

INFORME DE RESULTADOS

RETO PYTHON - TEAM CHALLENGE - SPRINT 1 - CIBERSEGURIDAD

Elaborado por:	Fecha de creación:
Víctor Manuel Martínez Barberá	03/06/2024

En relación al reto practico indicado, se han practicado las instrucciones recibidas en los archivos en pdf habiendo obtenido un resultado satisfactorio, como se puede observar en las siguientes capturas:

1 •— Una vez configurada la imagen .ova del reto, se abrió en el Virtual Vox esta maquina, aperturando además mi Kali Linux, ejecutando el comando ip a, con el cual se muestra información detallada sobre las interfaces de red de tu sistema Linux, tanto las físicas como las virtuales, usándose para verifacas si tus interfaces están bien configuradas, con las IPs esperadas. También se puede usar esta información para hacer nuestro scripts.

```
y ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:1c:12:50 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.4/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
        valid_lft 474sec preferred_lft 474sec
    inet6 fe80::a00:27ff:fe1c:1250/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Extraemos que nuestra interface de red es eth0.

2 •- Ejecutamos el comando `arp-scan -I eth0 -l`, con el cual realizamos un mapeo de la red buscando los dispositivos que están conectados a la red `eth0`, sus IPs, para ellos envía solicitudes ARP (protocolo de resolución de direcciones) a todas las direcciones de mi red, esperando respuesta de las mismas. Este comando es muy útil en ciberseguridad para mapear la red en busca de intrusiones no autorizadas.

```
> sudo arp-scan -I eth0 -l
[sudo] password for vicevil:
Interface: eth0, type: EN10MB, MAC: 08:00:27:1c:12:50, IPv4: 10.0.2.4
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      52:54:00:12:35:00      (Unknown: locally administered)
10.0.2.2      52:54:00:12:35:00      (Unknown: locally administered)
10.0.2.3      08:00:27:45:d5:c5      (Unknown)
10.0.2.15     08:00:27:c3:3a:68      (Unknown)

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.039 seconds (125.55 hosts/sec). 4 responded
```

En el resultado observamos 4 direcciones IPs (10.0.2.1, 10.0.2.2 y 10.0.2.3), siendo éstas utilizadas por VirtualBox para facilitar la comunicación entre el sistema anfitrión y el sistema invitado, por lo que la IP donde realizar la conexión segura.

3 •- Se procede a establecer la conexión segura, tunelizada y cifrada simétricamente mediante clave pública entre el sistema anfitrión y la maquina `Reto_python`, siendo utilizado en ciberseguridad, para automatizar de tareas periódicas repetitivas en servidores remotos mediante scripts.

```
> ssh user1@10.0.2.15
The authenticity of host '10.0.2.15 (10.0.2.15)' can't be established.
ED25519 key fingerprint is SHA256:HkyjvIvLqRiz0qjlfLrfHaHUmKRRKMcRhYtkxwu6N9Q.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.15' (ED25519) to the list of known hosts.
user1@10.0.2.15's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

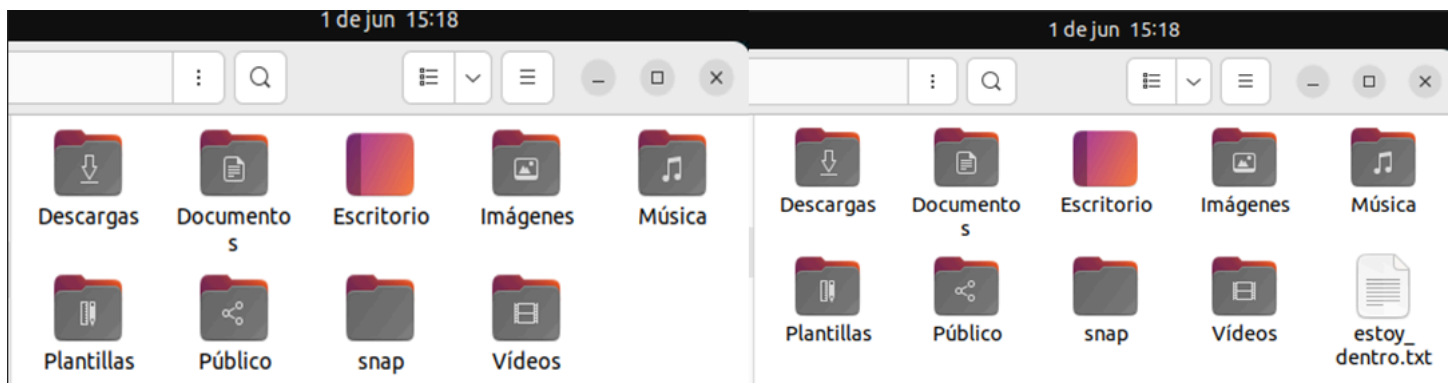
Se pueden aplicar 275 actualizaciones de forma inmediata.
67 de estas son actualizaciones de seguridad estándares.
Para ver estas actualizaciones adicionales, ejecute: apt list --upgradable

*** System restart required ***
Last login: Wed May  8 23:24:38 2024 from 10.0.2.4
user1@ubuntu-VirtualBox:~$
```

Aquí ya estaríamos conectados, por lo que podríamos ejecutar comandos, transferir archivos, instalar programas y realizar otras tareas de forma segura (los administradores de sistemas se conectan así, y además consume menos recursos que el TeamViewer).

4 •— finalmente hacemos la comprobación si realmente están conectados, creando un archivo `estoy_dentro.txt` desde la maquina anfitrión, siendo la misma positiva:

```
user1@ubuntu-VirtualBox:~$ ls
Descargas Documentos Escritorio Imágenes Música Plantillas Público snap Videos
user1@ubuntu-VirtualBox:~$ touch estoy_dentro.txt
user1@ubuntu-VirtualBox:~$ ls
Descargas Documentos Escritorio estoy_dentro.txt Imágenes Música Plantillas Público snap Videos
user1@ubuntu-VirtualBox:~$
```



NIVELES

NIVEL1:

```
root@ubuntu-VirtualBox:/home# cd user1
root@ubuntu-VirtualBox:/home/user1# ls
Descargas Documentos Escritorio Imágenes Música Plantillas Público snap Videos
root@ubuntu-VirtualBox:/home/user1# cd Documentos
root@ubuntu-VirtualBox:/home/user1/Documentos# ls
nivel1.py README.txt
root@ubuntu-VirtualBox:/home/user1/Documentos# nano nivel1.py
root@ubuntu-VirtualBox:/home/user1/Documentos# python3 nivel1.py
La pass de user2 es: S0Easy&Gu1d3d
root@ubuntu-VirtualBox:/home/user1/Documentos#
```

NIVEL 2: S0Easy&Gu1d3d

```

GNU nano 6.2 nivel2.py
import time
def loading():
    signs = ["|", "/", "-", "\\"]
    for i in range(100):
        print(signs[i%4].format(i%4), end='\r')
        time.sleep(0.1)
def menu():
    print("Bienvenido al panel de administración.")
    print("Iniciando el motor de búsqueda")
    loading()
opcion = ''
# La pass de user3 es N0C0mm3nts!
while opcion != 'n':
    menu()
    opcion = input("Parece que el servicio está detenido. ¿Quieres volver a intentarlo?")

```

NIVEL 3 N0C0mm3nts!

```

import requests
import base64
def decode():
    base64_message = 'SDRyZEMwZGU='
    message_bytes = base64.b64decode(base64_message)
    message = message_bytes.decode('ascii')
    return message
def menu():
    print("Bienvenido al panel de administración.")
    print("Iniciando el motor de búsqueda")
    menu()
user = input("Introduce el nombre de usuario: ")
passw = input("Introduce la pass: ")
if (user == "devadmin") and (passw == "H4rdC0ding&F4il"):
    print("La pass de user4 es: " + decode())
else:
    print("El usuario no existe")

```

```

root@ubuntu-VirtualBox:/home/user3/Documentos# nano nivel3.py
root@ubuntu-VirtualBox:/home/user3/Documentos# python3 nivel3.py
Bienvenido al panel de administración.
Iniciando el motor de búsqueda
Introduce el nombre de usuario: devadmin
Introduce la pass: H4rdC0ding&F4il
La pass de user4 es: H4rdC0de

```

NIVEL4: H4rdC0de

```

import sys
import time
import base64

def decode():
    base64_message = 'UEByYW1ldDNycw=='
    message_bytes = base64.b64decode(base64_message)
    message = message_bytes.decode('ascii')
    return message

def loading():
    signs = ["|", "/", "-", "\\"]
    for i in range(40):
        print(signs[i%4].format(i%4), end='\r')
        time.sleep(0.1)

def menu():
    print("Bienvenido al panel de administración.")
    print("Iniciando el motor de búsqueda")
    loading()

args = sys.argv
menu()

if (len(args)%2 == 0) and (args[4] == 'debug'):
    print("La pass de user5 es: " + decode())
else:
    print("Parece que el servicio está detenido. Vuelve a intentarlo")

```

Aquí tengo que interpretar el código python. Este comienza con el `arg0` y así sucesivamente, así que `arg0` es el propio script, `arg1,2` y `3` son los otros 3 argumentos, que cumple el total de argumentos el cociente es cero, y el ultimo deber ser `= debug`

```

root@ubuntu-VirtualBox:/home/user4/Documentos# nano nivel4.py
root@ubuntu-VirtualBox:/home/user4/Documentos# python3 nivel4.py argumento1 argumento2 debug
Bienvenido al panel de administración.
Iniciando el motor de búsqueda
La pass de user5 es: P@ramet3rs

```

NIVEL5: P@ramet3rs

```

import requests
import base64

def decode():
    base64_message = 'RXh0M3JuYWxLM1k='
    message_bytes = base64.b64decode(base64_message)
    message = message_bytes.decode('ascii')
    return message

response = requests.get('https://gist.githubusercontent.com/poundifdef/fd0901799a4aeb3a1f3956cfdb3c7746/raw/36f07f44ba9f80eb93fbb336e2bdf4096adeb484/trollface.txt')
key = str(response.text).split('\n')[-2].strip()

user_response = str(input("¿Cómo puedo ayudarte? \n"))

if user_response == key:
    print("La pass de user6 es: " + decode())
else:
    print("No puedo ayudarte con eso")

```

Al parecer que el enlace web le faltan elementos y no llega a ningún sitio ni da respuesta, por lo que procedo a investigar por la web llegando a la web original :

["https://gist.githubusercontent.com/poundifdef/fd0901799a4aeb3a1f3956cfdb3c7746/raw/36f07f44ba9f80eb93fbb336e2bdf4096adeb484/trollface.txt"](https://gist.githubusercontent.com/poundifdef/fd0901799a4aeb3a1f3956cfdb3c7746/raw/36f07f44ba9f80eb93fbb336e2bdf4096adeb484/trollface.txt)

Una vez hecho esto incluyo en el script una simple respuesta para que me de la clave a introducir en el mensaje, devolviéndose la password del user6

```

import requests
import base64

def decode():
    base64_message = 'RXh0M3JuYWxLM1k='
    message_bytes = base64.b64decode(base64_message)
    message = message_bytes.decode('ascii')
    return message

response = requests.get('https://gist.githubusercontent.com/poundifdef/fd0901799a4aeb3a1f3956cfdb3c7746/raw/36f07f44ba9f80eb93fbb336e2bdf4096adeb484/trollface.txt')
key = str(response.text).split('\n')[-2].strip() # Obtiene la clave del archivo

print("La clave es: " + str(key)) # Imprime la clave

user_response = str(input("¿Cómo puedo ayudarte? \n"))

if user_response == key:
    print("La pass de user6 es: " + decode())
else:
    print("No puedo ayudarte con eso")

```

✓ 11.1s

La clave es: -:o+/:...
La pass de user6 es: Ext3rnalK3Y

NIVEL6: Ext3rnalK3Y

Al no encontrar la bases de datos nombrada en el archivo python procedo a buscar el archivo oculto en la misma carpeta:

"find / -name ".registered_users.db" 2>/dev/null"

Una vez ubicado el archivo, se comprueba que no es una BBDD, así que se trata como un archivo normal

Procedo a ver el contenido del archivo , ejecutando la siguiente función, y comprando que hay mas de un millón de registros:


```
def read_db():
    with open(PATH_BBDD, 'r') as file:
        lines = file.readlines()
        for line in lines:
            print(line.strip()) # Imprime cada línea (email) en el archivo

read_db()
```

Por lo que borramos los ultimo registros, dejando hasta el 999998.

```
def delete_last_lines(filename, n):
    # Lee todas las líneas del archivo
    with open(filename, 'r') as file:
        lines = file.readlines()

    # Reescribe el archivo con todas las líneas excepto las últimas n
    with open(filename, 'w') as file:
        for line in lines[:-n]:
            file.write(line)

delete_last_lines(PATH_BBDD, 9)
```

Ahora ya ejecutamos dos veces el script del ejercicio, resultando la contraseña

```
> python nivel6.py
Bienvenido al concurso 1.000.000
Regístrate para ganar un fantástico premio. ¿Serás tu el participa
nte 1.000.000?
Introduce tu dirección de mail: casies@ciber.es
Estamos verificando si eres el participante 1.000.000 ...
Lo sentimos, no eres el participante 1.000.000
> python nivel6.py
Bienvenido al concurso 1.000.000
Regístrate para ganar un fantástico premio. ¿Serás tu el participa
nte 1.000.000?
Introduce tu dirección de mail: victormbar@gmail.com
Estamos verificando si eres el participante 1.000.000 ...
Enhorabuena, has ganado!
La pass de user7 es: =0neMllion
```

NOTA: Coincide con el resultado que da la función decode al ejecutarla.

NIVEL 7: =0neMllion

Para la clave del User 8 Aparecen dos archivos.py:

```

def getotp():
    ...return str(random.randint(10000000,99999999))

def decode_pass():
    ...base64_string = "TGEGcGFzcyBkZSB1c2VyOCBlczogVENQIVMwY2szdA=="
    ...base64_bytes = base64_string.encode("ascii")

    ...sample_string_bytes = base64.b64decode(base64_bytes)
    ...sample_string = sample_string_bytes.decode("ascii")
    ...return sample_string

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.bind(("127.0.0.1", 8050))
conn.listen(1)
cli, addr = conn.accept()
otp = ""

while True:

    recibido = cli.recv(1024)

    print("Recibo conexion de la IP: " + str(addr[0]) + " Puerto: " + str(addr[1]))

    comando = recibido.decode()

    if comando == "user":
        otp=getotp()
        cli.send(otp.encode())
    elif comando == otp:
        cli.send(decode_pass().encode())
    else:
        cli.send("404 Command not found".encode())

```

```

host = '127.0.0.1'
port = 8050

obj = socket.socket()
obj.connect((host, port))

print("Conectado al servidor")

while True:
    mens = input("Comando >> ")

    obj.send(mens.encode())

    respuesta = obj.recv(1024)

    print(respuesta.decode())

obj.close()

print("Conexión cerrada")

```

Una vez observados, se comienza ejecutando independientemente la función “decode_pass” , que nos muestra la contraseña valida.


```
def getotp():
    return str(random.randint(10000000,99999999))

def decode_pass():
    base64_string = "TGEgcGFzcyBkZSB1c2VyOCBlczogVENQIVMwY2szdA=="
    base64_bytes = base64_string.encode("ascii")

    sample_string_bytes = base64.b64decode(base64_bytes)
    sample_string = sample_string_bytes.decode("ascii")
    return sample_string
decode_pass()
```

✓ 0.0s

'La pass de user8 es: TCP!S0ck3t'

NIVEL8:TCP!S0ck3t

EL juego del trilero•– Se ejecuta la función “getNewPosition” observando un patrón que se repite:

```
def getNewPosition(timeexec,round):
    base = timeexec.split(':')[round%3]
    position = (int(base)+round)%3
    return str(position + 1)
```

LA SERIE ES 3,2,3,3,2,3,3,2,3,3,2....

Aplicando dicho patrón, y siguiendo la secuencia lógica de respuestas hallada:

```
> python nivel8.py
```



```
20:00:57 ...
¿Puedes adivinar dónde está la bolita 5 veces seguidas?
Estamos reordenando los vasos
Introduce el número del vaso (1, 2, 3): 3
Interesante ...
Estamos reordenando los vasos
Introduce el número del vaso (1, 2, 3): 2
He visto gente con suerte... pero nunca tanta como tú
Estamos reordenando los vasos
Introduce el número del vaso (1, 2, 3): 3
Interesante ...
Estamos reordenando los vasos
Introduce el número del vaso (1, 2, 3): 3
Esta no la ví venir
Estamos reordenando los vasos
Introduce el número del vaso (1, 2, 3): 2
He visto gente con suerte... pero nunca tanta como tú
No se como lo has hecho pero has ganado ...
Aquí tienes la pass de user9:
N0tS0Rand0m!
```

NOTA: igual que anteriormente, misma contraseña que sale al ejecutar la función decode

NIVEL9: N0tS0Rand0m!

Se procede a decodificar la parte que viene en la función execute_low:
"user10 - Encoded pass: 6e6957267376697250796669646f4d" en hexadecimal,
creando una función para ello:

#vamos a realizar una función de decode en base hexadecimal

```
def decode_hex(hex_string):  
    bytes_object = bytes.fromhex(hex_string)  
    ascii_string = bytes_object.decode("ASCII")  
    return ascii_string
```

```
print(decode_hex(baseHX_message))
```

✓ 0.0s

niW&svirPyfidoM

Se observa también que en la función `execute_high` aparece otro `print` con la información `"Hex(Reverse(password))"`, por lo que se intuye que hay que revertir la contraseña decodificada:

```
clave_hex= "niW&svirPyfidoM"  
clave_hex_invertida = clave_hex[::-1]  
clave_hex_invertida
```

✓ 0.0s

'ModifyPrivs&Win'

NIVEL10: ModifyPrivs&Win