

Cyber@UC Meeting 90

MBE: Basic Binary Exploitation



Hacking



*Information- and
Cyber Security*

If You're New!

- Join our Slack: cyberatuc.slack.com
- Check out our website: cyberatuc.org
- Organization Resources on our Wiki: wiki.cyberatuc.org
- **SIGN IN!** (*Slackbot will post the link in #general every Wed@6:30*)
- Feel free to get involved with one of our committees:
Content Finance Public Affairs Outreach Recruitment Lab
- Ongoing work in our research lab!



Announcements

- Bi-weekly lab events!
 - Socket Programming!
- Organization planning meeting Sunday, all are welcome to attend
- Dodgeball Thursday
- CTF team



BATTELLE

***WEDNESDAY APRIL 10TH,
2019***

***GUEST SPEAKER:
AARON MCCANTY***

RE/VR AUTOMATION

SATURDAY APRIL 20TH, 2019

***FULL DAY EVENT
11AM - 4PM***

VIDEO GAME + CTF = 🤖

COLUMBUS OH

Weekly News

Chinese national arrested carrying malware

- Charged with lying to a federal agent
- Carrying 4 Phones, 2 Chinese passports, a laptop, and a USB drive containing malware
- Event that she said she was attending was non existent
- Also said she was there to use the pool
- President was at the resort at the same time
- She did not actually use the pool

<https://www.nytimes.com/2019/04/02/us/mar-a-lago-zhang-chinese-secret-service.html>



Basic Binary Exploitation

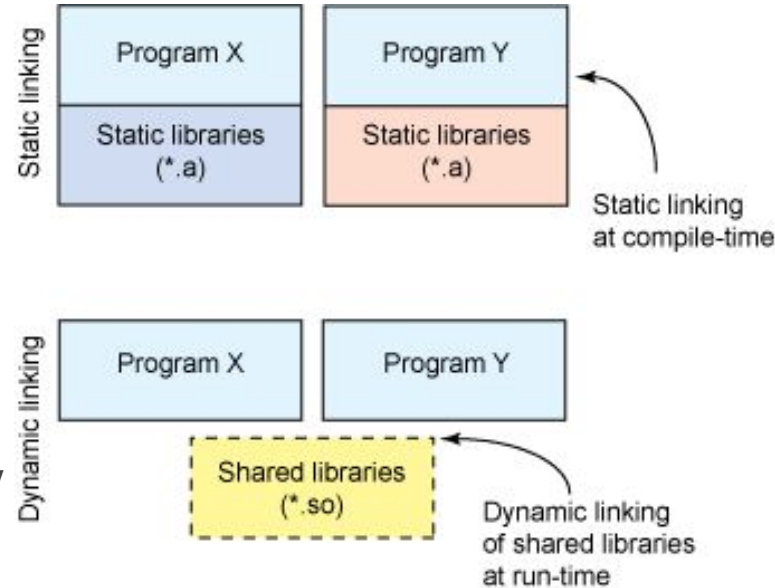
What are binary files?

- Source Code is plain text
- Source Code is compiled to Assembly which is also plain text
- Assembly is assembled to an Object file which is made of relocatable machine code
- Object Files are linked to each other and libraries into Binary Files
- Binary File have all dependencies resolved



What are binary files? (cont.)

- Binary files are typically one of these formats:
 - Portable Executable (PE) - used by Windows
 - Executable and Linkable Format (ELF) - used by everything else
- Both formats have support for static linking and dynamic linking
 - ELF uses object (.o) and shared object (.so)
 - PE uses executable (.exe) and dynamic linking library (.dll)



Source Code

```
int main()
{
    printf("Hello World!\n");
}
```

Compile



Assembly

```
_main EBEC
push ebp
mov ebp, esp
push OFFSET 00020005
call DWORD PTR __imp_printf
add esp, 4
xor eax, eax
pop ebp
set al
_main EBEC
TEXT EBC8
END
```

Assemble



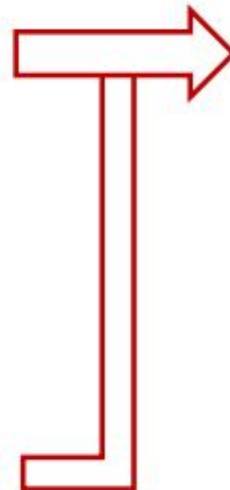
Object File

```
68 00 30 40 00 FF 15 90
50 C3 88 40 5A 00 00 66
33 C0 EB 34 80 00 0C 00
58 05 00 00 75 EA 00 00
40 00 75 DC 33 C0 83 89
01 18 00 40 00 0F 95 C0
15 7C 20 40 00 59 66 FF
3A 20 40 00 03 80 33 40
30 40 00 89 01 80 00 38
09 01 18 27 05 00 00 18
40 00 00 75 0C 68 EA 12
50 EA 40 05 00 00 83 30
FF FF 15 44 20 40 00 59
EA 04 04 00 00 01 58 30
00 FF 35 54 30 40 00 60
```

Libraries

```
68 00 30 40 00 FF 15 90
50 C3 88 40 5A 00 00 66
33 C0 EB 34 80 00 0C 00
58 05 00 00 75 EA 00 00
40 00 75 DC 33 C0 83 89
01 18 00 40 00 0F 95 C0
15 7C 20 40 00 59 66 FF
3A 20 40 00 03 80 33 40
30 40 00 89 01 80 00 38
09 01 18 27 05 00 00 18
40 00 00 75 0C 68 EA 12
50 EA 40 05 00 00 83 30
FF FF 15 44 20 40 00 59
EA 04 04 00 00 01 58 30
00 FF 35 54 30 40 00 60
```

Link



Binary File

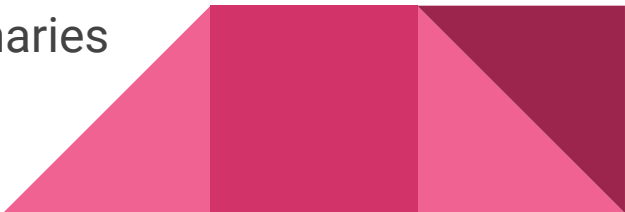
```
68 00 30 40 00 FF 15 90
50 C3 88 40 5A 00 00 66
33 C0 EB 34 80 00 0C 00
58 05 00 00 75 EA 00 00
40 00 75 DC 33 C0 83 89
01 18 00 40 00 0F 95 C0
15 7C 20 40 00 59 66 FF
3A 20 40 00 03 80 33 40
30 40 00 89 01 80 00 38
09 01 18 27 05 00 00 18
40 00 00 75 0C 68 EA 12
50 EA 40 05 00 00 83 30
FF FF 15 44 20 40 00 59
EA 04 04 00 00 01 58 30
00 FF 35 54 30 40 00 60
```

Tools

Static Analysis (not running):

- **strings** - dumps “readable” data from a binary file
- **file** - identifies a file format based on magic
- **md5sum** - gets md5 sum of a file
- **objdump** - converts binaries to assembly
- **binwalk** - searches for files in files

Dynamic Analysis:

- **IDA/GHIDRA/BinNin/R2** - Disassembly / visualize binaries
 - **GDB** - GNU Debugger
 - **GDB:GEF** - Extension for GDB
- 

GDB:GEF Setup

- Installation from (github)
 - `wget -O ~/.gdbinit-gef.py -q https://github.com/hugsy/gef/raw/master/gef.py`
 - `echo source ~/.gdbinit-gef.py >> ~/.gdbinit`
- If you already have GDB:PEDA it's similar but still actively developed so just use PEDA for today
- GEF adds a few UI improvements to keep you from repeating commands and extends the capabilities of GDB



Crackme's

- Files meant to be reverse engineered
 - Example: Battelle's Goat challenge
- Search for "RPISEC MBE" then download the challenges.zip from the class site
 - <http://security.cs.rpi.edu/courses/binexp-spring2015/>

