Exploiting Game Engines For Fun & Profit

Luigi Auriemma & Donato Ferrante

Paris, May 2013
Re-VWho?

- Vulnerability Research
- Consulting
- Penetration Testing

REVULN.com
Agenda

• Introduction

• Game Engines

• Attacking Game Engines
  – Fragmented Packets
  – Compression
  – Game Protocols
  – MODs
  – Master Servers

• Real World

• Conclusion

Theory about how to find vulnerabilities in game engines

Real world examples
Introduction

- Thousands of potential attack vectors (games)
- Millions of potential targets (players)

Very attractive for attackers
But wait...

Gamers
But wait... did you know...

• **Unreal Engine** => Licensed to **FBI** and **US Air Force**
  – In March 2012, the FBI licensed Epic's Unreal Development Kit to use in a simulator for training.
But wait... did you know...

- **Real Virtuality** $\Rightarrow$ It’s used in military training simulators
  - VBS1
  - VBS2
But wait... did you know...

- **Virtual3D** => Mining, Excavation, Industrial, Engineering and other GIS & CAD-based Visualizations with Real-time GPS-based Animation and Physical Simulation on a Virtual Earth => SCADA
But wait... did you know...

Different people but they have something in common...

**They are potential attack vectors**

- When they go back home, *they play games*
- When they play games, *they become targets*
- And most importantly, *their Companies become targets*
Game Engines
A Game Engine is the Kernel for a Game.
Game Engines [ and LEGO ]

- A Game Engine is basically a **Pre-Built Piece** where developers can plug new pieces on...
Game Engines [ Examples ]

• Several games share the same game engine

• The most popular game engines on the market are:
  – **Source**: Team Fortress 2, DOTA 2, Half Life 2, etc.
  – **CryEngine**: Crysis series
  – **UnrealEngine**: Unreal Tournament series
  – **idTech**: Quake series, DOOM 3, etc.

• But.. We are **NOT** developers. We are **bug-hunters**, we care about the **consequences** of using game engines :]

ReVuIn Ltd.
**Game Engines [ BugMath ]**

- **Some Math**
  - 1 Game $\Rightarrow$ 1 Game Engine
  - 1 Game Engine $\Rightarrow$ N Games

- **In Other Words**
  - 1 vulnerability in a Game $\Rightarrow$ 1 Game affected
  - 1 vulnerability in a Game Eng. $\Rightarrow$ N Games affected
Is this game **using** a Game Engine?

- Be careful before making assumptions
  - i.e. “This game has **NO** game engine!”

- Every Game has a Game Engine
  - Even PONG..
  - Game Engine functionality must be there

- It’s just a matter of how many other games share the same engine
Attacking Games

Without Considering Game Engines

• Just see unrelated/isolated components

• Missing a potential big attack vector..
  – Reducing the impact of potential issues

• If we don’t take in account Game Engines...

—FAIL
Attacking Games via Game Engines

- Even **the smallest issue in a game engine** can be a very valuable issue
- We **can affect several different targets at once**
- If we take in account engines...

---

**WIN**

---

Checkmate
Attacking Game Engines
The Attack Plan

Fragmented Packet
Compression
Custom
Game Protocols
The Attack Plan

- Fragmented Packet
- Compression
- Custom
- Game Protocols
Fragmented Packets

- **Network** support level
- Used in the **TCP-Over-UDP** implementation
- A fragmented packet is a UDP packet:
  1. **POS**: position of the current packet in the given stream
  2. **SIZE**: current data size
  3. **DATA**: current data
  4. **OTHER**: implementation dependent stuff
- Requires 2 engine actions: **Splitting** and **Rebuilding**
Splitting
Splitting

Original Packet
Hello Game!

Engine (Splitting)

Fragmented Packets

PKT>1:6:Hello
PKT>2:4:Game
PKT>3:1:!

POS
SIZE
DATA
Rebuilding
Rebuilding [ SUPPOSED ]

- Read Frag. Header
  - POS
  - LEN
- Place DATA in PKT_STREAM[POS]
Rebuilding [ ACTUAL ]

PKT_STREAM[-1] = AAAAA....AAAAA

PKT> -1:65:A..A

Game Engine

Hello

Game Engine Allocated Buffer

Memory

ReVuln Ltd.
while( true )
{
    [ do stuff ]
    pkt = get_packet( )
    buff = allocate( pkt.size )
    buff[ pkt.pos ] = pkt.data
    [ do more stuff ]
}
Fragmented Packets [ WaitWhat ]

• Corner-cases are the best :]

• What about truncated fragmented packets?

• ENGINE SPECIFIC

• RARE
  bad packet => drop packet

• USUAL
  mixing data coming from different packets, and so on...
  Hello memory corruption :]
Fragmented Packets [ Examples ]

• Several **Games, Game Engines** and **libraries** affected:
  – Source Engine
    • Counterstrike Source
    • Team Fortress 2
    • More..
  – CryEngine
  – American’s Army 3
  – ENet Library
  – **Others**...
Fragmented Packets [ Exploitation ]

- Easy to exploit fragmented packet issues
- Game engines are usually written in C++
- Tons of function pointers around
- Need more :]

ReVuln Ltd.
The Attack Plan

Fragmented Packet

Compression

Custom.

Game Protocols
Compression

Algorithms
Compression

Index Numbers

“Structures”

Algorithms

Other
Compression [ Index Numbers ]

• A way to **represent** numbers

• **Store/Transmit numbers** in an efficient way
  – Using the minimum amount of bits

• **Number** = [sequence of bits] != 4-bytes
  – **Average** case: 1,2 bytes
  – **Worst** case: 5 bytes

• **Two types:**
  – **Unsigned**
  – **Signed**
Compression [ Index Numbers ]

• General way for 32 bit (unsigned):
  – 7 bits, value
  – 1 bit, has next (byte) check

• To get an idea:
  – Number fits in 7 bits =>
  – Or it needs more bits =>
Compression [ Index Numbers ]

- General way for 32 bit (signed):
  - 1st byte:
    - 1 bit, sign
    - 6 bits, value
    - 1 bit, has next (byte) check
  - From the 2nd onwards:
    - same as the unsigned version
Compression [ Index Numbers ]

• Looking for interesting bugs?
  – Think about **flipping the first/last bit**

• Very often **integer overflows**

• Easy to exploit
The Attack Plan

Fragmented Packet

Compression

Customization

Game Protocols

ReVuln Ltd.
Game Protocol [ Opcodes ]

• Suppose that we have found a vulnerability in a game engine shared between 2 different games.

• We have found the game protocol, and the opcodes for the first protocol handshake are:

  Client → 00 11 22 33 44

  bb 11 22 33 44 ← Server

ReVuln Ltd.
Game Protocol [ Opcodes ]

GAME 1

00 11 22 33 44

bb 11 22 33 44

GAME 2

df 11 22 33 44

la 11 22 33 44

OBFUSCATION

ReVuln Ltd.
Game Protocol [ Opcodes ]

• Why should we care?
  – If we want to be able to write cross-game exploits we need to understand these concepts

• Interesting approach:
  – Protocol Tables
Game Protocol

**Base opcodes**
provided by the engine

- OP_1
- OP_2
- OP_3
- OP_4
- OP_5
- OP_6

**Permutation-based**
approach
provided by the engine but
used by the game developers

- OP_4
- OP_5
- OP_6
- OP_1
- OP_2
- OP_3

**Function-based**
approach
provided by the engine but
used by the game developers

- OP_1(x)
- OP_2(x)
- OP_3(x)
- OP_4(x)
- OP_5(x)
- OP_6(x)
Game Protocol [ Runtime Generation ]

- **The protocol table appears in memory (only) at Runtime**

- **Good news**
  it’s constant for each game

- **Bad news**
  we need to get the table for each Game using the target Game Engine

---

The protocol table appears in memory (only) at runtime. Good news: it’s constant for each game. Bad news: we need to get the table for each Game using the target Game Engine.
Game Protocol [ Exploitation ]

X_Game_Exploit ~ Engine_Exploit_Template(G)

```
\text{Fix}[0] \quad \text{var}[G, x] \quad \text{Fix}[2...8] \quad \text{var}[G, y] \quad \text{Fix}[10...16]
```

```
\text{Tables}[G] = \begin{align*}
\x73 & \x55 & \x89 & \x12 \\
\x11 & \x60 & \x40 & \x65 \\
\x23 & \x19 & \xdf & \x08
\end{align*}
```
The Attack Plan

- Fragmented Packet
- Compression
- Game Protocols
- Custom.
Customization [ MODs ]

• Game engines allow users to load custom MODs:
  – Animations
  – Maps
  – Model
  – Sounds
  – Etc

• Maps are interesting because:
  – Complex binary formats (fuzzing..)
  – Complex parsing routines (IDA..)
  – Automatically downloaded from the Servers
  – A bug mine :]

ReVuln Ltd.
Customization [ CMD line ]

- Game engines allow users to start games with *custom command line arguments*

- Usually local issues/features => local exploit => sad..

- **But!** Thanks to **Origin** and **Steam** an attacker can exploit these local issues/features remotely.
  - Hello **RCE :]**
  - Please refer to our previous research on Origin and Steam security for additional info.
Customization [ CMD line ]

- Command line switch to check for interesting effects:

1) **Devmode**: to enable most of the fun things :
   - Supposed to be used to debug/test/mess with customizations

2) **Loading**: to load arbitrary files in memory or on arbitrary locations on the victim’s system
   - Supposed to be used to load external (local) content like:
     - maps
     - sounds
     - models
     - Etc.

3) **Logging**: to write custom files on the victim’s system
   - Supposed to be used to log game customization or in-game events
Customization [ CMD line ]

**Expected** Usage (local exec):

```
gameX.exe -map myNewAmazingMap
```

**Unexpected** Usage (remote exec):

```html
<a href="steam://start=GameX&Map=veryMALICIOUSwebsite.com/map">

URI  RUN GAME  PARAMS={ local cmd-line args }
```

Please refer to our paper on Steam for a real/complete steam:// link example.
Master Servers
Master Servers [ What ]

• Master Servers are online database for games
  – Info about Servers => Hosted by Companies & Players
  – Sometimes info about Clients => Players

• Useful for developers
  – Matchmaking

• Useful for players
  – Finding match to join

• Useful for attackers
  – Finding victims/targets
Master Servers [ How ]

List_servers( GAME_ENGINE_X )

ip₁, ip₂, ip₃, …, ipₕ

Master Server

Game\textsubscript{A} Servers

Exploit!

Game\textsubscript{B} Servers
Real World
idTech 4 (0-days)

Quake Wars

Brink

QUAKE 4
idTech 4 [ The Function ]

- idTech 4, exposes an interesting function
  - idBitMsg::ReadData(..)

- This function is used both:
  - Server-side
  - Client-side

- Attackers have twice the fun
idTech 4 [ The Function ]

• This function is available in all the games using this engine

• But some games don’t call the function in a vulnerable way, like **DOOM 3**

• For other games there are several places where there is a call to this function...
idTech 4 [ The Function ]

• In Quake Wars
  – the function is called in a bad way **Client-side**

• In Brink
  – the function is called in a bad way **Server-side**

• Let’s take a look at some **0-days** related to this function...
int idBitMsg::ReadData( void *data, int length ) const {
    int cnt;
    ReadByteAlign();
    cnt = readCount;

    if ( readCount + length > curSize ) {
        if ( data ) {
            memcpy( data, readData + readCount, GetRemaingData() );
        }
        readCount = curSize;
    } else {
        if ( data ) {
            memcpy( data, readData + readCount, length );
        }
        readCount += length;
    }

    return ( readCount - cnt );
}
int idBitMsg::ReadData( void *data, int length ) const {
    int cnt;
    ReadByteAlign();
    cnt = readCount;

    if ( readCount + length > curSize ) {
        if ( data ) {
            memcpy( data, readData + readCount, GetRemaingData() );
        }
        readCount = curSize;
    } else {
        if ( data ) {
            memcpy( data, readData + readCount, length );
        }
        readCount += length;
    }

    return ( readCount - cnt );
}
int idBitMsg::ReadData( void *data, int length ) const {
  int cnt;
  ReadByteAlign();
  cnt = readCount;

  if ( readCount + length > curSize ) {
    if ( data ) {
      memcpy( data, readData + readCount, GetRemainingData() );
    }
    readCount = curSize;
  } else {
    if ( data ) {
      memcpy( data, readData + readCount, length );
    }
    readCount += length;
  }

  return ( readCount - cnt );
}
idTech 4 [ BUG #2 ]

### curSize

- 0x28

### readCount

- 0x29

```
0070BE35  MOV EDX, DWORD PTR DS:[ESI+4]
0070BE38  SUB EAX, EDI  ; 0x28 - 0x29
0070BE3A  PUSH EAX     ; /n
0070BE3B  ADD EDX, EDI ; |
0070BE3D  PUSH EDX     ; |src
0070BE3E  PUSH ECX     ; |dest
0070BE3F  CALL <JMP.&MSVCR90.memcpy>
```

MEMCPY( dest, src, n )
idTech 4 [ BUG #2 - EXPLOIT ]

Packet 2 – TRUNCATED
BUFF = REABCDE

Packet 1 – COMPLETE
BUFF = XXEVULN

GAME ENGINE

Memory

Packet 1 → GAME ENGINE
Packet 2 → GAME ENGINE
Packet 3 → GAME ENGINE
Packet 4 → GAME ENGINE
The caller does NOT verify the length parameter (like in Brink)

```cpp
int idBitMsg::ReadData( void *data, int length ) const {
    int cnt;
    ReadByteAlign();
    cnt = readCount;

    if ( readCount + length > curSize ) {
        if ( data ) {
            memcpy( data, readData + readCount, GetRemaingData() );
        }
        readCount = curSize;
    } else {
        if ( data ) {
            memcpy( data, readData + readCount, length );
        }
        readCount += length;
    }

    return ( readCount - cnt );
}
```

**BUG #3**
idTech 4 [ BUG #3 ]

1400 bytes -> SRC

1000 bytes -> DEST

0070BE38  SUB EAX, EDI
0070BE3A  PUSH EAX ; /n
0070BE3B  ADD EDX, EDI ; |
0070BE3D  PUSH EDX ; |src
0070BE3E  PUSH ECX ; |dest
0070BE3F  CALL <JMP.&MSVCR90.memcpy>

SRC up to 1400 bytes & DEST max 1000 bytes
The **GetInfo** packet is handled in an interesting way.

The engine checks if the packet has been sent from the Master Server:
- `q4master.idsoftware.com`

But an attacker can **spoof the IP** of the Master Server.

And..
idTech 4 [ Quake 4 ] (0-day)

10051B30  `.  55       PUSH EBP
10051B31  `.  8BEC     MOV EBP,ESP
10051B33  `.  83E4 F8  AND ESP,FFFFFFFF8
10051B36  `.  6A FF    PUSH -1
10051B38  `.  68 072E2810 PUSH 10282E07
10051B3D  `.  64:A1 00000000 MOV EAX,DWORD PTR FS:[0]
10051B43  `.  50     PUSH EAX
10051B44  `.  64:8925 00000000 MOV DWORD PTR FS:[0],ESP
10051B4B  `.  81EC 28050000 SUB ESP,528
10051BB7  `.  8BCE     MOV ECX,ESI
10051BB9  `.  E8 30381D00 CALL ReadBits
10051BBB  `.  E8 E1371D00 CALL ReadBits
10051C06  `.  6A F0     PUSH -10
10051C08  `.  8BCE     MOV ECX,ESI
10051C0A  `.  E8 E1371D00 CALL ReadBits
10051C31  `.  6A F0     PUSH -10
10051C33  `.  8BCE     MOV ECX,ESI
10051C35  `.  E8 B6371D00 CALL ReadBits
10051C50  `.  8B4D 08  MOV ECX,DWORD PTR SS:[EBP+8]
10051C53  `.  6A 20     PUSH 20
10051C55  `.  E8 96371D00 CALL ReadBits
10051C5A  `.  8B0D 04842F10 MOV ECX,DWORD PTR DS:[102F8404]
10051C60  `.  50     PUSH EAX
10051C61  `.  8907     MOV DWORD PTR DS:[EDI],EAX

; Installs SE handler 10282E07
; /Arg1 = -10
; \Quake4Ded.ReadBits (loop 1)
; /Arg1 = -10
; \Quake4Ded.ReadBits (loop 2)
; /Arg1 = -10
; \Quake4Ded.ReadBits (loop 3)
; /Arg1 = 20
; \Quake4Ded.ReadBits (our value)
; stack based buffer-overflow
Customized Engines [ Unreal Engine 3 ]
Customized Engines [ Unreal Engine 3 ]

- Some games use customized versions of this engine.
- But they don’t always change for the better...
- Especially from the Security point-of-view.
- The following slides give examples of issues introduced by customizations for the Unreal Engine 3.
• Devs added **RCON** support:
  
  + 1 new port
  
  custom protocol
  
  =
  
  ----------------------
  
  several new issues
  
  (wchar_t *)buff[size] = 0

• Some **RCON** affected commands:
  
  – **CT** <negative number> => 16-bit off the buffer set to 0
  
  – **CT** <negative number> => stack-based overflow
Monday Night Combat (0-day)

- Array overflow => Heap Corruption => RCE
  - Caused by additional Steam-related commands
    - STEAM_AUTHBLOB SUBBLOB=123 NUMSUBBLOBS=1 AUTHBLOBSTRING=aa...aa

```
00A8B9EA  . 03C0    ADD EAX,EAX
; array[SUBBLOB][12]
00A8B9EC  . 8B4C02 04    MOV ECX,DWORD PTR DS:[EDX+EAX+4]
00A8B9F0  . 3BCD    CMP ECX,EBP
; ECX must be 0 or 1
00A8B9F2  . 74 09    JE SHORT MNCDS.00A8B9FD
[...]
00A8B9FD > 8D4C24 28    LEA ECX,DWORD PTR SS:[ESP+28]
00A8BA01  . 51    PUSH ECX
00A8BA02  . 8D0C02    LEA ECX,DWORD PTR DS:[EDX+EAX]
; heap corruption with AUTHBLOBSTRING
00A8BA05  . E8 C6C59EFF    CALL MNCDS.00477FD0
00A8BA0A  . 8D4C24 28    LEA ECX,DWORD PTR SS:[ESP+28]
00A8BA0E  . C78424 C0[..]FF    MOV DWORD PTR SS:[ESP+8C0],-1
00A8BA19  . E8 52C3C3FF    CALL MNCDS.006C7D70
00A8BA1E  . E9 3E0E0000    JMP MNCDS.00A8C861
```
CryEngine 3 (0-days)
CryEngine 3 [ Fragmented Packet ]

- **_OPCODE**
- **ID**
- **FIELD 1**
- **FIELD 2**
- **DATA**

- Variable size
- 8 bit
- 16 bit
- 4 bit
CryEngine 3 [ Bug #1 ]

Integer Overflow

Memory Corruption
CryEngine 3 [ Bug #1 ]

Integer Overflow

Memory Corruption
Integer Overflow
Via Fragmented Packets [0-day]

395818D7  MOV EDX, DWORD PTR DS:[ESI]    ; packet size (=2) < 4
[...]  
395818E3  SUB EDX, 4                      ; 2 - 4
395818E6  PUSH EDX
395818E7  ADD EAX, 4
395818EA  PUSH EAX
395818EB  LEA ECX, [EDI+ECX+23]
395818EF  PUSH ECX
395818F0  CALL <JMP.&MSVCR100.memcpy>

CRYSSIS_OPCODE [0x93]
ID [truncated]

Just a 2-byte packet
CryEngine 3 [ Bug #2 ]

Integer Overflow

Memory Corruption
Heap Overflow
Via Fragmented Packets

CRYSIS_OPCODE (0x93)
ID
COUNTER
DATA

EXPLOIT WORKFLOW

SEND
RECV

SLEEP( 100 )
Heap Overflow
Via Fragmented Packets (0-day)

39581C0F  MOV EAX, DWORD PTR DS:[ESI]
39581C11  MOV EDX, DWORD PTR SS:[ESP+1C]
39581C15  MOV DWORD PTR DS:[EDX], EAX
39581C17  LEA ECX, [ESI+4]
39581C1A  AND EAX, FFFFFFFFC
39581C1D  MOV EDX, DWORD PTR DS:[EAX+0C]
39581C20  PUSH ECX
39581C21  PUSH EDI
39581C22  CALL EDX
Conclusion

• Game engines are **crucial** for games

• Game engine issues affect **sets of games**

• Games are **no longer for kids**

• Master servers can be used to conduct **distributed/targeted attacks** against Companies or Players

• Game security is **scary for players**

• And **awesome for Security Researchers :]**
References

• A paper about engine bugs is available at:
  – http://revuln.com/files/ReVuln_Game_Engines_0days_tale.pdf

• Steam and Origin papers:
Thanks! Questions?

revuln.com

info@revuln.com

FOLLOW US: @REVULN