HONEYD

(OPEN SOURCE HONEYPOT SOFTWARE)

Author:

Avinash Singh

Avinash Singh is a **Technical Evangelist** currently worksing at Appin Technology Lab, Noida.

Educational Qualification:

- B.Tech from Punjab Technical University

He holds the following certifications in the field of Ethical Hacking & Information Security Appin Certified Ethical Hacker (ACEH).
He has also done training in the field of Network Security, which includes IDS (Intrusion Detection Systems), Firewalls and Honeypot Technology.
He is also trained in other fields such as Linux, Microsoft Certified IT Professional (MCITP), Firewall and CISCO Certified Network Associate.

Got an article published in Pentest magazine and also published a **Research Paper** on VoIP Hacking published on packetstormsecurity.com

He is also the **admin** and lead **author** at **TechSpectrum.in**.

He has a total experience of around 2.5 year in the field of Training and Security, and has **successfully delivered more than 60 workshops and training programmes till now.**

Table of contents :

1. INTRODUCTION

Honeyd improves cyber security by providing mechanisms for threat detection and assessment. It also deters adversaries by hiding real systems in the middle of virtual systems. Honeyd is an application which enables the setup of multiple virtual honeypots on a single machine, each with different characteristics and services. Let's first have a look at the honeypot technology.

2. WHAT IS A HONEYPOT ??

A honeypot is a system which is acting as a potential target for an attacker. The system itself though isn't of much value to the operator as no valuable information or important services are located on that machine – it's the opposite. All services running on a honeypot aren't used in the productive environment. The services aren't promoted and so there shouldn't be any productive traffic going for these systems. Due to this fact, all traffic heading and reaching a honeypot is of potential value and should be analyzed. A honeypot doesn't need to deal with false positives like an intrusion detection system as there are simply no false positives – all traffic is suspicious as there shouldn't be any traffic because nobody knows of the system, no productive services are running and the system is not involved in "normal" activities.

2.1 TWO HONEYPOT CATEGORIES

Two categories of honeypots have evolved **– research and productive honeypots**.

Research honeypots are used primarily for research activities like detecting new kind of attacks, retrieving new hacker tools or to get a better knowledge about the attackers, their background, activities and goals. Research honeypots are valuable for developing new IDS signatures, analyze new attack tools or detect new ways of hidden communications or distributed denial of service (DDoS) tools. Research honeypots normally have great logging capabilities to log a hacker's activity once the attacks started or he gained root access.

The other category, the **productive honeypots**, is mostly used to distract an attacker from the real target. A honeypot is used as a bait to bind his attacking attempts as long as possible to the unproductive honeypot in order to gain time and protect the productive environment in the meantime. A productive honeypot is primarily not interested in gaining new knowledge about the blackhat community – its main interest is the protection of the real servers. Productive honeypots sometimes are also used to gather enough evidence for a successful prosecution of a hacker – But this application is still controversial and the legal side of such procedures is also not clear.

2.2 Level of Involvement

Besides the two usage categories of honeypots we already seen, there are also three different technical implementations of honeypots. The essential factor to distinguish here is the "level of involvement". A honeypot is acting like a "normal" server to the attacker – he offers certain services on different ports and could have certain vulnerabilities.

Depending on the usage of a honeypot, having some real services on that machine is not always desired or even needed. It could be enough to have a simple listener bound to a port which just writes all incoming packets to a file and never answers to the received request. For catching an infected Microsoft Internet Information Server this is enough, no real IIS is needed. On the other hand, to study a hacker's social network and ways of communicating it could be necessary to "offer a real shell" and allow the attacker to gain root privileges. Once a hacker is root on a system it could be very interesting to see what he's going to do and for what he does need his newly gained system. These different honeypots can be described with the level of involvement

• **Low involvement:** They are listening on a certain port for incoming connections. All packets are logged. No answer to the request is sent. Low involvement honeypots have no interaction with the attacker. No traffic is ever leaving the honeypot – It's a simple logging machine.

• **Mid involvement:** Mid involvement honeypots also listen on different ports. But in contradiction to low involvement they send information back to the attacker. A request is answered and the attacker has the possibility to issue commands. Normally, mid involvement honeypots don't use real daemons, instead scripts or small programs are used to imitate the behavior of a service. The provided functionality depends on the script – in most cases, the provided commands are very limited. The big advantage of using such scripts is their logging capabilities and the circumvention of possible vulnerabilities of real services.

• High involvement: High involvement honeypots are the most advanced honeypots. They use real daemons and provide the full set of functionality. An attacker can do whatever he could do to a productive system – no limitations in functionality, vulnerability or behavior. Unfortunately, logging all attempts with high details isn't always easy and the risk of a compromise is growing. Mostly, high involvement honeypots are used when a compromise of a system is desired.

3. HONEYD – A VIRTUAL HONEYPOT

Honeyd is a freely available framework for setting up virtual honeypots. With honeyd it is possible to setup honeypots with different personalities and services on one machine. Honeyd emulates the different operating system's IP stack and binds certain script to a desired port to emulate a specific service. Honeyd is able to fool network fingerprinting tools to think they are dealing with a real operating system ranging from a Windows NT to an AIX box. Even different router's IP stacks can be emulated. Honeyd relies on the nmap fingerprinting file which is used to characterize different kind of operating systems and their IP stacks. Before honeyd is inserting a packet into the IP stream, the personality of the packet is adjusted according to the desired operating system and the corresponding TCP/IP flags. With honeyd it is even possible to emulate complex network architectures and their characteristics. Virtual routing topologies can be defined including different brands of routers, the latency of a network connection as well as the packet loss. When using tools to map the network (like traceroute), the network traffic appears to follow the configured routers and network connections.

The setup of virtual machines is very easy. A configuration file is used to tell honeyd what kind of operating system is desired, how it does respond to closed ports and what kind of service is listening on which port. Honeyd is capable of binding a script to a network port. The script can be a standard shell script which simulates a certain service. Most scripts are built as state machines where a command triggers a certain response or advances to a new state with new possibilities. Scripts for the most popular well known services like SMTP, HTTP and telnet are available at several locations on the Internet.

3.1 CHOSEN SETUP

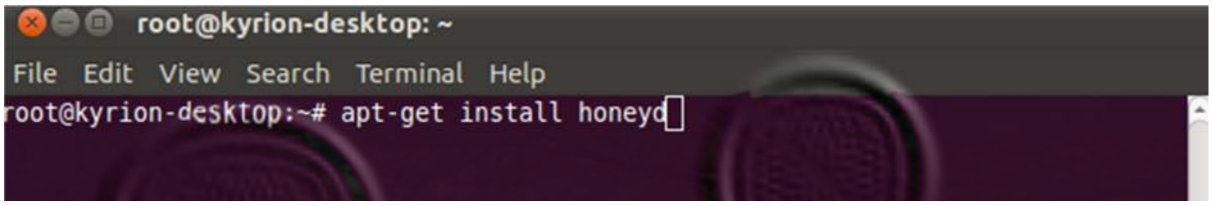
As honeyd runs on a Linux/UNIX/BSD so i chose the following setup :

Linux OS – Ubuntu 10.10 (Debian based) installed on a system with 1 GB of RAM and an Ethernet Card.

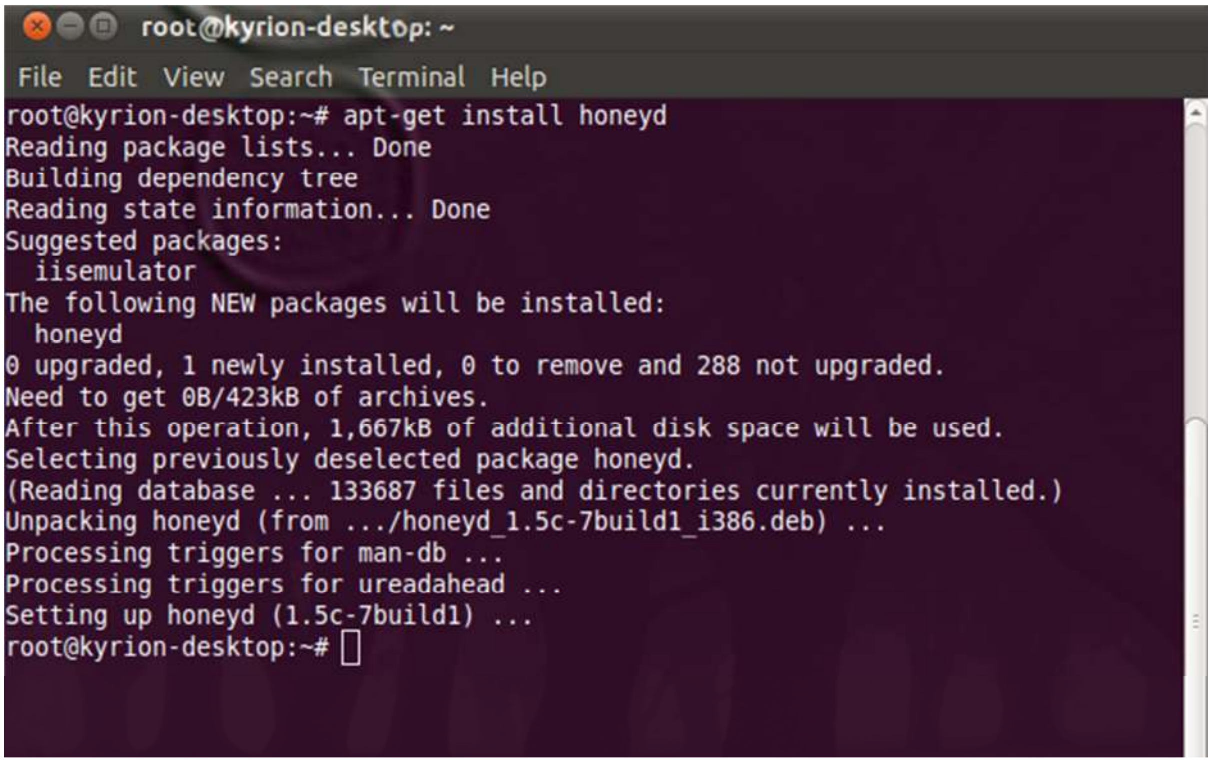
3.2 HONEYD INSTALLATION

In debian based linux versions honeyd can be installed by the following command:

apt-get install honeyd



```
root@kyrion-desktop:~# apt-get install honeyd
```

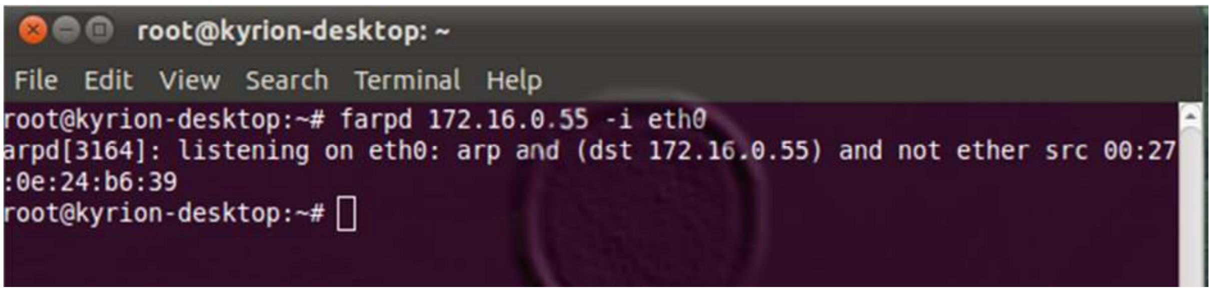


```
root@kyrion-desktop:~# apt-get install honeyd
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  iisemulator
The following NEW packages will be installed:
  honeyd
0 upgraded, 1 newly installed, 0 to remove and 288 not upgraded.
Need to get 0B/423kB of archives.
After this operation, 1,667kB of additional disk space will be used.
Selecting previously deselected package honeyd.
(Reading database ... 133687 files and directories currently installed.)
Unpacking honeyd (from .../honeyd_1.5c-7build1_i386.deb) ...
Processing triggers for man-db ...
Processing triggers for ureadahead ...
Setting up honeyd (1.5c-7build1) ...
root@kyrion-desktop:~# 
```

3.3 FARPD

The honeypot is going to run on a machine with a specific IP (eg 192.168.1.22) .Our honeypot (eg 192.168.1.50) would be visible from the Internet with a quantity of open ports. But the problem is that the router don't know our honeypot (192.168.1.50). To solve this we run in Honeyd computer:

farpd 192.168.1.50 –i eth0



```
root@kyrion-desktop:~# farpd 172.16.0.55 -i eth0
arpd[3164]: listening on eth0: arp and (dst 172.16.0.55) and not ether src 00:27
:0e:24:b6:39
root@kyrion-desktop:~# 
```

farpd is a program made by Niels Povos. With that program the computer with Honeyd (192.168.1.22) will send is MAC address when a ARP request is made to the network. This ARP request happens because the router don't know who is 192.168.1.50. After 192.168.1.22 sending his MAC address the router will send the package to Honeyd computer (192.168.1.22), and Honeyd program will take care of them, sending it to the virtual host.

3.4 CONFIGURATION SETTINGS

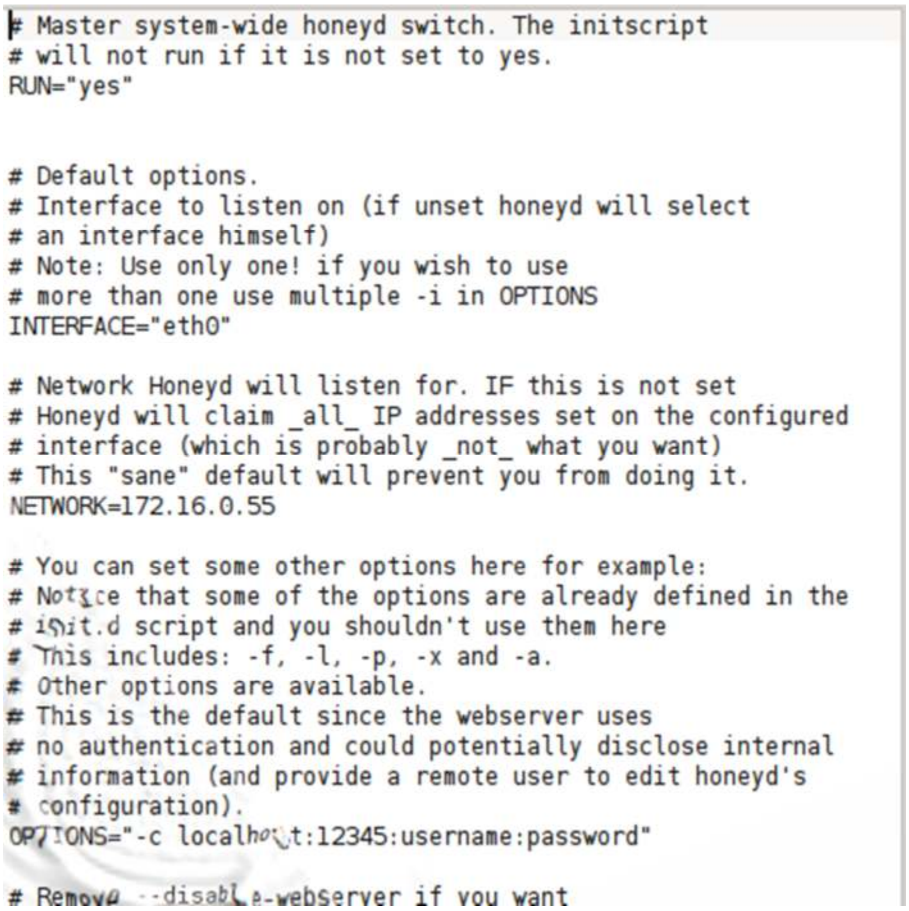
Now we have to configure how Honeyd will run, the file can be found in **/etc/default/honeyd**

RUN=”yes”

INTERFACE=”eth0”

NETWORK=192.168.1.50

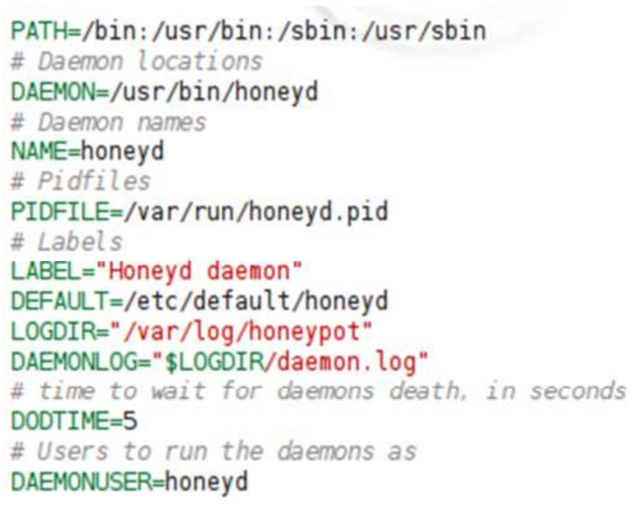
OPTIONS=”-c localhost:12345:username:password”



```
# Master system-wide honeyd switch. The initscript
# will not run if it is not set to yes.
RUN="yes"


# Default options.
# Interface to listen on (if unset honeyd will select
# an interface himself)
# Note: Use only one! if you wish to use
# more than one use multiple -i in OPTIONS
INTERFACE="eth0"

# Network Honeyd will listen for. IF this is not set
# Honeyd will claim _all_ IP addresses set on the configured
# interface (which is probably _not_ what you want)
# This "sane" default will prevent you from doing it.
NETWORK=172.16.0.55

# You can set some other options here for example:
# Notice that some of the options are already defined in the
# init.d script and you shouldn't use them here
# This includes: -f, -l, -p, -x and -a.
# Other options are available.
# This is the default since the webserver uses
# no authentication and could potentially disclose internal
# information (and provide a remote user to edit honeyd's
# configuration).
OPTIONS="-c localhost:12345:username:password"

# Remove --disable-webserver if you want
```

The -c flag will collect to us some statistics, that we will put in a pie chart further. This flag receives the hostname, the port, username and password to can access to the statistics.

You may want to configure also the **/etc/init.d/honeyd** file, here are the first lines of the file:

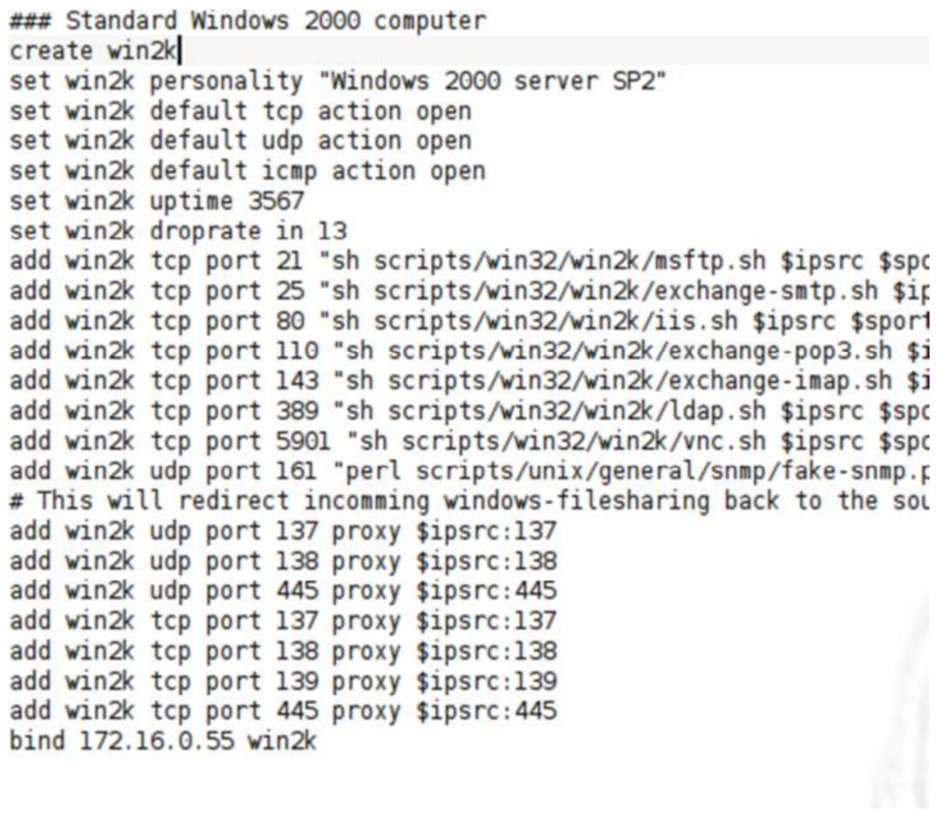


```
PATH=/bin:/usr/bin:/sbin:/usr/sbin
# Daemon locations
DAEMON=/usr/bin/honeyd
# Daemon names
NAME=honeyd
# Pidfiles
PIDFILE=/var/run/honeyd.pid
# Labels
LABEL="Honeyd daemon"
DEFAULT=/etc/default/honeyd
LOGDIR="/var/log/honeypot"
DAEMONLOG="$LOGDIR/daemon.log"
# time to wait for daemons death, in seconds
DODTIME=5
# Users to run the daemons as
DAEMONUSER=honeyd
```

Note that log files from Honeyd will be written in LOGDIR directory.

3.5 CONFIGURATION OF HONEYPOT

The following is a default configuration with some editing, a lot of these can be found on internet.



```
### Standard Windows 2000 computer
create win2k
set win2k personality "Windows 2000 server SP2"
set win2k default tcp action open
set win2k default udp action open
set win2k default icmp action open
set win2k uptime 3567
set win2k droprate in 13
add win2k tcp port 21 "sh scripts/win32/win2k/msftp.sh $ipsrc $spc
add win2k tcp port 25 "sh scripts/win32/win2k/exchange-smtp.sh $ip
add win2k tcp port 80 "sh scripts/win32/win2k/iis.sh $ipsrc $sport
add win2k tcp port 110 "sh scripts/win32/win2k/exchange-pop3.sh $i
add win2k tcp port 143 "sh scripts/win32/win2k/exchange-imap.sh $i
add win2k tcp port 389 "sh scripts/win32/win2k/ldap.sh $ipsrc $spc
add win2k tcp port 5901 "sh scripts/win32/win2k/vnc.sh $ipsrc $spc
add win2k udp port 161 "perl scripts/unix/general/snmp/fake-snmp.p
# This will redirect incomming windows-filesharing back to the sou
add win2k udp port 137 proxy $ipsrc:137
add win2k udp port 138 proxy $ipsrc:138
add win2k udp port 445 proxy $ipsrc:445
add win2k tcp port 137 proxy $ipsrc:137
add win2k tcp port 138 proxy $ipsrc:138
add win2k tcp port 139 proxy $ipsrc:139
add win2k tcp port 445 proxy $ipsrc:445
bind 172.16.0.55 win2k
```

Each system is first created with a **create command.**

The system then is further specified and configured with **add and set commands**.

With the set **personality command**, a personality is assigned to a created system. It is further possible to choose the default action for the supported network protocols like block, reset or open. If the default value is set to be open, all ports for the desired protocol are in a listening state. The value reset defines all ports to be closed while block is used to drop all packets for the designated protocol.

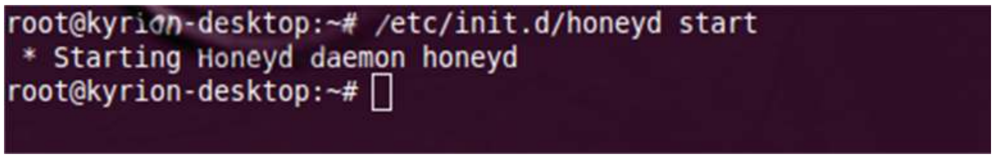
Adding services, therefore binding scripts to a certain port, is done by using the **add command.** Instead of binding a script to a port it is also possible to forward the traffic to another IP by using **the keyword proxy**.

Here I create a win2k operative system (Microsoft Windows 2000 with SP2) with a lot of open ports {23,21,25,80,110,143,389,5901,137,138,139}TCP and {161,137,138,445}UDP. These port's must be open in your router, and pointing to the honeypot – (eg 172.16.0.55).

3.6 STARTING THE HONEYPOT

And now the time to **start Honeyd,** the following command is used to start honeyd :

/etc/init.d/honeyd start



```
root@kyrion-desktop:~# /etc/init.d/honeyd start
 * Starting Honeyd daemon honeyd
root@kyrion-desktop:~# 
```

/etc/init.d/honeyd stop to stop the service

3.7 TESTING HONEYPOT

1. LOCALLY

Started testing Honeyd locally , (i.e accessing virtual host from the hosting machine) using the sample configuration file "config.sample " by redirecting the traffic for the 10.0.0.0/8 network to the physical machines loopback interface. First add the route in the routing table to direct Honeyd traffic to the loopback.

route -n add -net 10.0.0.0/8 gw 127.0.0.1

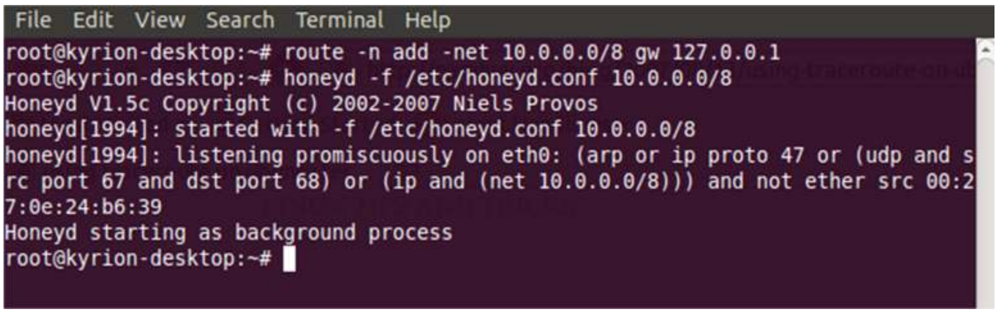
Start Honeyd by the simple command below, and check that it is running under list of running process, or check any other way.

honeyd -f config.sample 10.0.0.0/8

-f :- load configuration from file

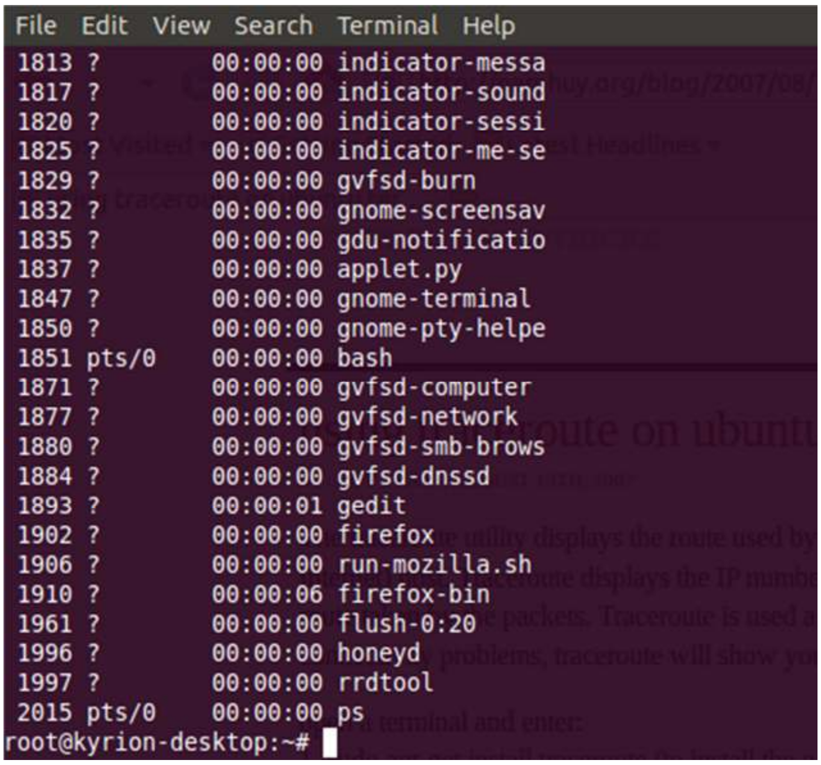
Config.sample – path of honeyd.conf file or the file in which honeyd configuration is stored. The following configuration was used :

http://www.citi.umich.edu/u/provos/honeyd/config.localhost



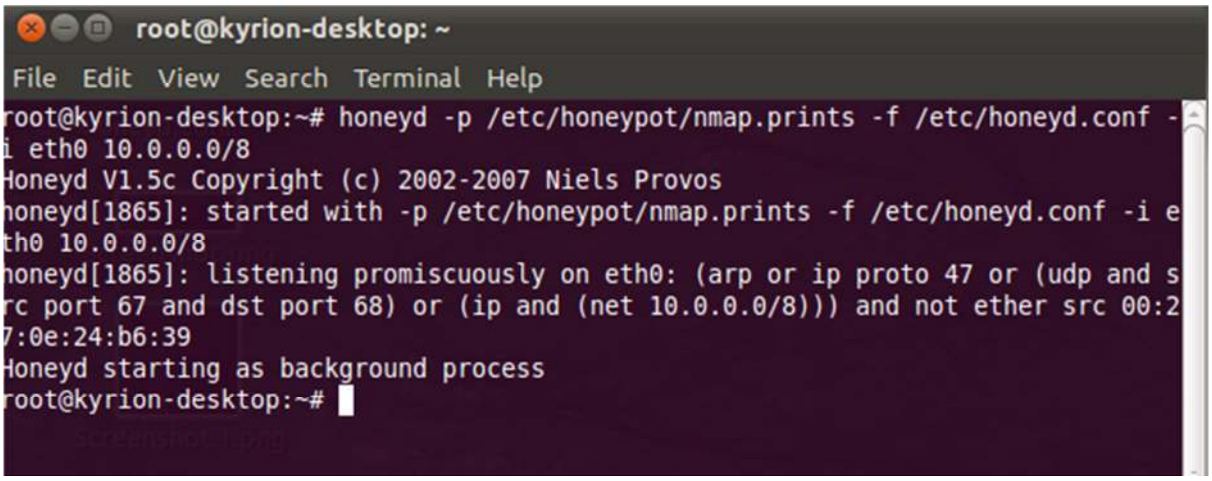
```
root@kyrion-desktop:~# route -n add -net 10.0.0.0/8 gw 127.0.0.1
root@kyrion-desktop:~# honeyd -f /etc/honeyd.conf 10.0.0.0/8
Honeyd V1.5c Copyright (c) 2002-2007 Niels Provos
honeyd[1994]: started with -f /etc/honeyd.conf 10.0.0.0/8
honeyd[1994]: listening promiscuously on eth0: (arp or ip proto 47 or (udp and s
rc port 67 and dst port 68) or (ip and (net 10.0.0.0/8))) and not ether src 00:2
7:0e:24:b6:39
Honeyd starting as background process
root@kyrion-desktop:~#
```

Honeyd can be seen in the list of runnin process :



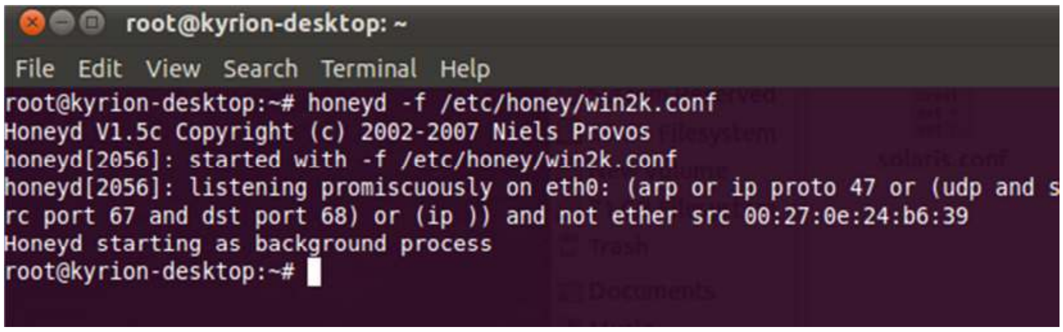
```
1813 ?        00:00:00 indicator-messa
1817 ?        00:00:00 indicator-sound
1820 ?        00:00:00 indicator-sessi
1825 ?        00:00:00 indicator-me-se
1829 ?        00:00:00 gvfsd-burn
1832 ?        00:00:00 gnome-screensav
1835 ?        00:00:00 gdu-notificatio
1837 ?        00:00:00 applet.py
1847 ?        00:00:00 gnome-terminal
1850 ?        00:00:00 gnome-pty-helpe
1851 pts/0    00:00:00 bash
1871 ?        00:00:00 gvfsd-computer
1877 ?        00:00:00 gvfsd-network
1880 ?        00:00:00 gvfsd-smb-brows
1884 ?        00:00:00 gvfsd-dnssd
1893 ?        00:00:01 gedit
1902 ?        00:00:00 firefox
1906 ?        00:00:00 run-mozilla.sh
1910 ?        00:00:06 firefox-bin
1961 ?        00:00:00 flush-0:20
1996 ?        00:00:00 honeyd
1997 ?        00:00:00 rrdtool
2015 pts/0    00:00:00 ps
root@kyrion-desktop:~#
```

Starting honeyd with nmap configurations :



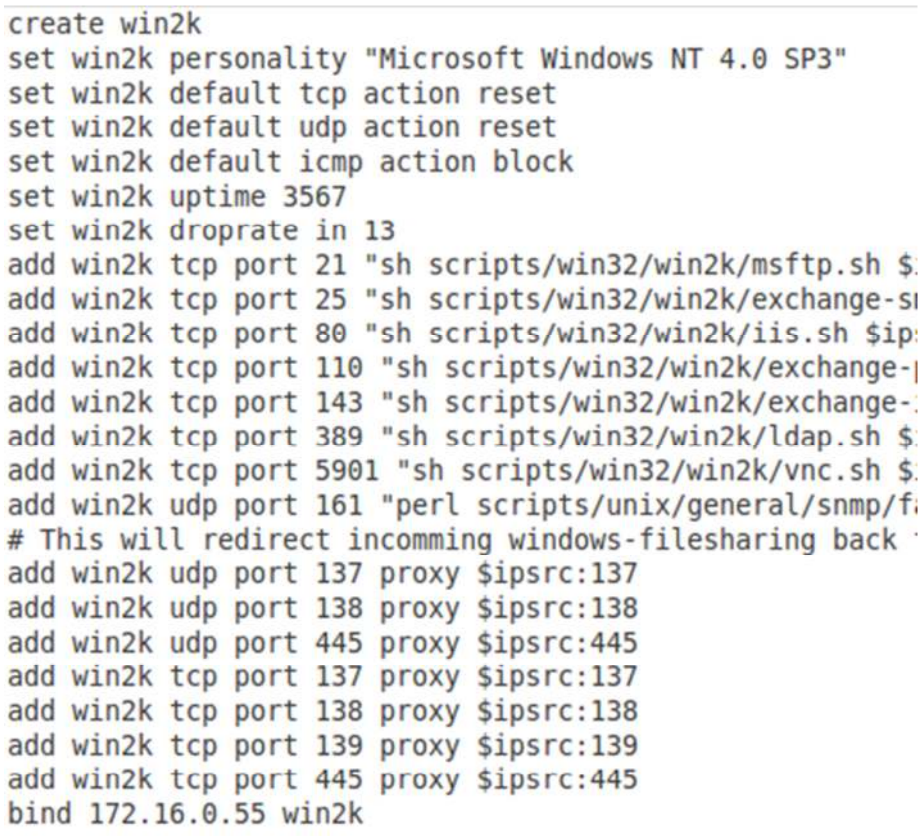
```
root@kyrion-desktop:~# honeyd -p /etc/honeypot/nmap.prints -f /etc/honeyd.conf -
i eth0 10.0.0.0/8
Honeyd V1.5c Copyright (c) 2002-2007 Niels Provos
honeyd[1865]: started with -p /etc/honeypot/nmap.prints -f /etc/honeyd.conf -i e
th0 10.0.0.0/8
honeyd[1865]: listening promiscuously on eth0: (arp or ip proto 47 or (udp and s
rc port 67 and dst port 68) or (ip and (net 10.0.0.0/8))) and not ether src 00:2
7:0e:24:b6:39
Honeyd starting as background process
root@kyrion-desktop:~#
```

2. ON NETWORK



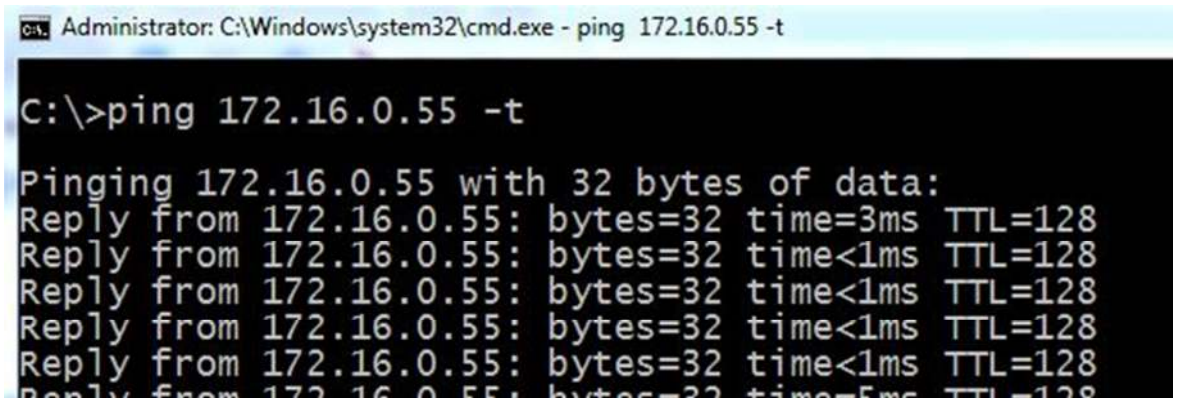
```
root@kyrion-desktop:~# honeyd -f /etc/honey/win2k.conf
Honeyd V1.5c Copyright (c) 2002-2007 Niels Provos
honeyd[2056]: started with -f /etc/honey/win2k.conf
honeyd[2056]: listening promiscuously on eth0: (arp or ip proto 47 or (udp and s
rc port 67 and dst port 68) or (ip )) and not ether src 00:27:0e:24:b6:39
Honeyd starting as background process
root@kyrion-desktop:~#
```

The following configuration was used to emulate a windows machine :



```
create win2k
set win2k personality "Microsoft Windows NT 4.0 SP3"
set win2k default tcp action reset
set win2k default udp action reset
set win2k default icmp action block
set win2k uptime 3567
set win2k droprate in 13
add win2k tcp port 21 "sh scripts/win32/win2k/msftp.sh $
add win2k tcp port 25 "sh scripts/win32/win2k/exchange-s
add win2k tcp port 80 "sh scripts/win32/win2k/iis.sh $ip
add win2k tcp port 110 "sh scripts/win32/win2k/exchange-
add win2k tcp port 143 "sh scripts/win32/win2k/exchange-
add win2k tcp port 389 "sh scripts/win32/win2k/ldap.sh $
add win2k tcp port 5901 "sh scripts/win32/win2k/vnc.sh $
add win2k udp port 161 "perl scripts/unix/general/snmp/f
# This will redirect incomming windows-filesharing back
add win2k udp port 137 proxy $ipsrc:137
add win2k udp port 138 proxy $ipsrc:138
add win2k udp port 445 proxy $ipsrc:445
add win2k tcp port 137 proxy $ipsrc:137
add win2k tcp port 138 proxy $ipsrc:138
add win2k tcp port 139 proxy $ipsrc:139
add win2k tcp port 445 proxy $ipsrc:445
bind 172.16.0.55 win2k
```

Pinging the virtual honeypot for testing :



Administrator: C:\Windows\system32\cmd.exe - ping 172.16.0.55 -t

```
C:\>ping 172.16.0.55 -t

Pinging 172.16.0.55 with 32 bytes of data:
Reply from 172.16.0.55: bytes=32 time=3ms TTL=128
Reply from 172.16.0.55: bytes=32 time<1ms TTL=128
Reply from 172.16.0.55: bytes=32 time<1ms TTL=128
Reply from 172.16.0.55: bytes=32 time<1ms TTL=128
Reply from 172.16.0.55: bytes=32 time<1ms TTL=128
```

3.8 PORT BEHAVIOR

TCP (default is Open)
- Open: Respond with Syn/Ack, establish connection
- Block: Drop packet and do not reply
- Reset: Respond with RST
- Tarpit: Sticky connection

UDP (default is Closed)
- Open: No response
- Block: Drop packet and do not reply
- Reset: Respond with ICMP port error message

ICMP (default is Open)
- Open: Reply to ICMP packets
- Block: Drop packet and do not reply

OTHER CONFIGURATIONS (EMULATIONS)

A lot of these configurations can be found on the internet. Some of them are available on http://www.honeyd.org/configuration.php . This includes network emulation also.