Hunting In Memory

April 19, 2017

Jared Atkinson, Joe Desimone

ENDGAME.

Who are we

- Jared Atkinson
  - Defensive Service Technical Director - Specter Ops
  - Microsoft Cloud and Datacenter Management MVP (PowerShell)
  - Lead Developer PowerForensics
- Joe Desimone
  - Senior Malware Researcher - Endgame
  - Developer, Hunter, Reverse Engineer

SPECTER OPS

ENDGAME.

Overview

- Why hunting in memory is important
- Memory based attacker techniques
- Existing tools and approaches
- New powershell tool for hunting at scale

SPECTER OPS

ENDGAME.

Importance of Memory Hunting

- Memory resident malware has been in use for over a decade, and is now ubiquitous
- Once a staple of ‘APT’; now commonplace for crimeware
- Designed to evade PSPs and YOU
- Great signal to noise ratio; easy button hunting

Attacker Techniques

- Classic memory/shellcode injection
- Reflective DLLs
- Memory Module
- Process and Module Hollowing
- PEB Unlinking
- Gargoyle (ROP/APCs)

SPECTER OPS

ENDGAME.

Classic Injection

- OpenProcess - Grab handle to target process
- VirtualAllocEx - Allocate a new chunk of memory in targe
- WriteProcessMemory - Write the shellcode/payload into target
- CreateRemoteThread - Start a new thread to execute the payload

Classic Injection - Poison Ivy

```
push 40
push 3000
push dword ptr ss:[ebp+10]
push 0
push dword ptr ss:[ebp+C]
call dword ptr ds:[esi+B1]          [esi+B1]:VirtualAllocEx
push eax
lea edi,dword ptr ss:[ebp-4]
push edi
push dword ptr ss:[ebp+10]
push dword ptr ss:[ebp+14]
push eax
push dword ptr ss:[ebp+C]
call dword ptr ds:[esi+B5]          [esi+B5]:WriteProcessMemory
pop eax
```

SPECTER OPS

ENDGAME.

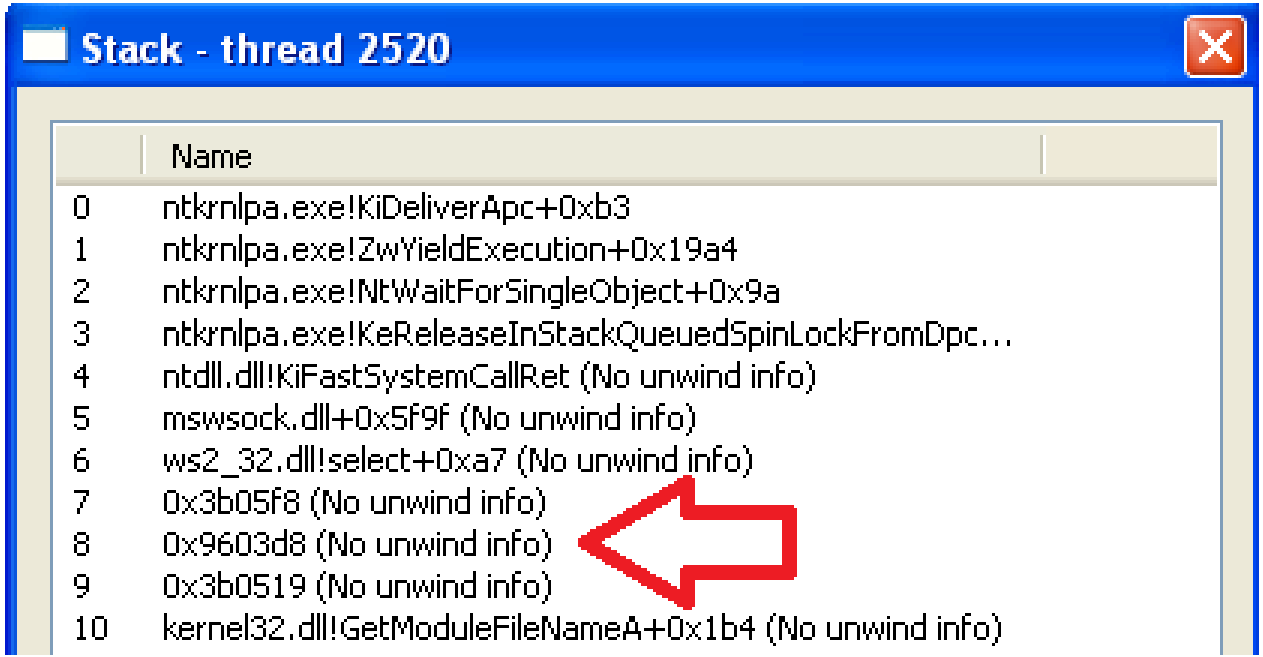
Poison Ivy

```
push 0
push dword ptr ss:[ebp-F80]
call dword ptr ds:[esi+C9]          [esi+C9]:CreateRemoteThread
push eax
push dword ptr ss:[ebp-F80]
call dword ptr ds:[esi+A1]          [esi+A1]:CloseHandle
pop eax
```

Poison Ivy

0x960000	Private	8 kB	RWX
0x970000	Private	4 kB	RWX
0x980000	Private	4 kB	RWX
0x990000	Private	4 kB	RWX
0x9a0000	Private	4 kB	RWX
0x9b0000	Private	4 kB	RWX
0x9c0000	Private	4 kB	RWX
0x9d0000	Private	4 kB	RWX
0x9f0000	Private	4 kB	RWX

Poison Ivy Thread



Reflective DLL Injection

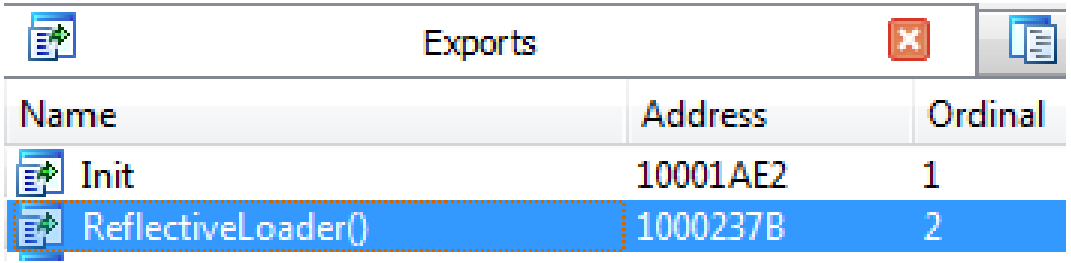
- DLL that maps itself into memory - original design and code by Steven Fewer [1]
- Handy from attacker perspective - makes for a ‘dumb’ injector
- No longer have to code in assembly (like PI)
- Very common technique (ex: meterpreter, powershell empire)
- Allocate memory, map sections, resolve imports, fixup relocations, call entry

[1] https://github.com/stephenfewer/ReflectiveDLLInjection

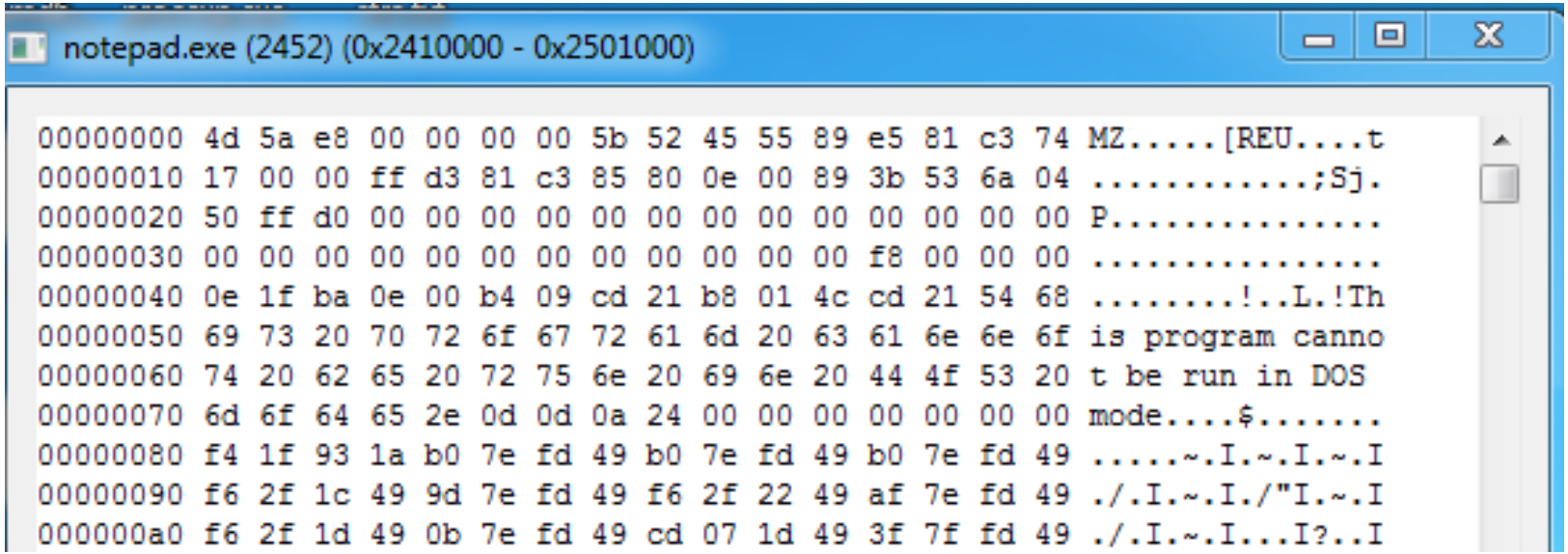
Meterpreter

- Classic DLL Reflection, such as meterpreter, is easy to find

0x2410000	Private: Commit	964 kB	RWX
0x1f20000	Private: Commit	396 kB	RWX
0x1d90000	Private: Commit	936 kB	RWX
0x220000	Private: Commit	128 kB	RWX



Meterpreter



SPECTER OPS

ENDGAME.

Memory Module

- Similar to Reflective technique, except loader does all the work [1]
- Payload DLL doesn’t need any special modifications
- Loader re-implements LoadLibrary(), but works on a buffer in memory
- Can map into local or remote process [2]
- Typical implementations avoid RWX

[1] Memory Module - https://github.com/fancycode/MemoryModule
[2] Manual Map - https://github.com/DarthTon/Blackbone

NetTraveler - Memory Layout

- Uses legitimate looking permissions

0x3f0000	Private	108 kB	RW
0x3f0000	Private: Commit	4 kB	RW
0x3f1000	Private: Commit	40 kB	RX
0x3fb000	Private: Commit	8 kB	R
0x3fd000	Private: Commit	52 kB	RW
0x40a000	Private: Reserved	4 kB	

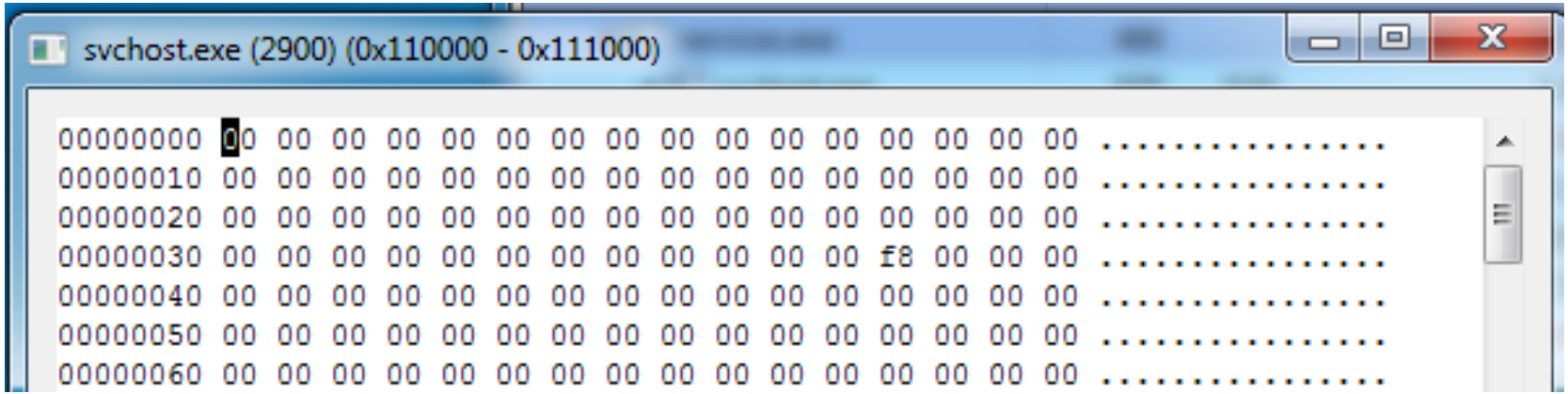
NetTraveler - Active Thread

```
14  winhttp.dll!WinHttpSendRequest+0x2fe6 (No unwind info)
15  winhttp.dll!WinHttpSendRequest+0x3069 (No unwind info)
16  winhttp.dll!WinHttpSendRequest+0x212 (No unwind info)
17  0x3f6e82 (No unwind info)
18  0x3f679e (No unwind info)
19  0x3f5d65 (No unwind info)
```

Winnti - Memory Layout

◢ 0x110000	Private	28 kB	RWX
0x110000	Private: Commit	4 kB	RWX
0x111000	Private: Commit	12 kB	RX
0x114000	Private: Commit	4 kB	R
0x115000	Private: Commit	4 kB	RW
0x116000	Private: Rese...	4 kB	

Winnti - Header Wipe



Process Hollowing

- Create new, suspended process
- Allocate new memory, unmap (hollow) existing code
- Write payload
- Redirect execution - SetThreadContext() and ResumeThread()
- Stealthy variants
  - Create/Map sections to avoid WriteProcessMemory
  - Modify entry point instead of SetThreadContext

SPECTER OPS

ENDGAME.

DarkComet - Process Hollowing

SPECTER OPS

ENDGAME.

DarkComet

Name	Base address	Size	Description
dark.exe	**0x400000**	**732 kB**	**WEbhu2 NXchu6**
advapi32.dll	0x76d60000	640 kB	Advanced Windows 32 F

0x3e0000	Mapped: Commit	96 kB	R	C:\Users\Joe
0x400000	Private: Commit	732 kB	RWX	
0x4c0000	Mapped: Commit	28 kB	R	

SPECTER OPS

ENDGAME.

Module Overwriting

- Up until now, all examples have lead to non-image backed code executing
- Module Overwriting avoids this, making it more difficult to detect
- Flame and Careto are examples
- Map an unused module into target process
- Overwrite legitimate module with payload
- Odinaff had a similar trick but overwrote its own executable

Odinaff

In Memory

MajorLinkerVersion	0A
MinorLinkerVersion	00
SizeOfCode	00001200
SizeOfInitializedData	00001200
SizeOfUninitializedData	00000000
AddressOfEntryPoint	00002180
FileAlignment	00000200
MajorOperatingSystemVers...	0005
MinorOperatingSystemVer...	0001

On Disk

MajorLinkerVersion	06
MinorLinkerVersion	00
SizeOfCode	00002000
SizeOfInitializedData	00006000
SizeOfUninitializedData	00000000
AddressOfEntryPoint	00001D04
FileAlignment	00001000
MajorOperatingSystemVers...	0004
MinorOperatingSystemVer...	0000

SPECTER OPS

ENDGAME.

PEB Unlinking

- Not an in memory technique, but rather an evasion
- Hide loaded DLL from security products, admins, hunters
- HackingTeam used this technique in their RAT
- Flame also unlinked shell32.dll
- To find peb unlinking, you could compare what the Win32 API reports as ‘loaded’ versus what you find is actually loaded with VirtualQuery/GetSectionName

SPECTER OPS

ENDGAME.

Gargoyle

- Technique developed by Josh Lospinoso to hide injected code from security products
- Payload lies dormant, with read only permissions
- Periodically ‘wakes up.’ Sets payload executable with an asynchronous procedure call and ROP. Permissions reverted, cycle repeats.
- https://jlospinoso.github.io/security/assembly/c/cpp/developing/software/2017/03/04/gargoyle-memory-analysis-evasion.html

Available Tools

- Volatility / malfind
- GRR
- Rekall
- inVtero

SPECTER OPS

ENDGAME.

Detecting Injection

w/ PowerShell

ENDGAME.

PSReflect

- PowerShell module written by Matt Graeber (@mattifestation)
  - https://github.com/mattifestation/PSReflect
- Avoids the compilation artifacts associated with P/Invoke
  - IMO the cleanest way to deal with Win32 API from PowerShell
- Library to abstract the complexities of calling Win32 functions via Reflection
- Intuitive “domain specific language” for defining enums, structs, and P/Invoke function signatures
- Must include PSReflect code in your scripts/modules

Get-InjectedThread

- Built on PSReflect

- PowerShell function to identify injected threads via detection methodology:
  - Use Windows Toolhelp API to get all threads
  - Iterate through each thread
  - Identify the thread's Base (Memory) Address
  - Query the memory page for which the Base Address belongs to
  - Check if the memory page's state is MEM_COMMIT
  - Check if the memory page's type is not MEM_IMAGE

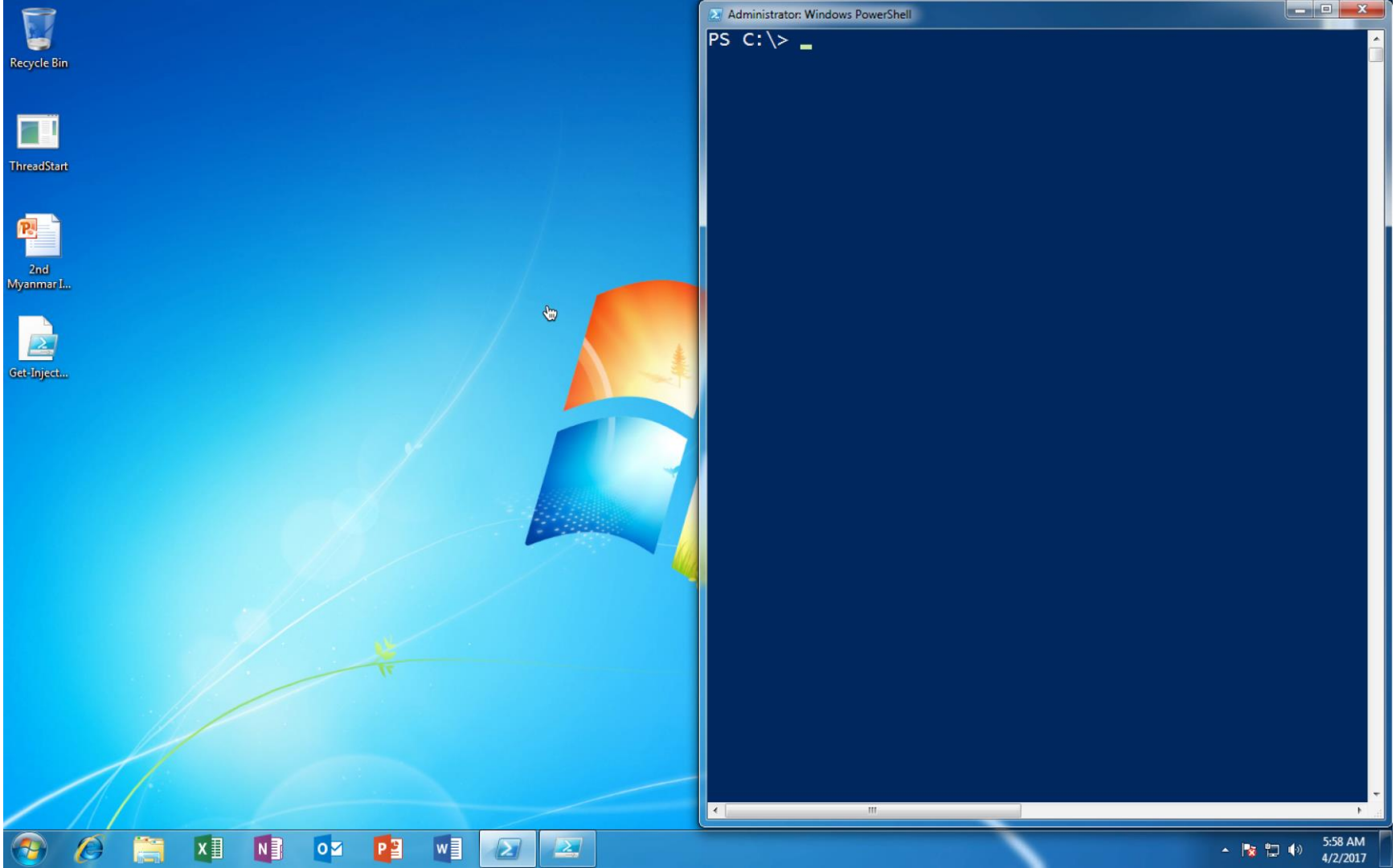
- Returns details regarding offending process and thread
  - Check memory page permissions
  - Look for unnecessary privileges or integrity level
  - Identify abnormal user tokens

Get-InjectedThread Output

- Injected Process Information
  - Process Id
  - Name
  - File Path (PEB and EPROCESS)
  - Command Line
- Thread Information
  - Thread Id
  - Unique Thread Token
  - Base Priority
  - Does thread have unique token?
- Memory Segment
  - Base Address
  - Size
  - Protection
  - State
  - Type
  - First 100 Bytes
- Token (Thread or Process)
  - Integrity Level
  - Enabled Privileges
  - SID / UserName
  - Logon Session Start Time
  - Logon Type
  - Authentication Package

SPECTER OPS

ENDGAME.


Recycle Bin
ThreadStart
2nd Myanmar I...
Get-Inject...
Administrator: Windows PowerShell
PS C:\>
5:58 AM
4/2/2017

Our Threat - 9002 Trojan

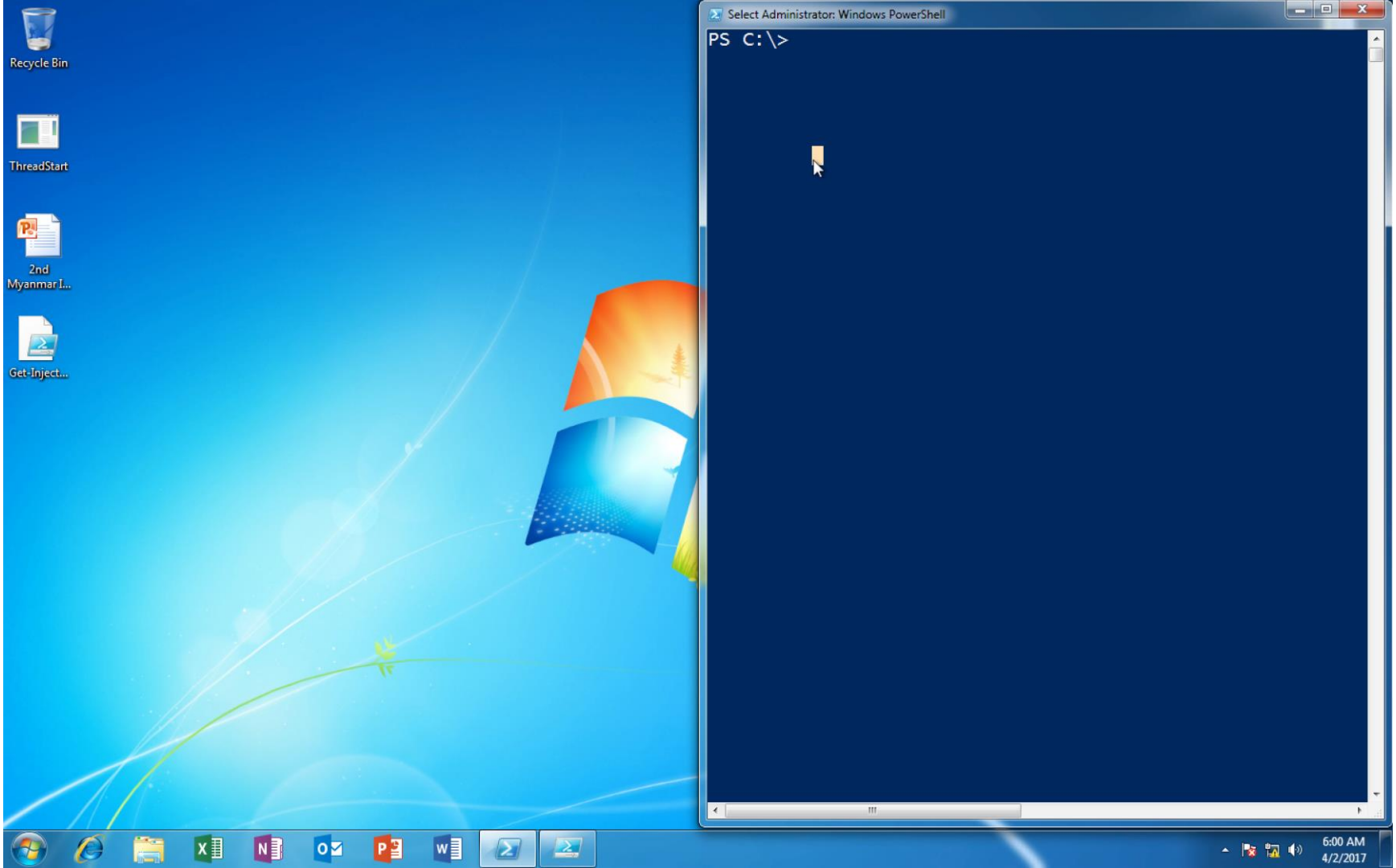
- Delivered as .zip file via Google Drive

- Zip archive contains one file
  - 2nd Myanmar Industrial Resource Development Symposium.exe
  - File has PowerPoint icon to trick users into opening

- Drops files upon execution
  - %USERPROFILE%\<random>\RealNetwork.exe (Legitimate Application)
  - %USERPROFILE%\<random>\main.dll (Loaded by MPAMedia.dll)
  - %USERPROFILE%\<ramdom>\mpaplugins\MPAMedia.dll (DLL Side Loading)
  - Ppt
    - Opened in PowerPoint to keep up the ruse

SPECTER OPS

ENDGAME.

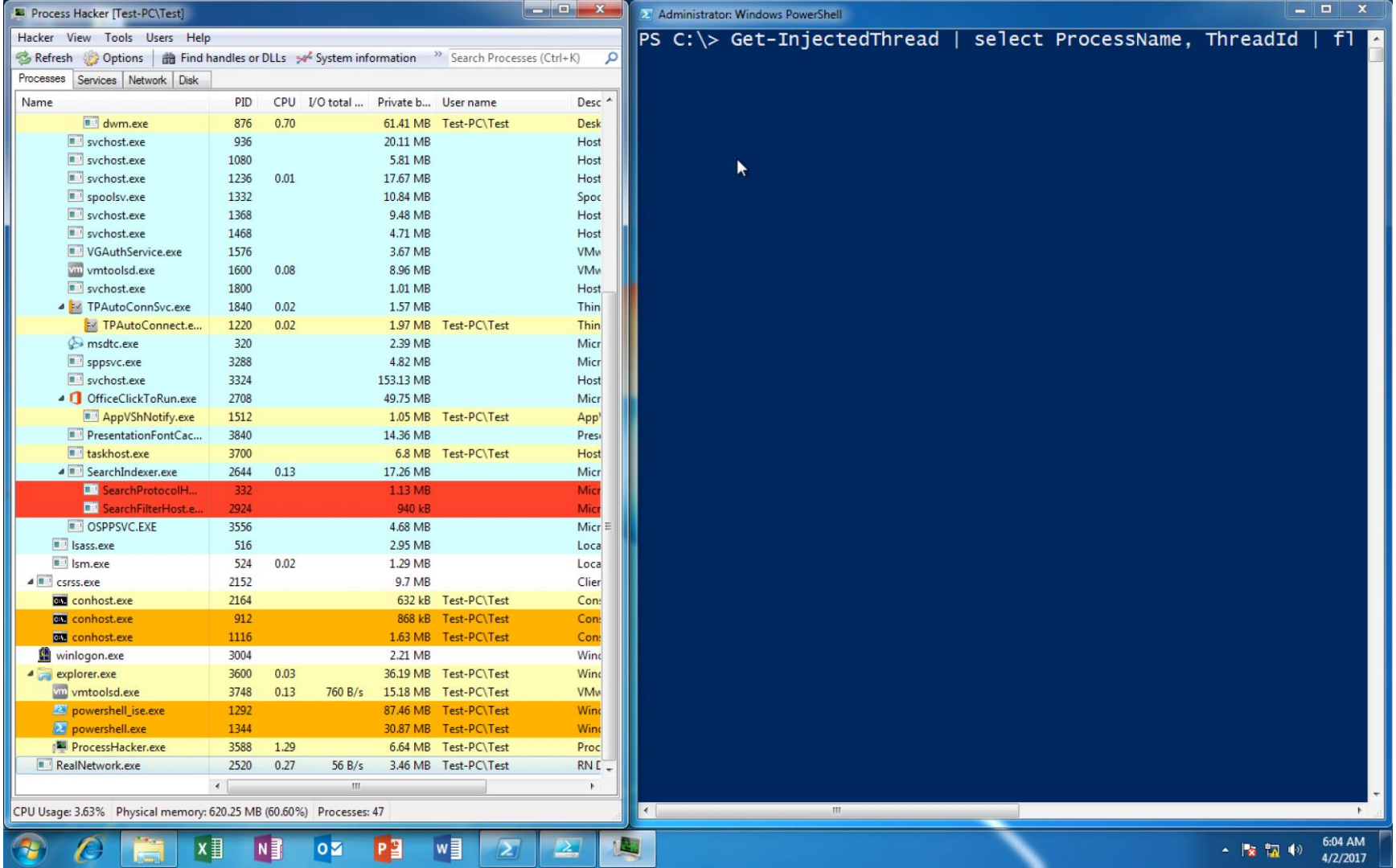

Recycle Bin
ThreadStart
2nd Myanmar I...
Get-Inject...
Select Administrator: Windows PowerShell
PS C:\>
6:00 AM
4/2/2017

Response

- Kill Thread
  - Stop-Thread
  - Built on Window's TerminateThread API
- Process Minidump
  - Out-Minidump (PowerSploit)
  - https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Out-Minidump.ps1
- Thread Dump
  - Dump-Thread

SPECTER OPS

ENDGAME.


Process Hacker [Test-PC\Test]
Hacker View Tools Users Help
Refresh Options Find handles or DLLs System information Search Processes (Ctrl+K)
Processes Services Network Disk
Name PID CPU I/O total ... Private b... User name Desc
dwm.exe 876 0.70 61.41 MB Test-PC\Test Desk
svchost.exe 936 20.11 MB Host
svchost.exe 1080 5.81 MB Host
svchost.exe 1236 0.01 17.67 MB Host
spoolsv.exe 1332 10.84 MB Spoo
svchost.exe 1368 9.48 MB Host
svchost.exe 1468 4.71 MB Host
VGAuthService.exe 1576 3.67 MB VMw
vmtoolsd.exe 1600 0.08 8.96 MB VMw
svchost.exe 1800 1.01 MB Host
TPAutoConnSvc.exe 1840 0.02 1.57 MB Thin
TPAutoConnect.e... 1220 0.02 1.97 MB Test-PC\Test Thin
msdtc.exe 320 2.39 MB Micr
sppsvc.exe 3288 4.82 MB Micr
svchost.exe 3324 153.13 MB Host
OfficeClickToRun.exe 2708 49.75 MB Micr
AppVShNotify.exe 1512 1.05 MB Test-PC\Test App
PresentationFontCac... 3840 14.36 MB Pres
taskhost.exe 3700 6.8 MB Test-PC\Test Host
SearchIndexer.exe 2644 0.13 17.26 MB Micr
SearchProtocolH... 332 1.13 MB Micr
SearchFilterHost.e... 2924 940 kB Micr
OSPPSVC.EXE 3556 4.68 MB Micr
lsass.exe 516 2.95 MB Loca
lsm.exe 524 0.02 1.29 MB Loca
csrss.exe 2152 9.7 MB Clier
conhost.exe 2164 632 kB Test-PC\Test Cons
conhost.exe 912 868 kB Test-PC\Test Cons
conhost.exe 1116 1.63 MB Test-PC\Test Cons
winlogon.exe 3004 2.21 MB Winl
explorer.exe 3600 0.03 36.19 MB Test-PC\Test Winl
vmtoolsd.exe 3748 0.13 760 B/s 15.18 MB Test-PC\Test VMw
powershell_ise.exe 1292 87.46 MB Test-PC\Test Winl
powershell.exe 1344 30.87 MB Test-PC\Test Winl
ProcessHacker.exe 3588 1.29 6.64 MB Test-PC\Test Proc
RealNetwork.exe 2520 0.27 56 B/s 3.46 MB Test-PC\Test RN D
CPU Usage: 3.63% Physical memory: 620.25 MB (60.60%) Processes: 47
Administrator: Windows PowerShell
PS C:\> Get-InjectedThread | select ProcessName, ThreadId | fl
6:04 AM
4/2/2017

ENDGAME.

Questions?