

Abstract

Creating a simple Honeypot using python and using a Honeypot created by PenBox on a Ubuntu Linux Virtual machine to demonstrate how a Honeypot can protect a system from cyberattacks.

Introduction

On a Ubuntu Virtual machine, two Honeypots will be demonstrated-- one being a simple Honeypot coded in Python with the usage of its socket library, and the other is a Honeypot created by using PentBox. The Ubuntu Virtual machine will be the target system hosting these Honeypots, and the Kali Linux Virtual machine will be the attacker.

Basic Python Honeypot

First, in order to create the Python Honeypot, we must learn about the sockets available on the Linux system by looking at the socket man page with the command:

man socket

Commands used:

man socket (socket manual page)

```
evhx@evhx-VirtualBox:~$ man socket
evhx@evhx-VirtualBox:~$
```

```
SOCKET(2)                                Linux Programmer's Manual                                SOCKET(2)
NAME
    socket - create an endpoint for communication
SYNOPSIS
    #include <sys/types.h>                /* See NOTES */
    #include <sys/socket.h>

    int socket(int domain, int type, int protocol);
DESCRIPTION
    socket() creates an endpoint for communication and re-
    turns a file descriptor that refers to that endpoint.
    The file descriptor returned by a successful call will
    be the lowest-numbered file descriptor not currently
    open for the process.

    The domain argument specifies a communication domain;
    this selects the protocol family which will be used for
    communication. These families are defined in
    <sys/socket.h>. The formats currently understood by the
    Linux kernel include:

    Name                Purpose                Ma
n page
    AF_UNIX              Local communication      un
lx(7)
    AF_LOCAL             Synonym for AF_UNIX
    AF_INET              IPv4 Internet protocols    ip
(7)
    AF_AX25              Amateur radio AX.25 protocol  ax
25(4)
    AF_IPX               IPX - Novell protocols
    AF_APPLETALK         AppleTalk                dd
p(7)
    AF_X25               ITU-T X.25 / ISO-8208 protocol  x2
5(7)
```

The commands that we will be using from the socket manual page will be:

AF_INET (Internet Domain)

SOCK_STREAM (Stream Socket)

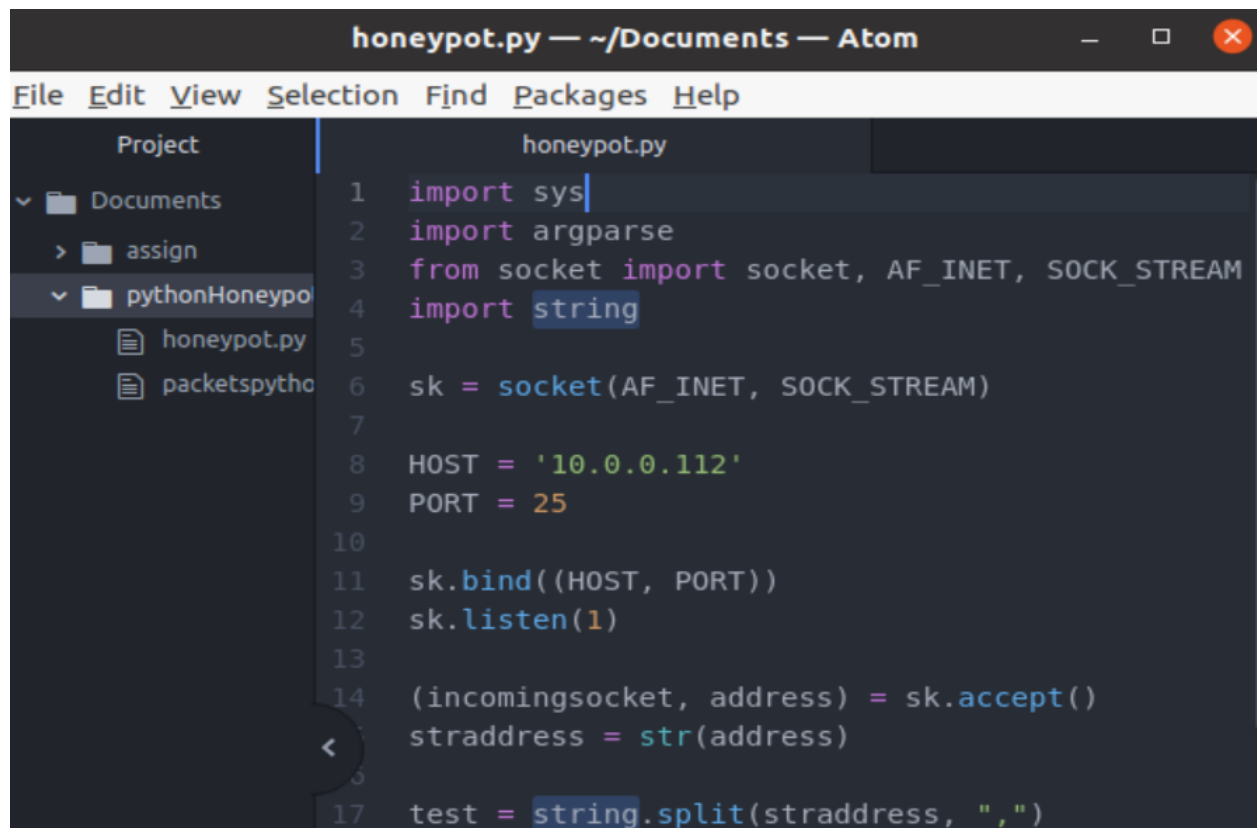
Which will create a stream socket in the Internet domain. Then code in the host machine's ip address, and the port number that will be used. The host and port then need to be binded in order to allow a process to specify the address of the socket.

```
sk.bind((HOST, PORT))
```

To specify the maximum number of queued connected, you add:

```
sk.listen(1) (n = number of allowed connections)
```

The rest of the code would be different ways to react to the machine attempting to access this port.

A screenshot of an Atom text editor window titled "honeypot.py — ~/Documents — Atom". The left sidebar shows a project tree with "Documents" expanded, containing "assign" and "pythonHoneyPot". The "pythonHoneyPot" folder is selected, showing "honeypot.py" and "packetspytho". The main editor area displays the Python code for "honeypot.py".

```
1 import sys
2 import argparse
3 from socket import socket, AF_INET, SOCK_STREAM
4 import string
5
6 sk = socket(AF_INET, SOCK_STREAM)
7
8 HOST = '10.0.0.112'
9 PORT = 25
10
11 sk.bind((HOST, PORT))
12 sk.listen(1)
13
14 (incomingsocket, address) = sk.accept()
15 straddress = str(address)
16
17 test = string.split(straddress, ",")
```

After the simple Honeypot Python program is created, run the program using Python in its home directory using the command:

```
sudo python honeypot.py
```

Once the honeypot is running on the Ubuntu Virtual machine, on the Kali Virtual machine an nmap can be run to check the Ubuntu's machines open ports using the command:

```
nmap -v 10.0.0.112
```

The Python Honeypot is running on port 25, so if a machine were to try to access it, the Honeypot will inform the Ubuntu machine. To demonstrate this, on the Kali Linux machine run the command:

```
telnet 10.0.0.112 25
```

Commands used:

sudo	(root access)
python	(to run Python)
honeypot.py	(name of the Honeypot program)
nmap -v 10.0.0.112	(network scanner used on address 10.0.0.112)
telnet 10.0.0.112 25	(remotely connect to address with usage of a port)

```
(kali㉿kali)-[~]
└─$ nmap -v 10.0.0.112
Starting Nmap 7.91 ( https://nmap.org )
Initiating Ping Scan at 20:25
Scanning 10.0.0.112 [2 ports]
Completed Ping Scan at 20:25, 0.00s elapse (0.000s latency)
Initiating Parallel DNS resolution of 1 host at 20:25
Completed Parallel DNS resolution of 1 host at 20:25, 0.00s elapse
Initiating Connect Scan at 20:25
Scanning 10.0.0.112 [1000 ports]
Discovered open port 23/tcp on 10.0.0.112
Discovered open port 25/tcp on 10.0.0.112
Discovered open port 22/tcp on 10.0.0.112
Discovered open port 631/tcp on 10.0.0.112
Completed Connect Scan at 20:25, 0.00s elapse
Nmap scan report for 10.0.0.112
Host is up (0.0070s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
631/tcp   open  ipp
```

```
(kali㉿kali)-[~]
└─$ telnet 10.0.0.112 25
Trying 10.0.0.112...
Connected to 10.0.0.112.
Escape character is '^]'.
└─
```

The Python Honeypot was able to detect both instances of connection attempts from the Kali Linux machine. As previously stated, the maximum allowed queues was only 1 (listen(1)), so the Honeypot is terminated after a single attempt, which can be seen after running another nmap command:

```
nmap -v 10.0.0.112 -p 25
```

This shows that port 25 was now closed.

Commands used:

```
nmap -v 10.0.0.112 -p 25    (network scanner used on address 10.0.0.112 with port 25 )
```

In conclusion, creating a simple Python Honeypot mostly requires you to understand the sockets on a Linux machine and how foreign connections are created between machines.

```
evhx@evhx-VirtualBox:~/Documents/pythonHoneypot$ sudo python honeypot.py
Connection attempt on port 25 from 10.0.0.94 : 32884
evhx@evhx-VirtualBox:~/Documents/pythonHoneypot$ sudo python honeypot.py
Connection attempt on port 25 from 10.0.0.94 : 32890
```

```
(kali㉿kali)-[~]
└─$ nmap -v 10.0.0.112 -p 25
Starting Nmap 7.91 ( https://nmap.org )
Initiating Ping Scan at 19:01
Scanning 10.0.0.112 [2 ports]
Completed Ping Scan at 19:01, 0.00s elapsed
Initiating Parallel DNS resolution of 1 host
Completed Parallel DNS resolution of 1 host
Initiating Connect Scan at 19:01
Scanning 10.0.0.112 [1 port]
Completed Connect Scan at 19:01, 0.00s elapsed
Nmap scan report for 10.0.0.112
Host is up (0.00033s latency).

PORT      STATE SERVICE
25/tcp    closed smtp
```

Honeypot using PentBox

Another great way to test the usage of a Honeypot would be to try out the many open-source Honeypots available. Here we will try PentBox, which provides many open-source security tools for networks and systems. To start on the Ubuntu Virtual machine, we will disable the firewall using the command:

```
sudo i                                (if root access is wished for)
ufw disable
```

On the Kali Linux machine, we can ping the Ubuntu machine to test if its running on the same network with the command:

```
ping 10.0.0.112
```

Commands used:

ufw	(default firewall configuration tool)
ufw disable	(disable firewall)
ping 10.0.0.112	(check if running on the same network)

```
evhx@evhx-VirtualBox:~$ sudo -i
root@evhx-VirtualBox:~# ufw disable
Firewall stopped and disabled on system startup
```

```
(kali㉿kali)-[~]
└─$ ping 10.0.0.112
PING 10.0.0.112 (10.0.0.112) 56(84) bytes of data:
64 bytes from 10.0.0.112: icmp_seq=1 ttl=64 time=0.225 ms
64 bytes from 10.0.0.112: icmp_seq=2 ttl=64 time=0.229 ms
64 bytes from 10.0.0.112: icmp_seq=3 ttl=64 time=0.238 ms
64 bytes from 10.0.0.112: icmp_seq=4 ttl=64 time=0.235 ms
64 bytes from 10.0.0.112: icmp_seq=5 ttl=64 time=0.241 ms
64 bytes from 10.0.0.112: icmp_seq=6 ttl=64 time=0.236 ms
64 bytes from 10.0.0.112: icmp_seq=7 ttl=64 time=0.225 ms
64 bytes from 10.0.0.112: icmp_seq=8 ttl=64 time=0.219 ms
64 bytes from 10.0.0.112: icmp_seq=9 ttl=64 time=0.241 ms
64 bytes from 10.0.0.112: icmp_seq=10 ttl=64 time=0.226 ms
64 bytes from 10.0.0.112: icmp_seq=11 ttl=64 time=0.254 ms
64 bytes from 10.0.0.112: icmp_seq=12 ttl=64 time=0.201 ms
64 bytes from 10.0.0.112: icmp_seq=13 ttl=64 time=0.240 ms
64 bytes from 10.0.0.112: icmp_seq=14 ttl=64 time=0.238 ms
64 bytes from 10.0.0.112: icmp_seq=15 ttl=64 time=0.218 ms
64 bytes from 10.0.0.112: icmp_seq=16 ttl=64 time=0.229 ms
64 bytes from 10.0.0.112: icmp_seq=17 ttl=64 time=0.226 ms
64 bytes from 10.0.0.112: icmp_seq=18 ttl=64 time=0.227 ms
64 bytes from 10.0.0.112: icmp_seq=19 ttl=64 time=0.155 ms
64 bytes from 10.0.0.112: icmp_seq=20 ttl=64 time=0.242 ms
64 bytes from 10.0.0.112: icmp_seq=21 ttl=64 time=0.639 ms
64 bytes from 10.0.0.112: icmp_seq=22 ttl=64 time=0.302 ms
```

On the Ubuntu machine, PentBox will be downloaded from the PentBox repository using the command:

```
git clone https://www.github.com/technicaldada/pentbox
```

After the repository is downloaded, we go to the PentBox directory and check its contents with the ls command.

```
cd pentbox && ls
```

In the pentbox directory, there is a compressed file named 'pentbox.tar.gz', which will be decompressed using the command:

```
tar -xzf pentbox.tar.gz
```

After decompressing the file, a directory called 'pentbox-1.8' will be available, which is where PentBox can be used.

```
cd pentbox-1.8 && ls
```

Commands used:

git clone https://www.github.com/technicaldada/pentbox	(retrieve a git repository)
cd pentbox	(change directory)
tar -xzf pentbox.tar.gz	(decompress file)
ls	(list files)

```
root@evhx-VirtualBox:~# git clone https://www.github.com/technicaldada/pentbox
Cloning into 'pentbox'...
warning: redirecting to https://github.com/technicaldada/pentbox.git/
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 21 (delta 0), reused 0 (delta 0), pack-reused 17
Unpacking objects: 100% (21/21), 2.11 MiB | 6.46 MiB/s, done.
root@evhx-VirtualBox:~#
```

```
root@evhx-VirtualBox:~# cd pentbox
root@evhx-VirtualBox:~/pentbox#
```

```
root@evhx-VirtualBox:~/pentbox# ls
pentbox-1.8  pentbox.tar.gz  README.md
root@evhx-VirtualBox:~/pentbox# cd pentbox-1.8 && ls
changelog.txt  COPYING.txt  lib  other  pb_update.rb  pentbox.rb  readme.txt  todo.txt  tools
root@evhx-VirtualBox:~/pentbox/pentbox-1.8#
```

Finally to run PentBox, execute the command:

```
./pentbox.rb
```

To use the honeypot:

> 2- Network tools

> 3- Honeypot

Then you can choose to either run a 'Fast Auto Configuration' of the Honeypot, or be able to choose your own ports and messages with the 'Manual Configuration' option.

```
PentBox 1.8

.!!!!!!:
~::~!!!!!!:
:$$NWX!!:
$$$$$##WX!:
$$$$$ $$$UX
^$$$$B $$$$
**$bd$$$$
****

.::!!!!!!:
.!!!!!!:
.!!!!!!UWWW$$$
.!!!!!!XUWW$$$$$$$$$$$P
.<!!!!UW$$$$ $$$$$$$$#
:!!UW$$$$$$$$$ 4$$$$$*
$$$$$$$$$$$$$ d$$R*
'*$$$$$$$$$$$$$o+#
*****

----- Menu          ruby2.7.0 @ x86_64-linux-gnu
1- Cryptography tools
2- Network tools
3- Web
4- Ip grabber
5- Geolocation ip
6- Mass attack
7- License and contact

1- Net DoS Tester
2- TCP port scanner
3- Honeypot
4- Fuzzer
5- DNS and host gathering
6- MAC address geolocation (samy.pl)
0- Back
-> 3

// Honeypot //

You must run PentBox with root privileges.

Select option.

1- Fast Auto Configuration
2- Manual Configuration [Advanced Users, more options]
-> 1

HONEYPOT ACTIVATED ON PORT 80 (2021-12-02 19:58:20 -0800)
```

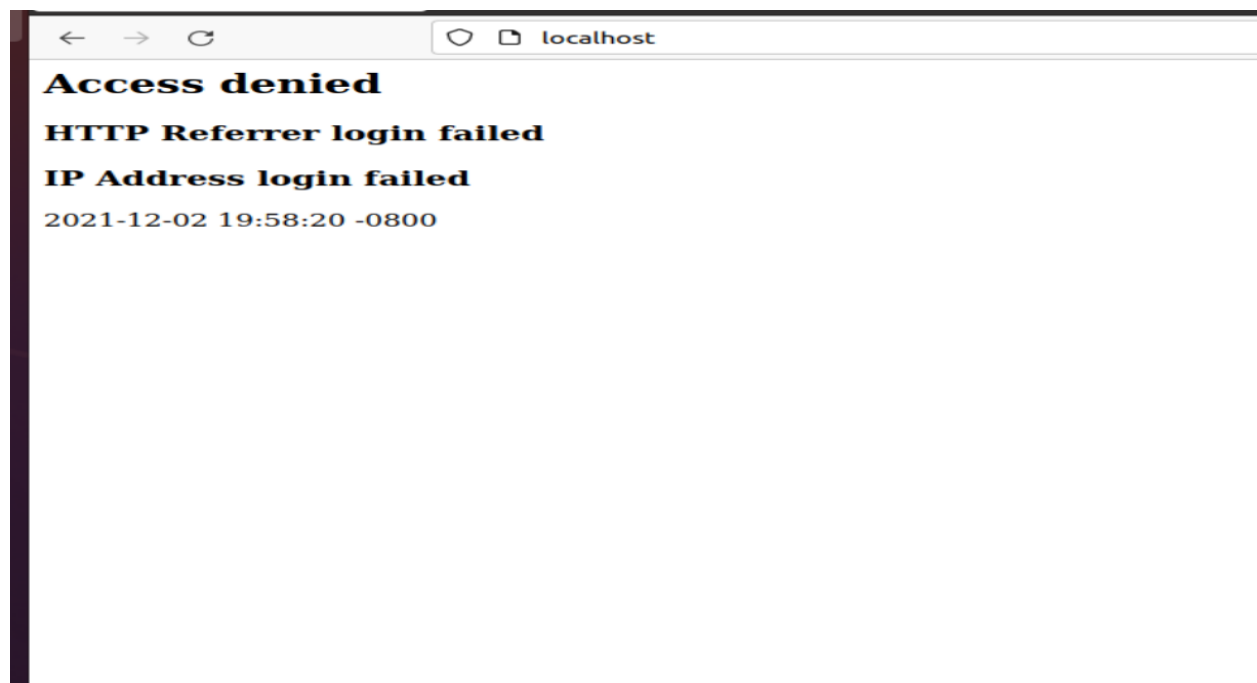
The Honeypot was previously activated on port 80. Port 80 is a port used for http, so to test this out, open up a browser on the Ubuntu machine and access port 80 using the command:

localhost:80

You can see the access to this port was denied because of the Honeypot, and the intrusion attempt shows that the attempt came from address '127.0.0.1', which is the localhost.

```
INTRUSION ATTEMPT DETECTED! from 127.0.0.1:41104 (2021-12-02 19:59:19 -0800)
-----
GET / HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:94.0) Gecko/20100101 Firefox/94.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

INTRUSION ATTEMPT DETECTED! from 127.0.0.1:41106 (2021-12-02 19:59:22 -0800)
-----
GET /favicon.ico HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:94.0) Gecko/20100101 Firefox/94.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://localhost/
Sec-Fetch-Dest: image
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
```



Now we can do the same but with the Kali Linux machine. On the Kali Linux machine we will scan the Ubuntu machine using nmap, which will show all the open ports. Here we see that port 80 is open, which would be great as an attacker considering you simply need to enter the ip address and port number of the system you're trying to access. The attempt is then blocked by the Honeypot.

```
nmap -v 10.0.0.112
```

To concatenate the output into a file, simply add:

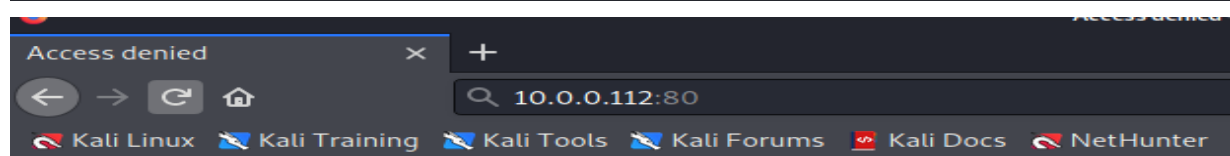
```
>> ~/Desktop/nmap.output.txt      (file location/name)
```

Commands used:

```
nmap -v 10.0.0.112    (network scanner used on address 10.0.0.112      )
```

```
(kali@kali)-[~]
$ nmap -v 10.0.0.112
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-23 11:14 EST
Initiating Ping Scan at 11:14
Scanning 10.0.0.112 [2 ports]
Completed Ping Scan at 11:14, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 11:14
Completed Parallel DNS resolution of 1 host. at 11:14, 0.03s elapsed
Initiating Connect Scan at 11:14
Scanning 10.0.0.112 [1000 ports]
Discovered open port 23/tcp on 10.0.0.112
Discovered open port 80/tcp on 10.0.0.112
Discovered open port 631/tcp on 10.0.0.112
Completed Connect Scan at 11:14, 1.23s elapsed (1000 total ports)
Nmap scan report for 10.0.0.112
Host is up (0.00055s latency).
Not shown: 995 closed ports
PORT      STATE      SERVICE
23/tcp    open       telnet
80/tcp    open       http
631/tcp   open       ipp
4000/tcp  filtered  remoteanything
31337/tcp filtered  Elite

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1.32 seconds
```



Access denied

HTTP Referrer login failed

IP Address login failed

2021-12-02 19:58:20 -0800

The attempt from the Kali Linux machine will be shown on the Honeypot. Then we will open port 66, and attempt to access it using the Kali Linux machine using telnet.

```
INTRUSION ATTEMPT DETECTED! from 10.0.0.94:34738 (2021-12-02 21:27:10 -0800)
-----
GET /favicon.ico HTTP/1.1
Host: 10.0.0.112
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

You must run PentBox with root privileges.

Select option.

- 1- Fast Auto Configuration
- 2- Manual Configuration [Advanced Users, more options]

-> 2

Insert port to Open.

-> 66

Insert false message to show.

-> DID YOU TRY TO ACCESS MY COMPUTER?!?! HOW DARE YOU!

Save a log with intrusions?

(y/n) -> y

Save a log with intrusions?

(y/n) -> y

Log file name? (incremental)

Default: */pentbox/other/log_honeypot.txt

->

Activate beep() sound when intrusion?

(y/n) -> y

HONEYPOT ACTIVATED ON PORT 66 (2021-12-02 21:38:43 -0800)

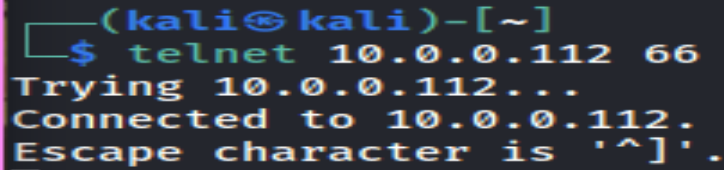
To remotely connect from the Kali Linux machine to the Ubuntu machine use the command:
(after using nmap to check the ports of course)

```
telnet 10.0.0.112 66
```

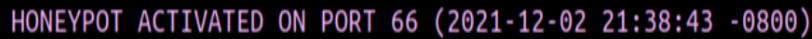
You then see the attempt on the Honeypot once again.

Commands used:

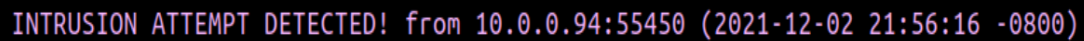
```
telnet 10.0.0.112 66 (remotely connect to address with usage of a port )
```

A terminal window with a dark background. The prompt is (kali@kali)-[~]. The user enters the command \$ telnet 10.0.0.112 66. The output shows 'Trying 10.0.0.112...' followed by 'Connected to 10.0.0.112.' and 'Escape character is '^]'.

```
(kali@kali)-[~]  
$ telnet 10.0.0.112 66  
Trying 10.0.0.112...  
Connected to 10.0.0.112.  
Escape character is '^]'.  
_
```

A log entry on a dark background showing 'HONEYPOT ACTIVATED ON PORT 66' followed by a timestamp and timezone.

```
HONEYPOT ACTIVATED ON PORT 66 (2021-12-02 21:38:43 -0800)
```

A log entry on a dark background showing 'INTRUSION ATTEMPT DETECTED!' followed by source IP, port, timestamp, and timezone. A dashed line follows.

```
INTRUSION ATTEMPT DETECTED! from 10.0.0.94:55450 (2021-12-02 21:56:16 -0800)  
-----
```

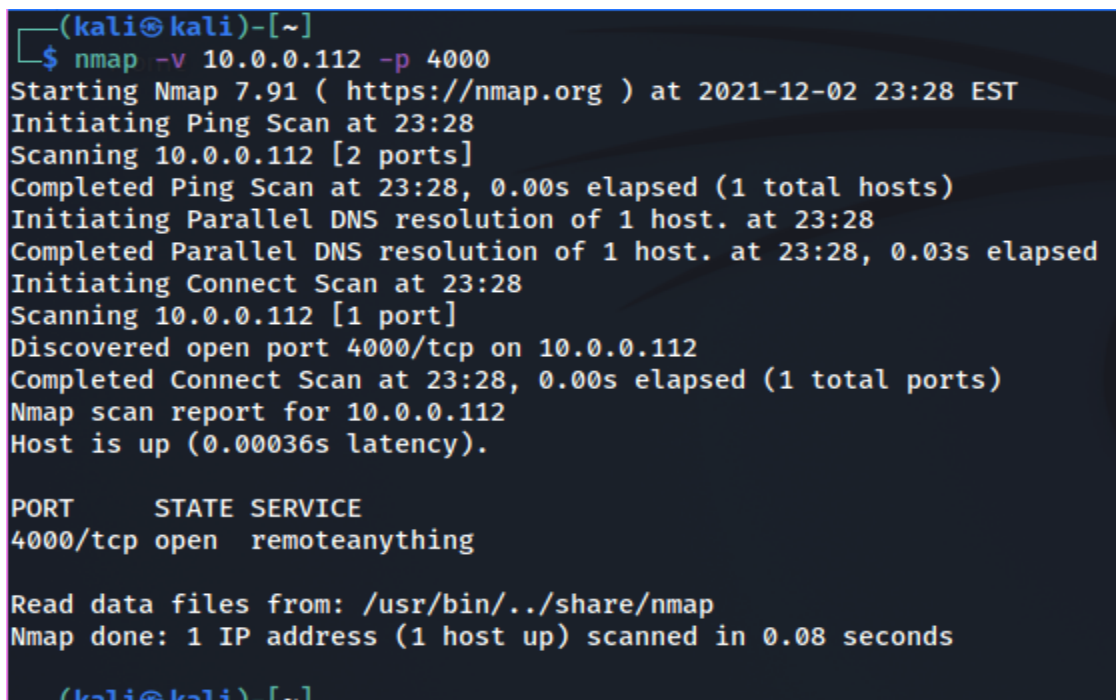
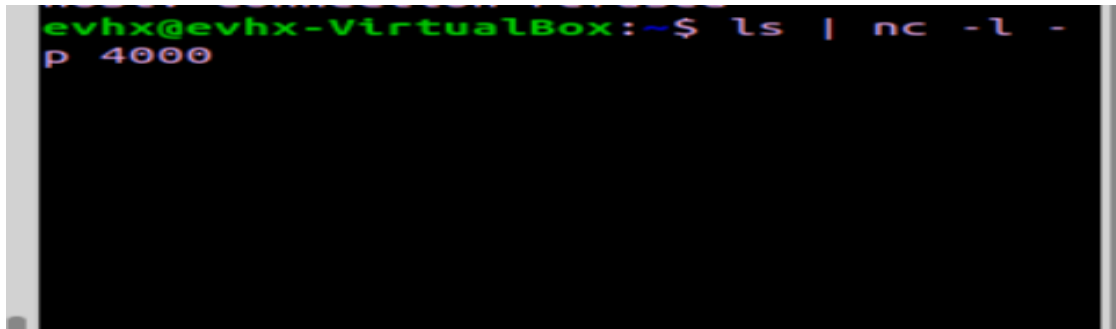
Now lets simply create a port listener using the command:

```
ls | nc -l -p 4000
```

Using nmap we can see that this port is open and allows 'remote anything'.

Commands used:

```
ls | nc -l -p 4000      (list and listen on port 4000  )
```



Using the Kali Linux machine, lets remotely connect to port 4000 on the Ubuntu machine using the command:

```
telnet 10.0.0.112 4000
```

We can then clearly see that the Ubuntu machine was easily accessible by using telnet on the Kali Linux machine.

Commands used:

telnet 10.0.0.112 4000 (remotely connect to address with usage of a port)

```
(kali㉿kali)-[~]  
$ telnet 10.0.0.112 4000  
  
Trying 10.0.0.112...  
Connected to 10.0.0.112.  
Escape character is '^['.  
Desktop  
Documents  
Downloads  
Music  
Pictures  
Public  
snap  
Templates  
Videos  
  
^C  
exit  
clear  
^]  
  
'^['  
fgfg  
hello  
how are you?  
this isnt supposed to work!  
meowmeow
```

```
p 4000  
  
exit  
clear  
^]  
  
'^['  
fgfg  
hello  
how are you?  
this isnt supposed to work!  
meowmeow
```

Another port that we can test is the CUPS port on 631, which is a printing system service on Linux systems. To open the CUPS port, execute the commands:

```
sudo apt-get install cups -y
sudo systemctl start cups
```

You must allow listening on all interfaces in the 'cupsd.conf' file, and then restart CUPS with the commands:

```
sudo apt-get install cups -y
sudo systemctl start cups
```

Commands used:

sudo apt-get install cups -y	(install CUPS)
sudo systemctl start cups	(start CUPS)
sudo nano /etc/cups/cupsd.conf	(write in the CUPS file)
sudo systemctl restart cups	(Restart CUPS)

```
evhx@evhx-VirtualBox:~$ sudo apt-g
et install cups -y
[sudo] password for evhx:
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package -y
evhx@evhx-VirtualBox:~$ sudo syste
mctl start cups
evhx@evhx-VirtualBox:~$ sudo syste
mctl enable cups
Synchronizing state of cups.servic
e with SysV service script with /l
ib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sy
sv-install enable cups
evhx@evhx-VirtualBox:~$ sudo nano
/etc/cups/cupsd.conf
evhx@evhx-VirtualBox:~$ sudo syste
mctl restart cups
evhx@evhx-VirtualBox:~$
```

Using nmap on the Kali Linux machine, you can see that port 631 is now open with service 'ipp', which is the Internet Printing Protocol, so it means that anyone can connect to printers on port 631, which can lead to information disclosure and the manipulation of print jobs. Unfortunately, if the port is already open, the Pentbox Honeypot cannot be placed and another Honeypot would have to be added to port 631.

```
(kali㉿kali)-[~]
└─$ nmap -v 10.0.0.112 -p 631
Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-03 01:09 EST
Initiating Ping Scan at 01:09
Scanning 10.0.0.112 [2 ports]
Completed Ping Scan at 01:09, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 01:09
Completed Parallel DNS resolution of 1 host. at 01:09, 0.03s elapsed
Initiating Connect Scan at 01:09
Scanning 10.0.0.112 [1 port]
Discovered open port 631/tcp on 10.0.0.112
Completed Connect Scan at 01:09, 0.00s elapsed (1 total ports)
Nmap scan report for 10.0.0.112
Host is up (0.00053s latency).

PORT      STATE SERVICE
631/tcp   open  ipp

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.08 seconds

(kali㉿kali)-[~]
```

Conclusion

Honeypots are great to have for their security benefits, since it's basically creating an extra layer of protection as a decoy that is capable of stopping and monitoring cyberattacks. Hackers will be wasting their time trying to attack the Honeypot thinking they're attacking your system.

Honeypots combined with firewalls work great since firewalls keep attackers out of the network, and Honeypots are used to gather information from those same attackers, similar to having a shield against your enemies but also analyzing the ways they attack as well.