

Способы внедрения в процесс

Часть 1

Способы

- Классический способ - в память хорошего процесса копируется путь к вредной .dll
- PE инъекция - в память хорошего процесса копируется весь вредный файл
- Выкорчевывание (Process Hollowing) - уже загруженный в память хороший процесс перезаписывается полностью вредным
- Похищение потока - **SIR** (**S**uspend, **I**nject, **R**esume) - подмена EIP в существующем хорошем потоке
- SetWindowsHookEx() - перехват вызовов
- Отправление реестра - внедрение в процесс берет на себя сам реестр
- APC
- EWTI с помощью SetWindowLong()
- SHIMS
- Userland rootkit

Классика

AdjustTokenPrivileges(SE_DEBUG_PRIVILEGE)

// Повышаем права

CreateTool32Snapshot()

// Получаем **список** процессов

Process32First()

// Ищем подходящий и получаем **processid**

Process32Next()

OpenProcess()

// Открываем и получаем **handle**

VirtualAllocEx()

// Выделяем память и получаем **адрес**

WriteProcessMemory()

// Пишем путь к .dll в память

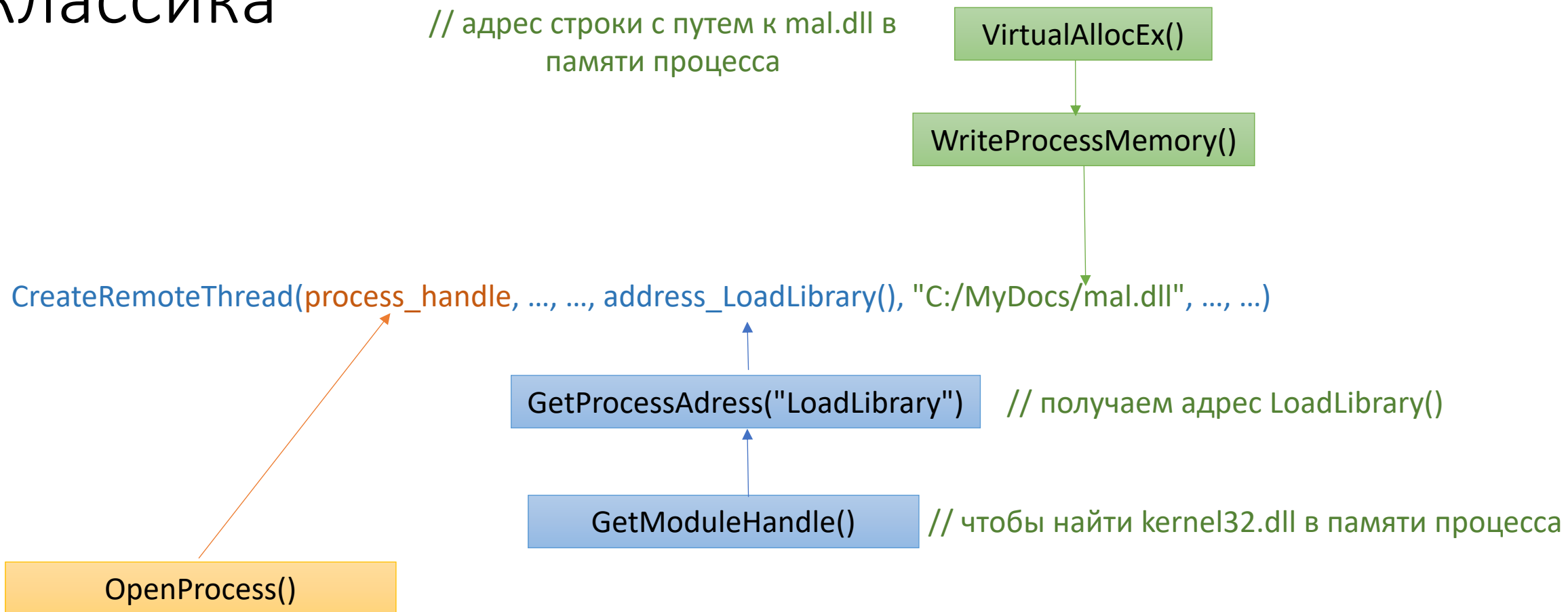
CreateRemoteThread()

NtCreateThreadEx()

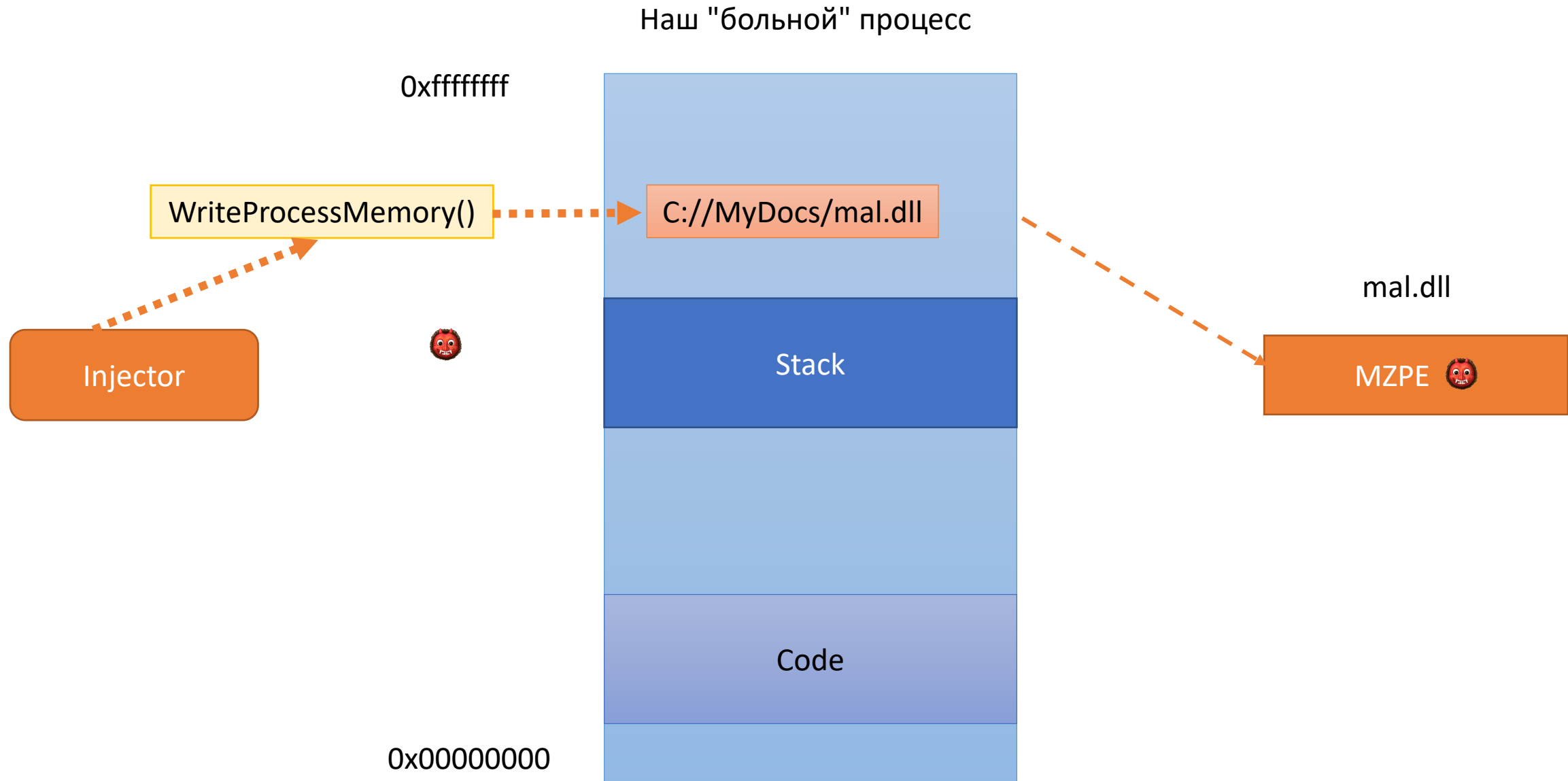
RtlCreateUserThread()

// Вызываем любую из функций и вызываем свою mal.dll

Классика



Классика



Классика

```
call    VirtualAllocEx
push    40h                ; flProtect
push    3000h              ; flAllocationType
push    esi                ; dwSize
push    edi                ; lpAddress
push    ebx                ; hProcess
call    VirtualAllocEx
mov     ebp, eax
test    ebp, ebp
jz      short loc_40AFA4
```

```
lea     eax, [esp+24h+NumberOfBytesWritten]
push    eax                ; lpNumberOfBytesWritten
push    esi                ; nSize
push    0                  ; lpModuleName
call    GetModuleHandleA_0
push    eax                ; lpBuffer
push    edi                ; lpBaseAddress
push    ebx                ; hProcess
call    WriteProcessMemory
cmp     esi, [esp+24h+NumberOfBytesWritten]
ja      short loc_40AFA4
```

```
lea     eax, [esp+24h+ThreadId]
push    eax                ; lpThreadId
push    0                  ; dwCreationFlags
mov     eax, [esp+2Ch+lpParameter]
push    eax                ; lpParameter
mov     eax, [esp+30h+lpStartAddress]
push    eax                ; lpStartAddress
push    0                  ; dwStackSize
push    0                  ; lpThreadAttributes
push    ebx                ; hProcess
call    CreateRemoteThread
push    ebx                ; hObject
call    CloseHandle
mov     [esp+24h+var_1C], ebp
```

```
loc_40AFA4:
mov     eax, [esp+24h+var_1C]
add     esp, 14h
pop     ebp
pop     edi
pop     esi
pop     ebx
retn
sub_40AF08 endp
```

Классика

- Необходимо иметь mal.dll на диске
- Различные механизмы защиты агрядся на CreateRemoteThread()

РЕ инъекция

AdjustTokenPrivileges(SE_DEBUG_PRIVILEGE)

// Повышаем права

CreateTool32Snapshot()

// Получаем **список** процессов

Process32First()

Process32Next()

// Ищем подходящий и получаем **processid**

OpenProcess()

// Открываем и получаем **handle**

VirtualAllocEx()

// Выделяем память и получаем **адрес**

WriteProcessMemory()

// Пишем .dll в память процесса целиком

Custom_FixRelocationTable()

// правим таблицы релокаций

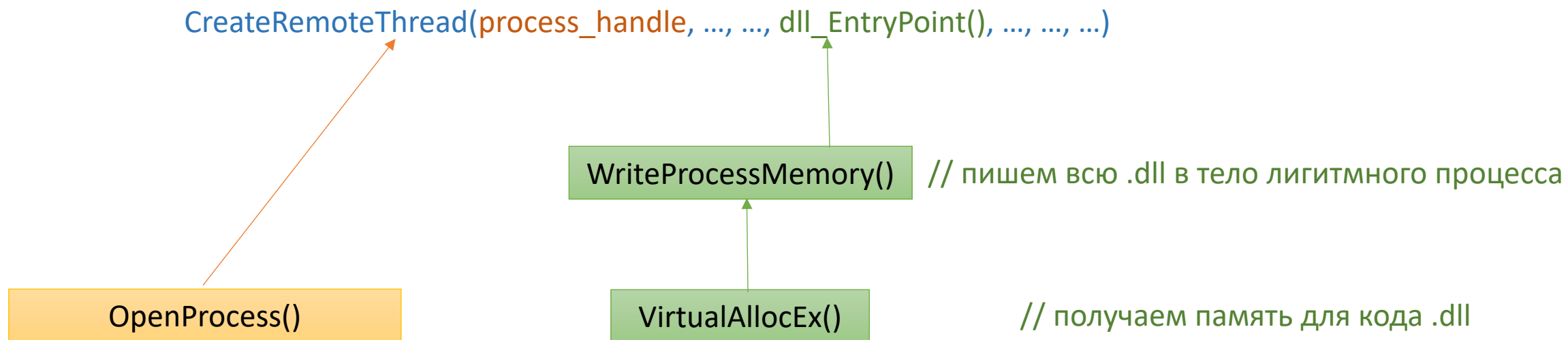
CreateRemoteThread()

NtCreateThreadEx()

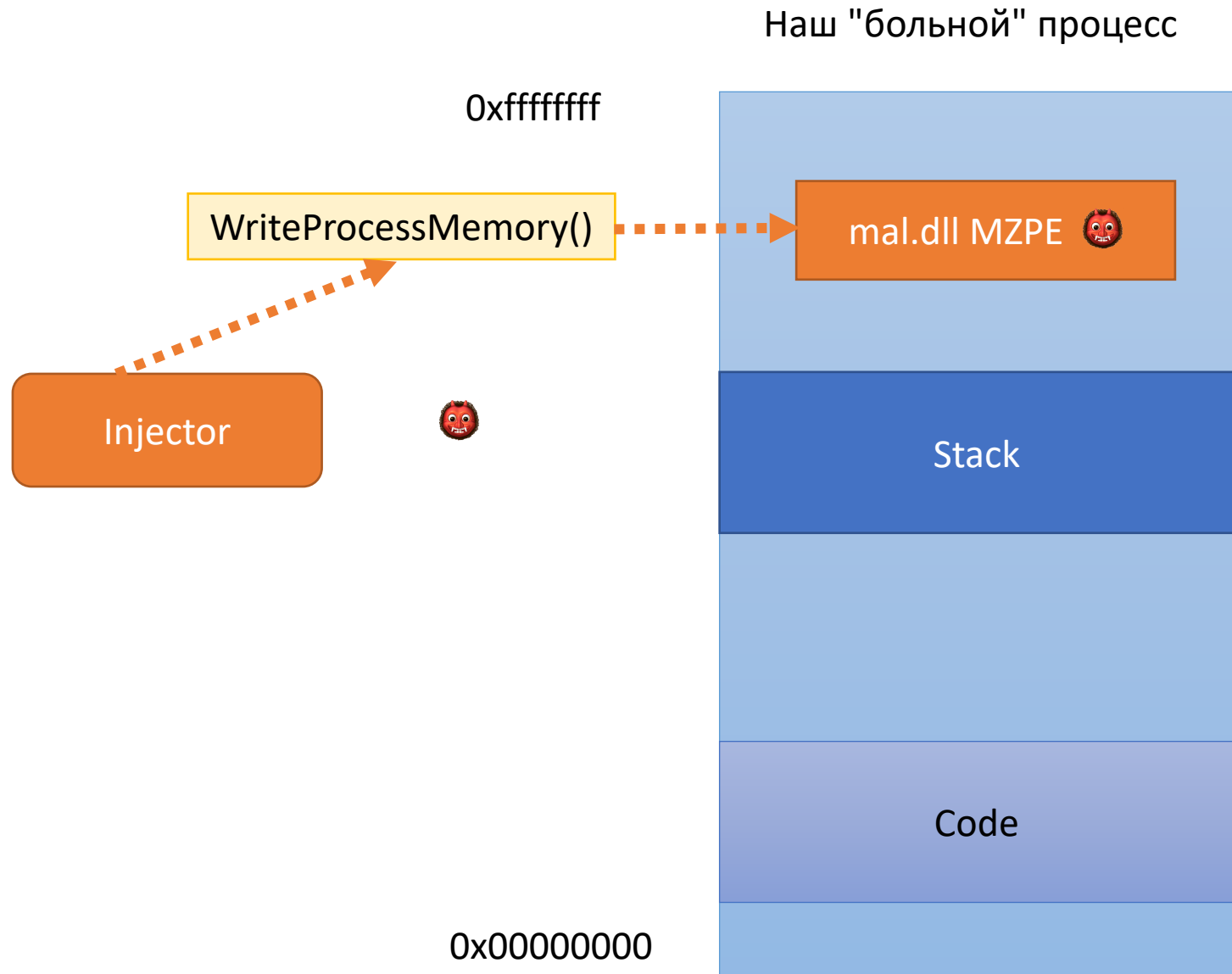
RtlCreateUserThread()

// Вызываем любую из функций и вызываем свою mal.dll

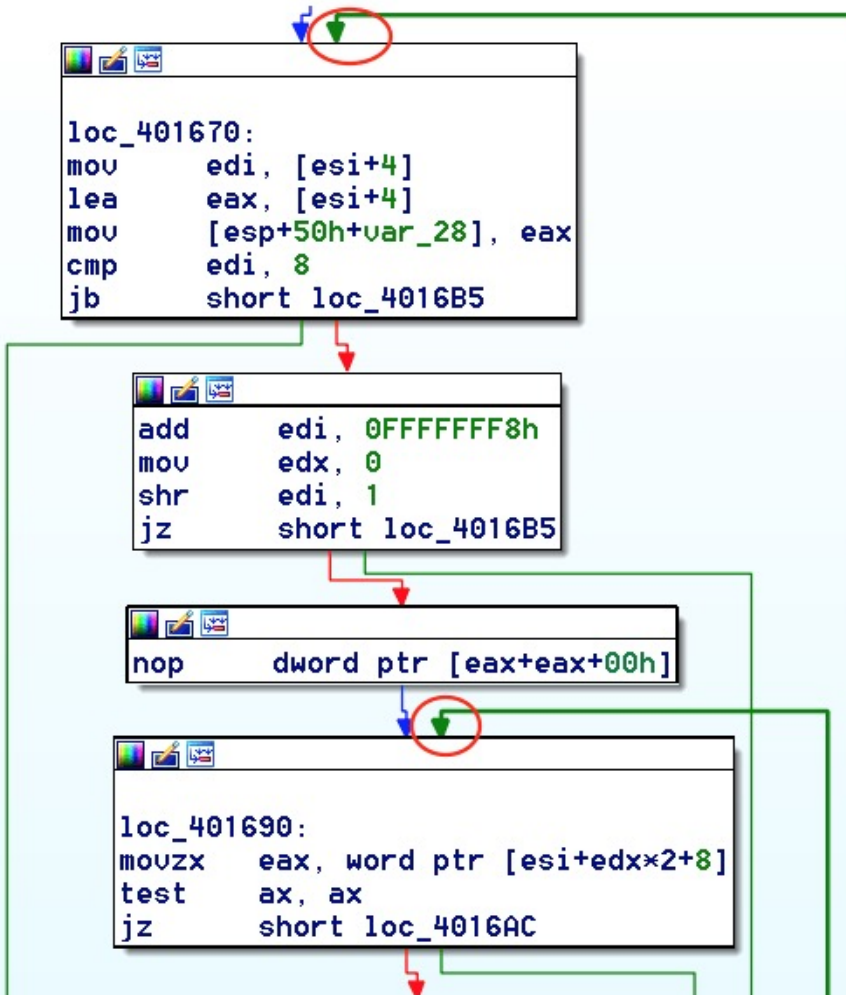
РЕ инъекция



РЕ инъекция



РЕ инъекция

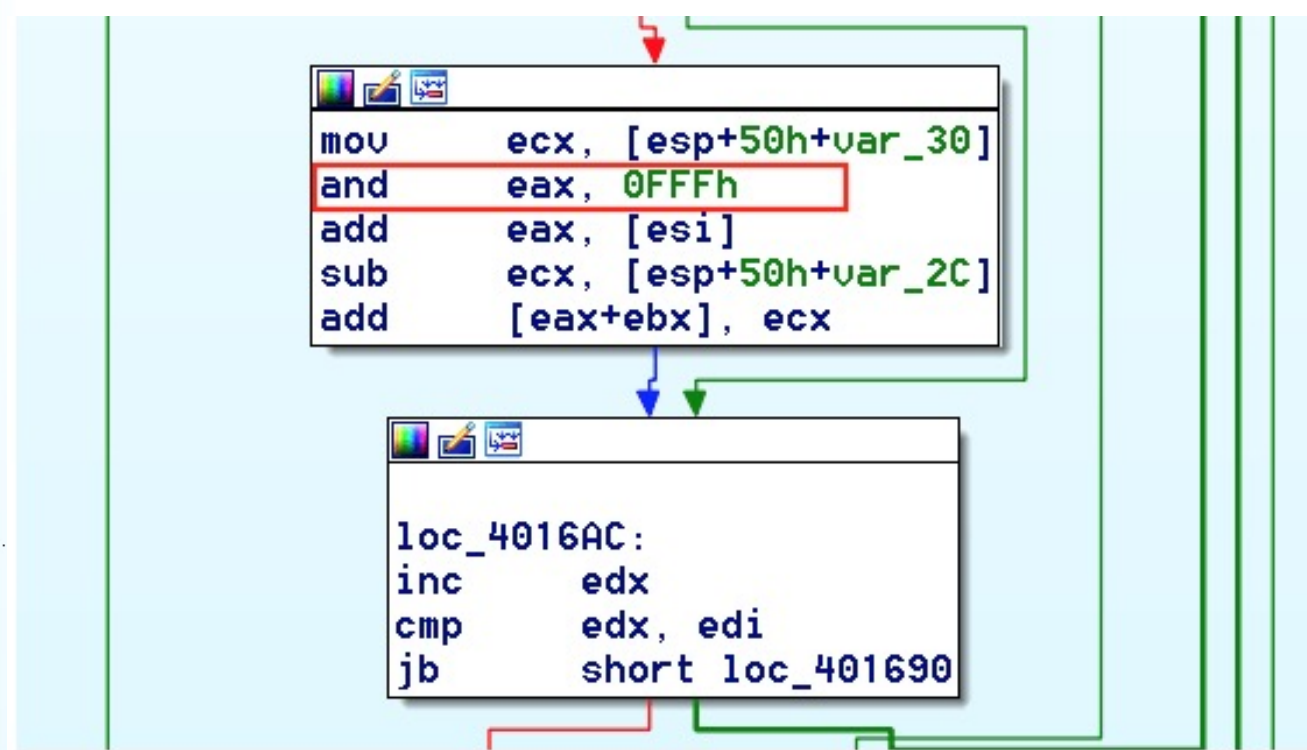


```
loc_401670:  
mov     edi, [esi+4]  
lea     eax, [esi+4]  
mov     [esp+50h+var_28], eax  
cmp     edi, 8  
jb      short loc_4016B5
```

```
add     edi, 0FFFFFFF8h  
mov     edx, 0  
shr     edi, 1  
jz      short loc_4016B5
```

```
nop     dword ptr [eax+eax+00h]
```

```
loc_401690:  
movzx   eax, word ptr [esi+edx*2+8]  
test    ax, ax  
jz      short loc_4016AC
```



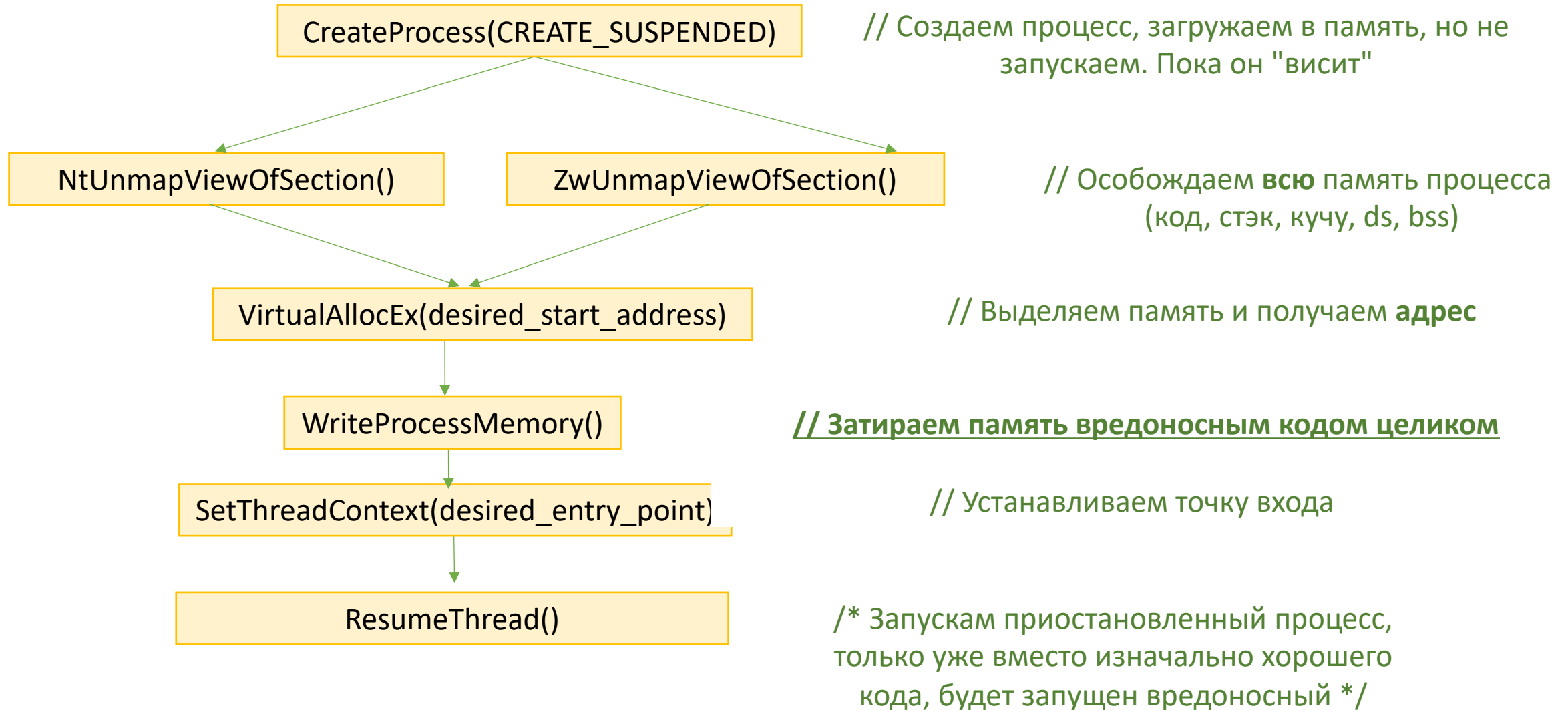
```
mov     ecx, [esp+50h+var_30]  
and     eax, 0FFFh  
add     eax, [esi]  
sub     ecx, [esp+50h+var_2C]  
add     [eax+ebx], ecx
```

```
loc_4016AC:  
inc     edx  
cmp     edx, edi  
jb      short loc_401690
```

РЕ инъекция

- Нет необходимости иметь mal.dll на диске
- Остается проблема с API функциями CreateRemoteThread() и LoadLibrary(), то есть техника не скрытая

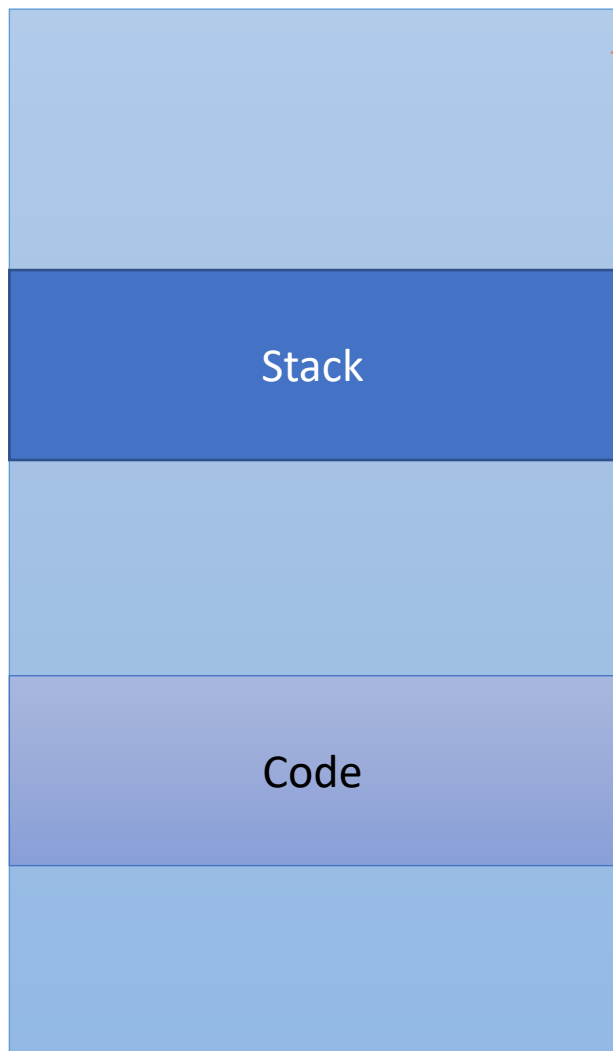
Process Hollowing



Process Hollowing

Создан хороший здоровый процесс, а...

0xffffffff

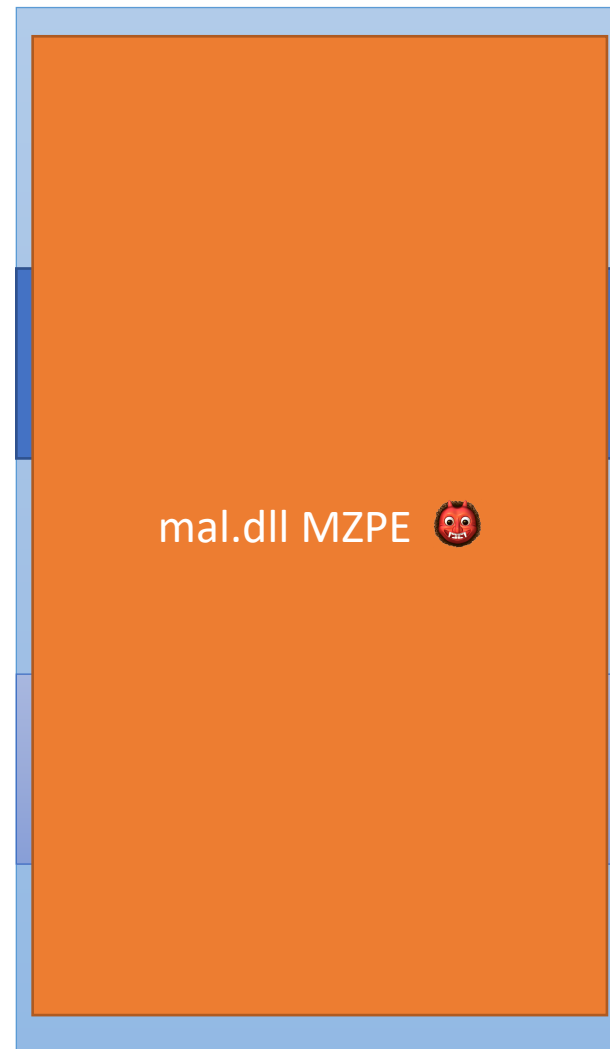


WriteProcessMemory()

Injector


...запущен плохой и вредный

0xffffffff



Process Hollowing

```
call     @System@@@FillChar$qqrpv1c ; System::__linkproc__ FillChar(void *,int,char)
mov     [ebp+StartupInfo.cb], 44h
lea     eax, [ebp+ProcessInformation]
push    eax                ; lpProcessInformation
lea     eax, [ebp+StartupInfo]
push    eax                ; lpStartupInfo
push    0                  ; lpCurrentDirectory
push    0                  ; lpEnvironment
push    4                  ; dwCreationFlags    Process created in suspended state
push    0                  ; bInheritHandles
push    0                  ; lpThreadAttributes
push    0                  ; lpProcessAttributes
mov     eax, [ebp+var_8]
call     @System@@@LStrToPChar$qqrx17System@AnsiString ; System::__linkproc__ LStrToPChar(System::AnsiString)
push    eax                ; lpCommandLine
push    0                  ; lpApplicationName
call     CreateProcessA
test    eax, eax
jz      loc_45B12C
```



```
lea     eax, [ebp+lpAddress]
call    sub_45AD34
mov     [ebp+lpContext], eax
cmp     [ebp+lpContext], 0
jz      loc_45AFF2
```

Process Hollowing

```
nov    eax, [ebp+lpContext]
nov    dword ptr [eax], 10007h
nov    eax, [ebp+lpContext]
push   eax                ; lpContext
nov    eax, [ebp+ProcessInformation.hThread]
push   eax                ; hThread
call   GetThreadContext
test   eax, eax
jz     loc_45AFE2
```

```
lea    eax, [ebp+NumberOfBytesRead]
push   eax                ; lpNumberOfBytesRead
push   4                  ; nSize
lea    eax, [ebp+Buffer]
push   eax                ; lpBuffer
mov     eax, [ebp+lpContext]
mov     eax, [eax+0A4h]
add     eax, 8
push   eax                ; lpBaseAddress
mov     eax, [ebp+ProcessInformation.hProcess]
push   eax                ; hProcess
call   ReadProcessMemory
mov     eax, [edi+34h]
cmp     eax, [ebp+Buffer]
jnz     short loc_45AF27
```

```
nov    eax, [edi+34h]
push   eax                ; BaseAddress
nov    eax, [ebp+ProcessInformation.hProcess]
push   eax                ; ProcessHandle
call   NtUnmapViewOfSection
test   eax, eax
jnz     short loc_45AF0C
```

Hollowing out the process

Process Hollowing

- Нет необходимости иметь mal.dll на диске
- Остается проблема с API функциями CreateRemoteThread() и LoadLibrary(), то есть техника не скрытая

SIR

AdjustTokenPrivileges(SE_DEBUG_PRIVILEGE)

// Повышаем права

CreateTool32Snapshot()

// Получаем **список** процессов

Thread32First()

// Ищем подходящий поток и получаем **threadid**

Thread32Next()

// Открываем и получаем **handle**

OpenThread()

// Ставим исполнение потока на паузу

SuspendThread()

// Выделяем память и получаем **адрес**

VirtualAllocEx()

// Пишем путь к .dll + LoadLibrary()_address в
память или шелкод

WriteProcessMemory()

SetThreadContext()

// меняем значение EIP регистра, устанавливая
на шелкод или на EP .dll

ResumeThread()

// запускаем вредоносный процесс

SIR

Наш "больной" процесс

WriteProcessMemory()

Injector

2

SetThreadContext(_ARM64_NT_CONTEXT)

EIP = 0xffff1234

3

Приостановленный поток 1

Поток 2

Поток 3

Stack

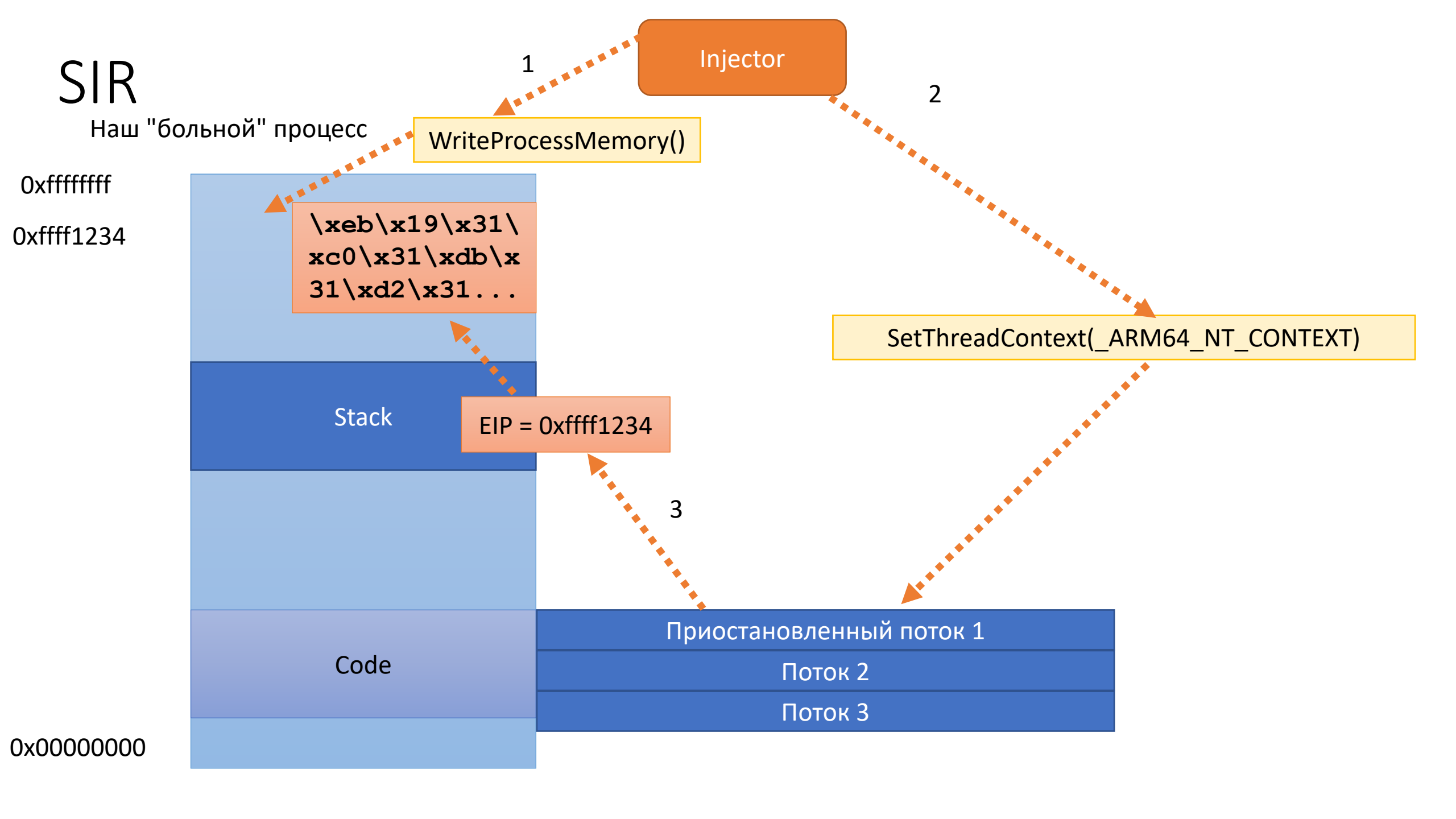
Code

\xeb\x19\x31\x
c0\x31\xdb\x
31\xd2\x31...

0xffffffff

0xffff1234

0x00000000



SIR

```
call    OpenThread
mov     [ebp+hThread], eax
cmp     [ebp+hThread], 0
jz      loc_503053
```

```
mov     eax, [ebp+hThread]
push    eax ; hThread
call    SuspendThread
mov     [ebp+var_0C], 10007h
lea     eax, [ebp+var_0C]
push    eax ; lpContext
mov     eax, [ebp+hThread]
push    eax ; hThread
call    GetThreadContext
mov     eax, [ebp+var_24]
mov     [ebp+var_104], eax
mov     eax, [ebp+var_3C]
mov     [ebp+var_100], eax
push    offset aLoadlibrarya_1 ; "LoadLibraryA"
push    offset aKernel32_dll_3 ; "kernel32.dll"
call    GetModuleHandleA
push    eax ; hModule
call    GetProcAddress
mov     [ebp+var_FC], eax
mov     edx, [ebp+var_8] ; unsigned int
mov     eax, [ebp+ProcessHandle] ; this
call    @Advapihook@InjectString$qqruipc ; Advapihook::InjectString(uint,char *)
mov     [ebp+var_F8], eax
cmp     [ebp+var_F8], 0
jz      short loc_503053
```

SIR

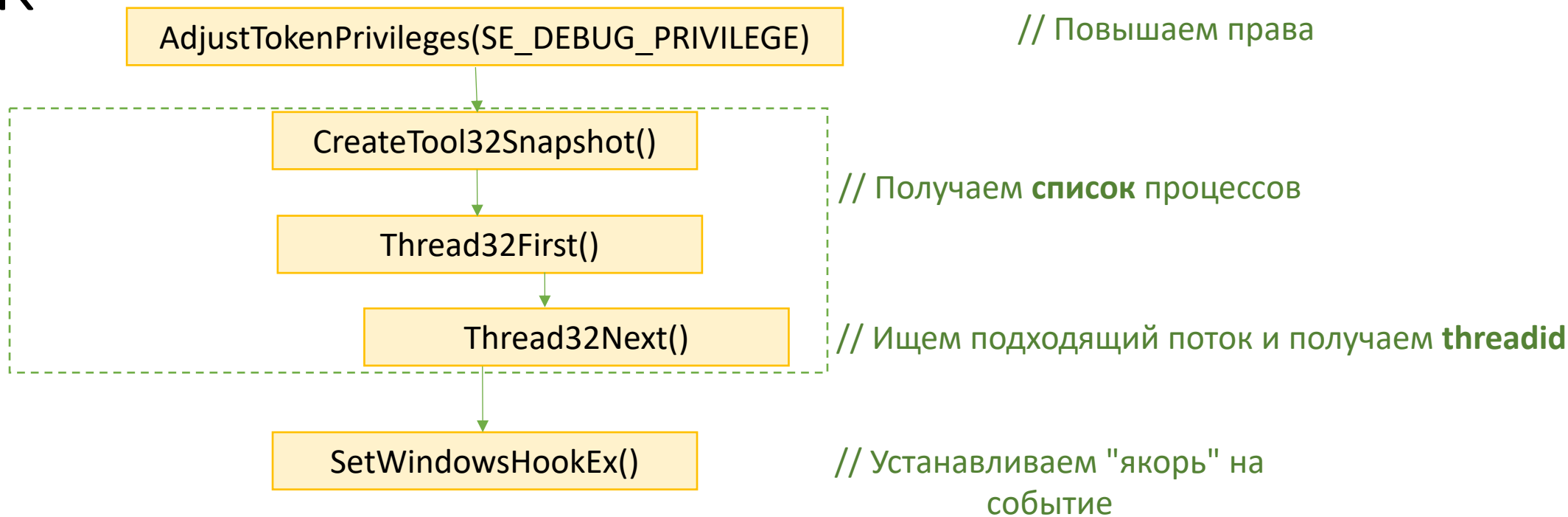
```
lea     edx, [ebp+var_104] ; unsigned int
mov     ecx, 10h           ; void *
mov     eax, [ebp+ProcessHandle] ; this
call    @Advapihook@InjectMemory$qqruipvui ; Advapihook::InjectMemory(uint,void *,uint)
mov     [ebp+var_3C], eax
mov     eax, offset sub_502F28 ; this
call    @Advapihook@SizeOfProc$qqrpv ; Advapihook::SizeOfProc(void *)
mov     ecx, eax           ; void *
mov     edx, offset sub_502F28 ; unsigned int
mov     eax, [ebp+ProcessHandle] ; this
call    @Advapihook@InjectMemory$qqruipvui ; Advapihook::InjectMemory(uint,void *,uint)
mov     [ebp+var_24], eax
lea     eax, [ebp+var_DC]
push    eax                ; lpContext
mov     eax, [ebp+hThread]
push    eax                ; hThread
call    SetThreadContext
mov     eax, [ebp+hThread]
push    eax                ; hThread
call    ResumeThread
mov     [ebp+var_9], 1
```

```
loc_503053:
mov     al, [ebp+var_9]
mov     esp, ebp
pop     ebp
retn
@Advapihook@InjectDllAlt$qqruipc endp
```

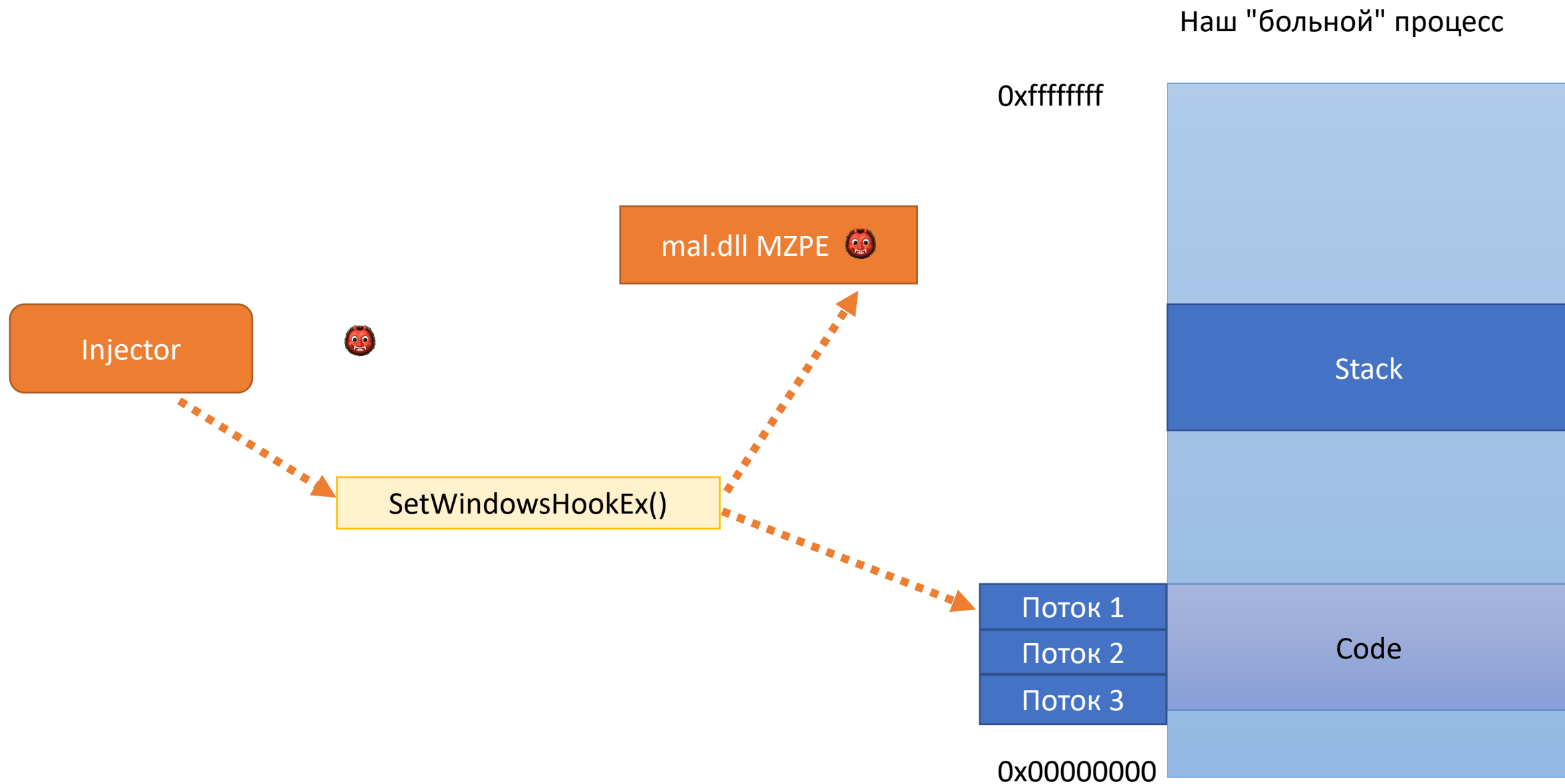
SIR

- Не так "шумно", потому что не создается процесс, извращения творятся с уже запущенным
- Проблема: если вдруг процесс приостановлен во время системного вызова - программа крашнется. Для этого, более "умные" трояны могут проверять значение EIP и отложить инъект, если его значение находится в пределах ntdll.dll

HOOK



HOOK



HOOK

//ищем адрес своей
вредоносной функции

SetWindowsHookEx(event_type, target_mal_function_address, module, thread_id)

LoadLibrary()

GetProcAddress()

WH_KEYBOARD
WH_MOUSE
CBT ... etc

Поток, в который
внедряем код

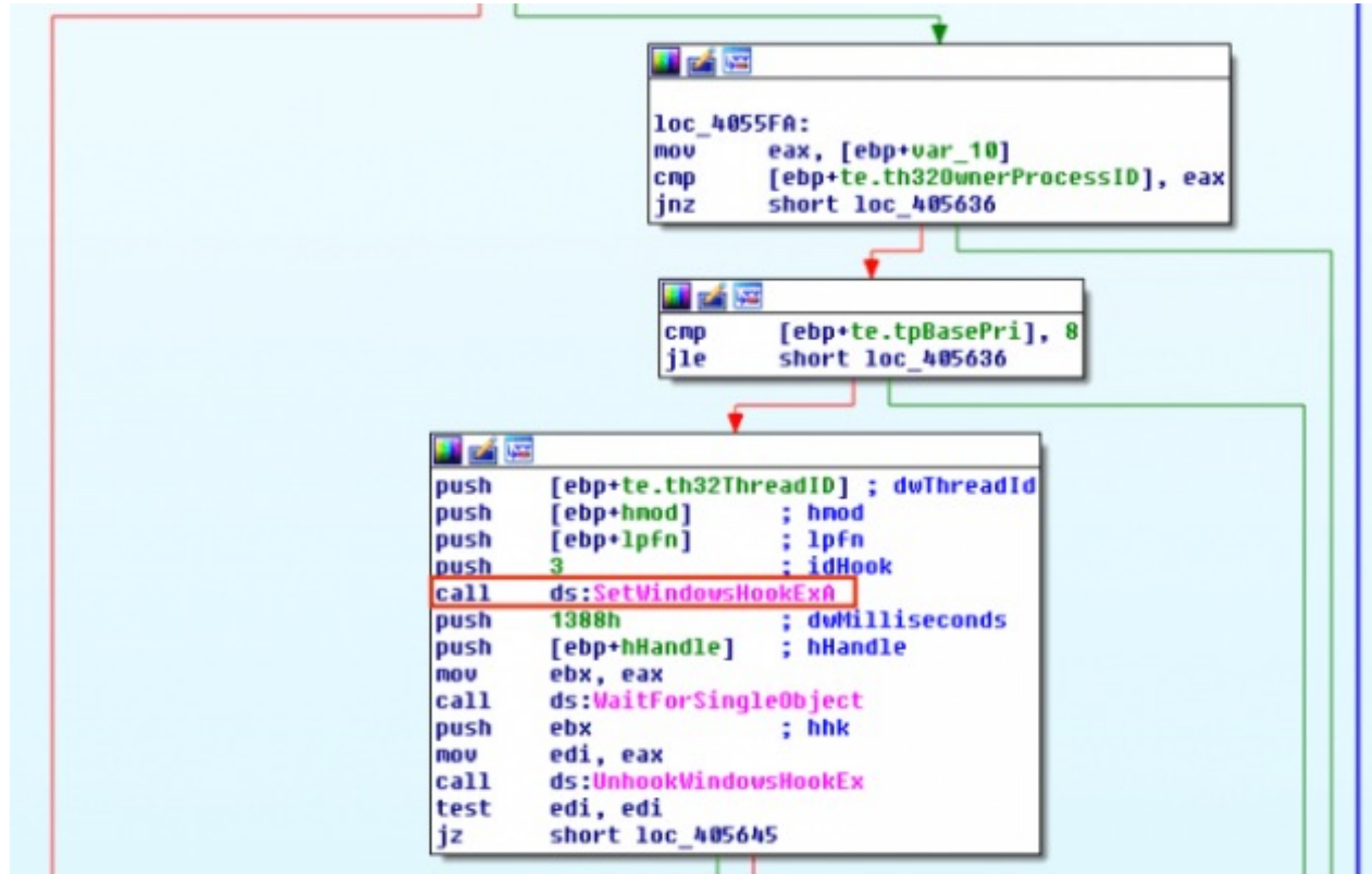
// если указать "0", тогда все потоки будут
реагировать на событие, но это шумно

HOOK

```
loc_405588:  
lea     eax, [ebp+pFileName]  
push    eax ; lpLibFileName  
call    ds:LoadLibraryV  
push    offset aMyprocedure ; "MyProcedure"  
push    eax ; hModule  
mov     [ebp+hmod], eax  
call    ds:GetProcAddress  
push    esi ; th32ProcessID  
push    4 ; dwFlags  
mov     [ebp+lpfn], eax  
call    ds:CreateToolhelp32Snapshot  
mov     esi, ds:Thread32Next  
lea     ecx, [ebp+te]  
push    ecx  
mov     [ebp+lpData], eax  
mov     [ebp+te.dwSize], 1Ch  
push    eax  
jnp     short loc_40563D
```

```
loc_40563D:  
call    esi ; Thread32Next  
test    eax, eax  
jnz     short loc_4055FA
```

HOOK



REGISTRY

HKLM/Software/Microsoft/WindowsNT/CurrentVersion/Windows/Appinit_Dlls

HKLM/Software/Wow6432Node/Microsoft/WindowsNT/CurrentVersion/Windows/Appinit_Dlls

// при загрузке User32.dll будет загружаться mal.dll

HKLM/System/CurrentControlSet/Control/Session Manager/AppCertDlls

/* при вызовах CreateProcess(), CreateProcessAsUser(), CreateProcessWithLogonW(),
CreateProcessWithTokenW(), WinExec() будет вызываться mal.dll */

HKLM/Software/Wow6432Node/Microsoft/WindowsNT/CurrentVersion/image file execution options

/* IFEO - обычно используется для аттача дебаггера. Меняется поле "Debugger Value" */

REGISTRY

```
push    0                ; lpClass
push    0                ; Reserved
push    offset aSoftwareMicr_0 ; "SOFTWARE\\Microsoft\\Windows NT\\Curren"...
push    80000002h        ; hKey
call    RegCreateKeyExA
test    eax, eax
jnz     short loc_403DD2
```

```
lea     ebx, [esp+1018h+Dst]
push    ebx              ; lpString
call    lstrlenA
inc     eax
push    eax              ; cbData
push    ebx              ; lpData
push    1                ; dwType
push    0                ; Reserved
push    offset aAppinit_dlls ; "Appinit_DLLs"
mov     eax, [esp+102Ch+phkResult]
push    eax              ; hKey
call    RegSetValueExA
mov     eax, [esp+1018h+phkResult]
push    eax              ; hKey
call    RegCloseKey
mov     bl, 1
```

```
loc_403DD2:
mov     eax, ebx
add     esp, 1010h
pop     esi
pop     ebx
retn
sub_403C80 endp
```

REGISTRY

- Сам процесс инжекта наиболее незаметен
- Однако остается проблема присутствия .dll на диске
- Остаются "ошметки в реестре"

ИСТОЧНИКИ

Ten Process Injection Techniques: A Technical Survey Of Common And Trending Process Injection Techniques:

<https://www.endgame.com/blog/technical-blog/ten-process-injection-techniques-technical-survey-common-and-trending-process>