A Process Is No One:

Hunting for Token Manipulation

@jaredcatkinson

- Adversary Detection Technical Lead @ SpecterOps
- Developer:
  - PowerForensics
  - Uproot
  - ACE
  - PSReflect-Functions
- Microsoft MVP - Cloud and Data Center Management (PowerShell)
- Former:
  - U.S. Air Force Hunt Team
  - Veris Group's Adaptive Threat Division

@robwinchester3

- Adversary Detection Lead @ SpecterOps
- Contributor:
  - Co-author of ACE
  - HELK
- Former:
  - U.S. Air Force Red Team
  - Veris Group's Adaptive Threat Division

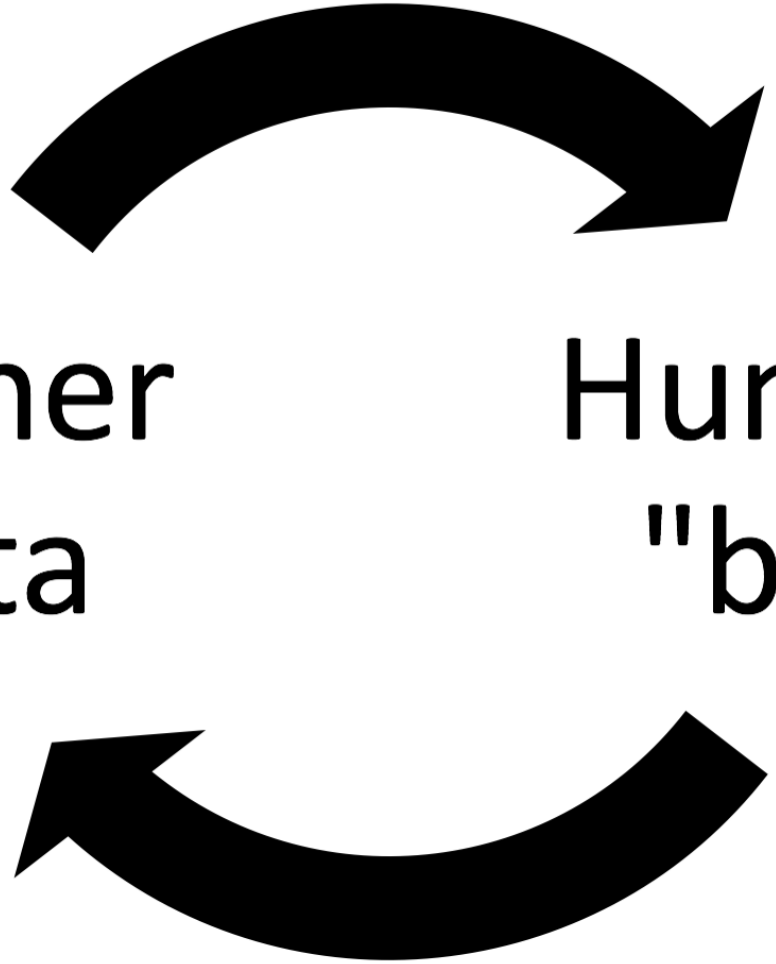
Overview

- What is “Hunt”?
- Attacker TTP
- Creating a Useful Hypothesis
- Case Study
  - Detecting Access Token Manipulation

What is “Hunt”?

- Actively searching for malicious activity in the environment that has evaded current in place defenses
- Rooted in the assume breach mentality

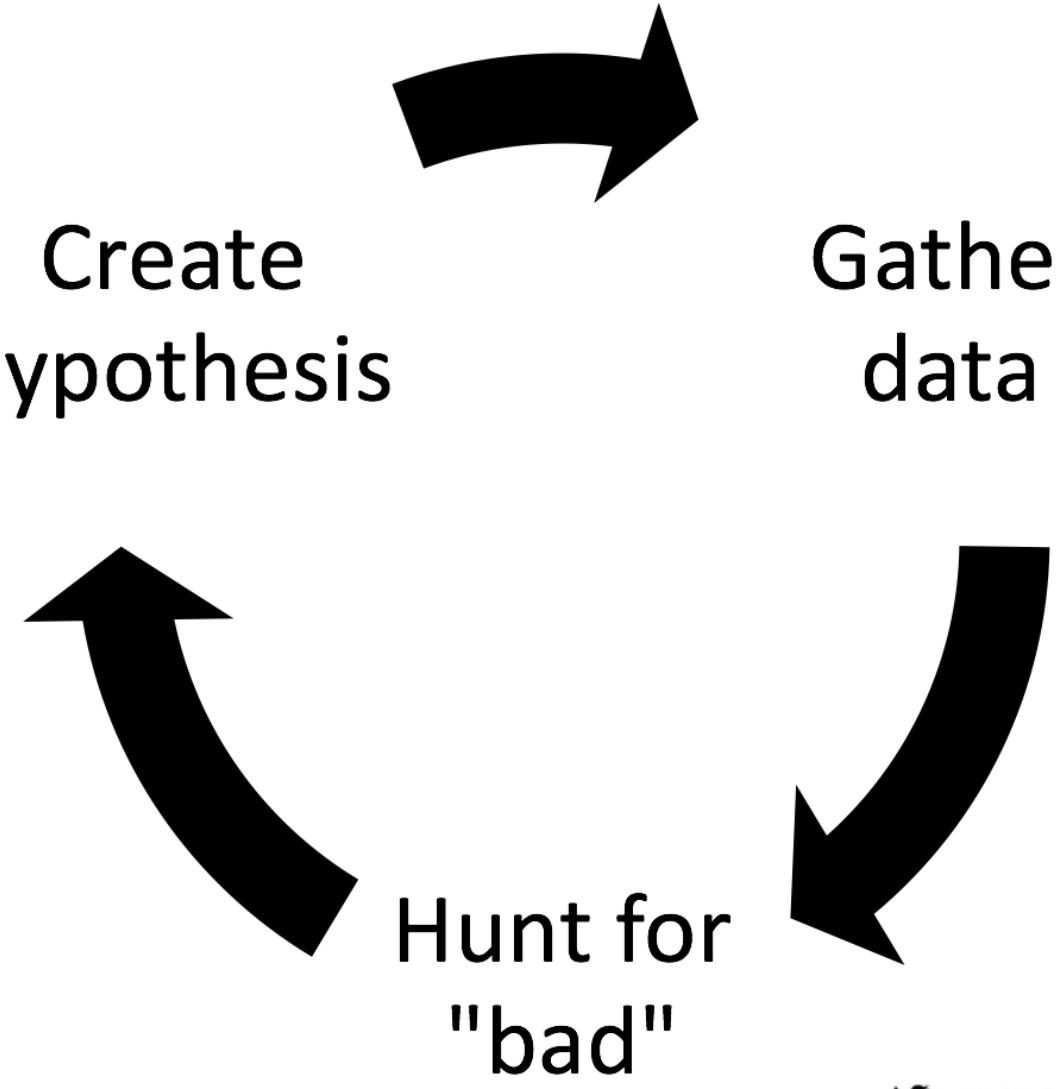
Generic Hunt Process



"Generic" Hunt Process Problems

- Gather data
  - What data should we collect?
  - Why are we collecting that data?
  - SIEMs, like Splunk, are expensive…
  - Open source alternatives, like ELK, are technically free but cost time
- Hunt for bad
  - What are you looking for in the gathered data?
  - What is "good" activity that can safely be filtered?
  - How much time do you have to search through the data?
  - Must balance "Hunting" time with "Investigation" time

Hypothesis Driven Hunting



Hypothesis Driven Hunting Benefits

- Focuses data collection effort
- Provides a specific goal for the team
- Combats data collection for data collection sake
- Helps eliminate “analysis paralysis”
- Track hypotheses over time
  - What Tactics are you not covering
  - Identify knowledge gaps that training can help fill
  - Inform purchasing decisions moving forward

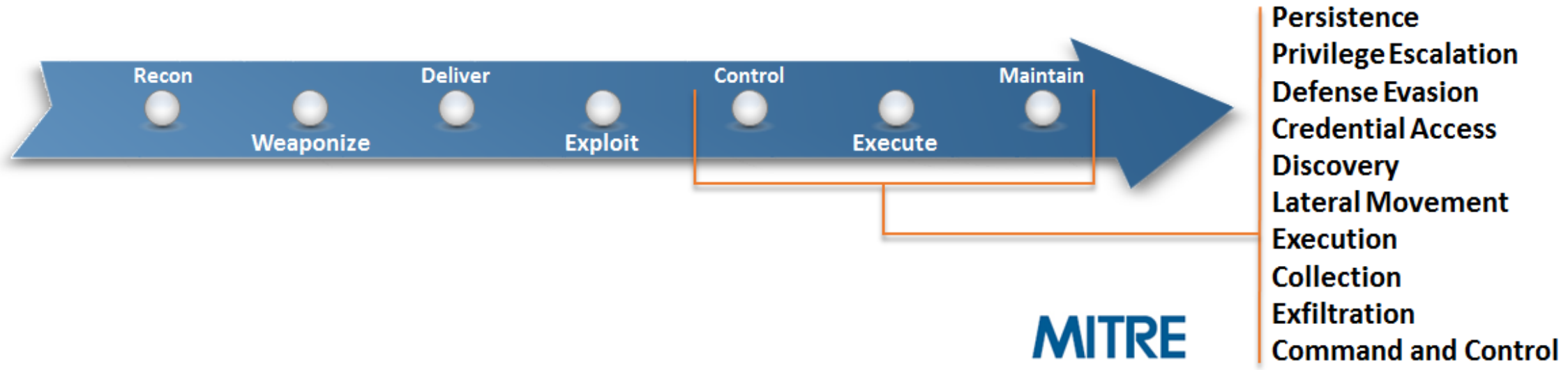
Great, so how do I make a hypothesis?

- This is the focus of the presentation!
- We will walk you through how to make a hypothesis that will actually result in something tangible
- Will do a practical demonstration on using this process to create and execute a hypothesis for Access Token Manipulation

First, what are we looking for?

- Assume Breach
  - Tons of organizations have been breached
  - It's a matter of **WHEN** not **IF** a breach will occur
- Focus on post-exploitation activity
  - Most in place defenses tools focus on preventing/detecting the initial attack
    - Firewalls
    - Anti-Virus
    - Intrusion Detection/Prevention Systems
  - If you can stop the attack before the objective is achieved, the attack is still stopped

MITRE Cyber Attack Lifecycle



https://attack.mitre.org/wiki/Main_Page

MITRE ATT&CK Framework

- A body of knowledge for cataloging an adversary activity during the attack cycle
  - Similar to how OWASP defines application security vunerabilities
  - Used as a reference for both offense and defense
    - Red Canary's Atomic Red Team Project (https://github.com/redcanaryco/atomic-red-team)
  - Includes Windows, Unix, and MacOS TTPs
- Categories loosely correspond with the attack cycle
  - Persistence
  - Lateral Movement
  - etc.

MITRE ATT&CK Framework

Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Execution	Collection	Exfiltration	Command and Control
Accessibility Features	Accessibility Features	Binary Padding	Brute Force	Account Discovery	Application Deployment Software	Command-Line Interface	Audio Capture	Automated Exfiltration	Commonly Used Port
AppInit DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	Application Window Discovery	Exploitation of Vulnerability	Execution through API	Automated Collection	Data Compressed	Communication Through Removable Media
Authentication Package	Bypass User Account Control	Code Signing	Credential Manipulation	File and Directory Discovery	Logon Scripts	Execution through Module Load	Clipboard Data	Data Encrypted	Connection Proxy
Basic Input/Output System	DLL Injection	Component Firmware	Credentials in Files	Local Network Configuration Discovery	Pass the Hash	Graphical User Interface	Data Staged	Data Transfer Size Limits	Custom Command and Control Protocol
Bootkit	DLL Search Order Hijacking	Component Object Model Hijacking	Exploitation of Vulnerability	Local Network Connections Discovery	Pass the Ticket	InstallUtil	Data from Local System	Exfiltration Over Alternative Protocol	Custom Cryptographic Protocol
Change Default File Association	Exploitation of Vulnerability	DLL Injection	Input Capture	Network Service Scanning	Remote Desktop Protocol	MSBuild	Data from Network Shared Drive	Exfiltration Over Command and Control Channel	Data Encoding
Component Firmware	File System Permissions Weakness	DLL Search Order Hijacking	Network Sniffing	Peripheral Device Discovery	Remote File Copy	PowerShell	Data from Removable Media	Exfiltration Over Other Network Medium	Data Obfuscation
Component Object Model Hijacking	Legitimate Credentials	DLL Side-Loading	Two-Factor Authentication Interception	Permission Groups Discovery	Remote Services	Process Hollowing	Email Collection	Exfiltration Over Physical Medium	Fallback Channels
DLL Search Order Hijacking	Local Port Monitor	Disabling Security Tools		Process Discovery	Replication Through Removable Media	Regsvcs/Regasm	Input Capture	Scheduled Transfer	Multi-Stage Channels
External Remote Services	New Service	Exploitation of Vulnerability		Query Registry	Shared Webroot	Regsvr32	Screen Capture		Multiband Communication
File System Permissions Weakness	Path Interception	File Deletion		Remote System Discovery	Taint Shared Content	Rundll32	Video Capture		Multilayer Encryption
Hypervisor	Scheduled Task	File System Logical Offsets		Security Software Discovery	Third-party Software	Scheduled Task			Remote File Copy
Legitimate Credentials	Service Registry Permissions Weakness	Indicator Blocking		System Information Discovery	Windows Admin Shares	Scripting			Standard Application Layer Protocol
Local Port Monitor	Web Shell	Indicator Removal from Tools		System Owner/User Discovery	Windows Remote Management	Service Execution			Standard Cryptographic Protocol
Logon Scripts		Indicator Removal on Host		System Service Discovery		Third-party Software			Standard Non-Application Layer Protocol
Modify Existing Service		Install Root Certificate		System Time Discovery		Windows Management Instrumentation			Uncommonly Used Port
Netsh Helper DLL		InstallUtil				Windows Remote Management			Web Service
New Service		Legitimate Credentials							
Path Interception		MSBuild							
Redundant Access		Masquerading							
Registry Run Keys / Start Folder		Modify Registry							
Scheduled Task		NTFS Extended Attributes							
Security Support Provider		Network Share Connection Removal							
Service Registry Permissions Weakness		Obfuscated Files or Information							
Shortcut Modification		Process Hollowing							
Web Shell		Redundant Access							
Windows Management Instrumentation Event Subscription		Regsvcs/Regasm							
Winlogon Helper DLL		Regsvr32							
		Rootkit							
		Rundll32							
		Scripting							
		Software Packing							
		Timestomp							

TTPs: What are they?

- We will explain the meaning of TTPs with an example of a Car
- **Tactics** - The employment and ordered arrangement of forces in relation to each other
  - Preventative Maintenance
- **Techniques** - Non-prescriptive ways or methods used to perform missions, functions, or tasks
  - Changing Oil
- **Procedures** - Standard, detailed steps that prescribe HOW to perform specified tasks
  - Detailed manufacturer's instructions for oil change

Tactics & Techniques

- Tactics
  - Sorted by column headers

Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Execution	Collection	Exfiltration	Command and Control

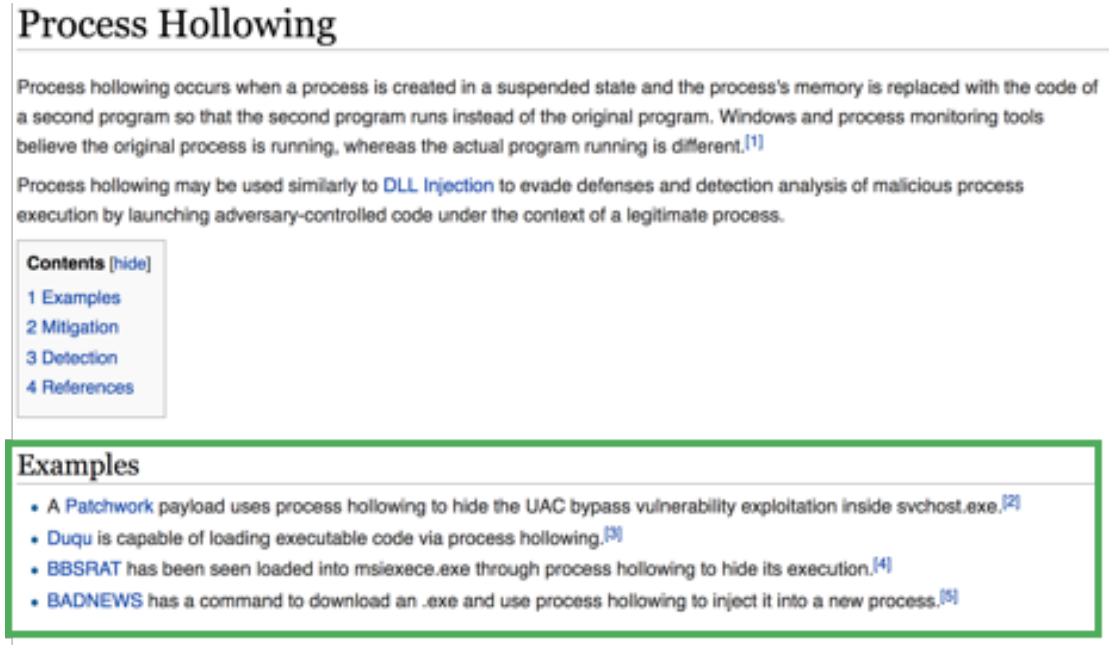
- Techniques
  - Represented by individual entries in ATT&CK matrix
  - 217 techniques currently documented
  - Windows, Linux, MacOS

Credential Access
Brute Force
Credential Dumping
Credential Manipulation
Credentials in Files
Exploitation of Vulnerability
Input Capture
Network Sniffing
Two-Factor Authentication Interception

https://attack.mitre.org/wiki/Main_Page

Procedures

- In the detailed information of each technique specific examples or threats are included as available
- Not all procedures are included, but the data set is large and growing



Process Hollowing

Process hollowing occurs when a process is created in a suspended state and the process's memory is replaced with the code of a second program so that the second program runs instead of the original program. Windows and process monitoring tools believe the original process is running, whereas the actual program running is different.[1]

Process hollowing may be used similarly to DLL Injection to evade defenses and detection analysis of malicious process execution by launching adversary-controlled code under the context of a legitimate process.

Contents [hide]

1 Examples
2 Mitigation
3 Detection
4 References

Examples

- A Patchwork payload uses process hollowing to hide the UAC bypass vulnerability exploitation inside svchost.exe.[2]
- Duqu is capable of loading executable code via process hollowing.[3]
- BBSRAT has been seen loaded into msiexece.exe through process hollowing to hide its execution.[4]
- BADNEWS has a command to download an .exe and use process hollowing to inject it into a new process.[5]

Why is this useful?

- Focus on detecting the behavior, not hashes and specific tool signatures
- Reference throughout the hypothesis process
- Tracking hunt history (technique coverage)
- Plan/chart future hunt activity
- Identify areas (Tactics) lacking coverage

Enter the Hunt Hypothesis

Our Hypothesis Process

- 5 step process to create meaningful hunt hypotheses
  1) Identify the Tactic and Technique
  2) Identify the Procedure(s)
  3) Identify the Collection Requirements
  4) Identify the Scope
  5) Document Excluded Factors

- Intended to be used to create hunt hypotheses to be completed in one week

Phase 1: Identify the Tactic & Technique

- High level what are you looking for?
- Used to track interest (tactics) over time in environment
- Attacks rarely use only one Tactic or Technique
  - Specifically focus your efforts

Phase 2: Identify the Procedures

- Specific examples and implementations of the selected technique
- Frequently found in APT reports, threat intelligence, etc.
- Understand and examine the different procedures
- What can and cannot be easily changed across all of the procedures?
- Perform research to understand the basic concepts of each procedure

Phase 3: Identify Collection Requirements

- Bulk of the research time
- Replicate malicious activity in the lab
- Identify common behaviors
- Identify high false positives
  - If possible, test in a small portion of the network
- Should result in a POC collection capability which gathers desired data

Phase 4: Identify the Scope

- Two factors for scope:
  - Time
    - Length of execution window
    - We recommend starting with week long execution windows
  - Number of data sources to collect
    - Can be host or network information
    - How much data can be collected in the timeframe?
    - How much data can be analyzed in the timeframe?
- Primarily based on collection requirements
- Scope may be limited due to limited collection capability

Phase 5: Document Excluded Factors

- What things were you unable to include in the hypothesis at each level?
  - What TTPs were not able to be researched during this hunt?
  - Technical collection limitations?
  - Political limitations?
  - Scope limitations?
- Will feed future hunt hypotheses
- Informs future technology purchases
- Quantifies the effects of scope limitation

What do we have at the end?

- Specific behavior being “hunted” for in the environment
- Understanding of the attack technique
- Knowledge of data required to detect the activity

Benefit of this process

- Focuses hunt efforts to have a tangible result
- Incremental improvement of security posture over time
- Can transition reported attack techniques to real security quickly

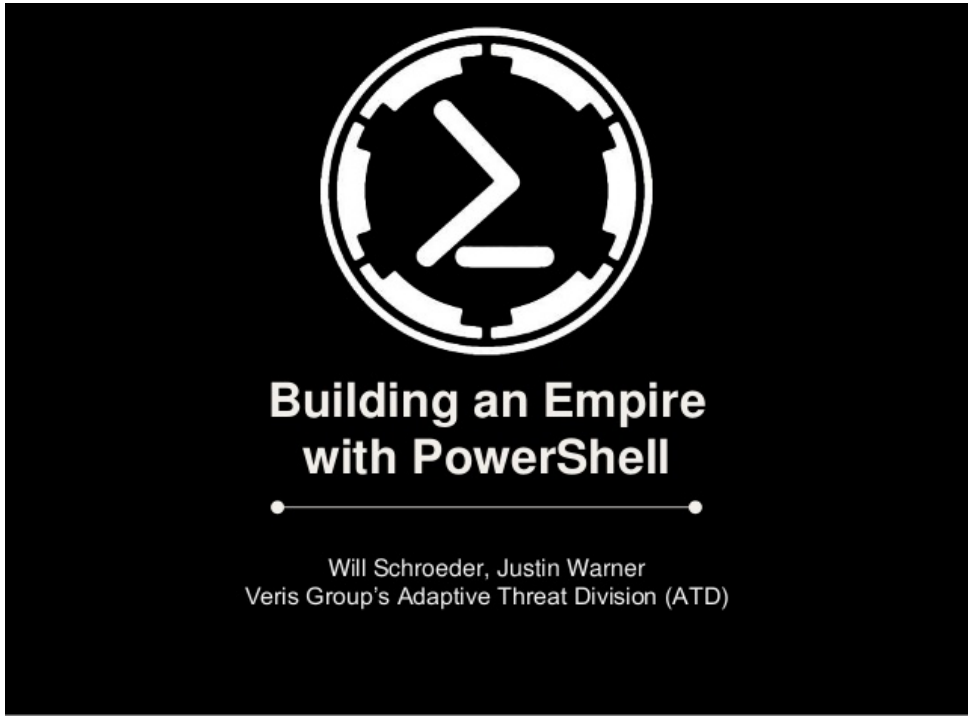

JIMMY
Kimmel
LIVE!

Case Study

Detecting Access Token Manipulation

CISO went to Hacker Summercamp

- Our situation:
  - Small security budget
  - No EDR capability (agent-based monitoring)
  - Poor lateral network visibility
  - Lots of local administrators
- Our task:
  - Can we detect it?
  - Can we stop it?
  - Have we been affected by it?
- Our reality:
  - Don’t want to focus on the Empire specifically, but maybe its capabilities



Phase 1: Tactic and Technique(s)

- Empire is commonly used for:
  - Privilege Escalation
    - Access Token Manipulation
    - Bypass User Account Control
  - Defense Evasion
    - DLL Injection
  - Credential Access
    - Credential Dumping
  - Lateral Movement
    - Pass the Hash
    - Pass the Ticket
    - Windows Management Instrumentation
- Tactic - Privilege Escalation [1]
- Technique - Access Token Manipulation [2]

[1]https://attack.mitre.org/wiki/Privilege_Escalation
[2]https://attack.mitre.org/wiki/Technique/T1134

Access Token Manipulation

- Windows uses **access tokens** to determine the security context of a running process or thread
- Attackers can manipulate their applied access token to access securable objects or perform privileged operations that they previously were not able to do.
- Procedures
  - Token Impersonation (Theft)
  - Create a Process with a Token
  - Create Token → Impersonate Token

https://attack.mitre.org/wiki/Technique/T1134

Windows Authentication Overview

- Windows creates a **logon session** upon successful authentication
  - User credentials (if any) are stored in lsass.exe
    - Credentials may be used later for Single Sign On
  - Credentials are tied to the logon session
  - Typical credentials: NTLM hash, Kerberos tickets, plaintext passwords
- **Tokens** define the security context of a process/thread
  - When a process/thread wants to act in a user context it uses a token
    - Interact with a securable object or perform an action that requires privilege
  - Tokens are tied to logon sessions and determine how the cred is used

Thread/Process → Token → Logon Session → Credential

Logon Session Types

- The **logon session type** is the only thing that matters for credential/token theft
  - NOT the token type. Credentials are tied to a logon session, NOT a token!
- Types
  - **Network** logon (type 3): the client proves they have credentials, but does not send them to the server (credentials are NOT in memory)
  - **Non-Network** logon (Interactive/NetworkCleartext/etc.): the client sends credentials to the service (credentials are in lsass.exe)
- Implication:
  - If Logon Session Type is 3 (Network logon) then there is no credential/token to steal

What is an Access Token?

- Kernel object that describes the security context of a process/thread
- Contains the following information:
  - User Account Security Identifier (SID)
  - Group SIDs
  - Logon SID (Logon Session Identifier)
  - List of Privileges held by user or groups
  - Token Integrity Level
  - Impersonation Level
  - Optional list of restricting SIDs

https://msdn.microsoft.com/en-us/library/windows/desktop/aa374909(v=vs.85).aspx

Token Types & Impersonation Levels

- **1) Primary** - a *process* token
  - OS uses token's credentials to authenticate remotely.
- **2) Impersonation** - a *thread* token
  - Threads use Impersonation tokens to impersonate other security contexts
  - OS ***might*** use token's credentials to authenticate remotely
- Token Impersonation Levels
  - **Anonymous** - Remote server cannot identify/impersonate client
  - **Identification** - Remote server can identify user, but not impersonate
  - **Impersonation** - The remote server can identify and impersonate the client across one computer boundary
  - **Delegation** - The server can impersonate the client on across multiple boundaries, and can make calls on behalf of the client.

Who has seen or done this??

```
root@kali: ~
ile Edit View Search Terminal Help

 Id  Type                  Information                          Connection
 --  ----                  -----------                          ----------
 1   meterpreter x86/win32  NOSECURITY\paranoid @ PARANOIDPC  192.168.56.85:443
-> 192.168.56.83:62675 (192.168.56.83)

sf exploit(ask) > run

*] Started reverse handler on 192.168.56.85:4444
*] UAC is Enabled, checking level...
*] The user will be prompted, wait for them to click 'Ok'
*] Uploading EwINpSJetr.exe - 73802 bytes to the filesystem...
*] Executing Command!
*] Sending stage (885806 bytes) to 192.168.56.83
*] Meterpreter session 3 opened (192.168.56.85:4444 -> 192.168.56.83:62745) at
015-08-24 15:31:13 +0300

eterpreter > getuid
erver username: NOSECURITY\paranoid
eterpreter > getsystem
..got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
eterpreter > getuid
erver username: NT AUTHORITY\SYSTEM
eterpreter >
```

Image Credit: https://dfirblog.wordpress.com/2015/10/24/protecting-windows-networks-uac/

Access Token Manipulation Overview

- Every process has a **primary token** that describes the security context of the user account associated with the process
- By default, threads use the process' primary token
- Threads can impersonate a client account
  - The thread will have both a **primary** and **impersonation token** in this case
- Token Impersonation
- Create a Process with a Token
- Make and Impersonate Token

Phase 2: Identify the Procedures

- Technique - Access Token Manipulation
  - Token Impersonation
  - Create a Process with a Token
  - Make and Impersonate Token

Token Impersonation (Theft)

- Situation:
  - Your target user has a non-network Logon Session on the system.
- Assuming admin rights, you can directly impersonate the token.
- DuplicateToken(Ex)
  - Creates a new access token that duplicates an existing token.[1]
  - Can use returned token w/ ImpersonateLoggedOnUser or SetThreadToken
- ImpersonateLoggedOnUser
  - Lets the calling thread impersonate a logged on user's security context.[2]
  - Works with **primary** and **impersonation** tokens.[2]
- SetThreadToken
  - Assigns an **impersonation token** to a thread.[3]

[1]https://msdn.microsoft.com/en-us/library/windows/desktop/aa446617(v=vs.85).aspx
[2]https://msdn.microsoft.com/en-us/library/windows/desktop/aa378612(v=vs.85).aspx
[3]https://msdn.microsoft.com/en-us/library/windows/desktop/aa379590(v=vs.85).aspx

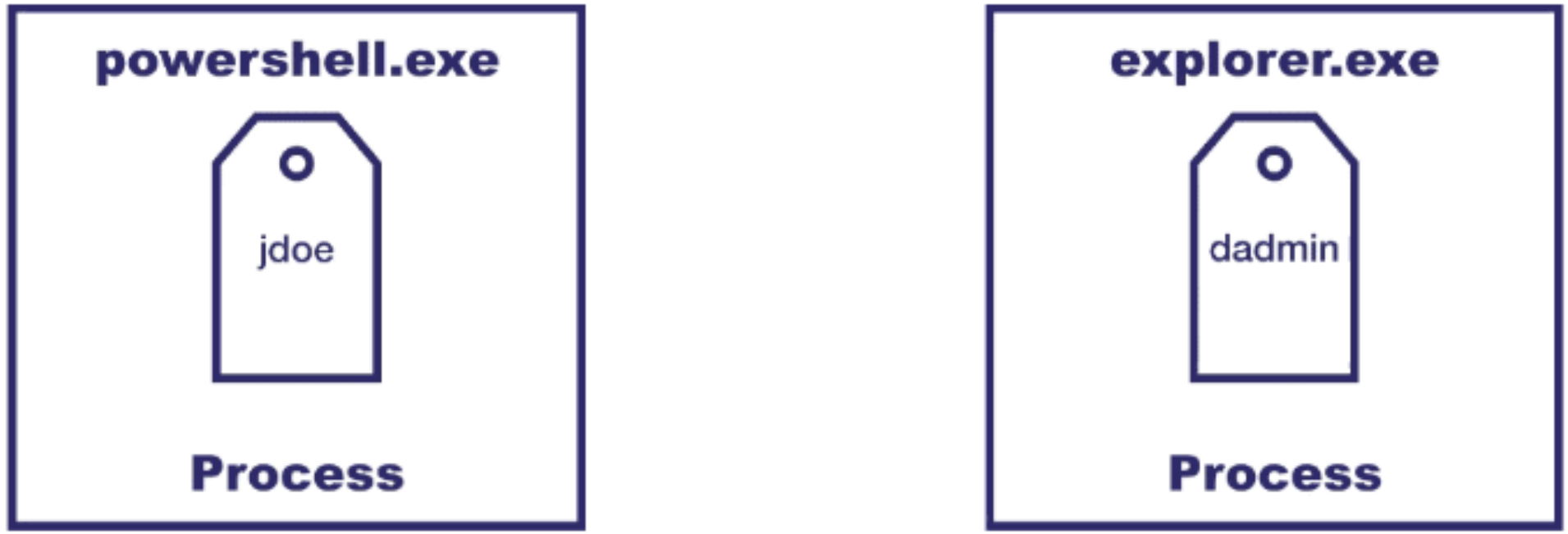
TOKEN IMPERSONATION/THEFT

Create a Process with a Token

- Situation:
  - You want a quick way to create a process with a security context for a different user account.
- DuplicateToken(Ex)
  - Creates a new access token that duplicates an existing token.[1]
  - Can use returned token w/ ImpersonateLoggedOnUser or SetThreadToken
- CreateProcessWithTokenW
  - Creates a new process and its primary thread.[1]
  - Process runs in the security context of the user account represented by the specified token.[1]

[1]https://msdn.microsoft.com/en-us/library/windows/desktop/ms682434(v=vs.85).aspx

CREATE PROCESS WITH TOKEN

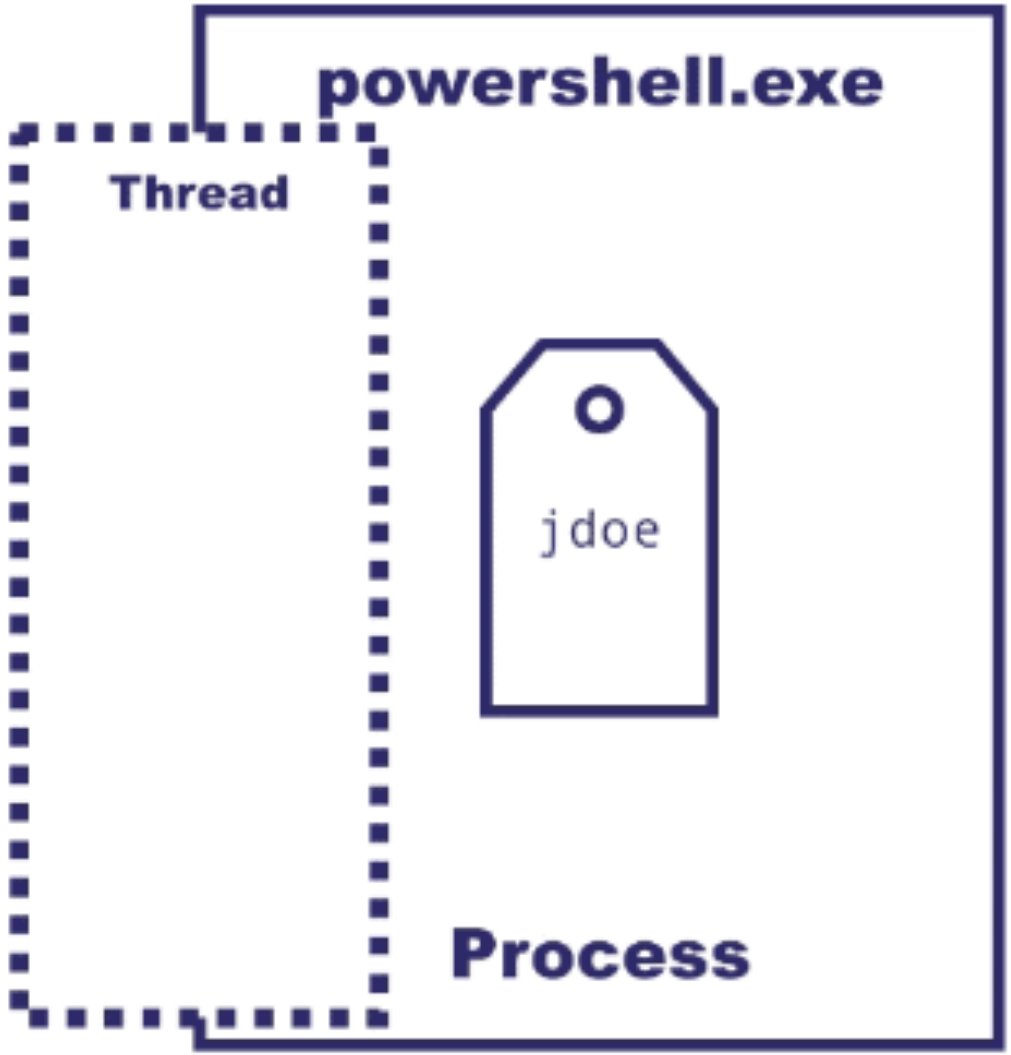


Make and Impersonate Token[1]

- Situation:
  - You have a username and password, but the user is not logged on
- LogonUser
  - Set the **dwLogonType** to LOGON32_LOGON_NEW_CREDENTIALS (type 9)
  - Creates a NewCredential Logon Session for specified user and password
    - Local authentication will use the parent process' user
    - Network authentication will use the specified user account
  - Returns a copy of the new Logon Session's access token
- SetThreadToken
  - Assigns an impersonation token to a thread.

[1]https://blog.cobaltstrike.com/2015/12/16/windows-access-tokens-and-alternate-credentials/

MAKE AND IMPERSONATE TOKEN



Phase 2: Identify the Procedures

- Technique - Access Token Manipulation
  - Token Impersonation
  - Create a Process with a Token
  - Make and Impersonate Token
- **Procedure**
  - Token Impersonation
  - Create and Impersonate Token

Phase 3: Collection Requirements

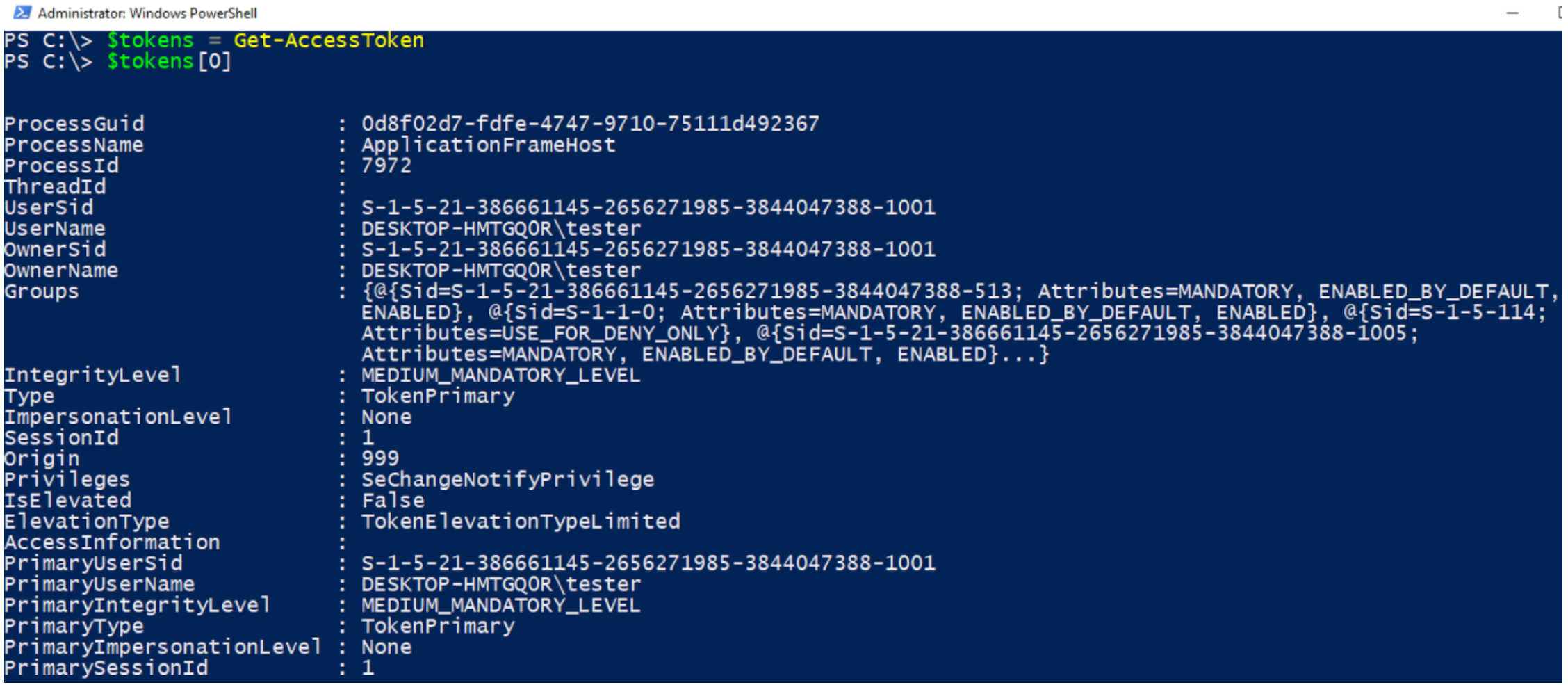
- Interact with known Access Token Manipulation tools to identify collection requirements
  - Incognito (Meterpreter)
  - Invoke-TokenManipulation (PowerSploit/PowerShell Empire)
  - Cobalt Strike
- Collect relevant data points
  - Access Tokens for each process and thread

Collecting - Access Tokens

- Enumerate processes/threads (*Get-Process*)
- **OpenProcess**[1] - Returns a handle to a process object
- **OpenProcessToken**[2] - Opens an access token associated with a process
- **OpenThread**[3] - Returns a handle to a Thread object
- **OpenThreadToken**[4] - Opens an access token associated with a thread
- **GetTokenInformation**[5] - Retrieves a specified type of information about an access token

[1]https://msdn.microsoft.com/en-us/library/windows/desktop/ms684320(v=vs.85).aspx
[2]https://msdn.microsoft.com/en-us/library/windows/desktop/aa379295(v=vs.85).aspx
[3]https://msdn.microsoft.com/en-us/library/windows/desktop/ms684335(v=vs.85).aspx
[4]https://msdn.microsoft.com/en-us/library/windows/desktop/aa379296(v=vs.85).aspx
[5]https://msdn.microsoft.com/en-us/library/windows/desktop/aa446671(v=vs.85).aspx

Get-AccessToken[1]



https://gist.github.com/jaredcatkinson/17698b39efd72f976a6a846ec3a8eacd

Collecting - Ticket Granting Tickets

- Enumerate LSA Logon Sessions
  - **LsaEnumerateLogonSessions**[1] - Returns a handle to an array of session data structures.
  - **LsaGetLogonSessionData**[2] - Queries each session handle for its associated information (logon type, user, etc.).
- Request each Logon Session's Ticket Granting Ticket
  - **LsaRegisterLogonProcess**[3] - Establishes a connection to the Local Security Authority Server.
  - **LsaCallAuthenticationPackage**[4] - Calls a specified function implemented by an authentication package (Kerberos).
  - **LsaDeregisterLogonProcess**[5] - Closes the connection to the Local Security Authority Server.

[1]https://msdn.microsoft.com/en-us/library/windows/desktop/aa378275(v=vs.85).aspx
[2]https://msdn.microsoft.com/en-us/library/windows/desktop/aa378290(v=vs.85).aspx
[3]https://msdn.microsoft.com/en-us/library/windows/desktop/aa378318(v=vs.85).aspx
[4]https://msdn.microsoft.com/en-us/library/windows/desktop/aa378261(v=vs.85).aspx
[5]https://msdn.microsoft.com/en-us/library/windows/desktop/aa378269(v=vs.85).aspx

Get-KerberosTicketGrantingTicket[1]

```
PS C:\> $tickets[0]


ServiceName                   : krbtgt
ClientName                    : localadmin
DomainName                    : CITADEL.COVERTIUS.LOCAL
TargetDomainName              : CITADEL.COVERTIUS.LOCAL
AltTargetDomainName           : CITADEL.COVERTIUS.LOCAL
SessionKeyType                : aes256_cts_hmac_sha1_96
SessionKey                    : {28, 236, 204, 223...}
TicketFlags                   : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyExpirationTime             : 1/1/1601 1:00:00 AM
StartTime                     : 11/28/2017 3:03:23 AM
EndTime                       : 11/28/2017 1:03:23 PM
RenewUntil                    : 12/4/2017 5:18:22 PM
TimeSkew                      : 0
EncodedTicketSize             : 1150
EncodedTicket                 : {97, 130, 4, 122...}
SessionLogonId                : 431425084
SessionUserName               : localadmin
SessionUserPrincipalName      : localadmin@citadel.covertius.local
SessionLogonType              : RemoteInteractive
SessionAuthenticationPackage  : Kerberos
SessionLogonServer            : zeus
SessionLogonDomain            :
PSComputerName                : 10.1.10.101
RunspaceId                    : f14b26e7-6e19-4925-b963-12077ea9907d
```

https://gist.github.com/jaredcatkinson/c95fd1e4e76a4b9b966861f64782f5a9#file-get-kerberosticketgrantingticket-ps1

Phase 4: Identify the Scope

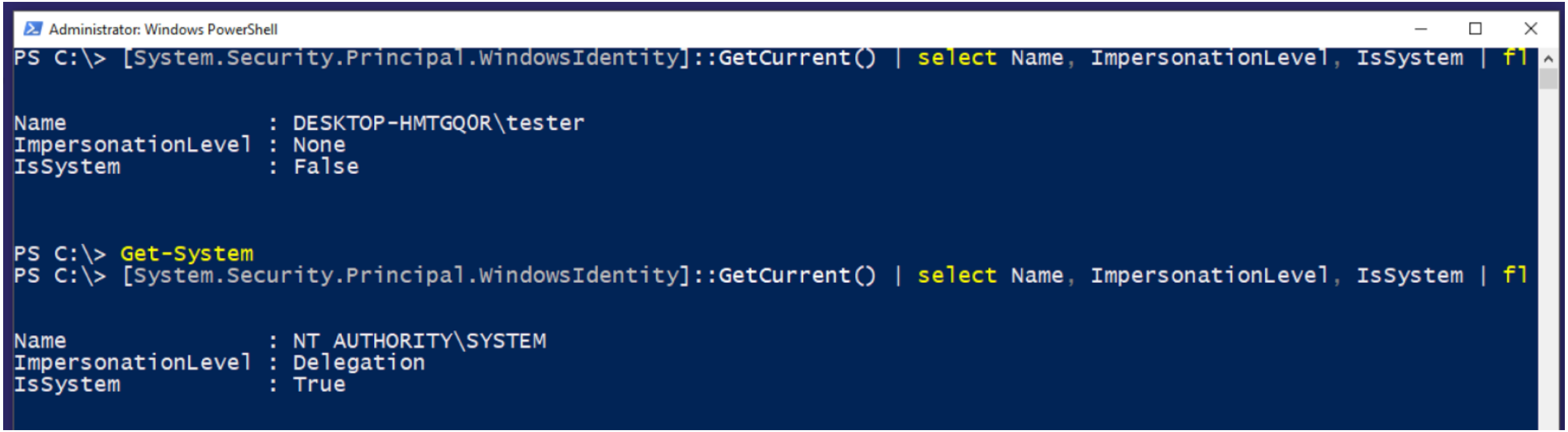
- Our timeframe:
  - One week
- Our environment:
  - 3 Domains
  - 1 Linux Server (non-domain joined)
  - 9 Windows Workstations
  - 2 Windows Servers
  - No sensitive production systems
- Our scope:
  - 3 domains
  - Windows Servers and Workstations (11)

Phase 5: Document Excluded Factors

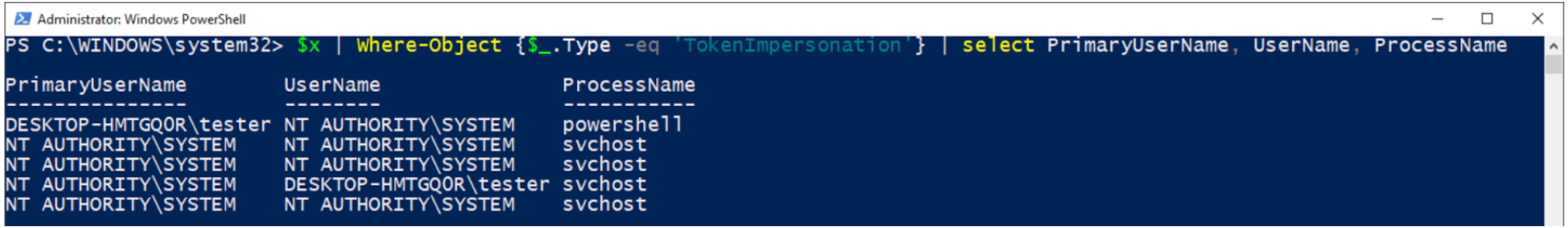
- Privilege Escalation
  - Bypass User Account Control
  - New Service
- Credential Access
  - Credential Dumping
  - Account Manipulation
  - Security Support Providers
- Lateral Movement
  - Pass the Hash
  - Pass the Ticket
- Scope
  - Lacked credentials for the Linux Server

Hunt Execution Demo!!

Adversary Runs Get-System



Query Impersonation Tokens




```
ProcessName       : powershell
PrimaryUserName   : DESKTOP-HMTGQOR\tester
UserName          : DESKTOP-HMTGQOR\tester
Type              : TokenPrimary
IntegrityLevel    : HIGH_MANDATORY_LEVEL

ProcessName       : powershell
PrimaryUserName   : DESKTOP-HMTGQOR\tester
UserName          : NT AUTHORITY\SYSTEM
Type              : TokenImpersonation
IntegrityLevel    : SYSTEM_MANDATORY_LEVEL
```

Stolen Token in Detail

```
ProcessGuid               : d7da1078-b93a-4ef4-99e2-beea5c497b9f
ProcessName               : powershell
ProcessId                 : 3628
ThreadId                  : 11104
UserSid                   : S-1-5-18
UserName                  : NT AUTHORITY\SYSTEM
OwnerSid                  : S-1-5-32-544
OwnerName                 : BUILTIN\Administrators
Groups                    : {@{Sid=S-1-5-32-544; Attributes=ENABLED_BY_DEFAULT, ENABLED, OWNER}, @{Sid=S-1-1-0;
                            Attributes=MANDATORY, ENABLED_BY_DEFAULT, ENABLED}, @{Sid=S-1-5-11; Attributes=MANDATORY,
                            ENABLED_BY_DEFAULT, ENABLED}, @{Sid=S-1-16-16384; Attributes=INTEGRITY, INTEGRITY_ENABLED}}
IntegrityLevel            : SYSTEM_MANDATORY_LEVEL
Type                      : TokenImpersonation
ImpersonationLevel        : SecurityDelegation
SessionId                 : 1
Origin                    : 0
Privileges                : SeTcbPrivilege;SeProfileSingleProcessPrivilege;SeIncreaseBasePriorityPrivilege;SeCreatePermanentPr
                            ivilege;SeDebugPrivilege;SeAuditPrivilege;SeChangeNotifyPrivilege;SeImpersonatePrivilege;SeCreateG
                            lobalPrivilege
IsElevated                : True
ElevationType             : TokenElevationTypeDefault
AccessInformation         : 2106006828064
PrimaryUserSid            : S-1-5-21-386661145-2656271985-3844047388-1001
PrimaryUserName           : DESKTOP-HMTGQOR\tester
PrimaryIntegrityLevel     : HIGH_MANDATORY_LEVEL
PrimaryType               : TokenPrimary
PrimaryImpersonationLevel : None
PrimarySessionId          : 1
```

Establish PSRemoting Session to Targets

Administrator: Windows PowerShell

```
PS C:\> $1

 Id Name            ComputerName    ComputerType    State         ConfigurationName     Availability
 -- ----            ------------    ------------    -----         -----------------     ------------
  1 WinRM1          10.1.20.236     RemoteMachine   Opened        Microsoft.PowerShell     Available
  2 WinRM2          10.1.20.123     RemoteMachine   Opened        Microsoft.PowerShell     Available
  3 WinRM3          10.1.20.105     RemoteMachine   Opened        Microsoft.PowerShell     Available
  4 WinRM4          10.1.10.101     RemoteMachine   Opened        Microsoft.PowerShell     Available
  5 WinRM5          10.1.10.152     RemoteMachine   Opened        Microsoft.PowerShell     Available
  6 WinRM6          10.1.10.173     RemoteMachine   Opened        Microsoft.PowerShell     Available
  7 WinRM7          10.1.10.107     RemoteMachine   Opened        Microsoft.PowerShell     Available
  8 WinRM8          10.1.10.51      RemoteMachine   Opened        Microsoft.PowerShell     Available
  9 WinRM9          10.1.10.25      RemoteMachine   Opened        Microsoft.PowerShell     Available
 10 WinRM10         10.1.30.100     RemoteMachine   Opened        Microsoft.PowerShell     Available
 11 WinRM11         10.1.30.152     RemoteMachine   Opened        Microsoft.PowerShell     Available
```

Query Kerberos TGTs in Each Session

Administrator: Windows PowerShell

```
PS C:\> $tickets = Invoke-Command -Session $1 -FilePath C:\demo\BlackHatEU\Get-KerberosTicketGrantingTicket.ps1
Unable to Impersonate System Token
    + CategoryInfo          : OperationStopped: (Unable to Impersonate System Token:String) [], RuntimeException
    + FullyQualifiedErrorId : Unable to Impersonate System Token
    + PSComputerName        : 10.1.30.152

PS C:\> $tickets | Where-Object {$_.SessionLogonType -eq 'NewCredentials'} | select SessionLogonId, SessionUserName, ClientName,
 PSComputerName

SessionLogonId SessionUserName ClientName PSComputerName
-------------- --------------- ---------- --------------
      31048020 SYSTEM          mnelsonDA  10.1.20.123
      44932836 arobbins_da     mnelsonDA  10.1.20.105
      44928598 arobbins_da     mnelsonDA  10.1.20.105
      45133054 arobbins_da     mnelsonDA  10.1.20.105
      24508547 SYSTEM          mnelsonDA  10.1.10.107
      15745484 SYSTEM          wschroeder 10.1.10.107
      30247392 SYSTEM          bharris_a  10.1.10.152
     103599649 arobbins_da     arobbins   10.1.10.25
     107121830 arobbins_da     arobbins   10.1.10.25
     252487656 john            mwright_a  10.1.10.101
     252478810 john            arobbins   10.1.10.101
     252421172 john            arobbins   10.1.10.101
     252365680 john            arobbins   10.1.10.101
     252234507 john            mwright_a  10.1.10.101
     252220963 john            mwright_a  10.1.10.101
     251281527 john            amurphy    10.1.10.101
     251058902 john            mwright_a  10.1.10.101
     251007545 john            sqlsvc     10.1.10.101
      94005877 bharris_a       mwright_a  10.1.10.101
      92196334 john            bharris_a  10.1.10.101
      91987531 john            bharris_a  10.1.10.101
      90082821 john            mwright_a  10.1.10.101
```

Viewing Malicious Session in Detail

```
PS C:\> $tickets | Where-Object {$_.SessionLogonId -eq 251007545}


ServiceName                   : krbtgt
ClientName                    : sqlsvc
DomainName                    : CITADEL.COVERTIUS.LOCAL
TargetDomainName              : CITADEL.COVERTIUS.LOCAL
AltTargetDomainName           : CITADEL.COVERTIUS.LOCAL
SessionKeyType                : rc4_hmac
SessionKey                    : {149, 14, 59, 61...}
TicketFlags                   : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyExpirationTime             : 1/1/1601 1:00:00 AM
StartTime                     : 11/16/2017 6:31:48 PM
EndTime                       : 11/16/2017 8:46:44 PM
RenewUntil                    : 11/16/2017 8:46:44 PM
TimeSkew                      : 0
EncodedTicketSize             : 1098
EncodedTicket                 : {97, 130, 4, 70...}
SessionLogonId                : 251007545
SessionUserName               : john
SessionUserPrincipalName      :
SessionLogonType              : NewCredentials
SessionAuthenticationPackage  : Negotiate
SessionLogonServer            :
SessionLogonDomain            :
PSComputerName                : 10.1.10.101
RunspaceId                    : f14b26e7-6e19-4925-b963-12077ea9907d
```