Reverse Engineering a (M)MORPG

Antonin Beaujeant

17-19 October 2017

MMXVII

HACK.LU

AUTHOR

Name: Antonin Beaujeant (@beaujeant)

Job: Pentest - R&D

- Previous
- Now

Interest:

- Reverse engineering
- Hardware hacking
- CTF

AGENDA

- Introduction (12m)
- Game structure (8min)
- Define targets (8min)
- Find local saved data (10min)
- RE network protocol (45min)
- Building a Wireshark dissector (25min + 15min)
- Building an asynchronous proxy (python) (55min + 10min)
- RE binary (90min)
- Binary patching (30min)
- Library hooking (30min)

Introduction

17-19
October
2017
MMXVII
HACK.LU

INTRODUCTION

Game: Pwn Adventure 3

Genre: (M)MORPG - Team Adventure Quest

Developper: Vector35 (**https://vector35.com**)

Game Engine: Unity (cross-platform)

Event: Ghost In The Shellcode CTF 2015 (ShmooCon)

RE complexity: Medium (no obfuscation, no signature/encryption, all-in-one logic file)

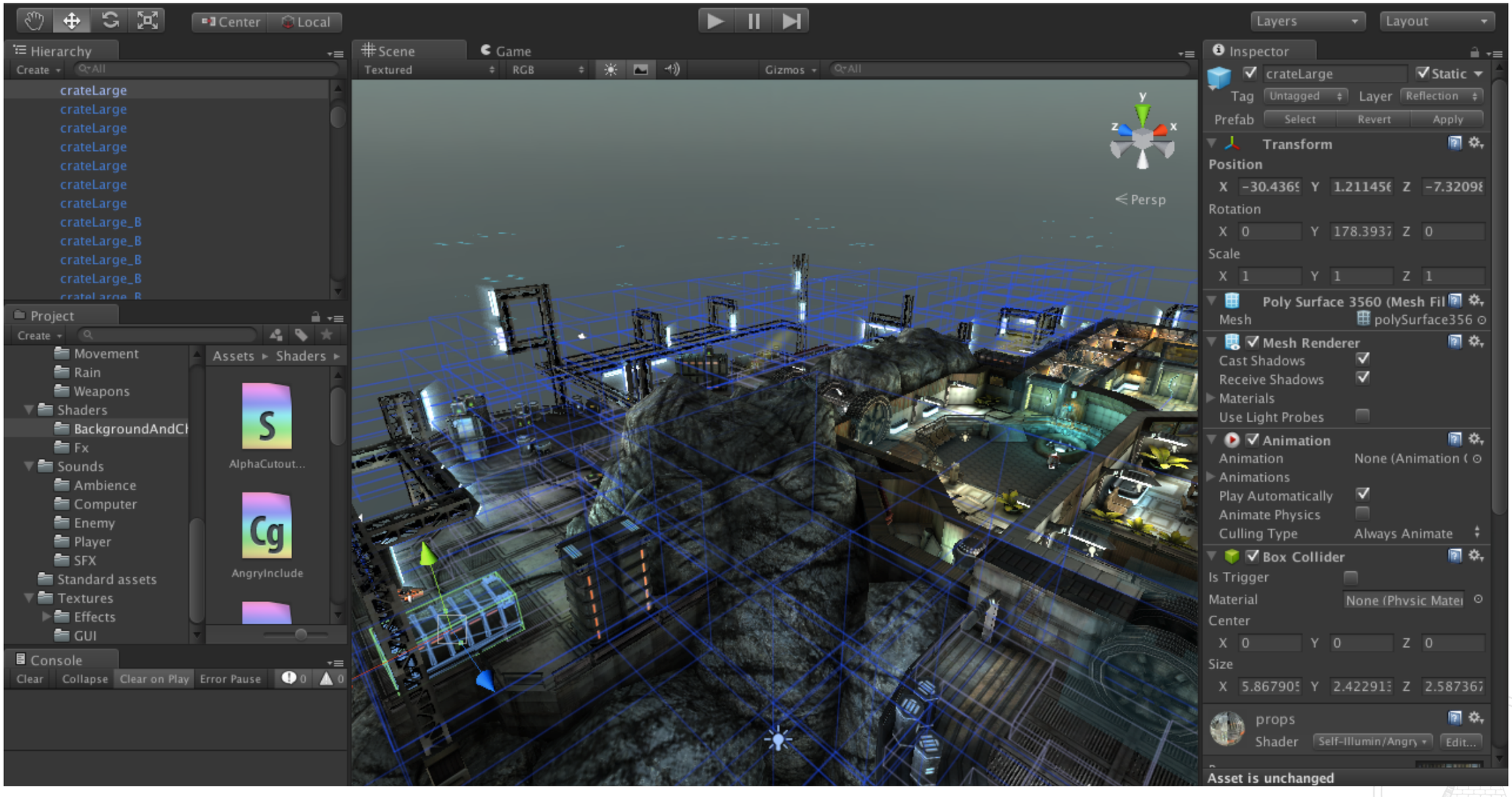
INTRODUCTION

List of **quests**:

- Fire and Ice (RE binary)
- *Pirates Treasure (crack me)*
- 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)*

17-19 October 2017
MMXVII
HACK.LU

Game Structure



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

17-19 October 2017
MMXVII
HACK.LU

GAME STRUCTURE

Pwn Adventure 3 structure:

- Master game server - Login, team, characters, assign instance
- Game server - Game instances
- 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 execution 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)

17-19 October 2017
MMXVII
HACK.LU

DEFINE TARGET

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 player
- Remove DRM
- Generate money (in game - real life)

DEFINE TARGET

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 weakness

DEFINE TARGET

Targets for this workshop**:**

- Spawn wherever we want
- Pick up remote items
- Find secret to unlock quest
- Find vulnerability to kill boss
- Run faster
- Jump higher
- Teleport anywhere

DEFINE TARGET

Where to look?

- Network communication (if online)
- Local saved data
- Client game logic binary
- Server game logic binary (if online)
- Rendering engine

DEFINE TARGET

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

17-19
October
2017
MMXVII
HACK.LU

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

Lab 1: Find local saved data

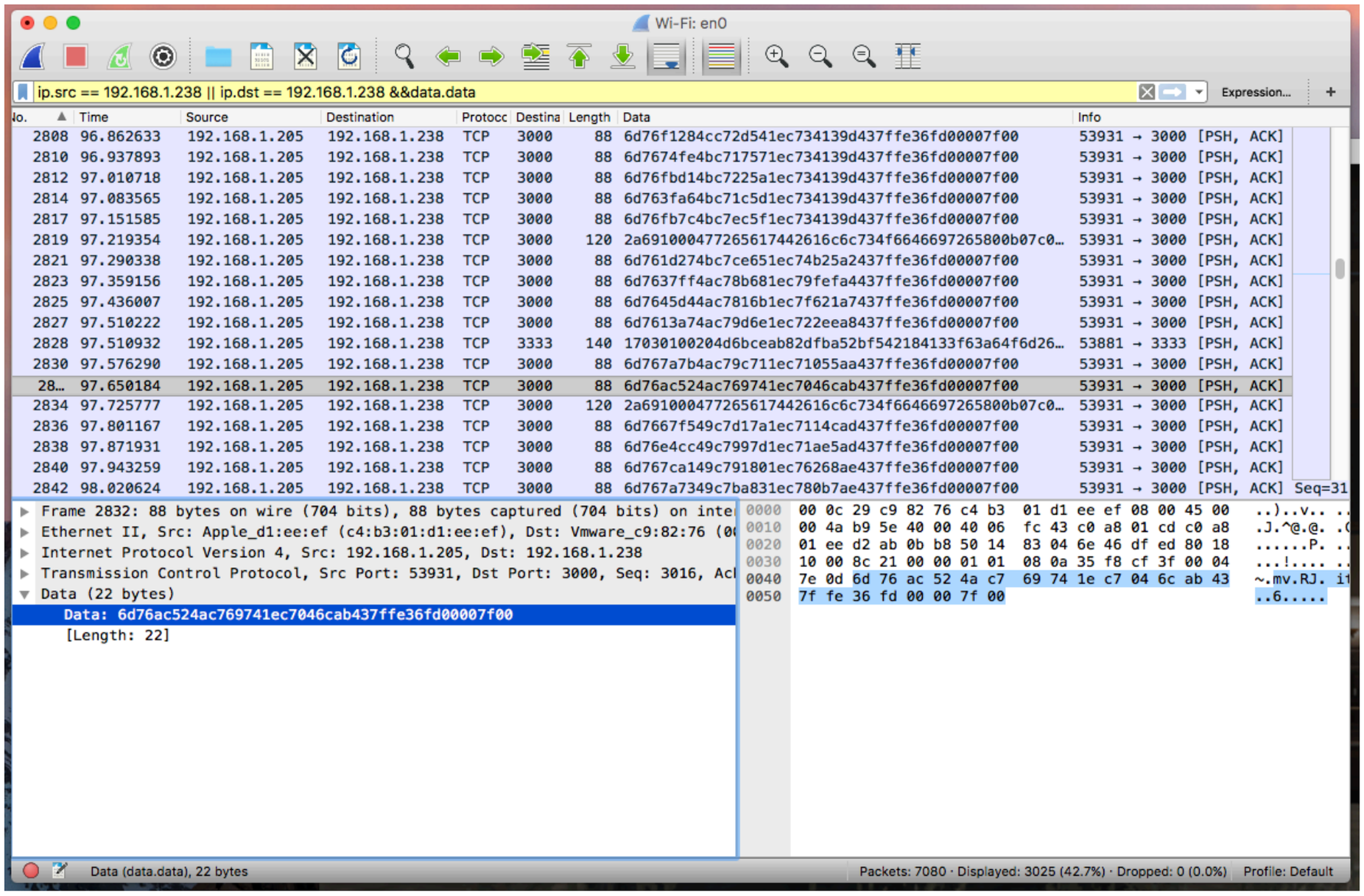
Find new/deleted files:

- `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`

RE Network Protocol



17-19 October 2017
MMXVII
HACK.LU

RE NETWORK PROTOCOL

Data expected:

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

RE NETWORK PROTOCOL

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
  - Identifier (when sequence order is not predictable)
  - Delimiter (when size is not predicable)

RE NETWORK PROTOCOL

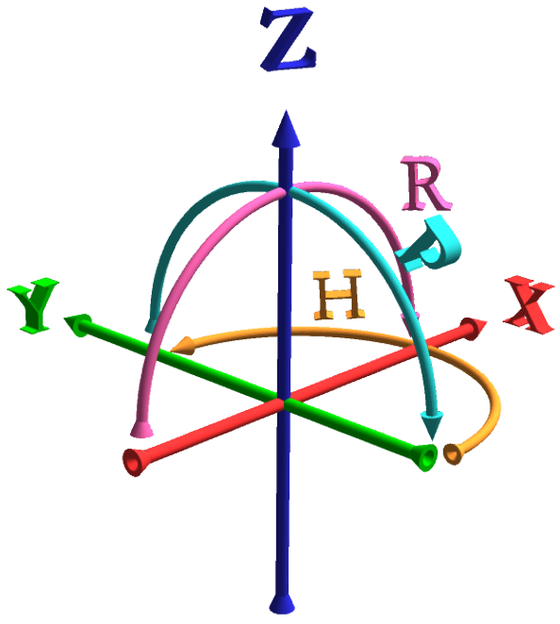
Methodology for *binary* protocol using *unknown* format**:**

- Understand your target
- Identify IP(s), port(s) and content (encrypted? Format?)
- Build use cases (e.g. don’t move, shoot, etc)
- Look at network behaviour (size, frequency, pattern, etc)
- Raise assumptions
  - Identify the packet
    - Isolate variables
    - Locate the variable

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

RE NETWORK PROTOCOL

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



RE NETWORK PROTOCOL

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

No mouvement:

(tcp.port >= 3000 && tcp.port <= 3010) && tcp.len > 0 — Expression… +

No.	Source	Destination	Protoco	Length	Info	Data
6	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c75f867e440000000000000000
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 …	6d76687150c758eb60c75f867e440000000000000000
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 …	6d76687150c758eb60c75f867e440000000000000000
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 …	6d76687150c758eb60c75f867e440000000000000000
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 …	6d76687150c758eb60c75f867e440000000000000000
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 …	6d76687150c758eb60c75f867e440000000000000000
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 …	6d76687150c758eb60c75f867e440000000000000000
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 …	6d76687150c758eb60c75f867e440000000000000000
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 …	6d76687150c758eb60c75f867e440000000000000000
31	192.168.2.1…	192.168.2.1	TCP	68	3000 → 62634 …	0000

Structure: -

17-19 October 2017
MMXVII
HACK.LU

RE NETWORK PROTOCOL

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

Jump:

ip.dst == 192.168.2.130 && (tcp.port >= 3000 && tcp.port <= 3010) && tcp.len > 0 — Expression… +

No.	Source	Destination	Protoco	Length	Info	Data
324	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c75f867e44000000000000000
327	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c75f867e44000000000000000
330	192.168.2.1	192.168.2.130	TCP	91	62634 → 3000 …	6a70016d76687150c758eb60c770ff804400000000000000..
334	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c72f96844000000000000000
338	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c7a114884000000000000000
341	192.168.2.1	192.168.2.130	TCP	91	62634 → 3000 …	6a70006d76687150c758eb60c749ab8b4400000000000000..
345	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c76ab78e4000000000000000
349	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c71bbc904000000000000000
352	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c7933792400000000000000
355	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c7000993400000000000000
358	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c78a5493400000000000000
361	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c72f0393400000000000000
364	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c74d0d92400000000000000

Structure: - - - - - - - - - - Z Z Z - - - - - - - - -

RE NETWORK PROTOCOL

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

Move forward:

ip.dst == 192.168.2.130 && (tcp.port >= 3000 && tcp.port <= 3010) && tcp.len > 0 — Expression… +

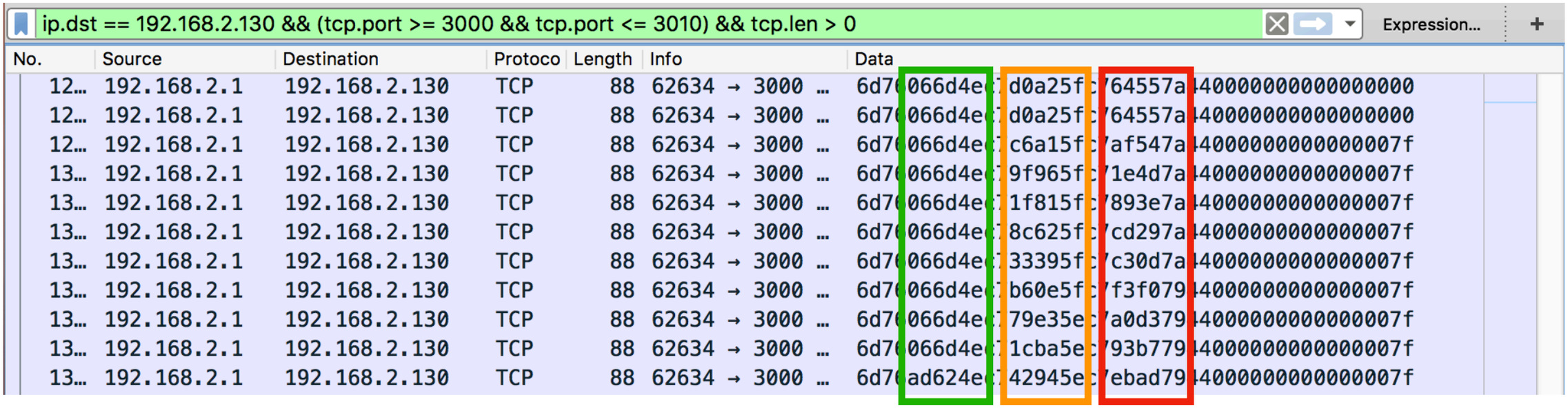
No.	Source	Destination	Protoco	Length	Info	Data
954	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c75f867e400000000000000000
957	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76687150c758eb60c75f867e400000000000000000
960	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76957050c758eb60c7a9897e400000000000007f00
963	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76d86650c758eb60c783b07e400000000000007f00
966	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76285150c758eb60c707077f400000000000007f00
969	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76a23250c758eb60c7cd807f400000000000007f00
972	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d765e0450c758eb60c7ae1c80400000000000007f00
975	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d7651d64fc758eb60c7877880400000000000007f00
978	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d763ea94fc758eb60c76ed280400000000000007f00
981	192.168.2.1	192.168.2.130	TCP	88	62634 → 3000 …	6d76627e4fc758eb60c79cd880400000000000007f00

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

RE NETWORK PROTOCOL

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

Strafe right:



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

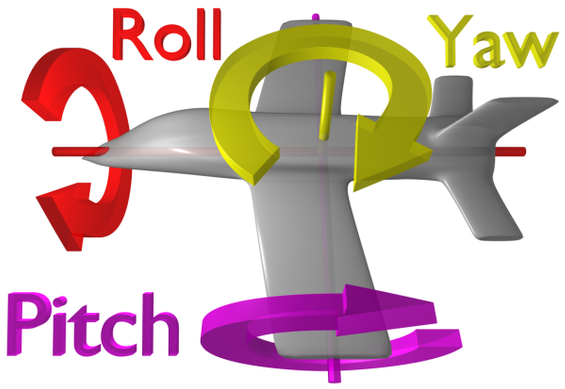
RE NETWORK PROTOCOL

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

Structure: – – X X X – Y Y Y – Z Z Z 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

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



RE NETWORK PROTOCOL

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

Look left:

ip.dst == 192.168.2.130 && (tcp.port >= 3000 && tcp.port <= 3010) && tcp.len > 0 — Expression… +

No.	Source	Destination	Protoco	Length	Info	Data
119	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76ad6250c714c15ec7353a8d440000000000000000
122	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76ad6250c714c15ec7353a8d440000000000000000
126	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76ad6250c714c15ec7353a8d44000[0]a104[0]0000000
130	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76ad6250c714c15ec7353a8d44000[0]2d0b[0]0000000
133	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76ad6250c714c15ec7353a8d44000[0]9f11[0]0000000
136	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76ad6250c714c15ec7353a8d44000[0]351a[0]0000000
139	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76ad6250c714c15ec7353a8d44000[0]9528[0]0000000
142	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76ad6250c714c15ec7353a8d44000[0]cb2e[0]0000000
145	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76ad6250c714c15ec7353a8d44000[0]2635[0]0000000
148	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76ad6250c714c15ec7353a8d44000[0]2836[0]0000000

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

RE NETWORK PROTOCOL

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

Look up:

ip.dst == 192.168.2.130 && (tcp.port >= 3000 && tcp.port <= 3010) && tcp.len > 0 — Expression… +

No.	Source	Destination	Protoco	Length	Info	Data
717	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76cf5050c7eca75ec7a17d8d440000df2600000000
720	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76cf5050c7eca75ec7a17d8d440000df2600000000
723	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76cf5050c7eca75ec7a17d8d441600df2600000000
726	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76cf5050c7eca75ec7a17d8d44b000df2600000000
729	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76cf5050c7eca75ec7a17d8d44e600df2600000000
732	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76cf5050c7eca75ec7a17d8d44e600df2600000000
735	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76cf5050c7eca75ec7a17d8d442901df2600000000
743	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76cf5050c7eca75ec7a17d8d445501df2600000000
746	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76cf5050c7eca75ec7a17d8d445501df2600000000
749	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76cf5050c7eca75ec7a17d8d44a601df2600000000
752	192.168.2.1	192.168.2.130	TCP	88	63093 → 3000 …	6d76cf5050c7eca75ec7a17d8d44a602df2600000000

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

RE NETWORK PROTOCOL

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

Look up:

ip.dst == 192.168.2.130 && (tcp.port >= 3000 && tcp.port <= 3010) && tcp.len > 0 — Expression… +

No.	Source	Destination	Protoco	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 X Y Y Y Y Z Z Z Z P P YA YA R R U S

RE NETWORK PROTOCOL

Lab 2: 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 3: Identify and dissect the “mana” and “chat” packets

Ressource:

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

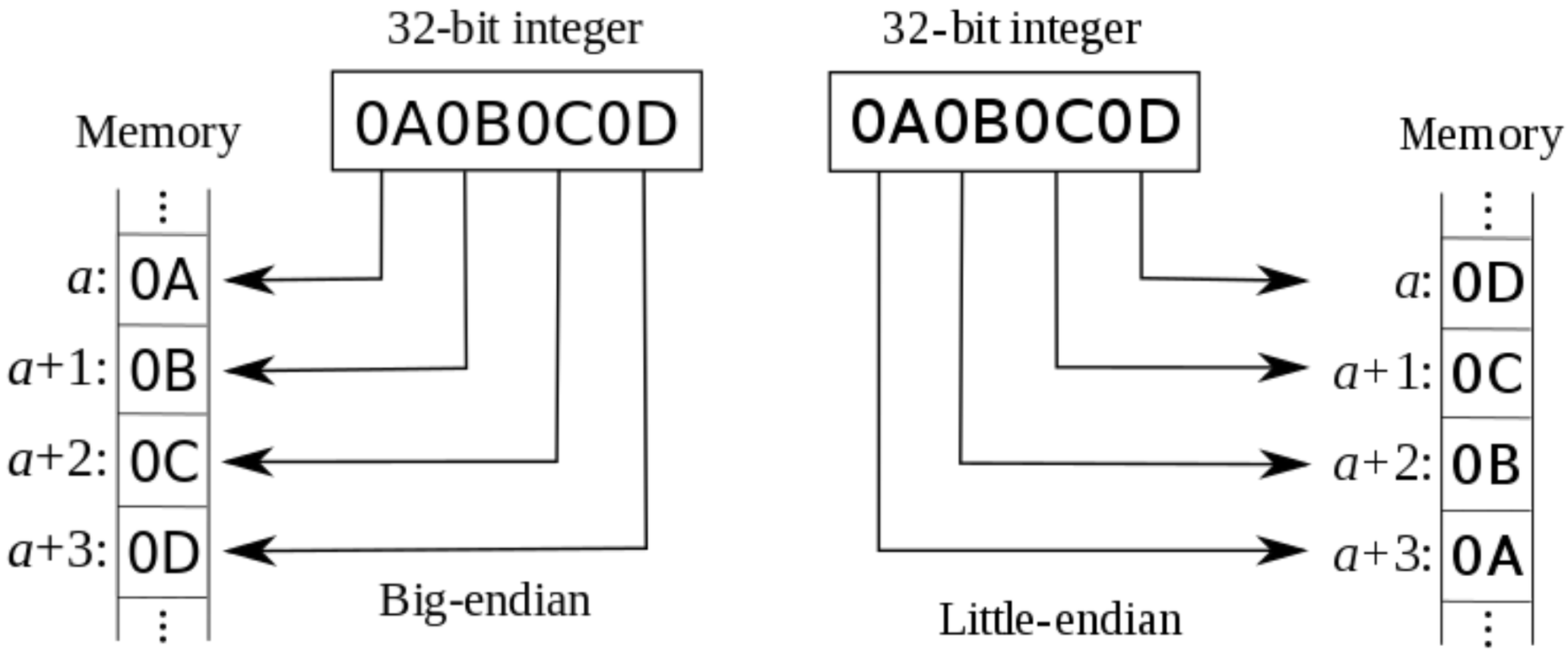
RE NETWORK PROTOCOL

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

Endianness:



RE NETWORK PROTOCOL

Complete list of reversed packets:

https://github.com/Foxmole/PwnAdventure3/blob/master/protocol.md

Building a Wireshark Dissector

17-19
October
2017
MMXVII
HACK.LU

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 - Game server 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)

...
```

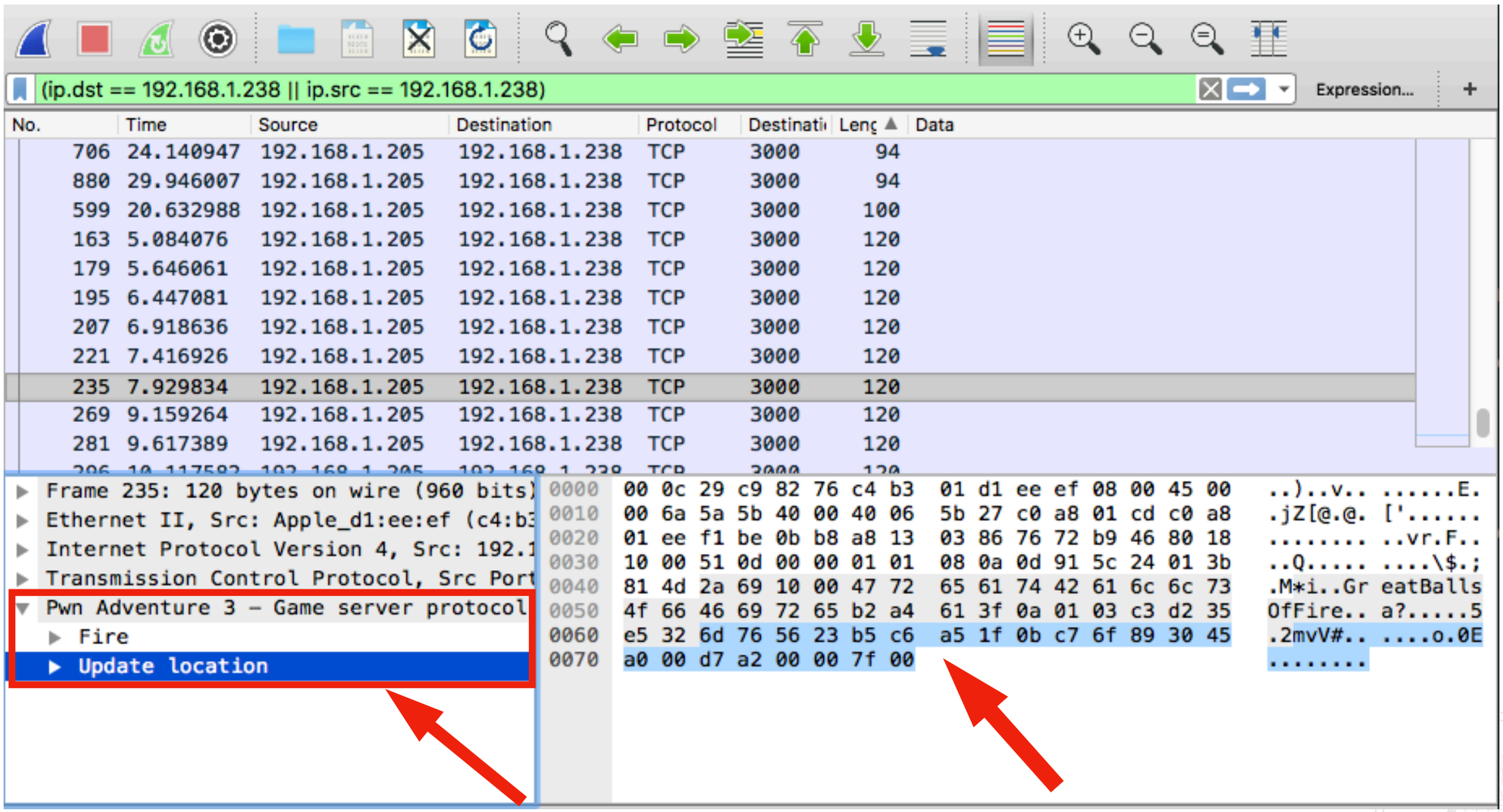
BUILDING DISSECTOR

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

```
function PWN3.dissector (buffer, pinfo, tree)
    -- Dissection code
    -- buffer is the data in the TCP packet
    -- tree is the root element in the visual tree structure

end
```

BUILDING DISSECTOR

Wireshark **visual tree structure**:



17-19 October 2017
MMXVII
HACK.LU

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 (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

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
```

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 mane" 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, which might be part of the data

BUILDING DISSECTOR

Install dissector:

- http://10.13.110.24/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 4: Build the rest of the dissector

Ressources:

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

BUILDING DISSECTOR

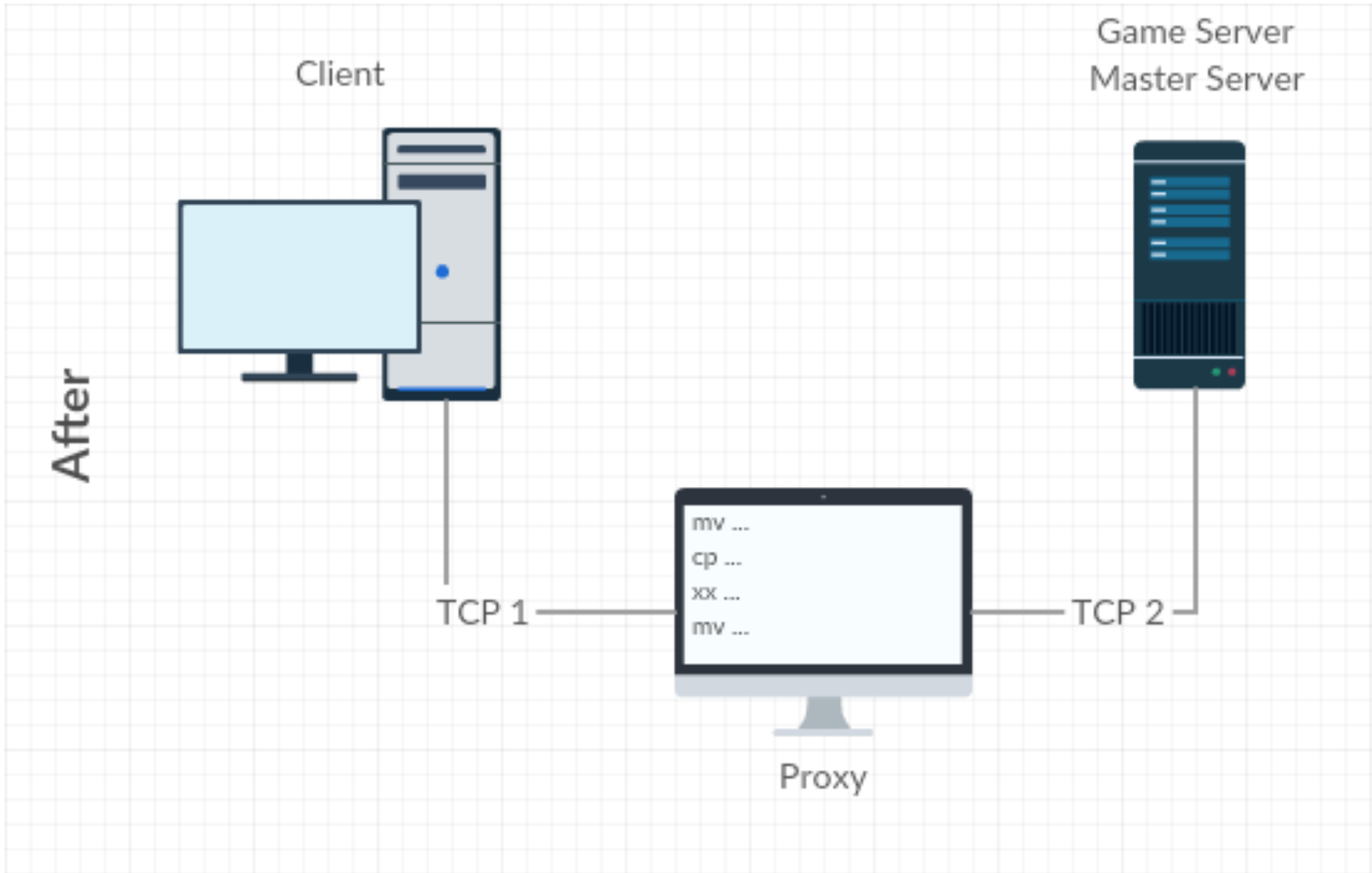
Complete dissector:

https://github.com/Foxmole/PwnAdventure3/blob/master/pwn3.lua

BUILDING DISSECTOR

Demo: Play game with dissector

Building an Asynchronous Proxy



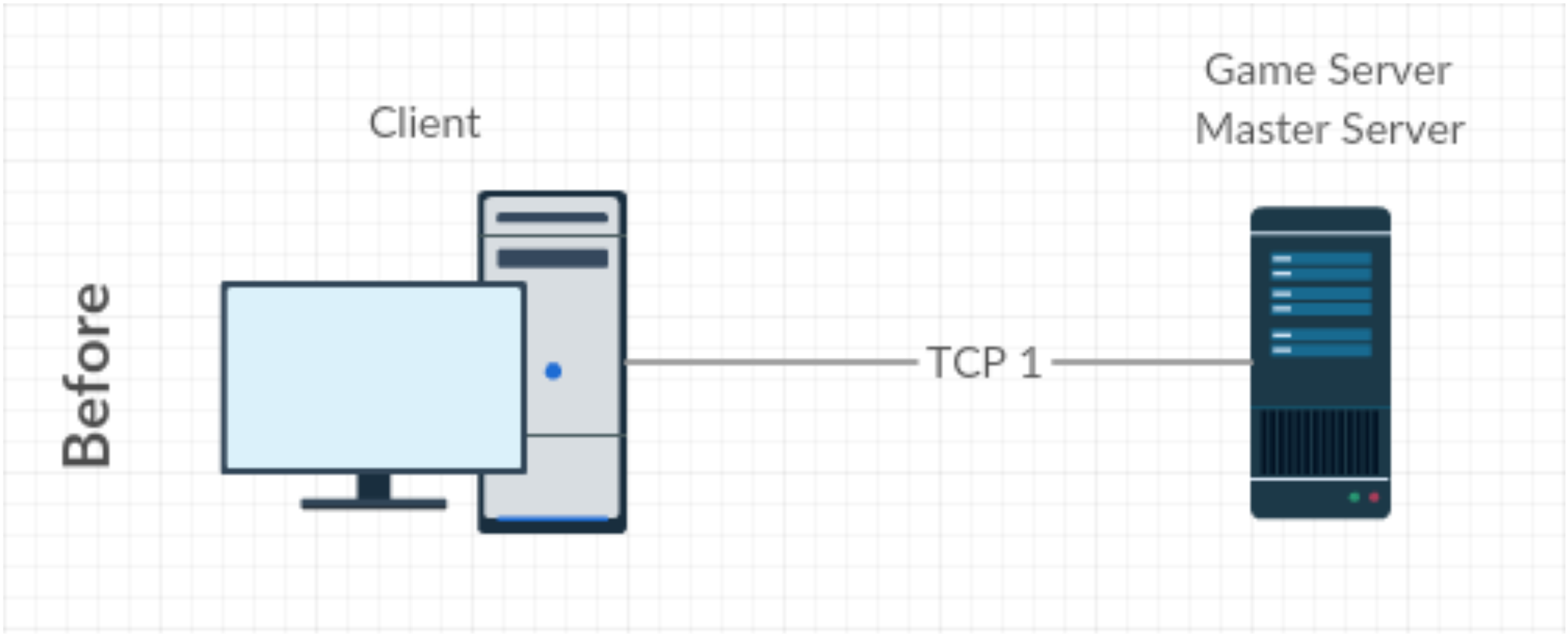
17-19 October 2017
MMXVII
HACK.LU

BUILDING ASYNC PROXY

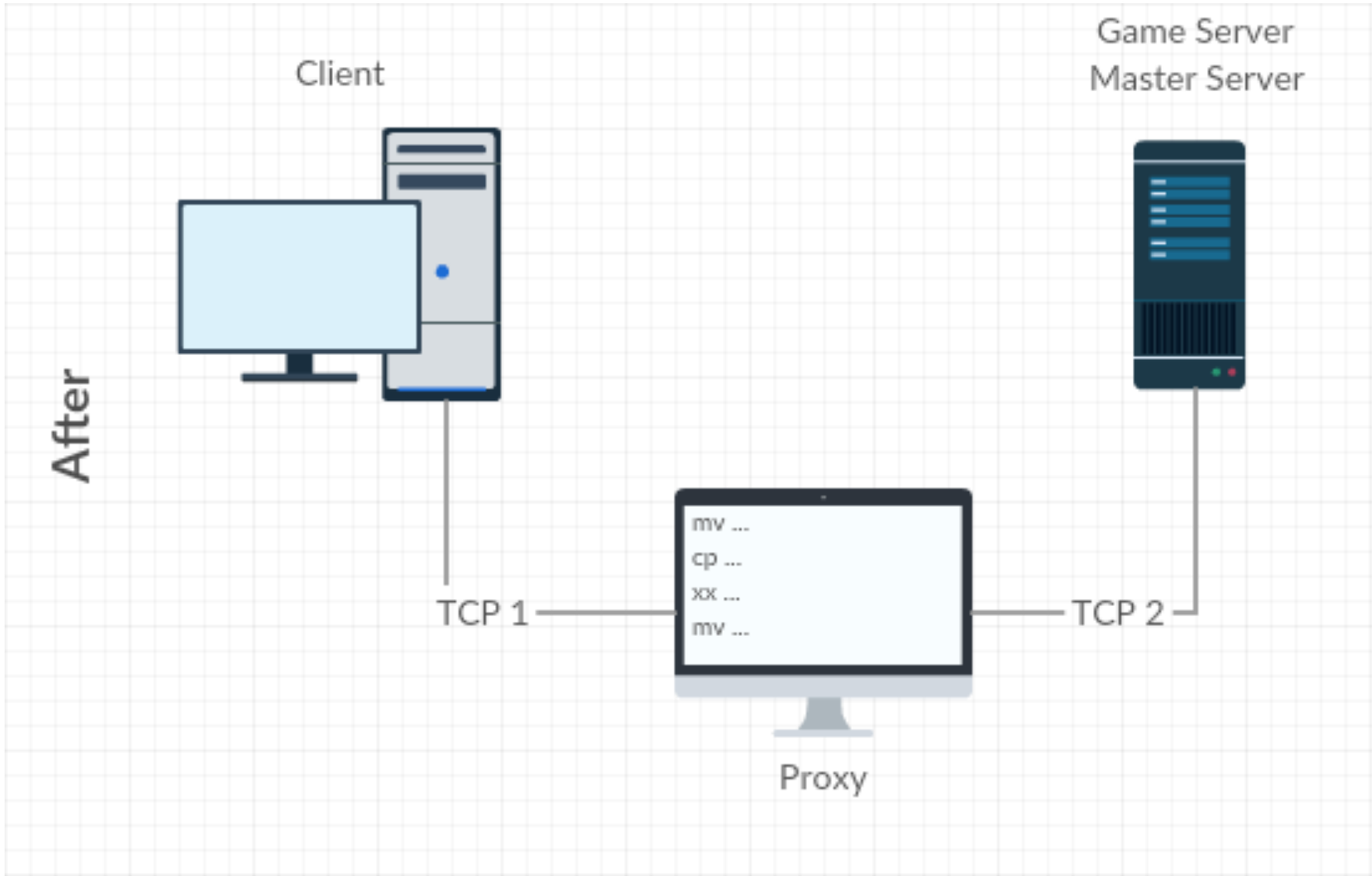
Proxy **script**:

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

BUILDING ASYNC PROXY



BUILDING ASYNC PROXY



BUILDING ASYNC PROXY

Re-routing the traffic:

- Editing */etc/hosts* file:

```
127.0.0.1     pwn3.hackeduniverse.com
```

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**:

- Buffer `write_buffer` for each TCP connection
- `handle_read()`: “Called when the asynchronous loop detects that a `read()` call on the channel’s socket will succeed”
- When `read()` is called, the data is `parsed()` then copied in the buffer of the other connection

```
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
```

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` (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

PICK UP ANY ITEM

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

`65 65` = Identifier (`e e`)

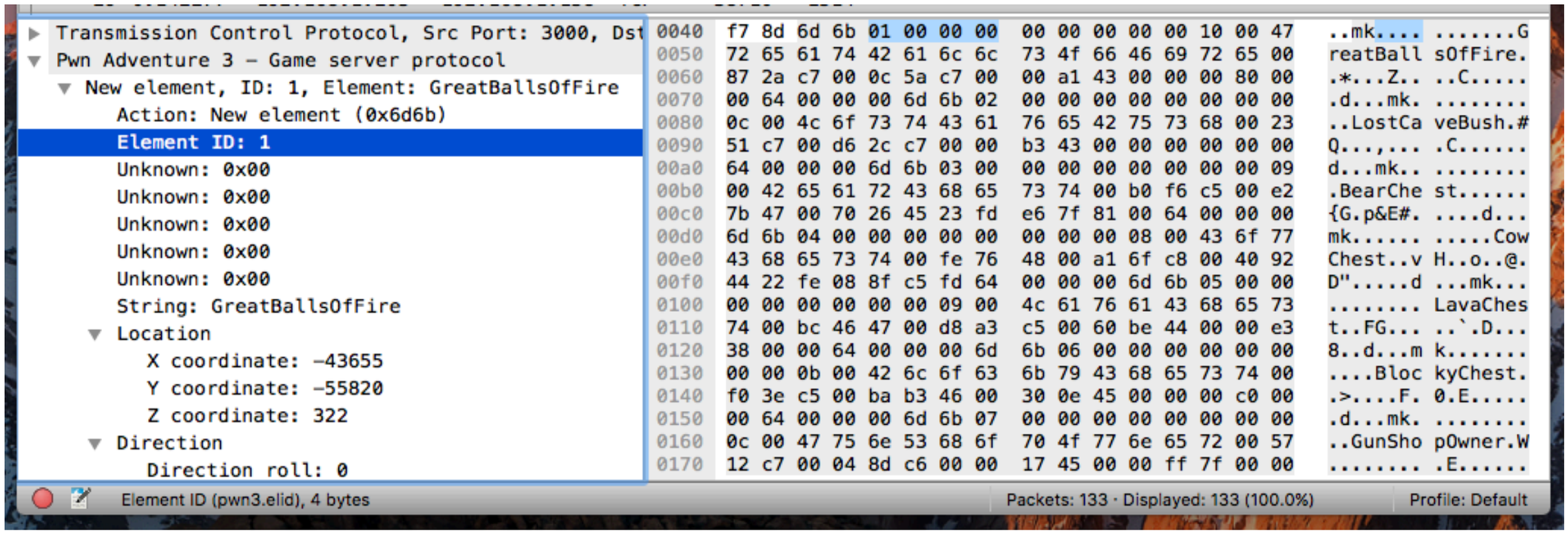
`N` = Element identifier

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

PICK 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)



17-19 October 2017
MMXVII
HACK.LU

PICK UP ANY ITEM

When to send?

- Decide when to trigger
- Easy to trigger

PICK 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
```

PICK 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
```

PICK UP ANY ITEM

Demo: Get the Great Balls of Fire

PICK UP ANY ITEM

It doesn’t work… **Why**?

PICK 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)

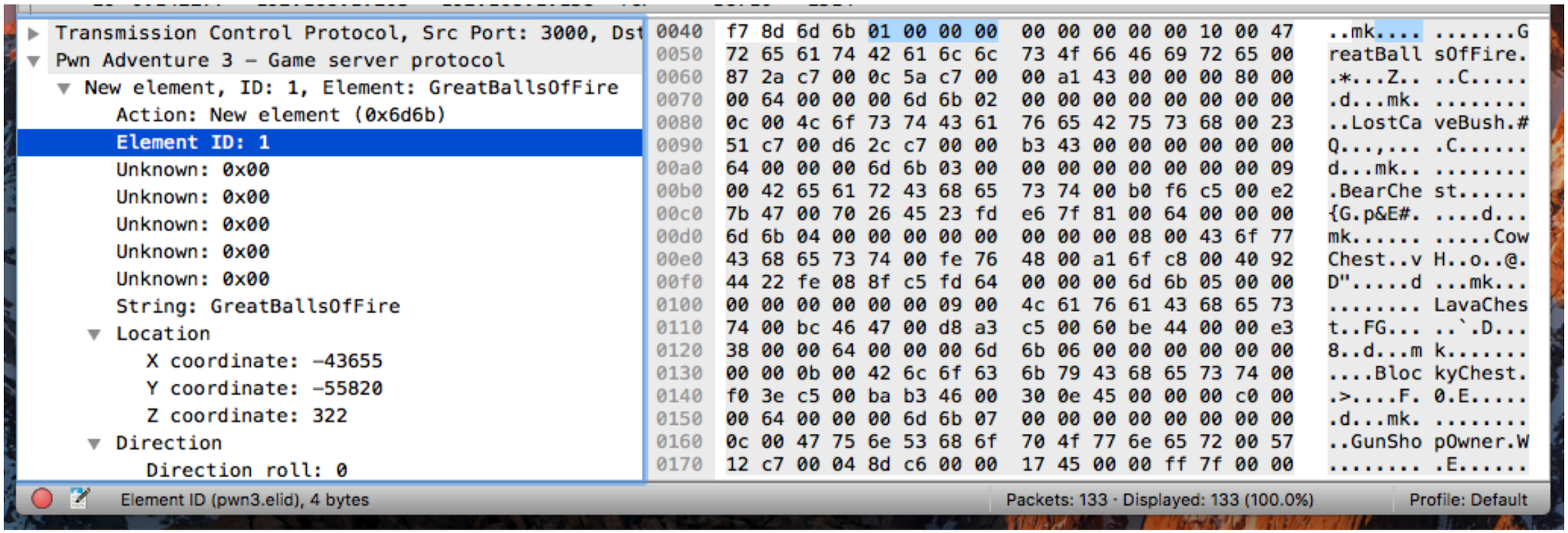
`F` = Direction where I move

`S` = Direction where I strafe

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

PICK UP ANY ITEM

What location to fake?



17-19 October 2017
MMXVII
HACK.LU

PICK 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("GreatBallsOfFire") in p_out:
        p_out += loc(-43655.0, -55820.0, 322.0)
        p_out += getme(1)

    return p_out
```

PICK UP ANY ITEM

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

CREATE ELEMENTS

Let’s generate **new element**!

17-19 October 2017
MMXVII
HACK.LU

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`

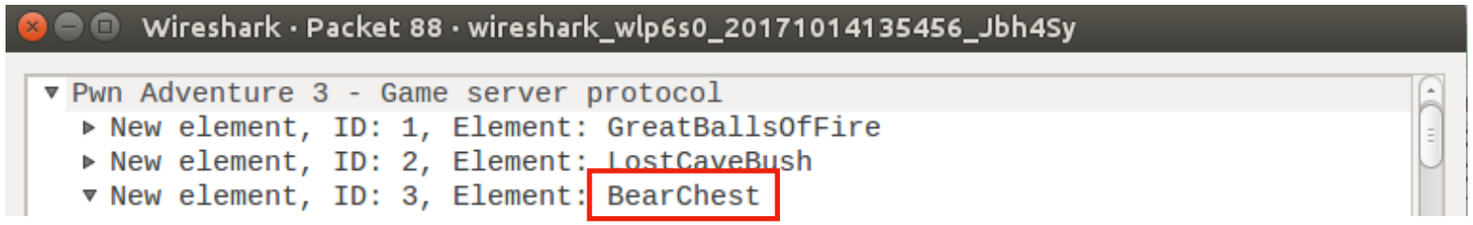
CREATE ELEMENTS

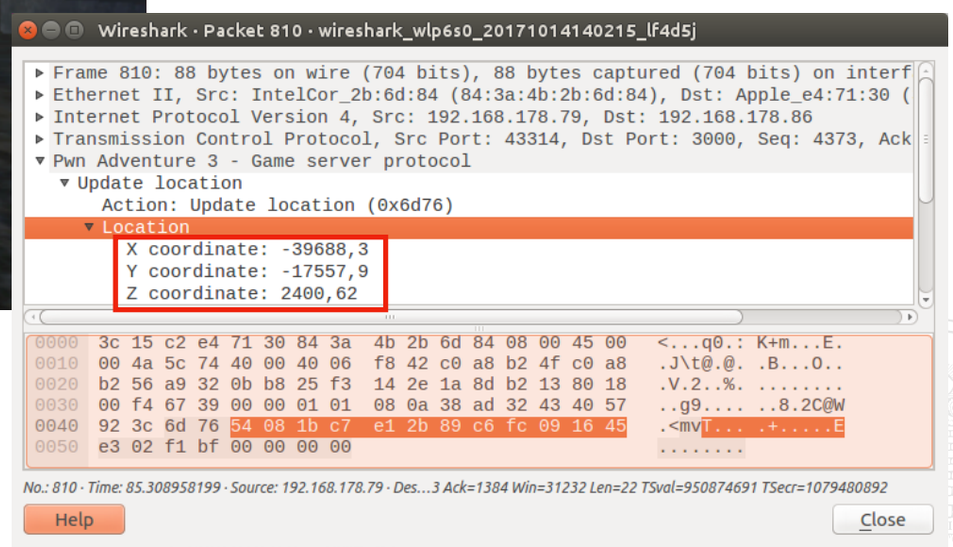
Create element packet:

```
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("fffHHHBBBB", x, y, z, 0, 0, 0, 0, 0, 0, 0)
return packet
```

CREATE ELEMENTS

What to create and **where** to put it?





17-19 October 2017
HACK.LU

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("fffHHHBBBB", 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

BUILDING ASYNC PROXY

Lab 5: 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

LOOT AND MONEY

Time to get **rich**!

LOOT AND 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 AND 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 AND MONEY

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

Why?

LOOT AND 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

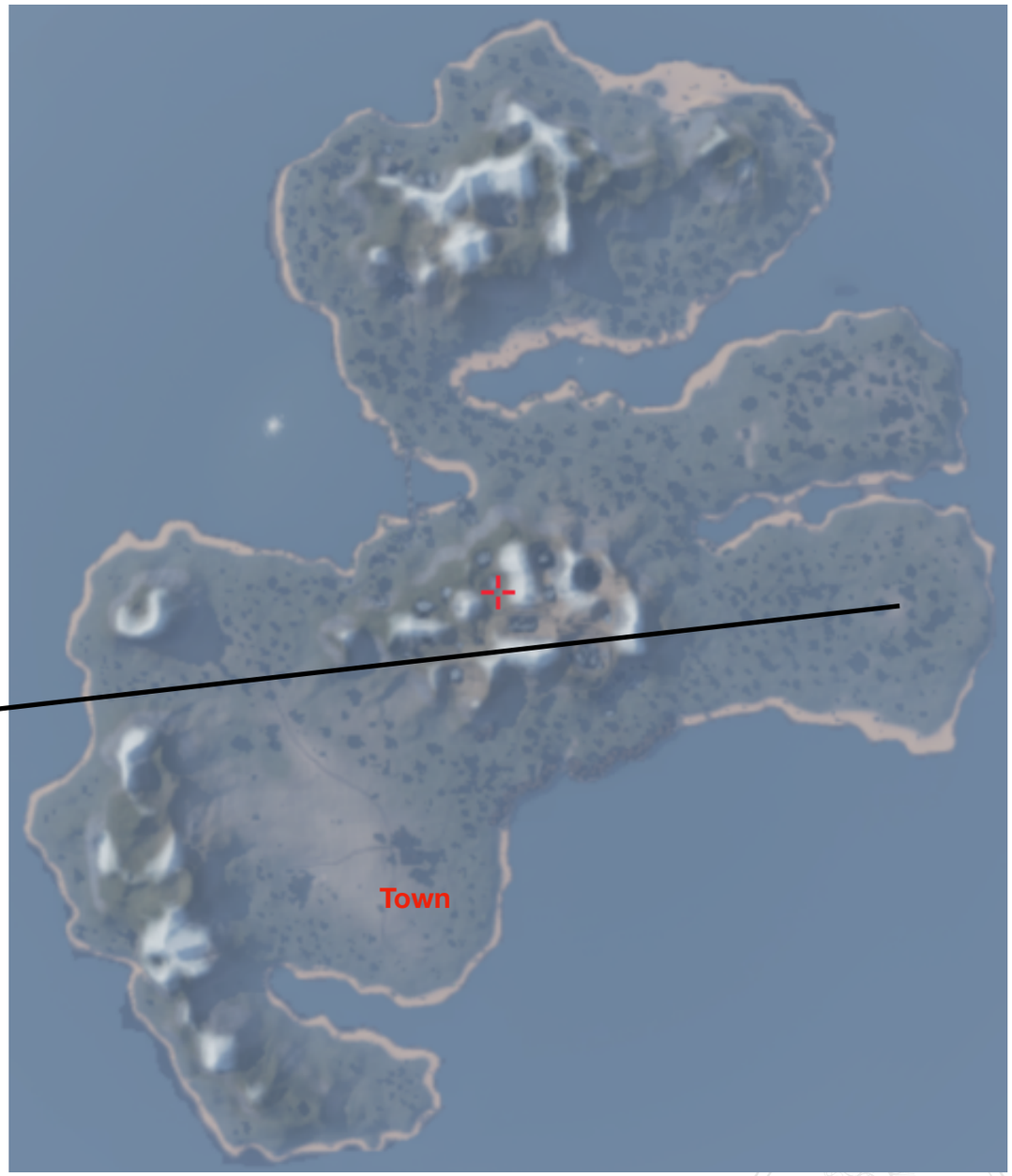
Resource:

- https://github.com/Foxmole/PwnAdventure3/blob/master/pwn3proxy.py

UNBEARABLE REVENGE

List of **quests**:

- Fire and Ice (RE binary)
- Pirates Treasure (crack me)
- 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)



17-19 October 2017
MMXVII
HACK.LU

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, 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

UNBEARABLE REVENGE

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

- Spawn on top
- Activate chest remotely

```
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("fffHHHBB", 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)
```

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":

    # 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*”

FIRE AND ICE

List of **quests**:

- Fire and Ice (RE binary)
- Pirates Treasure (crack me)
- 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)

17-19 October 2017
MMXVII
HACK.LU

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, 0)

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

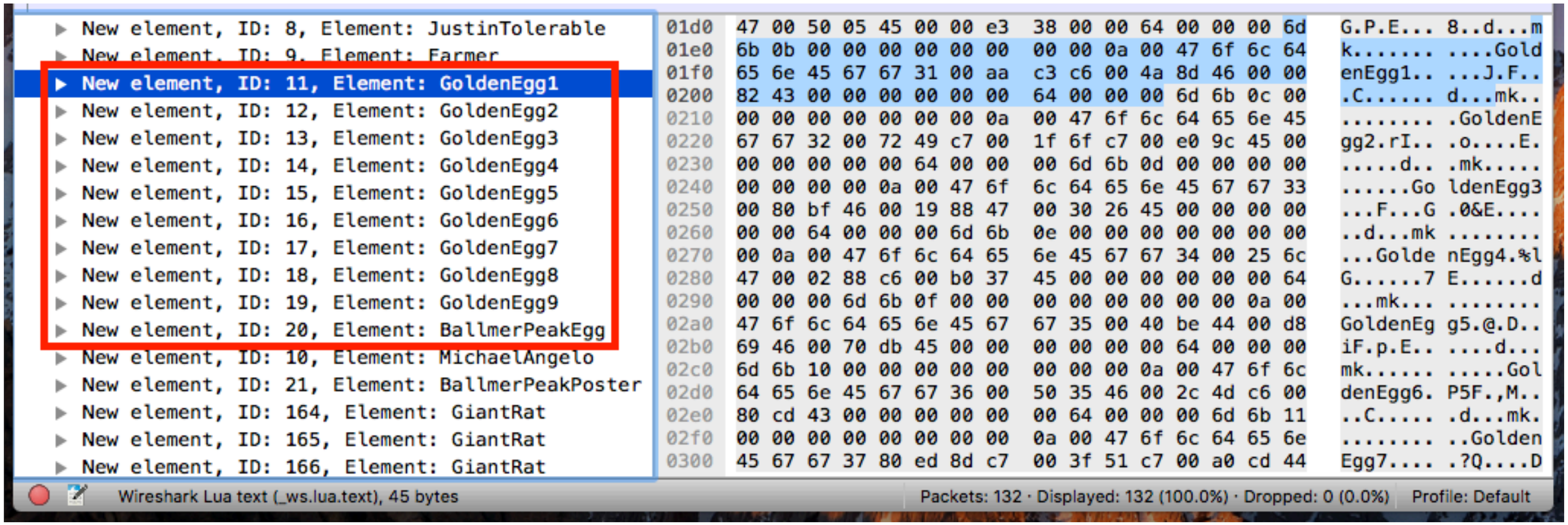
EGG HUNTER

Demo: Start Egg Finder quest

EGG HUNTER

What to do?

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



17-19 October 2017
MMXVII
HACK.LU

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("fffHHHBB", 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("egg3") in p_out:
        p_out += loc(24512.0, 69682.0, 2659.0)
        p_out += getme(13)

    ...

    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)
```

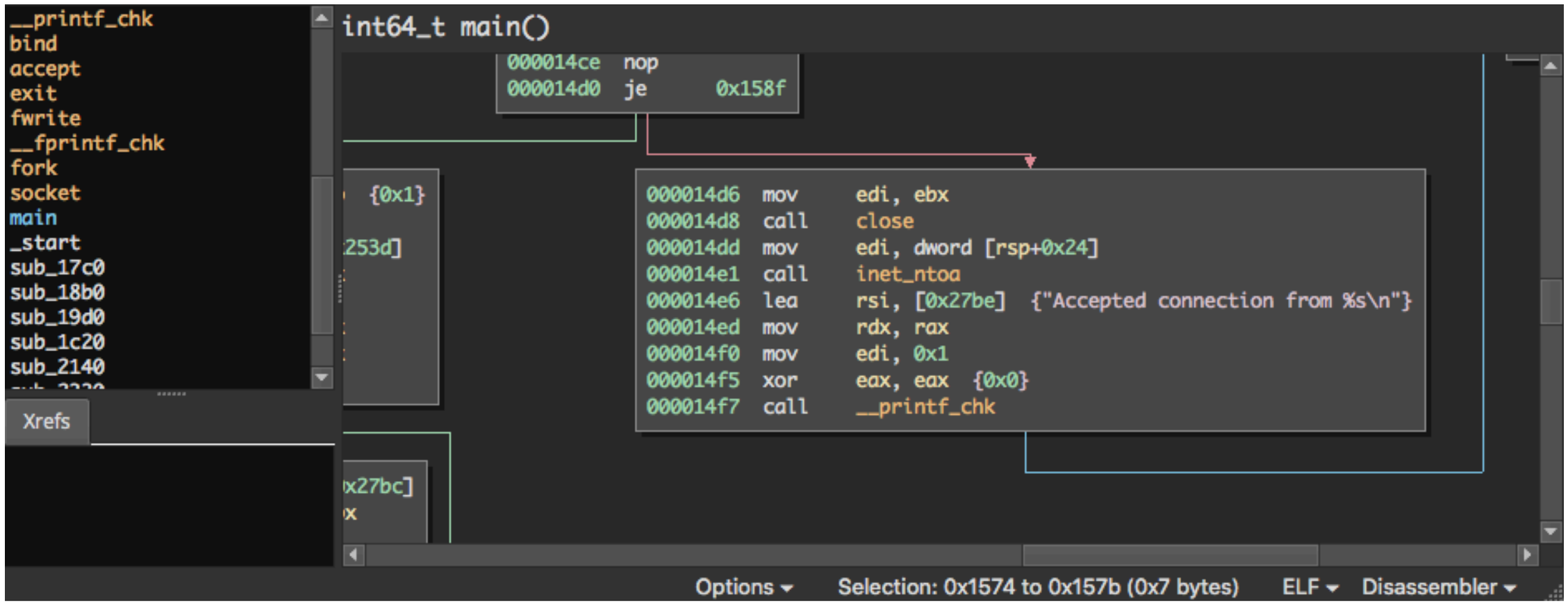
EGG HUNTER

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



17-19 October 2017
MMXVII
HACK.LU

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

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

RE BINARY

What **tools** to use?

- Disassembler
  - Hopper (free for x64)
  - **Binary Ninja**
  - IDA Pro
- Debugger
  - gdb
  - OllyDbg

Assembly:

- Low-level programming language
- Close to machine code instruction

RE BINARY

Limited set of **instructions**:

- mov
- call
- test
- cmp
- jmp, jnz, jb, …
- …

Variables are stored in different **areas**:

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

EGG FINDER

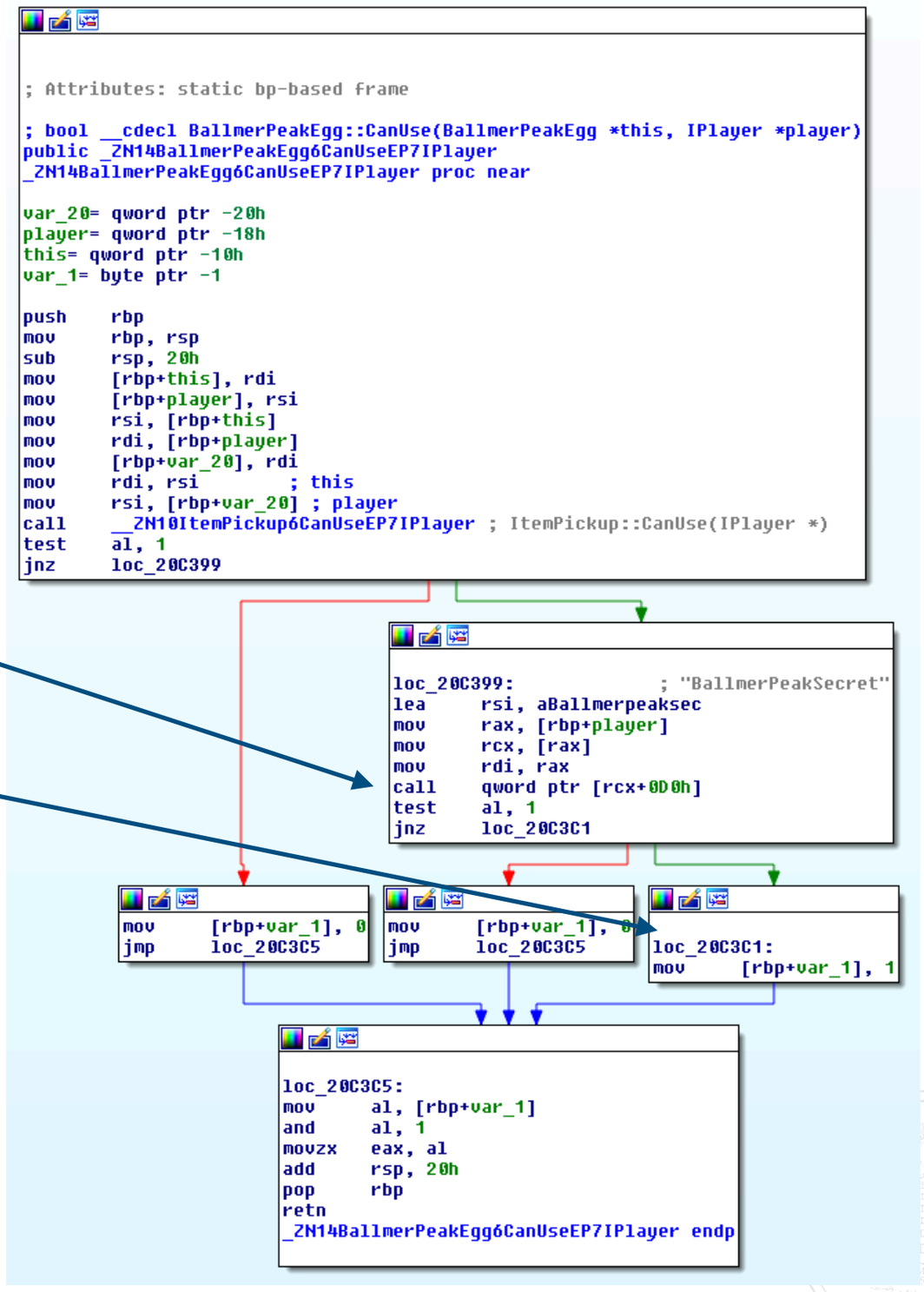
Look for the *Ballmer Peak Egg* related function:

Functions window

Function name	Segment	Start	Length	Locals	Arguments	R	F	L	S	B	T	=
BallmerPeakEgg::~BallmerPeakEgg()	.plt	0000000000111060	00000006	00000000	00000000	R	.	.	.	.	T	.
ActorFactory<BallmerPeakEgg>::ActorFactory(void)	.plt	0000000000113AF0	00000006	00000000	00000000	R	.	.	.	.	T	.
BallmerPeakEgg::~BallmerPeakEgg()	.plt	00000000001158B0	00000006	00000000	00000000	R	.	.	.	.	T	.
BallmerPeakEgg::BallmerPeakEgg(void)	.plt	0000000000115BB0	00000006	00000000	00000000	R	.	.	.	.	T	.
BallmerPeakEgg::BallmerPeakEgg(void)	.plt	000000000011DAB0	00000006	00000000	00000000	R	.	.	.	.	T	.
ActorFactory<BallmerPeakEgg>::ActorFactory(void)	.plt	0000000000125A80	00000006	00000000	00000000	R	.	.	.	.	T	.
BallmerPeakEgg::BallmerPeakEgg(void)	.text	000000000018E3A0	0000001B	00000018	00000000	R	.	.	S	B	T	.
BallmerPeakEgg::BallmerPeakEgg(void)	.text	000000000019EA00	000000F6	00000078	00000000	R	.	.	S	B	T	.
ActorFactory<BallmerPeakEgg>::ActorFactory(void)	.text	000000000019FAC0	0000001B	00000018	00000000	R	.	.	S	B	T	.
ActorFactory<BallmerPeakEgg>::ActorFactory(void)	.text	00000000001A5E10	00000039	00000018	00000000	R	.	.	S	B	T	.
ActorFactory<BallmerPeakEgg>::CreateActor(void)	.text	00000000001A5E70	0000003B	00000038	00000000	R	.	.	S	B	T	.
BallmerPeakEgg::CanUse(IPlayer *)	.text	000000000020C360	00000073	00000028	00000000	R	.	.	S	B	T	.
BallmerPeakEgg::~BallmerPeakEgg()	.text	000000000020C3E0	0000001B	00000018	00000000	R	.	.	S	B	T	.
BallmerPeakEgg::~BallmerPeakEgg()	.text	000000000020C400	00000028	00000018	00000000	R	.	.	S	B	T	.
BallmerPeakEgg::~BallmerPeakEgg()	.text	000000000020C430	0000001B	00000018	00000000	R	.	.	S	B	T	.

EGG FINDER

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


    int ret;
    int canuse;




    ret = ItemPickup::CanUse(this, player);
    if (ret)
    {

        ret = Player::????(“BallmerPeakSecret”);
        if (ret)
        {
            canuse = true;
        }
        else
        {
            canuse = false;
        }

    }
    else
    {
        canuse = false;
    }

    return canuse; // We want to return true



}
```



17-19 October 2017
HACK.LU

EGG FINDER

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

- gdb
- Breakpoint at `call qword ptr [rcx+0D0h]`

```
$ 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 FINDER

Let’s trigger the function **BallmerPeakEgg:CanUse**:

- Type BallmerPeakEgg in the chat box

```
def parse(p_out):

    p_in = ""
def getme(item):
    return "ee" + struct.pack("I", item)

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

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 FINDER

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

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

EGG FINDER

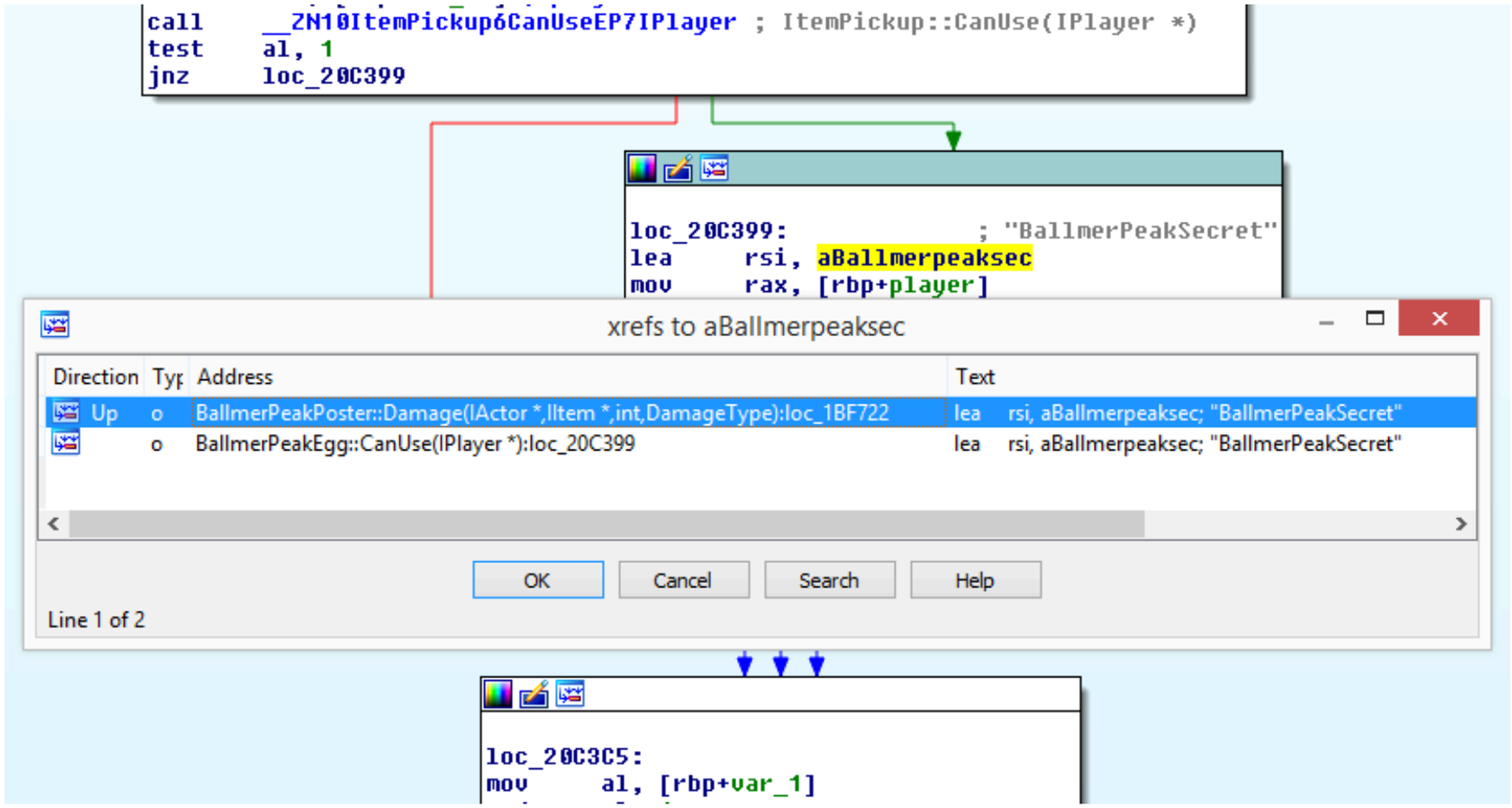
Final pseudo-code translation for **BallmerPeakEgg:CanUser**:

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

    int ret;
    int canuse;

    ret = ItemPickup::CanUse(this, player); // Seems to be always true
    if (ret)
    {

        ret = Player::HasPickedUp(this, “BallmerPeakSecret”);
        if (ret)
        {
            canuse = true;
        }
        else
        {
            canuse = false;
        }

    }
    else
    {
        canuse = false;
    }

    return canuse; // We want to return true

}
```

EGG FINDER

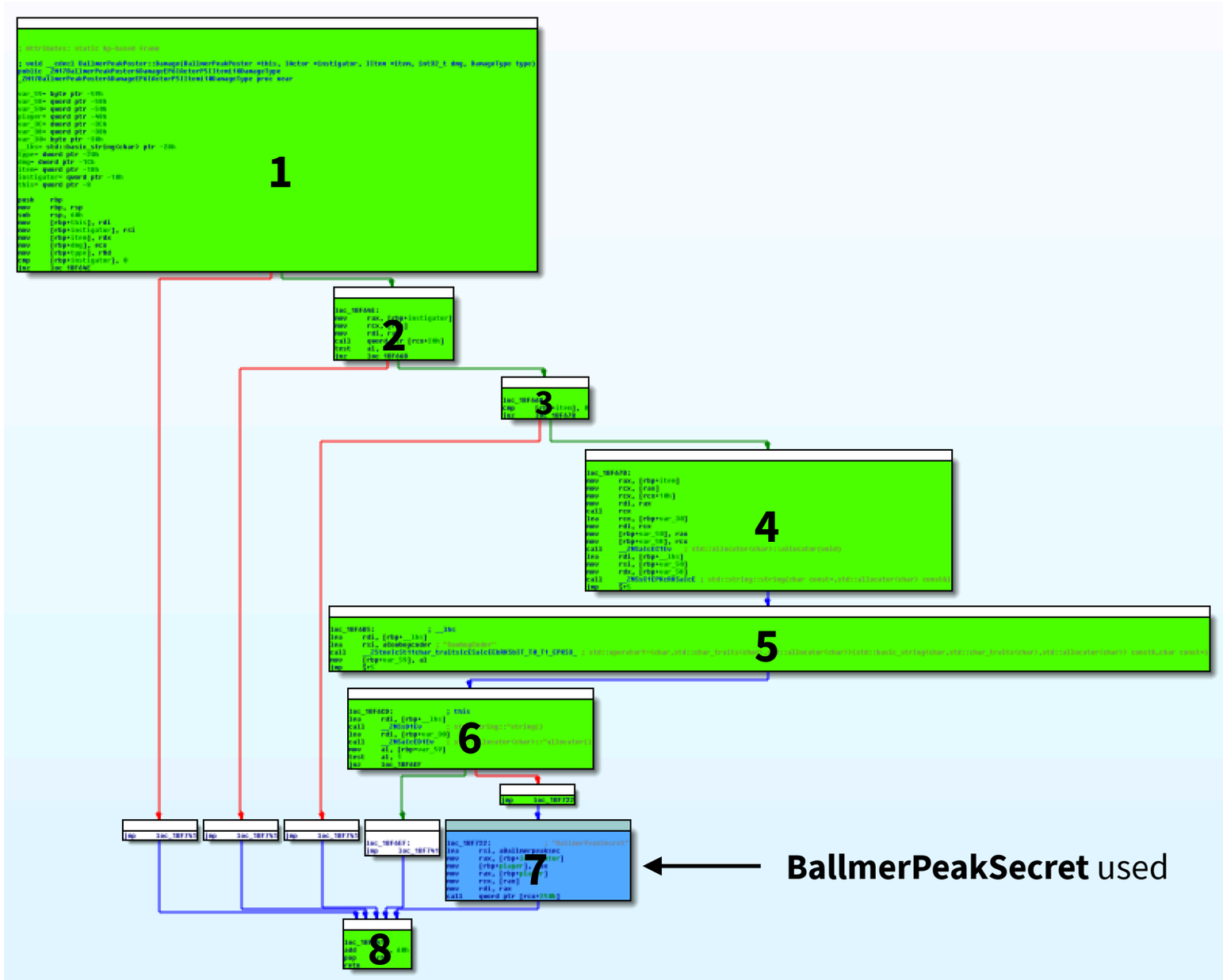
It seems that at some point, we need to **pick up** a "**BallmerPeakSecret**" in order to **use** (pick up) the **BallmerPeakEgg**.

Where is the "**BallmerPeakSecret**" string also used?



EGG FINDER

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



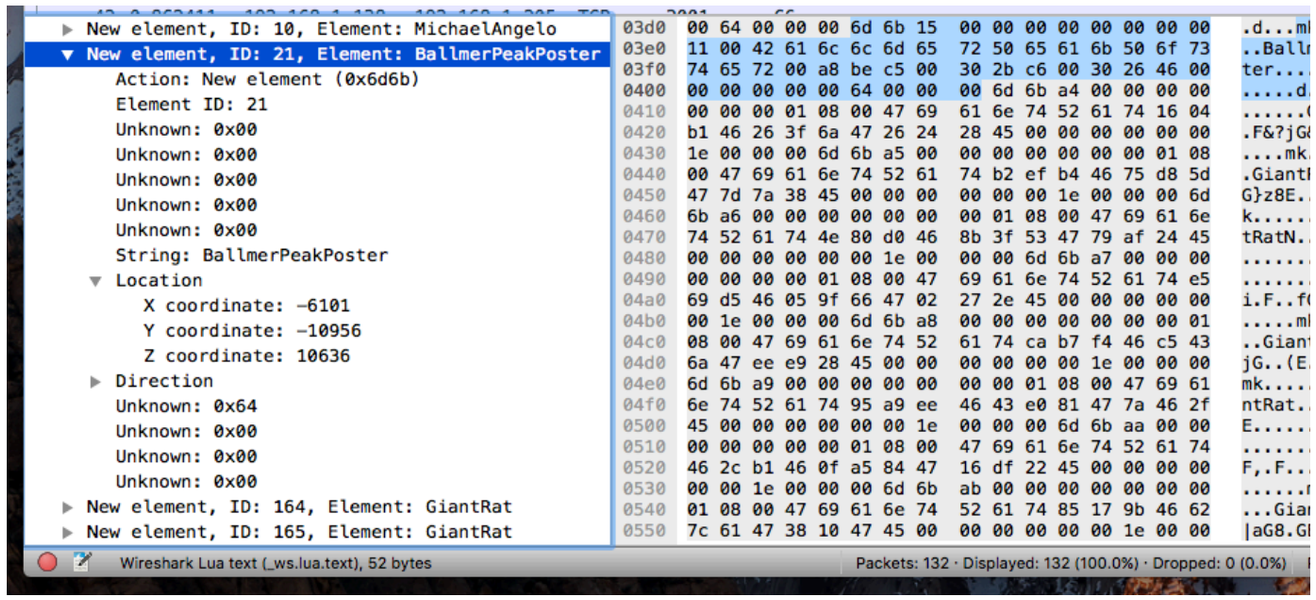
EGG FINDER

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

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

EGG FINDER

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, 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 FINDER

1. Entry point (disassembler)

```
; 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
```

rdi rsi rdx ecx r8d

Assign arguments to local variables

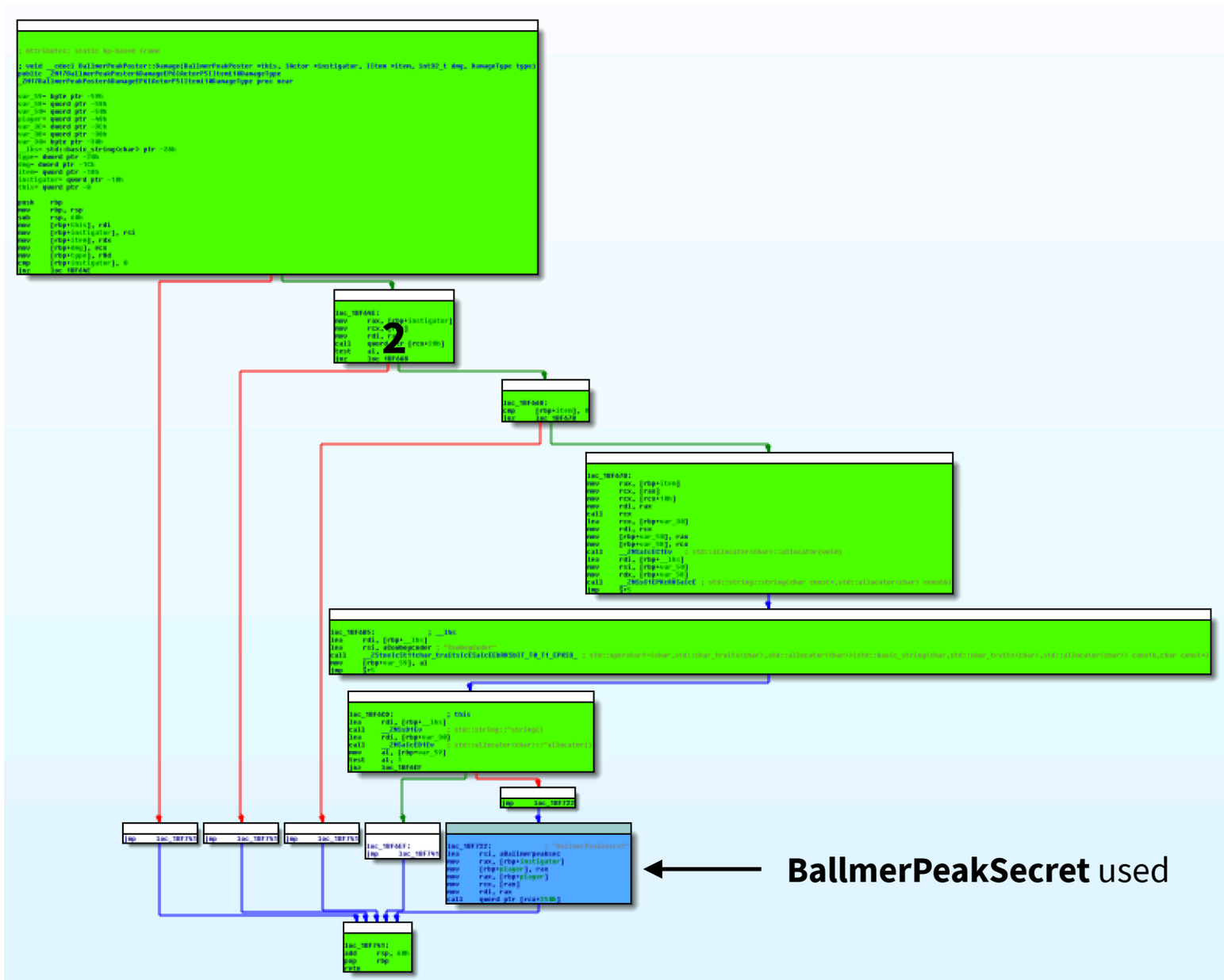
Make sure the argument instigator != 0

1. Entry point (debugger)

```
(gdb) x/x $rsi
0x5e0fcb0: 0x230c2980
```

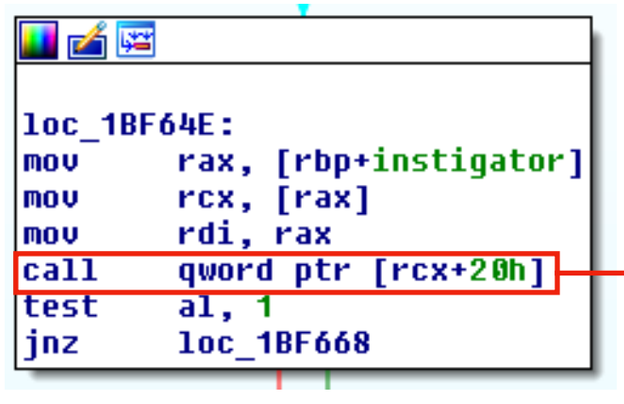
EGG FINDER

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



EGG FINDER

2. Block (disassembler)



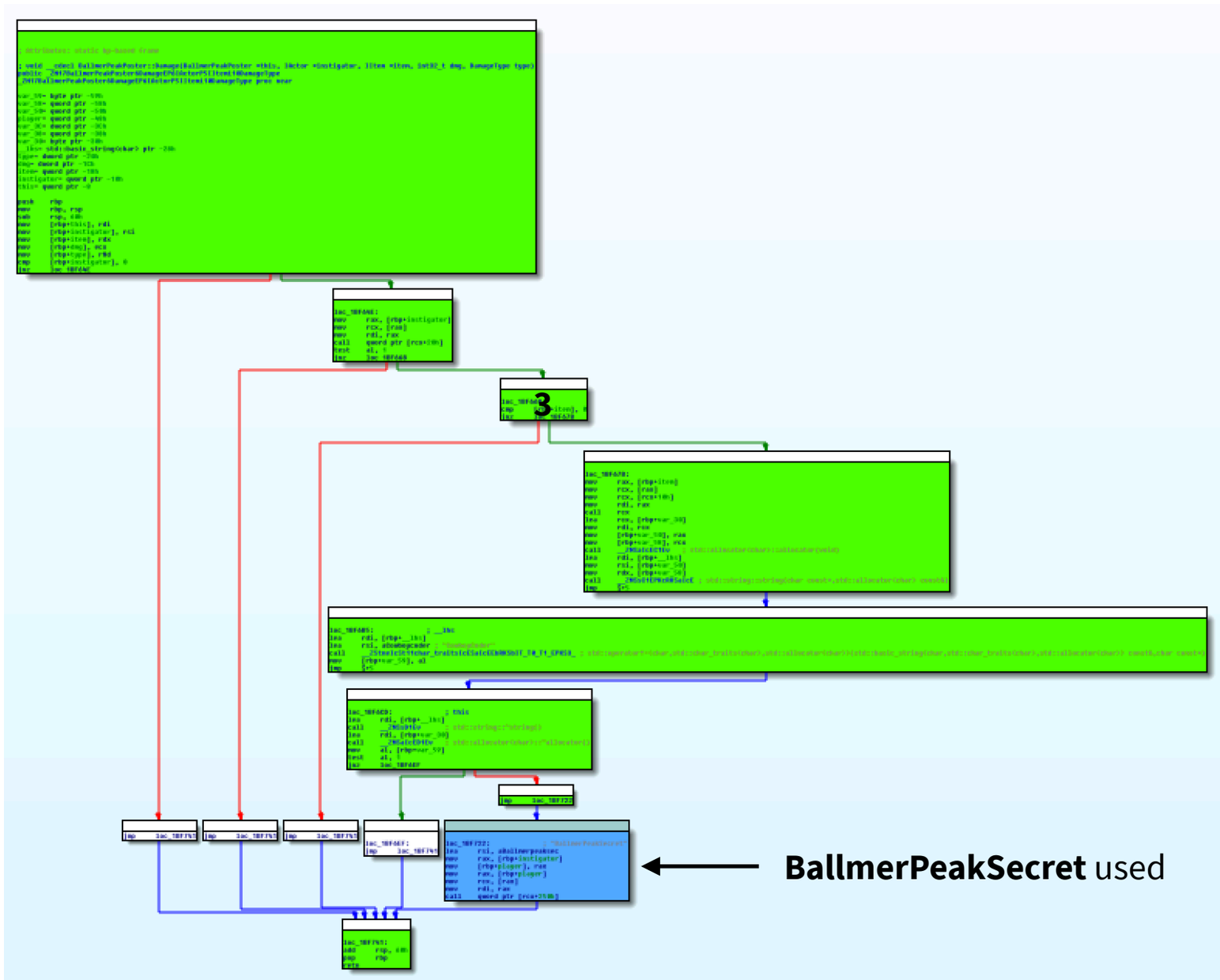
What function is called?

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=0x5e0fcb0)
(gdb) finish
Value returned is $1 = true
```

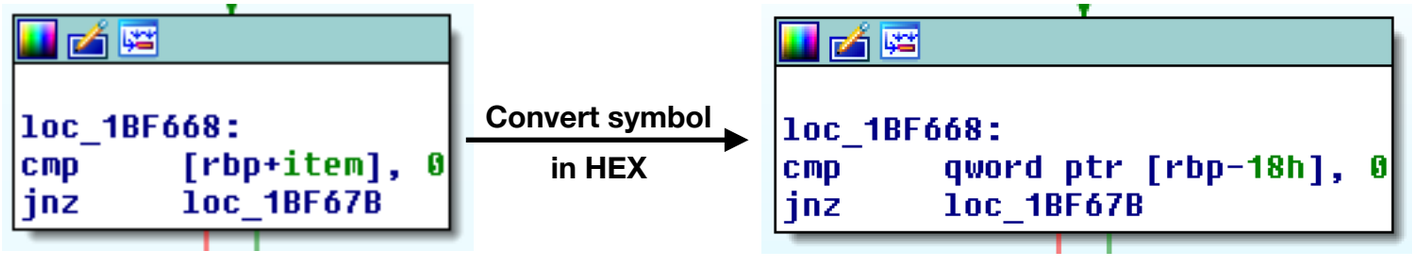
EGG FINDER

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



EGG FINDER

3. Block (disassembler)

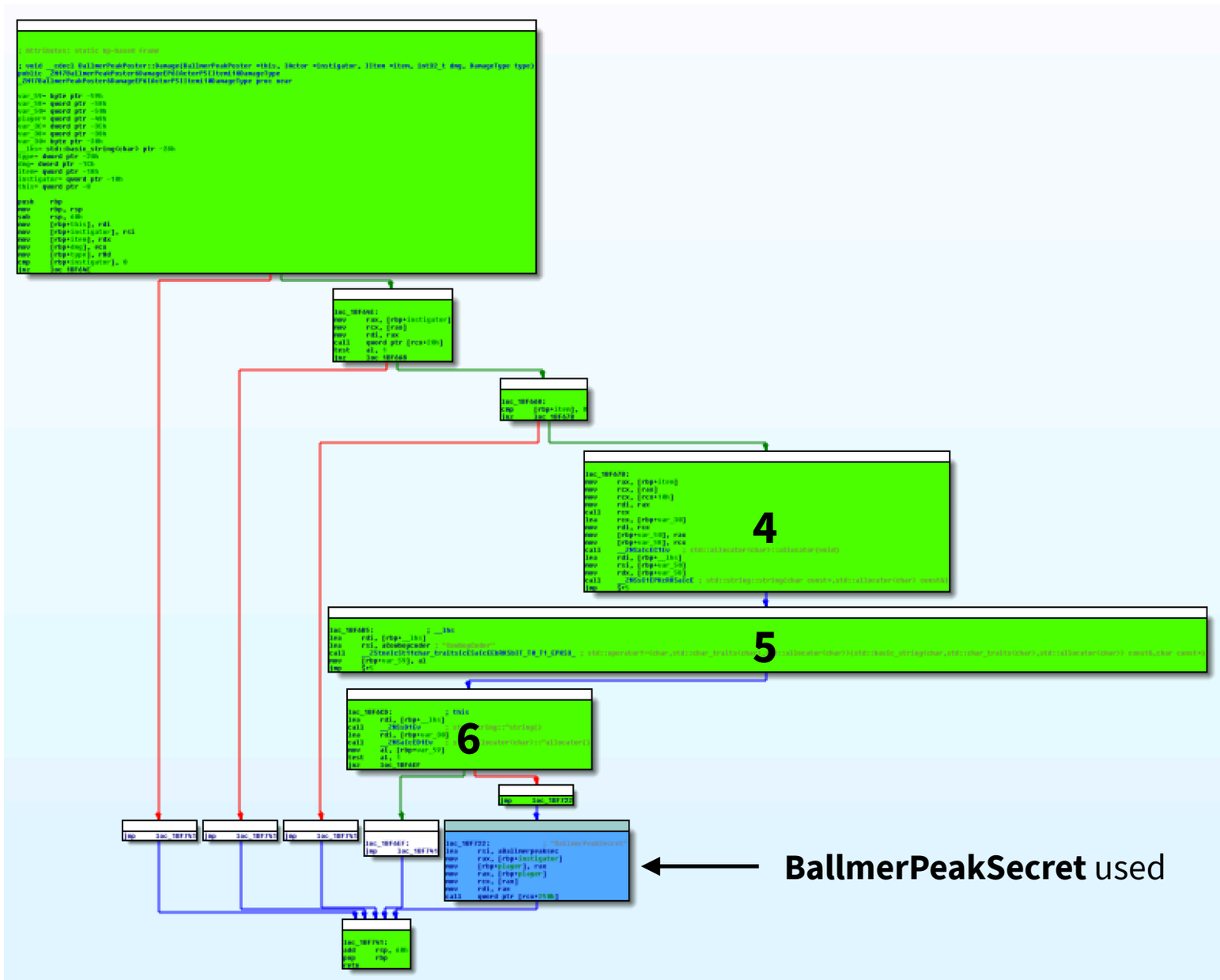


3. Block (debugger)

```
(gdb) x/xg $rbp-0x18
0x7fffd46c5988: 0x0000000005bc8290
```

EGG FINDER

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



EGG FINDER

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_1BF6B5:             ; __lhs
lea     rdi, [rbp+__lhs]
lea     rsi, aCowboycoder ; "CowboyCoder"
call    __ZStneIcSt11char_traitsIcESaIcEEbRKSbIT_T0_T1_EPKS3_ ; -
mov     [rbp+var_59], al
jmp     $+5
```

3

```
loc_1BF6CD:             ; this
lea     rdi, [rbp+__lhs]
call    __ZNSsD1Ev      ; std::string::~string()
lea     rdi, [rbp+var_30]
call    __ZNSaIcED1Ev   ; std::allocator<char>::~allocator()
mov     al, [rbp+var_59]
test    al, 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
0x7fffd46c5978: 0x0000000005db9d38
(gdb) x/s 0x0000000005db9d38
0x5db9d38: "GreatBallsOfFire"
```

EGG FINDER

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
```

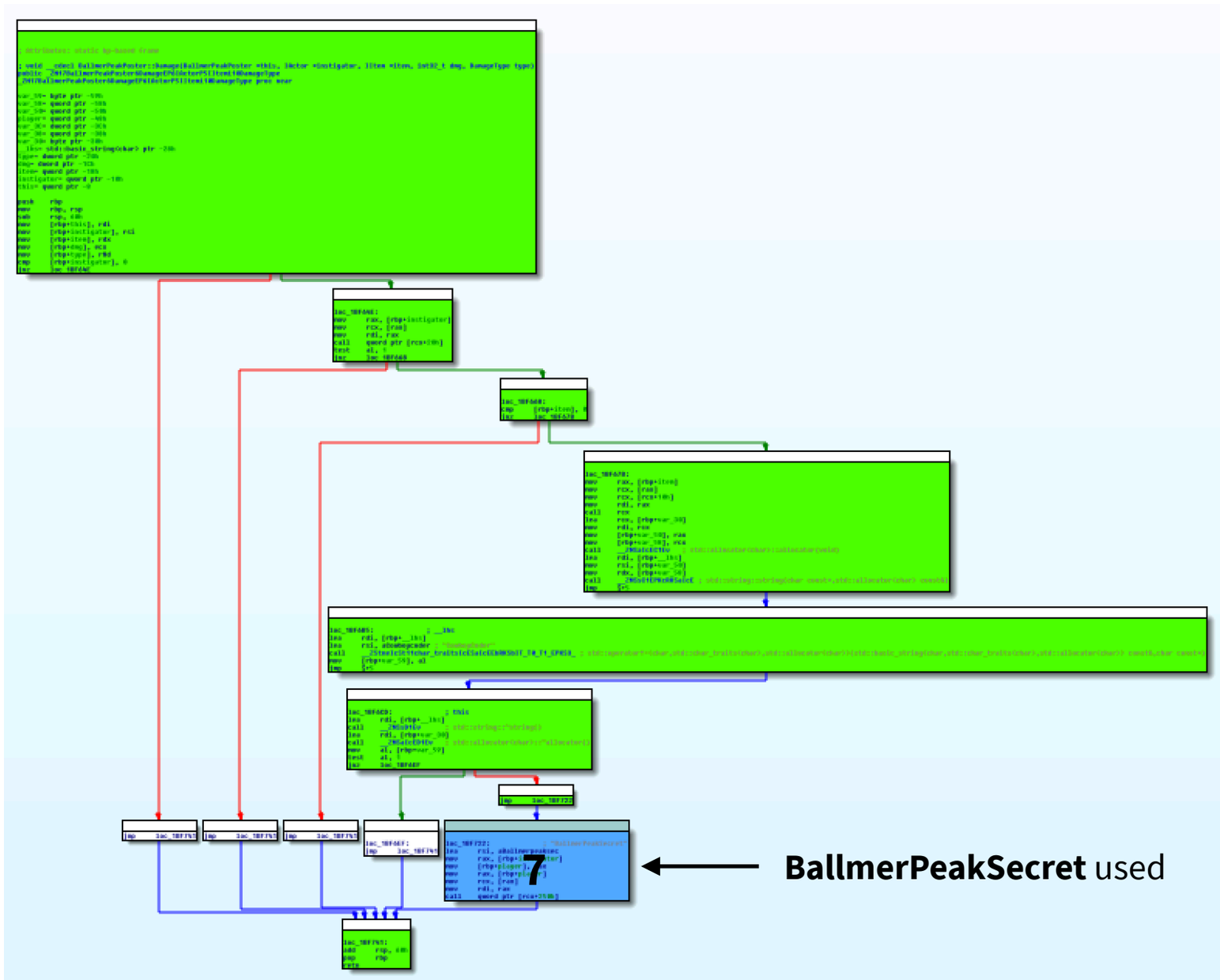
Once bought, **come back** to *Ballmer Peak*:

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

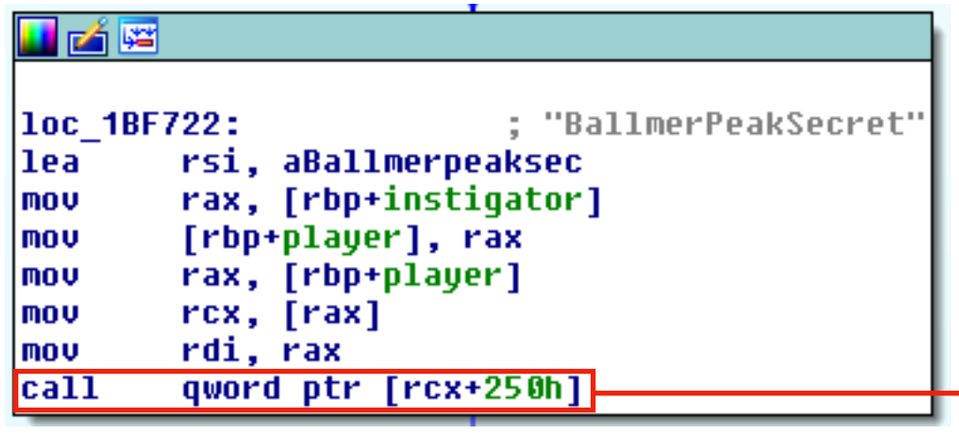
And **shoot** at the *poster*

EGG FINDER

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



7. Block (disassembler)



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 FINDER

`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 FINDER

Demo: Get that Ballmer Peak Egg

FIRE AND ICE

List of **quests**:

- Fire and Ice (RE binary)
- *Pirates Treasure (crack me)*
- *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, 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”

17-19 October 2017
MMXVII
HACK.LU

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

List all **functions** in **Magmarok object:**

Function name	Segment	Start
Magmarok::Damage(IActor *,IItem *,int,DamageType)	.text	000000000013AF30
Magmarok::GetAggressionRadius(void)	.text	000000000013BEE0
Magmarok::GetAttackTime(void)	.text	000000000013BF00
Magmarok::GetDeathMessage(void)	.text	000000000013BDA0
Magmarok::GetDisplayName(void)	.text	000000000013BD60
Magmarok::GetMaxHealth(void)	.text	000000000013BD50
Magmarok::GetMaximumDamageDistance(void)	.text	000000000013BE10
Magmarok::GetRangedAttackDistance(void)	.text	000000000013BE50
Magmarok::IsElite(void)	.text	000000000013BD80
Magmarok::Magmarok(void)	.plt	000000000010E6B0
Magmarok::Magmarok(void)	.text	000000000013ACF0
Magmarok::OnKilled(IActor *,IItem *)	.text	000000000013AE50
Magmarok::OnPrepareAttack(Actor *)	.text	000000000013B8F0
Magmarok::ShouldAttack(void)	.text	000000000013B0B0
Magmarok::ShouldAttackFromRange(void)	.text	000000000013BE30
Magmarok::ShouldAttackMultipleTargets(void)	.text	000000000013BEB0
Magmarok::ShouldMove(void)	.text	000000000013BE90
Magmarok::ShouldWander(void)	.text	000000000013BE70
Magmarok::Tick(float)	.text	000000000013B0D0
Magmarok::~Magmarok()	.plt	000000000011AE00
Magmarok::~Magmarok()	.plt	0000000000124650
Magmarok::~Magmarok()	.text	000000000013BD00
Magmarok::~Magmarok()	.text	000000000013BD20
Magmarok::~Magmarok()	.text	000000000013BF20

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 weapon/spell will give half damage

FIRE AND ICE

Potential **integer overflow**?

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

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.”

```
1111111111111111111111111111111111111111111111111111111111111111 = 18446744073709551615
1111111111111111111111111111111111111111111111111111111111111110 = 18446744073709551614
                         .    .    .
0000000000000000000000000000000000000000000000000000000000000010 = 2
0000000000000000000000000000000000000000000000000000000000000001 = 1
0000000000000000000000000000000000000000000000000000000000000000 = 0
1111111111111111111111111111111111111111111111111111111111111111 = -1
1111111111111111111111111111111111111111111111111111111111111110 = -2
```

FIRE AND ICE

Potential **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:

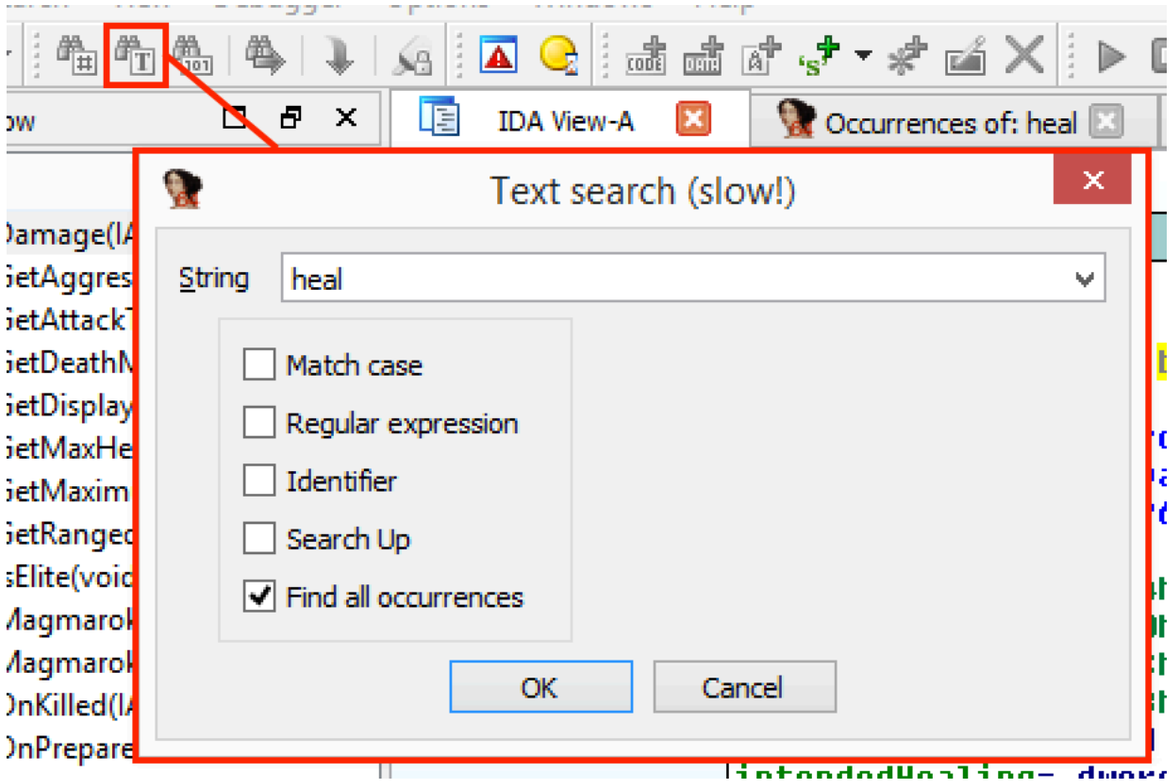
```
Magmarok::Damage(weaponDmg, weaponType) {

    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

What is **`Magmarok::Tick`** function?

A “real time” function that is triggered regularly in order to update the state and action 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 **prevent** the first **integer overflow**:

```
maximumHealing = 10000 - magmarok.currentHealth # 0x13AFD3

if (intendedHealing > maximumHealing) # 0x13AFDC
{
    intendedHealing = maximumHealing  # 0x13AFE8
}
```

rax = Magmarok object
verify with debugging

```
mov     ecx, 2710h
mov     rax, 4
cvtsi2ss xmm1, rax
movss   xmm2, [rbp+var_3C]
mulss   xmm2, xmm0
mulss   xmm2, xmm1
cvttss2si edx, xmm2
mov     [rbp+intendedHealing], edx
mov     rax, [rbp+var_38]
sub     ecx, [rax+38h]
mov     [rbp+maximumHealing], ecx
mov     ecx, [rbp+intendedHealing]
cmp     ecx, [rbp+maximumHealing]
jbe     loc_13AFEB
```

```
mov     eax, [rbp+maximumHealing]
mov     [rbp+intendedHealing], eax
```

```
loc_13AFEB:
mov     eax, 0
sub     eax, [rbp+intendedHealing]
mov     [rbp+dmg], eax
jmp     loc_13B023
```

FIRE AND ICE

Comparison done with **JBE**:

```
cmp     ecx, [rbp+maximumHealing]
jbe     loc_13AFEB
```

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

FIRE AND ICE

What if **`magmarok.currentHealth`** is **higher** than **10000**?

```
maximumHealing = 10000 - magmarok.currentHealth

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

`maximumHealing` will be **negative**:

```
00000000000000000000000000000000000000000000000000000000000000010 = 2
00000000000000000000000000000000000000000000000000000000000000001 = 1
00000000000000000000000000000000000000000000000000000000000000000 = 0
11111111111111111111111111111111111111111111111111111111111111111 = -1
11111111111111111111111111111111111111111111111111111111111111110 = -2
```

E.g. if **`maximumHealing`** = -1, 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
```

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

Binary Patching

17-19 October 2017
MMXVII
HACK.LU

BINARY PATCHING

When you **run** an **application**, you **execute instructions** that has 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**:

Functions window

Function name	Segment	Start
ClientHandler::Sprint(void)	.text	0000000000141AC0
ClientHandler::Sprint(void)	.plt	000000000011A2A0
ClientHandler::Sprint(void)::$_5::operator() const(void)	.text	0000000000148A90
ClientWorld::Sprint(bool)	.text	000000000013C540
GameServerConnection::Sprint(bool)	.text	00000000001C3690
GameServerConnection::Sprint(bool)	.plt	000000000011C590
GameServerConnection::Sprint(bool)::$_47::operator() const(void)	.text	00000000001C8890
LocalWorld::Sprint(bool)	.text	00000000001BDAB0
Player::GetJumpSpeed(void)	.text	000000000015EA80
Player::GetJumpSpeed(void)	.plt	000000000010ED90
Player::GetSprintMultiplier(void)	.text	000000000015EA60
Player::GetSprintMultiplier(void)	.plt	0000000000117CD0
Player::GetWalkingSpeed(void)	.text	000000000015EA40
Player::GetWalkingSpeed(void)	.plt	0000000000123FB0
Player::SetSprintState(bool)	.text	00000000001515C0
Player::SetSprintState(bool)	.plt	000000000010F160
ServerWorld::Sprint(bool)	.text	0000000000227D50

17-19 October 2017
MMXVII
HACK.LU

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
```

BINARY PATCHING

Tool list:

- Python 
- Capstone 
- Keystone 

BINARY PATCHING

Find the **offset**:

```
.text:000000000015EA60
.text:000000000015EA60 push    rbp
.text:000000000015EA61 mov     rbp, rsp
.text:000000000015EA64 mov     rax, 3
.text:000000000015EA6E cvtsi2ss xmm0, rax
.text:000000000015EA73 mov     [rbp+this], rdi
.text:000000000015EA77 pop     rbp
.text:000000000015EA78 retn
```

BINARY PATCHING

Verify **new instruction** won't **overwrite** the next **legitimate instruction** (`cvtsi2ss xmm0, rax`). For this, we need to **calculate** the **size** 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 **size**: If It 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/Foxmole/PwnAdventure3/blob/master/binpatcher.py

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

FIRE AND ICE

Demo: Let’s run supa fast!

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
```

17-19 October 2017
MMXVII
HACK.LU

LIBRARY HOOKING

Binary patching:

- Edit binary to modify the binary in our advantage
- Small changes: Overwrite one instruction
- If we want to add more complex logic:
  - Code cave
  - Hooking

CODE CAVE

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)

CODE CAVE

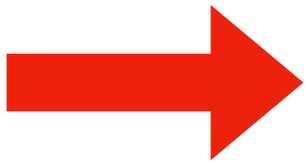
	ASM	HEX

Initial function	push ebp
	mov ebp, esp
	sub esp, 0x30
	push eax
	push ecx
	push 0x24
	push eax
	call sub_401020
	
	...
		55
		89 e5
		83 ec 30
		50
		51
		6a 24
		50
		e8 fc ff ff ff
		
		...

Another function	push ebp
	mov ebp, esp
	sub esp, 0x30
	sub eax, ebx
	push edi
	
	...
		55
		89 e5
		83 ec 30
		29 d8
		57
		
		...

Code cave	nop nop nop nop nop ...	90 90 90 90 90 ...

code cave



	ASM	HEX

Initial function	push ebp
	mov ebp, esp
	sub esp, 0x30
	jmp code_cave
	
	call sub_401020
	
	...
		55
		89 e5
		83 ec 30
		e9 a0 41 06 00
		
		e8 fc ff ff ff
		
		...

Another function	push ebp
	mov ebp, esp
	sub esp, 0x30
	sub eax, ebx
	push edi
	
	...
		55
		89 e5
		83 ec 30
		29 d8
		57
		
		...

Code cave	push eax push ecx push 0x24 push eax pushad pushfd new code add esp, 0xa0 popfd popad jmp init_after_jmp ...	50 51 6a 24 50 60 9c new code 83 c4 0a 9d 61 e9 b0 30 fc ff ...

17-19 October 2017
MMXVII
HACK.LU

CODE CAVE

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 (C/C++)
- Use *LD_PRELOAD* (in Linux)
  - Preloading a library
  - Its functions will be used before others of the same name in later libraries

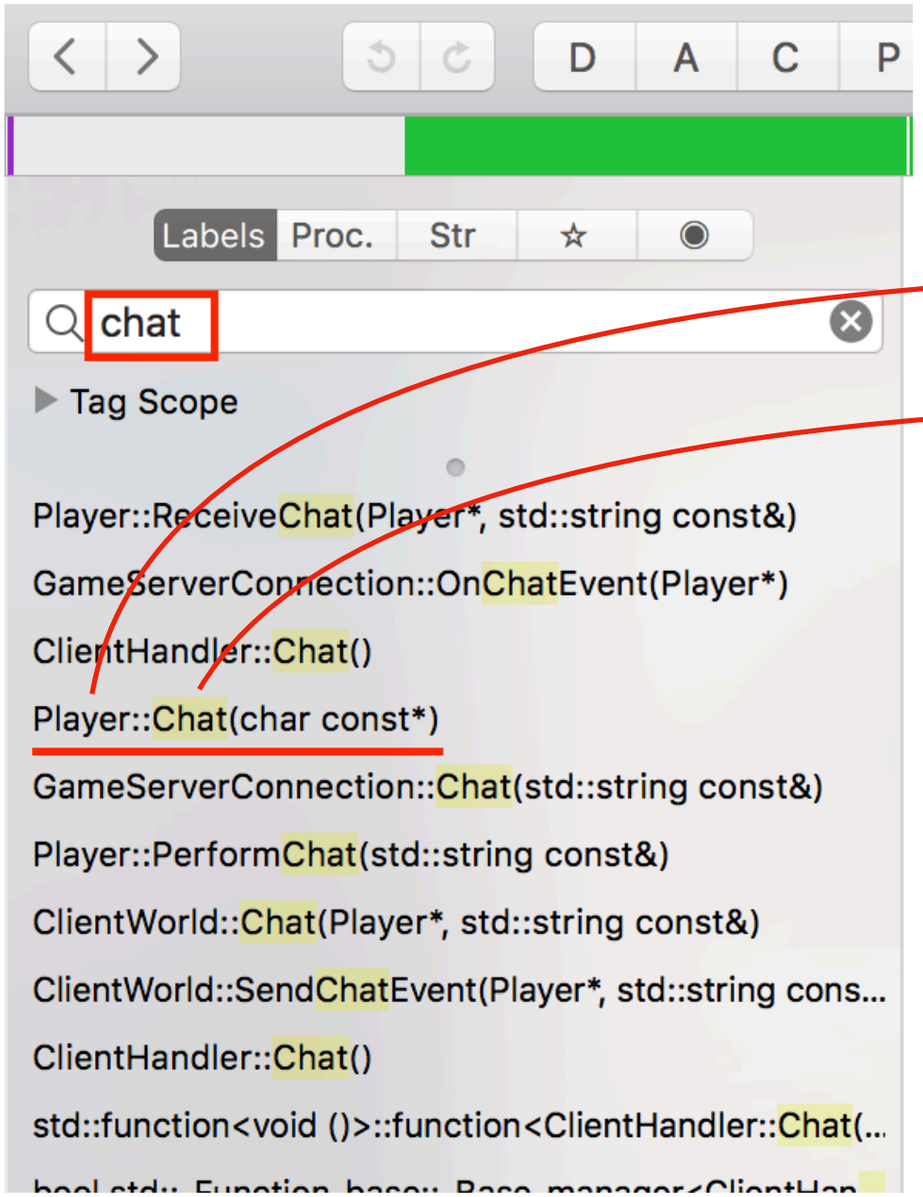
LIBRARY HOOKING

How to **trigger** the *hack*?

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

LIBRARY HOOKING

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 hook.so hook.cc
```

Run and **load** the **shared object**:

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

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

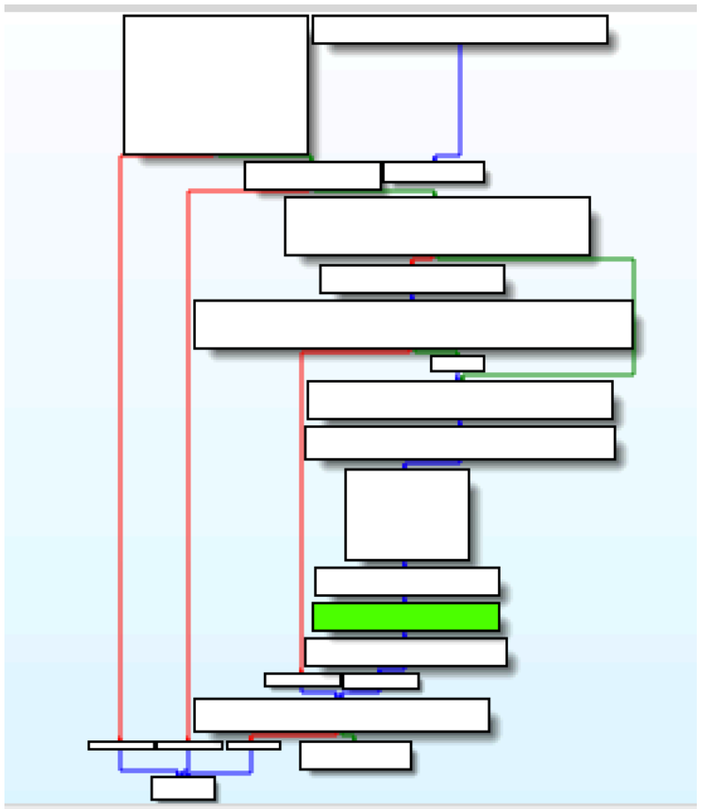
LIBRARY HOOKING

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:



```
mov     rcx, [rbp+var_70]
mov     [rbp+var_A0], rsi
mov     rsi, rcx
mov     rcx, [rbp+var_98]
mov     r8, [rbp+var_A0]
call    r8
jmp     $+5
```

```
loc_158432:             ; pos
        lea     rsi, [rbp+choice]
        mov     rdi, [rbp+var_70] ; this
        call    __ZN5Actor11SetPositionERK7Vector3 ; Actor::SetPosition(Vector3 const&)
        jmp     $+5
```

```
loc_158444:             ; rot
        lea     rsi, [rbp+choice.rotation]
        mov     rdi, [rbp+var_70] ; this
        call    __ZN5Actor11SetRotationERK8Rotation ; Actor::SetRotation(Rotation const&)
        jmp     $+5
```

Vector3 type:

```
struct Vector3 {
    float x;
    float y;
    float z;
};
```

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(this, &p);
    }

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

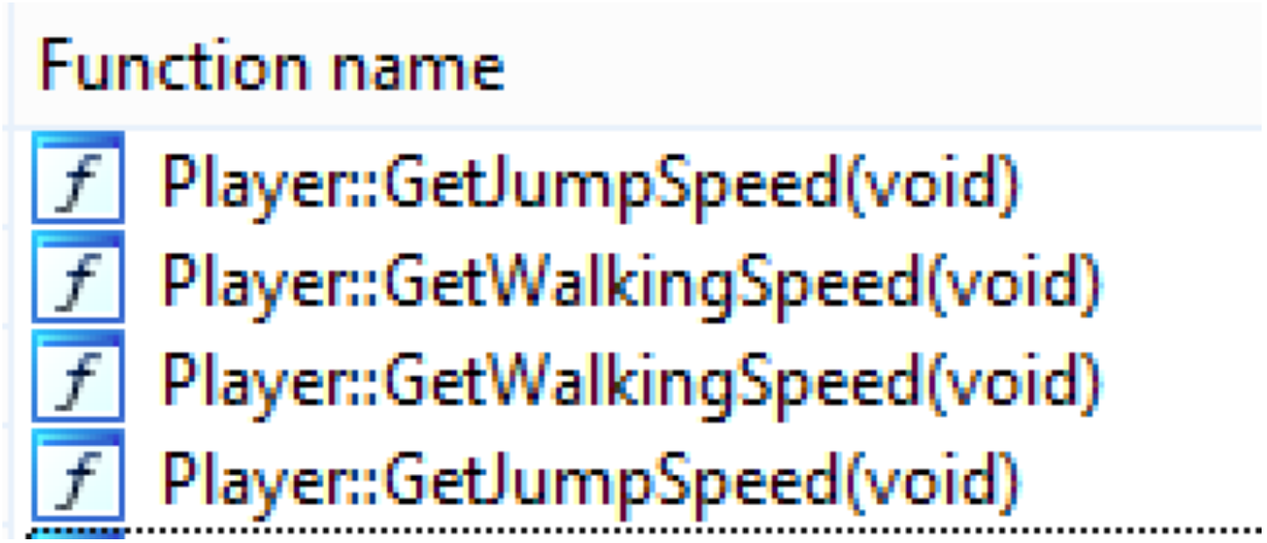
LIBRARY HOOKING

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**:



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
```

17-19 October 2017
MMXVII
HACK.LU

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 orignal 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/Foxmole/PwnAdventure3/blob/master/hook.cc

FUTURE WORK

Homework:

- Reverse 3333 (SSL/TLS)
- Wall hack

Want **more**?

- Vector35 trainings
- https://www.reddit.com/r/REGames/

Questions?

17-19 October 2017
MMXVII
HACK.LU