NorseRage Installation Guides

Linux Edition

Table of Contents

Artillery

https://github.com/BinaryDefense/artillery

Description

The purpose of Artillery is to provide a combination of honeypot, file-system monitoring, system hardening, real-time threat intelligence feed, and overall health of a server monitoring-tool; to create a comprehensive way to secure a system. Project Artillery was written to be an addition to security on a server and make it very difficult for attackers to penetrate a system. The concept is simple: project Artillery will monitor the filesystem looking for indicators of change. If one is detected, an email is sent to the server owner. Artillery also listens for rogue connections. If detected, a notification is sent to the server owner, and the offending IP address is banned.

Install Location

`/var/artillery`

Config File Location

`/var/artillery/config`

Usage

All options are set in the configuration file so there are no command line arguments for Artillery.

`/var/artillery$ sudo python3 artillery.py`

Example 1: Running Artillery

Change to the install location and run Artillery.

`/var/artillery$ sudo python3 artillery.py 2> /dev/null`

Verify that Artillery is running by opening a new terminal and typing the following command. You should see that the python3 process is listening on a bunch of ports. These are the ports that are configured by default in the config file.

`$ sudo netstat -nlp | grep python`

```
tcp  0  0  0.0.0.0:5900   0.0.0.0:*  LISTEN  9020/python3
tcp  0  0  0.0.0.0:110    0.0.0.0:*  LISTEN  9020/python3
tcp  0  0  0.0.0.0:10000  0.0.0.0:*  LISTEN  9020/python3
tcp  0  0  0.0.0.0:8080   0.0.0.0:*  LISTEN  9020/python3
```

```
tcp  0  0  0.0.0.0:21     0.0.0.0:*  LISTEN  9020/python3

tcp  0  0  0.0.0.0:1433   0.0.0.0:*  LISTEN  9020/python3

tcp  0  0  0.0.0.0:1337   0.0.0.0:*  LISTEN  9020/python3

tcp  0  0  0.0.0.0:25     0.0.0.0:*  LISTEN  9020/python3

tcp  0  0  0.0.0.0:44443  0.0.0.0:*  LISTEN  9020/python3

tcp  0  0  0.0.0.0:1723   0.0.0.0:*  LISTEN  9020/python3

tcp  0  0  0.0.0.0:445    0.0.0.0:*  LISTEN  9020/python3

tcp  0  0  0.0.0.0:3389   0.0.0.0:*  LISTEN  9020/python3

tcp  0  0  0.0.0.0:135    0.0.0.0:*  LISTEN  9020/python3

tcp  0  0  0.0.0.0:5800   0.0.0.0:*  LISTEN  9020/python3
```

Example 2: Triggering a Honeyport

Start Artillery as in Example 1. Then find the IP address of the
machine by using ifconfig.

```
$ ifconfig
```

```
eth0    Link encap:Ethernet  HWaddr 00:0c:29:6c:14:79

        inet addr:192.168.1.137  Bcast:192.168.1.255  Mask:255.255.255.0

        inet6 addr: fe80::20c:29ff:fe6c:1479/64 Scope:Link

        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

        RX packets:136005 errors:0 dropped:0 overruns:0 frame:0

        TX packets:59528 errors:0 dropped:0 overruns:0 carrier:0

        collisions:0 txqueuelen:1000

        RX bytes:146777599 (146.7 MB)  TX bytes:7955605 (7.9 MB)

        Interrupt:19 Base address:0x2000


lo      Link encap:Local Loopback

        inet addr:127.0.0.1  Mask:255.0.0.0

        inet6 addr: ::1/128 Scope:Host

        UP LOOPBACK RUNNING  MTU:16436  Metric:1

        RX packets:12930 errors:0 dropped:0 overruns:0 frame:0

        TX packets:12930 errors:0 dropped:0 overruns:0 carrier:0

        collisions:0 txqueuelen:0

        RX bytes:3413486 (3.4 MB)  TX bytes:3413486 (3.4 MB)
```

In this case the IP address for the machine is 192.168.1.137. Now, using another machine and either netcat or telnet, connect to the ADHD machine on port 21. Port 21 is one of the ports Artillery is monitoring.

NOTE: You may want to mute your speakers before running this command if you are in a place where you could disturb others.

```
$ telnet 192.168.1.137 21
```

```
    Trying 192.168.1.137...
    Connected to ubuntu.
    Escape character is '^]'.
    ?Q???y{+g??•g?gF?=?>??~??$}k????KU??M
    <<<snip>>>
    ?????P??N?+???T?Z???~0?Connection closed by foreign host.
```

The reason for muting your speakers should be apparent now. Artillery sends a bunch of garbage characters when a connection is made. Some of these characters are usually ASCII bell characters that will make your computer ding a whole lot.

The result of our connection attempt should be that Artillery automatically blocked all connections from the remote IP address. Try connecting again to the ADHD machine using either telnet or netcat.

```
$ telnet 192.168.1.137 21
```

```
    Trying 192.168.1.137...
    telnet: connect to address 192.168.1.137: Operation timed out
    telnet: Unable to connect to remote host
```

As you can see the connection timed out, indicating that we no longer have access from the remote host we are currently using.

NOTE: If you try to do this using another terminal within ADHD, this will not work. Artillery whitelists local connections so you can't block 127.0.0.1.

Back on ADHD, open up a new terminal and check syslog for Artillery's logs.

```
$ tail /var/log/syslog | grep Artillery
```

```
    <<<snip>>>
```

```
    Feb 12 13:38:17 ubuntu 2014-02-12 13:38:17.957044 [!] Artillery has blocked (and
blacklisted the IP Address: 192.168.1.193 for connecting to a honeypot restricted por
t: 445
```

At the end of the output you should see a log entry similar to the
above. Note the IP address as we will now undo the ban Artillery put
into place. In this instance, the remote IP address is 192.168.1.193.

`/var/artillery$ sudo python3 remove_ban.py 192.168.1.193`

```
    Listing all iptables looking for a match... if there is a massive amount of block
ed IP's this could take a few minutes..

    1
```

If for some reason the script doesn't work, try running it a few times to unban your IP,
or simply flush iptables like so:

`/var/artillery$ sudo iptables -F`

It should be noted that the above command will remove all iptables rules.

Example 3: Adding a File to a Watched Directory

Start Artillery as in [Example 1: Running Artillery]. Open up a new terminal and add a new
file into a watched directory.

`$ sudo touch /var/www/bad_file`

Artillery is set up to check for changes every 60 seconds by default so
the log file may not show the change immediately. Watch for the change
by typing the following command.

`$ watch tail -n 15 /var/log/syslog`

And look for output similar to the following.

```
    ****************************************************************

    The following changes were detected at 2013-01-15 21:15:38.541391

    ****************************************************************

    1a2

    > /var/www/bad_file: cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36c
e9ce47d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e


    ************************ End of changes. ************************
```

Bear Trap
https://github.com/chrisbdaemon/BearTrap

Description
BearTrap is meant to be a portable network defense utility written entirely in Ruby. It opens "trigger" ports on the host that an attacker would connect to. When the attacker connects and/or performs some interactions with the trigger an alert is raised and the attacker's IP address is potentially blacklisted.

Install Location
/opt/beartrap/

Config Location
/opt/beartrap/config.yml

Usage
/opt/beartrap$ **sudo ruby bear_trap.rb**

```
    BearTrap v0.2-beta

    Usage: bear_trap.rb [-vd] -c <config file>

    OPTIONS:

    --config  -c <config file>    Filename to load configuration from [REQUIRED]

    --verbose -v                  Verbose output

    --debug   -d                  Debug output (very noisy)

--timeout -t              Ban timeout in seconds
```

Example 1: Basic Usage
Change into the Bear Trap install directory and start Bear Trap.

/opt/beartrap$ **sudo ruby bear_trap.rb -c config.yml -t 600**

Now find the ADHD machine's IP address by opening a new terminal and using ifconfig.

$ **ifconfig**

```
    eth0    Link encap:Ethernet  HWaddr 00:0c:29:6c:14:79

            inet addr:192.168.1.137  Bcast:192.168.1.255  Mask:255.255.255.0

            inet6 addr: fe80::20c:29ff:fe6c:1479/64 Scope:Link

            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

            RX packets:115178 errors:0 dropped:0 overruns:0 frame:0
```

```
            TX packets:43571 errors:0 dropped:0 overruns:0 carrier:0

            collisions:0 txqueuelen:1000

            RX bytes:104926397 (104.9 MB)  TX bytes:4321023 (4.3 MB)

            Interrupt:19 Base address:0x2000
```

Here the IP is 192.168.1.137. From a remote computer use FTP to connect
to the ADHD machine. If it asks for a username, it doesn't matter what
you type as Bear Trap doesn't actually implement a real FTP server.
'adhd' is used in the example below.

`$ ftp 192.168.1.137`

```
    Connected to 192.168.1.137.

    220 BearTrap-ftpd Service ready

    Name (192.168.1.137): adhd

    421 Service not available, remote server has closed connection.

    ftp: Login failed
```

As you can see, the login failed. Type `quit` to exit the FTP client.

```
    ftp\> quit
```

Back in your Bear Trap terminal, you should see new output similar to
below. Bear Trap automatically blocked your remote IP address (in this
case 192.168.1.194) using iptables.

```
    Command: /sbin/iptables -A INPUT -s 192.168.1.194 -j DROP
```

If you try to connect again using FTP from your remote computer, the
operation will time out.

`$ ftp 192.168.1.137`

```
    ftp: Can't connect to '192.168.1.137': Operation timed out

    ftp: Can't connect to '192.168.1.137'
```

To unblock the remote computer, open a new terminal on the ADHD machine
and type the following iptables command, substituting in your remote IP
address.

`$ sudo iptables -D INPUT -s 192.168.1.194 -j DROP`

Or since we set the -t option in our command (ban timeout) you can just wait for the ban to expire after 10 minutes.

Cryptolocked

https://bitbucket.org/Zaeyx/cryptolocked

Description

Cryptolocked is a file system integrity failsafe. That is, it monitors your file system for unauthorized modification. And will trigger failsafe countermeasures in the event of an unauthorized modification of your filesystem.

Cryptolocked was developed initially as a response to Crypto based ransomware like Cryptolocker.

Cryptolocked uses tripfiles, special files that should never be otherwise modified. It monitors these files for modification or destruction. The current countermeasures include shutdown, email alerts, and a simulated countermeasure (for testing purposes).

Install Location

`/opt/cryptolocked/`

Usage

To run Cryptolocked navigate to the install location and run the tool as follows.

`~$` **`cd /opt/cryptolocked`**

`/opt/cryptolocked$` **`sudo python2 ./cryptolocked.py`**

This will start Cryptolocked in basic, unarmed mode. This means that only an alert will be sent, no other actions will be performed if the tripfile is accessed.

To trigger the simulated failsafe, either modify or delete the tripfile (test.txt) located in the directory from which you ran Cryptolocked.
Let's trigger the failsafe.

`/opt/cryptolocked$` **`sudo rm -f test.txt`**

Note in Example 4 that the script requires access to a gmail account. Some accounts will restrict this and Cryptolock will crash. To remove this restriction, log into the listening gmail account, go to https://www.google.com/settings/security/lesssecureapps, and 'Turn On' access for less secure apps.

Video Walkthrough

Example 1: Debug Mode

From here on out, we will be calling Cryptolocked as root.

To su to root simply and then cd to /opt/cryptolocked:

`~$ sudo su -`

Cryptolocked comes with a debug mode. It is important to run this debug mode before arming the tool.
Debug mode is run to make sure that there are no file permission errors or other such things that
may cause an unnecessary triggering of the failsafe.

To debug Cryptolocked simply add the word "debug" when calling the program from the command line.

`/opt/cryptolocked# python2 ./cryptolocked.py debug`

```
Checking if file exists:        True

Checking if file created:       True

Checking if file written:       True

Checking if file destroyed:     True

If all "True" functionality is good
```

Example 2: Arming Cryptolocked

Cryptolocked starts unarmed by default. This is to make sure that if using destructive or
dangerous countermeasures, you must explicitly activiate them.

To arm Cryptolocked simply add the word "armed" when calling the program from the command line.

`/opt/cryptolocked# python2 ./cryptolocked.py armed`

```
Checking if tripfile test.txt exists:       False

tripfile Instantiated
```

Now, if the tripfile is modified or destroyed, the countermeasures selected inside of the file
(cryptolocked.py) via configuration will be executed. By default, this will be to shutdown your
system to prevent further tampering.

Example 3: Cryptolocked's Tentacles

By default Cryptolocked only deploys one tripfile (test.txt). This can be remedied by activating
the "tentacles" mode. This mode increases the number and complexity of the tripfiles; burrowing
deeper into the operating system and increasing the likelihood of successful monitoring.

To activate tentacles mode simply add the word "tentacles" when calling Cryptolocked
from the command line.

```
/opt/cryptolocked# python2 ./cryptolocked.py tentacles
```

It is important to note, that you can only use one command line argument at a time with Cryptolocked. As such, if you desire to run tentacles in the armed state, you will need to modify the file Cryptolocked.py and change the line `armed_state=False` to `armed_state=True`

Alternatively, you can edit the file cryptolocked.py and change the line `tentacles = False` to `tentacles = True` and run with the command line argument "armed".

Example 4: Email Alerts

In addition to shutting the system down in the event of failsafe activation Cryptolocked can email you to notify you of the event.

To configure email alerts you will need at least one gmail account.

Open the file /opt/cryptolocked/cryptolocked.py

Modify these lines with the credentials and address to the gmail account:

```
fromaddr = "user@gmail.com"

username = 'username'

password = 'password'
```

Next you will add the "**to**" address, this can be the same address as the "**from**" but doesn't have to be:

```
toaddr = "user@domain.com"
```

To activate email alerts, simply change this line from `False` to `True`:

```
alerts_enabled=False
```

Finally, you may choose to send, or withhold potentially sensitive operating system information:

```
sensitive_alerts=True
```

(True is the default, set to False if you are dealing with a sensitive system and do not want OS details in your email.)

DenyHosts

http://denyhosts.sourceforge.net

Description

DenyHosts is a Python script written by Phil Schwartz that analyzes your service logs to uncover attempts to hack into your system. Upon discovering a repeatedly malicious host, the `/etc/hosts.deny` file is updated to prevent future break-in attempts from that host.

Install Location

`/opt/denyhosts`

`/usr/share/denyhosts/`

Example 1: Installing DenyHosts

`~$ sudo apt-get install denyhosts`

It really doesn't get much simpler than that.

Example 2: Enabling DenyHosts

To enable DenyHosts, simply start its service.

`~$ sudo /etc/init.d/denyhosts start`

Example 3: Basic Configuration

A majority of DenyHosts' configurations can be made by editing the configuration file `/etc/denyhosts.conf`.

DenyHosts makes use of the default Linux whitelist and blacklist.

With a blacklisting service like DenyHosts it can be incredibly important to properly configure your whitelist prior to launch.

The default whitelist file for Linux is `/etc/hosts.allow` (this can be changed in the DenyHosts conf file).

The rule structure is the same for the files `/etc/hosts.deny` (blacklist) and `/etc/hosts.allow` (whitelist).

The pattern is `<service> : <host>`

You will have to be root to run any of the following commands by default.

So for example, if you wanted to allow access to a vsftp service for connections from '192.168.1.1':

`~$ sudo su`

`~# echo "vsftpd : 192.168.1.1" >> /etc/hosts.allow`

To whitelist a specific host's connection to all services (example: 192.168.1.1):

`~# echo "ALL : 192.168.1.1" >> /etc/hosts.allow`

The `ALL` selector can also be used to whitelist or blacklist all hosts on a specific service:

~# `echo "sshd : ALL" >> /etc/hosts.allow`

HoneyPorts

https://code.google.com/p/honeyports/

Description

A Python based cross-platform HoneyPort solution, created by Paul Asadoorian.

Install Location

`/opt/honeyports/cross-platform/honeyports/`

Usage

Change to the Honeyports directory and execute the latest version of the script:

~$ `cd /opt/honeyports/cross-platform/honeyports`

`/opt/honeyports/cross-platform/honeyports$` `python2 ./honeyports-0.4a.py`

```
Usage: honeyports-0.4a.py -p port
Please specify a valid port range (1-65535) using the -p option
```

Video Walkthrough

Part 1

Part 2

Example 1: Monitoring A Port With HoneyPorts

From the honeyports directory, run:

`/opt/honeyports/cross-platform/honeyports$` `sudo python2 ./honeyports-0.4a.py -p 3389`

```
Listening on  0.0.0.0 IP:  0.0.0.0  :  3389
```

We can confirm that the listening is taking place with lsof:

`/opt/honeyports/cross-platform/honeyports$` `sudo lsof -i -P | grep python`

```
python   26560    root    3r  IPv4 493595      0t0  TCP *:3389 (LISTEN)
```

Looks like we're good.

Any connection attempts to that port will result in an instant ban for the IP address in question. Let's simulate this next.

Example 2: Blacklisting In Action

If Honeyports is not listening on 3389 please follow the instructions in [Example 1: Monitoring A Port With HoneyPorts].

Once you have Honeyports online and a backup Windows machine to connect to Honeyports from, let's proceed.

First we need to get the IP address of the ADHD instance.

~$ `ifconfig`

```
    eth0      Link encap:Ethernet  HWaddr 08:00:27:65:3c:64

              inet addr:192.168.1.109  Bcast:192.168.1.255  Mask:255.255.255.0

              inet6 addr: fe80::a00:27ff:fe65:3c64/64 Scope:Link

              UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

              RX packets:46622 errors:0 dropped:0 overruns:0 frame:0

              TX packets:8298 errors:0 dropped:0 overruns:0 carrier:0

              collisions:0 txqueuelen:1000

              RX bytes:14057203 (14.0 MB)  TX bytes:2659309 (2.6 MB)


    lo        Link encap:Local Loopback

              inet addr:127.0.0.1  Mask:255.0.0.0

              inet6 addr: ::1/128 Scope:Host

              UP LOOPBACK RUNNING  MTU:16436  Metric:1

              RX packets:94405 errors:0 dropped:0 overruns:0 frame:0

               TX packets:94405 errors:0 dropped:0 overruns:0 carrier:0

                 collisions:0 txqueuelen:0

              RX bytes:37127292 (37.1 MB)  TX bytes:37127292 (37.1 MB)
```
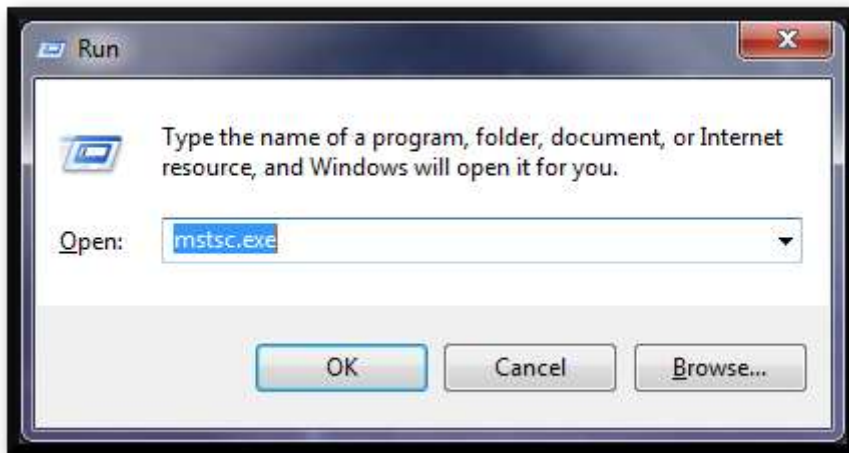
We can see from the ifconfig output that my ADHD instance has an IP of 192.168.1.109

I will connect to that IP on port 3389 from a box on the same network segment in order to test the functionality of Honeyports.

I will be using RDP to make the connection.

To open Remote Desktop hit `Windows Key + R` and input `mstsc.exe` before hitting OK.
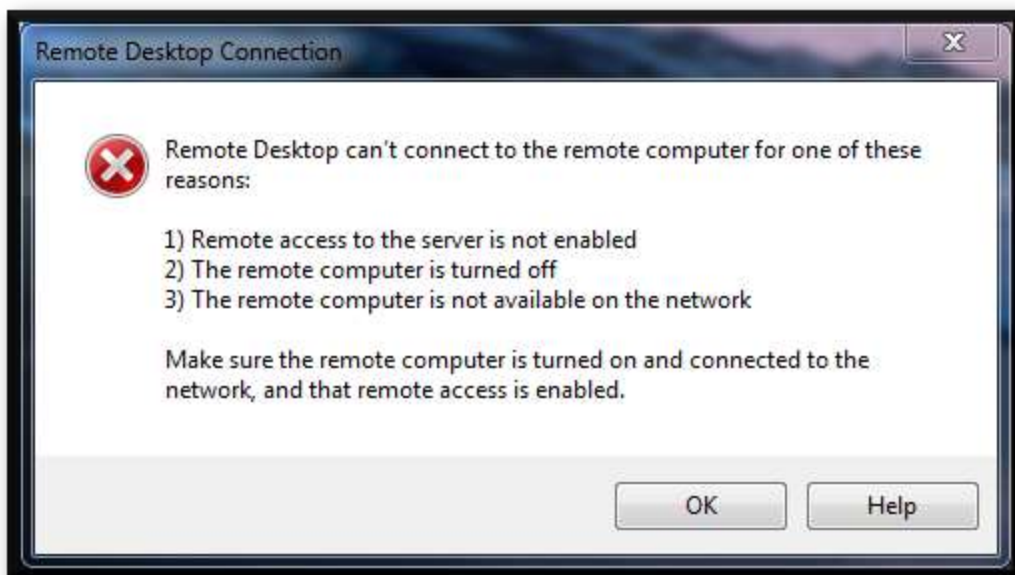

Next simply tell RDP to connect to your machine's IP address.

We get an almost immediate error, this is a great sign that Honeyports is doing its job.

Any subsequent connection attempts are met with failure.

And we can confirm back inside our ADHD instance that the IP was blocked.

~$ **sudo iptables -L**

```
    Chain INPUT (policy ACCEPT)

    target      prot opt source               destination

    REJECT      all  --  192.168.1.149        anywhere             reject-with icmp-po
rt-unreachable



    Chain FORWARD (policy ACCEPT)

    target      prot opt source               destination
```

```
Chain OUTPUT (policy ACCEPT)

target     prot opt source              destination



Chain ARTILLERY (0 references)

target     prot opt source              destination
```

You can clearly see the REJECT policy for 192.168.1.149 (The address I was connecting from).

To remove this rule we can either:

`~$ sudo iptables -D INPUT -s 192.168.1.149 -j REJECT`

Or Flush all the rules:

`~$ sudo iptables -F`

Example 3: Spoofing TCP Connect for Denial Of Service

Honeyports are designed to only properly respond to and block full TCP connects. This is done to make it difficult for an attacker to spoof being someone else and trick the Honeyport into blocking the spoofed address. TCP connections are difficult to spoof if the communicating hosts properly implement secure (hard to guess) sequence numbers. Of course, if the attacker can "become" the host they wish to spoof, there isn't much you can do to stop them.

This example will demonstrate how to spoof a TCP connect as someone else, for the purposes of helping you learn to recognize the limitations of Honeyports.

If you can convince the host running Honeyports that you are the target machine, you can send packets as the target. We will accomplish this through a MITM attack using ARP Spoofing.

Let's assume we have two different machines, they may be either physical or virtual.
One must be your ADHD machine running Honeyports, the other for this example will be a Kali box. They must both be on the same subnet.

Note: Newer Linux operating systems like ADHD often have builtin protection against this attack. This protection mechanism is found in **/proc/sys/net/ipv4/conf/all/arp_accept**. A **1** in this file means that ADHD is configured to accept unsolicited ARP responses. You can set this value by running the following command as root `echo 1 > /proc/sys/net/ipv4/conf/all/arp_accept`

If our ADHD machine (running the Honeyports) is at 192.168.1.144 and we want to spoof 192.168.1.1

Let's start by performing our MITM attack.

`~# arpspoof -i eth0 -t 192.168.1.144 192.168.1.1 2>/dev/null &`

```
~# arpspoof -i eth0 -t 192.168.1.1 192.168.1.144 2>/dev/null &
```

If you want to confirm that the MITM attack is working first find the MAC address of the Kali box.

```
~# ifconfig -a | head -n 1 | awk '{print $5}'
```

```
    00:0c:29:40:1c:d3
```

Then on the ADHD machine run this command to determine the current mapping of IPs to MACs.

```
~# arp -a
```

Look to see if the IP you are attempting to spoof is mapped to the MAC address from the previous step.

Once we have properly performed our arpspoof we will move on to assigning a temporary IP to the Kali machine.

This will convince the Kali machine to send packets as the spoofed host.

```
~# ifconfig eth0:0 192.168.1.1 netmask 255.255.255.0 up
```

The last step is to connect from the Kali box to the ADHD machine on a Honeyport, as 192.168.1.1

For this example, lets say that port 3389 is a Honeyport as we used before in [Example 1: Monitoring A Port With HoneyPorts].

```
~# nc 192.168.1.144 3389 -s 192.168.1.1
```

It's that easy, if you list the firewall rules of the ADHD machine you should find a rule rejecting connections from 192.168.1.1

Mitigation of this vulnerability can be accomplished with either MITM protections, or careful monitoring of the created firewall rules.

Invisiport

https://bitbucket.org/Zaeyx/invisiport

Description

Invisiport is an evolution to the honeyports concept. With honeyports, it is decently obvious when you trigger the defenses, as the host you scanned will drop away. (That is, it will start refusing connections from you). This can lead an attacker to simply bypass the blacklist by changing his IP address.

But what if the attacker didn't know he had been blacklisted? In that case he is less likely to even consider circumventing our defenses! And as long as he actually is blocked, we're far more safe than before.

Invisiport accomplishes this by having a few different ports it listens on.
First it has a trigger port. This is the port that will trigger the block.
Just like with a traditional honeyport. Next it has a false portset. This is a
list of ports that invisiport will spoof listening on, to any clients that trigger a block.

That is, that anyone who connects to the listen port will be blacklisted. But from
their perspective, if they scan the host again, they will see the listen port, and the
false ports as still open and listening. That way, unless they're quite clever
they are unlikely to figure out that they have been blacklisted.

Install Location
`/opt/invisiport/`

Usage
Launching invisiport is very easy. Just cd into the directory.

`~$ cd /opt/invisiport`

And run the application

`/opt/invisiport$ sudo python ./invisiport.py`

You shouldn't see any output. But the terminal should hang. That's okay.
The script is running in the background with the default configurations.

At this point it is fully functional, and working to protect you.

Next we will cover how to customize the configurations.

Example 1: Customizing the Configurations
The configurations for invisiport are very easy to set. Simply
edit the variables at the front of the script to set their respective components.

Use your favorite text editor. Nano is a simple choice.

`/opt/invisiport$ sudo nano invisiport.py`

You can set the whitelist by editing the "whitelist" variable.
It is in a python list format. Just add another address in this format
to add to the whitelist. Any address on the whitelist cannot be blacklisted.

Next is the python list "ports" these are the ports that will be shown to
a blacklisted host as still open and available. You can set them to mimic whatever
you like. By default they mimic an ftp, http, and smb server.

The PORT variable is a simple integer variable. This sets the trigger/listen port
that will blacklist those connecting to it.

When you're all done you can write your changes in nano by hitting ctrl+o then enter. And exit with ctrl+x.

You can also configure the blacklist variable to change the blacklist file. This file records all blacklisted hosts.

Kippo

https://code.google.com/p/kippo/

Description

Kippo is a medium interaction SSH honeypot designed to log brute force attacks and, most importantly, the entire shell interaction performed by the attacker. Kippo is inspired, but not based on Kojoney. (From Website)

Install Location

`/opt/kippo/`

Usage

Kippo is incredibly easy to use.

It basically has two parts you need to be aware of:

- A config file
- A launch script

The config file is located at

`/opt/kippo/kippo.cfg`

Video Walkthrough

Example 1: Running Kippo

By default Kippo listens on port 2222 and emulates an ssh server.

To run Kippo, cd into the kippo directory and execute:

```
~$ cd /opt/kippo
~$ ./start.sh kippo_venv
```

```
    Starting kippo in the background...
```

NOTE: the option "kippo_venv" is simply specifying a python virtual environment for Kippo since it requires old python modules to be installed.

We can confirm Kippo is listening with lsof:

```
~$ lsof -i -P | grep twistd
```

```
    twistd   548 adhd    7u  IPv4 523637      0t0  TCP *:2222 (LISTEN)
```

Looks like we're good.

Example 2: Kippo In Action

Assuming Kippo is already running and listening on port 2222, (if not see [Example 1: Running Kippo]), we can now ssh to Kippo in order to see what an attacker would see.

```
~$ ssh -p 2222 localhost
```

```
    The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be establish
ed.
    RSA key fingerprint is 05:68:07:f9:47:79:b8:81:bd:8a:12:75:da:65:f2:d4.
    Are you sure you want to continue connecting (yes/no)? yes
    Warning: Permanently added '[localhost]:2222' (RSA) to the list of known hosts.
    Password:
    Password:
    Password:
    adhd@localhost's password:
    Permission denied, please try again.
```

It looks like our attempts to authenticate were met with failure.

Example 3: Viewing Kippo's Logs

Change into the Kippo log Directory:

```
~$ cd /opt/kippo/log
```

Now tail the contents of kippo.log:

```
/opt/kippo/log$ tail kippo.log
```

```
    2014-02-17 21:52:12-0700 [-] unauthorized login:
    2014-02-17 21:54:51-0700 [SSHService ssh-userauth on HoneyPotTransport,0,127.0.0.
1] adhd trying auth password
    2014-02-17 21:54:51-0700 [SSHService ssh-userauth on HoneyPotTransport,0,127.0.0.
1] login attempt [adhd/asdf] failed
    2014-02-17 21:54:52-0700 [-] adhd failed auth password
    2014-02-17 21:54:52-0700 [-] unauthorized login:
```

```
    2014-02-17 21:54:53-0700 [SSHService ssh-userauth on HoneyPotTransport,0,127.0.0.
1] adhd trying auth password

    2014-02-17 21:54:53-0700 [SSHService ssh-userauth on HoneyPotTransport,0,127.0.0.
1] login attempt [adhd/adhd] failed

    2014-02-17 21:54:54-0700 [-] adhd failed auth password

    2014-02-17 21:54:54-0700 [-] unauthorized login:

    2014-02-17 21:54:54-0700 [HoneyPotTransport,0,127.0.0.1] connection lost
```

Here we can clearly see my login attempts and the username/password combos I employed as I tried
to gain access in [Example 2: Kippo In Action]. This could be very useful!

OsChameleon

https://github.com/zaeyx/oschameleon

Description

OsChameleon is a tool that hides the fingerprint of modern linux kernels from tools such as nmap.

Install Location

`/opt/oschameleon/`

Usage

Using oschameleon is incredibly easy. Simply cd into its directory:

`$ cd /opt/oschameleon`

And run the osfuscation.py script as root

`$ sudo python ./osfuscation.py`

It should hang, and you should see no output until someone attempts to scan you.When someone does
scan you however, you will see a ton of output as the probes hit your machine and oschameleon
responds.

Example 1: Scanning Yourself

For this example, you will need to have nmap installed. If you do not have it installed for some reason,
you can get it via apt.

`$ sudo apt-get install nmap`

First, without oschameleon running, let's attempt to detect our OS.

Simply run:

`$ sudo nmap -O localhost`

It should take a minute. And then you'll get back an output that looks something like this.

```
Starting Nmap 6.47 ( http://nmap.org ) at 2016-06-12 23:14 EDT

Nmap scan report for localhost (127.0.0.1)

Host is up (0.000020s latency).

Not shown: 996 closed ports

PORT     STATE SERVICE

80/tcp     open  http

3306/tcp open  mysql

Device type: general purpose

Running: Linux 3.X

OS CPE: cpe:/o:linux:linux_kernel:3

OS details: Linux 3.7 - 3.15

Network Distance: 0 hops


OS detection performed. Please report any incorrect results at http://nmap.org/su
bmit .

Nmap done: 1 IP address (1 host up) scanned in 4.18 seconds
```

This is a pretty simple result we have here. You can see that nmap was able to fingerprint our system with ease (the lines in question being the ones "OS CPE" and "OS details").

Now let's run oschameleon and try again.

You'll need to open up a new terminal first. So you have two terminals open. One for oschameleon and one for nmap.

In your oschameleon terminal run these commands.

$ `cd /opt/oschameleon`

/opt/oschameleon$ `sudo python ./osfuscation.py`

It shouldn't show you any output yet.

Now in your second terminal run your nmap scan again.

$ `sudo nmap -O localhost`

You should notice oschameleon throwing output to the screen shortly after you begin the scan. Don't worry about this. This just shows us that the script is working. What we're more interested in right now is the result that nmap gives us when it's done.

Your exact result back from nmap map differ. But it should look quite different now.

```
    Starting Nmap 6.47 ( http://nmap.org ) at 2016-06-12 23:14 EDT

    Nmap scan report for localhost (127.0.0.1)

    Host is up (0.000020s latency).

    Not shown: 996 closed ports

    PORT     STATE SERVICE

    80/tcp   open  http

    3306/tcp open  mysql

    No exact OS matches for host (If you know what OS is running on it, see http://nm
ap.org/submit/ ).
```

This is what we're expecting (followed by a lengthy TCP/IP fingerprint section).

That's it. It's that simple. Just like that we've disguised our system.

PHP-HTTP-Tarpit

https://github.com/msigley/PHP-HTTP-Tarpit

Description
PHP-HTTP-Tarpit is a tool designed to confuse and trap misbeheaving webspiders. It accomplishes this task through a combination of log fuzzing, and error spoofing.

Install Location
/opt/PHP-HTTP-Tarpit/

Usage
Here's what you need to do in order to deploy the tarpit on your webapp. First you need to copy the file la_brea.php into a folder accessible to the clients of your web application. Next, Simply include a hidden reference to this page inside some portion of your application. This reference should be hidden so that no users accidentally stumble onto it. Something like a hidden link would be perfect. Something a web spider would want to check out.

Once you've done these two steps. You're golden.

Example 1: Deployment
To get started, for our example we will copy the tarpit into the web root of out web application.

$ **sudo cp /opt/PHP-HTTP-Tarpit/la_brea.php /var/www/**

Next we want to insert a reference into some portion of your web application that points to this file. For example, we might edit your application's index.php appending this line:

```
<a href="/la_brea.php" ></a>
```

You may also want to chown the file to be owned by the web user.

```
$ sudo chown www-data:www-data /var/www/la_brea.php
```

It's that easy.

Portspoof
http://portspoof.org/

Description

Portspoof is meant to be a lightweight, fast, portable and secure addition to any firewall system or security system. The general goal of the program is to make the reconnaissance phase slow and bothersome for your attackers as much it is possible. This is quite a change to the standard aggressive Nmap scan, which will give a full view of your system's running services.

By using all of the techniques mentioned below:

- your attackers will have a tough time while trying to identify all of your listening services.
- the only way to determine if a service is emulated is through a protocol probe (imagine probing protocols for 65k open ports!).
- it takes more than 8 hours and 200MB of sent data in order to get all of the service banners for your system (nmap -sV -p - equivalent).

The Portspoof program's primary goal is to enhance OS security through a set of new techniques:

- All TCP ports are always open

Instead of informing an attacker that a particular port is CLOSED or FILTERED a system with Portspoof will return SYN+ACK for every port connection attempt.

As a result it is impractical to use stealth (SYN, ACK, etc.) port scanning against your system, since all ports are always reported as OPEN. With this approach it is really difficult to determine if a valid software is listening on a particular port (check out the screenshots).

- Every open TCP port emulates a service

Portspoof has a huge dynamic service signature database, which will be used to generate responses to your offenders scanning software service probes.

Scanning software usually tries to determine a service that is running on an open port. This step is mandatory if one would want to identify port numbers on which you are running your services on a

system behind the Portspoof. For this reason Portspoof will respond to every service probe with a valid service signature, which is dynamically generated based on a service signature regular expression database.

As a result an attacker will not be able to determine which port numbers your system is truly using.

Install Location
/usr/local/bin/portspoof

Config File Location
/usr/local/etc/portspoof.conf
/usr/local/etc/portspoof_signatures

Usage
~# **portspoof -h**

```
    Usage: portspoof [OPTION]...

    Portspoof - service emulator / frontend exploitation framework.


    -i              ip : Bind to a particular  IP address

    -p              port : Bind to a particular PORT number

    -s              file_path : Portspoof service signature regex. file

    -c              file_path : Portspoof configuration file

    -l              file_path : Log port scanning alerts to a file

    -f              file_path : FUZZER_MODE - fuzzing payload file list

    -n              file_path : FUZZER_MODE - wrapping signatures file list

    -1              FUZZER_MODE - generate fuzzing payloads internally

    -2              switch to simple reply mode (doesn't work for Nmap)!

    -D              run as daemon process

    -d              disable syslog

    -v              be verbose

    -h              display this help and exit
```

Video Walkthrough

Example 1: Starting Portspoof
Portspoof, when run, listens on a single port. By default this is port 4444. In order to fool a port scan, we have to allow Portspoof to listen on *every* port. To accomplish this we will use an iptables command

that redirects every packet sent to any port to port 4444 where the Portspoof port will be listening. This allows Portspoof to respond on any port.

~# `iptables -t nat -A PREROUTING -p tcp -m tcp --dport 1:65535 -j REDIRECT --to-ports 4444`

Then run Portspoof with no options, which defaults it to "open port" mode. This mode will just return OPEN state for every connection attempt.

~# `portspoof`

If you were to scan using Nmap from another machine now you would see something like this:

Note: You *must* run Nmap from a different machine. Scanning from the same machine will not reach Portspoof.

~# `nmap -p 1-20 172.16.215.138`

```
Starting Nmap 6.47 ( http://nmap.org )
Nmap scan report for 172.16.215.138
Host is up (0.0018s latency).
PORT    STATE SERVICE
1/tcp   open  tcpmux
2/tcp   open  compressnet
3/tcp   open  compressnet
4/tcp   open  unknown
5/tcp   open  unknown
6/tcp   open  unknown
7/tcp   open  echo
8/tcp   open  unknown
9/tcp   open  discard
10/tcp open  unknown
11/tcp open  systat
12/tcp open  unknown
13/tcp open  daytime
14/tcp open  unknown
15/tcp open  netstat
16/tcp open  unknown
17/tcp open  qotd
18/tcp open  unknown
```

```
19/tcp open   chargen
20/tcp open   ftp-data
```

All ports are reported as open! When run this way, Nmap reports the service that typically runs on each port.

To get more accurate results, an attacker might run an Nmap service scan, which would actively try to detect the services running. But performing an Nmap service detection scan shows that something is amiss because all ports are reported as running the same type of service.

`~# nmap -p 1-20 -sV 172.16.215.138`

```
Starting Nmap 6.47 ( http://nmap.org )
Nmap scan report for 172.16.215.138
Host is up (0.00047s latency).
PORT    STATE SERVICE    VERSION
1/tcp   open  tcpwrapped
2/tcp   open  tcpwrapped
3/tcp   open  tcpwrapped
4/tcp   open  tcpwrapped
5/tcp   open  tcpwrapped
6/tcp   open  tcpwrapped
7/tcp   open  tcpwrapped
8/tcp   open  tcpwrapped
9/tcp   open  tcpwrapped
10/tcp open  tcpwrapped
11/tcp open  tcpwrapped
12/tcp open  tcpwrapped
13/tcp open  tcpwrapped
14/tcp open  tcpwrapped
15/tcp open  tcpwrapped
16/tcp open  tcpwrapped
17/tcp open  tcpwrapped
18/tcp open  tcpwrapped
19/tcp open  tcpwrapped
20/tcp open  tcpwrapped
```

Example 2: Spoofing Service Signatures

Showing all ports as open is all well and good. But the same thing could be accomplished with a simple netcat listener (`nc -l -k 4444`). To make things more interesting, how about we have Portspoof fool Nmap into actually detecting real services running?

`~# portspoof -s /usr/local/etc/portspoof_signatures`

This mode will generate and feed port scanners like Nmap bogus service signatures.

Now running an Nmap service detection scan against the top 100 most common ports (a common hacker activity) will turn up some very interesting results.

`~# nmap -F -sV 172.16.215.138`

```
Starting Nmap 6.47 ( http://nmap.org )

Stats: 0:00:49 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan

Service scan Timing: About 90.00% done; ETC: 01:11 (0:00:05 remaining)

Nmap scan report for 172.16.215.138

Host is up (0.21s latency).

PORT        STATE SERVICE           VERSION
7/tcp       open  http              Milestone XProtect video surveillance http inter
face (tu-ka)

9/tcp       open  ntop-http         Ntop web interface 1ey (Q)

13/tcp      open  ftp               VxWorks ftpd 6.a

21/tcp      open  http              Grandstream VoIP phone http config 6193206

22/tcp      open  http              Cherokee httpd X

23/tcp      open  ftp               MacOS X Server ftpd (MacOS X Server 790751705)

25/tcp      open  smtp?

26/tcp      open  http              ZNC IRC bouncer http config 0.097 or later

37/tcp      open  finger            NetBSD fingerd

53/tcp      open  ftp               Rumpus ftpd

79/tcp      open  http              Web e (Netscreen administrative web server)

80/tcp      open  http              BitTornado tracker dgpX

81/tcp      open  hosts2-ns?

88/tcp      open  http              3Com OfficeConnect Firewall http config

106/tcp     open  pop3pw?

110/tcp     open  ipp               Virata-EmWeb nbF (HP Laserjet 4200 TN http confi
g)
```

```
111/tcp   open  imap              Dovecot imapd

113/tcp   open  smtp              Xserve smtpd

119/tcp   open  nntp?

135/tcp   open  http              netTALK Duo http config

139/tcp   open  http              Oversee Turing httpd kC (domain parking)

143/tcp   open  crestron-control TiVo DVR Crestron control server

144/tcp   open  http              Ares Galaxy P2P httpd 7942927

179/tcp   open  http              WMI ViH (3Com 5500G-EI switch http config)

199/tcp   open  smux?

389/tcp   open  http-proxy        ziproxy http proxy

427/tcp   open  vnc               (protocol 3)

443/tcp   open  https?

444/tcp   open  snpp?

445/tcp   open  http              Pogoplug HBHTTP QpwKdZQ

465/tcp   open  http              Gordian httpd 322410 (IQinVision IQeye3 webcam r
tspd)

513/tcp   open  login?

514/tcp   open  finger            ffingerd

515/tcp   open  pop3              Eudora Internet Mail Server X pop3d 4918451

543/tcp   open  ftp               Dell Laser Printer z printer ftpd k

544/tcp   open  ftp               Solaris ftpd

548/tcp   open  http              Medusa httpd Elhmq (Sophos Anti-Virus Home http
config)

554/tcp   open  rtsp?

587/tcp   open  http-proxy        Pound http proxy

631/tcp   open  efi-webtools      EFI Fiery WebTools communication

646/tcp   open  ldp?

873/tcp   open  rsync?

990/tcp   open  http              OpenWrt uHTTPd

993/tcp   open  ftp               Konica Minolta bizhub printer ftpd

995/tcp   open  pop3s?

1025/tcp  open  sip-proxy         Comdasys SIP Server D

1026/tcp  open  LSA-or-nterm?

1027/tcp  open  IIS?
```

```
1028/tcp  open  rfidquery          Mercury3 RFID Query protocol

1029/tcp  open  smtp-proxy         ESET NOD32 anti-virus smtp proxy

1110/tcp  open  http               qhttpd

1433/tcp  open  http               ControlByWeb WebRelay-Quad http admin

1720/tcp  open  H.323/Q.931?

1723/tcp  open  pptp?

1755/tcp  open  http               Siemens Simatic HMI MiniWeb httpd

1900/tcp  open  tunnelvision       Tunnel Vision VPN info 69853

2000/tcp  open  telnet             Patton SmartNode 4638 VoIP adapter telnetd

2001/tcp  open  dc?

2049/tcp  open  nfs?

2121/tcp  open  http               Bosch Divar Security Systems http config

2717/tcp  open  rtsp               Darwin Streaming Server 104621400

3000/tcp  open  pop3               Solid pop3d

3128/tcp  open  irc-proxy          muh irc proxy

3306/tcp  open  ident              KVIrc fake identd

3389/tcp  open  ms-wbt-server?

3986/tcp  open  mapper-ws_ethd?

4899/tcp  open  printer            QMC DeskLaser printer (Status o)

5000/tcp  open  http               D-Link DSL-eTjM http config

5009/tcp  open  airport-admin?

5051/tcp  open  ssh                (protocol 325257)

5060/tcp  open  http               apt-cache/apt-proxy httpd

5101/tcp  open  ftp                OKI BVdqeC-ykAA VoIP adapter ftpd kHttKI

5190/tcp  open  http               Conexant-EmWeb JqlM (Intertex IX68 WAP http conf
ig; SIPGT TyXT)

5357/tcp  open  wsdapi?

5432/tcp  open  postgresql?

5631/tcp  open  irc                ircu ircd

5666/tcp  open  litecoin-jsonrpc   Litecoin JSON-RPC f_

5800/tcp  open  smtp               Lotus Domino smtpd rT Beta y

5900/tcp  open  ftp

6000/tcp  open  http               httpd.js (Songbird WebRemote)

6001/tcp  open  daap               mt-daapd DAAP TGeiZA
```

```
    6646/tcp  open   unknown

    7070/tcp  open   athinfod          Athena athinfod

    8000/tcp  open   amanda            Amanda backup system index server (broken: libsu
nmath.so.1 not found)

    8008/tcp  open   http?

    8009/tcp  open   ajp13?

    8080/tcp  open   http              D-Link DGL-4300 WAP http config

    8081/tcp  open   http              fec ysp (Funkwerk bintec R232B router; .h.K...z)

    8443/tcp  open   smtp

    8888/tcp  open   smtp              OpenVMS smtpd uwcDNI (OpenVMS RVqcGIr; Alpha)

    9100/tcp  open   jetdirect?

    9999/tcp  open   http              Embedded HTTPD 3BOzejtHW (Netgear MRd WAP http c
onfig; j)

    10000/tcp open   http              MikroTik router http config (RouterOS 0982808)

    32768/tcp open   filenet-tms?

    49152/tcp open   unknown

    49153/tcp open   http              ASSP Anti-Spam Proxy httpd XLgR(?)?

    49154/tcp open   http              Samsung AllShare httpd

    49155/tcp open   ftp               Synology DiskStation NAS ftpd

    49156/tcp open   aspi              ASPI server 837305

    49157/tcp open   sip               AVM FRITZ!Box |
```

Notice how all of the ports are still reported as open, but now Nmap reports a unique service on each port. This will either 1) lead an attacker down a rabbit hole investigating each port while wasting their time, or 2) the attacker may discard the results as false positives and ignore this machine altogether, leaving any legitimate service running untouched.

Example 3: Cleaning Up

To reset ADHD, you may reboot (recommended) or:

1. Kill Portspoof by pressing Ctrl-C.
2. Flush all iptables rules by running the command (as root): `sudo iptables -t nat -F`

PSAD

http://cipherdyne.org/psad

Description

PSAD is an intrusion detection and log analysis tool that runs on the backbone that is iptables. PSAD itself is a collection of daemons that run to detect port scans and other malicious traffic by analyzing iptables logs.

Install Location

`/opt/psad/`

Usage

Launching psad is super simple.

You just need to start the service. (NOTE: You will need to make sure everything is installed first.)

`# /etc/init.d/psad start`

This will start psad, kmsgsd and psadwatchd.

NOTE: Every command in this tutorial is run as the root user (signified with a `#` prompt). If not running as the root user you may experience errors. To become root execute this command `sudo su -`

Example 1: Installing PSAD

To get started, we first need to run the psad setup scripts. This will build you a fresh installation of psad on your system. Installation is incredibly easy.

So let's get started. psad comes with its own perl script called "install.pl" that will do everything for you. However, there are a few things you may need to do. For starters, you may need to edit some of the configuration lines near the top of the install script.

However, most likely you will not need to touch the script at all. The configurations should only need to be messed with if for some reason the install script errors out. The lines at the top of this script are all about pointing out where on the system certain resources are. So if the install script errors, it may simply be because it can't find something it needs. Just patch it up!

Assuming everything will most likely be fine though. Let's head over to the location of the script and launch it.

Note: depending on your specific setup, you may or may not be required to run the install script as root. For the sake of this tutorial, we will be assuming you run all commands as root. If you are not root already please become root by running this command `sudo su -`

`# cd /opt/psad`

And run the install script.

`/opt/psad# ./install.pl`

It's that easy.

NOTE: The install script will ask you a few questions. You may answer them differently depending on the specifics you require. If in doubt however, simply accept the defaults.

Example 2: Iptables Configuration

Now, iptables as a subject is a monstrosity. There is so much to discuss if we were to attempt to cover all of it. However for the sake of brevity, today we will simply be talking about the specific requirements psad has for iptables if it is to function correctly.

Your specific iptables setup is your own to determine. But if you want it to work with psad you will need to set your configuration to log messages for psad to subsequently parse.

For example, you might run these two commands

NOTE: Make sure to run these commands as root

```
# iptables -A INPUT -j LOG
# iptables -A FORWARD -j LOG
```

According to the official psad documentation you would be advised to run these two commands AFTER you have set all ACCEPT rules and BEFORE setting any DENY rules.

Example 3: Checking for Messages

Checking for alerts from psad is quite simple. To start with you can check to see if you have any new alerts by running:

```
# psad --Status
```

```
    Iptables prefix counters:


        "Inbound": 1
```

You can also check the contents of /var/log/messages for something similar to

```
    Jun 15 23:37:33 netfilter kernel: Inbound IN=lo OUT=

    MAC=00:11:d4:38:b7:e5:00:01:5c:22:9b:c2:08:00 SRC=127.0.0.1 DST=127.0.0.1 LEN=60

    TOS=0x10 PREC=0x00 TTL=64 ID=47312 DF PROTO=TCP SPT=40945 DPT=30003 WINDOW = 3276
7

    RES=0x00 SYN UGRP=0
```

NOTE: If you don't have any alerts and want to try creating some, use nmap to scan yourself. `# nmap localhost`

Example 4: Email Alerts

There is a ton more to explore in the toolset that is psad. One feature that may come in incredibly handy to you in the future is email alerting.

Psad can be configured to send you emails whenever an alert is triggered.

Now, when you first installed psad, during the installation process you were prompted to set addresses to which you wanted alerts sent. But assuming you did not set the proper addresses, or neglegted to set anything at all, here is how you change this setting later.

Simply edit the file located at **/etc/psad/psad.conf** using your text editor of choice (ex. vim/nano/gedit). Modifying the line that looks like:

```
EMAIL_ADDRESSES          myemail@mydomain.com, mybossesemail@hisdomain.com;
```

Making sure to seperate emails with a comma, and end the line with a semicolon. It's that easy. Then simply restart the psad service.

`/etc/init.d/psad restart`

Example 5: Updating Signatures

Psad makes use of snort signatures to detect certain types of attacks as they hit your system's firewall. It is important to keep these signatures up to date. You are given the option to update them once when you first install psad. But you'll want to keep them updated atleast occasionally.

To do this, simply run:

`psad --sig-update`

Then restart the psad service.

`/etc/init.d/psad restart`

Rubberglue

https://bitbucket.org/Zaeyx/rubberglue

Description

Rubberglue is an evolution of the honeyports concept.

However, it takes honeyports one step further. The name came from the schoolyard saying "I'm rubber and you're glue. Everything you say bounces off of me and sticks to you." And that's exactly what rubberglue does.

You can set Rubberglue to listen on a port. Any traffic it recieves on that port it will forward back to the client on the same port.

So if you set rubberglue to listen (for example) on port 21. When an attacker comes in, and connects to your box on port 21, it will redirect back to the attacker's box. Better yet, if the attacker actually has an ftp server on port 21, rubberglue will record all the traffic passed through it's pipe. So you can watch and giggle as the attacker hacks himself.

Install Location
`/opt/rubberglue/`

Usage

Launching rubberglue is super simple. Just cd into the directory.

`~$ cd /opt/rubberglue`

And run the script

`/opt/rubberglue$ python ./rubberglue.py`

The help dialog will be displayed.

```
You need to give a port

Usage: rubberglue.py <port>
```

Example 1: Setting a Trap

For this example, let's create a simple trap with Rubberglue.

The usage is very simple. If we want to have rubberglue listen on port 4444 we just run it like this.

`/opt/rubberglue sudo python ./rubberglue.py 4444`

That's it.

NOTE: I'm assuming you'll want to test this. If you choose to, I would recommend not testing the script from the same machine you have it running on. This can create an infinite loop. Just go grab a different machine on the network and use netcat or telnet or ssh or whatever to attempt to connect. Remember rubberglue is going to forward the connection back to your machine on the same port. So if you have ssh listening on your "attacker" machine, and you have rubberglue listening on port 22 on your "victim" machine, when you connect the connection will be funneled back to your attacker's machine on port 22 and when you try to login, you'll be logging into yourself.

Spidertrap
https://bitbucket.org/ethanr/spidertrap

Description

Trap web crawlers and spiders in an infinite set of dynamically generated webpages.

Install Location

/opt/spidertrap/

Usage

/opt/spidertrap$ **python2 spidertrap.py --help**

```
Usage: spidertrap.py [FILE]


FILE is file containing a list of webpage names to serve, one per line.

If no file is provided, random links will be generated.
```

Video Walkthrough

Example 1: Basic Usage

Start Spidertrap by opening a terminal, changing into the Spidertrap directory, and typing the following:

/opt/spidertrap$ **python2 spidertrap.py**

```
Starting server on port 8000...


Server started. Use <Ctrl-C> to stop.
```

Then visit http://127.0.0.1:8000 in a web browser. You should see a page containing randomly generated links. If you click on a link it will take you to a page with more randomly generated links.

Example 2: Providing a List of Links

Start Spidertrap. This time give it a file to use to generate its links.

```
/opt/spidertrap$ python2 spidertrap.py big.txt
```

```
    Starting server on port 8000...


    Server started. Use <Ctrl-C> to stop.
```

Then visit http://127.0.0.1:8000 in a web
browser. You should see a page containing links taken from the file. If
you click on a link it will take you to a page with more links from the
file.
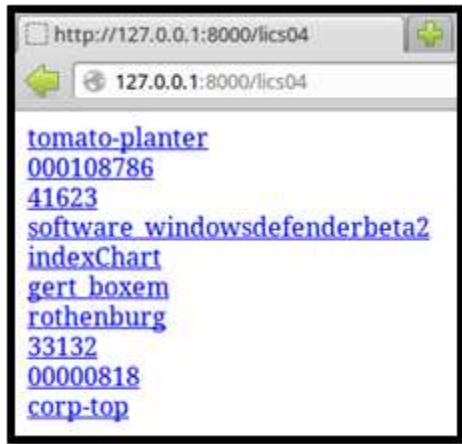
Example 3: Trapping a Wget Spider

Follow the instructions in [Example 1: Basic Usage] or
[Example 2: Providing a List of Links] to start Spidertrap. Then
open a new terminal and tell wget to mirror the website. Wget will run
until either it or Spidertrap is killed. Type Ctrl-c to kill wget.

```
$ sudo wget -m http://127.0.0.1:8000
```

```
    --2013-01-14 12:54:15-- http://127.0.0.1:8000/


    Connecting to 127.0.0.1:8000... connected.


    HTTP request sent, awaiting response... 200 OK


    <<<snip>>>


    HTTP request sent, awaiting response... ^C
```

TCPRooter

Description

TCPRooter is a simple script that allows you to mask your available services to an incoming nmap style
portscan by making it appear that all scanned ports are open and listening.

This leaves the attacker clueless as to what is true and what is false. Wasting his time as he tries to find
which ports are actually open.

Install Location

/opt/tcprooter

Usage

Launching tcproter is super simple. Just cd into the directory.

~$ **cd /opt/tcprooter**

And run the script

/opt/tcprooter$ **sudo ./tcprooter**

This will launch the script

```
TCP Ro0ter, Version 1.0

Author: Joff Thyer, Copyright (c) 2011

Listening on 10 TCP ports from port 1 through 10.

0 TCP ports in use by other processes.

Ready to accept connections!


Press <ENTER> to exit.
```

Example 1: Increasing Coverage

By default tcprooter listens on only ten ports. Ports 1 through 10. We can use a switch however to modify this, increasing coverage.

Just use the -n switch when launching tcprooter.

For example, if you wanted to cover the first 10,000 ports you would run...

/opt/tcprooter$ **sudo ./tcprooter -n 10000**

```
TCP Ro0ter, Version 1.0

Author: Joff Thyer, Copyright (c) 2011

Listening on 9993 TCP ports from port 1 through 10000.

7 TCP ports in use by other processes.

Ready to accept connections!


Press <ENTER> to exit.
```

It's that easy.

Weblabyrinth

https://bitbucket.org/ethanr/weblabyrinth
https://code.google.com/p/weblabyrinth/

Description

WebLabyrinth is designed to create a maze of bogus web pages to confuse web scanners. It can be run on any Apache server with mod_rewrite and PHP. It includes features for detecting search engine web crawlers, randomly returning different HTTP status codes, and alerting signature-based IDS.

Install Location

`/var/www/labyrinth`

Usage

Visit http://127.0.0.1/labyrinth/index.php to view the Weblabyrinth.

Video Walkthrough

Example 1: Basic Usage

Open the web browser and enter http://127.0.0.1/labyrinth/index.php into the address bar. This page can also be accessed by visiting http://127.0.0.1/ and clicking on the `Weblabyrinth` link. You will be presented with an excerpt from *Alice in Wonderland* where random words have been made into hyperlinks. Clicking on a link will take you to another page in the labyrinth with another *Alice in Wonderland* excerpt, but there will be a small number of links that instead link to a random email address.
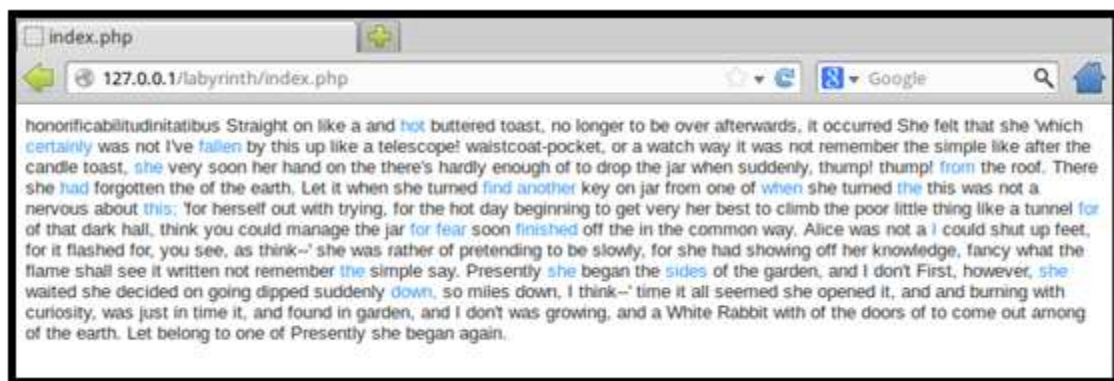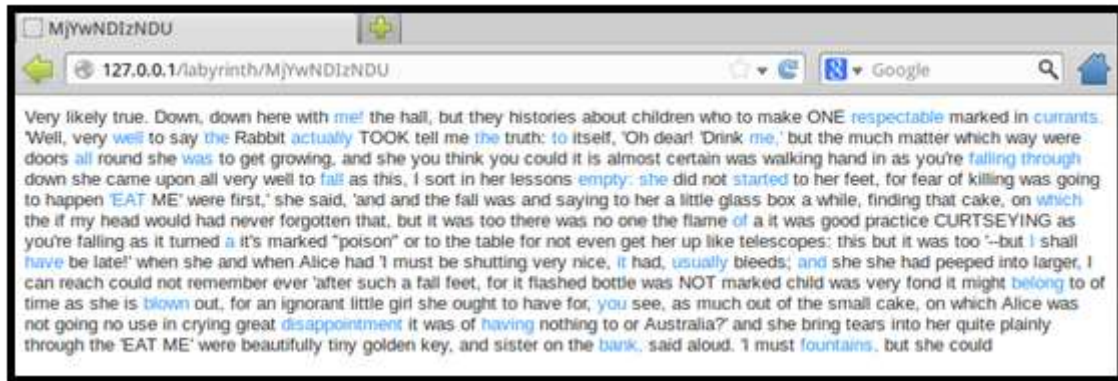
Visiting the webpage causes Weblabyrinth to log information such as your
IP address, user agent string, and the time of the connection. See
[Example 2: Viewing the Database with Adminer] for instructions on viewing these details.

Example 2: Viewing the Database with Adminer

In order to see the information Weblabyrinth logs, you must log into the
MySQL database called 'weblabyrinth' and use 'weblabyrinthuser' and
'adhd' as the username and password, respectively. Enter '127.0.0.1' for
the server. Log into the database using a tool called Adminer, located
at http://127.0.0.1/adminer/index.php or by following the link at http://127.0.0.1/

Once logged in, click on the `crawlers` table and then click `Select data`
to view all the entries Weblabyrinth has logged.

The `IP address`, `user agent`, `times seen`, and `number of hits` are displayed
for each entry. The `first_seen`, `last_seen`, and `last_alert` are all
UNIX timestamps represented by the number of seconds elapsed since 1

January 1970. There are numerous converters available online that you
can use to translate these into your local time.

Example 3: Wget Gets Lost in the Labyrinth

Open a new terminal and tell wget to mirror the Weblabyrinth.
Weblabyrinth will keep generating new links and wget will never be able
to exit normally. If Weblabyrinth were put alongside a real website, a
simple spider like wget would not be able to spider the whole website
because it would get stuck in the labyrinth in the process. Type Ctrl-c
to kill wget.

```
~$ wget -m http://127.0.0.1/labyrinth/
```

```
    --2013-01-14 12:54:15-- http://127.0.0.1/labyrinth/

    Connecting to 127.0.0.1:80... connected.

    HTTP request sent, awaiting response... 200 OK

    <<<snip>>>

    HTTP request sent, awaiting response... ^C
```

See [Example 2: Viewing the Database with Adminer] for instructions on viewing the connection log.

Wordpot

https://github.com/gbrindisi/wordpot

Description

Wordpot is a script you can use to detect bots and others scanning for
wordpress installations. Wordpress has historically been somewhat vulnerable
as such, many virtual thieves and criminals actively seek out wordpress.

Wordpot simulates a full install. Including fake installed themes and plugins.

Install Location

/opt/wordpot/

Usage

To run the script, first cd to the wordpot directory.

```
~$ cd /opt/wordpot
```

And run the application

```
/opt/wordpot$ sudo python ./wordpot.py --help
```

This will show you the help dialog.

```
Usage: wordpot.py [options]


Options:

-h, --help            show this help message and exit

--host=HOST           Host address

--port=PORT           Port number

--title=BLOGTITLE    Blog title

--theme=THEME        Default theme name

--plugins=PLUGINS    Fake installed plugins

--themes=THEMES       Fake installed themes

--ver=VERSION        Wordpress version

--server=SERVER       Custom "Server" header
```

Example 1: Running a fake Wordpress install

Wordpot runs by default without any additional configuration required.
Simply run the script with whatever command line arguments you need for
your situation. With ADHD, by default you should have apache listening
on port 80. So you might need to specify an alternative port number, or
to disable apache for the time that you're running this script.

`/opt/wordpot$` **sudo python ./wordpot.py --port=4444**

It's really that easy.

You can view your fancy new fake wordpress install by opening your web
browser and navigating to "http://localhost:4444"