

Forum:[Hack Forums](#)

Thread: [\[Rust\] Obfuscate your program's Strings and Function name using Rust - Part 1](#)

Infinity. - October 27, 2024, 4:23 am

? Rust Guide: String & Function Obfuscation ?

Hi mates, I'm back again with a new guide, and today we shall dive into string + function name obfuscation. I will show you how to encrypt a string using a simple XOR cipher and decrypt it at runtime.

Additionally, we shall use function pointers to indirectly call secret functions, which simulates function name obfuscation to hide what the function actually does.

? What is XOR Encryption? ?

First, let's get familiar with XOR encryption. When I look up a definition, I get something like this:

Quote: "XOR encryption is a simple encryption technique that uses the XOR (exclusive or) bitwise operation. The XOR operation compares two bits:"

What does it mean by "comparing two bits"?

If the bits are the same (both 0 or both 1), the result is 0.
Let's look at some bit comparisons:

0 XOR 0 = 0
1 XOR 1 = 0

If the bits are different (one is 1 and the other is 0), the result is 1.

1 XOR 0 = 1
0 XOR 1 = 1

This behavior makes XOR perfect for encryption because XOR-ing the same value twice restores the original data. This means we can use the same operation for both encryption and decryption.

For example:
A XOR B = C (Encryption)
C XOR B = A (Decryption)

? Simple Example: Encrypting the letter 'A' ?

Let's encrypt the letter 'A' (ASCII value 65) using a simple key like 55.

Convert A (65) and 55 to binary:

65 in binary: 1000001

55 in binary: 110111

We need to make both fit into 8 bits (1 byte), so we add leading 0s to align them:

65 becomes 01000001

55 becomes 00110111

Now, let's XOR the bits:

First bit: 0 XOR 0 = 0

Second bit: 1 XOR 0 = 1

Third bit: 0 XOR 1 = 1

Fourth bit: 0 XOR 1 = 1

Fifth bit: 0 XOR 0 = 0

Sixth bit: 0 XOR 1 = 1

Seventh bit: 0 XOR 1 = 1

Eighth bit: 1 XOR 1 = 0

The result is 01110110, which is 118 in decimal or the letter 'v' in ASCII. So, we have successfully encrypted 'A' to 'v'.

? Let's Implement XOR Encryption in Rust ?

Now, let's apply this in a Rust program. You can install Rust tools from:

<https://www.rust-lang.org/tools/install>

Once installed, create a new project using cargo by typing:

Code cargo new fallen-obfuscate

In the main.rs file, we'll write the following code to perform string obfuscation and function name obfuscation:

1. XOR Function (xor_str):

This function takes two arguments:

og_string: The original string to be obfuscated.

key: The XOR key (an 8-bit unsigned integer).

The function converts the string into its byte form using `.bytes()`. Then, for each byte, it applies the XOR operation with the key (`b ^ key`). The result is a vector of bytes (`Vec<u8>`), which represents the obfuscated string.

2. De-XOR Function (dexor_str):

This function takes the obfuscated string (as bytes) and the same key used for XOR. It reverses the XOR operation ($b \oplus \text{key}$) to get back the original string. The bytes are converted back to characters, and the result is collected into a string.

3. Secret Functions:

`secret_func`: Prints "Secret society is illuminati confirmed!!".
`secret_func_2`: Prints "ssh key for root@hackforums.net".

These functions simulate some sensitive actions or messages that you might want to obfuscate.

4. Function Pointer Alias (FnPtr):

`FnPtr` is defined as a type alias for a function pointer. In Rust, `fn()` is a function that takes no arguments and returns nothing (`()`).

5. Main Function:

The key used for XOR is `0xAA`.

We XOR the string "PM me for zelos wallet keys" by calling `xor_str`, passing the original string and the key.

We then deobfuscate it at runtime using `dextr_str` and print both the XOR'ed and decoded versions.

6. Function Name Obfuscation:

We create a function pointer table (`func_table`) that holds both `secret_func` and `secret_func_2`. We select `i = 1`, which refers to the second function in the table (`secret_func_2`).

Instead of calling `secret_func_2` directly, we call it indirectly through the function pointer, making it harder to track which function is being invoked.

Thank you for reading everyone. I know this is just basics, later on I will cover more advanced obfuscation and how does obfuscation helps us evading AV.

Thank you and have a nice day!

latencyx - October 27, 2024, 7:52 am

very interesting, especially in the Rust language

Infinity. - October 28, 2024, 3:32 am

(Oct 27, 2024 07:52 AM)latencyx Wrote: [\[CITE\]](#) very interesting, especially in the Rust language
Thank you. I am hoping to spread more rust based guides.

PitbullT0r - October 28, 2024, 7:35 am

Thank you brother for this usefull and high quality thread as always!

Hect0r - October 31, 2024, 3:57 am

Hey, very interesting thread, Personally I don't know the Rust language but I find it very interesting, maybe in the future I will start studying it. Thank you for sharing this with the community

Infinity. - October 31, 2024, 3:58 am

(Oct 28, 2024 07:35 AM)PitbullT0r Wrote: [\[CITE\]](#) Thank you brother for this usefull and high quality thread as always!

Thanks mate. I shall see to create more such threads in the near future

Hopsin - November 3, 2024, 3:12 pm

Excellent. Keep up the good work!

koovoo - November 13, 2024, 9:48 pm

Good, im waiting the cpp tut :)

Infinity. - May 23, 2025, 3:53 am

Why am I not able to edit this thread and remove the Fallen material in it?