GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at http://www.giac.org/registration/gsec

GIAC Security Essentials Certification (GSEC)
Practical Assignment
Version 1.4b

Second Generation Honeynet Honeywall
Author: Brough Davis

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Table of Contents

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

1. Introduction

This paper will attempt to explain the common components of a second generation (GenII) honeywall in a honeynet honeypot. Before the honeywall is discussed the concept of a honeynet honeypot will be explained. The rest of the paper goes on to describe how to install and configure a GenII honeywall.

Creating a honeywall can be quite time consuming. Trying to figure out ways to automate a lot of the installation and configuration process can save a lot of time. A PXE/Kickstart method of quickly creating a Honeywall system was developed and is shared in the Appendix. For information on this method a website was created for public reference and input at http://www.honeywall.org.

2. What is a Honeynet Honeypot?

Since a honeywall is a component of a honeynet honeypot, in order to explain what a honeywall is it important to give a brief explanation on what a honeynet honeypot is. A honeynet is a type of honeypot. A honeypot is a general term used to describe a system that finds some value in gathering information from an unauthorized or illicit use of a system resource. There are two types of honeypots; low-interactive and high-interactive. A low-interactive honeypot is an application that is quickly configured and maintained that takes care of all data control, data capture, and alerts for a honeypot system. An example of a low-interactive honeypot is honeyd. A high-interactive honeypot is a system that is highly configurable across each component for data control, data capture, and alerts. A honeynet is not only an example of a high-interactive honeypot but also a conceptual network architecture.

A honeynet is a network that's sole purpose is to analyze new exploits to unknown vulnerabilities for a system on that network. The Honeynet is widely used for research that helps keep the security industry one step closer to new exploits and attacks from the black hat hacking community. While there will always be a new exploit for an unknown vulnerability, the honeynet allows for an organization to be aware of the vulnerability much sooner. The less time it takes for an organization to find out what vulnerabilities its systems have the more time that organization has to fix the vulnerability.

The honeynet project sprang out of the imagination from a group of security industry professionals in 1999. The non-profit organization has been going strong ever since trying to raise awareness and to help provide a central location to provide open source tools for honeynet research. More information around the honeynet project can be found on the official website http://www.honeynet.org.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Many low-interactive honeypots focuses around a sole server that is configured to look like a standard system to a hacker. The system has programs or configurations that attempts to keep the hacker(s) interested in the honeypot servers and away from production equipment. Some honeypot systems go one step further and jail or slow the hacker down with the bogus services running on the honeypot. One such honeynet device called LaBrea (http://labrea.sourceforge.net/labrea-info.html), which finds un-used IP space in the production network and opens up fake services to attract hackers. The fake services basically confuse or slow down any hacker progress or information gathering. This paper does not discuss specific honeypot server configurations. More information on honeypots can be found at http://www.tracking-hackers.com

To put it very simply the honeynet is comprised of "bait" victim systems and a gateway device in front of the "bait" victim systems. These victim systems can be any standard OS. One large value out of the honeynet is that by using standard server equipment a real world attack can be seen exactly like in a production environment. It is the gateway system known as the honeywall that this paper focuses on.

2.1. Why Have a Honeynet?

Honeynets provide a great safe testing ground for systems that a company maybe testing or already implement. Educational/Research groups are interested in new and emerging threats that may exist on the internet. For companies; a honeynet may provide a good means of testing how secure a system(s) that is similar to a production system(s). Honeynets require a lot of maintenance to watch for alerts from possible attacks. Not only do the alerts need to be watched but also any new Intrusion Detection rules to keep up with new attacks that are emerging. Honeynets also require the use of extra equipment and resources in order to create and maintain a honeynet. Also, there is no guarantee that a possible vulnerability on a system will be attacked. Time sensitive security projects are hindered by this "waiting around" model. These reasons are why most honeynets are mainly found in research or hobby networks and are not commonly found in non-security focused commercial company networks. Another reason why companies are not fully embracing honeynets is for legal reasons discussed in the next section.

2.2. Are Honeynets legal?

This question has recently come up in the last few years or so. It seems the controversy around a honeynet is similar to what some people consider un-lawful entrapment.

For example the developer of honeyd, a low-interactive honeypot, had some problems hosting his application on a Michigan ISP due to the Michigan state

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

law. The clause that is currently in dispute is a small section from the Michigan Penal Code Act, Section 750.540c, that states

"(1) A person shall not assemble, develop, manufacture, possess, deliver, offer to deliver, or advertise an unlawful telecommunications access device or assemble, develop, manufacture, possess, deliver, offer to deliver, or advertise a telecommunications device intending to use those devices or to allow the devices to be used to do any of the following or knowing or having reason to know that the devices are intended to be used to do any of the following:"

…

"b)Conceal the existence or place of origin or destination of any telecommunications service"

To try to get around this clause some people put up a banner for incoming services (e.g. telnet, SSH, etc) to waive the hackers rights to being monitored. The following example banner was taken from a security focus paper by Lance Spitzner at http://www.securityfocus.com/infocus/1703.

```
##########################################################
#              !READ BEFORE CONTINUING!
#  This system is for the use of authorized users only.
#  By using this computer you are consenting to having
#  all of your activity on this system monitored and
#  disclosed to others.
##########################################################
```

This paper does not discuss other state or US federal legislations tied to honeynets. It does seem that people creating honeynets are passing them off as pure research projects or are moving such development to overseas honeynet friendly regions. For example the developer of honeyd has since moved his development to the Netherlands.

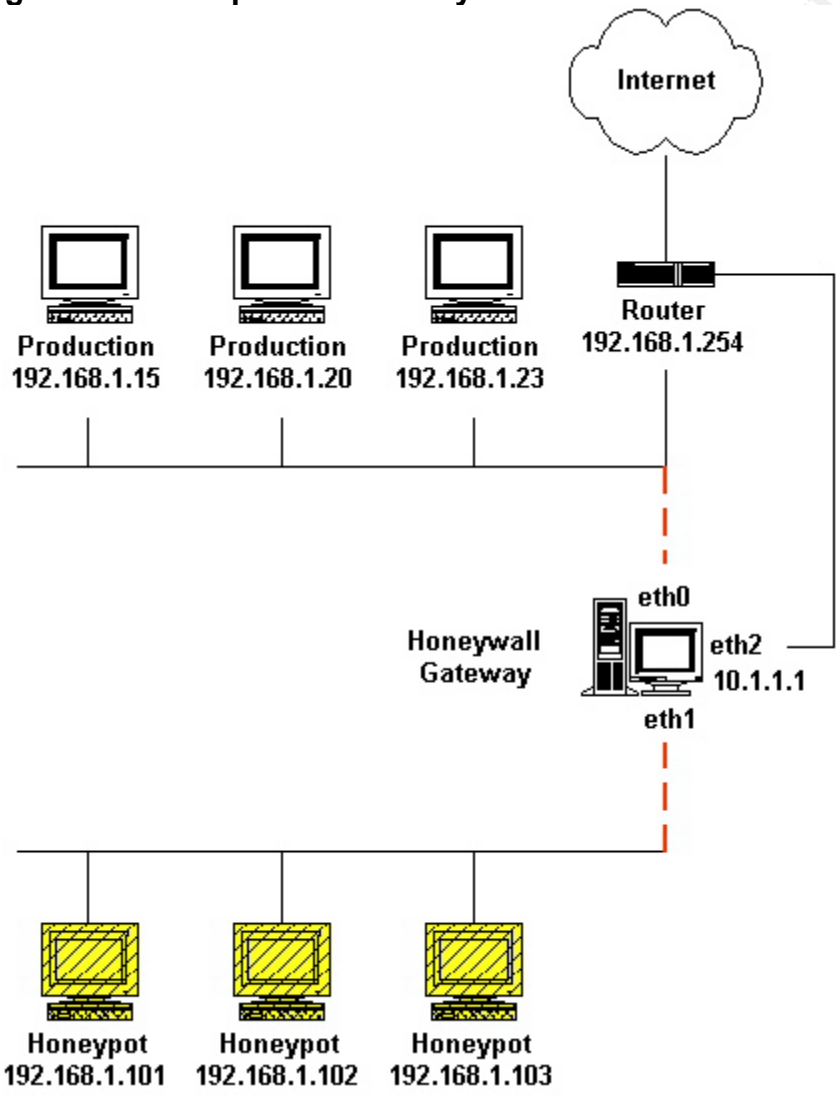
While this paper does not delve heavily into the legal ramifications of honeynets it is the opinion of this author that limiting honeynet development would not benefit the research and commercial sectors to help protect against new forms of internet attacks/exploits. For more information on finding out how to support the continuation of similar technology development please go to the electronic frontier foundations website at http://www.eff.org/.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

3. What is a Honeywall?

The gateway device in a honeynet honeypot is called a honeywall. Diagram 1 describes where a honeywall is usually placed in a honeynet network. The honeywall can be considered the main point of entry and exit for all network traffic for a honeynet honeypot. This allows for complete control and analysis of all network traffic to and from a honeynet system.

Diagram 1 – Example GenII Honeynet



There have been different phases of development around the honeynet. Most recently there has been a second generation design that focuses on the honeywall design. When the first generation honeynet was discusses the honeywall was basically a router that performed Network Address Translation.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

One major flaw to this design is that a remote hacker can now know an extra "hop" exists before the "bait" honeypot systems. Most hackers want to know as much as possible about a possible network as possible. Part of this is just the natural curiosity but mainly it's to figure out if there are any firewalls or intrusion detection systems that maybe tracking their movements. If the honeywall has an IP address the hacker can see this could send red flags to the hacker that there is some firewall/ids/honeywall in front of the bait systems. This prompted the next generation honeywall to move to a bridged environment in which a remote hacker would have no idea that traffic was passing through a honeywall.

4. Honeywall Components

The following sections describe the various components of a honeywall as similar to the description by the honeynet.org technical whitepaper. The honeynet project white papers describes a honeywall to have three main objectives; Data Control, Data Capture, and Alerting.

4.1. Data Control

Data control's main purpose is to make sure that the honeywall goes un-noticed and protects the rest of the internet from compromised honeynet "bait" systems.

4.1.1. Bridging

One main distinction of a second generation honeynet honeywall is the bridging capability of the honeywall. As discussed earlier the honeynet has evolved to be as anonymous as possible to the remote hacker. Configuring a system that bridges traffic is discussed later.

4.1.2. Outbound Filtering

Another main distinction of the second generation honeynet honeywall is the rate limiting feature. Rate limiting allows the honeywall to protect the rest of the internet from Denial of Service attacks that may be coming from a compromised honeynet system.

An application commonly used to control network filtering is **iptables**. The filtering a honeywall does is not very sophisticated. One main reason is that the honeynet should allow any hacker through the honeywall into the honeynet honeypot. The methodology is the opposite for outgoing network traffic. Once a honeynet system has been compromised it is important to set iptables filters to limit what a compromised system can send out to the internet. Since Denial of Service attacks are very common from compromised machines an iptables feature is used to rate limit the amount of traffic coming from the honeynet destined to the internet. This iptables feature must be enabled by compiling the Linux OS Kernel with rate limiting features. Configuration of this feature is described later.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The honeynet project was nice enough to provide standard rule sets to enable rate limiting as well as other filters (e.g. honeywall management). These rules can be found at the following website: http://www.honeynet.org/tools/rc.firewall

4.1.3. Network Intrusion Detection/Prevention

A common Network Intrusion Detection application that is commonly used is **snort**. The honeywall needs to prevent harmful traffic **to** the internet while listening on the internal (honeynet side) interface. The common snort implementation uses the opposite methodology in which snort tries to detect harmful traffic coming **from** the internet while listening on the external (internet side) interface.

Another objective of the honeywall is to be proactive in stopping harmful attacks out to the internet. It is important to reiterate that the honeynet objective is to let hackers in but not let them use a compromised system to attack internet clients outside the honeynet. Because of this proactive requirement a hybrid snort application was developed named **snort_inline**. Snort_inline is referred to as a Network Intrusion Prevention (NIP) system. A system with snort_inline can be placed in the network as a routing gateway (e.g. GenI honeynet) or a bridged device (e.g. GenII honeynet).

Snort_inline was developed from the honeynet project. Since snort_inline was developed for a honeywall the network definitions defined in the snort_inline configuration file (snort_inline.conf) already have the networks reversed from what a standard snort implementation would look like. For more information on snort_inline see the following website http://sourceforge.net/projects/snort-inline/

4.2. Data Capture

The corner stone of a honeywall is capturing the actions of an attempted or successful attack on a honeynet honeypot system. In order to do this both iptables and snort are used to capture suspicious network traffic.

While snort_inline is used to control outbound traffic to the internet, snort can be used to detect and log intruder network traffic coming from the internet. When trying to diagnose how a hacker compromises a honeynet system having network traffic logs from snort is important to quickly figure out if a known attack or exploit was used. If an attacker uses an un-known exploit or attack that snort does not recognize snort dumps all network traffic it sniffs into a common logging format to go over later for packet by packet analysis.

Honeynet System Tools

While not the subject of this paper it is important to note that capturing data on the honeynet honeypot system is key to figuring out how the system gets compromised at a system level. Applications such as sebek exist that secretly

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

track system actions such as user key strokes and reports them back to a logging server. For more information on sebek go to http://www.honeynet.org/tools

4.3. Alert

It is important to be notified as quickly as possible when snort_inline and/or iptables log on certain suspicious traffic. It is this reason that a log monitor that reads all system log files can alert the honeynet administrator when a snort rule is matched. One program that is commonly used for this purpose is swatch. Swatch installation and configuration is described later in this document.

5. Creating a Honeywall

This section explains how to install and configure all the pieces that make up a honeywall in a second generation honeynet network.

Some work has been done on creating bundled applications that do all of the Data Control, Data Capture, and Alert functions. Some of these efforts are open source research projects while others are more commercial ventures. One application that is getting more notice is the honeyd application. For more information please see the following website:
http://www.citi.umich.edu/u/provos/honeyd/

5.1. Linux OS Installation

Linux was chosen as the honeywall operating system since most honeywall components were developed in a Linux environment. Honeywalls could theoretically be developed on any type of system such as BSD, Solaris, or Windows platforms. For this paper a minimum install of Redhat 9.0 was done.

5.2. Kernel Modification

After installing Redhat 9.0 the kernel needs to be upgraded and compiled for bridging and special iptables options (e.g. queuing). The kernel can be downloaded from http://www.kernel.org
After downloading the kernel, decompressing it, and untar'ng it (in /usr/src), the following compiling procedures were performed.

```
/usr/src/linux-2.4.21
make menuconfig (see kernel options)
make dep
change Makefile (name Extraversion)
make bzImage
cp /usr/src/linux-2.4.21/arch/i386/boot/bzImage /boot/bzImage.name
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
cp /usr/src/linux-2.4.21/.config /boot/config.name
edit /boot/grub/grub.conf
```

Note: The ncurses package may need to be installed before 'make menuconfig' is performed.

Kernel Options

The following kernel options need to be enabled for IPTABLES (all sub-options) and 802.1d (bridging). The following is an excerpt out of the .config file after 'make menuconfig'

```
CONFIG_IP_NF_CONNTRACK=y
CONFIG_IP_NF_FTP=y
CONFIG_IP_NF_AMANDA=y
CONFIG_IP_NF_TFTP=y
CONFIG_IP_NF_IRC=y
CONFIG_IP_NF_IPTABLES=y
CONFIG_IP_NF_MATCH_LIMIT=y
CONFIG_IP_NF_MATCH_MAC=y
CONFIG_IP_NF_MATCH_PKTTYPE=y
CONFIG_IP_NF_MATCH_MARK=y
CONFIG_IP_NF_MATCH_MULTIPORT=y
CONFIG_IP_NF_MATCH_TOS=y
CONFIG_IP_NF_MATCH_ECN=y
CONFIG_IP_NF_MATCH_DSCP=y
CONFIG_IP_NF_MATCH_AH_ESP=y
CONFIG_IP_NF_MATCH_LENGTH=y
CONFIG_IP_NF_MATCH_TTL=y
CONFIG_IP_NF_MATCH_TCPMSS=y
CONFIG_IP_NF_MATCH_HELPER=y
CONFIG_IP_NF_MATCH_STATE=y
CONFIG_IP_NF_MATCH_CONNTRACK=y
CONFIG_IP_NF_FILTER=y
CONFIG_IP_NF_TARGET_REJECT=y
CONFIG_IP_NF_NAT=y
CONFIG_IP_NF_NAT_NEEDED=y
CONFIG_IP_NF_TARGET_MASQUERADE=y
CONFIG_IP_NF_TARGET_REDIRECT=y
CONFIG_IP_NF_NAT_AMANDA=y
CONFIG_IP_NF_NAT_LOCAL=y
CONFIG_IP_NF_NAT_IRC=y
CONFIG_IP_NF_NAT_FTP=y
CONFIG_IP_NF_NAT_TFTP=y
CONFIG_IP_NF_MANGLE=y
CONFIG_IP_NF_TARGET_TOS=y
CONFIG_IP_NF_TARGET_ECN=y
CONFIG_IP_NF_TARGET_DSCP=y
CONFIG_IP_NF_TARGET_MARK=y
CONFIG_IP_NF_TARGET_LOG=y
CONFIG_IP_NF_TARGET_ULOG=y
CONFIG_IP_NF_TARGET_TCPMSS=y
CONFIG_IP_NF_ARPTABLES=y
CONFIG_IP_NF_ARPFILTER=y
CONFIG_BRIDGE=y
```

IP QUEUE

ip_queue needs to be installed as a kernel module for snort_inline to run.

For the life of me I couldn't find the kernel option to enable the module and had to compile it manually.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
# cd /usr/src/linux-2.4.21/net/ipv4/netfilter
# gcc -D__KERNEL__ -I/usr/src/linux-2.4.21/include -Wall -Wstrict-
prototypes -Wno-trigraphs -O2 -fno-strict-aliasing -fno-common -fomit-
frame-pointer -pipe -mpreferred-stack-boundary=2 -march=i686 -DMODULE -
DMODVERSIONS -include /usr/src/linux-2.4.21/include/linux/modversions.h
-nostdinc -iwithprefix include -DKBUILD_BASENAME=ip_queue  -c -o
ip_queue.o ip_queue.c
```

Once the ip_queue source is compiled the module could now be added using the following commands.

```
# cp ip_queue.o /lib/modules/2.4.21/kernel/net/ipv4/netfilter
# cd /lib/modules/2.4.21/kernel/net/ipv4/netfilter
# insmod ip_queue.o
```

5.3. Bridging

Download and install bridge utilities
wget http://bridge.sourceforge.net/bridge-utils/bridge-utils-0.9.6.tar.gz

```
./configure
make
make install
```

After installation the path is in /usr/local/sbin/brctl

brgup.sh script for turning up bridged interface. The following is incorporated into the rc.firewall script. See IPTABLES section for more information.

```
#!/bin/sh
ifconfig eth0 0.0.0.0 down
ifconfig eth1 0.0.0.0 down
brctl addbr br0
brctl stp br0 off
brctl addif br0 eth0
brctl addif br0 eth1
ifconfig br0 0.0.0.0 up
ifconfig eth0 up
ifconfig eth1 up
ifconfig eth2 10.1.1.1/24 up
```

Note: The br0 interface does not have an IP address that can be reached. It is important for the honeywall to have a third interface (eth2) that is protected on an internal network segment for management purposes.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

5.4. Iptables

Upgrading to 1.2.8 is recommended.

upgrade/install

```
wget http://www.iptables.org/files/iptables-1.2.8.tar.bz2
bzcat iptables-1.2.8.tar.bz2 |tar xvf -
```

verify

```
md5sum iptables-1.2.8.tar.bz2 | diff md -
md5 sum posted on website
```

Install

```
make KERNEL_DIR=/usr/src/linux-2.4.21
make install KERNEL_DIR=/usr/src/linux-2.4.21
```

rc.firewall

Honeynet was good enough to create a firewall script that helped automate the following tasks:

- Bringing up the Bridged interfaces
- Enabling Management Access
- Enable IP Queuing rules for Snort_inline to access packets
- Enable Rate Limiting on Outbound Packets for DoS prevention
- Enable Rate Limiting on Logging

Verifying Byte Counters

```
iptables -x -v --line-numbers -L FORWARD
```

Test IP tables with Bridging:

```
Iptables -P FORWARD DENY
```

Ping from internet client to x.x.x.5 (honey pot). Result should be rejected.

Rate Limit Test:

Reference http://iptables-tutorial.frozentux.net/iptables-tutorial.html#LIMIT-MATCHTXT

```
iptables -A FORWARD -p icmp --icmp-type echo-request -i eth0 -m limit -
-limit <RATE>/<SCALE> --limit-burst <RATE> -j ACCEPT
iptables -A FORWARD -j DROP
```

RATE = 10
SCALE=minute

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

With the above example an ICMP packet will get accepted 10 per minute or 1 every 6 seconds. You can test this with a simple ping. Since the default ping request is every second you can look at the sequence or time stamp to verify.

```
nettool# ping honeypot
64 bytes from honeypot: icmp_seq=6  ttl=64 time=0.311 ms
64 bytes from honeypot: icmp_seq=12 ttl=64 time=0.316 ms
64 bytes from honeypot: icmp_seq=18 ttl=64 time=0.382 ms
```

IPTABELS - syslog Note:
In order to not have IPTABLES log to the console but still log to /var/log/messages the klogd options need to be changed. The easy way to do this was edit the /etc/sysconfig/syslog options file that is read by the /etc/init.d/syslog startup script. The "-c 3" option is added to allow kernel.errs and higher to log to console only and let the kernel.warnings (which is what IPTABLES logs to) log to /var/log/messages.

/etc/sysconfig/syslog

```
SYSLOGD_OPTIONS="-m 0"
KLOGD_OPTIONS="-x -c 3"
```

5.5. Snort Inline

Snort Inline is a modified version of SNORT that is used to work with IPTABLES in order to create a Network Intrusion Prevention System. Unlike a NID (Network Intrusion Detection) system a NIP system proactively filters harmful traffic based on the existing rules. The Honeywall will be a sort of NIP system but instead of protecting the internal host (honeypot) we want to protect the rest of the world once the honeypot has become compromised.

In order for snort_inline to work properly, you must download and compile the iptables code to include "make install-devel" (www.iptables.org). This will install the libipq library that allows snort_inline to interface with iptables. Also, you must build and install LibNet, which is available from www.packetfactory.net.

```
wget Libnet from http://www.packetfactory.net/libnet/dist/libnet.tar.gz
tar -zxvf libnet.tar.gz
./configure
make
make install
```

```
wget snort_inline-2.0.0-1.tar.gz from http://snort-inline.sourceforge.net
tar -zxvf snort_inline-2.0.0.1.tar.gz
./configure --enable-inline
make
make install
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

test:

```
snort_inline -i eth1 -v -l /var/log/snort
```

or

```
snort_inline -i eth1 -d -l /var/log/snort -b -A full -s
```

Want to monitor internal interface “eth1” to focus on suspicious traffic.
-b => binary mode (can read with snort –r or tcpdump –r)
-A => Alerts with
-s => sends to syslog (/var/log/messages on Linux)

snort_inline.conf
copy snort_inline.conf example file to /etc/snort_inline/

classification.conf
copy classification.conf example to /etc/snort_inline/drop_rules

reference.conf
copy reference.conf example to /etc/snort_inline_drop_rules

download rules
wget http://www.snort.org/dl/rules/snortrules-stable.tar.gz

unpack to /etc/snort

```
tar -zxvf snortrules-stable.tar.gz -C /etc/snort
```

start snort_inline

```
snort_inline -i eth1 -d -Q -l /var/log/snort -b -c /etc/snort_inline/snort_inline.conf -A
full -s
```

-Q is the queing option that allows snort_inline to communicate with IPTABLES
The standard SNORT rules have been created with the intention to protect an internal system. The rules need to be reversed in order to protect the rest of the world from the honeypot. This can be accomplished by using the script (convert.sh) that was provided by the snort_inline distribution.

Copy convert.sh to /etc/snort/rules and execute

Note: The rc.firewall script supplied by snort_inline assumes that the firewall kernel options were installed as kernel modules. If you compile the firewall options directly into the kernel you need to comment out the modprobe and insmod statements in the script before executing it.

5.6. Swatch

Swatch looks for user defined strings across all system logs and alerts them to the admin via email or output to another file. Swatch is very useful for centrally defining what system events need to alert the system administrator.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Download swatch 3.0.8
http://swatch.sourceforge.net/

Decompress / Untar / Compile

```
tar -zxvf
perl Makefile.PL
make
make test
make install
make realclean
```

example /etc/swatch.conf file

```
watchfor /snort/
        echo = normal
        mail = admin@company.com, subject= Snort!
```

start swatch

```
swatch --config-file /etc/swatch.conf
```

Now any snort or snort_inline events will be emailed to admin@company.com with subject "Snort!".

6. Conclusion

Hopefully this document has provided a high level understanding of what a honeywall is and how it functions in a second generation honeynet honeypot. The instructions for installing and configuring the honeywall components will hopefully be a helpful guideline. Similar installation procedures will try to be maintained on http://www.honeywall.org to help honeywall newcomers find answers around configuration and installation issues.

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Appendix – PXE & Kickstart Info

The following information is a brief overview of the configuration files needed to help automate a lot of the installation/configuration of honeywalls through PXE, kickstart, and rpm packages. For more information please refer to http://www.honeywall.org

There are a few components to automating a honeywall system.

- DHCP Server
- TFTP Server
- PXE Boot IMAGE
- Linux Kickstart
- NFS Server

Basically Once these pieces are configured the process flow will look like the following

1.) New Server Boot from Network (PXE compliant) DHCP Request
2.) DHCP Server hears request and returns client IP and PXE boot image location
3.) New Server loads PXE boot image via TFTP
4.) PXE boot image configuration file points at linux image and Kickstart configurations
5.) New Server loads linux image and reads kickstart configuration file via NFS
6.) New Server installs Linux and honeywall specific packages
7.) New Server reboots and new Honeywall system is Born!

Kickstart Configuration File - ks.cfg

```
install
text
nfs --server=172.31.0.5 --dir=/usr/src/RH9/temp
lang en_US
langsupport --default en_US.iso885915 en_US.iso885915
keyboard us
mouse none
network --bootproto=dhcp --device=eth1
rootpw --iscrypted $1$TPsdfeeOp$a8H6XBCb7.8CbrY1HC9iF.
authconfig --enableshadow --enablemd5
timezone America/Tijuana
bootloader
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
zerombr yes
clearpart --all --drives=hda
part / --fstype ext2 --size=2100 --ondisk=hda
part swap --size=128 --grow --maxsize=256 --ondisk=hda

# check out packages list RedHat/base/comps.xml
%packages
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
binutils
cpp
glibc-devel
libgcc
gcc
ssh
libpcap
ncftp

%post
rpm -ivh ftp://linux.server.com/RPM/linux-2.4.21-1.rpm
rpm -ivh ftp://linux.server.com/RPM/snort-2.0.0-1.rpm
rpm -ivh ftp://linux.server.com/RPM/snort_inline-2.0.0-1.rpm
```

DHCP Server Configuration - /etc/dhcpd.conf

```
allow booting;
allow bootp;

subnet 172.31.0.0 netmask 255.255.255.0 {
option domain-name-servers 10.0.0.10, 10.0.0.6;
option routers 172.31.0.1;

group {
          next-server 172.31.0.5;
          filename "pxelinux.0";

        host red-client1 {
           hardware ethernet 00:D0:B7:0E:13:14;
           fixed-address 172.31.0.7;
        }
}
}
```

start DHCP Server & enable startup script

```
/etc/init.d/dhcpd start
chkconfig –level 345 dhcpd on
```

PXE Configuration (Using pxelinux)

- copy the pxelinux.0 image from the syslinux distribution to /tftpboot/
- create /tftpboot/pxelinux.cfg
- create a filed called default (or system specific by mac address)
- copy RedHat 9 pxeboot images from distribution CD to tftpboot

kernel image

```
cp /mnt/cdrom/images/pxeboot/vmlinuz /tftpboot/linux.1
```

ramdisk

```
cp /mnt/cdrom/images/pxeboot/initrd /tftpboot/linux.2
```

/tftpboot/pxelinux.cfg/default

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
DEFAULT Linux
DISPLAY hello.msg
PROMPT 1
LABEL Linux
  KERNEL pxelinux.1
  APPEND vga=788 initrd=pxelinux.2 ks=nfs:172.31.0.5:/kickstart/ks.cfg
```

TFTP Server

- Install tftp (rpm)
- Enable tftp server in xinet.d/tftp
- Restart xinet

NFS Server

Install NFS Server (rpm)
Configure /etc/export for kickstart and Linux Distribution locations

/etc/exports

```
/usr/src/RH9            *(ro,no_root_squash)
/kickstart              *(ro,no_root_squash)
```

start NFS

```
/etc/init.d/nfs start
```

enable export

```
exportfs -av
```

RPM packages

To speed up the honeywall install process, RPM *.spec files could be created for snort_inline and the linux 2.4.21 pre-compiled (bridging/queueing) kernel in order to include in the kickstart configuration process. For more information see http://www.honeywall.org.

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Bibliography

"Know Your Enemy: GenII Honeynets" The Honeynet Project,27 June 2003, URL: http://www.honeynet.org/papers/gen2/

"Know Your Enemy: Honeynets" The Honeynet Project, 18 January 2003, URL: http://www.honeynet.org/papers/honeynet/

"The Michigan Penal Code (Excerpt)", 25 August 2003, URL: http://www.michiganlegislature.org/printDocument.asp?objName=mcl-750-540c&version=txt

"Honeyd – Network Rhapsody for You", Center for Information Technology Integration, 1 August 2003, URL: http://www.citi.umich.edu/u/provos/honeyd/

Lance Spitzner "Learning with honeyd" Security Focus, 20 January, 2003 URL: http://www.securityfocus.com/infocus/1659

Lance Spitzner "Honeypots, Are They Legal?" Security Focus, 20 January, 2003 URL: http://www.securityfocus.com/infocus/1703.

Lance Spitzner "Honeypots: Definitions and Value of Honeypots" 29 May, 2003 URL: http://www.tracking-hackers.com/papers/honeypots.html

Tom Liston, "Tom Liston talks about LaBrea" , February 2003, URL: http://labrea.sourceforge.net/Intro-History.html

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.