Intrusion Detection System honeypots

Master program System and Network Administration

University of Amsterdam
In cooperation with SURFnet

Mark Meijerink, Jonel Spellen

February 13, 2006

Chapter 1

Summary

With the immense popularity of the Internet and the growing number of households with a broadband Internet connection more and more computers are being attacked by hackers, script-kiddies and malware. To be able to keep your network clean of these illegitimate activities you have to monitor your network so you can take actions against thesethe activities get detected. Honeypots are systems which can be used for this goal. activities when they get detected.

SURFnet is a Internet Service Provider for research centers, school and universitities and connects these connected parties to the Internet. SURFnet is rolling out a Distributed Intrusion Detection System (D-IDS) to detect illegitimate network activity in the LANs of the connected parties. Sensors in those LANs are connected to the honeypot at SURFnet. The honeypot gathers information about the detected illegitimate network activity. Via a webinterface the connected partie can see the information about the detected malware in their network.

In our research project we researched the honeypot SURFnet is using in their IDS and we looked at other honeypots which could be interesting for SURFnet to apply in their IDS.

The conclusion of our research project is that Nepenthes is the honeypot that meets most of the requirements of SURFnet. Nepenthes emulates the most known vulnerabilities. To detect zero-day attacks we advised to add the Argos high-interaction honeypot to their IDS. When Argos is added to their IDS they can detect most of the illegitimate network activity.

Chapter 2

Preface

Students of the master study System and Network Administration have to do two research projects on a certain topic. We have chosen a research project about honeypots. Almost avery household as an Internet connection nowadays and more and more households are connected by a broadband Internet connection. Computers with a broadband Internet connection are willing targets for hackers and script-kiddies. These computers can be compromised by the attackers and then be used for spamming, keylogging and botnetting. With an IDS you can detect compromise attacks or malware spreading so administrators can take action against these activities. A system which can be used for detecting unauthorized activities on the network is a honeypot. In our research project we have researched the honeypot SURFnet is using in their IDS and looked at other honeypots that could be part of their IDS. In this report you can read our findings.

2.1 Acknowledgments

We want to thank Rogier Spoor, Kees Trippelvitz and Jan van Lith at SURFnet for their advice and feedback during the research project. We also want to thank Herbert Bos and Georgios Portokalidis for their contribution to the research project about the Argos Honeypot.

Contents

Chapter 3

Research goals

When we started the research project we made a projectdescription. In this chapter we will describe the research goals we have defined for this research project. The full research description, written in Dutch, can be found in the appendix as Appendix A: Projectomschrijving. The research goals we have defined are listed below.

- Analyse Nepenthes in catching malware
- Find interesting tools and honeypots for the IDS
- Find protection methods against TCP fingerprinting

In the analysis of Nepenthes we will look at the way Nepenthes is designed to catch malware. We will try to find out which malware is catched and which malware isn't. Our second research goal is to find honeypots and tools which SURFnet can use to expand their IDS. If a honeypot is interesting to look at we may do some tests. We will compare these honeypots with Nepenthes and give an advice. The final goal is to find protection methods against TCP fingerprinting tools. It depends on the time we have left after two research goals mentioned earlier in we can start with a research about TCP fingerprinting. With tools like nmap and xprobe you can make a fingerprint of a system and get to know which operating system it runs. Our goal is to find methods to masquerade the real operating system.

Chapter 4

Related work

Last year Kees Trippelvitz and Harm-Jan Blok from the master study System and Network Administration have done a research project for SURFnet last summer. They made a proposition [1] for the design and architecture of an IDS what SURFnet can use to detect illegitimate network activity on the connected parties LANs. They designed an scalable and zero maintenance architecture. Kees Trippelvitz and Harm-Jan Blok based their research project about other projects which were done before. These project were about using standard applications as sensors, a centralized log analysis for IDSs [3] and the use of high interactive windows honeypots [4].

Chapter 5

Introduction to IDSs and honeypots

This report is about IDSs and honeypots. This chapter gives an overview of IDSs, honeypots and the SURFnets IDS Service.

5.1 IDS

In general an IDS detects unwanted manipulations to systems. The manipulations may take the form of attacks by skilled malicious hackers, or script kiddies using automated tools. An IDS is required to detect all types of malicious network traffic and computer usage that can't be detected by tools like a firewall. This includes network attacks against vulnerable services, data driven attacks on applications, host based attacks such as privilege escalation, unauthorized logins and access to sensitive files, and malware (viruses, trojan horses, and worms).

Within IDS we separate three kind of IDS. Network IDSs, host-based IDSs and hybrid IDSs. A Network IDS identifies intrusions by examining network traffic. Network IDSs gain access to network traffic by connecting to a hub, network switch configured for port mirroring, or network tap. A Host-based IDS consists of an agent on a host which identifies intrusions by analyzing system calls, application logs, file-system modifications and other host activities and state. A Hybrid IDS combines both approaches. Host agent data is combined with network information to form a comprehensive view of the network. This paragraph is based on the the text of Wikipedia.org [15] about IDS.

5.2 Honeypots

A honeypot is a trap set to detect, deflect or in some manner counteract attempts at unauthorized use of information systems. A honeypot appears to be part of a network but which is actually isolated and protected. A honeypot seems to contain information or a resource that would be of value to attackers. A honeypot is valuable as a surveillance and early-warning tool. Honeypots should have no production value and hence should not see any legitimate traffic or activity. Whatever they capture can then be surmised as malicious or unauthorized. Honeypots can generally be divided into different categories, low-interaction and high-interaction honeypots. Low-interaction honeypots emulate services. Honeyd, Mwcollect and Nepenthes are low-interaction honeypots which can be used to collect autonomously spreading malware. Automated attacks are not only logged, the daemons extract information how to obtain the malware binaries from the exploit payload using known patterns and then actively download a sample. High interaction honeypots like Argos offer a full operating system to the attacker and when the attackers tries to do something malicious the honeypot will shut down and makes dumps of memory and disk to get information about what the attacker was trying to do. This paragraph is based on the the text of Wikipedia.org [16] about Honeypots.

5.3 SURFnet's IDS Service

SURFnet is going to roll-out a Distributed IDS with multiple sensors and one honeypot. This D-IDS is in testing fase now. In their current approach a workstation in the LAN of the connected party is used as a sensor. The workstation is turned into a sensor by booting it from an USB stick containing the SURFnet D-IDS sensor software. This USB stick contains a remastered Knoppix distribution and uses openVPN to start a layer-2 tunnel to the D-IDS server. The layer-2 tunnel is put in bridging mode with the network interface of the sensor. Next, a DHCP request is made from the D-IDS server through the tunnel into the client LAN. This request allows the D-IDS server to obtain an IP-address on the client LAN and then bind it on a virtual interface containing a honeypot. Virtually, the D-IDS server will be present on the client LAN and attackers will think they are attacking a host on the client LAN. The honeypot that is being used on the D-IDS server is Nepenthes, which is able to emulate vulnerabilities. If an attacker triggers the honeypot it is considered a malicious attack and the honeypot attempts to retrieve the malware that an attacker tries to put on the host. All attacks are logged into a PostgreSQL database and users are able to view detailed information about the attacks through a web interface. For more information and the latest news you can go the SURFnet IDS website [2]. This paragraph is based on the information found on the SURFnet's IDS website and the research report of Kees Trippelvitz ans Harm-Jan Blok [1].

Chapter 6

Analysis of the Nepenthes honeypot

Nepenthes is a low-interaction honeypot that emulates known vulnerabilities worms use to spread and Nepenthes catches these worms.

6.1 Detection method

Nepenthes has a lot of vulnerabilities it can emulate. Every vulnerability module emulates a vulnerability in a service which runs on a network port. Nepenthes also starts services like FTP, POP3, IMAP to collect data. When an attacker connects to a service Nepenthes will open the network port of that service and asks every registered vulnerability module on that port to start a shellcodehandler for accepting the data. When the data comes in, the data is sent to all the created shellcodehandlers. As the packets send by the attacker do not fit what the shellcodehandlers expect then they tell the socketmanager to stop sending packets to the shellcode handler. Every vulnerability module has a certain amount of functionality to interact with the attackers. When a vulnerability is exploited the malware will send shellcode to the network port. When the shellcode is received the shellcodehandler will pass the shellcode to a second shellcode handler. This handler will use perl regular expressions to detect the type of shellcode. Lets say this is a shellcode to create a windows shell on a certain port. The shell will be started and the shellcodehandler will wait for further instructions. Most malware will now try to download a file and execute it. Nepenthes will download the file but will not executed it.

6.2 Vulnerabilities

Nepenthes catches malware what tries to exploit known vulnerabilities. In this part of the research we describe the emulated vulnerabilities, the way Nepenthes

handles unknown vulnerabilities and a short analysis of Nepenthes and vulnerability emulation.

6.2.1 Emulated vulnerabilities

Nepenthes emulates known vulnerabilities. In this paragraph we give the list of vulnerabilities which Nepenthes can emulate. We used the documentation which is available at the website of the Nepenthes project [5]. The following vulnerabilites were emulated in Nepenthes version 0.1.6.

Portnr.	Vulnerability
	vuln-netdde, emulates the netdde vulnerability
42	vuln-winsm, emulates the wins vulnerabiltiy
80	vuln-asn1, emulates the asn1 vulnerability
80	vuln-iis, handles some different bugs in microsoft iis5
135	vuln-dcom, emulates the dcom vulnerabilty
139	vuln-netbiosname, replies netbiosnames on valid requests
443	vuln-iis, handles some different bugs in microsoft iis5
445	vuln-dcom, emulates the dcom vulnerabilty
445	vuln-lsass, emulates the lsass vulnerabiltiy
445	vuln-asn1, emulates the asn1 vulnerability
1023	vuln-sasserftpd, handles bug in sasserftp
1025	vuln-dcom, emulates the dcom vulnerabilty
1434	vuln-mssql, emulates the mssql vulnerability
2103	vuln-msmq, handles the MSMQ bug found in 2005
2105	vuln-msmq, handles the MSMQ bug found in 2005
2107	vuln-msmq, handles the MSMQ bug found in 2005
2745	vuln-bagle, emulates the bagle backdoor
3127	vuln-mydoom, emulates the mydoom backdoor
3140	vuln-optix, emulates the optix vulnerability
5000	vuln-upnp, emulates the upnp vulnerability
5554	vuln-sasserftpd, handles bug in sasserftp
17300	vuln-kuang2, emulates the kuang2 vulnerability
27374	vuln-sub7, emulates the sub7 vulnerability

6.2.2 Known vulnerabilities

We used the list of known vulnerabilities we found on the website of the Cert Command Center [9]. This list contains vulnerabilities for applications and operating systems. We filtered out the application vulnerabilities so the list of operating systems remains.

Effected Service	Portnr/Protocol	Related Information
tftp	69/udp	CA-2003-20: W32/Blaster worm
http	80/tcp	CA-2002-27: Apache/mod_ssl Worm

		CA-2003-09: Buffer Overflow in Core Microsoft Windows DLL
epmap	135/tcp	
	135/udp	CA-2003-16: Buffer Overflow in Microsoft RPC
		CA-2003-19: Exploitation of Vulnerabilities in Microsoft RPC Interface
		CA-2003-20: W32/Blaster worm
		CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows
netbios-ns	137/udp	CA-2003-08: Increased Activity Targeting Windows Shares
		CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows
netbios-dgm	138/udp	CA-2003-08: Increased Activity Targeting Windows Shares
		CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows
netbios-ssn	139/tcp	
	139/udp	CA-2003-03: Buffer Overflow in Windows Locator Service
		CA-2003-08: Increased Activity Targeting Windows Shares
		CA-2003-16: Buffer Overflow in Microsoft RPC
		CA-2003-19: Exploitation of Vulnerabilities in Microsoft RPC Interface
		CA-2003-20: W32/Blaster worm
		CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows
https	443/tcp	CA-2002-27: Apache/mod_ssl Worm
microsoft-ds	445/tcp	
	445/udp	CA-2003-03: Buffer Overflow in Windows Locator Service
		CA-2003-16: Buffer Overflow in Microsoft RPC
		CA-2003-19: Exploitation of Vulnerabilities in Microsoft RPC Interface
		CA-2003-20: W32/Blaster worm
		CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows
http-rpc	593/tcp	CA-2003-20: W32/Blaster worm
		CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows
unassigned*	1052/tcp	CA-2002-27: Apache/mod_ssl Worm
unassigned*	1978/udp	CA-2002-27: Apache/mod_ssl Worm
globe	2002/udp	CA-2002-27: Apache/mod_ssl Worm
ctx-bridge	3127/tcp	Current Activity 01/26/04: W32/Mydoom.A or W32/Novarg
		Current Activity 02/10/04: W32/Mydoom.C or W32.HLLW.Doomjuice
unassigned*	4156/udp	CA-2002-27: Apache/mod_ssl Worm
unassigned*	4444/tcp	CA-2003-20: W32/Blaster worm

6.2.3 Unknown vulnerabilities

As we mentioned in the paragraph about the detection method the vulnerbility modules start a shellcodehandler when a connection is made to the network port they are registered to. If none of the shellcodehandlers can handle the given shellcode Nepenthes will make a hexdump of the shellcode to disk and will create a message in the logfile. These hexdumps can be used by developers to create a new shellcodehandler or by anti-virus companies. Connections made on network ports on which no vulnerability modules or other modules are registered are not noticed by Nepenthes. No logs will be created.

6.2.4 Analysis

Nepenthes detects malware using use well known vulnerabilities. Nepenthes gives very few false positive and false negative events. This is because Nepenthes is looking at the known vulnerabilities and uses shellcode comparison.

Most of the known vulnerabilities in Windows operating systems are emulated by Nepenthes. Vulnerabilities in other software the the operating system are not fully covered. Nepenthes simulates a lot of vulnerabilities in applications besides the operating system. Vulnerebilities in applications like Sendmail are not simulated by Nepenthes but Nepenthes does open the ports 21 for FTP, 25 for SMTP and 110 for POP3.

Malware using vulnerabilities not covered by the Nepenthes modules can not be identified. Nepenthes will create a log in the logfile and will make an hexdump of the communication to disk. Trough this hexdump developers can look at the code that the malware is trying to execute. Developers can also make a new module to emulate the vulnerability and interact with the malware so the malware can be identified in the future.

6.3 Features

- Detects attacks on known vulnerabilities
- Low rate of false positives and false negatives
- Downloads malware
- Modular

6.4 Supporting community

The Nepenthes developers can be contacted in different ways for support. On their website they have a mailing-list [6], a forum [7] and a contact [8] section. Nepenthes also has an IRC channel. The IRC channel is #Nepenthes. The main developer can always be contacted and to answer your questions.

6.5 Development

At January 15th 2006 version 0.1.6 was released. The Nepenthes project is under continious development. In the release history of Nepenthes we see a new version every one or two months. The developers keep on writing new modules for Nepenthes.

6.6 Extendability

Nepenthes is build modular and can be easily extended with new modules. When Nepenthes is started it will go true the modules directory and starts loading the modules. Nepenthes gives the developer 8 example modules with examples for using modules for dns resolving, file downloading and geological IP location resolving.

Chapter 7

Alternative honeypots and tools

In the current IDS Nepenthes is the honeypot. Our goal was to look at tools and other honeypots with could be used to improve the IDS. Because the lack of time we were only able to analyse other honeypots. We looked at many honeypots but most of them are no longer under development. In our research we looked at low-interaction honeypots, high-interaction honeypots and a honeywall. The honeypots we analyzed were Honeyd, Mwcollect, Honeynet and Argos. Honeyd and Mwcollect are low-interactive honeypots just like Nepenthes, Argos is a high-interactive honeypot and Honeynet is a honeywall.

7.1 Honeyd

Honeyd [11] is developed by Niels Pavos of the University of Michigan. The first versions were written for the unix platform but meanwhile Honeyd has also been ported to the Windows platform.

7.1.1 Detection method

Honeyd uses a tool called arpd to route the illegitimate network traffic to unused IP addresses to the Honeyd honeypot because every connection attempt to an unused IP address is a possible attack. Honeyd will take the identity of the unused IP address and will interact with the attacker.

Honeyd uses virtual hosts to communicate with the attacker. A virtual host is simulated at stack level so tools like nmap and xprobe will not be able to get a fingerprint of the honeypot server but will get the operating system of the virtual system. The virtual hosts are defined in the configuration file of Honeyd. When you create a virtual host you can open tcp and upd ports, bind scripts to ports, set the personality of the virtual host and bind an IP address to a virtual host.

We can open all ports if we want and we can write scripts who handle the connections. For more information about how to create virtual hosts and networks we advice you to read the documents "Simulating Networks with Honeyd" [12] and the webpage "Open Source Honeypots: Learning with Honeyd" [13].

7.1.2 Vulnerabilities

Although we can create scripts to handle connections there have only been two scripts written to catch malware. One script to catch the kuang2 worm and one script to catch the mydoom worm. There have been developers who created a mail server emulator and telnet emulator which we can use to detect connections on these ports and look at the instructions given by the attacker. With Honeyd you can see connections made to opened ports but to see which vulnerability the malware is trying to exploit you should capture all the instructions and execute these instructions on a machine running a real operating system.

7.1.3 Features

- simulates virtual hosts
- service configuration via configuration file
- simulates operating systems on TCP/IP stack level
- simulates network topologies
- subsystem virtualization

7.1.4 Supporting community

The community behind Honeyd seems to be very small. When you take a look at the forum we see many questions but not so many answers. Only Niels Pavos gives answers to the questions so it seems that he is the only one who is really into Honeyd.

7.1.5 Development

Niels Pavos released a new version of Honeyd at December 31st 2005. However this is just a beta test release. Although this new version was released there are no new scripts available to detect the vulnerabilities malware are trying to exploit. The last script was added on June 11th 2004.

7.1.6 Extendability

Honeyd uses scripts to handle connections made on certain ports. Developers can develop their own scripts and bind them to a port to handle connections made on that port.Honeyd uses plugins to extend it's functionality. These can be written by developers also.

7.2 Mwcollect

Mwcollect [26] is a low-interaction honeypot. Honeynet is partly funded by the Honeynet project. Mwcollect is designed like Nepenthes. The current devloper of Nepenthes decided to start his own project after he had a different point of view how Mwcollect should be developed. He took the source and started the Nepenthes project.

7.2.1 Detection method

The honeypots on a operating system. The Mwcollect deamon openes ports which are commonly attacked by malware. By simulating some well known vulnerabilities on these ports, malware will exploit these ports and send its shellcodes to the Mwcollect deamon.The deamon will parse the exploited packets, will search in the shellcodes, interprets the shellcode and then take further action to download de malware. Once the malware is catched it will be saved to disk.

7.2.2 Vulnerabilities

The vulnerabilities which Mwcollect can detect are found in the source code. There are only a few vulnerabilities that Mwcollect can emulate. Mwcollect emulates the vulnerabilities listed below.

- MS03-026 Buffer Overrun In RPC Interface Could Allow Code Execution, port 135.
- MS04-011 Remote Code Execution in LSASS Service, port 445.
- MS05-039 Vulnerability in Plug and Play Could Allow Remote Code Execution and Elevation of Privilege, port 445.
- MS05-051 Remote Code Execution in MS Microsoft Distributed Transaction (MSDTC), port 1025.

7.2.3 Features

- Detects attacks on known vulnerabilities.
- Modular
- Downloads malware
- Low false positives and false negatives

7.2.4 Supporting community

Mwcollect is partly funded by The Honeynet Project [17]. Mwcollect has a big support community. They don't have a forum where someone can post messages. But for viewing modules or shellcodes you can use their svn(subversion) repository or Trac's SVN front-end (web based svn repository). For reporting bugs they have a ticket system where someone can post found bugs. Mwcollect also has a IRC channel to communicate with the developers. The IRC channel is #Mwcollect.

7.2.5 Development

In december 2005 they released version 3.02. This means that they are still developing Mwcollect application. But the development of modules and shellcode is going very slow. There are not much vulnerability emulation modules made for the program.

7.2.6 Extendability

Just like Nepenthes modules and shellcodes can be written to extend Mwcollect's features.

7.3 Honeynets

Honeynet is developed by The Honeynet Project to capture information at a network. The primary purpose of the honeynet is to gather information on security threads. The tool is a high-interaction honeypot to capture extensive information. It act as a gateway, called Honeywall, by collecting data flow from and to the honeypot(s) on a network. The Honeynet is a collection of tools compiled into on CDROM which can be downloaded from The Honeynet Project [17] site. The latest Honeywall CDROM is called "Roo".

7.3.1 Detection method

The detection method [29] is explained based on the figure below. The principle part of the Honeynet is the gateway which is called the honeywall. The honeywall is the one that separates the honeypots victims from the rest of the world. All the traffic entering or leaving the honeypots must pass through the honeywall. The honeywall can be configured as a layer 2 or layer 3 routing gateway. But a layer 2 bridge is better because in bridge mode it is harder to detect by attackers. By bridging the interfaces the gateway won't have an IP address. The only assigned IP address is a secure network used for administrative purpose only.

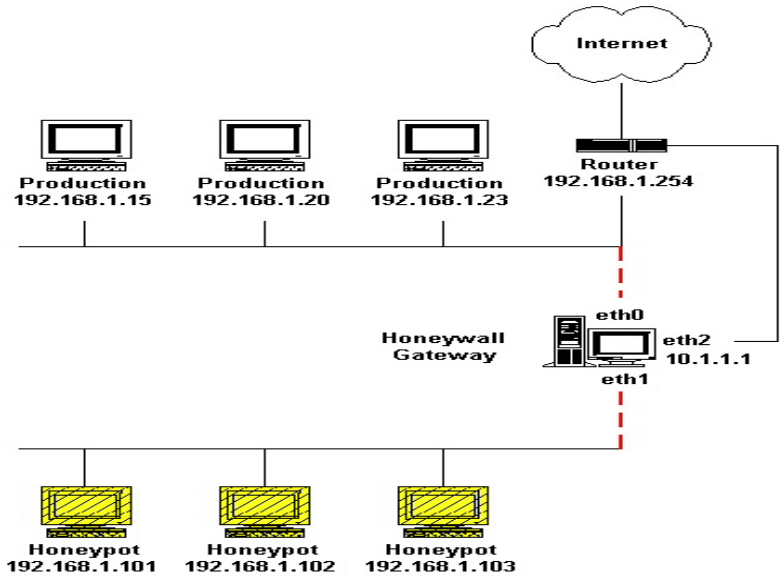

Figure 7.1: Honeywall architecture

For the honeynet to work successfully it has to be deployed properly or it will fail to capture attackers activity. The honeywall consists of three parts: data control, data capture and data collection. Data control: once a honeypot within the honeynet is compromised, honeynet have to contain the activity and ensure the honeypots are not used to harm non honeynet systems. There must be some means of controlling how traffic can flow in and out of the honeynet, without attackers detecting control activities. Data control always takes priority over data capture. Data capture: captures all activity within the honeynet and the information that enters and leaves the honeynet, without attackers knowing they are being watched. The data capturing is the monitoring and logging of all the attackers activities within the honeynet. This captured data is analyzed to learn the tools, tactics and motives of the attackers. Data collection: When the honeynet is part of a distributed environment the captured data is securely forwarded to a centralized data collection point. This allows captured data from honeynet sensors to be centrally collected for analysis and archiving.

7.3.2 Vulnerabilities

The honeynet does not actually emulate vulnerabilities. A honeypot behind the honeywall could be used to emulate vulnerabilities. It logs inbound and outbound data flow passing through the honeywall. This data can then be used for further analysis. By analyzing data new exploits and vulnerabilities can be found.

7.3.3 Features

- Data control
- Data capture
- Data analysis

7.3.4 Supporting community

The Honeynet Project has created a document called “Honeynet Definitions, Requirements, and Standards”. The purpose of the document is to give organizations the flexibility to build a honeynet that fit into their environment and their goals. The document defines in details how the organizations can deploy their honeynet environment effectively and securely and allowing different honeynets to work together. The members of The Honeynet Project can be contacted by email.

7.3.5 Development

The development of honeynet is still in progress. In may 2005 they released the Honeywall CDROM called “Roo”. As they say in their whitepaper “Know

Your Enemy: Honeywall CDROM Roo" [28] they are not fully satisfied with the CDROM. They have more options planned to be added for the next release.

7.3.6 Extendability

The Honeywall is a Fedora 3 Linux distribution and it can be extended by adding more tools. By making use of the package management tools from Fedora more tools can be added.

7.4 Argos

Argos is a high-interaction honeypot for catching zero-day attacks like new worms. Unlike low-interaction honeypots Argos provides real services and a real operating system that the malware can try to compromise. Argos is based on Qemu [22], an fast emulator for multiple architectures like x86 and power-pc64. For more information about Qemu check the Qemu website [22] or the Qemu wikipedia page [23]. Argos extends Qemu by providing the functionality to taint and track memory and to generate footprints. Our focus will be on Argos itself. Argos is focused on attacks which are automated like worms and other malware and do not need any user input. Argos is designed to detect vulnerabilities which are used by malware to compromise computer systems and they are not interested on the payload given after the vulnerability is exploited. This paragraph is based on the paper "Argos: an Emulator for Fingerprinting Zero-Day Attacks" [24] written by Gerogios Portokalidis, Asia Slowinska and Herbert Bos [25] and the website of the Argos project [19].

7.4.1 Detection method

All incoming traffic is logged in a trace database via tcpdump and is send to the guest system that is running on Qemu. Every guest system has his own IP address. The whole system is based on tainting memory that arrives from untrusted sources(i.e. the network). When the network is written to the memory then these memory blocks are tainted. If memory is copied to a other memory block or is copied into a register then this new location is tainted too. The system keeps checking the tainted places.

Argos raises an alarm on several actions. When an attacker is trying to gain control over a system the attacker will try to redirect the control to instructions supplied by the attacker, the attackers shellcode. Argos can detect this action by continious checking the call, ret, jmp and longjmp instructions. These instructions can be used to change the position of the instruction pointer, EIP. Other ways that Argos uses to detect illegitimate use of memory are checks on string format vulnerabilities and checking if the Qemu execve() attributes are tainted. Format string vulnerabilities can be used to overwrite memory locations with illegitimate network data.

When an alarm is raised Argos starts the signature creation proces. Memory dumps and register dumps will be made and are written to a logfile. The authors also made a piece of forensic shellcode which they execute on the tainted code to get extra information about the attacked process, executable name, open files, open sockets, network port used, etc. With the information gathered by the forensic shellcode, the data in the trace database can be filtered on the process and network port. If the attack was a TCP connection then the TCP flow will be reconstructed. The following step is the creation of the signature. The signature is created with the gathered information as input value. Argos creates a flow signature and a packet signature. The flow signature is the sequence of bytes and the packet signature can be used for IDS and IPS systems.

The creation of signatures is not as good as they want it to be so they are still submitting their signatures to Sweetbait. Sweetbait is a system that correlates signatures form different sites and creates a new signature based on this correlation process. These signatures can be used for the creation of IPS rules for example snort_inline and for IDS systems to detect malware. The following figure is an overview of the Argos system as explained in this section.

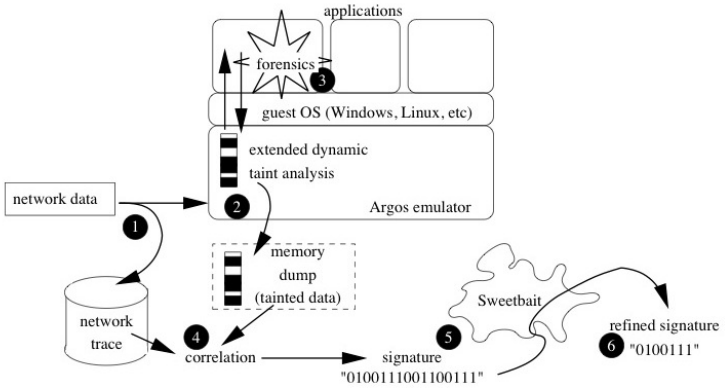

Figure 7.2: Argos overview

7.4.2 Vulnerabilities

The vulnerabilities that are catched by Argos depend on the virtual systems running on top of it. The virtual systems can run all possible several services on Windows or Unix bases operating systems. Argos can detect every attack on a vulnerability in a service. The more services the virtual system runs, the more vulnerabilities Argos can detect.

7.4.3 Features

- Emulation of multiple platforms
- Detects control flow and code execution attacks
- Uses physical memory addresses in stead of virtual memory adresses
- Forensic shellcode

7.4.4 Supporting community

The Argos project is part of the European FP6 Noah project [20] that aims to design a Pan-European Network of Advanced Honeypots. Their are quite a lot of parties involved in the FP6 Noah project from different countries like

Greece, Switzerland, the Netherlands, Germany and France. At the Vrije Universiteit of Amsterdam in the Netherlands Herbert Bos, Georgios Portokalidis, Asia Slowinska and some students are working on the project. Argos will be used as the honeypot in the Network of Advanced Honeypots. This could mean there will be a big supporting community. Herbert Bos and Georgios Portokalidis can be contacted by email or by phone. Their addresses can be found on the Argos website [19]. Argos is also part of the DeWorm project. In this project they plan to investigate a new approach called DeWorm which combines the deep scan and flow-based approaches to stop flash worms even if they are self-modifying. More information can be found on their website [21].

7.4.5 Development

Georgios, Hertbert and Asia are still working at the project at the Vrije Univerisiteit in Amsterdam in the Netherlands. Asia Salwinska in now working on the improvement of the signature creation. The future work is improving the signature generation. Their aim is to make better signatures then Sweetbait is making right now and in the future they want to distribute the signatures themselves. It depends on the growth of the Noah project which steps will be made do develop the Argos honeypot. The forensic shellcode will be released within a few weeks.

7.4.6 Implementation proposal

The current SURFnet architecture is based on three components. The sensors, the honeypot and the logserver. In this section we will focus on the honeypot itself. In the current situation the sensors used to send all of their network traffic from the LANs of the connected parties to the honeypot. The logging server is used to analyse and present the information gathered from the honeypot. We tried to find setups to implement Argos within the current architecture of IDS service. We will explain two setups of which we know they will work.

In the first setup Nepenthes will run as a subsystem of Honeyd and passing al unknown traffic to Argos. In this setup Nepenthes could identity malware and download files, Argos makes signatures of the malware unknown to Nepenthes, and Honeyd will protect the real operating system against TCP fingerprinting as explained in the paragraph about Honeyd. The developers of Nepenthes already tried this setup [27]. They did had a few problems but it worked. More research has te be done to solve these problems.

In the second setup Argos will run at a seperate system. The sensor in the connected party's LAN will setup two openVPN connections. One connection with the Nepenthes system and another connection with the Argos system. Nepenthes can be used for detecting malware trying to exploit well known vulnerabilities and Argos will be used to detect zero-day attacks.

With these two setups SURFnet's IDS should be able to detect malware using known vulnerabilities and malware attacking new, zero-day, vulnerabilities. These setups are two of setups possible. Problems with this setup could be

combining Nepenthes and Argos logfiles. We hope SURFnet can use our setup proposals to expand their IDS service of give them some ideas about how they could combine Nepenthes and Argos in their IDS service.

7.5 Analysis

In the following paragraph we will make a comparison between the honeypots we analysed earlier. We will compare the honeypots on the point listed below.

- Network monitoring
- Vulnerability emulation
- Supporting community
- Development activity

7.5.1 Network monitoring

The honeypots we analysed all have the capability to monitor the network on illegitimate network activity. Honeynet just logs the network activity and need administrators to analyse the logged data.

7.5.2 Vulnerability emulation

Nepenthes, Honeyd and Mwcollect can emulate vulnerabilities. Comparing these honeypots Nepenthes is the one which emulates the most vulnerabilities and is updated most often. A honeypot like Nepenthes or Mwcollect can also be implemented as a subsystem of Honeyd. One of the developers of Nepenthes tried it, but he came faced a couple of problems. More information on this test can be found on the website of Nepenthes [5]. Honeynet is a honeywall and cannot emulate vulnerabilities. The honeypot behind Honeynet can. Argos doens't emulate honeypots eather. The guest system is running a real operating system and provides real services.

7.5.3 Supporting community

Nepenthes community has different ways to contact them. They has a mailing-list, a forum, an contact emailaddress and a IRC channel. In the Nepenthes IRC channel you will find the developer and the members to talk to. Honeyd has a forum on their website where occasionally messages are posted. The most messages posted are answered by the developer Niels Pavos. Mwollect has an IRC channel where you can talk to the developer and members of the community. Argos community is growing bigger. On the Noah [20] website under partner section you can see the partners who are working together with the Argos project. Meaning if Argos grows the supporting community will also be bigger. Honeynet is part of the Honeynet project and has a big supporting community.

Concluding the different supporting communities, the supporting community of Nepenthes is the most active compare to the other. The Argos community will also be growing if the Noah project succeeds.

7.5.4 Development activity

All the alternative honeypots we analysed are still under development. Argos is still under heavy development and when Argos is implemented in the Noah project it will be developt a lot faster. Nepenthes is also under continious development. They bring out new modules on quite often. A new testversion of Honeyd has just been released but is still under development but we can not see any recent scripts and plugins submitted to the website

Chapter 8

Conclusion

The conclusions we can make after our research projects are the following. By comparing Nepenthes with the alternative honeypots on the aspects; network monitoring, vulnerabilities emulation, support community and development activity we conclude that Nepenthes is still the honeypot for SURFnet to use in their IDS. Nepenthes meets most of the requirements of SURFnet's IDS. Argos would be a good addition to the IDS to capture zero-day attacks in LANs of connected parties.

Chapter 9

Future work

Argos looks to be a very promising honeypot and when it will be used in the Noah project a lot of people will be working with the honeypot so it can be developed much quicker than it is now. Future work would be a project for the implementation of Argos. A great challenge of this project would be combining the logs created by Nepenthes and Argos.

In our research project we were only able to look at different honeypots and we dit not had the time do a research for interesting tools as well. In a following research project a research can be done to find interesting tools which can be used to improve the IDS.

Other future work could be the design of an architecure in which multiple honeypots could be added as end-points. The sensors will probably have to be modified to set up multiple openVPN connections to the different honeypots.

Bibliography

[1] Report SURFnet Intrusion Detection System, http://staff.science.uva.nl/ delaat/snb-2004-2005/p30/report.pdf

[2] SURFnet IDS project, http://ids.surfnet.nl

[3] Paper van onderzoek naar gecentraliseerde loganalyse voor IDS, http://staff.science.uva.nl/ delaat/snb-2004-2005/p16/report.pdf

[4] Paper van research projects naar het gebruik van high-interaction Windows honeypots, http://www.os3.nl/ bart/courses/IDS/bestanden/project/IDS_verslag.pdf

[5] Nepenthes project homepage, http://Nepenthes.sourceforge.net

[6] Nepenthes Mailing-list, https://sourceforge.net/mail/?group_id=137598

[7] Nepenthes Forum, http://sourceforge.net/forum/?group_id=137598

[8] Nepenthes Contact, http://Nepenthes.sourceforge.net/contact

[9] CERT Coordination Center, http://www.cert.org/current/services_ports.html

[10] Portlist of malware,http://www.chebucto.ns.ca/ rakerman/trojan-port-table.html

[11] Honeyd Virtual Honeypot hompage, http://www.Honeyd.org

[12] Simulating Networks with Honeyd written by R Chandran and S. Pakala, http://www.paladion.net/papers/simulating_networks_with_Honeyd.pdf

[13] Open Source Honeypots: Learning with Honeyd by Lance Spitzner, http://www.securityfocus.com/infocus/1659

[14] Wikipedia.org, http://www.wikipedia.org

[15] Wikipedia IDS, http://en.wikipedia.org/wiki/Intrusion-detection_system

[16] Wikipedia Honeypots, http://en.wikipedia.org/wiki/Honeypot_%28electronics%29

[17] The Honeynet Porject, http://www.honeynet.org

[18] Qdetect, http://www.quarantainenet.nl

[19] Argos, http://www.few.vu.nl/ porto/Argos

[20] Website of the Noah project, http://www.fp6-noah.org/

[21] Website of the DeWorm project,
http://www.cs.vu.nl/ herbertb/projects/deworm/

[22] Qemu projectsite, http://fabrice.bellard.free.fr/qemu/about.html

[23] Qemu wikipedia.org, http://en.wikipedia.org/wiki/QEMU

[24] Paper Argos: an Emulator for Fingerprinting Zero-Day Attacks written by Georgios Portokalidis, Asia Slowinska and Herbert Bos

[25] Hertbert Bos researcher and teacher at the Vrije Universiteit of Amsterdam,
http://www.cs.vu.nl/ herbertb/

[26] Mwcollect, http://www.Mwcollect.org

[27] Howto run Nepenthes as Honeyd subsystem,
http://nepenthes.sourceforge.net/howto:run_nepenthes_as_honeyd_subsystem

[28] Know Your Enemy: Honeywall CDROM,
http://www.honeynet.org/papers/cdrom/index.html

[29] Know Your Enemy: GenII Honeynets,
http://www.honeynet.org/papers/gen2/index.html

Appendix A: Projectomschrijving

Het begin van het onderzoek wordt de honeypot Nepenthes bekeken. Het onderzoek zal zich richten op de werking van Nepenthes in het vangen van malware. Aan de hand van de binnengekomen meldingen van Nepenthes zal bekeken worden of er nog malware tussendoor glipt zonder dat Nepenthes hier iets mee doet.

Als dit onderdeel is afgerond wordt gekeken naar het eventuele gebruik van andere honeypots (bv. mwcollect, qdetect, honeyd) en tools voor het IDS. Er worden in ieder geval een aantal andere honeypots onderzocht en eventueel getest. Van sommige honeypots is namelijk bekend dat de ontwikkeling stil staat. Binnen SURFNet is een aantal maanden terug gekozen voor Nepenthes, maar daarna is niet meer gekeken naar andere honeypots. Als er honeypots en tools zijn welke een meerwaarde bieden voor het huidige IDS dan wordt bekeken hoe deze kunnen worden opgenomen binnen het IDS.

Als de tijd het toe laat wordt ook nog gekeken naar TCP fingerprinting en naar een full interactive honeypot. Nepenthes is high interactive honeypot en een andere type honeypot is de full interactive honeypot. De honeypot laat de malware zijn gang gaan en als de malware klaar is wordt het systeem afgesloten om de schade te analyseren De sensoren binnen het IDS zijn gebaseerd op Knoppix. Het zou verdacht zijn als malware een TCP fingerprint doet en terugkrijgt dat het een Knoppix systeem is terwijl de malware net een aanval heeft gedaan op een lek in Microsoft Windows.

Appendix B: CERT vulnerability list

Service/Portnr	Used Protocol	Related Information
ICMP	0/icmp	Current Activity 08/18/2003: W32/Welchia Worm
ssh	22/tcp	CA-2002-36: Multiple Vulnerabilities in SSH Implementations
		CA-2003-24: Buffer Management Vulnerability in OpenSSH
smtp	25/tcp	CA-2003-07: Remote Buffer Overflow in Sendmail
		CA-2003-12: Buffer Overflow in Sendmail
		CA-2003-25: Buffer Overflow in Sendmail
domain	53/tcp	
	53/udp	CA-2002-31: Multiple Vulnerabilities in BIND
bootps	67/tcp	
	67/udp	CA-2003-01: Buffer Overflows in ISC DHCPD Minires Library
bootpc	68/tcp	
	68/udp	CA-2003-01: Buffer Overflows in ISC DHCPD Minires Library
tftp	69/udp	CA-2003-20: W32/Blaster worm
http	80/tcp	CA-2002-27: Apache/mod_ssl Worm
		CA-2002-33: Heap Overflow Vulnerability in Microsoft Data Access Components (MDAC)
		CA-2003-09: Buffer Overflow in Core Microsoft Windows DLL
		Current Activity 08/18/2003: W32/Welchia Worm
hosts2-ns	81/tcp	CA-2002-35: Vulnerability in RaQ Server Appliances
sunrpc	111/tcp	
	111/udp	CA-2002-26: Buffer Overflow in CDE ToolTalk
epmap	135/tcp	
	135/udp	CA-2003-16: Buffer Overflow in Microsoft RPC
		CA-2003-19: Exploitation of Vulnerabilities in Microsoft RPC Interface
		CA-2003-20: W32/Blaster worm
		Current Activity 08/18/2003: W32/Welchia Worm
		CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows
netbios-ns	137/udp	CA-2003-08: Increased Activity Targeting Windows Shares
		CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows
netbios-dgm	138/udp	CA-2003-08: Increased Activity Targeting Windows Shares
		CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows
netbios-ssn	139/tcp	
	139/udp	CA-2003-03: Buffer Overflow in Windows Locator Service
		CA-2003-08: Increased Activity Targeting Windows Shares
		CA-2003-16: Buffer Overflow in Microsoft RPC
		CA-2003-19: Exploitation of Vulnerabilities in Microsoft RPC Interface
		CA-2003-20: W32/Blaster worm
		CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows
https	443/tcp	CA-2002-27: Apache/mod_ssl Worm
snpp	444/tcp	CA-2002-35: Vulnerability in RaQ Server Appliances
microsoft-ds	445/tcp	
	445/udp	CA-2003-03: Buffer Overflow in Windows Locator Service
		CA-2003-08: Activity Targeting Windows Shares
		CA-2003-16: Buffer Overflow in Microsoft RPC
		CA-2003-19: Exploitation of Vulnerabilities in Microsoft RPC Interface
		CA-2003-20: W32/Blaster worm
		CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows
rtsp	554/tcp	VU#934932: RealNetworks media server RTSP protocol parser buffer overflow
http-rpc	593/tcp	CA-2003-20: W32/Blaster worm
		CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows
kerberos-adm	749/tcp	
	749/udp	CA-2002-29: Buffer Overflow in Kerberos Administration Daemon
pump	751/tcp	
	751/udp	CA-2002-29: Buffer Overflow in Kerberos Administration Daemon
unassigned*	1052/tcp	CA-2002-27: Apache/mod_ssl Worm
lotusnote	1352/tcp	CA-2003-11: Multiple Vulnerabilities in Lotus Notes and Domino
ms-sql-m	1434/udp	CA-2003-04: MS-SQL Server Worm
h323hostcal	1720/tcp	
	1720/udp	CA-2004-01: Multiple H.323 Message Vulnerabilities
unassigned*	1978/udp	CA-2002-27: Apache/mod_ssl Worm
globe	2002/udp	CA-2002-27: Apache/mod_ssl Worm
ctx-bridge	3127/tcp	Current Activity 01/26/04: W32/Mydoom.A or W32/Novarg
		Current Activity 02/10/04: W32/Mydoom.C or W32.HLLW.Doomjuice
unassigned*	4156/udp	CA-2002-27: Apache/mod_ssl Worm

unassigned*	4444/tcp	CA-2003-20: W32/Blaster worm
sip	5060/tcp	
	5060/udp	CA-2003-06: Multiple vulnerabilities in implementations of the Session Initiation Protocol (SIP)
sip	5061/tcp	CA-2003-06: Multiple vulnerabilities in implementations of the Session Initiation Protocol (SIP)
unassigned*	6129/tcp	Current Activity 12/26/2003: Systems compromised via buffer overflow in DameWare
unassigned*	6778/tcp	CA-2002-32: Backdoor in Alcatel OmniSwitch AOS
font-service	7100/tcp	CA-2002-34: Buffer Overflow in Solaris X Window Font Service