The Hardest CTF I've Ever Done

My Experiences Reverse Engineering an MMORPG

Dave Kukfa

/who

- Dave Kukfa
- Senior Computing Security student @ RIT
- Application Security
- Reverse Engineering
- https://kukfa.co
- @kukfa_

Agenda

- Project intro
  - Game background
  - Concept of a server emulator
  - Overview of MMO reversing process
  - Tools of the trade
- Technical analysis
  - Technical details of the reversing process
  - My experiences with my specific game
- Project demo
- Legal implications
- Questions

Audience Survey

Are there any...

- MMO players?
- CTF players?
- Game hackers?

Project Intro

Game Background

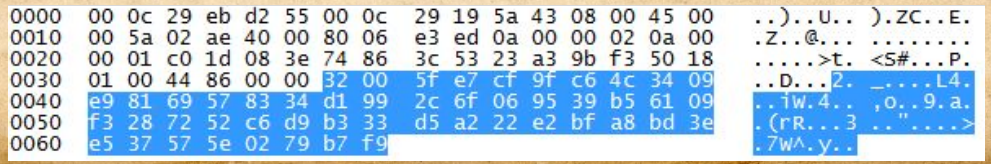
- Dungeon Runners (NCSoft, 2007)
- MMORPG
  - Online game where thousands of players interact in a virtual world
- Custom game engine
  - Scrapped several times until it became Dungeon Runners
- Decommissioned in 2010
  - Lack of profit
  - Servers shut down
- Game client has no server to connect to

Server Emulator

- An attempt to recreate an online game's server
- Lots of proprietary components
  - Communication protocol
  - Server architecture
  - Database schemas
  - Etc.
- Involves a great deal of reverse engineering
  - Taking something apart to see how it works
  - Very long, arduous process
- Implementation is generally hacky
  - Built in a 'guess & check' fashion

Typical Reversing Process

- Protocol analysis
  - Determine how the client and server communicate
  - Packet identification
    - Identify the different types of packets and their purpose
  - Packet structure
    - “What do these bytes represent or control?”
  - Encryption & checksum algorithms
  - Connection handoffs
    - “At what point does the authentication server direct the client to the game server?”
  - Serialization
    - “How are things like character data sent over the wire?”



Typical Reversing Process cont.

- Server design
  - Designing massive game servers is a very complex problem
  - Architecture
    - Many moving parts
      - Authentication server, game logic server, zone servers, database server…
    - Load balancing/caching
  - Database design
    - SQL vs NoSQL? Schema?
  - Code structure
  - Scalability
  - Backups/failover

Typical Reversing Process cont.

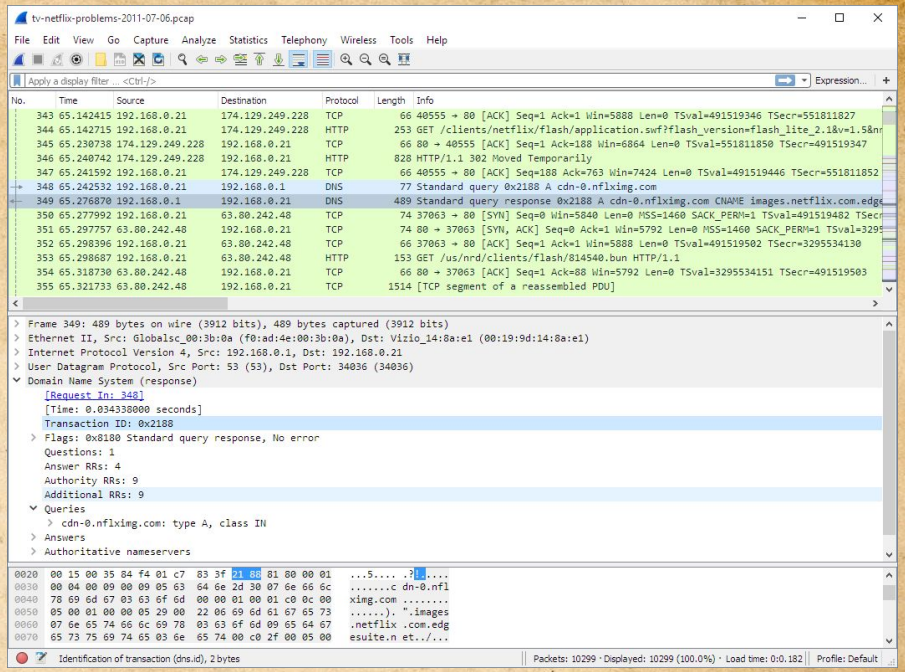
- Populate with data
  - Depends on how much data is stored client-side
    - Usually the bulk of it is (good!)
  - Vendor/merchant inventory
  - Quest info
  - Enemies in combat zones
  - Loot/drops
  - If you have it particularly bad:
    - Zone layouts
    - NPC locations
    - Dialogue

Tools of the Trade

- Disassembler
  - Transforms machine language into assembly language
  - Makes a compiled program (sort of) human-readable
  - Static analysis
  - IDA, Binary Ninja, Hopper, Radare2...
- Debugger
  - View the state of a running program
    - Current instruction, register values, memory
  - Dynamic analysis
  - IDA, OllyDbg, gdb, x64dbg...

Tools of the Trade cont.

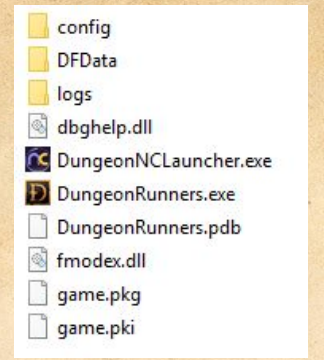
- Packet sniffer
  - Capture and inspect network traffic
  - Wireshark, tcpdump, Fiddler...
- Hex editor
  - Manipulate binary data (in hex form)
  - HxD, Vim, Hex Fiend...

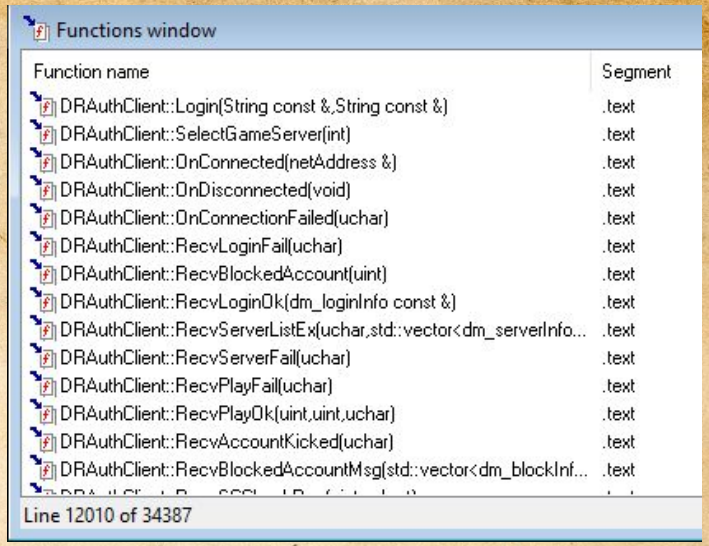


Technical Analysis

What I Had to Work With

- Dormant game client
- No game server protocol documentation
  - We'll talk about this in a bit
- No packet captures
- But!
  - Debug symbols
  - Verbose log output






```
12/09 09:50:39.875   832:    DRAuthClient::Login Connecting to Auth: '206.127.154.204:2110'
12/09 09:50:39.875   832:    DRAuthClient::Login changed state from 0 to 1.
12/09 09:50:40.171   2332:   DRAuthClient::OnConnected changed state from 1 to 2.
12/09 09:50:40.328   2332:   DRAuthClient::RecvLoginOk LoggedIn as simic024
12/09 09:50:40.328   2332:   DRAuthClient::RecvLoginOk changed state from 2 to 3.
12/09 09:50:40.500   2332:   DRAuthClient::RecvServerListEx changed state from 3 to 4.
```

Starting Out

- Game only ran if launched from the publisher's game portal
- Looked at command-line options
  - ran_from_launcher
- Looking through config folder
  - DungeonRunners.cfg
    - Specifies authentication server addr/port
    - Direct the client to connect to local server

DungeonRunners.cfg

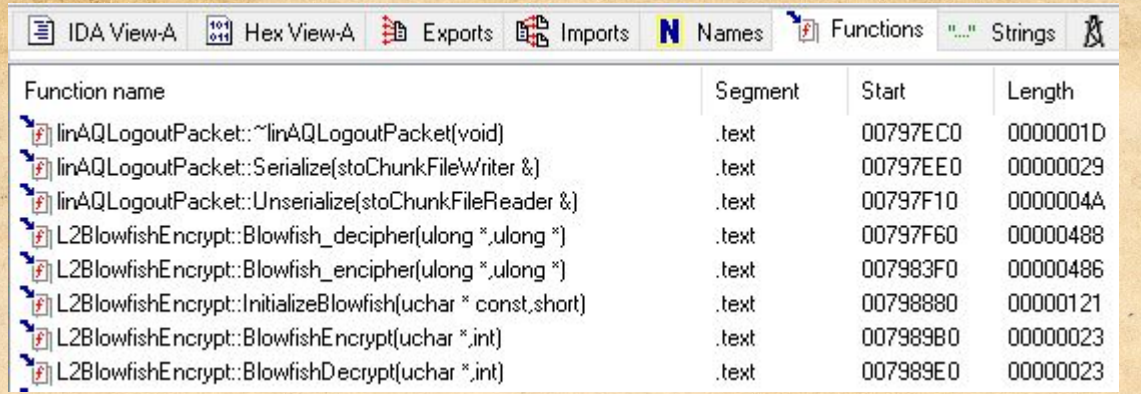
```
1  [AuthServer]
2  Address = auth.dungeonrunners.com
3  Port = 2110
4
5  [ResourceManager]
6  ResourceConfig = ResourceManager.cfg
```

IDA View-A | Hex View-A | Exports | Imports | Names | Functions | Strings | Structures | Enums

Address	Length	Type	String
"..." .rdata:0...	00000050	C	fmodSampleManager::Init Unknown Sound InputFileType '%s' in the config ...
"..." .rdata:0...	00000028	C	http://www.dungeonrunners.com/join.html
"..." .rdata:0...	00000018	C	.\\DungeonNCLauncher.exe
"..." .rdata:0...	00000056	C	%s /LauncherArgs=\" /LaunchGame=%s /StartLocation=\\\"{_stub_dir}\\\" \" ran_from_launcher
"..." .rdata:0...	0000002D	C	Failed to launch %s in %s with last error %u
"..." .rdata:0...	00000011	C	.\\LauncherID.ini

Auth Server Protocol

- So the client can connect to our custom auth server…
  - Now we need to understand the protocol
- Looking through functions list



Function name	Segment	Start	Length
linAQLogoutPacket::~linAQLogoutPacket(void)	.text	00797EC0	0000001D
linAQLogoutPacket::Serialize(stoChunkFileWriter &)	.text	00797EE0	00000029
linAQLogoutPacket::Unserialize(stoChunkFileReader &)	.text	00797F10	0000004A
L2BlowfishEncrypt::Blowfish_decipher(ulong *,ulong *)	.text	00797F60	00000488
L2BlowfishEncrypt::Blowfish_encipher(ulong *,ulong *)	.text	007983F0	00000486
L2BlowfishEncrypt::InitializeBlowfish(uchar * const,short)	.text	00798880	00000121
L2BlowfishEncrypt::BlowfishEncrypt(uchar *,int)	.text	007989B0	00000023
L2BlowfishEncrypt::BlowfishDecrypt(uchar *,int)	.text	007989E0	00000023

- ‘lin’ and ‘L2’ prefixes
- Right next to authentication functions

Lin and L2

- Some quick Googling reveals these refer to Lineage II
  - Another NCSoft MMO (2003 - present)
  - Several releases over the years
  - Game has been reverse engineered and publicly documented
- Could DR have borrowed L2's auth server?
  - Each major L2 release had a different protocol
  - Find an early version of the server and see if it works
- Hunted for an L2 server from around the same time period
  - Easier said than done
  - Old forums/dead links
  - archive.org is a life saver
- Found legacy auth server from 2004

Used to seeing...

After L2


Dungeon
RUNNERS
Choose Any World to Play On!
Your characters are available on all of them.
No worlds are online right now.
Okay
Cancel
User Name
dave
Password
****
Login
Selecting World...
Credits
Exit
Build: 666 (Final)
Copyright NCsoft. All Rights Reserved.

Re-Implementation

- DR did use L2's authentication protocols
  - This turns out to be a common practice for NCsoft MMOs
- While DR and the L2 server shared the same protocols, the L2 server was (obviously) built for L2
  - Not 100% compatible out of the box
  - Some things didn't work
- Instead of changing L2 code, let's build our own auth server!
- Good news: L2's protocols have been reversed and publicly documented
- Bad news...

It's never easy

Протокол Lineage II

Автор: *TechnoWiz@rd*
Последнее редактирование: 23 ноября 2007

Содержание

1. Общие сведения
2. Пакеты Client -> Login Server
3. Пакеты Login Server -> Client
4. Пакеты Game Server -> Client
5. Пакеты Client -> Game Server

1. Общие сведения

Каждый пакет состоит из размера пакета(2 байта), типа пакета(1 байт) и блока параметров(переменная длина). авторизации, в конце добавляется контрольная сумма и дополняется нулями так, чтобы размер пакета был кратен 8- быть рассчитана следующей функцией:

```
unsigned long checksum( unsigned char *packet, int count )
{
   long chksum = 0L;
   for( int i = 0; i < count; i += 4 ) chksum ^= *((unsigned long *)&raw[i]);
   return chksum;
};
```

Rosetta Stone

- After much help from Google Translate, I had a basic working prototype
  - Client logs in
  - Server sends a list of game servers (worlds)
  - Client selects a world and connects to it
- Issue: Client authentication packet did not follow protocol spec
  - Having trouble extracting player credentials
  - Username & password were unintelligible blobs of data
  - Needed to figure out how credentials were being encrypted

Finding Encryption Routine

```
; public: int __thiscall ncAuthClientImpl::SendLogin(char const *, char const *, unsigned int, unsigned short)
?SendLogin@ncAuthClientImpl@@QAEHPBD0IG@Z proc near
```

```
call    ?DesWriteBlock@L2DESEncrypt@@YAXPAXH@Z ; L2DESEncrypt::DesWriteBlock(void *,int)
```

```
; void __cdecl L2DESEncrypt::DesKeyInit(char const *)
?DesKeyInit@L2DESEncrypt@@YAXPBD@Z proc near
```

```
loc_79765B:
mov     dl, ds:byte_8534B4[eax]
xor     byte ptr [esp+ecx+0Ch+var_C], dl
lea     ecx, [esp+ecx+0Ch+var_C]
inc     eax
cmp     ds:byte_8534B4[eax], 0
jnz     short loc_797647
```

And the key is...

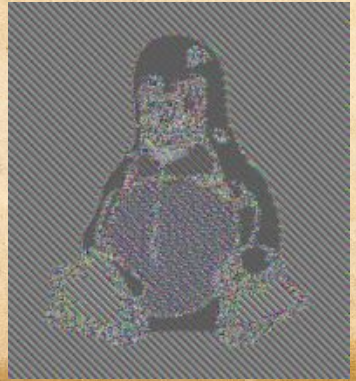
And the key is...

```
.rdata:008534B4 aTest                   db 'TEST',0
.rdata:008534B4
.rdata:008534B9                         db    0
.rdata:008534BA                         db    0
.rdata:008534BB                         db    0
```

...TEST with 4 null bytes

Credential Encryption

- Algorithm: DES
- Symmetric Key: TEST + 4 null bytes
- Mode of Operation: Take a guess, given our track record
  - Yeah, ECB
- Block size: 64 bits (8 bytes)
  - This presents an interesting problem



L2 Login specifications

- Max username length: 14 characters
- Max password length: 16 characters
- Total size: 30 characters → 30 bytes
  - ASCII encoded
- DES block size: 8 bytes
  - Can encrypt data with sizes 8 bytes, 16 bytes, 24 bytes, 32 bytes…
- What happens when the credential size isn't perfectly divisible by 8?
  - You might think it's padded to round up?

Nope!

- Username: 14characterusr
- Password: 16characterpassw
- Network traffic:

Username

>	.	©	)	ž	Ú	q	.	.	þ	Ç	ã	‰	ù
1	2	3	4	5	6	7	8	9	10	11	12	13	14

Password

.	¶	.	Ž	Š	.	.	.	ÿ	³	r	p	a	s	s	w
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Onwards

- At this point, the authentication server was pretty much figured out
- When the player selects a game server:
  - Game client will disconnect from the auth server
  - Attempts to connect to the game server
- Game server uses a custom (undocumented) protocol
  - Where do I start?

Starting Out

- Tried to mimic the protocol of the auth server
  - Didn't get too far (two different protocols)
- Good: Verbose game logs pointed out what was wrong
  - Can use this to fix problems and work out correct packet structure

```
04/19 10:30:59.138  3236:   DRAuthClient::SelectGameServer changed state from 4 to 5.
04/19 10:30:59.164  3236:   DRAuthClient::RecvPlayOk changed state from 5 to 8.
04/19 10:30:59.175  3236:   DFCSocketChannel::updateReceiveQueue Closing socket 10.0.0.1:7777 because
packet size 3277056 exceeds max 1048576
04/19 10:30:59.192  3236:   GatewayClient::UpdateAuthorize Lost connection during Authorization.
```

- 3277056 = 0x320100
  - Find where in the packet I'm sending these bytes
  - Modify it and see what happens afterwards

It's not always easy

- The game doesn't log every issue it has when things don't work
- Often need to dive in with a debugger and fix things
  - Identify functions related to the issue you're having
  - Get an idea of what each function is doing
  - "Are there any checks that are failing?"
  - "What is the expected value of this (register|return value|...)?"
  - Set a bunch of breakpoints
  - Compare actual values to expected values
- Can be very time-consuming and frustrating
  - That's why these projects take so long to complete
  - If the high-level code was spaghetti, you'll become real familiar with these steps

Packet Structures

- Eventually identified 4 different packet structures
  - Small tweaks between each variant
- Example:
  1. [1 byte] Packet type
  2. [3 bytes] Unknown data
  3. [4 bytes] Compressed size of packet data + 4 (little-endian)
  4. [4 bytes] Uncompressed size of packet data (little-endian)
  5. [Variable] Zlib-compressed packet data
- Different ‘packet types’ hold the same data, but are assigned to a specific packet structure
  - e.g. Packet type 0x02 can only be used with packet structure #4

Channel Types

- Packet data starts out with a 1-byte ‘channel type’
  - Essentially identifies what component of the game server we’re talking to
- Started filling in different values for the channel type and seeing what happens
- Identified a bunch of different channels:
  - CharacterManagerClient
    - Character selection
  - ClientEntityManager
    - Game entities
  - ZoneClient
    - Entering and leaving different zones (essentially maps)

Channel Data

- After the channel type, we can start feeding data to be sent to that channel
- Usually starts with an identifier for the packet we'll be sending
- e.g. for CharacterManagerClient:
  - 0x00: Connected
  - 0x01: Disconnected
  - 0x02: CharacterCreated
  - 0x03: GotCharacter
  - 0x04: Enter Character Creation Screen
- This is followed by additional data for that packet
  - e.g. a list of the player's characters, a serialized character...

Current State

- After piecing enough of these together, I was able to get some basic game functionality
  - Character serialization
  - Creating a character
  - Sending character list
- There's still a lot to be done
  - Loading into a zone
  - Creating game entities
    - Other players
    - Monsters
  - Synchronizing multiple clients
  - Properly architect the server

Project Demo

Legal Implications

Game Company Stances

- They (usually) don't like it
  - Large game companies will fight tooth and nail over intellectual property
  - Indie studios might be more relaxed, but it's a toss-up
- Some famous examples:
  - (2016) Nostalrius shut down
    - WoW private server with 150k+ active users
  - (2012) UMaple lawsuit
    - Ordered to pay $3.6M to Nexon
  - (2009) Scapegaming lawsuit
    - Ordered to pay $88M to Blizzard
  - (2008) Glider Bot lawsuit
    - Ordered to pay $6M to Blizzard

NCSoft History

- (2006) Lineage II FBI investigation & home raid
- (2007) City of Heroes Cease & Desist
- (2010) Aion C&D
- (2011) Tabula Rasa C&D
- (2012) Exteel C&D
- (2014) Lineage I C&D

EFF Exemptions

- EFF landed recent DMCA exemption for video game archiving
- Proposed to reduce legal uncertainty of reversing games for preservation purposes
- Catch: Not intended to apply to games with persistent worlds
  - Literally the definition of MMORPGs
- Interpretation has been debated
- General consensus is the exemption does not apply to MMO server emulators

Case for Dungeon Runners

- Attempts to purchase IP have gone nowhere
- Continue reversing the game
- Avoid piracy
  - Don’t distribute the game client
  - Don’t use NCSoft-owned server code
- Not meant to be commercial
  - Only intend to restore gameplay
- Hope for the best

Questions?

Thank you!

dkukfa@mail.rit.edu
https://kukfa.co
@kukfa_