesis
for
Candid Wüest (D-INFK)

Main Reader:	Dr. Diego Zamboni (IBM Rüschlikon Research Lab)
Alternate Reader:	Marc Rennhard
Issue Date:	1st November 2002
Submission Date:	28th February 2003
Title:	**Desktop Firewalls and Intrusion Detection**

A.1 Introduction

During the past few years, *desktop firewalls* have become more and more popular. In contrast to the traditional model where firewalls are recommended to be installed on dedicated, well-controlled machines, desktop firewalls are installed on end-user machines, together with a wide variety of tools and applications.

Regarding the increasing number of home users, desktop firewalls have been the next logical step in software evolution, as most home users are not experienced enough to set up a real firewall or are not willing to use a dedicated machine for it. Desktop firewalls are easy to set up, but they are not as powerful as normal firewalls. They are a trade-off between security and usability.

A.2 Motivation

The motivation for this thesis is twofold. First of all, it is not clear how much additional security desktop firewalls provide compared to a system that does not run a firewall at all. It could well be the case that desktop firewalls are vulnerable to similar types of attacks as the operating system on which they run.

Second, several corporations have desktop firewalls installed on many end-systems. This leads to a huge information source that normally isn't used. Correlating this information, one could for instance find out if a specific attack such as a port-scan was targeting just one machine or the whole network. Using this information could make it possible to rate events and decide if they are important or not. Another benefit is that one could reduce the number of generated alarms in the sense that only one event is generated by a port-scan instead of 255 events. Even some e-mail worms could be detected and prevented in this way.

A.3 Assignement

A.3.1 Objectives

The goal of this diploma thesis is to get a better understanding of the strengths and weaknesses of desktop firewalls and how desktop firewalls can be used to monitor the security state of a computer network. Often several desktop firewalls are deployed in a computer network. Therefore, it is not only of interest what a single desktop firewall can achieve but also what a combination of desktop firewalls can do. It can also be imagined that desktop firewalls provide information that is very useful in combination with other types of information such as the alarms generated by intrusion detection systems (IDSs). This is another area that should be investigated in the context of the diploma thesis.

A.4 Tasks

The following tasks have to be fulfilled:

- Select at least two different desktop firewall products, witch represent the most used in market. They should be from different vendors and can but donŠt have to, run on different operating systems. The aim is to see what information they provide and if it is possible to combine the – probably different – strengths of the different products.
- Analyze how easy or difficult it is to circumvent these desktop firewalls.
- Study the robustness of the desktop firewalls themselves against attacks. This includes the following two subtask:
  - Collect and analyze the resistance to known attacks, using security mailing lists as information source and testing with released exploits.

  - Design new attacks and analyze the resistance of desktop firewalls by modifying existing exploits, or by using discovered vulnerabilities to create new attacks.

- Evaluate how useful desktop firewalls are in detecting and defending against attacks. In particular, the following three areas should be addressed:
  - What are the logging capabilities of desktop firewalls?
  - What information is recorded and how easily can the recorded information be made available to other security tools?
  - What kinds of attacks can be detected by a single desktop firewall or a combination of desktop firewalls?
  - How can the information as provided by desktop firewalls be correlated with other types of security information such as alarms from intrusion detection systems?

- Write a small application that is able to read the different log files and generate event messages for IDS correlation tools.

A.5 Deliverables

- At the end of the second week, a detailed time schedule of the semester thesis must be given and discussed with the advisor.

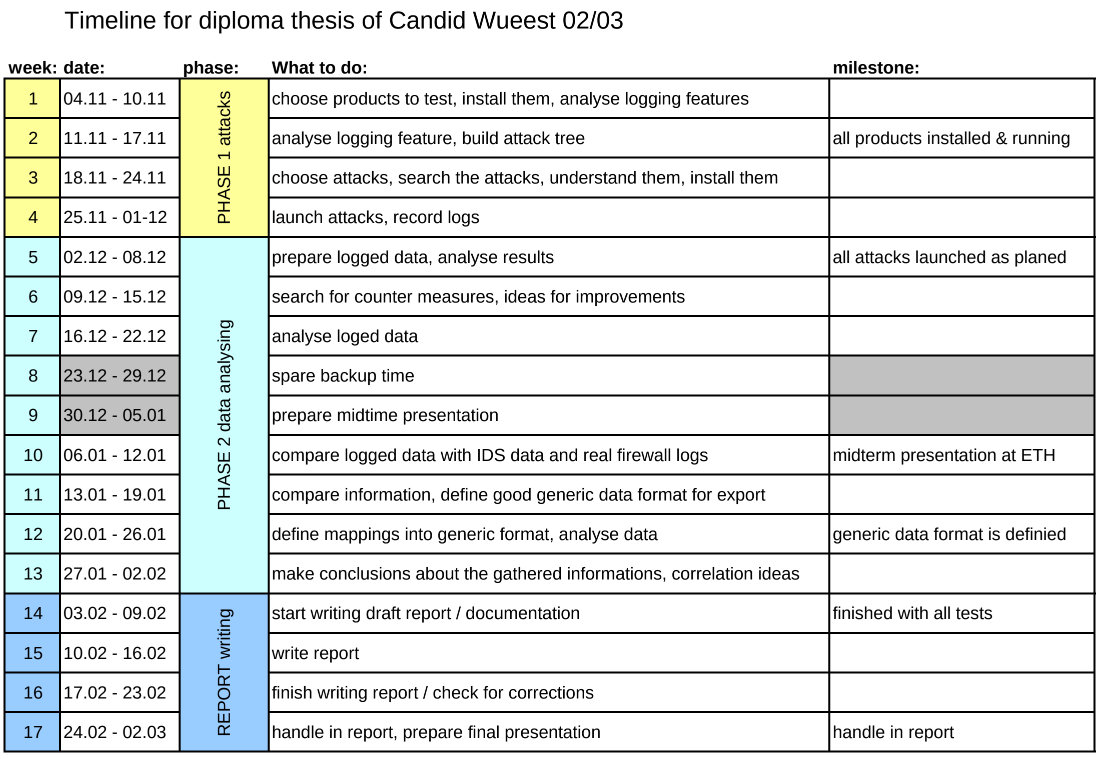
- At half time of the semester thesis, a short discussion of 15 minutes with Prof. B. Plattner and the advisors will take place. The student has to talk about the major aspects of the ongoing work. At this point, the student should already have a preliminary version of the written report, including a table of contents. This preliminary version should be brought along to the short discussion.

- At the end of the semester thesis, a presentation of 20 minutes must be given during the TIK or the communication systems group meeting. It should give an overview as well as the most important details of the work.

- The final report must be written in English. It must contain a summary written in both English and German, the assignment and the time schedule. Its structure should include an introduction, an analysis of related work, and a complete documentation of all used software tools. Two copies of the final report must be delivered to TIK.

Appendix B

Project Schedule



Timeline for diploma thesis of Candid Wueest 02/03

week:	date:	phase:	What to do:	milestone:
1	04.11 - 10.11	PHASE 1 attacks	choose products to test, install them, analyse logging features	
2	11.11 - 17.11		analyse logging feature, build attack tree	all products installed & running
3	18.11 - 24.11		choose attacks, search the attacks, understand them, install them	
4	25.11 - 01-12		launch attacks, record logs	
5	02.12 - 08.12	PHASE 2 data analysing	prepare logged data, analyse results	all attacks launched as planed
6	09.12 - 15.12		search for counter measures, ideas for improvements	
7	16.12 - 22.12		analyse loged data	
8	23.12 - 29.12		spare backup time	
9	30.12 - 05.01		prepare midtime presentation	
10	06.01 - 12.01		compare logged data with IDS data and real firewall logs	midterm presentation at ETH
11	13.01 - 19.01		compare information, define good generic data format for export	
12	20.01 - 26.01		define mappings into generic format, analyse data	generic data format is definied
13	27.01 - 02.02		make conclusions about the gathered informations, correlation ideas	
14	03.02 - 09.02	REPORT writing	start writing draft report / documentation	finished with all tests
15	10.02 - 16.02		write report	
16	17.02 - 23.02		finish writing report / check for corrections	
17	24.02 - 02.03		handle in report, prepare final presentation	handle in report

Appendix C

Desktop firewall rule editor details

The following Section explains all the options of the desktop firewall rule editors. As Zonealarm does not have a rule editor this desktop firewall will be skipped.

C.1 Symantec desktop firewall rule editor

These are the options that can be configured, when adding a custom rule in Symantec's desktop firewall.

- **Name:** The name of the new rule, as it should appear.
- **Action:** This can be either permit, block or ignore and defines what should be done with the traffic.
- **Direction:** Outbound, inbound or either, specifies the direction of traffic where this rule should be applied to.
- **Protocol**: This can be either TCP, UDP, TCP/UDP or ICMP and specifies the protocol that should be checked.
- **Application:** This can be a full path to an application including its name or "ANY" which means that no filtering will be applied.
- **Service:** There are four option each for remote and for local service:
  1. *single service*, defines one port number.
  2. *service range*, defines a port range.
  3. *list of services*, defines multiple port number, does not have to be a connected range.
  4. *any service*, specifies any port number.

- **Address:** There are four options for remote address:

  1. *remote address*, defines one IP address.
  2. *network address*, defines a network address including subnet mask.
  3. *address range*, defines multiple IP addresses.
  4. *any address*, defines any IP address.

  and two options for local address:

  1. *host address*, defines one IP address.
  2. *any address*, defines any IP address.

- **Logging:** Here we can specify to write a log event when this rule was matched. An interesting feature is, telling the desktop firewall to wait for a certain number of the same event before string to log it. This threshold can help eliminate false positives. In addition we can have it raise an alarm when the rule was matched, which then will be displayed in the task bar icon of the desktop firewall.

C.2 Sygate personal firewall rule editor

These are the options that can be configured, when adding a custom application rule in Sygate's personal firewall.

- **Name of Application:** Full path of the application.
- **Trusted IP addresses:** An IP address or a range of addresses that should be trusted. Any traffic coming from a trusted IP address will not be blocked for this application.
- **Remote ports:** Specifies a TCP or UDP single port or port range for the remote machine.
- **Act as client:** This means that it will be possible to send traffic to the configured ports.
- **Local ports:** Specifies a TCP or UDP single port or port range for the local machine.
- **Act as server:** This means that this application will open a listening port for incoming connections.
- **Allow ICMP traffic:** Enables this machine to use ICMP traffic.

- **Allow during screensaver mode:** If not checked, the application will be blocked during screensaver mode.
- **Enable scheduling:** By enabling this option the advanced rule will be limited to some time intervals. Specified by month, day, hour and minute. For example, this can be used to block all outgoing traffic after 19:00 when the office machine should be unused.

Another feature are the advanced rules, which can be configured in five different sections, specifying the characteristics of the rule.

- **General:** Specifies general information for the new rule.
  1. A name for the new rule.
  2. Choose the action to perform on the traffic: allow or block it.
  3. Specify the network interface card which we want this rule to applied to.
  4. Tell if it should apply in screensaver mode.
  5. Enable recording the event to the log file.
- **Hosts:** Specifies the remote host for which this rule should be applied. This can be one of the following:
  1. *All addresses*, means no filtering on IP addresses.
  2. *MAC address*, defines the MAC address of the machine.
  3. *IP addresses*, defines a single or multiple IP addresses.
  4. *Subnet*, defines the subnet including subnet mask.
- **Ports and Protocols:**
  1. *Protocol*, selects the protocol to watch, either ALL, TCP, UDP, ICMP, or IP Type.
  2. *Remote port*, specifies the remote port of the traffic.
  3. *Local port*, specifies the local port of the traffic.
  4. *Traffic direction*, specifies the direction to monitor, either incoming, outgoing or both.
- **Scheduling:** By enabling this option the advanced rule will be limited to some time intervals, specified by month, day, hour and minute.
- **Applications:** With this option we can specify the applications that should be affected by this rule.

Appendix D

Logfiles

The log files generated during the attack experiments in this thesis are included on the attached CD-ROM and in the Internet at URL
http://www.trojan.ch/DFW/

Bibliography

[1] M. Ranum, *Using desktop firewalls as basic IDSes*, 2002.
http://www.infosecuritymag.com/2002/oct/cooltools.shtml

[2] *Log file analyzer project on security focus*, 2003.
http://analyzer.securityfocus.com

[3] S. Boran, *An analysis of mini-firewalls for Windows Users*, 2003.
http://www.boran.com/security/sp/pf/pf_main20001023.html

[4] G. Bahadur, *Article of attacks against personal firewalls*, 2001.
http://www.infosecuritymag.com/articles/july01/cover.shtml

[5] *Leak test against desktop firewalls*, 2002.
http://www.pcflank.com/art21.htm

[6] *DShield, distributed intrusion detection system project*, 2003.
http://www.dshield.org

[7] *General informations about computer viruses*, 2003.
http://www.virusbtn.com

[8] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, N. Weaver, *The spreading of the Sapphire/Slammer worm*, 2003.
http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html

[9] Fyodor, *Information about idlScans with nmap*, 2002.
http://www.nmap.org/nmap/idlescan.html

[10] *Hexillion's online information gathering tool*, 2003.
http://www.hexillion.com

[11] *ProcessXplorer, a process explorer from sysinternals*, 2002.
http://www.sysinternals.com

[12] T. Bird, *This is a collection of links related to log file anayzer*, 2002.
http://www.counterpane.com/log-analysis.html

[13] *Analysis and surveys about computer security*, 2002.
http://www.cert.org/analysis/

[14] R. Graham, *Explains alarms seen in firewall logs*, 2000.
http://packetstormsecurity.org/papers/firewall/firewall-seen.htm

[15] M. Rauen, *Website with low level system programming source code*, 2002.
http://www.madshi.net

[16] W. Hua, J. Ohlund, B. Butterklee, *Microsoft's introduction to layered service provider*, 1999.
http://www.microsoft.com/msj/0599/layeredservice/layeredservice.htm

[17] D. Brent Chapman & Elizabeth D. Zwicky, *Building Internet Firewalls* , 1995, O'Reilly & Associates.

[18] DiamondCS, *How to shutdown zonalarm from a batch file*, 2000.
http://www.diamondcs.com.au/alerts/zonedown.txt

[19] L. Bowyer, *Article of tunneling with HTTP protocol and stegano messages in pictures*, 2002.
http://www.networkpenetration.com/protocol_steg.html

[20] Internet Security Systems, *IGMP fragmentation bug*, 1999.
http://www.iss.net/security_center/static/2341.php