Duping items in intersect (Or how to hack intersect and other games)

Introduction

This post is going to introduce people to the basics of getting started with hacking intersect. It will show you some of the tools needed, and how to put them to good use. This is just scraping the surface of what is possible. If you are interested in this topic you should try reading up on reverse engineering and c#.

Disclaimer

Any exploits that are found should first be reported to the intersect developers. You should do this privately and not via the bug tracker. They can be reached at: admin@ascensiongamedev.com. Abusing the flaw and not reporting it could be illegal. This guide is designed to be educational and help the developers secure their application.

Are we creating a usable hack:

Yes we are! This tutorial will show you how to duplicate items in Intersect, this will only work on version Build: 0.6.0.145 and below. I have reported the vulnerability and had made sure it was patched before releasing this tutorial.

What you need to know before using this tutorial

To fully understand this tutorial you should have a basic idea of how programming languages work, how computers work and how code is compiled. As well as a basic idea of how client/server applications work. Knowing C# is a plus as well.

Though all you really need to gain knowledge from this tutorial, is an eagerness to learn and googling skills. Hacking, computer programming or learning anything at the end of the day is just a lot of googling and practice.

Tools:

Notepad++ (optional)
dnSpy (x64)

How does it work

Intersect is made in a programming language called C#. If we have access to the source code we can change the behavior of the program. Because Intersect is an online system, all behavior should be authenticated by the server, so we should not be able to make client side changes that affect the game. But because the creators are human, they are going to make mistakes. Our job is to find spots they have failed to authenticate and take advantage of them. Or pass in values that the creators were not expecting to create undefined behavior.

Unfortunately for us, Intersect source code has not been released at the time of writing. Not to mention, most games we want to exploit probably will not have released their source code. While there are other ways to exploit games without using the source code or reverse engineering it. That is what we are concentrating on for this tutorial.

Fortunately for us, when a program is compiled it turns source code into machine code that is readable by the computer. This machine code can be read and used to reverse engineer the program to see what the original code may have looked like. While the functionality of the machine code is identical to the original source code, the decompiled version can differ significantly due to compiler optimizations. It is also designed for a computer to read, so it does not care about comments, or variable names. This makes the resulting assembly much harder to read.

While some languages get converted to assembly code on compilation, other languages such as C# get converted to Microsoft Intermediate language. This language contains much more information about the original source code. This makes are job much easier.

So how do we get the source code? Or a version that looks like it?

Let's get started

Step 1:

Download the tools listed above. Make sure that you get the x64 version of dnSpy as the intersect client is 64 bit. (The editor and patcher are 32 bit however).

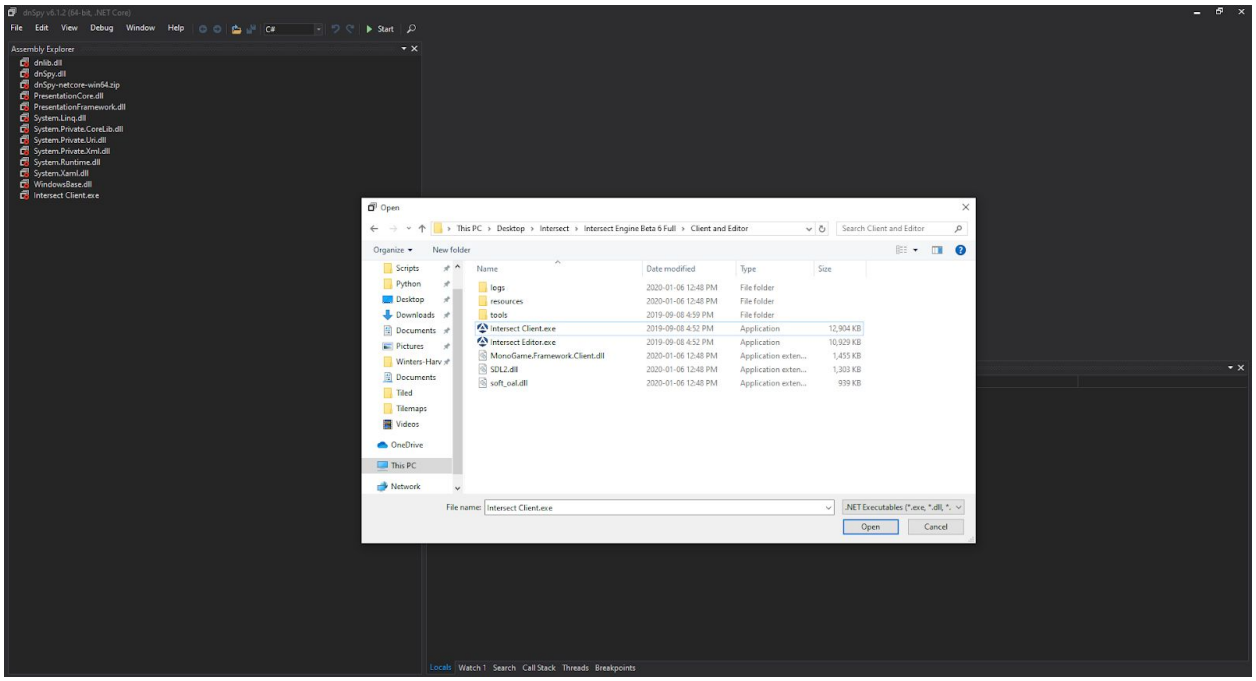
Step 2:

Download the game you wish to modify, or start up your own game and server.

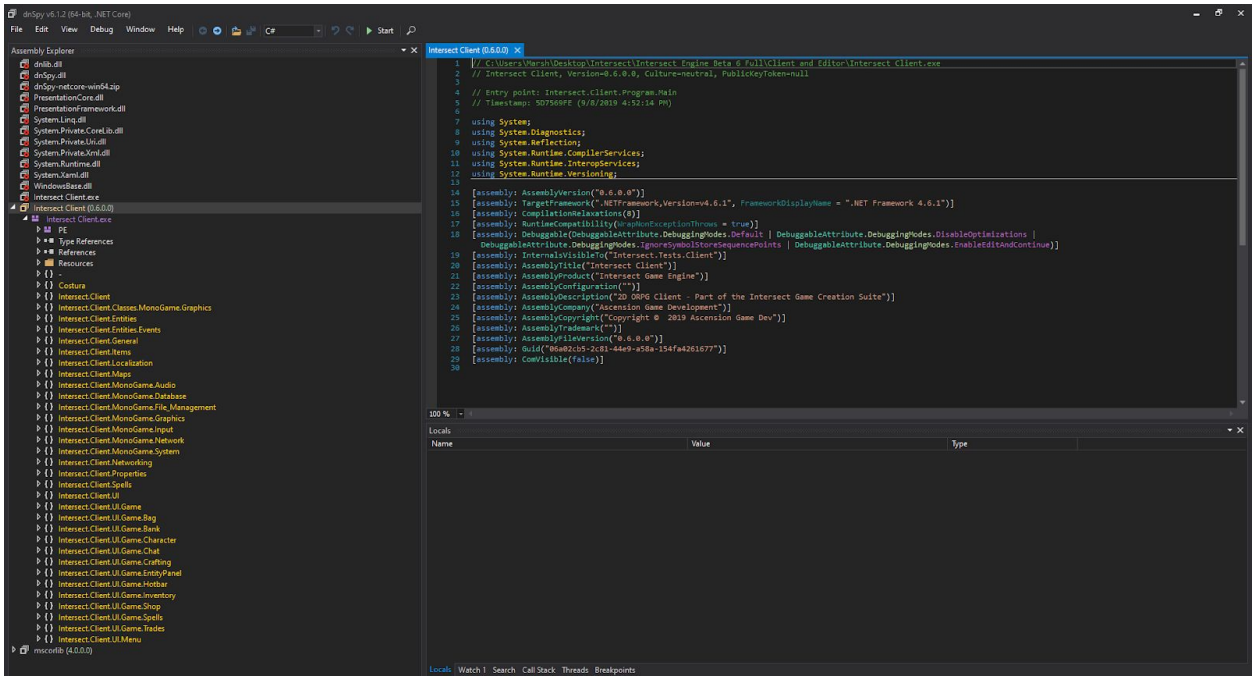
Step 3:

dnSpy is a .NET debugger and assembly editor. It can be used to translate a programs IR (intermediate representation) back to readable C#. This is not the exact source code that the developer wrote, but an estimation that will have the exact same behavior. You can also debug the application live, edit variables and even source code.

Open dnSpy.exe, go to file -> Open and select the game exe. Make sure you have the correct client executable. If they have a patcher the real client might be in the data folder. If dnSpy tells you that it is a 32 bit executable. You have selected the patcher and not the client.



You should now be greeted with this window. Your information might look a little different depending on what version you are debugging.

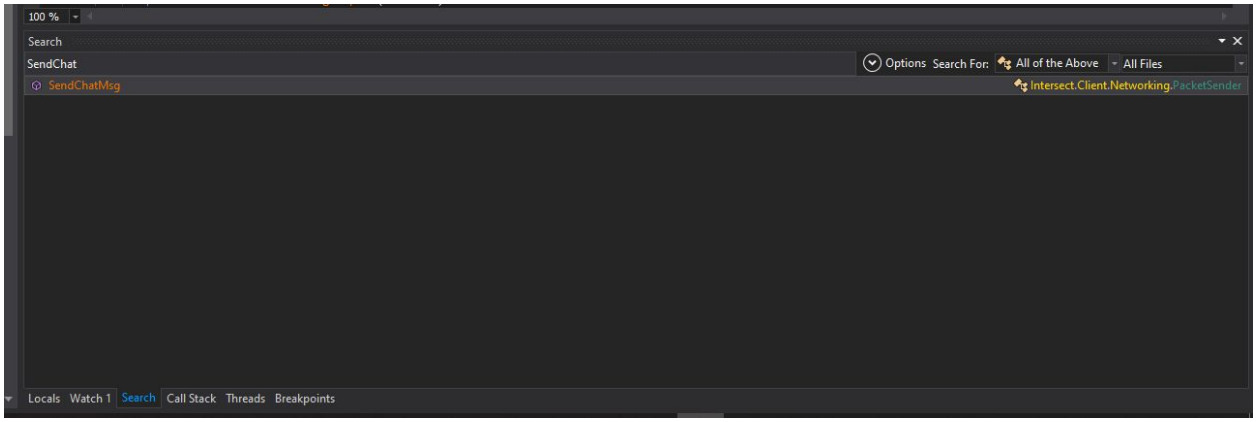


The left hand side of the window shows all the classes, namespaces and various files of the exe. The right hand side shows the source code.

Step 4

Our goal is to duplicate an item. First we have to think about how we normally get an item. We normally receive items by buying them in a shop, picking them up off the ground or perhaps as a reward from a quest. We need to find the code that sends this pickup item event from the client.

This can be done by searching various terms, or searching the source code. You can set a breakpoint in the code to see if the right section of code is being hit. I have already searched the source code and found the piece of code we are looking for.




```
        // Token: 0x060000EB RID: 235 RVA: 0x0000AB8E File Offset:
0x00008D8E
        public static void SendDropItem(int slot, int amount)
        {
            GameNetwork.SendPacket(new DropItemPacket(slot, amount));
        }

        // Token: 0x060000E9 RID: 233 RVA: 0x0000AB6F File Offset:
0x00008D6F
        public static void SendPickupItem(int index)
        {
            GameNetwork.SendPacket(new PickupItemPacket(index));
        }
```

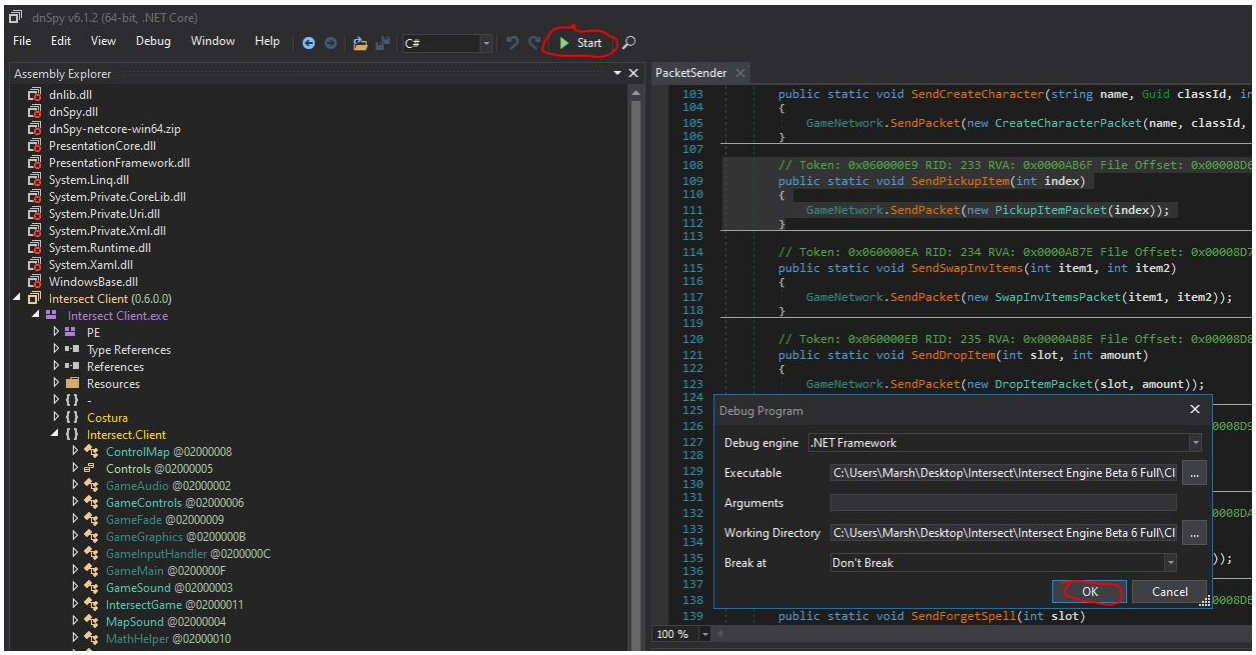
One of these functions sticks out, in the SendDropItem, we notice that it sends the amount of items dropped. This is probably for stackable items. What happens if we modified the code to drop more of an item then we actually posses? If we tell the server that we dropped 5 of an item when we only own one will five appear on the ground to be picked up?

While this could have easily worked, the server checks to see if more items then the player owns is dropped. So changing this to a higher value still only drops the max amount a player has. But this is what you need to do. Find all the functions like this and try modifying the values. Eventually you will find a value that is not checked by the server.

If you know about programming you know that sending values that the programmer would not expect can work well to Perhaps they have checked to make sure that the player owns enough of the item. But what happens if you drop 0 items? Or -1 items? Have they checked every case? These numbers could also trigger undefined behavior in the code that gives us what we want. This exact case happens in one of the above functions. Let's investigate.

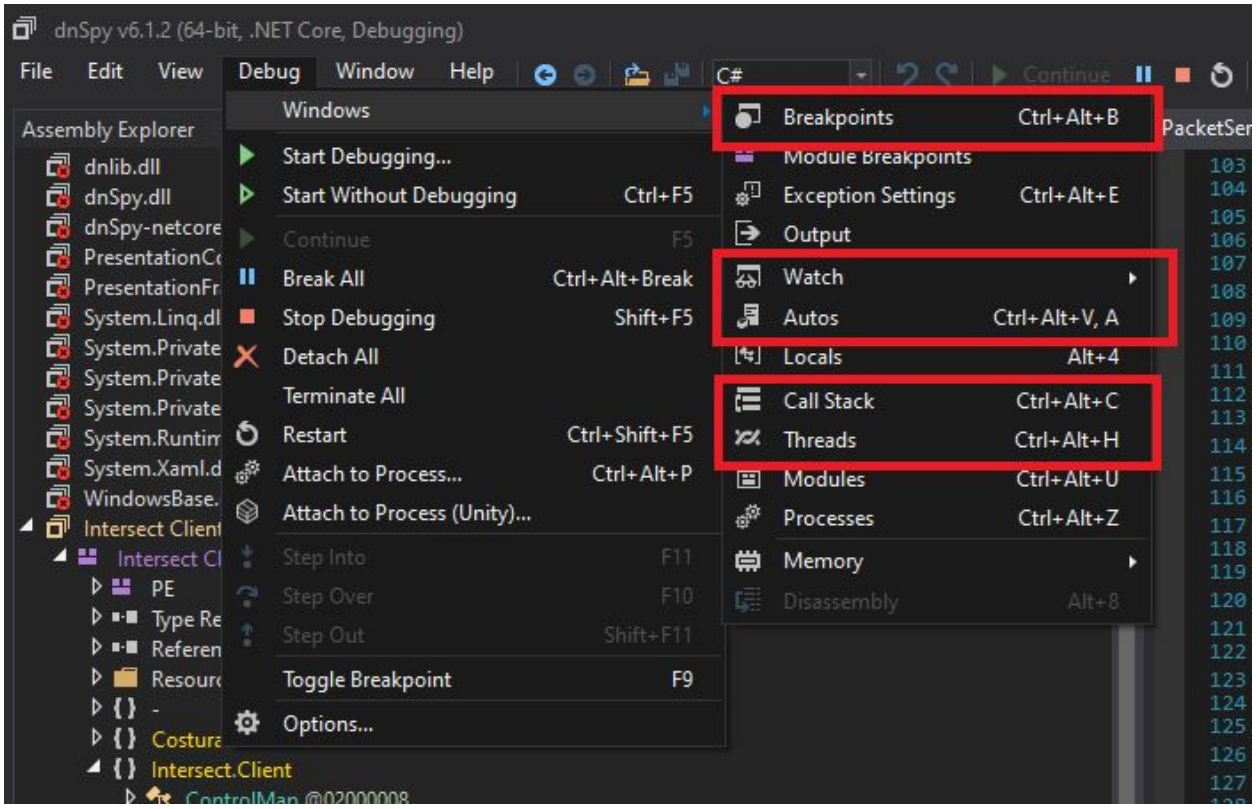
Step 5

Now that we have the functions that we are going to use. Let's start the program in the debugger. This will let us change the variables and source code while running the program. To do this hit the green start arrow. On the new dialog box that comes up. Simply hit ok.

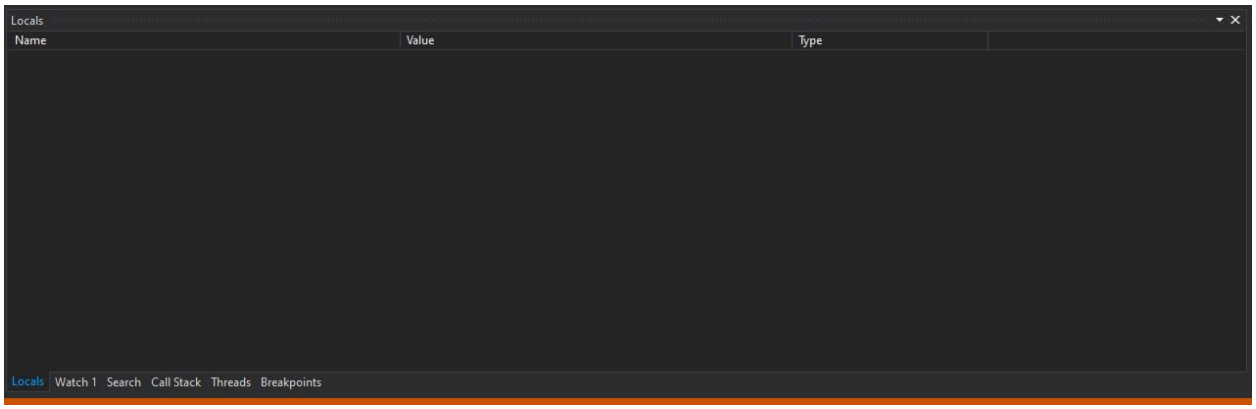


The game is now running, but the debugger is attached. You can hit the pause button to see what code is currently running. Just make sure that you resume the program after.

The default dnSpy view has a lot of the important debugging features missing. Once the program is running we should open all the windows we need. To do this goto Debug -> Windows and select the following items.

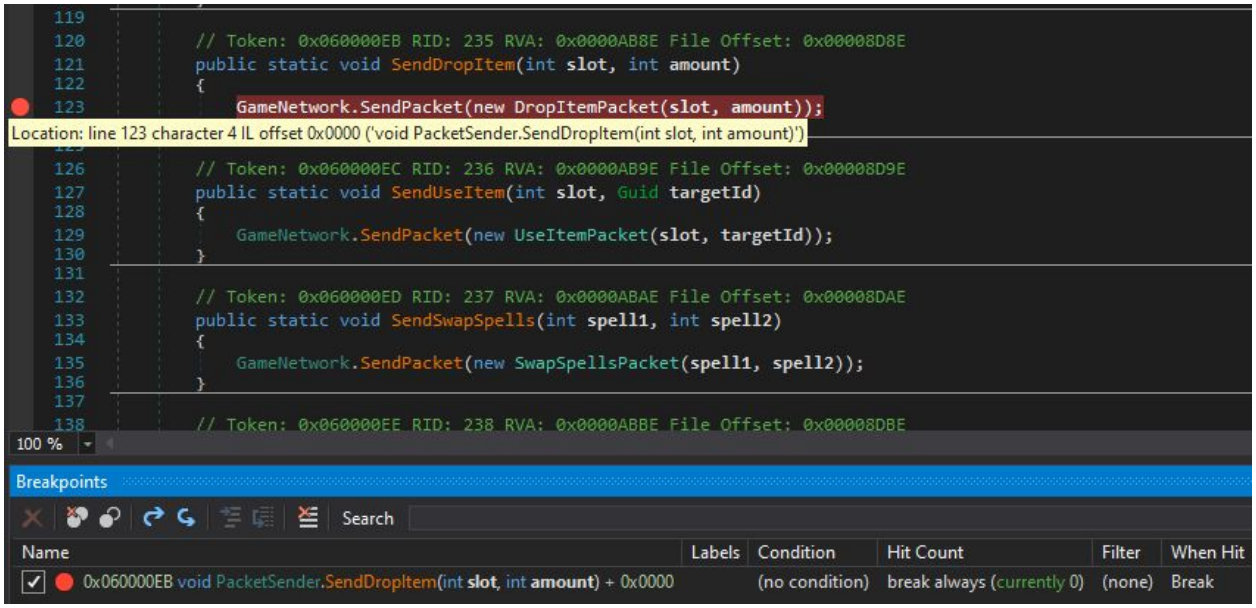


Your bottom window should now look like the following



Set a breakpoint on the line `GameNetwork.SendPacket(new DropItemPacket(slot, amount));`

To do this, click on the left hand column beside the line. A red dot should appear. You should also see the breakpoint in your breakpoints window.

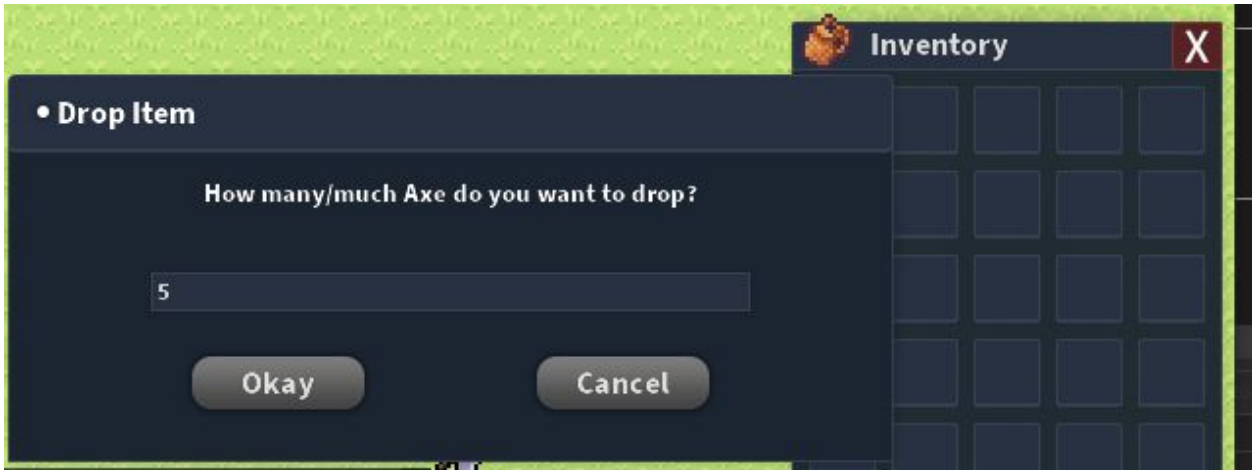


Now when you drop an item in the game, the debugger will stop in this function. When we are modifying it we must act quickly. Because it is an online game, it will kick us out if we spend to long debugging as it thinks we have disconnected since we did not send a packet in a long time.

We want to drop a non stackable item. When this function triggers we want to look in the auto window and change the amount value to -1 to confuse the server.

So get the item you want to duplicate and then drop it, dnSpy should now break into your function. Click on the locals window. You should see a “slot” and an “amount” field. Double click on the value column in your locals window and change the value to -1 and then press enter. The field might change to FFFFFFFF. Do not worry about that, it is just the hexadecimal representation of -1. Press F5 or hit the green arrow to start the program again. If you get disconnected message when resuming the game, you need to do the steps faster.

When the game is resumed, you will notice nothing has happened. The game did not drop any axes. It is now stuck in a confused state however. Try to drop the item again. You will now see this dialog.



The client is now confused and thinks that your non stackable item is stackable and giving you the option to drop more than one. Type in any number higher than 1. The game will break again, just resume it right away with no changes. You will now notice an item on the ground. Pick it up. You will see that the item is in your inventory, but the item is still on the ground as well. Pick it up again. You now have 2 of the item in your inventory. You have successfully confused the server and duplicated the item.

You can repeat these steps as many times as you want. Note that even if you type “5” you will still only get 2 of the item you drop.

Here is a video demonstrating the full process if the above text was not clear.

https://youtu.be/loPyMsx99t0

Final Notes

Try out all the other functions and see if you can find any more exploits. (I will give you a hint, you can change your sprite when creating a new character past the given class bounds). You can also change the gold amount in the trade menu.

Additional Reading

https://en.wikipedia.org/wiki/Undefined_behavior
https://en.wikipedia.org/wiki/Intermediate_representation
https://en.wikipedia.org/wiki/C_Sharp_(programming_language)
https://www.swarthmore.edu/NatSci/echeeve1/Ref/BinaryMath/NumSys.html
https://www.guru99.com/compiler-design-phases-of-compiler.html

https://docs.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-debugger?view=vs-2019