Global Information Assurance Certification Paper

A Honeypot Based Worm Alerting System

Jeff Kloet

January 3, 2005

GIAC Security Essentials Certification
Practical Assignment
Version 1.4c
Option 1

Table of Contents

1.0 Abstract 1
2.0 Why use a Honeypot to detect Worms? 1
3.0 Honeypots and Honeyd 2
4.0 Implementation Architecture Overview 2
5.0 Implementation Requirements 3
5.1 Software for the Honeypot 3
5.2 Software for the Syslog Server 3
5.3 Hardware for the Honeypot 3
5.4 Hardware for the Syslog Server 4
5.5 The Local Router 4
6.0 Installing and Configuring the Honeypot Server 4
6.1 Installing libevent 4
6.2 Installing libdnet 4
6.3 Installing libpcap 5
6.4 Installing Honeyd 5
6.5 Configuring Honeyd 6
6.6 The Honeyd Configuration File 7
6.7 The Honeyd Start-up Script 7
7.0 Configuring the Honeypot Server's Local Syslog Facility 7
8.0 Configuring Routes to the Honeyd Address Space 8
9.0 Configuring the Syslog Server 8
10.0 Testing the Honeypot and Email Alerts 10
10.1 Starting and Verifying that Honeyd is Working 11
10.2 Verifying the Syslog Server and Email Alerting 11
11.0 Eliminating False Positive Alerts 12
12.0 Conclusion 13
13.0 Credits 13
References 14

1.0 Abstract

Network administrators are always looking for simple and effective ways to make their company networks more secure and resilient from worms and viruses. It is a great thing to proactively address problems before they become mountains.

One way to 'sweeten' a private network is with a honeypot that sends alerts when worm and virus infected machines are crawling the network; those nasties are looking for another vulnerable box to infect. This paper describes how to implement a Honeyd based honeypot to provide notification via email when worm and virus network traffic is detected.

2.0 Why use a Honeypot to detect Worms?

One very popular definition of a computer worm is:

> *"A computer program that can run independently, can propagate a complete working version of itself onto other hosts on a network, and may consume computer resources destructively."*[1]

The very nature of a worm is to self propagate. The first order of business in propagation then is to find another target to infect. The most successful way to find targets is to scan the network for other systems that are also vulnerable to the worms exploit.

Following are three IP address worm scanning models[2] that a worm could use in the process of looking for other vulnerable systems:

1. **Random** – Target addresses are selected randomly.
2. **Targeted** – Target addresses are gleaned/derived from local system resources (host file, arp tables, DNS record lookups, etc).
3. **Localized** – Target addresses are the local subnet and those logically adjacent.

It is the localized scanning strategy that one can take advantage of in detecting the self propagating nature of a worm infected machine.

If there is a honeypot listening for network connections where there should not be any connections, one is able to alert oneself to this suspicious network activity.

---

[1] Cisco Systems, Inc. Documentation. 15 Nov 2004
<http://www.cisco.com/univercd/cc/td/doc/cisintwk/ita/w12.htm>
[2] Chen, Zesheng, Lixin Gao and Kevin Kwiat. Modeling the Spread of Active Worms. 2003
<http://www.ieee-infocom.org/2003/papers/46_03.PDF>

3.0 Honeypots and Honeyd

In the broad sense, a honeypot is a host or network that is exposed to a public network for the purpose of monitoring and collecting information about attacks. Typically, the honeypot host/network is purposely configured with one or more vulnerabilities which act as a lure for malicious activity.

Honeyd[3] is a simple yet powerful honeypot that has been developed by Niels Provos. It is an OpenSource project that is designed to run on UNIX systems.

Honeyd falls into a category of honeypot defined as 'low-interaction'. A low-interaction honeypot mimics components of a host's operating system. An attacker does not interact with the real services but rather with crafted simulations. The low-interaction honeypot may, for instance, simulate an SMTP service which is poorly configured as an open relay. It may mimic a TELNET service with a known vulnerability which could be used to compromise the host. It is important to understand that the actual low-interaction honeypot host system is not exposed to attack; it is only mimicking vulnerable services.

There is also a 'high-interaction' honeypot category. This type of honeypot is a real host system where nothing is simulated. Everything about what an attacker is doing on the system can be monitored for analysis.

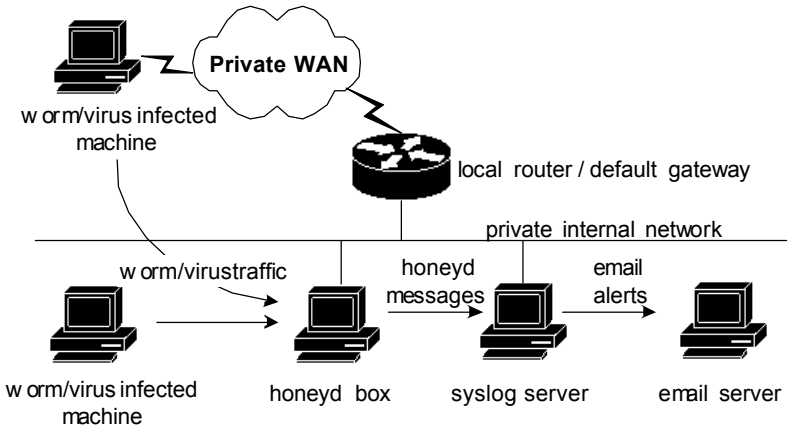
There are advantages and disadvantages with both low and high interaction honeypots. High-interaction honeypots are excellent for collecting information about an attacker's method however they are generally more complex to deploy and manage. Low-interaction honeypots are simple to deploy however they are limited in their scope of data collection.

Honeyd suits the purpose of a worm detection system when one simply wants to be alerted to any traffic that hits the honeypot.

4.0 Implementation Architecture Overview

A honeypot system that provides alerts of worm and virus activity requires a honeypot server and a means to deliver events to an administrators mailbox. A honeypot server is configured to forward all connection attempts to a syslog server. The syslog server in turn is configured to selectively email relevant events to the desired email account.

[3] Provos, Niels. Developments of the Honeyd Virtual Honeypot. 10 Oct 2004 < http://www.honeyd.org/>



5.0 Implementation Requirements

5.1 Software for the Honeypot

Honeyd will compile and run on BSD systems, GNU/Linux and Solaris. I chose to use RedHat Linux 9.0 for my honeypot server. Following is the software that is necessary to install and support Honeyd on a system:

Software	Download Site
honeyd-0.8b.tar.gz	http://www.honeyd.org/
libevent-0.9.tar.gz	http://www.monkey.org/~provos/libevent/
libdnet-1.8.tar.gz	http://libdnet.sourceforge.net/
libpcap-0.8.3.tar.gz	http://www.tcpdump.org/

The versions of software releases listed above are the ones that I used on RedHat 9. The versions that one requires for their system may differ.

5.2 Software for the Syslog Server

A syslog server is required that can filter on the message text and can forward received Syslog messages via email (SMTP). If one does not have a Syslog server then the licensed version of 'Kiwi Syslog Daemon' is an excellent choice. It is highly configurable and the service edition will run on Windows NT/2K/XP.

Kiwi Syslog Daemon	http://www.kiwisyslog.com/

5.3 Hardware for the Honeypot

A system that supports the version of BSD/Linux/Solaris is the hardware that

one should use. Anything manufactured within the last several years will likely be supported.

5.4 Hardware for the Syslog Server

The Kiwi Syslog Daemon requires a box that supports Windows NT/2K/XP. Anything manufactured within the last several years will be supported.

5.5 The Local Router

The local router/default gateway for the network that the Honeypot server is connected to needs to be able to be configured with static routes. This should not be a problem since even home DSL/Cable routers can support this. A desirable router feature, though not critical, is the ability to configure a static route to a null interface.

6.0 Installing and Configuring the Honeypot Server

6.1 Installing libevent

The software binaries for libevent[4] should be acquired and the following commands to install the software should be used:

```
# cp libevent-0.9.tar.gz /usr/src/redhat/SOURCES
# cd /usr/src/redhat/SOURCES
# tar -zxvf libevent-0.9.tar.gz
# cd /usr/src/redhat/SOURCES/libevent-0.9
# ./configure
# make
# make install
```

6.2 Installing libdnet

The software binaries for libdnet[5] should be acquired and the following commands to install the software should be used:

```
# cp libdnet-1.8.tar.gz /usr/src/redhat/SOURCES
# cd /usr/src/redhat/SOURCES
# tar -zxvf libdnet-1.8.tar.gz
# cd /usr/src/redhat/SOURCES/libdnet-1.8
# ./configure
```

[4] Provos, Niels. libevent - an event notification library. 7 Apr 2004 <http://www.monkey.org/~provos/libevent/>
[5] Song, Dug. libdnet. 22 May 2004 <http://sourceforge.net/projects/libdnet/>

```
# make
# make install
```

6.3 Installing libpcap

The software binaries for libpcap[6] should be acquired and the following commands to install the software should be used:

```
# cp libpcap-0.8.3.tar.gz /usr/src/redhat/SOURCES
# cd /usr/src/redhat/SOURCES
# tar -zxvf libpcap-0.8.3.tar.gz
# cd /usr/src/redhat/SOURCES/libpcap-0.8.3
# ./configure
# make
# make install
```

6.4 Installing Honeyd

The software binaries for Honeyd[7] should be acquired and the following commands and procedures to install the software should be used:

```
# cp honeyd-0.8b.tar.gz /usr/src/redhat/SOURCES
# cd /usr/src/redhat/SOURCES
# tar -zxvf honeyd-0.8b.tar.gz
# cd /usr/src/redhat/SOURCES/honeyd-0.8b
```

As per the 'Honeyd FAQ'[8], one[9] may have to run:

```
# ldconfig –v[10]
```

and I had to edit the '/etc/ld.so.conf' file to add '/usr/local/lib'.

Honeyd can be run in normal and in debug mode. A syslog mask determines what information gets logged to the local syslog facility. By default, Honeyd does not log connection attempts to syslog when it is run in normal mode. However it is important for Honeyd to log connection attempts because one can use the local syslog facility to push them to an external syslog server for further processing. Honeyd's logging behaviour is changed by modifying the mask in the source code. This should be done before Honeyd is compiled and installed.

To make Honeyd log information in normal mode like it does in debug mode, edit "/usr/src/redhat/SOURCES/honeyd-0.8b/honeyd.c", look for:

---

[6] Lawrence Berkeley National Laboratory. tcpdump/libpcap. 22 June 2004 <http://www.tcpdump.org/>
[7] Provis, Niels. Honeyd Downloads and Releases 20 Apr 2004 <http://www.honeyd.org/release.php>
[8] Provos, Niels. Honeyd Frequently Asked Questions. 20 Dec 2004 <http://honeyd.org/faq.php>
[9] I did with RedHat 9.
[10] The Honeyd FAQ says to use the –m parameter however this is not an option in RedHat 9.

```
setlogmask(LOG_UPTO(LOG_INFO))
```

… and change it to:

```
setlogmask(LOG_UPTO(LOG_DEBUG))
```

The Honeyd software can then be compiled and installed using the following commands:

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

6.5 Configuring Honeyd

Honeyd basically emulates host computers on a network. Honeyd can be configured to simply open a connection in response to a request or it can be configured to run a script that might make it appear to be a valid service on a real host computer. One instucts Honeyd what IP address or addresses to listen on and associates those addresses with a computer personality/profile.

As an example configuration, Honeyd can be instructed to listen on unused address space in a network and can be made to respond openly to any connection requests. If one is using 192.168.0.0 addressing for their network and has assigned subnets starting with 192.168.20.0/24 and up, then I suggest that Honeyd listen on a subnet lower than what is currently being used, for example 192.168.10.0/24. If one is using 10.0.0.0 addressing and they are using 10.20.0.0/16 and above, then perhaps Honeyd could be instructed to listen on network 10.1.0.0/16.

The reason the honeypot should listen on this unused space is because normal traffic within the network will not be destined to this addressing. It is highly probable to be worm or virus traffic when traffic destined to that addressing does occur. At the very least it is suspicious and worth further investigation.

One instructs Honeyd what networks to listen on when it is started. If one wants Honeyd to listen and respond on the example networks described above, then Honeyd would be started with the following options:

```
# honeyd  -f ./usr/local/share/honeyd/honeyd.conf
192.168.10.0/24 10.1.0.0/16
```

The –f option tells Honeyd where to read its configuration file from.

6.6 The Honeyd Configuration File

A number of sample configuration files can be found in "/usr/local/share/honeyd". For this example implementation, a simple configuration is used that tells Honeyd to open a connection for any incoming TCP, UDP and ICMP traffic.

One of the sample configuration files (config.honeyd for instance) may be modified or a new file created and the following settings entered:

```
create default
set default personality "Microsoft Windows XP Professional SP1"
set default default tcp action open
set default default udp action open
set default default icmp action open
```

Save the file as "/usr/local/share/honeyd/honeyd.conf".

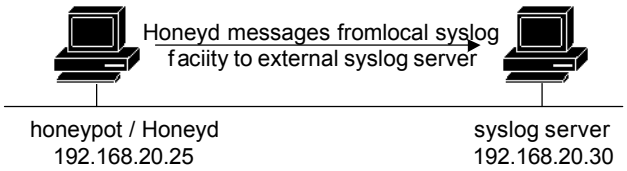
6.7 The Honeyd Start-up Script

Honeyd can be started from the command line however it is easier to create a script that contains the start-up options. The script will contain:

```
# Honeyd start-up script

/usr/local/bin/honeyd  \
      -f ./usr/local/share/honeyd/honeyd.conf \
      192.168.10.0/24 10.1.0.0/16
```

7.0 Configuring the Honeypot Server's Local Syslog Facility

When one changes the syslog mask for Honeyd in normal mode, it logs all events and connection requests to the local syslog facility. The Honeyd messages are automatically logged into /var/log/messages, however we want to also have them sent to the external syslog server where they will be processed to generate email alerts.



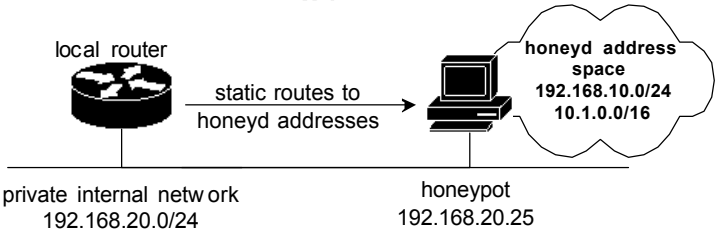
Given that the Syslog server's TCP/IP address is 192.168.20.30, edit the /etc/syslog.conf file on the Honeyd box and add/insert the 'daemon.*' line listed below.

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                          /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none
/var/log/messages

# For Honeyd, forward daemon messages to remote syslog server
daemon.*                                        @192.168.20.30

# etc…
```

This will instruct the local syslog service to forward all daemon generated messages to the remote syslog server. One can also put the DNS name in place of the TCP/IP address.

8.0 Configuring Routes to the Honeyd Address Space

A network needs to know where to send the traffic that is trying to hit the Honeyd network address space. Honeyd supports a number of ways to do this however the easiest is to create a static route that points to the actual network address of the honeypot.



Given that the honeypot IP address is 192.168.20.25, static routes like the following (Cisco IOS example) should be added:

```
ip route 192.168.10.0 255.255.255.0  192.168.20.25
ip route 10.1.0.0 255.255.0 192.168.20.25
```

9.0 Configuring the Syslog Server

With the honeypot server configured to forward all 'daemon.*' messages, the external syslog server will need to filter through them and make a decision about which ones to generate email alerts on.

One may use the licensed version of Kiwi Syslog Daemon[11] to process the

[11] Kiwi Enterprises. Kiwi Syslog Daemon. 2004 <http://www.kiwisyslog.com/>

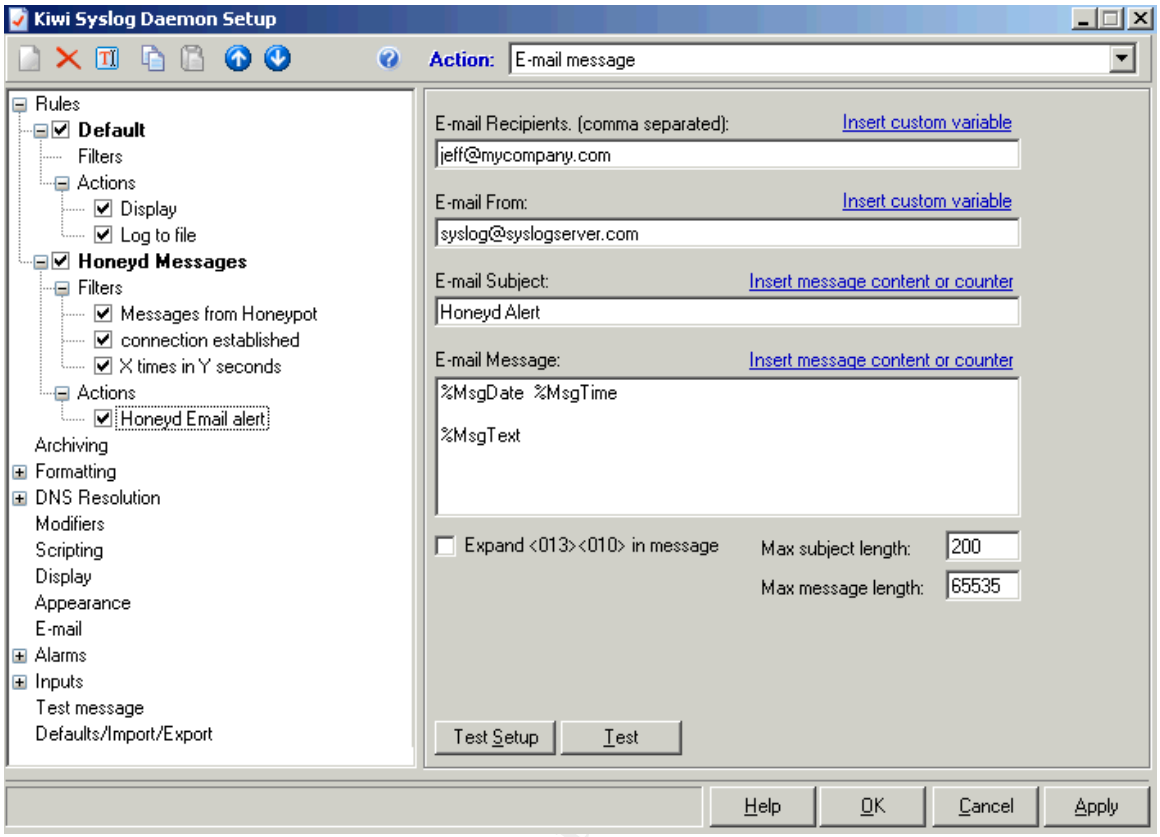
messages sent from the Honeyd server. The application should be installed on a Windows NT/W2K/XP machine and configured as a service so that it is always up and running.

In Kiwi's setup screen, a new Syslog rule should be added and called 'Honeyd Messages'. The rule should be built with the following filters and action:

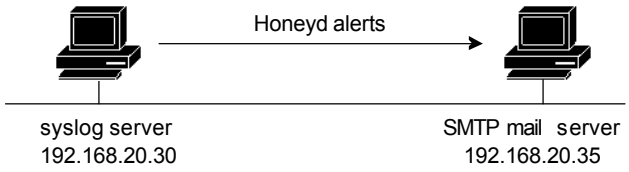
Rule	Type	Name	Field-Type	Conditions
Honeyd Messages	Filter	Messages from Honeypot	IP address-IP range	start 192.168.50.25, end 192.168.50.25
Honeyd Messages	Filter	Connection established	Message text-RegExp	Include "honeyd" AND "connection established"
Honeyd Messages	Filter	X times in Y seconds	Flags/Counters-Threshold	true if event occurs 5 times in 60 seconds
Honeyd Messages	Action	Honeyd Email alert	E-mail message	

The 'X times in Y seconds' threshold filter is useful in minimizing false positives. There likely will be non worm/virus traffic that hits the Honeyd address space however it probably will not have the repetitive nature of a worm/virus. If one says that the message must occur a number of times within a time period, it will prevent the syslog server from notifying the administrator of traffic that is not of interest. As an example, one may choose 5 occurrences in 60 seconds. This may need to be tweaked to achieve the desired goal of no unnecessary alerts.

The email action/alert should be configured with the desired 'E-mail Recipients' and appropriate settings for the 'Email From' and 'E-mail Subject' fields. While the information in the 'E-mail Message' could be as detailed as desired, it may be appropriate to simply include the date, time and actual message from Honeyd.



The Kiwi Syslog Daemon needs to know where to relay the email alerts.



The SMTP server information must be entered in the 'Email' section of the Setup screen.

One should verify that the Syslog server can send an email message to their mailbox. With Kiwi, a test message can be sent in the 'E-mail' setup screen or from the 'Honeyd Email Alert' action created for the 'Honeyd Messages' rule.

10.0 Testing the Honeypot and Email Alerts

It is important to test the configuration in order to verify operation. Before starting, the 'X times in Y seconds' threshold filter on the Syslog server 'Honeyd Messages' rule should be disabled so that a single connection request will

generate an alert.

10.1 Starting and Verifying that Honeyd is Working

The Honeyd start-up script that was created earlier can be used to start Honeyd:

```
# cd /usr/local/share/honeyd
# ./start-honeyd
  *** You should see Honeyd start with the following output ***
Honeyd V0.8b Copyright (c) 2002-2004 Niels Provos
honeyd[11770]: started with -f
/usr/local/share/honeyd/honeyd.conf 192.168.10.0/24 10.1.0.0/16
honeyd[11770]: listening promiscuously on eth0: (arp or ip
proto 47 or (ip and (net 192.168.10.0/24 or net 10.1.0.0/16)))
and not ether src 00:b0:d0:82:1b:3d
Honeyd starting as background process
#
```

The /var/log/messages file should be monitored with the following command:

```
# tail -f /var/log/messages
```

From another machine, some traffic should be created that hits Honeyd's address space. For example:

```
C:\>ping 192.168.10.10

Pinging 192.168.10.10 with 32 bytes of data:

Reply from 192.168.10.10: bytes=32 time=3ms TTL=128
Reply from 192.168.10.10: .... etc

C:\>telnet 192.168.10.15
```

One should then see Honeyd logging entries to '/var/log/messages' that are similar to the following:

```
Dec 29 11:40:59 localhost honeyd[12919]: Sending ICMP Echo
Reply: 192.168.10.10 -> 192.168.1.101
Dec 29 11:41:02 localhost last message repeated 3 times
Dec 29 11:41:10 localhost honeyd[12919]: Connection request:
tcp (192.168.1.101:1315 - 192.168.10.15:23)
Dec 29 11:41:10 localhost honeyd[12919]: Connection
established: tcp (192.168.1.101:1315 - 192.168.10.15:23)
```

These same messages will be forwarded to the external syslog server because the 'daemon.*' line was added to the syslog.conf file.

10.2 Verifying the Syslog Server and Email Alerting

The same messages that Honeyd logs to the local syslog facility will be sent to the external syslog server. If using the Kiwi Syslog Service, these messages should appear in the default Manager Display window.

Date	Time	Priority	Hostname	Message
12-29-2004	11:41:11	Daemon .Debug	192.168.1.10 3	honeyd[12919]: Connection established: tcp (192.168.1.101:1315 - 192.168.10.15:23)
12-29-2004	11:41:11	Daemon .Debug	192.168.1.10 3	honeyd[12919]: Connection request: tcp (192.168.1.101:1315 - 192.168.10.15:23)
12-29-2004	11:41:11	Daemon .Debug	192.168.1.10 3	last message repeated 3 times
12-29-2004	11:40:59	Daemon .Debug	192.168.1.10 3	honeyd[12919]: Sending ICMP Echo Reply: 192.168.10.10 -> 192.168.1.101

The last Honeyd message that contains the string 'Connection established' will satisfy the Syslog server's 'Honeyd Messages' filter and will trigger an email alert. A message similar to the following should appear in the administrators mailbox:

```
From: syslog@syslogserver.com
Sent: December 29, 2004 11:41 AM
To: Jeff K
Subject: Honeyd Alert

2004-12-29  11:41:11

honeyd[12919]: Connection established: tcp (192.168.1.101:1315 -
192.168.10.15:23)
```

If the alerts are not reaching the users mailbox, it should be confirmed that the (Kiwi) Syslog server itself can send a test message through.

11.0 Eliminating False Positive Alerts

Once the honeypot is up and running for a while, the user may find that alerts will be generated from non-worm/virus traffic that hits the Honeyd address space. The reasons for this traffic will vary and it should be determined why it is occurring before it can be discounted it as a false positive.

False positive alerts are annoying and counter productive. Following are some ways to deal with false positive traffic:

1. Set and/or tune a threshold filter on the Syslog 'Honeyd Messages' rule. Adding a threshold filter to the Syslog server honeypot rule was described

in section 9.0, 'Configuring the Syslog Server'.

2. If the false positive traffic is of a re-occurring nature (in type and destination), create a static route on the local default router that sinks the traffic to a null interface or eliminate the target address(es) from Honeyd's address space. For example, if Honeyd is listening on the network 192.168.10.0/24 and non-worm traffic regularly hits the address 192.168.10.22, one can filter the false positive traffic by:

   a) Creating a static route to a null interface by entering the following command on a router (Cisco IOS example):

   ```
   ip route 192.168.10.22 255.255.255.255 Null0
   ```

   b) Starting Honeyd by specifying a range of addresses that does not include 192.168.10.22 (the IP address(es) or network that Honeyd listens on is specified using CIDR[12] notation or address range):

   ```
   honeyd  -f honeyd.conf 192.168.10.1-21 192.168.10.23-254
   ```

12.0 Conclusion

This honeypot implementation is not the end all to worm and virus traffic detection however it is an excellent tool in ones arsenal. What makes this Honeyd configuration so sweet is that, once deployed, it will be next to zero maintenance and one will only be notified when there is a true threat.

13.0 Credits

I would like to thank Niels Provos and the 'Honeypots' mailing list (honeypots@securityfocus.com) for helping with Honeyd-0.8b technical issues.

[12] See http://infocenter.guardiandigital.com/manuals/IDDS/node9.html for an explanation of CIDR notation.

References

Chen, Zesheng, Lixin Gao and Kevin Kwiat. Modeling the Spread of Active Worms. 2003 <http://www.ieee-infocom.org/2003/papers/46_03.PDF>

Cisco Systems, Inc. Documentation. 15 Nov 2004 <http://www.cisco.com/univercd/cc/td/doc/cisintwk/ita/w12.htm>

Kiwi Enterprises. Kiwi Syslog Daemon. 24 August 2004 <http://www.kiwisyslog.com/whats_new_syslog.htm>

Lawrence Berkeley National Laboratory. tcpdump/libpcap. 22 June 2004 <http://www.tcpdump.org/>

Motlekar, Shaheem. Honeypots: Frequently Asked Questions. 25 Mar 2004 <http://www.tracking-hackers.com/misc/faq.html>

Oudot, Laurent. Fighting Internet Worms With Honeypots. 23 Oct 2003 <http://www.securityfocus.com/infocus/1740>

Provos, Niels. Developments of the Honeyd Virtual Honeypot. 10 Oct 2004 <http://www.honeyd.org/>

Provos, Niels. Honeyd Frequently Asked Questions. 20 Dec 2004 <http://honeyd.org/faq.php>

Provos, Niels. Honeyd - Network Rhapsody for You. 12 Aug 2004 <http://www.citi.umich.edu/u/provos/honeyd/>

Provos, Niels. libevent - an event notification library. 7 Apr 2004 <http://www.monkey.org/~provos/libevent/>

Red Hat, Inc.. Red Hat Linux 9 Manual Pages. 2004 <http://www.redhat.com>

Song, Dug. libdnet. 22 May 2004 <http://sourceforge.net/projects/libdnet/>

Spitzner, Lance. Honeypots. 29 May 2003 <http://www.tracking-hackers.com/papers/honeypots.html>

Staniford, Stuart, Vern Paxson and Nicholas Weaver. How to Own the Internet in Your Spare Time. 14 May 2002 <http://www.icir.org/vern/papers/cdc-usenix-sec02/>