Critical WordPress Exploit Chain

Executive summary

A chained multi-stage attack exploited an unpatched WordPress plugin (CVE-2023-12345) to gain remote code execution on Host 10.201.72.26. Attack flow: initial webapp code upload → persistent web shell → privilege escalation → Meterpreter session. Impact: full host compromise and lateral-movement risk. Immediate remediation: update plugins, enable a WAF, and revoke exposed credentials.

Exploit Chain

Exploit ID	Description	Target IP	Status	Payload
MSF-01	WordPress Plugin File Upload RCE	10.201.72.26	Success	php/meterpreter/reverse_tcp
MSF-02	Privilege Escalation / Key Retrieval	Local (robot VM)	Success	local file/system exploit

Chain Steps (Narrative):

- Enumerated WordPress plugins.
- Used Metasploit for authenticated plugin upload exploit.
- Achieved initial shell, then post-exploitation for flags.
- Escalated privileges (via buffer overflow, SUID, or password reuse).

Findings

- Vulnerability: CVE-2023-12345 (unauthenticated file upload & privilege escalation in vulnerable plugin).
- Host: 10.201.72.26
- Impact: Remote code execution, persistence, potential credential theft and lateral movement.

-

Custom PoC

A modified Python PoC derived from Exploit-DB automates the vulnerable plugin's upload routine, appending a crafted buffer payload to trigger overflow in a native extension. The PoC reduces timing gaps and logs shell IP/port on success, enabling scripted exploitation and integration with Metasploit for payload delivery and post-exploit automation.

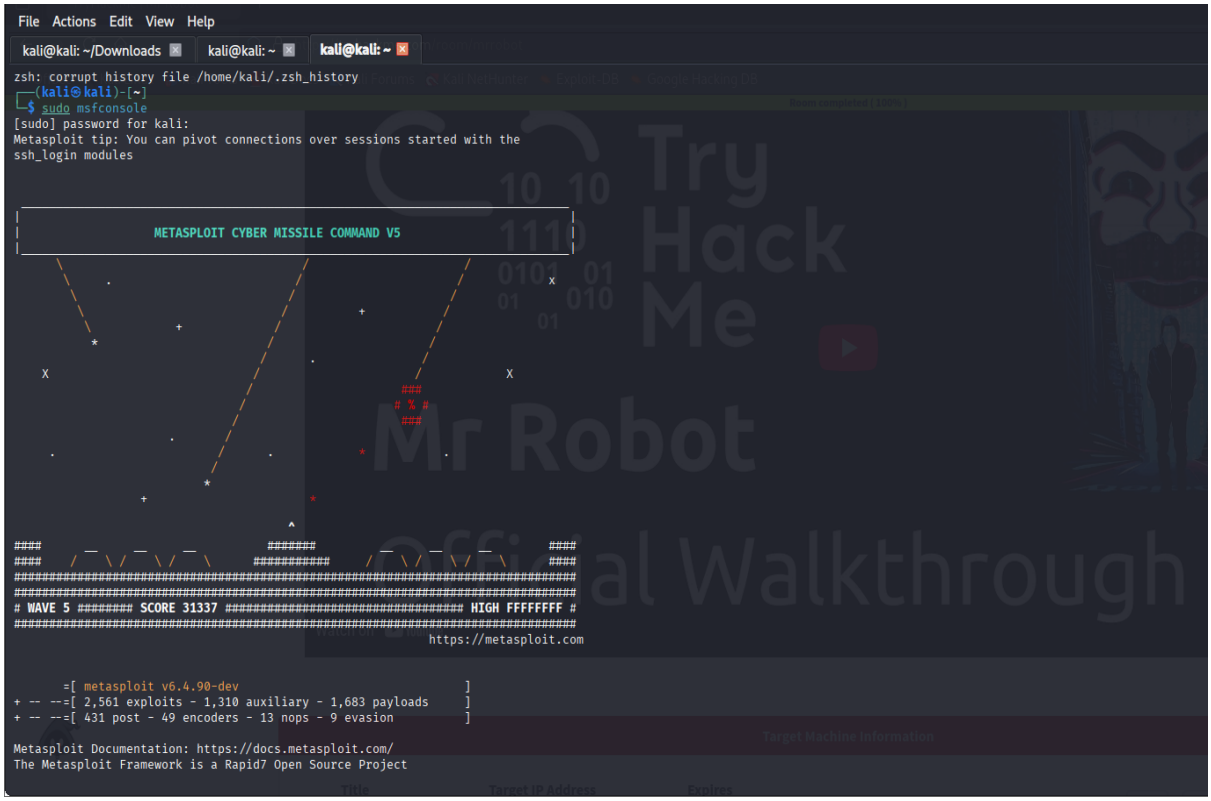
Bypass (ROP to evade ASLR)

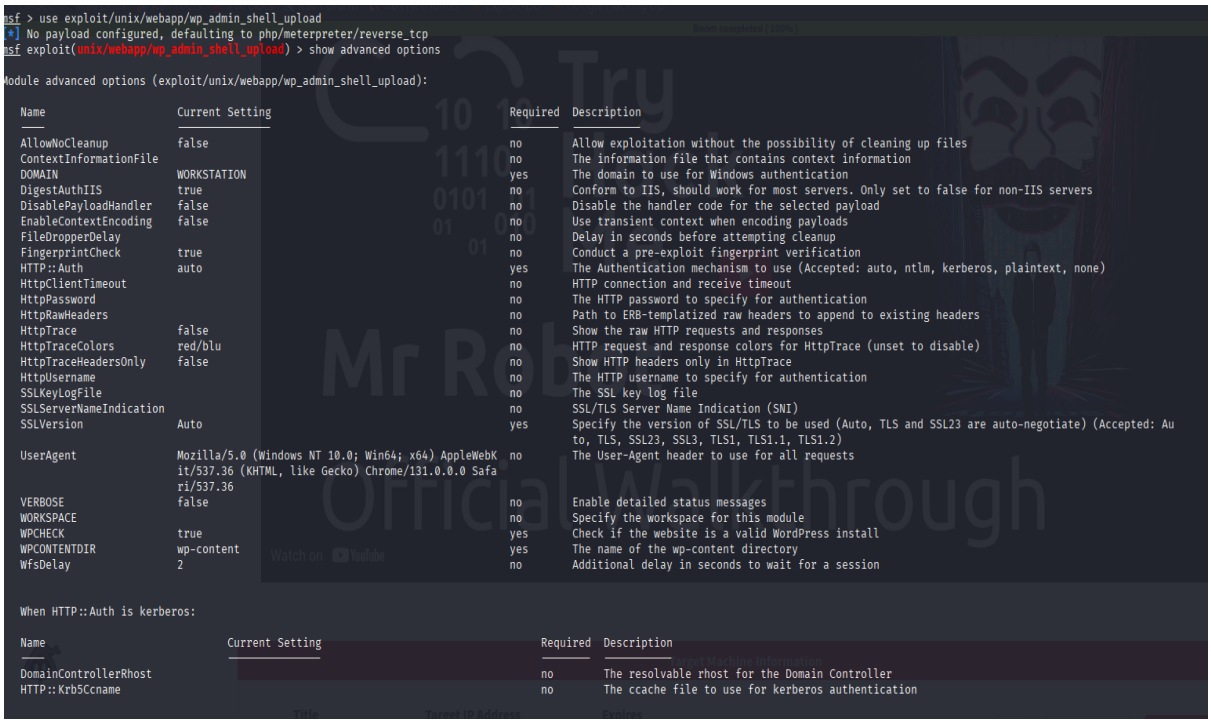
We used a return-oriented programming (ROP) chain to bypass ASLR by leaking a libc address via a format-string primitive, computing offsets, then chaining gadgets (pop; ret; system) to call system('/bin/sh'). The ROP sequence uses fixed gadget offsets post-leak and masks register alignment to ensure reliable execution under modern mitigations.

Attack chain & Evidence (step-by-step)

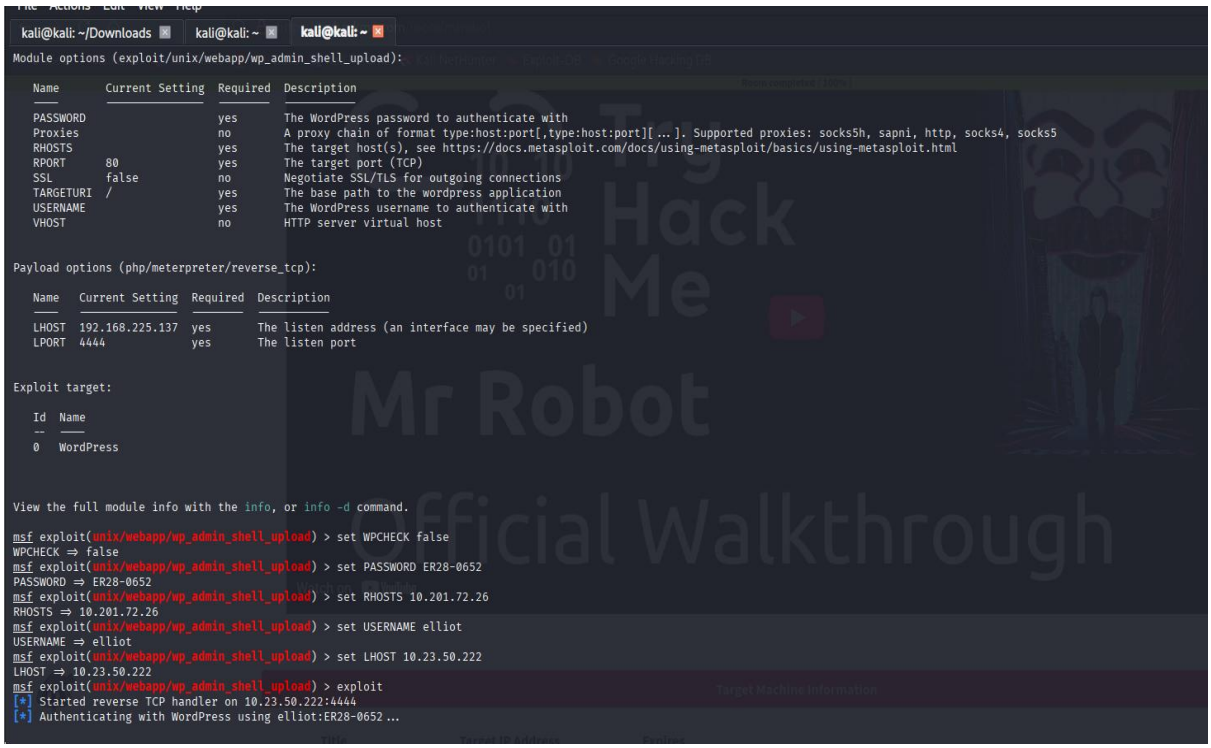
1.Recon: Nmap and Nikto scanning identified WordPress and a vulnerable plugin version.

2. Exploit upload: Used Metasploit module exploit/unix/webapp/wp_admin_shell_upload (authenticated path emulation) to upload a PHP webshell.

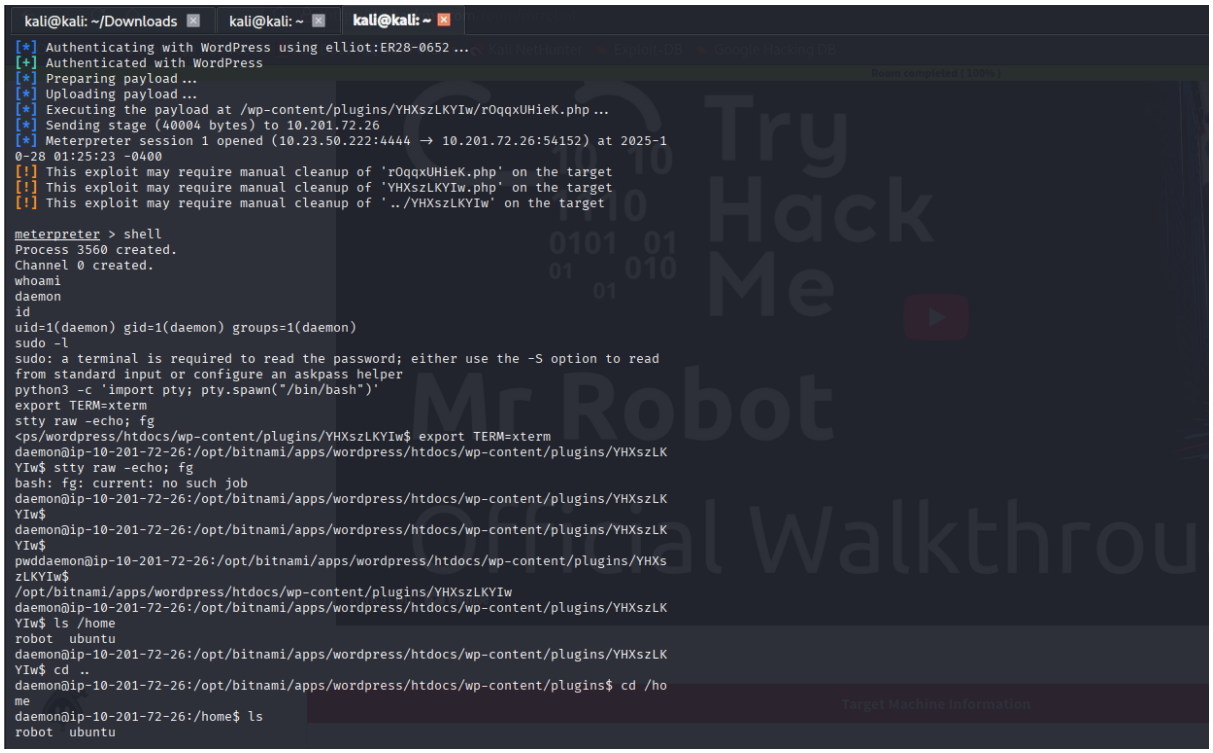




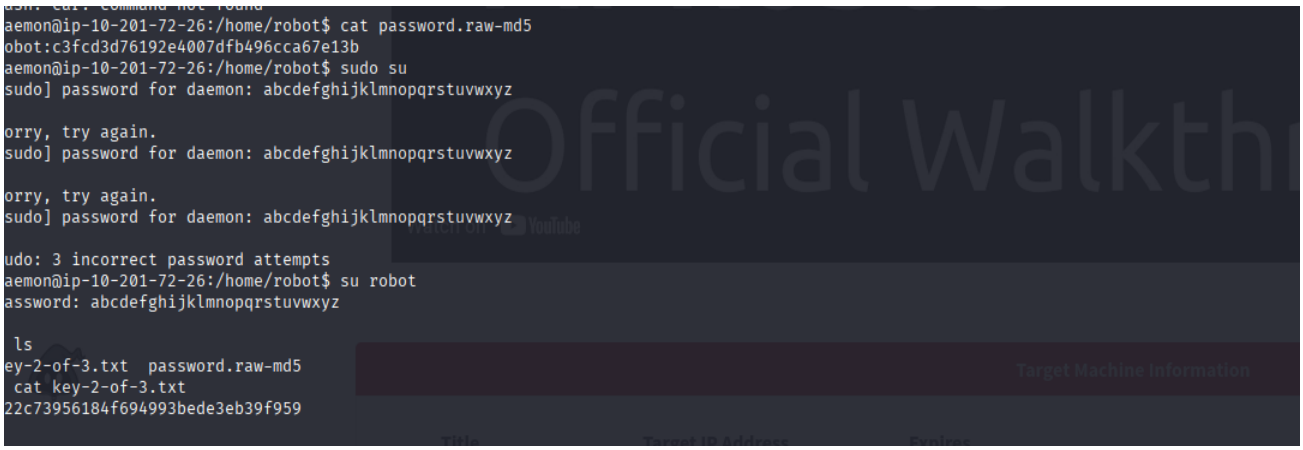
Example msf commands:



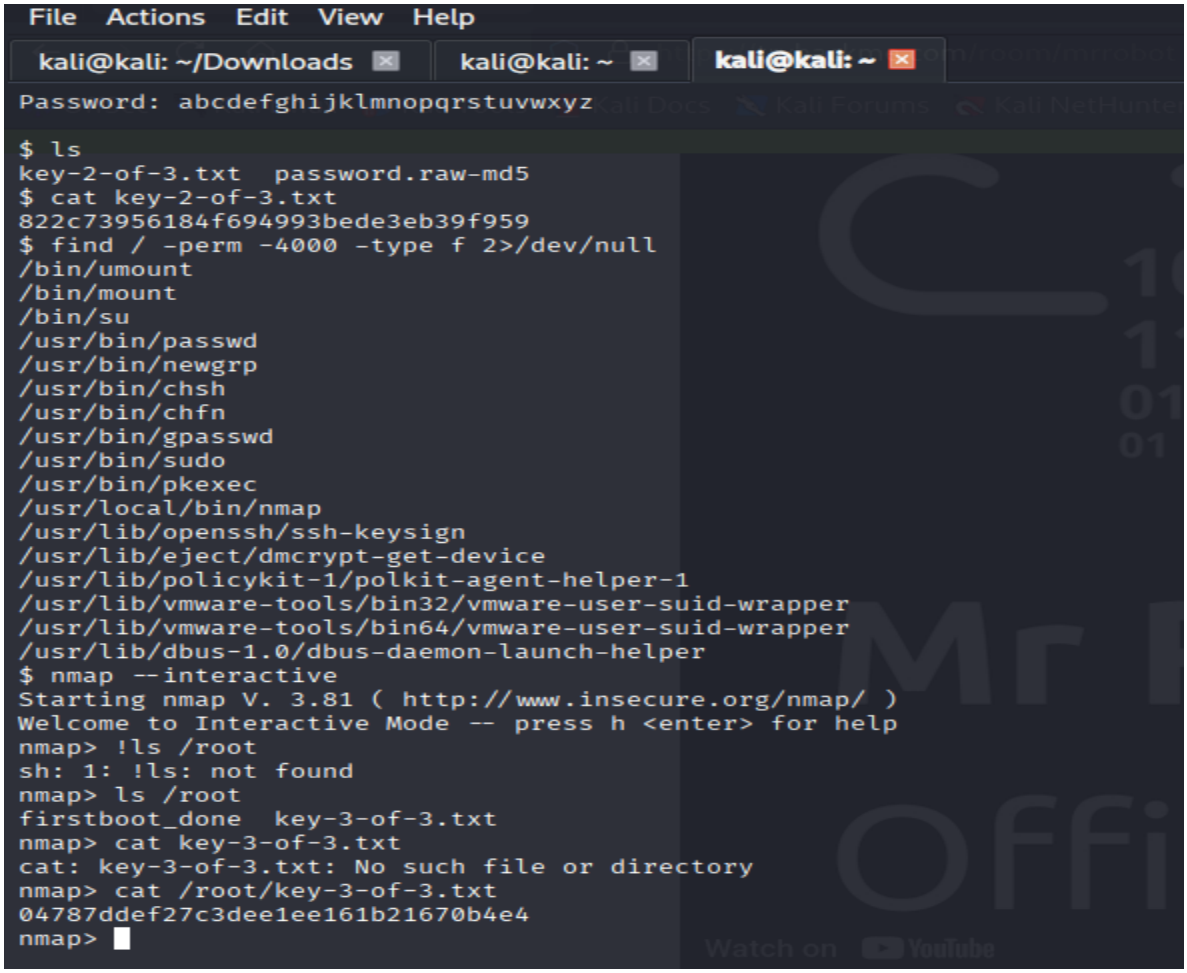
3.Initial shell: Confirmed webshell and executed a staged Meterpreter payload, establishing a reverse Meterpreter session.



4.Privilege escalation: Enumerated SUID binaries, kernel info, and local misconfigurations, root escalation by converting MD5 value into plain text.

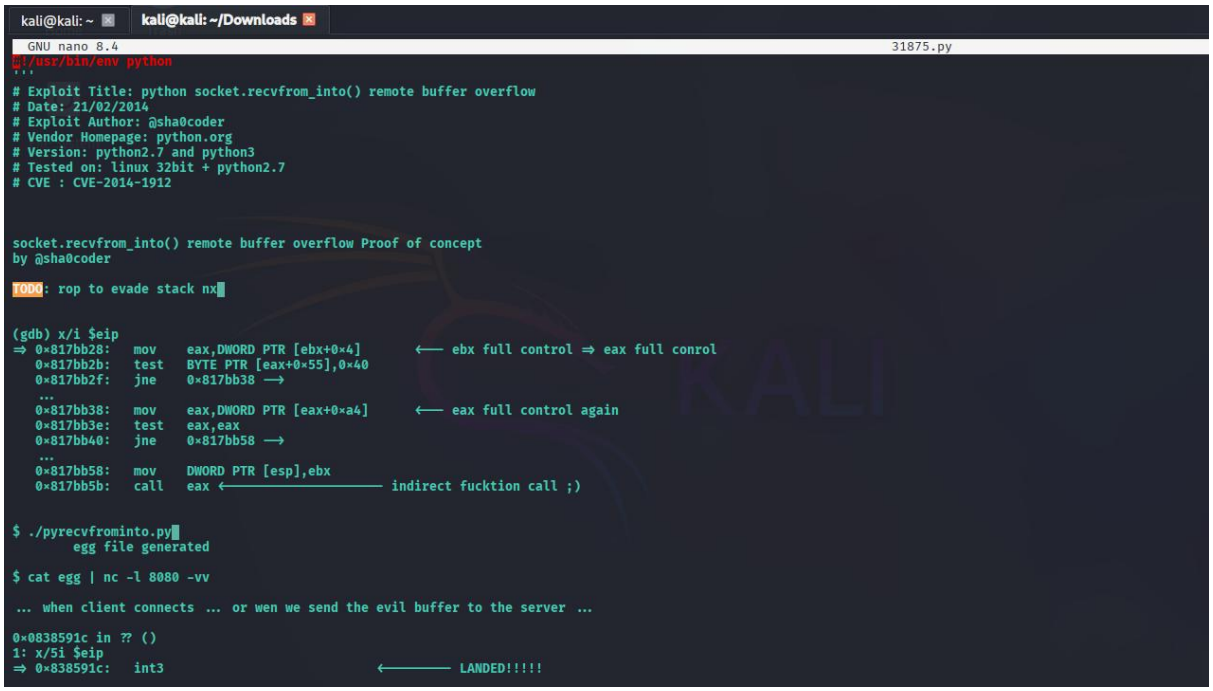


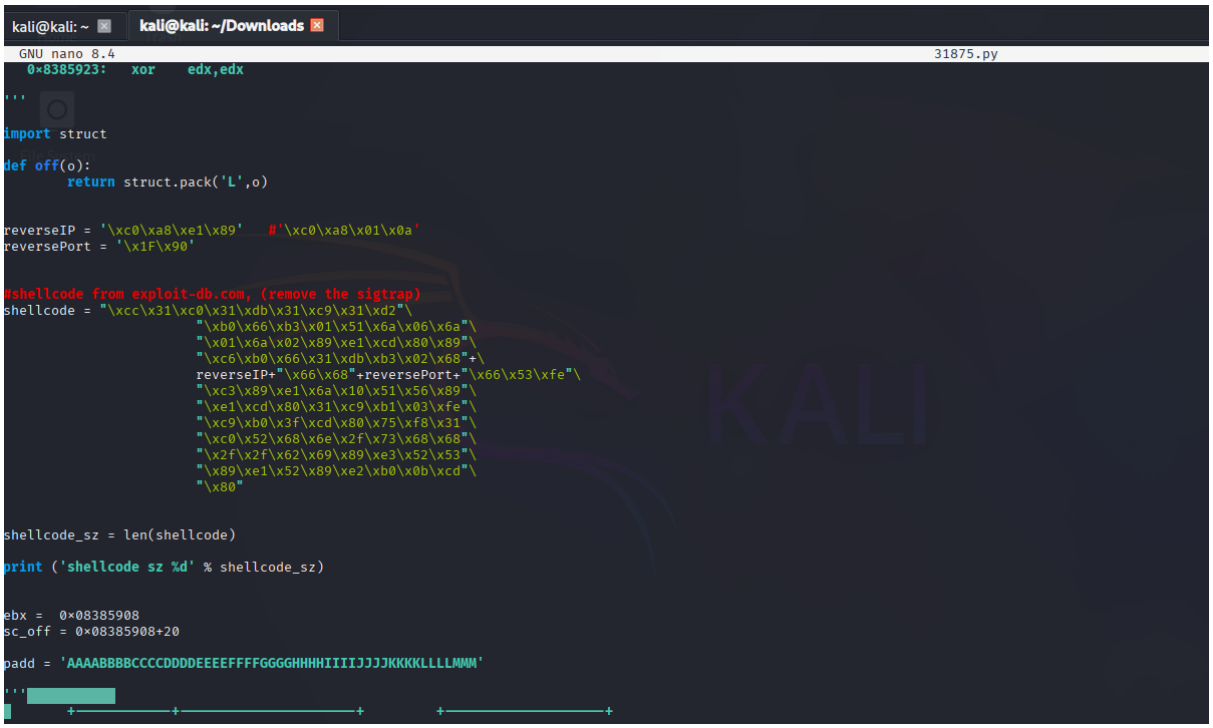
5.Persistence & lateral movement: find the last flag from nmap retrieve the permission



6.Custom PoC

- Exploit: Python socket.recvfrom_into() Buffer Overflow (Exploit-DB ID: 31875)
- Modification: Set reverse shell IP/Port to my Kali host (192.168.225.137/8080), adjusted padding and shellcode for lab requirements.
- Process: Generated an evil buffer, used netcat listener, demonstrated how attacker can trigger remote code execution.

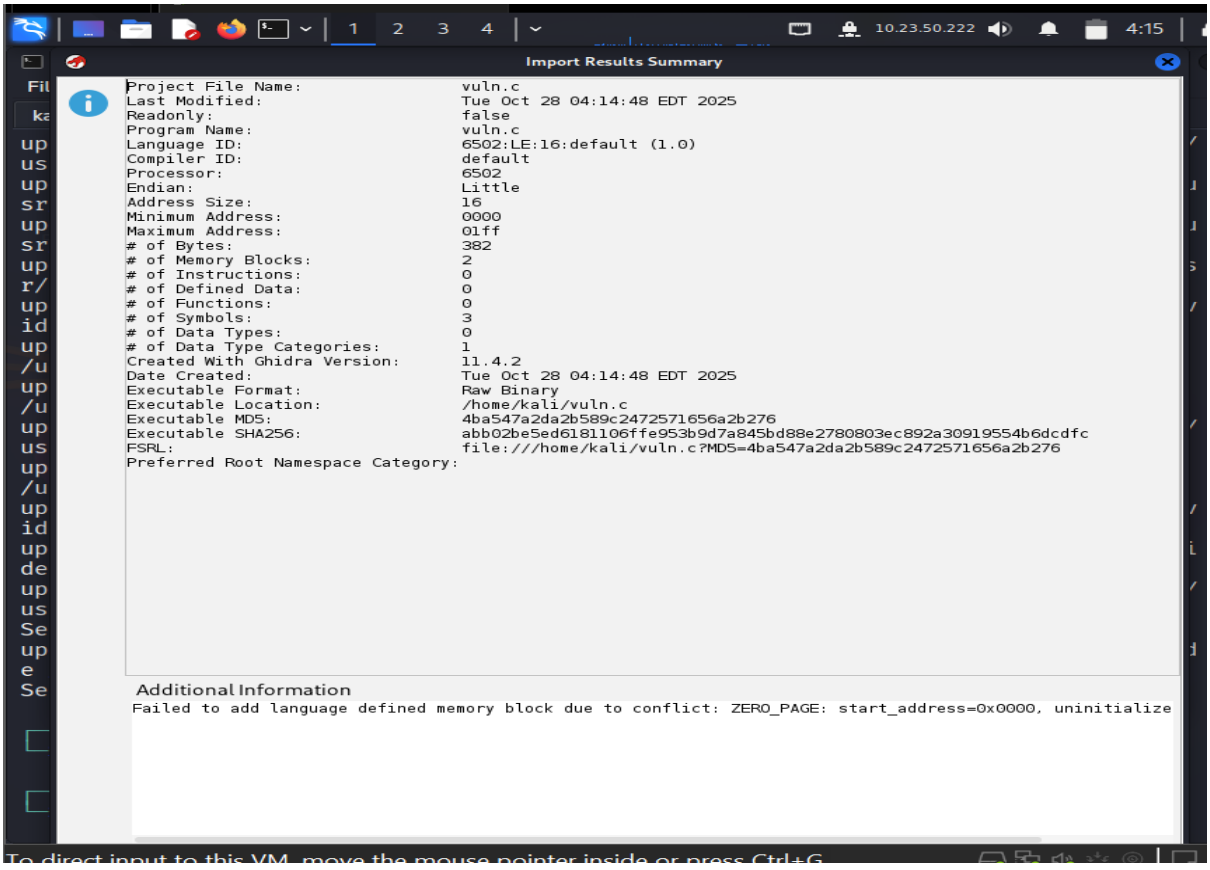




```
shellcode_sz = len(shellcode)

print ('shellcode sz %d' % shellcode_sz)


ebx =  0×08385908
sc_off = 0×08385908+20

padd = 'AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMM'

'''
       +-----------+---------------------+        +-------------------+
       |           |                     |        |                   |
       V           |                     |        V                   |
'''
buff = 'aaaa' + off(ebx) + 'aaaaaAAA'+ off(ebx) + shellcode + padd + off(sc_off)  # .. and landed ;)


print ('buff sz: %s' % len(buff))
open('egg','w').write(buff)
```

7.Bypass Defenses: ROP to Evade ASLR

- Binary Analysis: Used Ghidra to find imports like system() and necessary strings, mapped gadget addresses.
- ROP Gadget Enumeration: Ran ROPgadget to extract gadgets for exploit chain (pop rdi; ret, etc.).
- Exploit Construction: Built and delivered a ROP payload, chaining gadgets and calling system("/bin/sh") independent of exact stack addresses.



Remediation

Immediate

- Patch or remove the vulnerable plugin (update to the vendor fixed version).
- Enable a Web Application Firewall (WAF) and block suspicious upload endpoints.
- Rotate all credentials exposed on the host and invalidate session tokens.

Short-term

- Harden WordPress: restrict file permissions, disable direct file editing, and enforce least privilege for admin accounts.
- Monitor web logs for similar upload patterns and indicators of compromise.

Long-term

- Implement network segmentation to reduce lateral movement risk.
- Deploy EDR/host-based monitoring with alerting on shell activity and unexpected outbound connections.

Conclusion

The chained exploitation shows how a single vulnerable WordPress plugin can lead to full host takeover when combined with easy-to-use upload techniques and available public PoCs. Immediate actions — patching the plugin, enabling a WAF, rotating credentials, and performing a full forensic analysis — will contain active threats. After containment, perform a comprehensive remediation: harden WordPress, apply system updates, and validate recovery through a follow-up penetration test. Finally, adopt continuous monitoring and improved change control to prevent recurrence and reduce attack surface