

# Intrusion Detection Honeypots on a SBC.

CHRIS "RATTIS" JENKS

#MISEC SOUTHFIELD – 2025-03-13

## Today's talk

- The problem I was trying to solve.
  - The constraints I had to work in.
- Quick Honeypot History
- OpenCanary
  - Why I couldn't go with pre-built
- Intrusion Detection Honeypots
  - No interaction option

We're going to be covering a few different topics. The initial problem of seeing internal traffic that could be malicious, the constraints I had to work in, some of the history of honeypots, OpenCanary and why it wouldn't have worked for me, and what I came up with to solve my original problem.

I'm also probably going to use some terms in way that most IT and Cybersecurity people haven't heard them before. It's because of my belief in Factor Analysis of Information Risk framework and their terminology which uses the terms in the same way as the insurance industry, and before IT borrowed the words and changed the meaning.



# The itch I needed to scratch

ORIGINAL NEED AND CONSTRAINTS

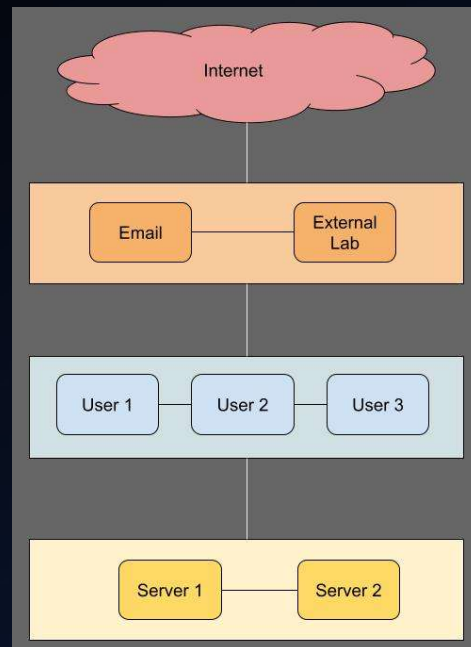
It's always that one spot that you can't quite reach with your current tools that sends you off to build a better back scratcher.

## ORIGINAL NEED

Detect potentially malicious “East – West” traffic on a multi-segment network.

Are we missing malicious internal traffic?

We had “North – South” visibility.



Once upon a time, the fine Senior SOC manger... No ok, really. I don't remember if it was in a daily handover or an all-hands meeting. But either way the Senior manager pointed out we could see traffic between zones, and traffic going to the internet. But we lacked visibility for potential malicious traffic between devices in a zone. I think this came up a few months after Not Petya.

The request was to find something that would work to monitor east west traffic. I thought let's do an internal honeypot. It won't be perfect, but it'll catch some of the potentially malicious actions by threats moving through the network. It won't see all the traffic, just the traffic reaching out to the honeypot, which wouldn't be common knowledge. Any traffic to it would have to be investigated.

## Constraints and Limitations


- Legal considered honeypots as entrapment devices.
  - I blame FIPS 39 (1976) and bad education / outdated views.
  - Lawyers insisted Honeypots were entrapment and we couldn't do that.
- The Proof of concept had to be cheap.
- Had to be Limited Handshake to Zero Interaction
  - The lawyers said interaction would be entrapment.

The response was “No, we can't have honeypots. Legal won't allow us to have entrapment devices.” Until I made this slide deck, I didn't know what they meant by that. We're talking poison not venom. The threat shouldn't be on the network and if they brush up against the device that's on them. We're not broadcasting it as something to look at and then injecting them when they get nearby, like a snake lashing out. (John Strand does a better job of explaining this in the Antispyhon's Active Defense and Cyber Deception pay what you can class). I was arguing this is more like a security camera in a hallway, or a silent alarm on a door.

The department budget was always getting cut, and we ran lean. Which is odd considering where I worked and how large they were; but that's how they ran it. Lean and tight. So, what ever I came up with as a POC had to be close to free.

Back to the entrapment thing, we had to set it up so the attacker if

they did touch it didn't want to be "enticed" to attack it.



# Quick History of Honeypots

NOT IN-DEPTH

## Before Honeypots were called Honeypots

- FIPS 39 (1976).
  - “The deliberate planting of apparent flaws in a system for the purpose of detecting attempted penetrations or confusing an intruder about which flaws to exploit.”
- Cliff Stoll
  - Fake Government Documents
- Bill Cheswick
  - Funky Unix Jail

As mentioned above, legal appeared to be better prepared than we were in the SOC for arguing for these devices being called entrapment devices. I never came across FIPS 39 until I was researching the history of honeypots. But the arguments that legal were making at the time, from what I remember, tied in to the enticing the user with a flawed system. Legal saw that as entrapment and wanted nothing to do with any tools like. Some days I wonder how I was allowed to use Canary tokens in documents we sent to threats, to try and figure out the threat's location.

The first time I really remember coming across honeypots was reading Cliff Stoll's book about what he did in the 80s. For those that don't know the Cuckoo's Egg, the computer network he was on was being used without paying for the time used. It turned out to be espionage based. To keep the threat actor online enough to trace the phone call in Germany, he created fake government documents to entice the threat to stay connected.



The next version of a honeypot I came across was when I was reading *Firewalls and Internet Security* by Bill Cheswick. In it he talks about creating a Unix Jail in a server to mess with a threat who accessed the system through an exploit. He made several choices in the interactions with the threat, to get more information and share what he was seeing in the logs.

# Honeypots

## TERM COINED BY LANCE SPITZNER – BUILD A HONEYPOT

- Deception Toolkit
- CyberCop Sting
- BackOfficer Friendly
- The HoneyNet Project
- HoneyD
- Labrea Tarpit



Image Credit : Peter Breunig, Goethe University Frankfurt

In both Cliff Stoll and Bill Cheswick's intrusion events, they didn't have a name for what they were doing. The name came about later, first seen in Lance Spitzner's white paper "Build a HoneyPot". At least that is the oldest known reference to the word honey pot being used in a computer security context. Prior to that, it was about an attractive person being used to get a possible asset in a compromising position.

The list in the slide is different honeypots over the course of time. The first 3 came out around the same time Spitzner was coining the term.

That photo was tied to an article about the oldest surviving honey pot being about 3500 years old. I just found it interesting (but my first degree was focused on Anthropology/Archaeology).

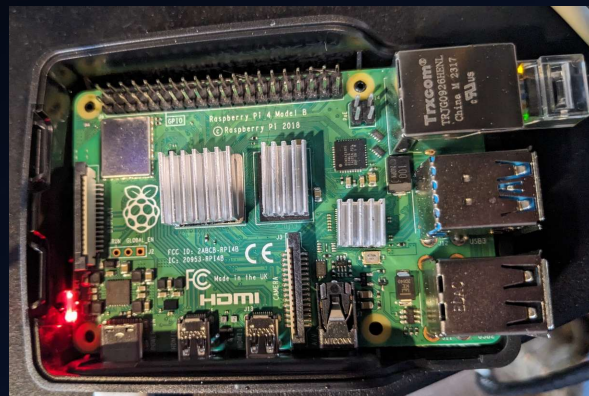
# OpenCanary

LOW INTERACTION HONEYPOT – NICE  
BUT VIOLATED MY PRIMARY  
CONSTRAINT

This is a great honeypot; but I couldn't use it. I first came across it in Chris Sanders's book *Intrusion Detection Honeypots*. Even though we were using other Thinkst products at the time, this was a no go with management. They said it fell under the entrapment argument of legal. Mainly because it was giving the threat a login screen to try and login to.

## “Multi-protocol network honeypot”

- Owned by Thinkst
- Available for free from GitHub
- Low Interaction Honeypot
- Can run on a Raspberry Pi
- Breaks the no-interaction rule
- Also needs rsyslog installed (Debian based, including the Pi)

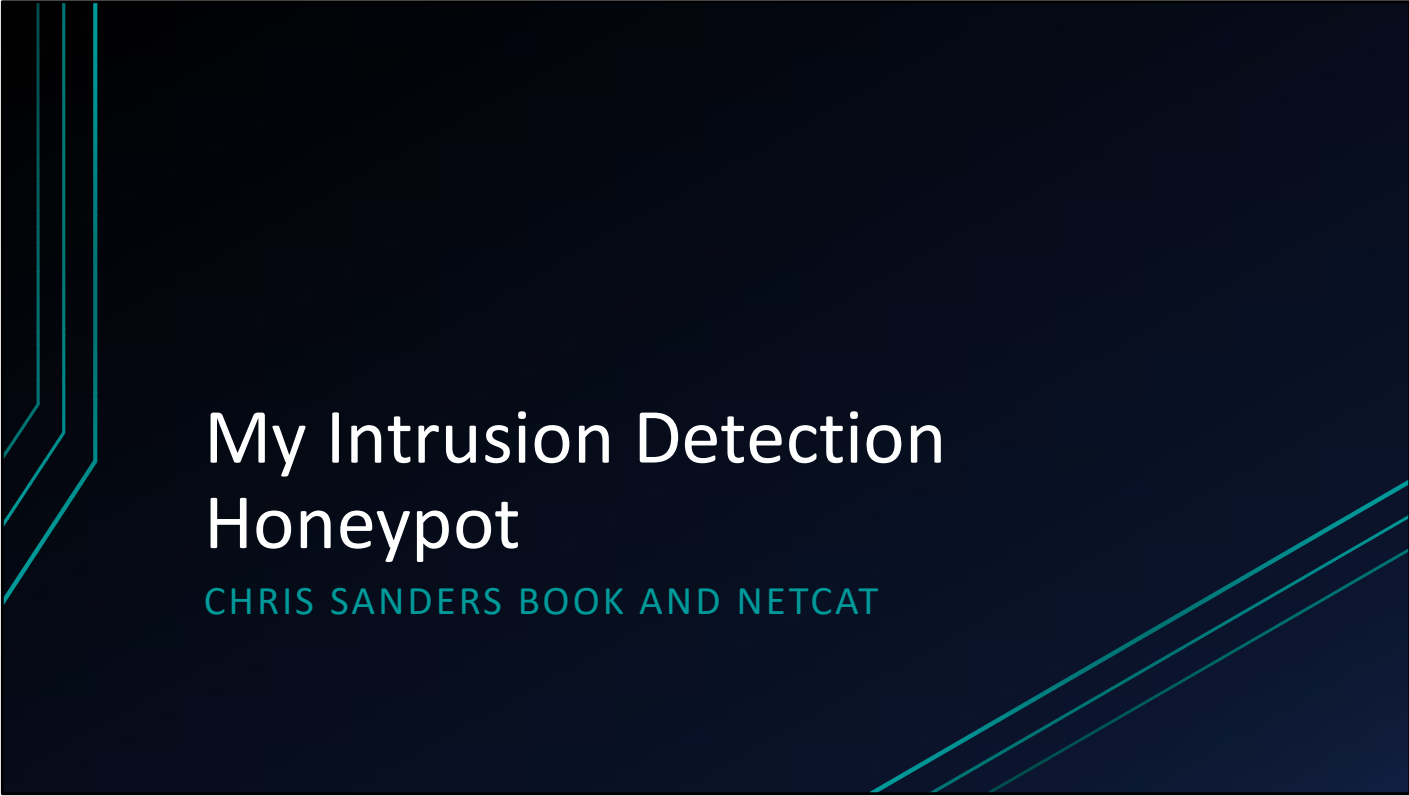


I had forgotten about this, until I was talking to the people at the Thinkst boot at the Gartner Risk summit last year (2024) in DC. I was talking about my NetCat based honeypot and what I was trying to do. The person I was talking to said to check out their free product to fit my need.

## What Services/Servers does it provide?

- SSH
- FTP
- GIT
- HTTP
- HTTP Proxy
- MSSQL
- MYSQL
- Telnet
- SNMP
- SIP
- VNC
- Redis
- TFTP
- NTP
- SMB (Samba with samba installed)

I like the different type of services that the OpenCanary emulate, but it looks like a server. So beyond breaking the no interaction rule, like the last slide said this also doesn't do a good job as looking like a user device in a user vlan / or segment of the network.

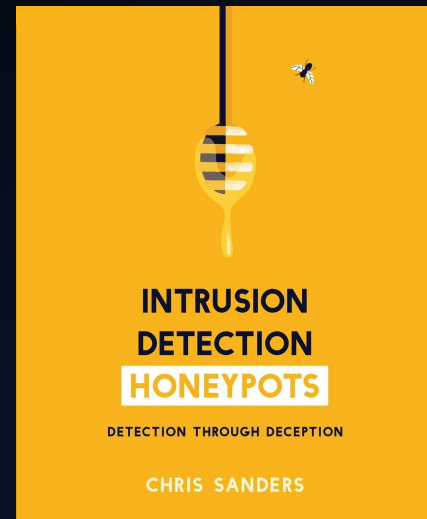
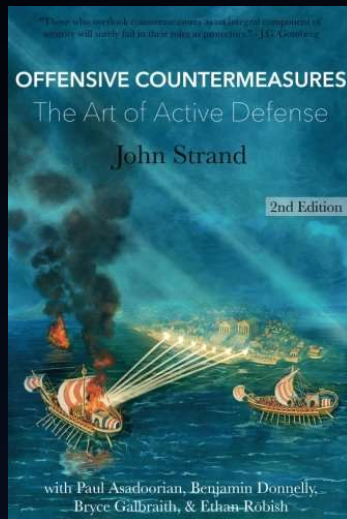


# My Intrusion Detection Honeypot

CHRIS SANDERS BOOK AND NETCAT

But let's get back to my issue, I still had that itch that needed scratching.

## The start of an Idea



Refining my idea, still using a honeypot and not calling it one, started when I took the Antisyphon course on Active Defense and Cyber Deception. I was trying to use the things I learned there to push back on the legal argument about entrapment and enticement. The course goes with the Offensive Counter measures book really well.

But the real key was when I read Intrusion Detection Honeypots. While it talked about OpenCanary and some other options, it also showed how to use NetCat to do the same job. You could build a persistent netcat listener, and have it return data based on what port the threat was hitting. Which is when I thought, if I can choose to not return a web page...

## What Service/Services can it run

- Netcat Listener / None, but can monitor for anything
  - The device is more a honey port device
- Know what is normal looking in your VLAN/Network
  - NMAP or other scanning – Make it look like any other expected device
- Zero Interaction (can be low interaction if you want)
  - Listener with no data returned to potential malicious traffic
- Catch extra log data with iptables

I could have NetCat listen on any port I wanted. I could set it up to log the connection to a file (adding a timestamp was something I got help from the Active Countermeasures discord server). But the best part, this was more like a video camera or a door alarm in the guard booth. It has open ports, but no response beyond a handshake. The listener isn't responding with data. There is nothing to interact with.

So, I want a device that looks like it belongs in the segment. Great, use nmap to scan the vlan and set a script up to open the ports of vlan devices to blend in.

Iptables came later, due to NetCat not logging nmap scans. I recalled that it used to in one of the iterations of the device. But I also changed from the Original NetCat, to Traditional NetCat, to BSD-netcat, to the nmap rewrite ncat.





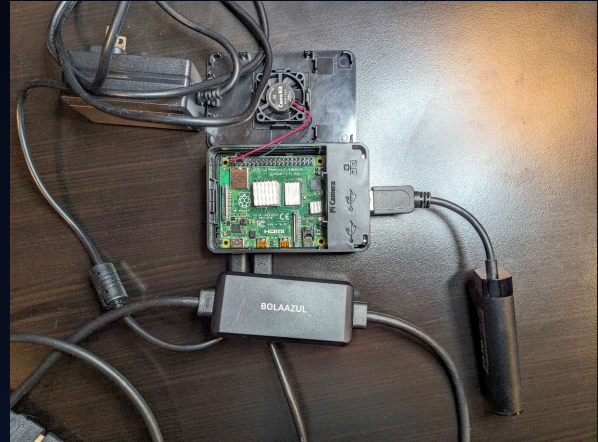
# Building my IDH

CHOICES

The next several slides, are meant as part walk through, part presentation. We're going to do it live. But I'll walk through my thoughts and steps on building and why I made some choices that I did.

## Hardware

- Virtual Machine
- Mini-pc
- Raspberry Pi
  - USB to Ethernet Optional (highly recommended)
  - POE (Optional)
  - HDMI to Display port converter (Optional)



The very first version of this was on a Virtual Machine on my work laptop. The problem was, I took that home with me. So potentially malicious traffic when I wasn't at work was missed.

I had some Raspberry Pis sitting around, so I used it. I got some extra parts for the last place I was at. All the monitors were dell dvi, so I took the hdmi-mini to HDMI convert, and searched for an HDMI to DisplayPort converter. The one I Got seemed to work well for what I needed.

Because Ncat can be set up to listen on an IP address. It's possible to use the USB to Ethernet as a management port and use that for SSH on that interface.

Limiting the actual SSH port via iptables is recommended too.

## NMAP to blend in

- `sudo nmap -T4 -Pn -n -p- -oA net_scan X.X.X.X/Y`
  - Gives multiple file formats
- `grep -o '[0-9]*' net_scan.txt | sort | uniq -c | sort -rn`
  - Get a list of the most used ports on the network (see next slide)
  - These will be used to build out the ports on the honeypot

The first step in setup is to blind in. To do so, you need to know what the network segment looks like. This goes back to running the nmap scan, on the network. I've learned over the years not to trust the deployment documentation, because the OS vendors like to make changes. Like adding port 7680 to make Windows Update less bandwidth intensive. The nmap scan also gives us a great time to update deployment documentation, if we have a say in that.

After scanning, grep the ports, and see what the top 10 or so ports are in the environment. Use those for the nmap listener.

## Example of open ports

```
(chrisj@kali-purple)-[~]
└─$ grep -o "[0-9]*" netscan_ports.txt | sort | uniq -c | sort -rn
15 22
 7 8009
 7 80
 7 443
 6 8443
 5 9000
 5 8012
 5 8008
 5 10101
 5 10001
 4 10007
 4 10005
 4 10002
 3 445
 3 3493
 3 139
 2 9080
 2 8080
 2 60000
 2 55443
 2 55442
 2 5357
 2 53
 2 5040
 2 49668
 2 3389
 2 135
```

This is what my home network looks like. Which reminds me, I need to redo all the vlans. For the presentations demonstration I'm going to run this on ports 22, 80, 443, 8443, 8009, 445, 139, and 135.

## Build slide 1

- Not a deployable GitHub. But steps here: <https://github.com/crattis/idh-nc>
- Install a headless system (Debian net-inst, Raspberry Pi OS Lite (64-bit))
- Hide the hardware type by changing the mac address
  - Remember the blending in slide, what OUI shows up the most?
- Name so it blends in (match local naming conventions).
- Upgrades
- Install ncat, rsyslog, nmap, iptables, tmux, byobu, moreutils, lsof

I've put my script, and the auxiliary scripts on my GitHub page. This is not a cloneable repo, unless you want to do some post clone clean up.

In my case I'm setting up on a headless system running on a raspberry pi. When I build these on bare metal or in Virtual Machines, I use the Debian netinst iso image. I've used Debian since the 90s and like it more than other distros for different reasons. I think it has a better package management system for dealing with dependancies. It doesn't break functionality by forcing snaps or flatpaks that are isolated from each other (there is a time and place but not everything should be done that way).

Because we want to try and blend in, we'll want to re-run grep on the nmap output see what the standard hardware addresses are for the devices. While in this case the my pi uses a Pi mac address, my OpenCanary uses a Dell mac address.

Try to figure out the naming convention of the environment. For the Demo, I just used raspi-idh, but a lot of my network devices have raspi-<something> has their name. raspi-nas, raspi-hole, raspi-sensor, etc.

Since I used the Rasperry Pi imager to install the Raspberry Pi OS Lite (64-bit) image, I needed to do updates after install. Once the updates are done, I had to install other packages. I used nmaps version of netcat, rsyslog (for logging reasons), nmap for scanning, iptables for capturing traffic to non-NetCat ports, Byobu and Tmux for screen management, moreutils for ts to do timestamps. The downside to using LITE installs is some utilities may be missing.

## Build slide 2

- Install the script under `/opt/honeypot`
  - Update `port_array=(<num> <num> <num>)` in `honeypot.sh` for your ports
  - `chmod 755 /opt/honeypot/honeypot.sh`
- `mkdir /var/log/honeypot`
  - Set owner to `root:adm`
  - Change permissions to `775 (rwx,rwx,rx)`

Once you're ready to go / installs are complete. Copy the `honeypot.sh` scrip to `/opt/honeypot/honeypot.sh`. You can use other locations, like `/usr/local/bin`, but I got in the habit of putting things in `opt` a long time installed, never changed. Also, while this says `.sh` at the end, it's a Bash Script, taking advantage of some of the bash built in features. If your system doesn't have bash installed you may have to make some modifications.

While copying the script to the directory you'll need to set the array items for the script. Using the ports from my lab example, the array would `port_array=(22 80 443 8443 8009 445 139 135)`.

Once the file written change the permissions. The file and directory in my builds are `root:root`, and I set the permissions to owner: read, write, execute; group: read, execute, everyone else: read, execute (`rwxr-xr-x`). I never understood the +-permissions, the numbers just make more sense to me. But that was how I learned in college in the

90s.

Create and set the `/var/log/honeypot` to `root:adm` with the permission as `775 (rwxrwxr-x)`. You may need to do this again after NetCat creates the files.



## Build 3 – Testing before daemon

- `sudo /opt/honeypot/honeypot.sh # to open reserved ports`
- `cat, less, or tail /var/log/honeypot/<port number log>.log` or all logs
- `sudo pkill -9 ncat`
- `sudo rm /var/log/honeypot/*.log`

Test and tweak. I don't think I've had a build yet where testing and tweaking hasn't been an issue. This is also where I use byobu/tmux to split the full screen ssh connection (or the local terminal screen connection when I add a kvm). One side I run the honeypot command with sudo, and tail the logs the other frame I use to modify the script, install missing packages, kill the listeners (they don't quit when you ctrl-c in the run frame).

If you're running from the command line, it will sometimes show you the strings the threat uses. It was fun watching and telling the pen-test team what commands they were sending to my listener that would never reply to them.

## System Testing

```
pi@raspi-1dh:~$ sudo /opt/honeypot/honeypot.sh && tail -f /var/log/honeypot/*.log
==> /var/log/honeypot/135.log <==
2025-03-06 18:43:00:-0500 listner started

==> /var/log/honeypot/139.log <==
2025-03-06 18:43:00:-0500 listner started

==> /var/log/honeypot/22.log <==
2025-03-06 18:43:00:-0500 listner started

==> /var/log/honeypot/443.log <==
2025-03-06 18:43:00:-0500 listner started

==> /var/log/honeypot/445.log <==
2025-03-06 18:43:00:-0500 listner started

==> /var/log/honeypot/8009.log <==
2025-03-06 18:43:00:-0500 listner started

==> /var/log/honeypot/80.log <==
2025-03-06 18:43:00:-0500 listner started

==> /var/log/honeypot/8443.log <==
2025-03-06 18:43:00:-0500 listner started

==> /var/log/honeypot/8009.log <==
2025-03-06 18:43:00-0500 Ncat: Version 7.93 ( https://nmap.org/ncat )
2025-03-06 18:43:00-0500 Ncat: Listening on :::8009
2025-03-06 18:43:00-0500 Ncat: Listening on 0.0.0.0:8009
```

Example output of the log when testing. In this example, the “start of script” command is working, but the listener wasn’t listening on some ports. In this case 8009 was faster to start than the other ones, and ncat wrote the data below.

## Make it Auto Start

### **idh-nc.service**

#### [Unit]

Description=NetCat based Intrusion Detection Honeypot

ConditionPathExists=/opt/honeypot/honeypot.sh

#### [Service]

User=root

RemainAfterExit=yes

ExecStart=/opt/honeypot/honeypot.sh

#### [Install]

WantedBy=multi-user.target

If you want this to run like a daemon, note this will run ncat as root but I was ok with that because it's not responding to requests, create a file called idh-nc.service under /etc/systemd/system/idh-nc.service. Then systemctl enable idh-nc.service

## Capture other connection attempts - iptables

```
iptables -I INPUT -p tcp -m state --state NEW -m limit --limit 6/min -j LOG --log-prefix='[iptables_inbound]' --log-level 4
```

```
ip6tables -I INPUT -p tcp -m state --state NEW -m limit --limit 6/min -j LOG --log-prefix='[ip6tables_inbound]' --log-level 4
```

```
iptables-save -f /etc/iptables/rules.v4
```

```
ip6tables-save -f /etc/iptables/rules.v6
```

```
sudo apt install iptables-persistence
```

To capture other data like nmap scans that aren't being captured by the logging of the NetCat listener, use iptables. I know nftables replaced iptables. But my iptables commands still work, and I haven't learned nftables yet.

Create 2 rules. One for ipv4 and one for ipv6. This will monitor and log network connections and log them to `/var/log/syslog` with the string `iptables_inbound` or `ip6tables_inbound`.

After the rules are created, save them to create the restore file. Then install `iptables-persistence`. `Iptables-persistence` looks for the saved rules when installed and will provide errors if those files are not found. `Iptables-save` and `iptables-restore` are part of the `iptables` package.

## Seeing Data

- See real-time honeypot logs:
  - `tail -f /var/log/honeypot/*`
- See triggered firewall logs:
  - `sudo grep -E "ip6?tables" /var/log/syslog`

## Seeing entries in Honeypot Logs

```
==> /var/log/honeypot/80.log <==
2025-03-09 15:54:00-0400 Ncat: Version 7.93 ( https://nmap.org/ncat )
2025-03-09 15:54:00-0400 Ncat: Listening on :::80
2025-03-09 15:54:00-0400 Ncat: Listening on 0.0.0.0:80
2025-03-09 16:01:44:-0400 listner started
2025-03-09 16:01:45-0400 Ncat: Version 7.93 ( https://nmap.org/ncat )
2025-03-09 16:01:45-0400 Ncat: Listening on :::80
2025-03-09 16:01:45-0400 Ncat: Listening on 0.0.0.0:80
2025-03-09 16:19:14-0400 Ncat: Connection from 192.168.1.103.
2025-03-09 16:19:14-0400 Ncat: Connection from 192.168.1.103:43716.
2025-03-09 16:19:14-0400
```

## Seeing firewall entries in Syslog

```
pi@raspi-idh:~$ tail -f /var/log/syslog | grep -E "ip6?tables_inbound"
2025-03-09T16:18:49.449315-04:00 raspi-idh kernel: [ 1012.197372] [iptables_inbound]IN
=eth0 OUT= MAC=d8:3a:dd:6b:e4:08:bc:24:11:ec:07:45:08:00 SRC=192.168.1.103 DST=192.168
.1.105 LEN=44 TOS=0x00 PREC=0x00 TTL=56 ID=59424 PROTO=TCP SPT=39335 DPT=110 WINDOW=10
24 RES=0x00 SYN URGP=0
2025-03-09T16:19:14.095491-04:00 raspi-idh kernel: [ 1036.844793] [iptables_inbound]IN
=eth0 OUT= MAC=d8:3a:dd:6b:e4:08:bc:24:11:ec:07:45:08:00 SRC=192.168.1.103 DST=192.168
.1.105 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=4646 DF PROTO=TCP SPT=43716 DPT=80 WINDOW=6
4240 RES=0x00 SYN URGP=0
2025-03-09T16:20:51.239533-04:00 raspi-idh kernel: [ 1133.991304] [iptables_inbound]IN
=eth0 OUT= MAC=d8:3a:dd:6b:e4:08:bc:24:11:ec:07:45:08:00 SRC=192.168.1.103 DST=192.168
.1.105 LEN=44 TOS=0x00 PREC=0x00 TTL=58 ID=62855 PROTO=TCP SPT=65214 DPT=5900 WINDOW=1
024 RES=0x00 SYN URGP=0
2025-03-09T16:20:51.239905-04:00 raspi-idh kernel: [ 1133.991482] [iptables_inbound]IN
=eth0 OUT= MAC=d8:3a:dd:6b:e4:08:bc:24:11:ec:07:45:08:00 SRC=192.168.1.103 DST=192.168
.1.105 LEN=44 TOS=0x00 PREC=0x00 TTL=47 ID=51531 PROTO=TCP SPT=65214 DPT=1025 WINDOW=1
024 RES=0x00 SYN URGP=0
```

## Caveat


- Any port in use by the system will fail for Netcat.  
2025-03-06 18:43:00:-0500 listener started  
2025-03-06 18:43:00-0500 Ncat: Version 7.93 ( <https://nmap.org/ncat> )  
2025-03-06 18:43:00-0500 Ncat: bind to :::22: Address already in use.  
QUITTING.
- Using daemon to auto start NC will snag ssh port before sshd on reboot.
  - Change sshd port number before reboot.
- Nmap scans show up in firewall logs, not honeypot logs.



## Breaking 1 rule: Not completely zero interaction

```
└─$ nmap -T4 -Pn -n -p- 192.168.1.105
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-09 16:06 EDT
Nmap scan report for 192.168.1.105
Host is up (0.00027s latency).
Not shown: 65526 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
8009/tcp  open  ajp13
8443/tcp  open  https-alt
9122/tcp  open  grcmp
MAC Address: D8:3A:DD:6B:E4:08 (Raspberry Pi Trading)

Nmap done: 1 IP address (1 host up) scanned in 1.48 seconds
```



## THANK YOU

Links to sources

- @rattis – twitter, InfoSec. Exchange
- @rattis.bsky.social
- Blog: <https://www.rattis.net>
- GitHub: <https://github.com/crattis>
- LinkedIn:  
<https://www.linkedin.com/in/christopherjenks/>

Here are my current socials that I somewhat monitor.



# References

LINKS TO SOURCES

## Books

- Bill Cheswick - Firewalls and Internet Security
- Chris Sanders – Intrusion Detection Honeypots
- John Strand, et al. - Offensive Countermeasures: The art of Active Defense
- Cliff Stoll – The Cuckoo’s Egg

## Websites

- Antisyphon Training - Active Defense and Cyber Deception (pay what you can) <https://www.antisiphontraining.com/course/active-defense-and-cyber-deception-with-john-strand/>
- FIPS 39 - <https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub39.pdf>
- 3500-year-old honeypot - <https://www.heritagedaily.com/2021/04/3500-year-old-honeypot-oldest-direct-evidence-for-honey-collecting-in-africa/138695>
- Lance Spitzner – Build a Honeypot - <https://web.archive.org/web/20131230094328/http://www.spitzner.net:80/honeypot.html>
- HoneyNet Project - <https://www.honeynet.org/>