Intro to playing CTFs and Binary Exploitaiton

chiliz

Disclaimer

The opinions and positions expressed are mine only and do not represent the views of any current or previous employer, including Intel Corporation or its affiliates.

This presentation has no intention to advertise or devalue any current or future technology.

whoami

- Lisa / chiliz
- Since recently Security Researcher at Intel STORM Team ☺
- B.Sc in Automation and Mechatronics,

  Thesis about Automated Security Testing & Fuzzing
- CTF Player (when the time allows)

Agenda

- Part1: Intro to CTFs in general
  - Idea of CTFs & Motivation
  - Avoiding Pitfalls I
  - Different CTF Styles
  - Hands-on Example: picoCTF
  - Avoiding Pitfalls II
  - Training Ressources
  - Learning Ressources

- Part2: Intro to Binary Exploitation
  - Stack based Buffer Overflows and Return Oriented Programming

CTF: Capture The Flag

CTF: Capture The Flag

- Gamified approach to learn and train IT Security skills ("Hacking") without doing harm → solving 'real world puzzles'

CTF: Capture The Flag

- Gamified approach to learn and train IT Security skills ("Hacking") without doing harm → solving 'real world puzzles'

- Basic principle:
  - Look at a challenge
  - Find a vulnerability/way to solve the challenge

CTF: Capture The Flag

- Gamified approach to learn and train IT Security skills ("Hacking") without doing harm → solving 'real world puzzles'

- Basic principle:
  - Look at a challenge
  - Find a vulnerability/way to solve the challenge
  - Get the flag (textfile in a specifig format,
    e.g.: FLAG{4wes0me} , FLAG{Y4EY)

CTF: Capture The Flag

- Gamified approach to learn and train IT Security skills ("Hacking") without doing harm → solving 'real world puzzles'

- Basic principle:
  - Look at a challenge
  - Find a vulnerability/way to solve the challenge
  - Get the flag (textfile in a specifig format,
    e.g.: FLAG{4wes0me} , FLAG{Y4EY)

  - Your team gets points for every flag
  - Team with the most points wins the competition

CTF: Capture The Flag

- Consider challenges as untrusted code, if you run/debug them consider doing so in a VM.
- Either have one/two/many VMs with tools installed like here
- Or try out vagrant, which provides easy access to set up machines in < 5 minutes with a CLI

CTF: Capture The Flag

- Very **hands-on**, practical → actually doing something teaches you more than just reading about it

CTF: Capture The Flag

- Very **hands-on**, practical → actually doing something teaches you more than just reading about it
- Unlike entertaining escape rooms or Cluedo, the challenges are based on **real, occuring problems**
  - →The setting in CTFs often is staged and simplified, but most often they have a real core.

CTF: Capture The Flag

- Very **hands-on**, practical → actually doing something teaches you more than just reading about it
- Unlike entertaining escape rooms or Cluedo, the challenges are based on **real, occuring problems**
  →The setting in CTFs often is staged and simplified, but most often they have a real core.
- A way to **learn new** technologies, be creative and play around with them
  - You can use it as a kickstart motivation to look into something
  - Forces you to look deeper (in order to hack something, it's good to know it in-depth)

Avoiding Pitfalls I

- **Set realistic expectations**

Avoiding Pitfalls I

- **Set realistic expectations** → Going from 0 to hero sounds tempting, but don't expect it to happen over night. Hard CTFs are meant to be hard.

Avoiding Pitfalls I

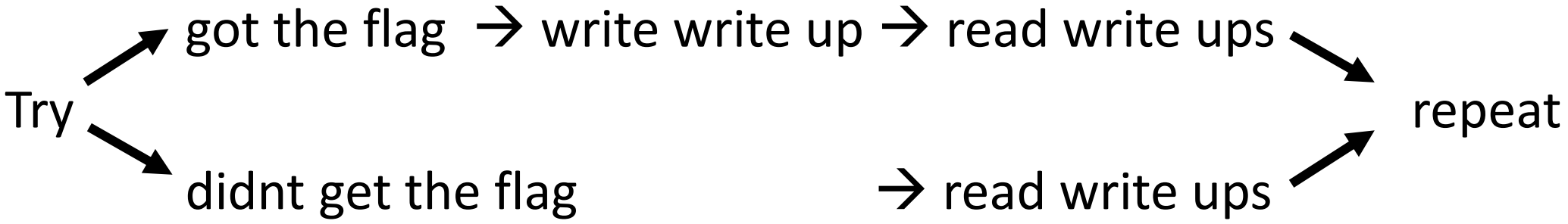
- **Set realistic expectations** → Going from 0 to hero sounds tempting, but don't expect it to happen over night. Hard CTFs are meant to be hard.
- Getting 0 flags in a DEFCON Qualifier is not a sign of failure*, it's just normal,* especially in the beginning

Avoiding Pitfalls I

- **Set realistic expectations** → Going from 0 to hero sounds tempting, but don't expect it to happen over night. Hard CTFs are meant to be hard.
- Getting 0 flags in a DEFCON Qualifier is not a sign of failure*, it's just normal,* especially in the beginning

Avoiding Pitfalls I

- **Set realistic expectations →** Going from 0 to hero sounds tempting, but don't expect it to happen over night. Hard CTFs are meant to be hard.
- Getting 0 flags in a DEFCON Qualifier is not a sign of failure*, it's just normal,* especially in the beginning



Avoiding Pitfalls I

My way was:

- Learn/Level up in junior CTFs

  (harder levels in junior CTFs are the easy ones in some main CTFs)

- Look at challenges from the main CTFs → understanding the challenge is the first challenge → Look at write ups what I got right/wrong, what techniques are there etc.

- Being able to solve something in the main CTFs eventually

Avoiding Pitfalls I

- CTFs are no scavenger hunts. While it might be slightly interesting or even a hint in rare cases, many times the backstory can be abstracted away. Challenge is most often an implementation or logic bug.
- Hints often come in form of a word game in the challenge name, code comments, or exagerrated statements (*This has to be secure now!!11, Here I fixed it, I heard encryption is very important so I wrote my own ...)*

Dont try to see hints where there are none.

- many pictures of flags -> probably a dad joke, not a hint

  -> No need to to look for secret connections between the flag pictures
- No need to translate a generated 'lorem ipsum'

Jeopardy Style CTFs

5 – 8 categories, each has challenges in various difficulties

The more difficult, the more points

WEB	REVERSE ENGINEERING	BINARY EXPLOITATION	CRYPTO	MISC	…
100	100	100	100	100	
200	200	200	200	200	
300	300	300	….		
400	400	…			
500	…				

Jeopardy Style CTFs

5 – 8 categories, each has challenges in various difficulties

Very common: Dynamic points

All challenges start at the highes points

with every solve the worth decreases

→ the less a challenge got solved, the more points it gets

(already earned points decrease too)

Different CTF Styles

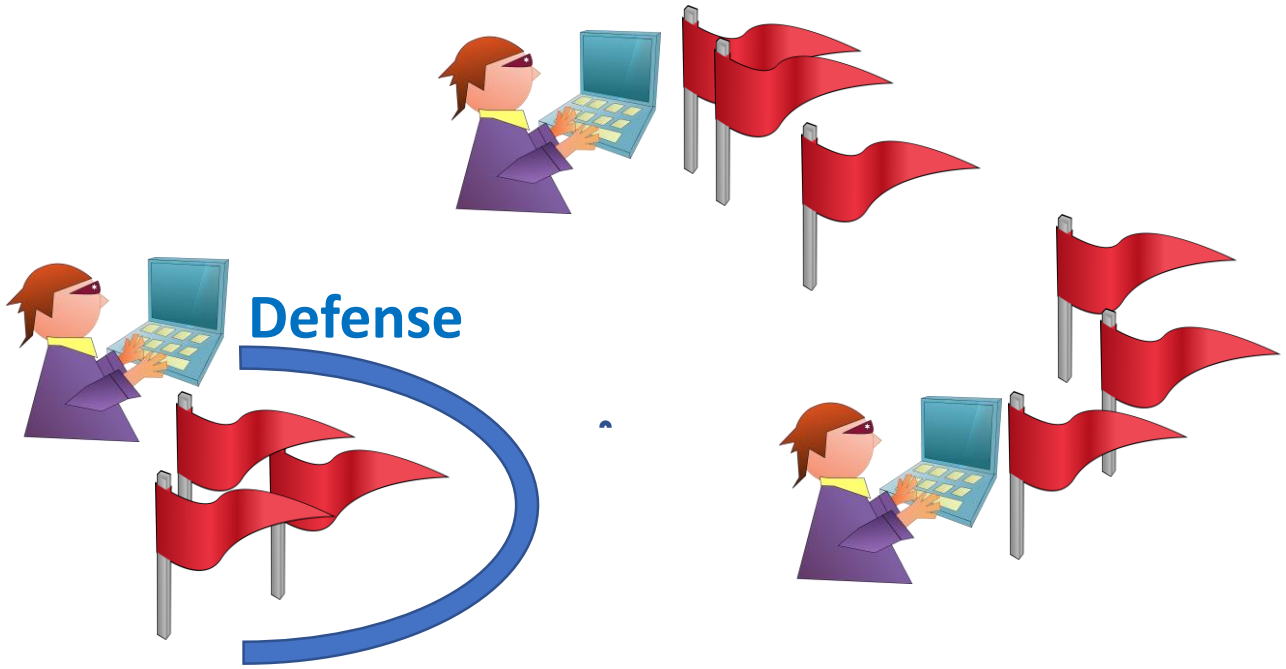
- Beginner CTFs: e.g. Junior C3 CTF, picoCTF, Junior Google CTF
- Jeopardy
- Attack & Defense

Different CTF Styles

- Beginner CTFs: e.g. Junior C3 CTF, picoCTF, Junior Google CTF
- Jeopardy
- Attack & Defense

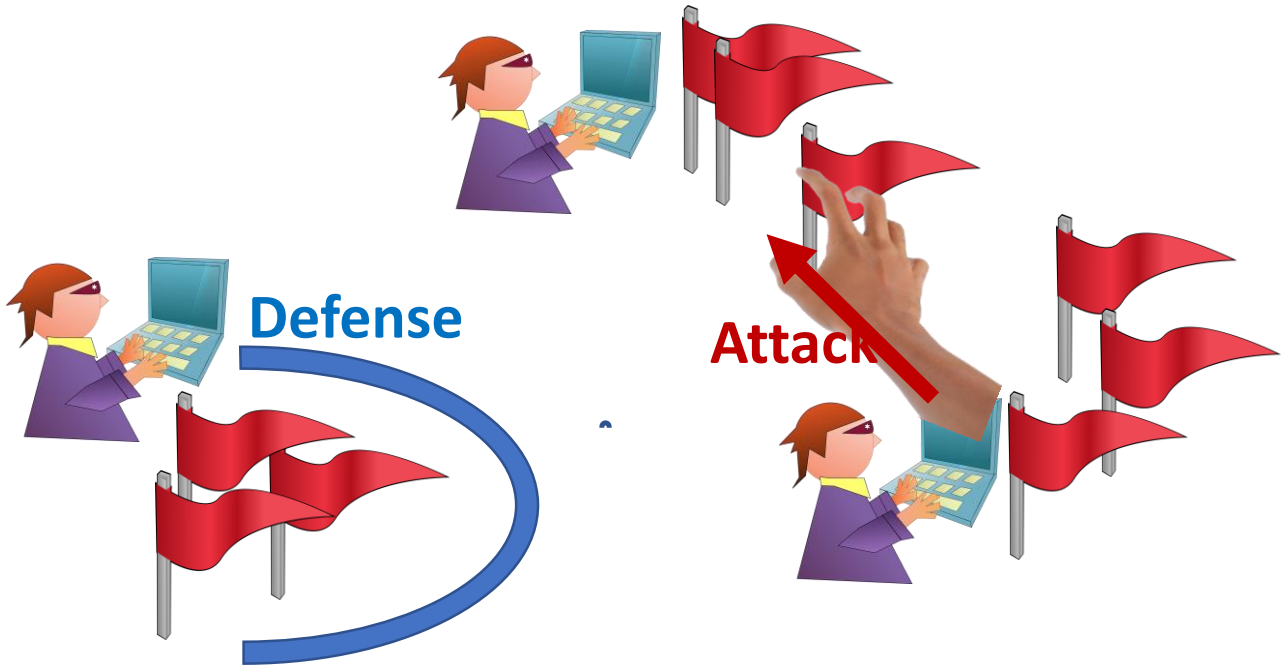
Different CTF Styles

- Beginner CTFs: e.g. Junior C3 CTF, picoCTF, Junior Google CTF
- Jeopardy
- Attack & Defense



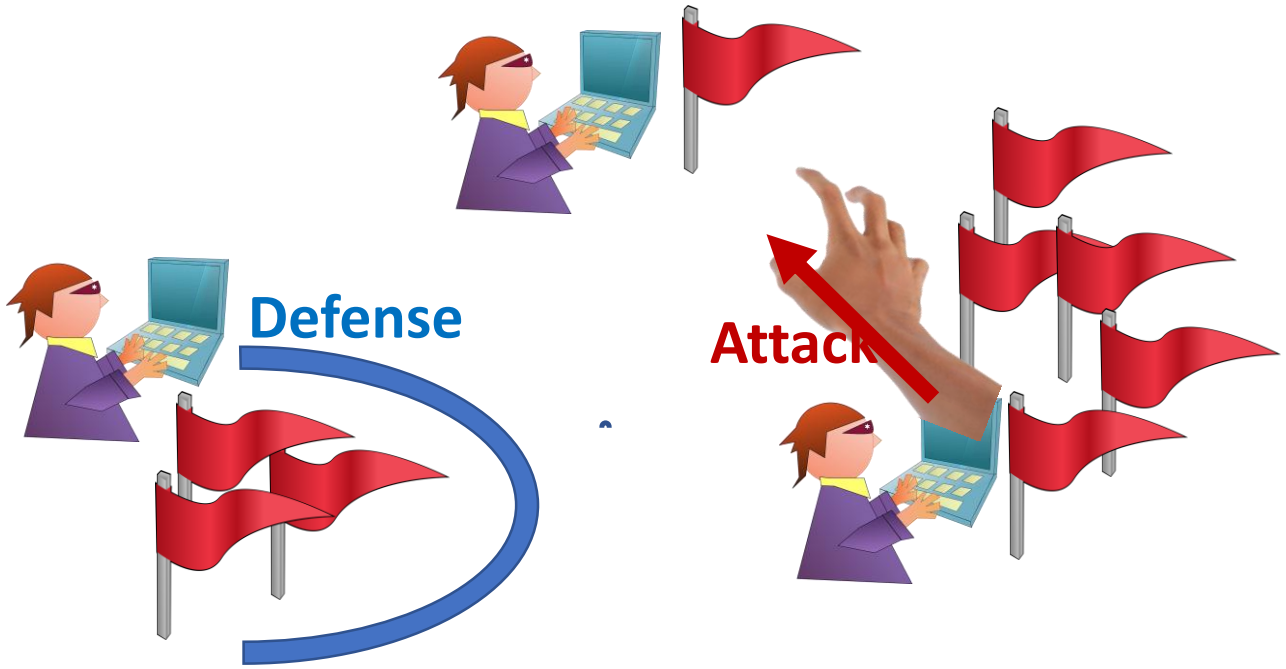
Different CTF Styles

- Beginner CTFs: e.g. Junior C3 CTF, picoCTF, Junior Google CTF
- Jeopardy
- Attack & Defense



Different CTF Styles

- Beginner CTFs: e.g. Junior C3 CTF, picoCTF, Junior Google CTF
- Jeopardy
- Attack & Defense



Different CTF Styles

- Beginner CTFs: e.g. Junior C3 CTF, picoCTF, Junior Google CTF
- Jeopardy
- Attack & Defense

- "special competitions" (e.g. 2-4 weeks instead of 48 hours):
  - FlareOn (Reversing, 6 weeks)
  - Advent/Easterhacking (OverTheWire Advent)

Experiment with Inputs and Outputs

- In challenges with an input (web, binary exploitation..), it is often helpful to analyze
  - Where/How is the input being processed? *(source)*
    Is it validated / sanitized?
  - Is there an interesting target function can be reached? /
    Where would it be interesting that the data could land? *(sink)*
  - Bring those two together

Experiment with Inputs and Outputs

Interesting Inputs Web:

- Any sequence of special characters
- Change request, analyze response

Interesting Inputs Binary Exploitation:

- A lot of of characters (e.g. 400 or 2000) to check for a Buffer Overflow
- If a number is asked, try a negative number or a giant number and check for strange behaviour (Integer Overflow/Underflow/Wraparound)
- %x , to check for a formatstring attack, if an address is printed (large integer number), this is an information leak

Experiment with Inputs and Outputs

Interesting Outputs:

- Any type of crash (or error message, in web)
- Any type of different behaviour from the expected
- similar inputs leading to different outputs
- Timing differences regarding processing data (web)

→Often an exploit is just a crafted, very specific sequence of input combinations, that makes use of their unintended outcome

→Sometimes to even to see a crash, a very specific input is needed, so sometimes a crash is hard to find.

Tools

- Spend more time understanding the problem/challenge than understanding the tool

  → Learn concepts, not shortcuts

Tools

- Spend more time understanding the problem/challenge than understanding the tool
  → Learn concepts, not shortcuts
- Tools can help with understanding what is going on, though

→ makes life easier

For beginner CTFs often additional tools are not ultimately needed (e.g. web challenges can be done in the browser), but visualization helps the understanding

Tools

Forensics:	Traffic-Analyzer	**Wireshark (.pcap)**
Web:		**Burp Suite, browser Developer tools**
Reversing:	Disassembler like	**IDA, ghidra**
		Hexeditor (e.g. for binary patching)
		Jadx for Java-Reversing
RE &		
Binary Exploitation:	Debugger like	**gdb-peda, gdb-gef (Linux)**
		Windbg(Windows)
	python-library	pwntools

Tools

- Automation Tools:

  My advice: Do it at least one time yourself, before using an out-of-the-box automated tool like SQLMap for SQL injections

A lot of CTF Challenges are designed in a way, that prevent just running a kali linux tool anyways, to keep it interesting → either you can configure it accordingly, or you have to find your own way.

(Pentesting training platforms like HackTheBox are different since they focus on a more realistic scenario, the usage of automated tools make more sense there)

Categories: Web

Hands-On picoCTF:

starts with looking at source code of site, developer tools etc.
check for robots.txt
session handling (check cookies etc.)

At a later point:

- tools like Burp Suite help to analyze the ongoings
- common attacks: SQL Injections, XSS, enumeration, CSRF

https://www.owasp.org/index.php/Germany/Projekte/Top_10
https://github.com/OWASP/CheatSheetSeries/tree/master/cheatsheets

Categories: Web

Tools:

Good working combination: Burp Suite, FoxyProxy, Cookie Quick Manager

Start analyzing:

FoxyProxy → Use Patterns

Burp Suite → Start capturing

→ Turn intercept off

Load Challenge website

Burp Suite → Tab History → find interesting entry

Burp Suite → Interesting entry → Send to repeater

→ change Tab to repeater

→ vary your request on the left, see if it changes something in the response (right)

Categories: Crypto

Beginner CTF: Ciphers like Caesar Ciphers, rot13 etc → online tools

Medium: attacking the implementation of a crypto algorithm

- Check the padding
- What is the source of randomness? Can it be influenced?
- Is the salt correctly done etc.

Hard: attacking the crypto (Math!)

Categories: Miscellaneous (Misc) & Stegano

- Misc:
  - Funfacts, challenges that did not fit in other categories, strange bugs
  - A lot of googling helps sometimes

- Steganography:
  - Hiding information (here: the flag) in valid cover data
  - Often used with pictures: the information is hidden in a specific part of a pixel. There are tools that sometimes help.

Categories: Reverse Engineering

- The given in reverse engineering is the endproduct, and the goal is to find out how it was constructed, or retrieving information out of memory.

- Very often this is an executable file, and what needs to be found is the logic in a higher (pseudo)language, e.g. Reconstructing pseudo C code of disassembly

Easy challenges: strings, file, hexdump etc.

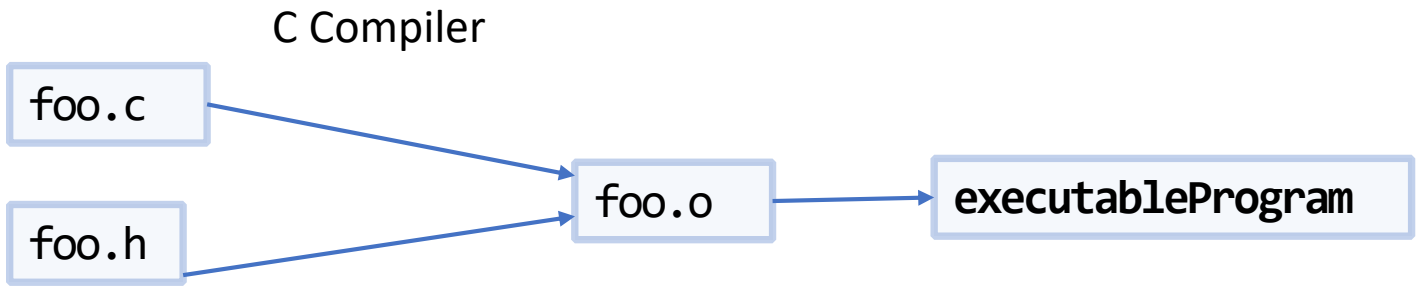
Other: open in graphic disassembly view, e.g. IDA

Intro – Assembly Basis

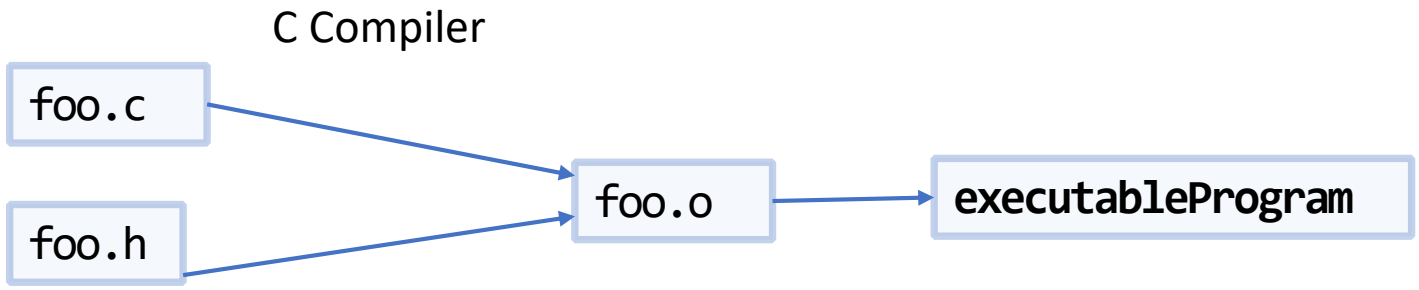
- Low-level programming language
- Every CPU architecture has its own assembly language
  - Intel x86 (32bit) or x86_64 (64bit)
  - ARM
- Usually not written by hand.
  A compiler is used to translate high-level Code

  (e.g. C) into assembler

Intro – C Programs to Assembly



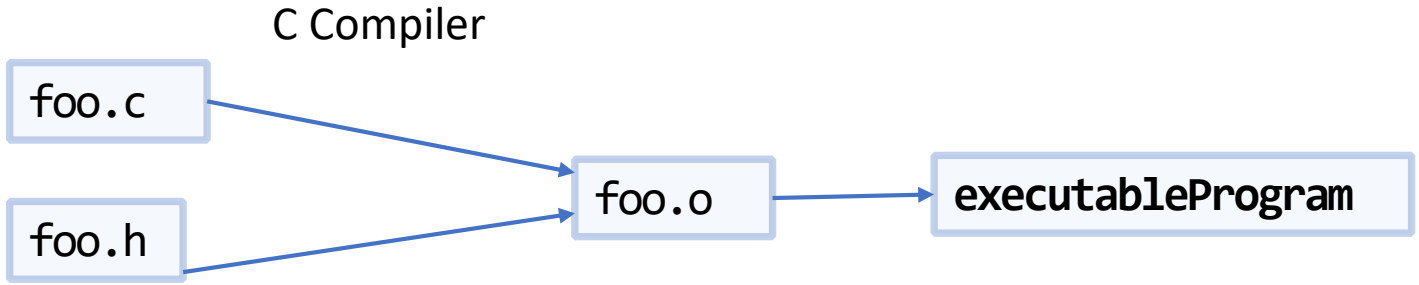
Intro – C Programs to Assembly




```
int a = 1;
int b = 2;
int c = a+b;
```

```
010110010101001 …
```

Intro – C Programs to Assembly



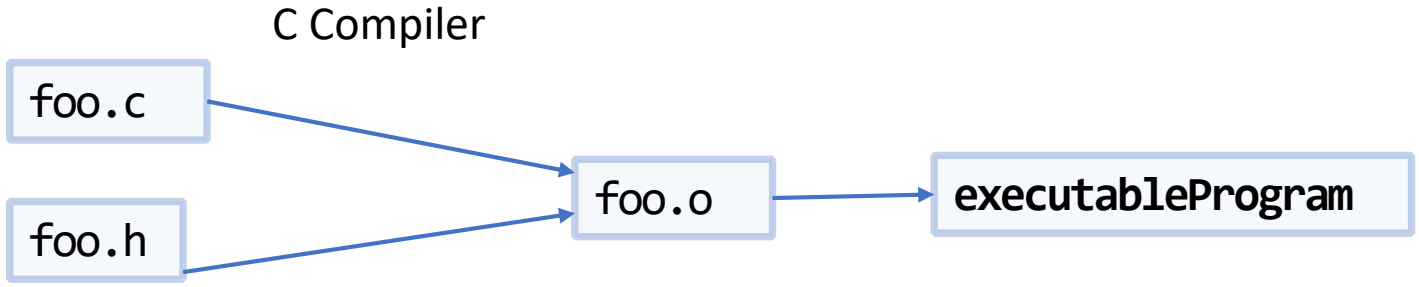

```
int a = 1;
int b = 2;
int c = a+b;
```

Assembling

```
Assembly Code:
    mov rax, 1
    mov rbx, 2
    add rax, rbx
```

```
010110010101001 …
```

Intro – C Programs to Assembly




```
int a = 1;
int b = 2;
int c = a+b;
```

010110010101001 …

Assembling

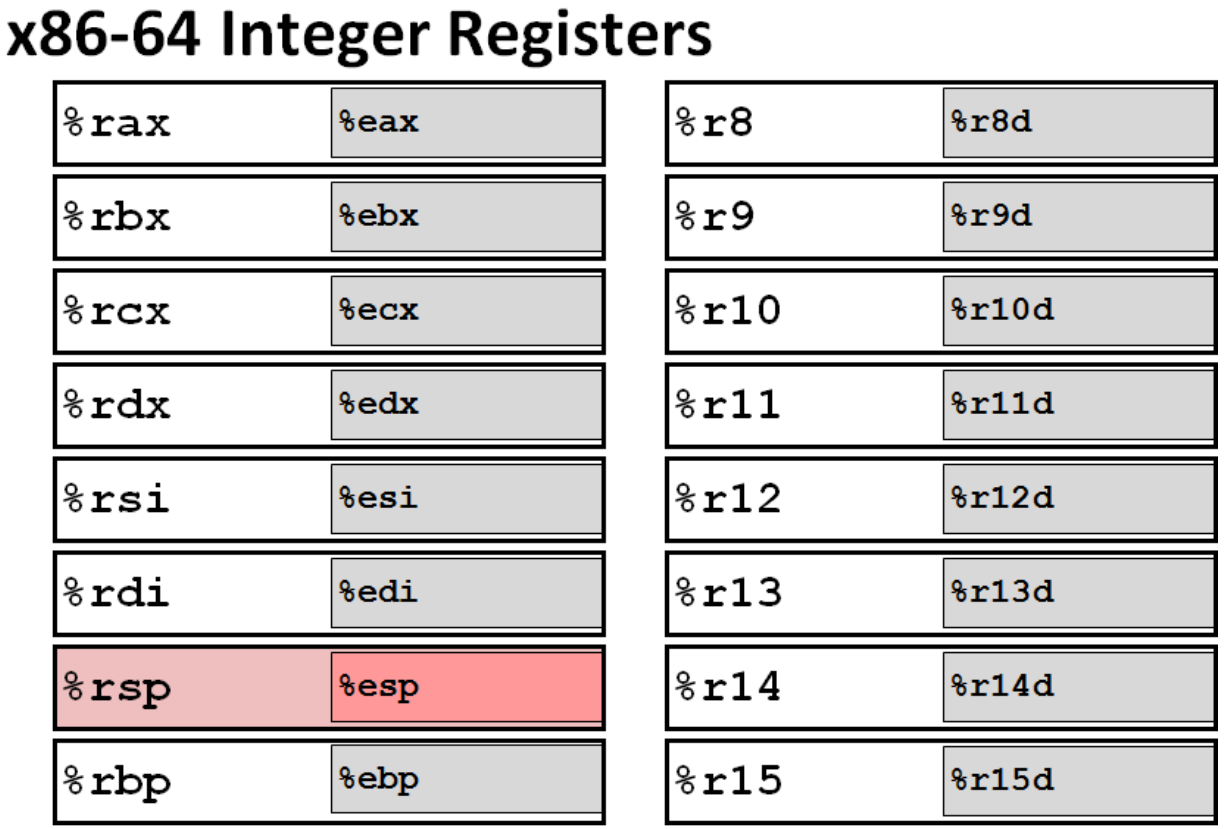
Assembly Code:

```
mov rax, 1
mov rbx, 2
add rax, rbx
```

Disassembling

CPU Registers (General Purpose)

- Small memory locations
  inside the CPU
- Very fast
- Used for local variables
  and other important values



- Size of a register depends on architecture
  (8 Byte for Intel x86_64)

Intro – Assembly Basis

- Instructions consist of:
  - Commands
  - Operands
- Can be translated to binary code (1:1 translation, reversible)

Intro – Assembly Basis

- Instructions consist of:
  - Commands
  - Operands
- Can be translated to binary code (1:1 translation, reversible)

mov eax, 1

binary code: 10111 000 01000000

(opcode mov) (eax) (value in little endian)

64 Bit – Calling convention Linux

- Function Arguments are stored in `RDI, RSI, RDX, RCX, R8, R9, XMM0–7` (in this order)
- Return value of a function is stored in RAX

`Important registers:`

- `RIP: Instruction Pointer`
- `RSP: Current Top of the Stack`

64 Bit – Calling convention Linux

- Move 2nd function argument in **RSI**
- Move 1st function argument in **RDI**

- Call to function
  - save **return address** on the stack to return to it later
  - Function gets executed
  - return to the address that is saved to the stack

- Execution continues, return value of function in **RAX**

Categories: Binary Exploitation

- Will be covered in Part 2!

Categories: Binary Exploitation

- Will be covered in Part 2!

Avoiding Pitfalls II

- Care for Team Communication (Pads, Discord/Mattermost, for each challenge a thread, linking them in the main thread works very well)
- Find a balance between "hopping from challenge to challenge" and "this is the only challenge I look at for 48 hours"

  (staring it to death really helps sometimes, though).

- Sleep. Breathe. Take Breaks.

- **Read the Write Ups Afterwards. Even if you never want to look at it again.**
  - If you want to look at the challenge after the CTF, before reading a write up, set a time limit for yourself. If you haven't looked at it in 10 months, will you really ever do it?

Training ressources

- CTF challenges availabe 365 days/year: https://picoctf.com , https://ringzer0ctf.com
- Crack me sites (Reversing): https://crackmes.one/
- Binary exploitation course: https://github.com/RPISEC/MBE
- Wargames (e.g. https://overthewire.org , https://pwnable.kr): Binary exploitation,
  - Exploit challenge0 to get from user0 → user1,
  - Exploit challenge1 to get from user1 → user2

  ...
  - Exploit challenge8 to get from user8 →root

- Hackthebox:
  - own the vulnerable machine.
  - pentesting-training: checking against vulnerable versions, looking at the whole system etc.
  - Youtube channel of IppSec has great walkthroughs: https://www.youtube.com/channel/UCa6eh7gCkpPo5XXUDfygQQA

Learning ressources

Streaming channels:

LiveOverflow Youtube Channel (Binary Series, CTF, all of them!)

GynvaelEN Youtube Channel: picoCTF walkthroughs (2019 &2018)

Ippsec Youtube Channel

MalwareTech Youtube Channel

RE- Online Workshops

https://www.begin.re/ (Ophir Harpaz)

https://malwareunicorn.org/workshops/ (Amanda Rosseau)

Damn Vulnerable Web App List

https://www.owasp.org/index.php/OWASP_Vulnerable_Web_Applications_Directory_Project/Pages/Offline