

IT-SECURITY CONFERENCE
TROOPERS18



REVERSE ENGINEERING
A (M)MORPG
Antonin Beaujeant

Author

Name: Antonin Beaujeant (@beaujeant)

Job: Pентest - R&D

- Previous
- Now



Interest:

- Reverse engineering
- Hardware hacking
- CTF



Agenda

- Introduction
- Game structure
- Define targets
- Find local saved data
- RE network protocol
- Building a Wireshark dissector
- Building an asynchronous proxy
- RE binary
- Binary patching
- Library hooking



Introduction



Introduction

Game: Pwn Adventure 3

Genre: (M)MORPG - Team Adventure Quest

Developer: Vector35 (<https://vector35.com>)

Game Engine: Unity (cross-platform)

Event: Ghost In The Shellcode CTF 2015 (ShmooCon)

RE complexity: Medium

- no obfuscation (~~packer, stripped, sandbox detection~~)
- no signature/encryption
- all-in-one logic file



Introduction

Client install:

- Go to <http://pwnadventure.com/#downloads>
- Select your platform (Windows, macOS or Linux)
- “chmod” if needed
- Run the launcher (download ~2.0 Go)
- Click “Play”
- (Windows) Instal DirectX



Introduction

Configure client:

- Edit MasterServer in server.ini
 - macOS: Pwn Adventure 3.app/Contents/PwnAdventure3/PwnAdventure3.app/Contents/UE4/PwnAdventure3/Content/Server/server.ini
 - LINUX: PwnAdventure3_Data/PwnAdventure3/PwnAdventure3/Content/Server/server.ini
 - Windows: PwnAdventure3_Launcher_Windows/PwnAdventure3_Data/PwnAdventure3/PwnAdventure3/Content/Server

```
[MasterServer]
Hostname=pwn3.server
Port=3333
```



Introduction

Configure hosts file:

- macOS: `sudo sh -c "echo '10.20.12.5 pwn3.server' >> /etc/hosts"`
- Linux: `sudo sh -c "echo '10.20.12.5 pwn3.server' >> /etc/hosts"`
- Windows
 - Go to `C:/Windows/System32/drivers/etc/`
 - Right click hosts > Open with > Notepad
 - Add the following line

```
10.20.12.5      pwn3.server
```



Introduction

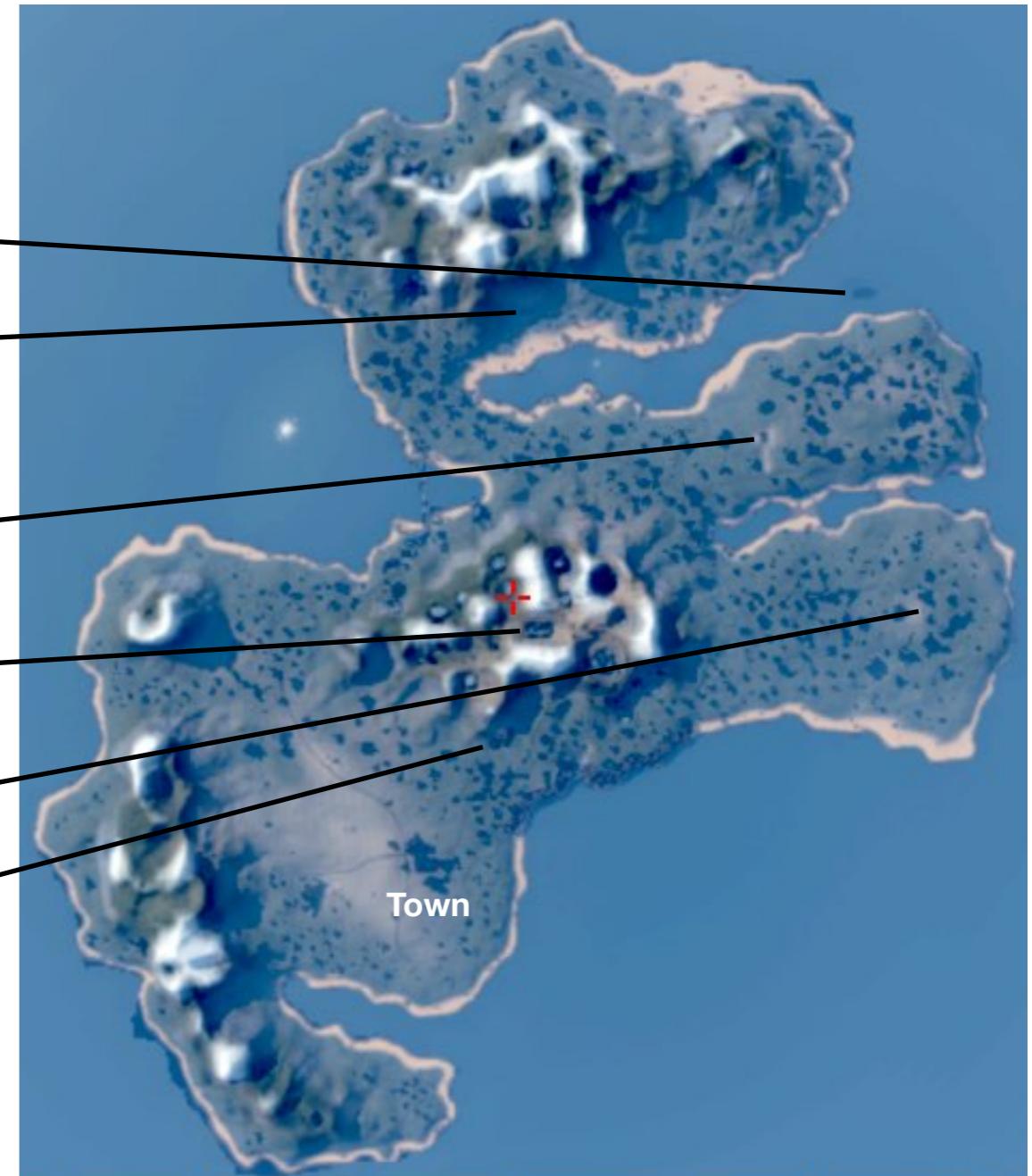
Gameplay: <https://youtu.be/PHZJ443zVM0>



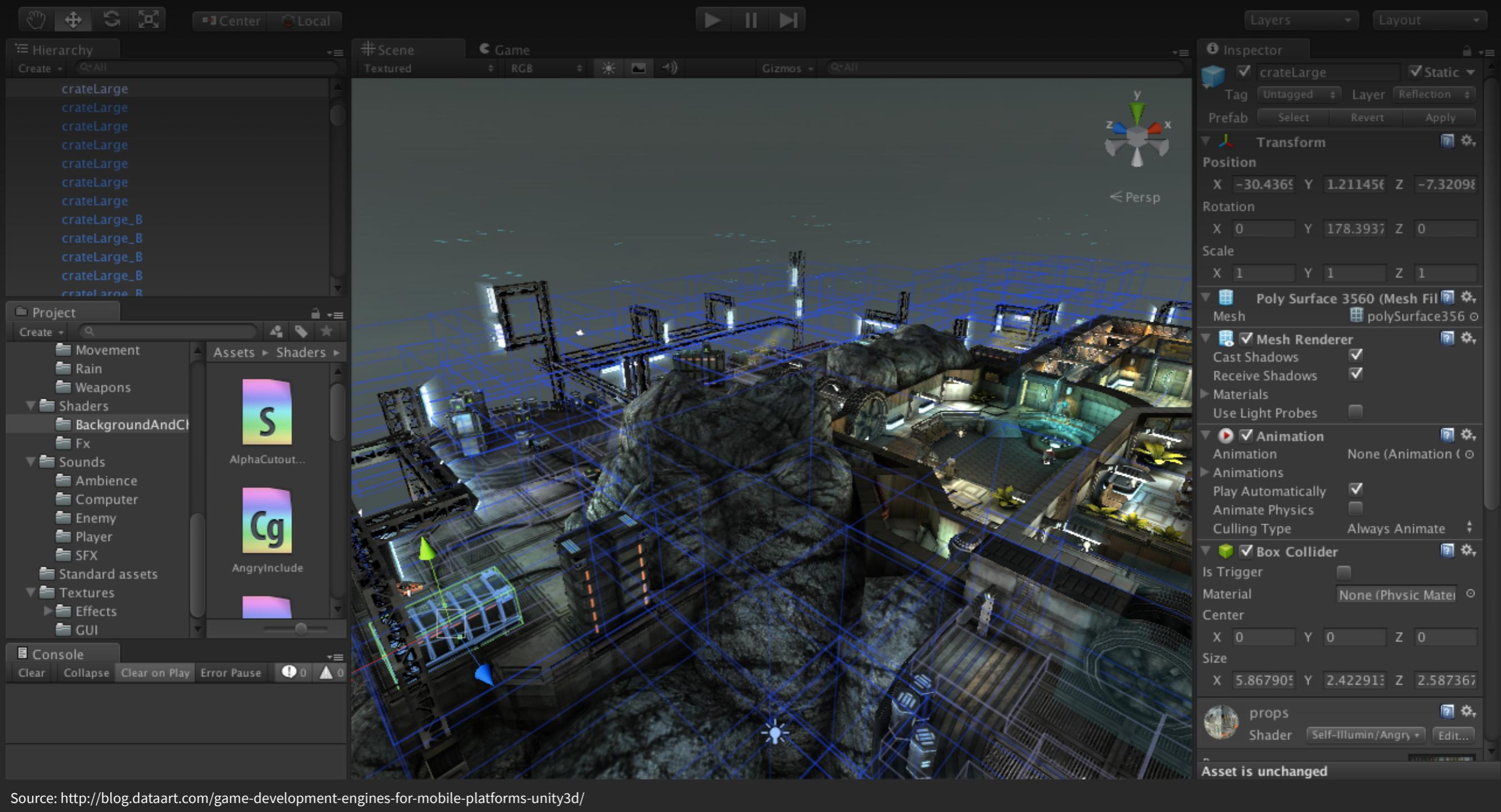
Introduction

List of quests:

- *Pirates Treasure (crack me)*
- [Fire and Ice \(RE binary\)](#)
- [Until the Cows Come Home \(binary patching\)](#)
- [Egg Hunter \(RE network & RE binary\)](#)
- [Unbearable Revenge \(RE network\)](#)
- *Blockys Revenge (logic gate)*
- *Overachiever (finish all achievements)*



Game structure



Source: <http://blog.dataart.com/game-development-engines-for-mobile-platforms-unity3d/>



Game structure

Pwn Adventure 3 structure:

Master game server

- Login
- Team
- Characters
- Assign instance

Game server

- Player location
- Enemy/NPC location

Server

Pwn Adventure 3

- Application
- Logic
- View

Client



Game structure

Application layer:

- Loading the game
- Receiving user inputs
- File/memory management
- Network communication

Game **logic**:

- Events (if player kill/pick up/... , then ...)
- States and data (items, NPC, enemy, player, quests, etc)
- Physics (gravity, hit box, movement, collision with wall, etc)

Game **view**:

- Graphic engine
- Audio



Game structure

Offline game:

- All files and executions are done locally
- Full control over the game
- Obfuscation/anti-RE is the *only* obstacle

Online game:

- Client game logic
- Server game logic
- Bi-directional network communication
- Unknown logic on server side
- Uncontrolled logic on server side



Define Targets



Source: For Honor (Ubisoft)



Define targets

Why reverse video game?

Fun 

- Modding: New weapons, maps, missions, skins, etc
- Modding: New features
- Find cheat codes

Profit 

- Make the game easier
- Advantage over other players
- Remove DRM
- Generate money (in game - real life)



Define targets

OFFLINE

Cheat against yourself
Defeat your own machine
You control your environment

VS

ONLINE

Cheat against other players
Defeat the host server
Outsmart others' security



Define targets

Top down approach

Play the game to identify what is valuable:

- Items (coins, weapons, spells, etc)
- State (quest unlocked, being level 42, etc)
- Increase specs (damage x10, health +1000, etc)
- Enhance capabilities (run faster, see through wall, jump higher, etc)

Identify where it is used, then reverse and exploit.

Bottom-up approach

Reverse the binary/network to identify potential weaknesses



Define targets

Let's play!



Define targets



- Spawn
- ○ Quests (workshop)
- Quests (not part of workshop)
- Fast travel



Define targets

Targets for this workshop:

- Spawn wherever we want
- Generate items/money
- Run faster
- Jump higher
- Teleport anywhere
- Increase damage
- Be invincible
- Kill instantly
- Be admin
- RCE on server



Define targets

Targets for this workshop:

- Spawn wherever we want
- ~~Generate items/money~~
- Run faster
- Jump higher
- Teleport anywhere
- ~~Increase damage~~
- ~~Be invincible~~
- ~~Kill instantly~~
- ~~Be admin~~
- ~~RCE on server~~
- Pick up remote items
- Find secret to unlock quest
- Find vulnerability to kill boss



Define targets

Where to look?

- Network communication
- Local saved data
- Client game logic binary
- Server game logic binary
- Rendering engine



Define targets

What to do?

- RE network protocol
- Edit local saved data
- RE game logic
- Patch binary/Hook libraries
- Edit rendering engine
- Build bots (automate task to be faster and more accurate)



Find local saved data



Image © Avon Fox
www.the-liberator.net
Image may be used unaltered
with this watermark intact.



Find local saved data

Saved data might contain **items** and **states**.

 Could be **encrypted** and/or **signed**.

Should be *most likely* **present** in **offline** game.

Should be *most likely* **not present** in **online** game.



Find local saved data

Techniques to **identify** files:

- File system comparison
 - Create a snapshot before specific action
 - Execute specific action (pick up item, gain experience points, etc)
 - Create another snapshot after specific action
 - Compare snapshots
- “Spying” file manipulation calls
 - Run the game
 - Attach a “spying” app on the process (game)
 - Filter logged calls (e.g. file manipulation only)
 - Execute specific action
 - Analyse collected logs



Find local saved data

Analyse modified files:

- Look at the filename and extension
- Look at the content
 - Encrypted?
 - Binary?
 - ASCII?
- Compare before vs after for changes
- Understand format and content
- Debug the binary and reverse functions that modify the file



Find local saved data

Lab 1.1: Find local saved data (Linux)

Find new/deleted files:

- `find / -type f -o -path /proc -prune > snapshot1`
- `find / -type f -o -path /proc -prune > snapshot2`
- `diff -crB snapshot1 snapshot2 > changes`

Find edited/removed files:

- `find / -path /proc -prune -o -path /sys -prune -o -path /usr -prune -o -type f -print0 | xargs -0 md5sum | tee md5sum.txt`
- `md5sum -c md5sum.txt 2> /dev/null | grep -i 'FAIL' > failed.txt`



Find local saved data

Lab 1.2: Find local saved data (Windows)

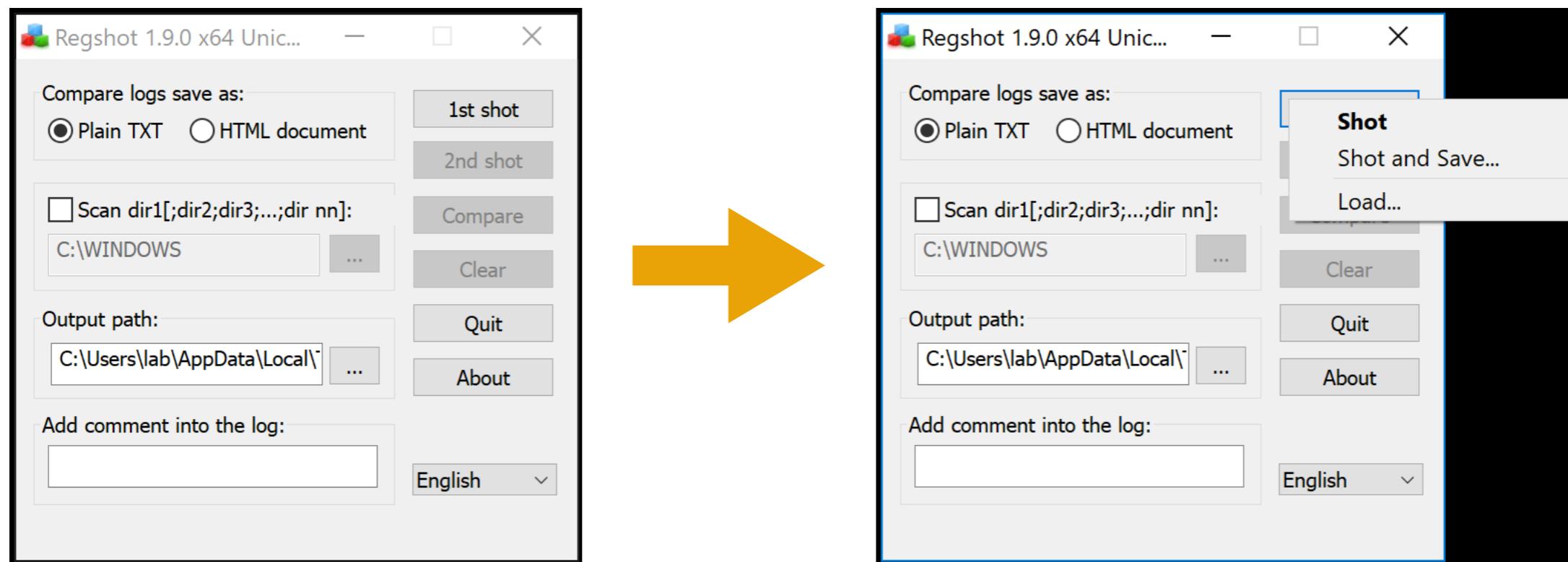
Use *regshot* to build the snapshots and make the comparison



Find local saved data

Lab 1.2: Find local saved data (Windows)

Create the first snapshot



Find local saved data

Lab 1.2: Find local saved data (Windows)

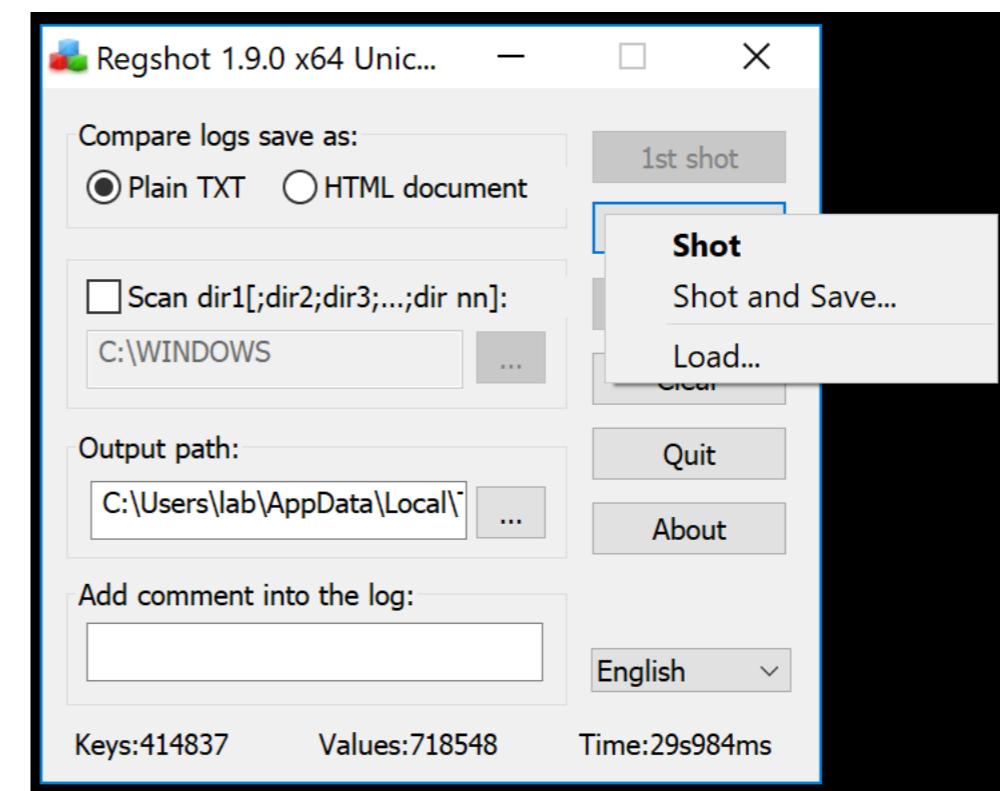
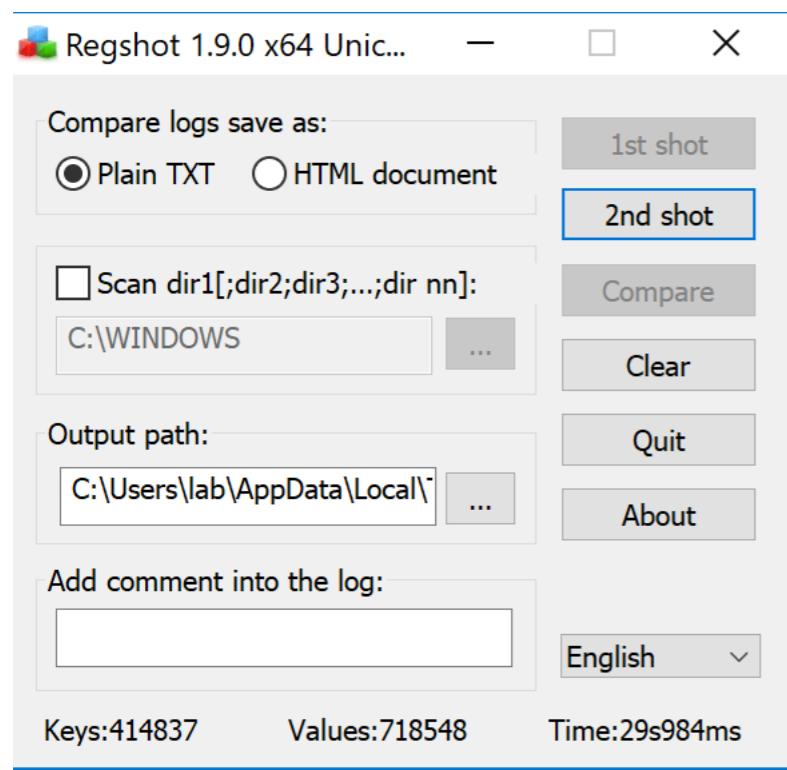
Run the game and execute specific action (e.g. purchase new items)



Find local saved data

Lab 1.2: Find local saved data (Windows)

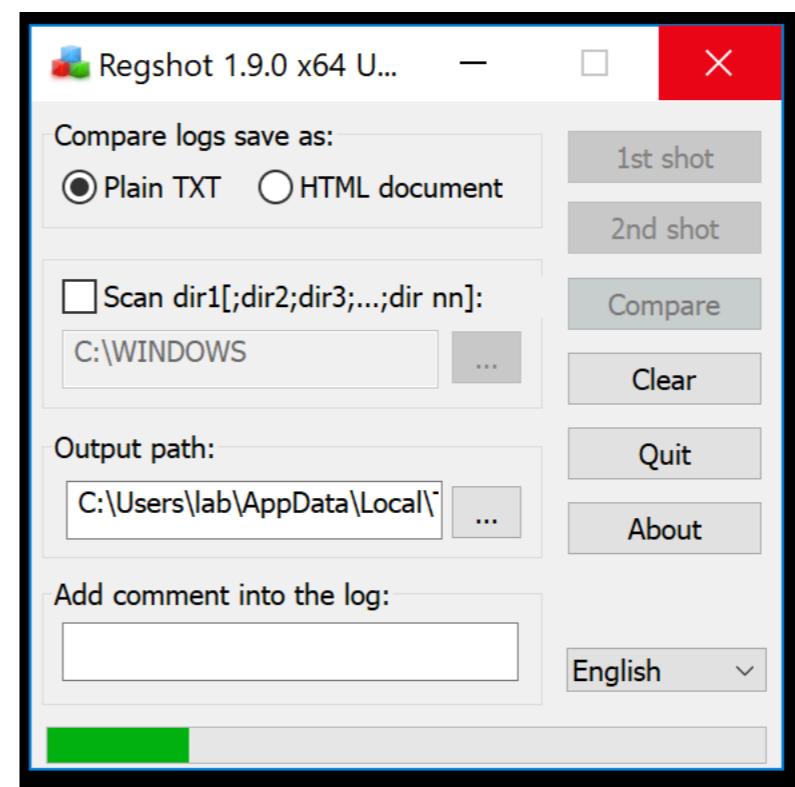
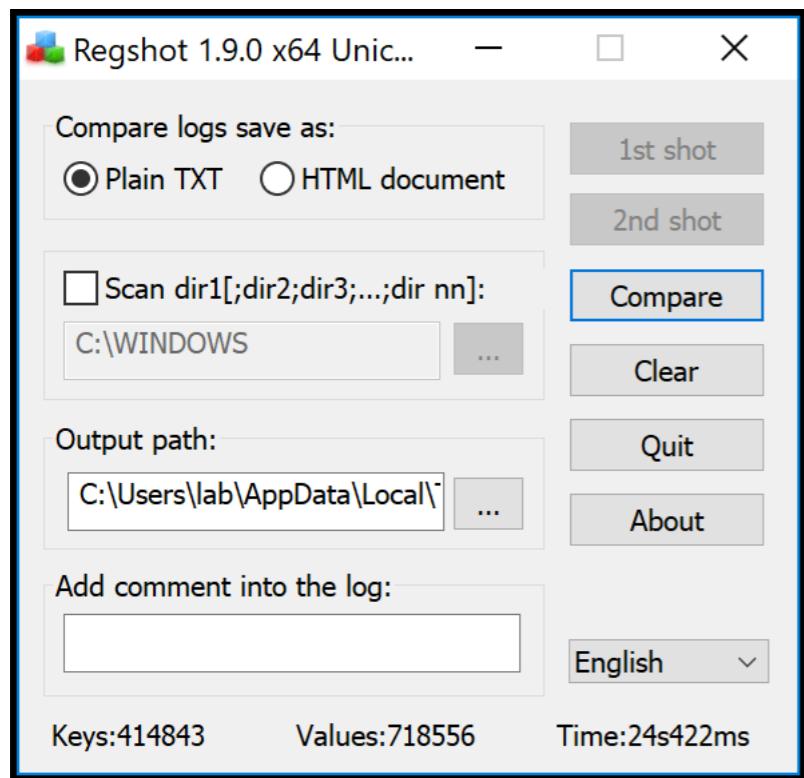
Create the second snapshot



Find local saved data

Lab 1.2: Find local saved data (Windows)

Compare the two snapshots



Find local saved data

Lab 1.2: Find local saved data (Windows)

Analyse result



RE network protocol

Wi-Fi: en0

src == 192.168.1.238

Time | Source | Destination | Protocol | Length | Data

08 96.862633 192.168.1.205 192.168.1.238 TCP 3000 88 6d7674fe4bc71757c734139d437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
10 96.937893 192.168.1.205 192.168.1.238 TCP 3000 88 6d76fb14bc7225a1ec734139d437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
12 97.010718 192.168.1.205 192.168.1.238 TCP 3000 88 6d763fa64bc71c5d1ec734139d437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
14 97.083565 192.168.1.205 192.168.1.238 TCP 3000 88 6d76fb7c4bc7ec5f1ec734139d437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
17 97.151585 192.168.1.205 192.168.1.238 TCP 3000 88 6d76fb7c4bc7ec5f1ec734139d437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
19 97.219354 192.168.1.205 192.168.1.238 TCP 3000 120 2a691000477265617442616c6c734f6646697265800b07c0... 53931 → 3000 [PSH, ACK]
21 97.290338 192.168.1.205 192.168.1.238 TCP 3000 88 6d761d274bc7ce651ec74b25a2437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
23 97.359156 192.168.1.205 192.168.1.238 TCP 3000 88 6d7637ff4ac78b681ec79fefafa437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
25 97.436007 192.168.1.205 192.168.1.238 TCP 3000 88 6d7645d44ac7816b1ec7f621a7437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
27 97.510222 192.168.1.205 192.168.1.238 TCP 3000 88 6d7613a74ac79d6e1ec722eea8437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
28 97.510932 192.168.1.205 192.168.1.238 TCP 3333 140 17030100204d6bceab82dfba52bf542184133f63a64f6d26... 53881 → 3333 [PSH, ACK]
30 97.576290 192.168.1.205 192.168.1.238 TCP 3000 88 6d767a7b4ac79c711ec71055aa437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
... 97.650184 192.168.1.205 192.168.1.238 TCP 3000 88 6d76ac524ac769741ec7046cab437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
34 97.725777 192.168.1.205 192.168.1.238 TCP 3000 120 2a691000477265617442616c6c734f6646697265800b07c0... 53931 → 3000 [PSH, ACK]
36 97.801167 192.168.1.205 192.168.1.238 TCP 3000 88 6d7667f549c7d17a1ec7114cad437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
38 97.871931 192.168.1.205 192.168.1.238 TCP 3000 88 6d76e4cc49c7997d1ec71ae5ad437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
40 97.943259 192.168.1.205 192.168.1.238 TCP 3000 88 6d767ca149c791801ec76268ae437ffe36fd00007f00 53931 → 3000 [PSH, ACK]
42 98.020624 192.168.1.205 192.168.1.238 TCP 3000 88 6d767a7349c7ba831ec780b7ae437ffe36fd00007f00 53931 → 3000 [PSH, ACK] Se

frame 2832: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface
Ethernet II, Src: Apple_d1:ee:ef (c4:b3:01:d1:ee:ef), Dst: Vmware_c9:82:76 (00:0c:29:c9:82:76)
Internet Protocol Version 4, Src: 192.168.1.205, Dst: 192.168.1.238
Transmission Control Protocol, Src Port: 53931, Dst Port: 3000, Seq: 3016, Ack:
Data (22 bytes)

Data: 6d76ac524ac769741ec7046cab437ffe36fd00007f00
[Length: 22]

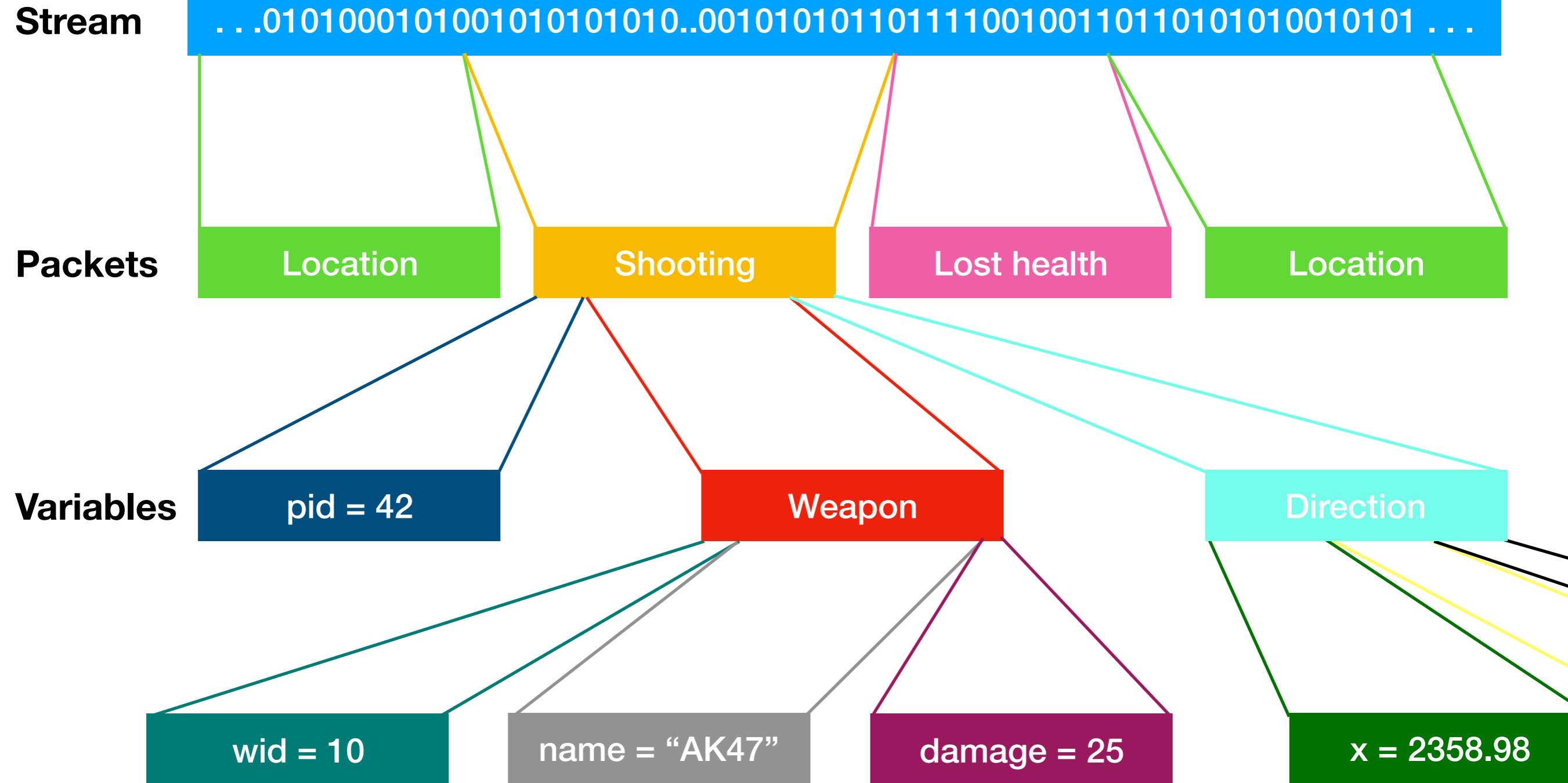
0000 00 0c 29 c9 82 76 c4 b3 01 d1 ee ef 08 00 45 00 ...v
0010 00 4a b9 5e 40 00 40 06 fc 43 c0 a8 01 cd c0 a8 ..J.^@..
0020 01 ee d2 ab 0b b8 50 14 83 04 6e 46 df ed 80 18
0030 10 00 8c 21 00 00 01 01 08 0a 35 f8 cf 3f 00 04 ...!..
0040 7e 0d 6d 76 ac 52 4a c7 69 74 1e c7 04 6c ab 43 ~.mv.R
0050 7f fe 36 fd 00 00 7f 00 ..6...

Data (data.data), 22 bytes

Packets: 7080 · Displayed: 3025 (42.7%) · Dropped: 0 (0.0%) · Profile: Def



RE network protocol



RE network protocol

Stream/data expected:

- Encrypted
- Encoded
- Clear text
 - Printable character
 - Binary



RE network protocol

Packet **Format** expected:

- Known format
 - HTTP: `username=john&password=letmein` (POST /login/)
 - JSON: `{"login": {"username": "john", "password": "letmein"}}`
 - XML: `<login><username>john</username><password>letmein</password></login>`
 - More...
- Unknown format
 - No identifier: both the client and the server know the sequence
 - Identifier: when sequence order is not predictable



RE network protocol

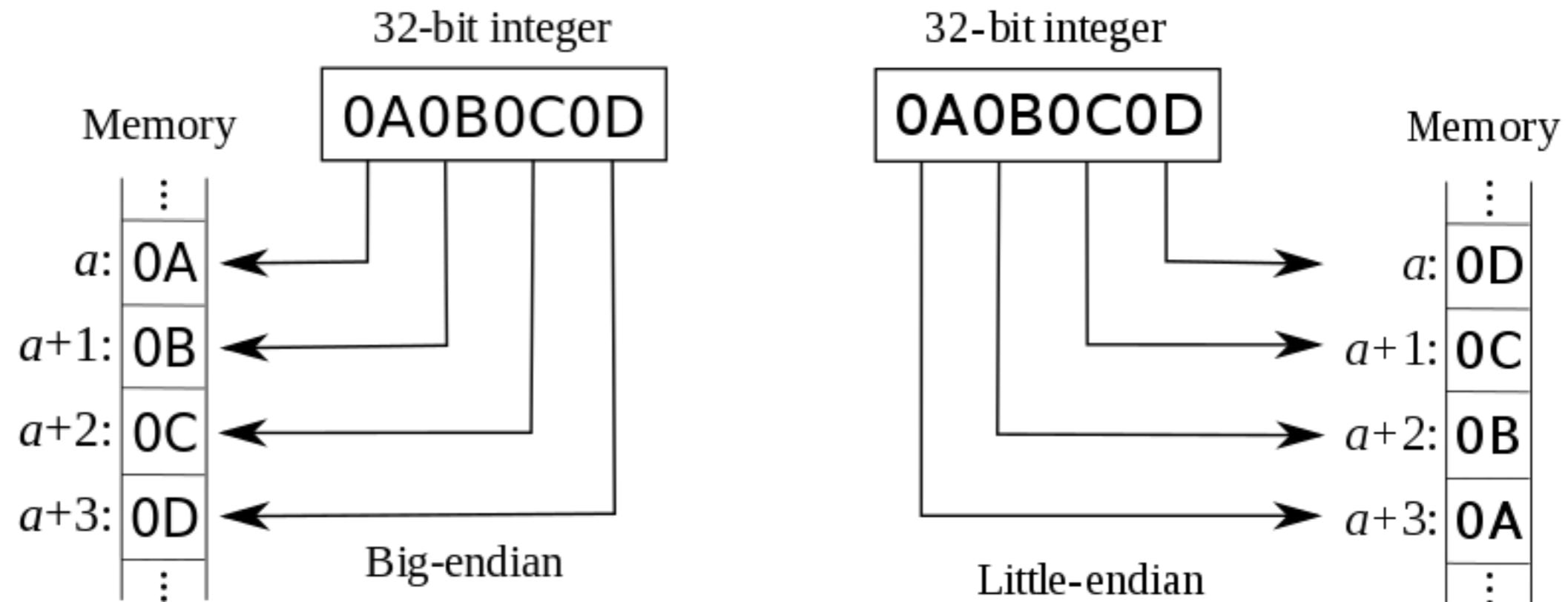
Variables expected:

- Integer: e.g. 265845
- Float: e.g. 125.89654
- Boolean: e.g. True
- String: e.g. “Hello World!”
 - Known length
 - Delimiter (e.g. NULL termination, or padding)
 - Announced length
- Struct (like in C)



RE network protocol

Endianness



RE network protocol

Lab 2: Identify game communication and format

Use the protocol analyser *Wireshark* to capture your network traffic. Open the game, create a new account, create a new character and start playing Pwn Adventure 3.



RE network protocol

Network communication:

- Master game server (TCP/3333)
 - SSL/TLS encrypted
 - Authentication
 - Character creation
 - Changing areas
- Game server (TCP/30xx)
 - Clear text
 - Binary
 - Traffic generated constantly when playing the game



RE network protocol

Let's focus on the **Game server** (TCP/30xx)



RE network protocol

Methodology for *binary* protocol using *unknown* format:

- Understand your target
- Identify IP(s), port(s) and content (encrypted? Format?)
- Define objective (e.g. location packet, new item packet, etc)
 - Build use cases (e.g. don't move, shoot, etc) based on assumptions to isolate the packet/variable
 - Look at network behaviour (changes, size, frequency, pattern, etc)
 - Identify the packet/variable



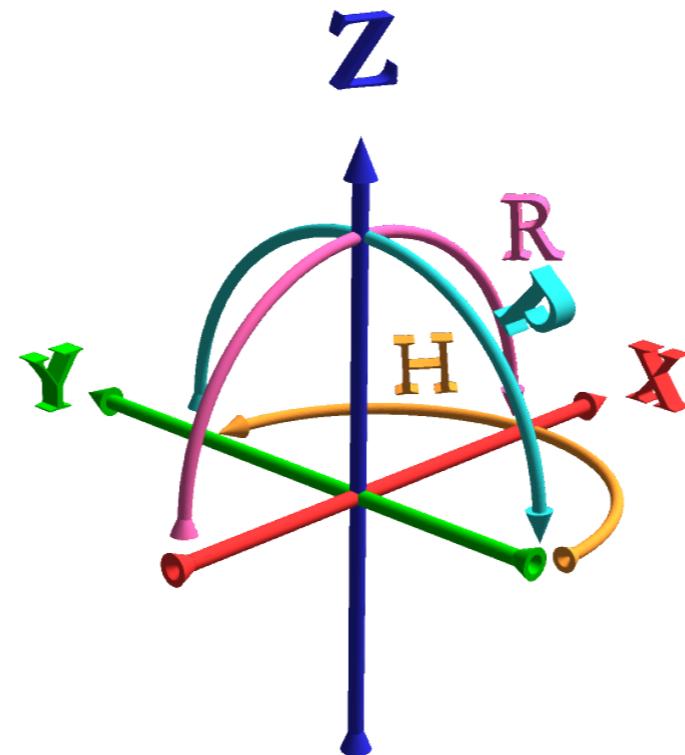
RE network protocol

*A lot of **guessing** and **assumption**, which requires **knowledge** to fasten the process.*



RE network protocol

Lab 3: Identify and dissect the “player location” packet



RE network protocol

Methodology for *binary* protocol using *unknown* format:

- Understand your target
- Identify IP(s), port(s) and content → **pwn3server:30xx
Binary (cleartext)**
- Define objective → **Player location**
 - Build use cases based on **assumptions** to isolate the packet:
 - *Send from client to game server*
 - *Coordinate format (X, Y, Z)*
 - *Send regularly*



RE network protocol

...

- Define objective → **Player location**
 - Build use cases based on assumptions to isolate the packet:
 - *Send from client to game server*
 - *Coordinate format (X, Y, Z)*
 - *Send regularly*
 - Don't move



RE network protocol

Lab 3: Identify and dissect the “player location” packet

No mouvement:

(tcp.port >= 3000 && tcp.port <= 3010) && tcp.len > 0							Expression...	+
No.	Source	Destination	Protocol	Length	Info	Data		
6	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000	6d76687150c758eb60c75f867e440000000000000000000000		
7	192.168.2.1...	192.168.2.1	TCP	68	3000 → 62634	0000		
9	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000	6d76687150c758eb60c75f867e440000000000000000000000		
10	192.168.2.1...	192.168.2.1	TCP	68	3000 → 62634	0000		
12	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000	6d76687150c758eb60c75f867e440000000000000000000000		
13	192.168.2.1...	192.168.2.1	TCP	68	3000 → 62634	0000		
15	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000	6d76687150c758eb60c75f867e440000000000000000000000		
16	192.168.2.1...	192.168.2.1	TCP	68	3000 → 62634	0000		
18	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000	6d76687150c758eb60c75f867e440000000000000000000000		
19	192.168.2.1...	192.168.2.1	TCP	68	3000 → 62634	0000		
21	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000	6d76687150c758eb60c75f867e440000000000000000000000		
22	192.168.2.1...	192.168.2.1	TCP	68	3000 → 62634	0000		
24	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000	6d76687150c758eb60c75f867e440000000000000000000000		
25	192.168.2.1...	192.168.2.1	TCP	68	3000 → 62634	0000		
27	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000	6d76687150c758eb60c75f867e440000000000000000000000		
28	192.168.2.1...	192.168.2.1	TCP	68	3000 → 62634	0000		
30	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000	6d76687150c758eb60c75f867e440000000000000000000000		
31	192.168.2.1...	192.168.2.1	TCP	68	3000 → 62634	0000		



RE network protocol

Methodology for *binary* protocol using *unknown* format:

- Understand your target
- Identify IP(s), port(s) and content → **pwn3server:30xx
Binary (cleartext)**
- Define objective → **Player location**
 - Build use cases based on assumptions → **Player location**
 - Look at network behaviour
 - Identify the packet



RE network protocol

Methodology for *binary* protocol using *unknown* format:

- Understand your target
- Identify IP(s), port(s) and content → **pwn3server:30xx
Binary (cleartext)**
- Define objective → **Finding X, Y and Z in location packet**
 - Build use cases based on **assumptions** to isolate the variable:
 - *Location on 3 axis*



RE network protocol

...

- Define objective → **Finding Z in location packet**
 - Build use cases based on assumptions to isolate the packet:
 - *Location on 3 axis*
 - Spawn
 - Don't move (not even the mouse)
 - Jump



RE network protocol

Lab 3: Identify and dissect the “player location” packet

Jump:



RE network protocol

...

- Define objective → **Finding X (or Y) in location packet**
 - Build use cases based on assumptions to isolate the packet:
 - Location on 3 axis
 - Spawn
 - Don't direction (mouse)
 - Walk straight then stop then walk backward



RE network protocol

Lab 3: Identify and dissect the “player location” packet

Move forward:

No.	Source	Destination	Protocol	Length	Info	Data
954	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76687150:758eb60c75f867e44000000000000000000
957	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76687150:758eb60c75f867e44000000000000000000
960	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76957050:758eb60c7a9897e44000000000000000000
963	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76d86650:758eb60c783b07e44000000000000000000
966	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76285150:758eb60c707077f44000000000000000000
969	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76a23250:758eb60c7cd807f44000000000000000000
972	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d765e0450:758eb60c7ae1c8044000000000000000000
975	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d7651d64f:758eb60c787788044000000000000000000
978	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d763ea94f:758eb60c76ed28044000000000000000000
981	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76627e4f:758eb60c79cd88044000000000000000000

Structure: - - X X X - - - - Z Z Z - - - - - - ? -



RE network protocol

...

- Define objective → **Finding Y (or X) in location packet**
 - Build use cases based on assumptions to isolate the packet:
 - Location on 3 axis
 - Spawn
 - Don't direction (mouse)
 - Strafe left then stop then strafe right



RE network protocol

Lab 2: Identify and dissect the “player location” packet

Strafe right:

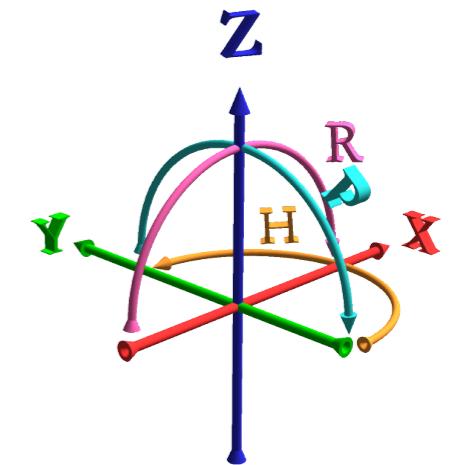
No.	Source	Destination	Protocol	Length	Info	Data
12...	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76066d4ec7d0a25fc764557a4400000000000000000000
12...	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76066d4ec7d0a25fc764557a4400000000000000000000
12...	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76066d4ec7c6a15fc7af547a440000000000000000007f
13...	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76066d4ec79f965fc71e4d7a440000000000000000007f
13...	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76066d4ec71f815fc7893e7a440000000000000000007f
13...	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76066d4ec78c625fc7cd297a440000000000000000007f
13...	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76066d4ec733395fc7c30d7a440000000000000000007f
13...	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76066d4ec7b60e5fc7f3f079440000000000000000007f
13...	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76066d4ec779e35ec7a0d379440000000000000000007f
13...	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76066d4ec71cba5ec793b779440000000000000000007f
13...	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 ...	6d76ad624ec742945ec7ebad79440000000000000000007f

Structure: - - X X X - Y Y Y - Z Z Z - - - - - ? ?



RE network protocol

Lab 2: Identify and dissect the “player location” packet



Structure: - - X X X - Y Y Y - Z Z Z - - - - - - - ? ?

Structure: - - X X X X Y Y Y Y Z Z Z Z - - - - - U S

Structure: ID ID X X X X Y Y Y Y Z Z Z Z - - - - - U S



RE network protocol

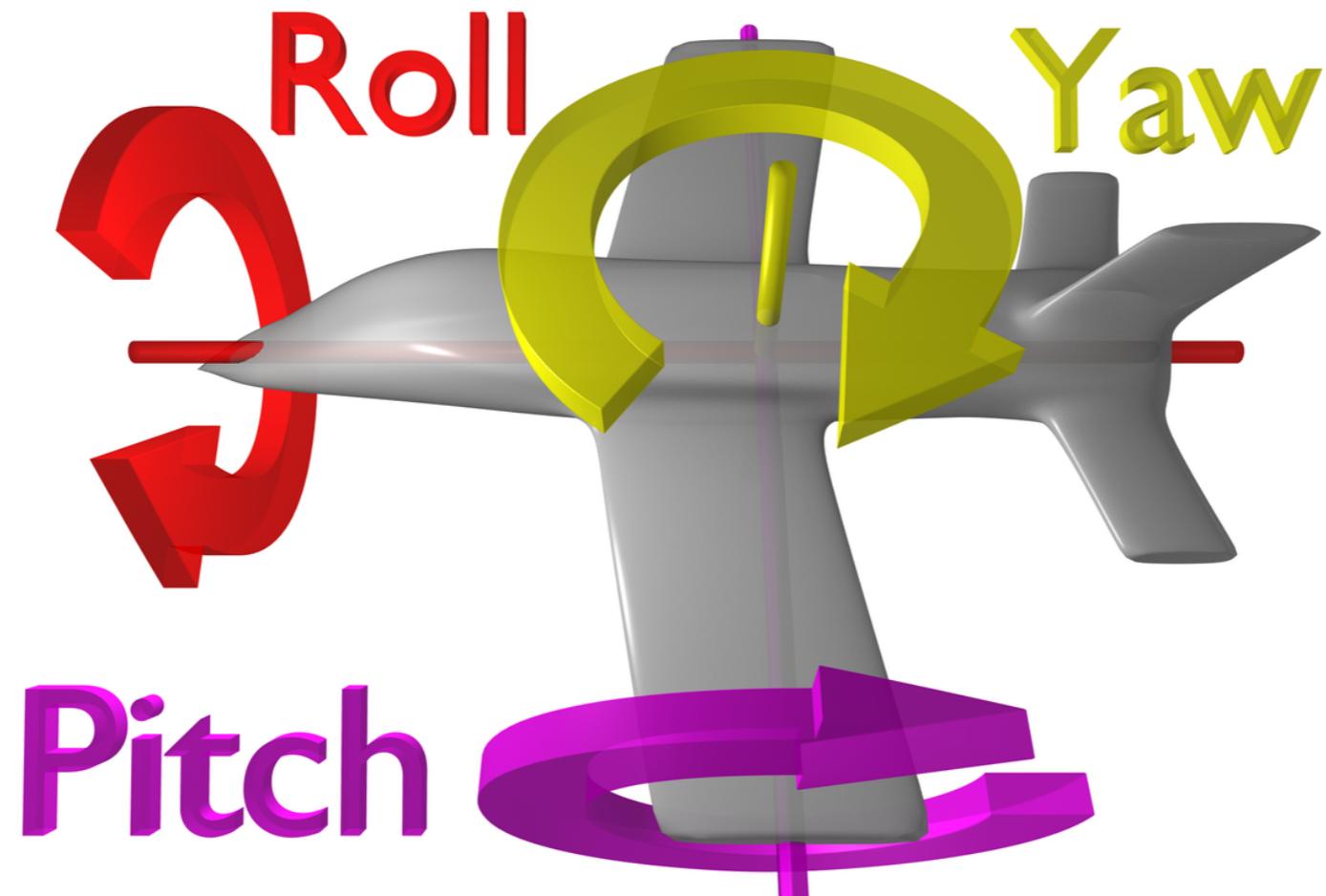
Methodology for *binary* protocol using *unknown* format:

- Understand your target
- Identify IP(s), port(s) and content → **pwn3server:30xx
Binary (cleartext)**
- Define objective → **Finding direction values**
 - Build use cases based on **assumptions** to isolate the variable:
 - *Based on yaw, pitch roll*
 - *At spawn, the direction is reset (0 yaw, 0 pitch and 0 roll)*



RE network protocol

Aircraft principal axes used in video games



RE network protocol

...

- Define objective → **Finding yaw**
 - Build use cases based on assumptions to isolate the packet:
 - *Based on yaw, pitch roll*
 - *At spawn, the direction is reset (0 yaw, 0 pitch and 0 roll)*
 - Spawn
 - Don't move the mouse
 - Use arrow key to look right, then stop, then look left



RE network protocol

Lab 2: Identify and dissect the “player location” packet

Look left:

Structure: ID ID X X X X Y Y Y Y Z Z Z Z - - YA YA - - U S



RE network protocol

...

- Define objective → **Finding pitch**
 - Build use cases based on assumptions to isolate the packet:
 - *Based on yaw, pitch roll*
 - *At spawn, the direction is reset (0 yaw, 0 pitch and 0 roll)*
 - Spawn
 - Use your mouse to look straight up, then stop, then look down



RE network protocol

Lab 2: Identify and dissect the “player location” packet

Look up:

No.	Source	Destination	Protocol	Length	Info	Data
717	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d4-0000df2600000000
720	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d4-0000df2600000000
723	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d4-1600df2600000000
726	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d4-b000df2600000000
729	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d4-e600df2600000000
732	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d4-e600df2600000000
735	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d4-2901df2600000000
743	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d4-5501df2600000000
746	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d4-5501df2600000000
749	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d4-a601df2600000000
752	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d4-a602df2600000000

Structure: ID ID X X X Y Y Y Z Z Z P P YA YA - - U S



RE network protocol

...

- Define objective → **Finding roll**
 - Build use cases based on assumptions to isolate the packet:
 - *Based on yaw, pitch roll*
 - *At spawn, the direction is reset (0 yaw, 0 pitch and 0 roll)*
 - Spawn
 - Use your mouse to look straight up
 - Use your mouse to look on the left



RE network protocol

Lab 2: Identify and dissect the “player location” packet

Look up/down:

No.	Source	Destination	Protocol	Length	Info	Data
965	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d44c936912600000000
968	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d44d538ed2500000000
971	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d449c39af25ffff0000
974	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d44983a8f25ffff0000
977	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d44c93b8f2500000000
980	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d44113d8f2500000000
983	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d44573d8f2500000000
986	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d44d33d8f25ffff0000
989	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d446b3e8f2500000000
992	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 ...	6d76cf5050c7eca75ec7a17d8d44ca3e8f25ffff0000

Structure: ID ID X X X Y Y Y Z Z Z P P YA YA R R U S



RE network protocol

Lab 3: Identify and dissect the “player location” packet

Structure: ID ID X X X X Y Y Y Y Z Z Z Z - - - - - - U S

Structure: ID ID X X X X Y Y Y Y Z Z Z Z P P YA YA R R U S



RE network protocol

Lab 4: Identify and dissect the “*mana*” and “*chat*” packets

Ressource:

- Wireshark filer: (tcp.port >= 3000 and tcp.port <=3010) and tcp.len > 0



RE network protocol

Packet **format**:

- Start with a 2-bytes identifier (e.g. mv = location, * i = fire, etc)
- Can be concatenated
- Strings is defined by length
- Number are represented in little-endian



RE network protocol

Complete list of reversed packets

[https://github.com/beaujeant/PwnAdventure3/
blob/master/Network/pwn3-gs.md](https://github.com/beaujeant/PwnAdventure3/blob/master/Network/pwn3-gs.md)

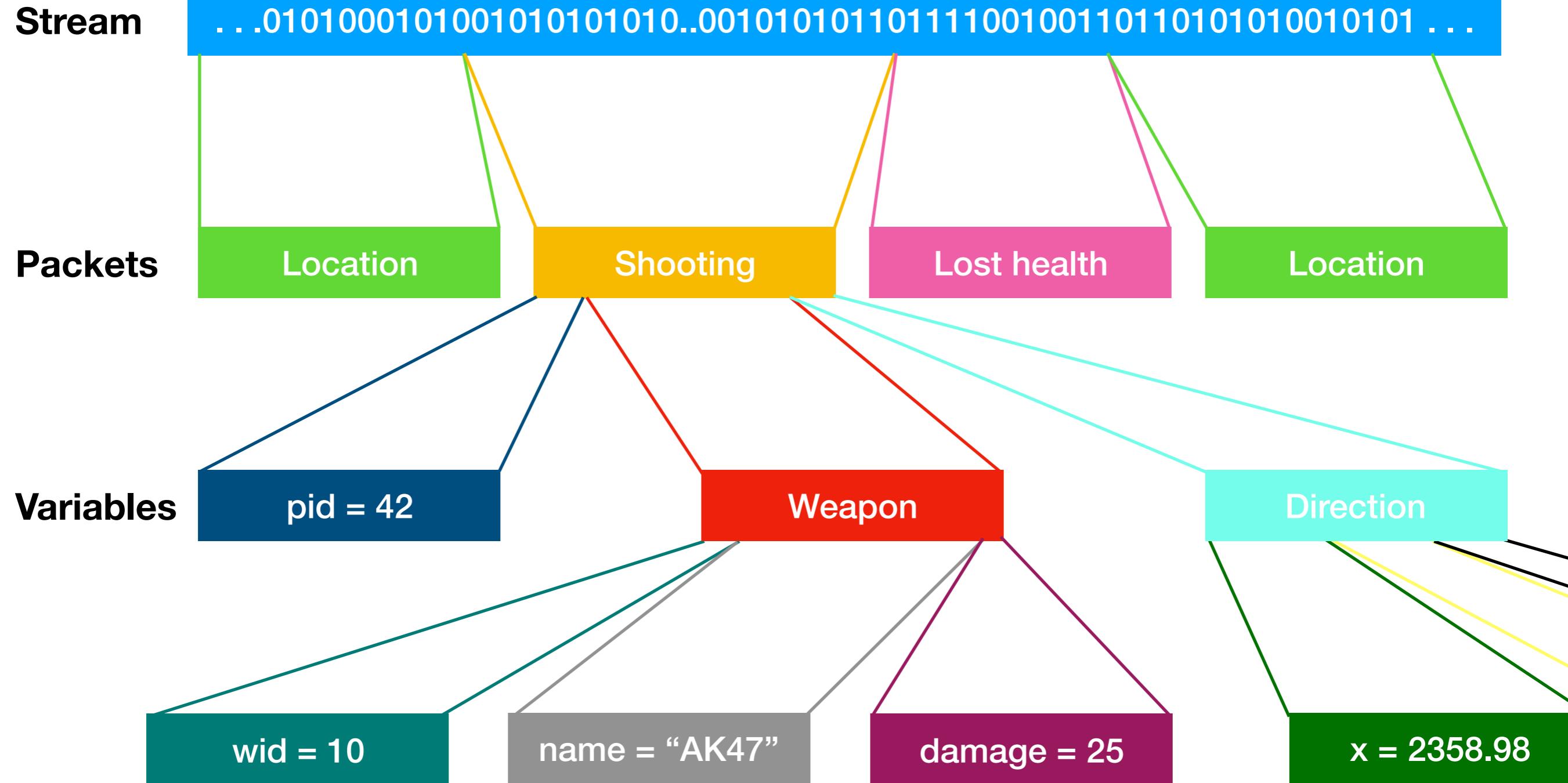


RE network protocol

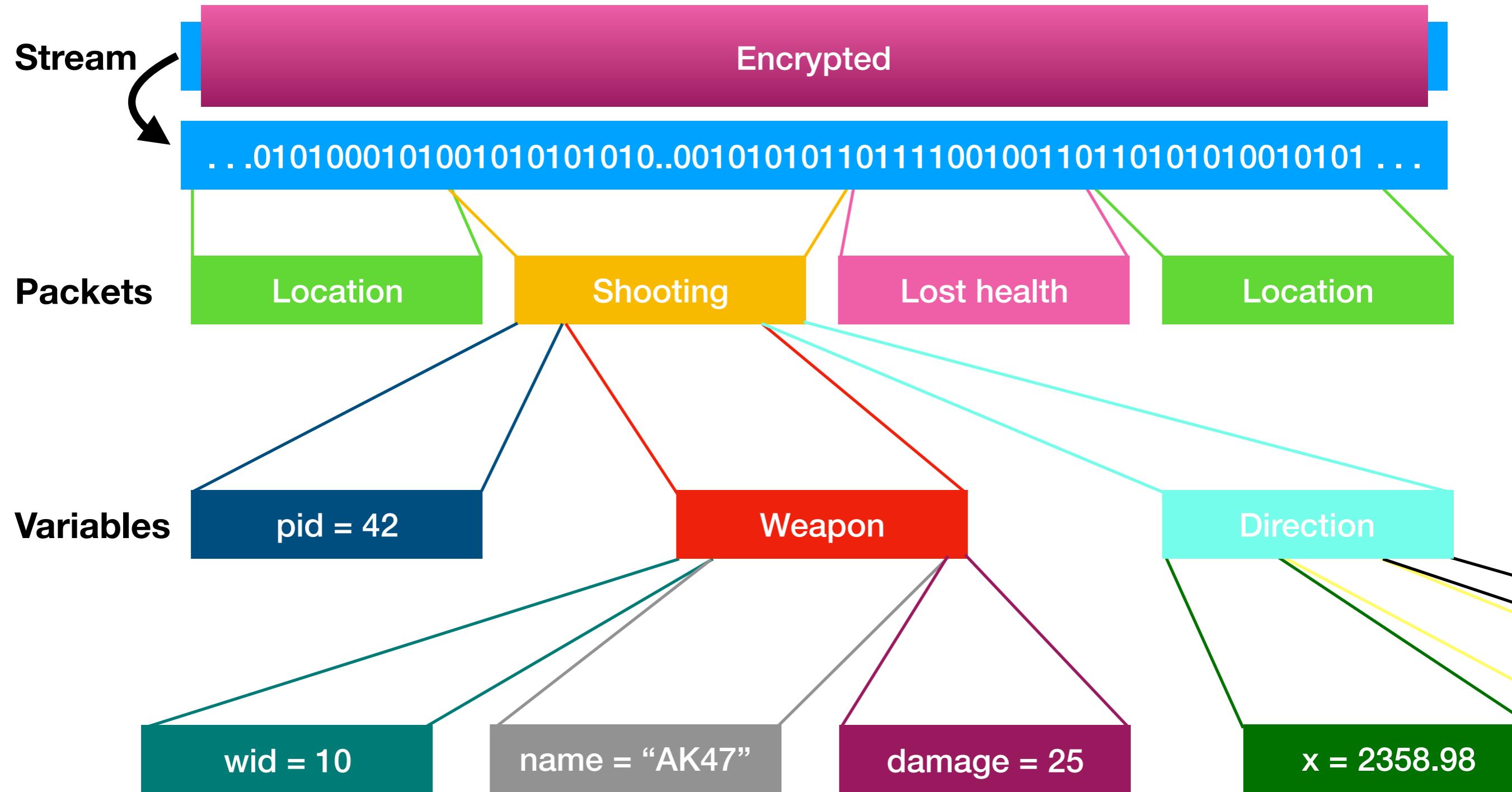
What about an **encrypted** protocol?
Let's have a look at the **Master server**.



RE network protocol



RE network protocol

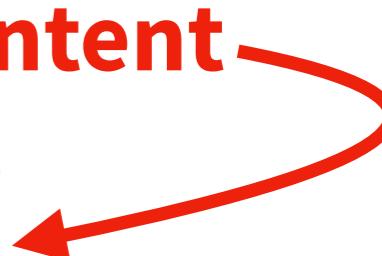


RE network protocol

Methodology for *binary* protocol using *unknown* format:

- Understand your target
- Identify IP(s), port(s) and content (encrypted? Format?)
- Define objective (e.g. location packet, new item packet, etc)
 - Build use cases (e.g. don't move, shoot, etc) based on assumptions to isolate the packet/variable
 - Look at network behaviour (changes, size, frequency, pattern, etc)
 - Identify the packet/variable

Decrypt content



RE network protocol

Decrypt content:

- Identify encryption algorithm
- Find a way to read cleartext
 - Debug/patch binary to leak content once decrypted
 - Reverse decryption (network stream or function), extract the key and build decryption
 - Reverse decryption, find a vulnerability and crack decryption



RE network protocol

Identify encryption **algorithm**:

- Find function that uses the encrypted protocol, e.g. *Login*
 - GameAPI::Login
 - MasterServerConnection::Login
 - SSLSocket



RE network protocol

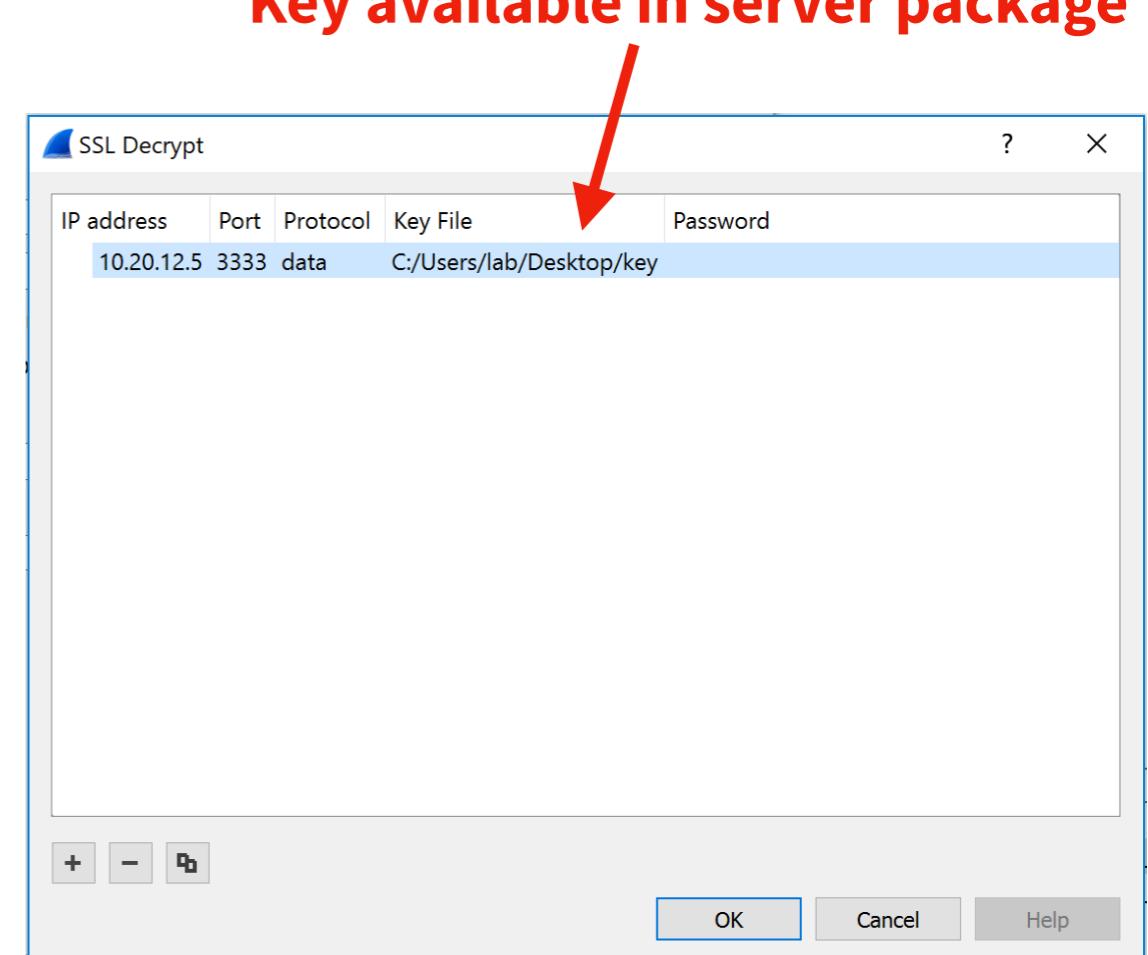
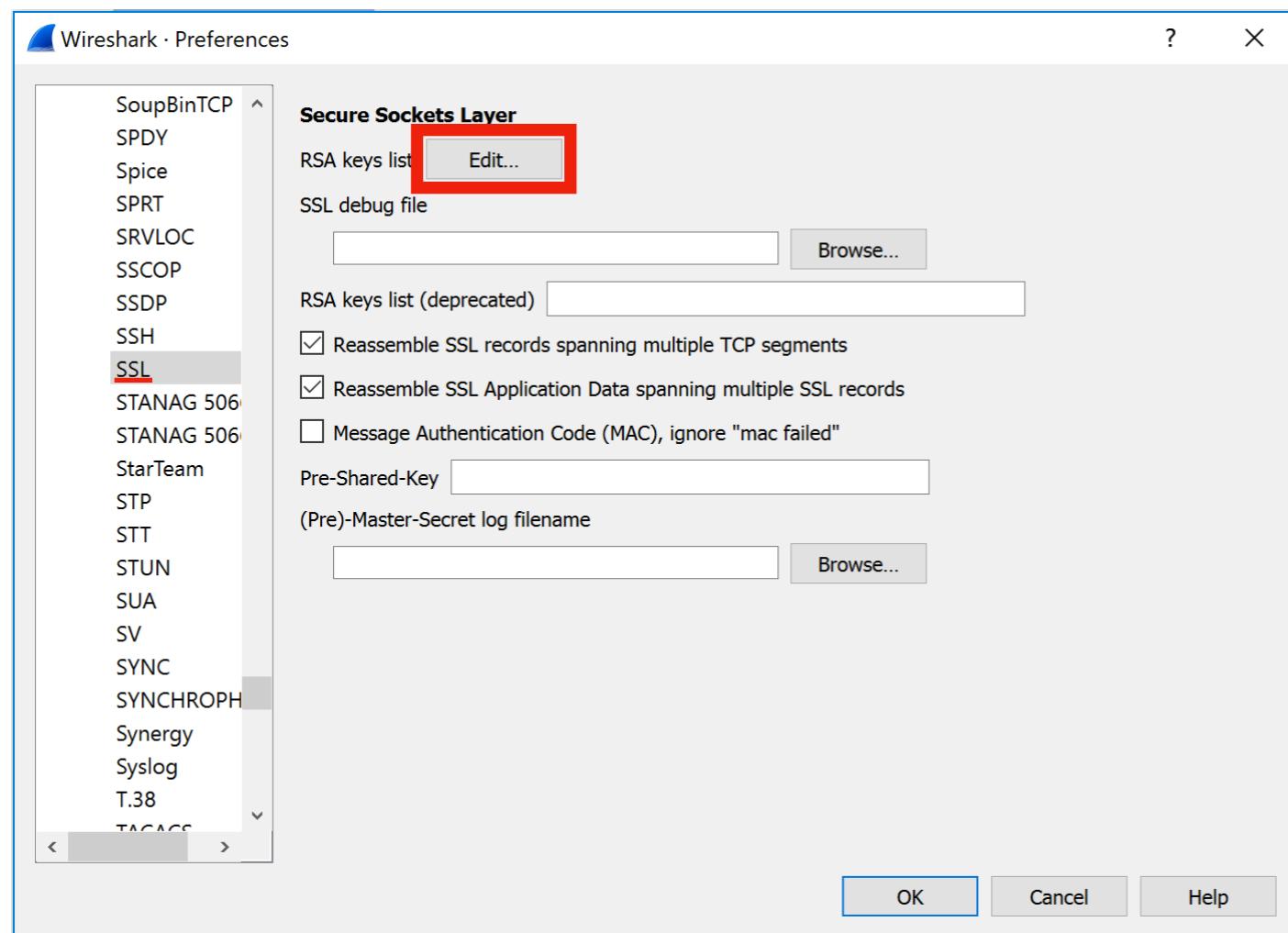
SSL is a known **cryptographic protocol**, which means no need to build an **encryption/decryption script**

- Python library
- Wireshark decryption



RE network protocol

Wireshark decryption:



Building a Wireshark dissector



Building dissector

What is Wireshark?

- Packet analyser
- Data → human readable representation
- Hundreds of supported protocols and media
- Possibility to add custom parser (dissector)



Building dissector

Two type of **dissectors**

Compiled dissector:

- Edit source code & compile
- Long term
- Parsing process faster

External **plugin**:

- Plugin in *Lua*
- Easy language
- Rapidly deployed



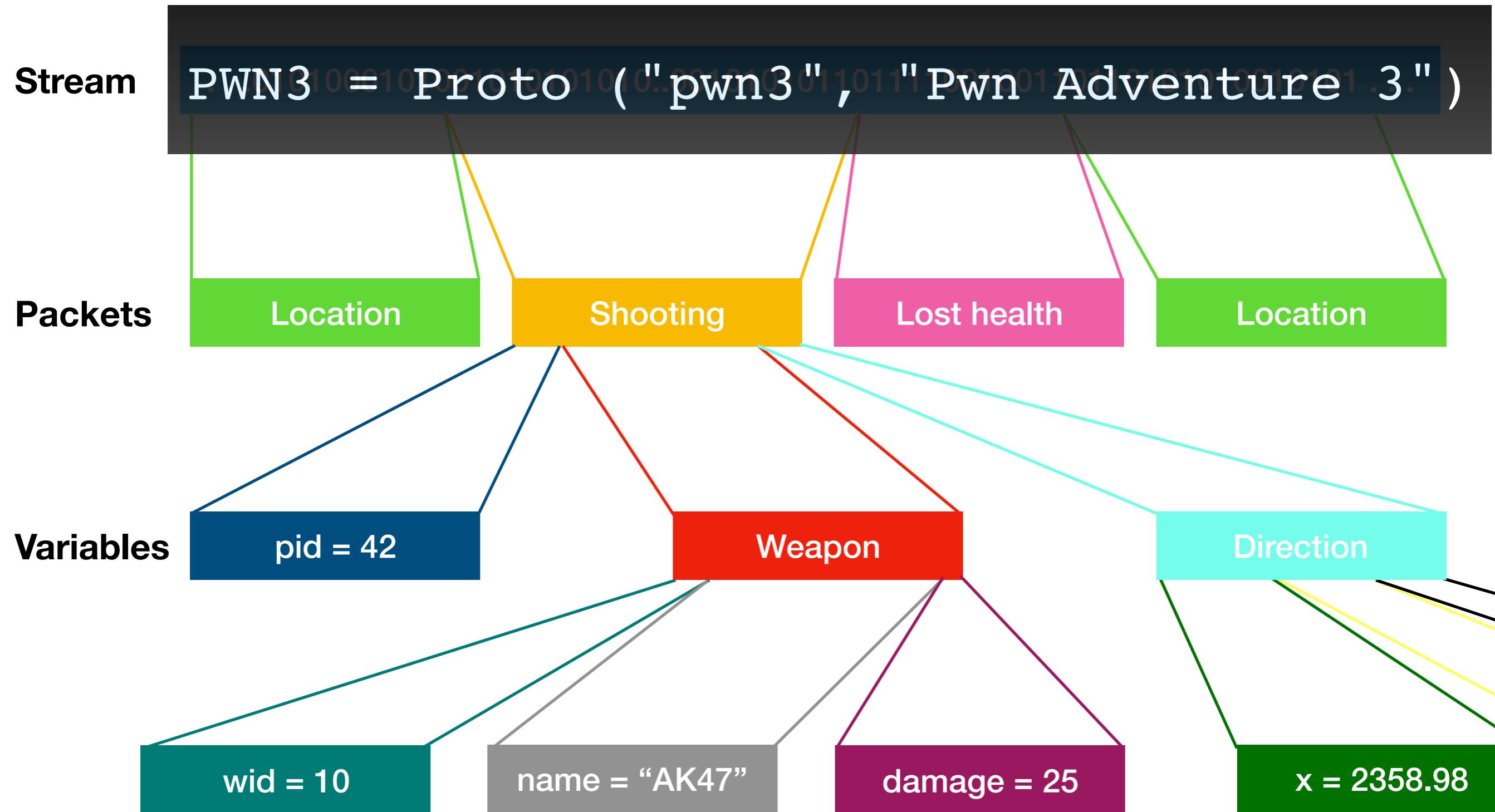
Building dissector

Create **new protocol** with Proto()

```
PWN3 = Proto ("pwn3", "Pwn Adventure 3")
```



RE network protocol



Building dissector

Define all **elements** (and type) of the protocol:

```
local f = PWN3.fields

local opcodes = {
    [0x2a69] = "Fire",
    [0x6d61] = "Update mana",
    [0x6d76] = "Update location",
    ...
}

f.opcode = ProtoField.uint16 ("pwn3.opcode", "Action", base.HEX, opcodes)

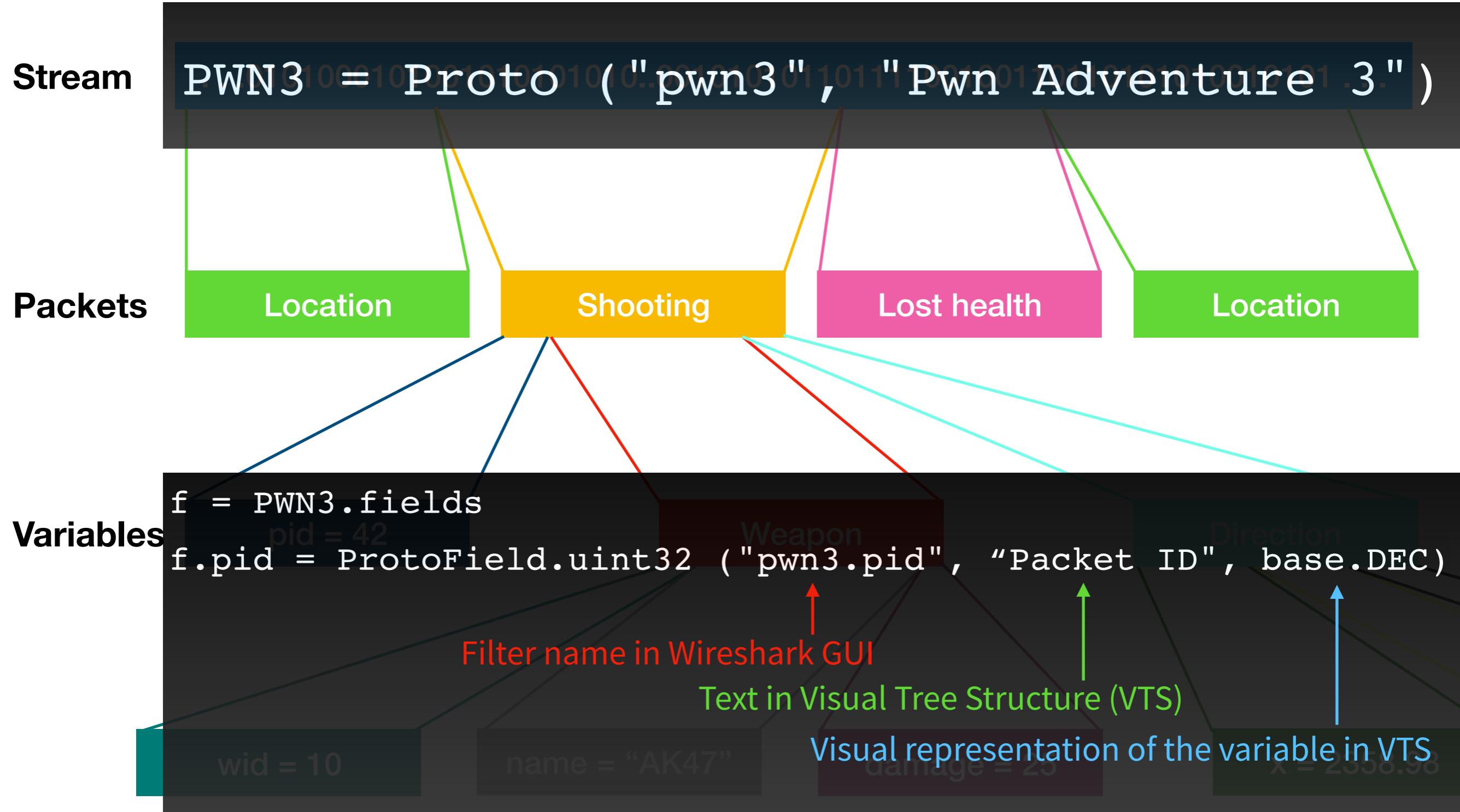
f.posx = ProtoField.new ("X coordinate", "pwn3.posx", ftypes.FLOAT)
f.posy = ProtoField.new ("Y coordinate", "pwn3.posy", ftypes.FLOAT)
f.posz = ProtoField.new ("Z coordinate", "pwn3.posz", ftypes.FLOAT)

f.mana = ProtoField.uint32 ("pwn3.mana", "Mana", base.DEC)

...
```

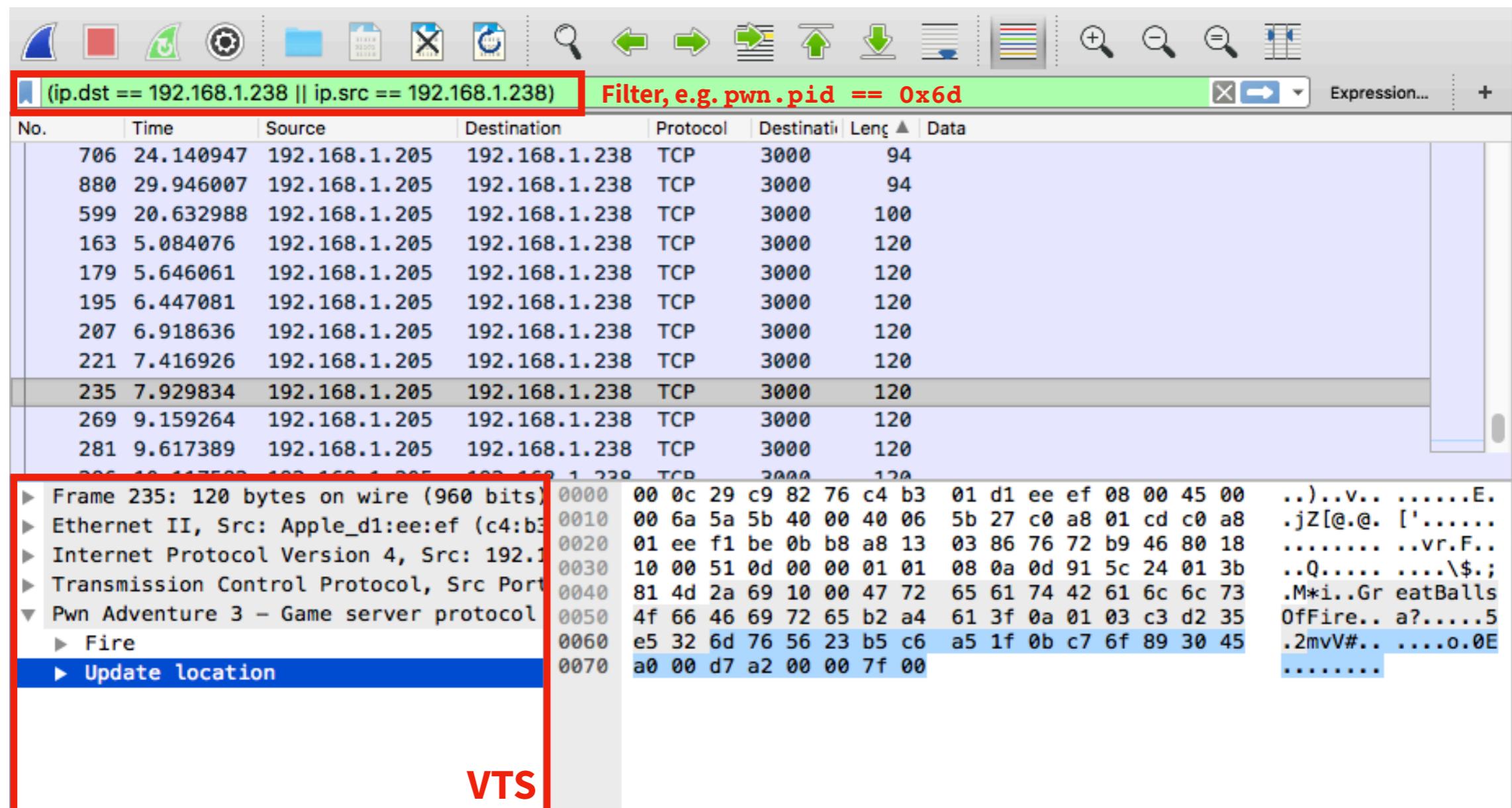


RE network protocol



Building dissector

Wireshark visual tree structure



Building dissector

Start **dissecting** with PROTO.dissector():

```
function PWN3.dissector (buffer, pinfo, tree)

    -- Dissection code
    -- buffer is the data in the TCP packet
    -- pinfo is the packet information (time, length, IP, etc)
    -- tree is the root element in the visual tree structure

end
```



Building dissector

Adding **new branch** in the *visual tree structure* with `tree:add`:

```
-- Add node to the root (right below the TCP element)
local subtree = tree:add (PROTO, buffer())

-- Add a node to subtree (created above)
local branch = subtree:add (buffer(OFFSET, LENGTH), "Text")

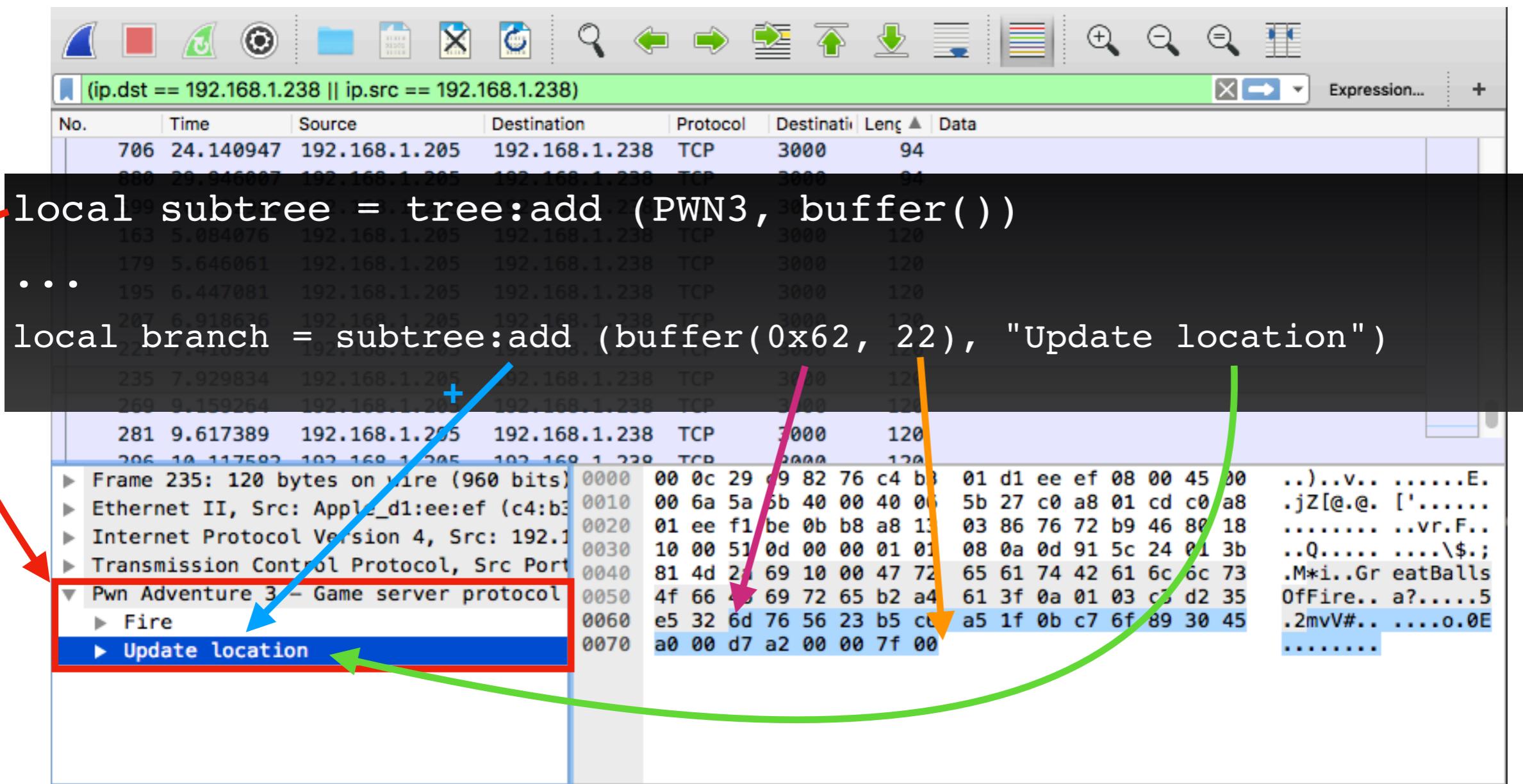
-- Add an element/variable to branch (created above)
branch:add (f.FMT, buffer(OFFSET, LENGTH))
```

- PROTO = PWN3
- BUFFER = buffer argument given in PROTO.dissector()
- f.FMT = format defined previously
- OFFSET / LENGTH = used to select a part of the buffer



Building dissector

Wireshark visual tree structure



Building dissector

We create a **loop** that **read the buffer**:

```
function PWN3.dissector (buffer, pinfo, tree)

    -- Create the "Pwn Adventure 3" node
    local subtree = tree:add (PWN3, buffer())

    -- Pointer to read through the buffer
    local offset = 0

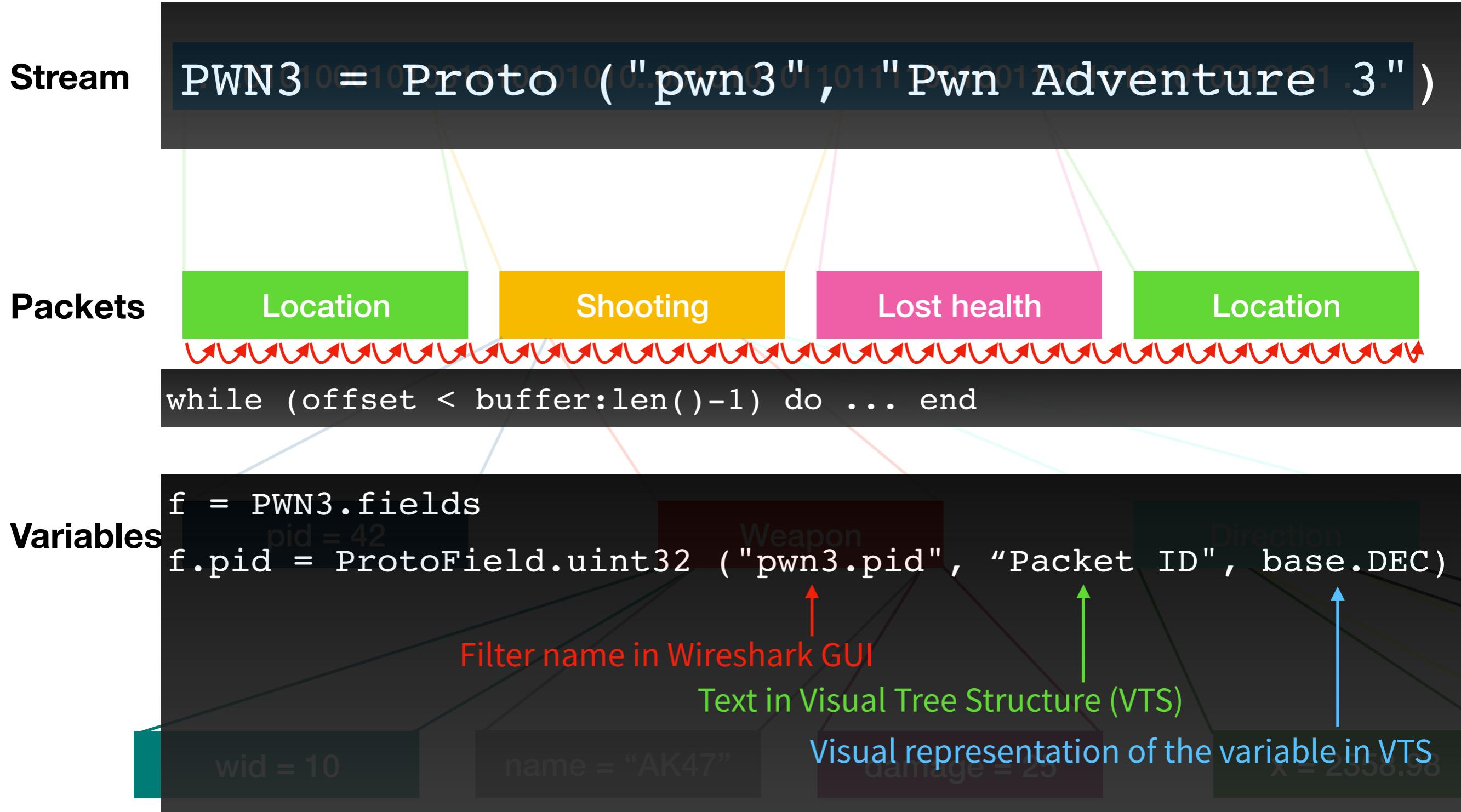
    -- Read until we reach the end of the buffer
    while (offset < buffer:len()-1) do

        ...

    end
end
```



RE network protocol



Building dissector

Each time we find a **known identifier**, the section is **parsed**:

```
while (offset < buffer:len()-1) do
    -- Reading two bytes
    -- opcode is the packet identifier number
    local opcode = buffer(offset, 2):uint()
    offset = offset + 2

    -- Update mana
    if (opcode == 0x6d61) then
        ...
        -- Fire
    elseif (opcode == 0x2a69) then
        ...
    end
end
```



Building dissector

If mana:

```
-- Adding a node "Update mana" in subtree
-- Using the name related to the opcode
local branch = subtree:add (buffer(offset-2, 6), opcodes[opcode])

-- Appending a text to node "Update mana"
branch:append_text ("", Mana: " .. buffer(offset, 4):le_uint())

-- Adding a node in the "Update mana" branch
-- Adding the opcode
branch:add (f.opcode, buffer(offset-2, 2))

-- Adding a node in the "Update mana" branch
-- Node mana level
branch:add_le (f.mana, buffer(offset, 4))
offset = offset + 4
```

```
▼ Pwn Adventure 3 - Game server protocol
  ▼ Update mana, Mana: 37
    Action: Update mana (0x6d61)
    Mana: 37
    Unknown: 0x00
```



Building dissector

If fire:

```
-- Getting the size of the weapon name
-- This is to calculate the total length of the packet
local length = buffer(offset, 2):le_uint()

-- Adding a node "Fire" in subtree
-- Using the name related to the opcode
-- The size of the packet is:
-- 2 bytes for the opcode
-- 2 bytes for the length of the weapons name
-- n bytes for the weapons name (stored in var length)
-- 12 bytes for the direction of the projectile
-- Total = 16 + length
local branch = subtree:add (buffer(offset-2, 16+length), opcodes[opcode])

-- Adding a node in the "Update mane" branch
-- Adding the opcode
branch:add (f.opcode, buffer(offset-2, 2))

-- Skip the length byte
offset = offset + 2

-- Appending a text to node "Fire"
branch:append_text (", Weapon: " .. buffer(offset, length):string())

-- Adding a node in the "Fire" branch
-- Node weapons name
branch:add (f.str, buffer(offset, length))
offset = offset + length

-- Adding a node in the "Fire" branch
-- Node direction of the projectile
addVectors (buffer, offset, branch)
offset = offset + 12
```

```
▼ Pwn Adventure 3 - Game server protocol
  ▼ Fire, Weapon: GreatBallsOfFire
    Action: Fire (0x2a69)
    String: GreatBallsOfFire
  ▼ Vectors
    Vector X: 3233823168
    Vector Y: 3239396352
    Vector Z: 932628512
```



Building dissector

We create **function** to add **direction/location/vectors** in tree:

```
function addVectors (vectors, offset, tree)

local branch

branch = tree:add (vectors(offset, 12), "vectors")

branch:add_le (f.vx, vectors(offset, 4))
branch:add_le (f.vy, vectors(offset + 4, 4))
branch:add_le (f.vz, vectors(offset + 8, 4))

end
```



Building dissector

Finally, we **register** the protocol to the associated port(s) with DissectorTable:

```
tcp_table = DissectorTable.get ("tcp.port")
tcp_table:add (3000, PWN3)
tcp_table:add (3001, PWN3)
...
...
```



Building dissector

Limitation with this design:

- If identifier not known, look for the next 2-bytes until one pair match
- Part of the data might match a known ID
- Start parsing from a wrong offset



Building dissector

Install dissector:

- <https://github.com/beaujeant/PwnAdventure3/blob/master/Network/draft-dissector.lua>
- Find plugin folder (in “About Wireshark”)
 - macOS: “Wireshark” > “About Wireshark” > “Folders”
 - Linux: “Help” > “About Wireshark” > “Folders”
 - Windows: “Help” > “About Wireshark” > “Folders”
- Save dissector in folder
- Menu “Analyse” > “Reload Lua Plugins”
- Verify in menu “Analyse” > “Enabled Protocol...”



Building dissector

Lab 5: Build the rest of the dissector

Ressources:

- Wireshark filer: (tcp.port >= 3000 and tcp.port <=3010) and tcp.len > 0
- <https://github.com/beaucheant/PwnAdventure3/blob/master/Network/pwn3-gs.md>
- <https://github.com/beaucheant/PwnAdventure3/blob/master/Network/draft-dissector.lua>
- Some OS need wireshark-devel



Building dissector

Complete dissector:

[https://github.com/beaujeant/PwnAdventure3/
blob/master/Network/pwn3-gs.lua](https://github.com/beaujeant/PwnAdventure3/blob/master/Network/pwn3-gs.lua)

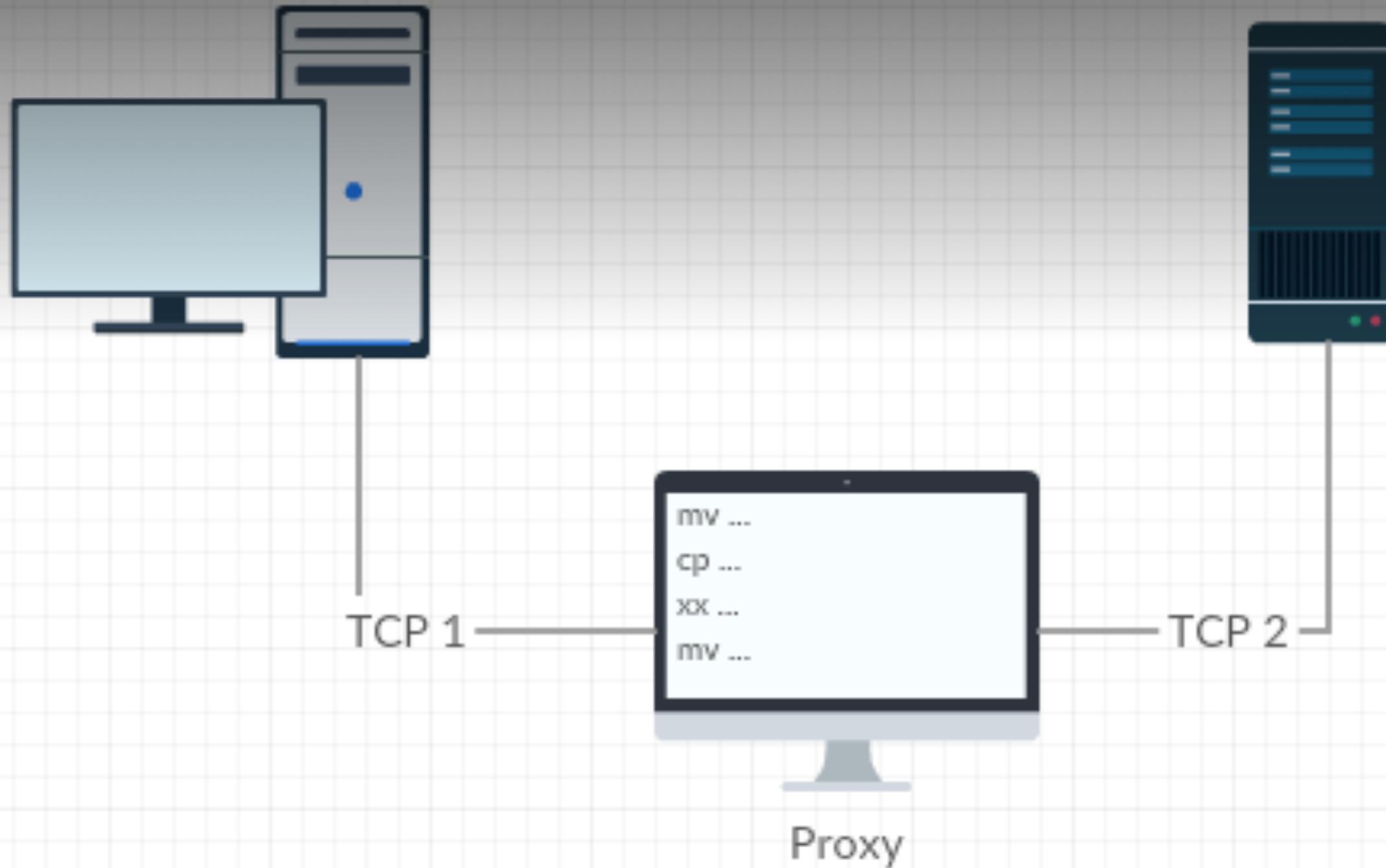


Building dissector

Demo: Play game with dissector



Building an asynchronous proxy



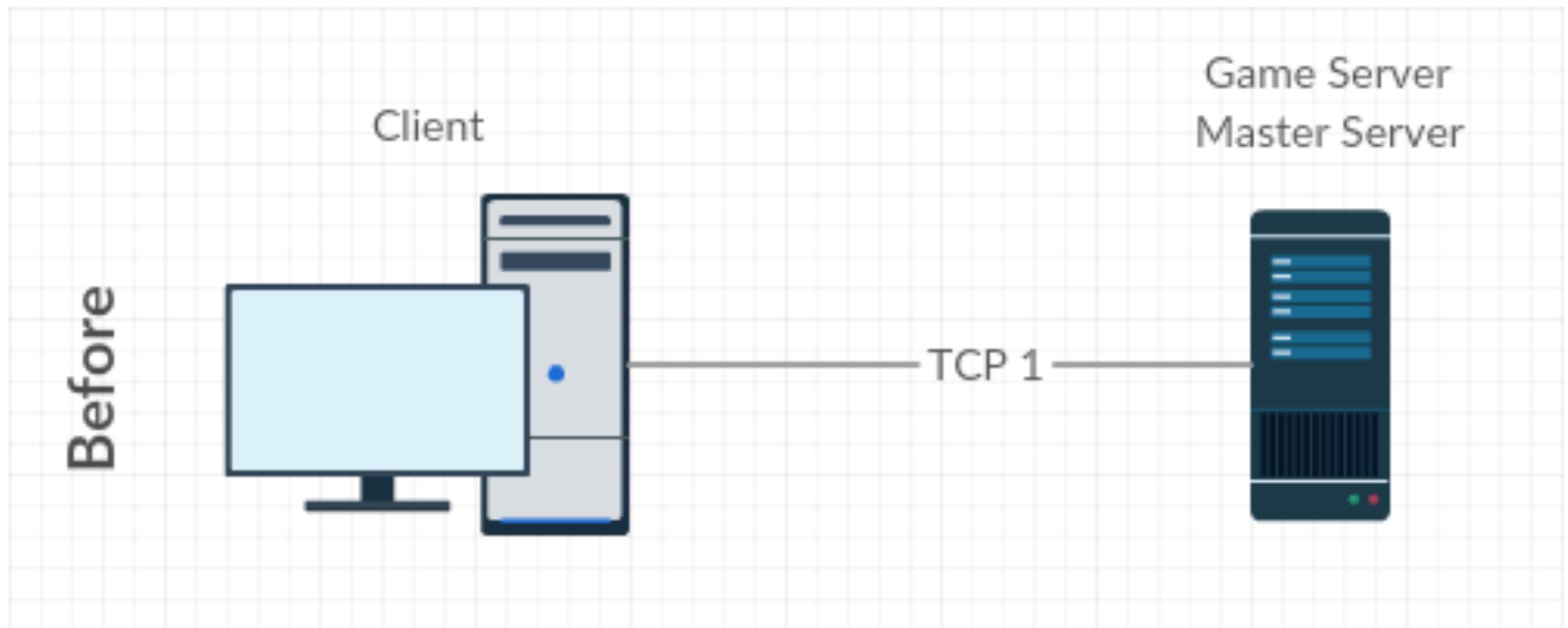
Building async proxy

Proxy **script**:

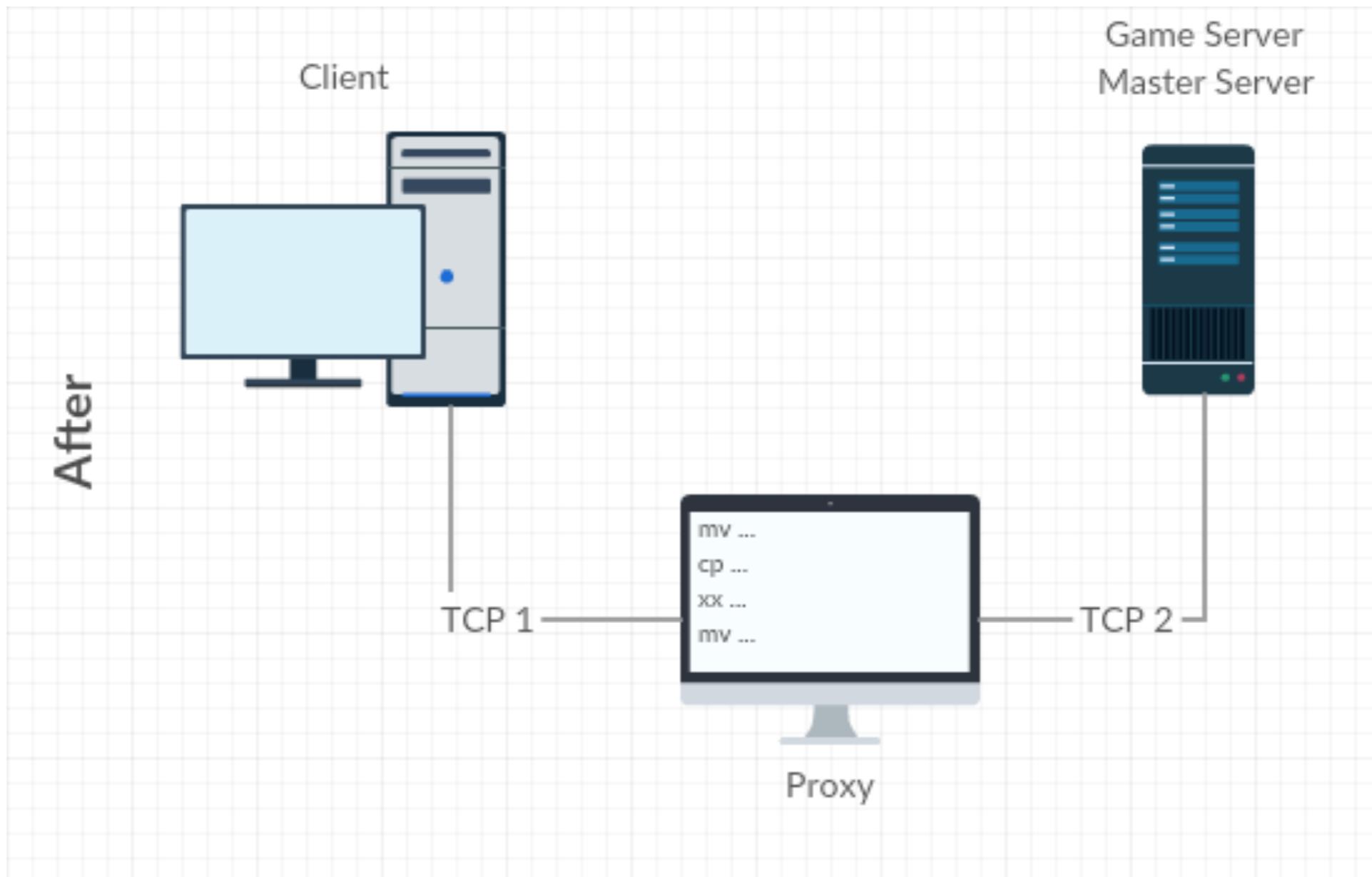
- Python 2.7
- Native libs: *asyncore*, *socket* and *struct*
- Parse / edit on the fly *in* and *out* communications



Building async proxy



Building async proxy



Building async proxy

Re-routing the traffic:

- Editing `/etc/hosts` file:

127.0.0.1 pwn3master



Building async proxy

Client to proxy connection (Master game server):

- Listener on port 3333

```
import asyncore
import socket

class ProxServer(asyncore.dispatcher):

    def __init__(self, src_port):
        self.src_port = src_port
        asyncore.dispatcher.__init__(self)
        self.create_socket(socket.AF_INET, socket.SOCK_STREAM)
        self.set_reuse_addr()
        self.bind(('0.0.0.0', src_port))
        self.listen(5)

if __name__ == '__main__':
    master = ProxServer(3333)
    asyncore.loop()
```



Building async proxy

Proxy to **server** connection (Master game server):

- Once client connected, create a new connection with server

```
class ProxServer(asyncore.dispatcher):  
  
    def __init__(self, src_port, dst_host, dst_port):  
        self.src_port = src_port  
        self.dst_host = dst_host  
        self.dst_port = dst_port  
        asyncore.dispatcher.__init__(self)  
        self.create_socket(socket.AF_INET, socket.SOCK_STREAM)  
        self.set_reuse_addr()  
        self.bind(('0.0.0.0', src_port))  
        self.listen(5)  
  
    def handle_accept(self):  
        pair = self.accept()  
        if not pair:  
            return  
        left, addr = pair  
        try:  
            right = socket.create_connection((self.dst_host, self.dst_port))  
        except socket.error, e:  
            if e.errno is not errno.ECONNREFUSED: raise  
            left.close()  
  
    def close(self):  
        asyncore.dispatcher.close(self)  
  
if __name__ == '__main__':  
  
    # 10.0.1.3 = Master Server  
    master = ProxServer(3333, '10.0.1.3', 3333)  
    asyncore.loop()
```



Building async proxy

Forward connections:

- Forward *input from client* ➔ *output to server*
- Forward *input from server* ➔ *output to client*

```
def handle_accept(self):  
    pair = self.accept()  
    if not pair:  
        return  
    left, addr = pair  
    try:  
        right = socket.create_connection((self.dst_host, self.dst_port))  
    except socket.error, e:  
        if e.errno is not errno.ECONNREFUSED: raise  
        left.close()  
  
    client = Sock(left)  
    server = Sock(right)  
  
    client.other = server  
    server.other = client
```



Building async proxy

New class **Sock**:

```
class Sock(asyncore.dispatcher):
    write_buffer = ''

    def readable(self):
        return not self.other.write_buffer

    def handle_read(self):
        self.other.write_buffer += self.recv(4096*4)

    def handle_write(self):
        if self.write_buffer:
            pkt = parse(self.write_buffer)
            sent = self.send(pkt)
            self.write_buffer = self.write_buffer[sent:]

    def handle_close(self):
        self.close()
        if self.other.other:
            self.other.close()
        self.other = None
```

Buffer that receives data to be forwarded

handle_read is called whenever the socket is ready to read

handle_write is called whenever the socket is ready to write



Building async proxy

- Proxy for **Master server** 

- Proxy for **Game server**:

```
if __name__ == '__main__':  
  
    master = ProxServer(3333, '10.0.1.3', 3333)  
    game = ProxServer(3000, '10.0.1.3', 3000)  
  
    print "Proxy ready..."  
    asyncore.loop()
```



Building async proxy

Now we just need to create the `parse()` function to **manipulate the data**



Building async proxy

What to do?

- Change spawning location
- Pick up any item
- Generate any item (weapons, ammunitions, etc)



Set spawning location

Spawn location packet:

```
[II  II]  [??  ??]  [XX  XX  XX  XX]  [YY  YY  YY  YY]  [ZZ  ZZ  ZZ  ZZ]  
[RR  RR]  [YY  YY]  [PP  PP]
```

I = Identifier can be multiple values

? = Always seen set as 0x00 0x00

X, Y and Z = Location where to spawn

R, Y and P = Set to zero (spawn always looking at the same direction)

Packet has always been seen *alone*.



Set spawning location

Identify generic spawn packet:

- 22 bytes long (always *alone*)
- 3rd and 4th = 0x00 (?? ??)
- Last 6 bytes = 0x00 (RR YY PP)

```
def parse(p_out):  
    if ( len(p_out) == 22 and p_out[2:4] == "\x00\x00" and  
        p_out[16:] == "\x00\x00\x00\x00\x00\x00" ):  
  
        # DO SOMETHING  
  
    return p_out
```



Set spawning location

What to do once we have **identifying** the *spawn packet*?

- Find the coordinate where we want to spawn
 - E.g. -39602.8, -18288.0, 2400.28 + 10000 (In town, high in the sky)
- Replace the coordinates with the new location
- Re-use the identifier
- Add \x00\x00
- Add the new coordinate [xx xx xx xx] [yy yy yy yy] [zz zz zz zz]
- Add \x00\x00\x00\x00\x00\x00\x00\x00 (direction)



Set spawning location

Parsing script to replace the spawn packet:

```
def parse(p_out):
    if ( len(p_out) == 22 and p_out[2:4] == "\x00\x00" and
        p_out[16:] == "\x00\x00\x00\x00\x00\x00" ):

        packet_id = p_out[:2] # We re-use the packet ID
        x = -39602.8
        y = -18288.0
        z = 2400.28 + 10000

        p_out = packet_id
        p_out += struct.pack("=HfffHHH", 0, x, y, z, 0, 0, 0)

    return p_out
```



Set spawning location

Demo: Spawn in the sky



Picking up any item

Pick up element packet: [65 65] [NN NN NN NN]

65 65 = Identifier (e e)

N = Element identifier

```
def getme(elem_id):  
    return "ee" + struct.pack("I", elem_id)
```



Picking up any item

What element number?

- Great balls of fire = 00 00 00 01
- Packet to sent: [65 65] [01 00 00 00] (little endian)

Transmission Control Protocol, Src Port: 3000, Dst		0040 f7 8d 6d 6b 01 00 00 00 00 00 00 00 00 00 00 00 10 00 47 ..mk....G	reatBall sOffFire.
Pwn Adventure 3 - Game server protocol		0050 72 65 61 74 42 61 6c 6c 73 4f 66 46 69 72 65 00 .*	Z.. ...C.....
New element, ID: 1, Element: GreatBallsOfFire		0060 87 2a c7 00 0c 5a c7 00 00 a1 43 00 00 00 80 00 .d...mk.	d...mk.
Action: New element (0x6d6b)		0070 00 64 00 00 00 6d 6b 02 00 00 00 00 00 00 00 00 00 00 ..LostCa veBush.#	..LostCa veBush.#
Element ID: 1		0080 0c 00 4c 6f 73 74 43 61 76 65 42 75 73 68 00 23 Q...,... .C.....	Q...,... .C.....
Unknown: 0x00		0090 51 c7 00 d6 2c c7 00 00 b3 43 00 00 00 00 00 00 00 00 d...mk..	d...mk..
Unknown: 0x00		00a0 64 00 00 00 6d 6b 03 00 00 00 00 00 00 00 00 00 09 .BearChe st.....	.BearChe st.....
Unknown: 0x00		00b0 00 42 65 61 72 43 68 65 73 74 00 b0 f6 c5 00 e2 {G.p&E#.d...	{G.p&E#.d...
Unknown: 0x00		00c0 7b 47 00 70 26 45 23 fd e6 7f 81 00 64 00 00 00 .mk.....Cow	.mk.....Cow
String: GreatBallsOfFire		00d0 6d 6b 04 00 00 00 00 00 00 00 00 08 00 43 6f 77 Chest..v H...o..@.	Chest..v H...o..@.
Location		00e0 43 68 65 73 74 00 fe 76 48 00 a1 6f c8 00 40 92 D".....d ...mk...	D".....d ...mk...
X coordinate: -43655		00f0 44 22 fe 08 8f c5 fd 64 00 00 00 6d 6b 05 00 00 .LavaChes	.LavaChes
Y coordinate: -55820		0100 00 00 00 00 00 00 09 00 4c 61 76 61 43 68 65 73 t..FG... ...`D...	t..FG... ...`D...
Z coordinate: 322		0110 74 00 bc 46 47 00 d8 a3 c5 00 60 be 44 00 00 e3 8..d...m k.....	8..d...m k.....
Direction		0120 38 00 00 64 00 00 00 6d 6b 06 00 00 00 00 00 00Bloc kyChest.Bloc kyChest.
Direction roll: 0		0130 00 00 0b 00 42 6c 6f 63 6b 79 43 68 65 73 74 00 .>....F. 0.E....	.>....F. 0.E....
		0140 f0 3e c5 00 ba b3 46 00 30 0e 45 00 00 00 c0 00 .d...mk.d...mk.
		0150 00 64 00 00 00 6d 6b 07 00 00 00 00 00 00 00 00 00 00 ..GunSho p0wner.W	..GunSho p0wner.W
		0160 0c 00 47 75 6e 53 68 6f 70 4f 77 6e 65 72 00 57E.....E.....
		0170 12 c7 00 04 8d c6 00 00 17 45 00 00 ff 7f 00 00	



Picking up any item

When to send?

- Decide when to trigger
- Easy to trigger



Picking up any item

Chat packet: [23 2A] [LL LL] [WW WW ...]

23 2A = Identifier (# *)

L = Size of the message

W = The message

```
def chat(msg):  
    return "#*" + struct.pack("H", len(msg)) + msg
```



Picking up any item

Script to get the **Great Balls of Fire** (v1):

```
def parse(p_out):

    def chat(msg):
        return "#*" + struct.pack("H", len(msg)) + msg

    def getme(elem_id):
        return "ee" + struct.pack("I", elem_id)

    if chat("GreatBallsOfFire") in p_out:
        p_out += getme(1)

    return p_out
```



Picking up any item

Demo: Get the Great Balls of Fire



Picking up any item

It doesn't work... **Why?**



Picking up any item

Location packet: [6D 76] [XX XX XX XX] [YY YY YY YY] [ZZ ZZ ZZ ZZ] [RR RR] [YY YY] [PP PP] [FF] [SS]

6D 76= Identifier (m v)

X, Y and Z = Position on the X, Y and Z axis of the map

R, Y and P = Direction where to look (roll, yaw, pitch)

F = Direction where I move

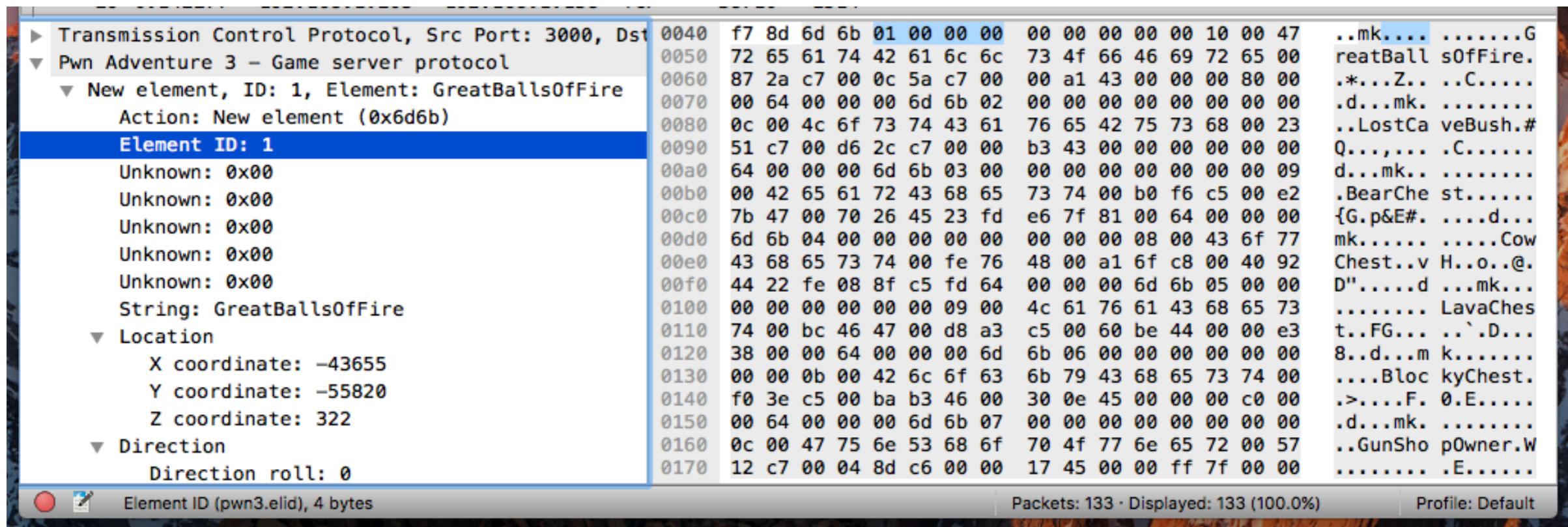
S = Direction where I strafe

```
def loc(x, y, z):  
    return "mv" + struct.pack("IIIfffBB", x, y, z, 0, 0, 0, 0, 0)
```



Picking up any item

What location to fake?



Picking up any item

Script to get the **Great Balls of Fire** (v2):

```
def parse(p_out):

    def chat(msg):
        return "#*" + struct.pack("H", len(msg)) + msg

    def getme(elem_id):
        return "ee" + struct.pack("I", elem_id)

    def loc(x, y, z):
        return "mv" + struct.pack("IIIfffBB", x, y, z, 0, 0, 0, 0, 0)

    if chat("egg1") in p_out:
        p_out += loc(-25045.0, 18085.0, 260.0)
        p_out += getme(11)

    return p_out
```



Picking up any item

Demo: Let's try again to get those Great Balls of Fire



Create elements

Let's generate **new element!**



Create elements

Communication flow:

Get any element:

```
| CLIENT | - - [chat] - - > | PROXY | - - - [chat] [location] [getelement] - - > | GAME SERVER |
| CLIENT | < - - [...] - - | PROXY | < - - - - - - - - - [...] - - - - - - - - - | GAME SERVER |
```

Create an element:

```
| CLIENT | - - - - [chat] - - - - - - - > | PROXY | - - - [chat] - - - > | GAME SERVER |
| CLIENT | < - - - [...] [addelement] - - - | PROXY | < - - - [...] - - - | GAME SERVER |
```



Create elements

Update proxy script:

```
def parse(p_out):  
  
    # DO SOMETHING  
  
    return (p_in, p_out)  
  
...  
  
class Sock(FixedDispatcher):  
  
    ...  
  
    def handle_write(self):  
        if self.write_buffer:  
            (p_in, p_out) = parse(self.write_buffer)  
            self.other.write_buffer += p_in  
            sent = self.send(p_out)  
            self.write_buffer = self.write_buffer[sent:]
```



Create elements

Create element packet:

```
[ 6D 6B] [NN NN NN NN] [?? ?? ?? ?? ??] [LL LL] [WW WW ...] [XX XX XX XX]  
[YY YY YY YY] [ZZ ZZ ZZ ZZ] [RR RR] [YY YY] [PP PP] [?? ?? ?? ??] [00 00]
```

6D 6B = Identifier (m k)

N = Element identifier

? = Always seen as 0x00

L = Length of the name

W = name of the object

X, Y and Z = Position on the X, Y and Z axis of the map

R, Y and P = Direction where to element is pointing to (roll, yaw, pitch)

? = Always seen as 0x00

00 00 = Always seen as 0x00 0x00



Create elements

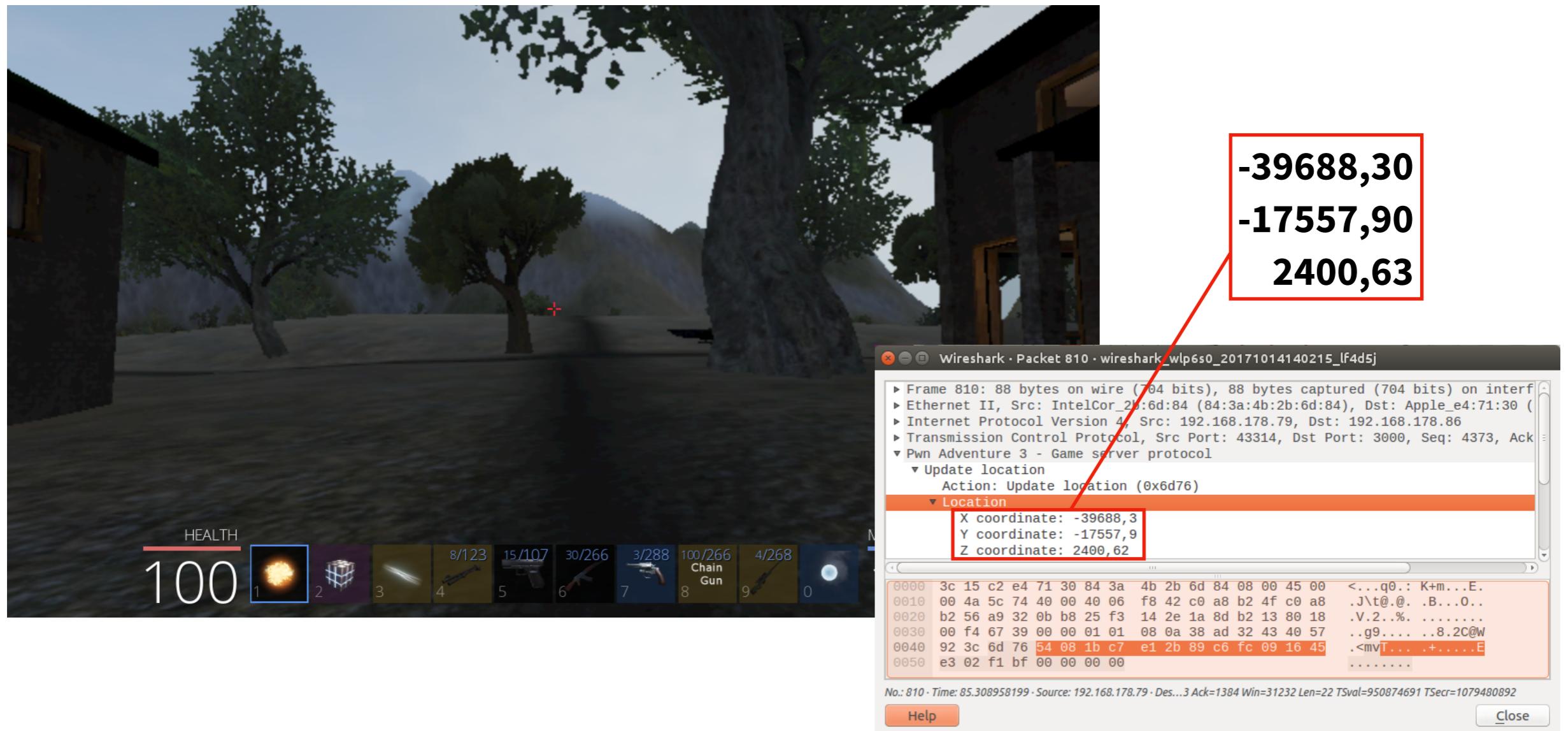
Create element packet:

```
def createel(elid, item, x, y, z):
    packet = 'mk'
    packet += struct.pack("I", elid)
    packet += "\x00\x00\x00\x00\x00"
    packet += struct.pack("H", len(item)) + item
    packet += struct.pack("ffffHHBBBB", x, y, z, 0, 0, 0, 0, 0, 0)
    return packet
```



Create elements

What to create and where to put it?



Create elements

Script to create a new **bear chest**:

```
def parse(p_out):  
  
    p_in = ""  
  
    def createel(elid, item, x, y, z):  
        packet = 'mk'  
        packet += struct.pack("I", elid)  
        packet += "\x00\x00\x00\x00\x00"  
        packet += struct.pack("H", len(item)) + item  
        packet += struct.pack("ffffHHBBBB", x, y, z, 0, 0, 0, 100, 0, 0, 0)  
        return packet  
  
    def chat(msg):  
        return "#*" + struct.pack("H", len(msg)) + msg  
  
    def newspawn(opcode, x, y, z):  
        return opcode + struct.pack("=HfffHHH", 0, x, y, z, 0, 0, 0)  
  
    # Coordinate Town  
    x = -39602.8  
    y = -18288.0  
    z = 2400.28  
  
    if len(p_out) == 22 and p_out[2:4] == "\x00\x00" and p_out[16:] == "\x00\x00\x00\x00\x00\x00":  
        p_out = newspawn(p_out[:2], x, y, z)  
  
    if chat("CreateChest") in p_out:  
        p_in += createel(1337, "BearChest", x + 500, y, z)  
  
    return (p_in, p_out)
```



Create elements

Demo: Create bear chest



Loot & Money

Time to get **rich!**



Loot & Money

New item in inventory packet:

[63 70] [LL LL] [WW WW ...] [QQ QQ QQ QQ]

63 70 = Identifier (c p)

L = Length of the name

W = Name of the element

Q = Quantity

```
def more(item, qt):
    return "cp" + struct.pack("H", len(item)) + item + struct.pack("I", qt)
```



Loot & Money

Script to loot bear skins:

```
def parse(p_out):  
  
    p_in = ""  
  
    def more(item, qt):  
        return "cp" + struct.pack("H", len(item)) + item + struct.pack("I", qt)  
  
    def chat(msg):  
        return "#*" + struct.pack("H", len(msg)) + msg  
  
    if chat("GetSkin") in p_out:  
        p_in += more("BearSkin", 100)  
  
    return (p_in, p_out)
```



Loot & Money

It works! But can we **trade** the item?

Why?



Loot & Money

- **Inventory** is stored on the **server side**
- Before any **transaction/usage**, the **server verify** if you actually have the **item** in your updated inventory stored on the **server side**
- Inventory is **updated** only with items looted/purchased **legitimately**



Building async proxy

Lab 6: Write banner on screen

Banner packet: [65 76] [LL LL] [WW WW ...] [LL LL] [WW WW ...]

65 76 = Identifier (e v)

L = Length of first message

W = First message

L = Length of second message

W = Second message



Building async proxy

Complete proxy:

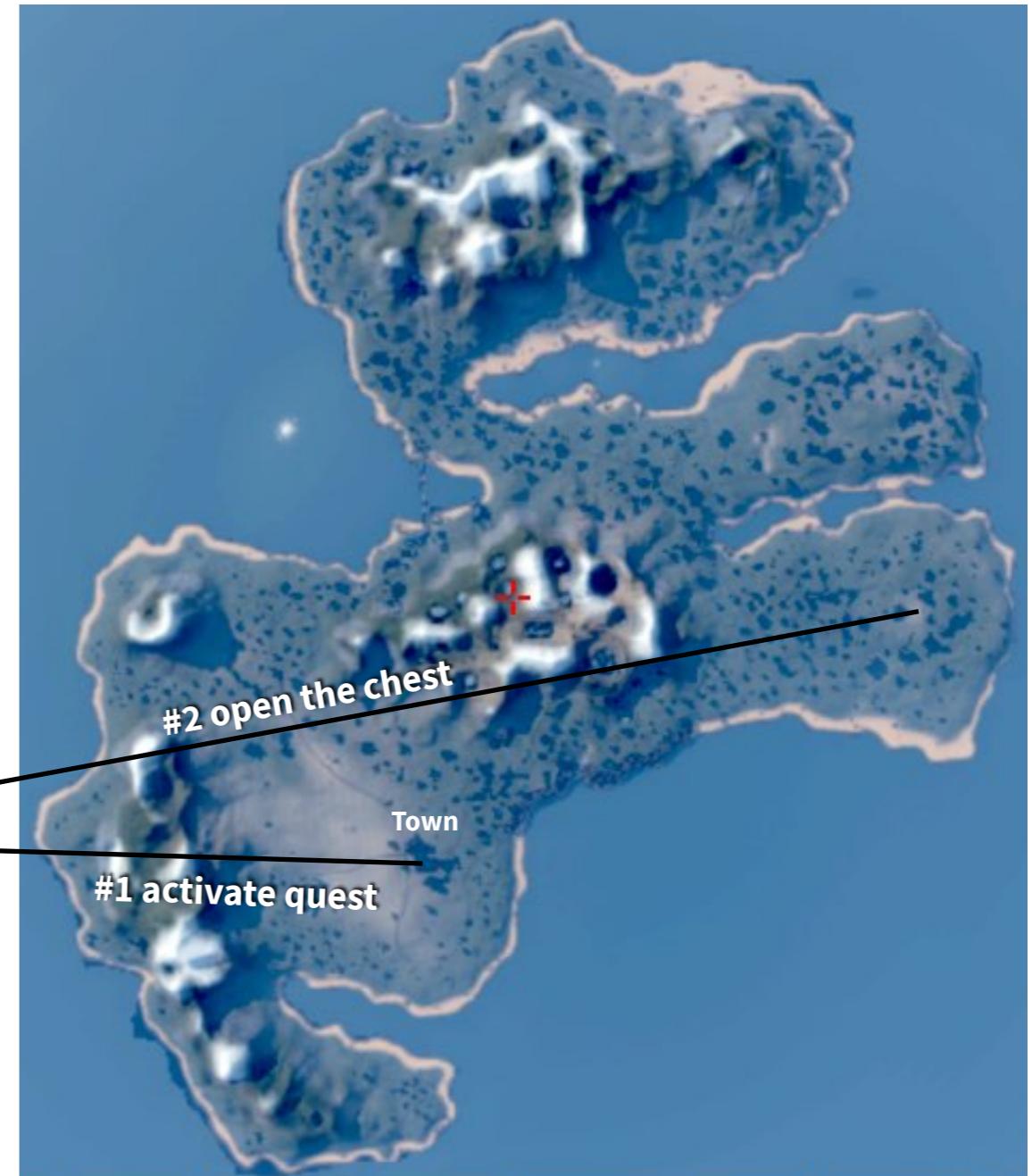
[https://github.com/beaujeant/
PwnAdventure3/blob/master/Network/
pwn3proxy.py](https://github.com/beaujeant/PwnAdventure3/blob/master/Network/pwn3proxy.py)



Unbearable revenge

List of quests:

- *Pirates Treasure* (crack me)
- Fire and Ice (RE binary)
- Until the Cows Come Home (binary patching)
- Egg Hunter (RE network & RE binary)
- Unbearable Revenge (RE network)
- *Blockys Revenge* (logic gate)
- *Overachiever* (finish all achievements)



Unbearable revenge

Where to start?

```
def parse(p_out):  
  
    p_in = ""  
  
    def newspawn(opcode, x, y, z):  
        return opcode + struct.pack("=HfffHHH", 0, x, y, z, 0, 0)  
  
    if ( len(p_out) == 22 and p_out[2:4] == "\x00\x00" and  
        p_out[16:] == "\x00\x00\x00\x00\x00\x00" ):  
  
        x = -7894.0  
        y = 64482.0 + 500  
        z = 2663.0  
  
        p_out = newspawn(p_out[:2], x, y, z)  
  
    return (p_in, p_out)
```



Unbearable revenge

Demo: Start Unbearable Revenge quest



Unbearable revenge

What to do?

- Unlock the chest (5 minutes)
- Stay in near the chest (around 10 meters diameter)
- Survive the different waves of bears
 - Bears
 - Angry bears



Loot & Money

How to protect you?



Unbearable revenge

Climb in the **tree?** **How** to get there?

```
def parse(p_out):  
  
    p_in = ""  
  
    def newspawn(opcode, x, y, z):  
        return opcode + struct.pack("=HfffHHH", 0, x, y, z, 0, 0, 0)  
  
    def loc(x, y, z):  
        return "mv" + struct.pack("ffffHHBB", x, y, z, 0, 0, 0, 0, 0)  
  
    def getme(item):  
        return "ee" + struct.pack("I", item)  
  
    # Bear Chest coordinates  
    x = -7894.0  
    y = 64482.0  
    z = 2663.0  
  
    if len(p_out) == 22 and p_out[2:4] == "\x00\x00" and p_out[16:] == "\x00\x00\x00\x00\x00\x00":  
        p_out = newspawn(p_out[:2], x, y, z + 3000)  
  
    if chat("UnlockChest") in p_out:  
        p_out += loc(x, y, z)  
        p_out += getme(3) # 3 = BearChest element ID  
  
    return (p_in, p_out)
```

Spawn on top

Activate the chest remotely



Unbearable revenge

Demo: Spawn in the tree



Unbearable revenge

Angry bear?

what are you doing?

Angry bear?

Stahp!

What to do now?



Unbearable revenge

Spawn **under** the chest:

- Since under the chest is the void, we create an element underneath to “fly”
- Found a special location where our head is stuck and we can’t fall

```
def parse(p_out):  
  
    p_in = ""  
  
    def newspawn(opcode, x, y, z):  
        return opcode + struct.pack("=HfffHHH", 0, x, y, z, 0, 0, 0)  
    if ( len(p_out) == 22 and p_out[2:4] == "\x00\x00" and  
        p_out[16:] == "\x00\x00\x00\x00\x00\x00\x00" ):  
  
        # Bear Chest coordinates  
        x = -7894.0  
        y = 64482.0  
        z = 2663.0  
  
        p_out = newspawn(p_out[:2], x, y, z - 244)  
  
    return (p_in, p_out)
```



Unbearable revenge

Demo: Spawn under the chest



Unbearable revenge

Gotcha!

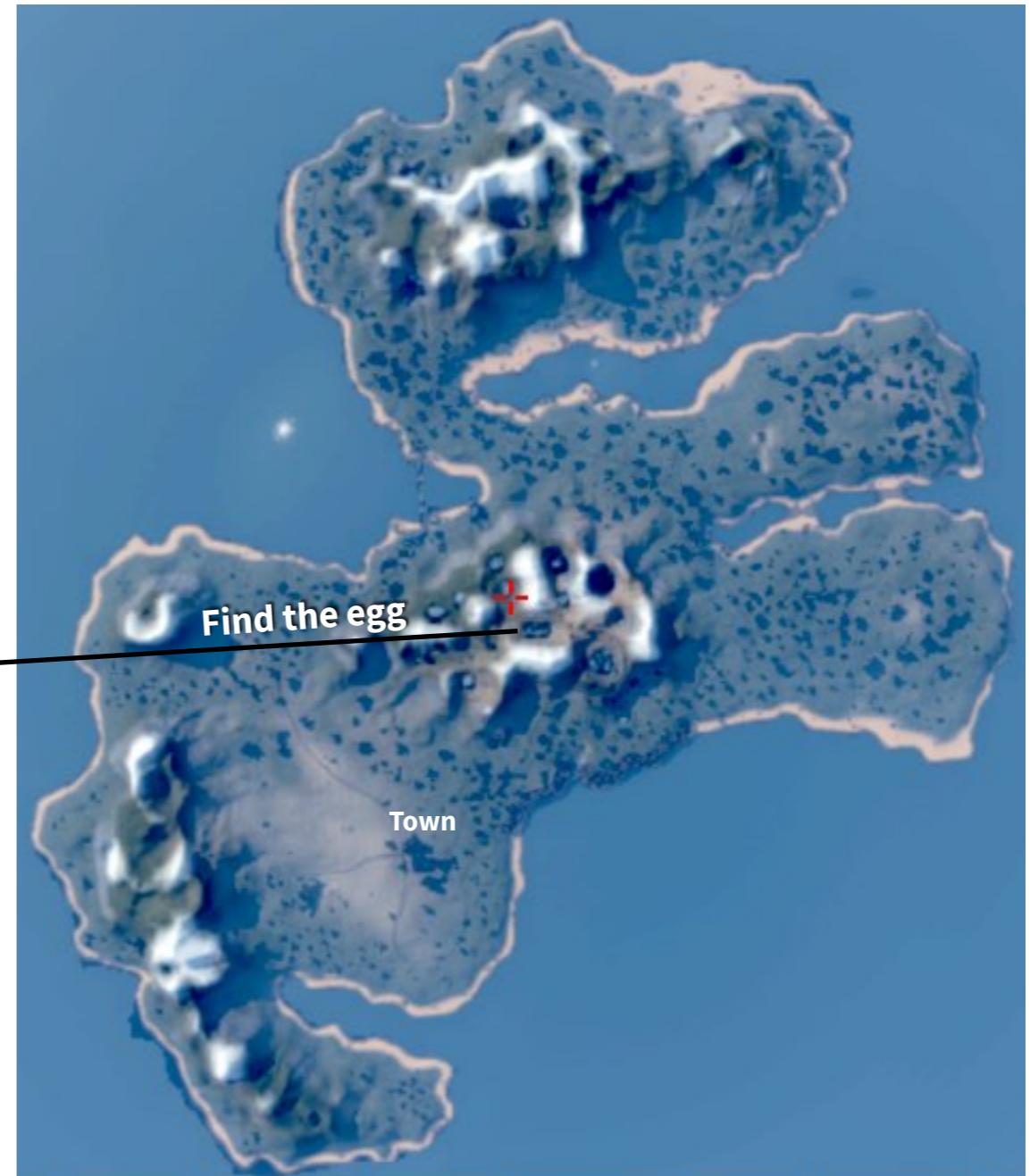
Flag: “*They couldnt bear the sight of you*”



Egg hunter

List of quests:

- *Pirates Treasure (crack me)*
- Fire and Ice (RE binary)
- Until the Cows Come Home (binary patching)
- Egg Hunter (RE network & RE binary)
- Unbearable Revenge (RE network)
- *Blockys Revenge (logic gate)*
- *Overachiever (finish all achievements)*



Egg hunter

Where to start?

```
def parse(p_out):  
  
    p_in = ""  
  
    def newspawn(opcode, x, y, z):  
        return opcode + struct.pack("=HfffHHH", 0, x, y, z, 0, 0)  
  
    if ( len(p_out) == 22 and p_out[2:4] == "\x00\x00" and  
        p_out[16:] == "\x00\x00\x00\x00\x00\x00\x00" ):  
  
        # First egg location  
        x = 24770.0  
        y = 69466.0  
        z = 2727.0  
  
        p_out = newspawn(p_out[:2], x, y, z)  
  
    return (p_in, p_out)
```



Egg hunter

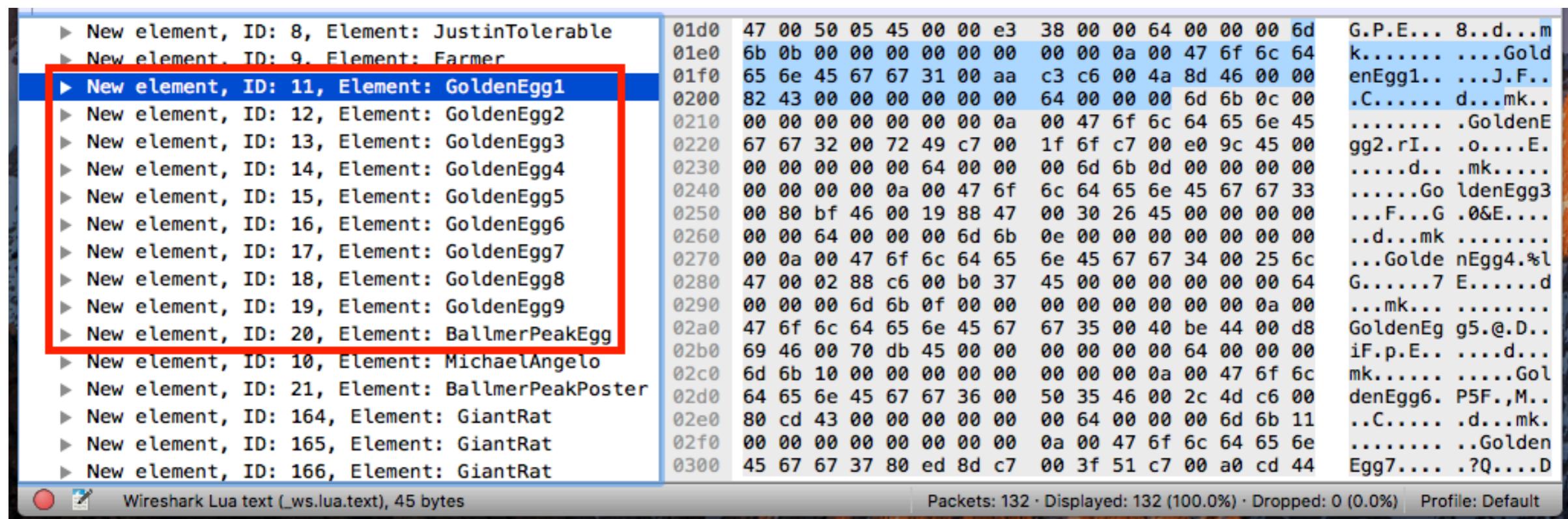
Demo: Start Egg Finder quest



Egg hunter

What to do?

- Collect all the eggs on the map
- Eggs are listed in the init packet



```
► New element, ID: 8, Element: JustinTolerable
► New element, ID: 9, Element: Farmer
► New element, ID: 11, Element: GoldenEgg1
► New element, ID: 12, Element: GoldenEgg2
► New element, ID: 13, Element: GoldenEgg3
► New element, ID: 14, Element: GoldenEgg4
► New element, ID: 15, Element: GoldenEgg5
► New element, ID: 16, Element: GoldenEgg6
► New element, ID: 17, Element: GoldenEgg7
► New element, ID: 18, Element: GoldenEgg8
► New element, ID: 19, Element: GoldenEgg9
► New element, ID: 20, Element: BallmerPeakEgg
► New element, ID: 10, Element: MichaelAngelo
► New element, ID: 21, Element: BallmerPeakPoster
► New element, ID: 164, Element: GiantRat
► New element, ID: 165, Element: GiantRat
► New element, ID: 166, Element: GiantRat
```

Hex	Text
01d0	47 00 50 05 45 00 00 e3 38 00 00 64 00 00 00 6d
01e0	6b 0b 00 00 00 00 00 00 00 00 0a 00 47 6f 6c 64
01f0	65 6e 45 67 67 31 00 aa c3 c6 00 4a 8d 46 00 00
0200	82 43 00 00 00 00 00 00 64 00 00 00 6d 6b 0c 00
0210	00 00 00 00 00 00 00 0a 00 47 6f 6c 64 65 6e 45
0220	67 67 32 00 72 49 c7 00 1f 6f c7 00 e0 9c 45 00
0230	00 00 00 00 00 64 00 00 00 6d 6b 0d 00 00 00 00
0240	00 00 00 00 0a 00 47 6f 6c 64 65 6e 45 67 67 33
0250	00 80 bf 46 00 19 88 47 00 30 26 45 00 00 00 00
0260	00 00 64 00 00 00 6d 6b 0e 00 00 00 00 00 00 00
0270	00 0a 00 47 6f 6c 64 65 6e 45 67 67 34 00 25 6c
0280	47 00 02 88 c6 00 b0 37 45 00 00 00 00 00 00 64
0290	00 00 00 6d 6b 0f 00 00 00 00 00 00 00 00 0a 00
02a0	47 6f 6c 64 65 6e 45 67 67 35 00 40 be 44 00 d8
02b0	69 46 00 70 db 45 00 00 00 00 00 00 64 00 00 00
02c0	6d 6b 10 00 00 00 00 00 00 00 00 0a 00 47 6f 6c
02d0	64 65 6e 45 67 67 36 00 50 35 46 00 2c 4d c6 00
02e0	80 cd 43 00 00 00 00 00 00 64 00 00 00 6d 6b 11
02f0	00 00 00 00 00 00 00 00 0a 00 47 6f 6c 64 65 6e
0300	45 67 67 37 80 ed 8d c7 00 3f 51 c7 00 a0 cd 44



Egg hunter

Script to collect them all:

```
def parse(p_out):  
  
    p_in = ""  
  
    def getme(item):  
        return "ee" + struct.pack("I", item)  
  
    def loc(x, y, z):  
        return "mv" + struct.pack("ffffHHBB", x, y, z, 0, 0, 0, 0, 0)  
  
    def chat(msg):  
        return "#*" + struct.pack("H", len(msg)) + msg  
  
    if chat("egg1") in p_out:  
        p_out += loc(-25045.0, 18085.0, 260.0)  
        p_out += getme(11)  
    if chat("egg2") in p_out:  
        p_out += loc(-51570.0, -61215.0, 5020.0)  
        p_out += getme(12)  
  
    ...  
  
    if chat("egg9") in p_out:  
        p_out += loc(65225.0, -5740.0, 4928.0)  
        p_out += getme(19)  
    if chat("BallmerPeakEgg") in p_out:  
        p_out += loc(-2778.0, -11035.0, 10504.0)  
        p_out += getme(20)  
  
    return (p_in, p_out)
```



Egg hunter

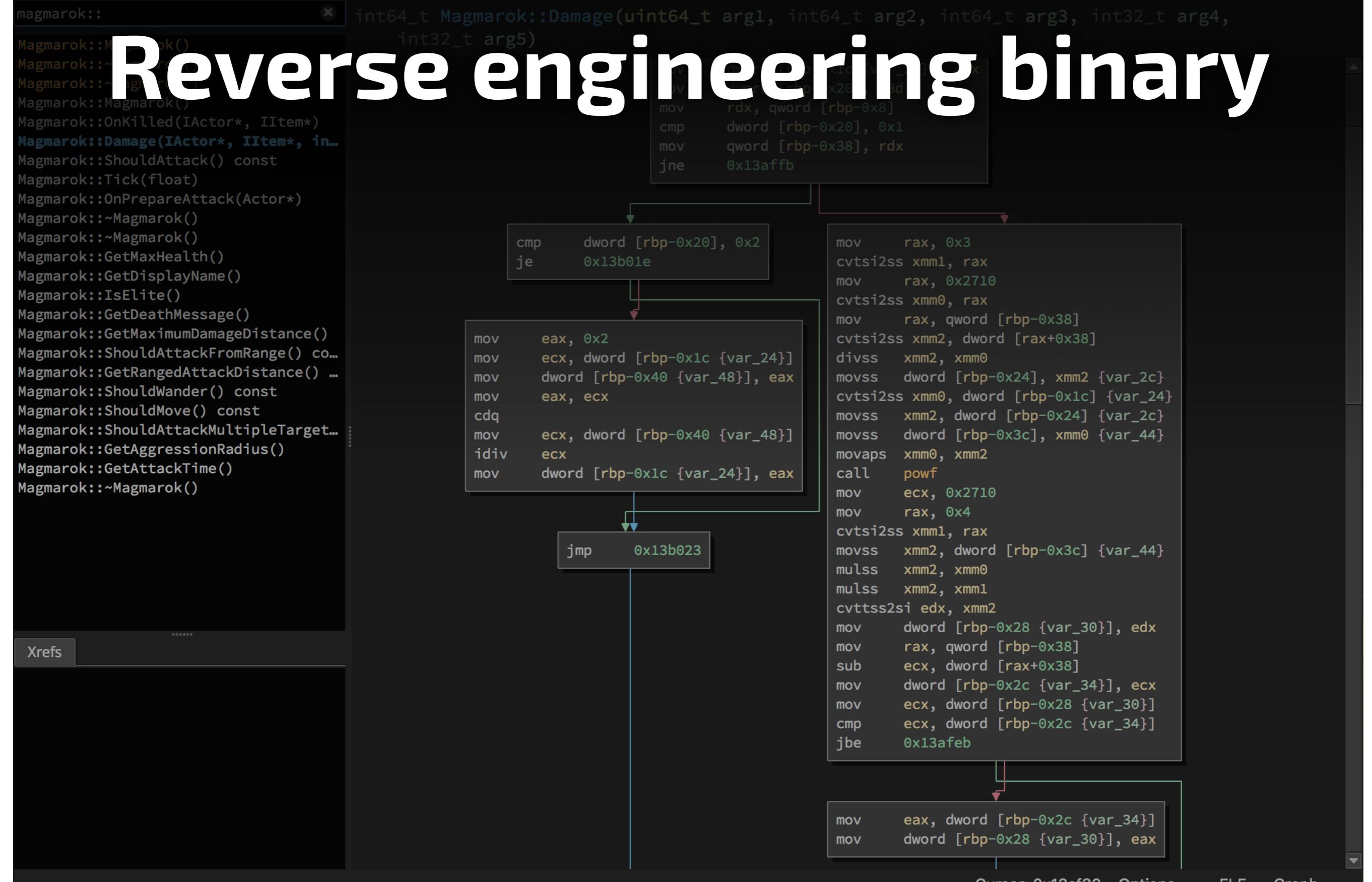
Once in a while, you will have the **game server** that **crash**. This might be due to the sudden jump between locations.

Eventually, you will be able to **collect all eggs** but the “*Ballmer Peak Egg*”.

Let’s have a closer look in... the **binary**!



Reverse engineering binary



RE binary

What binary to reverse?

- Game logic
 - Linux: PwnAdventure3/client/PwnAdventure3_Data/PwnAdventure3/PwnAdventure3/Binaries/Linux/libGameLogic.so
 - macOS: /Applications/Pwn\ Adventure\ 3.app/Contents/PwnAdventure3/PwnAdventure3.app/Contents/MacOS/GameLogic.dylib
 - Windows: PwnAdventure3_Data/PwnAdventure3/PwnAdventure3/Binaries/Win32/GameLogic.dll



RE binary

What binary to reverse?

```
:~$ file libGameLogic.so
libGameLogic.so: ELF 64-bit LSB shared object, x86-64, version 1
(SYSV), dynamically linked, not stripped
```

```
:~$ file GameLogic.dylib
GameLogic.dylib: Mach-O 64-bit dynamically linked shared library
x86_64
```

```
:~$ file GameLogic.dll
GameLogic.dll: PE32 executable (DLL) (GUI) Intel 80386, for MS
Windows
```



What **tools** to use?

- Disassembler
 - Hopper
 - **Binary Ninja**
 - IDA
- Debugger
 - gdb
 - OllyDbg



Assembly:

- Low-level programming language
 - Arithmetic operation
 - Logic operation
 - Data transfer
- Close to machine code instruction



Data: Binary

Representation:

- Hexadecimal
- Decimal - Integer - Float
- Char - String



RE binary

Variables are stored in different **areas** (x64):

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
 - RSI
 - RDI
 - RBP
 - RSP
 - R8
 - R9
 - R...
 - R15
 - XMM8
 - XMM9
 - XMM...
 - XMM15
 - RIP
 - RFLAGS
- Stack
 - Heap



RE binary

Limited set of **instructions**:

- mov
- lea
- push
- pop
- call
- test
- cmp
- jmp, jnz, jb, ...
- xor, or, and, not
- add, sub, mul, ...



RE binary

Limited set of instructions:

- mov
- lea
- push
- pop
- call

Move data:

`mov rax, [rbx]`

; move the value pointed by RBX (register) to RAX

`mov [rbx], rcx`

; move the value of RCX at the address pointed
; by RBX

`mov rdx, [rax-4]`

; move the value pointed by RAX with an offset
; of -4 into RDX

`mov [rdi+rcx], rdx`

; move the value of RDX at the address
; pointed by RDI with an offset of the value in RCX

`mov rdx, [rsi+4*rbx]` ; you got the idea. . .

test

cmp

jmp, jnz, jb

xor, or, and, not

add, sub, mul, ...



RE binary

Limited set of **instructions**:

- mov
- **lea**
- push
- pop
- call

Load effective address:

<code>lea rax, [rbx]</code>	; move the value in RBX (register) into RAX
<code>lea rbx, [rdx+4]</code>	; move the result of ((value in RCX) + 4) ; into RBX
<code>lea rcx, [0x1234]</code>	; move the value 0x1234 into RCX

- test
- cmp
- jmp, jne, jb...
- xor, or, and, not
- add, sub, mul, ...



RE binary

Limited set of **instructions**:

- mov
- lea
- **push**
- pop
- call
- test
- cmp
- jmp, jnz, jh, ...
- xor, or, and, not
- add, sub, mul, ...

Push stack:

```
push  rax      ; push the value in RAX on the stack ...  
push  [rbx]    ; push the value pointed by RBX on the stack
```



RE binary

Limited set of **instructions**:

- mov
- lea
- push
- pop
- call
- test
- cmp
- xor, or, and, not
- add, sub, mul, ...

Pop stack:

```
pop  rax      ; pop the value on top of the stack in RAX  
pop  [rbx]    ; pop the value on top of the stack in the memory  
              ; pointed by RBX
```



RE binary

Limited set of instructions:

- mov
- call a function:
 - The **call** instruction is used to call a function. It first saves (push) the address of the next instruction than jump to the address in the operant.
 - The instruction **ret** is used to return where the function was called. It pop the return address and jump to it.
- test
- jmp, jnp, jz, jb, ...
- push
- pop
- xor, or, and, not
- add, sub, mul, ...
- call / ret



RE binary

Limited set of **instructions**:

- mov
- lea **AND without saving the result:**
- push

```
test  rax, 0x01 ; RAX AND 0x01 (keep only the last bit)
jz    _func1   ; Jump if the AND result is 0
          ; Jump if RAX is even
```
- pop

```
test  rax, rax ; RAX AND RAX = RAX
jz    _func1   ; Jump if the AND result is 0
          ; Jump if RAX = 0
```
- call
- test
- cmp
- add, sub, mul, ...



RE binary

Limited set of **instructions**:

- mov
- lea
- push
- pop
- call
- test
- cmp
- im~~p~~, in~~z~~, jb, ...
Compares two operants and updates flags accordingly:
- add, sub, mul, ...

```
cmp    rax, 0x42    ; Compares the content of RAX with the value 0x42
jeq    _func1        ; If RAX == 0x42, then it jumps

cmp    rcx, rax     ; Compares the content of RCX and with the content of RAX
jge    _func1        ; Jump if RCX >= RAX
```



RE binary

Redirect the execution flow:

Limited set of instructions:

```
jmp _func1 ; Unconditional jump to _func1  
jnz _func1 ; Jump if the result of the previous logic/arithmetic  
             ; operation or comparison is different than 0  
● mov ; move  
je _func1 ; Jump if equal  
jne _func1 ; Jump if not equal  
jb _func1 ; Jump if below (unsigned)  
jbe _func1 ; Jump if below or equal (unsigned)  
jl _func1 ; Jump if below (signed)  
jle _func1 ; Jump if below or equal (signed)  
ja _func1 ; Jump if above (unsigned)  
jae _func1 ; Jump if above or equal (unsigned)  
jg _func1 ; Jump if greater (signed)  
jge _func1 ; Jump if greater or equal (signed)
```

- call

● cmp

● jmp, jnz, jb, ...

● xor, or, and, not

● add, sub, mul, ...



RE binary

Logic operation:

```
xor  rax, 0x42      ; xor the value in RAX with the value  
                     ; in RBX and save the result in RAX  
  
add  rbx, rcx        ; add the value in RBX with the value  
                     ; in RCX and save the value in RBX  
  
sub  [rdx], 0x20      ; subtraction  
inc  rax              ; increments rax  
dec  rbx              ; decrements rbx  
mul  rbx              ; multiplies value in RBX with the value in RAX  
                     ; and store the result higher part in EDX and  
                     ; the result lower part in RAX  
div  rbx              ; divides concat(RDX, RAX) by RBX and store the  
                     ; remainder in RDX and the quotient in RAX
```

• mov

• test

sub [rdx], 0x20

inc rax

dec rbx

mul rbx

div rbx

• pop

• call

• cmp

• jmp, jnz, jb, ..

• xor, or, and, not

• add, sub, mul, ...



RE binary

Variables are stored in different **areas** (x64):

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
 - RSI
 - RDI
 - RBP
 - RSP
 - R8
 - R9
 - R...
 - R15
 - XMM8
 - XMM9
 - XMM...
 - XMM15
 - RIP
 - RFLAGS
- Stack
 - Heap



RE binary

Variables are stored in different areas:

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
 - RSI
 - RDI
 - RBP
 - RSP
 - R8
 - R9
 - R10
 - R11
 - R12
 - R13
 - R14
 - R15
 - Stack
 - Heap
- Accumulator**
• I/O arithmetic
• Return value
- XMM8
 - XMM9
 - XMM10
 - XMM11
 - XMM12
 - XMM13
 - XMM14
 - XMM15
- RIP
 - RFLAGS



RE binary

Variables are stored in different areas:

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
 - RSI
 - RDI
 - RBP
 - RSP
 - R8
 - R9
 - R...
 - R15
- Stack
- Heap
- Base register
 - XMM8
 - XMM9
 - XMM...
 - XMM15
- Index
 - RIP
 - RFLAGS



RE binary

Variables are stored in different areas:

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
 - RSI
 - RDI
 - RBP
 - RSP
 - R8
 - R9
 - R...
 - R15
 - XMM8
 - XMM9
 - XMM...
 - XMM15
- Stack
- Heap
- RIP
- RFLAGS

- **Count register**
- **Loop / Iterative**
- **4th argument**



RE binary

Variables are stored in different areas:

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
 - RSI
 - RDI
 - RBP
 - RSP
 - R8
 - R9
 - R10
 - R11
 - R12
 - R13
 - R14
 - R15
 - Stack
 - Heap
 - XMM8
 - XMM9
 - XMM10
 - XMM11
 - XMM12
 - XMM13
 - XMM14
 - XMM15
 - RIP
 - RFLAGS
- Data register**
● I/O operation
● 3rd arguments



RE binary

Variables are stored in different areas:

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
- Stack
- Heap

- RSI
- RDI
- RBP
- RSP

- **Source**
- **String operations**
- **2nd argument**
- XMM8
- RIP
- R15
- XMM15



RE binary

Variables are stored in different areas:

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
- Stack
- Heap

- RSI
- RDI
- RBP
- RSP

- Destination
 - RIP
 - RDI
 - RBP
 - R15
- String operations
 - RSI
 - RDX
 - R14
 - XMM15
- 1st argument
 - RDI
 - RBP
 - R14
 - XMM15
- XMM15



RE binary

Variables are stored in different memory

• Arguments (R8 and R9)

• Registers • General purpose registers

- | | | | | |
|---------|-------|--------|----------|----------|
| • RAX | • RSI | • R8 | • XMM8 | • RIP |
| • RBX | • RDI | • R9 | • XMM9 | • RFLAGS |
| • RCX | • RBP | • R... | • XMM... | |
| • RDX | • RSP | • R15 | • XMM15 | |
| • Stack | | | | |
| • Heap | | | | |



RE binary

Variables are stored in different areas:

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
 - Stack
 - Heap
- Float operations**
- Return value**
- RSI
 - RDI
 - RBP
 - RSP
- R8
 - R9
 - R...
 - R15
- XMM8
 - XMM9
 - XMM...
 - XMM15
- RIP
 - RFLAGS



RE binary

Variables are stored in different areas:

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
- Stack
- Heap

Instruction pointer

- RSI
- RDI
- RBP
- RSP
- R9
- R...
- R15
- XMM9
- XMM...
- XMM15

• RIP

• RFLAGS



RE binary

- Status register

Variables are stored in different areas:

- Decision making

- Registers

- Carry flag

- Parity flag

- Adjust flag

- Zero flag

- Sign flag

- Trap flag

- Stack

- Interrupt enable flag

- Direction flag

- Overflow flag

- R8

- R9

- R...

- R15

- XMM8

- XMM9

- XMM...

- XMM15

- RIP

- RFLAGS



RE binary

Variables are stored in different areas:

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
 - RSI
 - RDI
 - RBP
 - RSP
 - R8
 - R9
 - R...
 - R15
 - XMM8
 - XMM9
 - XMM...
 - XMM15
 - RIP
 - RFLAGS
- Stack
 - Heap
- Dynamically allocated memory



RE binary

Variables are stored in different areas:

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
 - RSI
 - RDI
 - RBP
 - RSP
 - R8
 - R9
 - R...
 - R15
 - LIFO (push/pop)
 - XMM8
 - XMM9
 - XMM...
 - XMM15
 - Function arguments
 - Local variables
 - RIP
 - RFLAGS
- Stack
- Heap



RE binary

Variables are stored in different areas:

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
 - RSI
 - RDI
 - RBP
 - RSP
 - R8
 - R15
 - XMM8
 - XMM15
 - RIP
 - RFLAGS
- Stack pointer
- Push/Pop
- Stack
- Heap



RE binary

Variables are stored in different areas:

- Registers
 - RAX
 - RBX
 - RCX
 - RDX
 - Stack
 - Heap
 - RSI
 - RDI
 - RBP
 - RSP
- R8
- R9
- R10
- R11
- R12
- R13
- R14
- R15
- XMM8
- XMM9
- XMM10
- XMM11
- XMM12
- XMM13
- XMM14
- XMM15
- RIP
- RFLAGS

● Base pointer
● Stack frame



RE binary

RAX = ?
RDX = ?
Registers
RBP = 0xeffba0
RSP = 0xeffb78
RIP = →
• RAX
• RBX
• RCX
• RDX
• RSI
• RDI
• RBP
• RSP
• Stack
• Heap



main:

```
...  
push arg2  
push arg1  
call func1  
cmp rax, 0x42
```

...

• XMM8 • RIP

func1:

```
push rbp  
mov rbp, rsp  
sub rsp, 0x08 ; var1  
sub rsp, 0x08 ; var2  
mov rax, [rbp+0x10] ; arg1  
mov rdx, [rbp+0x18] ; arg2  
...  
mov rax, [rsp+0x08] ; var1  
mov rsp, rbp  
pop rbp  
ret
```



RE binary

RAX = ?
RDX = ?
Registers
RBP = 0xeffbba0
RSP = 0xeffb70
RIP = →
• RAX
• RBX
• RCX
• RDX
• Stack
• Heap



main:

...
→ push arg2
push arg1
call func1
cmp rax, 0x42

• XMM8 • RIP

func1:
push rbp
mov rbp, rsp
sub rsp, 0x08 ; var1
sub rsp, 0x08 ; var2

• XMM15
mov rax, [rbp+0x10] ; arg1
mov rdx, [rbp+0x18] ; arg2
...
mov rax, [rsp+0x08] ; var1
mov rsp, rbp
pop rbp
ret



RE binary

RAX = ?
RDX = ?
Registers
RBP = 0xeffbba0
RSP = 0xeffb68
RIP = →
• RAX
• RBX
• RCX
• RDX
• Stack
• Heap



main:

...
push arg2
push arg1
→ call func1
cmp rax, 0x42

• XMM8 • RIP

func1:
• XMM9 • RFLAGS
push rbp
mov rbp, rsp
• XMM10
sub rsp, 0x08 ; var1
sub rsp, 0x08 ; var2

• XMM11
mov rax, [rbp+0x10] ; arg1
mov rdx, [rbp+0x18] ; arg2
...
mov rax, [rsp+0x08] ; var1
mov rsp, rbp
pop rbp
ret



RE binary

RAX = ?
RDX = ?
Registers
RBP = 0xeffba0
RSP = 0xeffb60
RIP = →
• RAX
• RBX
• RCX
• RDX
• Stack
• Heap



main:

...
push arg2
push arg1
call func1
cmp rax, 0x42

...

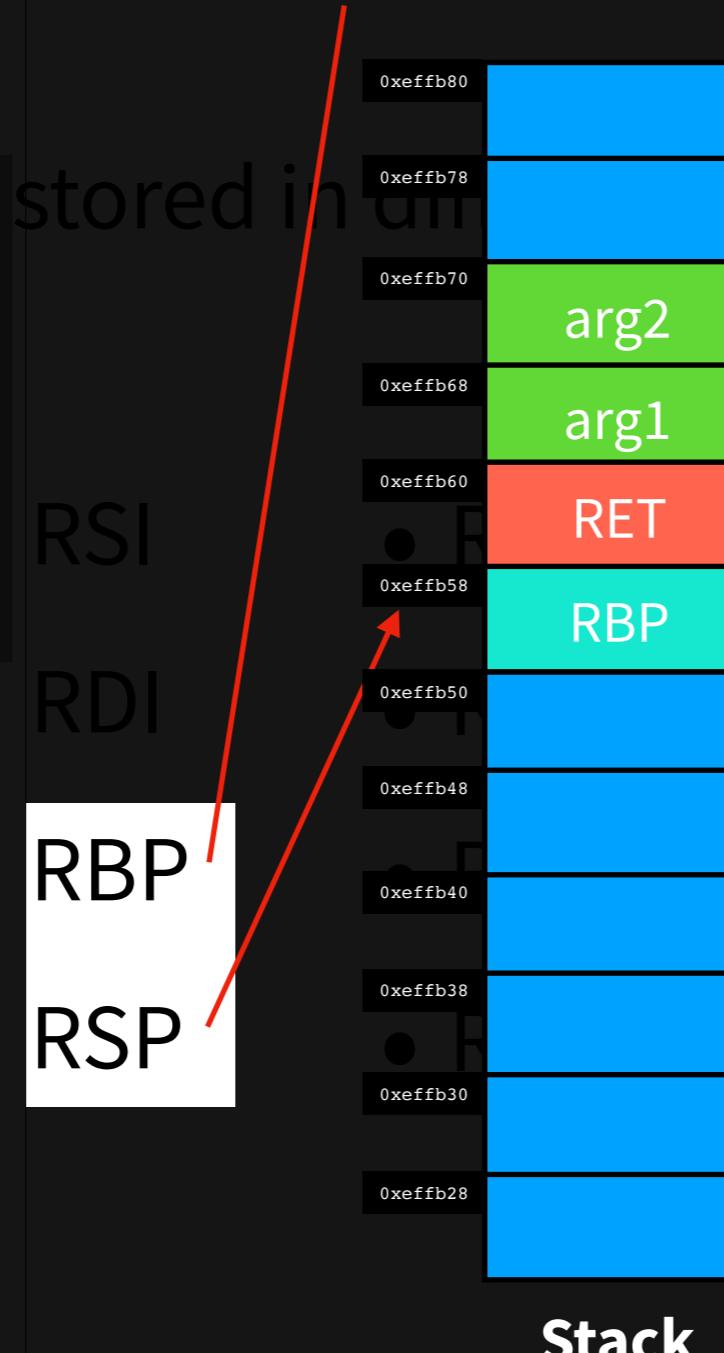
func1:

push rbp
mov rbp, rsp
sub rsp, 0x08 ; var1
sub rsp, 0x08 ; var2
mov rax, [rbp+0x10] ; arg1
mov rdx, [rbp+0x18] ; arg2
...
mov rax, [rsp+0x08] ; var1
mov rsp, rbp
pop rbp
ret



RE binary

RAX = ?
RDX = ?
Registers
RBP = 0xeffba0
RSP = 0xeffb58
RIP = →
• RAX
• RBX
• RCX
• RDX
• Stack
• Heap



main:

```
...  
push arg2  
push arg1  
call func1  
cmp rax, 0x42
```

• XMM8 • RIP

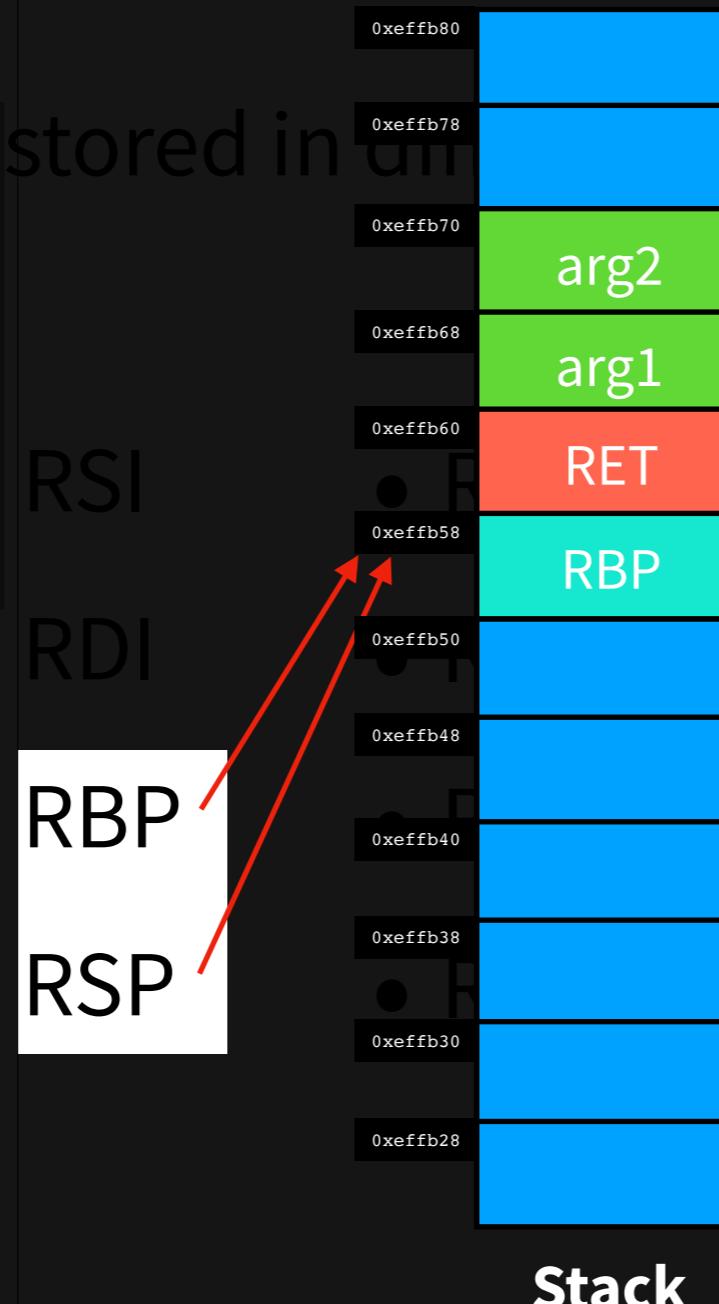
func1:
• XMM9 • RFLAGS
push rbp
→ mov rbp, rsp
• XMM10
sub rsp, 0x08 ; var1
sub rsp, 0x08 ; var2

• XMM11
mov rax, [rbp+0x10] ; arg1
mov rdx, [rbp+0x18] ; arg2
...
mov rax, [rsp+0x08] ; var1
mov rsp, rbp
pop rbp
ret



RE binary

- RAX = ?
- RDX = ?
- Registers
 - RBP = **0xeffb58**
 - RSP = **0xeffb58**
 - RIP = →
 - RBX
 - RDI
 - RCX
 - RDX
 - Stack
 - Heap



main:

```
...  
push arg2  
push arg1  
call func1  
cmp rax, 0x42
```

XMM8

func1:

```
push rbp  
mov rbp, rsp  
sub rsp, 0x08 ; var1  
sub rsp, 0x08 ; var2  
mov rax, [rbp+0x10] ; arg1  
mov rdx, [rbp+0x18] ; arg2  
...  
mov rax, [rsp+0x08] ; var1  
mov rsp, rbp  
pop rbp  
ret
```

RIP

RFLAGS

```
XMM9  
sub rsp, 0x08 ; var1  
sub rsp, 0x08 ; var2
```

XMM10

```
mov rax, [rbp+0x10] ; arg1  
mov rdx, [rbp+0x18] ; arg2
```

...
XMM11

```
mov rax, [rsp+0x08] ; var1  
mov rsp, rbp
```

XMM12

```
pop rbp
```

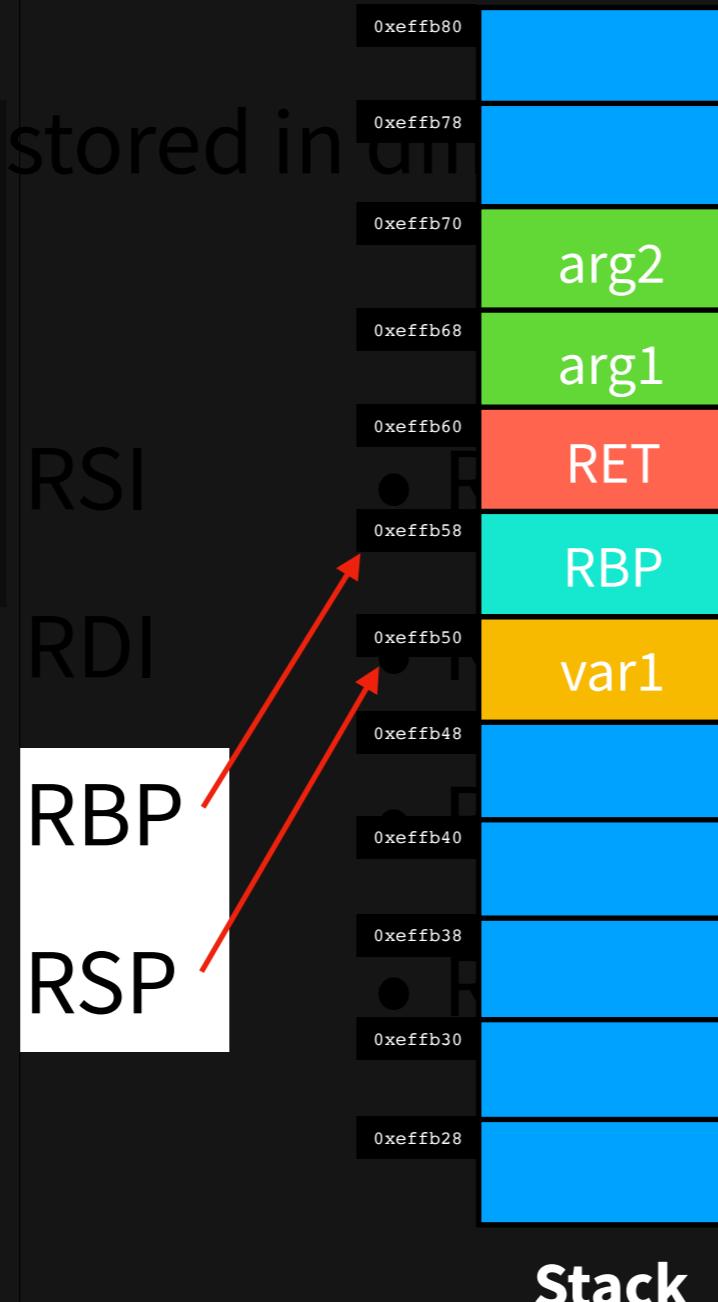
XMM13

```
ret
```



RE binary

- RAX = ?
- RDX = ?
- Registers
 - RBP = 0xeffb58
 - RSP = 0xeffb50
 - RIP = →
 - RBX
 - RDI
 - RCX
 - RDX
 - Stack
 - Heap



main:

```
...  
push arg2  
push arg1  
call func1  
cmp rax, 0x42
```

...
XMM8

RIP

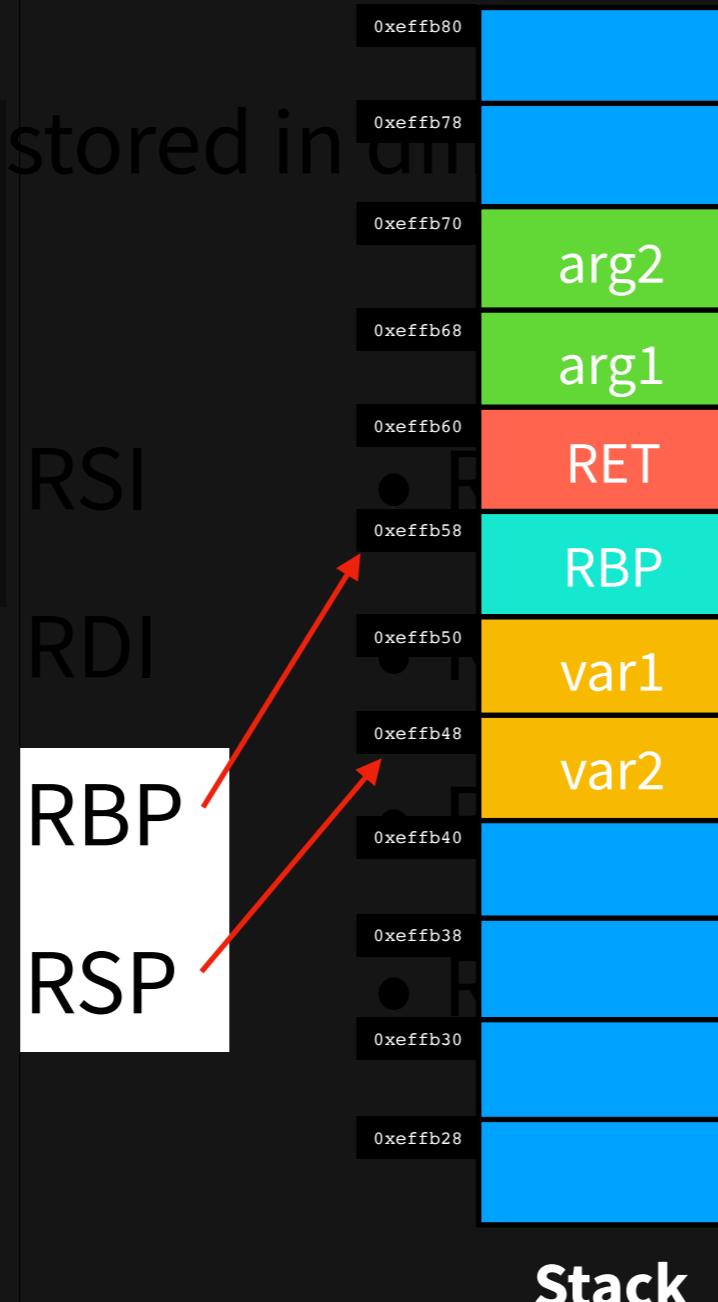
func1:

```
push rbp  
mov rbp, rsp  
sub rsp, 0x08 ; var1  
sub rsp, 0x08 ; var2  
mov rax, [rbp+0x10] ; arg1  
mov rdx, [rbp+0x18] ; arg2  
...  
mov rax, [rsp+0x08] ; var1  
mov rsp, rbp  
pop rbp  
ret
```



RE binary

- RAX = ?
- RDX = ?
- Registers
 - RBP = 0xeffb58
 - RSP = 0xeffb48
 - RIP = →
 - RBX
 - RDI
 - RCX
 - RDX
 - Stack
 - Heap



main:

```
...  
push arg2  
push arg1  
call func1  
cmp rax, 0x42
```

...

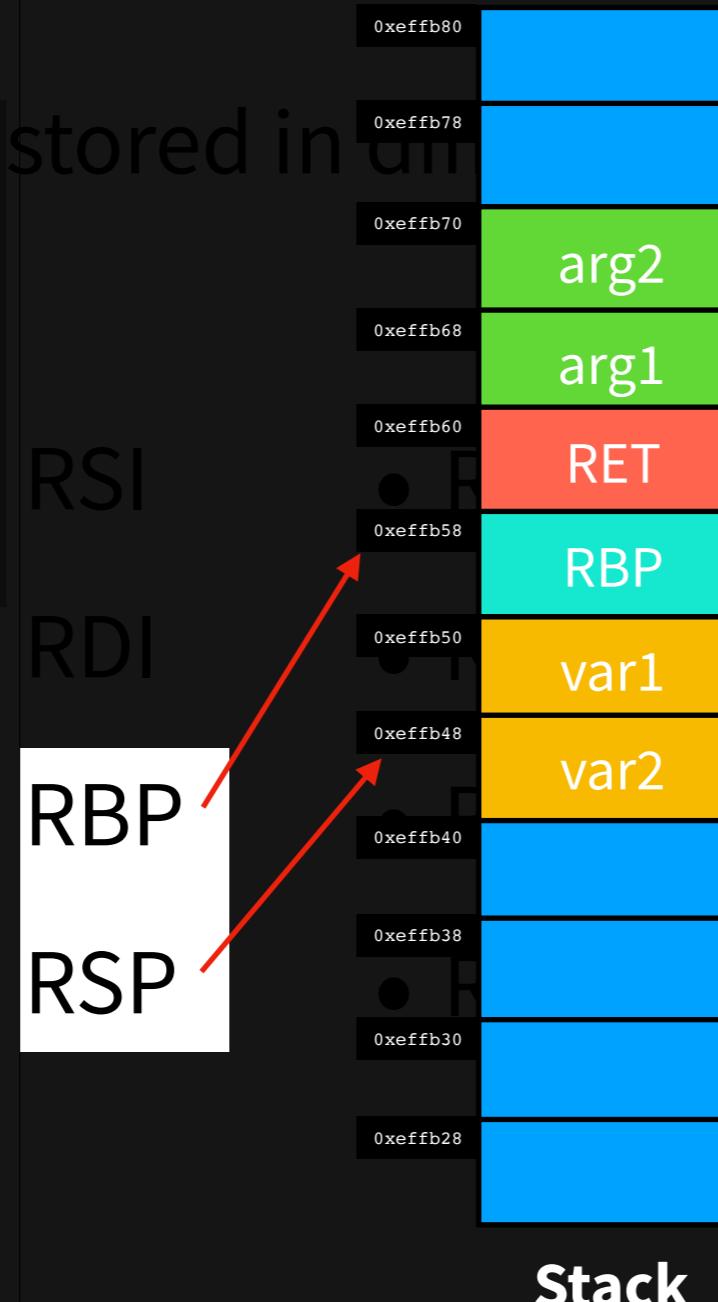
func1:

```
push rbp  
mov rbp, rsp  
sub rsp, 0x08 ; var1  
sub rsp, 0x08 ; var2  
...  
mov rax, [rbp+0x10] ; arg1  
mov rdx, [rbp+0x18] ; arg2  
...  
mov rax, [rsp+0x08] ; var1  
mov rsp, rbp  
pop rbp  
ret
```



RE binary

- RAX = ?
- RDX = ?
- Registers
 - RBP = 0xeffb58
 - RSP = 0xeffb48
 - RAX
 - RIP = →
 - RBX
 - RDI
 - RCX
 - RDX
 - Stack
 - Heap



main:

```
...  
push arg2  
push arg1  
call func1  
cmp rax, 0x42
```

...

XMM8 RIP

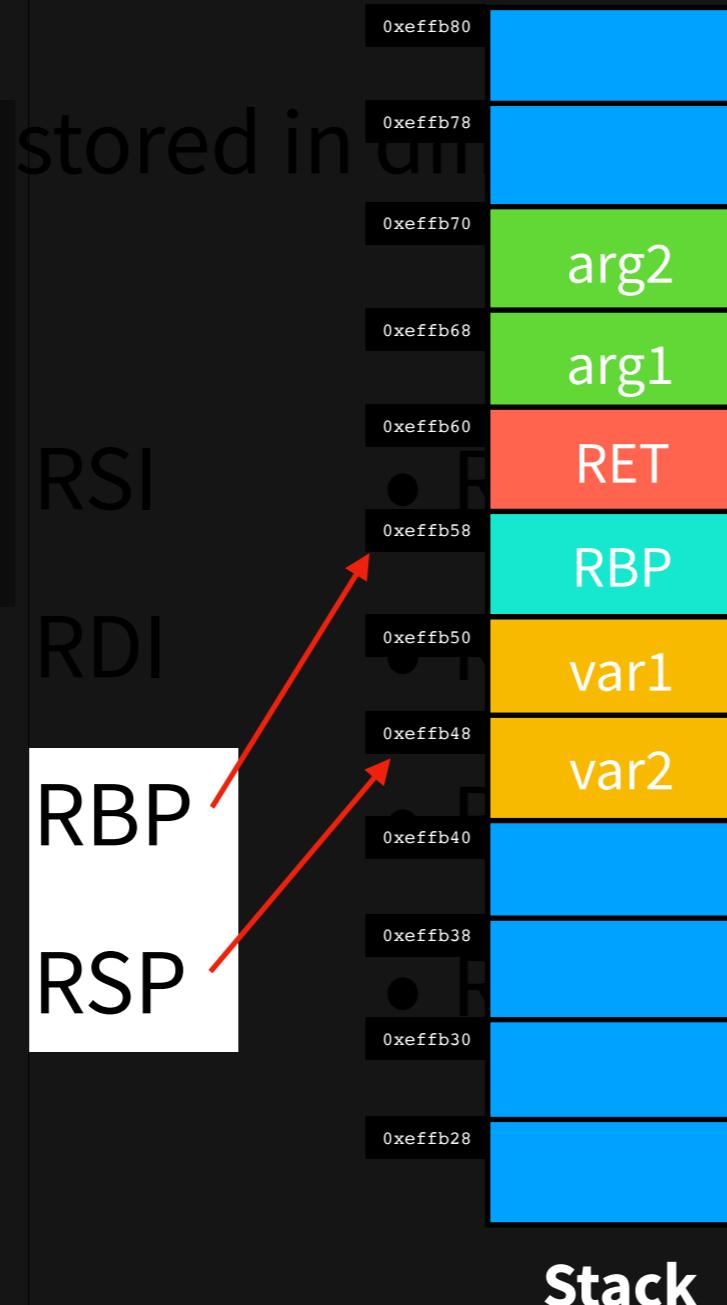
func1:

```
push rbp  
mov rbp, rsp  
sub rsp, 0x08 ; var1  
sub rsp, 0x08 ; var2  
...  
mov rax, [rbp+0x10] ; arg1  
mov rdx, [rbp+0x18] ; arg2  
...  
mov rax, [rsp+0x08] ; var1  
mov rsp, rbp  
pop rbp  
ret
```



RE binary

RAX = arg1
RDX = ?
Registers
RBP = 0xeffb58
RSP = 0xeffb48
RIP = →
• RBX
• RCX
• RDX
• Stack
• Heap



main:

...
push arg2
push arg1
call func1
cmp rax, 0x42

...
XMM8

• RIP

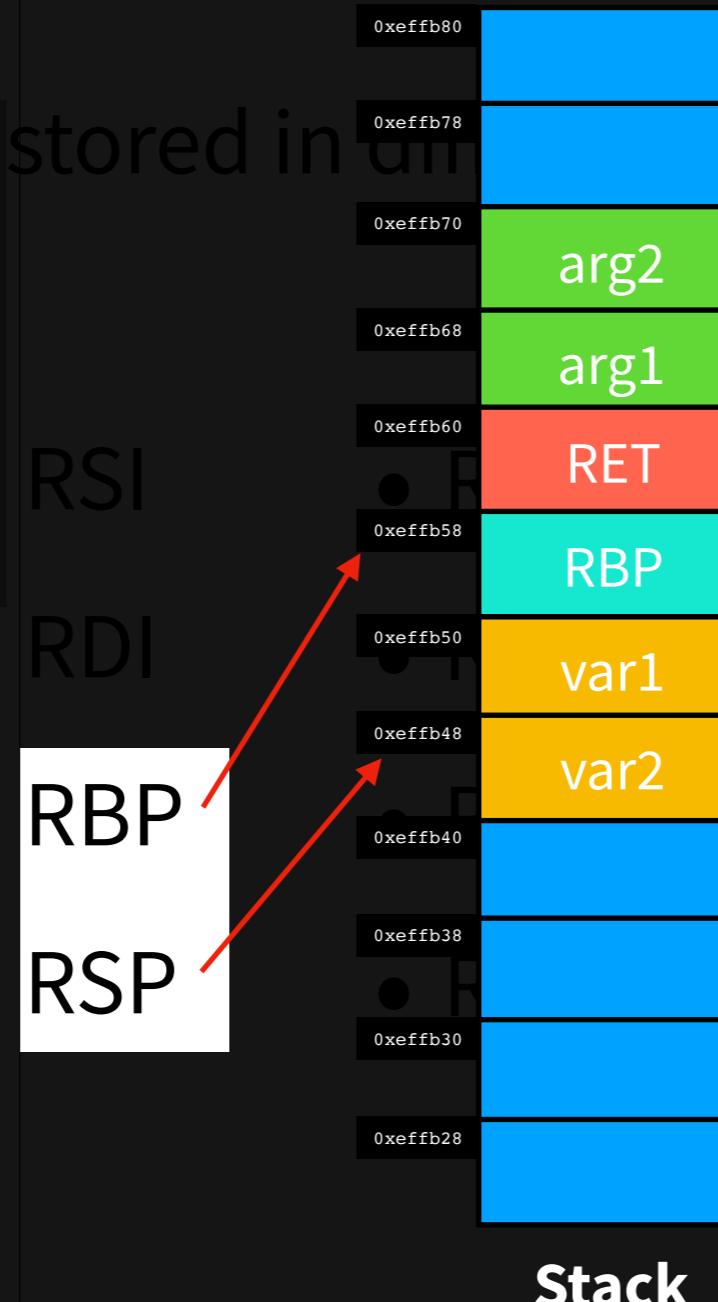
func1:

push rbp
mov rbp, rsp
sub rsp, 0x08 ; var1
sub rsp, 0x08 ; var2
mov rax, [rbp+0x10] ; arg1
→ mov rdx, [rbp+0x18] ; arg2
...
mov rax, [rsp+0x08] ; var1
mov rsp, rbp
pop rbp
ret



RE binary

- RAX = arg1
- RDX = arg2
- Registers
 - RBP = 0xeffb58
 - RSP = 0xeffb48
 - RIP = →
 - RBX
 - RDI
 - RCX
 - RDX
 - Stack
 - Heap



main:

```
...  
push arg2  
push arg1  
call func1  
cmp rax, 0x42
```

...
XMM8
RIP

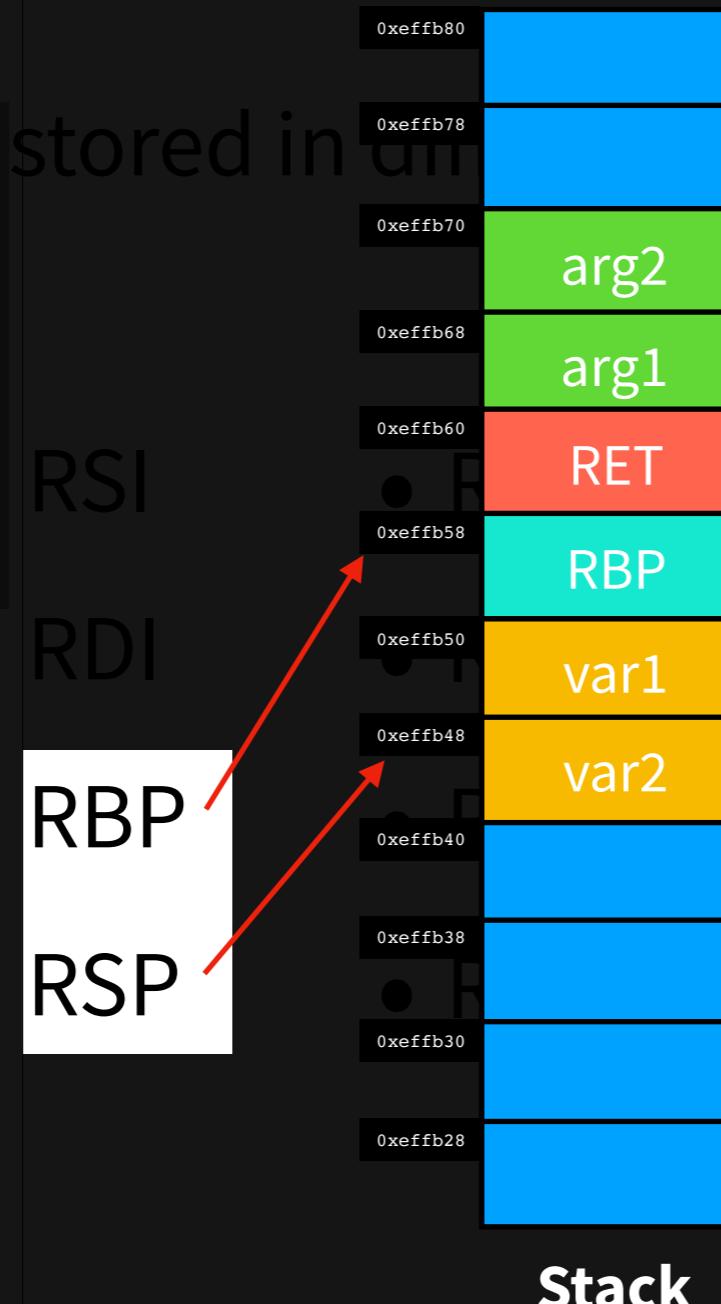
func1:

```
push rbp  
mov rbp, rsp  
sub rsp, 0x08 ; var1  
sub rsp, 0x08 ; var2  
...  
mov rax, [rbp+0x10] ; arg1  
mov rdx, [rbp+0x18] ; arg2  
...  
mov rax, [rsp+0x08] ; var1  
mov rsp, rbp  
pop rbp  
ret
```



RE binary

- RAX = ?
- RDX = ?
- Registers
 - RBP = 0xeffb58
 - RSP = 0xeffb48
 - RIP = →
 - RBX
 - RDI
 - RCX
 - RDX
 - Stack
 - Heap



main:

```
...  
push arg2  
push arg1  
call func1  
cmp rax, 0x42
```

...

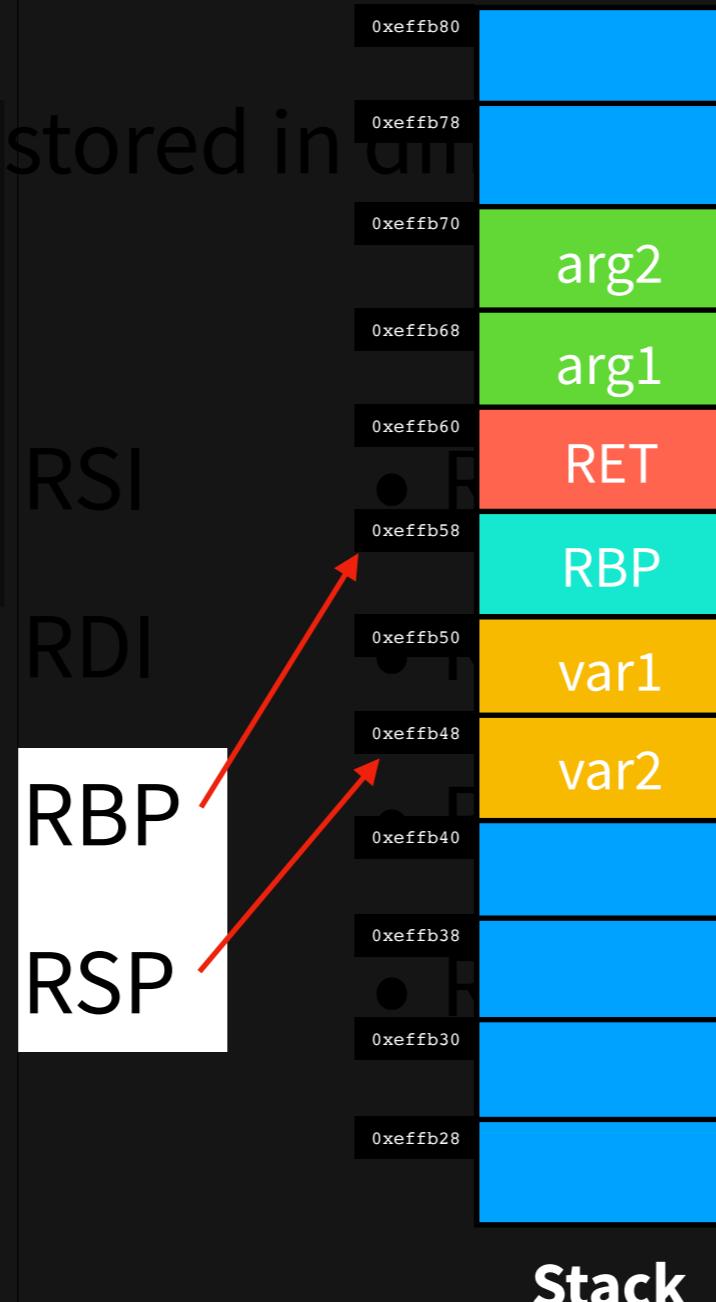
func1:

```
push rbp  
mov rbp, rsp  
sub rsp, 0x08 ; var1  
sub rsp, 0x08 ; var2  
...  
→ mov rax, [rbp+0x10] ; arg1  
mov rdx, [rbp+0x18] ; arg2  
...  
→ mov rax, [rsp+0x08] ; var1  
mov rsp, rbp  
pop rbp  
ret
```



RE binary

- RAX = var1
- RDX = ?
- Registers
 - RBP = 0xeffb58
 - RSP = 0xeffb48
 - RAX
 - RIP = →
 - RBX
 - RDI
 - RCX
 - RDX
 - Stack
 - Heap



main:

```
...  
push arg2  
push arg1  
call func1  
cmp rax, 0x42
```

...

XMM8 RIP

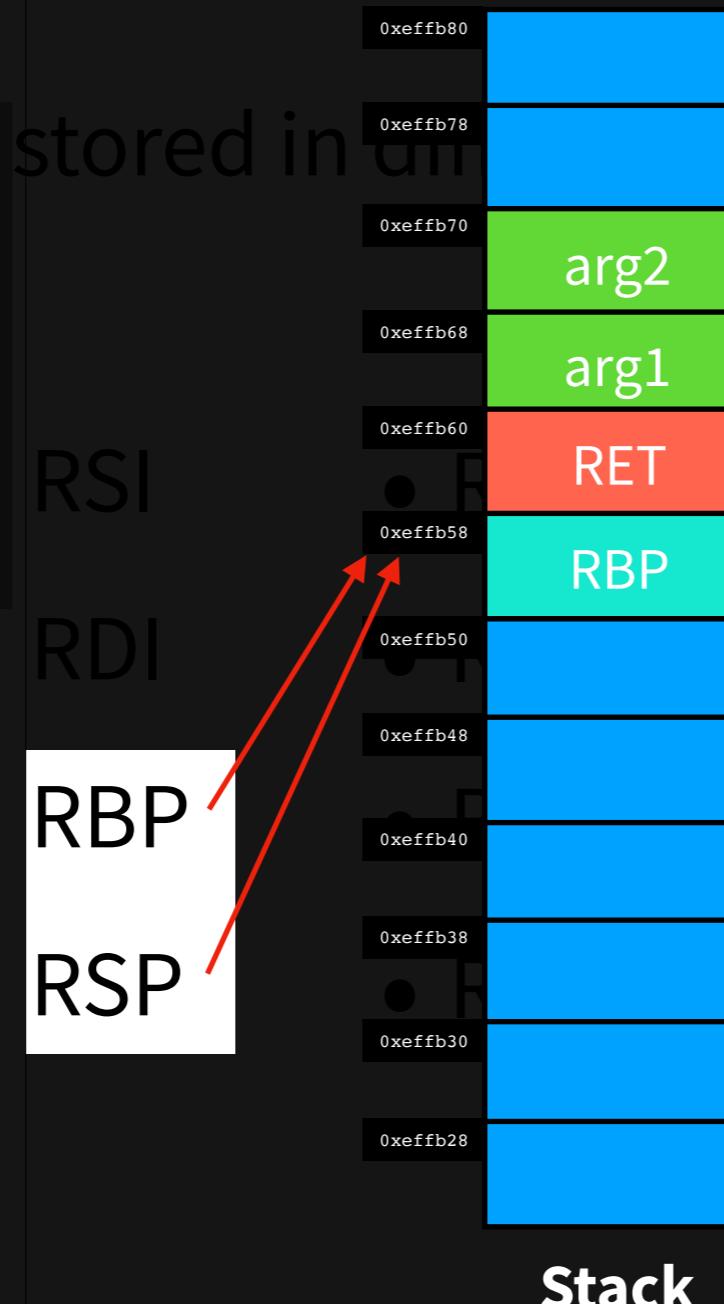
func1:

```
push rbp  
mov rbp, rsp  
sub rsp, 0x08 ; var1  
sub rsp, 0x08 ; var2  
...  
mov rax, [rbp+0x10] ; arg1  
mov rdx, [rbp+0x18] ; arg2  
...  
mov rax, [rsp+0x08] ; var1  
→ mov rsp, rbp  
pop rbp  
ret
```



RE binary

RAX = var1
RDX = ?
Registers
RBP = 0xeffb58
RSP = 0xeffb58
RIP = →
• RBX
• RCX
• RDX
• Stack
• Heap



main:

...
push arg2
push arg1
call func1
cmp rax, 0x42

...

• XMM8 • RIP

func1:

push rbp
mov rbp, rsp
sub rsp, 0x08 ; var1
sub rsp, 0x08 ; var2
mov rax, [rbp+0x10] ; arg1
mov rdx, [rbp+0x18] ; arg2
...
mov rax, [rsp+0x08] ; var1
mov rsp, rbp
→ pop rbp
ret



RE binary

RAX = var1
RDX = ?
Registers
RBP = 0xeffbba0
RSP = 0xeffb58
RIP = →
• RAX
• RBX
• RCX
• RDX
• Stack
• Heap



main:

...
push arg2
push arg1
call func1
cmp rax, 0x42

...

func1:

push rbp
mov rbp, rsp
sub rsp, 0x08 ; var1
sub rsp, 0x08 ; var2
...
mov rax, [rbp+0x10] ; arg1
mov rdx, [rbp+0x18] ; arg2
...
mov rax, [rsp+0x08] ; var1
mov rsp, rbp
pop rbp
→ ret



RE binary

RAX = var1
RDX = ?
• Registers
RBP = 0xeffba0
RSP = 0xeffb58
• RAX
RIP = →
• RBX
• RDI
• **RBP**
• RSP
• Stack
• Heap



main:

...
push arg2
push arg1
call func1
→ cmp rax, 0x42

• XMM8 • RIP

func1:
• XMM9 • RFLAGS
push rbp
mov rbp, rsp
• XMM10
sub rsp, 0x08 ; var1
sub rsp, 0x08 ; var2

• XMM11
• XMM12
mov rax, [rbp+0x10] ; arg1
mov rdx, [rbp+0x18] ; arg2
...
mov rax, [rsp+0x08] ; var1
mov rsp, rbp
pop rbp
ret



RE binary

RAX = var1
RDX = ?
• Registers
RBP = 0xeffba0
RSP = 0xeffb58
• RAX
RIP = →
• RBX
• RDI
• **RBP**
• RSP
• Stack
• Heap



main:

...
push arg2
push arg1
call func1
cmp rax, 0x42

...
• XMM8 • RIP

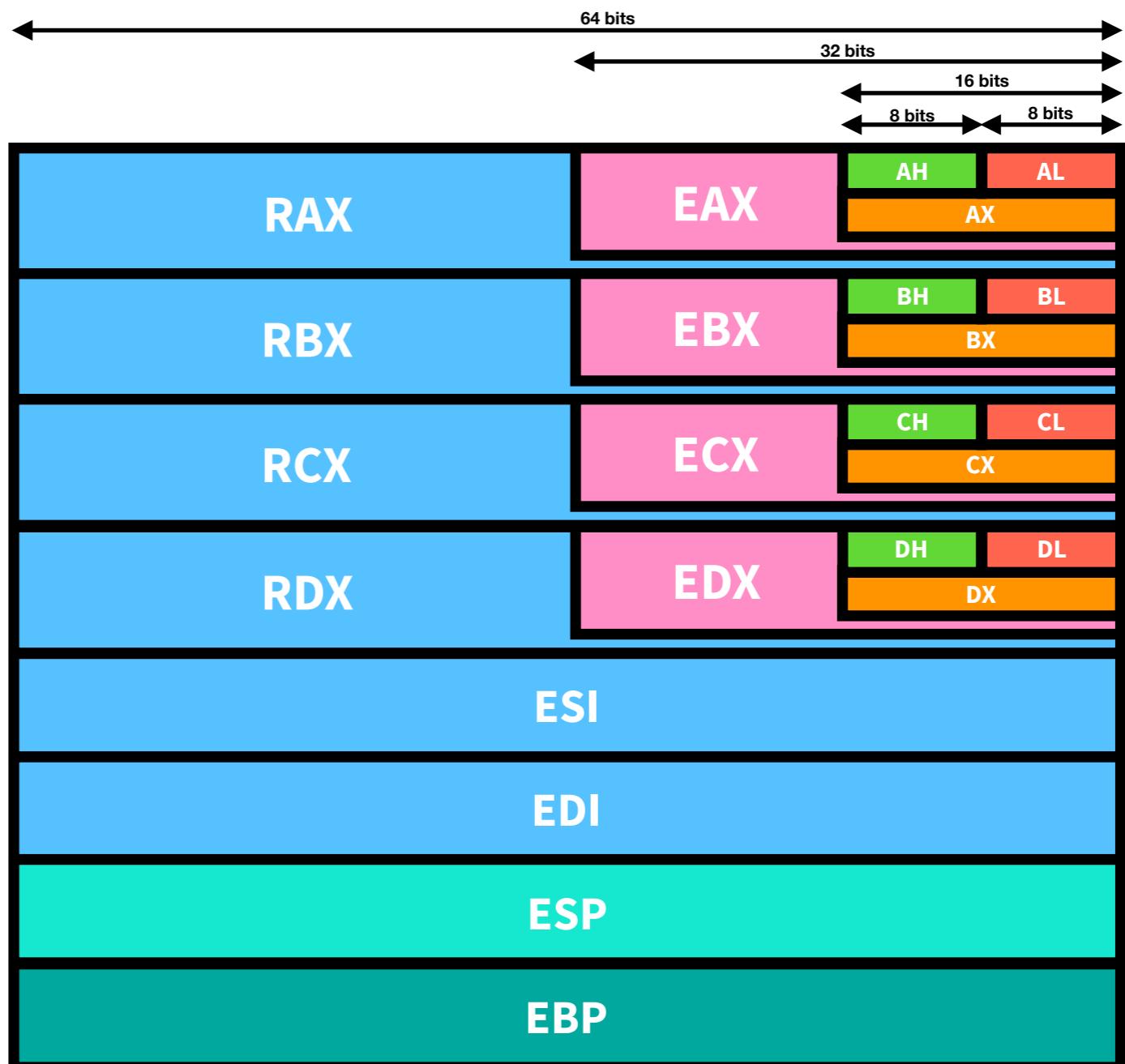
func1:

push rbp
mov rbp, rsp
sub rsp, 0x08 ; var1
sub rsp, 0x08 ; var2
mov rax, [rbp+0x10] ; arg1
mov rdx, [rbp+0x18] ; arg2
...
mov rax, [rsp+0x08] ; var1
mov rsp, rbp
pop rbp
ret



RE binary

Registers sizes:



Lab 7: What's the purpose of this function?

```
section .text
    global _start

_start:
    mov     rax, msg
    mov     rcx, len

_next:
    sub     rcx, 1
    cmp     byte [rax+rcx], 0x41
    jl      _next
    cmp     byte [rax+rcx], 0x5a
    jg      _next
    xor     byte [rax+rcx], 0x20
    test    rcx, rcx
    jne    _next
    ...

section .data

msg db "Hello World!", 0
len equ    $ - msg ; msg is pointing to the string array
                   ; equivalent of len = strlen(msg)
```



Binary patching



Binary patching

When you **run an application**, you **execute instructions** that have been writing by the developers for a **specific purpose**.

Sometimes, the **instructions restrict you**:

- An evaluation version of an application that exit after 30 minutes
- Your player in your MMORPG that cannot jump high enough to access hidden high location



Binary patching

The executed **instructions** are located on **your computer**, which you **control**:

- Read binary: Disassembling
- Execute binary: Debugging
- Edit binary: Patching

You can therefore **change** those **instructions** to **bypass restrictions** (e.g. cracking) but also to **improve** or **add new functionalities**.



Binary patching

Let's run faster!



Binary patching

Functions related to **movements**:

Function name	Segment	Start
f ClientHandler::Sprint(void)	.text	00000000000141AC0
f ClientHandler::Sprint(void)	.plt	0000000000011A2A0
f ClientHandler::Sprint(void)::\$_5::operator() const(void)	.text	00000000000148A90
f ClientWorld::Sprint(bool)	.text	0000000000013C540
f GameServerConnection::Sprint(bool)	.text	000000000001C3690
f GameServerConnection::Sprint(bool)	.plt	0000000000011C590
f GameServerConnection::Sprint(bool)::\$_47::operator() const(void)	.text	000000000001C8890
f LocalWorld::Sprint(bool)	.text	000000000001BDAB0
f Player::GetJumpSpeed(void)	.text	0000000000015EA80
f Player::GetJumpSpeed(void)	.plt	0000000000010ED90
f Player::GetSprintMultiplier(void)	.text	0000000000015EA60
f Player::GetSprintMultiplier(void)	.plt	00000000000117CD0
f Player::GetWalkingSpeed(void)	.text	0000000000015EA40
f Player::GetWalkingSpeed(void)	.plt	00000000000123FB0
f Player::SetSprintState(bool)	.text	000000000001515C0
f Player::SetSprintState(bool)	.plt	0000000000010F160
f ServerWorld::Sprint(bool)	.text	00000000000227D50



Binary patching

Player::GetSprintMultiplier(void) function:

Non-scalar types (including floats and doubles) are returned in the **XMM0** register

```
; Attributes: static bp-based frame
; float __cdecl Player::GetSprintMultiplier(Player *this)
public _ZN6Player19GetSprintMultiplierEv ; weak
_ZN6Player19GetSprintMultiplierEv proc near
    this= qword ptr -8
    push    rbp
    mov     rbp, rsp
    mov     rax, 3          ← 30
    cvtsi2ss xmm0, rax
    mov     [rbp+this], rdi
    pop    rbp
    retn
_ZN6Player19GetSprintMultiplierEv endp
```

```
float Player::GetSprintMultiplier() {
    return 3.0;
}
```



Binary patching

Tool list:

- Python



- Capstone



- Keystone



Binary patching

Find the **offset**:

```
.text:000000000015EA60  
● .text:000000000015EA60  
● .text:000000000015EA61  
● .text:000000000015EA64  
● .text:000000000015EA6E  
● .text:000000000015EA73  
● .text:000000000015EA77  
● .text:000000000015EA78
```

```
push    rbp  
mov     rbp, rsp  
mov     rax, 3  
cvttsi2ss xmm0, rax  
mov     [rbp+this], rdi  
pop    rbp  
ret
```



Binary patching

Verify **new instruction** won't **overwrite** the next **legitimate instruction** (in this case `cvttsi2ss xmm0, rax`). For this, we need to **calculate** the **size** it takes in memory of the **instruction** to replace:

```
from capstone import *

offset = 0x15ea64

with open( 'libGameLogic.so', 'rb' ) as f:
    binary = f.read()

cpt = 0
size = 0

md = Cs( CS_ARCH_X86, CS_MODE_64 )
for i in md.disasm( binary[ offset : offset+64 ], offset ):
    if not cpt:
        size = len(i.bytes)
    cpt += 1
```



Binary patching

Then we **assemble** the new **instruction** and **compare** the **sizes**: If it's equals or smaller, we can **overwrite** the **instruction** with padding:

```
from keystone import *

ks = Ks(KS_ARCH_X86, KS_MODE_64)
encoding, count = ks.asm(b'mov rax, 0x1e')

if len( encoding ) <= size:
    # padding with NOPs if shorter
    encoding = encoding + [0x90]*( size - len(encoding) )
    binary = binary[ : offset ] + ''.join(chr(c) for c in
binary) + BIN[ offset+size : ]
```



Binary patching

Finally, we just need to **save the new file**:

```
with open('libGameLogic.so', 'wb') as f:  
    f.write(binary)
```



Binary patching

Complete **patcher** script:

<https://github.com/beaujeant/PwnAdventure3/blob/master/Binary/binpatcher.py>

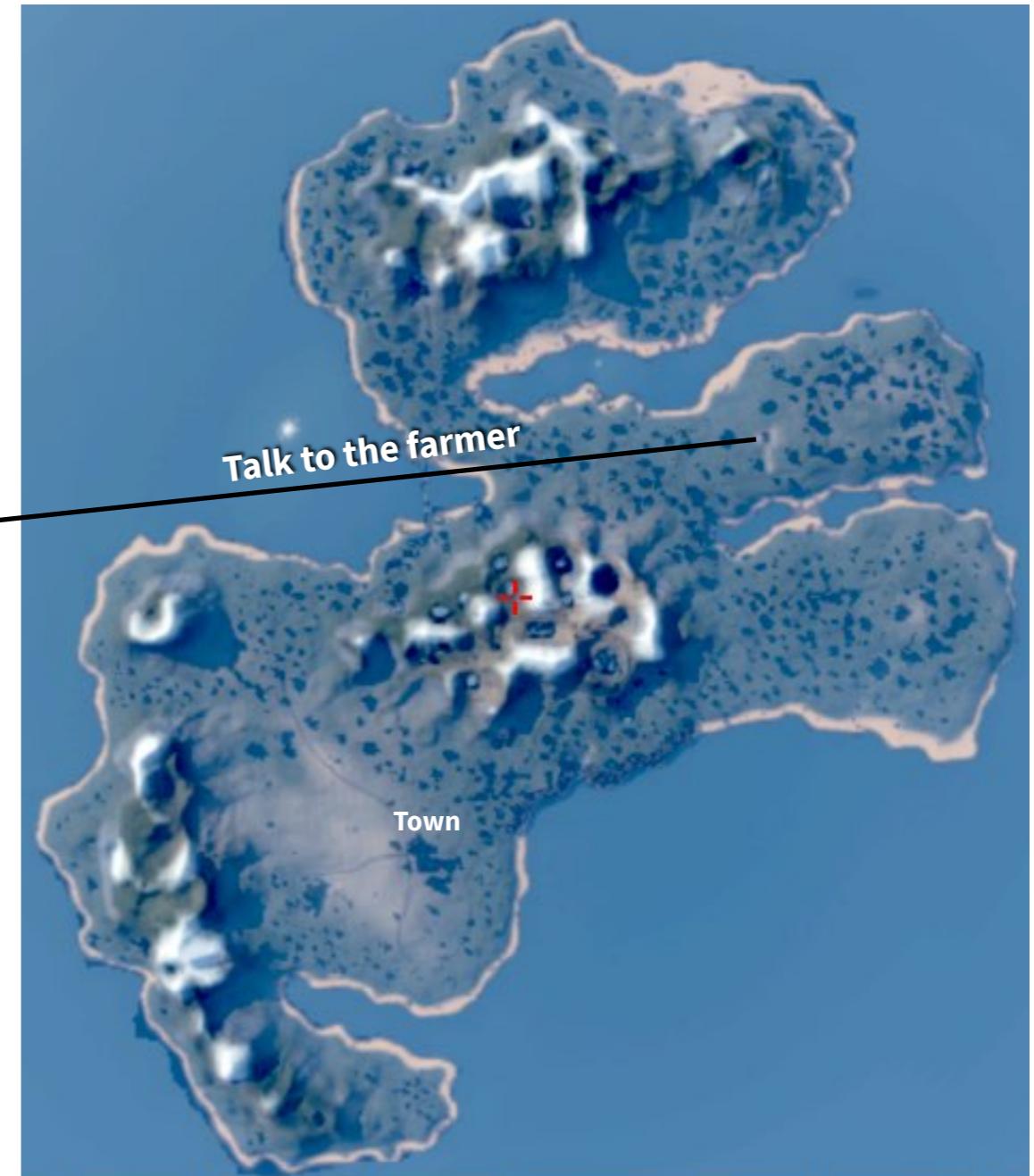
```
$ python binpatcher.py -f libGameLogic.so -o 0x15ea64  
-m 64 -i "movabs rax, 0x1e"
```



Until the Cows Come Home

List of quests:

- *Pirates Treasure (crack me)*
- Fire and Ice (RE binary)
- Until the Cows Come Home (binary patching)
- Egg Hunter (RE network & RE binary)
- Unbearable Revenge (RE network)
- *Blockys Revenge (logic gate)*
- *Overachiever (finish all achievements)*



Until the Cows Come Home

Demo: Follow the sun



Egg Hunter



Egg hunter

Look for the *Ballmer Peak Egg* related function:

Function name	Segment	Start	Length	Locals	Arguments	R	F	L	S	B	T	=
f BallmerPeakEgg::~BallmerPeakEgg()	.plt	0000000000111060	00000006	00000000	00000000	R	T	.
f ActorFactory<BallmerPeakEgg>::ActorFactory(void)	.plt	0000000000113AF0	00000006	00000000	00000000	R	T	.
f BallmerPeakEgg::~BallmerPeakEgg()	.plt	00000000001158B0	00000006	00000000	00000000	R	T	.
f BallmerPeakEgg::BallmerPeakEgg(void)	.plt	0000000000115BB0	00000006	00000000	00000000	R	T	.
f BallmerPeakEgg::BallmerPeakEgg(void)	.plt	000000000011DAB0	00000006	00000000	00000000	R	T	.
f ActorFactory<BallmerPeakEgg>::ActorFactory(void)	.plt	0000000000125A80	00000006	00000000	00000000	R	T	.
f BallmerPeakEgg::BallmerPeakEgg(void)	.text	000000000018E3A0	0000001B	00000018	00000000	R	.	.	S	B	T	.
f BallmerPeakEgg::BallmerPeakEgg(void)	.text	000000000019EA00	000000F6	00000078	00000000	R	.	.	S	B	T	.
f ActorFactory<BallmerPeakEgg>::ActorFactory(void)	.text	000000000019FAC0	0000001B	00000018	00000000	R	.	.	S	B	T	.
f ActorFactory<BallmerPeakEgg>::ActorFactory(void)	.text	00000000001A5E10	00000039	00000018	00000000	R	.	.	S	B	T	.
f ActorFactory<BallmerPeakEgg>::CreateActor(void)	---	000000001A5E70	00000028	00000038	00000000	R	.	.	S	B	T	.
f BallmerPeakEgg::CanUse(IPlayer *)	---	000000001A5E70	00000028	00000000	00000000	R	.	.	S	B	T	.
f BallmerPeakEgg::~BallmerPeakEgg()	---	000000000020C400	00000028	00000018	00000000	R	.	.	S	B	T	.
f BallmerPeakEgg::~BallmerPeakEgg()	.text	000000000020C430	0000001B	00000018	00000000	R	.	.	S	B	T	.

We want to “use” the egg



Egg hunter

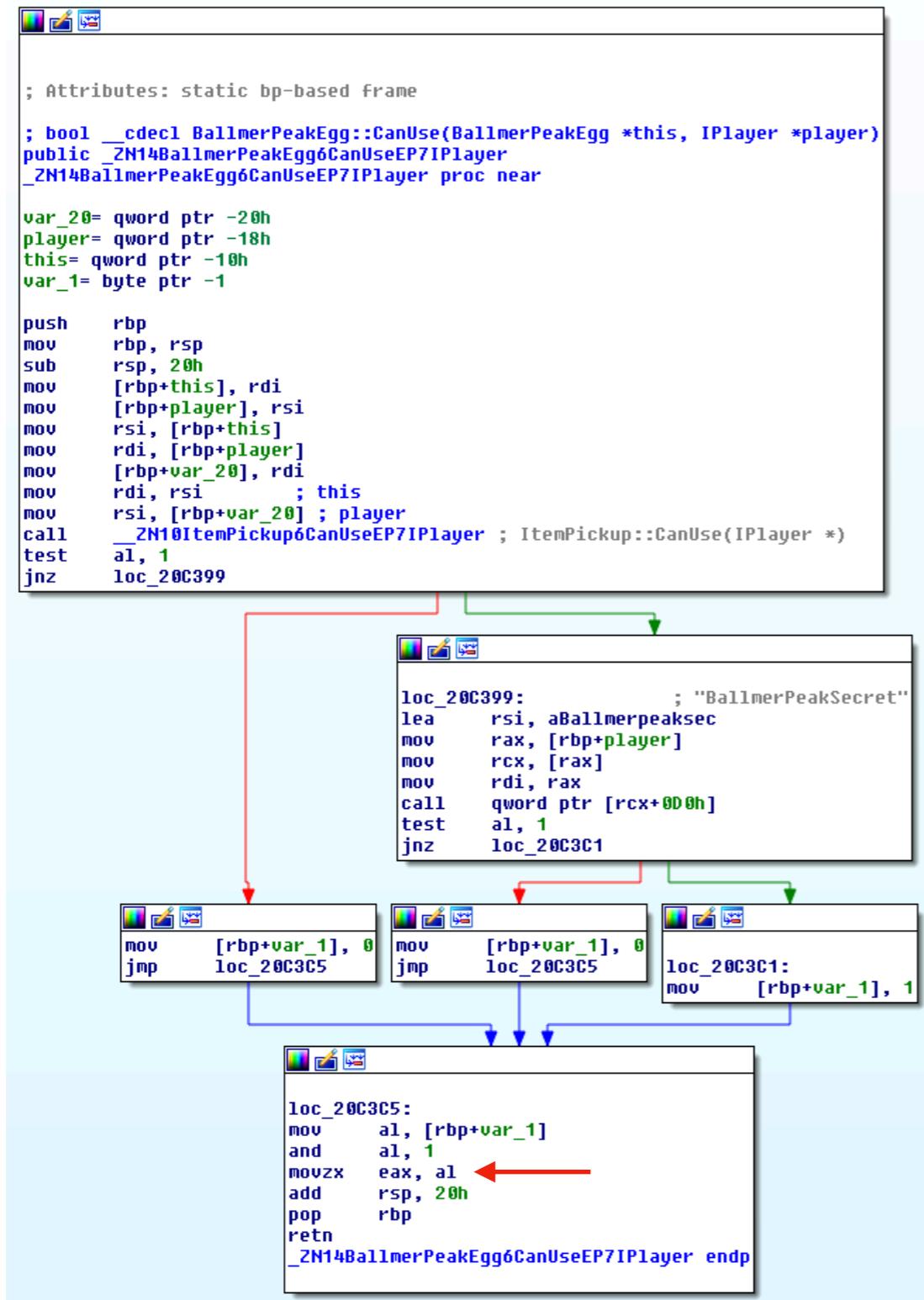


We want this function to return True

`rax (eax)` is usually the return value of the function



Egg hunter



We want this function to return True

`rax (eax)` is usually the return value of the function



Egg hunter



We want this function to return True

`rax (eax)` is usually the return value of the function



Egg hunter

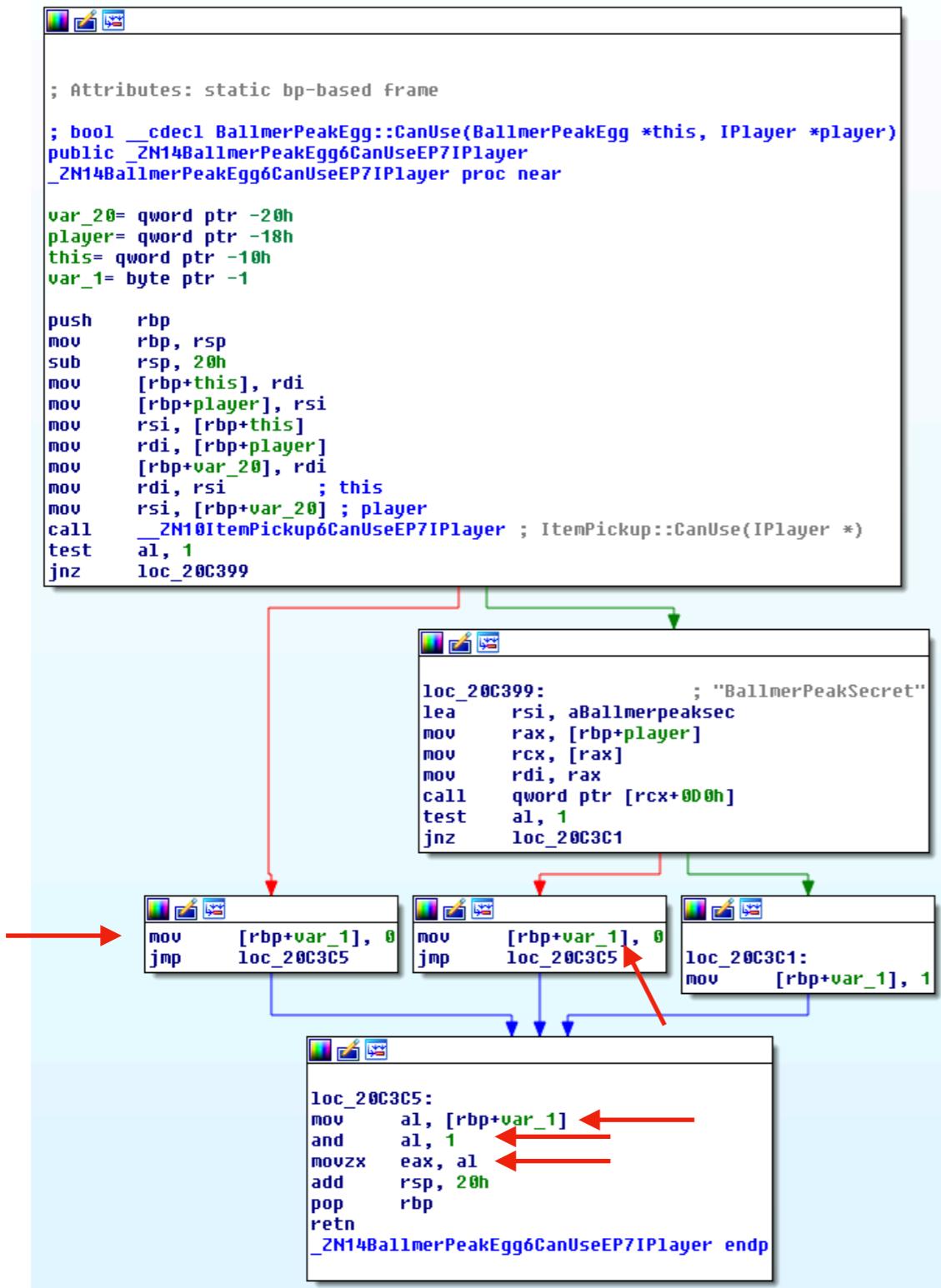


We want this function to return True

`rax (eax)` is usually the return value of the function



Egg hunter

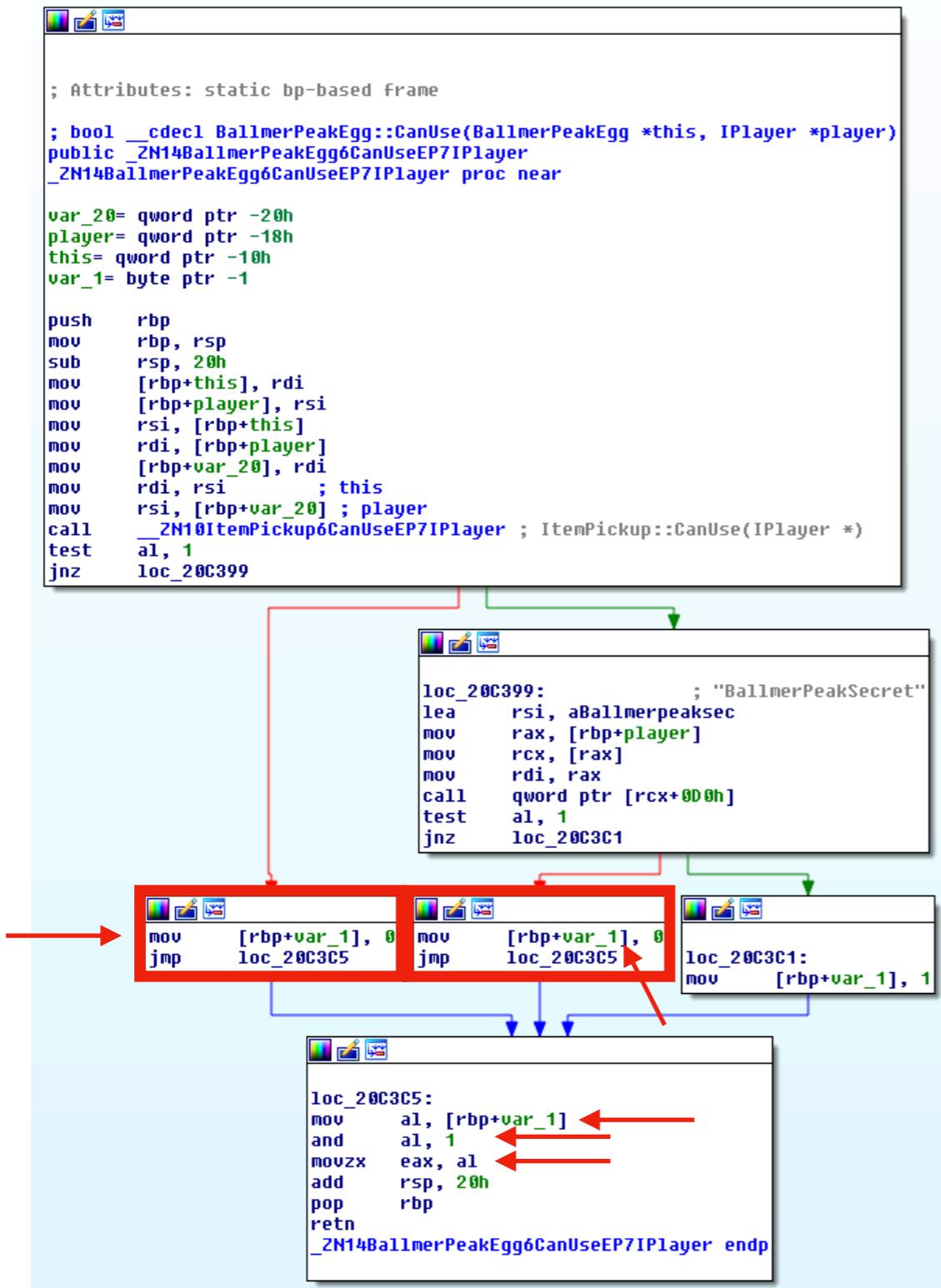


We want this function to return True

`rax (eax)` is usually the return value of the function



Egg hunter

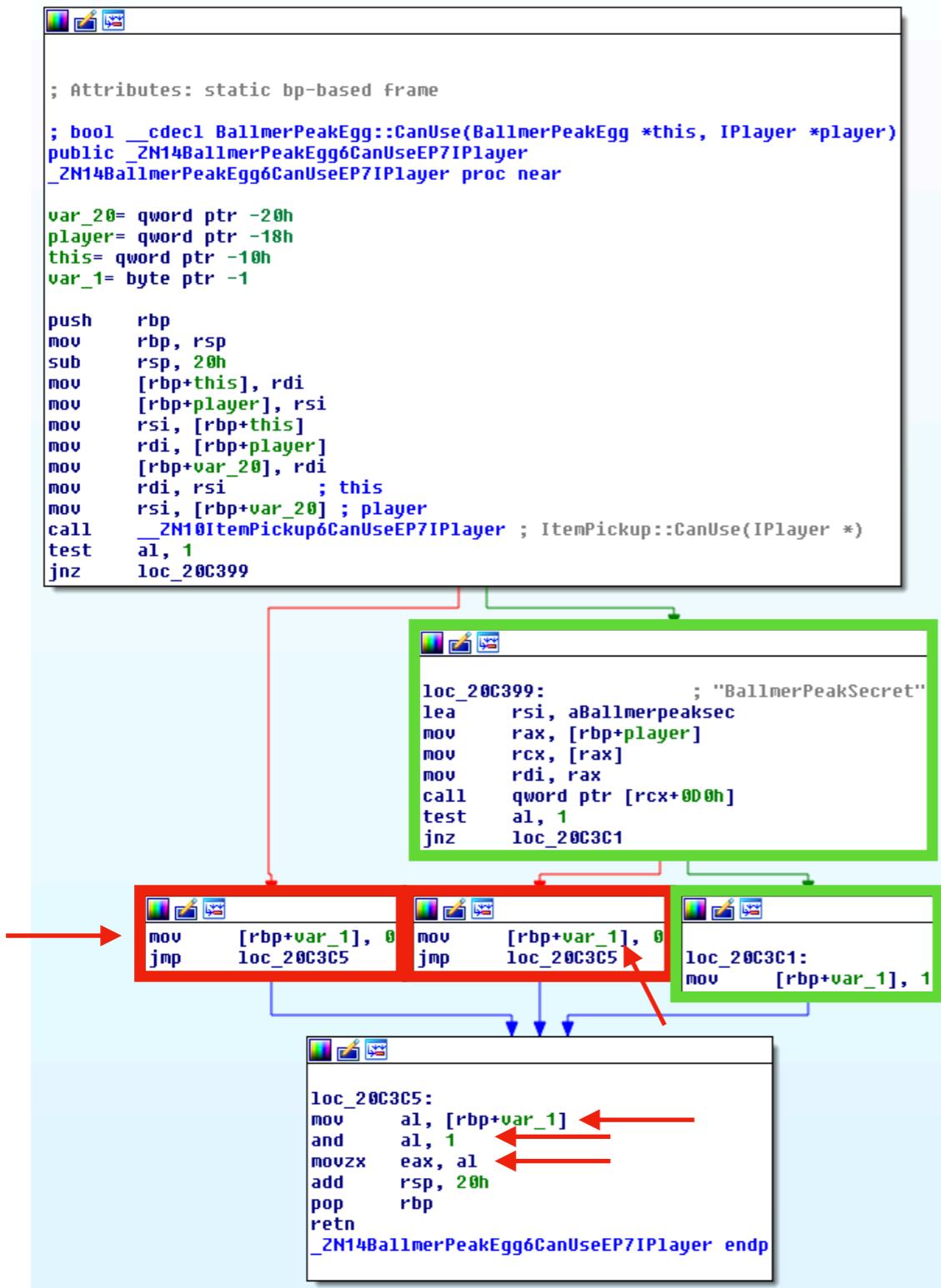


We want this function to return True

`rax (eax)` is usually the return value of the function



Egg hunter



We want this function to return True

`rax (eax)` is usually the return value of the function

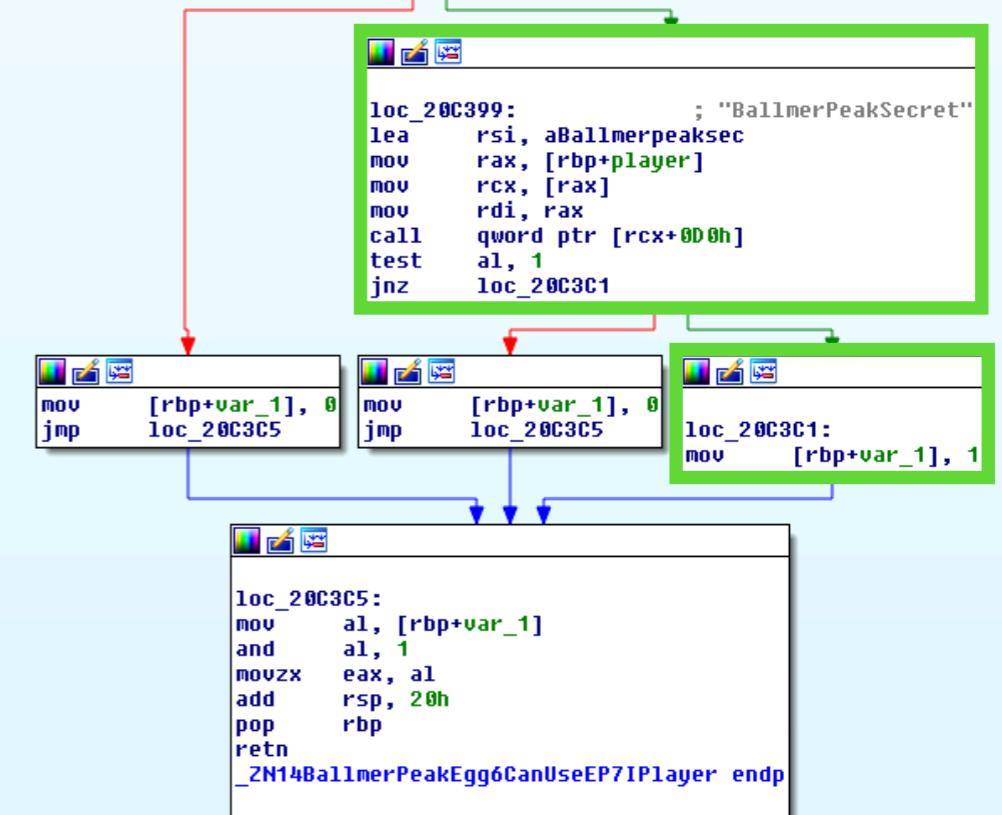


Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
    IPlayer *player) {
```

```
; Attributes: static bp-based frame  
  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push    rbp  
mov     rbp, rsp  
sub    rsp, 20h  
mov     [rbp+this], rdi  
mov     [rbp+player], rsi  
mov     rsi, [rbp+this]  
mov     rdi, [rbp+player]  
mov     [rbp+var_20], rdi  
mov     rdi, rsi        ; this  
mov     rsi, [rbp+var_20] ; player  
call    _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test    al, 1  
jnz    loc_20C399
```

```
}
```



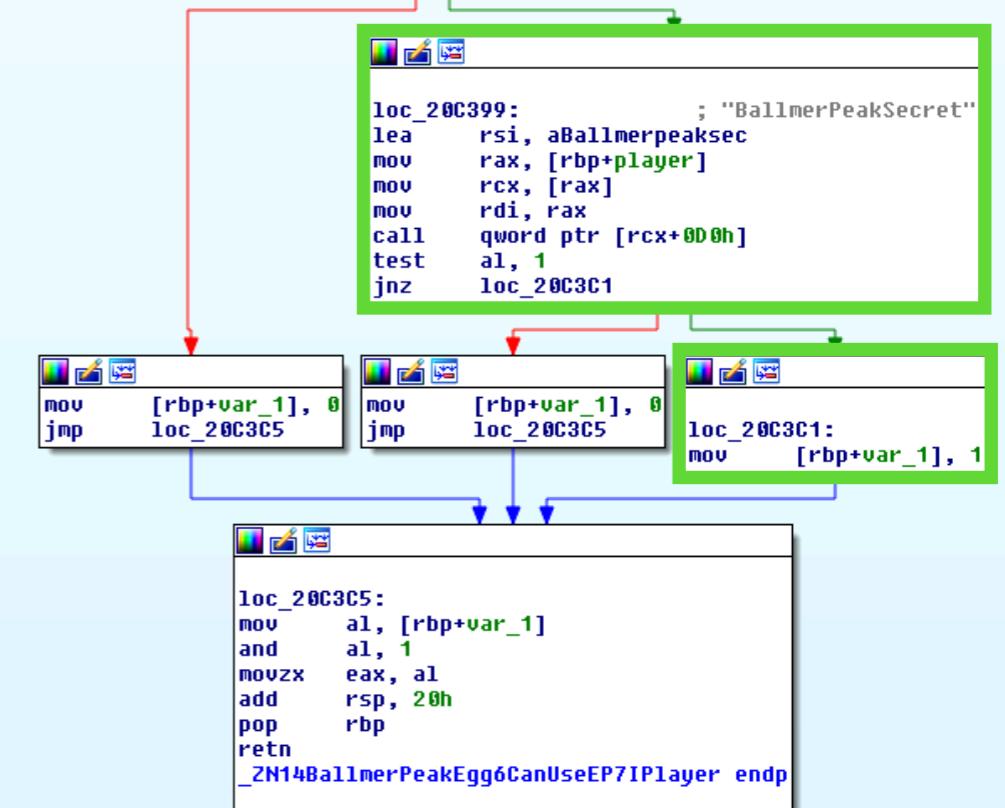
Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
    IPlayer *player) {
```

```
; Attributes: static bp-based frame  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
mov [rbp+this], rdi  
mov [rbp+player], rsi  
mov rsi, [rbp+this]  
mov rdi, [rbp+player]  
mov [rbp+var_20], rdi  
mov rdi, rsi ; this  
mov rsi, [rbp+var_20] ; player  
call _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test al, 1  
jnz loc_20C399
```

Prologue (setting the stack frame)

```
}
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
    IPlayer *player) {
```

```
; Attributes: static bp-based frame  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
mov [rbp+this], rdi  
mov [rbp+player], rsi  
mov rsi, [rbp+this]  
mov rdi, [rbp+player]  
mov [rbp+var_20], rdi  
mov rdi, rsi ; this  
mov rsi, [rbp+var_20] ; player  
call _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test al, 1  
jnz loc_20C399
```

Prologue (setting the stack frame)

```
}
```

```
loc_20C399: ; "BallmerPeakSecret"  
lea rsi, aBallmerpeaksec  
mov rax, [rbp+player]  
mov rcx, [rax]  
mov rdi, rax  
call qword ptr [rcx+000h]  
test al, 1  
jnz loc_20C3C1
```

```
mov [rbp+var_1], 0  
jmp loc_20C3C5
```

```
mov [rbp+var_1], 0  
jmp loc_20C3C5
```

```
loc_20C3C1:  
mov [rbp+var_1], 1
```

```
loc_20C3C5:  
mov al, [rbp+var_1]  
and al, 1  
movzx eax, al  
add rsp, 20h  
pop rbp  
retn  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer endp
```

Epilogue



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {
```

```
    int ret;
```

```
    bool canuse; | + BallmerPeakEgg* and 2 x IPlayer*
```

```
}
```

Call graph diagram illustrating the flow of control from C++ code to assembly and then to several assembly functions.

The C++ code is:

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse; | + BallmerPeakEgg* and 2 x IPlayer*
```

The assembly code is:

```
; Attributes: static bp-based frame  
;  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
mov [rbp+this], rdi  
mov [rbp+player], rsi  
mov rsi, [rbp+this]  
mov rdi, [rbp+player]  
mov [rbp+var_20], rdi  
mov rdi, rsi ; this  
mov rsi, [rbp+var_20] ; player  
call _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test al, 1  
jnz loc_20C399
```

Function loc_20C399:

```
loc_20C399: ; "BallmerPeakSecret"  
lea rsi, aBallmerpeaksec  
mov rax, [rbp+player]  
mov rcx, [rax]  
mov rdi, rax  
call qword ptr [rcx+0000h]  
test al, 1  
jnz loc_20C3C1
```

Function loc_20C3C5:

```
loc_20C3C5:  
mov [rbp+var_1], 0  
jmp loc_20C3C5
```

Function loc_20C3C5:

```
loc_20C3C5:  
mov [rbp+var_1], 0  
jmp loc_20C3C5
```

Function loc_20C3C1:

```
loc_20C3C1:  
mov [rbp+var_1], 1
```

Function loc_20C3C5:

```
loc_20C3C5:  
mov al, [rbp+var_1]  
and al, 1  
movzx eax, al  
add rsp, 20h  
pop rbp  
retn  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer endp
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {
```

```
    int ret;  
    bool canuse;
```

```
}
```

```
; Attributes: static bp-based frame  
;  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
  
mov [rbp+this], rdi  
mov [rbp+player], rsi  
mov rsi, [rbp+this]  
mov rdi, [rbp+player]  
mov [rbp+var_20], rdi  
mov rdi, rsi ; this  
mov rsi, [rbp+var_20] ; player  
call _ZN14ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test al, 1  
jnz loc_20C399
```

```
loc_20C399: ; "BallmerPeakSecret"  
lea rsi, aBallmerpeaksec  
mov rax, [rbp+player]  
mov rcx, [rax]  
mov rdi, rax  
call qword ptr [rcx+000h]  
test al, 1  
jnz loc_20C3C1
```

```
mov [rbp+var_1], 0  
jmp loc_20C3C5
```

```
mov [rbp+var_1], 0  
jmp loc_20C3C5
```

```
loc_20C3C1:  
mov [rbp+var_1], 1
```

```
loc_20C3C5:  
mov al, [rbp+var_1]  
and al, 1  
movzx eax, al  
add rsp, 20h  
pop rbp  
retn  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer endp
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
    IPlayer *player) {
```

```
    int ret;  
    bool canuse;
```

```
}
```

```
; Attributes: static bp-based frame  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push    rbp  
mov     rbp, rsp  
sub    rsp, 20h  
  
mov     [rbp+this], rdi  
mov     [rbp+player], rsi  
mov     rsi, [rbp+this]  
mov     rdi, [rbp+player]  
mov     [rbp+var_20], rdi  
mov     rdi, rsi      ; this  
mov     rsi, [rbp+var_20] ; player  
call    _ZN14ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test    al, 1  
jnz    loc_20C399
```

rdi = rdi
rsi = rsi

```
loc_20C399: ; "BallmerPeakSecret"  
lea     rsi, aBallmerpeaksec  
mov     rax, [rbp+player]  
mov     rcx, [rax]  
mov     rdi, rax  
call    qword ptr [rcx+000h]  
test    al, 1  
jnz    loc_20C3C1
```

```
mov     [rbp+var_1], 0  
jmp    loc_20C3C5
```

```
mov     [rbp+var_1], 0  
jmp    loc_20C3C5
```

```
loc_20C3C1:  
    mov     [rbp+var_1], 1
```

```
loc_20C3C5:  
    mov     al, [rbp+var_1]  
    and    al, 1  
    movzx  eax, al  
    add    rsp, 20h  
    pop    rbp  
    retn  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer endp
```

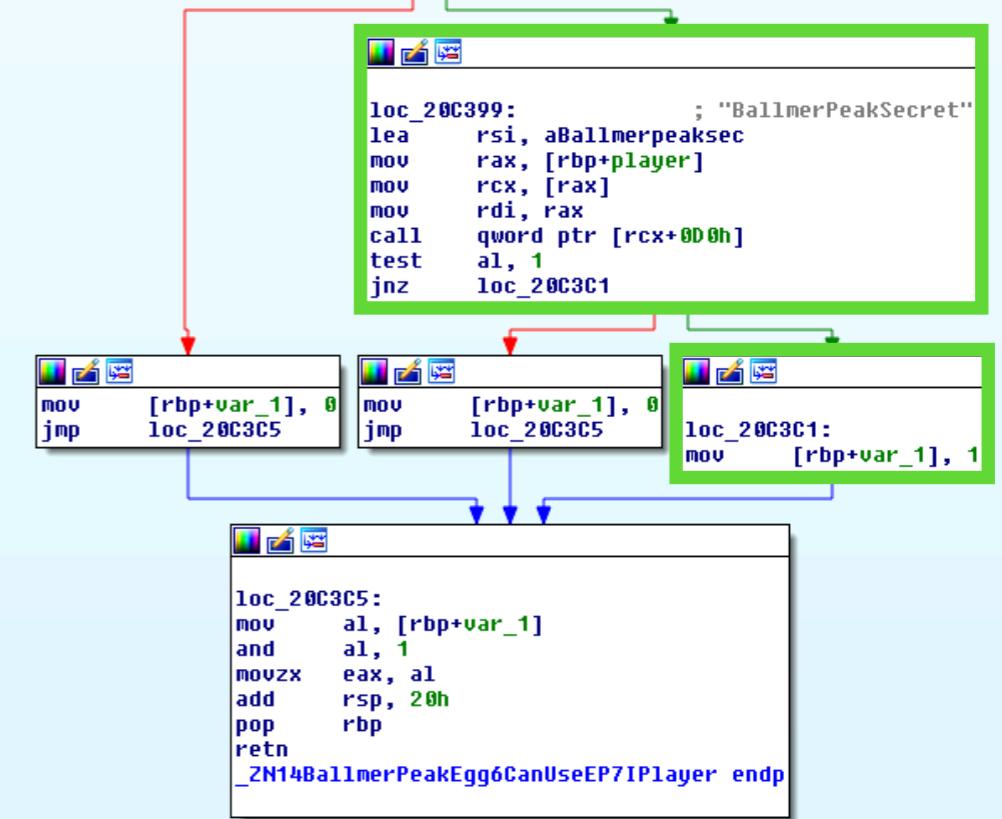


Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(          ,        );  
}  
}
```

rdi = rdi
rsi = rsi

```
; Attributes: static bp-based frame  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push   rbp  
mov    rbp, rsp  
sub    rsp, 20h  
mov    [rbp+this], rdi  
mov    [rbp+player], rsi  
mov    rsi, [rbp+this]  
mov    rdi, [rbp+player]  
mov    [rbp+var_20], rdi  
mov    rdi, rsi      ; this  
mov    rsi, [rbp+var_20] ; player  
call   _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test   al, 1  
jnz   loc_20C399
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, );  
}
```

The diagram illustrates the flow of control from the C++ source code to the generated assembly code and back. A yellow arrow points from the highlighted line in the C++ code to the corresponding assembly instruction in the first window. The assembly code shows the parameters being passed: `this` is `rdi`, and `player` is `rsi`. The assembly code is:

```
; Attributes: static bp-based frame  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
mov [rbp+this], rdi  
mov [rbp+player], rsi  
mov rsi, [rbp+this]  
mov rdi, [rbp+player]  
mov [rbp+var_20], rdi  
mov rdi, rsi ; this  
mov rsi, [rbp+var_20] ; player  
call _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test al, 1  
jnz loc_20C399
```

rdi = rdi
rsi = rsi

loc_20C399: ; "BallmerPeakSecret"
lea rsi, aBallmerpeaksec
mov rax, [rbp+player]
mov rcx, [rax]
mov rdi, rax
call qword ptr [rcx+0000h]
test al, 1
jnz loc_20C3C1

loc_20C3C5:
mov [rbp+var_1], 0
jmp loc_20C3C5

loc_20C3C5:
mov [rbp+var_1], 0
jmp loc_20C3C5

loc_20C3C1:
mov [rbp+var_1], 1

loc_20C3C5:
mov al, [rbp+var_1]
and al, 1
movzx eax, al
add rsp, 20h
pop rbp
retn
_ZN14BallmerPeakEgg6CanUseEP7IPlayer endp

}



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
}
```

The diagram illustrates the flow of control from the C++ source code to the generated assembly code and back. A yellow arrow points from the highlighted line in the C++ code to the corresponding assembly instruction in the first window. The assembly code shows the parameters being passed: `this` is `rdi`, and `player` is `rsi`. The assembly code then calls `ItemPickup::CanUse` at `loc_20C399`.

```
; Attributes: static bp-based frame  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
mov [rbp+this], rdi  
mov [rbp+player], rsi  
mov rsi, [rbp+this]  
mov rdi, [rbp+player]  
mov [rbp+var_20], rdi  
mov rdi, rsi ; this  
rsi, [rbp+var_20] ; player  
__ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
al, 1  
jnz loc_20C399
```

rdi = rdi
rsi = rsi

The assembly code at `loc_20C399` calls `BallmerPeakSecret` at `loc_20C3C1`. This function checks a condition and jumps to `loc_20C3C5` if it fails.

```
loc_20C399: ; "BallmerPeakSecret"  
lea rsi, aBallmerpeaksec  
mov rax, [rbp+player]  
mov rcx, [rax]  
mov rdi, rax  
call qword ptr [rcx+0000h]  
test al, 1  
jnz loc_20C3C1
```

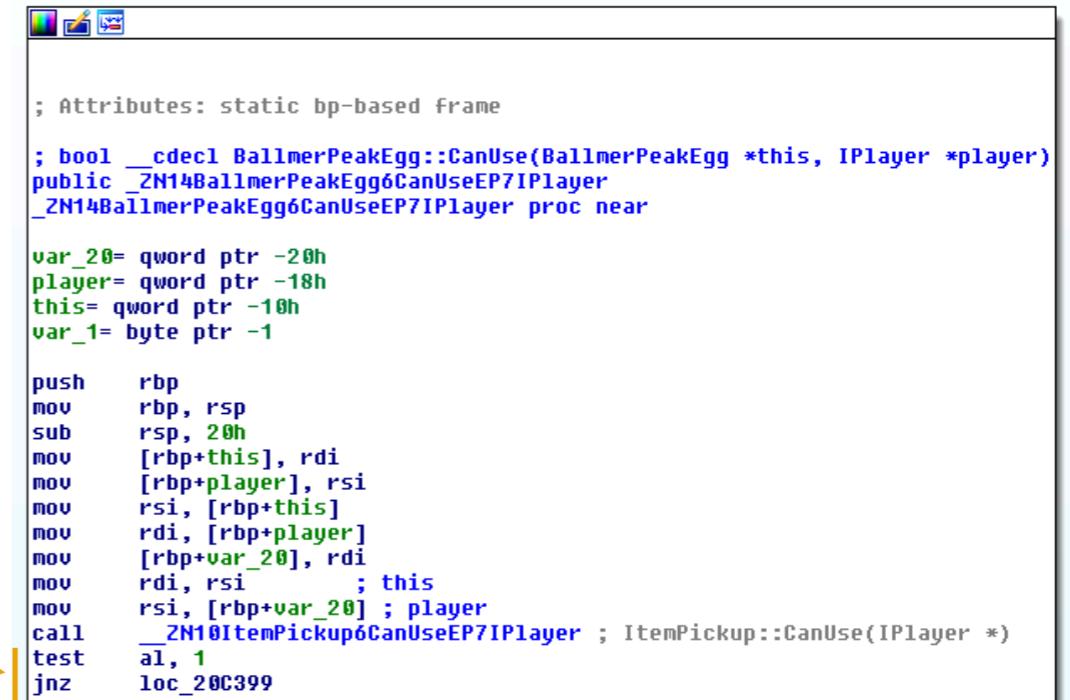
The assembly code at `loc_20C3C5` moves the value of `var_1` to `rsp` and then returns to the caller.

```
loc_20C3C5:  
mov al, [rbp+var_1]  
and al, 1  
movzx eax, al  
add rsp, 20h  
pop rbp  
retn  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer endp
```

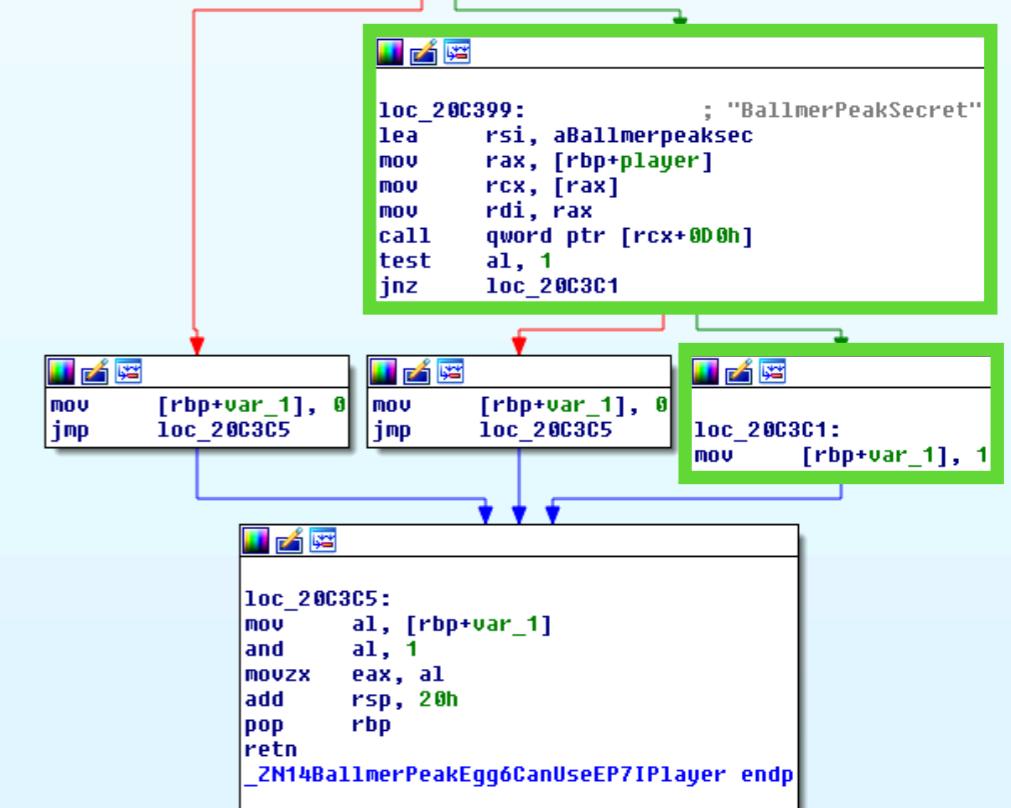


Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
  
    }  
    else  
    {  
    }  
}
```

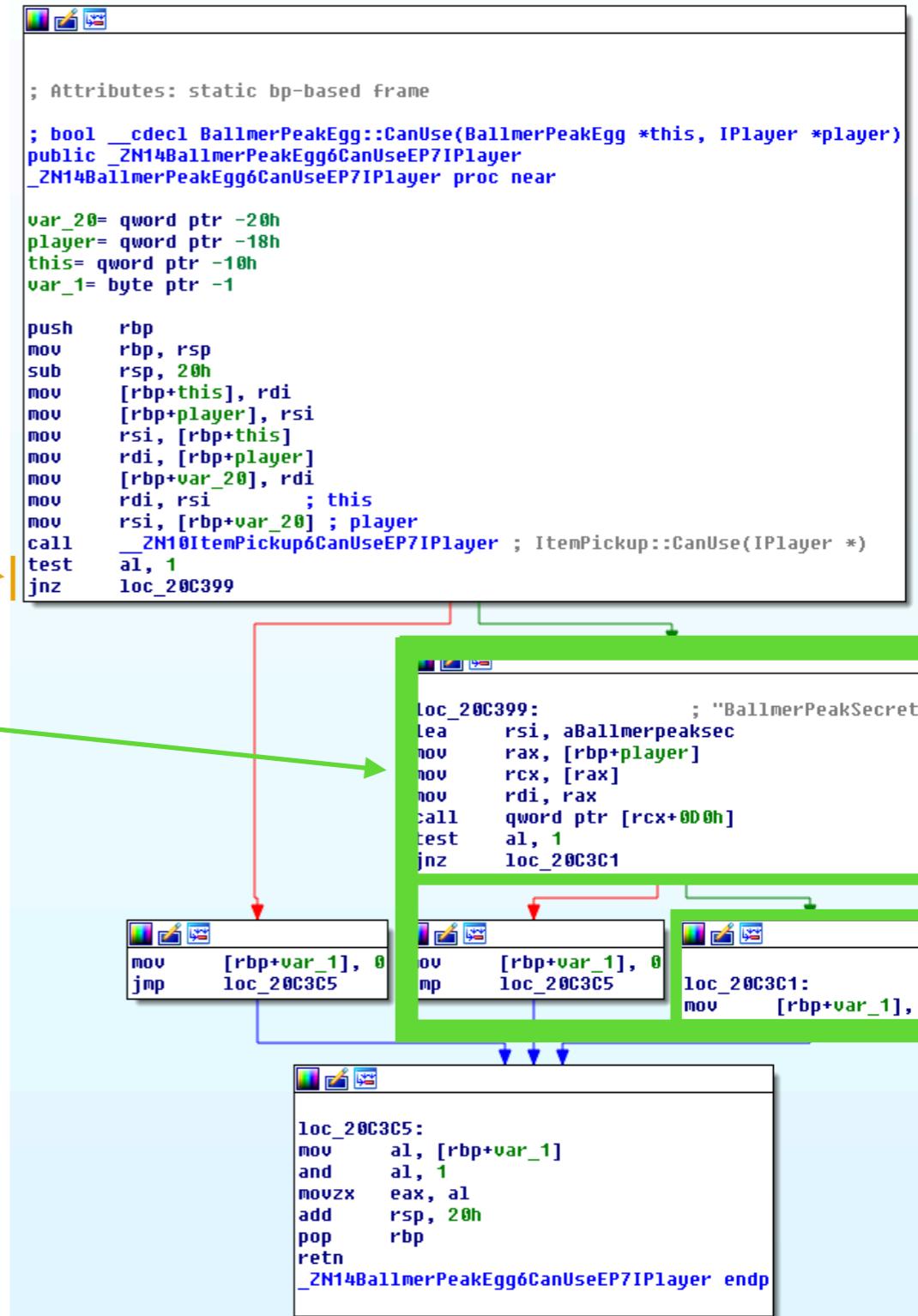


```
; Attributes: static bp-based frame  
;  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
mov [rbp+this], rdi  
mov [rbp+player], rsi  
mov rsi, [rbp+this]  
mov rdi, [rbp+player]  
mov [rbp+var_20], rdi  
mov rdi, rsi ; this  
mov rsi, [rbp+var_20] ; player  
call _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test al, 1  
jnz loc_20C399
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        // Green box highlights the if block  
    }  
    else  
    {  
        // Green box highlights the else block  
    }  
}
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        // Green box  
    }  
    else  
    {  
        // Red box  
    }  
}
```

```
; Attributes: static bp-based frame  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push   rbp  
mov    rbp, rsp  
sub    rsp, 20h  
mov    [rbp+this], rdi  
mov    [rbp+player], rsi  
mov    rsi, [rbp+this]  
mov    rdi, [rbp+player]  
mov    [rbp+var_20], rdi  
mov    rdi, rsi      ; this  
mov    rsi, [rbp+var_20] ; player  
call   _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test   al, 1  
jnz    loc_20C399
```

```
loc_20C399: ; "BallmerPeakSecret"  
lea    rsi, aBallmerpeaksec  
mov    rax, [rbp+player]  
mov    rcx, [rax]  
mov    rdi, rax  
call   qword ptr [rcx+000h]  
test   al, 1  
jnz    loc_20C3C1
```

```
loc_20C3C5:  
mov    [rbp+var_1], 0  
jmp    loc_20C3C5
```

```
loc_20C3C1:  
mov    [rbp+var_1], 1
```

```
loc_20C3C5:  
mov    al, [rbp+var_1]  
and    al, 1  
movzx eax, al  
add    rsp, 20h  
pop    rbp  
retn  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer endp
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        ret = IPlayer::????( , );  
    }  
    else  
    {  
    }  
}
```

```
; Attributes: static bp-based frame  
;  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
mov [rbp+this], rdi  
mov [rbp+player], rsi  
mov rsi, [rbp+this]  
mov rdi, [rbp+player]  
mov [rbp+var_20], rdi  
mov rdi, rsi ; this  
mov rsi, [rbp+var_20] ; player  
call _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test al, 1  
jnz loc_20C399
```

```
loc_20C399: ; "BallmerPeakSecret"  
lea rsi, aBallmerpeaksec  
mov rax, [rbp+player]  
mov rcx, [rax]  
mov rdi, rax  
call qword ptr [rcx+000h]  
test al, 1  
jnz loc_20C3C1
```

```
loc_20C3C5:  
mov [rbp+var_1], 0  
jmp loc_20C3C5
```

```
loc_20C3C5:  
mov [rbp+var_1], 0  
jmp loc_20C3C5
```

```
loc_20C3C1:  
mov [rbp+var_1], 1
```

```
loc_20C3C5:  
mov al, [rbp+var_1]  
and al, 1  
movzx eax, al  
add rsp, 20h  
pop rbp  
retn  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer endp
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        ret = IPlayer::????(player,  
                           );  
  
    }  
    else  
    {  
    }  
}
```

Call graph diagram illustrating the flow of control from C++ code to assembly and back. The C++ code calls `IPlayer::????`, which branches to two assembly paths based on the value of `var_1`. The first path leads to `loc_20C399`, which then branches to either `loc_20C3C1` or `loc_20C3C5`. The second path leads to `loc_20C3C5`. The assembly code includes comments like `; Attributes: static bp-based frame` and `; "BallmerPeakSecret"`.

```
; Attributes: static bp-based frame  
  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
mov [rbp+this], rdi  
mov [rbp+player], rsi  
mov rsi, [rbp+this]  
mov rdi, [rbp+player]  
mov [rbp+var_20], rdi  
mov rdi, rsi ; this  
mov rsi, [rbp+var_20] ; player  
call _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test al, 1  
jnz loc_20C399
```

The assembly flow continues from `loc_20C399` to either `loc_20C3C1` or `loc_20C3C5`. Both paths eventually converge at `loc_20C3C5`, which then leads to the end of the function at `_ZN14BallmerPeakEgg6CanUseEP7IPlayer endp`.

```
loc_20C399: ; "BallmerPeakSecret"  
lea rsi, aBallmerpeaksec  
mov rax, [rbp+player]  
mov rcx, [rax]  
mov rdi, rax  
call qword ptr [rcx+0000h]  
test al, 1  
jnz loc_20C3C1  
  
loc_20C3C1:  
mov [rbp+var_1], 0  
jmp loc_20C3C5  
  
loc_20C3C5:  
mov al, [rbp+var_1]  
and al, 1  
movzx eax, al  
add rsp, 20h  
pop rbp  
retn  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer endp
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        ret = IPlayer::????(player, "BallmerPeakSecret");  
    }  
    else  
    {  
    }  
}
```

Call graph diagram illustrating the flow of control from the C++ source code to the generated assembly code and back. The main function is BallmerPeakEgg::CanUse. It calls ItemPickup::CanUse, which returns true. Then it calls IPlayer::????, passing the player pointer and the string "BallmerPeakSecret". This leads to the assembly code at loc_20C399:

```
; Attributes: static bp-based frame  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
mov [rbp+this], rdi  
mov [rbp+player], rsi  
mov rsi, [rbp+this]  
mov rdi, [rbp+player]  
mov [rbp+var_20], rdi  
mov rdi, rsi ; this  
mov rsi, [rbp+var_20] ; player  
call _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test al, 1  
jnz loc_20C399
```

The assembly at loc_20C399 checks if the result is 1. If true, it branches to loc_20C3C1. If false, it branches to loc_20C3C5. Both paths lead to the same assembly code at loc_20C3C1:

```
loc_20C399: ; "BallmerPeakSecret"  
lea rsi, aBallmerpeaksec  
mov rax, [rbp+player]  
mov rcx, [rax]  
mov rdi, rax  
call qword ptr [rcx+0000h]  
test al, 1  
jnz loc_20C3C1
```

From loc_20C3C1, the assembly branches to loc_20C3C5 or loc_20C3C1 again. Both loc_20C3C5 and loc_20C3C1 lead to the final assembly code at loc_20C3C5:

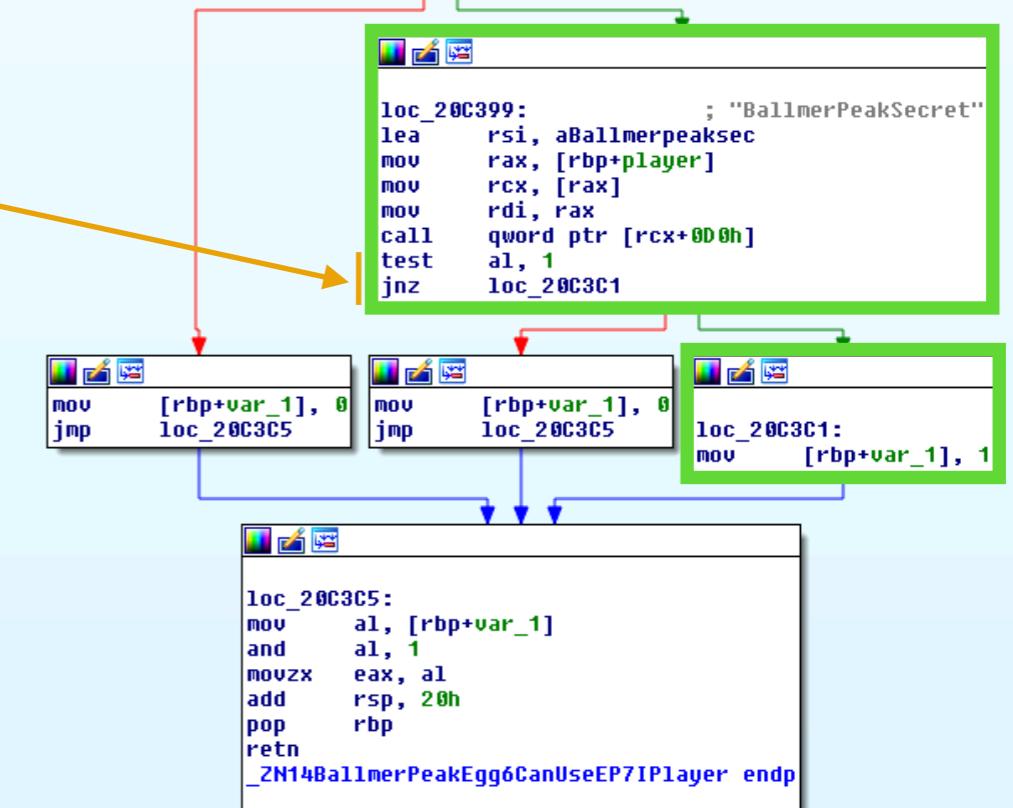
```
loc_20C3C5:  
mov al, [rbp+var_1]  
and al, 1  
movzx eax, al  
add rsp, 20h  
pop rbp  
retn  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer endp
```



Egg hunter

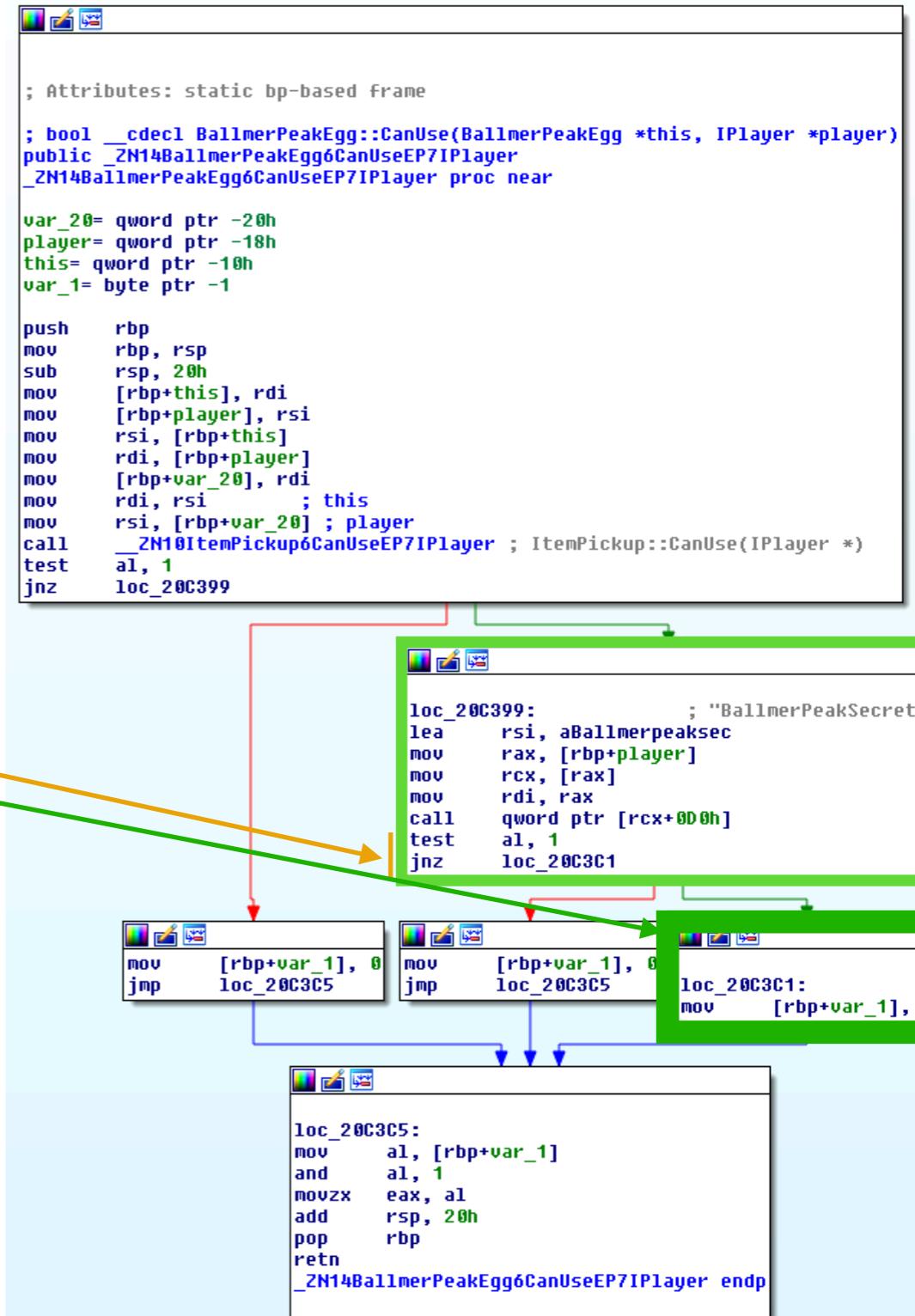
```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        ret = IPlayer::????(player, "BallmerPeakSecret");  
        if (ret)  
        {  
        }  
        else  
        {  
        }  
    }  
    else  
    {  
    }  
}
```

```
; Attributes: static bp-based frame  
;  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push   rbp  
mov    rbp, rsp  
sub    rsp, 20h  
mov    [rbp+this], rdi  
mov    [rbp+player], rsi  
mov    rsi, [rbp+this]  
mov    rdi, [rbp+player]  
mov    [rbp+var_20], rdi  
mov    rdi, rsi      ; this  
mov    rsi, [rbp+var_20] ; player  
call   _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test   al, 1  
jnz    loc_20C399
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        ret = IPlayer::????(player, "BallmerPeakSecret");  
        if (ret)  
        {  
            // [Redacted]  
        }  
        else  
        {  
        }  
    }  
    else  
    {  
        // [Redacted]  
    }  
}
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        ret = IPlayer::????(player, "BallmerPeakSecret");  
        if (ret)  
        {  
            // Green box  
        }  
        else  
        {  
            // Red box  
        }  
    }  
    else  
    {  
        // White box  
    }  
}
```

```
; Attributes: static bp-based frame  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push   rbp  
mov    rbp, rsp  
sub    rsp, 20h  
mov    [rbp+this], rdi  
mov    [rbp+player], rsi  
mov    rsi, [rbp+this]  
mov    rdi, [rbp+player]  
mov    [rbp+var_20], rdi  
mov    rdi, rsi      ; this  
mov    rsi, [rbp+var_20] ; player  
call   _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test   al, 1  
jnz    loc_20C399
```

```
loc_20C399: ; "BallmerPeakSecret"  
lea    rsi, aBallmerpeaksec  
mov    rax, [rbp+player]  
mov    rcx, [rax]  
mov    rdi, rax  
call   qword ptr [rcx+0000h]  
test   al, 1  
jnz    loc_20C3C1
```

```
loc_20C3C5:  
mov    [rbp+var_1], rax  
jmp    loc_20C3C5
```

```
loc_20C3C1:  
mov    [rbp+var_1], rax
```

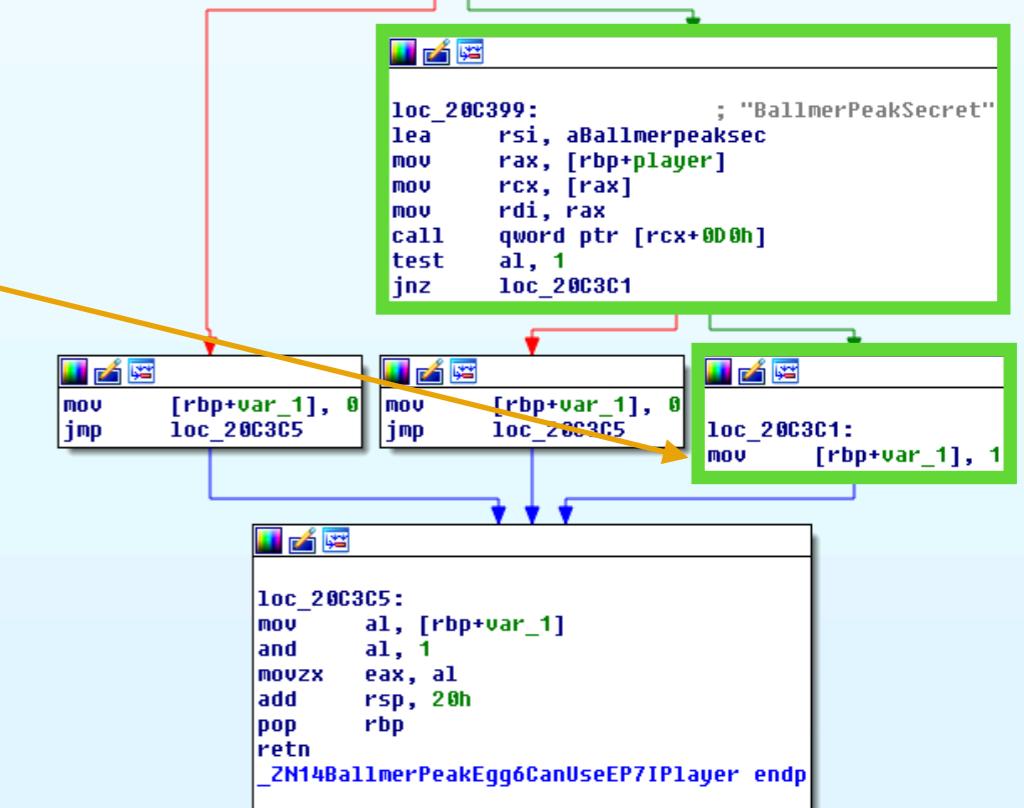
```
loc_20C3C5:  
mov    al, [rbp+var_1]  
and    al, 1  
movzx eax, al  
add    rsp, 20h  
pop    rbp  
retn  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer endp
```



Egg hunter

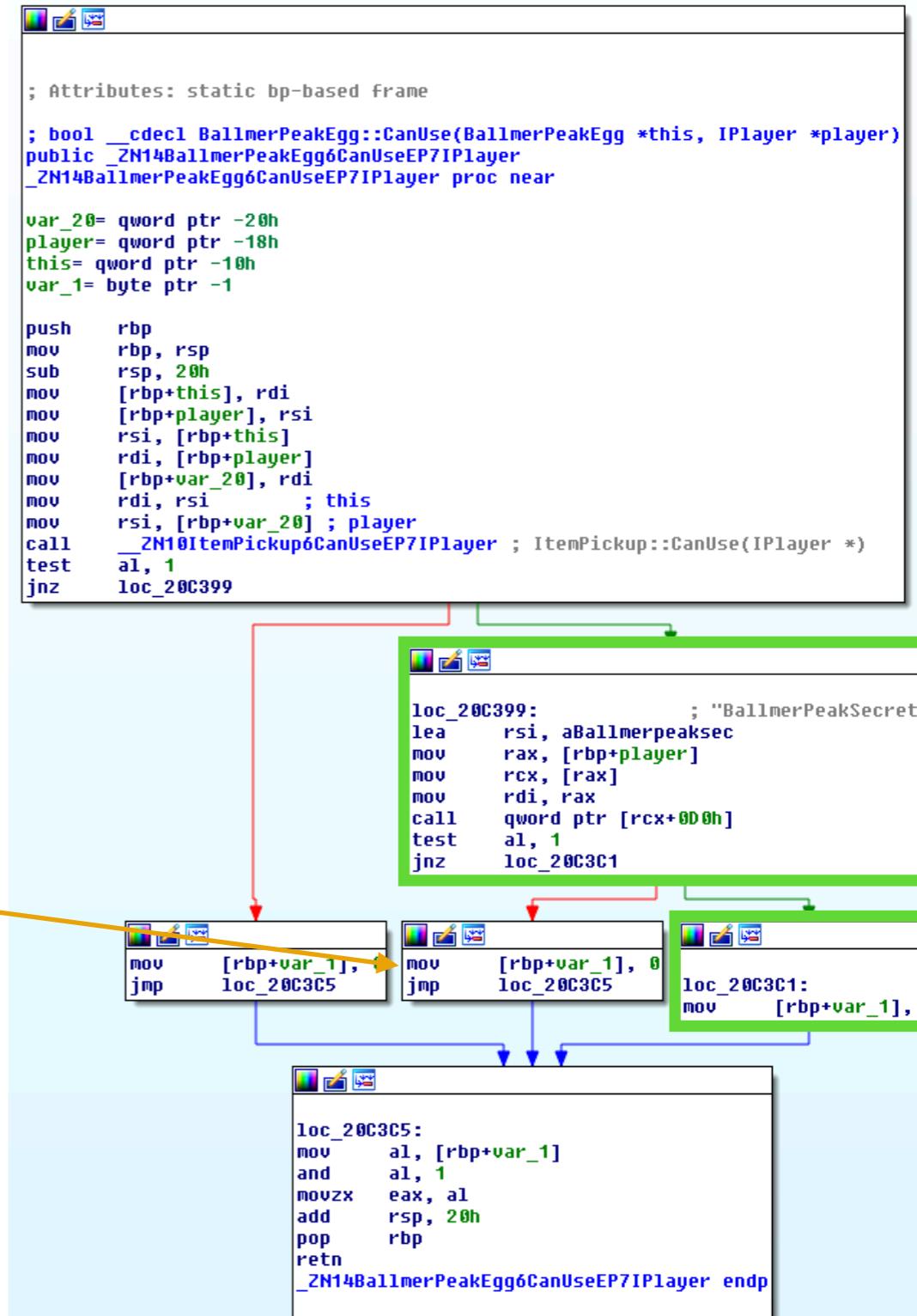
```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        ret = IPlayer::????(player, "BallmerPeakSecret");  
        if (ret)  
        {  
            canuse = true;  
        }  
        else  
        {  
        }  
    }  
    else  
    {  
    }  
}
```

```
; Attributes: static bp-based frame  
  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push   rbp  
mov    rbp, rsp  
sub    rsp, 20h  
mov    [rbp+this], rdi  
mov    [rbp+player], rsi  
mov    rsi, [rbp+this]  
mov    rdi, [rbp+player]  
mov    [rbp+var_20], rdi  
mov    rdi, rsi      ; this  
mov    rsi, [rbp+var_20] ; player  
call   _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test   al, 1  
jnz   loc_20C399
```



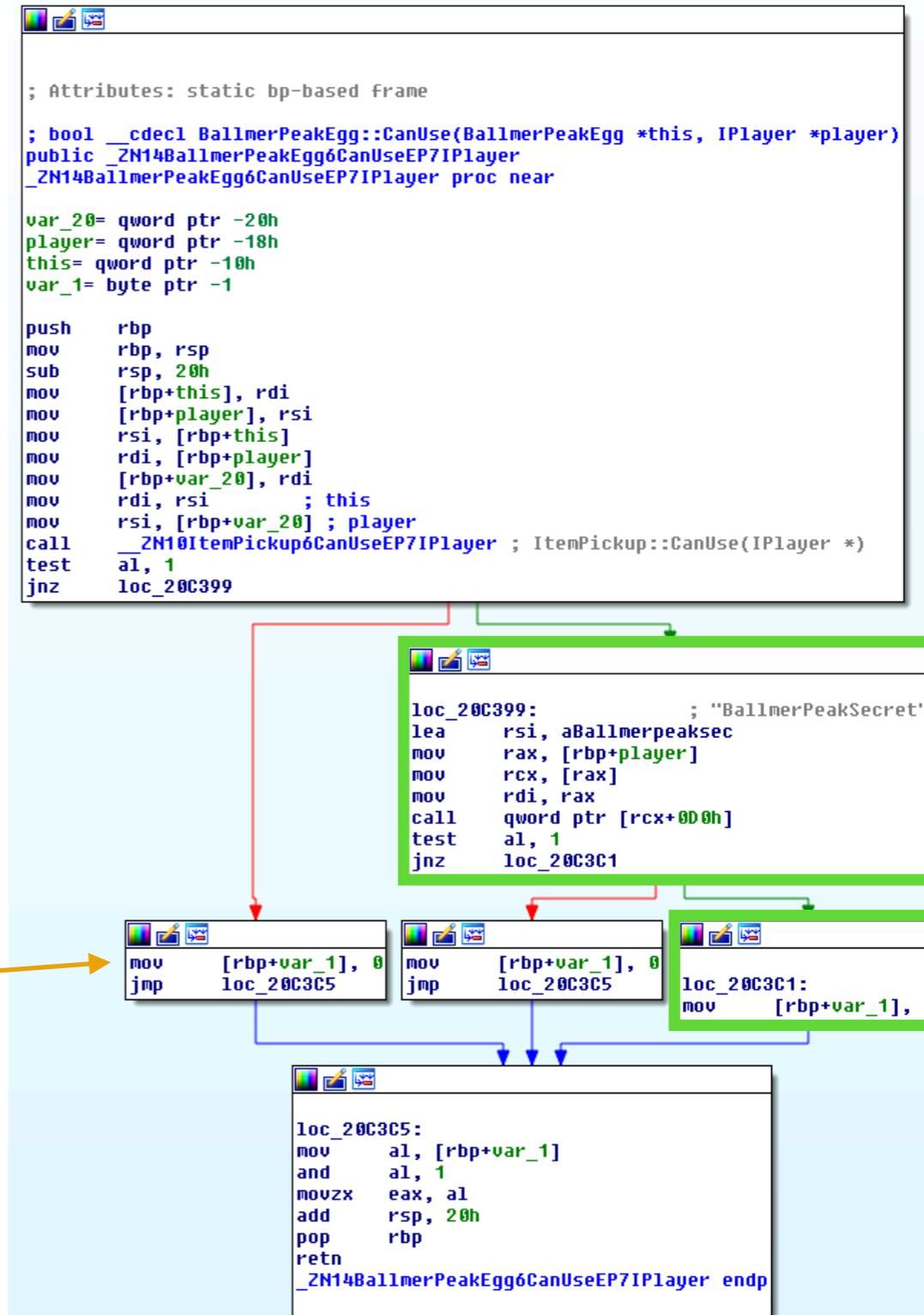
Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        ret = IPlayer::????(player, "BallmerPeakSecret");  
        if (ret)  
        {  
            canuse = true;  
        }  
        else  
        {  
            canuse = false;  
        }  
    }  
    else  
    {  
    }  
}
```



Egg hunter

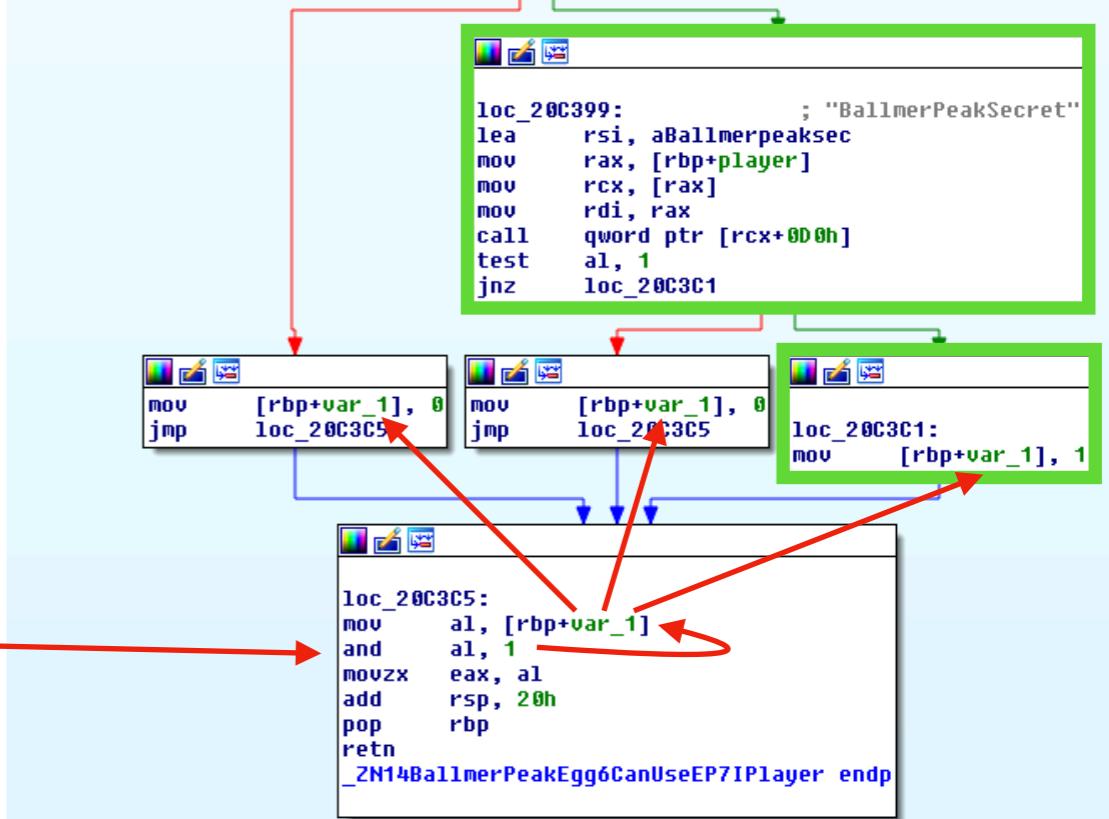
```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        ret = IPlayer::????(player, "BallmerPeakSecret");  
        if (ret)  
        {  
            canuse = true;  
        }  
        else  
        {  
            canuse = false;  
        }  
    }  
    else  
    {  
        canuse = false;  
    }  
}
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        ret = IPlayer::????(player, "BallmerPeakSecret");  
        if (ret)  
        {  
            canuse = true;  
        }  
        else  
        {  
            canuse = false;  
        }  
    }  
    else  
    {  
        canuse = false;  
    }  
  
    return canuse; // We want to return true  
}
```

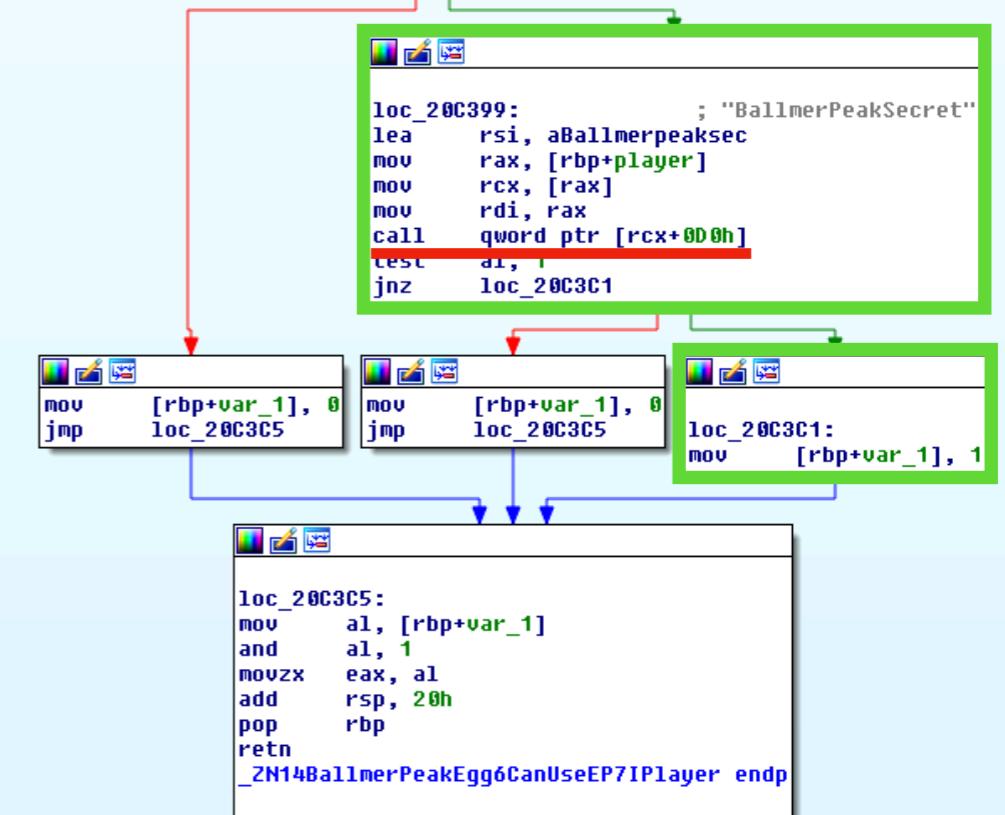
```
; Attributes: static bp-based frame  
  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
mov [rbp+this], rdi  
mov [rbp+player], rsi  
mov rsi, [rbp+this]  
mov rdi, [rbp+player]  
mov [rbp+var_20], rdi  
mov rdi, rsi ; this  
mov rsi, [rbp+var_20] ; player  
call _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test al, 1  
jnz loc_20C399
```



Egg hunter

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this,  
                      IPlayer *player) {  
  
    int ret;  
    bool canuse;  
  
    ret = ItemPickup::CanUse(this, player);  
    if (ret)  
    {  
        ret = IPlayer::????(player, "BallmerPeakSecret");  
        if (ret)  
        {  
            canuse = true;  
        }  
        else  
        {  
            canuse = false;  
        }  
    }  
    else  
    {  
        canuse = false;  
    }  
  
    return canuse; // We want to return true  
}
```

```
; Attributes: static bp-based frame  
  
; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)  
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer  
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near  
  
var_20= qword ptr -20h  
player= qword ptr -18h  
this= qword ptr -10h  
var_1= byte ptr -1  
  
push rbp  
mov rbp, rsp  
sub rsp, 20h  
mov [rbp+this], rdi  
mov [rbp+player], rsi  
mov rsi, [rbp+this]  
mov rdi, [rbp+player]  
mov [rbp+var_20], rdi  
mov rdi, rsi ; this  
mov rsi, [rbp+var_20] ; player  
call _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)  
test al, 1  
jnz loc_20C399
```



Egg hunter

How to find which **function** is **called**?

- Debugger (own server or offline)
 - Break
 - Step into
- Disassembler
 - vtable



Egg hunter

How to find which **function** is **called**?

- Debugger (own server or offline) 
 - Break
 - Step into
- Disassembler
 - vtable



Egg hunter

Debugger - **gdb** (on the **server**)

- Breakpoint at `BallmerPeakEgg::CanUse`

IF ONLINE

```
$ ps a
  PID  TTY      STAT TIME  COMMAND
 2334 pts/18  Sl+    0:00 ./MasterServer
 2410 pts/17  Sl+   11:08 ./PwnAdventure3Server
 2414 pts/17  Sl+   10:56 ./PwnAdventure3Server
 2416 pts/17  Sl+   12:35 ./PwnAdventure3Server
 3232 pts/1   Ss     0:00 bash
 3603 pts/1   R+    0:00 ps a
$ sudo su
# gdb -p 2410
(gdb) break BallmerPeakEgg::CanUse
(gdb) continue
```



Egg hunter

Debugger - **gdb** (on the **client**)

- Breakpoint at `BallmerPeakEgg::CanUse`

```
$ ps a
 PID  TTY      STAT   TIME   COMMAND
 2410 pts/17  Rl+   11:08  ./PwnAdventure3-Linux-Shipping
 3232 pts/1    Ss     0:00  bash
 3603 pts/1    R+     0:00  ps a
$ sudo su
# gdb -p 2410
(gdb) break BallmerPeakEgg::CanUse
(gdb) continue
```



Egg hunter

Let's trigger the function **BallmerPeakEgg:CanUse** (on the client):

- Offline mode constantly trigger the function
- If online, type BallmerPeakEgg in the chat box with the proxy enabled

```
def parse(p_out):  
  
    p_in = ""  
  
    def getme(item):  
        return "ee" + struct.pack("I", item)  
  
    def loc(x, y, z):  
        return "mv" + struct.pack("ffffHHBB", x, y, z, 0, 0, 0, 0, 0)  
  
    def chat(msg):  
        return "#*" + struct.pack("H", len(msg)) + msg  
  
    if chat("BallmerPeakEgg") in p_out:  
        p_out += loc(-2778.0, -11035.0, 10504.0)  
        p_out += getme(20)  
  
    return (p_in, p_out)
```



Egg hunter

The process **break** at **BallmerPeakEgg::CanUse**.

Now we can set a **breakpoint** at `call QWORD PTR [rcx+0xd0]`:

```
(gdb) set disassembly-flavor intel
(gdb) x/20i $rip
=> 0x7f3c6ff26374: mov rdi,QWORD PTR [rbp-0x18]
  0x7f3c6ff26378: mov QWORD PTR [rbp-0x20],rdi
  ...
  0x7f3c6ff263a7: mov rdi,rax
0x7f3c6ff263aa: call QWORD PTR [rcx+0xd0]
  0x7f3c6ff263b0: test al,0x1
  ...
  0x7f3c6ff263c5: mov al,BYTE PTR [rbp-0x1]
(gdb) break *0x7f3c6ff263aa
```



Egg hunter

Finally, we can see the **function called**:

```
(gdb) continue
(gdb) step
Player::HasPickedUp (this=0x72ed9e0, name=0x7f3c6ff473e9
"BallmerPeakSecret") at Player.cpp:864
```



Egg hunter

Final pseudo-code translation for **BallmerPeakEgg:CanUser:**

```
BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player) {
    int ret;
    bool canuse;

    ret = ItemPickup::CanUse(this, player); // Seems to be always true
    if (ret)
    {
        ret = Player::HasPickedUp(player, "BallmerPeakSecret");
        if (ret)
        {
            canuse = true;
        }
        else
        {
            canuse = false;
        }
    }
    else
    {
        canuse = false;
    }

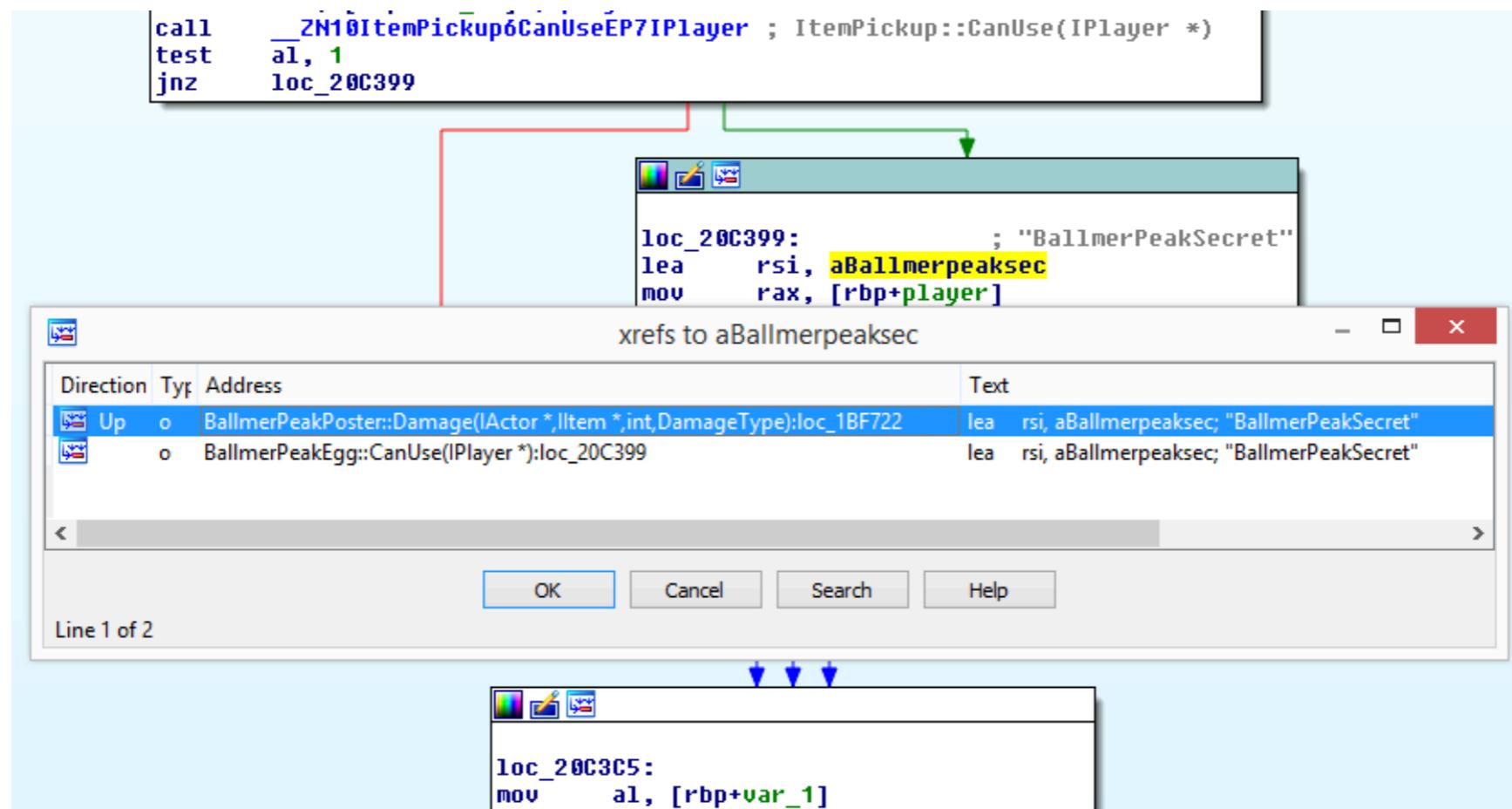
    return canuse; // We want to return true
}
```



Egg hunter

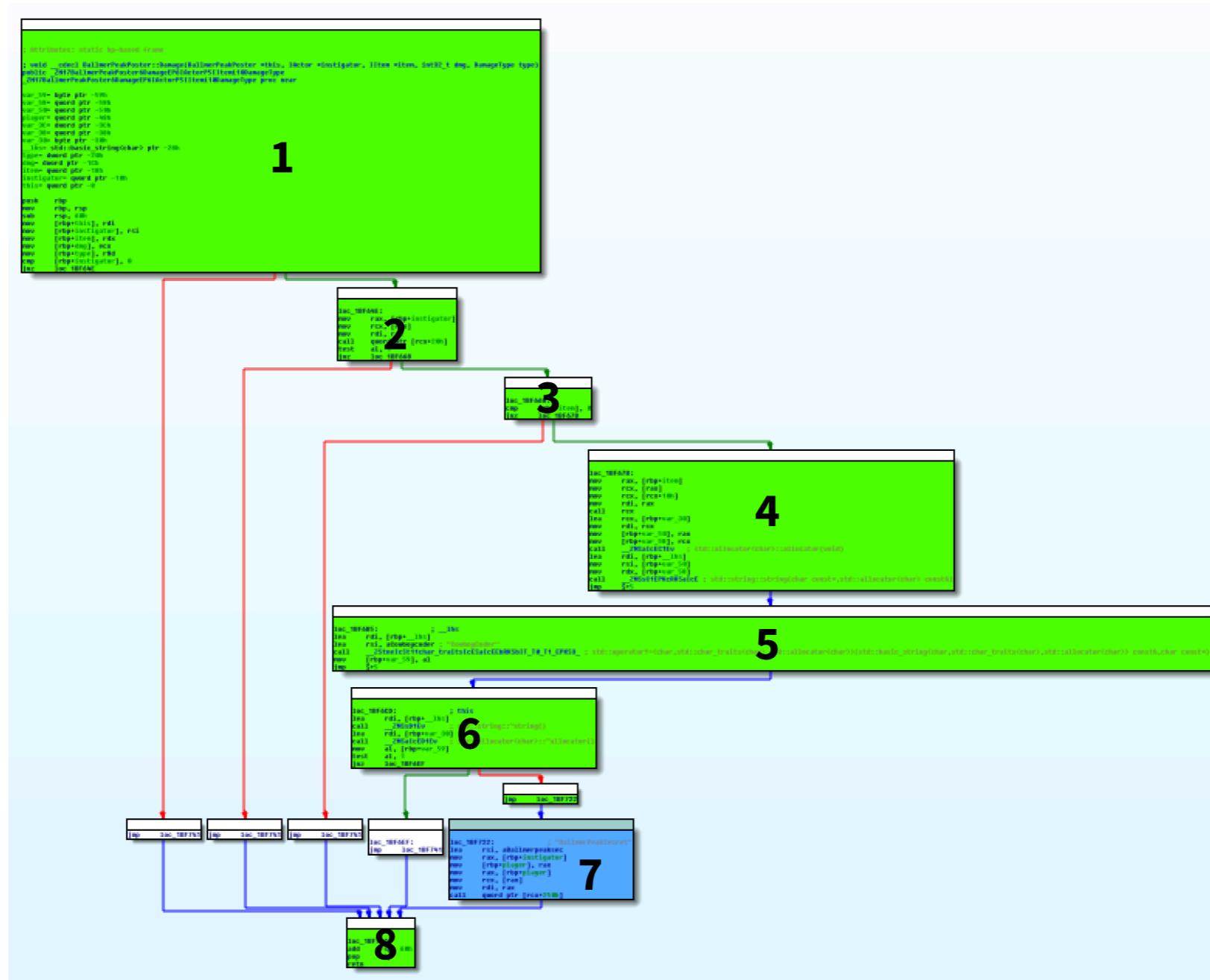
It seems that at some point, we need to **pick up** a “**BallmerPeakSecret**” in order to **use** (pick up) the **BallmerPeakEgg**.

Where is the “**BallmerPeakSecret**” string used?



Egg hunter

This is an **overview** of BallmerPeakPoster::Damage:



Egg hunter

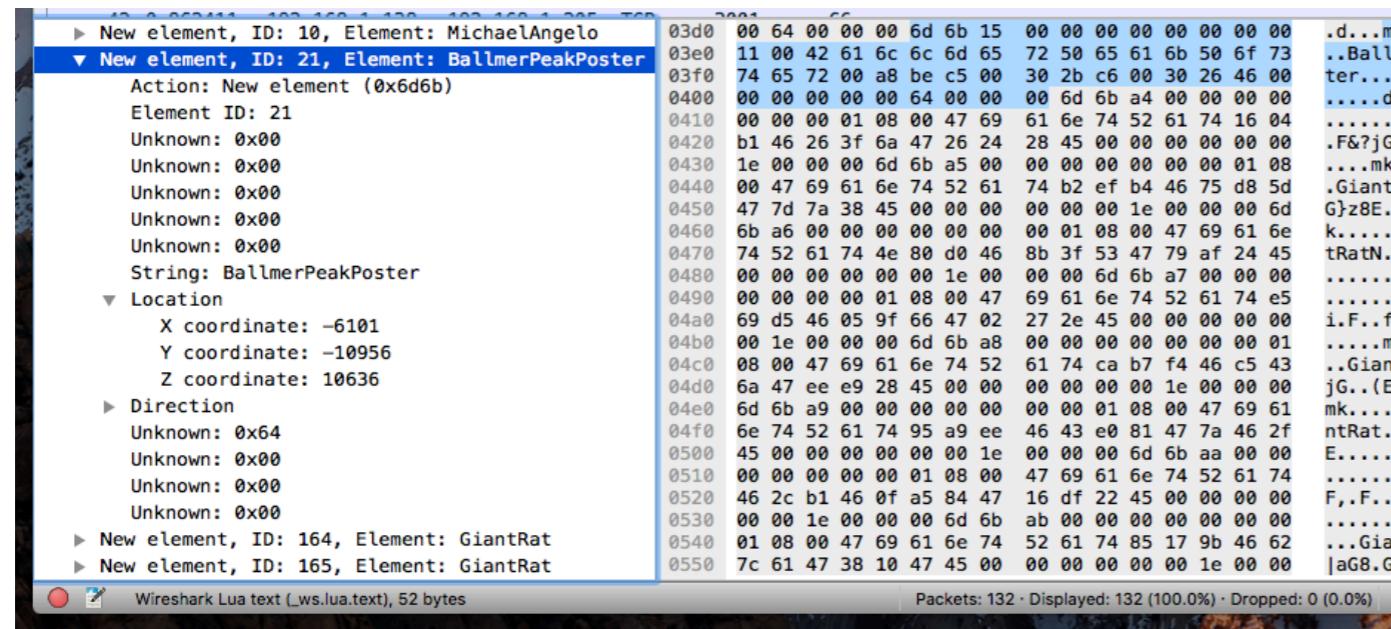
Set a **breakpoint** at **BallmerPeakPoster::Damage**:

```
(gdb) break BallmerPeakPoster::Damage  
(gdb) continue
```



Egg hunter

Go to **BallmerPeakPoster** and start fire *Great Balls of Fire* at it

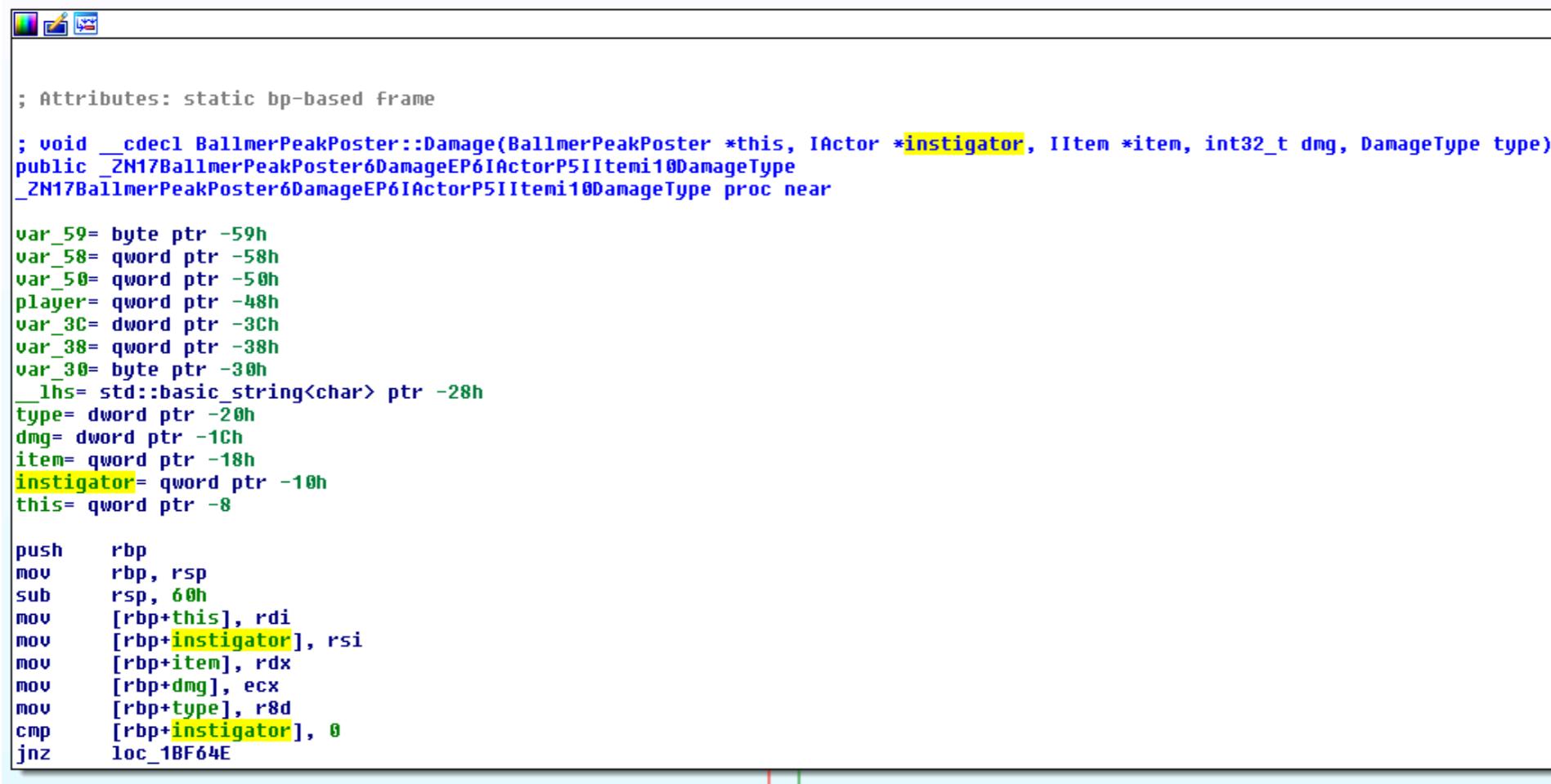


```
def parse(p_out):  
  
    p_in = ""  
  
    def newspawn(opcode, x, y, z):  
        return opcode + struct.pack("=HfffHHH", 0, x, y, z, 0, 0)  
  
    if len(p_out) == 22 and p_out[2:4] == "\x00\x00" and p_out[16:] == "\x00\x00\x00\x00\x00\x00":  
  
        # Ballmer Peak location  
        x = -6791.0  
        y = -11655.0  
        z = 10528.0  
  
        p_out = newspawn(p_out[:2], x, y, z)  
  
    return (p_in, p_out)
```



Egg hunter

1. Entry point (disassembler)



The screenshot shows a debugger's assembly view. The code is for the `BallmerPeakPoster::Damage` function. The assembly instructions are color-coded: `instigator` is yellow, `this` is green, and `item`, `dmg`, and `type` are blue. The assembly code includes variable declarations and their addresses:

```
; Attributes: static bp-based frame
; void __cdecl BallmerPeakPoster::Damage(BallmerPeakPoster *this, IActor *instigator, IItem *item, int32_t dmg, DamageType type)
public _ZN17BallmerPeakPoster6DamageEP6IActorP5IItemi10DamageType
_ZN17BallmerPeakPoster6DamageEP6IActorP5IItemi10DamageType proc near

var_59= byte ptr -59h
var_58= qword ptr -58h
var_50= qword ptr -50h
player= qword ptr -48h
var_3C= dword ptr -3Ch
var_38= qword ptr -38h
var_30= byte ptr -30h
__lhs= std::basic_string<char> ptr -28h
type= dword ptr -20h
dmg= dword ptr -1Ch
item= qword ptr -18h
instigator= qword ptr -10h
this= qword ptr -8

push    rbp
mov     rbp, rsp
sub    rsp, 60h
mov     [rbp+this], rdi
mov     [rbp+instigator], rsi
mov     [rbp+item], rdx
mov     [rbp+dmg], ecx
mov     [rbp+type], r8d
cmp     [rbp+instigator], 0
jnz    loc_1BF64E
```

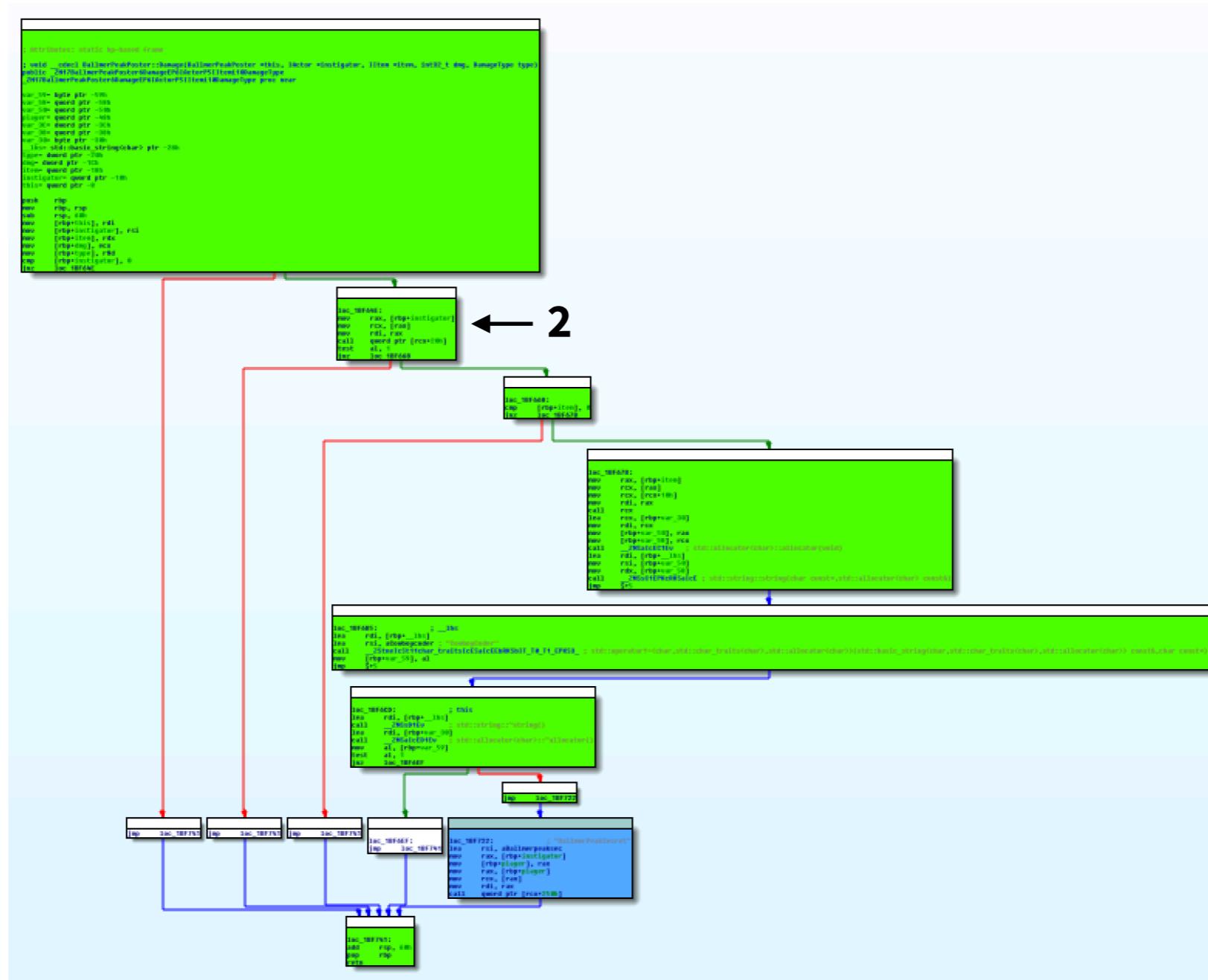
1. Entry point (debugger)

```
(gdb) x/x $rsi
0x5e0fc0: 0x230c2980
```



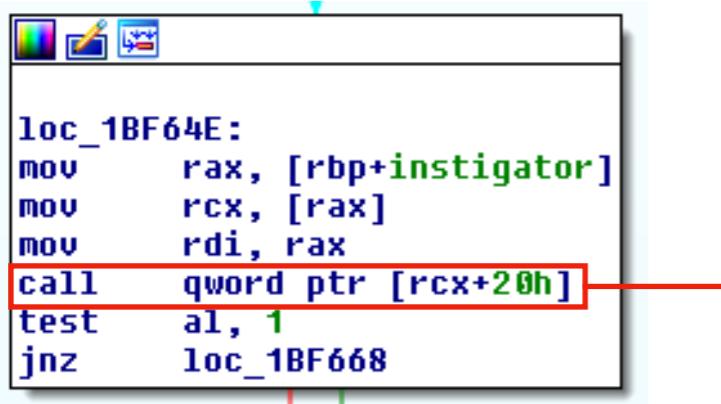
Egg hunter

This is an **overview** of BallmerPeakPoster::Damage:



Egg hunter

2. Block (disassembler)



```
loc_1BF64E:
mov    rax, [rbp+instigator]
mov    rcx, [rax]
mov    rdi, rax
call  qword ptr [rcx+20h]
test   al, 1
jnz   loc_1BF668
```

What function is called?

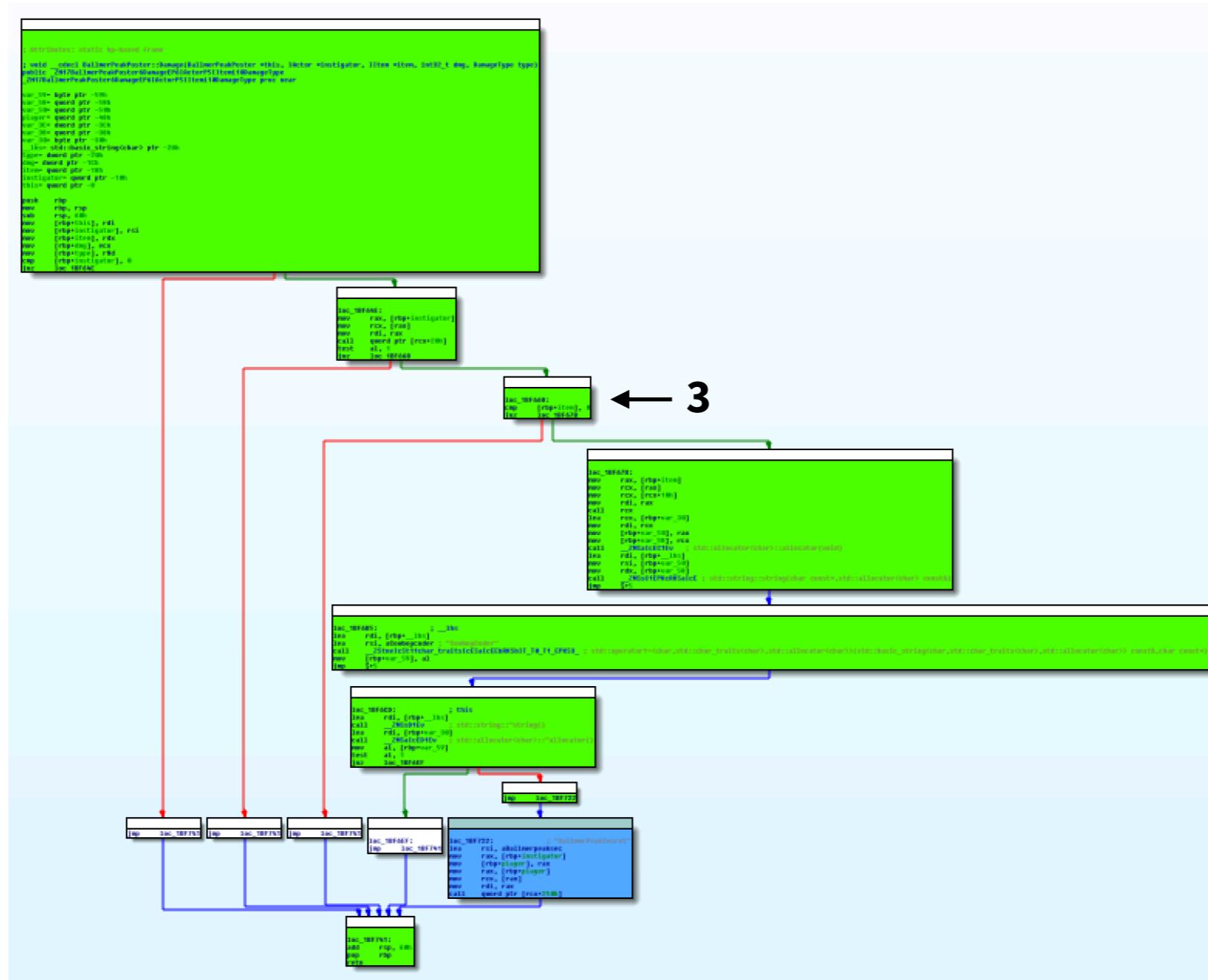
2. Block (debugger)

```
(gdb) set disassembly-flavor intel
(gdb) x/30i $rip
[...]
0x7f8622dcd655: mov rdi,rax
0x7f8622dcd658: call QWORD PTR [rcx+0x20]
0x7f8622dcd65b: test al,0x1
0x7f8622dcd65d: jne 0x7f8622dcd668
[...]
(gdb) break *0x7f8622dcd658
(gdb) break *0x7f8622dcd668
(gdb) continue
(gdb) step
Player::IsPlayer (this=0x5e0fc0)
(gdb) finish
Value returned is $1 = true
```



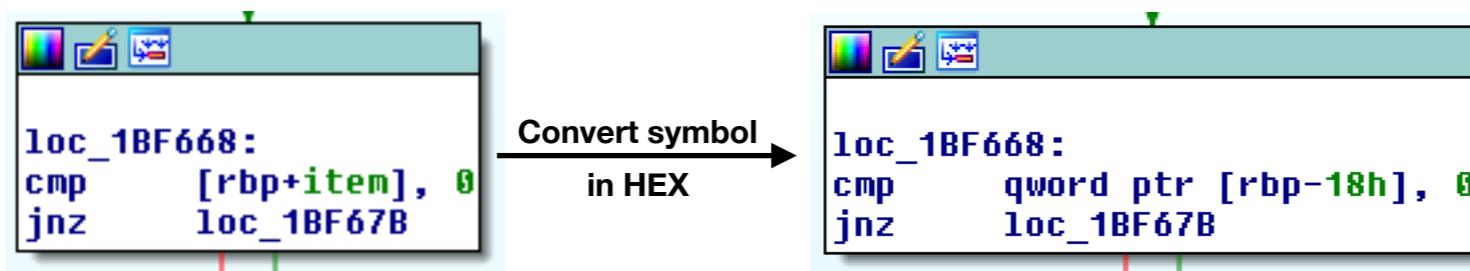
Egg hunter

This is an **overview** of BallmerPeakPoster::Damage:



Egg hunter

3. Block (disassembler)



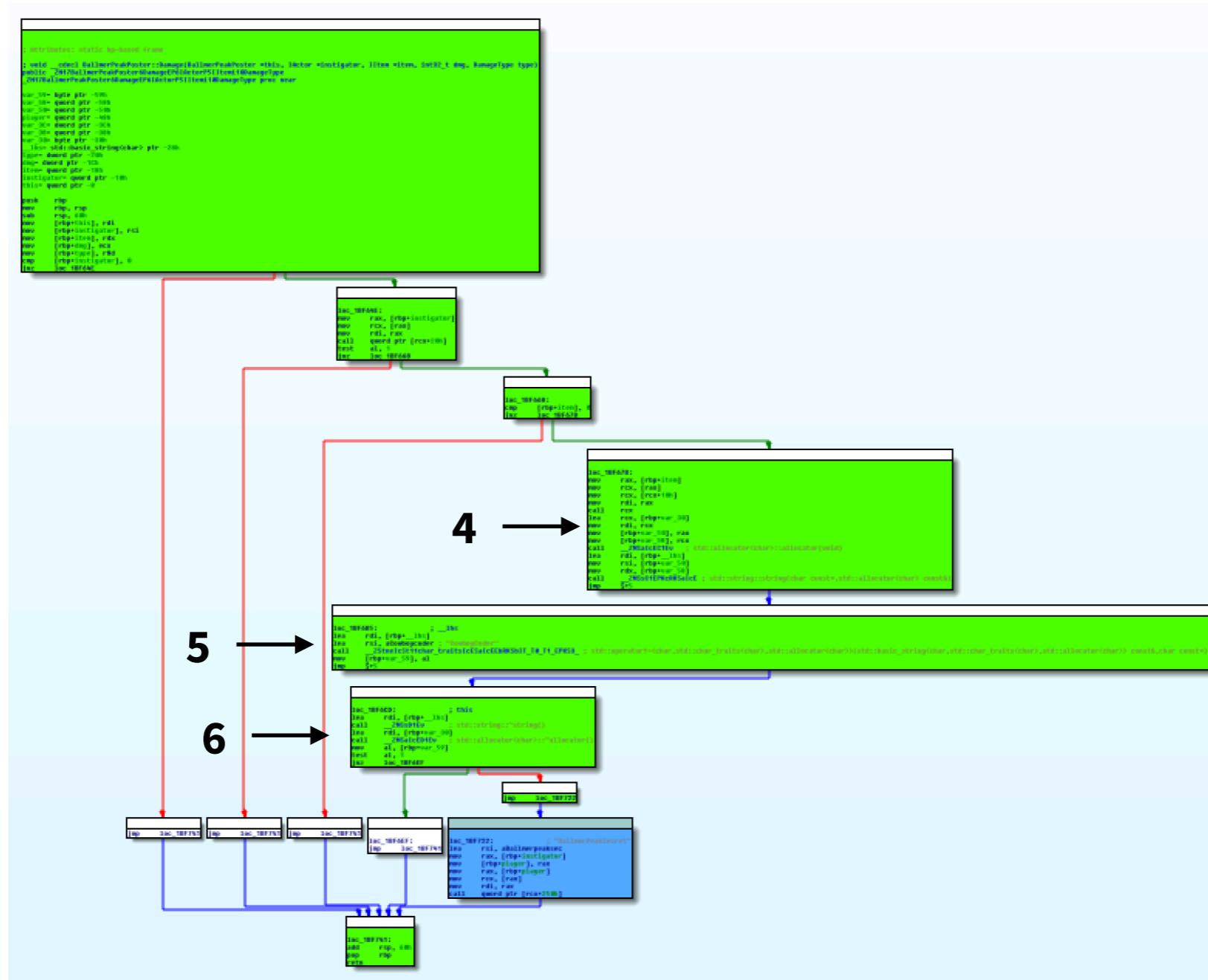
3. Block (debugger)

```
(gdb) x/xg $rbp-0x18
0x7ffffd46c5988: 0x0000000005bc8290
```



Egg hunter

This is an **overview** of BallmerPeakPoster::Damage:



Egg hunter

4, 5 and 6. Block (disassembler)

The image shows three vertically stacked debugger windows, each displaying assembly code:

- Top Window:** Shows assembly from address `loc_1BF67B:` to `jmp $+5`. It includes several `mov`, `call`, and `lea` instructions involving registers `rax`, `rcx`, `rdi`, `rsi`, and `rdx`, and memory locations like `[rbp+item]`, `[rbp+var_30]`, etc.
- Middle Window:** Shows assembly from address `loc_1BF6B5:` to `jmp $+5`. It includes `lea`, `call`, and `mov` instructions. A comment `; __lhs` is present above the first `lea` instruction, and a string "CowboyCoder" is shown in memory at `[rip+0x6a6b6]`.
- Bottom Window:** Shows assembly from address `loc_1BF6CD:` to `jnz loc_1BF6EF`. It includes `lea`, `call`, and `test` instructions. A comment `; this` is present above the first `lea` instruction.

4, 5 and 6. Block (debugger)

- In order to move to the **blue** box, AL should be different than 1 (True) **1**
- AL = var_59** **2**
- var_59** is the return value from **3**
- Once un-mangled, the function is:
`std::operator!=(lhs, rhs)`

```
(gdb) x/30i $rip
[...]
0x7f8622dcd6b5: lea rdi,[rbp-0x28]
0x7f8622dcd6b9: lea rsi,[rip+0x6a6b6] # 0x7f8622e37d76
0x7f8622dcd6c0: call 0x7f8622d1afb0
0x7f8622dcd6c5: mov BYTE PTR [rbp-0x59],al
0x7f8622dcd6c8: jmp 0x7f8622dcd6cd
[...]
(gdb) break *0x7f8622dcd6c0
(gdb) continue
(gdb) x/s $rsi
0x7f8622e37d76: "CowboyCoder"
(gdb) x/xg $rdi
0xffffd46c5978: 0x000000005db9d38
(gdb) x/s 0x000000005db9d38
0x5db9d38: "GreatBallsOfFire"
```



Egg hunter

4, 5 and 6. Block (disassembler)

The image shows three vertically stacked debugger windows, each displaying assembly code:

- Top Window:** Shows assembly from address `loc_1BF67B:` to `jmp $+5`. It includes several `mov`, `call`, and `lea` instructions involving registers `rax`, `rcx`, `rdi`, `rsi`, and `rdx`, and memory locations like `[rbp+item]`, `[rbp+var_30]`, etc.
- Middle Window:** Shows assembly from address `loc_1BF6B5:` to `jmp $+5`. It includes `lea`, `call`, and `mov` instructions. A comment `; __lhs` is present above the first `lea` instruction, and a string "CowboyCoder" is passed to a `__StneIcSt11char_traitsIcESaIcEEbRKSBIT_T0_T1_EPKS3_` function.
- Bottom Window:** Shows assembly from address `loc_1BF6CD:` to `jnz loc_1BF6EF 1`. It includes `lea`, `call`, and `test` instructions. The `call` instruction is annotated with `; std::string::~string()` and the `test` instruction with `al, 1`.

4, 5 and 6. Block (debugger)

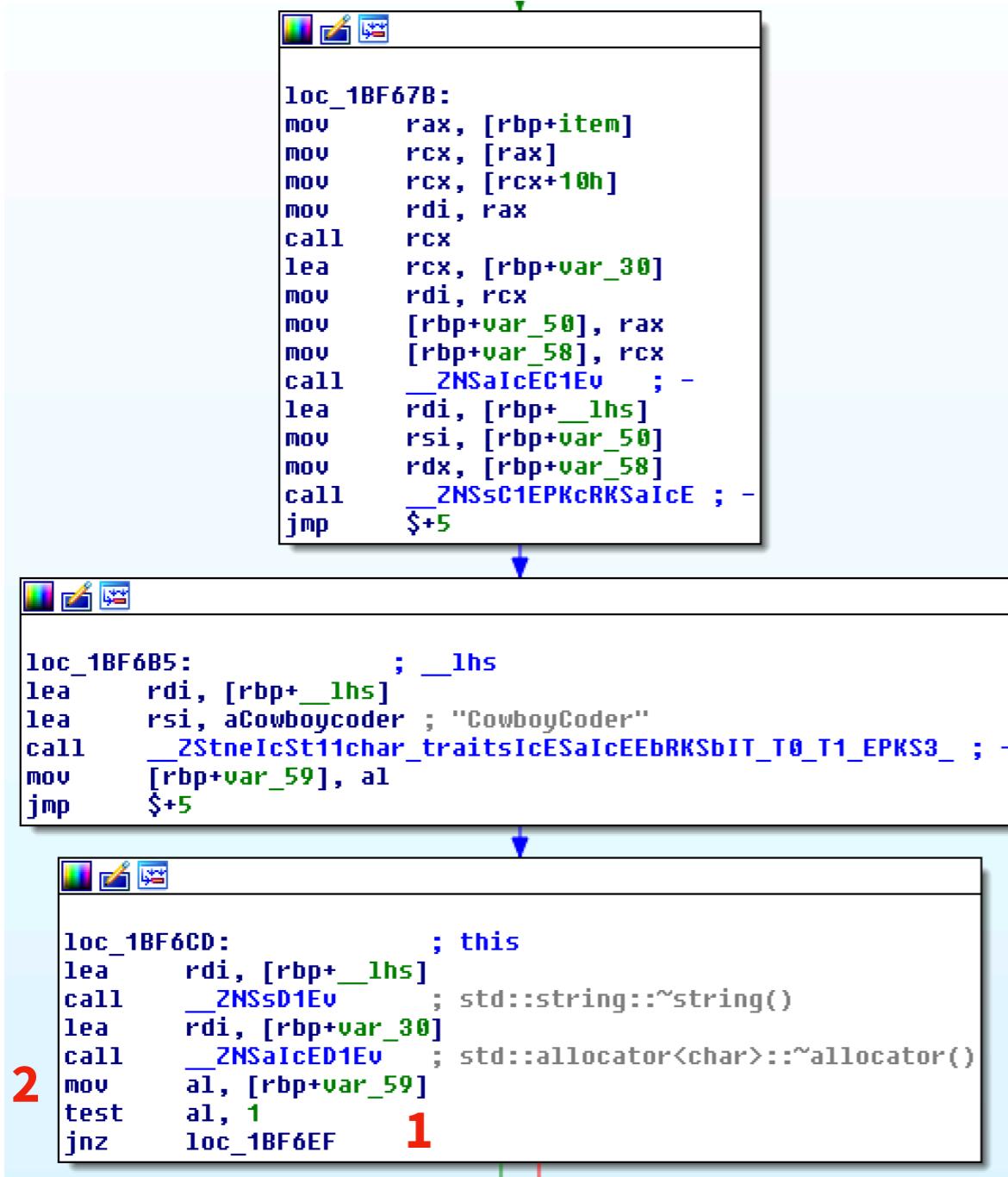
- In order to move to the **blue** box, AL should be different than 1 (True) **1**
- AL = var_59 2**
- var_59** is the return value from **3**
- Once un-mangled, the function is:
`std::operator!=(lhs, rhs)`

```
(gdb) x/30i $rip
[...]
0x7f8622dcd6b5: lea rdi,[rbp-0x28]
0x7f8622dcd6b9: lea rsi,[rip+0x6a6b6] # 0x7f8622e37d76
0x7f8622dcd6c0: call 0x7f8622d1afb0
0x7f8622dcd6c5: mov BYTE PTR [rbp-0x59],al
0x7f8622dcd6c8: jmp 0x7f8622dcd6cd
[...]
(gdb) break *0x7f8622dcd6c0
(gdb) continue
(gdb) x/s $rsi
0x7f8622e37d76: "CowboyCoder"
(gdb) x/xg $rdi
0xffffd46c5978: 0x000000005db9d38
(gdb) x/s 0x000000005db9d38
0x5db9d38: "GreatBallsOfFire"
```



Egg hunter

4, 5 and 6. Block (disassembler)



```
loc_1BF67B:
mov    rax, [rbp+item]
mov    rcx, [rax]
mov    rcx, [rcx+10h]
mov    rdi, rax
call   rcx
lea    rcx, [rbp+var_30]
mov    rdi, rcx
mov    [rbp+var_50], rax
mov    [rbp+var_58], rcx
call   __ZNSaIcEc1Ev ; -
lea    rdi, [rbp+_lhs]
mov    rsi, [rbp+var_50]
mov    rdx, [rbp+var_58]
call   __ZNSsC1EPKcRKSaIcE ; -
jmp   $+5

loc_1BF685:          ; __lhs
lea    rdi, [rbp+_lhs]
lea    rsi, aCowboyCoder ; "CowboyCoder"
call   __ZStneIcSt11char_traitsIcESaIcEEbRKSBIT_T0_T1_EPKS3_ ; -
mov    [rbp+var_59], al
jmp   $+5

loc_1BF6CD:          ; this
lea    rdi, [rbp+_lhs]
call   __ZNsD1Ev        ; std::string::~string()
lea    rdi, [rbp+var_30]
call   __ZNSaIcED1Ev ; std::allocator<char>::~allocator()
mov    al, [rbp+var_59]
test   al, 1             1
jnz   loc_1BF6EF
```

2

1

4, 5 and 6. Block (debugger)

- In order to move to the **blue** box, AL should be different than 1 (True) **1**
- AL = var_59 **2**
- var_59 is the return value from **3**
- Once un-mangled, the function is:
`std::operator!=(lhs, rhs)`

```
(gdb) x/30i $rip
[...]
0x7f8622dcd6b5: lea rdi,[rbp-0x28]
0x7f8622dcd6b9: lea rsi,[rip+0x6a6b6] # 0x7f8622e37d76
0x7f8622dcd6c0: call 0x7f8622d1afb0
0x7f8622dcd6c5: mov BYTE PTR [rbp-0x59],al
0x7f8622dcd6c8: jmp 0x7f8622dcd6cd
[...]
(gdb) break *0x7f8622dcd6c0
(gdb) continue
(gdb) x/s $rsi
0x7f8622e37d76: "CowboyCoder"
(gdb) x/xg $rdi
0xffffd46c5978: 0x000000005db9d38
(gdb) x/s 0x000000005db9d38
0x5db9d38: "GreatBallsOfFire"
```



Egg hunter

4, 5 and 6. Block (disassembler)

The screenshot shows three assembly code blocks from the Immunity Debugger:

- Block 1 (Top):**

```
loc_1BF67B:  
mov    rax, [rbp+item]  
mov    rcx, [rax]  
mov    rcx, [rcx+10h]  
mov    rdi, rax  
call   rcx  
lea    rcx, [rbp+var_30]  
mov    rdi, rcx  
mov    [rbp+var_50], rax  
mov    [rbp+var_58], rcx  
call   __ZNsaIcEc1Ev ; -  
lea    rdi, [rbp+_lhs]  
mov    rsi, [rbp+var_50]  
mov    rdx, [rbp+var_58]  
call   __ZNSSC1EPKcRKSaIcE ; -  
jmp   $+5
```
- Block 3 (Middle):**

```
loc_1BF6B5: ; _lhs  
lea    rdi, [rbp+_lhs]  
lea    rsi, aCowboyCoder ; "CowboyCoder"  
call   __ZStneIcSt11char_traitsIcESaIcEEbRKSBIT_T0_T1_EPKS3_ ; -  
mov    [rbp+var_59], al  
jmp   $+5
```
- Block 2 (Bottom):**

```
loc_1BF6CD: ; this  
lea    rdi, [rbp+_lhs]  
call   __ZNSSD1Ev ; std::string::~string()  
lea    rdi, [rbp+var_30]  
call   __ZNsaIcD1Ev ; std::allocator<char>::~allocator()  
mov    al, [rbp+var_59]  
test   al, 1  
jnz   loc_1BF6EF 1
```

4, 5 and 6. Block (debugger)

- In order to move to the **blue** box, AL should be different than 1 (True) **1**
- AL = var_59 **2**
- var_59 is the return value from **3**
- Once un-mangled, the function is:
`std::operator!=(lhs, rhs)`

```
(gdb) x/30i $rip  
[...]  
0x7f8622dcd6b5: lea rdi,[rbp-0x28]  
0x7f8622dcd6b9: lea rsi,[rip+0x6a6b6] # 0x7f8622e37d76  
0x7f8622dcd6c0: call 0x7f8622d1afb0  
0x7f8622dcd6c5: mov BYTE PTR [rbp-0x59],al  
0x7f8622dcd6c8: jmp 0x7f8622dcd6cd  
[...]  
(gdb) break *0x7f8622dcd6c0  
(gdb) continue  
(gdb) x/s $rsi  
0x7f8622e37d76: "CowboyCoder"  
(gdb) x/xg $rdi  
0xffffd46c5978: 0x000000005db9d38  
(gdb) x/s 0x000000005db9d38  
0x5db9d38: "GreatBallsOfFire"
```



Egg hunter

After the “*Unbearable Revenge*”, we should have enough money to **buy** the *Cowboy Coder from Major Payne in Town*:

```
# Coordinate Town  
x = -39602.8  
y = -18288.0  
z = 2400.28
```

Once bought, **come back** to *Ballmer Peak*:

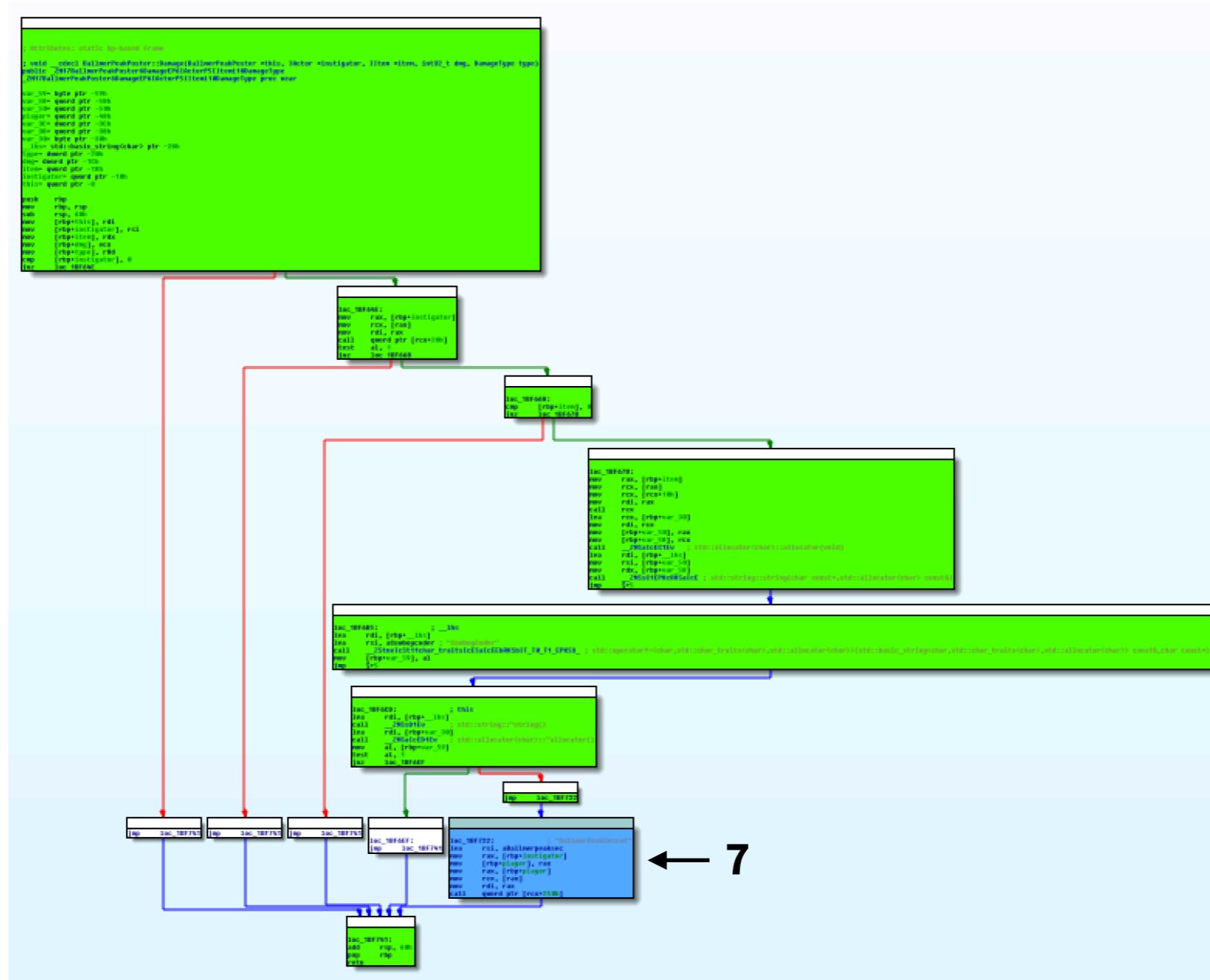
```
# Ballmer Peak location  
x = -6791.0  
y = -11655.0  
z = 10528.0
```

And **shoot** at the poster



Egg hunter

This is an **overview** of BallmerPeakPoster::Damage:



Egg hunter

7. Block (disassembler)

```
loc_1BF722:             ; "BallmerPeakSecret"
lea    rsi, aBallmerpeaksec
mov    rax, [rbp+instigator]
mov    [rbp+player], rax
mov    rax, [rbp+player]
mov    rcx, [rax]
mov    rdi, rax
call   qword ptr [rcx+250h]
```

What function is called?

7. Block (debugger)

```
(gdb) x/30i $rip
[...]
0x7f8622dcd722: lea rsi,[rip+0x6dcc0] # 0x7f8622e3b3e9
0x7f8622dcd729: mov rax,QWORD PTR [rbp-0x10]
0x7f8622dcd72d: mov QWORD PTR [rbp-0x48],rax
0x7f8622dcd731: mov rax,QWORD PTR [rbp-0x48]
0x7f8622dcd735: mov rcx,QWORD PTR [rax]
0x7f8622dcd738: mov rdi,rax
0x7f8622dcd73b: call QWORD PTR [rcx+0x250]
0x7f8622dcd741: add rsp,0x60
0x7f8622dcd745: pop rbp
0x7f8622dcd746: ret
[...]
(gdb) break *0x7f8622dcd73b
(gdb) continue
(gdb) step
Player::MarkAsPickedUp (this=0x619e010, name=0x7f8622e3b3e9 "BallmerPeakSecret")
```



Egg hunter

`Player::MarkAsPickedUp("BallmerPeakSecret")` is exactly what we needed!

Summary:

- If we shoot at the *Ballmer Peak Poster* with a *Cowboy Coder*, the function `Player::MarkAsPickedUp("BallmerPeakSecret")` will be called
- From now on, the function `Player::HasPickedUp("BallmerPeakSecret")` will return *True*
- Which means `BallmerPeakEgg::CanUse(...)` will also return *True*



Egg hunter

Demo: Get that Ballmer Peak Egg



Egg hunter

What if we had **no server** or **couldn't trigger the function** locally?

- Disassembler
- vtable



Egg hunter

What's the **purpose** of vtable?

```
Class IActor () {                                int main () {
    ...
}

Class Actor : IActor() {                         ...
    ...
}

Class Enemy : Actor() {                          Enemy *e;
    ...                                              Vector3 *v;
}

Class Bear : Enemy () {                           if (rand() % 2) {
    ...                                              e = new Bear();
}

Class GiantRat : Enemy () {                      } else {
    ...
}

    virtual int GetMaxHealth(void) {...};
    virtual Vector3 *GetPosition(void) {...};
    virtual void SetSpawner(Spawner *) {...};
    ...
}

    virtual int GetMaxHealth(void) {...};
    virtual Vector3 *GetPosition(void) {...};
    virtual void SetSpawner(Spawner *) {...};
    ...
}
```



Egg hunter

What's the **purpose** of vtable?

```
Class IActor () {                                int main () {
    ...
}

Class Actor : IActor() {                         ...
    ...
}

Class Enemy : Actor() {                          Enemy *e;
    ...                                              Vector3 *v;
}

Class Bear : Enemy () {                         if (rand() % 2) {
    ...
    e = new Bear();
}

} else {

Class GiantRat : Enemy () {                     e = new GiantRat();
    ...
    virtual int GetMaxHealth(void) {...};
    virtual Vector3 *GetPosition(void) {...};
    virtual void SetSpawner(Spawner *) {...};
    ...
}

Class GiantRat : Enemy () {                     }
    ...
    virtual int GetMaxHealth(void) {...};
    virtual Vector3 *GetPosition(void) {...};
    virtual void SetSpawner(Spawner *) {...};
    ...
}
```

The code illustrates the use of a vtable. The `IActor` interface defines a single method. The `Actor` class implements this interface. The `Enemy` and `Bear` classes inherit from `Actor`. The `GiantRat` class also inherits from `Enemy`. In the `main` function, an `Enemy` pointer `e` is created and assigned either a `Bear` or a `GiantRat` object based on a random value. The `v` variable is then set to the result of calling the `GetPosition` method via the vtable, which dispatches to the appropriate implementation in `Bear` or `GiantRat`.



Egg hunter

What's the **purpose** of vtable?

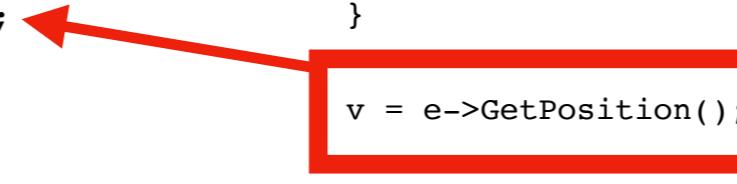
```
Class IActor () {                                int main () {
    ...
}

Class Actor : IActor() {                         ...
    ...
}

Class Enemy : Actor() {                          Enemy *e;
    ...                                              Vector3 *v;
}

Class Bear : Enemy () {                         if (_rand() % 2) {
    ...
    virtual int GetMaxHealth(void) {...};           e = new Bear();
    virtual Vector3 *GetPosition(void) {...};        } else {
    virtual void SetSpawner(Spawner *) {...};       e = new GiantRat();
    ...
}

Class GiantRat : Enemy () {                     }
    ...
    virtual int GetMaxHealth(void) {...};
    virtual Vector3 *GetPosition(void) {...};
    virtual void SetSpawner(Spawner *) {...};
    ...
}
```



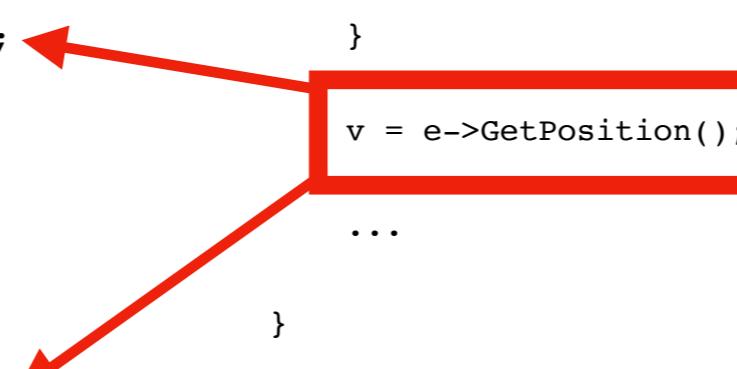
```
v = e->GetPosition();
```



Egg hunter

What's the **purpose** of vtable?

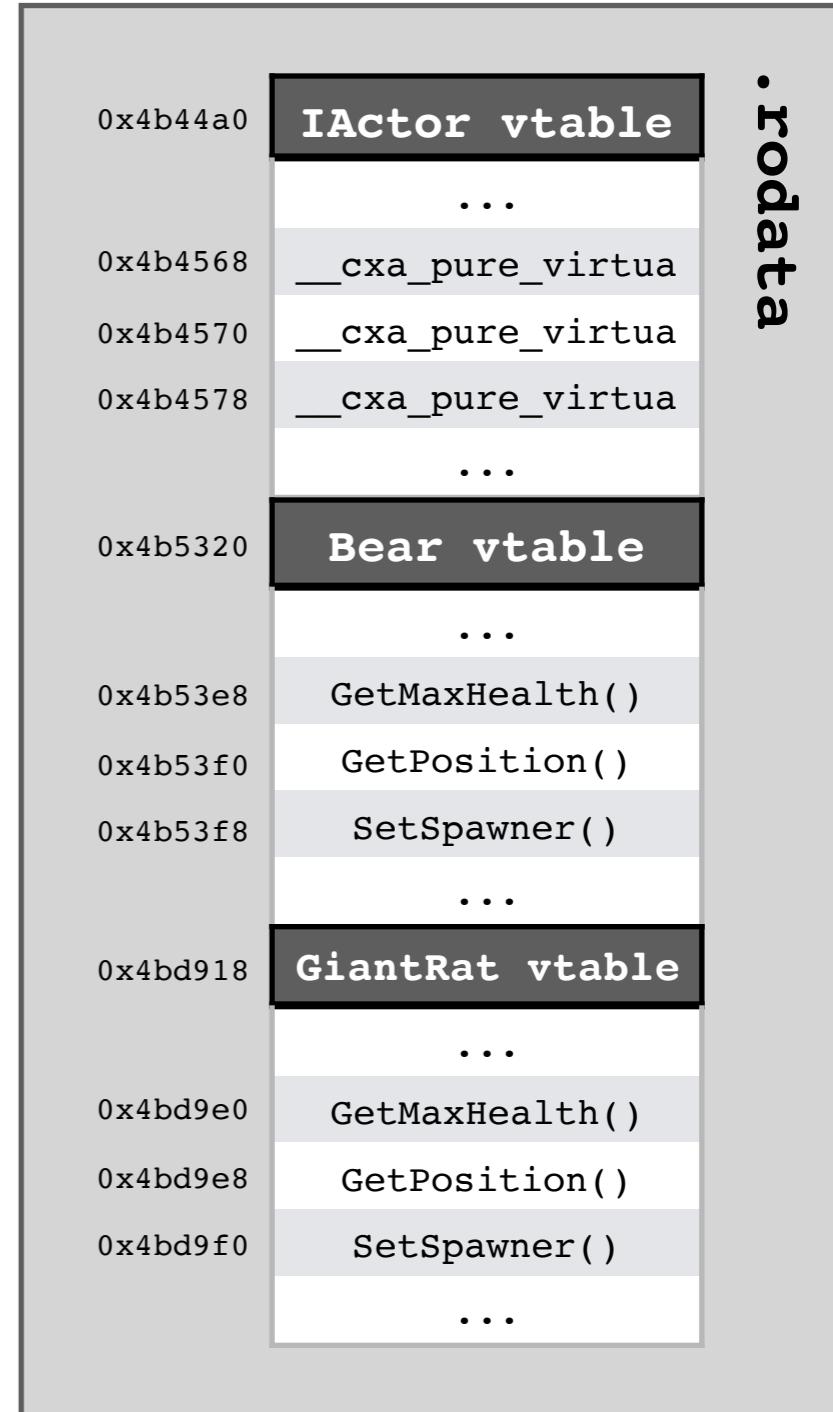
```
Class IActor () {  
    ...  
}  
  
Class Actor : IActor() {  
    ...  
}  
  
Class Enemy : Actor() {  
    ...  
}  
  
Class Bear : Enemy () {  
    ...  
    virtual int GetMaxHealth(void) {...};  
    virtual Vector3 *GetPosition(void) {...}; ←  
    virtual void SetSpawner(Spawner *) {...};  
    ...  
}  
  
Class GiantRat : Enemy () {  
    ...  
    virtual int GetMaxHealth(void) {...};  
    virtual Vector3 *GetPosition(void) {...};  
    virtual void SetSpawner(Spawner *) {...};  
    ...  
}  
  
int main () {  
    ...  
    Enemy *e;  
    Vector3 *v;  
    if (rand() % 2) {  
        e = new Bear();  
    } else {  
        e = new GiantRat();  
    }  
    v = e->GetPosition();  
    ...  
}
```



Egg hunter

In C++, for each **type** having **virtual function(s)**, the **compiler** create a **table** that contains **function pointers**: the **vtable**

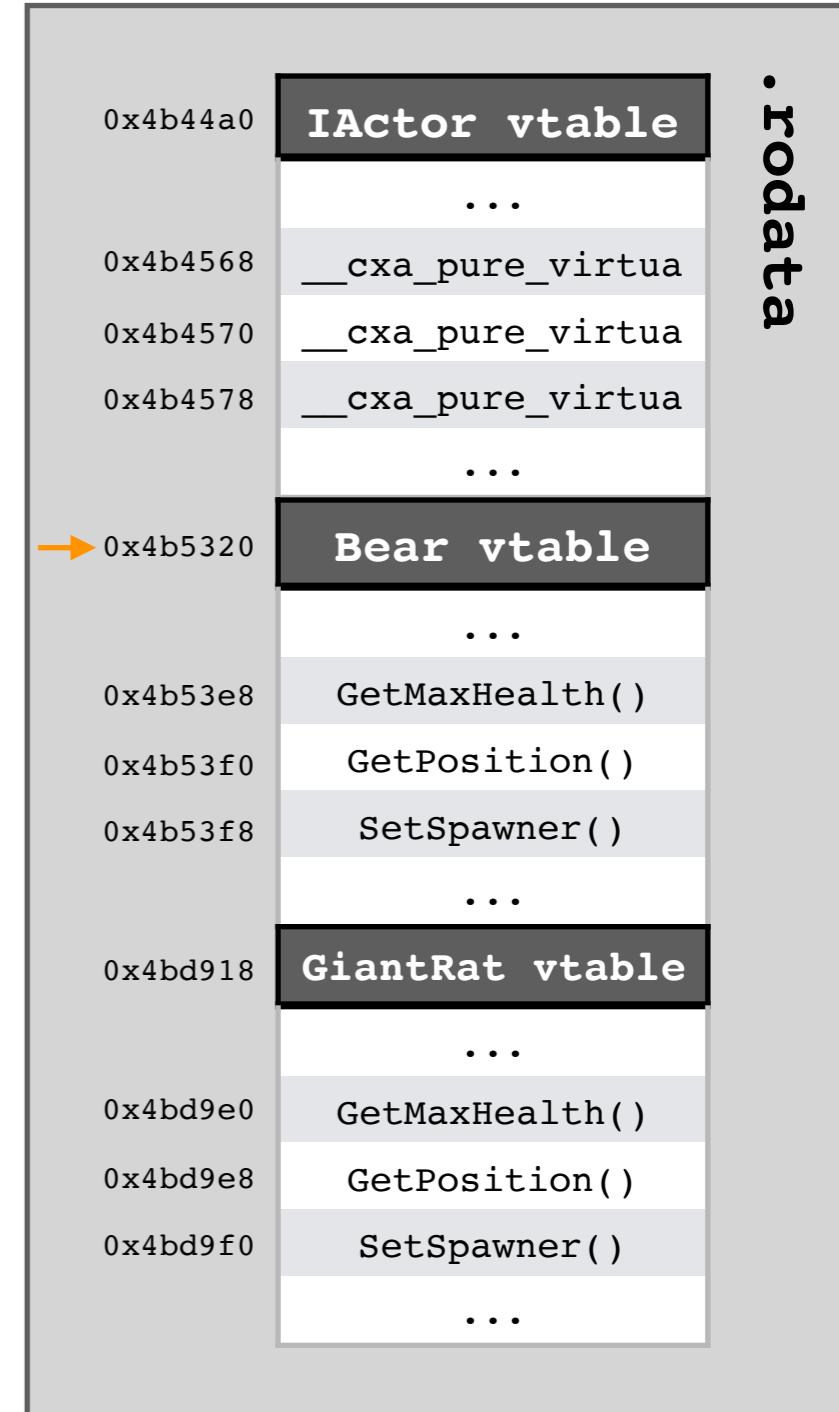
```
; var_1 = Bear object
mov    rax, [rbp+var_1]
mov    rcx, [rax]
mov    rsi, rbx
mov    rdi, rax
call   qword ptr [rcx+0xd0]
```



Egg hunter

In C++, for each **type** having **virtual function(s)**, the **compiler** create a **table** that contains **function pointers**: the **vtable**

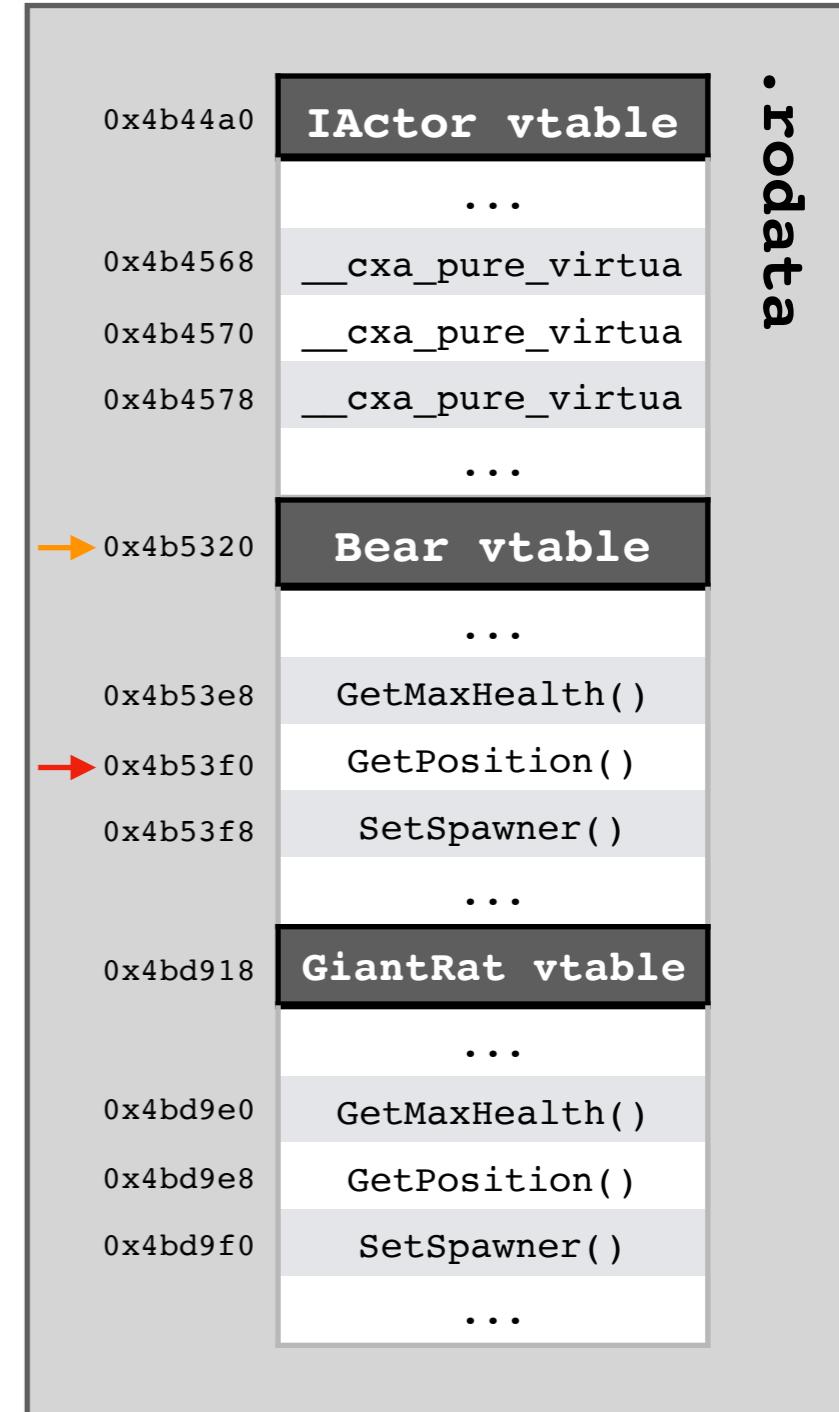
```
; var_1 = Bear object
mov    rax, [rbp+var_1]
mov    rcx, [rax]
mov    rsi, rbx
mov    rdi, rax
call   qword ptr [rcx+0xd0]
```



Egg hunter

In C++, for each **type** having **virtual function(s)**, the **compiler** create a **table** that contains **function pointers**: the **vtable**

```
; var_1 = Bear object
mov    rax, [rbp+var_1]
mov    rcx, [rax]
mov    rsi, rbx
mov    rdi, rax
call   qword ptr [rcx+0xd0]
```



Egg hunter

In C++, for each **type** having **virtual function(s)**, the **compiler** create a **table** that contains **function pointers**: the **vtable**

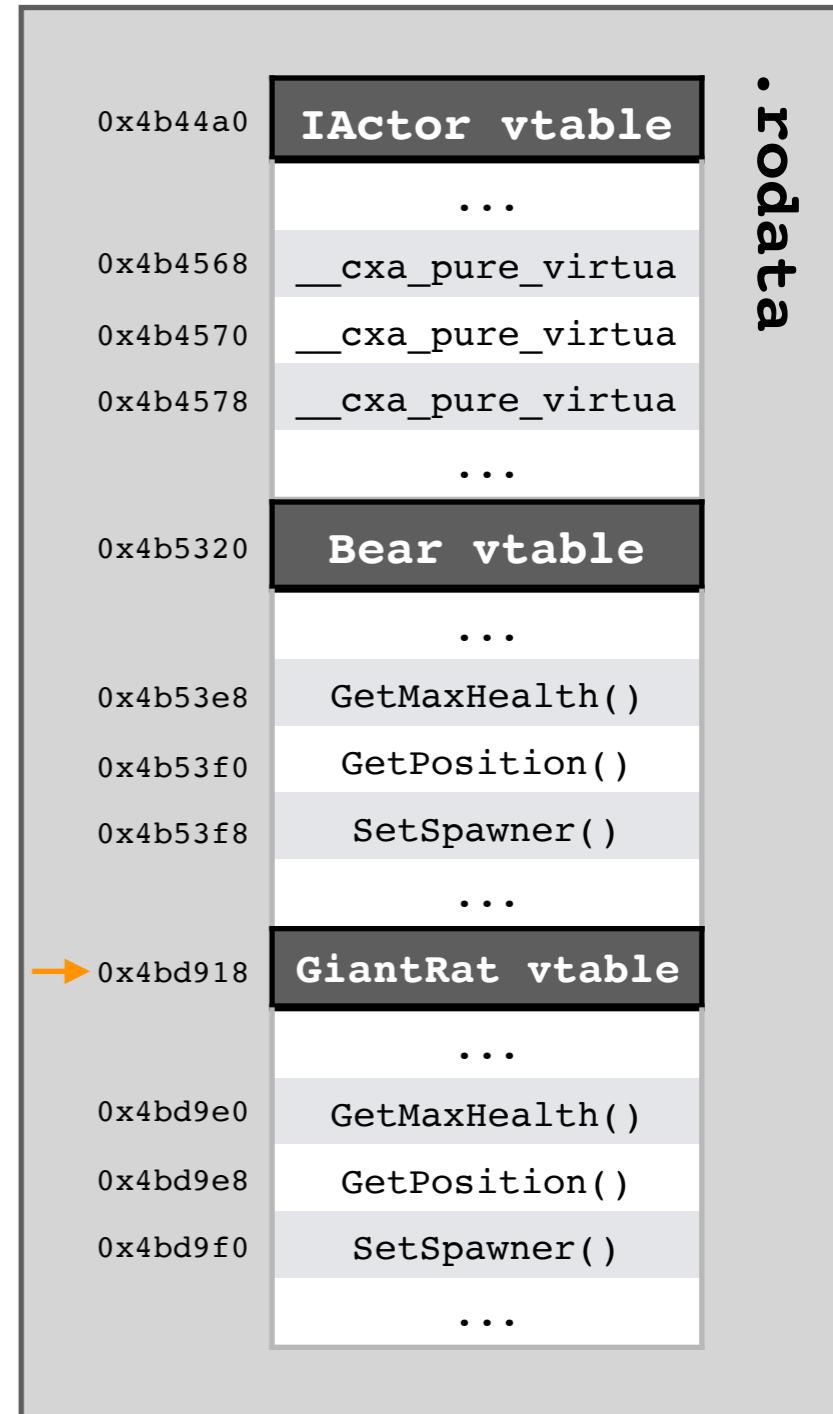
```
; var_1 = GiantRat object
mov    rax, [rbp+var_1]
mov    rcx, [rax]
mov    rsi, rbx
mov    rdi, rax
call   qword ptr [rcx+0xd0]
```



Egg hunter

In C++, for each **type** having **virtual function(s)**, the **compiler** create a **table** that contains **function pointers**: the **vtable**

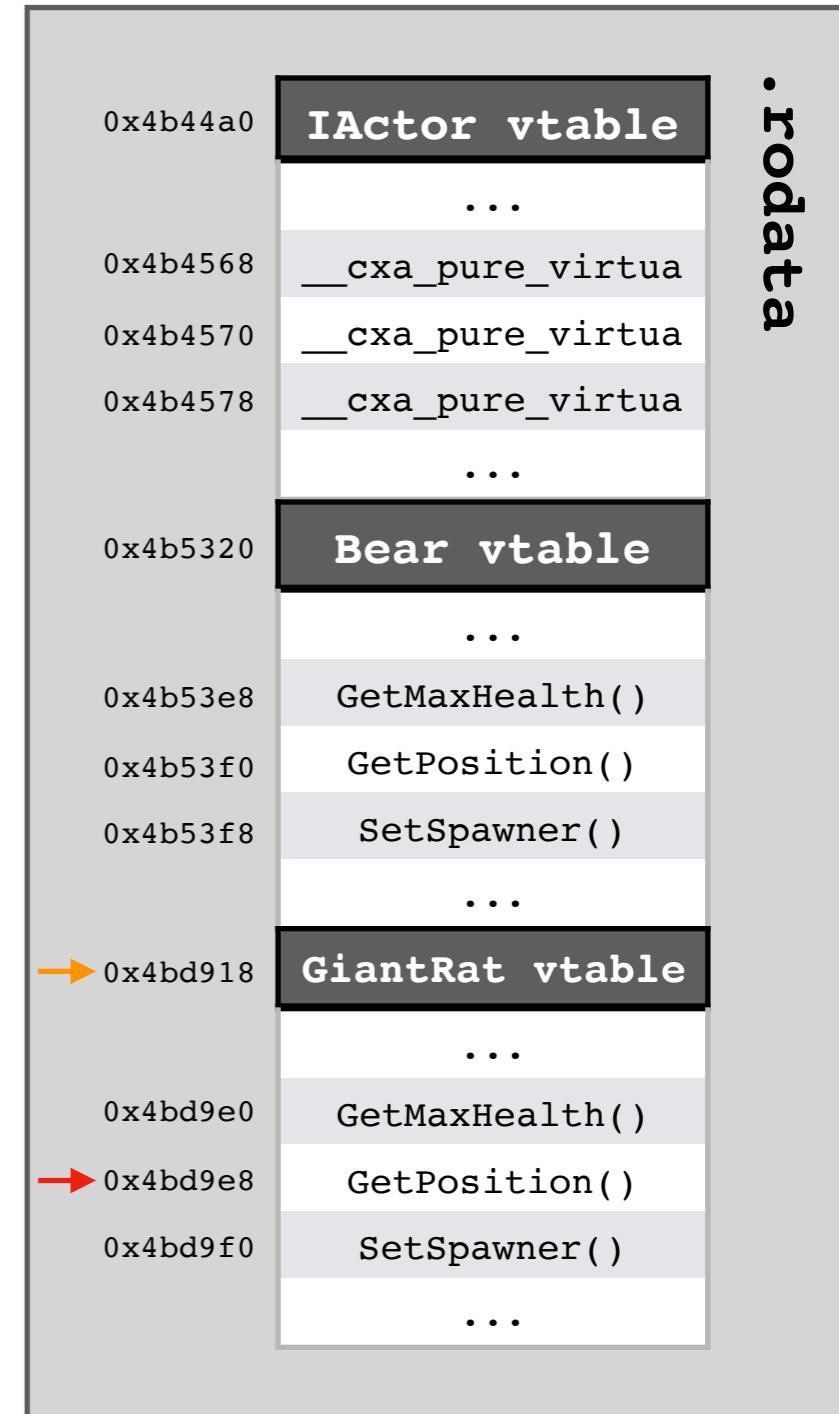
```
; var_1 = GiantRat object
mov    rax, [rbp+var_1]
mov    rcx, [rax]           ; Virtual table pointer
mov    rsi, rbx
mov    rdi, rax
call   qword ptr [rcx+0xd0]
```



Egg hunter

In C++, for each **type** having **virtual function(s)**, the **compiler** create a **table** that contains **function pointers**: the **vtable**

```
; var_1 = GiantRat object
mov    rax, [rbp+var_1]
mov    rcx, [rax]
mov    rsi, rbx
mov    rdi, rax
call   qword ptr [rcx+0xd0]
```



Egg hunter

```

BallmerPeakEgg::CanUse(BallmerPeakEgg *this,
                      IPlayer *player) {

    int ret;
    bool canuse;

    ret = ItemPickup::CanUse(this, player);
    if (ret)
    {
        ret = IPlayer::????(player, "BallmerPeakSecret");
        if (ret)
        {
            canuse = true;
        }
        else
        {
            canuse = false;
        }
    }
    else
    {
        canuse = false;
    }

    return canuse; // We want to return true
}

```

; Attributes: static bp-based frame

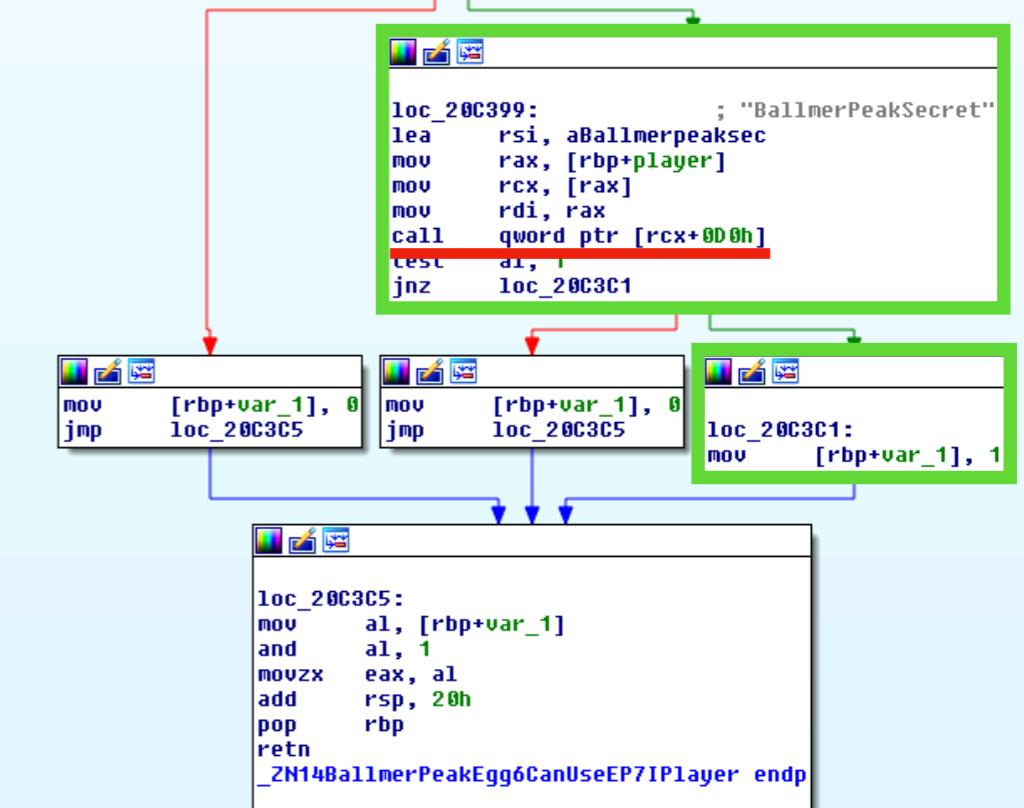
```

; bool __cdecl BallmerPeakEgg::CanUse(BallmerPeakEgg *this, IPlayer *player)
public _ZN14BallmerPeakEgg6CanUseEP7IPlayer
_ZN14BallmerPeakEgg6CanUseEP7IPlayer proc near

var_20= qword ptr -20h
player= qword ptr -18h
this= qword ptr -10h
var_1= byte ptr -1

push    rbp
mov     rbp, rsp
sub    rsp, 20h
mov     [rbp+this], rdi
mov     [rbp+player], rsi
mov     rsi, [rbp+this]
mov     rdi, [rbp+player]
mov     [rbp+var_20], rdi
mov     rdi, rsi      ; this
mov     rsi, [rbp+var_20] ; player
call   _ZN10ItemPickup6CanUseEP7IPlayer ; ItemPickup::CanUse(IPlayer *)
test   al, 1
jnz    loc_20C399

```



Egg hunter

The screenshot shows the IDA Pro interface with the following windows:

- Functions window**: Shows two functions: `IPlayer::IPlayer(void)` and `IPlayer::IPlayer(void)`. The second one is highlighted with a red border.
- IDA View-A**: Displays assembly code for the constructor. It includes comments like "; Attributes: static bp-based frame" and "; void __cdecl IPlayer::IPlayer(IPlayer *this)". The assembly instructions show the setup of a static bp-based frame, pushing rbp, moving rbp to rsp, calculating a pointer to _ZTV7IPlayer, adding 10h to it, moving the result to rdi, and finally moving rdi to rax. The code ends with a ret instruction and a note "; } // starts at 15C4C0".
- Hex View-1**, **Structures**, and **Enums** tabs are also visible at the top.

A red arrow points from the text "Search for IPlayer::IPlayer" to the highlighted function in the Functions window.

Search for IPlayer::IPlayer

```
; Attributes: static bp-based frame  
;  
; void __cdecl IPlayer::IPlayer(IPlayer *this)  
public __ZN7IPlayerC2Ev ; weak  
__ZN7IPlayerC2Ev proc near  
  
    this= qword ptr -8  
  
    ; __ unwind {  
    push    rbp  
    mov     rbp, rsp  
    mov     rax, cs:_ZTV7IPlayer_ptr  
    add     rax, 10h  
    mov     [rbp+this], rdi  
    mov     rdi, [rbp+this]  
    mov     [rdi], rax  
    pop     rbp  
    retn  
    ; } // starts at 15C4C0  
__ZN7IPlayerC2Ev endp
```



Egg hunter

The screenshot shows the IDA Pro interface with two main windows: 'Functions window' on the left and 'IDA View-A' on the right.

Functions window:

- Function name: `IPlayer::IPlayer(void)` (highlighted in yellow)
- Function name: `IPlayer::IPlayer(void)` (highlighted in red)
- Segment: `.plt` (highlighted in yellow)
- Segment: `.text` (highlighted in red)

IDA View-A:

- Assembly code:

 - `; Attributes: static bp-based frame`
 - `; void __cdecl IPlayer::IPlayer(IPlayer *this)`
 - `public _ZN7IPlayerC2Ev ; weak`
 - `_ZN7IPlayerC2Ev proc near`
 - `this= qword ptr -8`
 - `; __unwind {`
 - `push rbp`
 - `mov rbp, rsp`
 - `mov rax, cs:_ZTV7IPlayer_ptr`
 - `add rax, 10h`
 - `mov [rbp+this], rdi`
 - `mov rdi, [rbp+this]`
 - `mov [rdi], rax`
 - `pop rbp`
 - `ret`
 - `; } // starts at 15C4C0`
 - `_ZN7IPlayerC2Ev endp`

Annotations:

- A red arrow points from the text "Search for IPlayer::IPlayer" to the function name `IPlayer::IPlayer(void)` in the Functions window.
- A red arrow points from the text "Select the ref in .text" to the segment `.text` in the Functions window.



Egg hunter

The image shows a screenshot of the IDA Pro debugger interface. On the left, the 'Functions window' displays two entries for the function `IPlayer::IPlayer(void)`. The second entry is highlighted with a red border. A red arrow points from the text 'Search for IPlayer::IPlayer' to the first entry. Another red arrow points from the text 'Select the ref in .text' to the segment column, where '.text' is also highlighted with a red border. On the right, the 'IDA View-A' window shows the assembly code for the constructor. The code is annotated with comments explaining its behavior:

```
; Attributes: static bp-based frame  
;  
; void __cdecl IPlayer::IPlayer(IPlayer *this)  
public _ZN7IPlayerC2Ev ; weak  
_ZN7IPlayerC2Ev proc near  
  
    this= qword ptr -8  
  
    ; __unwind {  
    push    rbp  
    mov     rbp, rsp  
    mov     rax, cs:_ZTV7IPlayer_ptr  
    add     rax, 10h  
    mov     [rbp+this], rdi  
    mov     rdi, [rbp+this]  
    mov     [rdi], rax  
    pop     rbp  
    retn  
    ; } // starts at 15C4C0  
_ZN7IPlayerC2Ev endp
```



Egg hunter

The screenshot shows the IDA Pro interface with two main windows: 'Functions window' on the left and 'IDA View-A' on the right.

Functions window:

- Function name: `IPlayer::IPlayer(void)`
- Function name: `IPlayer::IPlayer(void)`

Segment:

- `.plt`
- `.text`

A red arrow points from the text "Search for IPlayer::IPlayer" to the function names in the Functions window. Another red arrow points from the text "Select the ref in .text" to the segment list. A third red arrow points from the text "ZTV7IPlayer = vtable for IPlayer" to the assembly code in the IDA View-A window.

IDA View-A:

```
; Attributes: static bp-based frame
; void __cdecl IPlayer::IPlayer(IPlayer *this)
public _ZN7IPlayerC2Ev ; weak
_ZN7IPlayerC2Ev proc near

this= qword ptr -8

; __unwind { ZTV7IPlayer = vtable for IPlayer
push    rbp
mov     rbp, rsp
mov     rax, cs:_ZTV7IPlayer_ptr
add     rax, 10h
mov     [rbp+this], rdi
mov     rdi, [rbp+this]
mov     [rdi], rax
pop     rbp
ret
; } // starts at 15C4C0
_ZN7IPlayerC2Ev endp
```



Egg hunter

The screenshot shows the IDA Pro interface with two main windows: 'Functions window' on the left and 'IDA View-A' on the right.

In the 'Functions window', under 'Function name', two entries are listed: `IPlayer::IPlayer(void)` and `IPlayer::IPlayer(void)`. The second entry is highlighted with a yellow background. A red arrow points from the text 'Search for IPlayer::IPlayer' to the first entry. Another red arrow points from the text 'Select the ref in .text' to the second entry.

In the 'Segment' column, the '.text' segment is selected, indicated by a red underline.

In the 'IDA View-A' window, the assembly code for the `IPlayer::IPlayer` constructor is shown:

```
; Attributes: static bp-based frame
; void __cdecl IPlayer::IPlayer(IPlayer *this)
public _ZN7IPlayerC2Ev ; weak
_ZN7IPlayerC2Ev proc near

    this= qword ptr -8

    ; __unwind { _ZTV7IPlayer = vtable for IPlayer
    push    rbp
    mov     rbp, rsp
    mov     rax, cs:_ZTV7IPlayer_ptr
    add     rax, 10h
    mov     [rbp+this], rdi
    mov     rdi, [rbp+this]
    mov     [rdi], rax
    pop     rbp
    retn
; } // starts at 15C4C0
_ZN7IPlayerC2Ev endp
```

A red box highlights the assembly code starting with `_ZTV7IPlayer = vtable for IPlayer`. A red arrow points from the text 'Select the ref in .text' to the `_ZTV7IPlayer` label. Another red arrow points from the text 'Search for IPlayer::IPlayer' to the `IPlayer::IPlayer` constructor entry in the Functions window.



```
• ot:00000000004BEA18 _ZTV10HandCannon_ptr dq offset _ZTV10HandCannon
ot:00000000004BEA18 ; DATA XREF: HandCannon::HandCannon(void)+1F↑r
ot:00000000004BEA18 ; `vtable for 'HandCannon
• ot:00000000004BEA20 _ZTV7IPlayer_ptr dq offset _ZTV7IPlayer ; DATA XREF: IPlayer::IPlayer(void)+4↑r
ot:00000000004BEA20 ; `vtable for 'IPlayer
• ot:00000000004BEA28 _ZTV21FastTravelDestination_ptr dq offset _ZTV21FastTravelDestination
```



```
• ot:00000000004BEA18 _ZTV10HandCannon_ptr dq offset _ZTV10HandCannon
  ot:00000000004BEA18 ; DATA XREF: HandCannon::HandCannon(void)+1F↑r
  ot:00000000004BEA18 ; `vtable for 'HandCannon
• ot:00000000004BEA20 _ZTV7IPlayer_ptr dq offset _ZTV7IPlayer ; DATA XREF: IPlayer::IPlayer(void)+4↑r
  ot:00000000004BEA20 ; `vtable for 'IPlayer
• ot:00000000004BEA28 _ZTV21FastTravelDestination_ptr dq offset _ZTV21FastTravelDestination
```



```
ata.rel.ro:00000000004B5100          public _ZTV7IPlayer ; weak
ata.rel.ro:00000000004B5100 ; `vtable for 'IPlayer
• ata.rel.ro:00000000004B5100 _ZTV7IPlayer    dq 0           ; DATA XREF: LOAD:0000000000017648↑o
ata.rel.ro:00000000004B5100           ; .got:_ZTV7IPlayer_ptr↓o
ata.rel.ro:00000000004B5100           ; offset to this
• ata.rel.ro:00000000004B5108      dq offset _ZTI7IPlayer ; `typeinfo for 'IPlayer
• ata.rel.ro:00000000004B5110      dq offset __cxa_pure_virtual
• ata.rel.ro:00000000004B5118      dq offset __cxa_pure_virtual
• ata.rel.ro:00000000004B5120      dq offset __cxa_pure_virtual
• ata.rel.ro:00000000004B5128      dq offset __cxa_pure_virtual
• ata.rel.ro:00000000004B5130      dq offset __cxa_pure_virtual
• ata.rel.ro:00000000004B5138      dq offset __cxa_pure_virtual
```



```
• ot:00000000004BEA18 _ZTV10HandCannon_ptr dq offset _ZTV10HandCannon
  ot:00000000004BEA18 ; DATA XREF: HandCannon::HandCannon(void)+1F↑r
  ot:00000000004BEA18 ; `vtable for 'HandCannon
• ot:00000000004BEA20 _ZTV7IPlayer_ptr dq offset _ZTV7IPlayer ; DATA XREF: IPlayer::IPlayer(void)+4↑r
  ot:00000000004BEA20 ; `vtable for 'IPlayer
• ot:00000000004BEA28 _ZTV21FastTravelDestination_ptr dq offset _ZTV21FastTravelDestination
```



```
ata.rel.ro:00000000004B5100           public _ZTV7IPlayer ; weak
ata.rel.ro:00000000004B5100 ; `vtable for 'IPlayer
• ata.rel.ro:00000000004B5100 _ZTV7IPlayer    dq 0 ; DATA XREF: LOAD:000000000017648↑o
ata.rel.ro:00000000004B5100           ; .got:_ZTV7IPlayer_ptr↓o
ata.rel.ro:00000000004B5100           ; offset to this
• ata.rel.ro:00000000004B5108        dq offset _ZTI7IPlayer ; `typeinfo for 'IPlayer
• ata.rel.ro:00000000004B5110        dq offset __cxa_pure_virtual
• ata.rel.ro:00000000004B5118        dq offset __cxa_pure_virtual
• ata.rel.ro:00000000004B5120        dq offset __cxa_pure_virtual
• ata.rel.ro:00000000004B5128        dq offset __cxa_pure_virtual
• ata.rel.ro:00000000004B5130        dq offset __cxa_pure_virtual
• ata.rel.ro:00000000004B5138        dq offset __cxa_pure_virtual
```

Function defined
in class IPlayer



Egg hunter

We need to find a **class** that **inherit** from **IPlayer**:

The screenshot shows the IDA Pro interface with several windows open:

- Functions window**: Shows two entries: `IPlayer::IPlayer(void)` and `IPlayer::IPlayer(void)`. The second entry is highlighted with a yellow box and has a red arrow pointing to it from the text "Search for IPlayer::IPlayer".
- IDA View-A**: Shows assembly code:

```
; Attributes: thunk  
;  
; void __cdecl IPlayer::IPlayer(IPlayer *this)  
__ZN7IPlayerC2Ev proc near  
jmp    cs:off_4C7A88  
__ZN7IPlayerC2Ev endp
```
- xrefs to IPlayer::IPlayer(void)**: A dialog box showing one entry:

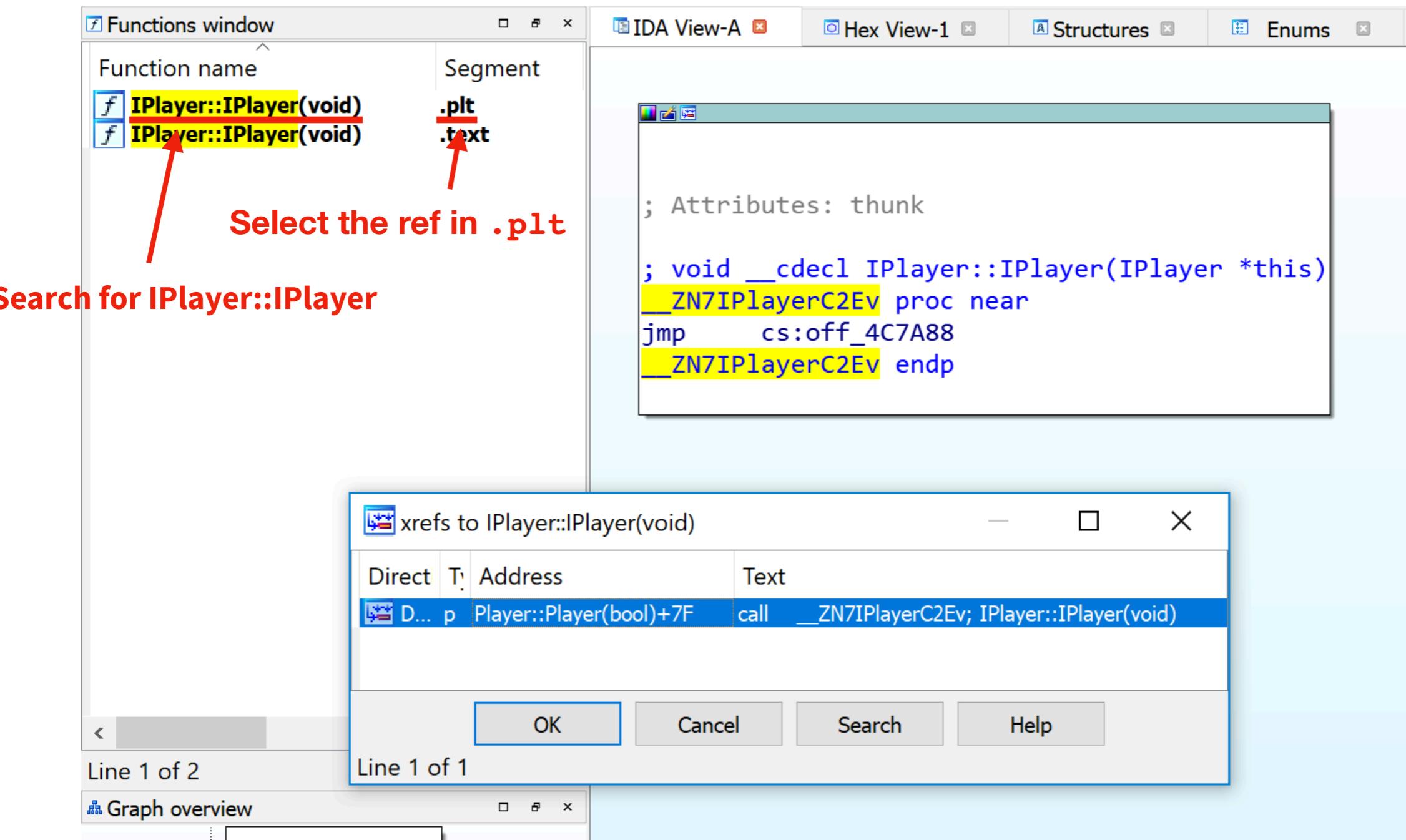
Direct	T	Address	Text
D...	p	Player::Player(bool)+7F	call __ZN7IPlayerC2Ev; IPlayer::IPlayer(void)

Buttons at the bottom: OK, Cancel, Search, Help.
- Hex View-1**, **Structures**, and **Enums** tabs are also visible at the top.



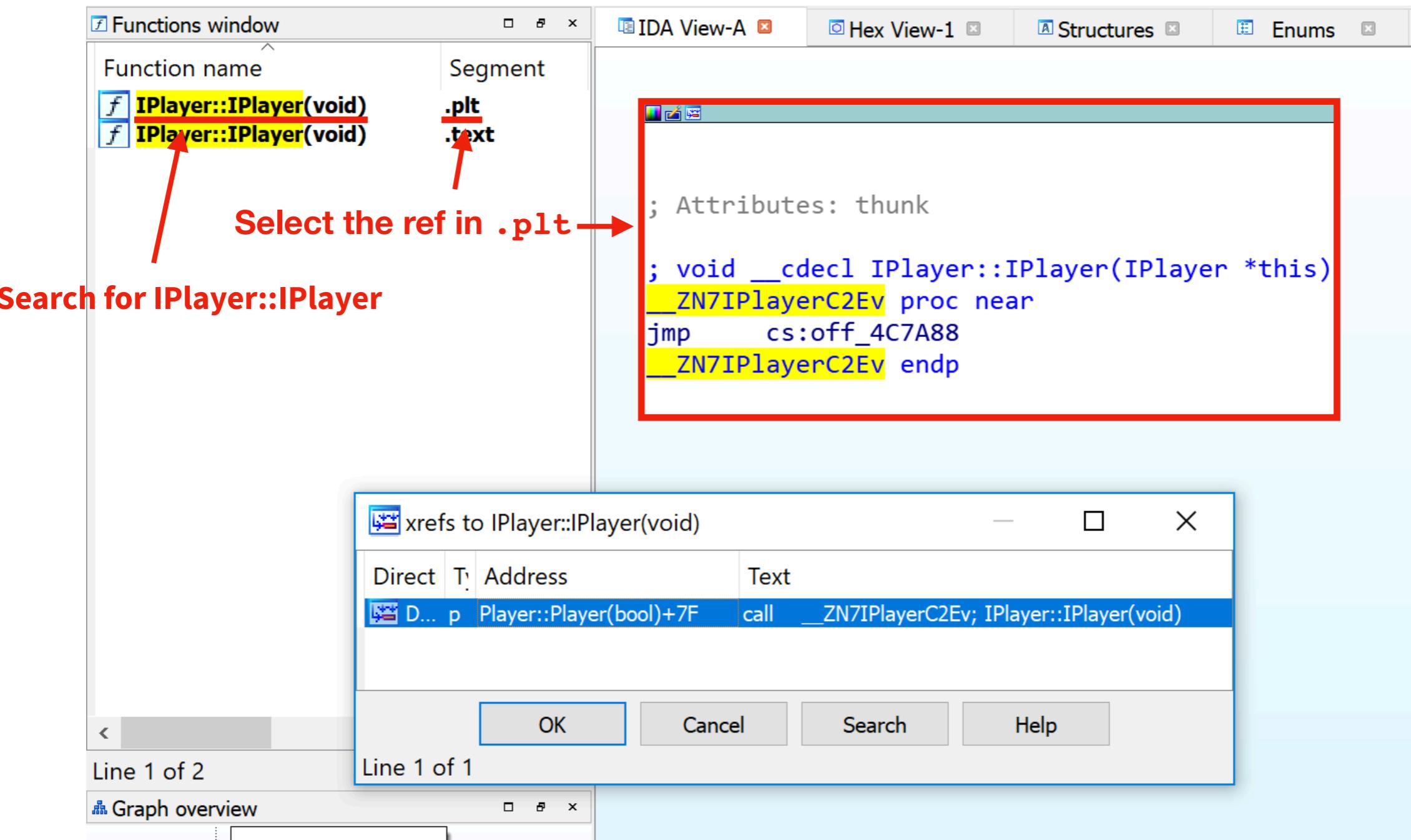
Egg hunter

We need to find a **class** that **inherit** from **IPlayer**:



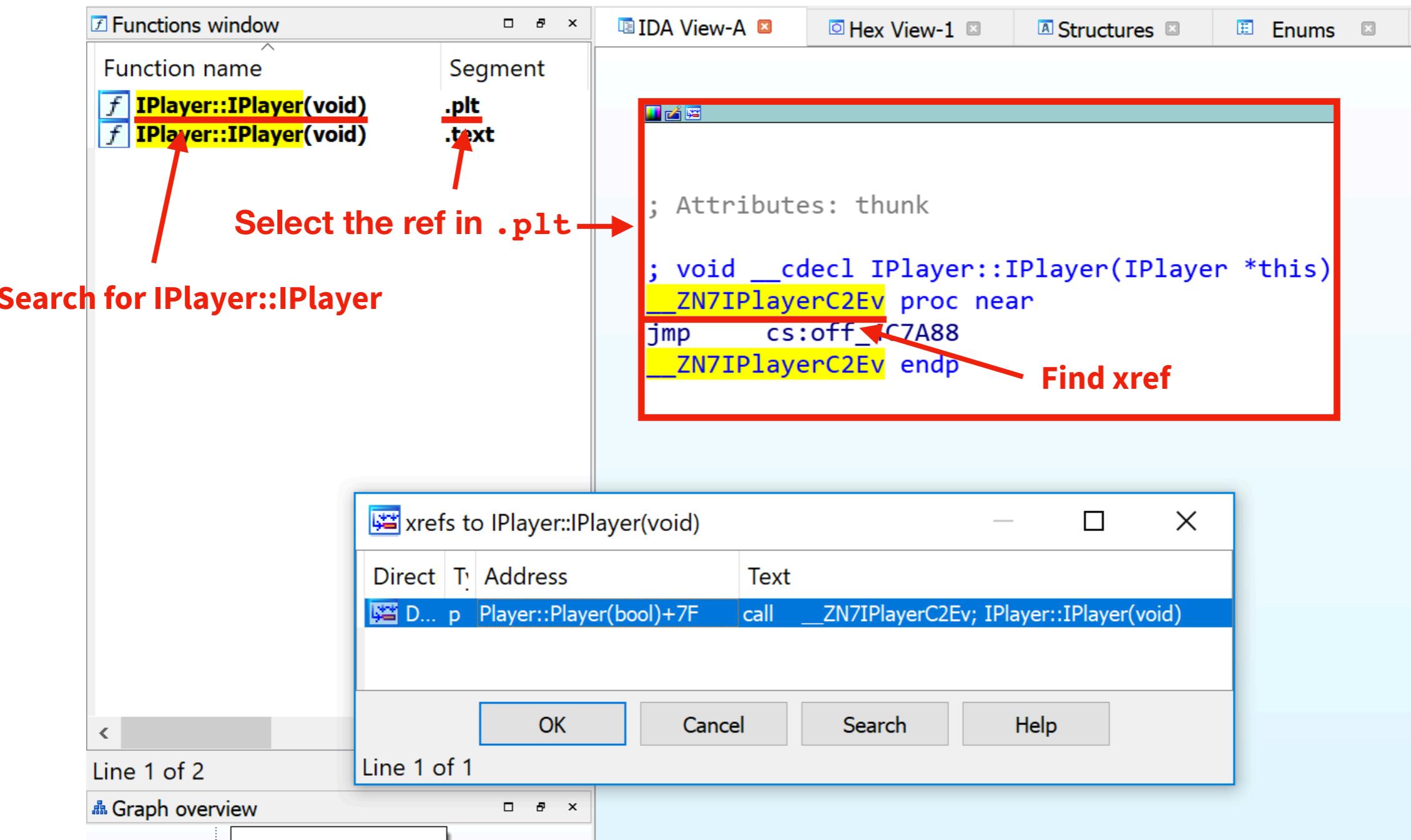
Egg hunter

We need to find a **class** that **inherit** from **IPlayer**:



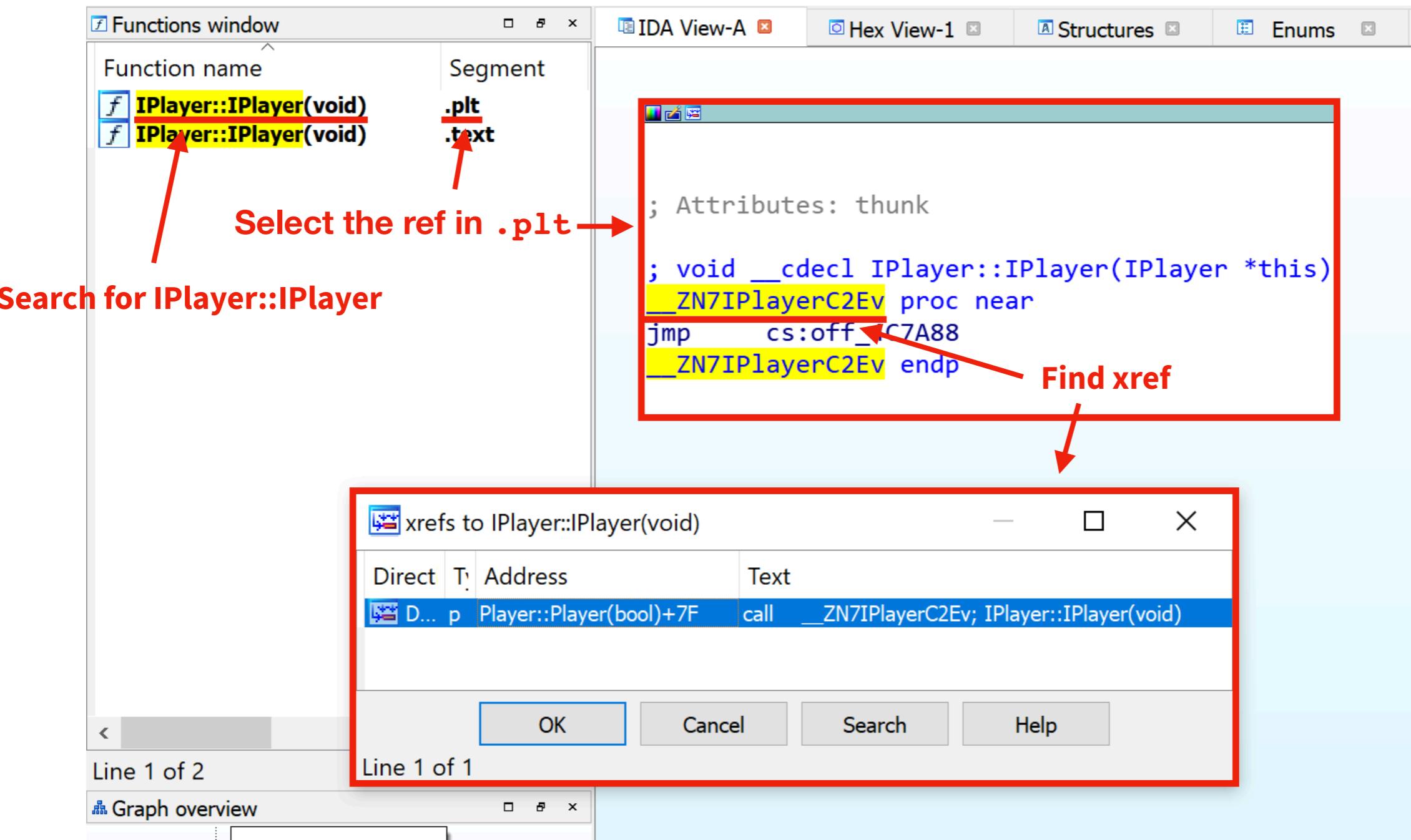
Egg hunter

We need to find a **class** that **inherit** from **IPlayer**:



Egg hunter

We need to find a **class** that **inherit** from **IPlayer**:



Egg hunter

We need to find a **class** that **inherit** from **IPlayer**:



Egg hunter

Inherit

```
loc_14EF42:          ; this
    lea    rdi, [rbp+blueprintName]
    call   __ZNSsD1Ev      ; std::string::~string()
    lea    rdi, [rbp+var_20]
    call   __ZNSaIcED1Ev   ; std::allocator<char>::~allocator()
    mov    rdi, [rbp+var_50]
    add    rdi, 0A8h        ; this
    call   __ZN7IPlayerC2Ev ; IPlayer::IPlayer(void)
    mov    rdi, cs:_ZTV6Player_ptr
    mov    rax, rdi
    add    rax, 10h
    mov    rcx, [rbp+var_50]
    mov    [rcx], rax
    add    rdi, 3B0h
    mov    [rcx+0A8h], rdi
    add    rcx, 0B8h
    ; try {
    mov    rdi, rcx        ; this
    mov    [rbp+var_58], rcx
    call   __ZNSsC1Ev      ; std::string::string(void)
    ; } // starts at 14EF8E
    jmp   $+5
```



Egg hunter

Inherit → call **ZN7IPlayerC2Ev**; IPlayer::IPlayer(void)
vtable for Player → mov rdi, cs:_ZTV6Player_ptr

```
loc_14EF42:          ; this
    lea    rdi, [rbp+blueprintName]
    call   __ZNSsD1Ev      ; std::string::~string()
    lea    rdi, [rbp+var_20]
    call   __ZNSaIcED1Ev   ; std::allocator<char>::~allocator()
    mov    rdi, [rbp+var_50]
    add    rdi, 0A8h        ; this
    call   ZN7IPlayerC2Ev ; IPlayer::IPlayer(void)
    mov    rdi, cs:_ZTV6Player_ptr
    mov    rax, rdi
    add    rax, 10h
    mov    rcx, [rbp+var_50]    vtable for Player
    [rcx], rax
    add    rdi, 3B0h
    mov    [rcx+0A8h], rdi
    add    rcx, 0B8h
; try {
    mov    rdi, rcx        ; this
    mov    [rbp+var_58], rcx
    call   __ZNSsC1Ev      ; std::string::string(void)
; } // starts at 14EF8E
    jmp   $+5
```



Egg hunter

Inherit →

```
loc_14EF42:          ; this
    lea    rdi, [rbp+blueprintName]
    call   __ZNSsD1Ev      ; std::string::~string()
    lea    rdi, [rbp+var_20]
    call   __ZNSaIcED1Ev   ; std::allocator<char>::~allocator()
    mov    rdi, [rbp+var_50]
    add    rdi, 0A8h        ; this
    call   __ZN7IPlayerC2Ev ; IPlayer::IPlayer(void)
    mov    rdi, cs:_ZTV6Player_ptr
    mov    rax, rdi
    add    rax, 10h
    mov    rcx, [rbp+var_50]    vtable for Player
    mov    [rcx], rax
    add    rdi, 3B0h
    mov    [rcx+0A8h], rdi    vtable = vtable + 0x3B0 offset
    add    rcx, 0B8h
; try {
    mov    rdi, rcx        ; this
    mov    [rbp+var_58], rcx
    call   __ZNSsC1Ev      ; std::string::string(void)
; } // starts at 14EF8E
    jmp   $+5
```



```
• 00000000004BECB0 _ZTV16EnemyAttackState_ptr dq offset _ZTV16EnemyAttackState
 00000000004BECB0 ; DATA XREF: EnemyAttackState::EnemyAttackState(AIActor *)+27↑r
 00000000004BECB0 ; `vtable for 'EnemyAttackState'
• 00000000004BECB8 _ZTV6Player_ptr dq offset _ZTV6Player ; DATA XREF: Player::Player(bool)+84↑r
 00000000004BECB8 ; Player::~Player() +F↑r
 00000000004BECB8 ; `vtable for 'Player'
• 00000000004BECC0 _ZTV25HolyHandGrenadeProjectile_ptr dq offset _ZTV25HolyHandGrenadeProjectile
```



```
• 00000000004BECB0 _ZTV16EnemyAttackState_ptr dq offset _ZTV16EnemyAttackState
00000000004BECB0 ; DATA XREF: EnemyAttackState::EnemyAttackState(AIActor *)+27↑r
00000000004BECB0 ; `vtable for'EnemyAttackState
• 00000000004BECB8 _ZTV6Player_ptr dq offset _ZTV6Player ; DATA XREF: Player::Player(bool)+84↑r
00000000004BECB8 ; Player::~Player()+F↑r
00000000004BECB8 ; `vtable for'Player
• 00000000004BECC0 _ZTV25HolyHandGrenadeProjectile_ptr dq offset _ZTV25HolyHandGrenadeProjectile
```



```
.rel.ro:00000000004B4970 public _ZTV6Player
.rel.ro:00000000004B4970 ; `vtable for'Player
• .rel.ro:00000000004B4970 _ZTV6Player dq 0 ; DATA XREF: LOAD:000000000029438↑o
.rel.ro:00000000004B4970 ; .got:_ZTV6Player_ptr↓o
.rel.ro:00000000004B4970 ; offset to this
• .rel.ro:00000000004B4978 dq offset _ZTI6Player ; `typeinfo for'Player
.rel.ro:00000000004B4980 dq offset _ZN6PlayerD2Ev ; Player::~Player()

...
.rel.ro:00000000004B4D18 dq offset _ZTI6Player ; `typeinfo for'Player
.rel.ro:00000000004B4D20 dq offset _ZThn168_N6Player17GetActorInterfaceEv ; `non-virtual thunk to'Player::GetActorInterface(void)
.rel.ro:00000000004B4D28 dq offset _ZThn168_NK6Player13IsLocalPlayerEv ; `non-virtual thunk to'Player::IsLocalPlayer(void)

...
.rel.ro:00000000004B4DE8 dq offset _ZThn168_N6Player15GetItemCooldownEP5IItem ; `non-virtual thunk to'Player::GetItemCooldown(I
.rel.ro:00000000004B4DF0 dq offset _ZThn168_N6Player11HasPickedUpEPKc ; `non-virtual thunk to'Player::HasPickedUp(char const*)
.rel.ro:00000000004B4DF8 dq offset _ZThn168_N6Player14MarkAsPickedUpEPKc ; `non-virtual thunk to'Player::MarkAsPickedUp(char co
```



```

• 0000000004BECB0 _ZTV16EnemyAttackState_ptr dq offset _ZTV16EnemyAttackState
0000000004BECB0 ; DATA XREF: EnemyAttackState::EnemyAttackState(AIActor *)+27↑r
0000000004BECB0 ; `vtable for'EnemyAttackState
• 0000000004BECB8 _ZTV6Player_ptr dq offset _ZTV6Player ; DATA XREF: Player::Player(bool)+84↑r
0000000004BECB8 ; Player::~Player()+F↑r
0000000004BECB8 ; `vtable for'Player
• 0000000004BECC0 _ZTV25HolyHandGrenadeProjectile_ptr dq offset _ZTV25HolyHandGrenadeProjectile

```

+ 0x3b0 offset



```

.rel.ro:0000000004B4970      public _ZTV6Player
.rel.ro:0000000004B4970 ; `vtable for'Player
• .rel.ro:0000000004B4970 _ZTV6Player    dq 0 ; DATA XREF: LOAD:000000000029438↑o
.rel.ro:0000000004B4970 ; .got:_ZTV6Player_ptr↓o
.rel.ro:0000000004B4970 ; offset to this
• .rel.ro:0000000004B4978    dq offset _ZTI6Player ; `typeinfo for'Player
.rel.ro:0000000004B4980    dq offset _ZN6PlayerD2Ev ; Player::~Player()

...
.rel.ro:0000000004B4D18    dq offset _ZTI6Player ; `typeinfo for'Player
• .rel.ro:0000000004B4D20    dq offset _ZThn168_N6Player17GetActorInterfaceEv ; `non-virtual thunk to'Player::GetActorInterface(void)
• .rel.ro:0000000004B4D28    dq offset _ZThn168_NK6Player13IsLocalPlayerEv ; `non-virtual thunk to'Player::IsLocalPlayer(void)

...
.rel.ro:0000000004B4DE8    dq offset _ZThn168_N6Player15GetItemCooldownEP5IItem ; `non-virtual thunk to'Player::GetItemCooldown(I
.rel.ro:0000000004B4DF0    dq offset _ZThn168_N6Player11HasPickedUpEPKc ; `non-virtual thunk to'Player::HasPickedUp(char const*)
.rel.ro:0000000004B4DF8    dq offset _ZThn168_N6Player14MarkAsPickedUpEPKc ; `non-virtual thunk to'Player::MarkAsPickedUp(char co

```



```

• 00000000004BECB0 _ZTV16EnemyAttackState_ptr dq offset _ZTV16EnemyAttackState
00000000004BECB0 ; DATA XREF: EnemyAttackState::EnemyAttackState(AIActor *)+27↑r
00000000004BECB0 ; `vtable for 'EnemyAttackState'
• 00000000004BECB8 _ZTV6Player_ptr dq offset _ZTV6Player ; DATA XREF: Player::Player(bool)+84↑r
00000000004BECB8 ; Player::~Player() +F↑r
00000000004BECB8 ; `vtable for 'Player'
• 00000000004BECC0 _ZTV25HolyHandGrenadeProjectile_ptr dq offset _ZTV25HolyHandGrenadeProjectile

```

+ 0x3b0 offset

```

    mov    rax, [rbp+player]
    mov    rcx, [rax]
    mov    rdi, rax
    call   qword ptr [rcx+0D0h]

```

```

.rel.ro:00000000004B4970      public _ZTV6Player
.rel.ro:00000000004B4970 ; `vtable for 'Player'
• .rel.ro:00000000004B4970 _ZTV6Player    dq 0 ; DATA XREF: LOAD:000000000029438↑o
.rel.ro:00000000004B4970 ; .got:_ZTV6Player_ptr↓o
.rel.ro:00000000004B4970 ; offset to this
• .rel.ro:00000000004B4978      dq offset _ZTI6Player ; `typeinfo for 'Player'
.rel.ro:00000000004B4980      dq offset _ZN6PlayerD2Ev ; Player::~Player()

...
.rel.ro:00000000004B4D18      dq offset _ZTI6Player ; `typeinfo for 'Player'
• .rel.ro:00000000004B4D20      dq offset _ZThn168_N6Player17GetActorInterfaceEv ; non-virtual thunk to 'Player::GetActorInterface(void)'
• .rel.ro:00000000004B4D28      dq offset _ZThn168_NK6Player13IsLocalPlayerEv ; non-virtual thunk to 'Player::IsLocalPlayer(void)'

...
.rel.ro:00000000004B4DE8      dq offset _ZThn168_N6Player15GetItemCooldownEP5IItem ; non-virtual thunk to 'Player::GetItemCooldown(I
.rel.ro:00000000004B4DF0      dq offset _ZThn168_N6Player11HasPickedUpEPKc ; non-virtual thunk to 'Player::HasPickedUp(char const*)
.rel.ro:00000000004B4DF8      dq offset _ZThn168_N6Player14MarkAsPickedUpEPKc ; non-virtual thunk to 'Player::MarkAsPickedUp(char co

```



```

• 00000000004BECB0 _ZTV16EnemyAttackState_ptr dq offset _ZTV16EnemyAttackState
00000000004BECB0 ; DATA XREF: EnemyAttackState::EnemyAttackState(AIActor *)+27↑r
00000000004BECB0 ; `vtable for 'EnemyAttackState'
• 00000000004BECB8 _ZTV6Player_ptr dq offset _ZTV6Player ; DATA XREF: Player::Player(bool)+84↑r
00000000004BECB8 ; Player::~Player() +F↑r
00000000004BECB8 ; `vtable for 'Player'
• 00000000004BECC0 _ZTV25HolyHandGrenadeProjectile_ptr dq offset _ZTV25HolyHandGrenadeProjectile

```

+ 0x3b0 offset

```

    mov    rax, [rbp+player]
    mov    rcx, [rax]
    mov    rdi, rax
    call   qword ptr [rcx+0D0h]

```

```

.rel.ro:00000000004B4970      public _ZTV6Player
.rel.ro:00000000004B4970 ; `vtable for 'Player'
• .rel.ro:00000000004B4970 _ZTV6Player    dq 0 ; DATA XREF: LOAD:000000000029438↑o
.rel.ro:00000000004B4970 ; .got:_ZTV6Player_ptr↓o
.rel.ro:00000000004B4970 ; offset to this
• .rel.ro:00000000004B4978      dq offset _ZTI6Player ; `typeinfo for 'Player'
.rel.ro:00000000004B4980      dq offset _ZN6PlayerD2Ev ; Player::~Player()

...
.rel.ro:00000000004B4D18      dq offset _ZTI6Player ; `typeinfo for 'Player'
• .rel.ro:00000000004B4D20      dq offset _ZThn168_N6Player17GetActorInterfaceEv ; non-virtual thunk to 'Player::GetActorInterface(void)'
• .rel.ro:00000000004B4D28      dq offset _ZThn168_NK6Player13IsLocalPlayerEv ; non-virtual thunk to 'Player::IsLocalPlayer(void)'

...
.rel.ro:00000000004B4DE8      dq offset _ZThn168_N6Player15GetItemCooldownEP5IItem ; non-virtual thunk to 'Player::GetItemCooldown(I
• .rel.ro:00000000004B4DF0      dq offset _ZThn168_N6Player11HasPickedUpEPKc ; non-virtual thunk to 'Player::HasPickedUp(char const*)
• .rel.ro:00000000004B4DF8      dq offset _ZThn168_N6Player14MarkAsPickedUpEPKc ; non-virtual thunk to 'Player::MarkAsPickedUp(char co

```



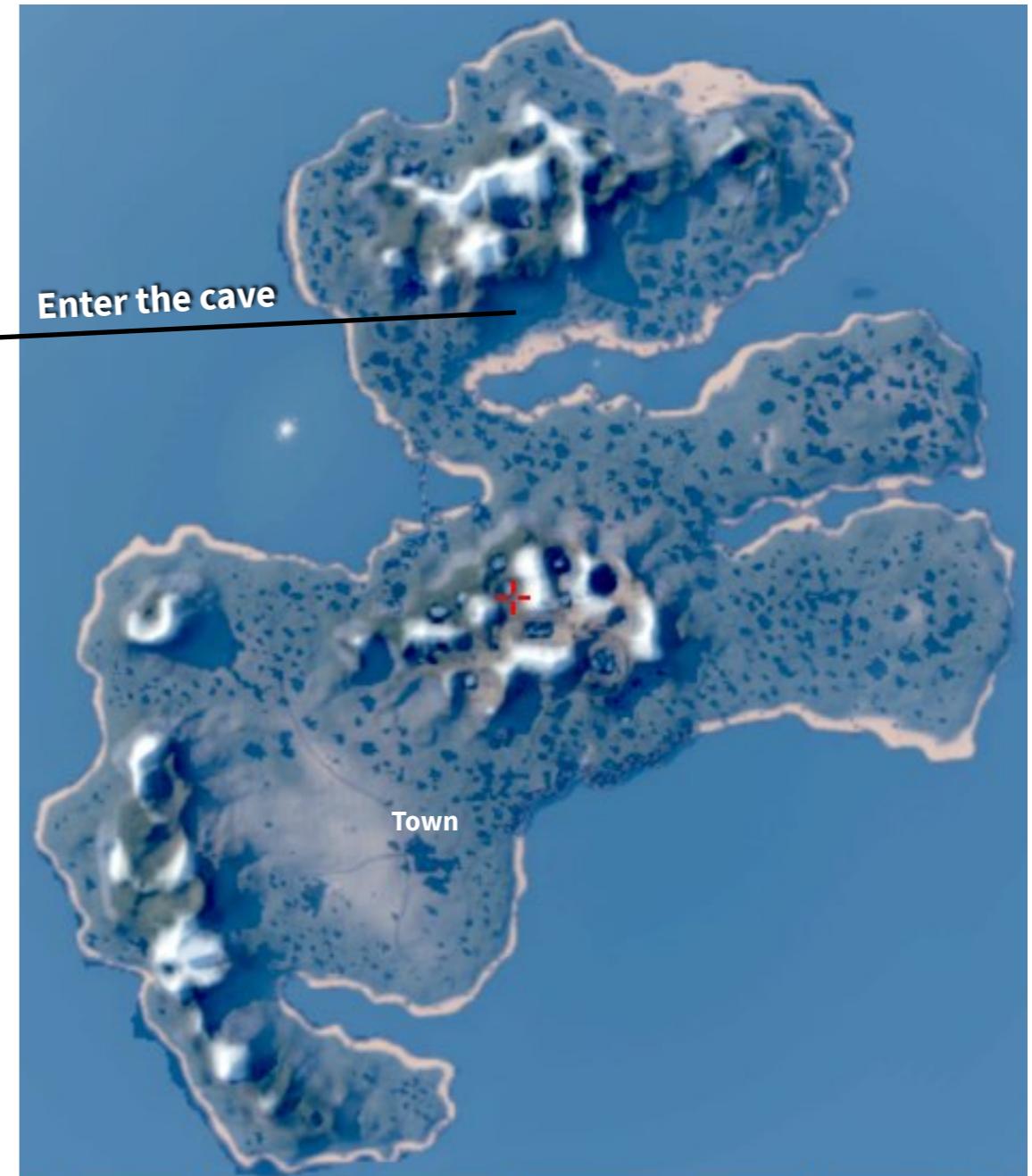
Fire and Ice



Fire and ice

List of quests:

- *Pirates Treasure (crack me)*
- [Fire and Ice \(RE binary\)](#)
- Until the Cows Come Home (binary patching)
- Egg Hunter (RE network & RE binary)
- Unbearable Revenge (RE network)
- *Blockys Revenge (logic gate)*
- *Overachiever (finish all achievements)*



Fire and ice

Where to start?

```
def parse(p_out):  
  
    p_in = ""  
  
    def newspawn(opcode, x, y, z):  
        return opcode + struct.pack("=HfffHHH", 0, x, y, z, 0, 0)  
  
        if len(p_out) == 22 and p_out[2:4] == "\x00\x00" and p_out[16:] ==  
"\\x00\\x00\\x00\\x00\\x00\\x00":  
  
            # Lava Cave location  
            x = 50876.0  
            y = -5243.0  
            z = 1645.0  
  
            p_out = newspawn(p_out[:2], x, y, z)  
  
    return (p_in, p_out)
```



Fire and ice

What to do?

- “Kill Magmarok”



Fire and ice

Demo: Try to kill Magmarok



Fire and ice

What we noticed:

- *Ice* seem to give *more* damage
- Weapon and static links barely give any damage
- *Fire* seem to *heal* Magmarok
- When Magmarok reach *half life*, it start a spell to *regenerate* its entire life bar



Fire and ice

List all functions in Magmarok object:

Function name	Segment	Start
f Magmarok::Damage(IActor *, IItem *, int, DamageType)	.text	0000000000013AF30
f Magmarok::GetAggressionRadius(void)	.text	0000000000013BEE0
f Magmarok::GetAttackTime(void)	.text	0000000000013BF00
f Magmarok::GetDeathMessage(void)	.text	0000000000013BDA0
f Magmarok::GetDisplayName(void)	.text	0000000000013BD60
f Magmarok::GetMaxHealth(void)	.text	0000000000013BD50
f Magmarok::GetMaximumDamageDistance(void)	.text	0000000000013BE10
f Magmarok::GetRangedAttackDistance(void)	.text	0000000000013BE50
f Magmarok::IsElite(void)	.text	0000000000013BD80
f Magmarok::Magmarok(void)	.plt	0000000000010E6B0
f Magmarok::Magmarok(void)	.text	0000000000013ACF0
f Magmarok::OnKilled(IActor *, IItem *)	.text	0000000000013AE50
f Magmarok::OnPrepareAttack(Actor *)	.text	0000000000013B8F0
f Magmarok::ShouldAttack(void)	.text	0000000000013B0B0
f Magmarok::ShouldAttackFromRange(void)	.text	0000000000013BE30
f Magmarok::ShouldAttackMultipleTargets(void)	.text	0000000000013BEBO
f Magmarok::ShouldMove(void)	.text	0000000000013BE90
f Magmarok::ShouldWander(void)	.text	0000000000013BE70
f Magmarok::Tick(float)	.text	0000000000013B0D0
f Magmarok::~Magmarok()	.plt	0000000000011AE00
f Magmarok::~Magmarok()	.plt	00000000000124650
f Magmarok::~Magmarok()	.text	0000000000013BD00
f Magmarok::~Magmarok()	.text	0000000000013BD20
f Magmarok::~Magmarok()	.text	0000000000013BF20



Fire and ice

List all functions in Magmarok object:

Function name	Segment	Start
f Magmarok::Damage(IActor *, IItem *, int, DamageType)	.text	0000000000013AF30
f Magmarok::GetAggressionRadius(void)	.text	0000000000013BEE0
f Magmarok::GetAttackTime(void)	.text	0000000000013BF00
f Magmarok::GetDeathMessage(void)	.text	0000000000013BDA0
f Magmarok::GetDisplayName(void)	.text	0000000000013BD60
f Magmarok::GetMaxHealth(void)	.text	0000000000013BD50
f Magmarok::GetMaximumDamageDistance(void)	.text	0000000000013BE10
f Magmarok::GetRangedAttackDistance(void)	.text	0000000000013BE50
f Magmarok::IsElite(void)	.text	0000000000013BD80
f Magmarok::Magmarok(void)	.plt	0000000000010E6B0
f Magmarok::Magmarok(void)	.text	0000000000013ACF0
f Magmarok::OnKilled(IActor *, IItem *)	.text	0000000000013AE50
f Magmarok::OnPrepareAttack(Actor *)	.text	0000000000013B8F0
f Magmarok::ShouldAttack(void)	.text	0000000000013B0B0
f Magmarok::ShouldAttackFromRange(void)	.text	0000000000013BE30
f Magmarok::ShouldAttackMultipleTargets(void)	.text	0000000000013BEBO
f Magmarok::ShouldMove(void)	.text	0000000000013BE90
f Magmarok::ShouldWander(void)	.text	0000000000013BE70
f Magmarok::Tick(float)	.text	0000000000013B0D0
f Magmarok::~Magmarok()	.plt	0000000000011AE00
f Magmarok::~Magmarok()	.plt	00000000000124650
f Magmarok::~Magmarok()	.text	0000000000013BD00
f Magmarok::~Magmarok()	.text	0000000000013BD20
f Magmarok::~Magmarok()	.text	0000000000013BF20



Fire and ice

Magmarok::Damage pseudo-code:

```
Magmarok::Damage(weaponDmg, weaponType) {  
  
    FIRE = 1  
    ICE = 2  
  
    if (weaponType == FIRE) # 0x13AF4F  
    {  
        healthFactor = magmarok.currentHealth / 10000 # 0x13AF87  
        healthMultiplier = pow(healthFactor, 3) # 0x13AFA2  
        intendedHealing = healthMultiplier * weaponDmg # 0x13AFC0  
        intendedHealing = intendedHealing * 4 # 0x13AFC4  
  
        maximumHealing = 10000 - magmarok.currentHealth # 0x13AFD3  
  
        if (intendedHealing > maximumHealing) # 0x13AFDC  
        {  
            intendedHealing = maximumHealing # 0x13AFE8  
        }  
  
        weaponDmg = 0 - intendedHealing # 0x13AFF3  
    }  
    else  
    {  
        if (weaponType != ICE) # 0x13AFFB  
        {  
            # If not ice nor fire  
            weaponDmg = weaponDmg / 2 # 0x13B019  
        }  
    }  
  
    . . .
```

```
    . . .  
  
    if (magmarok.something == 1) # 0x13B027  
    {  
        if (weaponDmg <= 0) # 0x13B034  
        {  
            if (magmarok.something == 1) # 0x13B060  
            {  
                if (weaponType != ICE) # 0x13B06D  
                {  
                    weaponDmg = weaponDmg * 4 # 0x13B07D  
                }  
            }  
        }  
        else  
        {  
            weaponDmg = weaponDmg / 2 # 0x13B052  
        }  
    }  
  
    damage(magmarok, weaponDmg, weaponType) # 0x13B09E  
}
```



Fire and ice

Damage summary:

- Fire spell will give negative damage (healing) proportionally to Magmarok's health
- Ice spell will give normal damage
- All other weapons/spells will give half damage



Fire and ice

Healing is an interesting feature. If we heal enough Magmarok, we might trigger an *integer overflow* and kill the beast.

Potential **integer overflow**?



Fire and ice

What is an **integer overflow**?

“If the variable has a signed integer type, a program may make the assumption that a variable always contains a positive value. An integer overflow can cause the value to wrap and become negative, which violates the program’s assumption and may lead to unexpected behavior.”



Fire and ice

Signed integer

S = sign

V = value



Fire and ice

Unsigned integer

S = sign

V = value



Fire and ice

Potential **integer overflow**?

Can we throw fire to Magmarok to increase his health until integer overflow?

Does not seem to be possible due to:

```
maximumHealing = 10000 - magmarok.currentHealth  
  
if (intendedHealing > maximumHealing)  
{  
    intendedHealing = maximumHealing  
}
```



Fire and ice

Magmarok::Damage pseudo-code:

```
Magmarok::Damage(weaponDmg, weaponType) {  
  
    FIRE = 1  
    ICE = 2  
  
    if (weaponType == FIRE) # 0x13AF4F  
    {  
        healthFactor = magmarok.currentHealth / 10000    # 0x13AF87  
        healthMultiplier = pow(healthFactor, 3)            # 0x13AFA2  
        intendedHealing = healthMultiplier * weaponDmg    # 0x13AFC0  
        intendedHealing = intendedHealing * 4              # 0x13AFC4  
  
        maximumHealing = 10000 - magmarok.currentHealth # 0x13AFD3  
  
        if (intendedHealing > maximumHealing) # 0x13AFDC  
        {  
            intendedHealing = maximumHealing # 0x13AFE8  
        }  
  
        weaponDmg = 0 - intendedHealing # 0x13AFF3  
    }  
    else  
    {  
        if (weaponType != ICE) # 0x13AFFB  
        {  
            # If not ice nor fire  
            weaponDmg = weaponDmg / 2 # 0x13B019  
        }  
    }  
  
    . . .
```

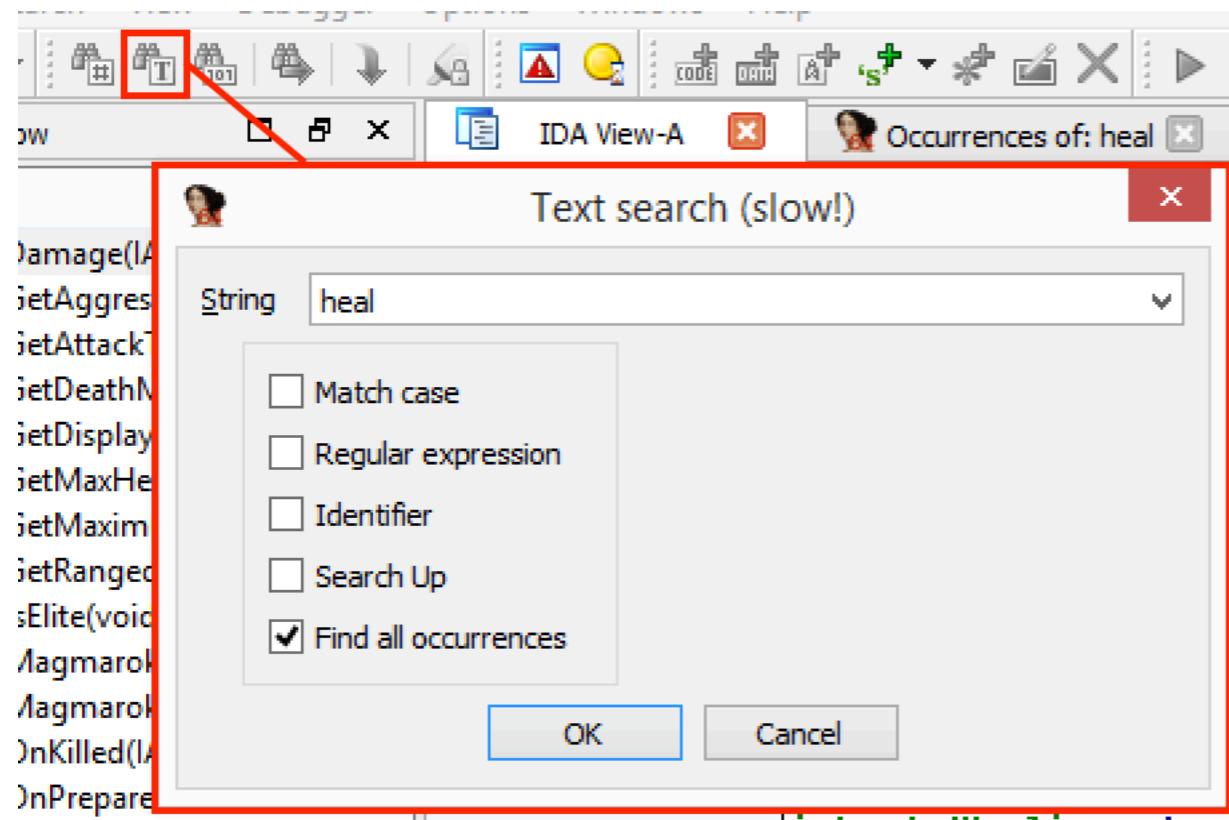
```
    . . .  
  
    if (magmarok.something == 1) # 0x13B027  
    {  
        if (weaponDmg <= 0)      # 0x13B034  
        {  
            if (magmarok.something == 1)    # 0x13B060  
            {  
                if (weaponType != ICE)      # 0x13B06D  
                {  
                    weaponDmg = weaponDmg * 4 # 0x13B07D  
                }  
            }  
        }  
        else  
        {  
            weaponDmg = weaponDmg / 2 # 0x13B052  
        }  
    }  
  
    damage(magmarok, weaponDmg, weaponType) # 0x13B09E  
}
```



Fire and ice

Where is the Magmarok regeneration code?

Let's search for "heal":



Filter for Magmarok:: functions

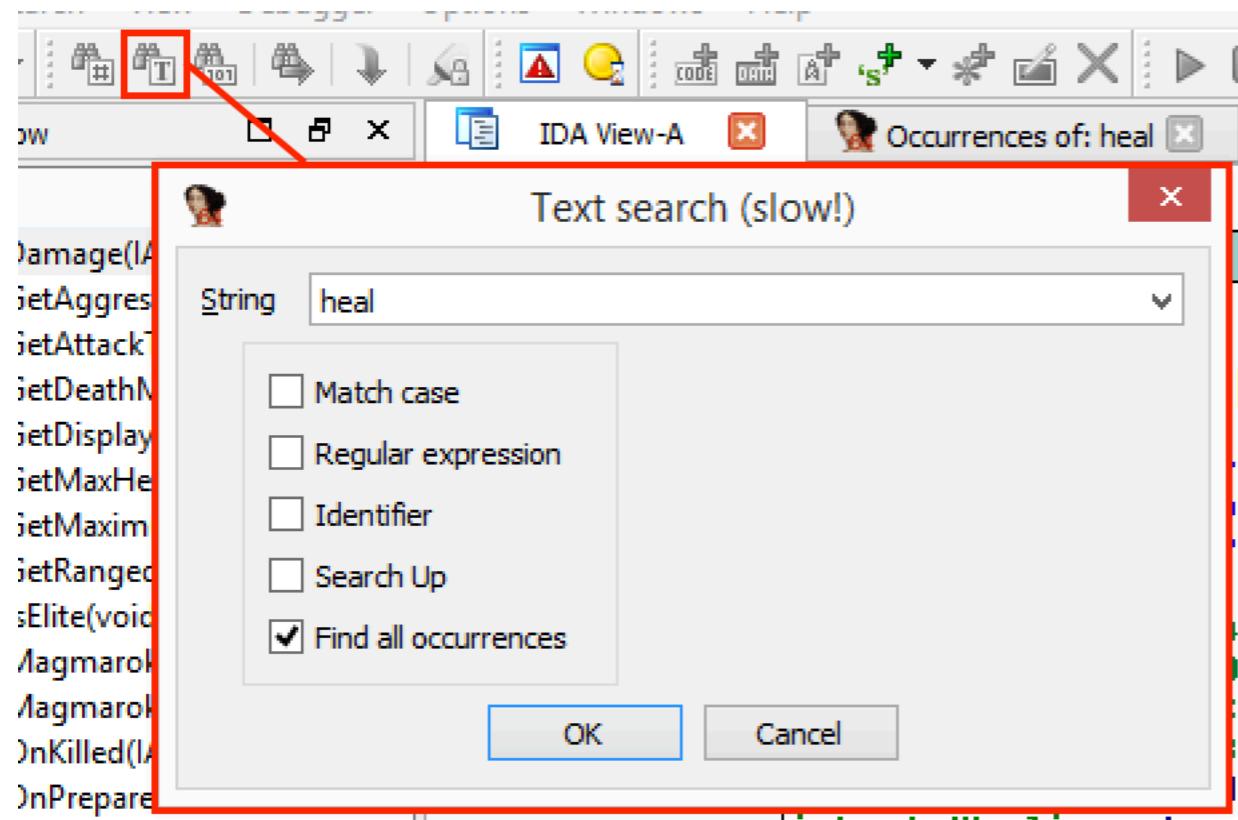
- Magmarok::Damage
- Magmarok::Tick
- Magmarok::GetMaxHealth



Fire and ice

Where is the Magmarok regeneration code?

Let's search for "heal":



Filter for Magmarok:: functions

- Magmarok::Damage
- Magmarok::Tick
- Magmarok::GetMaxHealth



Fire and ice

What is Magmarok::Tick function?

A “real time” function triggered regularly in order to update the state and actions of Magmarok.

```
[...]  
  
if (magmarok.healing == False) # 0x13B122  
{  
    if (magmarok.health > 0)    # 0x13B32F  
    {  
        if (magmarok.health < 5000) # 0x13B343  
        {  
            magmarok.healing = True # 0x13B357  
            updateState(magmarok, "Healing") # 0x13B3CD  
        }  
    }  
}  
else  
{  
    if (magmarok.health > 0) # 0x13B166  
    {  
        newHealth = magmarok.health + 4975    # 0x13B182  
        sendHealthUpdate(magmarok, newHealth) # 0x13B1B5  
        triggerEvent(magmarok, "Heal")       # 0x13B227  
        triggerEvent(magmarok, "Healing")     # 0x13B2A9  
    }  
}  
[...]
```



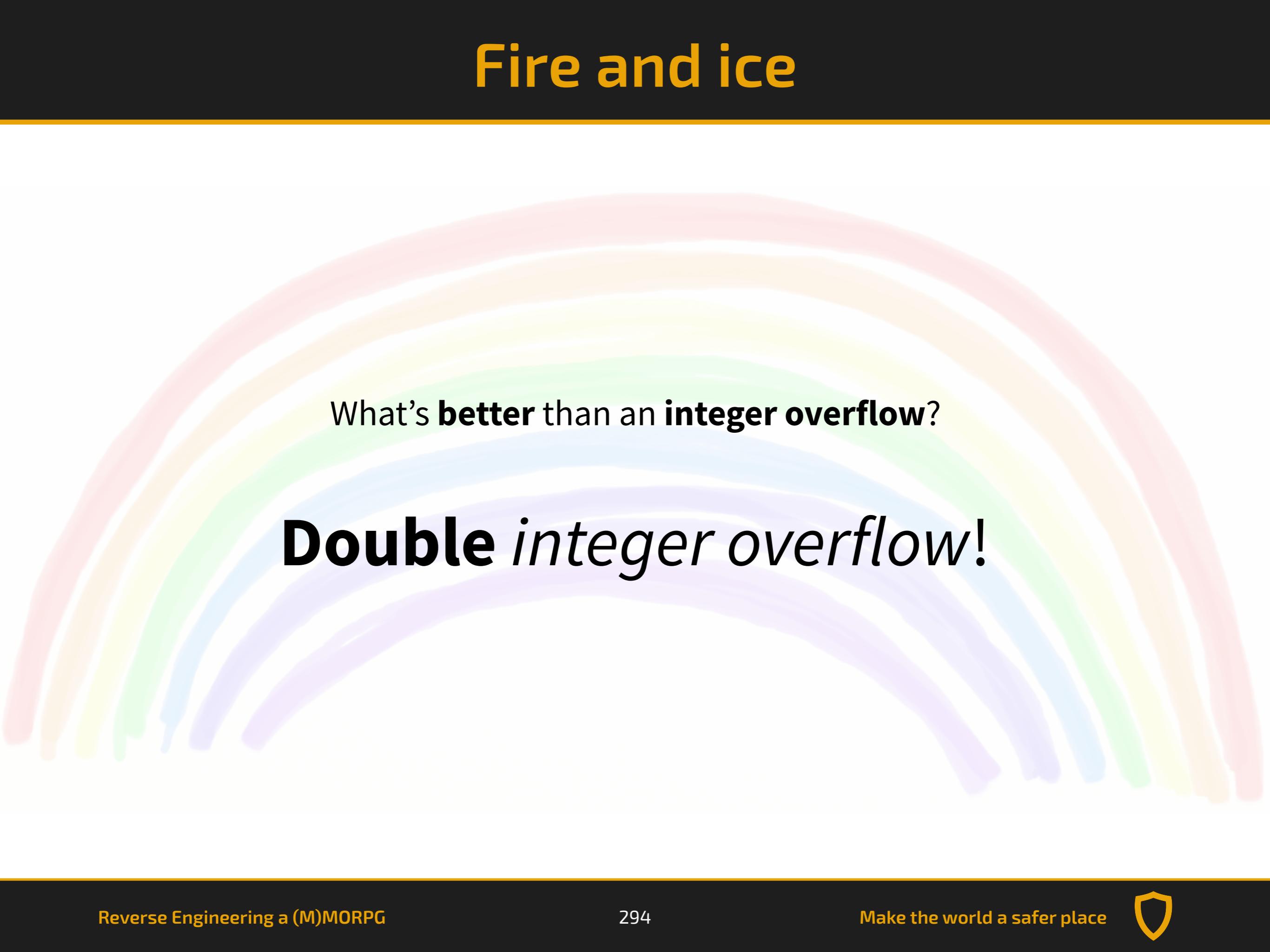
Fire and ice

Tick summary:

- When health < 5000 HP, switch to *healing* mode
- When in *healing* mode, next tick grants 4975 HP
- It takes around 4 seconds for healing



Fire and ice



What's **better** than an **integer overflow**?

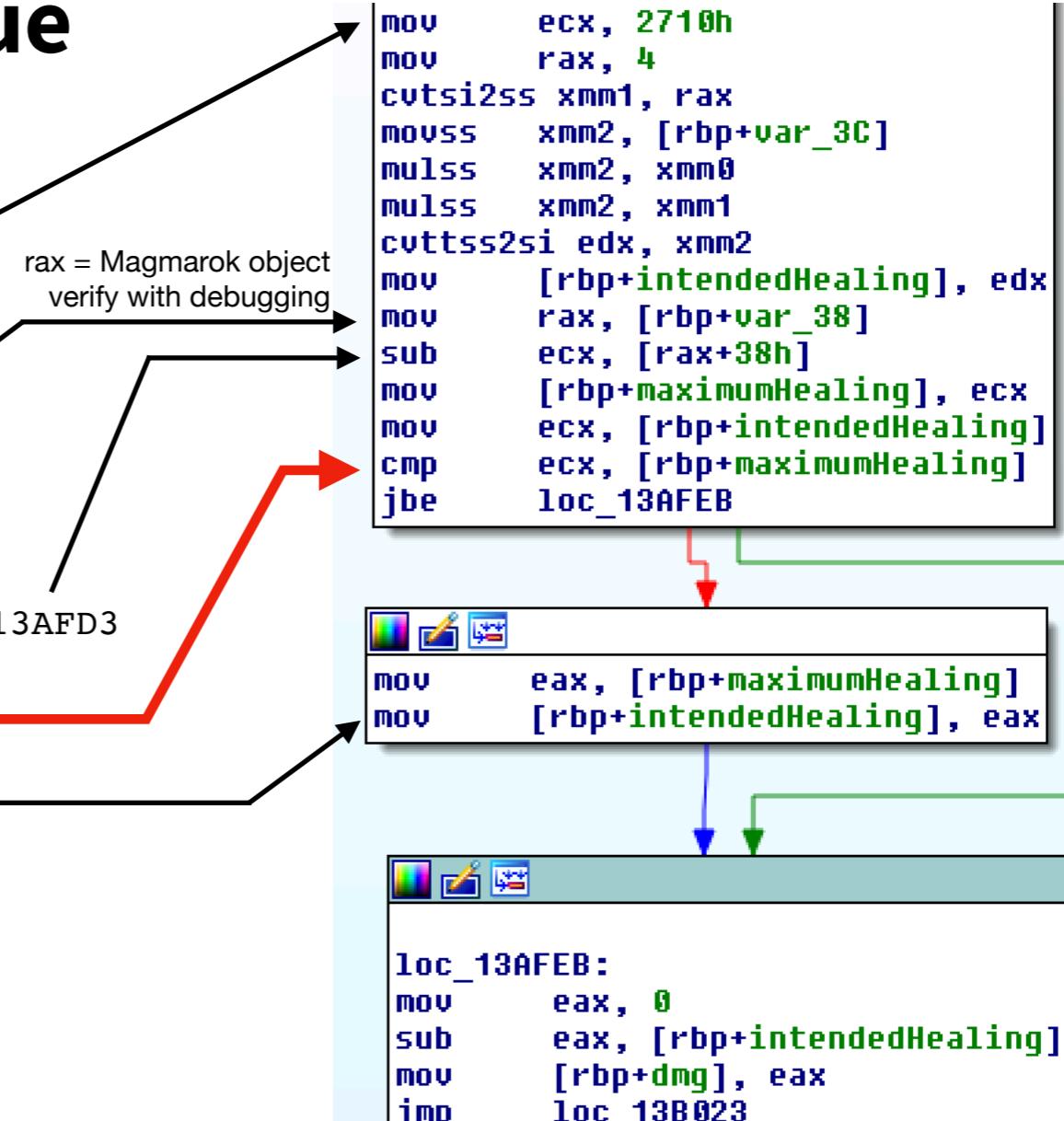
Double integer overflow!



Fire and ice

Let's go back to the **initial issue** that **prevents** the first **integer overflow**:

```
maximumHealing = 10000 - magmarok.currentHealth # 0x13AFD3  
  
if (intendedHealing > maximumHealing) # 0x13AFDC  
{  
    intendedHealing = maximumHealing # 0x13AFE8  
}
```



Fire and ice

Comparison done with JBE:

cmp
jbe
ecx, [rbp+maximumHealing]
loc_13AFEB

Instruction	Description	signed-ness	Flags	short jump opcodes	near jump opcodes
JO	Jump if overflow		OF = 1	70	OF 80
JNO	Jump if not overflow		OF = 0	71	OF 81
JS	Jump if sign		SF = 1	78	OF 88
JNS	Jump if not sign		SF = 0	79	OF 89
JE JZ	Jump if equal Jump if zero		ZF = 1	74	OF 84
JNE JNZ	Jump if not equal Jump if not zero		ZF = 0	75	OF 85
JB JNAE JC	Jump if below Jump if not above or equal Jump if carry	unsigned	CF = 1	72	OF 82
JNB JAE JNC	Jump if not below Jump if above or equal Jump if not carry	unsigned	CF = 0	73	OF 83
JBE JNA	Jump if below or equal Jump if not above	unsigned	CF = 1 or ZF = 1	76	OF 86
JA JNBE	Jump if above Jump if not below or equal	unsigned	CF = 0 and ZF = 0	77	OF 87



Fire and ice

What if magmarok.currentHealth is higher than 10000?

```
maximumHealing = 10000 - magmarok.currentHealth
```

```
if (intendedHealing > maximumHealing)
{
    intendedHealing = maximumHealing
}
```

`maximumHealing` should be **negative** but will be interpreted as a **huge number**:

E.g. if `maximumHealing` = 18446744073709551615, the `if()` condition `intendedHealing > maximumHealing` will not be `true`



Fire and ice

This means as soon as Magmarok **health is higher than 10000**, we will be able to **increase** its health **infinitely**.

How to get `magmarok.health > 10000`?

```
if (magmarok.health < 5000):
    Heal for 4 seconds
    magmarok.health += 4975
```

← Magmarok::Tick

So we just need to **drop health** right **below 5000HP**, then we will have **4 seconds** to **increase** the **health** to **5026HP** (or more) so that when it **regenerate** its **health** by **4975HP**, the final health will be **10001HP** (or more).



Fire and ice

We don't have much **time** (4 seconds) to get the **health > 5000HP**. This means we need to be **accurate** and be right below 5000HP.

We will use the **proxy** to indicate **Magmarok's health**:

```
def parse(p_out):  
  
    p_in = ""  
  
    if "++" in p_out:  
        posi = p_out.index("++")  
        actor,health = struct.unpack("ii", p_out[posi+2 : posi+10])  
        print "HEALTH: " + str(actor) + " - " + str(health) + "hp"  
  
    return (p_out, p_in)
```



Fire and ice

Demo: Let's kill Magmarok



Library hooking



```
; Attributes: static bp-based frame
; void __cdecl Player::Chat(Player *this, const char *text)
public _ZN6Player4ChatEPKc
_ZN6Player4ChatEPKc proc near

var_58      = qword ptr -58h
var_50      = qword ptr -50h
var_48      = qword ptr -48h
var_40      = qword ptr -40h
var_38      = qword ptr -38h
var_20      = byte ptr -20h
var_18      = std::string ptr -18h
text        = qword ptr -10h
this        = qword ptr -8

push    rbp
mov     rbp, rsp
sub    rsp, 60h
mov     [rbp+this], rdi
mov     [rbp+text], rsi
mov     rdi, [rbp+this]
mov     rax, cs:GameWorld_ptr
mov     rax, [rax]
mov     rcx, [rax]
mov     rcx, [rcx+0A0h]
lea     rdx, [rbp+var_20]
mov     [rbp+var_38], rdi
mov     rdi, rdx
mov     [rbp+var_40], rsi
mov     [rbp+var_48], rdx
mov     [rbp+var_50], rax
mov     [rbp+var_58], rcx
call   __ZNSaIcEC1Ev ; std::allocator<char>::allocator(void)
lea     rdi, [rbp+var_18]
mov     rsi, [rbp+var_40]
mov     rdx, [rbp+var_48]
call   __ZNSsC1EPKcRKSaIcE ; std::string::string(char const*,std::allocator<char> const&)
jmp   $+5

; Attributes: static bp-based frame
; void __cdecl Player::Teleport(Player *this, const char *name)
public _ZN6Player8TeleportEPKc
_ZN6Player8TeleportEPKc proc near

var_58= qword ptr -58h
var_50= qword ptr -50h
var_48= qword ptr -48h
var_40= qword ptr -40h
var_38= qword ptr -38h
var_20= byte ptr -20h
var_18= std::string ptr -18h
name= qword ptr -10h
this= qword ptr -8

push    rbp
mov     rbp, rsp
sub    rsp, 60h
mov     [rbp+this], rdi
mov     [rbp+name], rsi
mov     rdi, [rbp+this]
mov     rax, cs:GameWorld_ptr
mov     rax, [rax]
mov     rcx, [rax]
mov     rcx, [rcx+98h]
lea     rdx, [rbp+var_20]
mov     [rbp+var_38], rdi
mov     rdi, rdx
mov     [rbp+var_40], rsi
mov     [rbp+var_48], rdx
mov     [rbp+var_50], rax
mov     [rbp+var_58], rcx
call   __ZNSaIcEC1Ev ; std::allocator<char>::allocator(void)
lea     rdi, [rbp+var_18]
mov     rsi, [rbp+var_40]
mov     rdx, [rbp+var_48]
call   __ZNSsC1EPKcRKSaIcE ; std::string::string(char const*,std::allocator<char> const&)
jmp   $+5
```



Library hooking

Binary patching:

- Edit binary to modify the instructions in our advantage
- Small changes: overwrite few instructions
- If we want to add more complex logic:
 - Code cave
 - Hooking



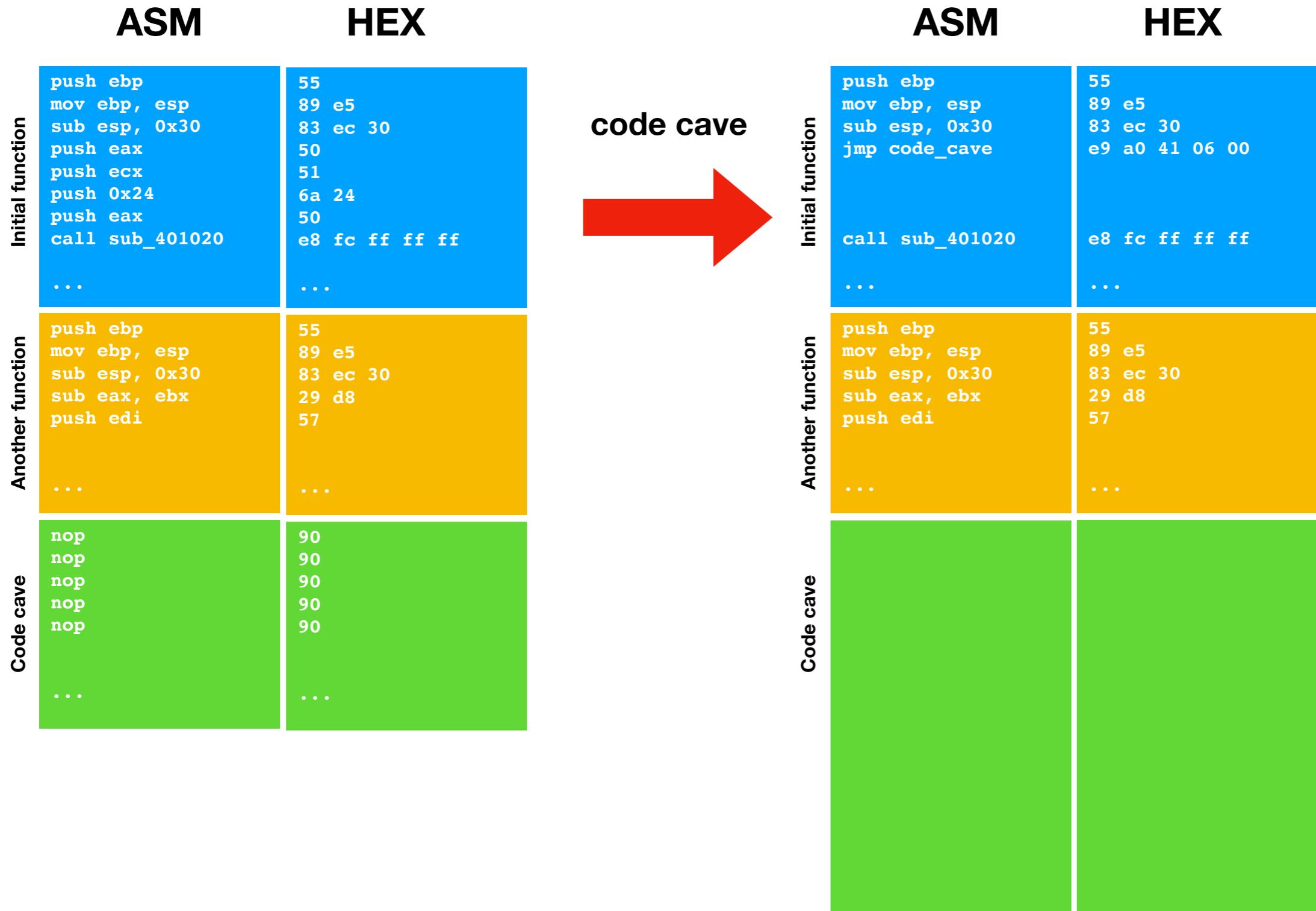
Library hooking

Code cave technique:

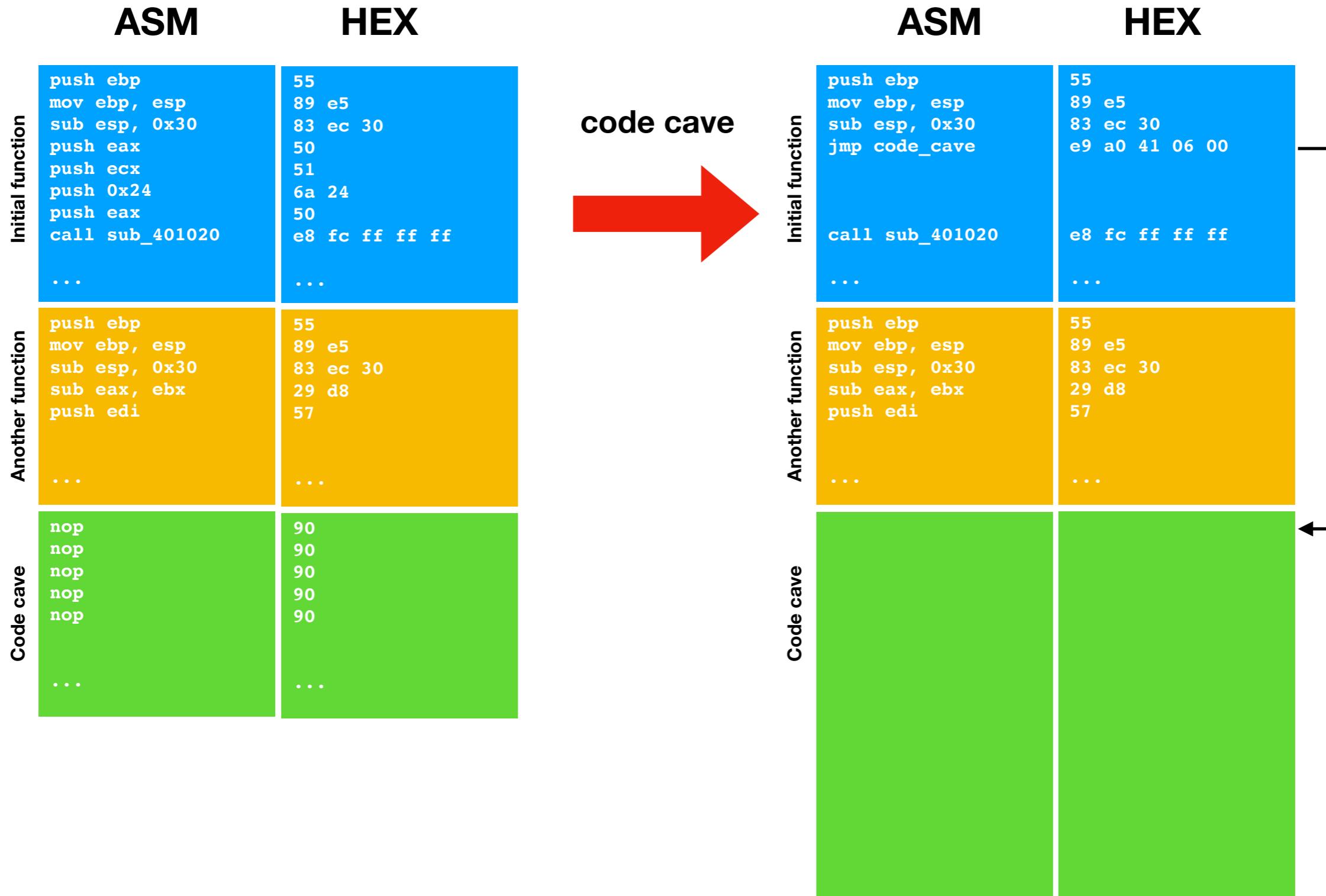
- Find an area in the binary that is not used (cave)
- Jump to that cave (overwrite instruction)
- Add overwritten instruction(s) at the beginning of the cave
- Save the registers
- Add new code (assembly instruction) in it
- Re-align stack and reset the registers at the end of the code cave
- Once reset, add a jump to come back in the initial function (right after the new jump)



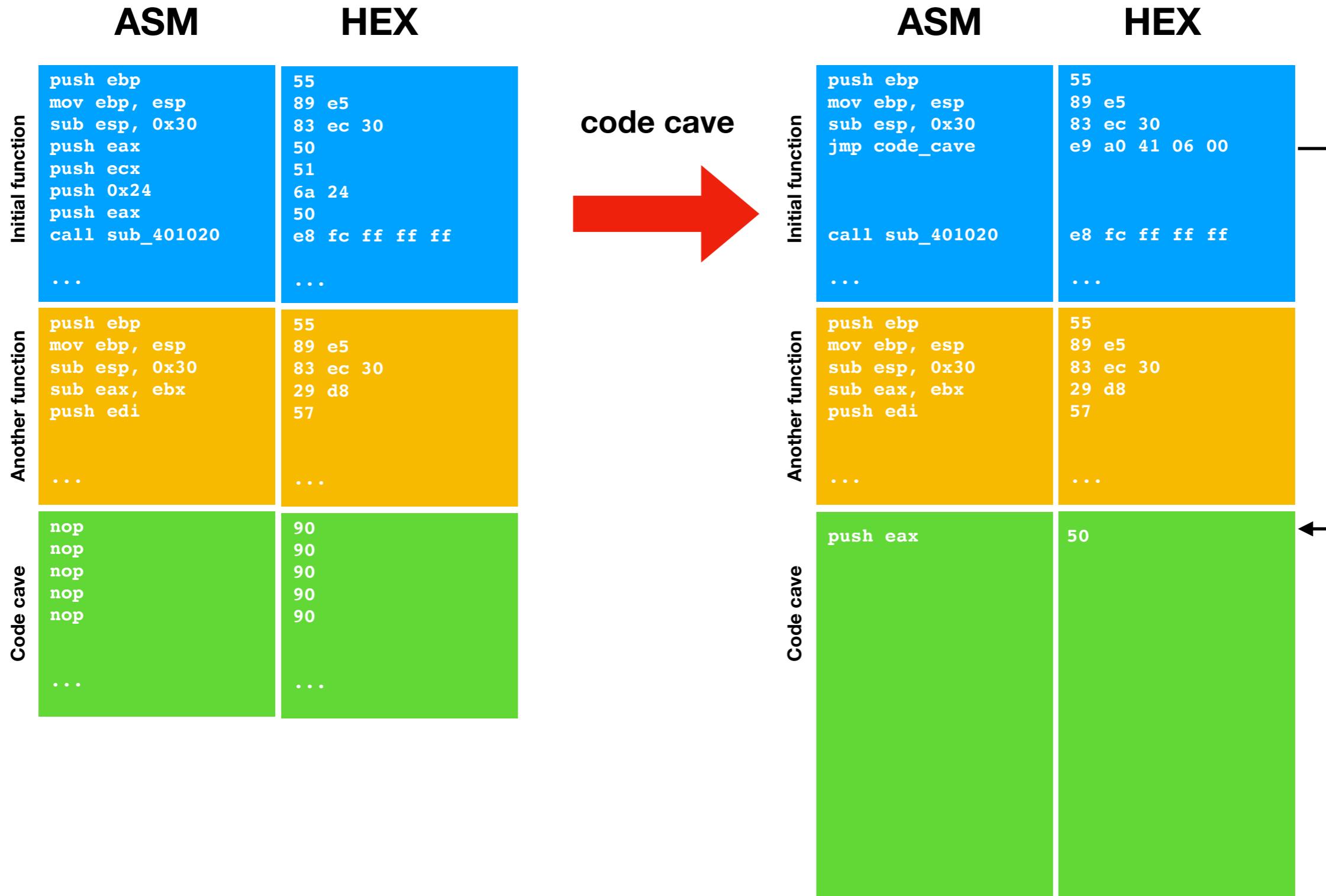
Library hooking



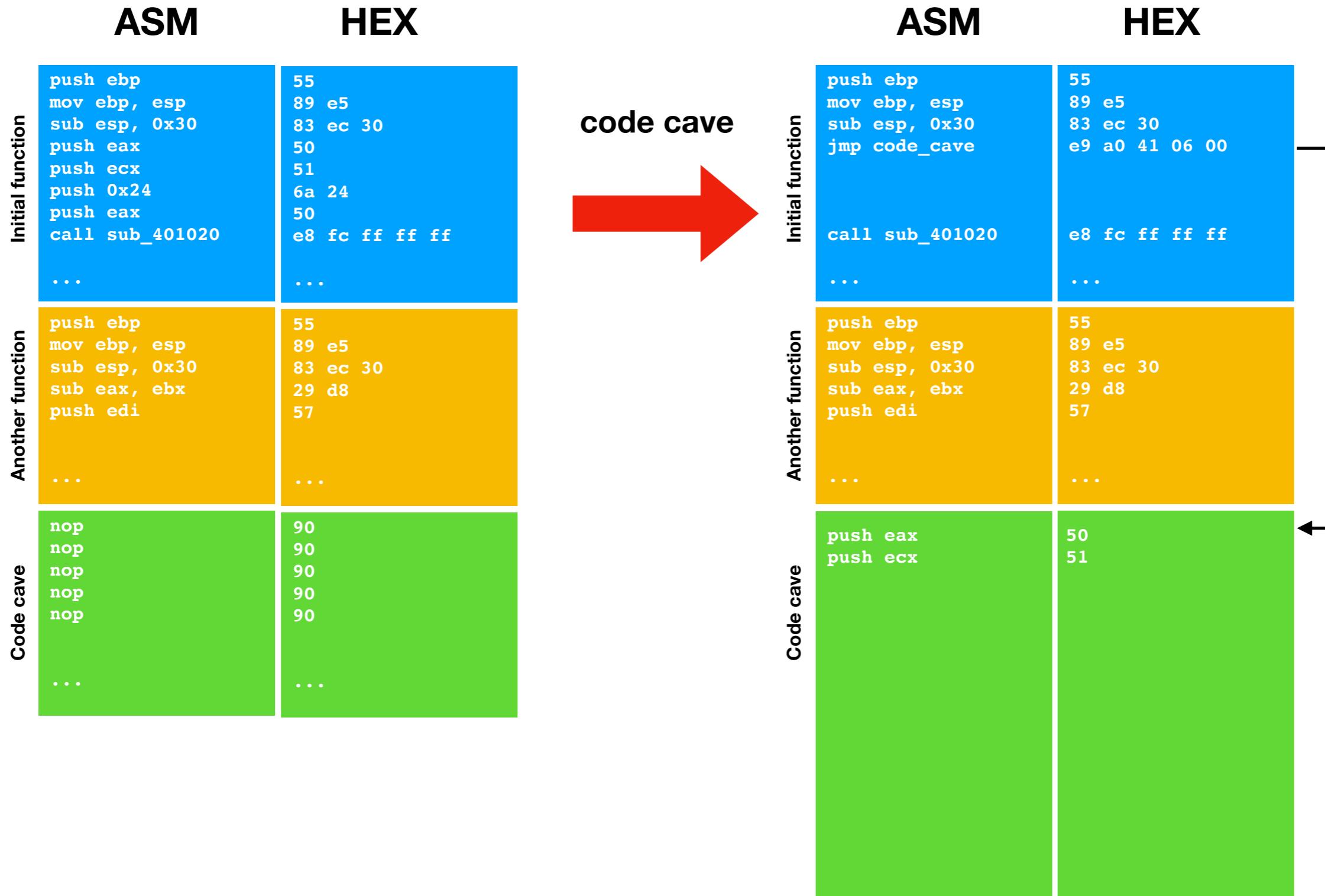
Library hooking



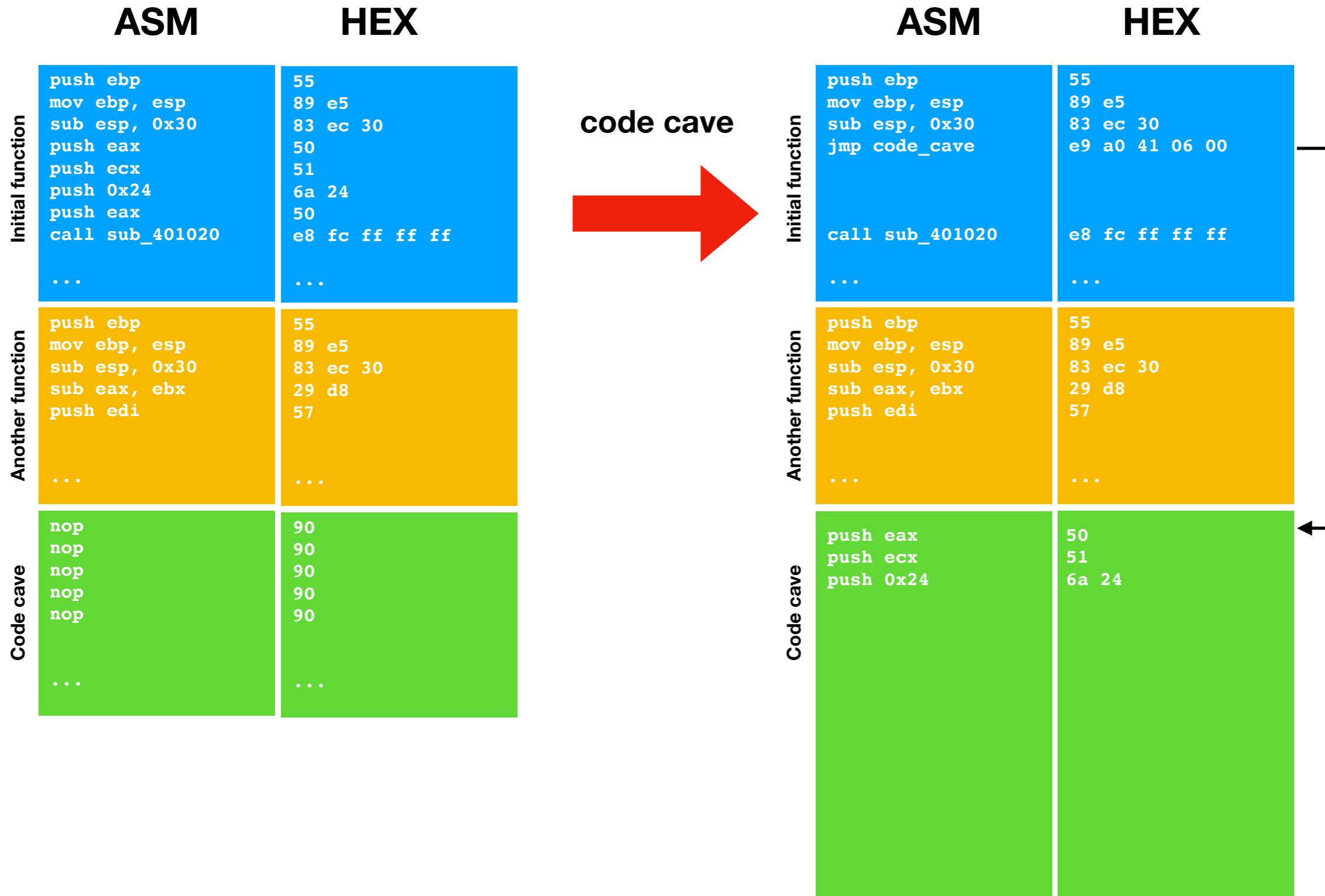
Library hooking



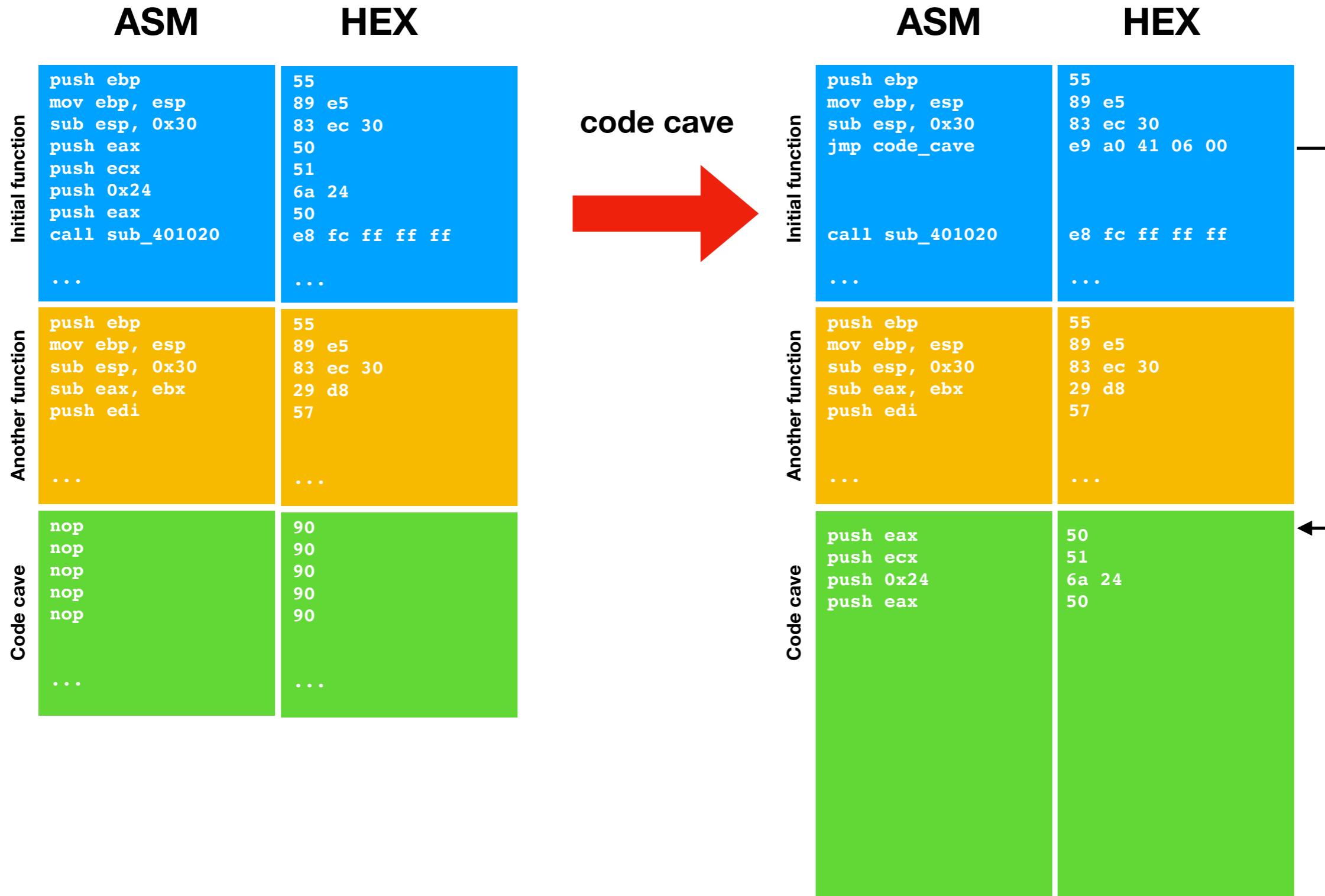
Library hooking



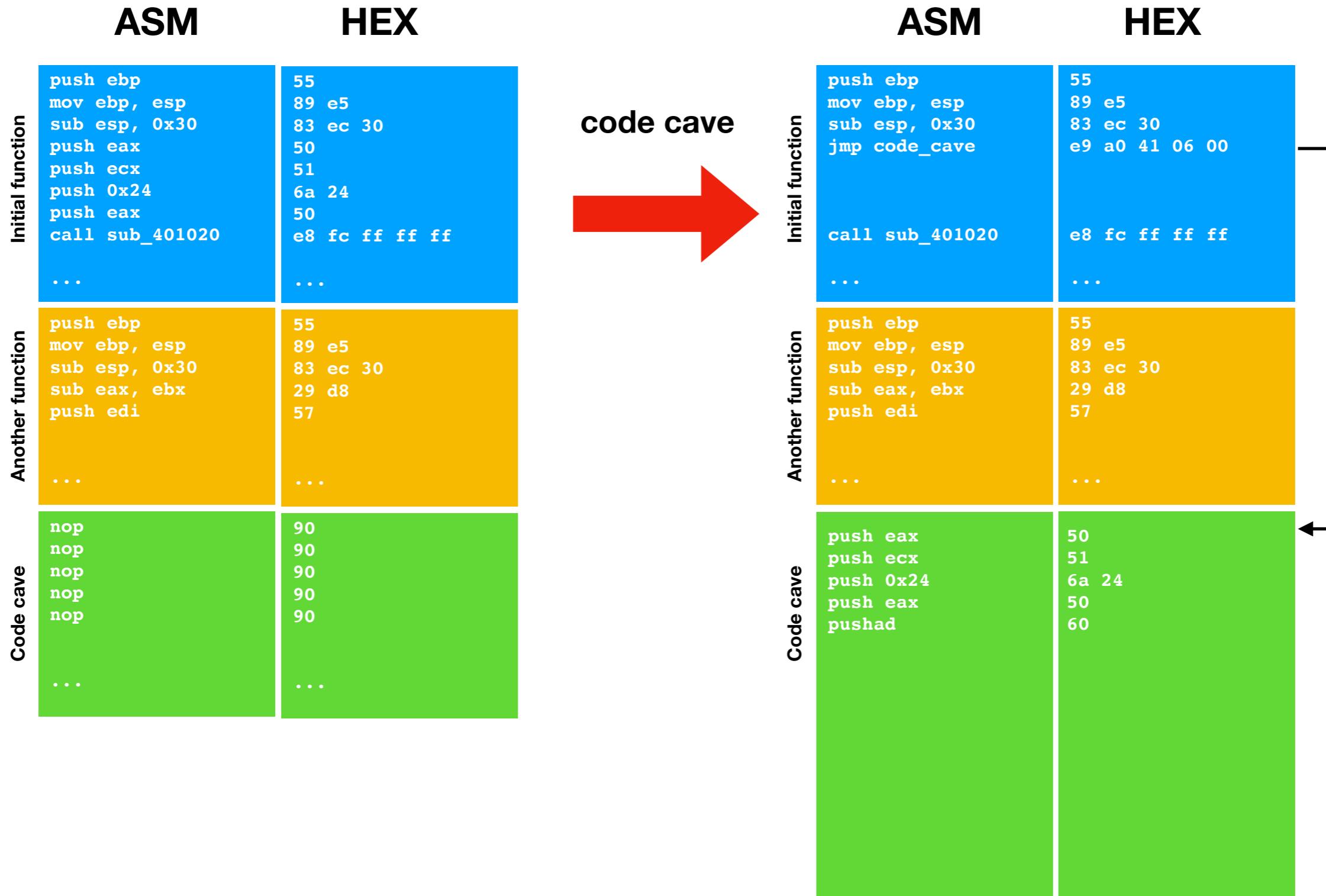
Library hooking



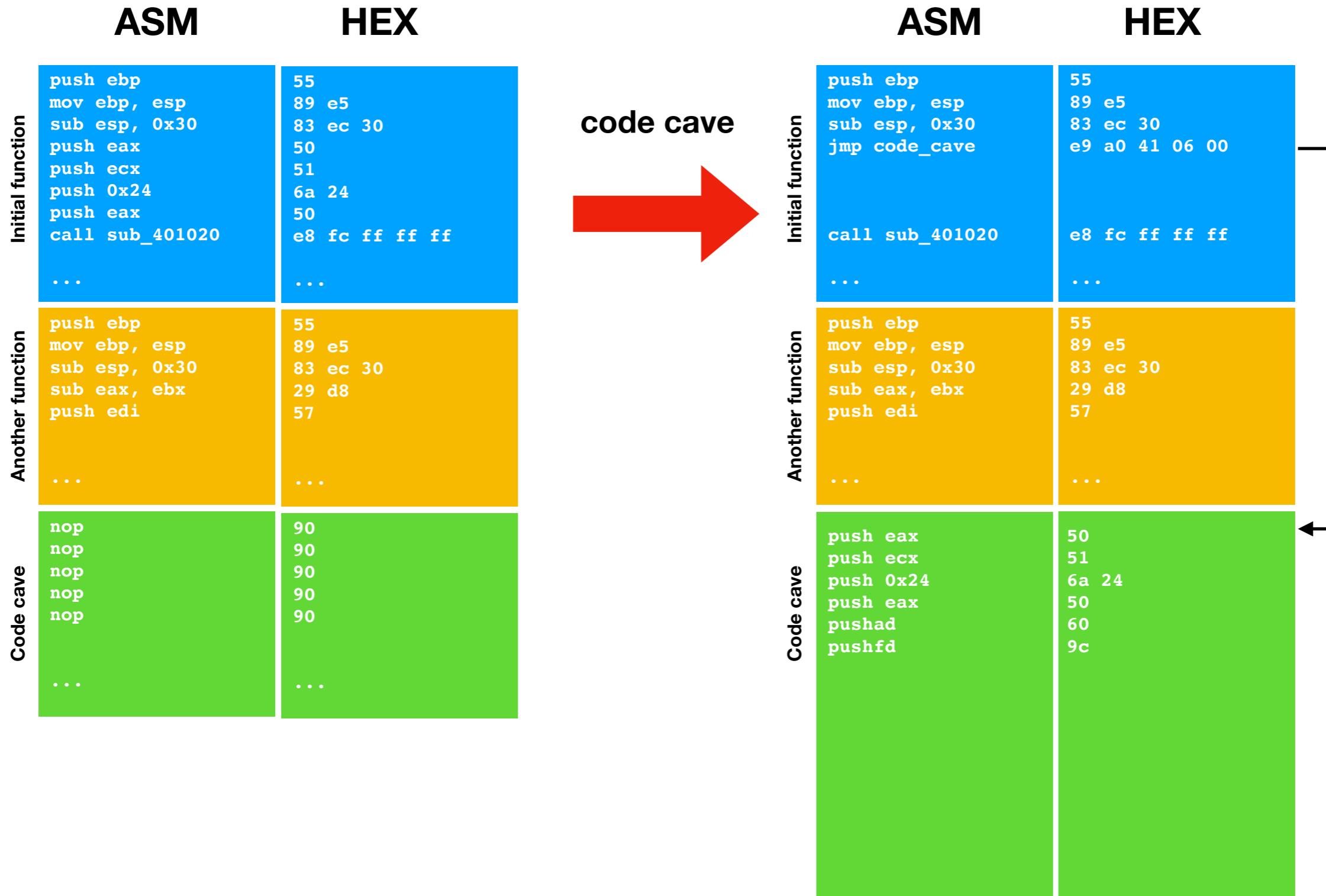
Library hooking



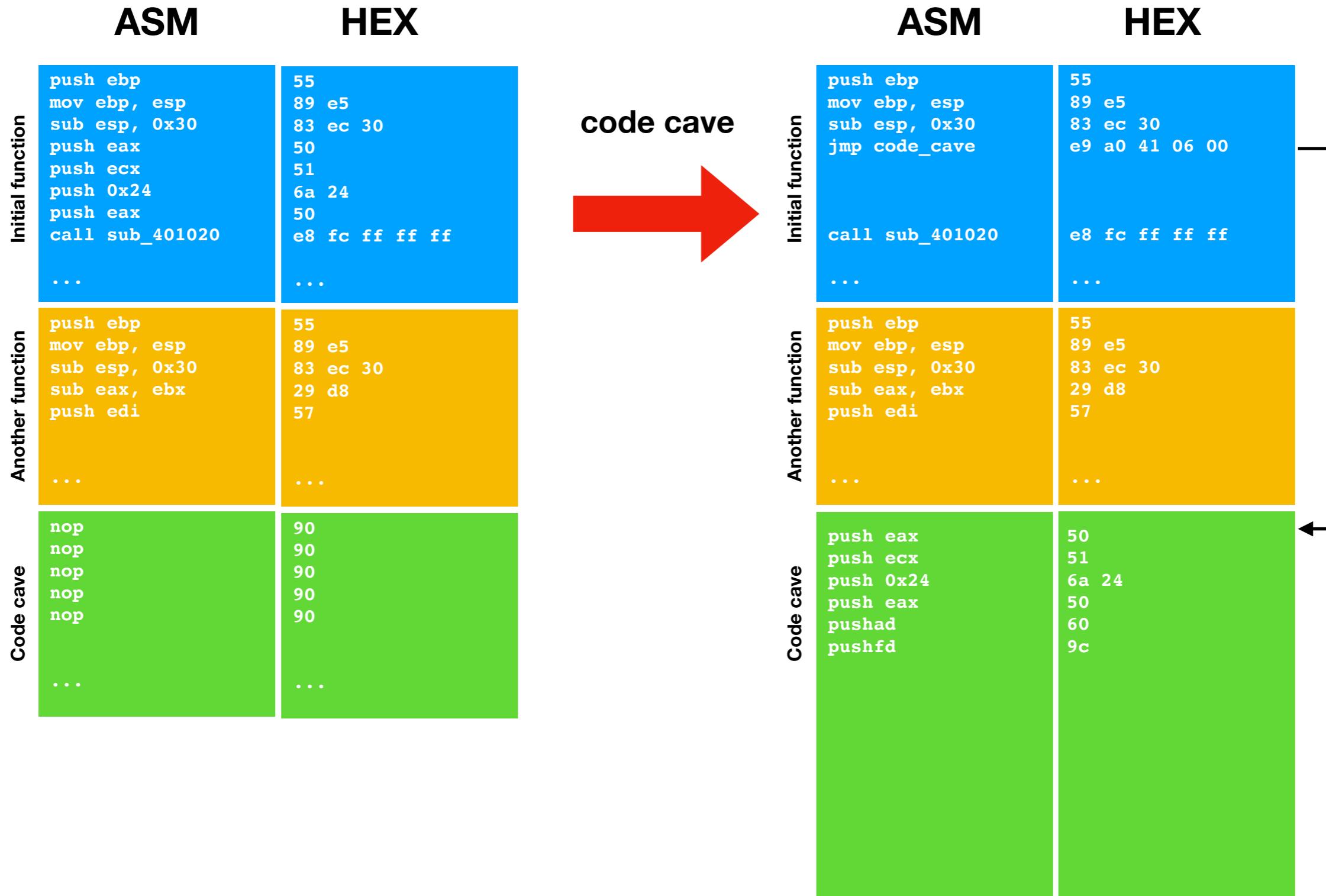
Library hooking



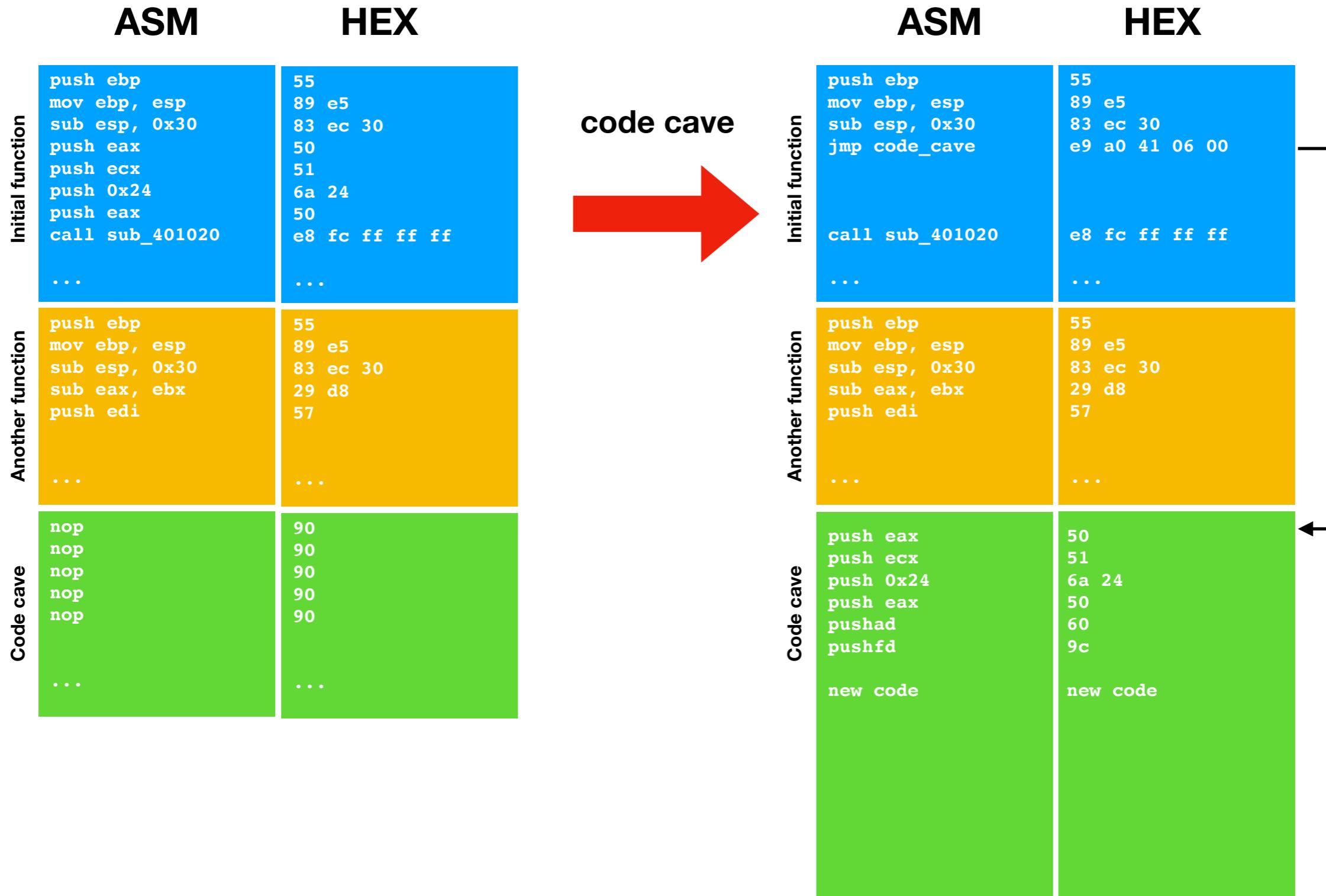
Library hooking



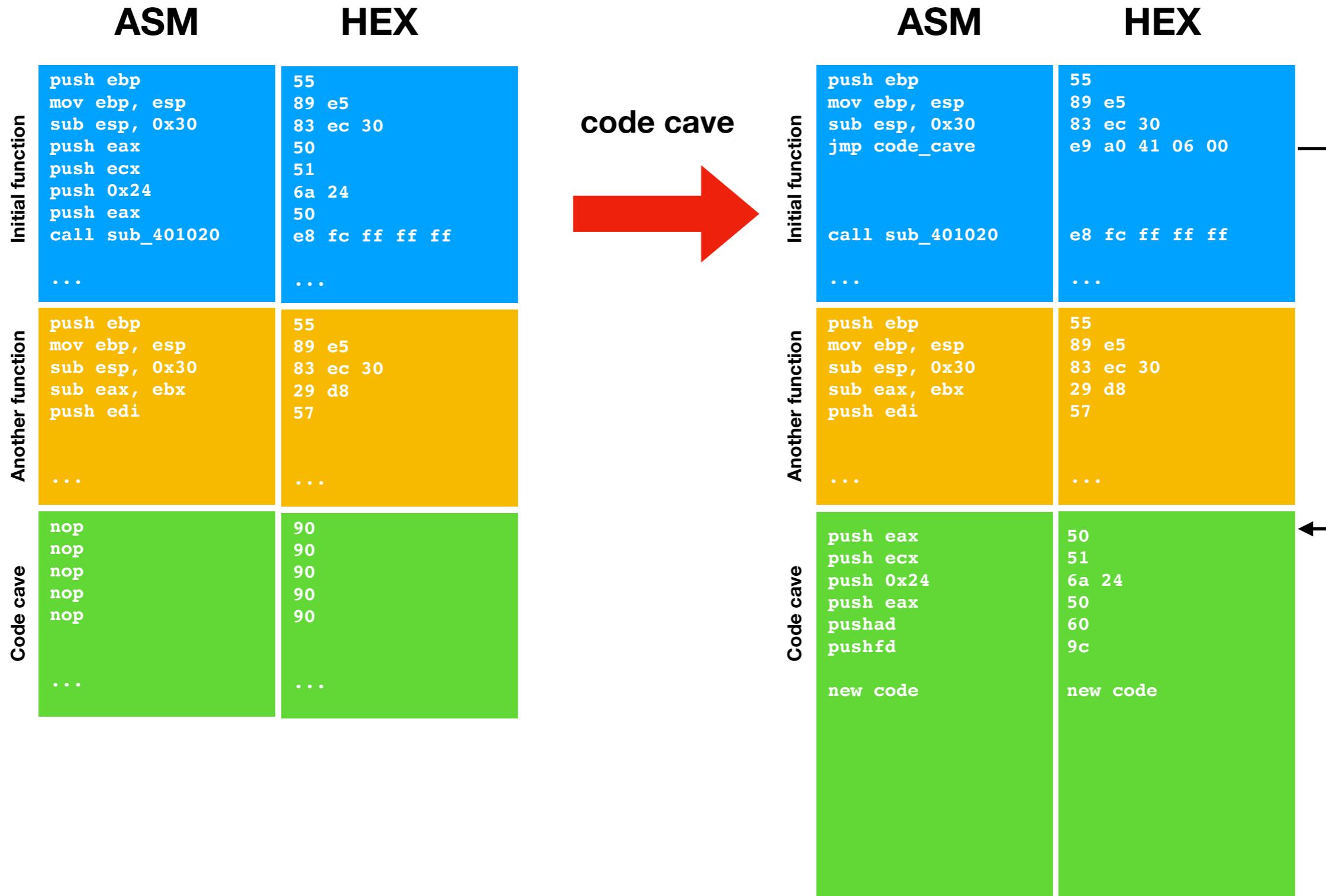
Library hooking



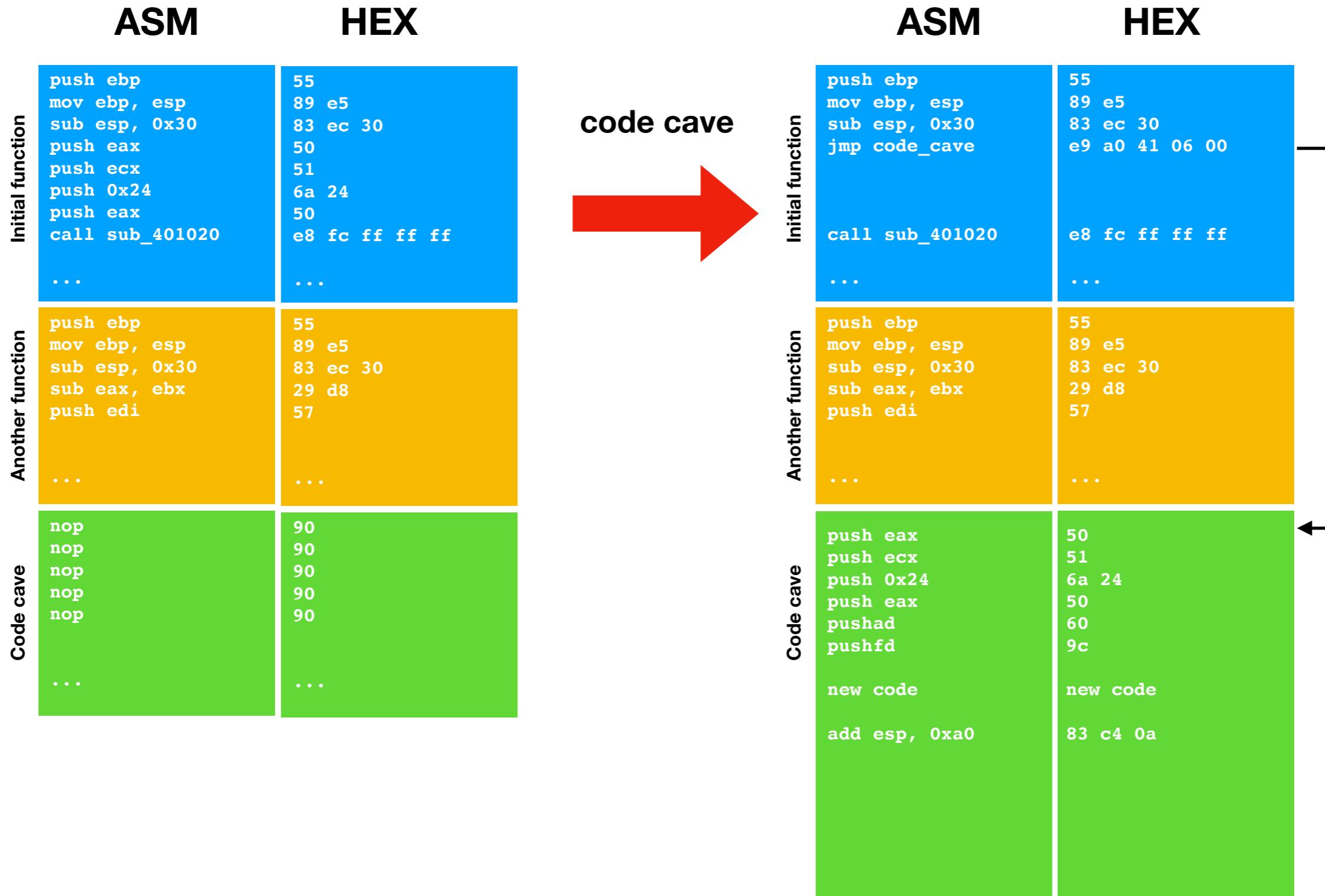
Library hooking



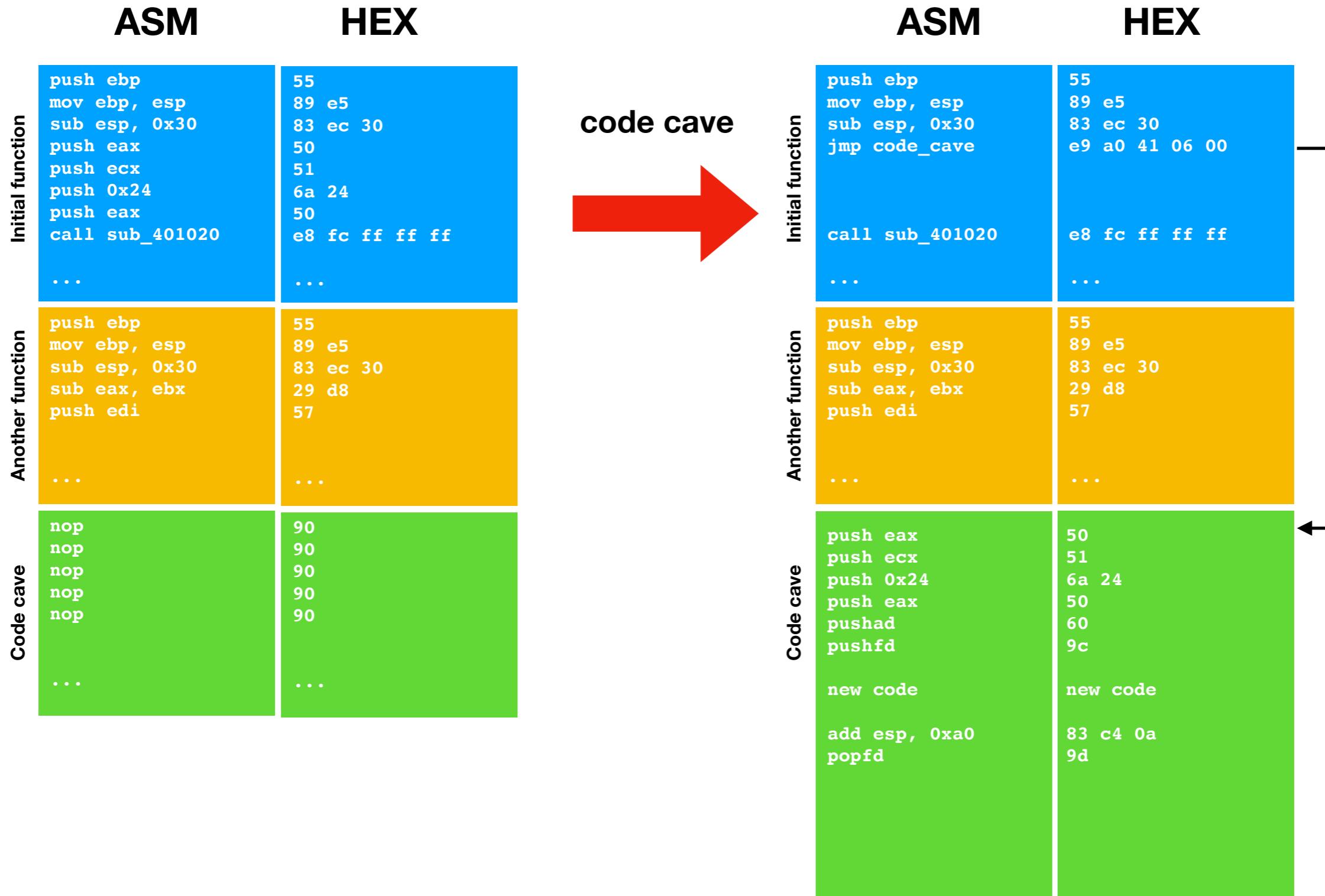
Library hooking



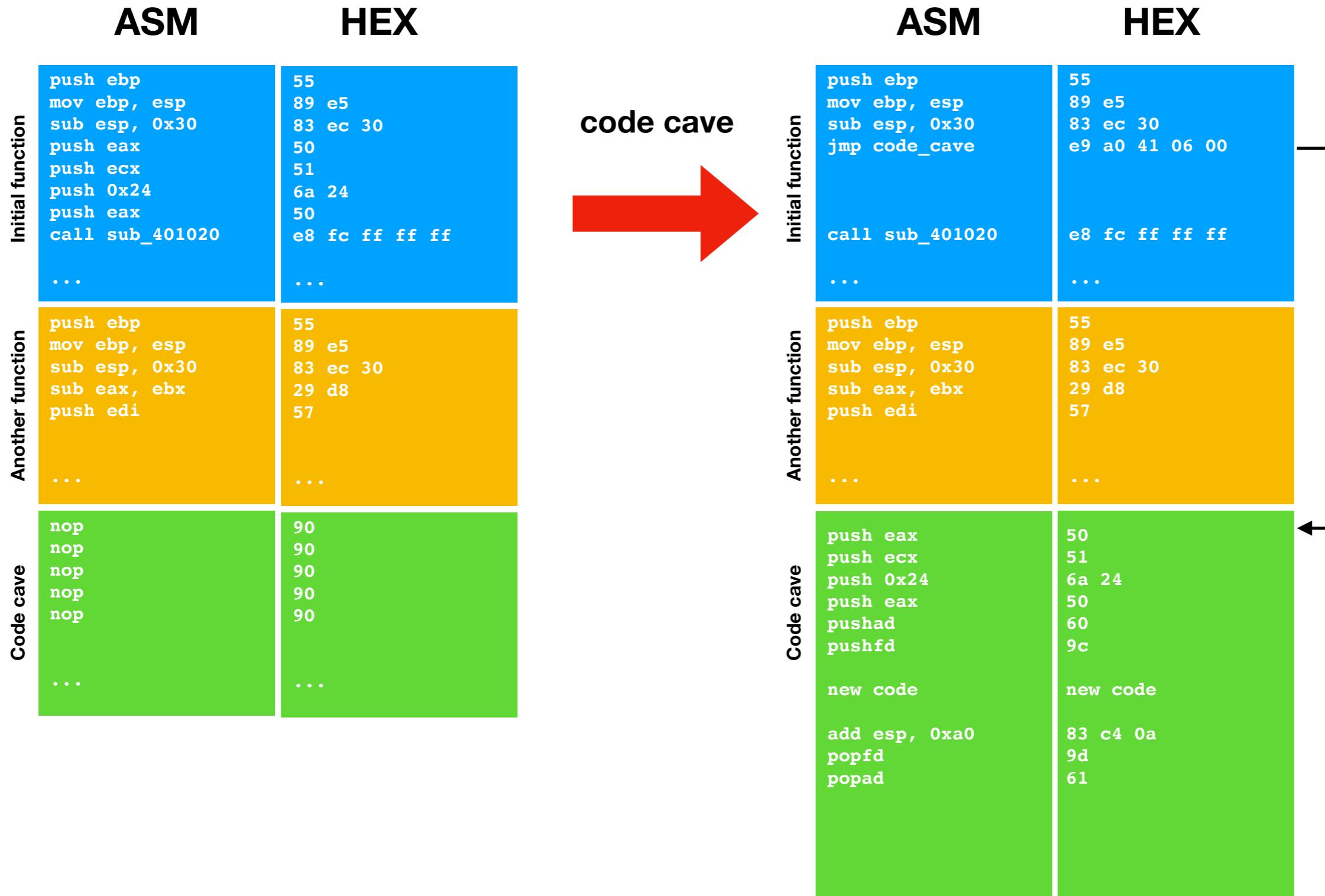
Library hooking



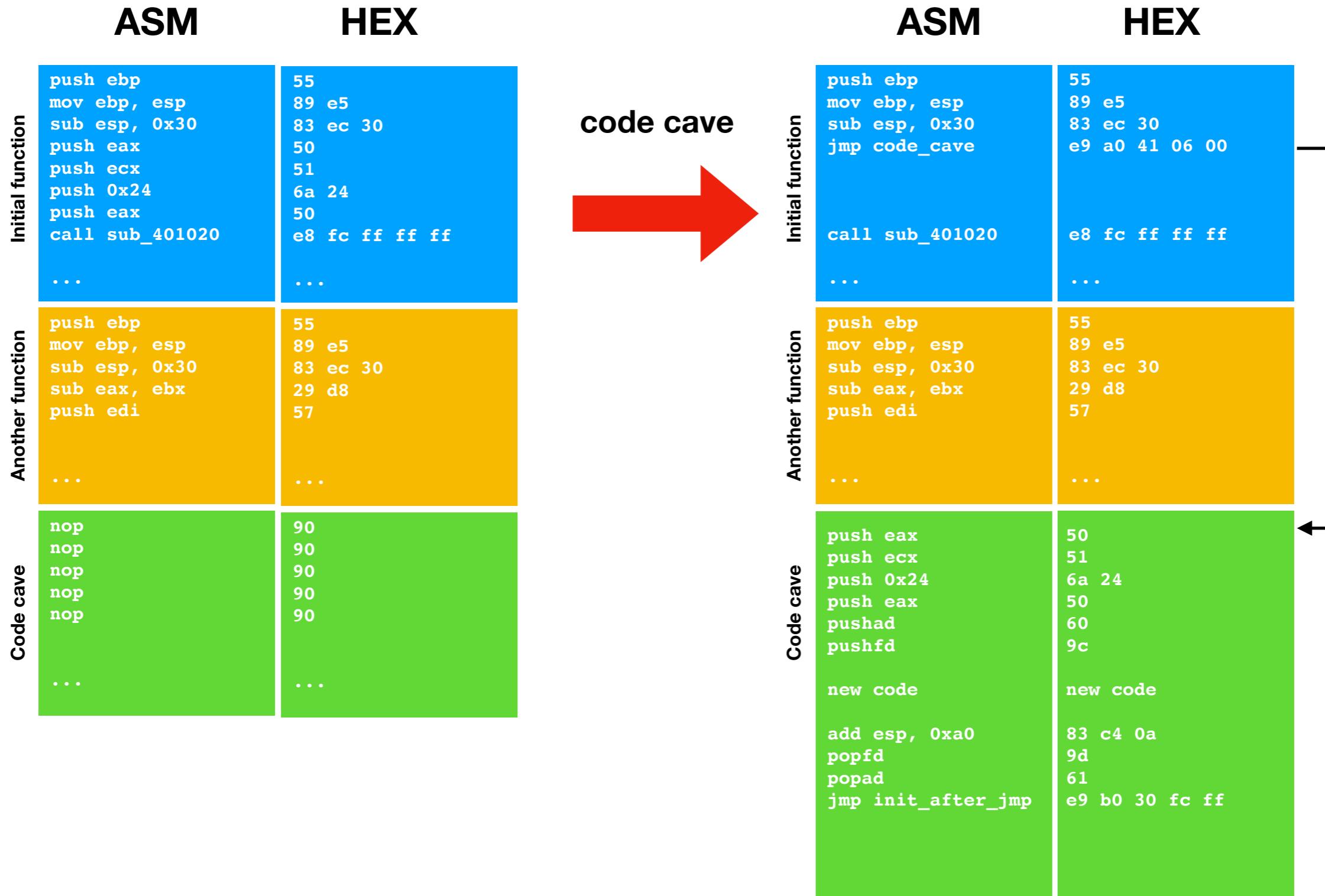
Library hooking



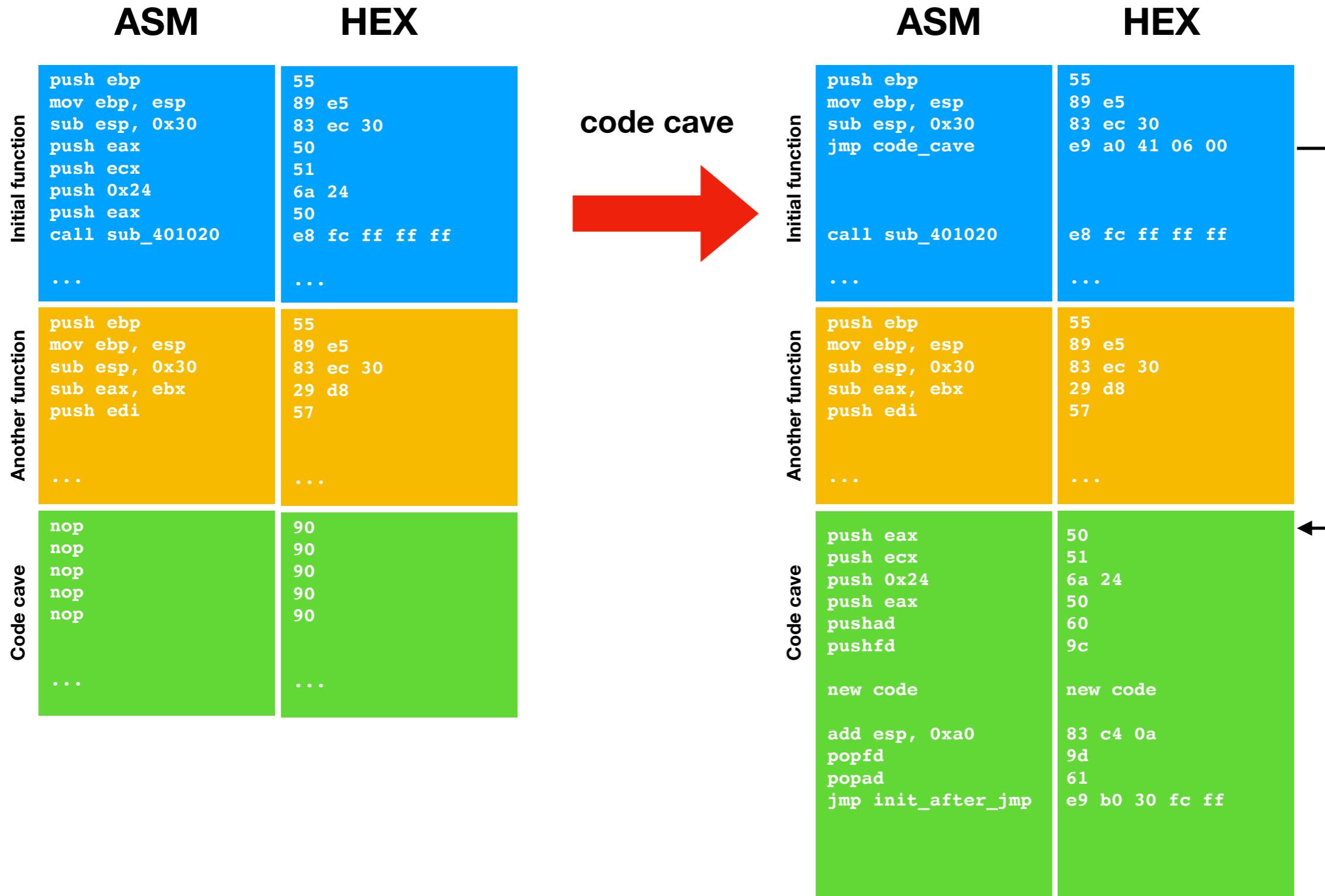
Library hooking



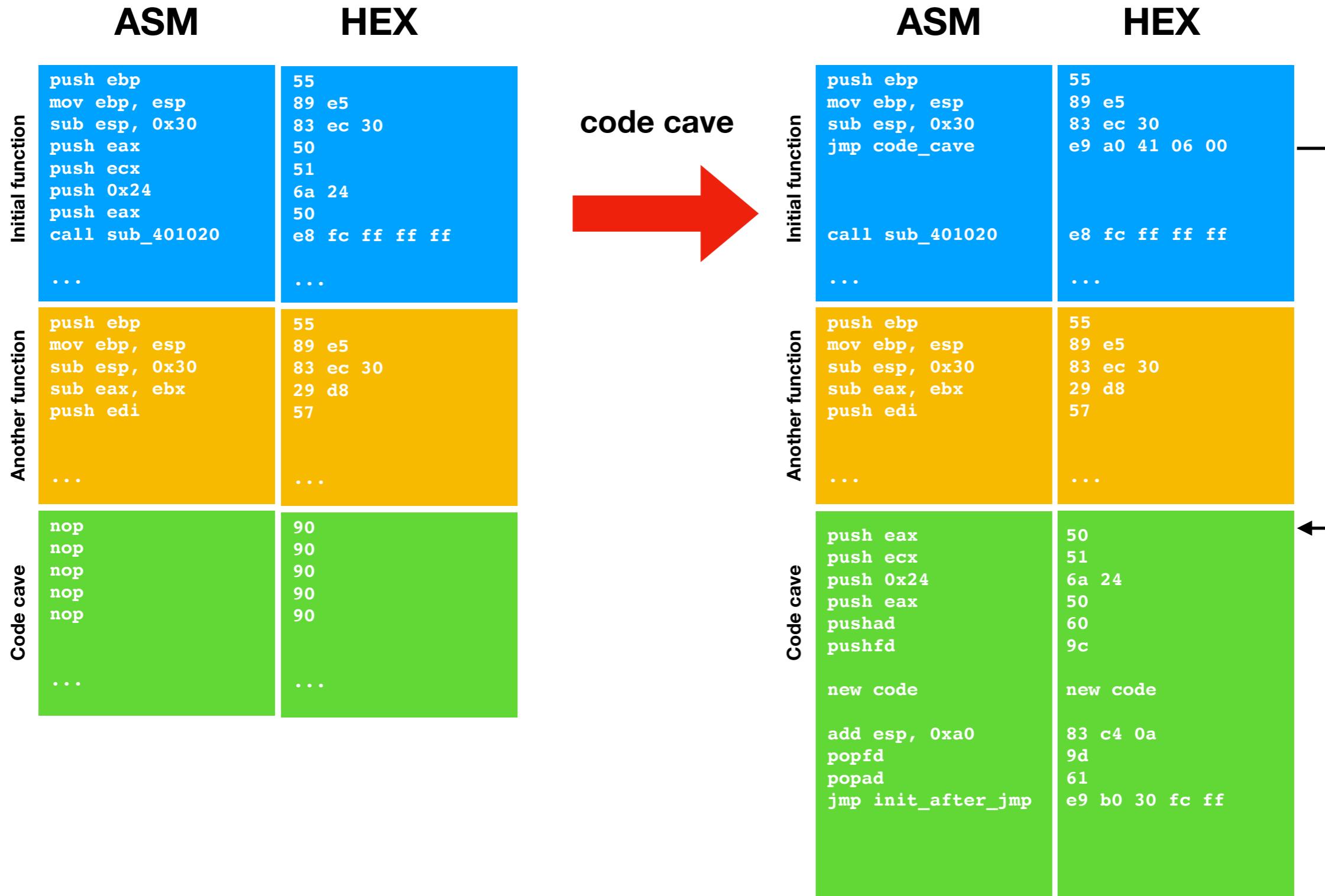
Library hooking



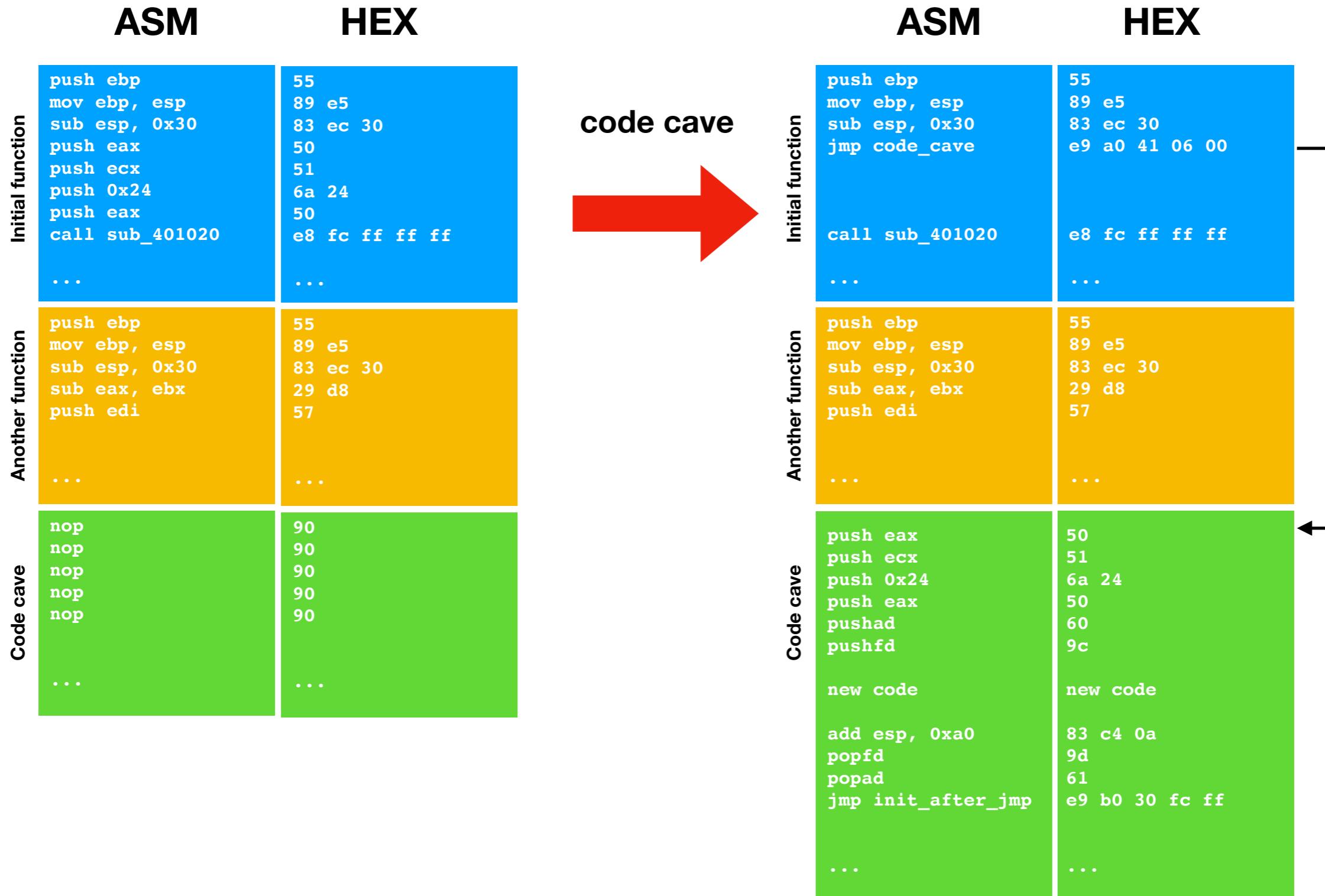
Library hooking



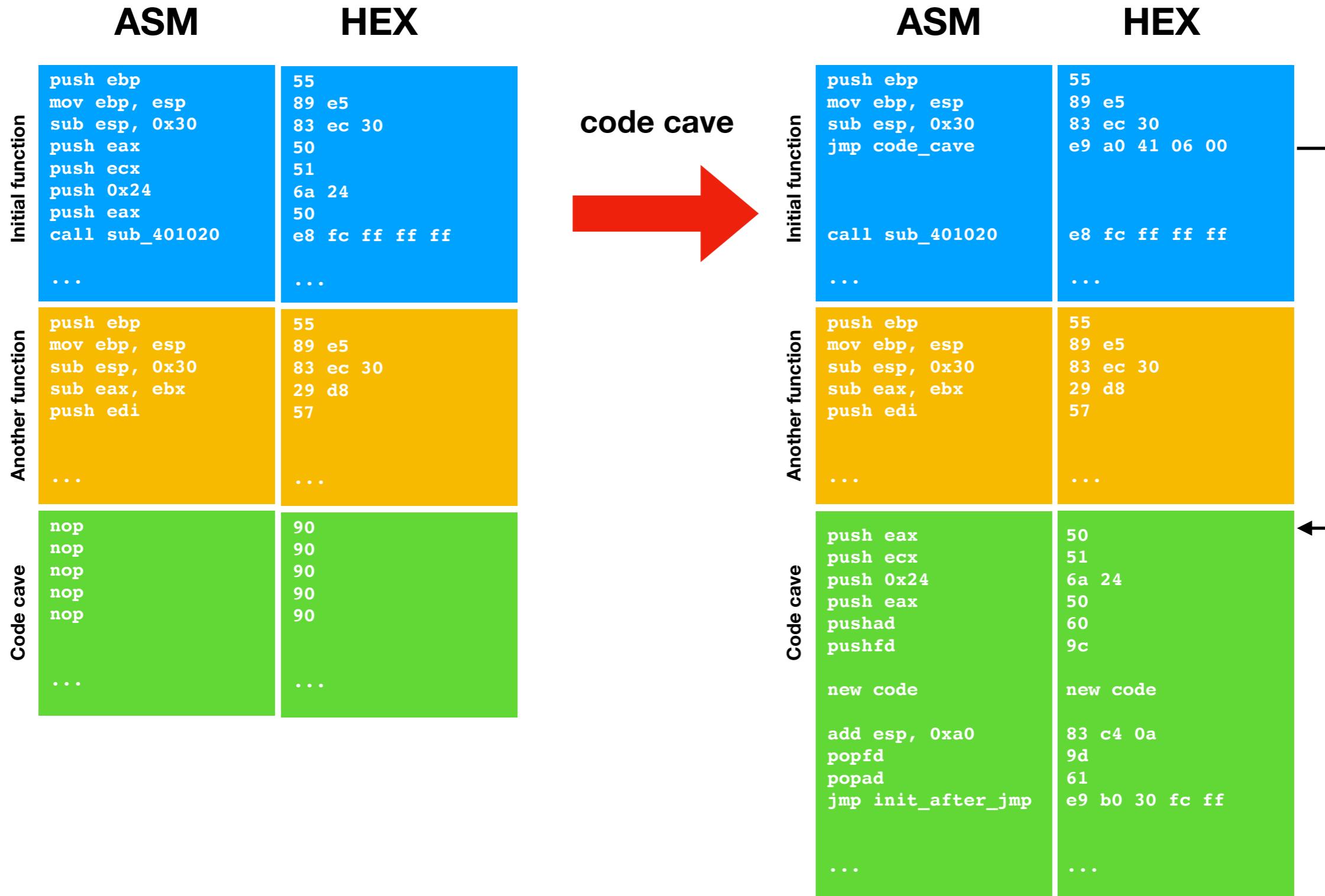
Library hooking



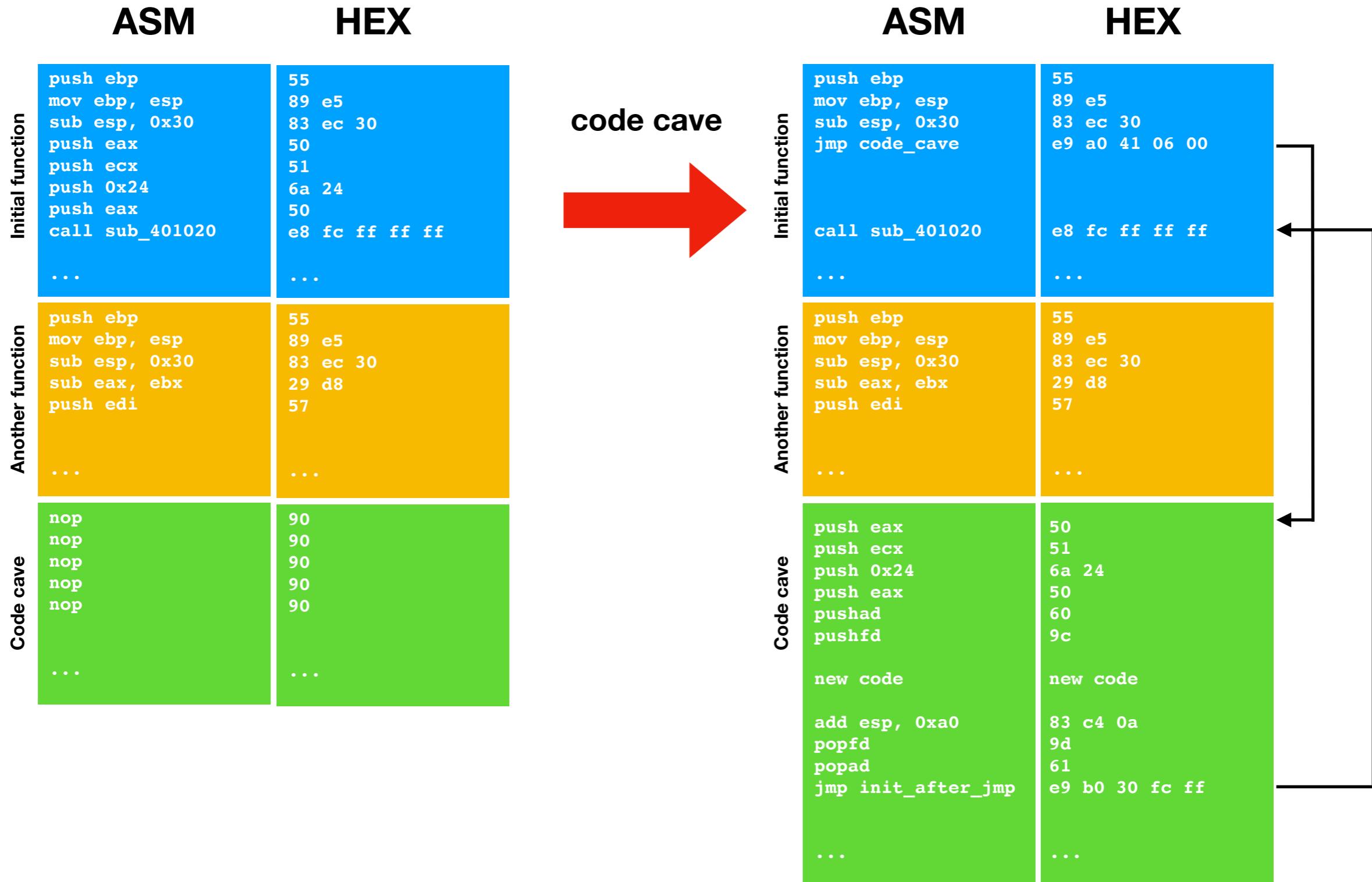
Library hooking



Library hooking



Library hooking



Library hooking

Code cave usage:

- Often used by malware developers to hide malicious code into benign applications
- Easy to break the application
- Requires to do the changes in assembly



Library hooking

Library hooking:

- Hijack the execution flow
- Code written with higher language (e.g. C/C++)
- Use *LD_PRELOAD* (in Linux)
 - Preloading a library
 - Its functions will be used before others of the same name in later libraries



Library hooking

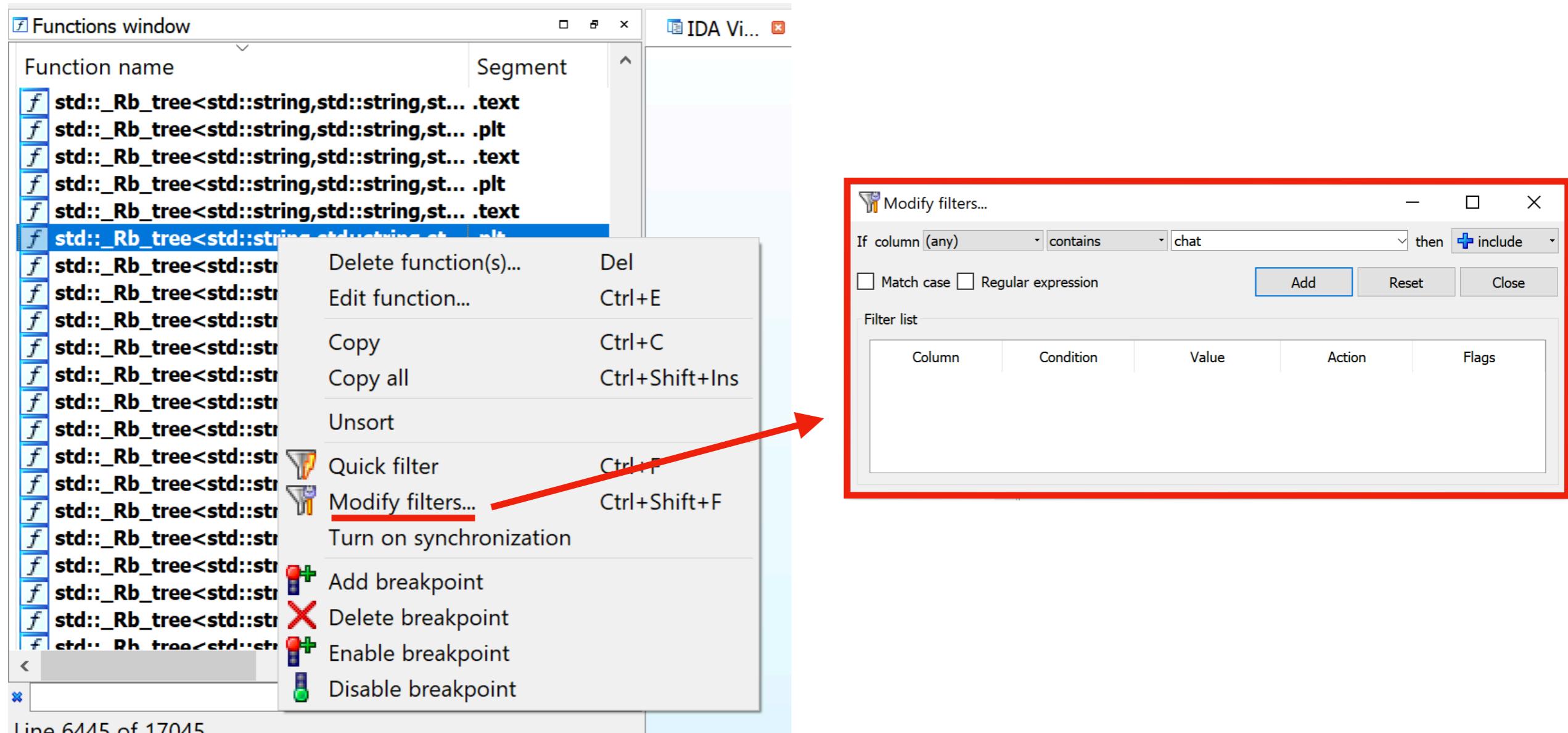
How to trigger the hack?

- Chat function
- Can be sent anytime
- Can contain custom text



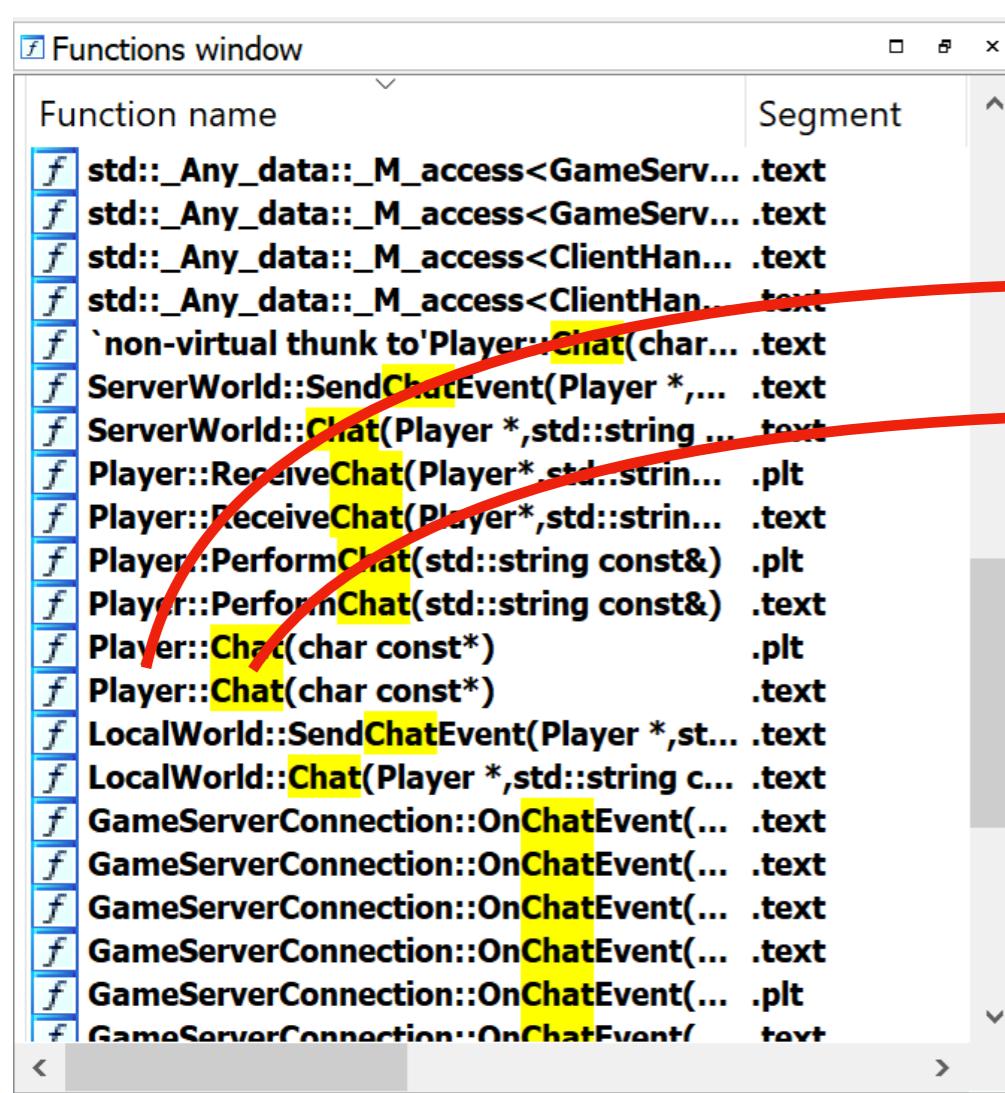
Library hooking

Find the function triggered when you **chat**:



Library hooking

What is Player::Chat structure?



```
// #define _GNU_SOURCE

#include <dlfcn.h>
#include <stdio.h>
#include <iostream>

class Player {
public:
    void Chat(const char *text);

};

typedef void (*orig_chat_f_type)(Player *, const char *);

void Player::Chat(const char *text)
{
    std::string str(text);
    std::cout << "Chat: " << str << "\n";

    // Call original Player::Chat() function
    orig_chat_f_type orig;
    orig = (orig_chat_f_type) dlsym(RTLD_NEXT,
                                    "_ZN6Player4ChatEPKc");
    orig(this, text);
}
```



Library hooking

Compile:

```
g++ -shared -fPIC -o chat.so chat.cc
```

Run and load the shared object:

```
LD_PRELOAD=`pwd`/chat.so ./PwnAdventure3-Linux-Shipping
```

Note: PwnAdventure is just a wrapper that actually call: PwnAdventure3_data/PwnAdventure3/PwnAdventure3/Binaries/Linux/PwnAdventure3-Linux-Shipping



Library hooking

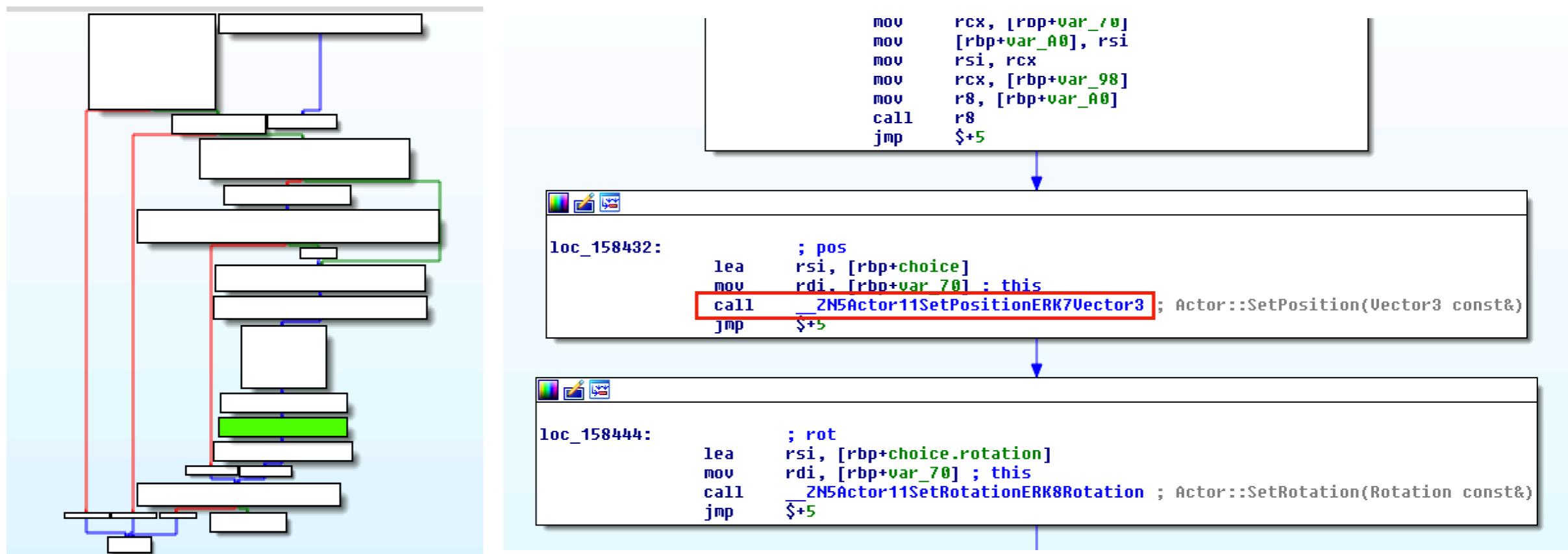
How to teleport:

- Create a function that changes X, Y, Z locally and update the server
- Use existing functions
 - Look for functions related to teleportation
 - Found Player::PerformTeleport



Library hooking

Player::PerformTeleport structure:



Library hooking

What is **Vector3** type?

```
(gdb) ptype Vector3
type = struct Vector3 {
    float x;
    float y;
    float z;
public:
    Vector3();
    Vector3(float, float, float);
    Vector3 operator*(float) const;
    Vector3 & operator*=(float);
    Vector3 operator+(const Vector3 &) const;
    Vector3 & operator+=(const Vector3 &);
    Vector3 operator-(const Vector3 &) const;
    Vector3 & operator-=(const Vector3 &);
    float MagnitudeSquared() const;
    float Magnitude() const;
    static float DistanceSquared(const Vector3 &, const Vector3 &);
    static float Distance(const Vector3 &, const Vector3 &);
    void Normalize();
    static Vector3 Normalize(const Vector3 &);
}
```



Library hooking

Use `dlsym` to find `Actor::SetPosition` address:

```
typedef void (*orig_pos_f_type)(Player *, const Vector3 *);  
orig_pos_f_type orig;  
orig = (orig_pos_f_type) dlsym(RTLD_NEXT, "_ZN5Actor11SetPositionERK7Vector3");
```



Library hooking

Teleport to *Michael Angelo* when **typing** *Michael*:

```
void Player::Chat(const char *text)
{
    std::string str(text);

    if (str.compare("Michael") == 0)
    {
        Vector3 pos;
        pos.x = 260255.0;
        pos.y = -249336.0;
        pos.z = 1476.0;

        const Vector3 p = pos;
        orig_pos_f_type orig;
        orig = (orig_pos_f_type) dlsym(RTLD_NEXT, "_ZN5Actor11SetPositionERK7Vector3");
        orig(this, &p);
    }

    // Call original Player::Chat() function
    orig_chat_f_type orig;
    orig = (orig_chat_f_type) dlsym(RTLD_NEXT, "_ZN6Player4ChatEPKc");
    orig(this, text);
}
```



Library hooking

How to move faster and jump higher?

- Changing sprint multiplier (binary patching)
- Changing default walking speed
- Changing default jump speed



Library hooking

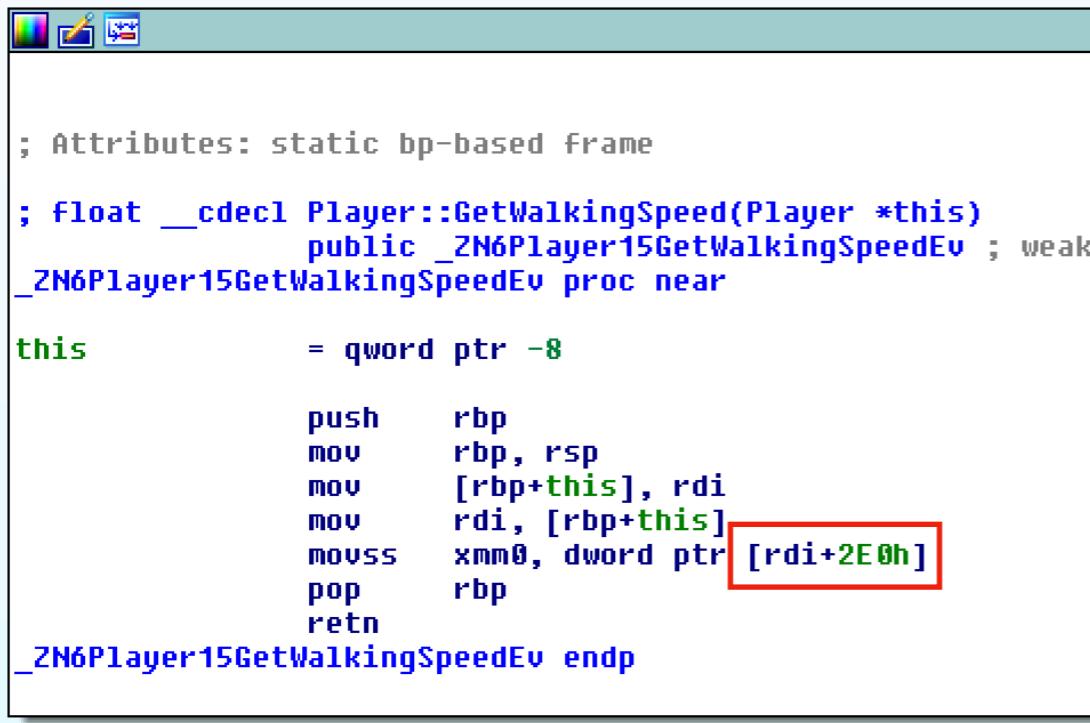
Function related to speed:

Function name
f Player::GetJumpSpeed(void)
f Player::GetWalkingSpeed(void)
f Player::GetWalkingSpeed(void)
f Player::GetJumpSpeed(void)

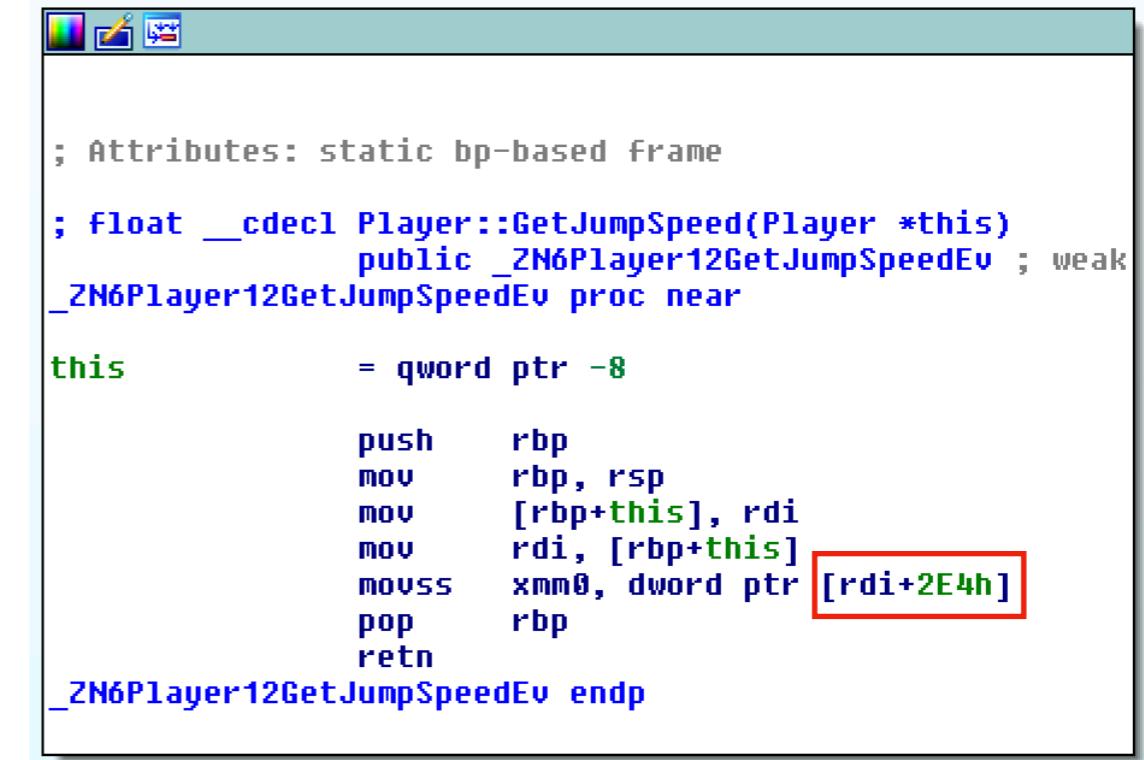


Library hooking

Function related to speed:



```
; Attributes: static bp-based frame
; float __cdecl Player::GetWalkingSpeed(Player *this)
    public _ZN6Player15GetWalkingSpeedEv ; weak
_ZN6Player15GetWalkingSpeedEv proc near
this      = qword ptr -8
    push    rbp
    mov     rbp, rsp
    mov     [rbp+this], rdi
    mov     rdi, [rbp+this]
    movss  xmm0, dword ptr [rdi+2E0h]
    pop    rbp
    retn
_ZN6Player15GetWalkingSpeedEv endp
```



```
; Attributes: static bp-based frame
; float __cdecl Player::GetJumpSpeed(Player *this)
    public _ZN6Player12GetJumpSpeedEv ; weak
_ZN6Player12GetJumpSpeedEv proc near
this      = qword ptr -8
    push    rbp
    mov     rbp, rsp
    mov     [rbp+this], rdi
    mov     rdi, [rbp+this]
    movss  xmm0, dword ptr [rdi+2E4h]
    pop    rbp
    retn
_ZN6Player12GetJumpSpeedEv endp
```

```
float Player::GetWalkingSpeed() {
    return this->walkingSpeed;
}
```

```
float Player::GetJumpSpeed() {
    return this->jumpSpeed;
}
```



Library hooking

Where is the **speed** value stored?

- `Player::GetWalkingSpeed()` and `Player::GetJumpSpeed()` return local variables
- Typical Setter/Getter



Library hooking

How to modify those values?

- Player *this is just a pointer
- Player structure is not define in our hook
- We cannot simply use the name
- Find the offset from the pointer
 - Player->WalkSpeed: offset 0x2e0 (736)
 - Player->JumpSpeed: offset 0x2e4 (740)
- What variables between beginning of object and walkSpeed?
 - Don't care, padding



Library hooking

New **hook** function:

```
class Player {  
  
public:  
    char padding[736];  
    float speed;  
    float jumpspeed;  
    void Chat(const char *text);  
  
};  
  
void Player::Chat(const char *text)  
{  
  
    std::string str(text);  
  
    if (str.compare("walk") == 0)  
    {  
        this->speed = this->speed * 4;  
    }  
  
    if (str.compare("jump") == 0)  
    {  
        this->jumpspeed = this->jumpspeed * 1.5;  
    }  
  
    // Call original Player::Chat() function  
    orig_chat_f_type orig;  
    orig = (orig_chat_f_type) dlsym(RTLD_NEXT, "_ZN6Player4ChatEPKc");  
    orig(this, text);  
}
```



Library hooking

Demo: You're now Superman, go kill each other!

[https://github.com/beaujeant/PwnAdventure3/
blob/master/Binary/hook-linux.cc](https://github.com/beaujeant/PwnAdventure3/blob/master/Binary/hook-linux.cc)



Building async proxy

Lab 7: Teleport to each eggs

Collect the location of each eggs and add functions to teleport to them (tp egg1, tp egg2, etc)



Building async proxy

Lab 8: Teleport to each eggs

Collect the location of each eggs and add functions to teleport to them (tp egg1, tp egg2, etc)



Future work

Homework:

- Reverse 3333 (SSL/TLS)
- Port Linux hook to Windows
- Wall hack

Want **more**?

- Vector35 trainings
- <https://www.reddit.com/r/REGames/>
- Attacking Network Protocols: A Hacker's Guide to Capture, Analysis, and Exploitation (James Forshaw)
- Fabien Sanglard's code review (fabiensanglard.net)



Questions?

@beaujeant

