Subject: wwnet
Posted by [EE]pickle-jucer on Fri, 19 Dec 2014 14:28:37 GMT
View Forum Message <> Reply to Message

this didn't seem to fit in any other sub-forum so I put it here.

Does anybody have a full copy of the "D:\renegade code 02-06-03\finalrenegade\wwnet" folder?

EDIT:or just wwpacket.cpp & wwpacket.h if you have them

Subject: Re: wwnet
Posted by jonwil on Sat, 20 Dec 2014 03:00:46 GMT
View Forum Message <> Reply to Message

What exactly do you want to know about the renegade netcode and what are you trying to do?

Subject: Re: wwnet
Posted by [EE]pickle-jucer on Sat, 20 Dec 2014 14:03:36 GMT
View Forum Message <> Reply to Message

First off let me clarify that i'm not trying to cheat/hack anything, I am trying to figure out the packets to make a game server, separate from the FDS. I know it wont be fully functional but I think it would be really cool if I could make an opensource game server for renegade that anyone could use, modify or contribute to.

the reason I'm looking for the netcode is that pure guesswork isn't getting me far, I've figured out the general layout of the first packet sent to the server from the client as layed out below, but as you can see there is a lot of "unknown"s and I don't even have a way to tell the packet apart from the rest.

join packet:
 4 byte crc32 of the rest of packet
 10 byte unknown
 1 byte nameLength
 [nameLength*sizeof(WCHAR)] Name
 11 byte unknown
 4 byte "!TT!" literal
 4 byte float TT scripts version number
 1 byte unknown
 1 byte exekeyLen
 [exekeyLen*byte] exeKey
 4 byte unknown(bandwidth?)

EDIT: upon looking at a snippet from cnetwork.cpp, i've realize the "exeKeyLen" & "exeKey" part are wrong

```
//
// Get player name
// This is not supposed to be empty, but if for whatever reason it it, we should
// just refuse, rather than crash.
//
WideStringClass player_name(0, true);
//packet.Get_Wide_Terminated_String(player_name.Get_Buffer(256), 256);
packet.Get_Wide_Terminated_String(player_name.Get_Buffer(256), 256, true);
if (player_name.Get_Length() == 0) {
        return REFUSAL_VERSION_MISMATCH;
}

// Get the clients password
WideStringClass password(0, true);
packet.Get_Wide_Terminated_String(password.Get_Buffer(256), 256, true);

// Get clients exe version
int client_exe_key = packet.Get(client_exe_key);
```

Subject: Re: wwnet
Posted by danpaul88 on Sat, 20 Dec 2014 17:15:25 GMT
View Forum Message <> Reply to Message

The FDS doesn't simply relay network packets you know, it has a complete implementation of the game engine and runs all the scripts etc that are present in a map. You'd have to reimplement basically the entire engine or embed in somehow (defeating the point of open source).

What is your goal with this exactly? There might be better approaches to take.

Subject: Re: wwnet
Posted by [EE]pickle-jucer on Sat, 20 Dec 2014 17:40:43 GMT
View Forum Message <> Reply to Message

danpaul88 wrote on Sat, 20 December 2014 10:15The FDS doesn't simply relay network packets you know, it has a complete implementation of the game engine and runs all the scripts etc that are present in a map.

I am well aware how the FDS works, I am currently just trying to understand the protocol but will end up working on this as well.

danpaul88 wrote on Sat, 20 December 2014 10:15You'd have to reimplement basically the entire engine or embed in somehow (defeating the point of open source).

Embedding is not the idea, I will end up rewriting all (or at least significant) portions of the FDS.

danpaul88 wrote on Sat, 20 December 2014 10:15
What is your goal with this exactly? There might be better approaches to take.

Reimplementing the FDS in a modern way.

edit: just realized this may come off as rude, that was not my intention

---

Subject: Re: wwnet
Posted by danpaul88 on Sat, 20 Dec 2014 20:08:55 GMT
View Forum Message <> Reply to Message

If you're thinking of making it multi threaded or asynchronous you'll find an awful lot of scripts etc will start to break in strange ways because they are designed for a single threaded server and make assumptions about various things not being volatile during an event

---

Subject: Re: wwnet
Posted by jonwil on Sun, 21 Dec 2014 04:21:56 GMT
View Forum Message <> Reply to Message

I can tell you right now that you will find it VERY diffivcult to do what you plan to do.
TT has been working on this for years as a team and there are still large chunks of the gqame engine we dont have any real knowledge of.
Do you knopw the W3D file format? How pathfinding works? How to do collision detection? How the physics sysrem works? How the game talks to WOL? How to read ASM and reverse engineer software? How network packets are compressed? How to load the mix files in the correct order and figure out which file to read if the same file is in more than one mix file?

Not to mention all the complex formulas for everything from damage to vehicle suspension to the way a soldiers legs animate.

Oh and btw any code you may have (including the cnetwork.cpp file you mention above), unless it specifically says its free to use you arent allowed to use it, doing so is a oopyright violation. (asaid code shouldnt be public anyway, it was leaked though a few unfortunate events) The code you can get on the official tt website is free to use though.

Subject: Re: wwnet
Posted by [EE]pickle-jucer on Sun, 21 Dec 2014 10:16:43 GMT
View Forum Message <> Reply to Message

jonwil wrote on Sat, 20 December 2014 21:21
Do you know the W3D file format? How pathfinding works? How to do collision detection? How the physics system works? How the game talks to WOL?
No, I don't know anything about these yet.

jonwil wrote on Sat, 20 December 2014 21:21
How to read ASM and reverse engineer software?

I've learned a bit over the past year or so, but I haven't learned enough to be able to understand a large portion of compiled code easily.

jonwil wrote on Sat, 20 December 2014 21:21How network packets are compressed?

Exactly what I'm trying to figure out right now.

jonwil wrote on Sat, 20 December 2014 21:21
How to load the mix files in the correct order and figure out which file to read if the same file is in more than one mix file?

I don't quite understand what you mean by this, but no I don't know anything about that.

jonwil wrote on Sat, 20 December 2014 21:21
Not to mention all the complex formulas for everything from damage to vehicle suspension to the way a soldiers legs animate.

I was under the assumption that the server would have nothing to do with the animation, if that isn't the case, I would definitely not know how to do that.

jonwil wrote on Sat, 20 December 2014 21:21
Oh and btw any code you may have (including the cnetwork.cpp file you mention above), unless it specifically says its free to use you arent allowed to use it, doing so is a oopyright violation. (asaid code shouldnt be public anyway, it was leaked though a few unfortunate events) The code you can get on the official tt website is free to use though.

I assumed it was something like that from the "Confidiential -- Westwood studios" at the top of each file, I was only looking at it to get a better understanding of the netcode, I hope that is okay.

All in all, I agree with you that I, by myself, probably wont be able to make a fully(if even at all) functional game server emulator. That certainly wont stop me from pursuing this as a small project. Heck, I'll be happy if I can just get the clients in game and the connection kept alive.

Thanks for your advice, I really appreciate it.

Subject: Re: wwnet
Posted by [EE]pickle-jucer on Fri, 01 Jul 2016 23:47:06 GMT
View Forum Message <> Reply to Message

Okay, so 1 and 1/2 years later and I've made some tiny progress . But I do have a few open questions if anybody would be willing answer them.

When I'm parsing raw packets sent between the client and server, this is the basic logic I'm following:

Step 1: Read in the CRC32 and compare it to the rest of the packet
Step 2: Read in the 2 byte header and bitmask out the PacketLength and IsMorePackets bit fields.
Step 3: Read in PacketLength many bytes and save them to later parse into Type, ID, SenderID, BitLength, etc.
Step 4: If(IsMorePackets == true){ goto Step 2 }

The 2 byte header is like such:
(1){1111111111}[11111]

```
() = (data >> 15) & 0x1    = IsMorePackets (Signifies if there is another packet after it.)
{} = (data >> 5)  & 0x3FF  = PacketLength (Packet length in bytes)
[] = (data >> 0)  & 0x1F   = Unknown (Somehow relates to decompression of repeated data?)
```

The question I have about this is about a function called when (Unknown - 1 > 0). The function in question, which is at ".text:0061BD90" (client address), seems to relate decompression of repeated data, but is too large for me to understand with my current knowledge of ASM. Does anybody know how this works, what it's supposed to do, or even its original name?

Also, while comparing the client<->server communication of the original client with a client running with TT scripts, I noticed a new packet type (8). This made me wonder, does TT hijack any fields that would make a client with TT scripts NOT be able to play on a server without TT scripts?

Subject: Re: wwnet
Posted by jonwil on Sat, 02 Jul 2016 02:16:00 GMT
View Forum Message <> Reply to Message

That function at 0061BD90 happens to be a function even TT hasn't been able to figure out.

As for TT vs non-TT, anything preventing communications between TT and non-TT systems is a bug, we (StealthEye specifically) put a lot of work into making sure it was possible to communicate between TT and non-TT.

What do you mean by "packet type" and do you have a list of the packet types you have identified so far?

Subject: Re: wwnet
Posted by [EE]pickle-jucer on Sat, 02 Jul 2016 03:36:32 GMT
View Forum Message <> Reply to Message

jonwil wrote on Fri, 01 July 2016 19:16That function at 0061BD90 happens to be a function even TT hasn't been able to figure out.
Ah, I see.

I probably should have clarified: I didn't notice anything broken, I was just wondering if there was breaking changes to the networking code that I needed to worry about in the future. It is really cool that TT was able to keep backwards compatibility along with the new features added!

The "packet type" I was talking about are the ones prefixed with "PACKETTYPE_", I don't know what they're referred to as normally because I just got them from the binary. Though, after a quick look at the strings from a TT bandtest.dll, it seems the new one I saw was probably "PACKETTYPE_RESOURCE_MANAGER"

So currently I have:

```
enum{
 PACKETTYPE_UNRELIABLE
 PACKETTYPE_RELIABLE
 PACKETTYPE_ACK
 PACKETTYPE_KEEPALIVE
 PACKETTYPE_CONNECT_CS
 PACKETTYPE_ACCEPT_SC
 PACKETTYPE_REFUSAL_SC
 PACKETTYPE_FIREWALL_PROBE

 // New from TT
 PACKETTYPE_RESOURCE_MANAGER
};
```

I'm currently able to parse all except: PACKETTYPE_UNRELIABLE, PACKETTYPE_FIREWALL_PROBE, (now) PACKETTYPE_RESOURCE_MANAGER, and partially PACKETTYPE_RELIABLE because I haven't implemented all of the NetClassIDs.

Subject: Re: wwnet
Posted by [EE]pickle-jucer on Fri, 15 Jul 2016 15:32:28 GMT
View Forum Message <> Reply to Message

Okay I've figured it out for the most part, it kind of clicked in my head while I was looking at some documents describing the Quake 3 networking code's delta compression.

I'm fairly certain that the function at 0061BD90 was originally named "PacketManagerClass::Reconstruct_From_Delta" and it reconstructs a packet with the given delta packet patch. Likewise, the function at 0061BB30, which I'm also fairly certain was named "PacketManagerClass::Build_Delta_Packet_Patch" does the opposite of the previous and generates a delta packet patch when given multiple packets.

I don't know if this will be useful to anyone else, but while I was trying to reverse engineer "PacketManagerClass::Reconstruct_From_Delta" I hooked the original function to jump into my code to see if it was working, this is what I ended up with (As far as I can tell with testing, it matches the functionality of the original 1:1):

View Code
```
#include <iostream>
#include <windows.h>
#include <cstdio>
#include <cstdint>
#include <cstring>

// PacketManagerClass::Build_Delta_Packet_Patch
// Original: 0x0061BB30

// PacketManagerClass::Reconstruct_From_Delta
// Original: 0x0061BD90
//



/* NOTES:
 *
 * If the lowest bit of the delta packet header is set (*delta_packet & 1), it signifies what I call
QWORD mode for lack of a better name.
 * QWORD mode simply has additional bits in the delta bitmask bytes which tell this function to
copy a QWORD (8 bytes) from some position in
 * the srcBuf to the same position in the dstBuf.
 */

int32_t __cdecl my_Reconstruct_From_Delta(uint8_t *srcBuf, uint8_t *dstBuf, uint8_t
*delta_packet, int32_t PacketLength, int32_t *delta_bytes_read)
{
 int32_t BitIndex;
 int32_t BitIndex_2;
 uint8_t *delta_byte_data_ptr;
 uint8_t *delta_byte_data_ptr_1;
 uint8_t *DiffBytePtr;
 int32_t *RelativeByteIndex;
 int32_t *CurrentRelativeByteIndex;
 int32_t DiffByteCount;
 int32_t OrigByteCount;
```

```
int32_t RelativeByteIndexes[1024];

// Check for null pointers
if (!srcBuf && !dstBuf && !delta_packet)
{
 return 0;
}

// Check for proper packet length
if (PacketLength > 500)
{
 return 0;
}

// Init some variables.
BitIndex = 0;
BitIndex_2 = 0;
delta_byte_data_ptr = delta_packet + 1;
delta_byte_data_ptr_1 = delta_packet + 1;
DiffByteCount = 0;
OrigByteCount = 0;

if (*delta_packet & 1){
 if (PacketLength - 7 > 0)
 {
  // Get the length needed for delta bitmask bytes
  // This rounds down to the nearest whole number
  uint32_t BitMaskByteCount = PacketLength / 8;

  int32_t currentOffset = 0;
  do
  {
   int8_t isBitSet = ((uint8_t)*delta_byte_data_ptr >> BitIndex++) & 1;
   if (BitIndex == 8)
   {
    ++delta_byte_data_ptr;
    BitIndex = 0;
   }
   if (isBitSet)
   {
    // Copy full QWORD (8 bytes), from
    memcpy(&dstBuf[currentOffset], &srcBuf[currentOffset], 8);
    OrigByteCount += 8;
   }
   currentOffset += 8;

   --BitMaskByteCount;
  } while (BitMaskByteCount);
 }

 // If there is any remainder of the packet length (because the previous section rounded down)
 if (PacketLength % 8) {
  int8_t isBitSet = ((uint8_t)*delta_byte_data_ptr >> BitIndex++) & 1;
  if (BitIndex == 8)
  {
   ++delta_byte_data_ptr;
   BitIndex = 0;
  }
  if (isBitSet)
  {
   return 0;
  }
 }
}

// This 'for' block marks where diff bytes are and places the indicies in the
// RelativeByteIndexes array.
int32_t targetIndex = 0;
int32_t currentIndex = 0;
for (; currentIndex < PacketLength; )
{
 // Checks for QWORD mode:
 // if the bit is set, it was already copied about so do not check for single byte differences.
 if (*delta_packet & 1) {
  int8_t isBitSet = ((uint8_t)*delta_byte_data_ptr_1 >> BitIndex_2++) & 1;
  if (BitIndex_2 == 8)
  {
   BitIndex_2 = 0;
   ++delta_byte_data_ptr_1;
  }
  if (isBitSet)
  {
   currentIndex += 8;
   continue;
  }
 }


 CurrentRelativeByteIndex = &RelativeByteIndexes[DiffByteCount];
 targetIndex = currentIndex + 8;
 do
 {
  if (currentIndex >= PacketLength)
   break;
  int8_t isBitSet = ((uint8_t)*delta_byte_data_ptr >> BitIndex++) & 1;
  if (BitIndex == 8)
  {
   ++delta_byte_data_ptr;
   BitIndex = 0;
  }
  if (isBitSet)
  {
   // Bit set:
   // Copy the original byte
   dstBuf[currentIndex] = srcBuf[currentIndex];
   ++OrigByteCount;
  }
  else
  {
   // Bit not set:
   // It uses a diff byte at this position, save the index to the array.
   *CurrentRelativeByteIndex = currentIndex;
   ++CurrentRelativeByteIndex;
   ++DiffByteCount;
  }
  ++currentIndex;
 } while (currentIndex < targetIndex);
 currentIndex = targetIndex;
}

// Get a pointer to the actual diff byte,
// if BitIndex > 0 then delta_byte_data_ptr still points to a bitmask byte so go 1 byte further.
DiffBytePtr = delta_byte_data_ptr;
if (BitIndex > 0){
 DiffBytePtr = delta_byte_data_ptr + 1;
}

// I beleive this would come out to the size of all the delta bytes, delta bitmasks, and the delta
header.
*delta_bytes_read = DiffByteCount + DiffBytePtr - delta_packet;
if (DiffByteCount > 0)
{
 // Get the first relative byte index
 RelativeByteIndex = &RelativeByteIndexes[0];

 // Loop over all of the diff bytes and place at the correct relative position in the dst buffer.
 for (int32_t i = DiffByteCount; i > 0; i--){
  dstBuf[*RelativeByteIndex++] = *DiffBytePtr++;
 }
}

// Returns the amount of diff bytes used + original bytes used
// This should match the original packet length, if it doesn't
// then something has gone wrong.
return DiffByteCount + OrigByteCount;

}


BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved){
if (fdwReason == DLL_PROCESS_ATTACH){
 // While its not reccomended to do alot in DllMain, this doesn't really need a new thread or
anything.

 void* Original_Reconstruct_From_Delta = (void*)0x0061BD90;
 size_t jmp_instruction_size = 5;

 // Change memory permissions
 DWORD flOldProtect;
 VirtualProtect((void*)Original_Reconstruct_From_Delta, jmp_instruction_size,
PAGE_EXECUTE_READWRITE, &flOldProtect);

 // Make a simple jmp patch/hook
 int32_t relativeOffset = (((int32_t)my_Reconstruct_From_Delta -
(int32_t)Original_Reconstruct_From_Delta) - jmp_instruction_size);
 *(BYTE*)Original_Reconstruct_From_Delta = 0xE9;
 *(int32_t*)((uint32_t)Original_Reconstruct_From_Delta + 1) = relativeOffset;

 // Restore memory permissions
 DWORD UnneededPerms;
 VirtualProtect((void*)Original_Reconstruct_From_Delta, jmp_instruction_size, flOldProtect,
&UnneededPerms);

 MessageBoxA(NULL, "Done hooking PacketManagerClass::Reconstruct_From_Delta!", "Done
hooking", 0);
}
return TRUE;
}
```

Subject: Re: wwnet
Posted by BillieJoe67 on Thu, 28 Jul 2016 13:57:33 GMT
View Forum Message <> Reply to Message

I have python code which can parse connection requests that may interest you. I tried pm'ing but apparently your username doesn't exist, do you have an email address you could pm me?

Subject: Re: wwnet
Posted by jonwil on Sat, 30 Jul 2016 04:38:06 GMT
View Forum Message <> Reply to Message

How did you reverse engineer that function?

Subject: Re: wwnet
Posted by [EE]pickle-jucer on Sat, 30 Jul 2016 20:39:24 GMT
View Forum Message <> Reply to Message

I have access to IDA Pro and the Hex-Rays decompiler through a friend of mine, so I was able to get it decompiled to (still very confusing) C code. After that it was just simplifying some the things that hex-rays produced (ex: inverted if conditions, bit shifts used as multiplication/division by powers of 2, etc).

As for the name, I was just guessing at the time of posting because of a single reference to it in the leaked code. But as I recently found out about the LFDS having debug symbols, I can confirm that it is right.

The code was very messy in the end, but it was enough to understand what it was doing so that I could re-implement it in another language.

Subject: Re: wwnet
Posted by jonwil on Sat, 30 Jul 2016 22:47:11 GMT
View Forum Message <> Reply to Message

IDA and Hex-Rays is what I use for my own reverse engineering
Without it, the custom scripts stuff wouldn't even exist.

Subject: Re: wwnet
Posted by jonwil on Mon, 04 Dec 2017 03:53:34 GMT
View Forum Message <> Reply to Message

Were you able to make any progress on this?
Were you able to reverse engineer PacketManagerClass::Build_Delta_Packet_Patch?
Can I use your clone of PacketManagerClass::Reconstruct_From_Delta if I decide its worth using?

Subject: Re: wwnet

Posted by [EE]pickle-jucer on Thu, 01 Feb 2018 09:43:53 GMT
View Forum Message <> Reply to Message

Whoops, sorry for the _really late_ response, I haven't been checking the forums very often.

jonwil wrote on Sun, 03 December 2017 20:53Were you able to make any progress on this? Were you able to reverse engineer PacketManagerClass::Build_Delta_Packet_Patch?
No, unfortunately. I never made any significant progress since I posted last time. My life got a bit more busy than I had liked with moving across the country, college, work, etc.

As far as I can remember, I thought it would be best to be able to parse them completely first, then try put them back together. So, with some really helpful information from BillieJoe67 I was trying to figure out how the world X/Y/Z position encoders were initialized from the level extents (instead of just reading the value from directly memory, which is what I was previously doing). That meant I first needed to understand how the chunked file format worked and what all the (micro)chunk ID's in the LSD file were.

I guess it all just became too complex and time-consuming with all the other stuff I had going on, so I stopped messing with it. I've recently been thinking about reverse engineering renegade again though, so I might try to figure out PacketManagerClass::Build_Delta_Packet_Patch soon, I'll post it here if I do.

jonwil wrote on Sun, 03 December 2017 20:53Can I use your clone of PacketManagerClass::Reconstruct_From_Delta if I decide its worth using?

Yes, absolutely. Feel free to use it for anything you want. You'll probably want to to clean up the code a bit though, looking at it now it seems that I couldn't even decide whether to use CamelCase or under_scored variable names hahahahaha.