# Fermín J. Serna – MSRC Engineering @ Microsoft EXPLOITS & MITIGATIONS: EMET

# Agenda

- Introduction
  - Exploits
  - Mitigations
- EMET
  - V.1.02 (public available)
  - Next version (soon to be available)
- QA

# Introduction

Who am I?

- Security Engineer at MSRC Engineering React (former SWI).
  - Technical background of MSRC for externally reported security issues: MSo8-67, Aurora IE case ...
- HPUX PA-RISC, Solaris SPARC: assembly, shellcode and exploit lover.
- Published several docs and exploits: Phrack (58-9) HP-UX PA-RISC exploitation techniques long time ago, ...
- Developer of a PAX implementation on win32 long time ago...

- A bit of history on the windows side...
  - Old days were good and easy to write exploits...
    - Smash the \* for fun and profit: stack, heap, vtable,...
  - These days is more difficult but challenges are fun
    - Better code...
    - Mitigations...
    - Mitigations...
    - Did I say mitigations?
    - No... wait... something is left...

## Mitigations!!!

# Exploits

#### Stack overflows

0X41414141

0X41414141

0X41414141

0X41414141

**AAAAAAAAAAAAAAA** AAAAAAAAAAAAAAAAA

Local Variables

**Target SEH record** 

Target saved EIP... one return needed

Target saved EBP... two returns needed

Target a local variable pointer.

# Exploits

#### SEH record overwrite





- Heap overflows
  - Target heap structures (unlink exploits)
  - Target an object vtable
  - Any variable/structure at the heap could be interesting for a write AV.

- C++ touch already freed objects
  - Somehow object was deleted but there are references to it on other objects... (ref counting bugs)
  - Target vtable calls: call [reg+offset]
  - Any member variable at the heap could be interesting for a write AV.

Return oriented programming

Crash at call [ecx + 100h] We assume attacker controls content of [ecx-500h,ecx+500h]

Attacker places at [ecx+100h] some fixed address of:

mov esp, ecx; «some other non-deref ins»; ret
Esp is pointing inside the controlled range
Next ret will grab arguments and saved eip from controlled range.

Chain of Virtualloc, strcpy, jmp will defeat DEP Full ASLR breaks ROP!!!!

# Mitigations

- Had been introduced historically in new OS and its Service Packs.
  - XPSP2 was a big security push: /SAFESEH, DEP, /GS, SAFE Unlinking...
  - Server 2k3 SP2: Improved GS
  - Vista: User mode ASLR + DEP (big win), heap structures encoded
  - Vista SP1: Kernel mode ASLR + SEHOP (disabled by default)
  - Server 2k8: SEHOP (enabled by default)

# Mitigations

#### GS (stack cookie)



# Mitigations

#### GS (stack cookie)

- Prevents
  - Saved Return address overwrite usage
  - Saved EBP overwrite usage
  - Local function pointers usage (new /GS)
- Does not prevent
  - SEH overwrite usage
  - Other local varibles usage
  - Weird overwrites MSo8-67

# Mitigations

#### DEP (Hardware one)

- Processor check check when executing instructions.
- Page's protection can be RWX (Read, Write and Executable) + other properties

Logic behind:

if (has\_exec\_bit((PAGE\_PROTECTION(eip))==FALSE)
{
 Raise\_exception();
}

# Mitigations

#### DEP (Hardware one)

- ASLR makes it more robust
- Full ASLR + DEP + no memory leaks = robust mitigation

#### Prevents

- Placing and executing shellcode at writable pages: recv() buffers, javascript strings, ...
- Old type of exploits need to be re-written

# Mitigations

#### ASLR

- Randomizes:
  - Modules base address
  - Heap allocations
  - Stack base
  - TEB address
- Prevents
  - Fixed address assumptions...
    - F.i. ROP or return into ntdll for DEP bypass (Uninformed – technique).

# Mitigations

#### ASLR



Region	Entropy
Image	8 bits
Неар	5 bits
Stack	14 bits

# Mitigations

- /SAFESEH (also referred as Software DEP)
  - When a SEH is called:
    - Verifies if SEH record is in executable memory
    - Verifies if the SEH function is inside a module
    - Also verifies if the function is defined in the module Safe SEH table located at the PE binary
  - If the PE file was not compiled with /SAFESEH it allows the handler execution
    - Attackers are taking advantage of this "app compat" scenario
    - 64 bits processes SEH works differently...

# Mitigations

#### SEHOP

- 32 bits OS mitigation
- When an exception happens the SEH chain is verified.
  - Last SEH record points to ntdll!FinalExceptionHandler? (SEHOP without ASLR = no sense)
    - Yes? chain is not corrupted
    - No? chain is corrupted and could lead to untrusted code execution... stop the process!

# Exploits

#### SEHOP operation record overwrite

#### Valid SEH Chain



Invalid SEH Chain N H app!\_main+ox1c 0x41414141?

#### Can't reach validation frame!



#### Purpose of EMET:

- Break current exploitation techniques, exploits and shellcodes
- Bring to older OS current mitigations
- Per process opt-in
- Test potential future OS mitiations
- Some of the mitigations can by bypassed.
  - Yes , we know... but they will break exploits as they are publicly developed currently.

#### ■ V1.02

- Free (with limited support)
- Published on November
- Mitigations:
  - Enable DEP per process
  - SEHOP (similar but no the same)
  - Common memory pages pre-allocation



#### DEP per process

- Easy one... enable DEP if SetProcessDepPolicy() exists in the system
- API available at XPSP3 +, Vista SP1+ and 2k8/Win7

#### SEHOP

- Similar to the OS SEHOP version
- OS version final «handler» is ntdll!FinalExceptionHandler
- EMET's version final «record» is at a random offset of a random page.
- A Vector Exception Handler (VEH) validates the exception chain before any SEH is executed
- Works downlevel!

### Common memory pages pre-allocation

- Common browser exloits rely on fixed addresses to be allocated... oxococococ, oxodododod, ...
- Pages address can be added at the registry
- Pre-allocate them as PAGE\_NOACCESS so a heap spray cannot get them
- NULL allocation in case a new technique for allocating it comes into play

- Next version...
  - Available soon
  - Potential new mitigations in next version:
    - EAT Hardware breakpoint mitigation
    - Stack pivot mitigation
    - Mandatory ASLR
  - Some can be circunvented... we know how to do it... but breaks a lot of currnet shellcodes/exploits
     ③



### EAT hardware breakpoint mitigation

- Metasploit's and most shellcodes do:
  - Go to the TEB fs:[o]
  - Grab a pointer to the PEB TEB[30] fs:[30]
  - Go the Ldr structures at the PEB
    - Linked list of loaded modules
    - Contains name and image base
  - Loop through all modules looking for specific exported functions at the Export Addess Table (EAT)
  - Once found go to EAT.AddressOffunctions[offset] for the address of the required function

### EAT hardware breakpoint mitigation

- X86 has 4 debug registers... acting as data breakpoints
- On windows 4 per thread since they are saved when OS switch to other tasks
- Place a 4 byte read data breakpoint at:
  - Kernel32!EAT.AddressOffunctions
  - Ntdll!EAT.AddressOfFunctions
- A VEH will monitor accesses to these: single step exception
  - Is eip inside a module? ③



### Stack pivot mitigation

- Hook some interesting functions: NtAllocateVirtualMemory, NtProtectVirtualMemory, LdrLoadDll, NtSetInformationProcess, ...
- At the hook check:
  - Is esp inside the bounds defined at the TEB: StackBase, StackLimit?
- Mitigates all aurora exploits I have seen ③



#### Mandatory ASLR

- ASLR aware modules contains DYNAMIC BASE on its DII Characteristics at the PE Header.
- Hook ntdll!LdrLoadDll, check if the module is ASLR aware.
  - Yes? Allow the dll load
  - No? Allocate a page at its imagebase and let the OS figure out if it can relocate it <sup>(C)</sup>
- Only works for dll loads after EMET has been loaded and not for imports...

# Greetings

 Thank you guys!
 MSRC/MSEC (former SWI): Andrew Roths, Matt Miller, Damian Hasse and Ken Johnson.

# Questions

Now or...

Twitter: fjserna http://zhodiac.hispahack.com