OWASP Slovenia

Enhanced Mitigation Experience Toolkit

Edi Strosar

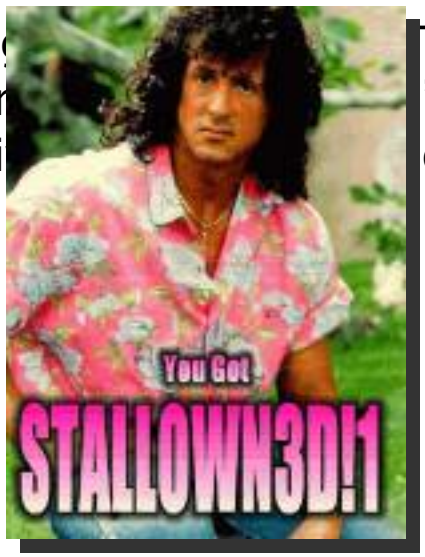
Ljubljana, 15.9.2011

Who's that dude?

- I am not:
  - CEH, CISSP, CISA, CCSP, MVP, RHCSS, Security+, GSEC, SCNP, NSCP, CCSA, CCSE, OSCP, ASS
  - LulzSec, Antisec, Anonymous, Ac1dB1tch3z, ZF0
- I am:
  -  @EdiStrosar
  - Hunting EIP for fun and challenge
  - No bugs will be killed today. Sorry guys :P

Kaj je EMET?

- Enhanced Miti Toolkit je program, ki pr ” preko 0-day in n-day ranlji omnilnika



EMET +

- Brezplačen in preprost za uporabo
- Uradno podprt s strani razvijalca
- V “arhaične” različice Windows (XP/2003) vnaša nekatere naprednejše MP mehanizme
- Blaži malomarnost/ignoranco programerjev glede (ne)uporabe stikal v prevajalniku
- Skape je v razvojni ekipi

EMET –

- Microsoft made it (can you spot the irony ?)
- For user space code only
- Nekateri programi so nekompatibilni z določenimi mehanizmi
- EMET != AV (yep, this is actually good)

EMET != čarobna paličica

- Znani so načini za premostitev vseh mehanizmov iz EMET repertoarja, vendar trenutno ni javno objavljene kode, ki bi to opravila zanesljivo in v eni potezi.
- Dejstvo: 0-day postopki za premagovanje MP mehanizmov imajo danes večjo vrednost kot 0-day ranljivosti same.

EMET != čarobna paličica

Delovanje

- Uporablja infrastrukturo “shim”, ki je uveljavljen način za hendlanje Application Compatibility skladnosti
- “Shim”
  - DLL, ki transparentno prestreza API klice (API hooking)
- EMET shim engine: implementiran v knjižnici emet.dll, ki je vrinjena v vse varovane procese

Memory protections (Windows)

- Integracija v sistem:
  - Compiled-in
  - Injected (DLLs, drivers)
  - Kernel modification* (PaX)
- ASLR (/DynamicBase)
- DEP (/NXCOMPAT)
- Guard Stack (/GS)
- SafeSEH (/SAFESEH)
- SEHOP (registry)
- MP mehanizmi zagotavljajo optimalno zaščito le pri komplementarni uporabi!

Memory protections (EMET)

- DEP, ASLR, SEHOP (system)
- Pseudo-mitigations (per-process):
  - Dynamic DEP
  - Mandatory ASLR
  - Heap spray preallocation
  - NULL page preallocation
  - EAT access filtering
  - Bottom-up randomization
- Novi MP mehanizmi bodo vključeni v EMET skladno z razvojem novih Xdev tehnik

ASLR / Rebasing

- Osnovni pogoj za izkoriščanje memory corruption vrzeli:
  - Predvidljiv naslov modula, pomnilniške strukture ali izvršljive kode
  - Statični moduli, memory disclosure leak, spraying, brute force*
- ASLR v platformo vnaša določeno entropijo:
  - **images** (dll/exe) ← rand. 2^8 (256 naslovov)
  - **PEB** / **TEB** ← rand. 2^4 (16 naslovov)
  - **sklad** ← rand. 2^{14} (16384 naslovov)
  - **kopica** ← rand. 2^5 (32 naslovov)
- Rebasing ← rand. 2^6 (64 naslovov)
  - Preferred Image Base naslov v modulu ni definiran
  - PE loader naloži modul na "random" lokacijo
- Mandatory ASLR* ← rand. 2^4 (16 naslovov)
  - Prealocira Image Base naslov (via ntdll!LdrLoadDll hook) in "prisili" modul v rebase

DEP / NX

- x86 ne ločuje med podatki in kodo
- NX preprečuje izvajanje kode na straneh, ki niso eksplicitno executable
- Hardware DEP (x64 + x86/PAE)
- Software DEP (x86)
  - dodatno preverjanje kazalca EH (preveri ali je destinacija EH mapirana MEM_IMAGE)
  - stack / heap sta še vedno izvršljiva
- DEP policies:
  - **OptIn** – sistemske aplikacije in aplikacije prevedene s stikalom /NXCOMPAT
  - **OptOut** – vse aplikacije na sistemu (razen eksplicitno navedenih izjem)
  - **AlwaysOn** – vse aplikacije na sistemu (brez izjem)
  - **AlwaysOff** – DEP je izključen
  - **Permanent DEP** – SetProcessDEPPolicy()
- Dynamic DEP:
  - emet.dll kliče kernel32!SetProcessDEPPolicy znotraj varovanega procesa

SafeSEH + SEHOP

- SafeSEH
  - ob prevajanju se v PE zapiše tabela z naslovi veljavnih EH procedur (naslova POP/POP/RET instrukcij ni v tabeli)
- SEHOP
  - preverjanje integritete verige SEH (preveri ali EH zadnjega člena verige vsebuje naslov ntdll!FinalExceptionHandler)
- SEHOP (EMET)
  - v SEH chain vrine Final Record z naključno vrednostjo (cookie), ki se preverja pred dispečanjem izjeme

Guard Stack

- Preprečuje prepisovanje povratnega naslova
  - Prolog = [→buffer][vars]:[cookie]:[EBP]:**[RET]**:[args]
  - Epilog = preverjanje vrednosti [cookie]
- Reorganizacija okvira sklada
  - Lokalne spremenljivke [vars] se preslikajo pred [buffer] kar preprečuje korupcijo kazalcev na funkcije

EMET pseudo mitigations

- NULL page preallocaton
  - Preprečuje izkoriščanje preko dereferenciranega kazalca NULL
- Heap spray preallocation
  - Prealocira najpogostejše “duality” (address/NOP) heap spray vrednosti
- EAT access filtering
  - Blokira izvajanje dinamične shellcode
  - HW BP na naslovu EAT.AddressOfFunctions polja v kernel32/ntdll
  - Ko se sproži breakpoint preveri ali je izvor kode, ki želi dostop do EAT mapiran MEM_IMAGE (.code segment)
- Bottom-up randomization
  - Alokacija pomnilnika za “bottom-up” strukture je včasih predvidljiva
  - 8-bitov entropije za Base naslove “bottom-up” alokacij (images/stack/heap)

Mitigation verification

- Process Explorer (Sysinternals/Microsoft)
- DEPend (Pax Team)
- Mona (Corelan Team)
- Narly (Nephi Johnson)
- Looking Glass (Errata Security)
- BinScope (Microsoft/SDL Initiative)

Let there be buffer overflow.

And there was stack buffer overflow.

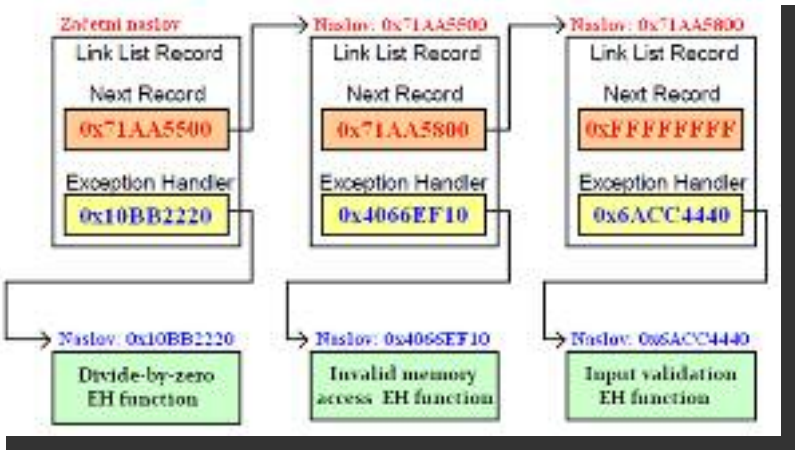
- Stack-based BOF nastane, ko je količina vhodnih podatkov večja od rezerviranega medpomnilnika
- Zapisovanje podatkov preko meja bufferja povzroči korupcijo struktur na skladu (RET, SEH, kazalci …)

Stack-based BOF
(mrinfo / XP)

Then he created SEH.

- SEH je nativni mehanizem za upravljanje izjem v Windows okolju
- Mehanizem je implementiran na strukturi povezane liste, ki povezuje ER elemente
- Vsak element v verigi SEH sestavljata:
  - Kazalec do procedure za obravnavo izjeme (EH)
  - Kazalec do naslednjega elementa v zaporedju (NR)

For exceptions to walk the SEH chain.



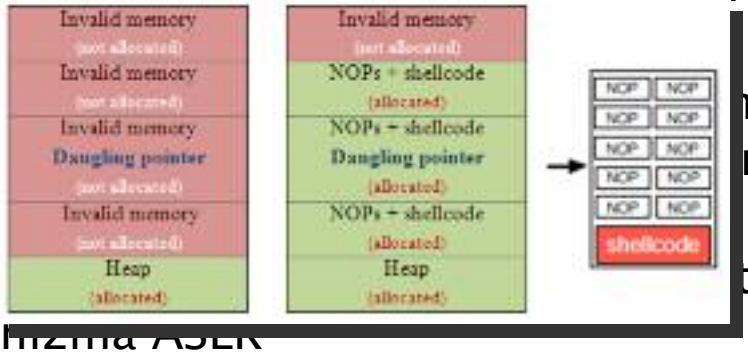
And it was good.

- Omogoča premostitev mehanizma Guard Stack
  - EH rutina se izvede pred fazo epiloga
- SEH smashing:
  - Prepiši EH z naslovom POP/POP/RET
    - Izvajanje se nadaljuje v SEH bloku, točno na naslovu NR
  - Prepiši NR z instrukcijo JMP+6

SEH corruption
(mrinfo / XP)

Be fruitful, and multiply, and heap spray.

- Namen razprševanja je večkratno repliciranje bloka
- Z zas … možnost, da bc … rovanem prost
- Bonu: … titev mehanizma ASLR



And it was so.

```
<SCRIPT language="javascript">
var nop = ("NOPs");
var shellcode = ("shellcode");

var nopsled = nop;
while(nopsled.length < 40000)        /* 0x40000 = 256KB NOPs */
  {
    nopsled+=nopsled;
  }

var spray = new Array();
for(i=0; i<1000; i++)                /* 1000 x (nopsled + shellcode) */
  {
    spray[i] = nopsled+shellcode;    /* 1000 x ~256KB = ~256MB zasebnega pomnilnika */
  }
</SCRIPT>
```

For every use after free known to 

```
7DC98C83  8B01        MOV EAX,DWORD PTR DS:[ECX]
7DC98C85  FF50 34     CALL DWORD PTR DS:[EAX+34]
7DC98C88  8B40 0C     MOV EAX,DWORD PTR DS:[EAX+C]
7DC98C8B  C3          RETN
7DC98C8C  90          NOP
7DC98C8D  90          NOP
7DC98C8E  90          NOP
7DC98C8F  90          NOP
7DC98C90  90          NOP
7DC98C91  8B41 18     MOV EAX,DWORD PTR DS:[ECX+18]
7DC98C94  84C0        TEST AL,AL

Registers (FPU)
EAX 01A71330
ECX AAAAAAAA
EDX 01A768B0
EBX AAAAAAAA
ESP 0013E35C
EBP 0013E37C
ESI 01A70900
EDI FFFFFFFF
EIP 7DC98C83 mshtml.7DC98C83
```

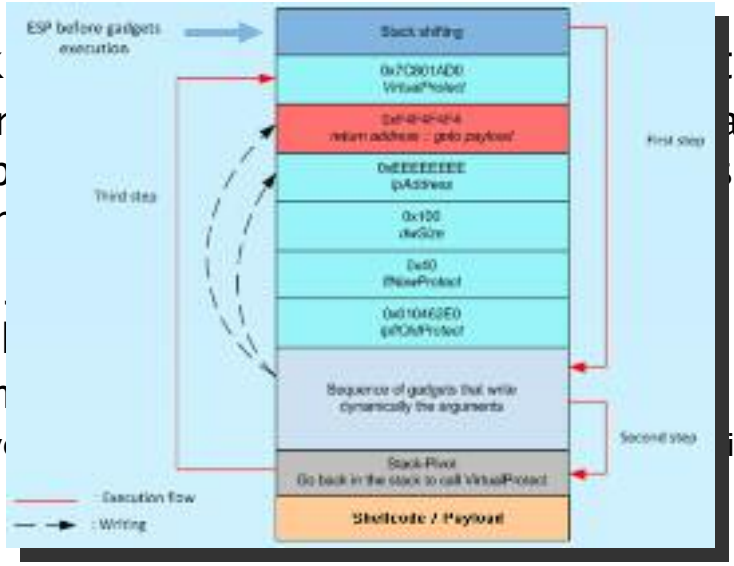
Heap spray
(Aurora)

On the 7th day he created ROP.

- Return-Oriented Programming / Code reuse
  - Omogoča premostitev mehanizma DEP/NX
  - Zametki v ret2libc → shellcode not required
- ROP 'n' Roll:
  - Iz instrukcij v .code segmentu sestavi zaporedje opravil, ki se zaključujejo z RET (gadget). Opravila združi v verigo ROP (ROP chain).

And the earth was filled with 0-days.

- ROP izk … trukcij”
  - Če začr … a prvem zlogu b … strukcijo začnem
- Na x86 …
  - Variabil …
  - Unalign …
    - if nativ … ion not /N



ROP
(Wireshark / Win7)

Got EMET ?

Zahvala

- Didier Stevens
- Ivan Lefou
- Peter Van Eechoutte (Corelan team)
- Fermin J. Serna (EMET devteam)
- OWASP Slovenija

Slikovno gradivo

- Peter Van Eeckhoutte (photo on slide 7)
- Donny Hubener (image on slide 20)
- Axel Souchet (image on slide 28)
- Mladina d.d. (photo on slide 33)

Vprašanja?