How can I use IDA to help decrypt W2M packet (v175.1 GMS)

25-07-16

Punon
Member

Rank	Member
Join Date	Apr 2014
Posts	68

How can I use IDA to help decrypt W2M packet (v175.1 GMS)

Hi everyone,

I've been trying to use the IDA but I still have no idea what it is capable of. Some questions I have in mind are:

- how does the IDA assist in decrypting packets?
- do the Decodes in IDA match the lengths of packets that are sent? (e.g decode1 = mplew.write?)
- why are some opcodes that are listed in CField in server source not present in IDA CField? (e.g. I can't find W2M in ID

Thanks in advance and I'd really appreciate any effort of helping!

EDIT: I don't mind reading previously posted guides if you wish to post some. I read the ones in Other Development-> and am still confused.

R

25-07-16

Eric
Moderator

Rank	Moderator
Join Date	Jan 2010
Location	DEV City
Posts	3,188

Re: How can I use IDA to help decrypt W2M packet (v175.1 GMS)

Well, it's kind of self explanatory, really.. IDA is reversing your unpacked MapleStory client and gives you the precious convert the assembly calls into C-Pseudocode. This being said, it's not only helping you "decrypting packets" (yes, it ha entire client encryption, keys, etc the whole process there you can copy or find if it were to change), but it also has the packet structures as well.

Of course they do! The Decode's are how nexon actually handles it, it's just their naming to read in and write out bytes Decode's in the client-side are known as the COutPacket on the server-side, and the client's Encode's are known as the CInPacket on the server-side. COutPacket is the equivalent to Odin's MaplePacketLittleEndianWriter, as CInPacket is e so SeekableLittleEndianAccessor. The bytes you see in packets all being outputted when you log? Most people sniff ar but this will actually tell you when the client requires an int here, a long there, and two shorts over there. Every byte p a packet is available in IDA.

So before I move on, here's some of this explained more. So in the client you have Decode calls and Encode calls. Wha doing is either a) requiring a certain length of a packet sent, or b) directly sending these amount of bytes to be read o server-side. Now Decode2 is the same as Encode2 as mplew.writeShort is the same as slea.readShort. Nexon uses the calls:

Decode1 -> The equivalent of mplew.write. In most libraries this is 'writeByte'.
Decode2 -> The equivalent of mplew.writeShort. In most libraries this is 'writeShort'.
Decode4 -> The equivalent of mplew.writeInt. In most libraries this is 'writeInt'.

The Decode's are really just saying the bytes it's encoding. If you know that there's 1 byte in a byte, 2 bytes in a short, an int, and 8 bytes in a long, that's the same as Nexon's Decode"X".

The one Decode/Encode call that doesn't follow that exact principle is DecodeBuffer. This is a void* datatype used her rather than your usual, it would be EncodeBuffer(pData, uLength). This means write pData that is uLength bytes long

of bytes. Now, you don't really have the ability to do this in Java, so you just make a few different calls that it is going t

1) EncodeBuffer(pData, 0xD). This is a 13 length buffer (0xD) that Nexon uses for names in some packets. In Odin, this writeAsciiString. The difference between this and your writeMapleAsciiString is that it HAS to be 13 bytes long. Your n 10bytes will crash. This is why you write(0) several times to make it exactly 13 bytes.
2) EncodeBuffer(pData, 0x8). This is a 8 length buffer (0x8) that Nexon uses for timestamps, serial numbers, and othe Odin, this is writeLong. Commonly found in Odin people don't realize this and do two writeInt's, or a writeInt(data) an writeInt(0) afterwards.
3) EncodeBuffer(pData, 0x8). This is a secondary buffer Nexon uses that writes doubles (again, a long is 8 bytes and s double). In Odin you don't really write doubles much, but when they do they shift around and write the first bytes as (example I can remember was their dUnitPrice in getNPCShop)

Ok, so why don't they all list in that location? Simple. Nexon organizes packets to be in their class files, public MapleSt emulators don't; instead, they make everything in one big file called MaplePacketCreator. Once you learn the basics o understand the Decode's, next you should learn the class setup. I'll break it down for you.

CField. So this is the biggest class of them all. ALL Field-related objects will be in this class. This means you will find Fie packets like the OX Quiz or some map notices or even Clocks will be here. Field calls to all of the objects like I mention will find Mob packets, Npc packets, Reactor packets, Summoned packets, etc. Each main class in Nexon terms is a "po is why you'll find CMobPool::OnPacket, CNpcPool::OnPacket, and so on. Field is generally the most important part upd updating the game.

To answer your question about not finding warpToMap, or in Nexon terms, OnSetField, it is because in CField they use ranges (a simple if-statement range of >= and <= will call to the outer subs). However, W2M isn't always there it seems most of the time it is. What you're looking for is CStage::OnPacket, which will hold the function CStage::OnSetField (th

So now you know that all field objects and content is in CField, CStage migrates you in (oh! and CStage::OnSetCashSh even new things like StarPlanet are also included here -- they are the major packets that get you ingame) we can mov the smaller subs.

So the first OnPacket you'd need to update first off is CLogin. This is your LoginPacket if you use Lithium. It's all login you'll need. Now remember when I said CStage wasn't always in CField? When you can't find it there, your next best b find it in CLogin.

Technically before CLogin is CClientSocket. This is your connection and how you AliveReq/Ack with the client. It is the OnConnect (in Odin terms the "handshake", or "getHello" packet). It is what initializes your IV sequences, and it is wha handles your PING/PONG packets. This class will usually have a call to CLogin, and I think CWvsContext too.

So the next big sub is CWvsContext. This is your OnPacket sub for nearly everything in the game. Your general handle your general packets are all within this class. This handles your inventories, your skills, your buffs, all User Interface re content and more, really.

Now I'm a bit of an oldie I guess, I work on v90 and not up-to-date. In v175, you're going to have a lot more bigger su going to use like Star Planet and that Monster Life stuff and so on. However, things like CField will always be the same reference the major classes you need, CStage will always get you ingame, CWvsContext will always handle your gener gameplay, CLogin and CClientSocket will always be the login/connection packets.

One last note before I hope this has helped. You ever notice how addCharLook and addCharStats etc is used? Technic was poorly done in Odin, but to be fair they didn't know. In reality these are actually more classes that call Encode's. F OnSelectWorld packet (in Odin this is getCharlist), you add addCharLook's. In IDA/Nexon terms, these are your Avatar Finding AvatarLook::Decode in your client will tell you all the information used there. Next is character stats. You see addCharStats used, I think in Odin only in getCharInfo (OnSetField), but this is in Nexon terms GW_CharacterStat::Enc holds all of your stats from your name, gender, map id, all the way to your player stats (str,dex,int,luk) to even things now like if your character is a burning character or not. and last but not least, is CharacterData. This is your Character much, packet-wise anyway. This is your inventories, your stats, your skills, your map transfers, everything in your get packet is just flags used and marked in CharacterData. When you do find OnSetField, make sure you find CharacterData::Decode in the client. This will be the toughest challenge to get working because nothing is all Decode have to keep track of the iPacket! (usually the 2nd or 3rd parameter that is used. It will be like CInPacket::Decode1(**a1**) something unless you are using a PDB)

What I highly suggest is looking over the v95 GMS PDB and the KMST PDB that was recently leaked. They are fully nar are very helpful, even at v175 to know how things work.

Hope this helps! Oh, and IDA/Client-related questions belong in Other Developments, not Server. :)

R

25-07-16

Punon

Member

Rank	Member
Join Date	Apr 2014

Re: How can I use IDA to help decrypt W2M packet (v175.1 GMS)

Thanks for your comprehensive reply, I can now understand the general whereabouts of major content like CLogin, C To be honest, I did not expect such an informative reply, thanks again!

Also whoops on the wrong section.

Posts	68

R

« Previous Thread | Next Thread »