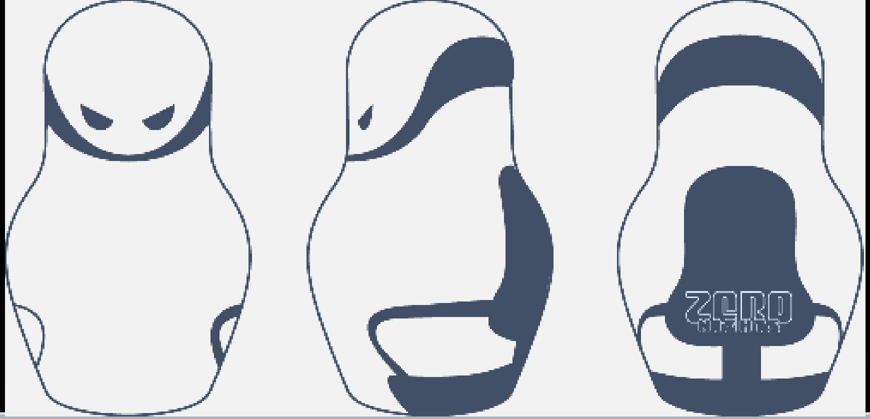


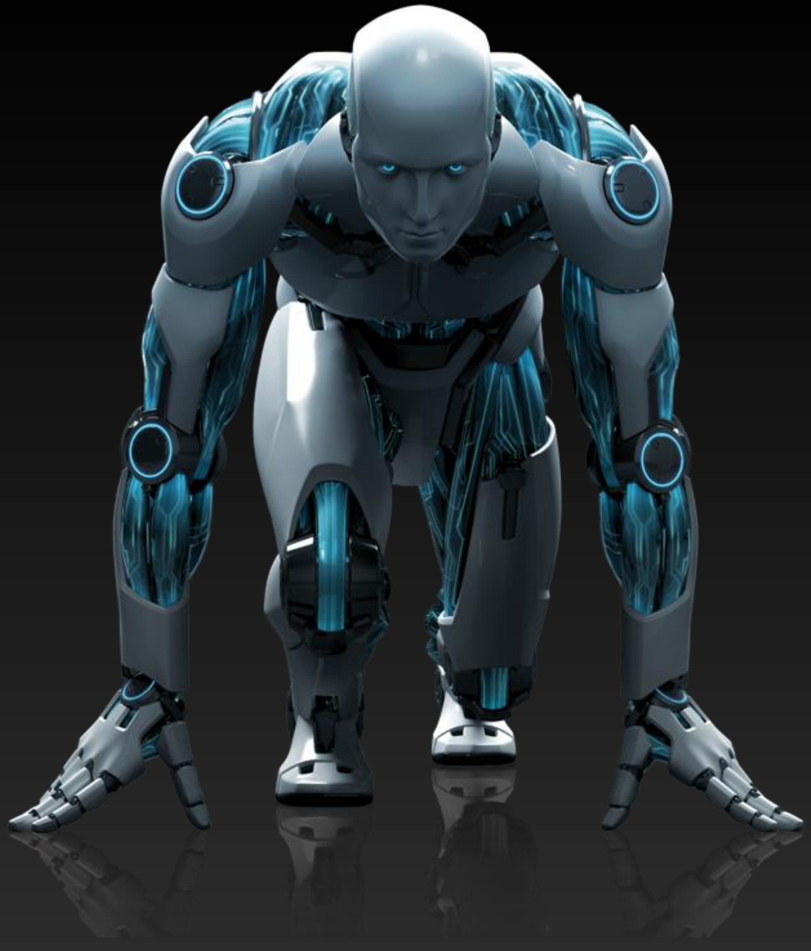
Win32/Flamer: Reverse Engineering and Framework Reconstruction

Aleksandr Matrosov

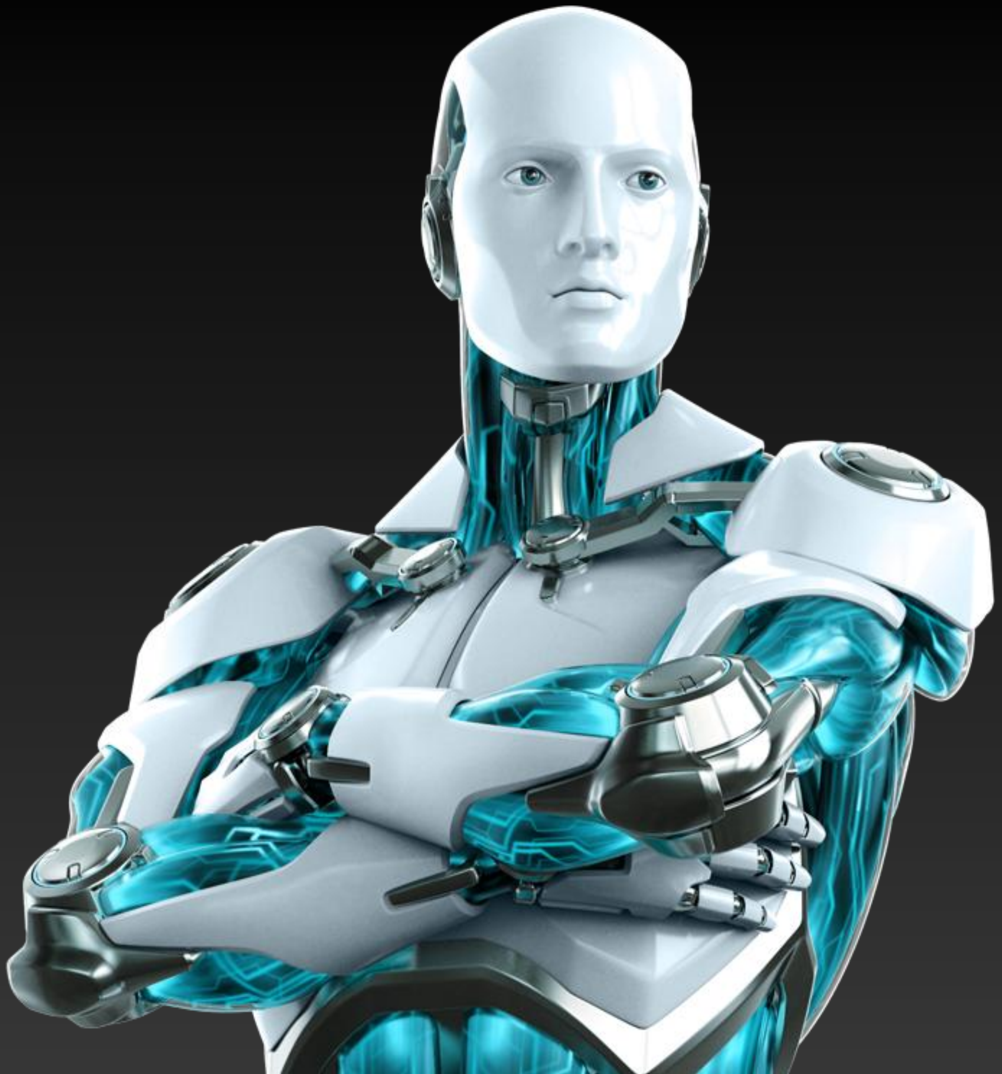
Eugene Rodionov

Outline of The Presentation

- Typical malware vs. Stuxnet/Flame
  - ✓ What the difference?
- Flamer code reconstruction problems
  - ✓ C++ code reconstruction
  - ✓ Library code identification
- Flamer framework overview
- Object oriented code reconstruction
- Relationship Stuxnet/Duqu/Flamer



Typical Malware vs. Stuxnet/Flamer



What’s the Difference?

VS.

What’s the Difference?

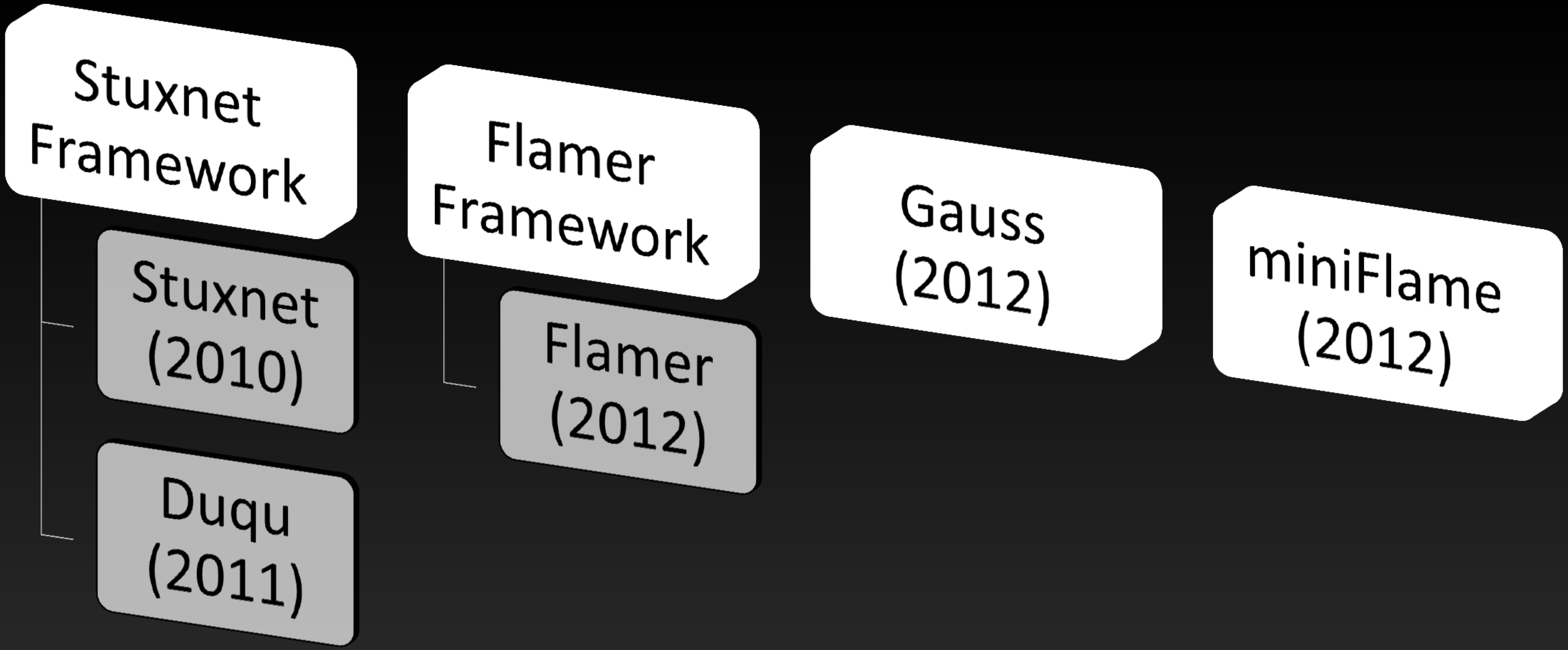
➢ Typical malware

- ✓ Different motivation, budget …
- ✓ Use 1-days for distribution
- ✓ Anti-stealth for bypassing AV
- ✓ Stealth timing: months
- ✓ Developed in C or C++ in C style
- ✓ Simple architecture for plugins
- ✓ Traditional ways for obfuscation:
    - packers
    - polymorphic code
    - vm-based protection
    - …

➢ Stuxnet/Flame …

- ✓ Different motivation, budget …
- ✓ Use 0-days for distribution
- ✓ Anti-stealth for bypassing all sec soft
- ✓ Stealth timing: years
- ✓ Tons of C++ code with OOP
- ✓ Industrial OO framework platform
- ✓ Other ways of code obfuscation:
    - tons of embedded static code
    - specific compilers/options
    - object oriented wrappers for typical OS utilities

Stuxnet/Duqu/Flamer/Gauss Appearance



Stuxnet Framework

Stuxnet (2010)

Duqu (2011)

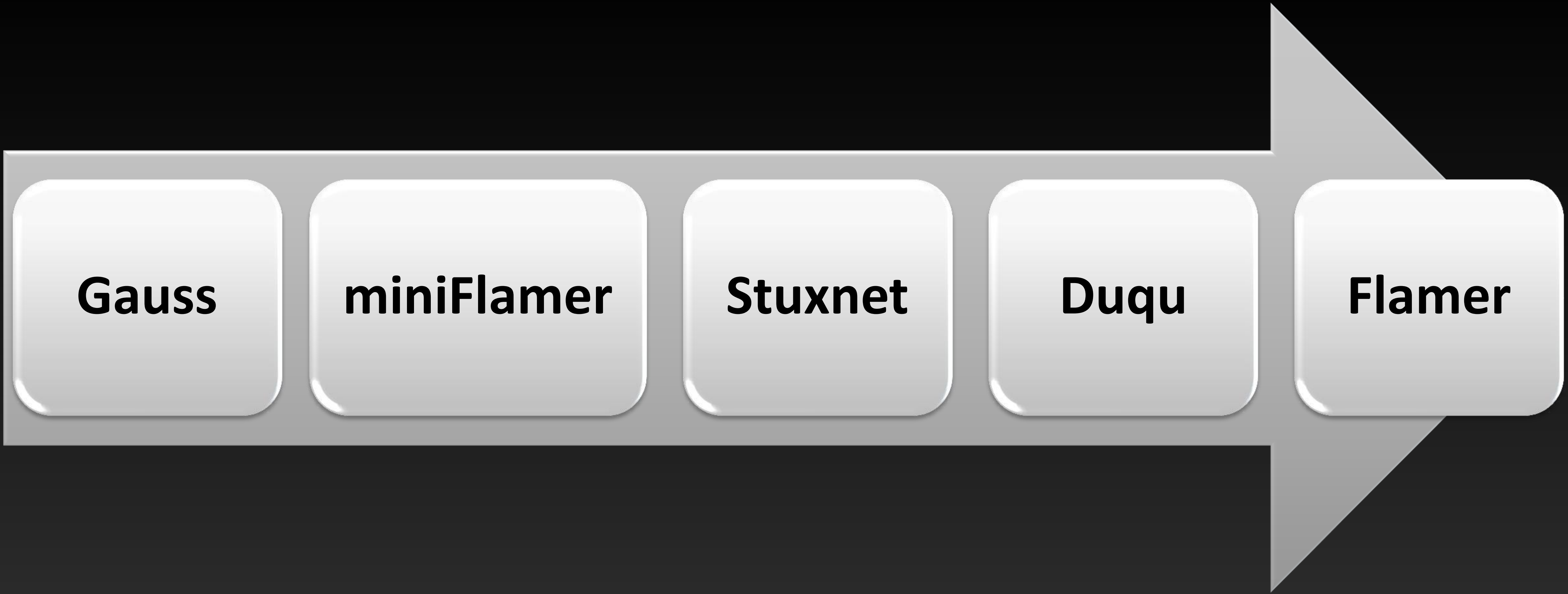
Flamer Framework

Flamer (2012)

Gauss (2012)

miniFlame (2012)

Code Complexity Growth



Gauss

miniFlamer

Stuxnet

Duqu

Flamer

Code Complexity Growth

C++ Code REconstruction Problems

C++ Code Reconstruction Problems

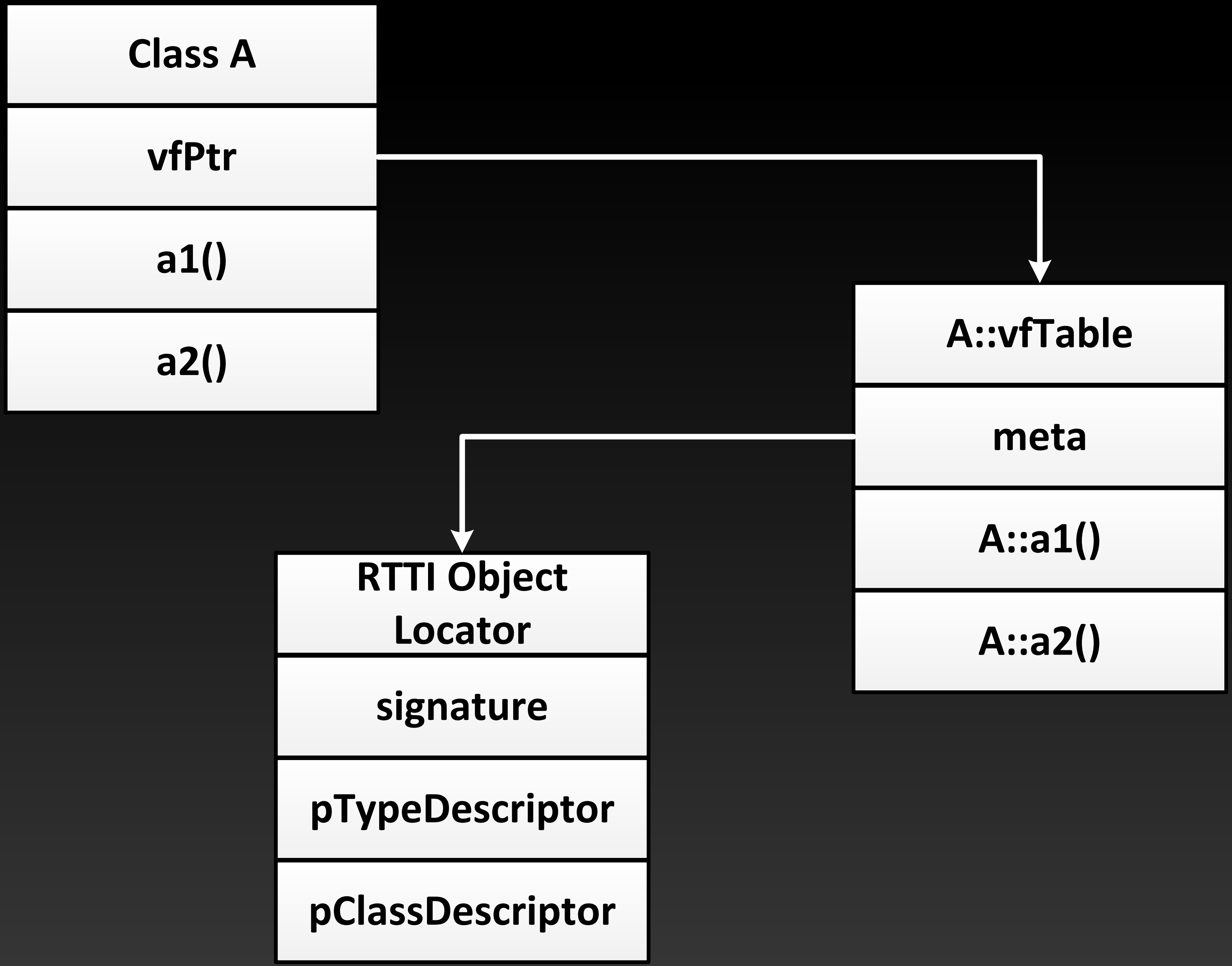
- Object identification
  - ✓ Type reconstruction

- Class layout reconstruction
  - ✓ Identify constructors/destructors
  - ✓ Identify class members
  - ✓ Local/global type reconstruction
  - ✓ Associate object with exact method calls

- RTTI reconstruction
  - ✓ Vftable reconstruction
  - ✓ Associate vftable object with exact object
  - ✓ Class hierarchy reconstruction

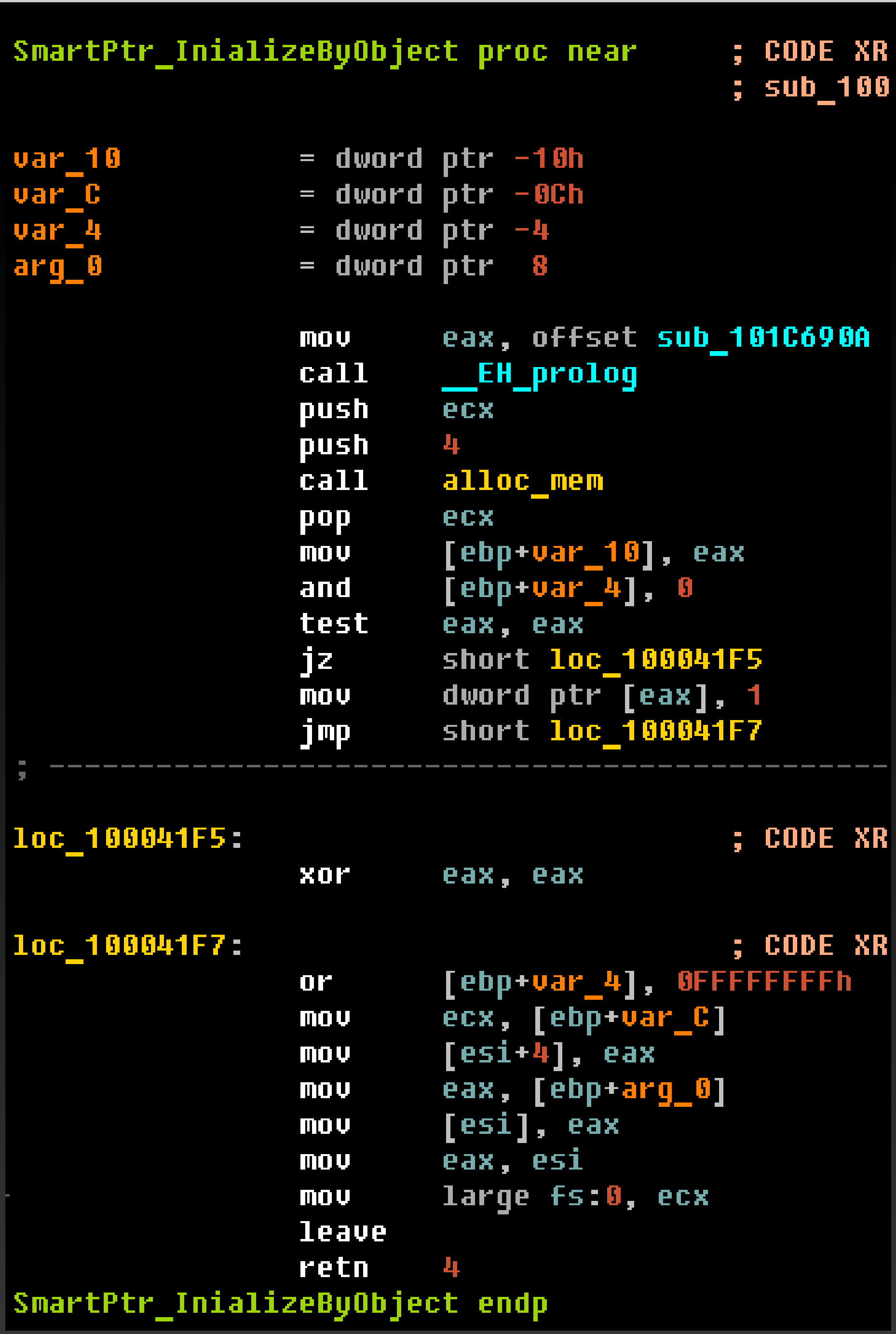
Depended by compiler design

C++ Code Reconstruction Problems

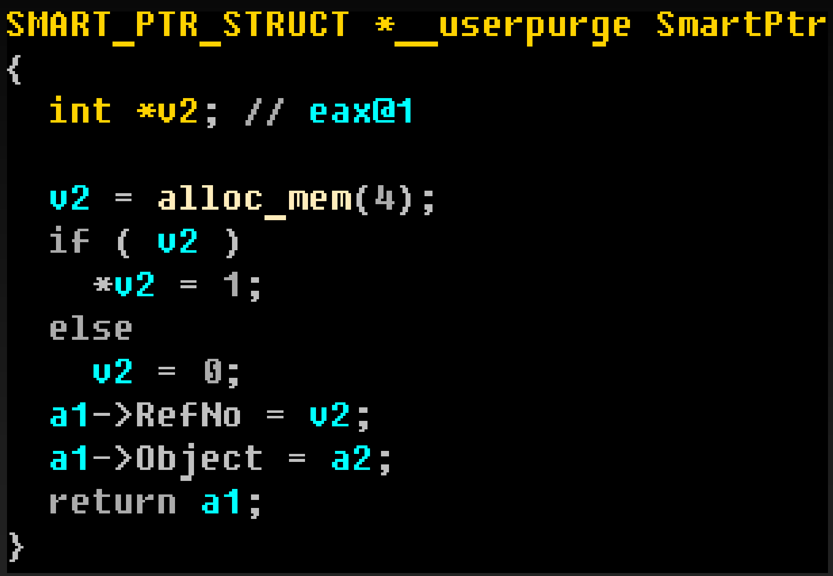


C++ Code Reconstruction Problems

Identify Smart Pointer Structure

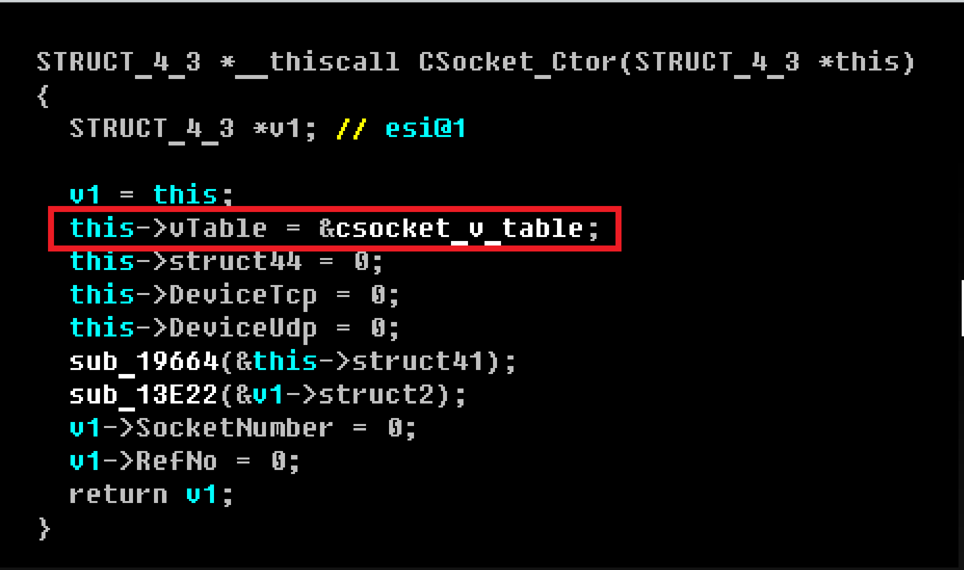


```
SmartPtr_InializeByObject proc near     ; CODE XR
                                        ; sub_100

var_10          = dword ptr -10h
var_C           = dword ptr -0Ch
var_4           = dword ptr -4
arg_0           = dword ptr  8

                mov     eax, offset sub_101C690A
                call    __EH_prolog
                push    ecx
                push    4
                call    alloc_mem
                pop     ecx
                mov     [ebp+var_10], eax
                and     [ebp+var_4], 0
                test    eax, eax
                jz      short loc_100041F5
                mov     dword ptr [eax], 1
                jmp     short loc_100041F7
; ---------------------------------------------

loc_100041F5:                           ; CODE XR
                xor     eax, eax

loc_100041F7:                           ; CODE XR
                or      [ebp+var_4], 0FFFFFFFFh
                mov     ecx, [ebp+var_C]
                mov     [esi+4], eax
                mov     eax, [ebp+arg_0]
                mov     [esi], eax
                mov     eax, esi
                mov     large fs:0, ecx
                leave
                retn    4
SmartPtr_InializeByObject endp
```

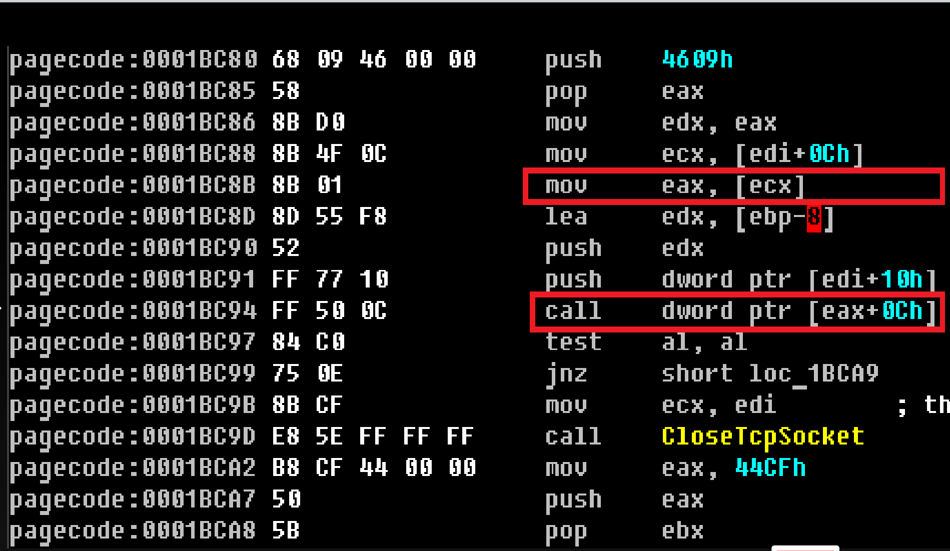


```
SMART_PTR_STRUCT *__userpurge SmartPtr
{
  int *v2; // eax@1

  v2 = alloc_mem(4);
  if ( v2 )
    *v2 = 1;
  else
    v2 = 0;
  a1->RefNo = v2;
  a1->Object = a2;
  return a1;
}
```

Identify Exact Virtual Function Call in vtable



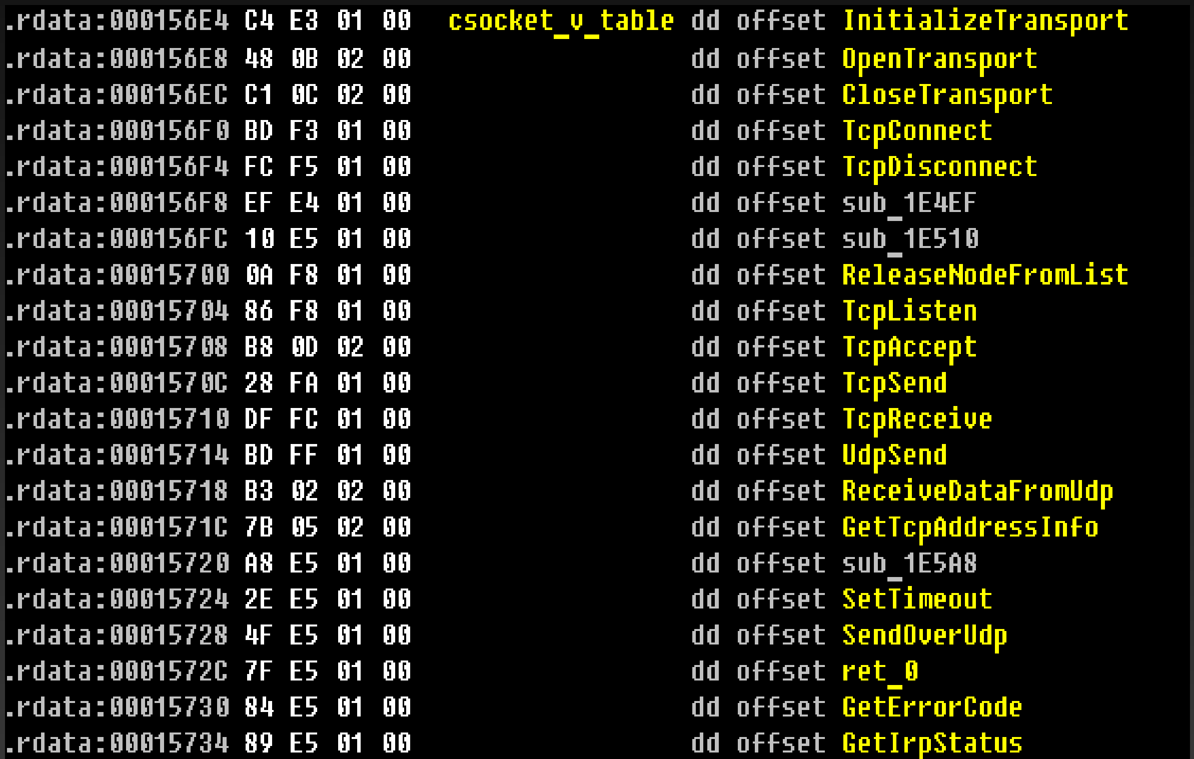
```
STRUCT_4_3 *__thiscall CSocket_Ctor(STRUCT_4_3 *this)
{
  STRUCT_4_3 *v1; // esi@1

  v1 = this;
  this->vTable = &csocket_v_table;
  this->struct44 = 0;
  this->DeviceTcp = 0;
  this->DeviceUdp = 0;
  sub_19664(&this->struct41);
  sub_13E22(&v1->struct2);
  v1->SocketNumber = 0;
  v1->RefNo = 0;
  return v1;
}
```



```
pagecode:0001BC80 68 09 46 00 00      push    4609h
pagecode:0001BC85 58                  pop     eax
pagecode:0001BC86 8B D0               mov     edx, eax
pagecode:0001BC88 8B 4F 0C            mov     ecx, [edi+0Ch]
pagecode:0001BC8B 8B 01               mov     eax, [ecx]
pagecode:0001BC8D 8D 55 F8            lea     edx, [ebp-8]
pagecode:0001BC90 52                  push    edx
pagecode:0001BC91 FF 77 10            push    dword ptr [edi+10h]
pagecode:0001BC94 FF 50 0C            call    dword ptr [eax+0Ch]
pagecode:0001BC97 84 C0               test    al, al
pagecode:0001BC99 75 0E               jnz     short loc_1BCA9
pagecode:0001BC9B 8B CF               mov     ecx, edi           ; this
pagecode:0001BC9D E8 5E FF FF FF      call    CloseTcpSocket
pagecode:0001BCA2 B8 CF 44 00 00      mov     eax, 44CFh
pagecode:0001BCA7 50                  push    eax
pagecode:0001BCA8 5B                  pop     ebx
```

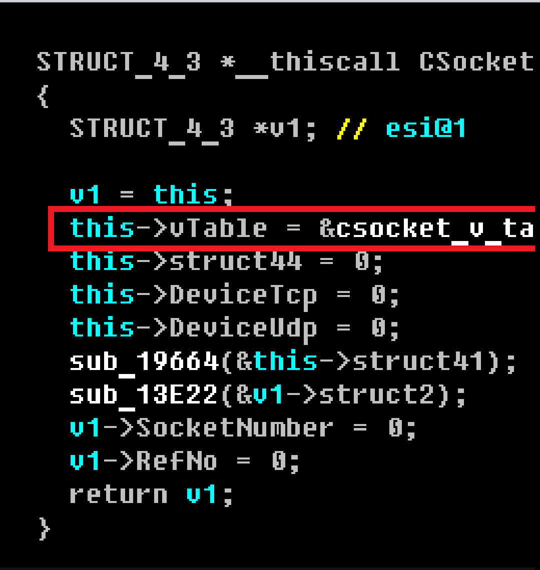
Load pointer to table of virtual methods

Call virtual method

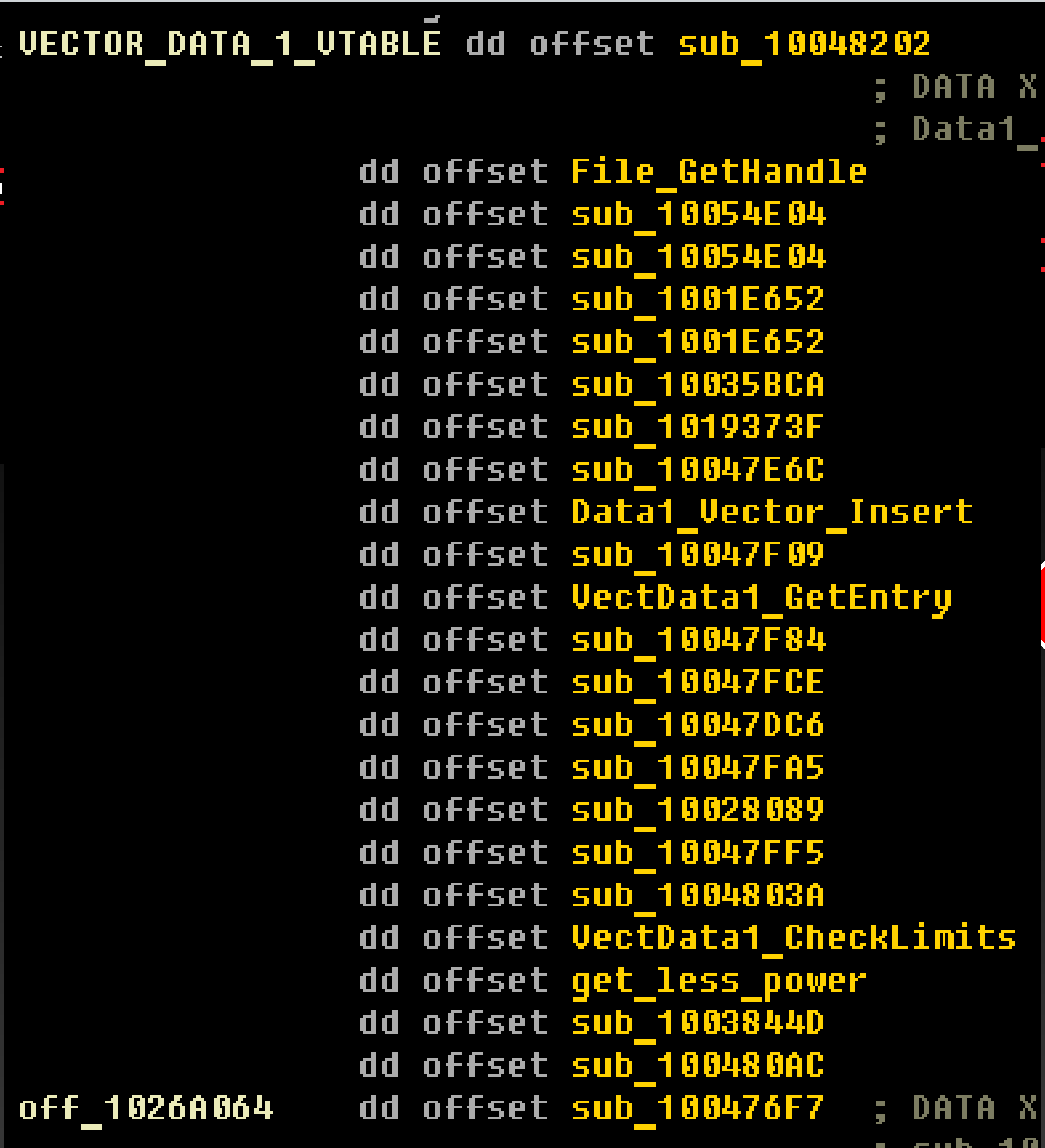


```
.rdata:000156E4 C4 E3 01 00   csocket_v_table dd offset InitializeTransport
.rdata:000156E8 48 0B 02 00                   dd offset OpenTransport
.rdata:000156EC C1 0C 02 00                   dd offset CloseTransport
.rdata:000156F0 BD F3 01 00                   dd offset TcpConnect
.rdata:000156F4 FC F5 01 00                   dd offset TcpDisconnect
.rdata:000156F8 EF E4 01 00                   dd offset sub_1E4EF
.rdata:000156FC 10 E5 01 00                   dd offset sub_1E510
.rdata:00015700 0A F8 01 00                   dd offset ReleaseNodeFromList
.rdata:00015704 86 F8 01 00                   dd offset TcpListen
.rdata:00015708 B8 0D 02 00                   dd offset TcpAccept
.rdata:0001570C 28 FA 01 00                   dd offset TcpSend
.rdata:00015710 DF FC 01 00                   dd offset TcpReceive
.rdata:00015714 BD FF 01 00                   dd offset UdpSend
.rdata:00015718 B3 02 02 00                   dd offset ReceiveDataFromUdp
.rdata:0001571C 7B 05 02 00                   dd offset GetTcpAddressInfo
.rdata:00015720 A8 E5 01 00                   dd offset sub_1E5A8
.rdata:00015724 2E E5 01 00                   dd offset SetTimeout
.rdata:00015728 4F E5 01 00                   dd offset SendOverUdp
.rdata:0001572C 7F E5 01 00                   dd offset ret_0
.rdata:00015730 84 E5 01 00                   dd offset GetErrorCode
.rdata:00015734 89 E5 01 00                   dd offset GetIrpStatus
```

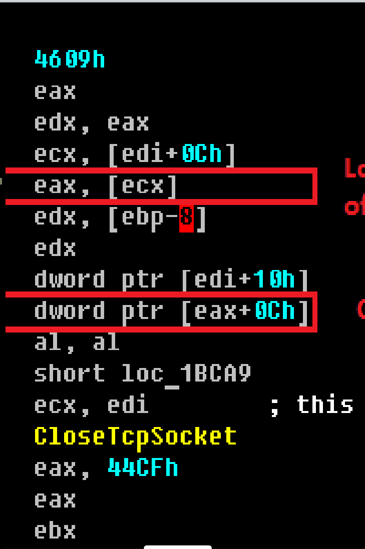
Identify Exact Virtual Function Call in vtable



```
STRUCT_4_3 *__thiscall CSocket
{
  STRUCT_4_3 *v1; // esi@1

  v1 = this;
  this->vTable = &csocket_v_ta
  this->struct44 = 0;
  this->DeviceTcp = 0;
  this->DeviceUdp = 0;
  sub_19664(&this->struct41);
  sub_13E22(&v1->struct2);
  v1->SocketNumber = 0;
  v1->RefNo = 0;
  return v1;
}
```



```
VECTOR_DATA_1_VTABLE dd offset sub_10048202
                                                ; DATA X
                                                ; Data1_
                     dd offset File_GetHandle
                     dd offset sub_10054E04
                     dd offset sub_10054E04
                     dd offset sub_1001E652
                     dd offset sub_1001E652
                     dd offset sub_10035BCA
                     dd offset sub_1019373F
                     dd offset sub_10047E6C
                     dd offset Data1_Vector_Insert
                     dd offset sub_10047F09
                     dd offset VectData1_GetEntry
                     dd offset sub_10047F84
                     dd offset sub_10047FCE
                     dd offset sub_10047DC6
                     dd offset sub_10047FA5
                     dd offset sub_10028089
                     dd offset sub_10047FF5
                     dd offset sub_1004803A
                     dd offset VectData1_CheckLimits
                     dd offset get_less_power
                     dd offset sub_1003844D
                     dd offset sub_100480AC
off_1026A064         dd offset sub_100476F7     ; DATA X
```

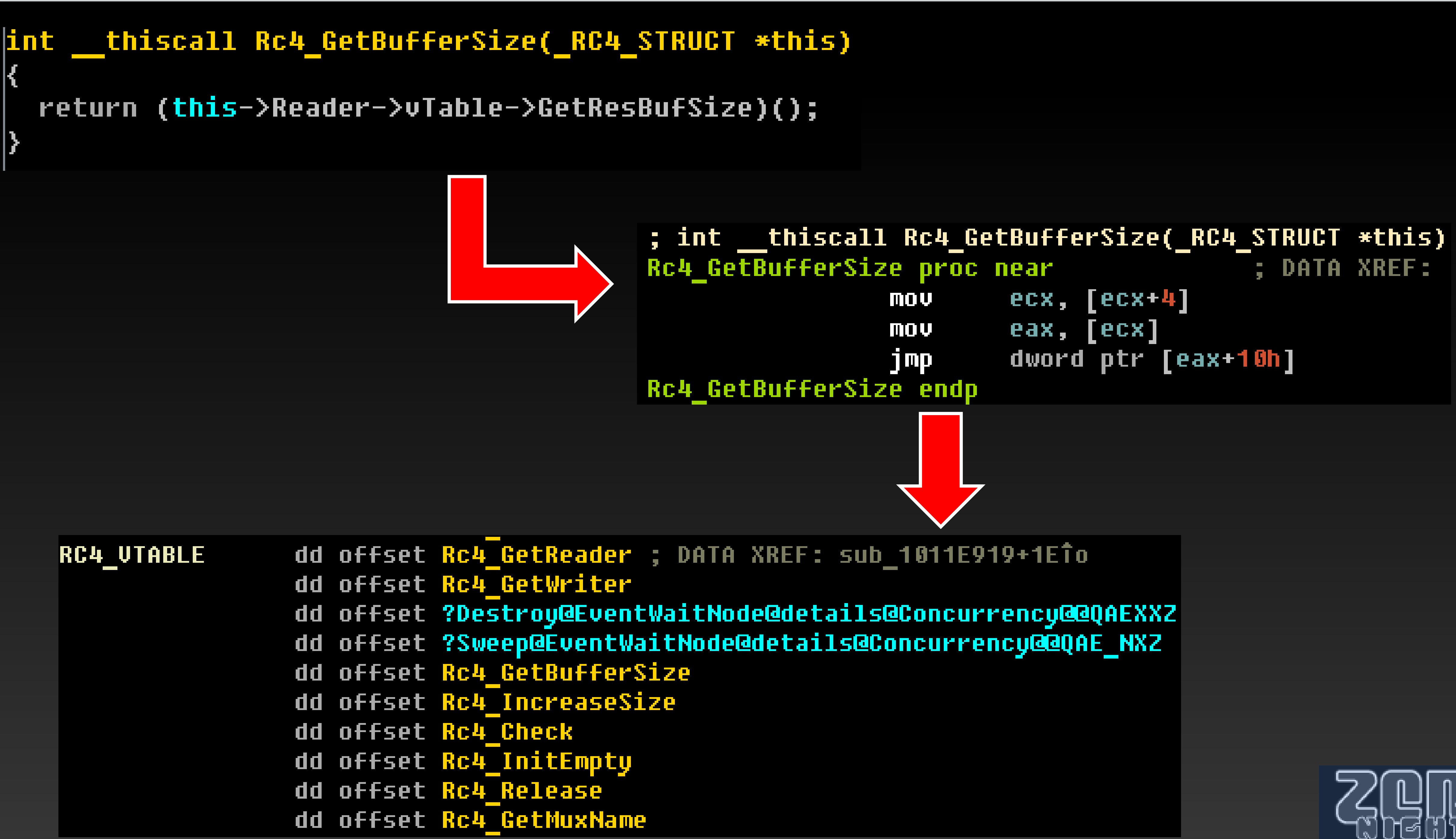


```
4609h
eax
edx, eax
ecx, [edi+0Ch]
eax, [ecx]
edx, [ebp-8]
edx
dword ptr [edi+10h]
dword ptr [eax+0Ch]
al, al
short loc_1BCA9
ecx, edi          ; this
CloseTcpSocket
eax, 44CFh
eax
ebx
```

Load pointer to table of virtual methods

Call virtual method

Identify Exact Virtual Function Call in vtable

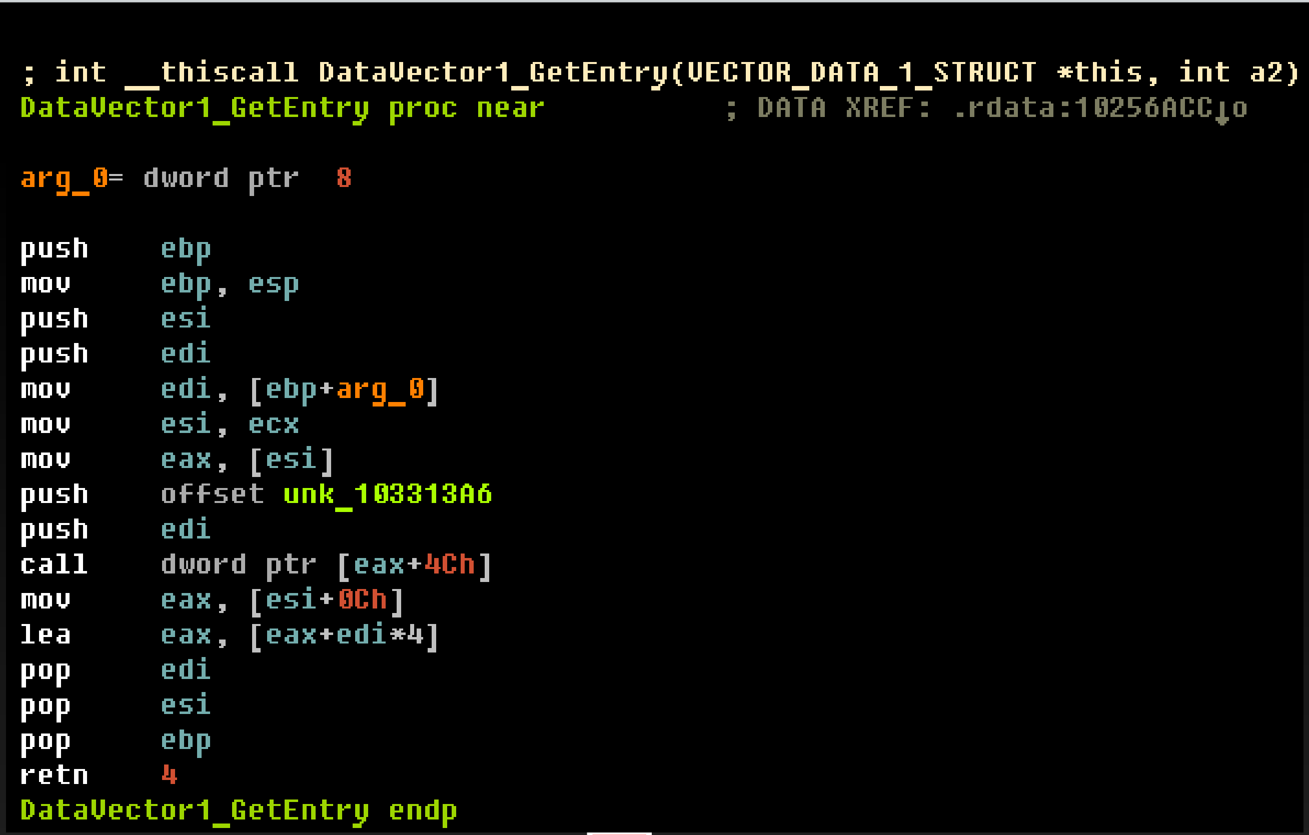


```
int __thiscall Rc4_GetBufferSize(_RC4_STRUCT *this)
{
  return (this->Reader->vTable->GetResBufSize)();
}
```

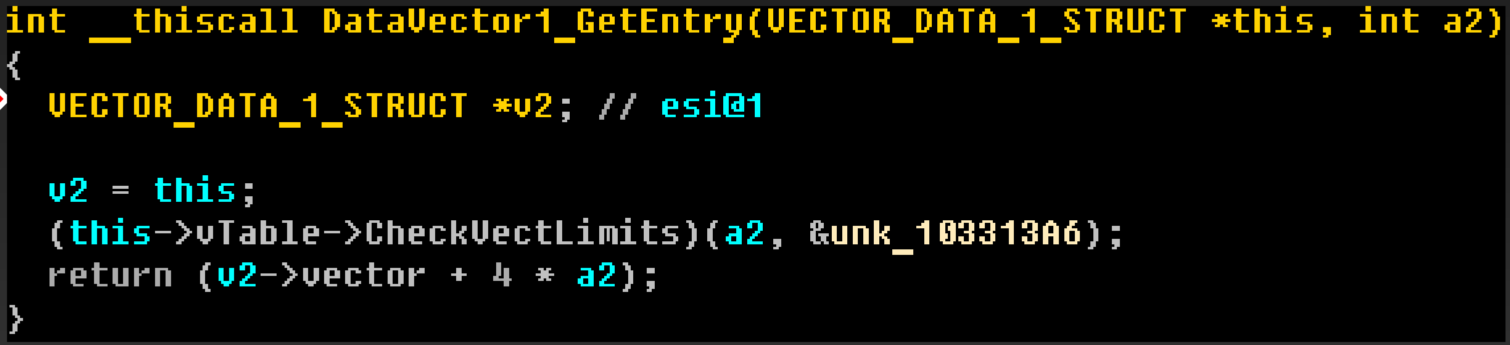
```
; int __thiscall Rc4_GetBufferSize(_RC4_STRUCT *this)
Rc4_GetBufferSize proc near                     ; DATA XREF:
                mov     ecx, [ecx+4]
                mov     eax, [ecx]
                jmp     dword ptr [eax+10h]
Rc4_GetBufferSize endp
```

```
RC4_VTABLE      dd offset Rc4_GetReader ; DATA XREF: sub_1011E919+1E↑o
                dd offset Rc4_GetWriter
                dd offset ?Destroy@EventWaitNode@details@Concurrency@@QAEXXZ
                dd offset ?Sweep@EventWaitNode@details@Concurrency@@QAE_NXZ
                dd offset Rc4_GetBufferSize
                dd offset Rc4_IncreaseSize
                dd offset Rc4_Check
                dd offset Rc4_InitEmpty
                dd offset Rc4_Release
                dd offset Rc4_GetMuxName
```

Identify Custom Type Operations

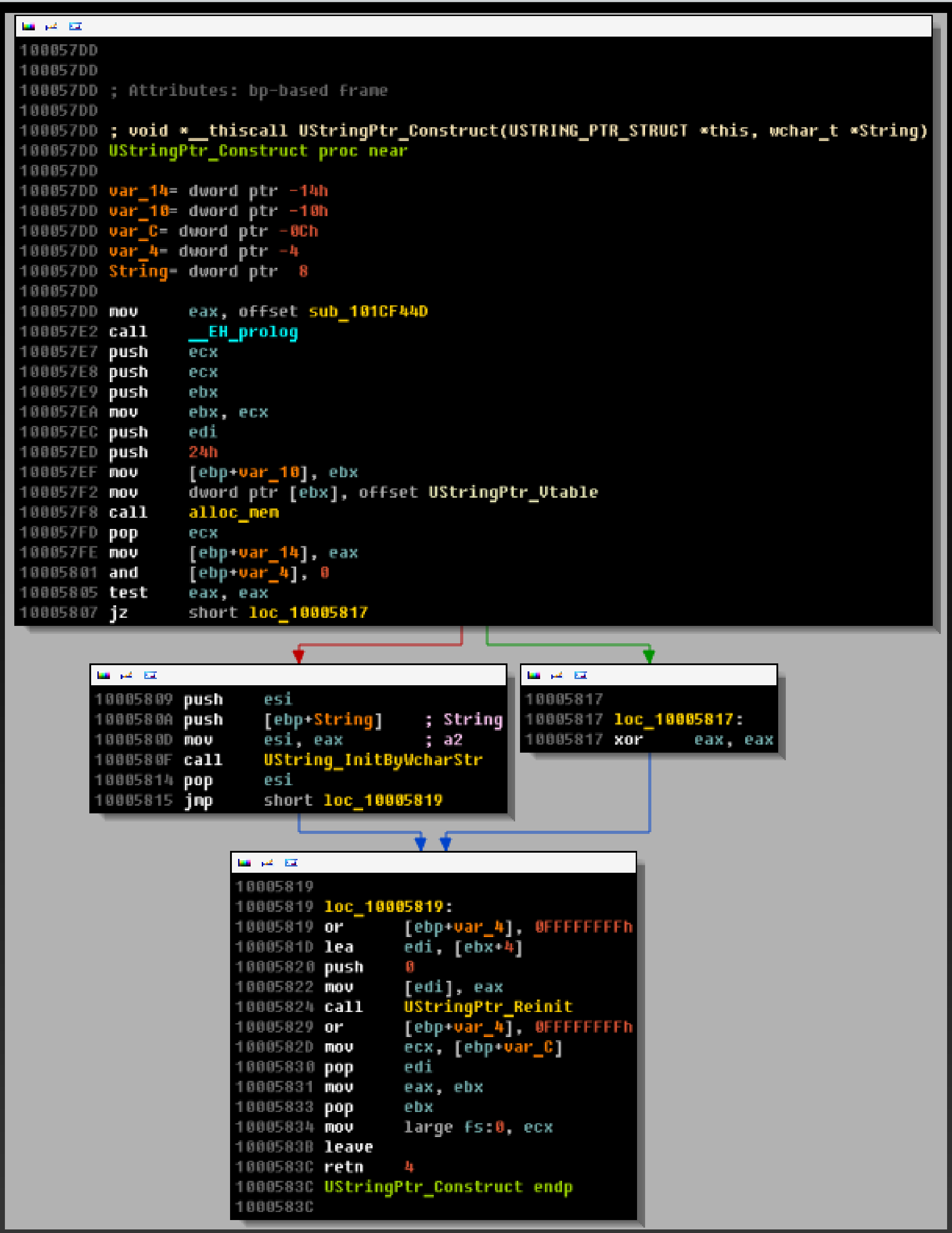


```
; int __thiscall DataVector1_GetEntry(VECTOR_DATA_1_STRUCT *this, int a2)
DataVector1_GetEntry proc near          ; DATA XREF: .rdata:10256ACC↓o

arg_0= dword ptr  8

push    ebp
mov     ebp, esp
push    esi
push    edi
mov     edi, [ebp+arg_0]
mov     esi, ecx
mov     eax, [esi]
push    offset unk_103313A6
push    edi
call    dword ptr [eax+4Ch]
mov     eax, [esi+0Ch]
lea     eax, [eax+edi*4]
pop     edi
pop     esi
pop     ebp
retn    4
DataVector1_GetEntry endp
```



```
int __thiscall DataVector1_GetEntry(VECTOR_DATA_1_STRUCT *this, int a2)
{
  VECTOR_DATA_1_STRUCT *v2; // esi@1

  v2 = this;
  (this->vTable->CheckVectLimits)(a2, &unk_103313A6);
  return (v2->vector + 4 * a2);
}
```

Identify Objects Constructors



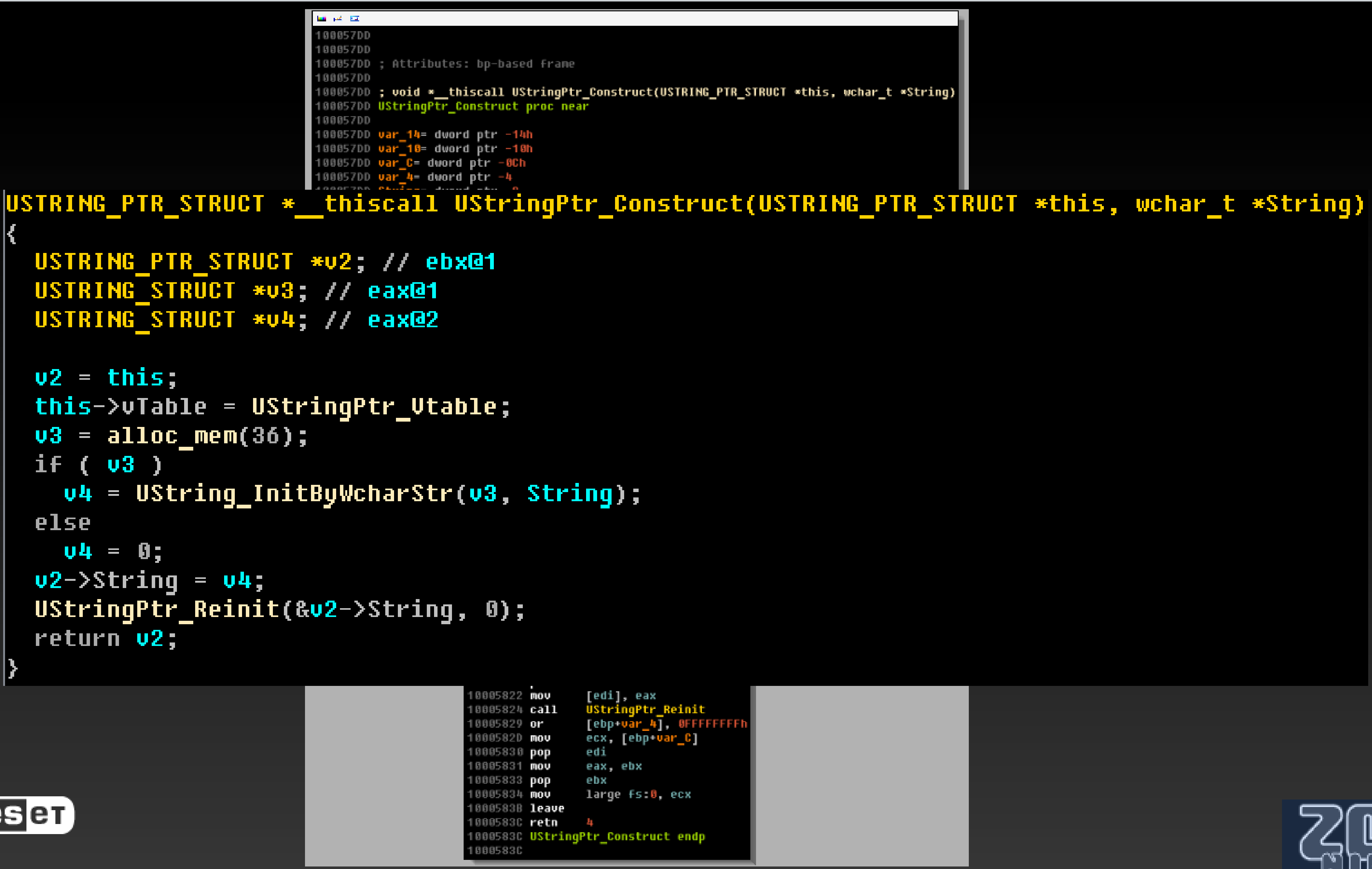
```
100057DD
100057DD
100057DD ; Attributes: bp-based frame
100057DD
100057DD ; void *__thiscall UStringPtr_Construct(USTRING_PTR_STRUCT *this, wchar_t *String)
100057DD UStringPtr_Construct proc near
100057DD
100057DD var_14= dword ptr -14h
100057DD var_10= dword ptr -10h
100057DD var_C= dword ptr -0Ch
100057DD var_4= dword ptr -4
100057DD String= dword ptr  8
100057DD
100057DD mov     eax, offset sub_101CF44D
100057E2 call    __EH_prolog
100057E7 push    ecx
100057E8 push    ecx
100057E9 push    ebx
100057EA mov     ebx, ecx
100057EC push    edi
100057ED push    24h
100057EF mov     [ebp+var_10], ebx
100057F2 mov     dword ptr [ebx], offset UStringPtr_Vtable
100057F8 call    alloc_mem
100057FD pop     ecx
100057FE mov     [ebp+var_14], eax
10005801 and     [ebp+var_4], 0
10005805 test    eax, eax
10005807 jz      short loc_10005817
```

```
10005809 push    esi
1000580A push    [ebp+String]    ; String
1000580D mov     esi, eax        ; a2
1000580F call    UString_InitByWcharStr
10005814 pop     esi
10005815 jmp     short loc_10005819
```

```
10005817
10005817 loc_10005817:
10005817 xor     eax, eax
```

```
10005819
10005819 loc_10005819:
10005819 or      [ebp+var_4], 0FFFFFFFFh
1000581D lea     edi, [ebx+4]
10005820 push    0
10005822 mov     [edi], eax
10005824 call    UStringPtr_Reinit
10005829 or      [ebp+var_4], 0FFFFFFFFh
1000582D mov     ecx, [ebp+var_C]
10005830 pop     edi
10005831 mov     eax, ebx
10005833 pop     ebx
10005834 mov     large fs:0, ecx
1000583B leave
1000583C retn    4
1000583C UStringPtr_Construct endp
1000583C
```

Identify Objects Constructors



```
100057DD
100057DD
100057DD ; Attributes: bp-based frame
100057DD
100057DD ; void *__thiscall UStringPtr_Construct(USTRING_PTR_STRUCT *this, wchar_t *String)
100057DD UStringPtr_Construct proc near
100057DD
100057DD var_14= dword ptr -14h
100057DD var_10= dword ptr -10h
100057DD var_C= dword ptr -0Ch
100057DD var_4= dword ptr -4
```

```
USTRING_PTR_STRUCT *__thiscall UStringPtr_Construct(USTRING_PTR_STRUCT *this, wchar_t *String)
{
  USTRING_PTR_STRUCT *v2; // ebx@1
  USTRING_STRUCT *v3; // eax@1
  USTRING_STRUCT *v4; // eax@2

  v2 = this;
  this->vTable = UStringPtr_Vtable;
  v3 = alloc_mem(36);
  if ( v3 )
    v4 = UString_InitByWcharStr(v3, String);
  else
    v4 = 0;
  v2->String = v4;
  UStringPtr_Reinit(&v2->String, 0);
  return v2;
}
```

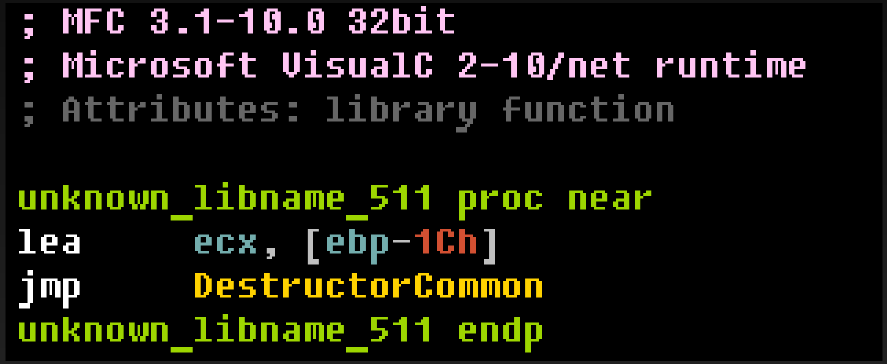
```
10005822 mov     [edi], eax
10005824 call    UStringPtr_Reinit
10005829 or      [ebp+var_4], 0FFFFFFFFh
1000582D mov     ecx, [ebp+var_C]
10005830 pop     edi
10005831 mov     eax, ebx
10005833 pop     ebx
10005834 mov     large fs:0, ecx
1000583B leave
1000583C retn    4
1000583C UStringPtr_Construct endp
1000583C
```

Library code identification problems

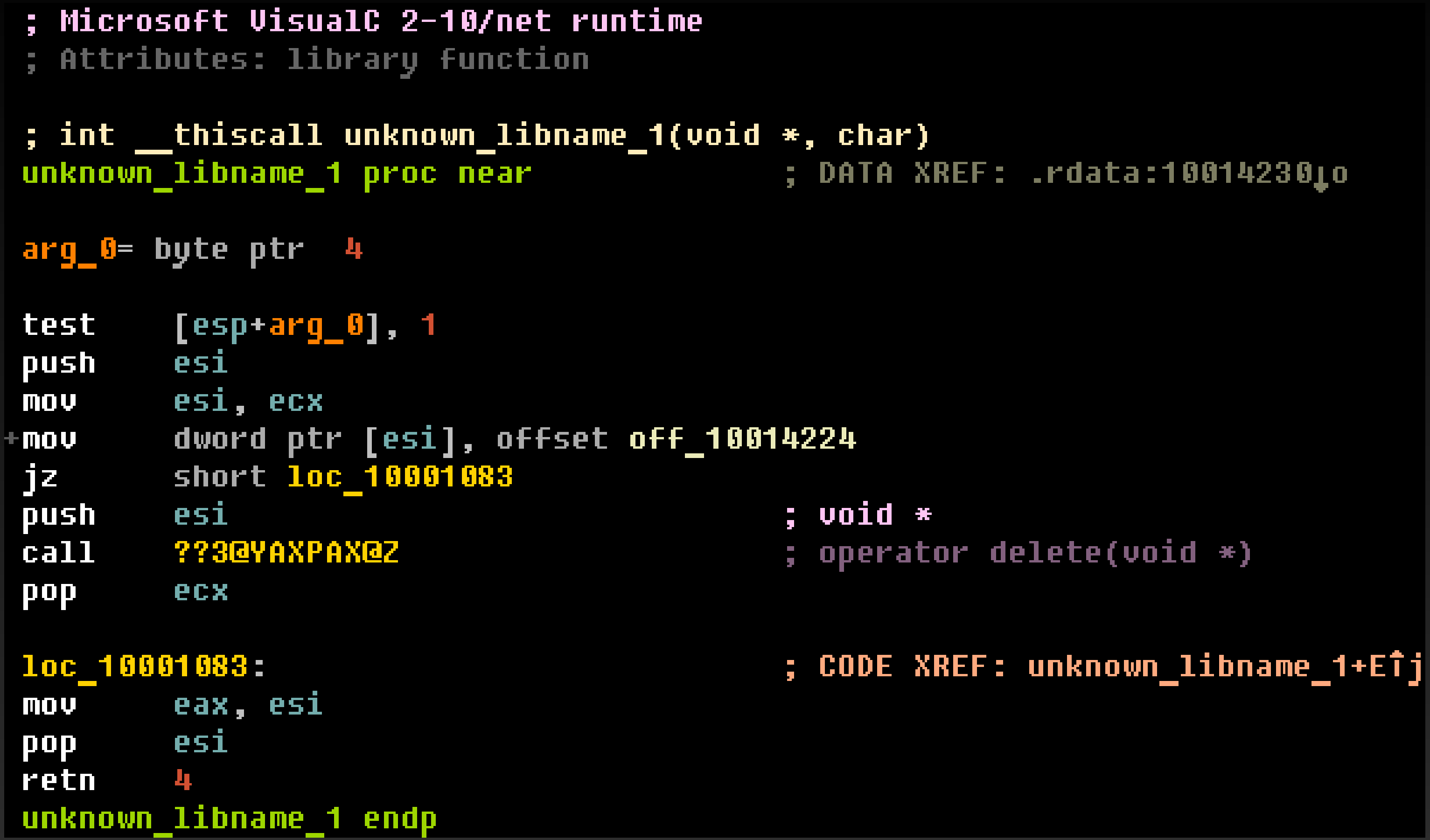
Library Code Identification Problems

- Compiler optimization
- Wrappers for WinAPI calls
- Embedded library code
  - ✓ Library version identification problem
- IDA signatures used syntax based detection methods
  - ✓ Recompiled libraries problem
  - ✓ Compiler optimization problem

Library Code Identification Problems

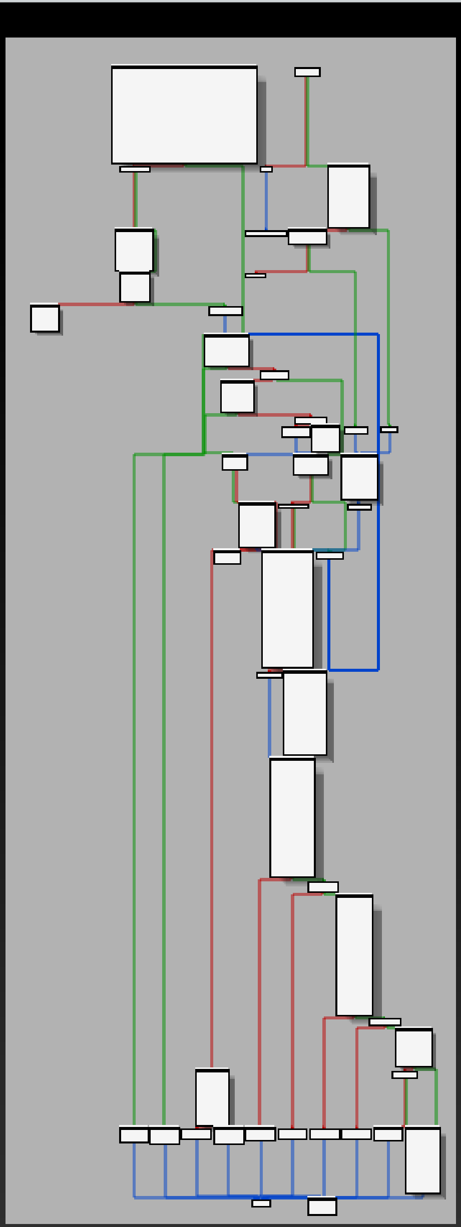


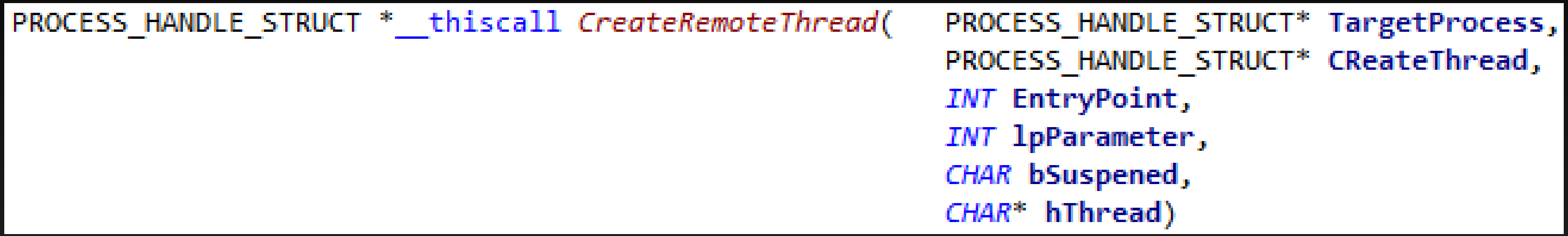
```
; MFC 3.1-10.0 32bit
; Microsoft VisualC 2-10/net runtime
; Attributes: library function

unknown_libname_511 proc near
lea     ecx, [ebp-1Ch]
jmp     DestructorCommon
unknown_libname_511 endp
```



```
; Microsoft VisualC 2-10/net runtime
; Attributes: library function

; int __thiscall unknown_libname_1(void *, char)
unknown_libname_1 proc near             ; DATA XREF: .rdata:10014230↓o

arg_0= byte ptr  4

test    [esp+arg_0], 1
push    esi
mov     esi, ecx
mov     dword ptr [esi], offset off_10014224
jz      short loc_10001083
push    esi                             ; void *
call    ??3@YAXPAX@Z                    ; operator delete(void *)
pop     ecx

loc_10001083:                           ; CODE XREF: unknown_libname_1+E↑j
mov     eax, esi
pop     esi
retn    4
unknown_libname_1 endp
```

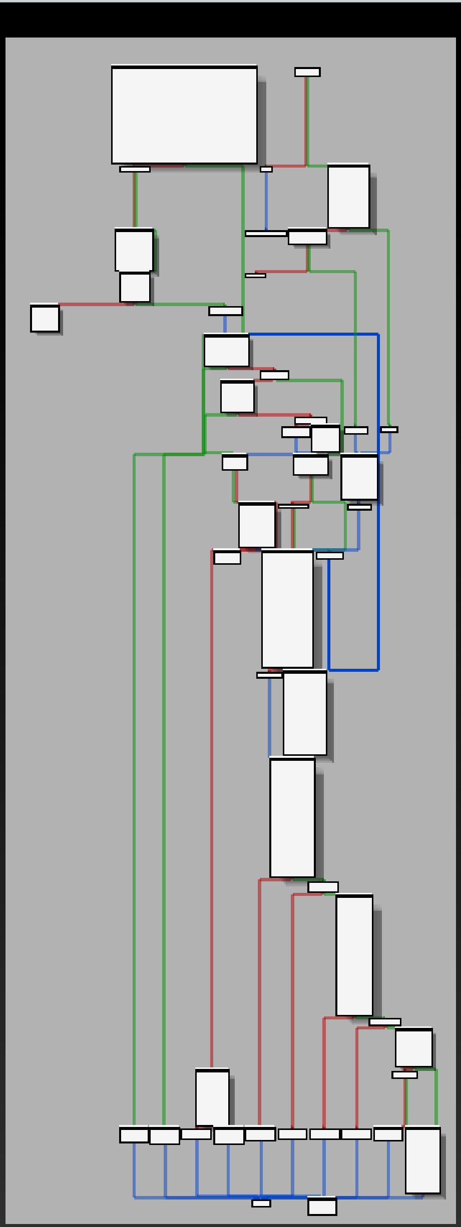
Object Oriented API Wrappers and Implicit Calls



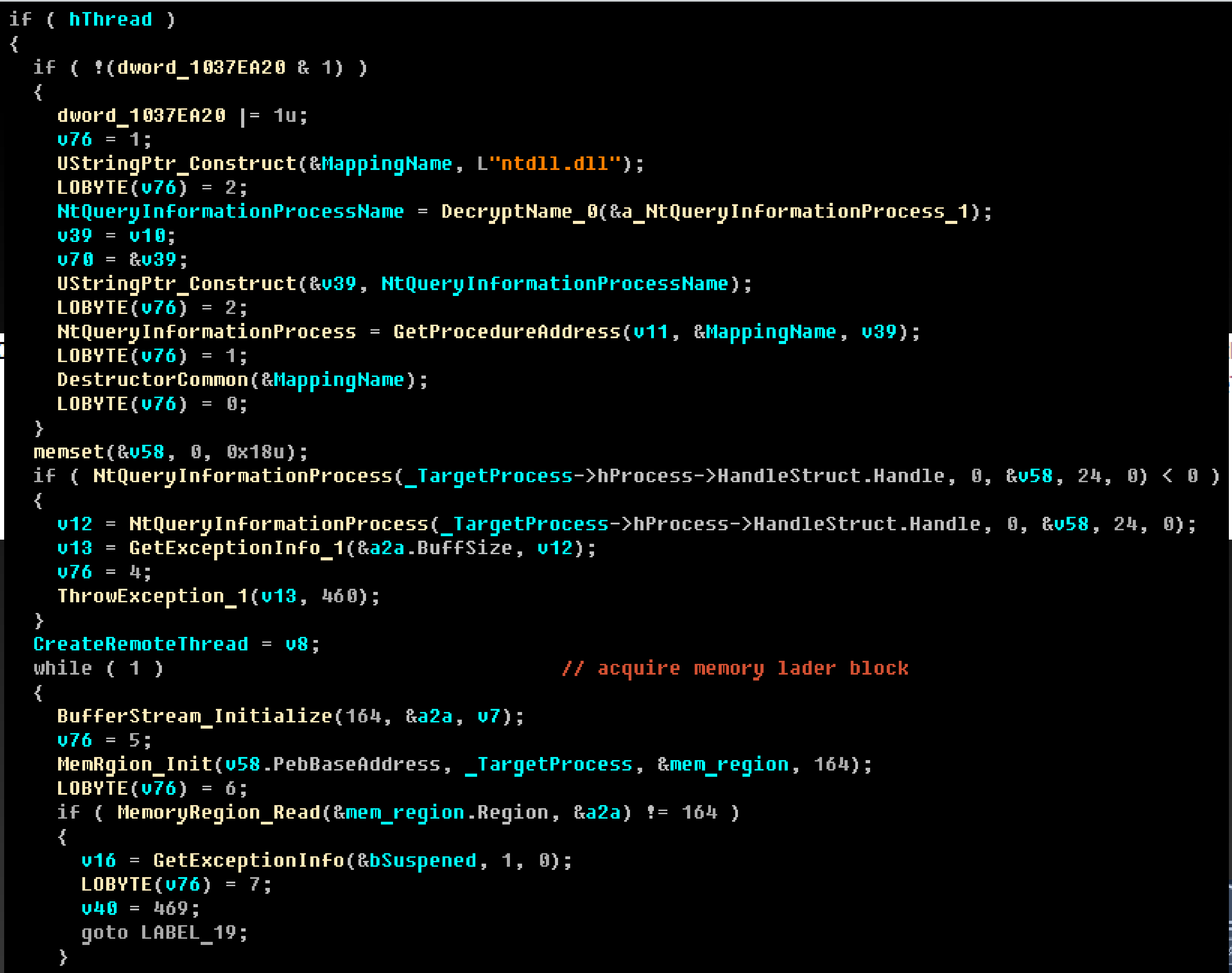


```
PROCESS_HANDLE_STRUCT *__thiscall CreateRemoteThread(    PROCESS_HANDLE_STRUCT* TargetProcess,
                                                          PROCESS_HANDLE_STRUCT* CReateThread,
                                                          INT EntryPoint,
                                                          INT lpParameter,
                                                          CHAR bSuspened,
                                                          CHAR* hThread)
```

Object Oriented API Wrappers and Implicit Calls



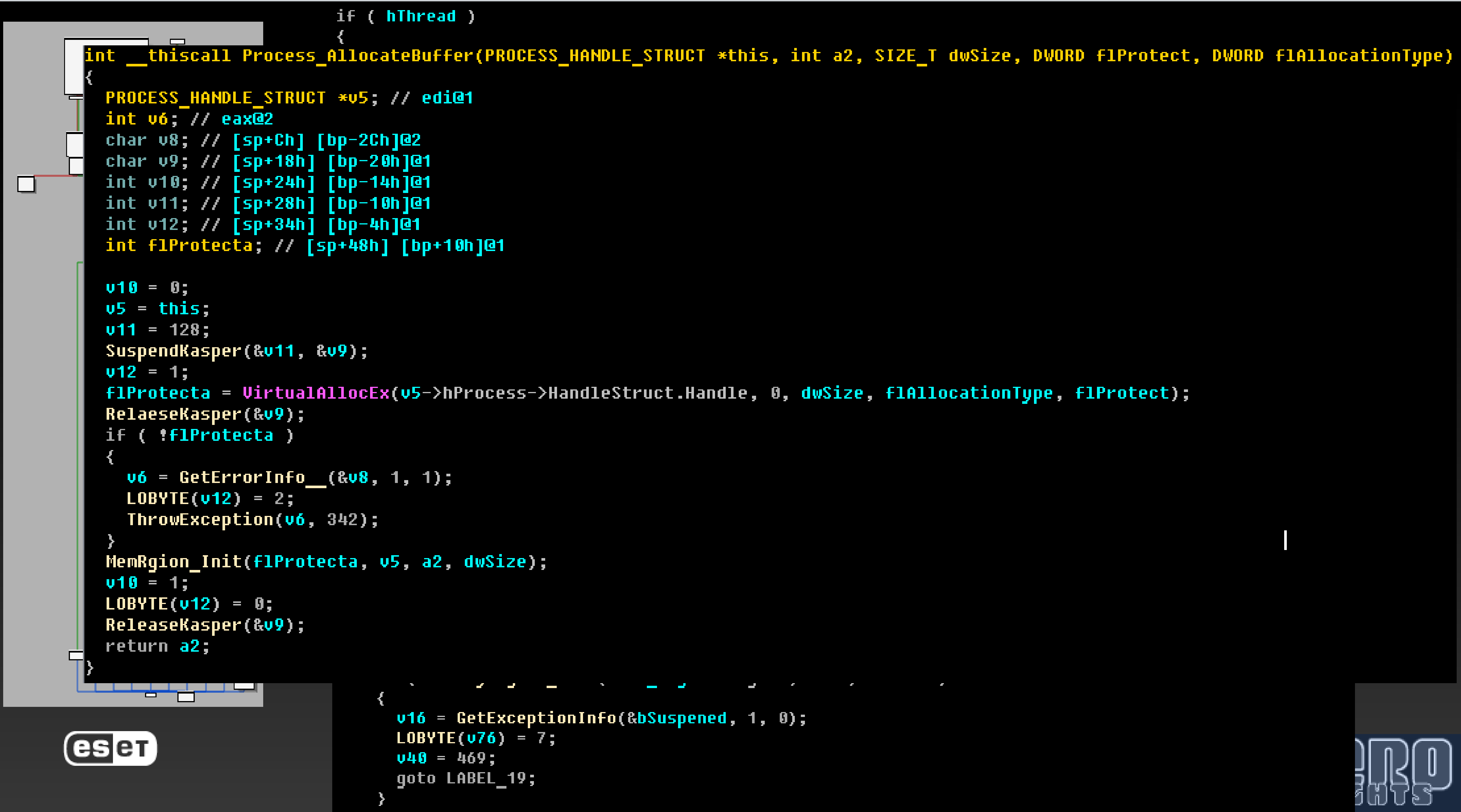
PRO



```
if ( hThread )
{
  if ( !(dword_1037EA20 & 1) )
  {
    dword_1037EA20 |= 1u;
    v76 = 1;
    UStringPtr_Construct(&MappingName, L"ntdll.dll");
    LOBYTE(v76) = 2;
    NtQueryInformationProcessName = DecryptName_0(&a_NtQueryInformationProcess_1);
    v39 = v10;
    v70 = &v39;
    UStringPtr_Construct(&v39, NtQueryInformationProcessName);
    LOBYTE(v76) = 2;
    NtQueryInformationProcess = GetProcedureAddress(v11, &MappingName, v39);
    LOBYTE(v76) = 1;
    DestructorCommon(&MappingName);
    LOBYTE(v76) = 0;
  }
  memset(&v58, 0, 0x18u);
  if ( NtQueryInformationProcess(_TargetProcess->hProcess->HandleStruct.Handle, 0, &v58, 24, 0) < 0 )
  {
    v12 = NtQueryInformationProcess(_TargetProcess->hProcess->HandleStruct.Handle, 0, &v58, 24, 0);
    v13 = GetExceptionInfo_1(&a2a.BuffSize, v12);
    v76 = 4;
    ThrowException_1(v13, 460);
  }
  CreateRemoteThread = v8;
  while ( 1 )                                    // acquire memory lader block
  {
    BufferStream_Initialize(164, &a2a, v7);
    v76 = 5;
    MemRgion_Init(v58.PebBaseAddress, _TargetProcess, &mem_region, 164);
    LOBYTE(v76) = 6;
    if ( MemoryRegion_Read(&mem_region.Region, &a2a) != 164 )
    {
      v16 = GetExceptionInfo(&bSuspened, 1, 0);
      LOBYTE(v76) = 7;
      v40 = 469;
      goto LABEL_19;
    }
```

Process,
Thread,

Object Oriented API Wrappers and Implicit Calls



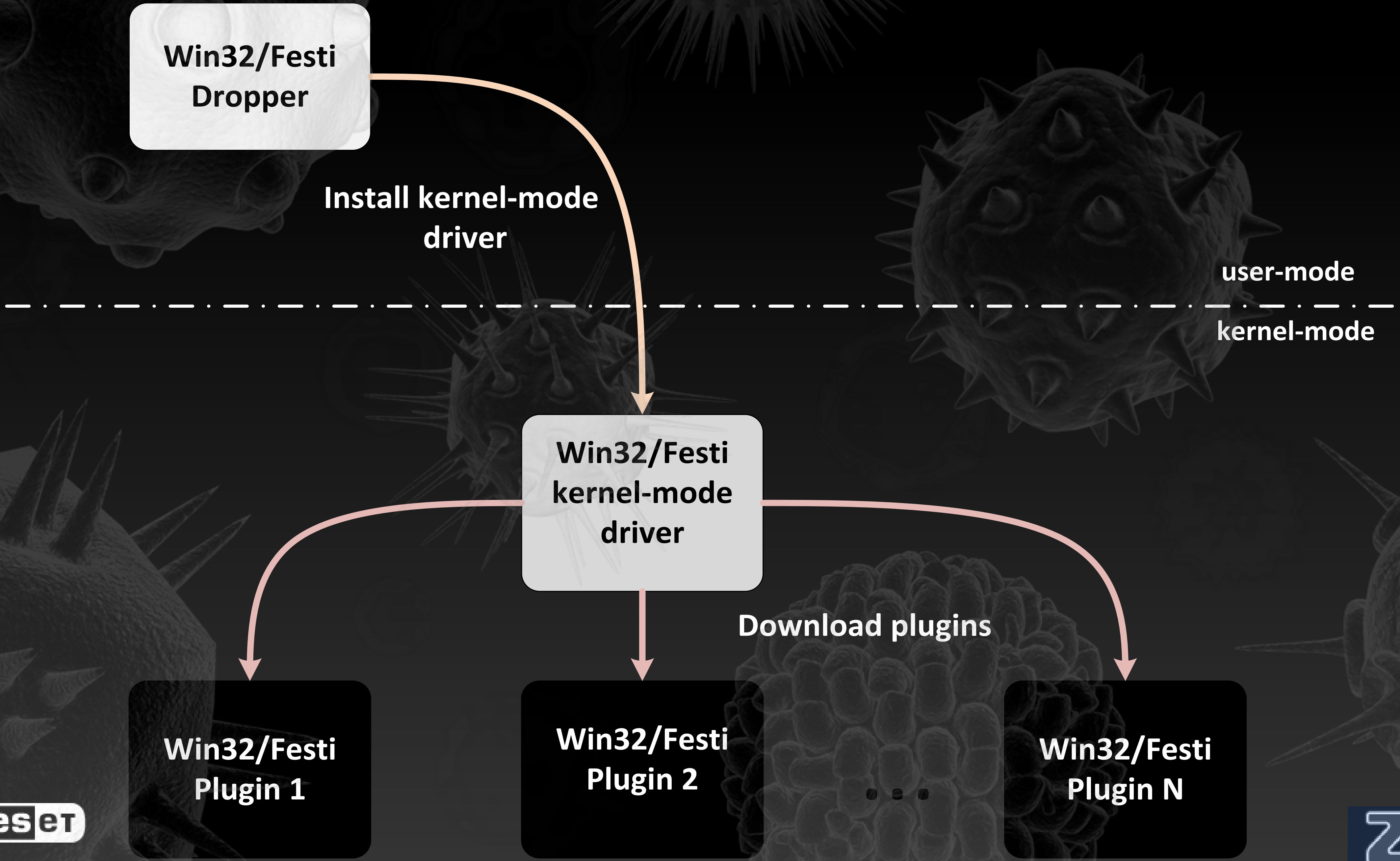
```
if ( hThread )
{
```

```
int __thiscall Process_AllocateBuffer(PROCESS_HANDLE_STRUCT *this, int a2, SIZE_T dwSize, DWORD flProtect, DWORD flAllocationType)
{
  PROCESS_HANDLE_STRUCT *v5; // edi@1
  int v6; // eax@2
  char v8; // [sp+Ch] [bp-2Ch]@2
  char v9; // [sp+18h] [bp-20h]@1
  int v10; // [sp+24h] [bp-14h]@1
  int v11; // [sp+28h] [bp-10h]@1
  int v12; // [sp+34h] [bp-4h]@1
  int flProtecta; // [sp+48h] [bp+10h]@1

  v10 = 0;
  v5 = this;
  v11 = 128;
  SuspendKasper(&v11, &v9);
  v12 = 1;
  flProtecta = VirtualAllocEx(v5->hProcess->HandleStruct.Handle, 0, dwSize, flAllocationType, flProtect);
  RelaeseKasper(&v9);
  if ( !flProtecta )
  {
    v6 = GetErrorInfo__(&v8, 1, 1);
    LOBYTE(v12) = 2;
    ThrowException(v6, 342);
  }
  MemRgion_Init(flProtecta, v5, a2, dwSize);
  v10 = 1;
  LOBYTE(v12) = 0;
  ReleaseKasper(&v9);
  return a2;
}
```

```
{
  v16 = GetExceptionInfo(&bSuspened, 1, 0);
  LOBYTE(v76) = 7;
  v40 = 469;
  goto LABEL_19;
}
```

Festi: OOP in kernel-mode

Main Festi Functionality store in kernel mode



Win32/Festi Dropper

Install kernel-mode driver

user-mode

kernel-mode

Win32/Festi kernel-mode driver

Download plugins

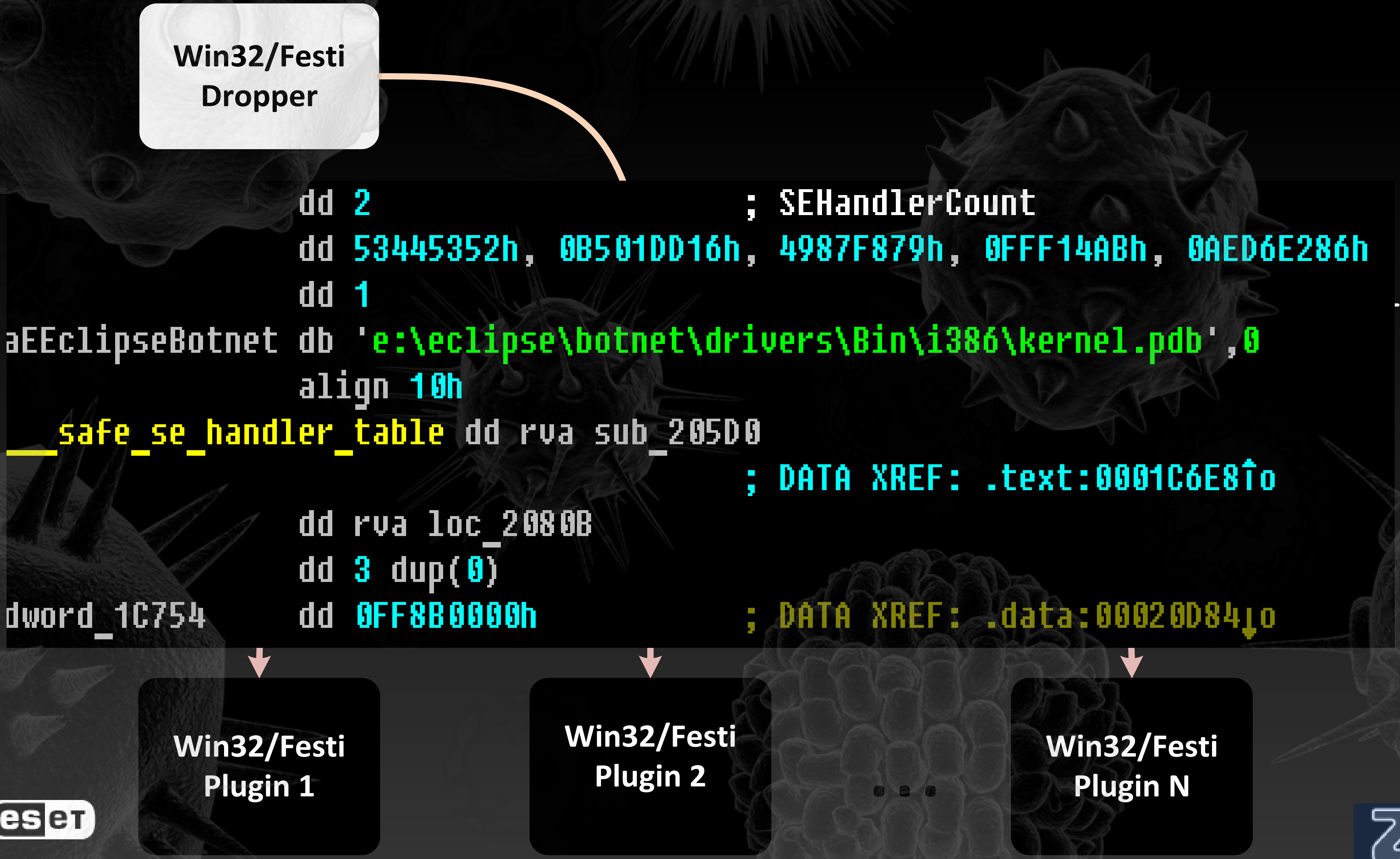
Win32/Festi Plugin 1

Win32/Festi Plugin 2

. . .

Win32/Festi Plugin N

Main Festi Functionality store in kernel mode



Win32/Festi Dropper

```
                dd 2                            ; SEHandlerCount
                dd 53445352h, 0B501DD16h, 4987F879h, 0FFF14ABh, 0AED6E286h
                dd 1
aEEclipseBotnet db 'e:\eclipse\botnet\drivers\Bin\i386\kernel.pdb',0
                align 10h
__safe_se_handler_table dd rva sub_205D0
                                                ; DATA XREF: .text:0001C6E8↑o
                dd rva loc_2080B
                dd 3 dup(0)
dword_1C754     dd 0FF8B0000h                   ; DATA XREF: .data:00020D84↓o
```

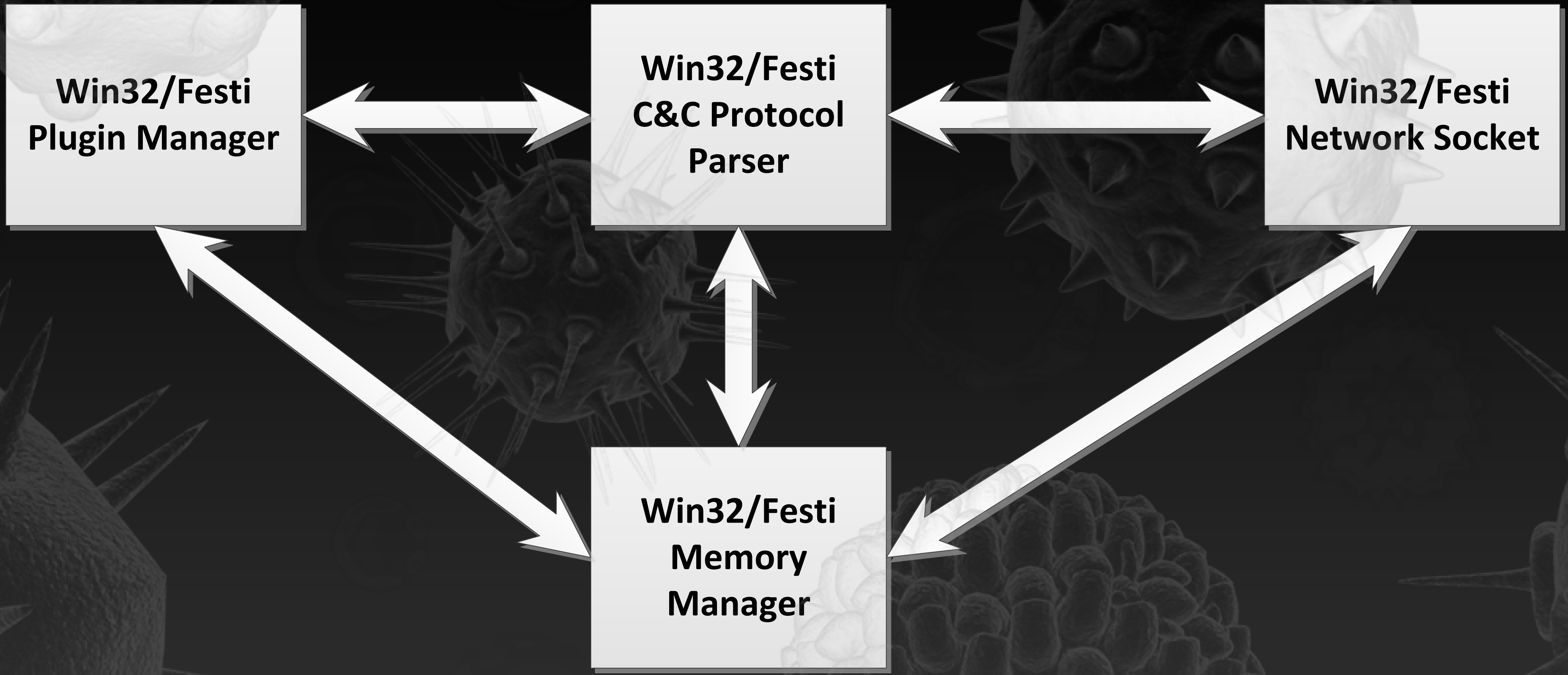
Win32/Festi Plugin 1

Win32/Festi Plugin 2

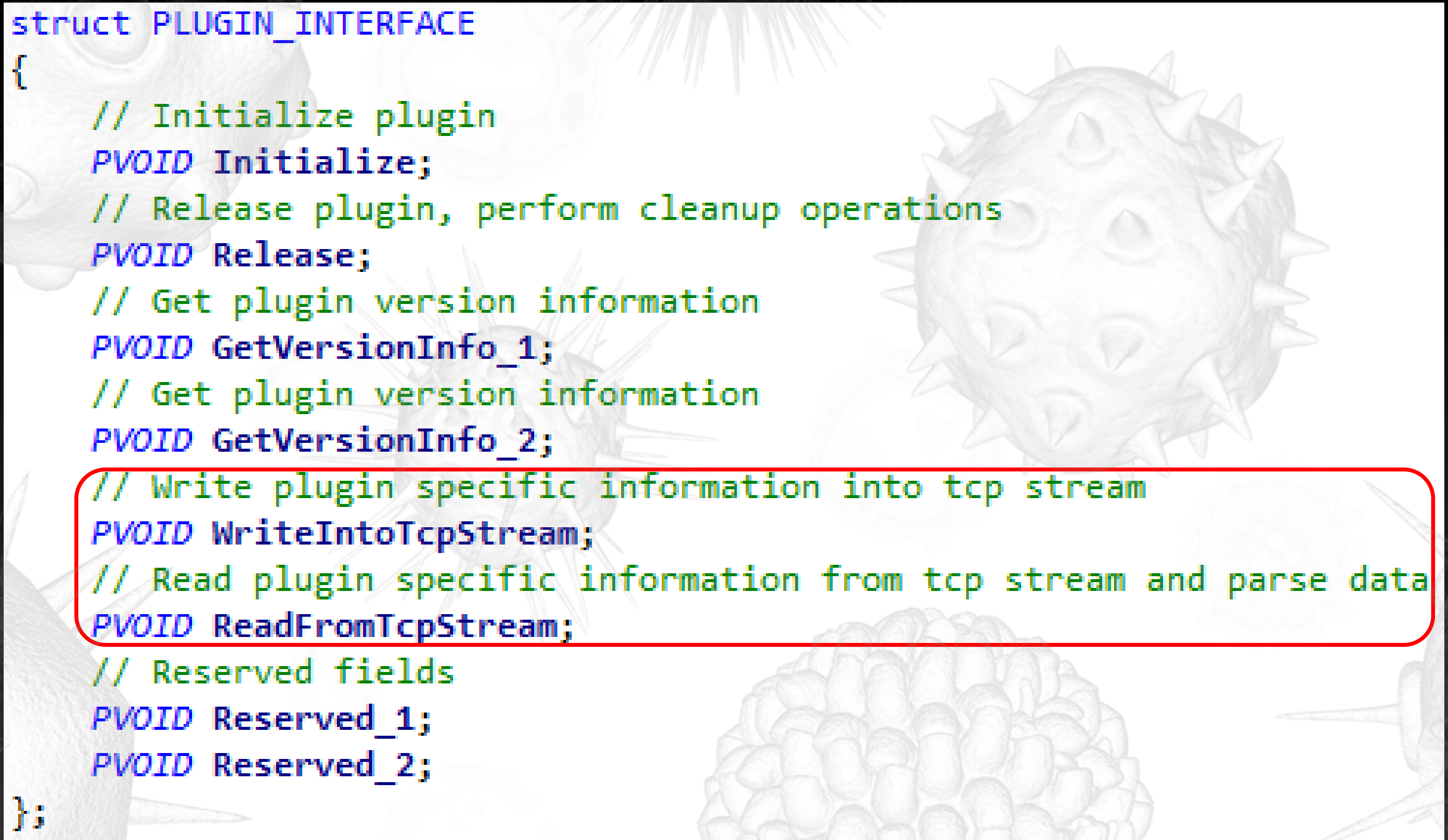
...

Win32/Festi Plugin N

Festi: Architecture


Win32/Festi Plugin Manager
Win32/Festi C&C Protocol Parser
Win32/Festi Network Socket
Win32/Festi Memory Manager

Festi: Plugin Interface



```
struct PLUGIN_INTERFACE
{
    // Initialize plugin
    PVOID Initialize;
    // Release plugin, perform cleanup operations
    PVOID Release;
    // Get plugin version information
    PVOID GetVersionInfo_1;
    // Get plugin version information
    PVOID GetVersionInfo_2;
    // Write plugin specific information into tcp stream
    PVOID WriteIntoTcpStream;
    // Read plugin specific information from tcp stream and parse data
    PVOID ReadFromTcpStream;
    // Reserved fields
    PVOID Reserved_1;
    PVOID Reserved_2;
};
```

Festi: Plugins

- **Festi plugins are volatile modules in kernel-mode address space:**
  - ✓ downloaded each time the bot is activated
  - ✓ **never stored** on the hard drive

- **The plugins are capable of:**
  - ✓ sending spam – *BotSpam.dll*
  - ✓ performing DDoS attacks – *BotDoS.dll*
  - ✓ providing proxy service – *BotSocks.dll*

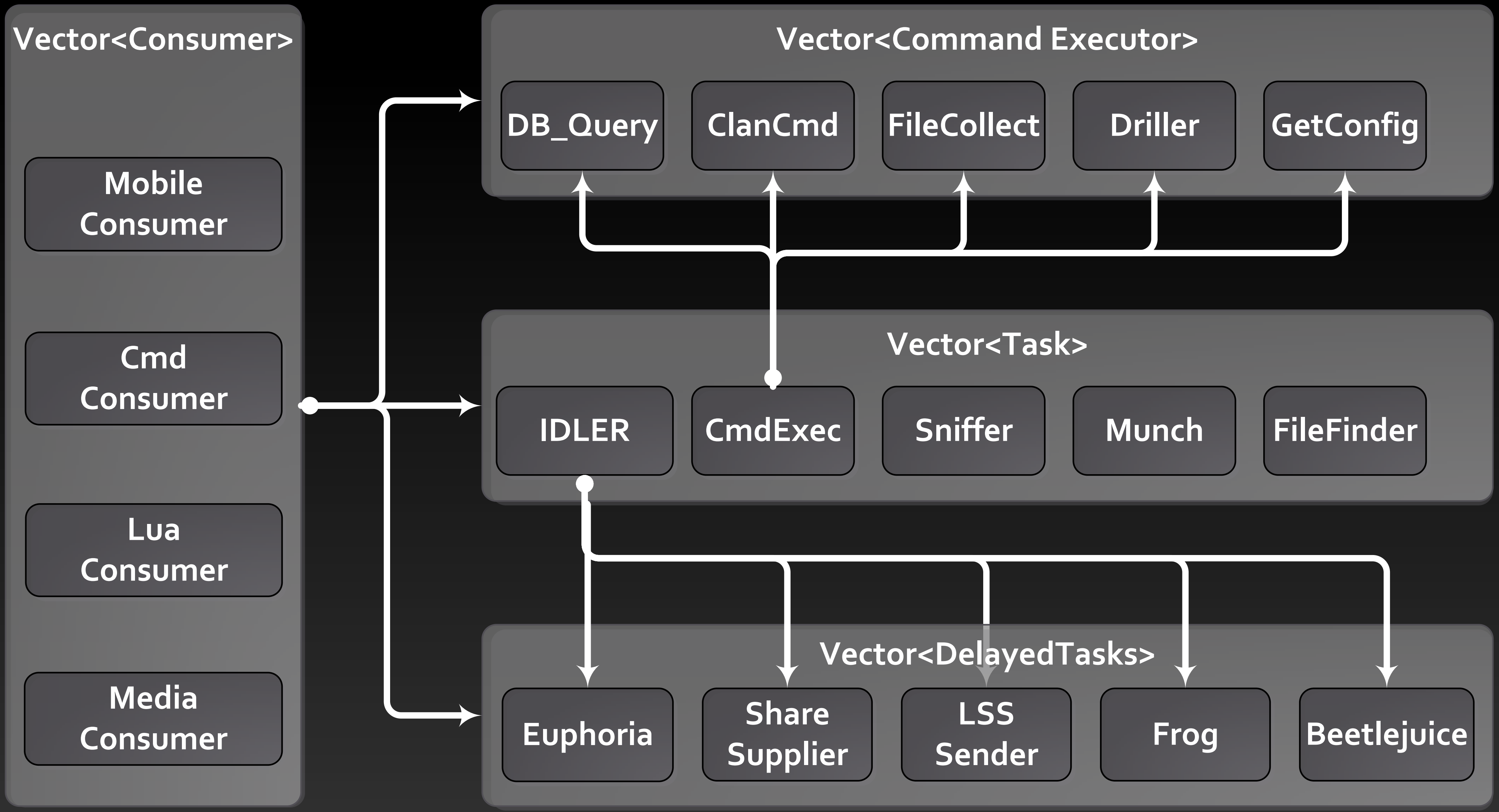
Flamer Framework Overview

An overview of the Flamer Framework

The main types used in Flamer Framework are:

- Command Executers –the objects exposing interface that allows the malware to dispatch commands received from C&C servers
- Tasks – objects of these type represent tasks executed in separate threads which constitute the backbone of the main module of Flamer
- Consumers – objects which are triggered on specific events (creation of new module, insertion of removable media and etc.)
- Delayed Tasks – these objects represent tasks which are executed periodically with certain delay.

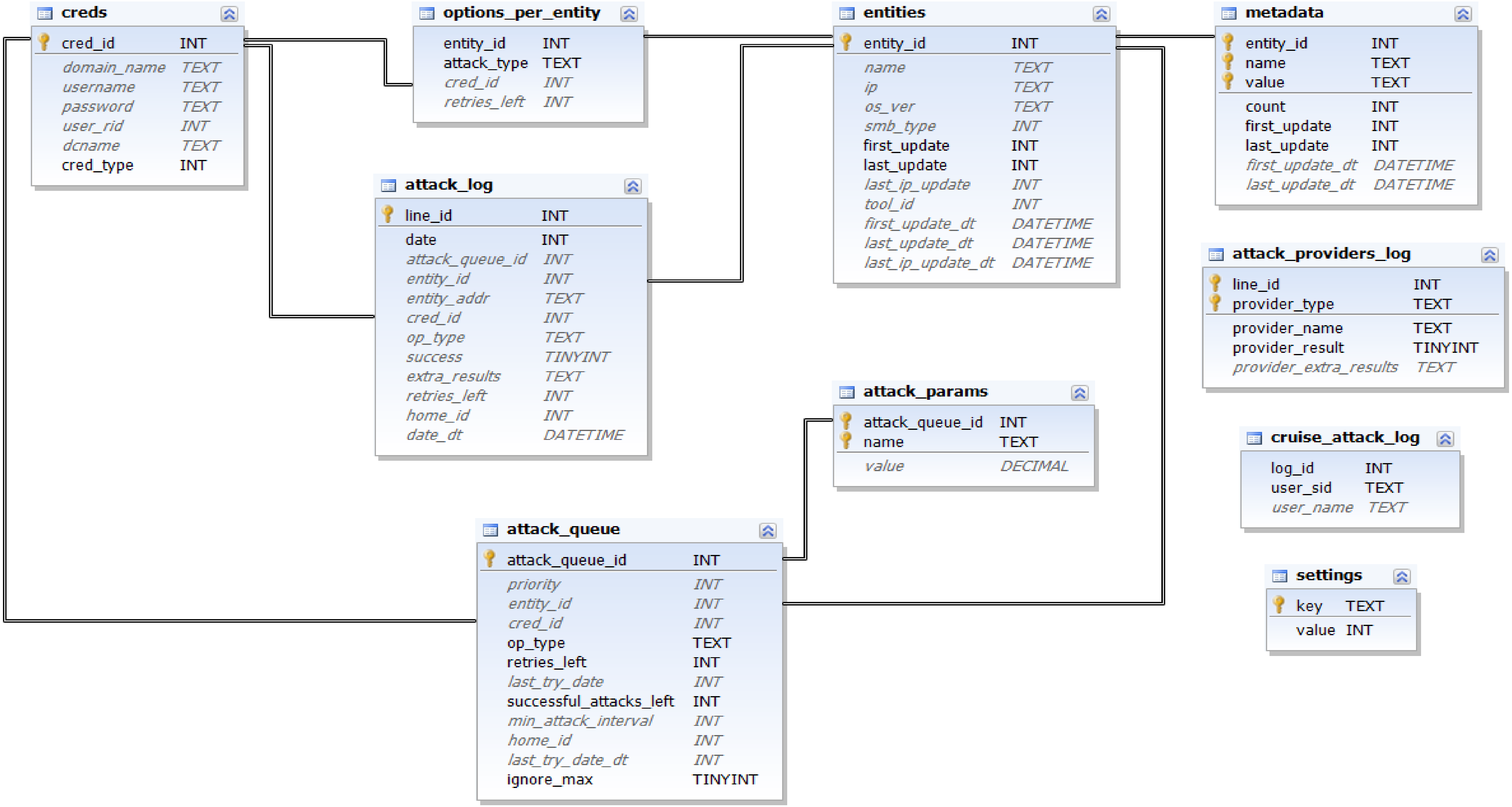
An overview of the Flamer Framework



Some of Flamer Framework Components

Security	Identifying processes in the systems corresponding to security software: antiviruses, HIPS, firewalls, system information utilities and etc.
Microbe	Leverages voice recording capabilities of the system
Idler	Running tasks in the background
BeetleJuice	Utilizes bluetooth facilities of the system
Telemetry	Logging of all the events
Gator	Communicating with C&C servers

Flamer SQL Lite Database Schema



creds

cred_id	INT
domain_name	*TEXT*
username	*TEXT*
password	*TEXT*
user_rid	*INT*
dcname	*TEXT*
cred_type	INT

options_per_entity

entity_id	INT
attack_type	TEXT
cred_id	*INT*
retries_left	*INT*

entities

entity_id	INT
name	*TEXT*
ip	*TEXT*
os_ver	*TEXT*
smb_type	*INT*
first_update	INT
last_update	INT
last_ip_update	*INT*
tool_id	*INT*
first_update_dt	*DATETIME*
last_update_dt	*DATETIME*
last_ip_update_dt	*DATETIME*

metadata

entity_id	INT
name	TEXT
value	TEXT
count	INT
first_update	INT
last_update	INT
first_update_dt	*DATETIME*
last_update_dt	*DATETIME*

attack_log

line_id	INT
date	INT
attack_queue_id	*INT*
entity_id	*INT*
entity_addr	*TEXT*
cred_id	*INT*
op_type	*TEXT*
success	*TINYINT*
extra_results	*TEXT*
retries_left	*INT*
home_id	*INT*
date_dt	*DATETIME*

attack_providers_log

line_id	INT
provider_type	TEXT
provider_name	TEXT
provider_result	TINYINT
provider_extra_results	*TEXT*

attack_params

attack_queue_id	INT
name	TEXT
value	*DECIMAL*

cruise_attack_log

log_id	INT
user_sid	TEXT
user_name	*TEXT*

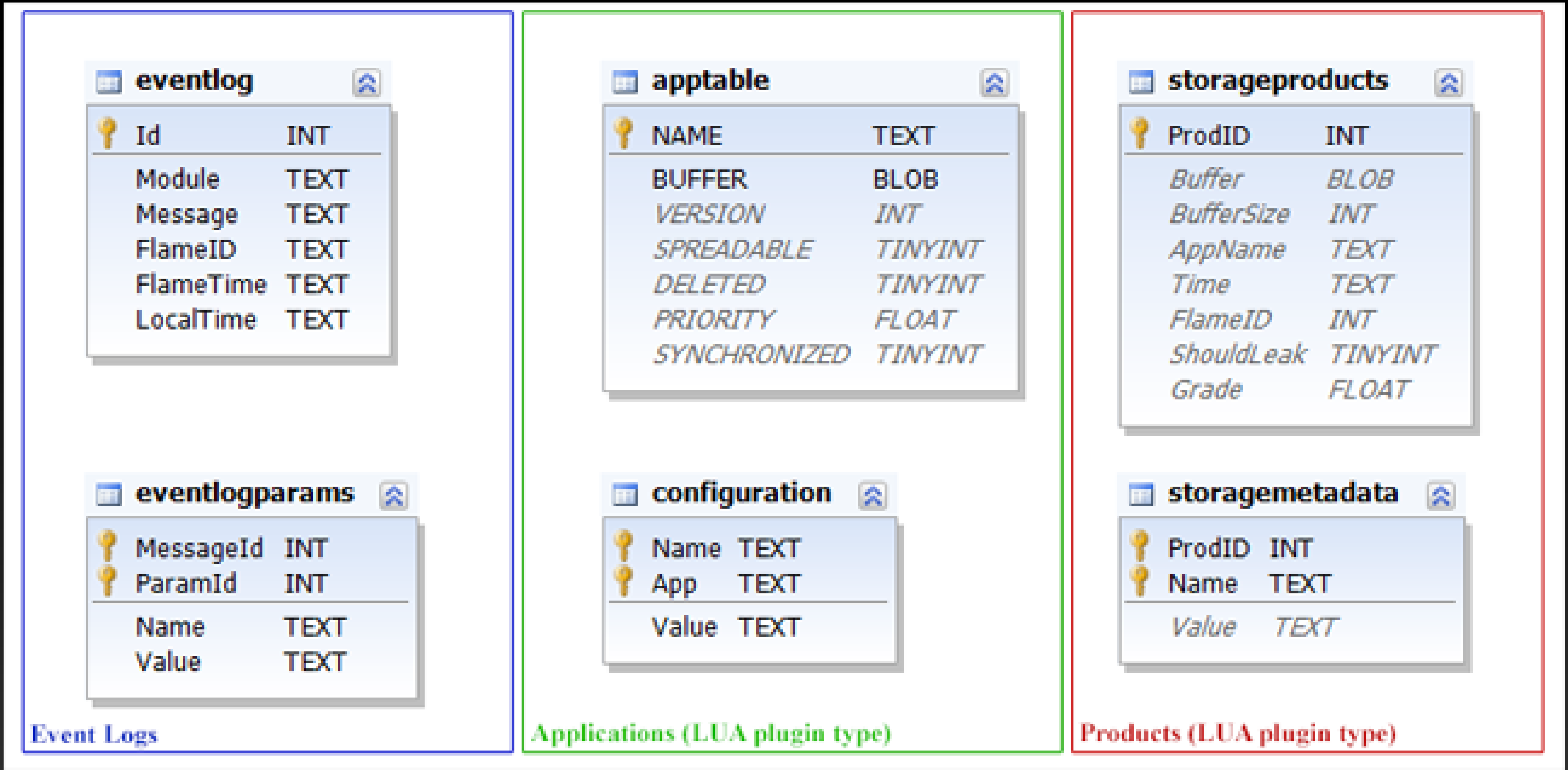
attack_queue

attack_queue_id	INT
priority	*INT*
entity_id	*INT*
cred_id	*INT*
op_type	TEXT
retries_left	INT
last_try_date	*INT*
successful_attacks_left	INT
min_attack_interval	*INT*
home_id	*INT*
last_try_date_dt	*INT*
ignore_max	TINYINT

settings

key	TEXT
value	INT

Flamer SQL Lite Database Schema



eventlog

🔑	Id	INT
	Module	TEXT
	Message	TEXT
	FlameID	TEXT
	FlameTime	TEXT
	LocalTime	TEXT

eventlogparams

🔑	MessageId	INT
🔑	ParamId	INT
	Name	TEXT
	Value	TEXT

Event Logs

apptable

🔑	NAME	TEXT
	BUFFER	BLOB
	VERSION	*INT*
	SPREADABLE	*TINYINT*
	DELETED	*TINYINT*
	PRIORITY	*FLOAT*
	SYNCHRONIZED	*TINYINT*

configuration

🔑	Name	TEXT
🔑	App	TEXT
	Value	TEXT

Applications (LUA plugin type)

storageproducts

🔑	ProdID	INT
	Buffer	*BLOB*
	BufferSize	*INT*
	AppName	*TEXT*
	Time	*TEXT*
	FlameID	*INT*
	ShouldLeak	*TINYINT*
	Grade	*FLOAT*

storagemetadata

🔑	ProdID	INT
🔑	Name	TEXT
	Value	*TEXT*

Products (LUA plugin type)

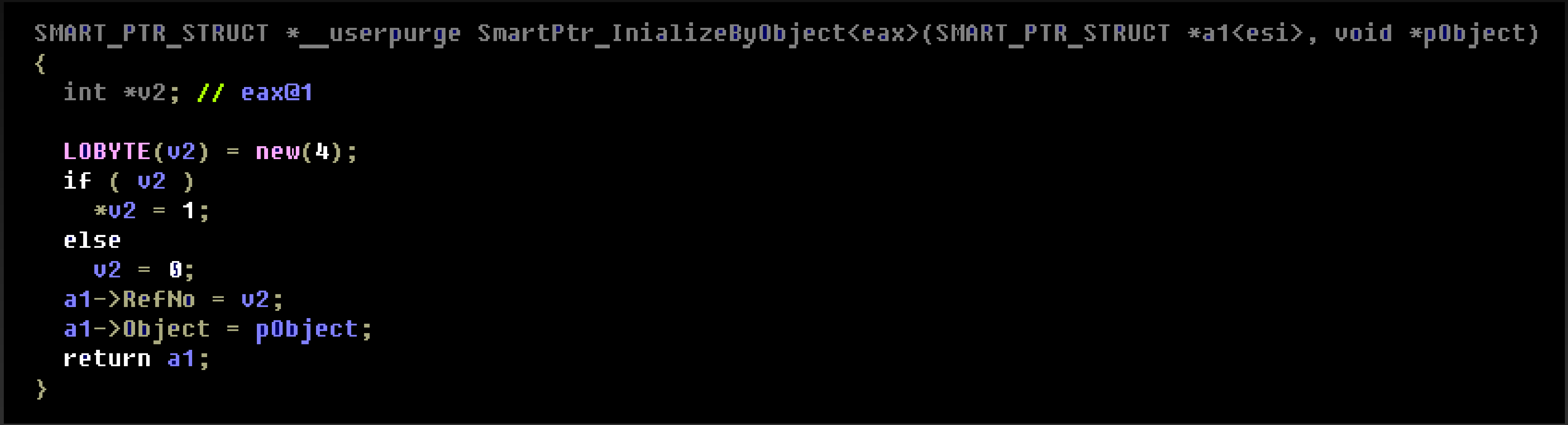
REconstructing Flamer Framework

Data Types Being Used

- Smart pointers
- Strings
- Vectors to maintain the objects
- Custom data types: wrappers, tasks, triggers and etc.

Data Types Being Used: Smart pointers

```
typedef struct SMART_PTR
{
   void        *pObject;    // pointer to the object
   int         *RefNo;      // reference counter
};
```



```
SMART_PTR_STRUCT *__userpurge SmartPtr_InializeByObject<eax>(SMART_PTR_STRUCT *a1<esi>, void *pObject)
{
  int *v2; // eax@1

  LOBYTE(v2) = new(4);
  if ( v2 )
    *v2 = 1;
  else
    v2 = 0;
  a1->RefNo = v2;
  a1->Object = pObject;
  return a1;
}
```

Data Types Being Used: Strings

```
struct USTRING_STRUCT
{
  void *vTable;            // pointer to the table
  int RefNo;               // reference counter
  int Initialized;
  wchar_t *UnicodeBuffer;  // pointer to unicode string
  char *AsciiBuffer;       // pointer to ASCII string
  int AsciiLength;         // length of the ASCII string
  int Reserved;
  int Length;              // Length of unicode string
  int LengthMax;           // Size of UnicodeBuffer
};
```

Data Types Being Used: Vectors

```
struct VECTOR
{
  void *vTable;            // pointer to the table
  int NumberOfItems;       // self-explanatory
  int MaxSize;             // self-explanatory
  void *vector;            // pointer to buffer with elements
};
```

- Used to handle the objects:
  - ✓ tasks
  - ✓ triggers
  - ✓ etc.

Using Hex-Rays Decompiler

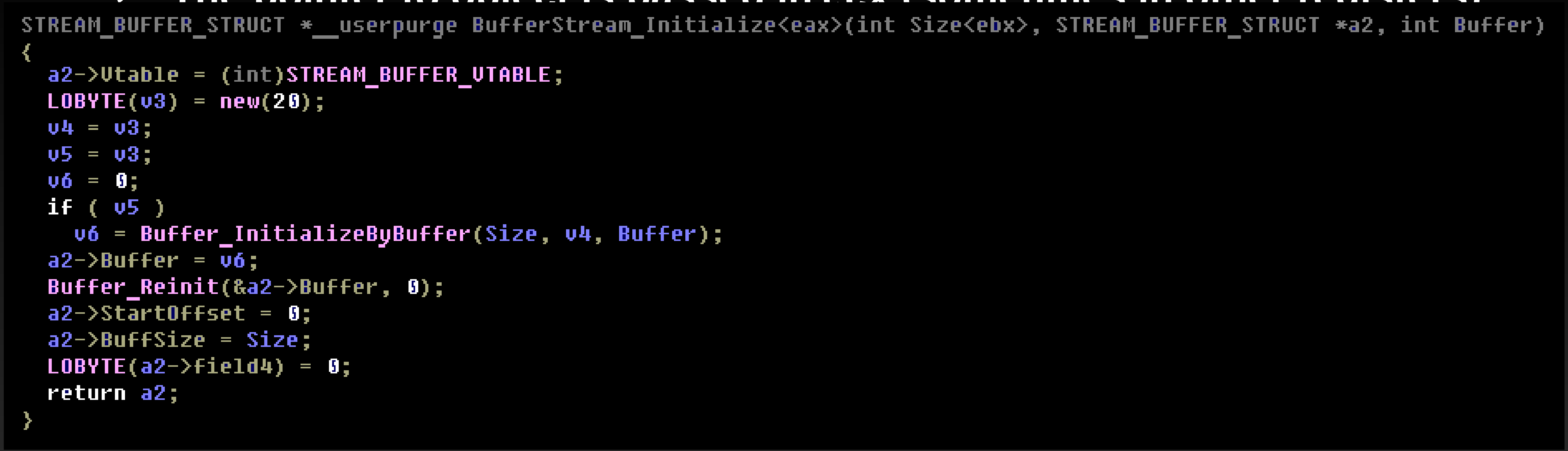
- **Identifying constructors/destructors**
  - ✓ Usually follow memory allocation
  - ✓ The pointer to object is passed in *ecx* (sometimes in other registers)

- **Reconstructing object’s attributes**
  - ✓ Creating custom type in “Local Types” for an object

- **Analyzing object’s methods**
  - ✓ Creating custom type in “Local Types” for a table of virtual routines

Using Hex-Rays Decompiler

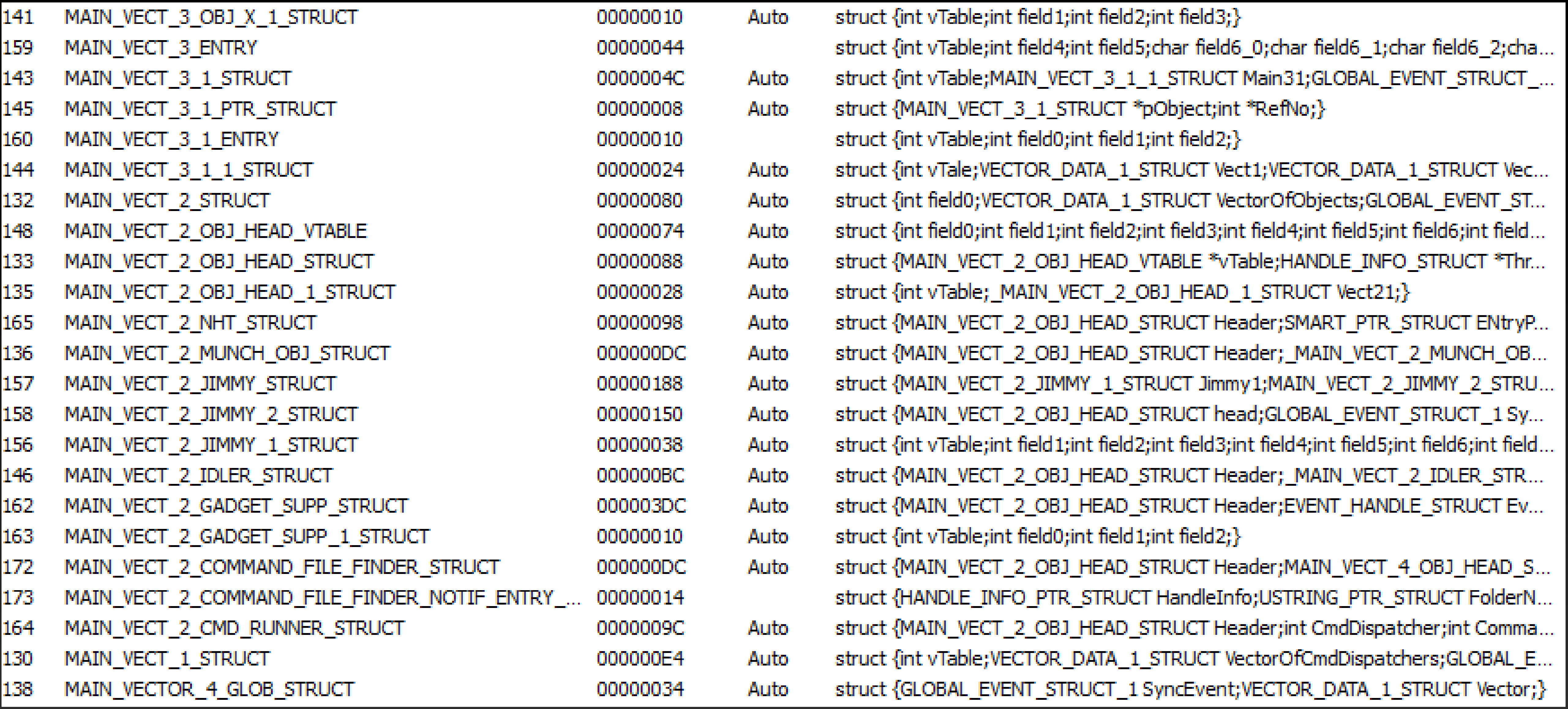
- **Identifying constructors/destructors**
  - ✓ **Usually follow memory allocation**
  - ✓ **The pointer to object is passed in *ecx* (sometimes in other registers)**



```
STREAM_BUFFER_STRUCT *__userpurge BufferStream_Initialize<eax>(int Size<ebx>, STREAM_BUFFER_STRUCT *a2, int Buffer)
{
  a2->Vtable = (int)STREAM_BUFFER_VTABLE;
  LOBYTE(v3) = new(20);
  v4 = v3;
  v5 = v3;
  v6 = 0;
  if ( v5 )
    v6 = Buffer_InitializeByBuffer(Size, v4, Buffer);
  a2->Buffer = v6;
  Buffer_Reinit(&a2->Buffer, 0);
  a2->StartOffset = 0;
  a2->BuffSize = Size;
  LOBYTE(a2->field4) = 0;
  return a2;
}
```

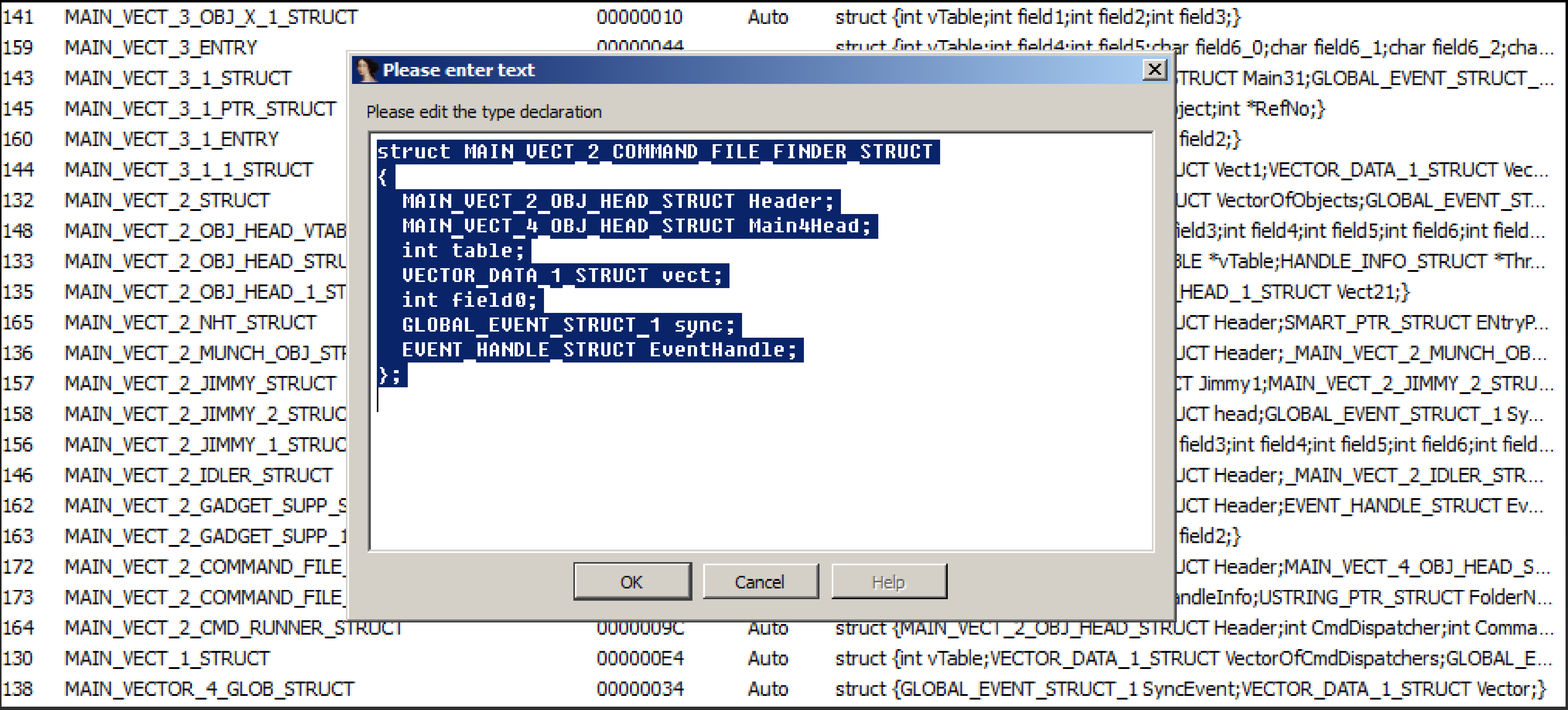
  - ✓ **Creating custom type in “Local Types” for a table of virtual routines**

Reconstructing Object’s Attributes



141	MAIN_VECT_3_OBJ_X_1_STRUCT	00000010	Auto	struct {int vTable;int field1;int field2;int field3;}
159	MAIN_VECT_3_ENTRY	00000044		struct {int vTable;int field4;int field5;char field6_0;char field6_1;char field6_2;cha...
143	MAIN_VECT_3_1_STRUCT	0000004C	Auto	struct {int vTable;MAIN_VECT_3_1_1_STRUCT Main31;GLOBAL_EVENT_STRUCT_...
145	MAIN_VECT_3_1_PTR_STRUCT	00000008	Auto	struct {MAIN_VECT_3_1_STRUCT *pObject;int *RefNo;}
160	MAIN_VECT_3_1_ENTRY	00000010		struct {int vTable;int field0;int field1;int field2;}
144	MAIN_VECT_3_1_1_STRUCT	00000024	Auto	struct {int vTale;VECTOR_DATA_1_STRUCT Vect1;VECTOR_DATA_1_STRUCT Vec...
132	MAIN_VECT_2_STRUCT	00000080	Auto	struct {int field0;VECTOR_DATA_1_STRUCT VectorOfObjects;GLOBAL_EVENT_ST...
148	MAIN_VECT_2_OBJ_HEAD_VTABLE	00000074	Auto	struct {int field0;int field1;int field2;int field3;int field4;int field5;int field6;int field...
133	MAIN_VECT_2_OBJ_HEAD_STRUCT	00000088	Auto	struct {MAIN_VECT_2_OBJ_HEAD_VTABLE *vTable;HANDLE_INFO_STRUCT *Thr...
135	MAIN_VECT_2_OBJ_HEAD_1_STRUCT	00000028	Auto	struct {int vTable;_MAIN_VECT_2_OBJ_HEAD_1_STRUCT Vect21;}
165	MAIN_VECT_2_NHT_STRUCT	00000098	Auto	struct {MAIN_VECT_2_OBJ_HEAD_STRUCT Header;SMART_PTR_STRUCT ENtryP...
136	MAIN_VECT_2_MUNCH_OBJ_STRUCT	000000DC	Auto	struct {MAIN_VECT_2_OBJ_HEAD_STRUCT Header;_MAIN_VECT_2_MUNCH_OB...
157	MAIN_VECT_2_JIMMY_STRUCT	00000188	Auto	struct {MAIN_VECT_2_JIMMY_1_STRUCT Jimmy1;MAIN_VECT_2_JIMMY_2_STRU...
158	MAIN_VECT_2_JIMMY_2_STRUCT	00000150	Auto	struct {MAIN_VECT_2_OBJ_HEAD_STRUCT head;GLOBAL_EVENT_STRUCT_1 Sy...
156	MAIN_VECT_2_JIMMY_1_STRUCT	00000038	Auto	struct {int vTable;int field1;int field2;int field3;int field4;int field5;int field6;int field...
146	MAIN_VECT_2_IDLER_STRUCT	000000BC	Auto	struct {MAIN_VECT_2_OBJ_HEAD_STRUCT Header;_MAIN_VECT_2_IDLER_STR...
162	MAIN_VECT_2_GADGET_SUPP_STRUCT	000003DC	Auto	struct {MAIN_VECT_2_OBJ_HEAD_STRUCT Header;EVENT_HANDLE_STRUCT Ev...
163	MAIN_VECT_2_GADGET_SUPP_1_STRUCT	00000010	Auto	struct {int vTable;int field0;int field1;int field2;}
172	MAIN_VECT_2_COMMAND_FILE_FINDER_STRUCT	000000DC	Auto	struct {MAIN_VECT_2_OBJ_HEAD_STRUCT Header;MAIN_VECT_4_OBJ_HEAD_S...
173	MAIN_VECT_2_COMMAND_FILE_FINDER_NOTIF_ENTRY_...	00000014		struct {HANDLE_INFO_PTR_STRUCT HandleInfo;USTRING_PTR_STRUCT FolderN...
164	MAIN_VECT_2_CMD_RUNNER_STRUCT	0000009C	Auto	struct {MAIN_VECT_2_OBJ_HEAD_STRUCT Header;int CmdDispatcher;int Comma...
130	MAIN_VECT_1_STRUCT	000000E4	Auto	struct {int vTable;VECTOR_DATA_1_STRUCT VectorOfCmdDispatchers;GLOBAL_E...
138	MAIN_VECTOR_4_GLOB_STRUCT	00000034	Auto	struct {GLOBAL_EVENT_STRUCT_1 SyncEvent;VECTOR_DATA_1_STRUCT Vector;}

Reconstructing Object’s Attributes



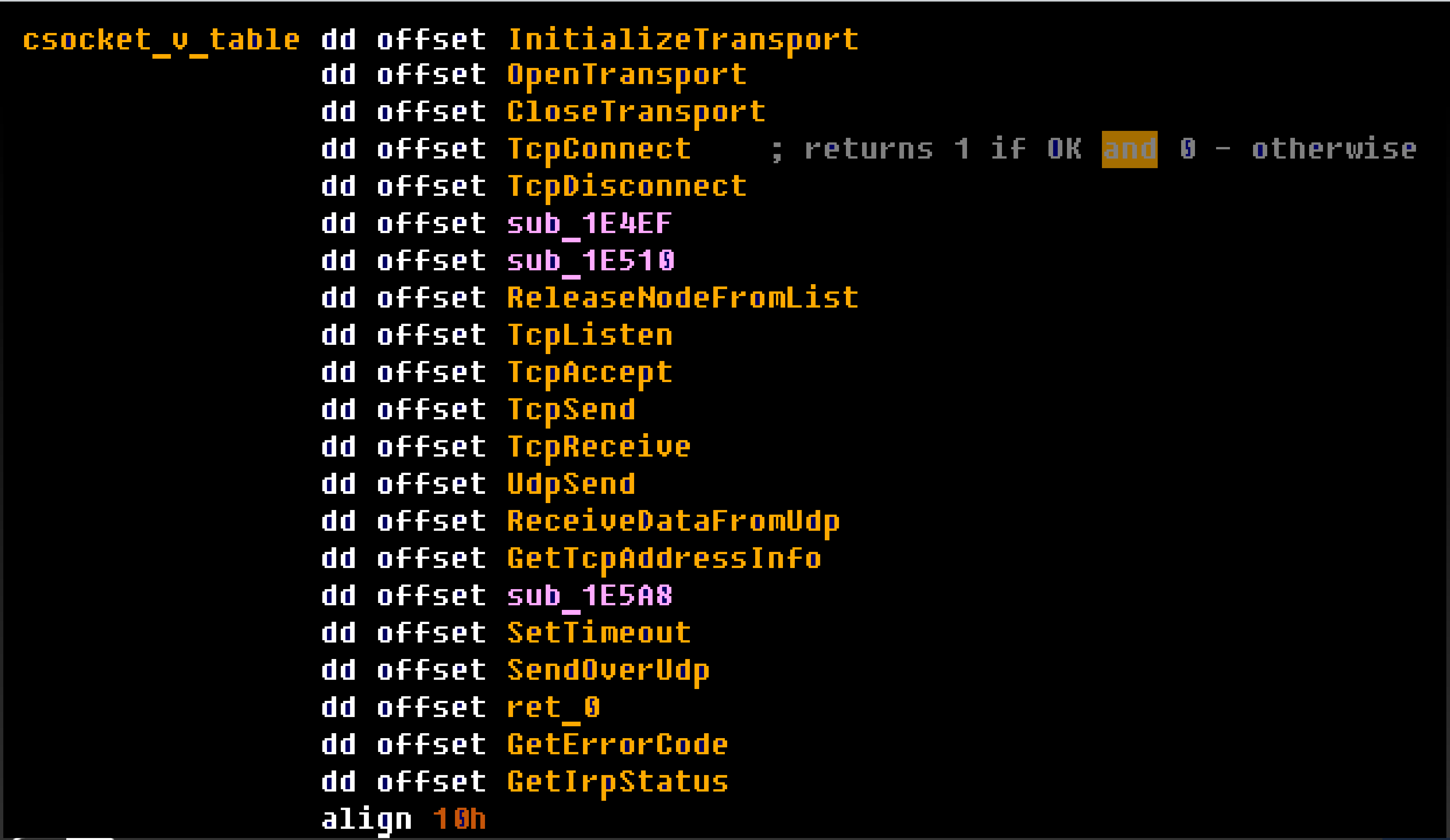
141	MAIN_VECT_3_OBJ_X_1_STRUCT	00000010	Auto	struct {int vTable;int field1;int field2;int field3;}
159	MAIN_VECT_3_ENTRY	00000044		struct {int vTable;int field4;int field5;char field6_0;char field6_1;char field6_2;cha...
143	MAIN_VECT_3_1_STRUCT			TRUCT Main31;GLOBAL_EVENT_STRUCT_...
145	MAIN_VECT_3_1_PTR_STRUCT			ject;int *RefNo;}
160	MAIN_VECT_3_1_ENTRY			field2;}
144	MAIN_VECT_3_1_1_STRUCT			JCT Vect1;VECTOR_DATA_1_STRUCT Vec...
132	MAIN_VECT_2_STRUCT			JCT VectorOfObjects;GLOBAL_EVENT_ST...
148	MAIN_VECT_2_OBJ_HEAD_VTAB			ield3;int field4;int field5;int field6;int field...
133	MAIN_VECT_2_OBJ_HEAD_STRU			3LE *vTable;HANDLE_INFO_STRUCT *Thr...
135	MAIN_VECT_2_OBJ_HEAD_1_ST			HEAD_1_STRUCT Vect21;}
165	MAIN_VECT_2_NHT_STRUCT			JCT Header;SMART_PTR_STRUCT ENtryP...
136	MAIN_VECT_2_MUNCH_OBJ_STF			JCT Header;_MAIN_VECT_2_MUNCH_OB...
157	MAIN_VECT_2_JIMMY_STRUCT			T Jimmy1;MAIN_VECT_2_JIMMY_2_STRU...
158	MAIN_VECT_2_JIMMY_2_STRUC			JCT head;GLOBAL_EVENT_STRUCT_1 Sy...
156	MAIN_VECT_2_JIMMY_1_STRUC			field3;int field4;int field5;int field6;int field...
146	MAIN_VECT_2_IDLER_STRUCT			JCT Header;_MAIN_VECT_2_IDLER_STR...
162	MAIN_VECT_2_GADGET_SUPP_S			JCT Header;EVENT_HANDLE_STRUCT Ev...
163	MAIN_VECT_2_GADGET_SUPP_1			field2;}
172	MAIN_VECT_2_COMMAND_FILE_			JCT Header;MAIN_VECT_4_OBJ_HEAD_S...
173	MAIN_VECT_2_COMMAND_FILE_			ndleInfo;USTRING_PTR_STRUCT FolderN...
164	MAIN_VECT_2_CMD_RUNNER_STRUCT	0000009C	Auto	struct {MAIN_VECT_2_OBJ_HEAD_STRUCT Header;int CmdDispatcher;int Comma...
130	MAIN_VECT_1_STRUCT	000000E4	Auto	struct {int vTable;VECTOR_DATA_1_STRUCT VectorOfCmdDispatchers;GLOBAL_E...
138	MAIN_VECTOR_4_GLOB_STRUCT	00000034	Auto	struct {GLOBAL_EVENT_STRUCT_1 SyncEvent;VECTOR_DATA_1_STRUCT Vector;}

Please enter text

Please edit the type declaration

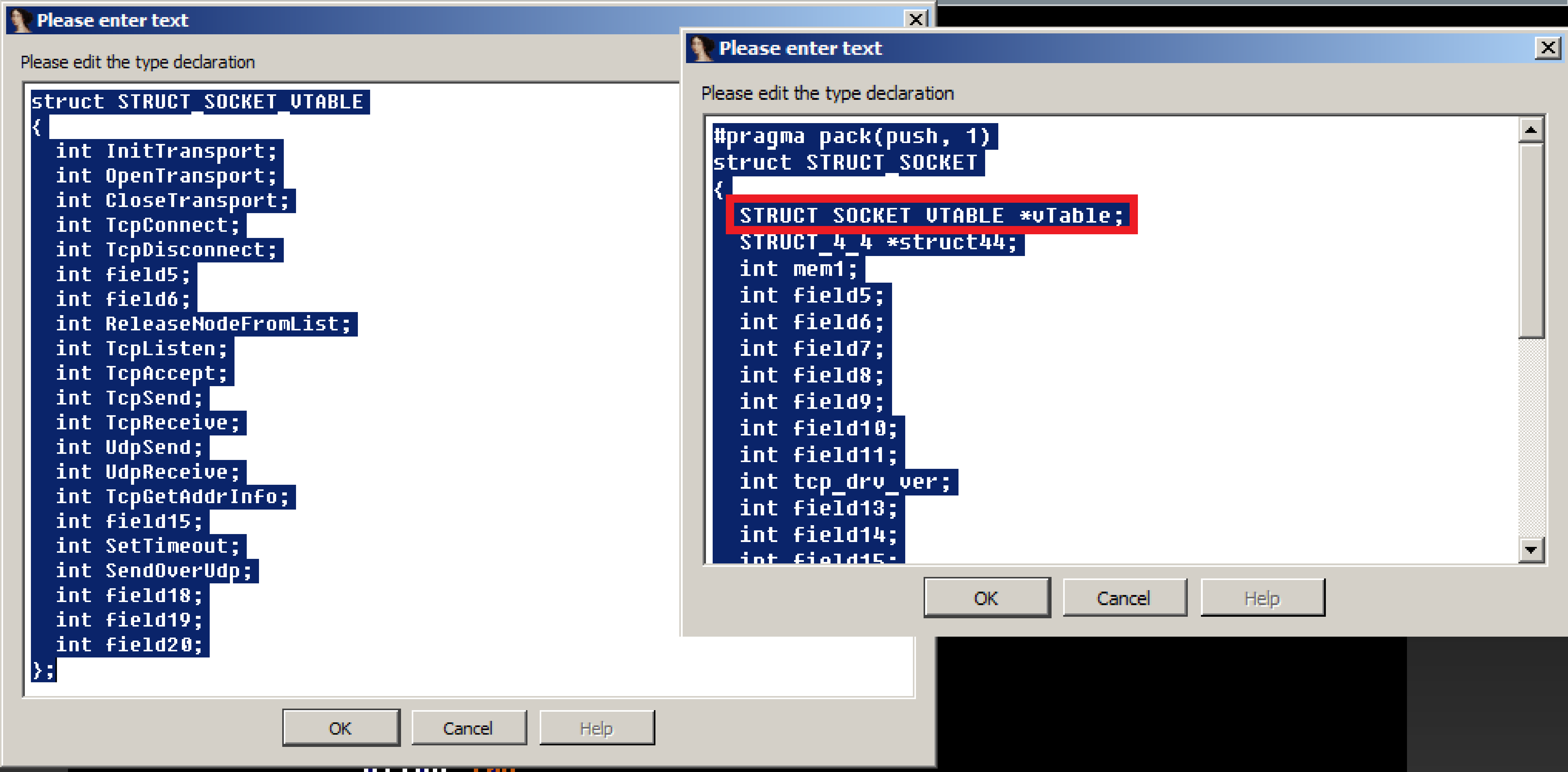
```
struct MAIN_VECT_2_COMMAND_FILE_FINDER_STRUCT
{
  MAIN_VECT_2_OBJ_HEAD_STRUCT Header;
  MAIN_VECT_4_OBJ_HEAD_STRUCT Main4Head;
  int table;
  VECTOR_DATA_1_STRUCT vect;
  int field0;
  GLOBAL_EVENT_STRUCT_1 sync;
  EVENT_HANDLE_STRUCT EventHandle;
};
```

Reconstructing Object’s Methods



```
csocket_v_table dd offset InitializeTransport
                dd offset OpenTransport
                dd offset CloseTransport
                dd offset TcpConnect     ; returns 1 if OK and 0 - otherwise
                dd offset TcpDisconnect
                dd offset sub_1E4EF
                dd offset sub_1E510
                dd offset ReleaseNodeFromList
                dd offset TcpListen
                dd offset TcpAccept
                dd offset TcpSend
                dd offset TcpReceive
                dd offset UdpSend
                dd offset ReceiveDataFromUdp
                dd offset GetTcpAddressInfo
                dd offset sub_1E5A8
                dd offset SetTimeout
                dd offset SendOverUdp
                dd offset ret_0
                dd offset GetErrorCode
                dd offset GetIrpStatus
                align 10h
```

Reconstructing Object’s Methods



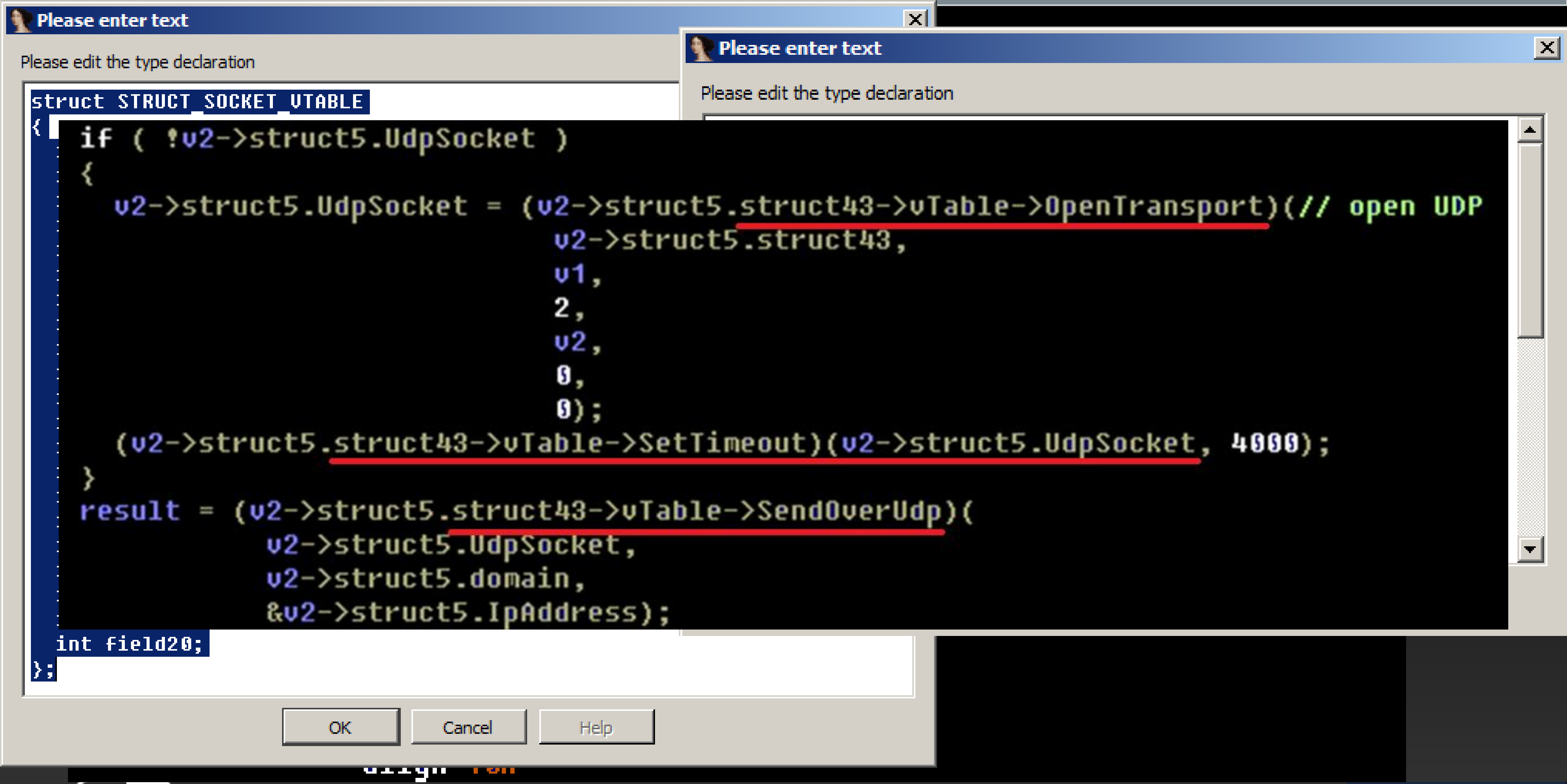
Please enter text
Please edit the type declaration
OK
Cancel
Help

```
struct STRUCT_SOCKET_VTABLE
{
  int InitTransport;
  int OpenTransport;
  int CloseTransport;
  int TcpConnect;
  int TcpDisconnect;
  int field5;
  int field6;
  int ReleaseNodeFromList;
  int TcpListen;
  int TcpAccept;
  int TcpSend;
  int TcpReceive;
  int UdpSend;
  int UdpReceive;
  int TcpGetAddrInfo;
  int field15;
  int SetTimeout;
  int SendOverUdp;
  int field18;
  int field19;
  int field20;
};
```

Please enter text
Please edit the type declaration
OK
Cancel
Help

```
#pragma pack(push, 1)
struct STRUCT_SOCKET
{
  STRUCT_SOCKET_VTABLE *vTable;
  STRUCT_4_4 *struct44;
  int mem1;
  int field5;
  int field6;
  int field7;
  int field8;
  int field9;
  int field10;
  int field11;
  int tcp_drv_ver;
  int field13;
  int field14;
  int field15;
```

Reconstructing Object’s Methods


Please enter text
Please edit the type declaration
Please enter text
Please edit the type declaration
OK
Cancel
Help

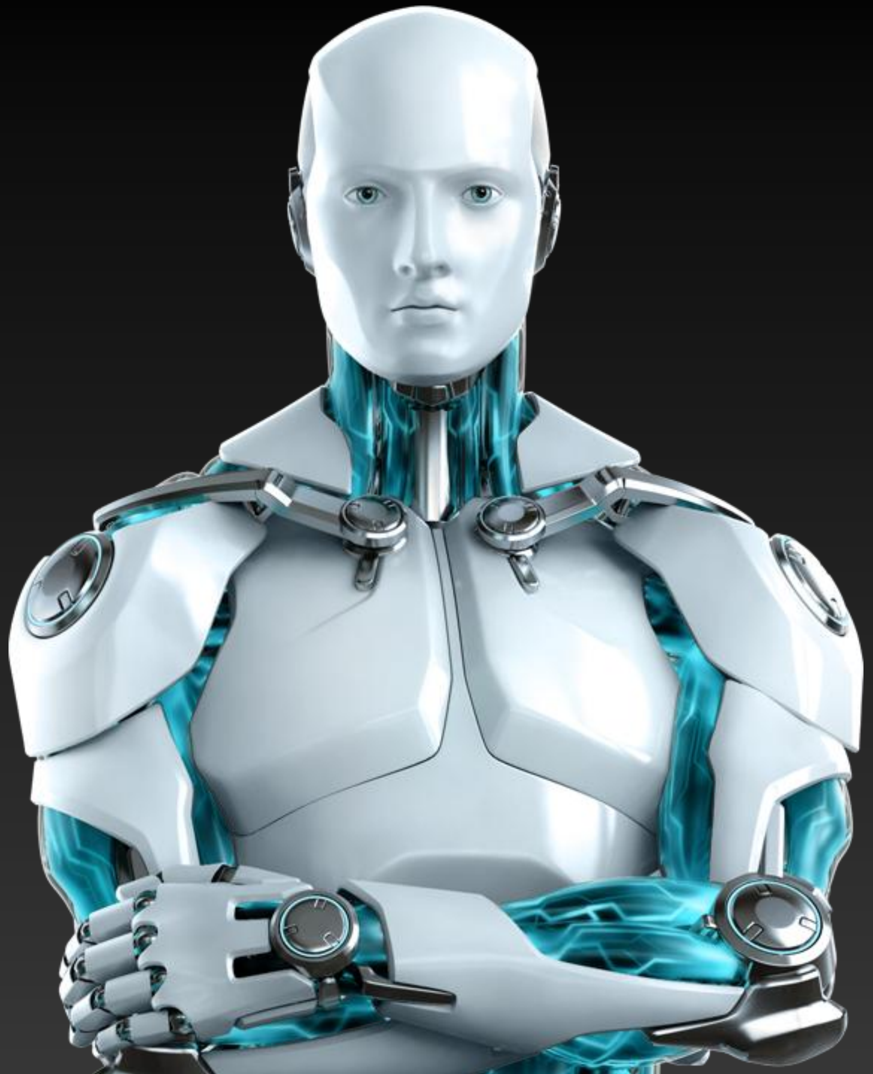
```
struct STRUCT_SOCKET_VTABLE
{
```

```
if ( !v2->struct5.UdpSocket )
{
  v2->struct5.UdpSocket = (v2->struct5.struct43->vTable->OpenTransport)(// open UDP
                            v2->struct5.struct43,
                            v1,
                            2,
                            v2,
                            0,
                            0);
  (v2->struct5.struct43->vTable->SetTimeout)(v2->struct5.UdpSocket, 4000);
}
result = (v2->struct5.struct43->vTable->SendOverUdp)(
           v2->struct5.UdpSocket,
           v2->struct5.domain,
           &v2->struct5.IpAddress);
```

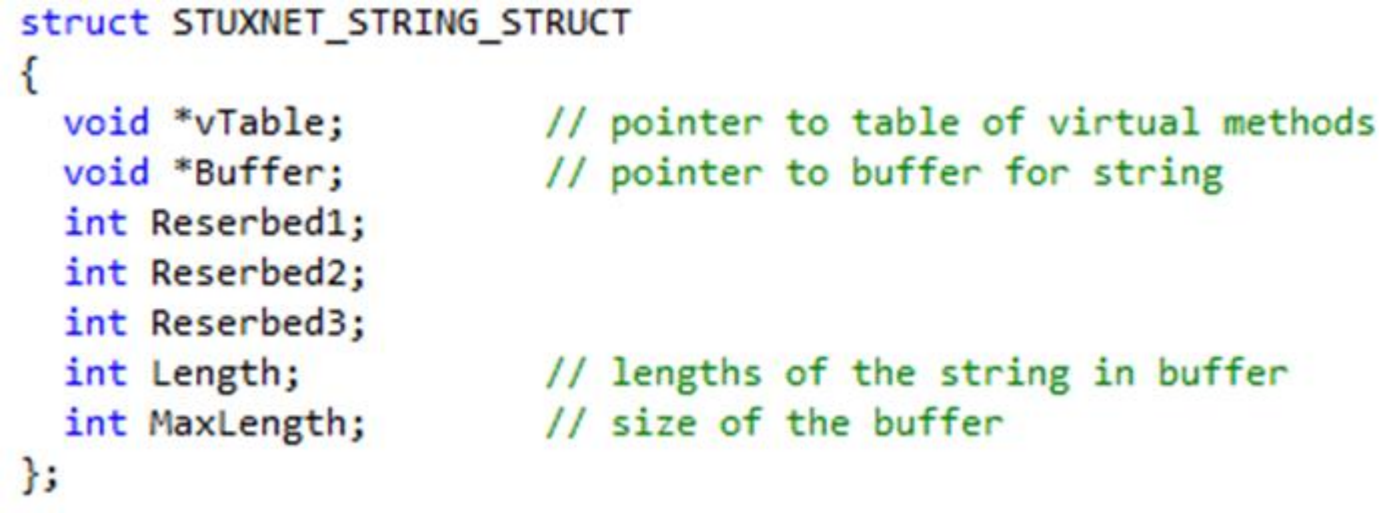
```
  int field20;
};
```

DEMO

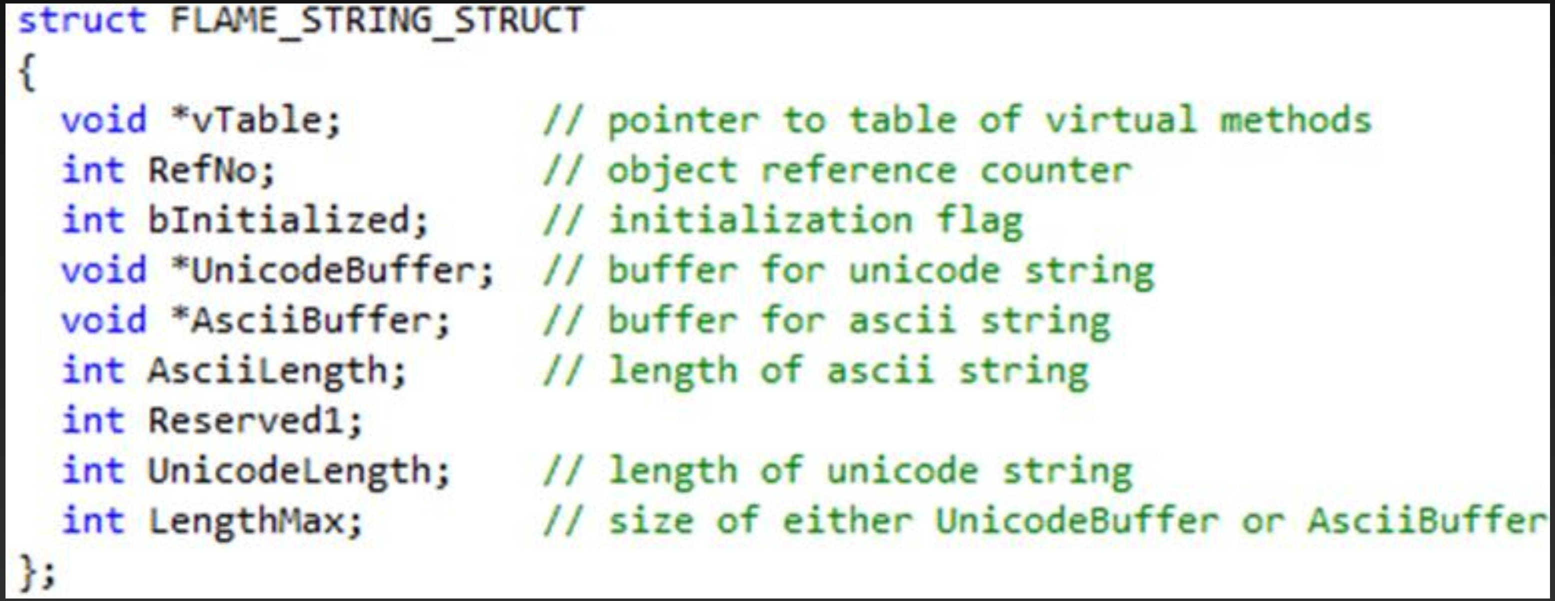
Relationship Stuxnet/Duqu/Gauss/Flamer



Source Code Base Differences



```
struct STUXNET_STRING_STRUCT
{
  void *vTable;           // pointer to table of virtual methods
  void *Buffer;           // pointer to buffer for string
  int Reserbed1;
  int Reserbed2;
  int Reserbed3;
  int Length;             // lengths of the string in buffer
  int MaxLength;          // size of the buffer
};
```



```
struct FLAME_STRING_STRUCT
{
  void *vTable;          // pointer to table of virtual methods
  int RefNo;             // object reference counter
  int bInitialized;      // initialization flag
  void *UnicodeBuffer;   // buffer for unicode string
  void *AsciiBuffer;     // buffer for ascii string
  int AsciiLength;       // length of ascii string
  int Reserved1;
  int UnicodeLength;     // length of unicode string
  int LengthMax;         // size of either UnicodeBuffer or AsciiBuffer
};
```

Exploit Implementations

Stuxnet	Duqu	Flame	Gauss
MS10-046 (LNK)		MS10-046 (LNK)	MS10-046 (LNK)
MS10-061 (Print Spooler)		MS10-061 (Print Spooler)	
MS08-067 (RPC)		MS08-067 (RPC)	
MS10-073 (Win32k.sys)			
MS10-092 (Task Scheduler)			
	MS11-087 (Win32k.sys)		

Exploit Implementations: Stuxnet & Duqu

- **The payload is injected into processes from both kernel-mode driver & user-mode module**

- **Hooks:**
  - ✓ ***ZwMapViewOfSection***
  - ✓ ***ZwCreateSection***
  - ✓ ***ZwOpenFile***
  - ✓ ***ZwClose***
  - ✓ ***ZwQueryAttributesFile***
  - ✓ ***ZwQuerySection***

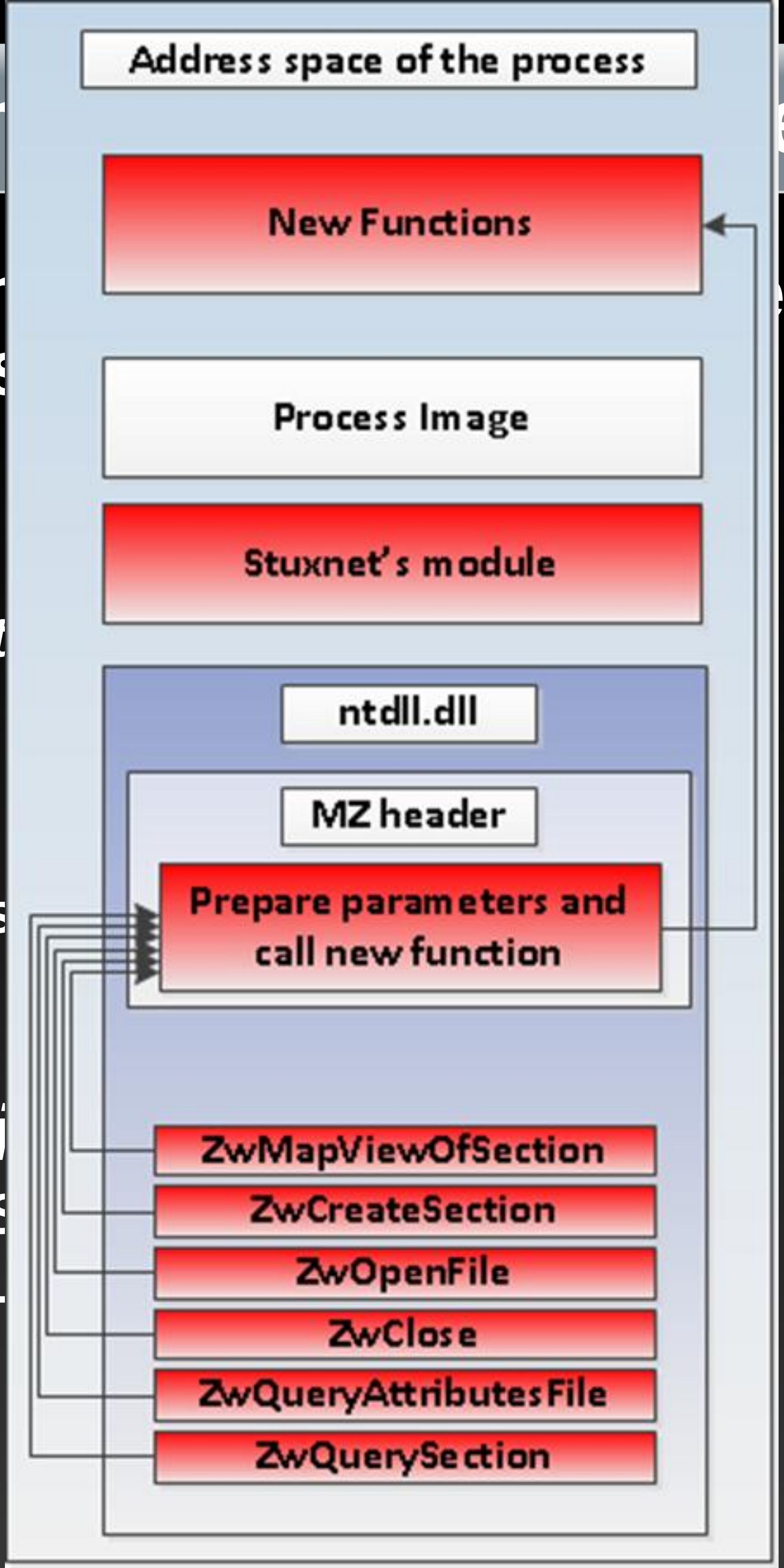
- **Executes *LoadLibraryW* passing as a parameter either:**
  - ✓ ***KERNEL32.DLL.ASLR.XXXXXXXX***
  - ✓ **SHELL32.DLL.ASLR.XXXXXXXX**

Exploit Implem et & Duqu

- ➢ The payload is in es from both kernel-mode driver & us

- ➢ Hooks:
  - ✓ *ZwMapViewOfSect*
  - ✓ *ZwCreateSection*
  - ✓ *ZwOpenFile*
  - ✓ *ZwClose*
  - ✓ *ZwQueryAttributes*
  - ✓ *ZwQuerySection*

- ➢ Executes *LoadLi* parameter either:
  - ✓ *KERNEL32.DLL.AS*
  - ✓ SHELL32.DLL.ASL



Injection mechanism: Flame

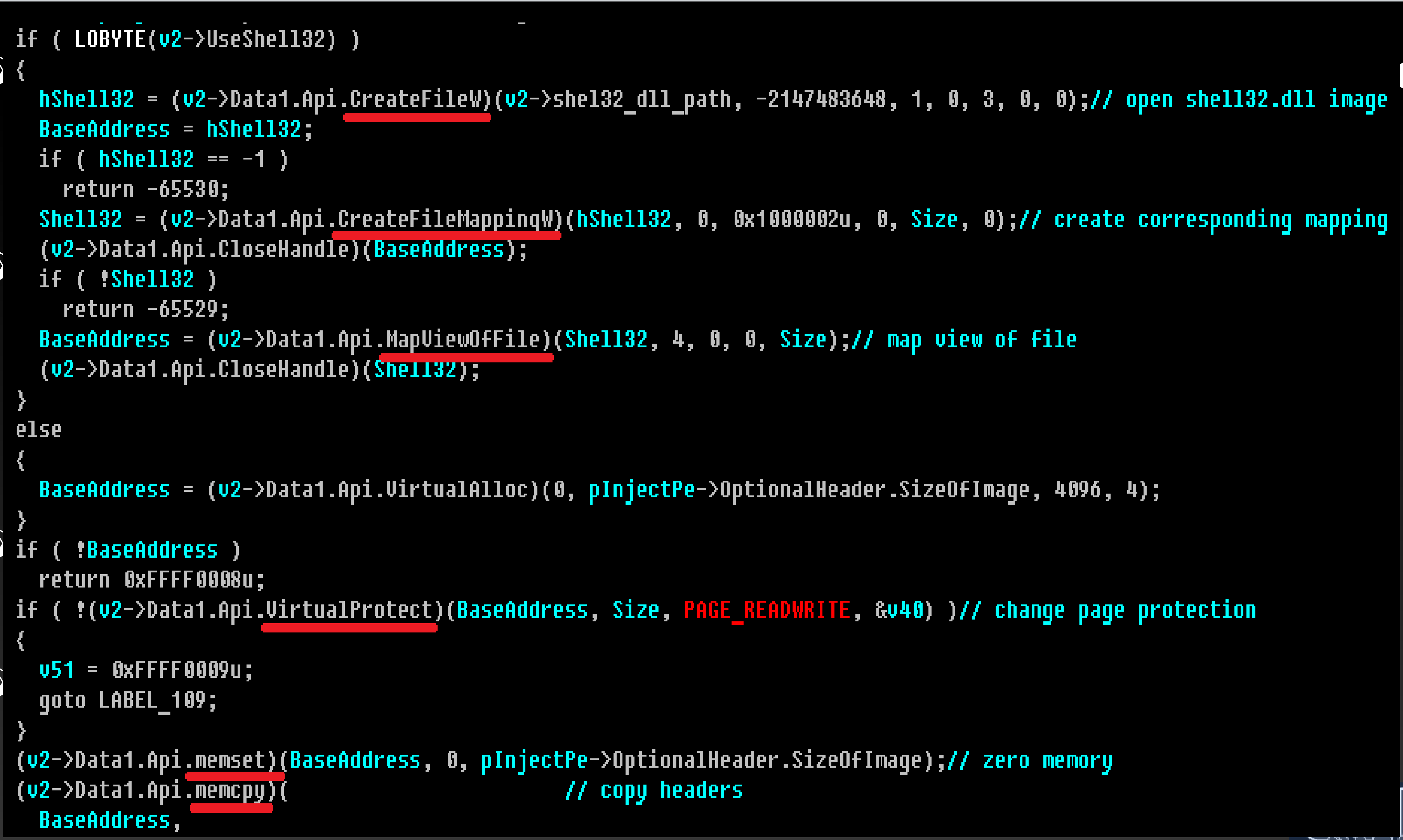
- The payload is injected into processes from user-mode module

- The injection technique is based on using:
  - ✓ *VirtualAllocEx*
  - ✓ *WriteProcessMemory\ReadProcessMemory*
  - ✓ *CreateRemoteThread\RtlCreateUserThread*

- The injected module is disguised as *shell32.dll*

- Hooks the entry point of *msvcrt.dll* by modifying PEB

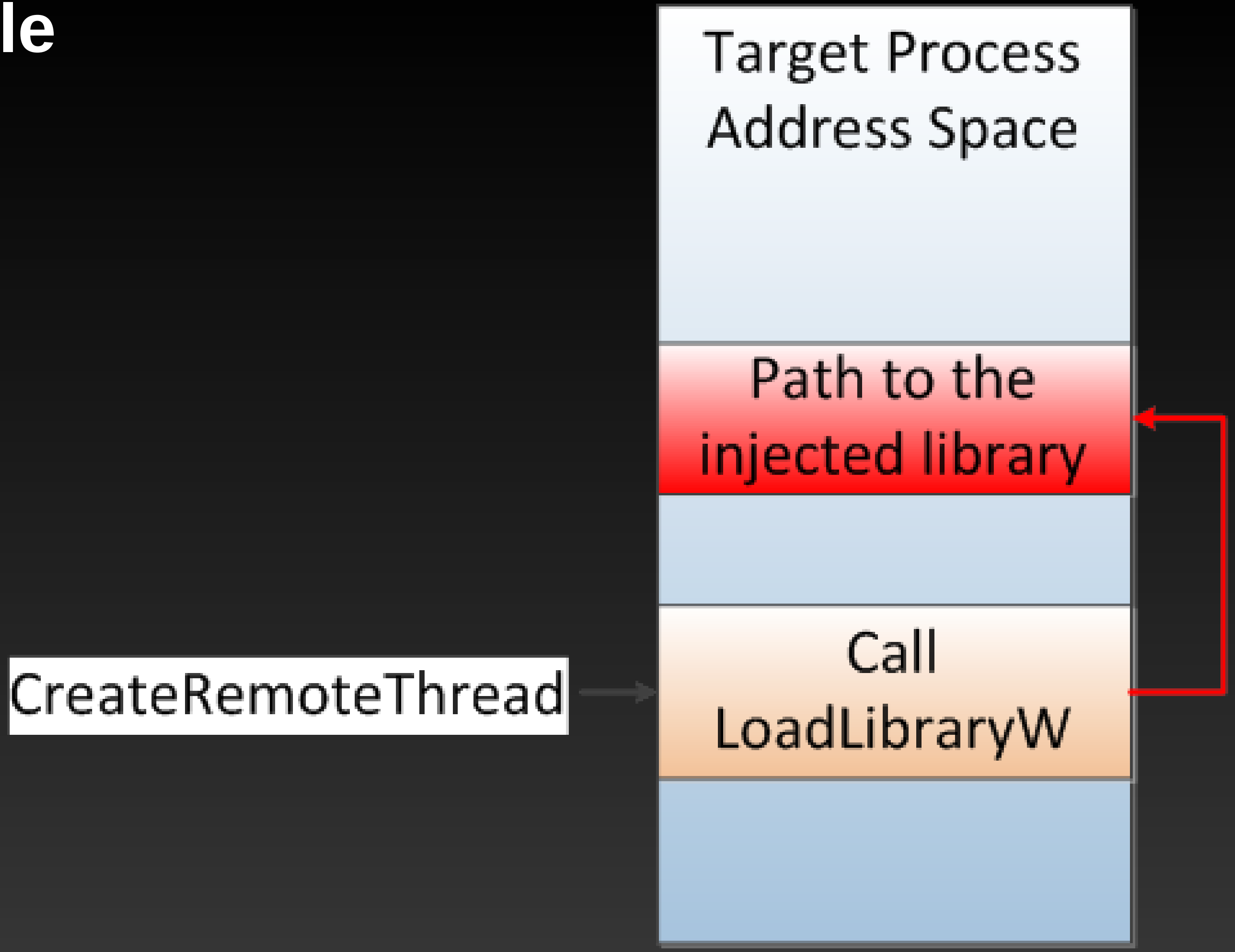
Injection mechanism: Flame

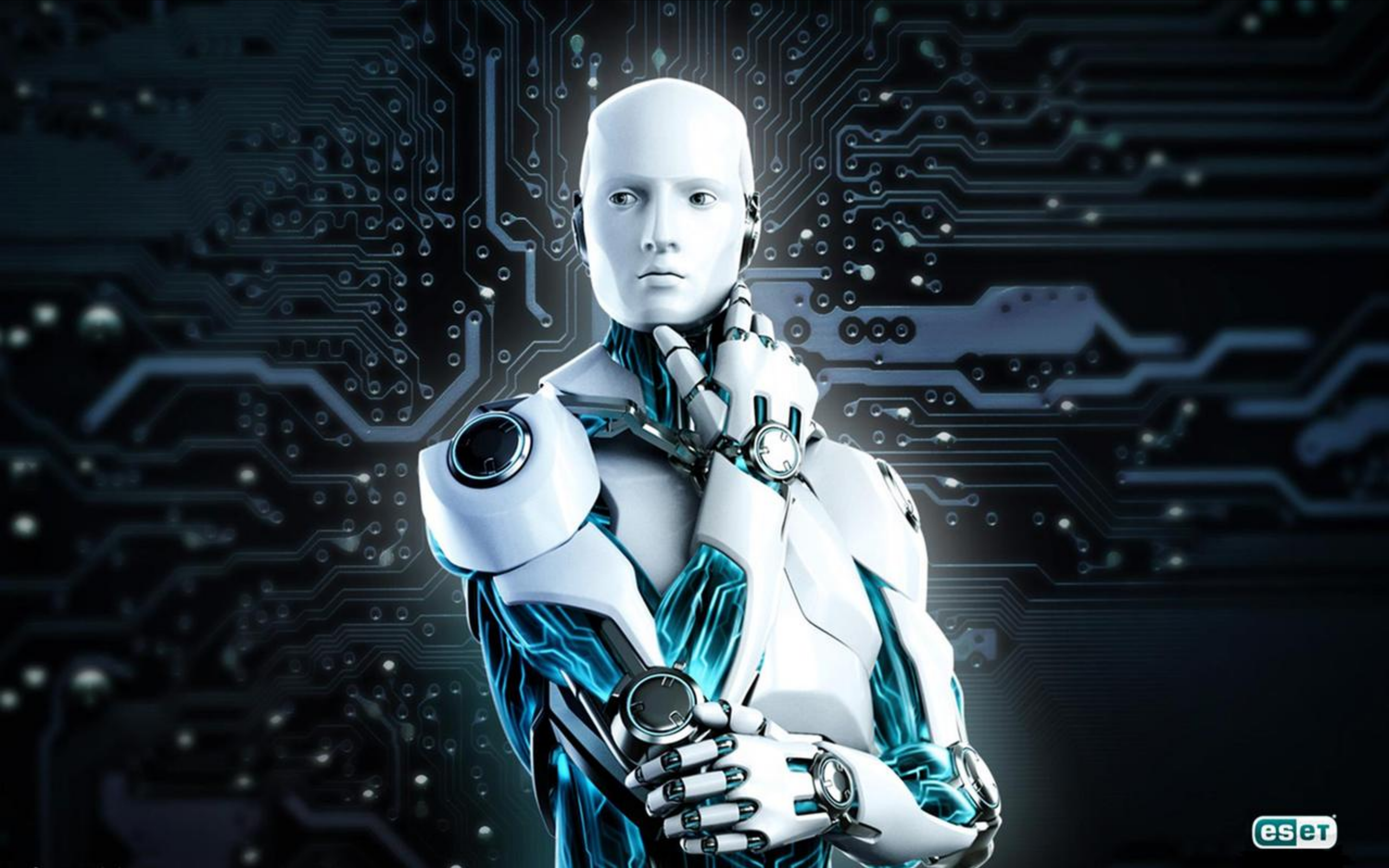


```
if ( LOBYTE(v2->UseShell32) )
{
  hShell32 = (v2->Data1.Api.CreateFileW)(v2->shel32_dll_path, -2147483648, 1, 0, 3, 0, 0);// open shell32.dll image
  BaseAddress = hShell32;
  if ( hShell32 == -1 )
    return -65530;
  Shell32 = (v2->Data1.Api.CreateFileMappingW)(hShell32, 0, 0x1000002u, 0, Size, 0);// create corresponding mapping
  (v2->Data1.Api.CloseHandle)(BaseAddress);
  if ( !Shell32 )
    return -65529;
  BaseAddress = (v2->Data1.Api.MapViewOfFile)(Shell32, 4, 0, 0, Size);// map view of file
  (v2->Data1.Api.CloseHandle)(Shell32);
}
else
{
  BaseAddress = (v2->Data1.Api.VirtualAlloc)(0, pInjectPe->OptionalHeader.SizeOfImage, 4096, 4);
}
if ( !BaseAddress )
  return 0xFFFF0008u;
if ( !(v2->Data1.Api.VirtualProtect)(BaseAddress, Size, PAGE_READWRITE, &v40) )// change page protection
{
  v51 = 0xFFFF0009u;
  goto LABEL_109;
}
(v2->Data1.Api.memset)(BaseAddress, 0, pInjectPe->OptionalHeader.SizeOfImage);// zero memory
(v2->Data1.Api.memcpy)(                         // copy headers
  BaseAddress,
```

Exploit Implementations: Gauss

- The payload is injected into processes from user-mode module




eset

Thank you for your attention!

Eugene Rodionov
rodionov@eset.sk
@vxradius

Aleksandr Matrosov
matrosov@eset.sk
@matrosov