Payload Encryption - XOR

Introduction

XOR encryption is the simplest to use and the lightest to implement, making it a popular choice for malware. It is faster than AES and RC4 and does not require any additional libraries or the usage of Windows APIs. Additionally, it is a bidirectional encryption algorithm that allows the same function to be used for both encryption and decryption.

XOR Encryption

The code snippet below shows a basic XOR encryption function. The function simply XORs each byte of the shellcode with a 1-byte key.

```
/*
	- pShellcode : Base address of the payload to encrypt
	- sShellcodeSize : The size of the payload
	- bKey : A single arbitrary byte representing the key for
encrypting the payload
*/
VOID XorByOneKey(IN PBYTE pShellcode, IN SIZE_T sShellcodeSize, IN BYTE
bKey) {
	for (size_t i = 0; i < sShellcodeSize; i++){
		pShellcode[i] = pShellcode[i] ^ bKey;
	}
}
```

Securing The Encryption Key

Some tools and security solutions can brute force the key which will expose the decrypted shellcode. To make the process of guessing the key more difficult for these tools, the code below performs a minor change and increases the keyspace of the key by making `i` a part of the key. With keyspace much larger now, it's more difficult to brute force the key.

```
/*
	- pShellcode : Base address of the payload to encrypt
	- sShellcodeSize : The size of the payload
	- bKey : A single arbitrary byte representing the key for
encrypting the payload
*/
VOID XorByiKeys(IN PBYTE pShellcode, IN SIZE_T sShellcodeSize, IN BYTE
bKey) {
```

```
        for (size_t i = 0; i < sShellcodeSize; i++) {
                pShellcode[i] = pShellcode[i] ^ (bKey + i);
        }
}
```

The code snippet above can still be hardened further. The snippet below performs the encryption process with a key, using every byte of the key repeatedly making it harder to crack the key.

```
/*
        - pShellcode : Base address of the payload to encrypt
        - sShellcodeSize : The size of the payload
        - bKey : A random array of bytes of specific size
        - sKeySize : The size of the key
*/
VOID XorByInputKey(IN PBYTE pShellcode, IN SIZE_T sShellcodeSize, IN PBYTE
bKey, IN SIZE_T sKeySize) {
        for (size_t i = 0, j = 0; i < sShellcodeSize; i++, j++) {
                if (j >= sKeySize){
                        j = 0;
                }
                pShellcode[i] = pShellcode[i] ^ bKey[j];
        }
}
```

Conclusion

It is recommended to utilize XOR encryption for small tasks, such as obscuring strings. However, for larger payloads, it is advised to use more secure encryption methods such as AES.