



ОНЛАЙН-ОБРАЗОВАНИЕ

Регистры процессора, работа с памятью

Особенности работы ПК



А зачем оно надо?

Мотивация



А зачем оно надо?

```
mov     [esi+4], ebx

label_getMem:
mov     ax, [esi]           ; CODE XREF: proc5_1+A61j
                                ; если eax = 0xffff, то
                                ; add eax,2 даст 0!
add     ax, 2
mov     [esi+2], ax
jz     short loc_8BCC4A1B
movzx   eax, ax
push   edi                 ; Tag
add     eax, 8
push   eax                 ; NumberOfBytes
push   ebx                 ; PoolType
call   ds:ExAllocatePoolWithTag

mov     [esi+4], eax
cmp     eax, ebx
jz     loc_8BCC4946
movzx   ecx, word ptr [esi+2]
mov     [eax], ecx
mov     eax, [esi+4]
mov     [eax+4], edi
movzx   ecx, word ptr [esi+2]
add     dword ptr [esi+4], 8
mov     eax, [esi+4]
push   ecx                 ; size_t
push   ebx                 ; int
push   eax                 ; void *
call   memset
add     esp, 0Ch
jmp     short label_copy

; -----
loc_8BCC4A1B:
mov     [esi+4], ebx           ; CODE XREF: proc5_1+F31j

label_copy:
                                ; CODE XREF: proc5_1+961j
                                ; proc5_1+1311j
movzx   eax, word ptr [esi]
push   eax                 ; size_t
push   [ebp+inBuff]        ; void *
push   dword ptr [esi+4]   ; void *
call   memcpy
movzx   eax, word ptr [esi]
```

А зачем оно надо?

```
.text:AD25A637      stosl
.text:AD25A638      mov     eax, [ebp+Buf]
.text:AD25A63B      lea    edx, [ebp+BufDest]
.text:AD25A641      sub    edx, eax
.text:AD25A643      pop    edi
.text:AD25A644
.text:AD25A644      loc_AD25A644:                ; CODE XREF: VulnProc+4A↓j
.text:AD25A644      mov    cl, [eax]
.text:AD25A646      mov    [edx+eax], cl
.text:AD25A649      inc    eax
.text:AD25A64A      test   cl, cl
.text:AD25A64C      jnz    short loc_AD25A644
.text:AD25A64E      lea    eax, [ebp+BufDest]
.text:AD25A654      push   eax                ; char *
.text:AD25A655      call  ds:_strupr
.text:AD25A65B      pop    ecx
.text:AD25A65C      call  sub_AD2599FA
.text:AD25A661      lea    eax, [ebp+P]
.text:AD25A667      push   eax
.text:AD25A668      call  sub_AD25A528
.text:AD25A66D      test   eax, eax
.text:AD25A66F      jge    short loc_AD25A689
.text:AD25A671      cmp    dword_AD25FC4C, 0
.text:AD25A678      jz     short loc_AD25A6D9
.text:AD25A67A      push   eax
.text:AD25A67B      push   offset aProcobsrvesEnu ; "ProcObsrves: EnumProcesses failed 0x%X\..."
.text:AD25A680      call  DbgPrint
.text:AD25A685      pop    ecx
.text:AD25A686      pop    ecx
.text:AD25A687      jmp    short loc_AD25A6D9
.text:AD25A689      ; -----
.text:AD25A689
.text:AD25A689      loc_AD25A689:                ; CODE XREF: VulnProc+6D↑j
.text:AD25A689      push   esi
.text:AD25A68A      mov    esi, [ebp+P]
.text:AD25A690
.text:AD25A690      loc_AD25A690:                ; CODE XREF: VulnProc+C3↓j
.text:AD25A690      mov    eax, [esi+44h]
.text:AD25A693      test   eax, eax
.text:AD25A695      jz     short loc_AD25A687
.text:AD25A697      cmp    dword_AD25F42C, 1
```

```
A problem has been detected and windows has been shut down to prevent damage to your computer.
```

```
A driver has overrun a stack-based buffer. This overrun could potentially allow a malicious user to gain control of this machine.
```

```
If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:
```

```
Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.
```

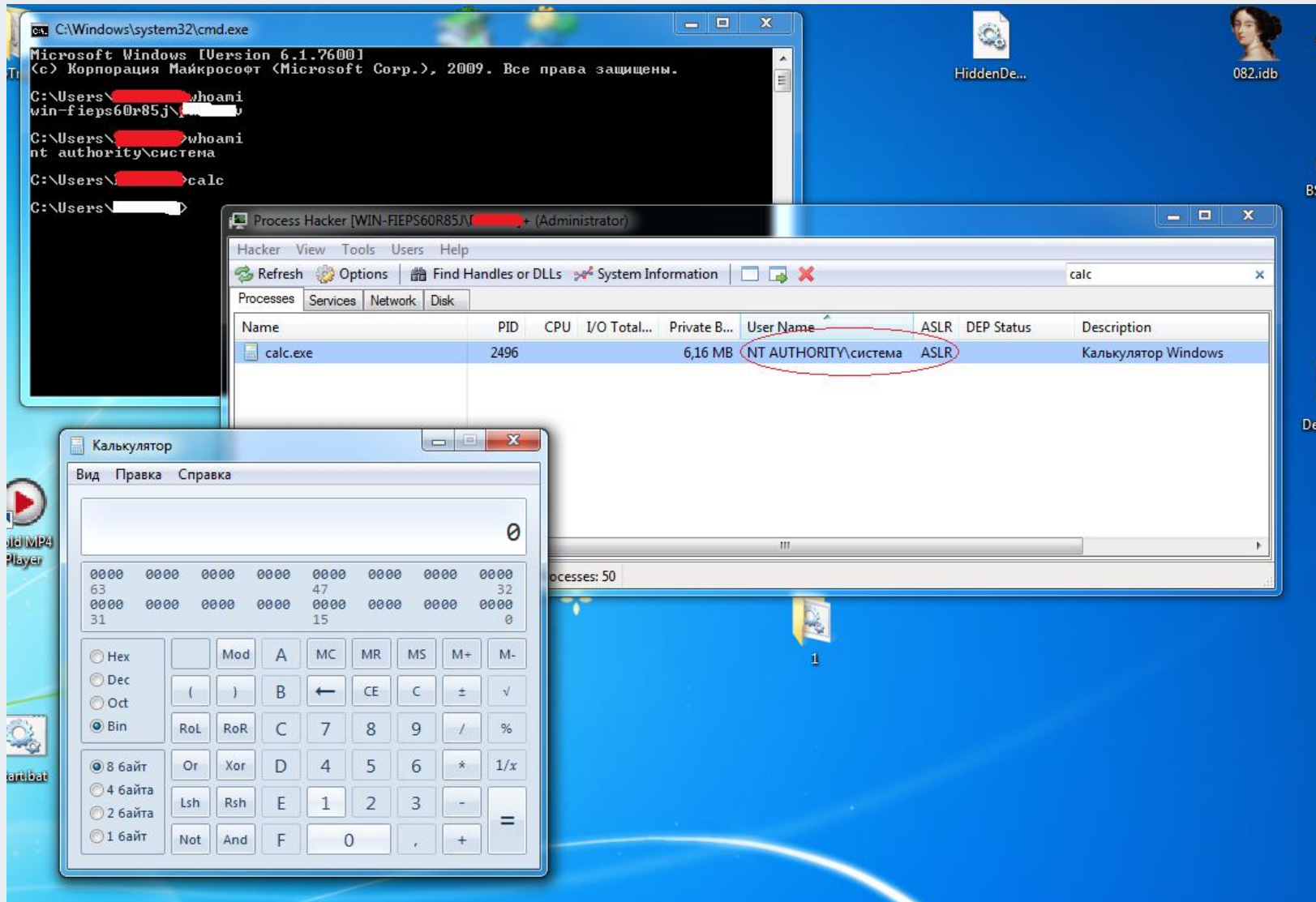
```
If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.
```

```
Technical information:
```

```
*** STOP: 0x000000F7 (0x41414141,0x00002CF0,0xFFFFD30F,0x00000000)
```

```
collecting data for crash dump ...  
initializing disk for crash dump ...
```

А зачем оно надо?





1971 год. Работал с 4х (!) разрядными данными

8080 – 1974 год. 8 разрядные данные, 64к память

8086 – 1978 год. 16 разрядные данные, 1М память

8088 – 1979 год. 16 разрядные данные, 1М память

80186 – 1983 год.

80286 – 1983 год. 16 разрядные данные, 16М память

80386 – 1983 год. 32 разрядные данные, 4Г память

1386 – 1987 год.

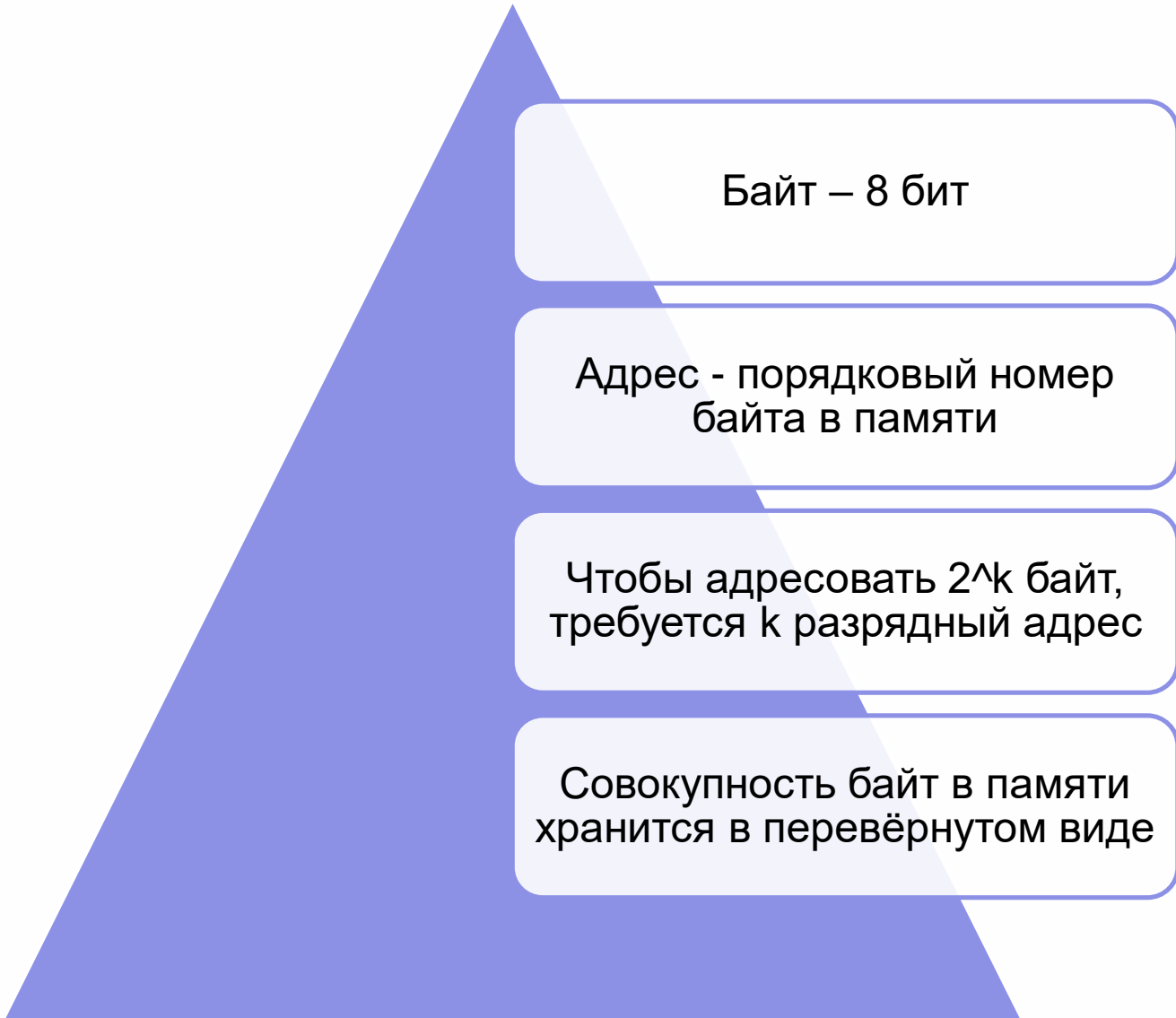
1486 – 1990 год.

Pentium – 1993 год. 64 разрядные данные.

Особенности работы ПК

Оперативная память





Байт – 8 бит

Адрес - порядковый номер
байта в памяти

Чтобы адресовать 2^k байт,
требуется k разрядный адрес

Совокупность байт в памяти
хранится в перевёрнутом виде

Регистры общего назначения

AX (accumulator) AH/AL

CX (count) CH/CL

DX (data) DH/DL

BX (base) BH/BL

Si (Source index)

Di (Destination Index)

Bp (Base Pointer)

Sp (Stack pointer)

Сегментные регистры

CS (Code Segment)

DS (Data Segment)

SS (Stack Segment)

ES (Extra Segment)

Instruction Pointer

IP

Регистр флагов

Флаги условий: cf, of, zf, sf, pf, af

Флаги состояний: df, if, tf

БИТЫ															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

16 регистров общего назначения

RAX		RCX				RDX				RBX			
EAX		ECX				EDX				EBX			
AX		CX				DX				BX			
AH	AL	CH CL				DH DL				BH BL			

RDI, RSI, RBP и RSP расширены для побайтного доступа!

К R1-R15 можно обращаться как к **D**WORD, **W**ORD, **B**YTE

RSP		RBP		RSI		RDI		Rx	
	ESP		EBP		ESI		EDI		RxD
	SP		BP		SI		DI		RxW
	SP L		BP L		SIL		DIL		RxB

X - цифра [0-15]

Зачем пишут x64 малварь?

*Если архитектура Windows
поддерживает выполнение x32 кода на
x64 OS системе*



Реальный режим (Real Mode)

Режим системного управления (System Management Mode)

Виртуальный режим i8086 (V86)

Защищенный режим (Protected Mode)

Модель памяти

В 8086 процессоре

Что интересного?

Сегментация

Логический адрес

Физический адрес

ВЫВОДЫ:

Номер сегмента всегда кратен $0x10$ → заканчивается на 0

Последняя цифра не записывается в сегментный регистр

Сегментные регистры всегда 16-разрядный

$\text{физический адрес} = \text{segment} * 0x10 + \text{offset}$

Диапазон адресов: $0 - 2^{20} - 1 = [0; 0xFFFFF]$

```
mov si, 0xFFFF
```

```
mov bx, [0xFFFF+0xF]
```

```
mov cx, [si+0xF]
```

```
00000000: BEFFFF      mov     si,-1 ; ' '  
00000003: 678B1D0E000100  mov     bx,[00001000E]  
0000000A: 8B4C0F        mov     cx,[si][00F]
```

Сегментация сокращает размер команд!

```
00000000: BEFFFF      mov     si,-1 ; ' '  
00000003: 678B1D0E000100  mov     bx,[00001000E]  
0000000A: 8B4C0F      mov     cx,[si][00F]
```

А что, если выражение получится $> 0xFFFFFFFF$?

`физический адрес = segment * 0x10 + offset`

Адресная шина процессора 8086 – 20и разрядная. Может адресовать 1Мб памяти

Диапазон логических адресов процессора 8086 –
[0000h:0000h – 0xffff:0xffff]



НО!

Тогда, диапазон Физических адресов:

[0 – 0xffff*0x10+0xffff] = [0 – 0x10ffef]

20 разрядов с [0-19]

Преобразование логического адреса в физический происходит по модулю `0xffff`

Тогда, диапазон логических адресов:
`[0000h:0000h - 0xffff:000f]`



Физический адрес максимального логического:
`[0 - 0xffff*0x10+0x000f] = [0 - 0xfffff]`

Все физические адреса, полученные из выходящих из диапазона логических «оборачиваются» ($\& 0\text{xFFFFFF}$)

Пример:

`[0000h:0000h - 0xffff:0x10]`



✓ $[0 - 0\text{xffff} * 0\text{x10} + 0\text{x10}] = [0 - 0\text{x100000} \& 0\text{FFFFFF}]$
 $= [0 - 0\text{x00000}]$

Итог:

```
Jmp 0:0 ~ jmp 0xFFFF:0x10
```

В реальном режиме 8086 используются младшие 20 адресных линий: A0-A19

Т.К. Более новые процессоры имеют ширину шины > 20, в реальном режиме адреса не оборачиваются, что приводит к нарушению обратной совместимости, поэтому адресная линия A20 по умолчанию блокируется

ДЗ

Tool chains

O T U S

Emu 8086

FASM

X64 Dbg

Ida Pro

HIEW

FAR MANAGER & CONEmu



Вопросы???





Пакулов Артур

A.Pakulov.Otus@Gmail.com

Спасибо
за внимание!

