

Forum:[Hack Forums](#)

Thread: [\[2/2\]\[ARM 64\] \[ASM\] \[TUTORIAL\] Manipulating Enum values parsed into functions in ASM](#)

Biggy_Asian - October 8, 2024, 6:14 am

[Credited User link \(click me. OFFSITE CAREFUL\)](#)

Hope the admins don't mind me giving credit although it is off platform...

I saw this posted on another forum related to game hacking/cheats. I think I attempted to post another thread that had all the links however I think it may have broken offsite rules or I just didn't submit it either way I am going to post the hidden content of the post but also include the true OP on the othersites link to the same post as I am not gonna claim their work for my own.

This is going to be the 1/2 in the 2 part posts I will send here. Like I said this is from a forum related to game hacking (mainly unity games) for ios however you can still 100000% take things away from this as it should give you an understanding of how ASM on ARM64 devices works when interacting with the target.

What are Enums?

An enum type is a special data type that enables for a variable to be a set of predefined constants. The variable must be equal to one of the values that have been predefined for it.

Using enums save time for developers and allow easier understanding of what a function is doing.

What do Enums look like?

Enums will be layed out like in the image below:

As you can see their layout is like a table. Since they're similar to a table we can treat them as such!

So 'Income' Will be index 0 and the one after will increment by one (current Index + 1 (0 + 1 in this case)).

How do we manipulate the Enums?

Okay so the way we would manipulate Enums is by offset patching (at least in this tutorial as this is really the only way I know).

For the offset patch example I will show you how to manipulate the given Enum to a function for it to give us money instead of taking out money.

Example:

The targeted function we want is the Apply Function as this either Adds or Removes currency

depending on the given CurrencyFlow:

Now since we have the function we want we will go to find a reference. After finding your reference to your target function we will want to figure out what is storing the enum parsed.

Below I have my reference of the Apply function including any parsed values:

Now that we have found it we will need to figure out which one of the X(number here) is parsing the Enum. However it will only be in the ones that are before the Wallet\$\$Apply call.

We can do this by going back and comparing the arguments positions in IDA and DNSpy:

As you can see the CurrencyFlow is stored in b]X2[/b]. So to make it not remove currency but instead add it we will need to change the flow that is parsed to Income:

To do this we will first get the offset of X2 and then write the index number of the Enum (In this case it will be Income) which will be 0.

[Correction](#) (Correction pasted below from original post):

Quote:Well, seem you wrong it should be x1, x2, x3,... x0 always pointer class for function non static

Remember, Tables and Enums both increment from 0 onward.

Code:

Code // Offset to edit: 0x3326E18 (Get the offset from IDA)

// This will be the code that makes the CurrencyFlow resolve to 'Income' Instead of 'Outcome'

MOV W3, #0 // This will convert t0: 03008052

Once you apply this patch to the given offset it will parse 'Income' Enum from the CurrencyFlow instead of 'Outcome'.

Now you understand how to read enums and parse your own!

Hope this helped you!