Intrusion Detection System Sensors

Extension of the SURFnet Intrusion Detection System Sensors to Microsoft Windows XP

Projectmentors:
R. Spoor
J. van Lith
K. Trippelvitz

Projectmembers:
Ing. Michael Rave
Ing. Coen Steenbeek

Version 1.0

February 2, 2007

Abstract

The SURFnet IDS is a distributed IDS, based on a client-server principle. The server-side runs honeypot and tunnel software. The client-side is represented by a dedicated sensors, which create tunnels to the server.

The functionality of the IDS system will be extended with this project. The target of this project is the development of a non dedicated Windows-based IDS Sensors, that can be used in the current infrastructure. During the project two possible solutions for the Windows Sensors were designed.

When a Windows sensor is attacked, in the first solution, it will forward the traffic to the "old-style" sensor, which will then forward it to the honeypot. The second solution is based on the same technique as the current sensor, namely to open a tunnel to the honeypot.

Both solutions were implemented and tested, and at this moment we recommend to use the direct second solution, because it is already been tested in a test environment and only needs to be tuned for optimal performance. Further research and testing needs to be done, before either of the solutions can be introduced in the production environment.

Acknowledgments

We would like to extend our gratitude to the following people and organizations for their assistance in our research.

SURFnet, Utrecht for giving us the time and resources to make this project possible

SURFnet IDS Team for the feedback and support during the project

Contents

Chapter 1

Introduction

This document describes the SURFnet IDS Sensors project. Starting with a little introduction about IDS's(Intrusion Detection System), that explains what an IDS can do and what kind of IDS's are available. After this introduction the current solution of SURFnet IDS will be described, followed by the research question and goal of this project.

The information that was found during the project, will be discussed in chapters 5 and 6. After the research two solutions were proposed, an indirect and a direct solution, which are described in the following chapter. Finally the conclusion of the project, how to intergrate these solutions and what future work must be done to complete the Windows sensors.

All our used sources can be found in the bibliography on page 22, for additional background information. The source of the open-sources tools that were used, can be found in the appendixes.

Chapter 2

Intrusion Detection Systems

To get a clear understanding of the distributed SURFnet IDS the basics of on IDS system will be described first. An Intrusion Detection System (IDS) is a system that can analyse and detect attacks. An IDS cannot stop hackers or crackers, it only gives information to understand the behaviour of intruders. This information can be used to prevent against future attacks or to decrease damage when an attacks occurs.

An IDS consists of several components. The *sensors* generate security-events. The *console* listens to these events, which are sent to the console by the sensors, and it also controls the sensors. All data is stored by the central *engine* and is for example correlated. The data can be used to get a better insight in the behaviour of the hackers, crackers, worms, bots, e.c. There are different manners to set up these components. This differences in setup is what distinguishes the different types of IDS's.

2.1 What are the differnt types of IDS's?

There are about five different types of IDSs according to Wikipedia[1], but the Network, Host-based and Hybrid IDS are used most commonly. No examples of an implementation of a protocol based IDS were found, and the protocol based IDS utility Secerno is no dedicated IDS, to indicate that these IDS's are not commonly used. The different types can be found in table 2.1.

Intrusion Detection System	**Scans**	**Example**
Network IDS	Network traffic and monitors multiple hosts	Snort
Protocol Based IDS	Certain protocol(s) for a specific service	N/A
Application Protocol Based IDS	Communication on application specific protocols	Secerno
Host-Based IDS	Internals of a computing system	Tripwire
Hybrid IDS	Combines one or more approaches	Prelude

Table 2.1: The different types of IDSs

Snort[2] is the de-facto standard for Network IDSs. It scans all incoming and outgoing network traffic. The traffic is compared with several rule sets. A security event occurs when the incoming or outgoing traffic matches a rule. This event is logged in a file or a database which can be used for better analysis.

Tripwire[3] is a tool that can analyse the hard disk of a computer. It can see whether the hard disk has been changed and what has been changed. This information gives a better understanding in the actions of an attacker when a computer system is compromised.

Hybrid IDS's like Prelude[4] combine several approaches in one solution. Prelude uses IDMEF (Intrusion Detection Message Exchange Format) to centralize the data which is generated by the different kind of IDS sensors (for instance Snort and Tripwire).

More information on the different IDSs can be found on the homepages of the different tools, which can be found in the bibliography of this report on page 22.

[1] `http://www.wikipedia.org`

Chapter 3

The SURFnet IDS

The SURFnet IDS[5] is a distributed IDS, based on a client-server principle. The client implements the IDS sensor and the server implements the engine and console.

The current sensor is a normal workstation that creates a layer-2 (OSI) VPN[23] tunnel to the IDS server. This is done by a modified version of the operation system Knoppix[1]. The current size of this operation system is approximately 150MB and is being spread on an USB-stick. It is required that the machine should be able to boot from an USB device. The layer-2 tunnel is set to bridging-mode so that the server is virtually located next to the client.

On the server runs a honeypot called Nepenthes[6], which simulates vulnerabilities on the layer-2 tunnels to the sensors. When an attacker takes advantage of these weaknesses, Nepenthes tries to communicate with the attacker in order to collect as much information about the attacker and the attack as possible. This information is stored in a PostgreSQL database on a logging server. In combination with Apache web service the information can be viewed through a web interface and conclusions can be drawn or they can take action to prevent new attacks.

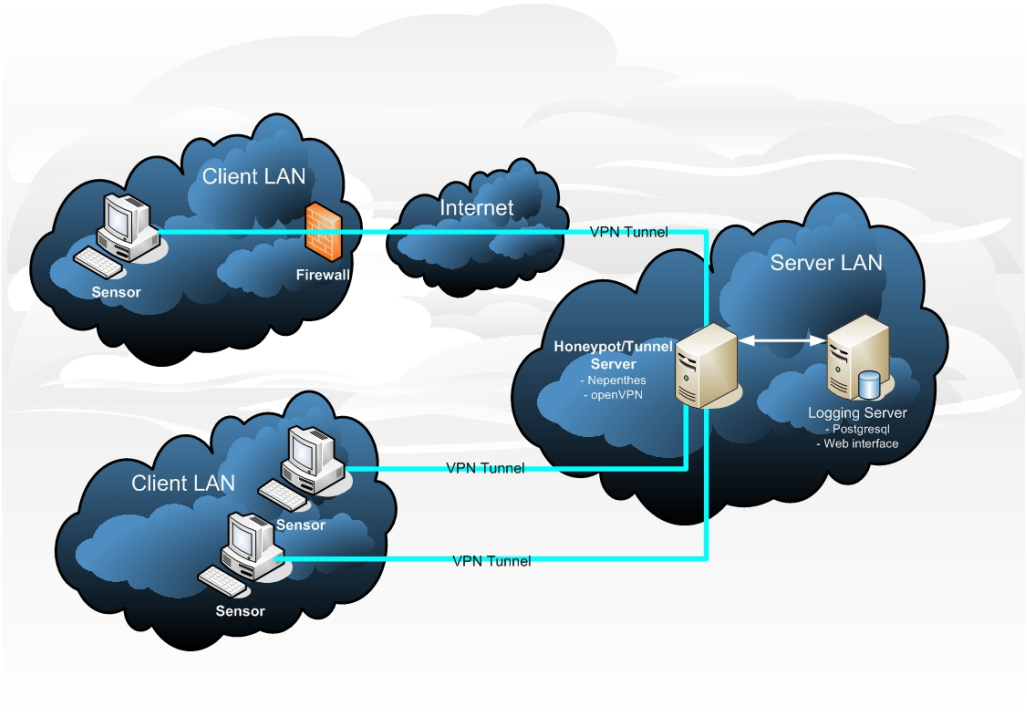

Figure 3.1: The distributed SURFnet IDS[25]

[1]For more information: `http://www.knoppix.org`

The sensors are scattered across the Netherlands. The owners of the sensors are able to go to the web interface, where information about there sensors can be found. For privacy reasons they only see there own sensors, not the sensors owned by other organisations. However, the administrators of the SURFnet IDS are able to see the information of all sensors.

The owners of the sensors decide themselves what they do with the information. They can for example take action to protect their cooperation from new attacks.

The next chapters describe how the SURFnet IDS solution can be extended, so that the dedicated (remastered)Knoppix machines are no longer a must for an organisation. When it is possible to install a sensor on a normal workstation, it is easier for an organisation to install multiple sensors in their network, so that more traffic can be analysed. The normal workstation can still be used normally, in contrast to the current, dedicated sensors.

Chapter 4

Problem Definition

SURFnet IDS is an open-source project set up with the assistance of SNE[1] (System and Network Engineering) students of the University of Amsterdam. At this moment the project is used by organisations in the United States, Germany, Sweden, Norway, Japan and Australia. The current version of the SURFnet IDS is a distributed solution. SURFnet developed it as a service for the organisations connected to SURFnet. At the moment there are about 40 unique sensors active in the Netherlands, United States, Australia and Sweden. Additional functionality of the IDS system is still being developed and extended. Because of that, our project is set up. To make the system more efficient investigation had to be conducted how to use the unused desktop computer ports as a sensor and connect them to the SURFnet IDS server. Research has to be done, how this solution can be build and can be implemented within the current infrastructure.

4.1 Research question

The purpose of this project can be summarized in one question, this question will be answered during the project.

> *How to give a desktop computer the same functionality as the old-style SURFnet IDS sensors without affecting the current functionality of the desktop computer?*

Thereby the following sub-questions will be utilized:

- How to obtain unused ports on Windows XP (Chapter 5)
- How to forward certain ports on Windows XP (Chapter 6)
- How to forward incoming traffic on certain ports to the honeypot without changing the source IP-address of the incoming packets. This will be the overall infrastructure of the new solution. (Chapter 7)

Chapter 3 described how the current sensor operates. A solutions has to be found to give the desktop computer the same functionality as the modified operation system Knoppix. Because the most clients run Microsoft Windows XP, the project is focussed to this operation system.

4.2 Research goal

At this moment a sensor costs IP-space and a dedicated workstation. When a sensor can be installed as a service on a normal desktop computer, all of the previous aspects are no issue any more, because the machine can still be used as a workstation. It is also possible to use more sensors at the same time. Also the quantity of the information that is collected with the current solution is less than with the new solution. This is because there are only a few, one till three systems, active as a sensor within an organisation. When sensor software can be installed on random desktop computers, more information can be gathered and more conclusions can be drawn. It also gives a better overview of the whole infrastructure of an organization, because sensors can be placed within the different network sectors.

[1] For more information: http://www.os3.nl

4.3 Start

This research started out to find a way how to find the unused ports on a normal Windows system. With that information ports can be forwarded to the current honeypot setup of SURFnet. With this information it is possible to design a solution to implement the Windows sensors.

Chapter 5

Unused ports

The first step of this project is determined how to detect which TCP and UDP ports are unused on a Windows workstation. It is easier to determine the used ports than to obtain the unused ports, since a normal workstation uses only twenty out of approxomately sixty-five thousand ports. Several solutions were found and tested, and the different options were listed in this chapter. With the use of table 5.1 is tried to compare the different solutions so the best options can be chosen.

5.1 Netstat

Netstat[9] is a standard Microsoft Windows XP tool. In a command shell the command netstat can be entered to get a listing of all the used TCP and UDP ports. Netstat can be used to show the active TCP-connections, the ports that the system listens on, Ethernet-statistics, the IP-routing table, IPv4- and IPv6-statistics. However, Netstat also works with the different Linux/Unix and Mac OS operating systems.

5.2 TCPView/TCPVcon

TCPView[10] is a tool that can be downloaded on the Microsoft website. It is originally written by Sysinternal, but this organisation has been annexed by Microsoft. TCPView shows detailed information of all TCP and UDP endpoints on a workstation in a graphical user interface. It also shows the local and remote addresses and the stat of the TCP connections. TCPVcon is the command-line version of TCPview, which gives the same information as the GUI version.

5.3 Netstatp

Netstatp[11] is the predecessor of netstat and also created by Sysinternals founder Mark Russinovich. The source is available on the internet. With the use of the source the C-code can be modified, so that it only gives the information that is needed in this project.

5.4 Fport

Fport[12] shows all open TCP and UDP ports per application. It is the same information that is generated with the *netstat -an* command, however Fport maps the used ports to the running processes with their process id, name and path.

5.5 Winpcap

Winpcap[13] is the Windows API verion of the well-known linux API libpcap. Winpcap gives a link-layer network access on a low-layer (kernel) level. This can be used by applications to for instance filter or

analyse the network traffic.

5.6 Windump

Windump[14] is the Windows version of Linux tcpdump tool. Windump uses the Winpcap library and drivers. It is also possible to download and modify the source code of Windump. A disadvantage, of this utility, is that it can only check the used ports if data is being send through these ports.

5.7 Nmap

Nmap[15] is an open-source program that can be used to check the security of a network. It has been written in C-code, so can be compiled for Windows too. Nmap can also be used to check the ports on a single host.

5.8 Filter driver

A filter driver[16] is a driver for a device that can be used to obtain specific information about that device. In our case it will mean that we would use a driver for the ethernet-device to check the(un)used ports.

5.9 Comparison

The distinguishing functions of the different tools are summarized in table 5.1. With the use of this table it is possible to choose the best tool for this project. The functions that were tested on are the following:

1. Open-source: if the source code can be obtained it can be modified to benefit the requirements of this project
2. Command-line: it is easier to modify command-line tool than a tool with a GUI (Graphical User Interface). The command-line tool can be scripted for instance.
3. Free: is it free to use.
4. Independent: does it need other system resources/tools?
5. Simple: is the source complicated or can it easily be adapted?
6. Fast: is it fast or a slow tool that uses a lot of resources?

Tool	**Open-source**	**Command-line**	**Free**	**Independent**	**Simple**	**Fast**
Netstat	-	+	+	+	+	+
TCPView	-	+	+	-	+	+
Netstatp	+	+	+	+	+	+
Fport	-	+	-	-	+	+
Winpcap	+	+	+	+	-	-
Windump	+	+	+	-	+	+
Nmap	+	+	+	+	+	-
Filter driver	n/a	n/a	+	-	-	n/a

Table 5.1: Comparison of the different (un)used port tools

5.9.1 Conclusion

Netstatp is the tool with the most positive aspects, as can be seen in table 5.1. Therefor this tool will be used in the new Windows-based sensor. With the use of the source code, modifications can be made so it only gives the information that is required. The information obtained by Netstatp can be used to verify if a port can be opened for use in the distributed IDS solution. The next chapter will discuss how the ports can be opened and all data can be deferred to the honeypot Nepenthes.

Chapter 6

Port Forwarding

Now the unused ports can be obtained with a modified Netstatp, these can be forwarded to the honeypot. The forwarding can be done by port forwarding applications. Some of the proprietary and open-source port forwarding utilities will be discussed in the following paragraphs.

6.1 Wintunnel

Wintunnel[20] is an open-source program written in Microsoft .NET and is originally used as a proxy service. That is why it has many functionalities. It is very simple to use Wintunnel as a port forwarding program. Wintunnel has the capability to be installed as a service so it can be easy started and stopped. When Wintunnel starts it will look for a file called WinTunnel.ini. This is the configuration file of WinTunnel and contains information such as local port, host IP-address, en host port. When another port needs to be forwarded, the service has to be restarted in order to make it work. The program is open-source, so it can be modified.

6.2 Porttunnel

Porttunnel[18] is a product of Steelbytes and can make port mappings. It has options for SMTP, FTP, SSL and TLS. It can also function as a proxy. Like Wintunnel, Porttunnel can also be installed as a service. A disadvantage about this product is that it is proprietary, so the source cannot be modified. The cost of Porttunnel is $100 for a server licence. All the extras it comes with are unneeded.

6.3 AUTAPF

NetworkActiv[17] is a company that has created various network based tools, like a webserver, a network monitor – like WireShark – and AUTAPF. It is not clear what AUTAPF means, but 'PF' probably means Port Forwarding. AUTAPF is a proprietary tool, which can forward selected ports to another ip and port combination with the use of a GUI. The tool can run as a service, however, to enable this function a certain payment has to be made. Disadvantages for us are the following: the tool cannot be modified and it is not possible to send command-line commands to the tool to add or delete port forwardings.

6.4 Trivial Port Forward

Trivial Port Forward[21] is an open source tool written in the program language C++. The website of the tool is basic and contains not much information. The use of Trivial Port Forward is very simple. A portforward can be added by the following command: */portforward [locale poort] [host ip] [host port].* During the testing phase of the tool it worked easy and benefited all the requirements for this project. However when the connection was closed on the host side of the port forward the tool would hang and use 100% cpu. So it still has a few bugs, but this tool is open source and can be modified.

6.5 TCP port rerouter

TCP Port Rerouter[19] is a port mapper written in Visual Basic and uses the Windows API. The product supports bridging and rerouting of a port. It cannot run as a background service, but will remain in the system tray.

6.6 Netsh

Netsh[22] is a command-line scripting utility that allows displaying or modifying the network configuration of a computer. This can be done locally or remotely. Netsh interacts with other operating system components using dynamic-link library (DLL) files. Each DLL helper file provides extra features and specific networking components command. For example, Dhcpmon.dll provides netsh the context and set of commands necessary to configure and manage DHCP servers. Netsh can simply be started by running 'netsh' in a command-line. A disadvantage of Netsh is that the Windows service, *Routing and Remote access* needs to be enabled. When this service is enabled, a reboot is required in order to make NAT work.

6.7 Comparison

Table 6.1 summarizes the distinguishing functions of the different utilities/solutions. With the use of this table it is possible to choose the best tool for the Windows sensors. The functions that are tested, are the following:

1. Open-source: if the source code can be obtained it can be modified to benefit the requirements of this project
2. Command-line: it is easier to adapt a command-line tool than a tool with a GUI. The command-line tool can be scripted for instance.
3. Free: is it free to use.
4. Independent: does it need other system resources/tools?
5. Simple: is the source complicated or can it easily be adapted?
6. Fast: is it fast or a slow tool that uses a lot of resources?

Tool/Solution	**Open-source**	**Command-line**	**Free**	**Independent**	**Simple**	**Fast**
Wintunnel	+	+	+	-	-	-
Trivial Port Forward	+	+	+	-	+	+
AUTAPF	-	-	-	+	-	+
TCP Port Forward	-	-	-	+	-	+
Porttunnel	-	-	-	+	-	+
Netsh	-	+	+	-	+	+

Table 6.1: Comparison of the different port forward tools

Trivial Port Forward is the tool that has the most positive aspects, so it would be natural to choose this option as our port forwarding utility. However, during the testing of the utility some limitations were exposed. These limitations are:

- Trivial Port Forward still has some bugs which have to be solved
- It is only possible to forward one port, so multiple processes have to be started in order to forward multiple ports
- Trivial Port Forward operates on layer-4 so the source IP-address of the forwarded packets are changed to the source IP-address of the host running Trivial Port Forward

Therefor Netsh is chosen to be used for the Windows sensors, because this is a standard utility of Windows and it can do pre-routing NAT (so it does not change the source IP-address).

Chapter 7

Solutions

Now that several possibilities for port forwarding and obtaining the (un)used ports of a workstation were found, a new solution for the Windows Sensors can be created. But how can the data be sent to the honeypot? Two different solutions were designed for this problem, which will be described in the next sections. After the development both solutions were implemented, so that they could be tested in actual practice.

7.1 Indirect Solution

This solution uses two different sensors, as can be seen on the left side of figure 7.1. The first sensor is the sensor which is attacked by the attacker, the sensor *Sensor Client*. The second sensor is the old-style sensor, the *Sensor Server*. The task of the Sensor Server is creating the layer-2 tunnel to the honeypot Nepenthes.

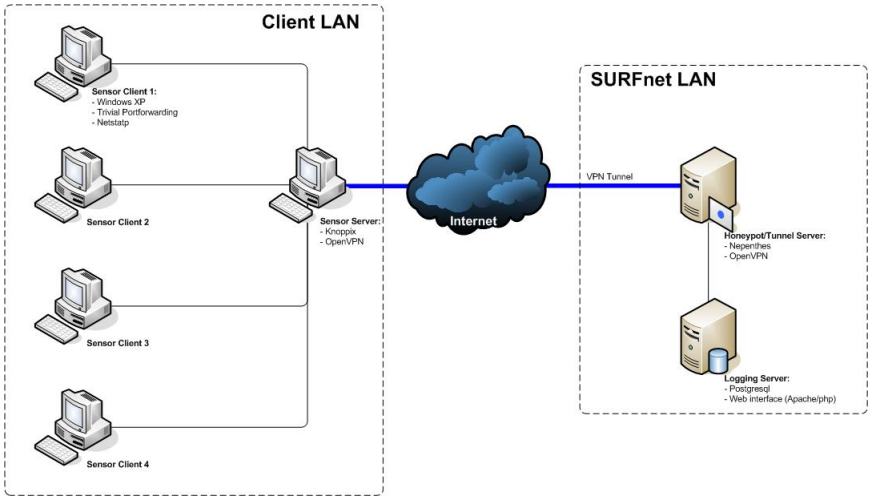

Figure 7.1: The indirect sensor solution

7.1.1 Sensor Client

The Sensor Client is the sensor which will be under attack. With the use of Netstatp and port forwarding this sensor sends the data to the honeypot. (Actually the data is send to the sensor server which sends it over its layer-2 tunnel to the honeypot.) The Client Sensor has to save the IP-address of the attacker so it can return the data to the attacker, when the traffic returns from the honeypot[1]. A big advantage

[1] Which can be compared to NAT

of this solution is that the Sensor Server does not need to be modified in any way. It only has to create the layer-2 tunnel, so the Sensor Client thinks that it can contact the honeypot server directly.

7.1.2 Modifications

To make this solution work, some modifications have to be made to the current system. The honeypot needs to know the IP-address of the attacker, so the sensor client needs to send this to the honeypot, as well as the name and IP-address of the sensor client itself so the honeypot can log this. The possibilities to solve this problem will be discussed in section7.3.2

The new Sensor Client is a Windows service that forwards the data which is sent to the unused ports of this client. The unused ports are obtained with the use of Netstatp. This data is then used by the port forward utility. The port forward utility sends all data which is send to the free ports to the Honeypot. The honeypot returns the data and the portforwarder moves the IP-packets back to the attacker.

7.2 Direct Solution

In this solution all sensors, just like the current sensors, make a VPN tunnel connection to the SURFnet honeypot server. The IP-addresses in the picture are not real IP-address but are only for better insight in the situtation.

7.2.1 Sensors

Each sensor has a network card with a public IP-address that is connected to the Internet. Also each sensor has a virtual network card with a static private IP-address, which is connected to the honeypot through the tunnel. Because Windows XP only supports a TAP virtual network card it is not possible to use a TUN virtual network card. TUN is a Virtual Point-to-Point network device and supports IP tunnelling. TAP is a virtual Ethernet network device and supports Ethernet tunnelling. For each sensor a new TAP device has to be created on the server. Each TAP device has a static IP-address. Because these clients are running Windows XP it is not possible to use the old Knoppix solution. When the Attacker attacks, for example, Sensor 1 will check with Netstatp if the port is in use. If the port is not in use it will forward the request with the use of Netsh to the honeypot through the virtual network card and VPN tunnel.

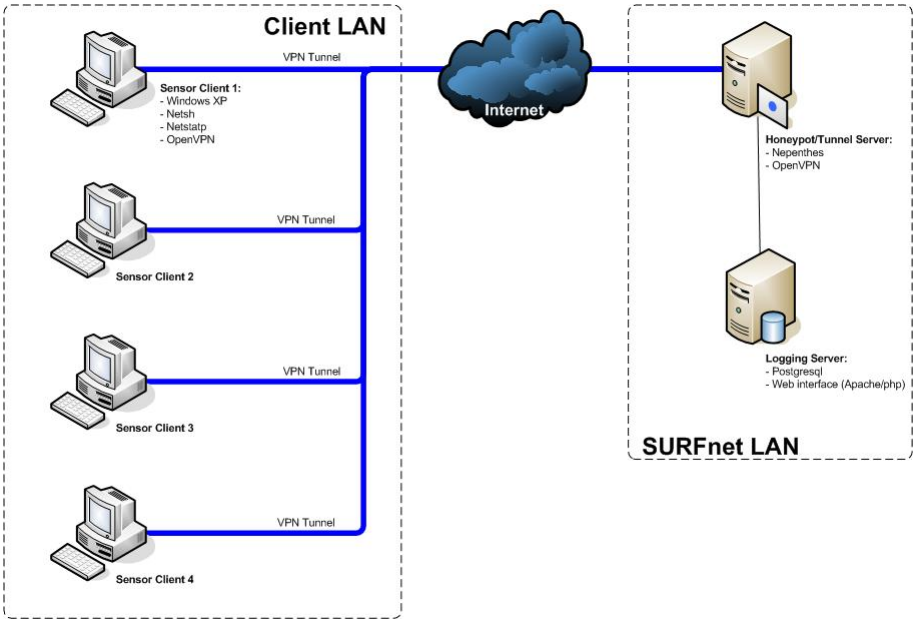

Figure 7.2: The direct sensor solution

7.3 Direct versus Indirect

The solutions discussed in the previous chapters were implemented in a test environment. During this implementation several problems were exposed. These challenges will be discussed before the solutions can be compared. The problems that occured during the implementation of the second solution, the direct one, could all be solved during the implementation. What these challenges were, and how they could be solved, can be found in the next section. The challenges with the indirect solution can be found in the following section that describes why IP-tunneling is needed.

7.3.1 Challenges Direct

With the use of port forwarding all traffic on the unused ports will be forwarded to the honeypot. The problem here is that the honeypot receives the source IP-address of the Sensor. This way, it is not possible to answer the attacker. To solve this, NAT (Network Address Translation) should be enabled on the sensor and source based routing on the honeypot. With the help of Netsh, which is available in Windows XP, it is possible to enable NAT together with port forwarding. NAT does not change the IP-address of the attacker while forwarding the package to the honeypot. So the honeypot thinks the attacker directly attacking the honeypot. When the honeypot receives the package it answers through the default gateway to the attacker. Now an attacker knows he/she is being forwarded. To solve this problem source based routing[7] could be used as it is being used now. With source based routing the honeypot answers through the TAP device on which he receives the package, so all traffic is routed through the layer-2 tunnel to the Windows sensor.

7.3.2 Challenges Indirect

The big advantage of this solution is that no changes had to be made to the current sensor, the Sensor Server. So no changes have to be made before the Windows Sensor Clients go live. However, this solution also has a disadvantage. This disadvantage will be described by the following example (figure 7.3).

There is an organisation X that has a workstation that booted the current sensor. The sensor has an IP-address, namely 192.87.1.24. When it booted it created a layer-2 tunnel to the SURFnet honeypot Server. This tunnel is a connection between two virtual devices on the sensor and the server, so the honeypot is virtually present in the LAN of organisation X. The honeypot also receives an IP-address from the DHCP server of organisation X, namely 192.87.1.25.

Now a Sensor Client is created on a normal workstation. The workstation has ip-adress 192.87.1.13. Netstat sees that port 80 is unused so it will start a port forward to 145.92.1.25. An attacker, with IP-address 123.0.0.123, attacks 145.92.1.13 on port 80 (step 1). Sensor Client 1 forwards all incoming traffic on port 80 to 145.92.1.25 (step 2). The honeypot server responds to the attack and returns the data to Sensor Client 1 on port 80 (step 3). Sensor Client 1 forwards the traffic to the attacker (step 4).

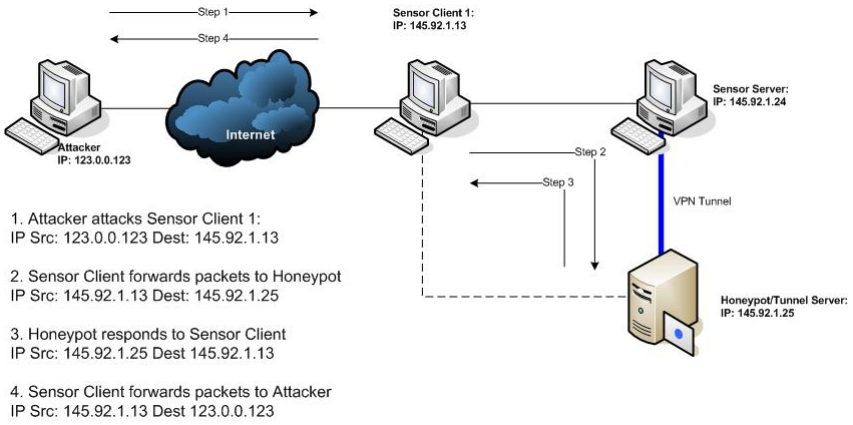

Figure 7.3: The indirect sensor solution

The problem occurs when the sensor client forwards the traffic to the honeypot, as can be seen in figure 7.3. It changes the source address to itself, so that the honeypot thinks the attacker has IP-address 145.92.1.13. However, the honeypot should know the IP-address of the attacker. It is not possible to use NAT[8] pre-routing to solve this problem, as has been used in the direct solution, because when the honeypot returns the data it will send it to its default gateway. The attacker will then receive data with the IP-source address of the honeypot instead the source-address of the Sensor Client, so all traffic should be routed through the Sensor Client.

The following are possible solution for this problem. The first is IP-tunneling[2]. It is possible to encapsulate an IP-packet in an IP-packet. The Sensor Client can then create an IP-tunnel between the Sensor Server and itself, so that the Sensor Server knows both IP-addresses, the attacker's IP-address in the encapsulated IP-packet and the sensor's IP-address in the encapsulation (as shown in figure 7.4). Several projects, like Linux-Virtual-Server[3], use IP-tunneling. These projects all concentrate on one operating system. No interoperable solutions were found, so it is not sure that this solution could work.

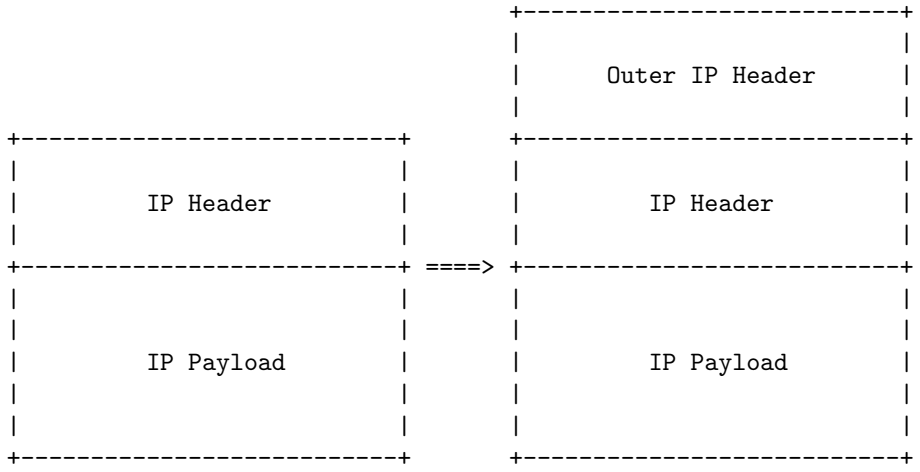

Figure 7.4: The encapsulation of IP-packets[25]

The second solution is IPSec[4]. IPSec can also encapsulate IP-packets in the same way like IP-tunneling does. The third solution is IPv6[5]. IPv6 also creates a tunnel between certain connections. There are more possible solutions, but because of the time limit of this project it was not possible to research these in great depth.

7.3.3 Comparison

As can be seen in the previous sections there were problems with the implementation of both solution. These problems could be solved during the implementation of the direct solution. For the problems of the indirect solution some more research needs to be done. Generally both solutions have their advantages and disadvantages. In this section these will be discussed.

Advantages Direct Solution

The direct solution has some advantages, which are the following:

- The first benefit of the direct solution is that it sends all the data over a secure VPN tunnel
- The second advantage is that the old-style sensors can still be used next to the new windows sensors

[2]RFC 4078 (IP Tunneling): `www.ietf.org/rfc/rfc4087.txt`
[3]for more information see: `http://www.linuxvirtualserver.org`
[4]RFC 2401 (IPSec): `www.ietf.org/rfc/rfc2401.txt`
[5]RFC 2460 (IPv6): `www.ietf.org/rfc/rfc2460.txt`

- Because the concept is equal to the old-style sensors the maintenance will be less
- The last benefit is that this solution is already working in our current setup at SURFnet.

Disadvantages Direct Solution

The disadvantages of this solution are the following:

- Every new sensor needs a new TAP device on the honeypot, which can makes things crowded when every organisation has 25 sensors.
- The source based routing tables contains many rules, because every new sensor needs a VPN tunnel and its own routing table entry. Currently the solutions has a limit of 260 VPN tunnels. If this solution is implemented a solution has to be found for this limitation.

Advantages Indirect Solution

The indirect solution also has its own advantages and disadvantages. The advantages are:

- It needs a Sensor Server, but this sensor is already present in the current setup
- The direct solution requires multiple VPN tunnels to the honeypot, this solution only needs one, which is allready present in the current setup
- Because of the single tunnel this solution offers a better structure

Disadvantages Indirect Solution

The following are disadvantages of the indirect solution:

- The IP-tunneling/IPSec/IPv6 implementation can make things complicated and it maybe implies that adaptions needs to be made to the Sensor Server (the old-style sensor)
- This solution could not be tested completely

Both solutions have their own advantages and disadvantages. So it will be not so easy to choose the best implementation. The direct solution can be implemented right away, whether the indirect still needs some research. In the next chapter our conclusion can be found on both solution as well as the future work that needs to be done.

Chapter 8

Conclusion

In this chapter the conclusion as well as the future research and development for the Windows IDS Sensors will be described. The new sensor section describes the proposal of the solutions that were designed during this report. The future work is a proposal towards future developments for the SURFnet IDS Sensors project.

8.1 The New Sensor

After the research two solutions were designed that both have their disadvantages and advantages, which were discussed in section 7.3.3. Both are a possible solution for SURFnet IDS.

The dedicated Sensor Server can be the old-style modified Knoppix sensor at the indirect solution. Also there are less VPN connections required, because only the sensor server has a VPN connection. This solution offers a better overview of all the sensors.

The honeypot server does currently not have the capability to handle much more then 260 VPN tunnels, at the direct solution. The amount of TAP devices is inconvenient. However, this solution has already been tested and therefor only needs to be tuned. If an owner of the sensor wants to use the existing sensor, it is possible to run both, the Windows sensor and the modified Knoppix sensor. All data is send from the sensor to the honeypot over a secure channel, because of the VPN tunnel.

At the end of this project the research question can be answered:

> *How to give a desktop computer the same functionality as the old-style SURFnet IDS sensors without affecting the current functionality of the desktop computer?*

With the help of tool like Netsh, Netstatp and Trivial Port Forward it is possible to create a Windows IDS Sensor that forwards unused ports, and still has the same functionality as before. The direct solution is already implemented in a testing environment. The other solution needs some more research and testing. The direct solution is most promising, because:

- It has been tested successful in a testing environment at SURFnet IDS
- It sends its data over a secure VPN tunnel
- Only small modifications have to be made to the honeypot
- No modifications have to be made to the current sensor
- Both, the current and the new sensors, can be used in conjunction

8.2 Future Work

Because of the time limit it was not possible to complete the project with a final – completely tested – solution. In order to help completing the project development stage the following future work is recommended to complete this project.

8.2.1 Sensors

At the moment the sensor is checking which ports are open and forwarding all other ports to the honeypot server. A better and possible even faster solution would be to make a list of all used ports and refresh this every x seconds. When an attacker attacks the sensor, the sensor checks whether the port is in use. If the reply is negative the sensor forwards the port to the honeypot. With this method you do not have to re-forward every port after a refresh of the used ports, this saves time and resources.

In order to keep the source IP-address intact with the indirect solution, the IP-packages must be encapsulated. This is already discussed on page 17. Because of the time limit of this project we recommend that another project is started that researches what the best option for the data transfer should be.

The project could be extended to work in corporation with the Windows XP Firewall. It can then check what ports are blocked/un-blocked to get a more information about the ports that are in use, for instance the Netbios ports that are in use by default.

Extending the SURFnet IDS with Windows-based sensors increases the functionality of the total system. It gathers more information to analyze and it enables to install sensors in all sectors of a network infrastructure.

Bibliography

[1] IDS
http://www.honeypots.net - Articles about IDS and honeypots
http://www.networkintrusion.co.uk/ids.htm - Different kinds of IDS's
http://en.wikipedia.org/wiki/Intrusion-detection_system - What is an IDS
http://www.windowsecurity.com/articles/Hids_vs_Nids_Part1.html - Host IDS vs. Network IDS

[2] Snort
http://www.snort.org - Homepage of the Snort Project

[3] Tripwire
www.utexas.edu/its/sds/faq/tripwire.html - Homepage of the Tripwire Project

[4] Prelude
http://www.prelude-ids.org - Homepage of the Prelude IDS Project

[5] SURFnet IDS
http://ids.surfnet.nl - Homepage SURFnet IDS
http://staff.science.uva.nl/~delaat/snb-2005-2006/p29/report.pdf - Pilot project of SURFnet IDS

[6] Nepenthes
http://nepenthes.mwcollect.org - Homepagina of the Nepenthes Project
http://sourceforge.net/projects/nepenthes - Download and information about Nepenthes at Sourceforge

[7] Source based routing
http://www.wlug.org.nz/SourceBasedRouting - Source based routing explained

[8] NAT
http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.txt - NAT Howto

[9] Netstat
http://en.wikipedia.org/wiki/Netstat - Information about Netstat
http://www.microsoft.com/technet/prodtechnol/windowsserver2003/nl/library/ServerHelp/7b3ae3c0-4b95-4cb7-a290-57b22824194b.mspx?mfr=true - Microsoft information about Netstat

[10] TCPview
www.microsoft.com/technet/sysinternals/utilities/TcpView.mspx - Microsoft information about TCPview
http://www.jsifaq.com/SF/Tips/Tip.aspx?id=9321 - Information about TCPview and Netstatp

[11] Netstatp
http://www.jsifaq.com/SF/Tips/Tip.aspx?id=9321 - Information about Netstatp and TCPview

[12] Fport
http://www.foundstone.com - Homepage of Fport
http://www.ibiblio.org/security/articles/fport.html - Article about Fport

[13] Winpcap
http://www.winpcap.org - Homepage of Winpcap

[14] WinDUMP
http://www.winpcap.org/windump - Homepage of WinDUMP

[15] Nmap
http://insecure.org/nmap - Homepage of Nmap

[16] Filter driver
http://www.microsoft.com/whdc/driver/filterdrv/default.mspx - How to edit the I/O of Windows drivers
http://www.pcausa.com/tdisamp/default.htm - TDI filter driver, used by firewalls
http://www.ndis.com/papers/winpktfilter.htm - NDIS filter driver, with pictures
http://www.codeproject.com/internet/drvfltip.asp - Driver hook, information and code examples
http://www.webopedia.com/TERM/B/Berkeley_Packet_Filter.html - Berkeley Packet Filter, filter used by Libpcap

[17] NetworkActiv
http://www.networkactiv.com - Homepage of NetworkActiv

[18] Porttunnel
http://www.steelbytes.com - Homepage of Steelbytes, manufacturer of Porttunnel

[19] TCP Port Rerouter
http://www.zdnet.fr/telecharger/windows/fiche/0,39021313,22036862s,00.htm - Article about TCP Port Rerouter

[20] Wintunnel
http://www.codeproject.com/cs/internet/WinTunnel.asp - Homepage of Wintunnel

[21] Portforward
http://rtfm.insomnia.org/~qg/portforward.html - Homepage of Trivial Port Forwarding

[22] Netsh
http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/netsh.mspx?mfr=true - Microsoft's webpage about Netsh
http://www.computerhope.com/netsh.htm - Explanation of the Netsh command

[23] OpenVPN
http://www.openvpn.org - Homepage of the OpenVPN Project

[24] Privacy
http://www.securityfocus.com/infocus/1703 - Article about security issues and privacy aspects of IDS systems in the US
http://www.securityfocus.com/news/4004 - Use a Honeypot, go to prison

[25] Figures
The sources of the various figures of this document:
Figure 3.1 : http://ids.surfnet.nl/global.php
Figure 7.4 : http://www.ietf.org/rfc/rfc2003.txt

Chapter 9

Appendixes

9.1 Netstatp

We have modified Netstatp. The following source code is the part where the ports are opened. In this example we try to open ports 22 to 80. The UsedTCPports array contains the used ports. If a port is not in use it can be forwarded in two manners. The first one is uses the Trivial Port Forwarding, and the second uses Netsh (which is used in the direct solution with layer-3 VPN tunnels).

```
//
// Now we are going to open the ports
//

for( want_to_open = 22; want_to_open < 81; want_to_open++) {

opencheck=0;

for( i = 0; i < tcpExTable->dwNumEntries; i++ ) {

if (UsedTCPports[i] == want_to_open){
opencheck=1;
}
}
if (opencheck == 0) {

//
//To open the ports 2 options can be used, the first is using a c++ script called portforward
//The second one is using the windows netsh solution
//

// Portforward.exe
sprintf(command, "start /B portforward.exe %d 192.87.118.61 %d", want_to_open, want_to_open);
system(command);

//Netsh
//sprintf(command, "netsh routing ip nat add portmapping lan tcp 0.0.0.0 %d 10.0.0.1 %d", want_to_op
//system(command);

printf("Opening port %d\n", want_to_open);
}
}
```

9.2 Port Forward

The following source code is a part of the Trivial Port Forwarding tool. Here the sockets are switched, so the incoming traffic is send to the outgoing traffic on the other socket. It will loop as long as data is recieved and send (The 100% CPU usage bug used to be in this part of the utility, because the utility was checking on the wrong parameters).

```
DWORD WINAPI reader(LPVOID lpParameter)
{
    SOCKET *socks = (SOCKET*)lpParameter;

    char buf[65536];
    int n;
    int m;

    while ((n = recv(socks[0], buf, sizeof(buf), 0)) != 0 && (m = send(socks[1], buf, n, 0)) != 0) {
    }

    closesocket(socks[0]);
    closesocket(socks[1]);

    return 0;
}

DWORD WINAPI writer(LPVOID lpParameter)
{
    SOCKET *socks = (SOCKET*)lpParameter;

    char buf[65536];
    int n;
    int m;
    while ((n = recv(socks[1], buf, sizeof(buf), 0)) != -1 && (m = send(socks[0], buf, n, 0)) != -1)
    }
    closesocket(socks[0]);
    closesocket(socks[1]);

    return 0;
}
```

9.3 Netsh

In order to use Netsh you fist have to run the script below and after that reboot the computer.

```
#Enabling Routing and Remote Access
Sc config remoteaccess start= auto
Net start remoteaccess
```

After the reboot run the following script to enable NAT and enabling port forwarding. The *portmapping* part can be repeated for each port.

```
#Enabling NAT and forwarding ports
Netsh routing ip nat install
Netsh routing ip net add interface INTERNET NIC full
Netsh routing ip nat add portmapping INTERNET NIC $proto 0.0.0.0 $port $tunnel_server_ip $port
```

If you don't reboot the computer after the first script NAT and port forwarding will not work.