CYBERARK®

LABS RESEARCH

PASS-THE-HASH DETECTION WITH WINDOWS EVENT VIEWER

Abstract

In this paper we will focus on detecting Pass-The-Hash attacks, after the credentials were stolen, via the event viewer. Pass-The-Hash is an attack technique that allows an attacker to start lateral movement in the network over NTLM protocol, in contrary to Over Pass-The-Hash which is based on Kerberos protocol, without the need for the user password. We will compare between legitimate and illegitimate NTLM connections, we will show what indictors can be used to distinguish between them and what we can conclude from that to build out an algorithm to demonstrate detection of Pass-the-Hash attacks. CyberArk Labs created a tool (Ketshash) that demonstrate the detection methods that we will talk about in this paper. This paper does not provide a 100% solution for Pass-The-Hash attack but it will show what can be done with the available tools and how to create a general view of the NTLM connections over the network.

Table of Contents

Figures 3
Tables 4
Finding an anchor 5
NTLM connections overview 6
Environment 6
Legitimate NTLM connections examples 7
RDP via NTLM connection with domain admin account 7
Mount administrative share via NTLM with domain admin account 9
Illegitimate NTLM connections examples 11
Pth-Winexe from Kali to Windows 10 11
Mounting administrative network share with Mimikatz from Windows 10 to Windows 7 12
Mounting administrative share from Windows 7 to Windows 10 with WCE 16
Identifying legitimate NTLM connections 18
Privileged NTLM connections 19
Putting the pieces together 19
A proposed alogrithm 19
Flow diagram 21
Advantages 22
Disadvantages 22
Things to keep in mind 22
Timing is everything 22
Sabotaging the event viewer 23
Summary 23
References 23

Figures

Figure 1 Event Properties: Event 4624 5
Figure 2 RDP via NTLM Simulation diagram 8
Figure 3 Mount Share Simulation diagram 9
Figure 4 PTH-Winexe Simulation Diagrahm 11
Figure 5 Passing the hash with Pth-winexe 11
Figure 6 Event ID 4672 indicates a privileged session initiation on the target machine 11
Figure 7 Mimikatz Simulation diagram 12
Figure 8 Passing the hash with Mimikatz 12
Figure 9 Compare between Event ID 4624: Windows 7 Event viewer- vs - Windows 10 Event viewer 13
Figure 10 Event ID 303 indicates a credentials use 13
Figure 11 Mimikatz use of CreateProcessWithLogonW 14
Figure 12 Ruas /netonly use of CreateProcessWithLogonW 14
Figure 13 Event ID 4656 indicates writing to LSASS memory 15
Figure 14 Mounting Administrative Share Simulation Diagram 16
Figure 15 Passing the hash with Windows Credentials Editor 16
Figure 16 WCE service creation log 17
Figure 17 WCE with two instances on Process Explorer 17
Figure 18 Logon session transferred through the pipeline 17
Figure 19 LSASS.exe loads wceaux.dll 17
Figure 20 Events ID 4768 and 4769 18
Figure 21 Correlating Events 4624 and 4672 by TargetLogonId and SubjectLogonId 19
Figure 22 Example for diagram based on section 4 22

Tables

Table 1 Microsoft Logon Types....7
Table 2 Event 4648 on the source machine (MARS-10)....8
Table 3 Event ID 4776 on the domain controller....8
Table 4 Event ID 4672 indicates a privileged session initiation on the target machine (MARS-7)....9
Table 5 Events ID 4648 and 4624 on the source machine (MARS-10)....10
Table 6 Events ID 4768 and 4769 indicate a Kerberos authentication on the domain controller....10
Table 7 Event ID 4624 with logon type 9 ("NewCredentials")....14
Table 8 Event ID 4776 on the domain controller....15
Table 9 Event ID 4672 indicates a privileged session initiation on the target machine (MARS-7)....15
Table 10 Event ID 4776 on the domain controller....18
Table 11 Event ID 4672 indicates a privileged session initiation on the target machine (MARS-10)....18

Finding an anchor

Once an attacker steals user's credentials, he can use it to open new session on the current computer or on a remote computer (if the stolen account has permissions on this computer). When he starts lateral movement over the network one of two protocols will be in use: NTLM (Pass-The-Hash) or Kerberos (Over Pass-The-Hash). Our focus is on Pass-the-Hash, therefore we will cover only connections over NTLM protocol. Pass-The-Hash is a bit more difficult to detect comparing to Over Pass-The-Hash as their Indicators of Compromises (IOCs) are more subtle. Information related to Over Pass-The-Hash can be found here (by Benjamin Delpy) and here (by Nikhil Mittal).

One of the triggers to use NTLM protocol is when a client authenticates to a server using IP address. This is the "anchor" that can be used to detect Pass-The-Hash attack over the network – NTLM connections. NTLM connections can be identified by monitoring events ID 4624 ("*An account was successfully logged on*") with logon type 3 ("*A user or computer logged on to this computer from the network*") and NTLM authentication package NTLM (or by logon process name NtLmSsp) which indicates the use of NTLM protocol for the authentication. See Figure 1 below for example.

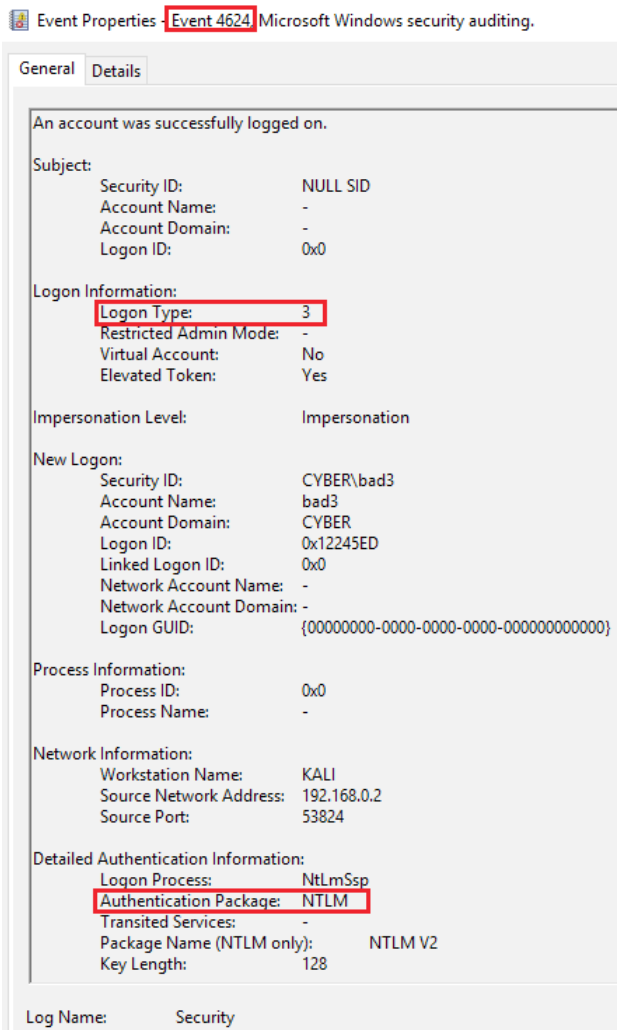

Figure 1. Event Properties: Event 4624

NTLM connections could also be detected by monitoring network traffic. Although NTLM authentication is an old authentication method and most of the authentications in the organizations today are based on Kerberos, some applications are still using it and according to Microsoft:

> **NTLM authentication is still supported and must be used for Windows authentication with systems configured as a member of a workgroup. NTLM authentication is also used for local logon authentication on non-domain controllers. Kerberos version 5 authentication is the preferred authentication method for Active Directory environments, but a non-Microsoft or Microsoft application might still use NTLM.**

NTLM connections overview

As NTLM authentication is used legitimately across the network, it is crucial to look at both legitimate and illegitimate NTLM authentications, which will also cover Pass-The-Hash scenarios. The next section covers several legitimate and illegitimate NTLM connections based on Pass-The-Hash techniques and the artifacts that can lead to detection.

We will show examples for legitimate and illegitimate NTLM connections:

- Legitimate:
  - RDP via NTLM connection with domain admin account
  - Mount administrative share via NTLM with domain admin account
- Illegitimate:
  - Pth-Winexe from Kali to Windows 10
  - Mounting administrative network share with Mimikatz from Windows 10 to Windows 7
  - Mounting administrative network share from Windows 7 to Windows 10 with WCE

Environment

We are using four machines:

- CYBER-DC (Server 2012R2) – 192.168.0.1
- KALI (Kali) – 192.168.0.2
- MARS-10 (Windows 10) – 192.168.0.3
- MARS-7 (Windows 7) – 192.168.0.4

UAC is in the top level.

Users:

- Student1 (standard user)
- Local_admin (local administrator)
- Legit1, legit2, bad1, bad2 and bad3 (domain admin users)

Tools:

- Mimikatz
- Pth-winexe – part of Kali's built-in tools
- Windows Credentials Editor (WCE)
- Invoke-Smbclient

Events map:

- 4768 - A Kerberos authentication ticket (TGT) was requested
- 4769 - A Kerberos service ticket was requested
- 4648 - A logon was attempted using explicit credentials
- 4672 - Special privileges assigned to new logon
- 4624 - An account was successfully logged on
- *4776 - The computer attempted to validate the credentials for an account. It is generated when the domain controller successfully authenticates a user via NTLM

* In all the examples where event 4776 was displayed, the authentication package was: MICROSOFT_AUTHENTICATION_PACKAGE_V1_0

Logon type	Logon title	Description
2	Interactive	A user logged on to this computer.
3	Network	A user or computer logged on to this computer from the network.
4	Batch	Batch logon type is used by batch servers, where processes may be executing on behalf of a user without their direct intervention.
5	Service	A service was started by the Service Control Manager.
7	Unlock	This workstation was unlocked.
8	NetworkCleartext	A user logged on to this computer from the network. The user's password was passed to the authentication package in its unhashed form. The built-in authentication packages all hash credentials before sending them across the network. The credentials do not traverse the network in plaintext (also called cleartext).
9	NewCredentials	A caller cloned its current token and specified new credentials for outbound connections. The new logon session has the same local identity, but uses different credentials for other network connections.
10	RemoteInteractive	A user logged on to this computer remotely using Terminal Services or Remote Desktop.
11	CachedInteractive	A user logged on to this computer with network credentials that were stored locally on the computer. The domain controller was not contacted to verify the credentials.

Table 1. Microsoft Logon Types

Taken from Microsoft documentation

Legitimate NTLM connections examples

RDP via NTLM connection with domain admin account

IT administrators, security operation center analysts and other privileged users, commonly use their non-privileged account to access personal information like emails. However, when executing sensitive tasks via RDP, for example, those users will commonly escalate their privileges using their privileged account credentials. When the user inserts the user name and password together with the target machine IP address, a legitimate NTLM authentication will be triggered.

To simulate such scenario, we used the non-privileged account *student1* and the privileged account *legit1* in the following way:

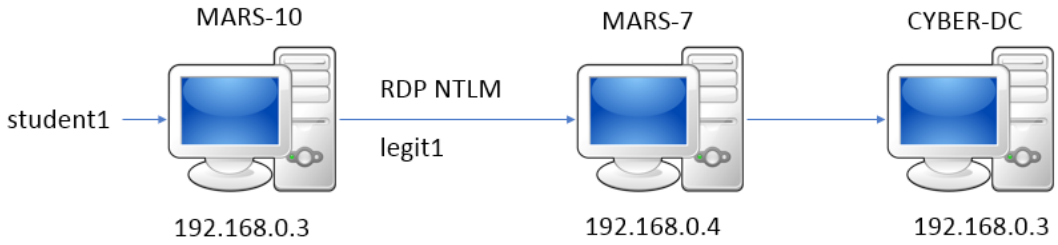

Figure 2. RDP via NTLM Simulation diagram

Our user, *student1* is logged on to *MARS-10* and connects via RDP to *MARS-7* by entering *MARS-7*'s IP address with the domain admin account legit1. When our user, student1 inserts legit1 credentials to escalate privileges and execute the task, event ID 4648 will be generated on the source machine (*MARS-10*) indicating explicit usage of credentials. The event indicates that credentials were inserted interactively (it doesn't have to be at the same time, e.g. password inserted to privileged scheduled task) by the user, which in other words mean that the current user has knowledge of the clear text password.

Source machine (MARS-10, 192.168.0.3)						
Code	**Subject User Sid (resolved)**	**Target Domain Name \Target User Name**	**Target machine**	**Process name**	**Subject Logon ID**	**Time seq.**
4648	CYBER\student1	CYBER\legit1	MARS-7.cyber.com	C:\Windows\System32\ LSASS.exe	0x2a9a2	#1

Table 2. Event 4648 on the source machine (MARS-10)

As the new credentials used by *student1* are domain credentials, the domain controller receives the authentication request, originating from target machine (*MARS-7*), which initiates the authentication process in the NTLM authentication scheme. The NTLM authentication attempt is registered as event ID 4776 on the domain controller.

Domain controller (CYBER-DC, 192.168.0.1)			
Code	**Target Domain Name \Target User Name**	**Workstation**	**Time seq.**
4776	legit1	MARS-10	#4

Table 3. Event ID 4776 on the domain controller

As soon as the privileged account, *legit1*, is granted **privileged** access to the target machine (*MARS-7*), event ID 4672 is generated, indicating an initiation of a privileged session on the target machine.

Target machine (MARS-7, 192.168.0.4)								
Code	**Subject User Sid (resolved)**	**Target Domain Name \Target User Name**	**Workstation Name**	**Logon Type**	**Authentication Package Name**	**Subject Logon ID**	**Target Logon ID**	**Time seq.**
4672	CYBER\legit1					**0xfcb23a**		#2
4624	S-1-0-0	CYBER\legit1	MARS-10	3	NTLM	0x0	**0xfcb23a**	#3

Table 4. Event ID 4672 indicates a privileged session initiation on the target machine (MARS-7)

In this scenario, of NTLM based RDP connection, the 4648 event is the most significant indication of a legitimate connection and authentication. In case of credentials theft and utilization of Pass-The-Hash, the malicious actor might have access to the password hash only. Therefore, when the credentials will be used or injected into a new or existing session, event 4648 might be absent and so exposing the potentially malicious activity.

Mount administrative share via NTLM with domain admin account

Similar to the previous legitimate scenario, sometimes the only sensitive tasks that these users will need is to check files on the server and it is more convenient to access these shares via SMB by mounting the administrative share. In other cases, the simultaneous remote connections can reach to its limit of remote connections and the only way to log on via RDP is by sending request for one of the users to log off which is unnecessary when the only needed task was to check files.

To simulate this scenario, we used the non-privileged account *student1* and the privileged account *legit1* in the following way:

student1 is logged on to *MARS-10*, he is using the *Run as different user* option to open an elevated command line interface (CLI) by providing a privileged account *legit2*'s credentials and mounts an administrative network share.

* Notice to the difference between right clicking on a file and choosing *Run as different user* to *Run as administrator*.

Run as administrator might validate the password, in some scenarios, with the cached passwords on the machine and therefore there will be no related events on the domain controller. In such cases all 4624 events will be assigned logon type 11.

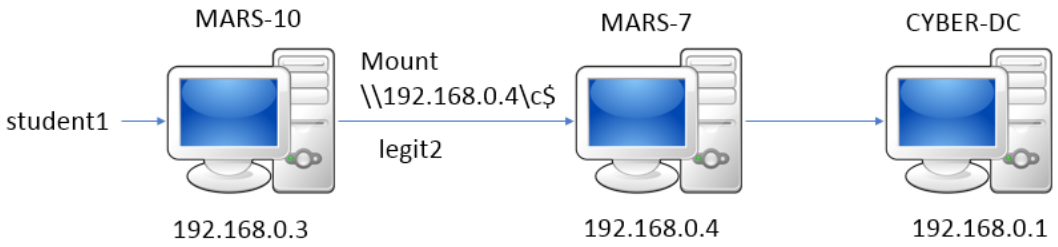

Figure 3. Mount Share Simulation diagram

When our *user*, *student1* inserts *legit1* credentials to escalate privileges and open the CLI, events ID 4648 and 4624 with logon type 2 (*Interactive*) will be generated on the source machine (*MARS-10*). Event ID 4648 was already mentioned in the previous example but the new event ID 4624 with logon type 2 (*Interactive*) is also an indication that credentials were inserted interactively by the user.

Source machine (MARS-10, 192.168.0.3)									
Code	**Subject User Sid (resolved)**	**Target Domain Name \Target User Name**	**Workstation Name**	**Logon Type**	**Authentication Package Name**	**Process Name\ Target machine**	**Subject Logon ID**	**Target Logon ID**	**Time seq.**
4648	CYBER\ student1	CYBER\legit2				C:\ Windows\ System32\ consent.exe localhost	0x2a9a2		#1
4624	S-1-0-0	CYBER\legit2	MARS-10	2	Negotiate		0x2a9a2	0x86747e	#2
4672	CYBER\ legit2						0x86747e		#3

Table 5. Events ID 4648 and 4624 on the source machine (MARS-10)

The domain controller receives this authentication request by the user *legit2* from the target machine (*MARS-7*) but unlike the previous example, this time it initiates a Kerberos authentication. The Kerberos authentication is registered as evens ID 4768 and 4769 on the domain controller.

Domain controller (CYBER-DC, 192.168.0.1)					
Code	**Subject User Sid (resolved)**	**Target Domain Name \Target User Name**	**Workstation**	**IP address**	**Time seq.**
4768		CYBER\legit2		192.168.0.3	#4
4769		CYBER\legit2		192.168.0.3	#5
4776	legit2		MARS-10		#9

Table 6. Events ID 4768 and 4769 indicate a Kerberos authentication on the domain controller

Once the privileged account, *legit2*, is mounting the administrative share, event ID 4672 is generated, indicating an initiation of a privileged session on the target machine (*MARS-7*).

Target machine (MARS-7, 192.168.0.4)								
Code	**Subject User Sid (resolved)**	**Target Domain Name \Target User Name**	**Workstation Name**	**Logon Type**	**Authentication Package Name**	**Subject Logon ID**	**Target Logon ID**	**Time seq.**
After mounting the network share								
4672	CYBER\legit2					**0x1149ef9**		#7
4624	S-1-0-0	CYBER\legit2	MARS-10	3	NTLM	0x0	**0x1149ef9**	#8

Table 2-3. Event ID 4672 indicates a privileged session initiation on the target machine (MARS-7)

In this scenario, of mounting administrative share, a new events ID 4624 with logon type 2, event 4768 and 4769 are another strong indication of a legitimate connection and authentication. We also had event ID 4648 but in some scenarios, like unlocking the workstation, we will have only event ID 4624 on the source machine (*MARS-10*) and events 4768 & 4769 on the domain controller.

Illegitimate NTLM connections examples

Pth-Winexe from Kali to Windows 10

Kali is a Linux distribution used for penetration testing. One of its tools is: Pth-Winexe. This tool is used to pass the hash by implementing a SMB client. This tool is also can be used by a real attacker that gained a privileged hash and wants to use it to connect to sensitive servers.

To simulate such a scenario, we used Kali's user *root* and the hash of a privileged account *bad1* in the following way:

root is logged on to *Kali* and pass the hash of the privileged account *bad1* to the tool *pth-winexe* together with the *MARS-10*'s IP and create a reverse shell to *MARS-10*.

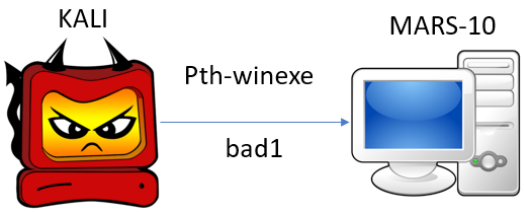

Figure 4. PTH-Winexe Simulation Diagrahm

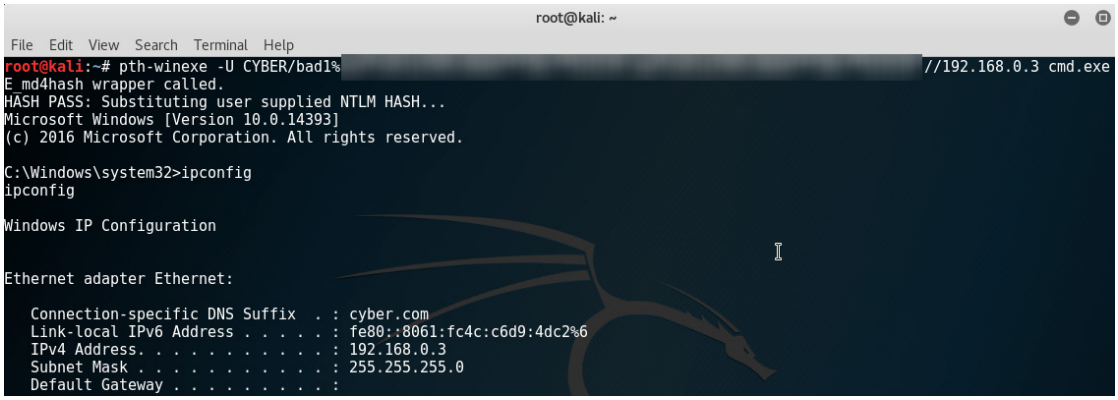

Figure 5. Passing the hash with Pth-winexe

Kali is a Linux distribution used as the source machine in this scenario and there are no logs on this machine.

After the user *root* executed *pth-winexe*, event 4672 is generated, indicating an initiation of a privileged session on the target machine (*MARS-10*).

Target machine (MARS-10, 192.168.0.3)									
Code	**Subject User Sid (resolved)**	**Target Domain Name \Target User Name**	**Workstation Name**	**Logon Type**	**IP address**	**Authentication Package Name**	**Subject Logon ID**	**Target Logon ID**	**Time seq.**
4672	CYBER\ bad1						0x8D7B5F		#1
4624	S-1-0-0	CYBER\bad1	KALI	3	192.168.0.2	NTLM	0x0	0x8D7B5F	#2
*4674	CYBER\ bad1						0x8D7B5F		#3

Figure 6. Event ID 4672 indicates a privileged session initiation on the target machine

* **4674** – "An operation was attempted on a privileged object." Only when enabling the security option "Audit: Audit the use of Backup and Restore privilege".

Domain controller (CYBER-DC, 192.168.0.1)				
Code	**Target User Name**	**Workstation**	**IP Address**	**Time seq.**
4776	bad1	KALI		#4

Table 3-2. Event ID 4776 on the domain controller

The tool *pth-winexe* is used as SMB client that installs the service *winexesvc* which creates a reverse shell. Similar tools for Windows can be found in the project Invoke-TheHash wrote by Kevin Robertson. One tool, the Invoke-SMBClient, is used as a SMB client and when running it on a Windows machine no logs are generated, only on the target. Because it is used as SMB client it doesn't need to write anything to LSASS.exe and no logon session is created on the source machine.

In this illegitimate scenario, passing the hash with implemented SMB client, there are no indications for a legitimate connection and therefore we will detect it as a suspicious connection.

Mounting administrative network share with Mimikatz from Windows 10 to Windows 7

One of the famous tools penetration testers and attackers are using for credentials-based attacks, wrote by Benjamin Delpy, is Mimikatz. This tool has the ability to extract credentials and also use them for attacks, in particular Pass-The-Hash.

To simulate such a scenario, we used the non-privileged account *student1*, privileged local account *local_admin* and privileged domain account *bad2* in the following way:

student1 is logged on to *MARS-10*, he is using Mimikatz, with local administrator account *local_admin*, to pass the hash of privileged account *bad2* in order to open an elevated command line interface (CLI). Then he mounts an administrative share by entering *MARS-7*'s IP address.

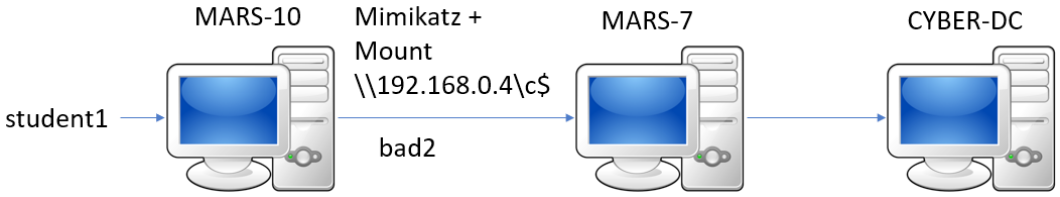

Figure 7. Mimikatz Simulation diagram

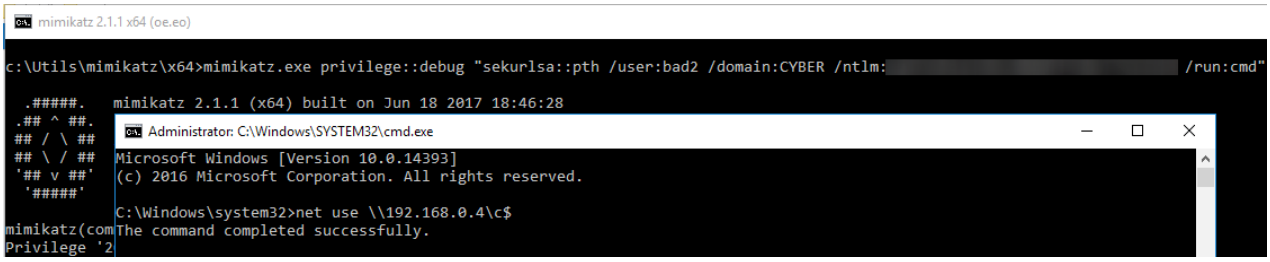

Figure 8. Passing the hash with Mimikatz

When our user, *local_admin* used the privileged account's hash bad2 to execute Pass-The-Hash attack and create the elevated CLI, event ID 4624 with logon type 9 (*NewCredentials*) will be generated on the source machine (*MARS-10*), indicating that a new logon session with two accounts has been created, the primary account (local_admin) is for local identity and the secondary (*bad2*) is for other network connections. The event could be a good indicator but it has number of drawbacks. The first drawback is that the same scenario on Windows 7 as the source machine has different results. The account (*bad2*) of the stolen hash is missing as depicted in the windows event captures in figure 9 (Win7 vs Win10).

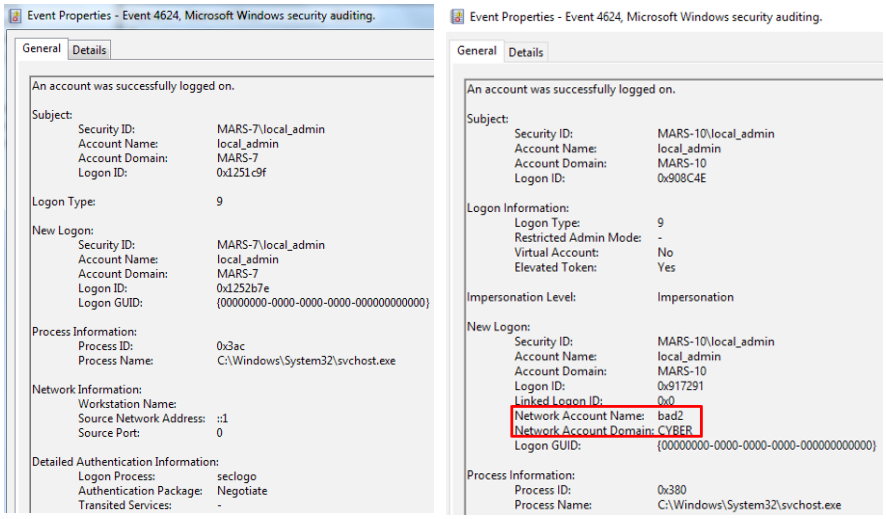

Figure 9. Compare between Event ID 4624: Windows 7 Event viewer- vs - Windows 10 Event viewer

A workaround for this can be by enabling "Microsoft-Windows-LSA/Operational" (disabled by default). Once event 4624 with logon type 9 occurred, at the same time ID 303 should appear:

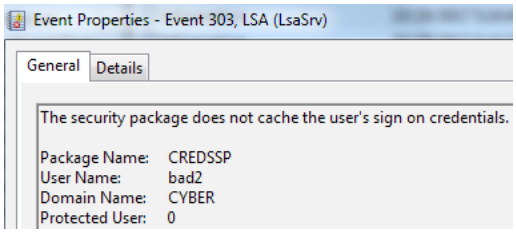

Figure 10. Event ID 303 indicates a credentials use

Another issue that came up is the gap of times between executing new logon session with Mimikatz to mounting administrative share. How much time backwards to look for event 4624 with logon type 9? Setting a constant time is not the best solution because it can cause false positive or false negative depending on the time that was set. For example, searching for event 4624 with logon type 9 one hour backward can cause false negative when the attacker waited two hours after opening the logon session and then executed commands on the remote machine.

Another option can be by creating a correlation. Once NTLM event was detected, searching on the source machine for event 4648 (with process id 0x4) at almost the same time (3-5 seconds before the NTLM event). Taking the **`SubjectLogonId`**'s value and search if there is an event 4624 (logon type 9) with the same value on the **`TargetLogonId`** variable. More generic approach will be to search for event 4648 **with** process id 0x4 at almost the same time of the NTLM event and mark it as illegitimate. It also possible to do it opposite and search for event 4648 **without** process id 0x4 and mark it as legitimate.

With all these suggestions, we need to recall that logon type 9 was not created especially for Mimikatz. It was used for applications that needed to work with different set of credentials locally than they do it remotely.

Microsoft even mentioned about the use of NewCredentials:

> This is useful in inter-domain scenarios where there is no trust relationship.

Mimikatz is using **CreateProcessWithLogonW** which together with the argument **LOGON _ NETCREDENTIALS _ ONLY** is generating event 4624 with logon type 9.

Module	API
mimikatz.exe	CreateProcessWithLogonW ("bad2", "CYBER", "", LOGON_NETCREDENTIALS_ONLY, NULL, "cmd", CREATE_NEW_CONSOLE | CREATE_SUSPENDED, NULL, NULL,

Figure 11. Mimikatz use of CreateProcessWithLogonW

An example for legitimate event 4624 with logon type 9 is the use of *runas /netonly* which acts similar to Mimikatz with the only exception that it doesn't change the logon session in LSASS.exe and doesn't use the **CREATE _ SUSPENDED** argument:

Module	API
runas.exe	CreateProcessWithLogonW ("bad2", "CYBER", [redacted], LOGON_WITH_PROFILE, NULL, "cmd", 0, NULL, NULL,

Figure 12. Ruas /netonly use of CreateProcessWithLogonW

After mounting the administrative share from the CLI, event ID 4648 will be generated on the source machine (*MARS-10*) indicating an explicit usage of credentials although no one entered new credentials. This is because the action of mounting an administrative share is a network task and because the logon session type is 9 (*NewCredentials*), it used its secondary account for this task which cause this event.

Source machine (MARS-10, 192.168.0.3)										
Code	**Subject User Sid (resolved)**	**Target Domain Name \Target User Name**	**Target Out-bound Domain Name \Target Outbound User Name**	**Worksta-tion Name**	**Logon Type**	**Auth. Package Name**	**Process name ma-chine \ Target Server Name**	**Subject Logon ID**	**Target Logon ID**	**Time seq.**
4624	MARS-10\ local_admin	MARS-10\lo-cal_admin	CYBER\bad2	MARS-10	9	Negotiate	C:\Windows\ System32\ svchost.exe	0x908C4E	**0x917291**	#1
4672	MARS-10\ local_admin							**0x917291**		#2
***4656**	MARS-10\ local_admin							0x908C4E		#3
After mounting the network share										
4648	MARS-10\ local_admin	CYBER\bad2					SYSTEM (PID 4) MARS-7. cyber.com	**0x917291**		#4

Table 7. Event ID 4624 with logon type 9 ("NewCredentials")

* This event will appear only if auditing on object has been enabled.

With event 4656 we can see writing to LSASS.exe memory and get more accurate detection:

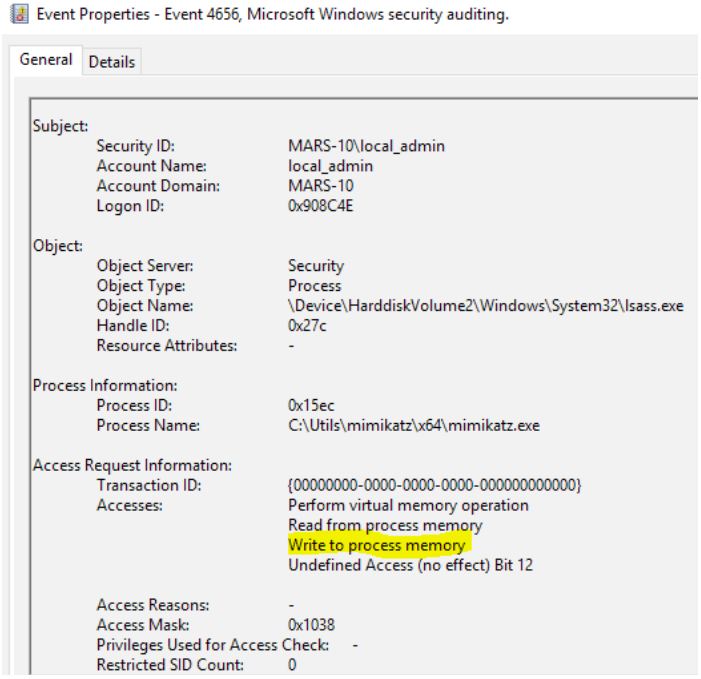

Figure 13. Event ID 4656 indicates writing to LSASS memory

This new information is applying to Windows 10 and Server 2016 and more information can be found here.

Same as in previous scenarios, the NTLM authentication attempt is registered as event ID 4776 on the domain controller.

Domain controller (CYBER-DC, 192.168.0.1)				
Code	**Target User Name**	**Workstation**	**IP Address**	**Time seq.**
4776	bad2	MARS-10		#7

Table 8. Event ID 4776 on the domain controller

As soon as the privileged account, *bad2*, is granted privileged access to the target machine (*MARS-7*), event ID 4672 is generated, indicating an initiation of a privileged session on the target machine (*MARS-7*).

Target machine (MARS-7, 192.168.0.4)								
Code	**Subject User Sid (resolved)**	**Target Domain Name \Target User Name**	**Workstation Name**	**Logon Type**	**Authentication Package Name**	**Subject Logon ID**	**Target Logon ID**	**Time seq.**
4672	CYBER\bad2					**0x11fca08**		#5
4624	S-1-0-0	CYBER\bad2	MARS-10	3	NTLM	0x0	**0x11fca08**	#6

Table 9. Event ID 4672 indicates a privileged session initiation on the target machine (MARS-7)

In this scenario, the event ID 4624 with logon type 9 is the most significant indication for the use of Mimikatz. But legitimate applications can also generate this event and therefore false positive can occurred.

Mounting administrative share from Windows 7 to Windows 10 with WCE

This scenario is similar to the previous scenario with Mimikatz but instead of using Mimikatz, WCE – Windows Credentials Editor by Hernan Ochoa. To simulate such scenario, we used the non-privileged account *student1*, privileged local account *local_admin* and privileged domain account *bad3* in the following way:

student1 is logged on to *MARS-7*, he is using WCE, with local administrator account *local_admin*, to pass the hash of privileged account *bad3* to the current logon session. Then he mounts an administrative share by entering *MARS-10*'s IP address.

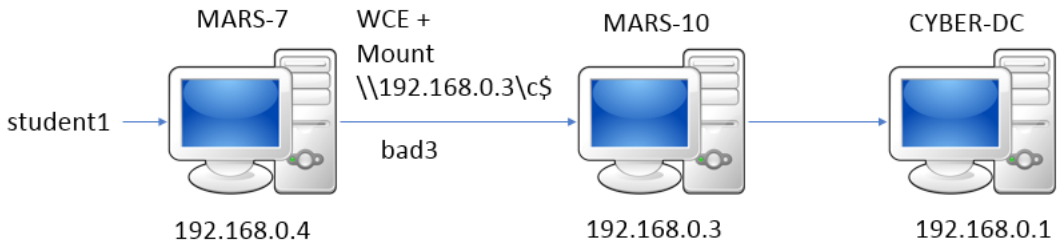

Figure 14. Mounting Administrative Share Simulation Diagram

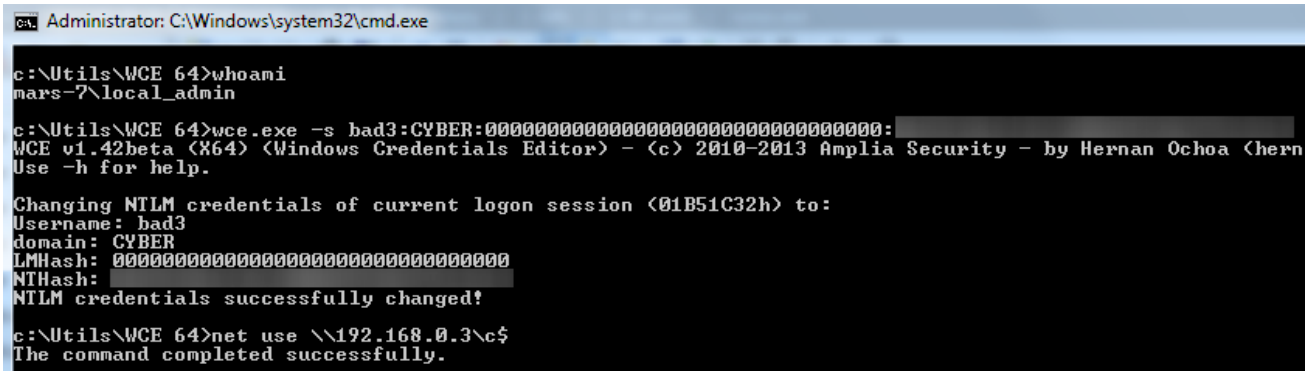

Figure 15. Passing the hash with Windows Credentials Editor

This scenario doesn't generate any trace logs on the security logs of the source machine (*MARS-7*). There are only new logs in the system events that indicates that a new service was created.

This is because WCE works differently from Mimikatz.

When we used Mimikatz to inject the hash into LSASS.exe memory and create new command line interface (CLI), logon event 4624 logon type 9 was generated. This is not the case when WCE (Windows Credentials Editor) is considered.

WCE doesn't use **`CreateProcessWithLogonW`** (although it has a call to this function in its code) in contrary to Mimikatz as we saw in the previous scenario.

WCE is creating a service named “WCESERVICE”:

```
A service was installed in the system.

Service Name:  WCESERVICE
Service File Name:  c:\Utils\WCE 64\wce.exe -S
Service Type:  user mode service
Service Start Type:  demand start
Service Account:  LocalSystem
```

Figure 16. WCE service creation log

The service has higher (SYSTEM) integrity over the process (High):

Process	CPU	Private Bytes	Working Set	PID	Integrity	Description	Company Name
SearchIndexer.exe		24,164 K	21,068 K	2636	System	Microsoft Windows Search I...	Microsoft Corporation
taskhost.exe	0.01	16,652 K	19,816 K	912	Medium	Host Process for Windows T...	Microsoft Corporation
PSEXESVC.exe		1,324 K	3,832 K	4072	System	PsExec Service	Sysinternals
wce.exe	< 0.01	824 K	2,632 K	1660	System		
lsass.exe	< 0.01	4,744 K	13,680 K	496	System	Local Security Authority Proc...	Microsoft Corporation
lsm.exe		2,296 K	4,312 K	504	System	Local Session Manager Serv...	Microsoft Corporation
csrss.exe	< 0.01	1,596 K	4,452 K	388	System	Client Server Runtime Process	Microsoft Corporation
winlogon.exe		2,416 K	6,908 K	424	System	Windows Logon Application	Microsoft Corporation
explorer.exe		19,328 K	33,380 K	2160	Medium	Windows Explorer	Microsoft Corporation
csrss.exe	0.05	10,560 K	9,348 K	2816	System	Client Server Runtime Process	Microsoft Corporation
winlogon.exe		2,820 K	7,528 K	2844	System	Windows Logon Application	Microsoft Corporation
explorer.exe	0.08	109,832 K	135,372 K	1036	Medium	Windows Explorer	Microsoft Corporation
x64dbg.exe	9.17	72,028 K	110,036 K	1588	High	x64dbg	
wce.exe	0.01	580 K	2,588 K	3528	High		

Figure 17. WCE with two instances on Process Explorer

The process is sending the logon session information through pipe \\.\pipe\WCEServicePipe to the service:

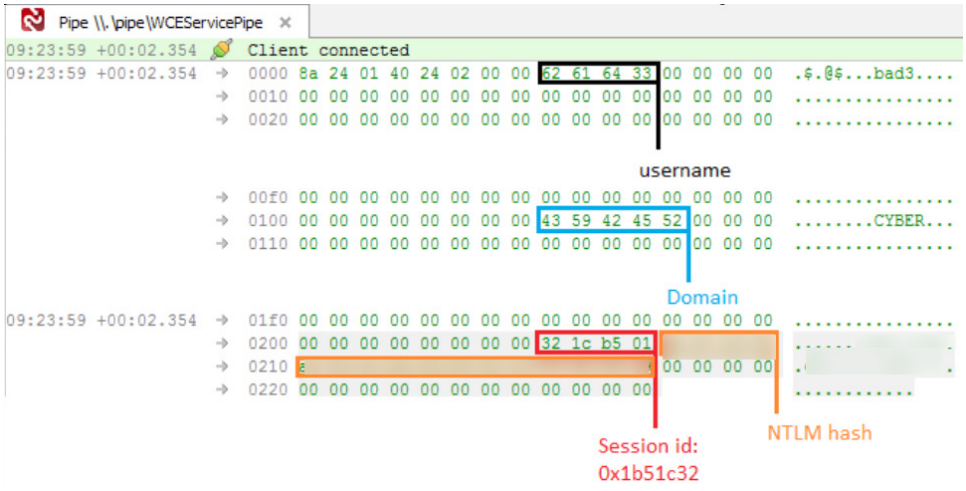

Figure 18. Logon session transferred through the pipeline

The service unpacks a binary resource named wceaux.dll. It injects the DLL to LSASS.exe

Process Na...	PID	Operation	Path	Result	Detail
lsass.exe	496	Load Image	C:\Windows\Temp\wceaux.dll	SUCCESS	Image Base: 0x7fefaaf0000, Image Size: 0x10000

Figure 19. LSASS.exe loads wceaux.dll

Then it uses the exported function **`WCEAddNTLMCredentials`** to change the logon session in LSASS.exe. At the time of writing this article, WCE doesn't support Windows 10. It supports Windows XP, Windows 2003, Vista, Windows 7 and Windows 2008 (all SPs, 32bit and 64bit versions). Same as in previous scenarios, the NTLM authentication attempt is registered as event ID 4776 on the domain controller.

Domain controller (CYBER-DC, 192.168.0.1)				
Code	**Target User Name**	**Workstation**	**IP Address**	**Time seq.**
4776	bad3	MARS-7		#3

Table 10. Event ID 4776 on the domain controller

As soon as the privileged account, *bad3*, is granted privileged access to the target machine (MARS-10), event ID 4672 is generated, indicating the initiation of a privileged session on the target machine (MARS-10).

Target machine (MARS-10, 192.168.0.3)								
Code	**Subject User Sid (resolved)**	**Target Domain Name \Target User Name**	**Workstation Name**	**Logon Type**	**Authentication Package Name**	**Subject Logon ID**	**Target Logon ID**	**Time seq.**
4672	CYBER\bad3					0xBAD9AB		#1
4624	S-1-0-0	CYBER\bad3	MARS-10	3	NTLM	0x0	0xBAD9AB	#2

Table 11. Event ID 4672 indicates a privileged session initiation on the target machine (MARS-10)

In this scenario, of using WCE to pass the hash, there was no indication for a legitimate nor illegitimate connection by the user *bad3*. In such case we will detect it as a suspicious connection.

Identifying legitimate NTLM connections

Identifying when the user provided the password interactively can filter out illegitimate connections such as Pass-The-Hash.

The following events can be used to identify a legitimate logon:

1. Events 4768 (TGT) and 4769 (TGS):
   These events are logged in domain controllers only and indicate a request for TGT or TGS tickets. The account and the workstation appear in the logs:

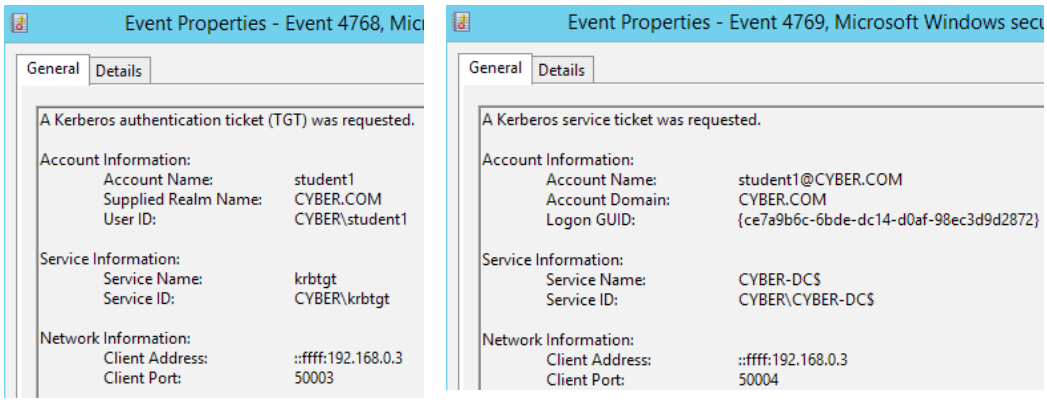

Figure 20. Events ID 4768 and 4769

2. Event 4624 with logon types: 2 (Interactive), 7 (Unlock), 10 (RemoteInteractive) or 11 (CachedInteractive).
   This event with the different logon types indicates when the user accesses the machine interactively.
3. Event 4648:
   This event indicates when a user or process are using alternate credentials.

Notice that events 4768 and 4769 can be used instead of event 4624 and vice versa to identify a legitimate logon. The difference between them is that 4768 and 4769 are generated on the domain controller while event 4624 is generated on the source machine.

Privileged NTLM connections

Not every NTLM connection is being used with privileged account. Focusing on the NTLM connections that are being used with privileged accounts can reduce the amount of false positive significantly. Event 4672 ("Special privileges assigned to new logon") let us know when a privileged account logs on.

Furthermore, checking to see if the user is privileged based on their membership in an AD group is not enough. There is a possible scenario were accounts are not part of any privileged AD group but have local administrator permissions on the target machine.

Creating correlation between the NTLM event (4624) to event 4672 is possible by checking that event 4624's **`TargetLogonId`** is the same as in event 4672's **`SubjectLogonId`**.

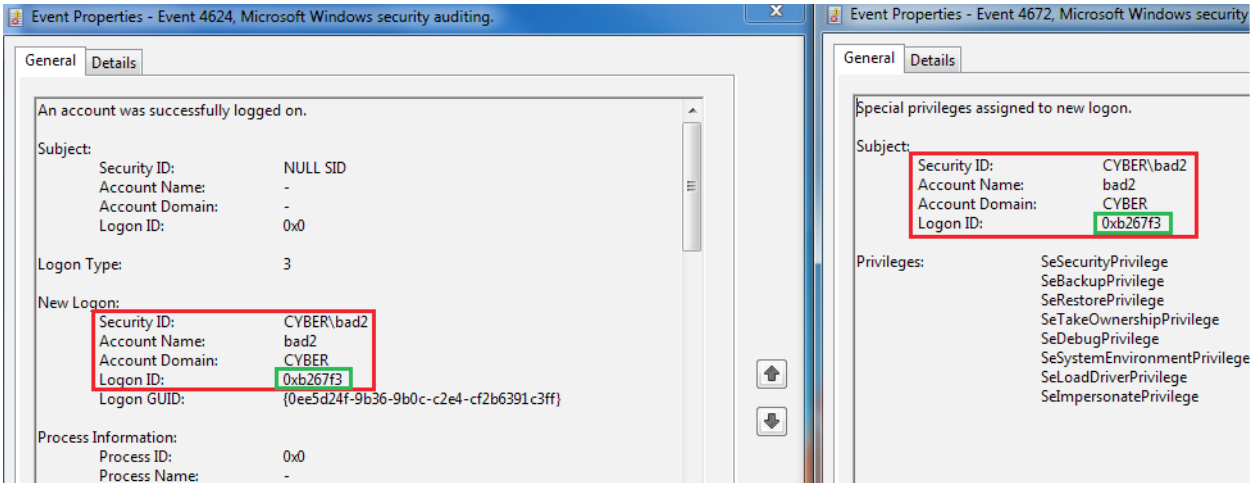

Figure 21. Correlating Events 4624 and 4672 by TargetLogonId and SubjectLogonId

Putting the pieces together

A proposed alogrithm

Using the above details, we will show how we can use it to detect pass-the-hash.

1. Monitor only privileged NTLM connections by correlating between event 4624 to event 4672 based on the logon ID.
2. [optional] Check that the computer is in active directory of the, if not => mark as suspicious
3. [optional] If the account from the NTLM event had previously (can set time or create correlation with the logon IDs) logon event 4624 with logon type 9 => mark as suspicious
   * With Windows 10 and Server 2016 there are logs that indicate writing to LSASS.exe's memory and provide very accurate detection.
4. If the account from the NTLM event generated a request for TGT (4768) or TGS (4769) from the source machine **`X`** hours before the NTLM event => mark as legitimate

5. If the account from NTLM event generated logon event 4624 (with logon types 2,7, 10 or 11) from the source machine **X** hours before the NTLM event => mark as legitimate
6. If the account from NTLM event generated event 4648 (explicit credential use) **without** process id **0x4** from the source machine **Y** seconds before the NTLM event => mark as legitimate
7. Everything else => mark as suspicious

Notes:

- Choosing between sections 4 and 5, will be the same as both of them deserve the same goal. It is preferred to use section 5 because it removes the dependency on getting the logs from a third machine - domain controller and anyway there is a need to check event 4648 on the source machine.
- Section 6 cannot replace 4 and 5 because there are scenarios when a privileged account is logged on interactively and connects to a remote machine over NTLM. In this case event 4648 won't appear.
- Section 3 is optional and will cause constant false positive for applications that generates event 4624 with logon type 9. Section 6 already covers section 3.
- Any change to the X and Y variables will change the level of false positive and false negative.
  The more X or Y will increase, less false positive and more false negative events will appear.
  The more X or Y will decrease, more false positive and less false negative events will appear.

To illustrate the principals of this research, CyberArk Labs has release a new tool that implements the detections described in this research paper in a tool called – Ketshash.

Flow diagram

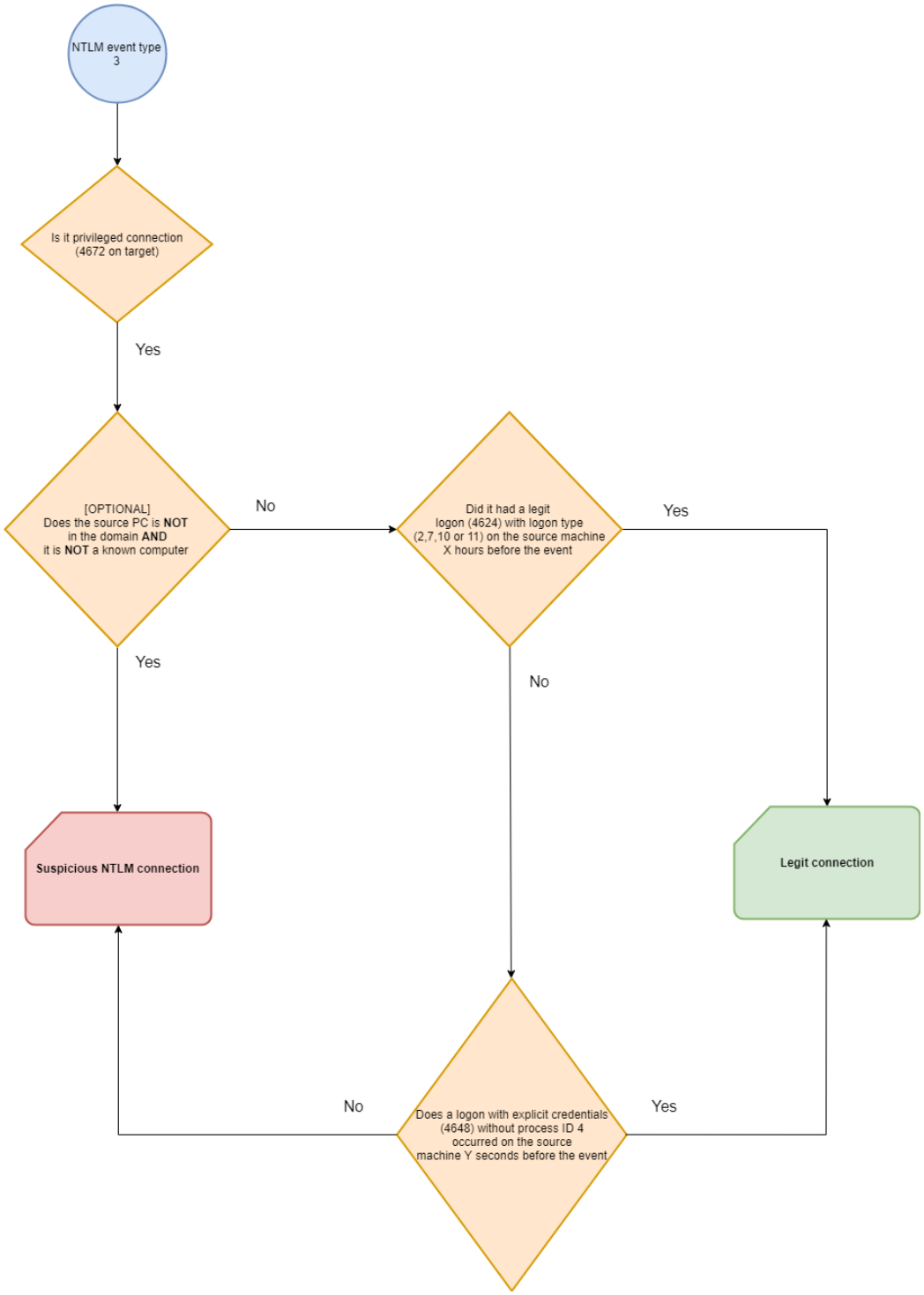

Figure 22. Example for diagram based on section 4

Advantages

- Covers all pass the hash attacks over the network

Disadvantages

- There is a gap that can lead to false negative or false positive, depending on the time that was set to check the legitimate logon events (4768, 4769, 4624 and 4648)
- Simulating legitimate events, such as creating privileged scheduled tasks with Pass-The-Hash which will generated event 4648, before the NTLM event can cause false negative
- Requires to be able to get logs from endpoints in the organization
- Requires for the involved machines to be synchronized on time

This is just one example on how to put this research into a detection approach for Pass-the-hash. More variants of this algorithm are possible with the research supplied.

If there are too many false positives it is possible to monitor only sensitive machines for NTLM connections and reduce the false positives. It is important to understand that although false positive events might appear, they are still NTLM connections. The NTLM protocol has been exposed to large amount of attacks over the years and it is recommended to use Kerberos authentication instead, as Microsoft mentioned here:

> **The Microsoft Kerberos security package adds greater security than NTLM to systems on a network.**

Things to keep in mind

When implementing any security detection there are always items to keep in mind.

Timing is everything

Timing is very important when creating correlations between the events. Machines with even the slightest asynchrony time between them can affect the results.

One example:

Machine	Current Time
Source Machine	16:00
Target Machine	15:55

A Legitimate NTLM event occurred on the target machine via NTLM at 15:55 after the user logged in at 16:00 on the source machine. In such cases, when searching for the legitimate logons prior to 15:55, no legitimate event will be found, producing a false negative alert.

One way to resolve this is to check the time of the source and the target machines in **parallel** (to prevent possible delays) and adjust one of them to the time of the other. Following our example, checking first the time difference between the source and the target yields -5. Therefore, searching for a legitimate logon should start from target machine's current time (15:55) + 5 minutes which is 16:00 on the source machine.

It also should be noted that even the time check itself might suffer from time delay. Checking the time on a machine can take time and therefore it should be measured in order to remove the overhead. For example, two machines that are synchronized, might be considered unsynchronized if on one machine, the time check takes 5 seconds.

In a perfect world, we would expect that all the machines will be synchronized by a time service and therefore be synchronized to the same time but this is often not the case in reality, machines are not always synchronized with the same time. It was also mentioned by Microsoft that it is possible for two computers to be often out of sync.

Sabotaging the event viewer

The detection in this paper is based on event viewer data and without logs it is not possible to detect suspicious NTLM connection. Using tools like Invoke-Phant0m it is possible to sabotage the event viewer or just changing the times on the source machine can affect the logs indirectly.

Stopping from event viewer to generate logs or hide them on the source machine will still be detected as illegitimate connection because without logs on the source machine it is not possible to mark the connection as legitimate.

A more concerning issue can be when the attacker is familiar with the detection algorithm and edits the logs on the source machine to look like a legitimate logon. In such case, the use of Kerberos events on the domain controller can have an advantage.

Another problem is when the attacker succeeds to hide the NTLM events (event 4624, logon type 3) on the target machine. In order to sabotage the logs, the attacker will need to connect the machine by NTLM connection which will generate a NTLM event. Forwarding the logs once they are generated can detect the NTLM connection before they are sabotaged.

Summary

This paper documented a way to detect suspicious NTLM connections in general and pass the hash in particular. A key recommendation to do this detection is to get the logs from the source machine. By changing some filtering parameters such as time, accounts (only privileged) and machines (only sensitive to the organization) the company wants to monitor, it is possible to reduce many false positives. All other detections that turn out to be false positives should warrant a check and determination if it is possible to change the authentication method to a more secure one.

We used an approach that requires to connect the machines directly, and while the mechanism to get the logs is out of scope for this research, it is worth noting that another approach could be by forwarding all the event logs to one place (like WEF server) which has its own advantages and disadvantages.

In the end, understanding the overall picture of NTLM connections, in particular privileged connections, in an organization can help be a catalyst for change to a more secure connection strategy and prevention of future attacks.

References

https://blogs.msdn.microsoft.com/chiranth/2013/09/20/ntlm-want-to-know-how-it-works/

https://support.microsoft.com/en-us/help/322979/kerberos-is-not-used-when-you-connect-to-smb-shares-by-using-ip-addres

https://www.sans.org/reading-room/whitepapers/testing/crack-pass-hash-33219

https://www.sans.org/reading-room/whitepapers/testing/pass-the-hash-attacks-tools-mitigation-33283

https://www.blackhat.com/docs/us-14/materials/us-14-Hathaway-Why-You-Need-To-Detect-More-Than-PtH-WP.pdf

TWC: Pass-the-Hash: How Attackers Spread and How to Stop Them

http://www.labofapenetrationtester.com/2017/08/week-of-evading-microsoft-ata-day2.html

http://blog.gentilkiwi.com/securite/mimikatz/overpass-the-hash

https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don't-Get-It-wp.pdf