

Forum:[Hack Forums](#)

Thread: [\[For beginners\] Shellcode Encryption and Why? | Static Scan Evasion](#)

Tentco - December 19, 2023, 3:38 am

Usually while using certain shellcode loader or Cobalt Strike beacon loaders, you might've noticed they are usually encrypted with various Encryptions, for eg - AES, XOR, RC4/5, Base64 etc... question is Why?

The answer is quite simple we encrypt the raw shellcode to evade/bypass shellcode signature.

Note - This will somewhat work on weak memory scanners, since this is only suitable for entropy hiding and static detection evasion, since you will need to decrypt it sometime in memory.

There's not much about this topic, you can use any sort of encryption, recommended to use AES.

XOR - A key is chosen and used to XOR each byte of the shellcode. To decrypt the shellcode, the same key is used to XOR each byte again, which reverses the encryption and recovers the original shellcode, this encryption is not strong.

There're tons of Encrypters available on Github, so I won't discuss about them here. For XOR, <https://github.com/DcardosoGH/ShellCode-Encrypt>

Usage is quite simple generate your shellcode(.bin) -> Encrypt it -> Add the generated key and shellcode encrypted bytes/data into your program, don't forget to use Runtime string obfuscation on your decryption key like [SkCrypt](#) or [XorStr](#).

```
Code char key[] = "Your Key";
char *shellcode_decrypt = new char[shellcodeSize];
int k = 0;
for (int i = 0; i < shellcodeSize; i++)
{
    if (k == sizeof key - 1)
        k = 0;
    shellcode_decrypt[i] = shellcode_encrypt[i] ^ key[k]; // shellcode_encrypt = encrypted raw
bytes.
    k++;
}
```

The decryption routine in this program performs an XOR operation on each byte of the encrypted shellcode using a secret key. The routine uses a for loop with an index variable i to iterate through each byte of the encrypted shellcode. The index variable k is used to access each byte in the key array.

If k reaches the end of the key array (indicated by k == sizeof key - 1), it is reset to 0. This ensures that the process of combining the encrypted shellcode with the key can continue indefinitely. The result of the XOR operation is stored in a new character array shellcode_decrypt, which is a decrypted version of the original encrypted shellcode.

The routine is somewhat same for AES, take encrypted shell code, decrypt It and allocate into a process's virtual address space.

You can find various encryption and loaders template on Github about this too, same goes for RC4 and vice-versa.

Bonus - We can manually add some junk instruction in our shellcode, let's see how.

1. Using [Ndisasm](https://nirx.net/ndisasm/) to get instructions format out of our shellcode alternatively use <https://defuse.ca/online-x86-assembler.htm#disassembly2> for your shellcode bytes.

2. Code cld

```
nop          ; Added a nop instruction
dec eax
nop          ; Added a nop instruction
and esp, byte -0x10
call 0xca
mov eax, eax ; Redundant mov instruction
nop          ; Added a nop instruction
inc ecx
push ecx
inc ecx
push eax
push edx
push ecx
mov eax, eax ; Redundant mov instruction
push esi
nop          ; Added a nop instruction
dec eax
xor edx, edx
gs dec eax
mov edx, [edx+0x60]
dec eax
mov edx, [edx+0x18]
nop          ; Added a nop instruction
dec eax
mov edx, [edx+0x20]
nop          ; Added a nop instruction
dec eax
mov esi, [edx+0x50]
dec eax
movzx ecx, word [edx+0x4a]
; Simple shellcode to open calc.exe, shortened for thread.
```

Adding junk instructions like nop and mov eax, eax that does nothing but server as placeholders are useful as they can avoid sig scan, you can easily write a program that does so, after that re-encrypt your shellcode and use it again.

In the next thread if I've time we will discuss about analyzing some samples using encrypted shellcode.

Again, shellcode encryption is nothing heavy just a light term for basic static scan evasion.

Keep in mind to, Encrypt your decryption key.

You can also decrypt the shellcode for a fraction of time and encrypt it back again.

BiNBo - December 19, 2023, 7:33 am

A wonderful guide and explanation, I would love to see more content like this in this sub-forum! Thank-you!

Nightmare - December 19, 2023, 10:05 am

HQ!! Whenever i see Tentco creating a thread i know it is HQ!!Thank you for this great Info!

Euaggelistes - December 19, 2023, 11:13 am

Thank you so much for providing this information! Please keep up the great work.

osn64698 - December 19, 2023, 9:22 pm

Good tutorial, A few suggestions:

1.

```
Code char key[] = "Your Key";
char *shellcode_decrypt = new char[shellcodeSize];
int k = 0;
for (int i = 0; i < shellcodeSize; i++)
{
    if (k == sizeof key - 1)
        k = 0;
    shellcode_decrypt[i] = shellcode_encrypt[i] ^ key[k]; // shellcode_encrypt = encrypted raw
bytes.
    k++;
}
```

The if logic can probably be replaced by a simple `(++k % sizeof(key) - 1)`

2.

Junk code wise, you can also push registers into the stack, save them, and recover later on, this gives a lot more control then simple "NOP" and "MOV EAX,EAX".

Stanley - December 25, 2023, 10:44 pm

I'm Stanley HF's helper bot. The OP (original poster) of this thread wants everyone to see this page and has elected to bump it to the top of the showthread listing. Since HF doesn't want a ton of LQP (Low Quality Posts) this is the best method of bumping your content. There is a small ?yte fee for each bump. Any OP can bump any of their threads on HF from the linked icon on the top of their first post.

If you haven't already read the first post of this thread we encourage you to do so.

Thanks,
Stanley