



POSITIVE  
TECHNOLOGIES

# Распаковка исполняемых файлов: статический и динамический подход

**Александр Лаухин**  
Эксперт PT ESC

**Алексей Вишняков**  
Эксперт PT ESC



[ptsecurity.com](https://ptsecurity.com)



# Содержание



- Коротко о PT Sandbox
- Portable Executable
- Упаковщики исполняемых файлов
- Динамическая распаковка
- Статическая распаковка
- Материалы и ссылки

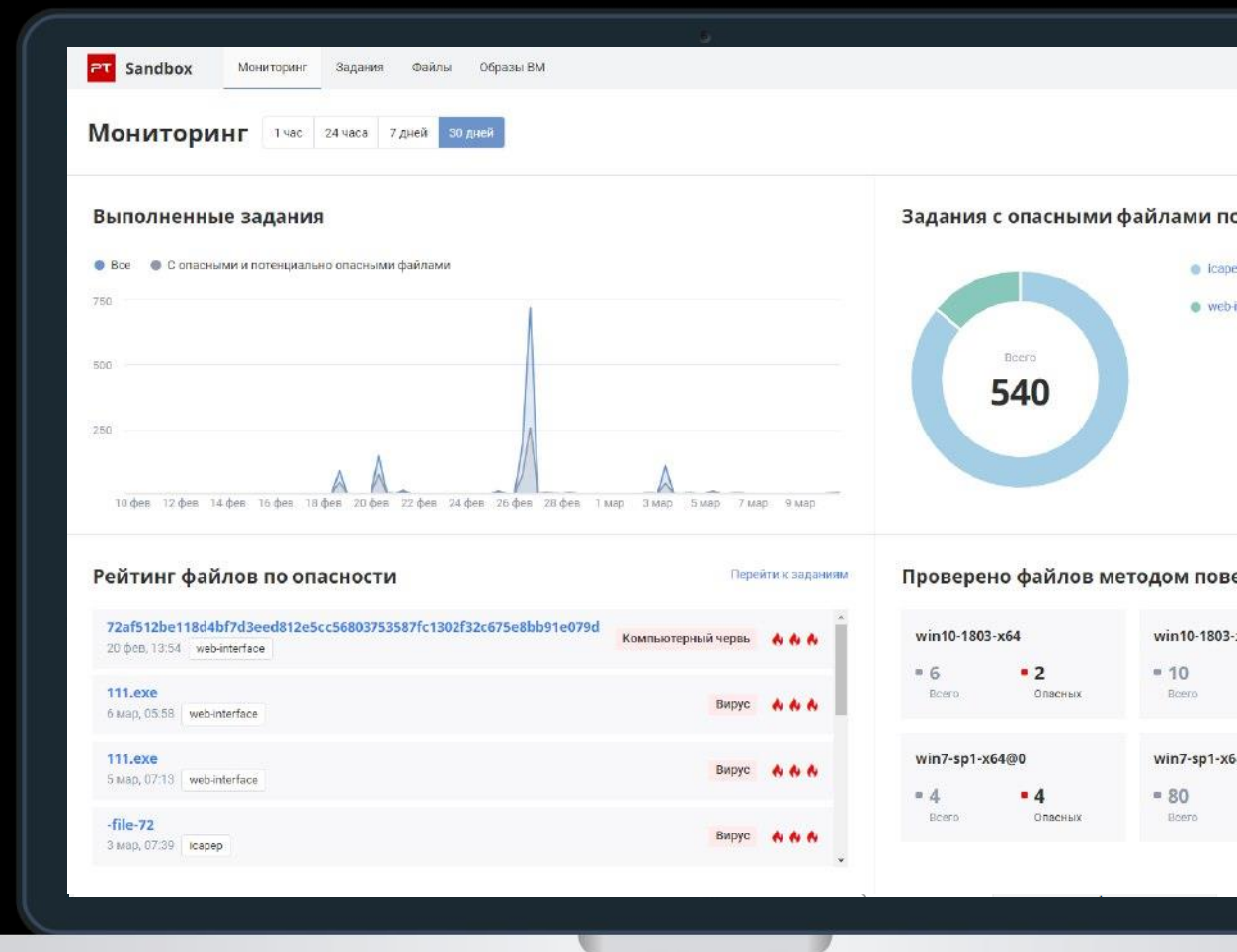


# PT Sandbox



Песочница для защиты от целевых и массовых атак с применением неизвестного вредоносного ПО и угроз нулевого дня.

- Обеспечивает комплексный анализ файлов и трафика (включая зашифрованный)
- Поддерживает гибкую настройку виртуальных сред и защищена от техник обхода песочниц
- Использует уникальные и наиболее актуальные знания для выявления угроз





# Содержание



- Коротко о PT Sandbox
- **Portable Executable**
- Упаковщики исполняемых файлов
- Динамическая распаковка
- Статическая распаковка
- Материалы и ссылки



# Portable Executable

PT



Форматы: PE32 и PE32+

Расширения: exe, dll, osx, sys, scr, ...

```
.01000000: 4D 5A 90 00-03 00 00 00-04 00 00 00-FF FF 00 00 MZÉ ♥ ♦
.01000010: B8 00 00 00-00 00 00 00-40 00 00 00-00 00 00 00 ґ @
.01000020: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.01000030: 00 00 00 00-00 00 00 00-00 00 00 00-E0 00 00 00 α
.01000040: 0E 1F BA 0E-00 B4 09 CD-21 B8 01 4C-CD 21 54 68 ґ||ґ ґo=!ґ@L=!Th
.01000050: 69 73 20 70-72 6F 67 72-61 6D 20 63-61 6E 6E 6F is program canno
.01000060: 74 20 62 65-20 72 75 6E-20 69 6E 20-44 4F 53 20 t be run in DOS
.01000070: 6D 6F 64 65-2E 0D 0D 0A-24 00 00 00-00 00 00 00 mode.ґ||ґ$
.01000080: 1D 83 C3 D1-59 E2 AD 82-59 E2 AD 82-59 E2 AD 82 ґâ|ґґґ;éґґ;é
.01000090: 50 9A 38 82-5A E2 AD 82-50 9A 2E 82-40 E2 AD 82 PÛ8éZґ;éPÛ.é@ґ;é
.010000A0: 50 9A 3E 82-4A E2 AD 82-59 E2 AC 82-29 E2 AD 82 PÛ>éJґ;éґґ%é)ґ;é
.010000B0: 50 9A 29 82-5F E2 AD 82-50 9A 39 82-58 E2 AD 82 PÛ)é_ґ;éPÛ9éXґ;é
.010000C0: 50 9A 3C 82-58 E2 AD 82-52 69 63 68-59 E2 AD 82 PÛ<éXґ;éRichґґ;é
.010000D0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.010000E0: 50 45 00 00-4C 01 03 00-18 C0 5B 4A-00 00 00 00 PE Lо♥↑L[ґ
.010000F0: 00 00 00 00-E0 00 02 01-0B 01 09 00-00 50 00 00 α 00đ00 P
.01000100: 00 10 00 00-00 90 00 00-A0 E5 00 00-00 A0 00 00 ► É áσ á
```



# PE. Optional Header

PT



-----OPTIONAL_HEADER-----		
[IMAGE_OPTIONAL_HEADER]		
0x0	Magic:	0x10B
0x10	AddressOfEntryPoint:	0xE5A0
...		
0x1C	ImageBase:	0x1000000
...		
0x46	DllCharacteristics:	0x8140
...		
-----Directories-----		
[IMAGE_DIRECTORY_ENTRY_EXPORT]		
0x60	VirtualAddress:	0x0
0x64	Size:	0x0
[IMAGE_DIRECTORY_ENTRY_IMPORT]		
0x68	VirtualAddress:	0xF774
0x6C	Size:	0x1E4
[IMAGE_DIRECTORY_ENTRY_RESOURCE]		
0x74	VirtualAddress:	0xF000
0x78	Size:	0x774
...		
[IMAGE_DIRECTORY_ENTRY_BASERELOC]		
0x88	VirtualAddress:	0xF958
0x8C	Size:	0x10
...		

1

2

3



# PE. Заголовки секций



-----PE Sections-----

[IMAGE\_SECTION\_HEADER]

```
0x0  Name: .text
0x8  VirtualSize: 0x9000 1
0xC  VirtualAddress: 0x1000
0x10 SizeOfRawData: 0x4400 2
0x14 PointerToRawData: 0x400
...
0x24 Characteristics: 0xE0000060
Flags:
IMAGE_SCN_CNT_CODE
IMAGE_SCN_CNT_INITIALIZED_DATA
IMAGE_SCN_MEM_EXECUTE
IMAGE_SCN_MEM_READ 3
IMAGE_SCN_MEM_WRITE
```



# PE. Таблица импорта

Таблица импорта состоит из массива дескрипторов импорта.

Дескриптор импорта содержит:

1. Адрес таблицы имен импорта
2. Адрес названия библиотеки
3. Адрес таблицы адресов импорта

Таблица имен импорта содержит порядковый номер функции в библиотеке (ordinal) или адрес её названия.

Таблица адресов импорта (IAT) содержит абсолютные адреса импортируемых функций.

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {  
    union {  
        DWORD Characteristics;  
        1 DWORD OriginalFirstThunk;  
    } DUMMYUNIONNAME;  
    DWORD TimeDateStamp;  
    DWORD ForwarderChain;  
    2 DWORD Name;  
    3 DWORD FirstThunk;  
} IMAGE_IMPORT_DESCRIPTOR, *PIMAGE_IMPORT_DESCRIPTOR;
```



# PE. Таблица релокаций

PE файлы:

- могут содержать абсолютные адреса
- компилируются с адресами для предпочтительного адреса загрузки (ImageBase)

При загрузке PE файла по другому адресу необходимо **перебазирование**. Для этих целей используется таблица релокаций.

Она состоит из блоков, содержащих:

1. Позиции абсолютных адресов в файле
2. Тип релокации

RELOCATION VIEWER					
Size of Relocations Table: 0000059Ch			Block Number: 0 <b>2</b>		
RVA	Items	Owner		Offset	Type
01001000h	38	.text	<b>1</b>	010011B0h	HIGHLOW (3)
01002000h	76	.text		010011BCh	HIGHLOW (3)
01003000h	54	.text		010011C0h	HIGHLOW (3)
01004000h	190	.text		0100134Ch	HIGHLOW (3)
01005000h	110	.text		01001350h	HIGHLOW (3)
01006000h	94	.text		010014BCh	HIGHLOW (3)
01007000h	54	.text		010014C0h	HIGHLOW (3)
01008000h	32	.text		01001527h	HIGHLOW (3)
01009000h	34	.text		0100154Ch	HIGHLOW (3)
				01001559h	HIGHLOW (3)
				01001565h	HIGHLOW (3)
				01001572h	HIGHLOW (3)
				0100157Eh	HIGHLOW (3)
				0100158Dh	HIGHLOW (3)
				01001599h	HIGHLOW (3)
				010015A4h	HIGHLOW (3)
				010015BBh	HIGHLOW (3)
				010015CEh	HIGHLOW (3)
				010015D6h	HIGHLOW (3)
				010015DEh	HIGHLOW (3)
				010015E6h	HIGHLOW (3)
				010015F2h	HIGHLOW (3)
				01001AEBh	HIGHLOW (3)
				01001B32h	HIGHLOW (3)
				01001B99h	HIGHLOW (3)
				01001BA0h	HIGHLOW (3)
				01001BBDh	HIGHLOW (3)
				01001BC7h	HIGHLOW (3)



# Содержание

The logo consists of the letters 'PT' in a stylized, bold, sans-serif font, positioned within a white square.

- Коротко о PT Sandbox
- Portable Executable
- **Упаковщики исполняемых файлов**
- Динамическая распаковка
- Статическая распаковка
- Материалы и ссылки



# Упаковщики исполняемых файлов



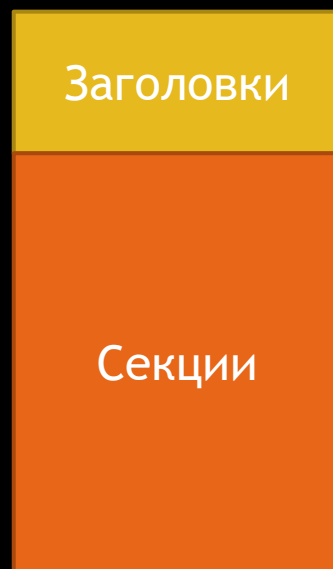
## Принцип работы:

- Сжатие секций
- Прикрепление загрузчика

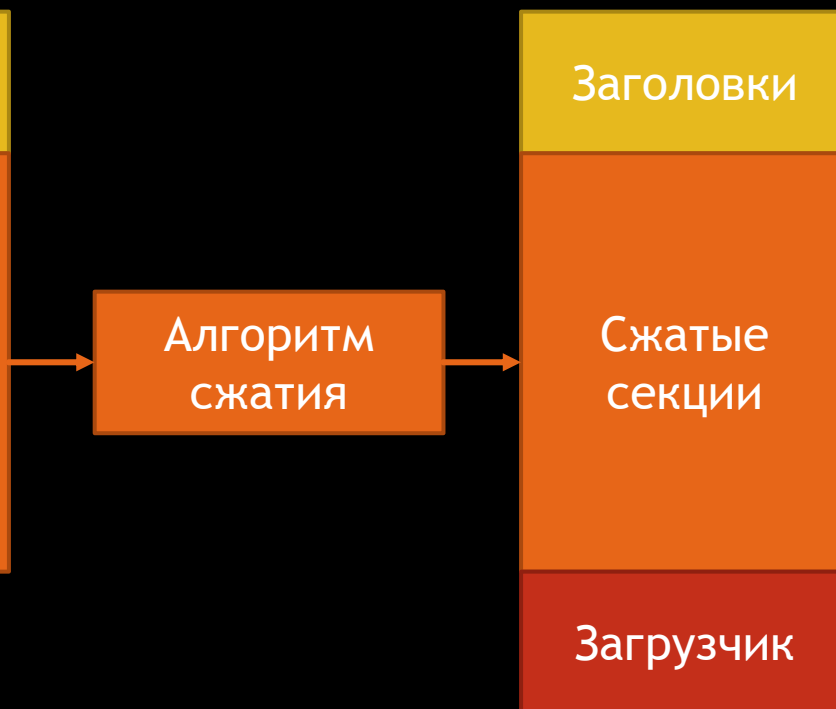
## Зачем применяют:

- Защита от реверс-инжиниринга
- Уменьшение размера файла
- Соккрытие вредоносного кода

### EXECUTABLE



### PACKED EXECUTABLE



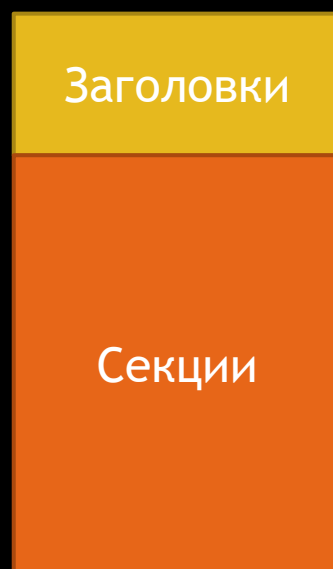


# Загрузчик

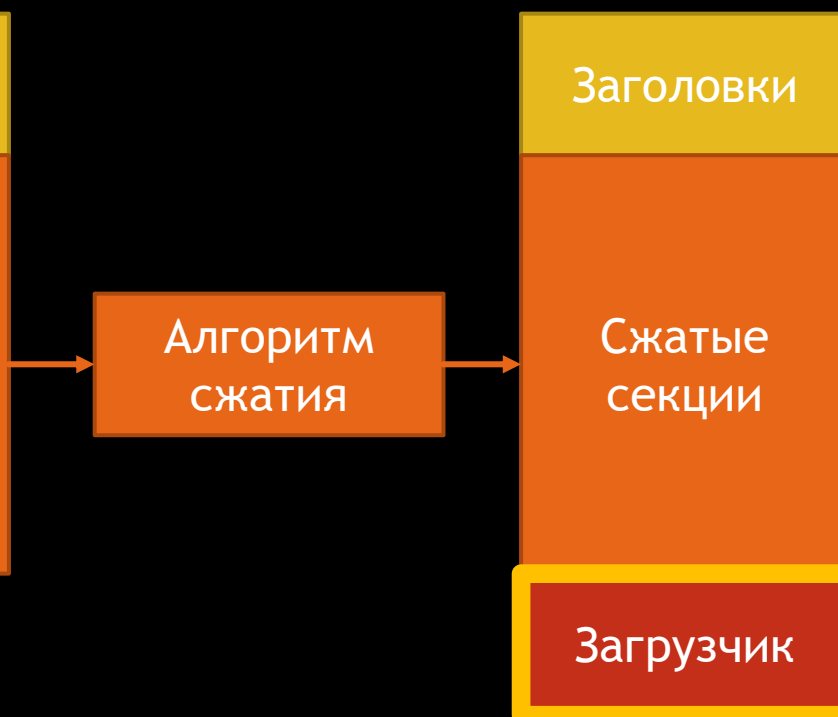
## Действия загрузчика:

- Распаковка образа в памяти
- Дефильтрация данных
- Перебазирование образа
- Получение импортов
- Передача выполнения на код исходной программы

### EXECUTABLE



### PACKED EXECUTABLE





# Распаковка

PT

Используется сжатие без потерь.

Наиболее часто используемые алгоритмы:

1. LZMA
2. APLib
3. Deflate
4. NRV
5. Алгоритм Хаффмана

```
get_gamma:
    add     dl, dl
    jnz     short loc_9D3724
    mov     dl, [esi]
    inc     esi
    adc     dl, dl

loc_9D3724:
    adc     ecx, ecx
    add     dl, dl
    jnz     short loc_9D372F
    mov     dl, [esi]
    inc     esi
    adc     dl, dl

loc_9D372F:
    jb      short get_gamma
    cmp     eax, 7D00h
    jnb     short domatch_with_2inc
    cmp     eax, 500h
    jb      short loc_9D3740
    inc     ecx
    push    esi
    mov     esi, edi
    sub     esi, eax
    rep     movsb
    pop     esi
    jmp     loc_9D3665

; -----
loc_9D3740:
    cmp     eax, 7Fh
    ja      short domatch_new_lastpos

domatch_with_2inc:
    add     ecx, 2
```



# Фильтрация в исполняемых файлах



Для улучшения степени сжатия применяется **фильтрация** данных.

В исполняемых файлах фильтруют инструкции, оперирующие относительными адресами.

Относительные call и jmp: 0xe8, 0xe9.

До фильтрации

```
.text:01002197 FF
.text:0100219D 50
.text:0100219F 57
.text:0100219F E8 CD 31 00 00
.text:010021A4 33 C0
.text:010021A6

.text:0100222B FF
.text:01002231 50
.text:01002232 57
.text:01002233 E8 39 31 00 00
.text:01002238 33 C0

        push    eax
        push    edi
        call    dummyfunc_sub_1005371
        xor     eax, eax
```

После фильтрации

```
seg000:01002197 FF
seg000:0100219D 50
seg000:0100219F 57
seg000:0100219F E8 71 53 00 00
seg000:010021A4 33 C0

seg000:0100222B FF
seg000:01002231 50
seg000:01002232 57
seg000:01002233 E8 71 53 00 00
seg000:01002238 33 C0

        push    eax
        push    edi
        call    near ptr unk_1007515
        xor     eax, eax

        push    eax
        push    edi
        call    sub_10075A9
        xor     eax, eax
```



# Дефильтрация

PT

Обратный процесс.

Дефильтрация относительных call и jmp (0xe8, 0xe9).

```
int unfilter_ct32_e8e9_bswap_le(uint8_t* buf, uint32_t buf_len, uint32_t addvalue)
{
    uint8_t* b = buf;
    uint8_t* b_end = b + buf_len - 5;
    uint32_t a = 0;
    do
    {
        if ((*b == 0xe8 || *b == 0xe9))
        {
            b += 1;
            a = (uint32_t)(b - buf);
            set_le32(b, get_be32(b) + (0 - a - addvalue));
            b += 4 - 1;
        }
    } while (++b < b_end);
    return 0;
}
```

```
unfilter:
    mov     ecx, 23959h

while_loop:
    mov     al, [edi]
    inc     edi
    sub     al, 0E8h ; 'è'

    cmp     al, 1
    ja      short unfilter

    mov     eax, [edi]
    mov     bl, [edi+4]
    xchg    al, ah
    rol     eax, 10h
    xchg    al, ah
    sub     eax, edi
    sub     bl, 0E8h ; 'è'
    add     eax, esi
    mov     [edi], eax
    add     edi, 5
    mov     al, bl
    loop    while_loop
```



# Заполнение таблицы адресов импорта

Загрузчик производит разбор таблицы импорта и заполняет IAT

```
HMODULE LoadLibrary(LPCSTR LibFileName);  
HMODULE GetModuleHandle(  
    LPCSTR lpModuleName  
);  
FARPROC GetProcAddress(  
    HMODULE hModule,  
    LPCSTR lpProcName  
);
```





# Перебазирование образа

Таблица релокаций состоит из блоков, каждый из которых состоит из:

- Информации о блоке (RelocationBlockInfo)
- Массива релокаций (RelocationEntry)

```
struct RelocationBlockInfo
{
    uint32_t* page_rva;
    uint32_t block_size;
};

struct RelocationEntry
{
    unsigned relocation_type : 4;
    unsigned : 0;
    unsigned relocation_offset : 12;
};
```

```
rebase_image:
    cmp     dword ptr [esi], 0
    jz      short rebase_done
    mov     ecx, [esi+4]
    sub     ecx, 8
    shr     ecx, 1
    mov     edi, [esi]
    add     edi, [ebp+488h]
    add     esi, 8

loc_100D1F5:
    mov     bx, [esi]
    shr     ebx, 0Ch
    cmp     ebx, 1
    jz      short reloc_type1_high
    cmp     ebx, 2
    jz      short reloc_type2_low
    cmp     ebx, 3
    jz      short reloc_type3_highlow
    jmp     short next_reloc

; -----
reloc_type1_high:
    mov     bx, [esi]
    and     ebx, 0FFFh
    add     [edi+ebx], ax
    jmp     short next_reloc

; -----
reloc_type2_low:
    mov     bx, [esi]
    and     ebx, 0FFFh
    add     [edi+ebx], dx
    jmp     short next_reloc

; -----
reloc_type3_highlow:
    mov     bx, [esi]
    and     ebx, 0FFFh
    add     [edi+ebx], edx
    jmp     short $+2

; -----
next_reloc:
    or      word ptr [esi], 0FFFFh
    add     esi, 2
    loop    loc_100D1F5
    jmp     short rebase_image
```

1

2



# Original Entry Point

PT

**OEP** (original entry point) — точка входа изначальной программы

Нахождение **OEP** и передача на неё выполнения

## UPX

```
push    ebx
push    edi
call    ebp
pop      eax
popa
lea     eax, [esp-80h]

restore_stack:
push    0
cmp     esp, eax
jnz     short restore_stack
sub     esp, 0FFFFFF80h
jmp     near ptr oep
```

## ASPack

```
prepare_oep:
mov     eax, 420Dh
push    eax
add     eax, [ebp+488h]
pop     ecx
or      ecx, ecx
mov     [ebp+40Eh], eax
popa
jnz     short oep_ret
mov     eax, 1
retn    0Ch

; -----
oep_ret:
push    0
retn
```

## PECompact2

```
pop     edx
jmp     short goto_oep
; -----
add     ecx, edx
push    8000h
push    0
push    edi
call    dword ptr [ecx]

goto_oep:
mov     eax, esi
pop     edx
pop     esi
pop     edi
pop     ecx
pop     ebx
pop     ebp
jmp     eax ; OEP
```



# Содержание



- Коротко о PT Sandbox
- Portable Executable
- Упаковщики исполняемых файлов
- **Динамическая распаковка**
- Статическая распаковка
- Материалы и ссылки



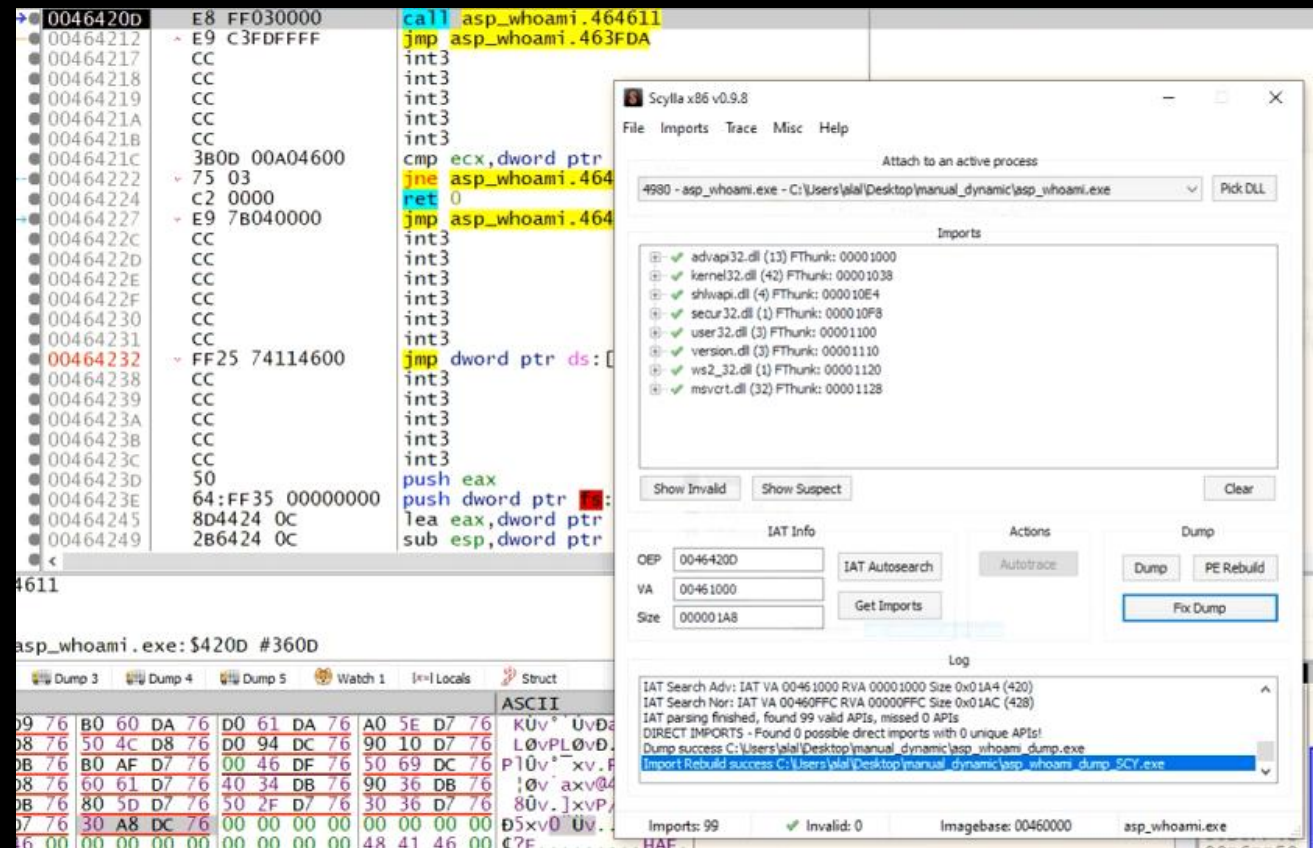
# Демо №1



# Динамическая распаковка

PT

- Определить OEP
- Создать дамп памяти
- Восстановить импорты
- Исправить дамп памяти





# Автоматизация динамической распаковки



**Speakeasy** — основанный на Unicorn фреймворк для эмуляции пользовательского режима и режима ядра OS Windows.

Основная задача — проведение динамического анализа вредоносного ПО.

Некоторые возможности:

- доступ к регистрам и памяти
- перехват вызовов WinAPI
- перехват выполнения инструкций
- перехват доступа к памяти



# Получение ОЕР

1. Определение границ секции загрузчика
2. Добавление перехвата на выполнение инструкций (hook\_sectionhop)
3. Проверка текущего адреса выполнения на принадлежность к секции загрузчика

Number	Name	VirtSize	RVA	PhysSize	Offset	Flag
1	.text	00009000	00001000	00004400	00000400	E0000060
2	.data	00001000	0000A000	00000200	00004800	C0000040
3	.rsrc	00001000	0000B000	00000200	00004A00	C0000040
4	.reloc	00001000	0000C000	00000600	00004C00	C2000040
5	.aspack	00002000	0000D000	00001A00	00005200	E0000060
6	.adata	00001000	0000F000	00000000	00006C00	E0000040

```
def set_loader_range(self, base, start, end):
    # ImageBase
    self.base = base
    # ImageBase + Loader section RVA
    self.start = start
    # ImageBase + Loader section RVA + Loader section virtual size
    self.end = end

def hook_sectionhop(self, emu, addr, size, ctx):
    3 if addr < self.start or addr > self.end:
        if self.mem_read(addr, 1) != b'\xc3':
            self.oep = addr
            self.done = True
            self.stop()
```



# Создание дампа памяти

Загрузчик ASPack затирает таблицу импортов.

Получение дампа памяти с таблицей импорта:

1. Первый вызов `GetModuleHandle` — для получения загрузчиком служебных функций.
2. Второй вызов `GetModuleHandle` — начало восстановления импортов (образ распакован, таблица импортов не затерта).

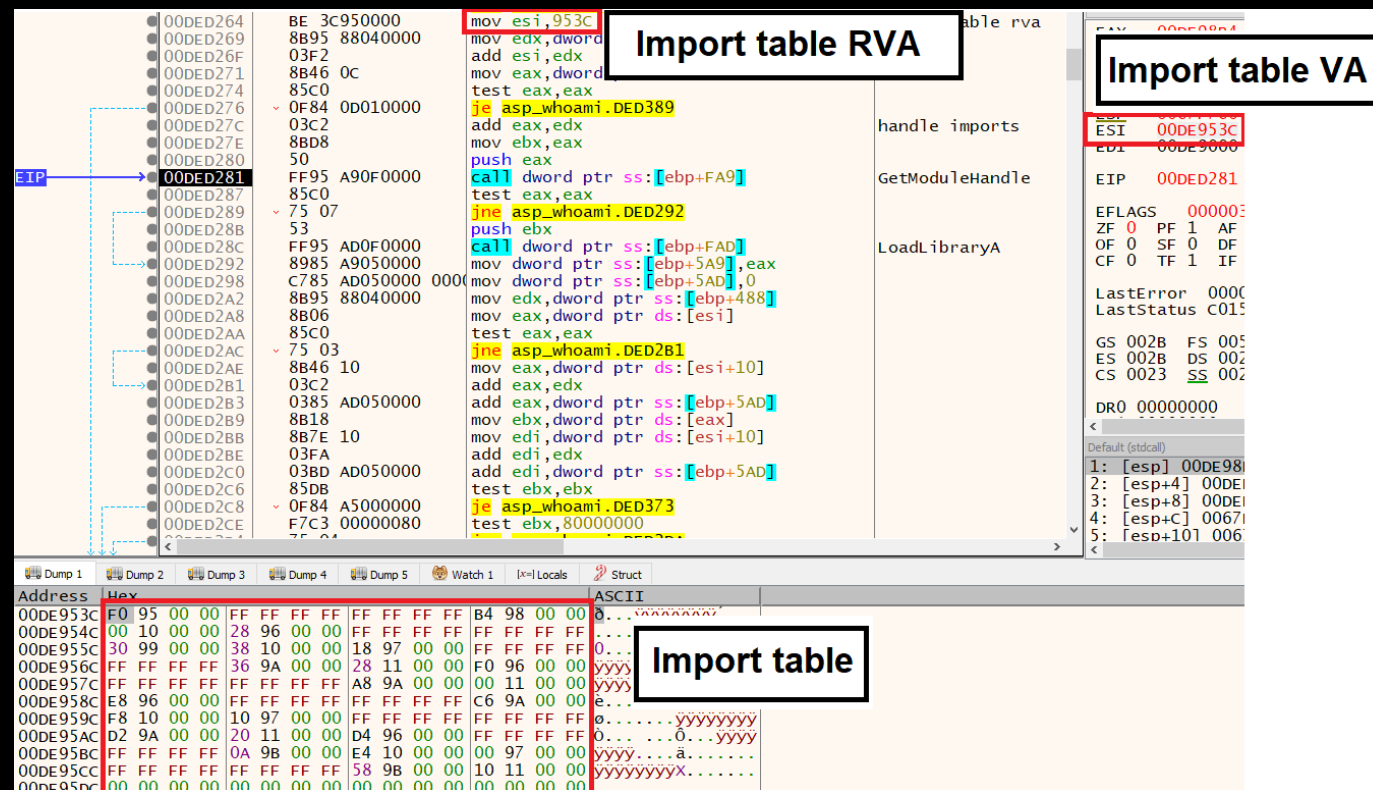
00DED03C	call dword ptr ss:[ebp+FA9]	1
00DED042	mov dword ptr ss:[ebp+48C],eax	
00DED048	mov esi,eax	
00DED04A	lea edi,77010A60 <kernel32.GetModuleHandleA>	
00DED04D	push edi	
00DED04E	push ebp	
00DED04F	call mov ebp,esp	
00DED055	pop ebp	2
00DED056	stosd	
	jmp dword ptr ds:[&GetModuleHandleA]	
00DED280	push eax	
00DED281	call dword ptr ss:[ebp+FA9]	
00DED287	test eax,eax	
00DED289	jne as77010A60 <kernel32.GetModuleHandleA>	
00DED28B	push edi	
00DED28C	call push ebp	
00DED292	mov ebp,esp	
00DED298	pop ebp	
00DED2A2	jmp dword ptr ds:[&GetModuleHandleA]	
00DED2A8	mov eax,dword ptr ds:[esi]	



## PT

Во втором вызове **GetModuleHandle** в регистре **ESI** содержится виртуальный адрес **таблицы импорта**.

Размер **таблицы импорта**  
вычисляется как количество  
вызовов **GetModuleHandle**,  
умноженное на размер дескриптора  
импорта (20 байт).





# Перехват GetModuleHandle



1. Первый вызов GetModuleHandle
2. Второй вызов — чтение адреса таблицы импорта, создание дампа памяти
3. Второй и последующие вызовы — вычисление размеров таблицы импорта

```
# hook for aspack
def hook_getmodulehandle(self, emu, api_name, func, params):
    '''HMODULE GetModuleHandle(
        LPCSTR lpModuleName
    );'''
    2 if self.idt_start is None and self.decompression_done:
        self.idt_start = self.reg_read(speakeasy.winenv.arch.X86_REG_ESI) - self.base
        print('Creating clean dump...')
        self.dump_on_hit()
    3 if self.idt_start is not None:
        self.idt_size += 0x14

    rv = 0
    mod_name, = params
    cw = self.get_char_width(api_name)
    if not mod_name:
        proc = emu.get_current_process()
        rv = proc.base
    else:
        lib = emu.read_mem_string(mod_name, cw)
        params[0] = lib
        sname, _ = os.path.splitext(lib)
        sname = speakeasy.windows.common.normalize_dll_name(sname)
        mods = emu.get_user_modules()
        for mod in mods:
            img = ntpath.basename(mod.get_emu_path())
            fname, _ = os.path.splitext(img)
            if fname.lower() == sname.lower():
                rv = mod.get_base()
                break
    1 self.decompression_done = True
    return rv
```



# Исправление дампа памяти



1. Выставление новой точки входа
2. Указание таблицы импорта
3. Отключение перемещения образа в памяти
4. Установка границ секций в файле

```
def fix_dump(dump, oep, idt_start, idt_size):
    pe = pefile.PE(data=dump)
    1 pe.OPTIONAL_HEADER.AddressOfEntryPoint = oep - pe.OPTIONAL_HEADER.ImageBase

    2 pe.OPTIONAL_HEADER.DATA_DIRECTORY[pefile.DIRECTORY_ENTRY['IMAGE_DIRECTORY_ENTRY_IMPORT']].VirtualAddress = idt_start
    pe.OPTIONAL_HEADER.DATA_DIRECTORY[pefile.DIRECTORY_ENTRY['IMAGE_DIRECTORY_ENTRY_IMPORT']].Size = idt_size

    3 if pe.OPTIONAL_HEADER.DllCharacteristics & pefile.DLL_CHARACTERISTICS['IMAGE_DLLCHARACTERISTICS_DYNAMIC_BASE']:
        pe.OPTIONAL_HEADER.DllCharacteristics ^= pefile.DLL_CHARACTERISTICS['IMAGE_DLLCHARACTERISTICS_DYNAMIC_BASE']
        print('ASLR disabled')

    last_section_num = 0
    image_end = 0
    4 for i, sect in enumerate(pe.sections):
        sect.PointerToRawData = sect.VirtualAddress
        sect.SizeOfRawData = sect.Misc_VirtualSize
        if image_end < sect.VirtualAddress + sect.Misc_VirtualSize:
            last_section_num = i

    # in case of uninitialized sections
    new_size = pe.sections[last_section_num].PointerToRawData + pe.sections[last_section_num].SizeOfRawData
    new_pe = pe.write()
    if new_size > len(new_pe):
        new_pe += (new_size - len(new_pe)) * b"\x00"

    return new_pe
```



# Демо №2



# Содержание

The logo consists of the letters 'PT' in a stylized, bold, sans-serif font, positioned within a white square.

- Коротко о PT Sandbox
- Portable Executable
- Упаковщики исполняемых файлов
- Динамическая распаковка
- **Статическая распаковка**
- Материалы и ссылки



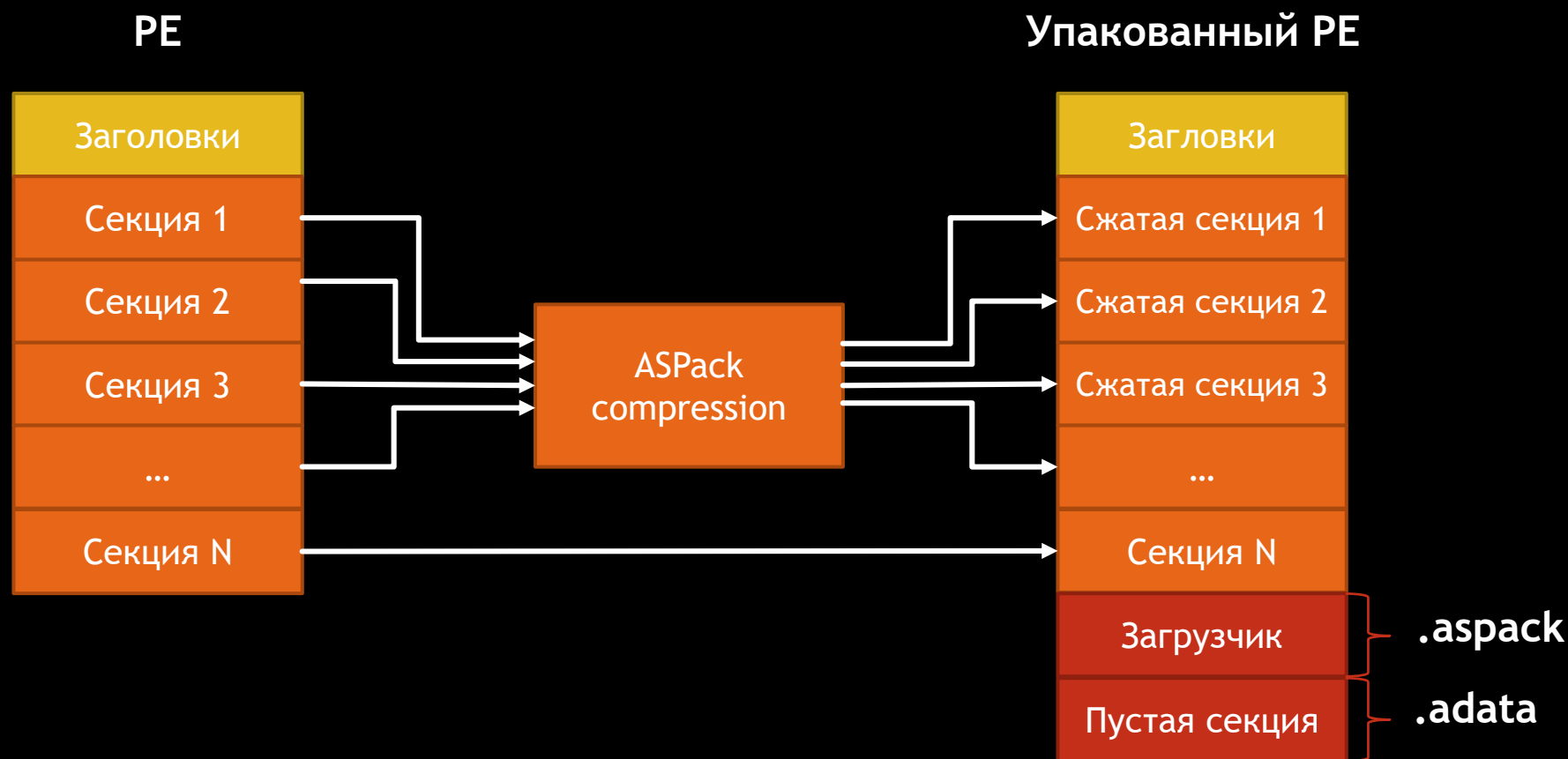
# Статическая распаковка

РТ





# Статическая распаковка ASPack



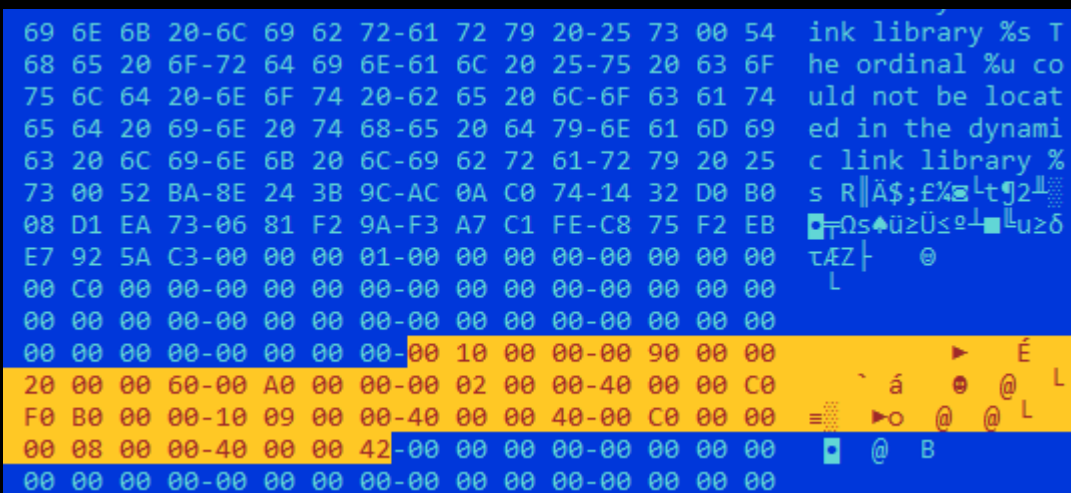


# Нахождение сжатых секций

В загрузчик записаны **относительные виртуальные адреса** и **размеры** сжатых блоков данных.

Информация о блоке сжатых данных:

1. Относительный виртуальный адрес блока
2. Размер блока
3. Только в новых версиях: права секций



```
struct AspackInfoOld
{
    // 1
    uint32_t packed_rva;
    // 2
    uint32_t packed_size;
};

struct AspackInfoNew
{
    // 1
    uint32_t packed_rva;
    // 2
    uint32_t packed_size;
    // 3
    uint32_t section_protection;
};
```



# Получение информации для распаковки ASPack

Получение информации о распаковке из кода:

1. Положение сжатых блоков
2. Адреса таблиц импорта и таблиц релокаций
3. Адрес точки входа

Адресация кода загрузчика происходит относительно содержимого регистра **EBP**.

00F4D00A	5D	pop ebp
00F4D00B	45	inc ebp
00F4D00C	55	push ebp
00F4D00D	C3	ret
00F4D00E	E8 01000000	call asp_whoami.F4D014
00F4D013	EB 5D	jmp asp_whoami.F4D072

00F4D097	8B03	mov eax,dword ptr ds:[ebx]
00F4D099	8785 00050000	xchg dword ptr ss:[ebp+599],eax
00F4D09F	8903	mov dword ptr ds:[ebx],eax
00F4D0A1	8DB5 c5050000	lea esi,dword ptr ss:[ebp+5C5]
00F4D0A7	833E 00	cmp dword ptr ds:[esi],0
00F4D0AA	0F84 0A010000	je asp_whoami.F4D1BA
00F4D0B0	6A 04	push 4
00F4D0B2	68 00100000	push 1000
00F4D0B7	68 00180000	push 1800



# Алгоритм сжатия ASPack



В упаковщике ASPack авторский алгоритм сжатия данных по словарю.

1. Функция декомпрессии в декомпилированном виде
2. Функция чтения заданного количества бит из закодированных данных
3. Функция чтения следующей закодированной последовательности

```
int __cdecl aspack_decompress(int a1, int a2, int a3, int a4)
{
    int v5; // [esp+0h] [ebp-354h] BYREF
    int v6[212]; // [esp+4h] [ebp-350h] BYREF

    init_decompressor(v6, a4);
    if ( !(unsigned __int8)build_unpacking_table(a1, a2) )
        return -1;
    if ( (unsigned __int8)decompress(a3, &v5) )
        return v5;
    return -1;
}
```

1

```
def get_n_bits(asp, bits):
    while asp.bits >= 8:
        b = asp.c_data[asp.ind]
        asp.word = ((asp.word << 8) | b) % 2 ** 32
        asp.bits = asp.bits - 8
        asp.ind += 1
    res = ((asp.word >> (8 - asp.bits)) & 0xFFFFF) >> (24 - bits)
    asp.bits += bits
    return res
```

2

```
def get_next_symbol(asp, num):
    if asp.bits >= 8:
        while True:
            asp.word = (asp.c_data[asp.ind] | (asp.word << 8)) % 2 ** 32
            asp.ind += 1
            asp.bits -= 8
            if asp.bits < 8: break
        comp_val = (asp.word >> (8 - asp.bits)) & 0xfffe00
        if comp_val >= asp.bs[num].ranges[8]:
            if comp_val >= asp.bs[num].ranges[10]:
                if comp_val >= asp.bs[num].ranges[11]:
                    if comp_val >= asp.bs[num].ranges[12]:
                        if comp_val >= asp.bs[num].ranges[13]:
                            numofbits = 0xf - (comp_val < asp.bs[num].ranges[14])
                        else: numofbits = 0xd
                    else: numofbits = 0xc
                else: numofbits = 0xb
            else: numofbits = 0xa - (comp_val < asp.bs[num].ranges[9])
        else: numofbits = asp.wrkmem[asp.bs[num].offs2 + (comp_val >> 0x10)]
        asp.bits += numofbits
        if numofbits > 1:
            pos = (((comp_val - asp.bs[num].ranges[numofbits - 1]) >> (0x18 - numofbits)) + asp.bs[num].ranges[numofbits + 0xf + 1]) * 4 + asp.bs[num].offs
        elif numofbits == 1:
            pos = (((comp_val - asp.bs[num].ranges[numofbits - 1]) >> (0x18 - numofbits)) + asp.bs[num].ranges[numofbits + 0xf + 1]) * 4 + asp.bs[num].offs
        else: return False
        res = struct.unpack("<I", asp.wrkmem[pos:pos+4])[0]
        return res
```

3



# Дефилтрация ASPack

PT

## Дефилтрация в ASPack:

- Относительные call и jmp (0xe8, 0xe9)
- Отфильтрованные инструкции обозначены **маркером**.

```
FF 3F 00 00 00      call    near ptr 1433F12h
E8 05 29 43 00      xor     esi, esi
33 F6
```

```
int unfilter_ct32_e8e9_rol32_le(uint8_t* buf, uint32_t buf_len, uint8_t cto, uint32_t addvalue)
{
    uint32_t a = 0;
    uint8_t* b = buf;
    uint32_t c = 0;
    do {
        if ((*b == 0xe8 || *b == 0xe9)) { 1
            if (*(b + 1) == cto) { 2
                c = get_le32(b + 1); 3
                a = rol32(c - cto, 0x18);
                set_le32(b + 1, a - addvalue - (b - buf)); 4
                b += 4;
            }
        }
        b++;
    } while (b < buf + buf_len - 5);
    return 0;
}
```

```
unfilter:
    or     ecx, ecx
    jz     short finish_unfilter
    js     short finish_unfilter
    1 lodsb
    cmp    al, 0E8h ; 'è'
    jz     short loc_100D134
    jmp     short $+2
; -----
loc_100D12C:
    cmp    al, 0E9h ; 'é'
    jz     short loc_100D134
; -----
loc_100D130:
    inc     ebx
    dec     ecx
    jmp     short unfilter
; -----
loc_100D134:
    3 mov     eax, [esi]
    jmp     short $+2
; -----
loc_100D138:
    2 cmp     byte ptr [esi], 5
    jnz     short loc_100D130
    4 and     al, 0
    rol     eax, 18h
    sub     eax, ebx
    mov     [esi], eax
    add     ebx, 5
    add     esi, 4
    sub     ecx, 5
    jmp     short unfilter
```



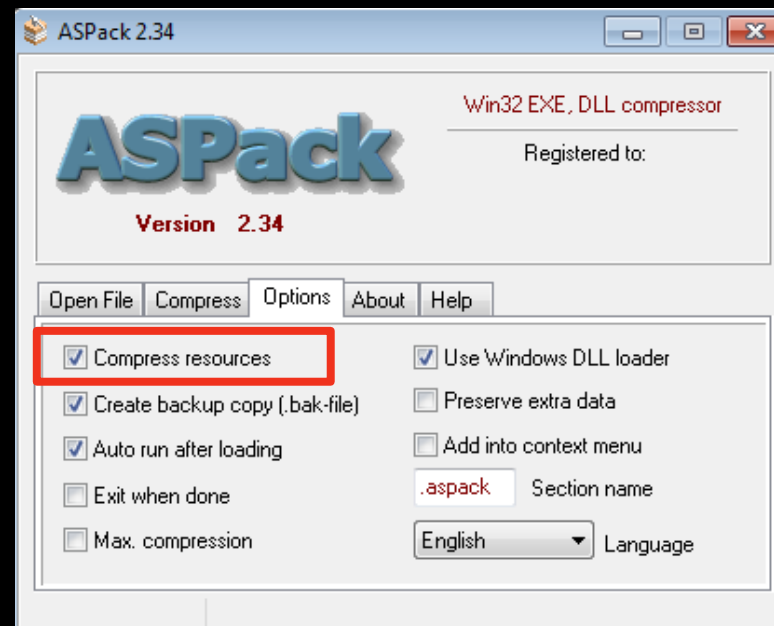
# Восстановление ресурсов

**Ресурсы** — данные, помещённые в специально отведённую для них область исполняемого файла (обычно .rsrc).

ASPack не сжимает:

1. Иконки (RT\_ICON)
2. Манифесты (RT\_MANIFEST)

В случае, если ресурсы сжаты, необходимо их собрать заново.





# Склеивание секций и исправление заголовков

```
# align sections to FileAlignment
for i, sect in enumerate(original_sects):
    if len(dec_sections):
        dec_sections[i] = dec_sections[i] + b"\x00" * (
            utils.align_up(len(dec_sections[i]), self.pe.OPTIONAL_HEADER.FileAlignment)
            - len(dec_sections[i])
        )
```

1

1. Выравнивание распакованных секций

2. Исправление заголовков секций

3. Исправление Optional Header

```
rawdataptr = self.pe.sections[0].PointerToRawData
for i, sect in enumerate(original_sects):
    if i != self.asp sect num and i != self.adata sect num:
        if sect is not None:
            if sect.sect_attr is not None:
                self.pe.sections[i].Characteristics = sect.sect_attr

self.pe.sections[i].SizeOfRawData = len(dec_sections[i])
self.pe.sections[i].PointerToRawData = rawdataptr
rawdataptr += self.pe.sections[i].SizeOfRawData
```

2

- Точка входа
- Размер образа в памяти
- Размер заголовков
- Адреса и размеры таблиц

3



# Статический распаковщик готов

PT

```
PS C:\Users\AL\Documents\PT\unpacker> .\test_samples\asp_whoami.exe
desktop-abbi4ol\al
PS C:\Users\AL\Documents\PT\unpacker> python .\unpacker.py -p ASPACK -i .\test_samples\asp_whoami.exe -o whoami.exe -v
Unpacking ASPACK...
Resources: fixed
Imports: fixed
Relocations: fixed
TLS: nothing to do
Unpacked
PE is written to: whoami.exe
PS C:\Users\AL\Documents\PT\unpacker> .\whoami.exe
desktop-abbi4ol\al
PS C:\Users\AL\Documents\PT\unpacker> |
```



# Бонус: как сломать стандартный распаковщик UPX

PT

## Названия секции UPX

Number	Name	VirtSize	RVA	PhysSize	Offset	Flag
1	UPX0	00009000	00001000	00000000	00000400	E0000080
2	UPX1	00005000	0000A000	00004800	00000400	E0000040
3	.rsrc	00001000	0000F000	00000A00	00004C00	C0000040

## Переименование секций

00	00	00	00-00	00	00	00-00	00	00	00-00	00	00	00	
56	53	43	30-00	00	00	00-00	00	12	00-00	10	00	00	vsce
00	00	00	00-00	04	00	00-00	00	00	00-00	00	00	00	
00	00	00	00-80	00	00	E0-56	53	43	31-00	00	00	00	αVSC1
00	B0	0B	00-00	10	12	00-00	A2	0B	00-00	04	00	00	α
00	00	00	00-00	00	00	00-00	00	00	00-40	00	00	E0	@



PT

```





struct UpxHeader {
    // 0x0
    uint32_t upx_magic; // 0x21585055 "UPX!"
    // 0x4
    uint8_t version; // upx version
    uint8_t format; // file format (e.g. dll, exe)
    uint8_t compression_method; // nrsv2b, lzma, zlib, etc
    uint8_t comp_level; //
    // 0x8
    uint32_t unpacked_adler; // decompressed checksum
    // 0xc
    uint32_t compressed_adler; // compressed checksum
    // 0x10
    uint32_t unpacked_len; //
    // 0x14
    uint32_t compressed_len;
    // 0x18
    uint32_t unpacked_filesize;
    // 0x1c
    uint8_t offset_filter; // type of filter
    uint8_t filter_cto; // filter cto
    uint8_t n_mru; //
    uint8_t header_checksum;
};

```

## UPX заголовков

00	00	00	00-00	00	00	00-00	00	00	00-00	00	00	00
00	00	00	00-00	00	00	00-00	00	00	33-2E	39	36	00
55	50	58	21-0D	09	02	08-AA	1E	E8	92-CE	A8	94	F3
B8	CA	00	00-91	45	00	00-00	A8	00	00-26	04	00	B1
6C	9A	A6	FB-9C	97	00	00-B4	03	D2	DC-EC	04	98	D3
34	4D	B3	1C-30	44	5A	74-BA	EE	33	4D-90	9E	00	C2

## Затирание UРХ заголовка

00	00	00	00-00	00	00	00-00	00	00	30-30	30	30	00	0000
56	53	43	2A-0D	09	08	0A-29	12	98	A5-04	F4	C0	C8	VSC*  \$yN\$  LL
3A	84	1D	00-03	9E	0B	00-00	A6	1C	00-26	27	00	D4	:ä↔  @L &'  L
72	49	F7	FF-68	00	12	59-20	E8	27	0E-07	52	59	C3	rI≈ h \$Y 0'♫•RY



# Полезные ссылки



## PT Sandbox

[ptsecurity.com/ru-ru/products/sandbox/](https://www.ptsecurity.com/ru-ru/products/sandbox/)



## PT ESC Threat Intelligence blog

[ptsecurity.com/ru-ru/research/pt-esc-threat-intelligence/](https://www.ptsecurity.com/ru-ru/research/pt-esc-threat-intelligence/)



## PT ESC Incident Response Alert

[ptsecurity.com/ru-ru/services/esc/](https://www.ptsecurity.com/ru-ru/services/esc/)



## Вопросы

[webinar@ptsecurity.com](mailto:webinar@ptsecurity.com)

Повышение привилегий в системе:  
детектирование техник на примере PT Sandbox

<https://www.ptsecurity.com/ru-ru/research/webinar/povyshenie-privilegij-v-sisteme-detektirovanie-tekhnik-na-primere-pt-sandbox/>

Плагин exploitmon для DRAKVUF.  
Обнаружение эксплуатации ядра ОС с  
помощью PT Sandbox

<https://www.ptsecurity.com/ru-ru/research/webinar/plugin-exploitmon-dlya-drakvuf-obnaruzhenie-ehkspluatacii-yadra-os-s-pomoshchyu-pt-sandbox/>

Александр Лаухин  
[alaukhin@ptsecurity.com](mailto:alaukhin@ptsecurity.com)

Алексей Вишняков  
[avishnyakov@ptsecurity.com](mailto:avishnyakov@ptsecurity.com)  
Twitter: @Vishnyak0v