Complete Opcodes Guide

By Xavier Geerinck

Revision

1. 2/11/2012 Initial Document
2. 4/11/2012 Added info to the CMSG and SMSG

# Purpose

The purpose of this document is to explain what opcodes are and how to find them, it is the book to read if you are interested to learn what opcodes means, what aslr is, what reverse engineering is and much more.

I hope that everything is written in such a way that it becomes clear for the reader to understand what is being told here.

# Tools Used

The tools used in this guide are

1. IDA Pro 6.1 With Hex Rays
2. IDA database by TOM\_RUS ( These can be found on ownedcore.com )
3. World Of Warcraft ( Our test subject )

# ASLR

ASLR stands for Adress Space Layout Randomization; this is the randomizing of the memory address where the module (.exe file, .dll file) is loaded.

The usefull thing is that Windows XP does NOT support ASLR, because of this the offset that will be added will be 0x00400000 (which is the standard offset) so we will get this offset when we disable ASLR.

## Rebased

If it’s rebased and you open the file in a decompiler like IDA then it will always start at a random offset, this is why we have to disable ASLR

## Disabling ASLR

There are 2 ways to disable ASLR, with a setdllcharacteristics tool that can be found on google, or through the Microsoft Visual Studio tool (which is the easiest way)

1. With Visual Studio

To disable ASLR in Visual Studio you just have to type this command line:

Editbin /dynamicbase:no wow.exe

1. With setdllcharacteristics

open up a command prompt and enter:

setdllcharacteristics –d wow.exe

( can be found here: <http://blog.didierstevens.com/2010/10/17/setdllcharacteristics/> )

## Not rebased

On the internet you will mainly see Rebased or Not Rebased when searching after opcodes or offsets, this is because of the random offset that gets added because of ASLR.

When they say this then we know that the offset starts at 0x00000000 then we can always find it in our IDA decompiler or any other decompiler when we got the same offset.

Example:

We got a offset like this:

009216D0 Script\_GetTime (Not rebased)

Then we know that we can find it in cheat engine (as an example) by going to WoW.exe + 0x009216D0 – 0x00400000 (the 0x00400000 is because we disabled ASLR )

And we can find it in IDA by going to the offset 009216D0 because the rebased offset is already off it.

Rebased

# Packets

## CMSG

The CMSG packet is a packet that gets sent from the client to the server.

## SMSG

The SMSG packet is a packet that gets sent from the server to the client.

## MSG

The MSG packet will be sent in both ways

# What are opcodes?

Opcodes, the term opcodes comes from the creation of the computer, it stands for Operation Code and specifies the operation to be performed.

Examples of operations to be performed are: register values, values in the stack, memory values, I/O ports, shift operations, copying of data, …

Complete example:

Opcode: ‘SMSG\_ACCOUNT\_DATA\_TIMES = 0xE48’

We can see that it holds the offset 0xE48

# Types of opcodes

## CMSG

The CMSG opcodes are the one being sent to the server, this means that the data is collected on client side, then compiled into a package, encrypted and sent to the server.

To catch these on the server we have to first fetch the packet on the server side, then decode it and then handle the data in it.

## SMSG

The structure of a SMSG packet can be found in the handlers of the client, these handlers will be called when a SMSG opcode arrives, there they get decrypted and used for the client. To make a proper handler on the server side we need to know this structure so that we can use the same structure on the server to assemble the package and then send it to the client.

Before WoW 4.0 all the handlers were in a function ClientServices\_\_SetMessageHandler(opcodeNumber, handlerFunctionPointer) so before 4.0 we could easily find them by searching after ClientServices\_\_SetMessageHandler(12345,

As of 4.0.6 this process got trickier because they are now changing the opcodes on every client update.

# Finding Structures

## CMSG

## SMSG

1. One of the methods WoW uses to randomise the opcodes is by storing all the handler pointers in a huge array at a particular offset, then the offset gets derived from the opcode number by a mathematical transformation. By finding this transformation we can convert back and forth between the offsets and the opcodes.

So as an example, if we would want to find a client handler for a specific SMSG opcode then we can convert it to an offset using the mathematical transformation that we found, if we then search the full decompile after ‘+ 8080’ where the 8080 stands for the opcode then we got the handler for this opcode.

1. For the second methods, WoW does not saves them in the main array, they are instead handled by a special function called NetClient\_\_ProcessSpecialOpcodes which calls the appropiate handler.

It maps the opcodes to small numbers that can also be converted by the mathematical transformation.

Then it calls a function for each opcode to read the data where we can see the packet structure. Then it calls the actual handler through a dword pointer to do the work required by the packet itself. You can find the function that is assigned to it by following the dword’s xref (shortcut X in IDA)

Once you have found the handler you can discover the packet structure by looking at CDataStore\_\_\*\*\* calls like CDataStore\_\_GetInt32, CDataStore\_\_GetInt64 etc.

Once you figured out tge opcode number and its structure you may want to see how the client reacts to getting this data, for that you can probably use a debugging tool in your server tos end the test data.

# Example

## Finding SMSG Structure

RealmConnection::HandleAuthChallenge = 0x007F45B0,

## Finding CMSG Structure

# Usefull Shortcuts in IDA

|  |  |
| --- | --- |
| Shortcut | Description |
| G | Go To Address |
| F5 | View pseudocode |
| X | Jump To XHref |

# References

### Opcode

<http://en.wikipedia.org/wiki/Opcode>

### Dump Thread

<http://www.ownedcore.com/forums/world-of-warcraft/world-of-warcraft-bots-programs/wow-memory-editing/376194-wow-5-0-5-16048-x86-info-dump-thread.html>

### IDA Signature creator:

<http://www.ownedcore.com/forums/world-of-warcraft/world-of-warcraft-bots-programs/wow-memory-editing/219433-ida-pluging-generate-patterns.html>

<http://www.ownedcore.com/forums/world-of-warcraft/world-of-warcraft-bots-programs/wow-memory-editing/341148-how-make-signature-search-ida.html>

### Finding Opcodes IDA:

<http://www.ownedcore.com/forums/world-of-warcraft/world-of-warcraft-bots-programs/wow-memory-editing/260952-tutorial-how-find-simple-stuff.html>

### Memory Reading / Writing

<http://www.ownedcore.com/forums/world-of-warcraft/world-of-warcraft-bots-programs/wow-memory-editing/276015-tutorial-starting-wow-memory-reading-writing.html>

### Info about how the packets works.

<http://dl.dropbox.com/u/26011806/Skripsie.pdf>