Proceedings of the 2006 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY, 21-23 June 2006

Design and Implementation of the Honey-DVD

Maximillian Dornseif Felix C. Freiling Nils Gedicke Thorsten Holz

Abstract— **Honeynets are a valuable source of data about techniques, tactics and motives of attackers in the Internet, but up to now they have been notoriously difficult to set up and maintain. This work describes the development and implementation of an easy to use, freely distributable, bootable solution on DVD for deploying honeynets. The system is based on a live Linux distribution and can be set up without installing anything on a local harddrive. It sets up a group of virtually emulated honeypots and links them together in a virtual network. Moreover, a honeywall is added to protect the honeypots. The whole honeynet is configured and maintained via a centralised controller software on the DVD which allows an easy configuration and automates all necessary procedures in the virtual network.**

I. Introduction

A *honeypot* is an information system (computer, router, switch, etc.) designed for being attacked and compromised [1]. It employs special monitoring software so that administrators can stealthily monitor the actions of the attackers. Networks of honeypots are called *honeynets*. Founded in 1999, the Honeynet Project [2] aims at using honeynets to learn more about "the tools, tactics and motives involved in computer and network attacks" on the Internet.

Honeynets have become a valuable source of information about malicious network activity [3], [4], [5]. But deploying a honeynet is not an easy task and involves some risk. For example, it is far from trivial to install the special monitoring software on the honeypot since it must be installed in such a way so that it becomes invisible even to the superuser of the honeypot in case an attacker aquires root priviliges. In such a case, an attacker might even use the honeypot as the source for further attacks and cause harm to other (non-honeypot) systems which might entail legal consequences.

To deal with the above problems, the honeynet project provides special software and deployment standards for honeynets. For example, the Honeynet Standards [6] define the basic functionality of a honeynet to consist of *Data Capture* and *Data Control*. Data Capture means that all activity within the honeynet and the information that enters and leaves the honeynet should be captured without attackers knowing they are being watched. Data Control means that once a honeypot within the honeynet is compromised, all malicious activity must be contained within the honeynet.

University of Mannheim, Germany

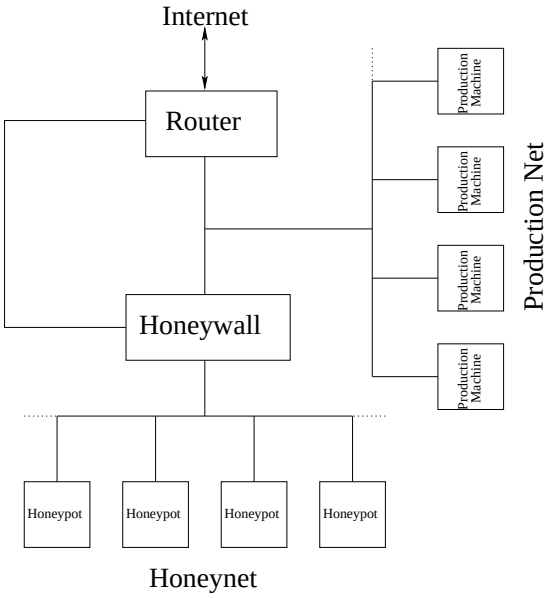

Fig. 1. Generic GenII honeynet layout

A good representative of honeynet technology are so-called second generation (GenII) honeynets. Figure 1 shows a typical layout. The central component of the honeynet is the *honeywall* which divides the network into the production network and the honeynet. The honeywall is equipped with three network interfaces. No IP address is assigned to the two network interfaces connected to the production network and the honeypots, and the traffic is directly forwarded between these two interfaces without increasing the time-to-live, thus it acts as a transparent bridge and is completely invisible to an adversary. The third interface, connected to the main network router, is for management purpose and does not interfere in any way with the honeynet traffic. Data Control and Data Capture are both implemented on the honeywall. Furthermore, every honeypot is equipped with tools to capture data in a stealthy way and export it to the honeywall.

Setting up a simple GenII honeynet entails a multitude of task, for example

- setting up the honeypots, configuring them, possibly involving different operating systems and different patch levels;
- setting up the honeywall and configuring it in a secure

yet stealthy way;
• setting up Data Capture mechanisms and configuring them.

The above points are only the most critical ones of the setup. The Honeynet Project provides a multi-volume series of documents about honeynets, their deployment and maintenance [2], [1], [7], [8]. But in general, the competence entry level of users of honeynet technology is relatively high, mostly at the level of expert system administrator.

One may argue that the complexity involved in setting up a honeynet is a good thing: Honeynets can be misused in the same way as any other security tool. However, we feel that new users with only modest administration experience should not be deterred so quickly from using and experimenting with this interesting technology. The Honeynet Project itself has taken giant steps to simplify the problem of honeynet deployment by creating the *honeywall CD-ROM*, a bootable CD which allowed the user to setup and maintain a honeywall in a quite intuitive and easy way. The latest development in this direction is a system called *Roo*, the so called Gen III honeynet. With the help of Roo, it is very easy to install a honeywall on any PC. However, the honeypots themselves must be installed manually. We are aware of other related work which uses virtual networks from bootable CDs as a training ground for data security exercises [9], but we know of no other work which has attempted to combine both attempts to setup a honeynet in a fully automated fashion.

In this paper we describe the design and implementation of the *Honey-DVD*, a solution for not only deploying a functional honeywall but a complete honeynet setup with not much more effort than booting the DVD. In the Honey-DVD, we combine different techniques which needed to be analysed during the design of the system. The first technology are *live Linux distributions*, distributions of the Linux system capable of booting from a CD or DVD and able to run completely in the memory without interacting with harddiscs. The second technology are *virtualisation techniques*, since we wanted to implement the complete honeynet setup as a virtual network of independent systems. The different virtualisation techniques available impose huge differences in their performance and usability with different operating systems. The third technique employed was *remote configuration*, since the Honey-DVD should be controlled using a centralised tool which should allow the user to configure every component of the honeynet like the honeypots' operating systems.

The resulting Honey-DVD system satisfies the following requirements:

• The Honey-DVD is bootable on every PC system able to boot from a DVD.
• The Honey-DVD provides a complete GenIII honeynet setup including different honeypots and the honeywall.
• The honeynet setup is simple but still flexible enough so that the Honey-DVD user can configure all important values.
• The maintenance of a running honeynet, including the basic capabilities for the analysis of the captured data, is possible right from the DVD.
• The Honey-DVD does not interfere with the data stored on the harddrives of the machine on which it is running.
• The Honey-DVD is freely distributable, i.e., it does not employ any commercial software.

The challenges of the Honey-DVD project lie in the trade-off between performance and DVD storage space. The contributions of this paper are the experiences we made and the lessons we learned by testing different parameter choices of this trade-off. The resulting DVD image can be downloaded from the authors' website for further experiments.

The paper is structured as follows: We report on the basic requirements and the available tools of the Honey-DVD in section II. We then elaborate on the design of the Honey-DVD components in section III. The implementation is described in section IV while section V reports on some performance measurements. Section VI concludes this paper.

II. Requirements and Available Tools

The Honey-DVD has two main ingredients: a live Linux distribution to enable autonomic bootup and a system virtualization technique to run the individual honeypots. A central requirement of this project was that the resulting DVD should be freely distributable. This is why only non-commercial open source projects were incorporated into the system.

A. Live Linux Distributions

A live Linux distribution is a distribution of the operating system Linux which is capable of starting and running from a CD or DVD without the need of being installed to a harddisc. Many live Linux distributions go even further and are able to run completely without using any harddisc space, which is also an integral part of the Honey-DVDs concept. The primary requirement for the chosen distribution were *stability*, i.e., the kernel and all software installed on the system should be fairly well tested versions, *security*, i.e., no known unpatched vulnerabilities, and *resource minimality*, i.e., the system should be very resource efficient since disk space, memory and CPU performance will obviously be critical issues, *maintainability*, i.e., it should be easy to update and patch main system components and software to enable stability and security, *compatibility* with most hardware, and it should support *easy customisation*, e.g., the startup sequence and automatic login behaviour should be configurable.

One of the distributions with the largest software repos-

itories is the Debian distribution. This distribution is easy to maintain (even without a running X server) and it is considered rather secure. We therefore chose this distribution as the basis for the Honey-DVD.

From all Debian-based systems available, the Knoppix [10] live Linux is probably one of the best and most common ones. Knoppix is very well documented and the automatic hardware detection and hardware compatibility are often considered better than the ones of commercial Linux distributions. Knoppix also allows on-the-fly decompression, which saves a lot of RAM and disk space; unused and unmodified data is deleted from the RAM-drive while virtually the directory structure is unchanged. However, Knoppix is one of the largest live Linux distributions available which makes it a bad candidate for the Honey-DVD.

Another live Linux distribution is GRML [11]. Based on the Knoppix distribution it comes with equal hardware compatibility. The differences to Knoppix in the base components are only minor, so that the Knoppix documentation is accurate to work with GRML. GRML is designed for recovery and administrator use so, for example, tools for system monitoring and testing are included. The overall structure of the GRML system, i.e., the startup scripts, configuration, etc., is easy to understand and the customisation can be done quite easily by using the standard Debian tools for installing or removing software. Furthermore, the standard GRML system is rather minimalistic; for example, it boots directly into a Linux zsh-prompt and by default allows access to three root and three user consoles only.

Overall, we chose GRML as the live Linux distribution for the Honey-DVD.

B. Virtualisation Techniques

Virtualisation environments allow to simulate multiple machines on a single physical machine. The simulated "virtual" machine will run individual honeypots which act as independent systems with their own operating system and virtual resources. Since there are not only many different providers of implementations, but also different ways of doing the virtualisation, the problem of choosing the right virtualisation software is not an easy one.

Following the classification of Singh [12], we decided that a *system emulator* would be the best way to combine performance with hard detectability. We are aware of the fact that most current virtualisation environments are detectable [13], therefore our goal is not to be completely stealth, but rather that it should be not too easy to detect the presence of a honeypot.

First we tested the more efficient *user mode emulation*. First experiments with Xen [14] and User Mode Linux (UML) [15], were encouraging since the maximum performance of the guest operating systems was nearly the same as on real machines and the timings of the virtual network were only slightly slower than those of the real machine. However, the network test with Xen showed that the connection was quite unstable (unexplainable packet losses and even complete link failures). Furthermore, a choice of both Xen and UML would have implied that only Linux could have been run as honeypot image. For Xen, some modified BSD versions are available and even a patch for Microsoft Windows XP was created (but due to some legal issues never officially released). But these modifications and patches only apply to newer system kernels what additionally diminishes the amount of usable operating systems. Moreover, the presence of UML can be easily detected by an attacker [13]. So unfortunately, user mode emulation did not turn out as the right choice.

In its book [1], the Honeynet Project [2] gives advice to use VMWare-Workstation [16], a full system emulator, to virtualise the honeypots. Although commercial (at the time of starting this research topic), this was the first choice for first experiments with full system emulation. Since beginning of 2006, VMware offers a version of VMware GSX server for free, so in the future this system could be used as a basis of a Honey-DVD. VMWare emulates multiple x86 based machines in one consistent framework and allows to customise the virtual systems, i.e., adding and removing resources or changing the network infrastructure, as needed in a very comfortable manner.

Another interesting candidate was the software QEMU [17], released under the GNU Lesser General Public License. QEMU allows to emulate x86/x86-64 PCs as well as PowerMac, PowerPC and SPARC Sun4m systems. It is not as comfortable as VMWare, since all configration is done via the commandline at startup and a text based monitor interface during runtime. It not only accepts disc images in its own *qcow* format, but also UMLs *cow* and some versions of VMWares *vmdk* format are supported. Other additional image formats are the easily exportable *raw*-, linux compressed *cloop*- and the simple *dd*-image. QEMU also comes with an accelerator module which is optional and allows QEMU to run most of the target application code directly on the host processor without the additional filtering for instruction sets if a PC is emulated on a PC. Unfortunately, this accelerator is a closed source proprietary product and redistribution is prohibited.

Overall we chose QEMU as the virtualisation software for the Honey-DVD.

III. Design of the Honey-DVD

After reviewing general requirements and tools for the Honey-DVD, we now discuss detailed requirements for the components used on the DVD. The main components of the Honey-DVD are the honeypots on the one side and the gateway (i.e., honeywall) on the other. These components must jointly perform different tasks, mainly Data Capture and Data Control.

A. Data Control

Following the honeynet standards [6], the Data Control mechanisms of the Honey-DVD will be implemented in two layers.

The first layer implements *connection rate limitation*. This is used to restrict the number of connections a honeypot can initiate. In this way it is, for example, possible to prevent the further distribution of malware from a honeypot or its participation in a distributed Denial-of-Service attack. The connection rate limitation can be easily implemented using standard tools like IPtables.

The second layer implements *extrusion prevention*. This means that the system prevents intrusions into other systems with the honeynet as a source. This is implemented using a modified version of the Snort intrusion detection system called snort_inline. It allows to analyse outbound packets for known attacks and gives the possibility of either dropping or modifying those packets.

B. Data Capture

Data Capture is performed by applying three complementary techniques. The first technique is *firewall logging*: since the honeypots normally do not originate any traffic, any activity on them is of interest. By logging every connection or connection attempt from or to a honeypot, the honeywall provides some basic information even if no actual data was transfered. For example portscans or single ping requests will be captured in this way.

The second techniques is classical *intrusion detection*: the maintainer of a honeynet will most likely want to be informed if a known attack on the system was tried. This information is not critical, but allows for example some statistical evaluations of what attacks are the most common and which backdoors the most known ones. Another reason to use an intrusion detection system is that such a system allows to log all network traffic for later use to do some offline analysis of the collected traffic with a vast number of tools.

Finally, the third technique is *data capture on the honeypots*: This technique is typical for a honeynet. While the honeywall has total control about the in- and outbound traffic, some of the intruders' actions are only "visible" on the attacked system itself. This can be because either the network connection is encrypted or because the activity of the tools used by the intruder are only local. The Honeynet Project uses Sebek [18], a system activity logger, developed specifically to meet honeynet needs. Sebek collects process tree, socket and file opening data and is available for Linux 2.4, Linux 2.6, all major BSD releases, and Win32. Roo [19], the new honeywall CDROM distributed by the Honeynet Project to build Gen III honeynets, implements a Sebek server. This means that the honeywall itself can collect the data retrieved by the Sebek client which is installed on the individual honeypots.

C. Honeypot Images

The final design decision targets the type of honeypots to include in the Honey-DVD. Because of the space restrictions of a DVD, the used images needed to trade off disk space for functionality.

A couple of design decisions had to be made in the selection of the honeypot images. First of all, to be used as a honeypot, the system should in some way be attractive for attackers. One way to achieve this is to install older versions of widely used operating systems instead of the newest and best patched ones. But of course a system which is too old would also be a bad choice, because no new security threats will be monitored on those systems. For us, a system which is about one year old seems to be the best choice.

Another requirement of the operating system is that it must be supported by Sebek. The documentation of the current version of the Linux client (at time of writing this was version 3.0.3) states, that it has been successfully tested on Redhat systems with kernel 2.4. The BSD versions are available for FreeBSD 5.3, NetBSD 2.0 and OpenBSD 3.7. The recently released Win32 version runs on Windows 2000, XP and 2003. However, Microsoft Windows as well as other commercial operating systems were excluded since we wanted to freely distribute the Honey-DVD. This also concerned the available services, i.e., services like a web- or FTP-server with a free implementation should exist for the operating system in use.

All in all the possible choices as honeypot images were reduced to the BSD and Linux distributions.

IV. Implementation of the Honey-DVD

We now report on how we implemented the Honey-DVD. When devising the network layout of the virtual honeynet, we initially followed the standards of the Honeynet Project [6]. However, we encountered several problems which we analyze in section IV-C. We finally describe the current layout of the Honey-DVD in section IV-D.

A. Initial Layout

The first version of the Honey-DVD was designed to follow the standards of the Honeynet Project [6] as closely as possible. But since the complete honeynet should run on only one physical machine and without the necessity of installing anything to a harddrive, some modifications where inevitable.

A.1 The Virtual Honeynet

The initial network layout of the Honey-DVD is given in figure 2. It follows the standard layout of a GenII honeynet and corresponds to what is called *self-contained virtual honeynet* in [1]. Some modifications had to be made, however. In a real network, several machines with their network interfaces would be connected to the gateway over a

switch or hub. In the virtual honeynet mentioned above all honeypots run in the VMWare framework, which internally uses a network bridge to implement the virtual network and further provides a single network interface on the real machine connecting to this network. However, QEMU does not provide such a framework. QEMU assigns a virtual network interface on the host system to every virtual network interface on the guest systems by using the TUN/TAP virtual point-to-point device driver. Hence every single virtual machine has its own network interfaces TUN/TAP counterpart on the real system that must be connected to the honeynet via an aditional network bridge *br0*. The base system is used as the gateway to connect this bridge to the interface *eth0* which is used as the connection to the external network. The Ethernet bridge implementation in the Linux kernel is not capable of directly connecting two Ethernet bridges. A consequence of this limitation is that the Honey-DVD has to use IPtables' NAT capabilities to connect the virtual network to the external network.

Like in other honeynets, the network interface *eth1* is used as a command interface which will not be connected to the honeynet, but will provide the means to export logs, send alarm signals to the administrator or to configure the honeynet from the outside via an SSH connection.

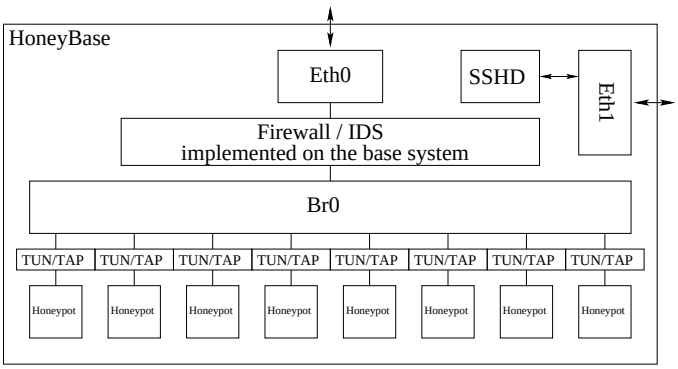

Fig. 2. Initial Network layout of the Honey-DVD

In addition to serving as a gateway, the base system also provides the main functionality of the honeywall. The key part of the honeywall is a script called *rc.firewall*. This script first handles the network setup, i.e., setting up the bridging and NAT to connect the external and internal networks and activating the local control interface. It then configures IPtables to enable connection rate limitation. Morover, outbound packets are passed to Snort_inline. It also configures IPtables so that Sebek packets do not leave the virtual network. The base system also provides the crucial functionality of the Sebek server.

B. Honeypot Configuration

To be as flexible as possible when deploying the honeynet, we attempted honeypot image reuse, i.e., it should be possible to reuse the base images and run several honeypots with different configurations from them. The individual configuration of these images should be done before startup.

To be able to change the configuration of the honeypots before starting the system, one must edit the relevant configuration files in the image (like for example */etc/rc.config* in FreeBSD 5.3). But to be able to run several honeypots from the same image, one would have to make copies of the image for every single honeypot and store them on the DVD. To avoid this, a second image for the configuration files of the honeypot base images was used. The virtual machine running the honeypot has two virtual harddrives. The first harddrive (*hda*) contains the complete system image of the honeypot. Every configuration file is "exported" to the second harddrive *(hdb)*. These files are then linked to their original locations.

Since the size of the second harddrive image is extremely small (10 MB) it is no problem to make copies of this image at runtime of the Honey-DVD and start different honeypots from the base images with their own copy of the configuration image.

During implementation we noticed that the FreeBSD standard file system is ufs is readable under Linux, but the write support is not reliable. Fortunately, FreeBSD has integrated support for the ext2 file system and would be able to use this as a source for the configuration. Unfortunately the ext2 support ends at the filechecking utility which is doing file system checks on all devices listed in */etc/fstab* during the boot sequence. The problem would be solvable by editing the fsck bootscript to exclude the ext2 file system from checks, but when editing the files on the ext2 image on the Linux system some inconsistencies occur which prevent the ext2 image from being remountable in the FreeBSD environment. These inconsistencies are mainly caused by interoperability problems with the ext2 handling of Linux and FreeBSD systems: if the Linux system changes the filesystem, FreeBSD marks this filesystem as corrupt. We did not examine the cause for this in more detail. The third and final possibility of solving this issue was to search the "least common denominator" file system: The FAT file system is supported by both system and although it is not the native system of either one it seems to have none of the above mentioned problems.

Another problem we encountered was that the boot sequence of FreeBSD accesses the main configuration files in a system state, where only the root file system is mounted read-only. The solution here was to edit and change the standard bootup procedure of FreeBSD. By just mounting the configuration filesystem before the contained values are needed and unmounting it again after they have been read, it is possible to bring the configuration file to the attention of the bootup process without interfering with the standard mounting process.

For the base version of the Honey-DVD only a few services besides SSH and telnet were chosen to be available on the honeypots for testing purposes. On the Redhat system these services were an Apache2 webserver and an FTP server called proftpd. Both services can be started as daemons with external configuration files which can easily be included in the configuration images for the honeypot. Similar to the RedHat system, we included in the FreeBSD system the Apache2 webserver as well as an FTP server. Additionally the FreeBSD system comes with a TFTP server and some minor services like *chargen*, *echo*, *time* etc.

B.1 The Honeynet Controller

We developed a honeynet control program for the Honey-DVD that is used to configure the honeynet before startup and during runtime. It provides a simple default configuration so that the honeynet can be used without user modification.

The default configuration can be customized in many ways, e.g., by configuring the honeywall, the number and kind of honeypots, and the honeypots themselves.

After configuring the honeynet, the controller writes out the previously defined configuration values to the appropriate files, brings up all real network devices and network bridges, starts the Data Capture and Data Control mechanisms of the Honey-DVD, and finally starts the honeypots.

During runtime, it is possible to change the honeynet using the controller program in many ways: For example, it is possible to reconfigure a honeypot (only additional services, no IP reconfiguration), to configure and start an additional honeypot (includes restart of the honeywall and hence a short break in the external connection of the honeynet), to restart the complete honeynet with the current configuration, to stop the honeynet for a complete reconfiguration, or to stop the honeynet and shutdown the Honey-DVD.

Additional functionalities like data analysis are only available outside of the controller and very limited due to the fact that the whole Honey-DVD system is running in the real machine's memory and hence the possibilities of storing great amounts of data are restricted. The Honey-DVD controller provides possibilities to export captured data like IPtables logs to a remote location on the network while Sebek and snort are both capable of exporting their data to SQL servers themselves. Maybe the most notable functionality we implemented in the controller is the possibility to take snapshots of the honeynet. This is implemented on top of QEMU's snapshot mechanism. To use this feature, some additional storage media must be connected to the Honey-DVDs host machine.

C. Analysis of Initial Prototype

Although fully functional, the first implementation had several problems which made it unacceptable for a "real" Honey-DVD. Firstly, since the honeynet is operating in NAT mode, the IP range of the honeynet is a different network from the external network. In bridging mode it would be possible to choose the IP range of the honeynet completely without such restrictions. Secondly, the honeynet Project now uses and propagates the Roo honeywall CD-ROM as the standard honeywall setup for their networks. The Roo system allows to gather data from multiple honeynets and offers some other advanced features [20].

Finally, the Honey-DVD's method for configuring the honeypots allows easy *fingerprinting* of the honeynet, i.e., an adversary that knows about the structure and key elements of the Honey-DVD can look for these things to identify honeynets based on it. For example, a Honey-DVD honeynet can be easily identified by looking for the particular symlinks, the centralised stored configuration files, or just the hard disk device: all honeypots use the same configuration and especially the hard disk device is rather small since it stores only the configuration information.

D. The Current Version of Honey-DVD

Based on the previous analysis the current version of the Honey-DVD was implemented. The key solution to the main problems was to add another virtual machine to the network on which the Roo honeywall was installed. Next to upgrading to Honeynet Project's best practices, this move solved the bridging problem and added some more flexibility. The main possible consequence, performance loss, was considered secondary.

Due to the static nature of the Honey-DVD, is is impossible to avoid fingerprinting of default or slighlty changed honeynet configurations. Nevertheless, one can try to make fingerprinting as hard as possible. A combination of methods is used in the current Honey-DVD to achieve this. For example, an additional script in the boot procedure replaces the symlinks with the real configuration files, deletes the files on the configuration harddisk, unmounts the configuration harddisk, removes the mount entry from */etc/fstab*, deletes the configuration folder, and selfdeletes the cleanup script. This procedure cannot prevent the scanning of the existing devices, but it helps masking the changes and reduces their obviousness.

The honeynet layout implemented in the current version of the Honey-DVD is shown in figure 3. Again the interfaces *eth0* and *eth1* are used for external and management connections. The additional network bridges *br1* and *br2* are used to connect the external and management interfaces of Roo to the external interfaces of the real machine, while the old bridge *br0* will be used to connect the honeypots to Roo's internal interface.

One drawback of Roo is that according to its manual [21] 5 GB of disk space are necessary for testing purpose and 10 GB for productive usage. As a result, it was necessary to change the medium for the Honey-DVD to a Double-

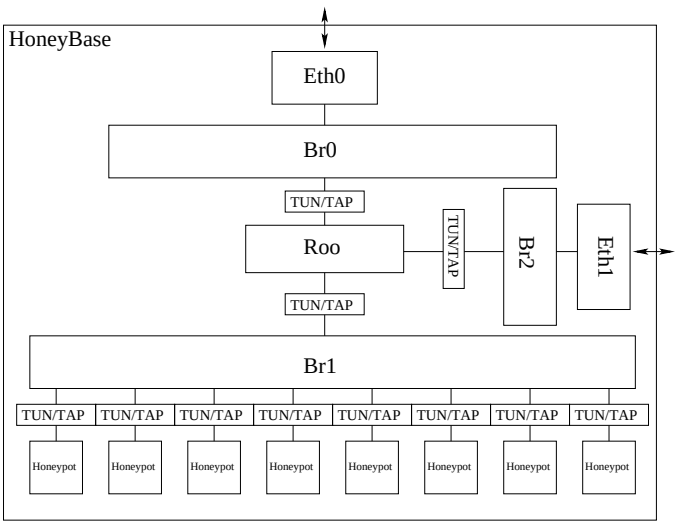

Fig. 3. Network layout of the Honey-DVD (current version)

Layer-DVD with a capacity of 8.5 GB. But also on this larger medium it was only possible to use a 4 GB image for the honeywall due to available disc space. We will discuss the consequences of this restriction later.

V. Performance of the Honey-DVD

After building the current version of the Honey-DVD we ran some tests to study the performance of its honeynet. These test were done on different systems to get an overview which components are critical for the usability of the implementation.

A. Test Setup

It was of great importance, that the only variables in the applied tests would be the components of the system running the Honey-DVD so we tested just a single (default) honeynet. The following test setup was used:

- The test was always done with two machines: (1) The test machine running the Honey-DVD. (2) The testing machine performing the tests and gathering the results.
- The honeywall management interface was added to the external network bridge as described earlier.
- Both machines were connected to a minimal network with a Cat 5 patch cable.

B. Test Parameters

The following test parameters were measured:

1. The booting time of the base system from Honey-DVD start until the Honey-DVD Controller is available.
2. The time needed to start the honeywall alone on the test system.
3. The time needed to start the default configuration.
4. The response time (ping) of a single honeypot without the honeywall or other active honeypots.
5. The response time (ping) of a single honeypot in the default configuration.
6. The response time of the service interface (Walleye) of the honeywall when no honeypot is running.
7. The response time of the Walleye interface of the honeywall in the default configuration.

Measured Parameter (with unit)	**Type 1** (5 runs)	**Type 2** (3 runs)	**Type 3** (2 runs)
1 (in min)	3	2.5–3.5	1–1.5
2 (in min)	17–23	13–15	6–7
3 (in min)	18–27	14–15	6–7
4 (in ms)	70	60–65	60
5 (in ms)	75–80	60	60
6 (in min)	2–3	1	0,5
7 (in min)	2–3	1	0,5

TABLE I
Test results. For a description of the measured parameters see section V-B.

B.1 Test Scenarios

The free variable of our tests was the configuration of the test machine. We used three different hardware setups that represent common classes of systems encountered in practice:

1. This setup used a typical example of memory restricted machines with average processor capabilities. The reference system was a one year old laptop (Intel Pentium M CPU at 1,6GHz, 512MB of RAM, 400MHz FSB, and a DVD-Drive speed of 8x).
2. This setup aimed at a standard desktop PC system with moderate memory. The reference system employed an Athlon-XP 1900+ CPU with CPU-Speed 1,6GHz, 1 GB of RAM, FSB 266MHz, and DVD-Drive speed 16x.
3. This setup represented the class of modern high end PCs. The representative of this class is a new desktop PC with maximum memory (Athlon 64 3200+ CPU, CPU-Speed 2GHz, 2GB of RAM, FSB 1GHz, DVD-Drive speed 16x):

C. Test Results and Analysis

Table I summarises the results of the the tests above. Values are given as minimum and maximum values of several system runs.

One observation which is common to all systems is that the honeywall is the all-dominant part of the Honey-DVD setup. It is the virtual system which takes the longest period of time to start and perform any activity and thus determines the overall performance of the Honey-DVD nearly independently of the running honeypots. This is mostly due to the small image size of the honeywall image on the DVD.

We further studied the performance loss due to the decision to virtualise the honeywall and installed the "unvirtualized" Roo honeywall to the reference system of setup 1 and measured the needed time for reconfiguration after the boot sequence. The result was a time of approximately two minutes. Comparing this time to the six minutes measured on the Honey-DVD the performance of the Honey-DVD's honeywall is about three times slower than on a real system. We found this performance overhead acceptable given the fact that a larger image size of the honeywall on the DVD would severely restrict the flexibility of the Honey-DVD alltogether.

Another important result is that the available memory in the real system running the Honey-DVD seems to be a bottleneck. Although the Honey-DVD is able to run on systems with 512MB RAM (test setup 1) it is quite obvious that the performance and usability of the Honey-DVD on this system is not really acceptable. Even the test system 2 with 1 GB RAM needed a lot of time to start the default network and had a large delay when accessing the Walleye interface.

VI. Conclusion and Future Work

The Honey-DVD is an easy to use method of deploying honeynets without the necessity of installing anything to a harddrive. To achieve this, different components were analysed and tested regarding the requirements of this work. The current setup fits onto a 8.5 GB Double Layer DVD and allows to deploy a fully functional honeynet on nearly every PC capable of booting from a DVD.

However, the current version of the Honey-DVD is still a testing release which is more a proof-of-concept system than a version for productive use. For example the Sebek sources are still visible on the honeypots and must be removed if intelligent attackers should be attracted. As another example for the "prototype-ness" of the Honey-DVD, all passwords in the honeynet are set to default values. Further development must refine these issues into a production system. Also possible as future work is to explore possibilities to reduce the necessary amount of memory. For example, one could use the upcoming remote logging capabilities of Roo. In this way it should be possible to reduce the size of the honeywall image to a minimum while exporting the captured data to a remote location. An additional possibility would be to implement a Roo compatible honeywall on the base system. Doing this, one would avoid the problem of fixed disc image sizes due to the dynamic nature of the live Linux memory management.

References

[1] The Honeynet Project, *Know Your Enemy – Learning about security threats*. Pearson Education, Inc., 2nd ed., 2004.

[2] "The Honeynet Project Homepage." `http://www.honeynet.org/`. Last checked: 12.2005.

[3] The Honeynet Project, "Know Your Enemy: Tracking Botnets," March 2005. `http://www.honeynet.org/papers/bots/`.

[4] The Honeynet Project, "Know Your Enemy: Phishing," May 2005. `http://www.honeynet.org/papers/phishing/`.

[5] N. Provos, "A virtual honeypot framework," in *Proceedings of 13th USENIX Security Symposium*, pp. 1–14, 2004.

[6] The Honeynet Project, "Honeynet definitions, requirements, and standards ver 1.6.0." `http://www.honeynet.org/alliance/requirements.html`, Oct. 2004. Last checked: 12.2005.

[7] The Honeynet Project, "Know your enemy: Honeynets," May 2005. `http://www.honeynet.org/papers/honeynet/index.html`, Last checked: 12.2005.

[8] The Honeynet Project, "Know your enemy: GenII honeynets," May 2005. `http://www.honeynet.org/papers/gen2/index.html`, Last checked: 12.2005.

[9] M. Dornseif, F. Freiling, M. Mink, and L. Pimenidis, "Teaching data security at university degree level," in *Proceedings of the IFIP TC11 WG11.3 Fourth World Conference on Information Security Education*, 2005.

[10] K. Knopper, "Knoppix - live linux." `http://www.knopper.net/knoppix/`. Last checked: 12.2005.

[11] "GRML – linux for geeks." `http://www.grml.org`. Last checked: 12.2005.

[12] A. Singh, "An introduction to virtualization." `http://www.kernelthread.com/publications/virtualization/`, 2005. Last checked: 12.2005.

[13] T. Holz and F. Raynal, "Detecting honeypots and other suspicious environments," in *Proceeedings of the 6th IEEE Information Assurance Workshop*, (West Point), IEEE, 2005.

[14] "The Xen virtual machine monitor." `http://www.cl.cam.ac.uk/Research/SRG/netos/xen/index.html`. Last checked: 12.2005.

[15] "The user-mode linux kernel home page." `http://user-mode-linux.sourceforge.net/`. Last checked: 12.2005.

[16] "Vmware – Virtual infrastructure software." `http://www.vmware.com/`. Last checked: 12.2005.

[17] "QEMU generic open source processor emulator." `http://www.qemu.org`. Last checked: 12.2005.

[18] The Honeynet Project, "Know your enemy: Sebek," Nov. 2003. `http://www.honeynet.org/papers/sebek.pdf`, Last checked: 12.2005.

[19] The Honeynet Project, "Know your enemy: Honeywall CDROM Roo," Aug. 2005. `http://www.honeynet.org/papers/cdrom/roo/index.html`, Last checked: 12.2005.

[20] E. Balas and C. Viecco, "Towards a third generation data capture architecture for honeynets," in *Proceeedings of the 6th IEEE Information Assurance Workshop*, (West Point), IEEE, 2005.

[21] The Honeynet Project, "The Roo unline user's manual." `http://www.honeynet.org/tools/cdrom/roo/manual/index.html`. Last checked: 12.2005.