Report
Threat Hunting Splunk

***Author** : M Arif Rizki*

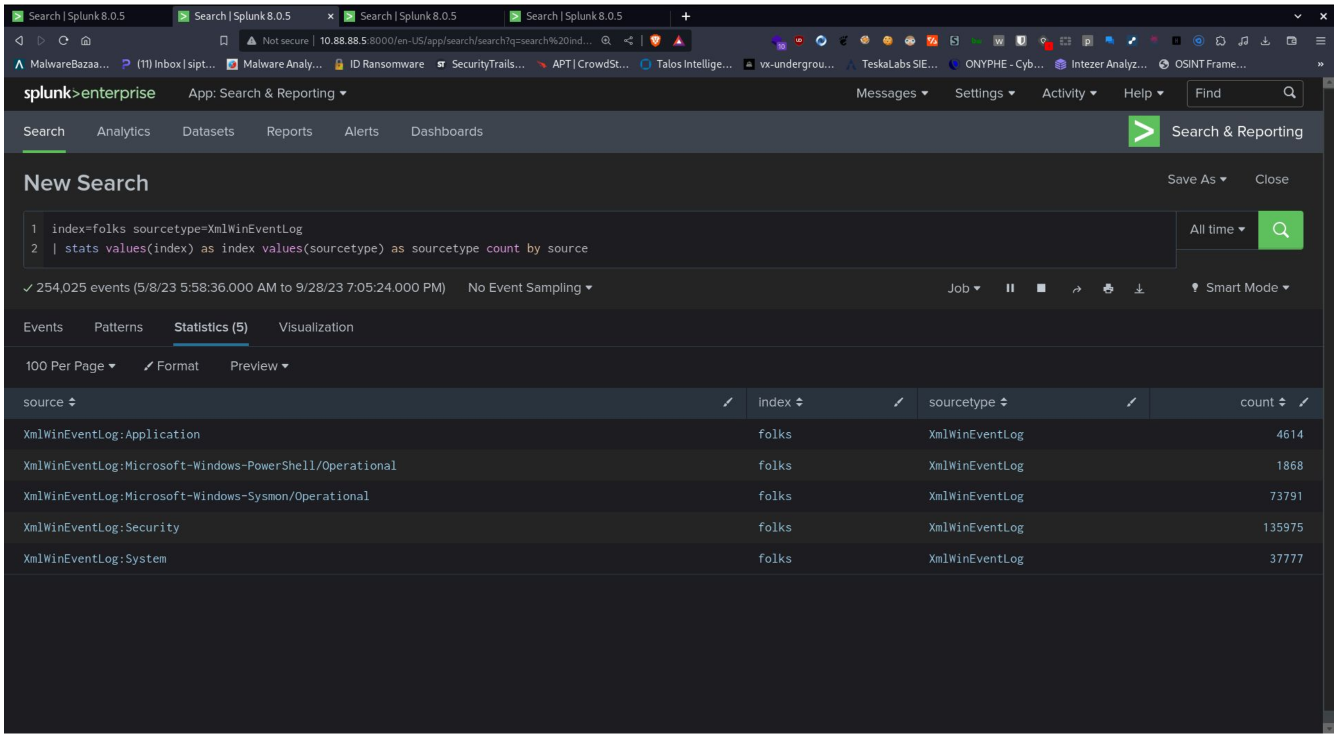
Overview

As a SOC analyst, you aim to investigate a security breach in an Active Directory network using Splunk SIEM (Security information and event management) solution to uncover the attacker's steps and techniques while creating a timeline of their activities. The investigation begins with network enumeration to identify potential vulnerabilities. Using a specialized privilege escalation tool, the attacker exploited an unquoted service path vulnerability in a specific process.

Once the attacker had elevated access, the attacker launched a DCsync attack to extract sensitive data from the Active Directory domain controller, compromising user accounts. The attacker employed evasion techniques to avoid detection and utilized pass-the-hash (pth) attack to gain unauthorized access to user accounts. Pivoting through the network, the attacker explored different systems and established persistence.

Throughout the investigation, tracking the attacker's activities and creating a comprehensive timeline is crucial. This will provide valuable insights into the attack and aid in identifying potential gaps in the network's security.

The log sources returned include :
“XmlWinEventLog:Security”: Contain events related to Windows authentication and security processes.
“XmlWinEventLog:Application”: Contain events logged by various applications and/or user programs, including any errors or info that an application is deigned to report.
“XmlWinEventLog:System”: Contain events logged by various Windows system components.
“XmlWinEventLog:Mircrosoft-Windows-Sysmon/Operational”: commonly used add-on for Windows logging. With Sysmon logs, you can detect malicious activity by tracking code behavior and network traffic, as well as create detections based on the malicious activity.
“XmlWinEventLog:Microsoft-Windows-PowerShell/Operational”: logs PowerShell activity.



```
index=folks sourcetype=XmlWinEventLog
| stats values(index) as index values(sourcetype) as sourcetype count by source
```

✓ 254,025 events (5/8/23 5:58:36.000 AM to 9/28/23 7:05:24.000 PM) No Event Sampling

source	index	sourcetype	count
XmlWinEventLog:Application	folks	XmlWinEventLog	4614
XmlWinEventLog:Microsoft-Windows-PowerShell/Operational	folks	XmlWinEventLog	1868
XmlWinEventLog:Microsoft-Windows-Sysmon/Operational	folks	XmlWinEventLog	73791
XmlWinEventLog:Security	folks	XmlWinEventLog	135975
XmlWinEventLog:System	folks	XmlWinEventLog	37777

Since we are investigating a breach into an **Active Directory** (**AD**) network, I decided to start hunting for common tactics and techniques used by threat actors once they have gained an initial foothold inside an AD network.

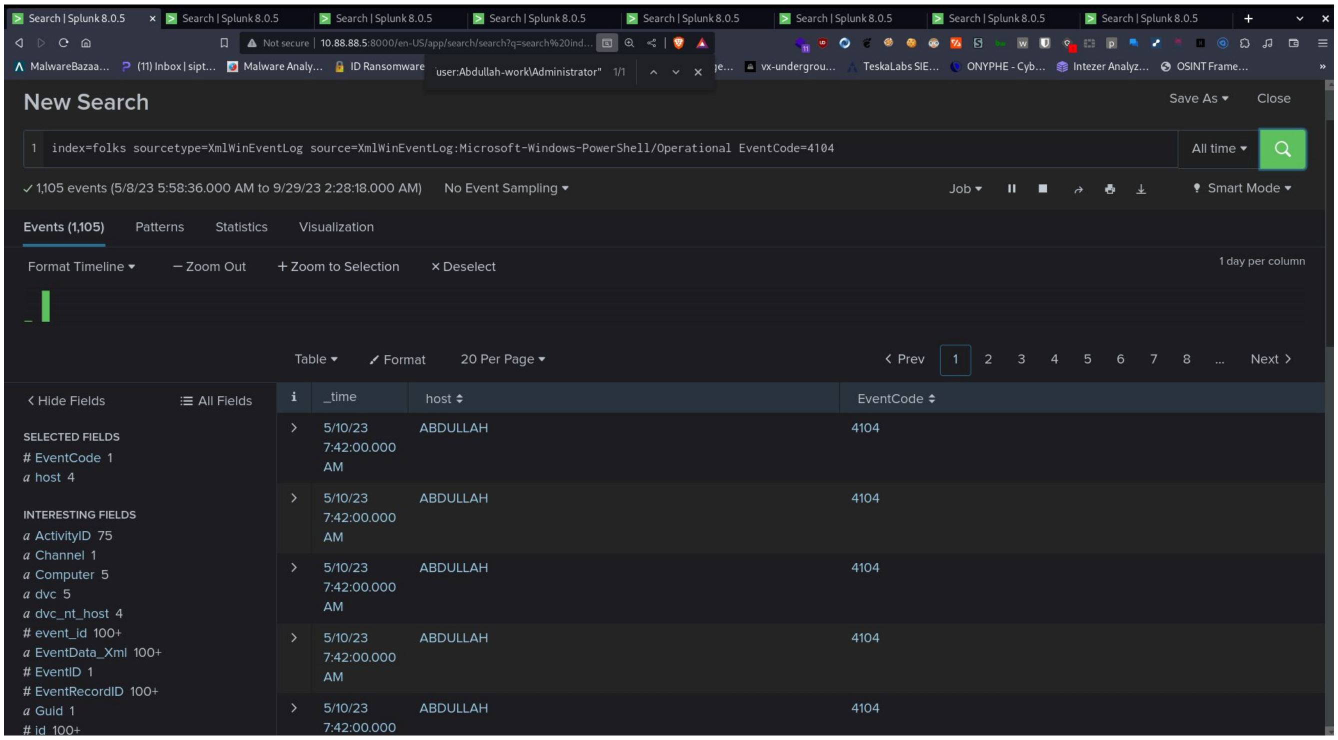
I decided to start by looking at the PowerShell logs, specifically event ID 4104, which is commonly used by adversaries (**MITRE T1059.001**).

Command and Scripting Interpreter: PowerShell : *https://attack.mitre.org/techniques/T1059/001/*

Event ID 4104 — Powershell Script Block Logging — Captures the entire scripts that are executed by remote machines. For example, obfuscated scripts that are decoded and executed at run time.
I used the following query to return PowerShell event ID's 4104 and spent sometime searching through the results.

Event ID 4104 : https://www.iblue.team/incident-response-1/logging-powershell-activities

Splunk Search :
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-PowerShell/Operational EventCode=4104


Search | Splunk 8.0.5
Not secure | 10.88.88.5:8000/en-US/app/search/search?q=search%20ind...
MalwareBazaa... (11) Inbox | sipt... Malware Analy... ID Ransomware vx-undergrou... TeskaLabs SIE... ONYPHE - Cyb... Intezer Analyz... OSINT Frame...
user:Abdullah-work\Administrator" 1/1
New Search
Save As
Close
All time
Job
Smart Mode
Format Timeline
Zoom Out
Zoom to Selection
Deselect
1 day per column
Table
Format
20 Per Page
Prev 1 2 3 4 5 6 7 8 ... Next
Hide Fields
All Fields

```
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-PowerShell/Operational EventCode=4104
```

✓ 1,105 events (5/8/23 5:58:36.000 AM to 9/29/23 2:28:18.000 AM) No Event Sampling

Events (1,105) Patterns Statistics Visualization

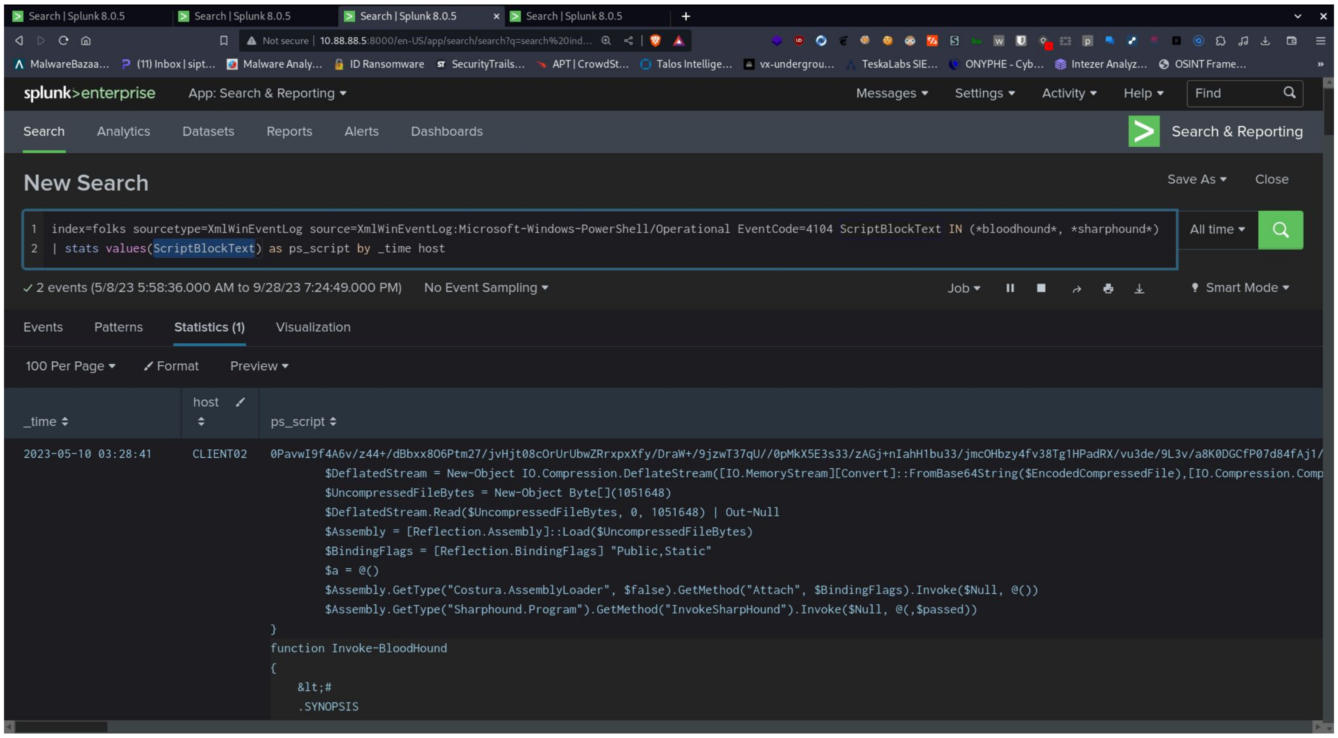
SELECTED FIELDS
EventCode 1
a host 4

INTERESTING FIELDS
a ActivityID 75
a Channel 1
a Computer 5
a dvc 5
a dvc_nt_host 4
# event_id 100+
a EventData_Xml 100+
EventID 1
EventRecordID 100+
a Guid 1
# id 100+

i	_time	host	EventCode
>	5/10/23 7:42:00.000 AM	ABDULLAH	4104
>	5/10/23 7:42:00.000 AM	ABDULLAH	4104
>	5/10/23 7:42:00.000 AM	ABDULLAH	4104
>	5/10/23 7:42:00.000 AM	ABDULLAH	4104
>	5/10/23 7:42:00.000	ABDULLAH	4104

Looking through the output, I discovered that the attacker had used Bloodhound to enumerate the active directory network. The original query above was further refined to return the time, host and user ID related to the PowerShell script execution.
Attackers have used Bloodhound to enumerate active network directories >
https://github.com/BloodHoundAD/BloodHound

Splunk Search :
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-PowerShell/Operational EventCode=4104 ScriptBlockText IN (*bloodhound*, *sharphound*)
| stats values(ScriptBlockText) as ps_script by _time host



```
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-PowerShell/Operational EventCode=4104 ScriptBlockText IN (*bloodhound*, *sharphound*)
| stats values(ScriptBlockText) as ps_script by _time host
```

✓ 2 events (5/8/23 5:58:36.000 AM to 9/28/23 7:24:49.000 PM)

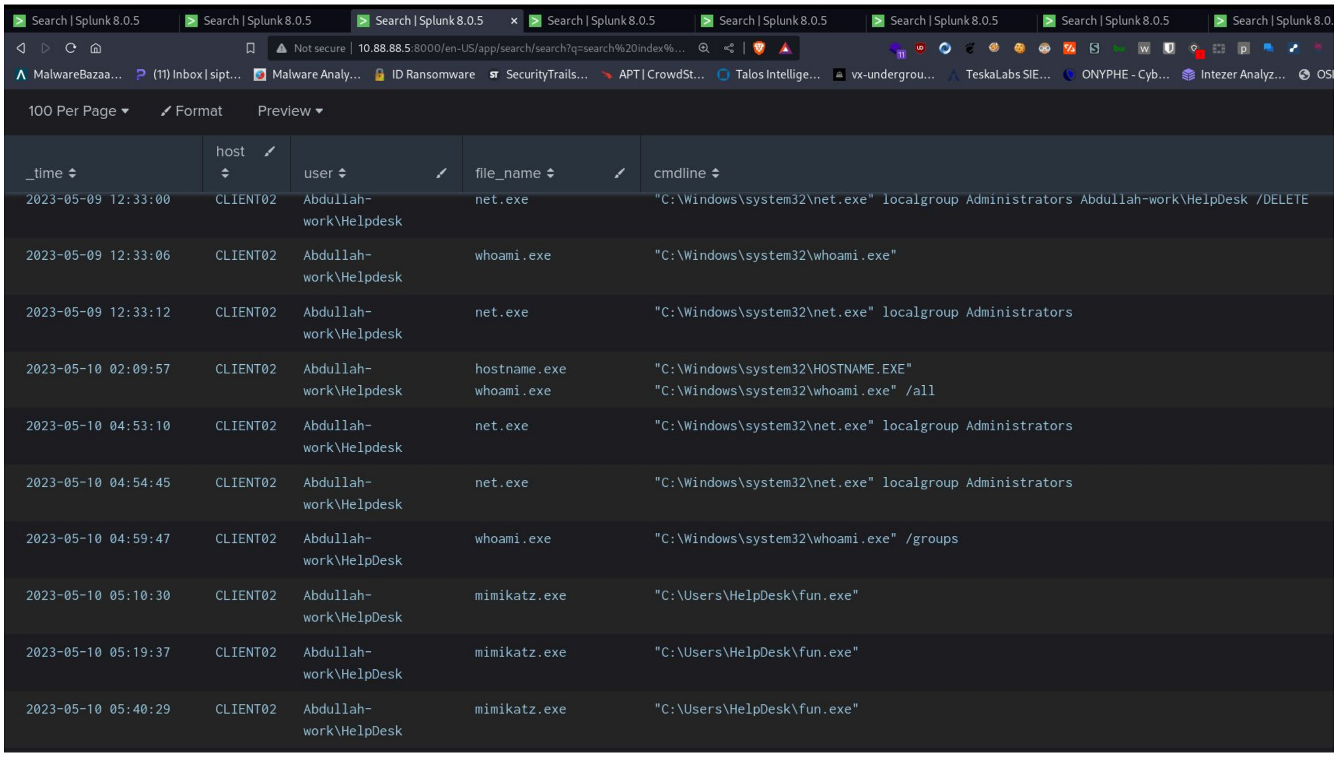
_time	host	ps_script
2023-05-10 03:28:41	CLIENT02	(see below)

```
0PavwI9f4A6v/z44+/dBbxx8O6Ptm27/jvHjt08cOrUrUbwZRrxpxXfy/DraW+/9jzwT37qU//0pMkX5E3s33/zAGj+nIahH1bu33/jmcOHbzy4fv38Tg1HPadRX/vu3de/9L3v/a8K0DGCfP07d84fAj1/
        $DeflatedStream = New-Object IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String($EncodedCompressedFile),[IO.Compression.Comp
        $UncompressedFileBytes = New-Object Byte[](1051648)
        $DeflatedStream.Read($UncompressedFileBytes, 0, 1051648) | Out-Null
        $Assembly = [Reflection.Assembly]::Load($UncompressedFileBytes)
        $BindingFlags = [Reflection.BindingFlags] "Public,Static"
        $a = @()
        $Assembly.GetType("Costura.AssemblyLoader", $false).GetMethod("Attach", $BindingFlags).Invoke($Null, @())
        $Assembly.GetType("Sharphound.Program").GetMethod("InvokeSharpHound").Invoke($Null, @(,$passed))
}
function Invoke-BloodHound
{
    &lt;#
    .SYNOPSIS
```

I can see that Bloodhound was executed on the host named “**CLIENT02**” (i.e. the compromised machine) at “2023–05–10 03:28:41”. Now that I know the compromised host is “**CLIENT02**”, I decided to start looking into Sysmon events.

I decided to look for command line activity by filtering for Sysmon Event ID 1 and setting a filter for the parent process to be PowerShell. I also set the time range for event since “2023–05–10”.

Splunk Search :
index=folks sourcetype=XmlWinEventLog
source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=1
ParentImage="C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"
| stats values(User) as user values(OriginalFileName) as file_name values(CommandLine) as cmdline by _time host



_time	host	user	file_name	cmdline
2023-05-09 12:33:00	CLIENT02	Abdullah-work\Helpdesk	net.exe	"C:\Windows\system32\net.exe" localgroup Administrators Abdullah-work\HelpDesk /DELETE
2023-05-09 12:33:06	CLIENT02	Abdullah-work\Helpdesk	whoami.exe	"C:\Windows\system32\whoami.exe"
2023-05-09 12:33:12	CLIENT02	Abdullah-work\Helpdesk	net.exe	"C:\Windows\system32\net.exe" localgroup Administrators
2023-05-10 02:09:57	CLIENT02	Abdullah-work\Helpdesk	hostname.exe whoami.exe	"C:\Windows\system32\HOSTNAME.EXE" "C:\Windows\system32\whoami.exe" /all
2023-05-10 04:53:10	CLIENT02	Abdullah-work\Helpdesk	net.exe	"C:\Windows\system32\net.exe" localgroup Administrators
2023-05-10 04:54:45	CLIENT02	Abdullah-work\Helpdesk	net.exe	"C:\Windows\system32\net.exe" localgroup Administrators
2023-05-10 04:59:47	CLIENT02	Abdullah-work\HelpDesk	whoami.exe	"C:\Windows\system32\whoami.exe" /groups
2023-05-10 05:10:30	CLIENT02	Abdullah-work\HelpDesk	mimikatz.exe	"C:\Users\HelpDesk\fun.exe"
2023-05-10 05:19:37	CLIENT02	Abdullah-work\HelpDesk	mimikatz.exe	"C:\Users\HelpDesk\fun.exe"
2023-05-10 05:40:29	CLIENT02	Abdullah-work\HelpDesk	mimikatz.exe	"C:\Users\HelpDesk\fun.exe"

the name of the compromised account : **Abdullah-work\Helpdesk**
Name of the compromised account: **CLIENT02**
Tools used by attackers : **bloodhound**

Looking at the first several events, I can see that the threat actor performed some enumeration with “whoami.exe” at “2023–05–10 02:09:57” (followed by the use of bloodhound at “2023–05–10 03:28:41”) and then some additional enumeration with “net.exe”. We can then see that the threat actor uses “mimkatz.exe”. This activity was all performed under the account “Abdullah-work\HelpDesk” (i.e. compromised account).

An unquoted service path vulnerability occurs when the file path associated with a Windows service does not have quotes (“”) (**MITRE T1574.009**).

Hijack Execution Flow: Path Interception by Unquoted Path :
https://attack.mitre.org/techniques/T1574/009/

Unsafe
C:\Program Files\MyService\MyService.exe

Safe
"C:\Program Files\MyService\MyService.exe"

An attacker could exploit this vulnerability by placing a malicious executable named "C:\Program.exe" to gain unauthorized access. As you can see there are some spaces in the PATH and Windows OS looks at the PATH like this when a service is starting.

C:\Program.exe

C:\Program Files.exe
C:\Program Files\Unquoted.exe
C:\Program Files\Unquoted Path.exe
C:\Program Files\Unquoted Path Service.exe
C:\Program Files\Unquoted Path Service\Common.exe
C:\Program Files\Unquoted Path Service\Common.exe

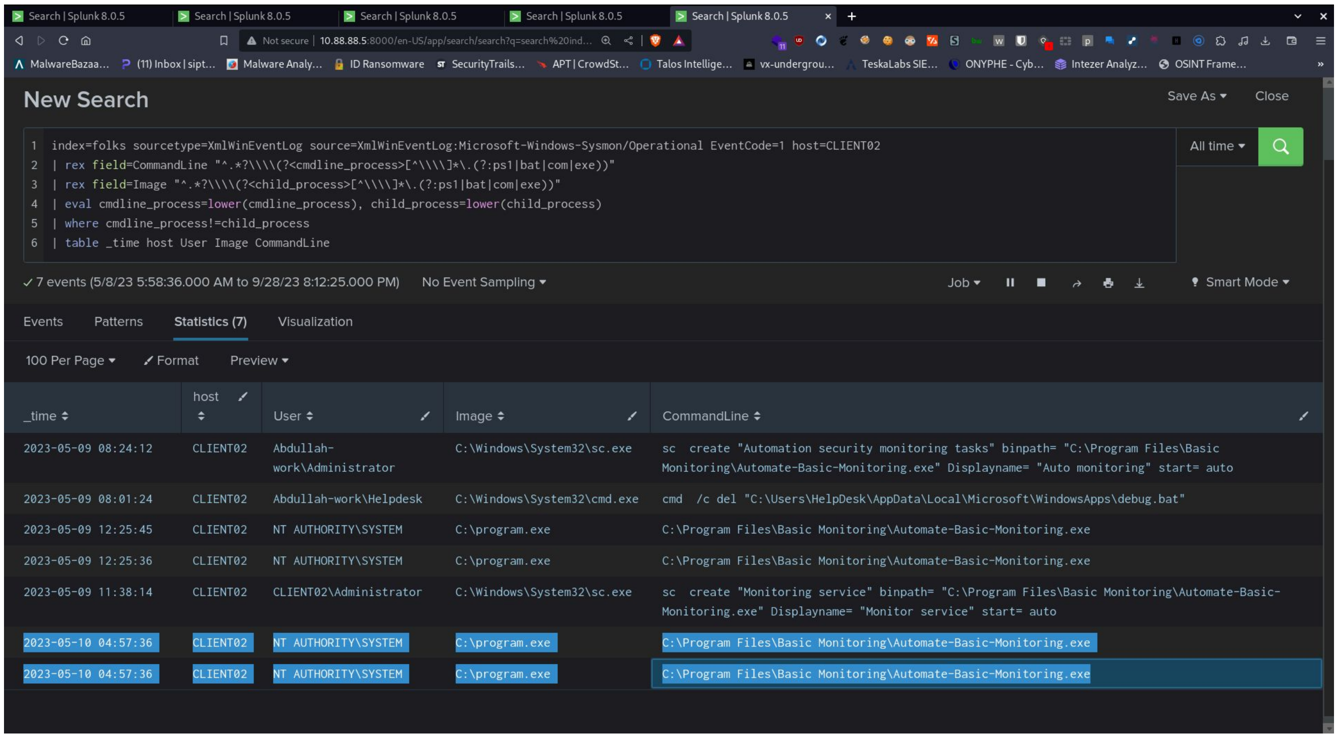
While doing some research, I came across a **Splunk detection** that looks for the abuse of unquoted service paths. The detection is searching for a discrepancy between the process name and the service process name. We can use a similar approach with Sysmon Event Code 1, as seen with the query below.

Detect Path Interception By Creation Of program exe :
https://research.splunk.com/endpoint/cbef820c-e1ff-407f-887f-0a9240a2d477/

Search Splunk :

```
index=folks sourcetype=XmlWinEventLog
source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational EventCode=1 host=CLIENT02
| rex field=CommandLine "^.*?\\\\(?<cmdline_process>[^\\\\]*\.(?:ps1|bat|com|exe))"
| rex field=Image "^.*?\\\\(?<child_process>[^\\\\]*\.(?:ps1|bat|com|exe))"
| eval cmdline_process=lower(cmdline_process), child_process=lower(child_process)
| where cmdline_process!=child_process
| table _time host User Image CommandLine
```

In the context of an unquoted service path vulnerability, the command line parameter typically includes the full path to the executable, whereas the Image field in the Sysmon event represents the actual executable file that was executed. Running this query, we get the following results.



```
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational EventCode=1 host=CLIENT02
| rex field=CommandLine "^.*?\\\\(?<cmdline_process>[^\\\\]*\.(?:ps1|bat|com|exe))"
| rex field=Image "^.*?\\\\(?<child_process>[^\\\\]*\.(?:ps1|bat|com|exe))"
| eval cmdline_process=lower(cmdline_process), child_process=lower(child_process)
| where cmdline_process!=child_process
| table _time host User Image CommandLine
```

✓ 7 events (5/8/23 5:58:36.000 AM to 9/28/23 8:12:25.000 PM)

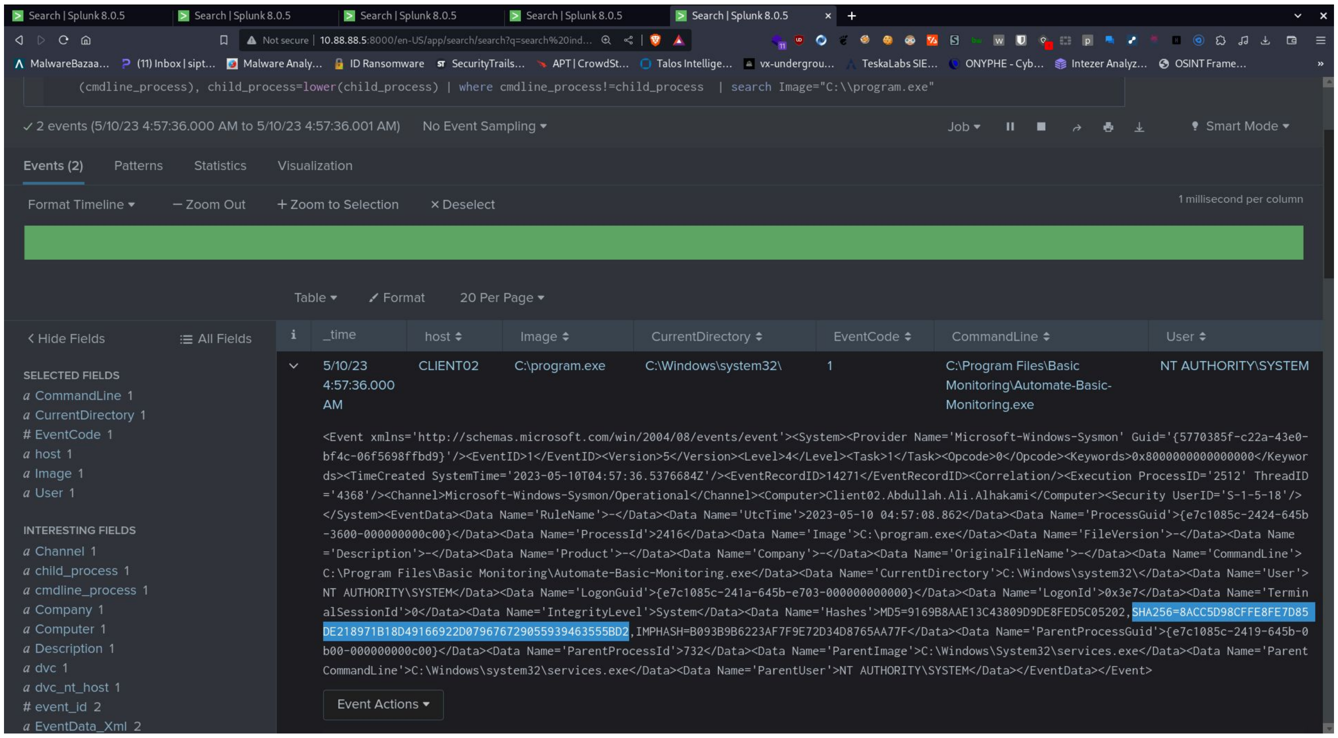
_time	host	User	Image	CommandLine
2023-05-09 08:24:12	CLIENT02	Abdullah-work\Administrator	C:\Windows\System32\sc.exe	sc create "Automation security monitoring tasks" binpath= "C:\Program Files\Basic Monitoring\Automate-Basic-Monitoring.exe" Displayname= "Auto monitoring" start= auto
2023-05-09 08:01:24	CLIENT02	Abdullah-work\Helpdesk	C:\Windows\System32\cmd.exe	cmd /c del "C:\Users\HelpDesk\AppData\Local\Microsoft\WindowsApps\debug.bat"
2023-05-09 12:25:45	CLIENT02	NT AUTHORITY\SYSTEM	C:\program.exe	C:\Program Files\Basic Monitoring\Automate-Basic-Monitoring.exe
2023-05-09 12:25:36	CLIENT02	NT AUTHORITY\SYSTEM	C:\program.exe	C:\Program Files\Basic Monitoring\Automate-Basic-Monitoring.exe
2023-05-09 11:38:14	CLIENT02	CLIENT02\Administrator	C:\Windows\System32\sc.exe	sc create "Monitoring service" binpath= "C:\Program Files\Basic Monitoring\Automate-Basic-Monitoring.exe" Displayname= "Monitor service" start= auto
2023-05-10 04:57:36	CLIENT02	NT AUTHORITY\SYSTEM	C:\program.exe	C:\Program Files\Basic Monitoring\Automate-Basic-Monitoring.exe
2023-05-10 04:57:36	CLIENT02	NT AUTHORITY\SYSTEM	C:\program.exe	C:\Program Files\Basic Monitoring\Automate-Basic-Monitoring.exe

In the results above, the Image field does not accurately reflect the full path to the executable, indicating a potential unquoted service path vulnerability. The executed process is “C:\Program Files\Basic Monitoring\Automate-Basic-Monitoring.exe” (i.e. the vulnerable service), but it is running with the image name of “program.exe”, which suggests an exploitation attempt.

The attacker used Unquoted Service Path to escalate privileges. the name of the vulnerable service : **Automate-Basic-Monitoring.exe**

we can check the “Hashes” field for the SHA256 of the “program.exe” file.

SHA256 of an executable file that elevates the attacker's privileges : SHA256 = **8ACC5D98CFFE8FE7D85DE218971B18D49166922D079676729055939463555BD2**



```
(cmdline_process), child_process=lower(child_process) | where cmdline_process!=child_process  | search Image="C:\\program.exe"
```

✓ 2 events (5/10/23 4:57:36.000 AM to 5/10/23 4:57:36.001 AM)   No Event Sampling

Events (2)   Patterns   Statistics   Visualization

1 millisecond per column

SELECTED FIELDS: a CommandLine 1, a CurrentDirectory 1, # EventCode 1, a host 1, a Image 1, a User 1

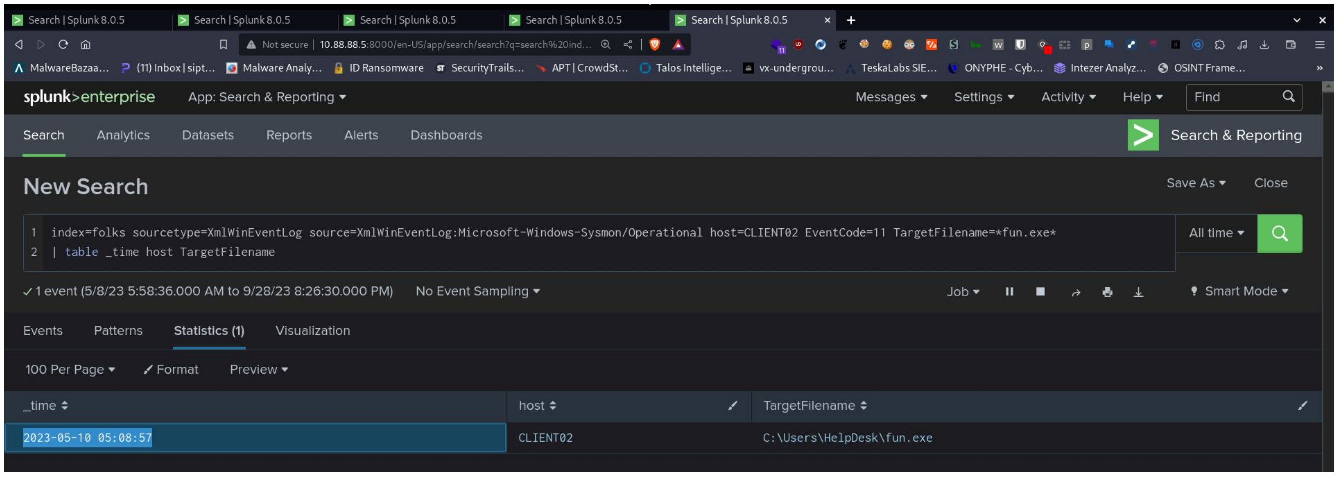
INTERESTING FIELDS: a Channel 1, a child_process 1, a cmdline_process 1, a Company 1, a Computer 1, a Description 1, a dvc 1, a dvc_nt_host 1, # event_id 2, a EventData_Xml 2

_time	host	Image	CurrentDirectory	EventCode	CommandLine	User
5/10/23 4:57:36.000 AM	CLIENT02	C:\program.exe	C:\Windows\system32\	1	C:\Program Files\Basic Monitoring\Automate-Basic-Monitoring.exe	NT AUTHORITY\SYSTEM

```
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider Name='Microsoft-Windows-Sysmon' Guid='{5770385f-c22a-43e0-
bf4c-06f5698ffbd9}'/><EventID>1</EventID><Version>5</Version><Level>4</Level><Task>1</Task><Opcode>0</Opcode><Keywords>0x8000000000000000</Keywor
ds><TimeCreated SystemTime='2023-05-10T04:57:36.5376684Z'/><EventRecordID>14271</EventRecordID><Correlation/><Execution ProcessID='2512' ThreadID
='4368'/><Channel>Microsoft-Windows-Sysmon/Operational</Channel><Computer>Client02.Abdullah.Ali.Alhakami</Computer><Security UserID='S-1-5-18'/>
</System><EventData><Data Name='RuleName'>-</Data><Data Name='UtcTime'>2023-05-10 04:57:08.862</Data><Data Name='ProcessGuid'>{e7c1085c-2424-645b
-3600-000000000c00}</Data><Data Name='ProcessId'>2416</Data><Data Name='Image'>C:\program.exe</Data><Data Name='FileVersion'>-</Data><Data Name
='Description'>-</Data><Data Name='Product'>-</Data><Data Name='Company'>-</Data><Data Name='OriginalFileName'>-</Data><Data Name='CommandLine'>
C:\Program Files\Basic Monitoring\Automate-Basic-Monitoring.exe</Data><Data Name='CurrentDirectory'>C:\Windows\system32\</Data><Data Name='User'>
NT AUTHORITY\SYSTEM</Data><Data Name='LogonGuid'>{e7c1085c-241a-645b-e703-000000000000}</Data><Data Name='LogonId'>0x3e7</Data><Data Name='Termin
alSessionId'>0</Data><Data Name='IntegrityLevel'>System</Data><Data Name='Hashes'>MD5=9169B8AAE13C43809D9DE8FED5C05202,SHA256=8ACC5D98CFFE8FE7D85
DE218971B18D49166922D079676729055939463555BD2,IMPHASH=B093B9B6223AF7F9E72D34D8765AA77F</Data><Data Name='ParentProcessGuid'>{e7c1085c-2419-645b-0
b00-000000000c00}</Data><Data Name='ParentProcessId'>732</Data><Data Name='ParentImage'>C:\Windows\System32\services.exe</Data><Data Name='Parent
CommandLine'>C:\Windows\system32\services.exe</Data><Data Name='ParentUser'>NT AUTHORITY\SYSTEM</Data></EventData></Event>
```

In Sysmon, we can use event code 11 to return file create operations and filter for the file name “fun.exe”.

Splunk Search :
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=11 TargetFilename=*fun.exe*
| table _time host TargetFilename

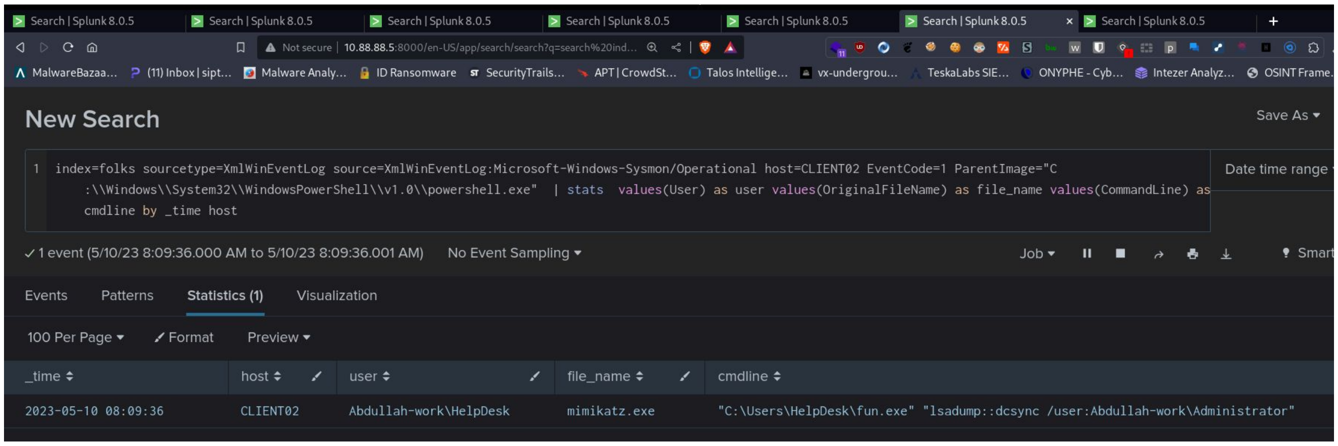


```
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=11 TargetFilename=*fun.exe*
| table _time host TargetFilename
```

✓ 1 event (5/8/23 5:58:36.000 AM to 9/28/23 8:26:30.000 PM)

_time	host	TargetFilename
2023-05-10 05:08:57	CLIENT02	C:\Users\HelpDesk\fun.exe

The attacker downloaded fun.exe? (24HOUR-UTC) : **2023-05-10 05:08:57**



```
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=1 ParentImage="C
    :\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"  | stats  values(User) as user values(OriginalFileName) as file_name values(CommandLine) as
    cmdline by _time host
```

✓ 1 event (5/10/23 8:09:36.000 AM to 5/10/23 8:09:36.001 AM)

_time	host	user	file_name	cmdline
2023-05-10 08:09:36	CLIENT02	Abdullah-work\HelpDesk	mimikatz.exe	"C:\Users\HelpDesk\fun.exe" "lsadump::dcsync /user:Abdullah-work\Administrator"

Command line used to launch DCSync attacks : **"C:\Users\HelpDesk\fun.exe" "lsadump::dcsync /user:Abdullah-work\Administrator"**

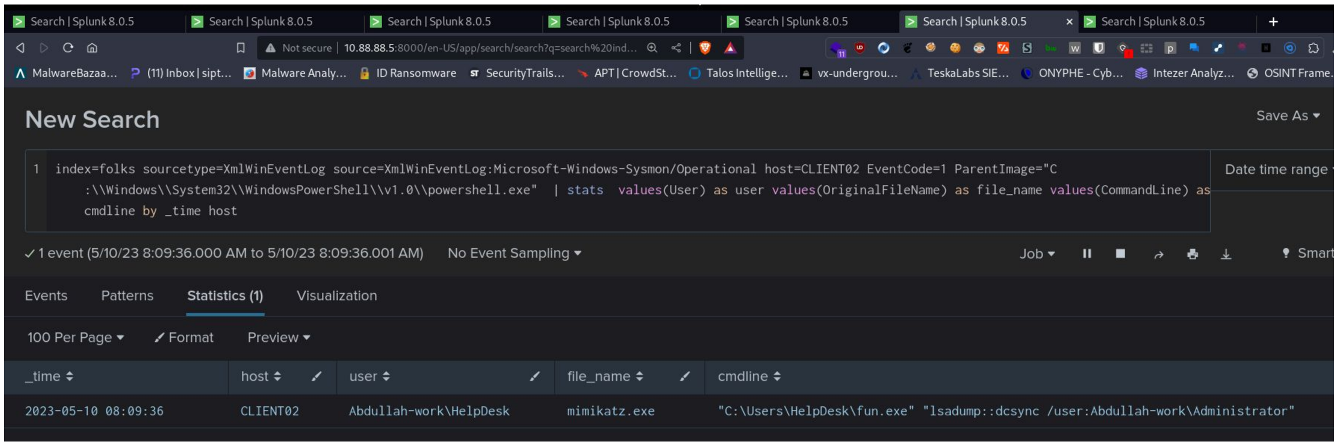
A DCSync attack is a method used to extract password hashes of user accounts from an Active Directory domain controller (**MITRE T1003.006**). The attacker impersonates a domain controller, requests replication data, specifically targeting user account objects, and retrieves the password hashes. With the hashes, the attacker can attempt to crack or decrypt them to obtain the users' passwords and gain unauthorized access to the network resources.

OS Credential Dumping: DCSync : https://attack.mitre.org/techniques/T1003/006/

To look for PowerShell command line activity.

Search Splunk :
index=folks sourcetype=XmlWinEventLog
source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=1
ParentImage="C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"
| stats values(User) as user values(OriginalFileName) as file_name values(CommandLine) as cmdline by _time host

If we scroll down to the last event, we can see that the attacker used "mimikatz.exe", which is the original name for "fun.exe" to perform a DCSync attack.


Search | Splunk 8.0.5
Not secure | 10.88.88.5:8000/en-US/app/search/search?q=search%20ind...
MalwareBazaa... (11) Inbox | sipt... Malware Analy... ID Ransomware SecurityTrails... APT | CrowdSt... Talos Intellige... vx-undergrou... TeskaLabs SIE... ONYPHE - Cyb... Intezer Analyz... OSINT Frame...
New Search
Save As
Date time range
Job
Smart
Events
Patterns
Statistics (1)
Visualization
100 Per Page
Format
Preview

```
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=1 ParentImage="C
    :\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"  | stats  values(User) as user values(OriginalFileName) as file_name values(CommandLine) as
    cmdline by _time host
```

✓ 1 event (5/10/23 8:09:36.000 AM to 5/10/23 8:09:36.001 AM)   No Event Sampling

_time	host	user	file_name	cmdline
2023-05-10 08:09:36	CLIENT02	Abdullah-work\HelpDesk	mimikatz.exe	"C:\Users\HelpDesk\fun.exe" "lsadump::dcsync /user:Abdullah-work\Administrator"

The original name of fun.exe : **mimikatz.exe**

OverPass-the-Hash (PtH) occurs when an attacker uses a password hash to authenticate as a user but also uses the password hash to create a valid Kerberos ticket (**MITRE T1550.002**). Looking at the results returned by the previously used SPL query, I can see that the attacker used "Rubeus.exe".

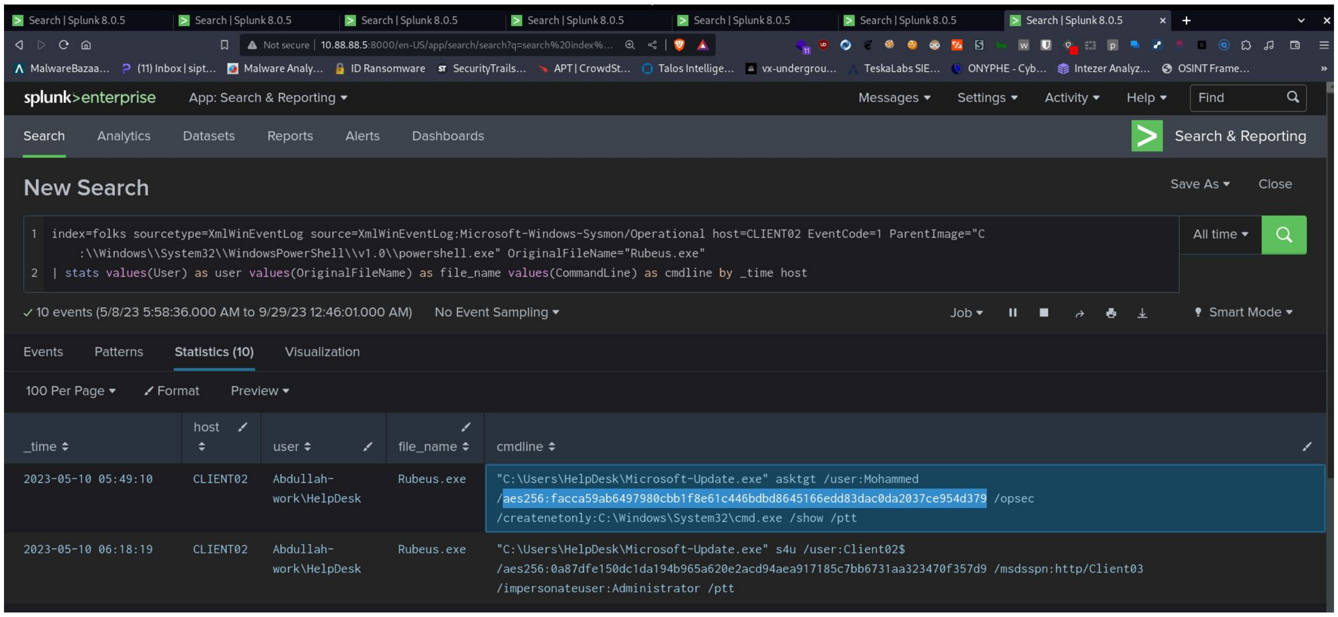
Use Alternate Authentication Material: Pass the Hash :
https://attack.mitre.org/techniques/T1550/002/

Search Splunk :
index=folks sourcetype=XmlWinEventLog
source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=1
ParentImage="C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"
| stats values(User) as user values(OriginalFileName) as file_name values(CommandLine) as cmdline by _time host

Rubeus is a tool that can be used to perform an Overpass the Hash attack (HackTricks). Using the query below, we can filter for "Rubeus.exe" and observe the AES256 hash of the account that was attacked.

Source Tools > https://book.hacktricks.xyz/windows-hardening/active-directory-methodology/over-pass-the-hash-pass-the-key
Source tools > https://github.com/GhostPack/Rubeus

Search Splunk :
index=folks sourcetype=XmlWinEventLog
source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=1
ParentImage="C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"
OriginalFileName="Rubeus.exe"
| stats values(User) as user values(OriginalFileName) as file_name values(CommandLine) as cmdline by _time host



```
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=1 ParentImage="C
    :\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe" OriginalFileName="Rubeus.exe"
| stats values(User) as user values(OriginalFileName) as file_name values(CommandLine) as cmdline by _time host
```

✓ 10 events (5/8/23 5:58:36.000 AM to 9/29/23 12:46:01.000 AM)   No Event Sampling   Statistics (10)

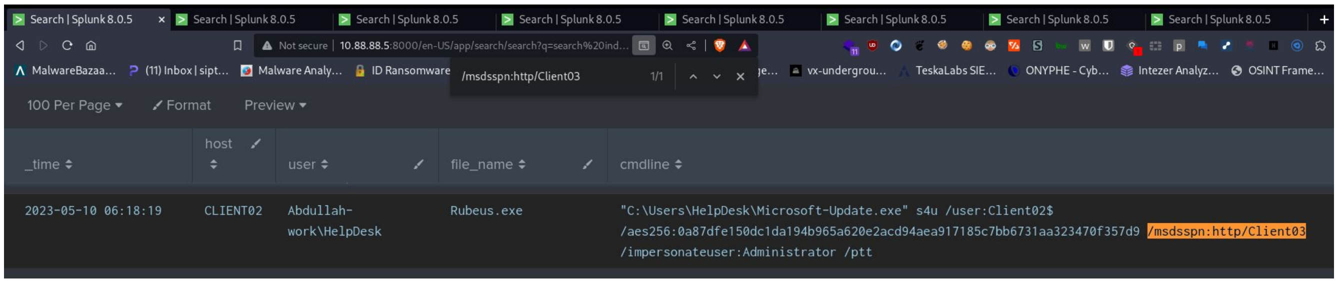
_time	host	user	file_name	cmdline
2023-05-10 05:49:10	CLIENT02	Abdullah-work\HelpDesk	Rubeus.exe	"C:\Users\HelpDesk\Microsoft-Update.exe" asktgt /user:Mohammed /aes256:facca59ab6497980cbb1f8e61c446bdbd8645166edd83dac0da2037ce954d379 /opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt
2023-05-10 06:18:19	CLIENT02	Abdullah-work\HelpDesk	Rubeus.exe	"C:\Users\HelpDesk\Microsoft-Update.exe" s4u /user:Client02$ /aes256:0a87dfe150dc1da194b965a620e2acd94aea917185c7bb6731aa323470f357d9 /msdsspn:http/Client03 /impersonateuser:Administrator /ptt

The attacker performed the Over-Pass-The-Hash technique. the AES 256 hash of the account he attacked : **facca59ab6497980cbb1f8e61c446bdbd8645166edd83dac0da2037ce954d379**

Once again, we can use the query from in up PowerShell command line activity and look through the events returned.

Search Splunk :
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=1 ParentImage="C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"
| stats values(User) as user values(OriginalFileName) as file_name values(CommandLine) as cmdline by _time host

I spotted the host "CLIENT03" being referenced in command line activity related to Rubeus.



/msdsspn:http/Client03 1/1

_time	host	user	file_name	cmdline
2023-05-10 06:18:19	CLIENT02	Abdullah-work\HelpDesk	Rubeus.exe	"C:\Users\HelpDesk\Microsoft-Update.exe" s4u /user:Client02$ /aes256:0a87dfe150dc1da194b965a620e2acd94aea917185c7bb6731aa323470f357d9 /msdsspn:http/Client03 /impersonateuser:Administrator /ptt

service did the attacker abuse to access the Client03 machine as Administrator : **http/Client03**

It appears the attacker leveraged Rubeus to perform a constrained delegation attack. Constrained delegation is a legitimate feature in Active Directory that allows a service to impersonate a user and access resources on their behalf. However, if not properly configured, it can be abused by attackers to gain unauthorized access to sensitive resources. Based on the command structure, we can see that the attacker abused the "Http/Client03" service.

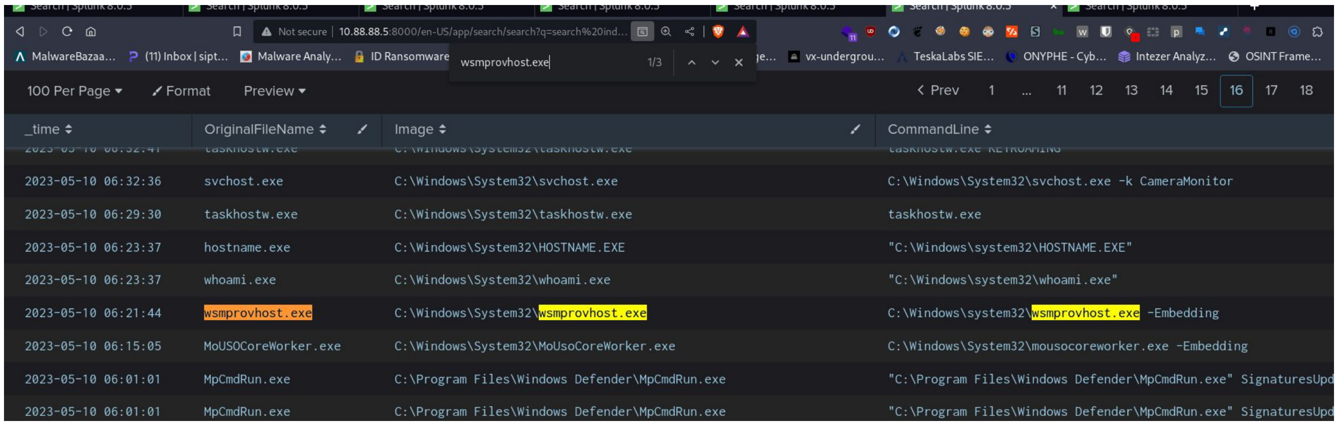
Attacking Kerberos Constrained Delegation : https://medium.com/r3d-buck3t/attacking-kerberos-constrained-delegations-4a0eddc5bb13

.\Rubeus.exe s4u /ticket:TGT_Ticket /msdsspn:"service/HOST" /impersonateuser:Administrator /ptt

we can take note of when the threat actor performed the constrained delegation attack to gain access to the host “CLIENT03” (i.e. 2023–05–10 06:18:19). Next, we can run the query below to check for what processes were spawned after the attacker logged in and set the time range to after “2023–05–10 06:18:19”.

Search Splunk :
index=folks sourcetype=XmlWinEventLog
source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational EventCode=1 host=CLIENT03
NOT CommandLine IN (*splunk*)
| table _time OriginalFileName Image CommandLine

I can see that the process “wsmprovhost.exe” was spawned at “2023–05–10 06:21:44”, which is an executable file that is associated with the Windows Remote Management (WinRM) service.

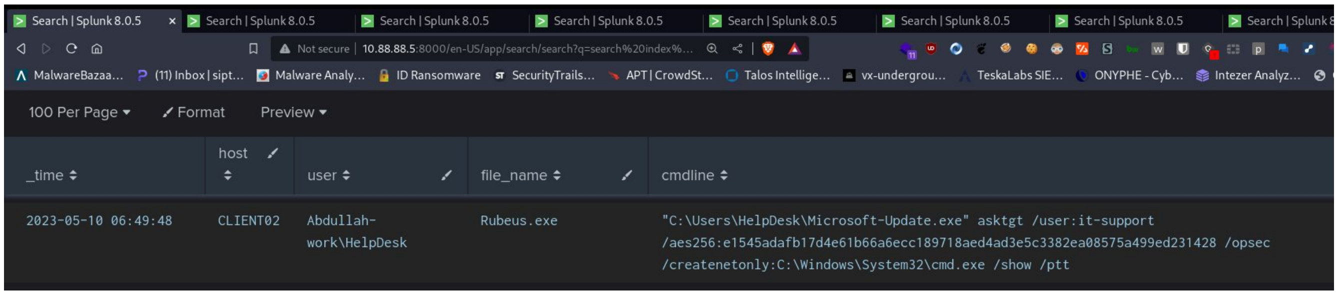


_time	OriginalFileName	Image	CommandLine
2023-05-10 06:32:36	svchost.exe	C:\Windows\System32\svchost.exe	C:\Windows\System32\svchost.exe -k CameraMonitor
2023-05-10 06:29:30	taskhostw.exe	C:\Windows\System32\taskhostw.exe	taskhostw.exe
2023-05-10 06:23:37	hostname.exe	C:\Windows\System32\HOSTNAME.EXE	"C:\Windows\system32\HOSTNAME.EXE"
2023-05-10 06:23:37	whoami.exe	C:\Windows\System32\whoami.exe	"C:\Windows\system32\whoami.exe"
2023-05-10 06:21:44	wsmprovhost.exe	C:\Windows\System32\wsmprovhost.exe	C:\Windows\system32\wsmprovhost.exe -Embedding
2023-05-10 06:15:05	MoUSOCoreWorker.exe	C:\Windows\System32\MoUsoCoreWorker.exe	C:\Windows\System32\mousocoreworker.exe -Embedding
2023-05-10 06:01:01	MpCmdRun.exe	C:\Program Files\Windows Defender\MpCmdRun.exe	"C:\Program Files\Windows Defender\MpCmdRun.exe" SignaturesUpd
2023-05-10 06:01:01	MpCmdRun.exe	C:\Program Files\Windows Defender\MpCmdRun.exe	"C:\Program Files\Windows Defender\MpCmdRun.exe" SignaturesUpd

The Client03 machine spawned a new process when the attacker logged on remotely. The process name : **wsmprovhost.exe**

For PowerShell command line activity and filtering for “it-support”, we can see that the account was compromised via Over-Pass-The-Hash-Attack on Client02 at “2023–05–10 06:49:48”.

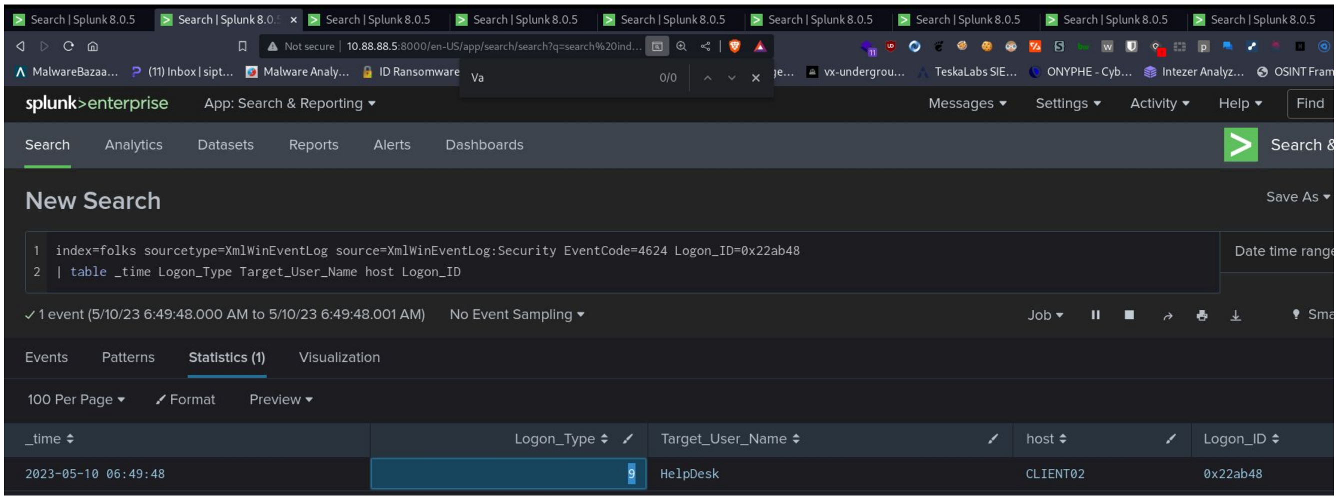
Search Splunk :
index=folks sourcetype=XmlWinEventLog
source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=1
ParentImage="C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"
| stats values(User) as user values(OriginalFileName) as file_name values(CommandLine) as cmdline by _time host



_time	host	user	file_name	cmdline
2023-05-10 06:49:48	CLIENT02	Abdullah-work\HelpDesk	Rubeus.exe	"C:\Users\HelpDesk\Microsoft-Update.exe" asktgt /user:it-support /aes256:e1545adafb17d4e61b66a6ecc189718aed4ad3e5c3382ea08575a499ed231428 /opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt

A successful pass-the-hash operation will also generate event 4624, which has the login type 9 as its logon. The executing user's Logon ID is recorded in this event, so we can cross-reference this event with the event that recorded the process creation.
Checking the "LogonId" field for this event, I can retrieve the event and then create a new SPL query that searches for 4624 events that contain the Logon ID "0x22ab48".

Search Splunk :
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Security EventCode=4624 Logon_ID=0x22ab48
| table _time Logon_Type Target_User_Name host Logon_ID



```
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Security EventCode=4624 Logon_ID=0x22ab48
| table _time Logon_Type Target_User_Name host Logon_ID
```

✓1 event (5/10/23 6:49:48.000 AM to 5/10/23 6:49:48.001 AM)   No Event Sampling

_time	Logon_Type	Target_User_Name	host	Logon_ID
2023-05-10 06:49:48	9	HelpDesk	CLIENT02	0x22ab48

The attacker compromises the it-support account. the logon type : **9**

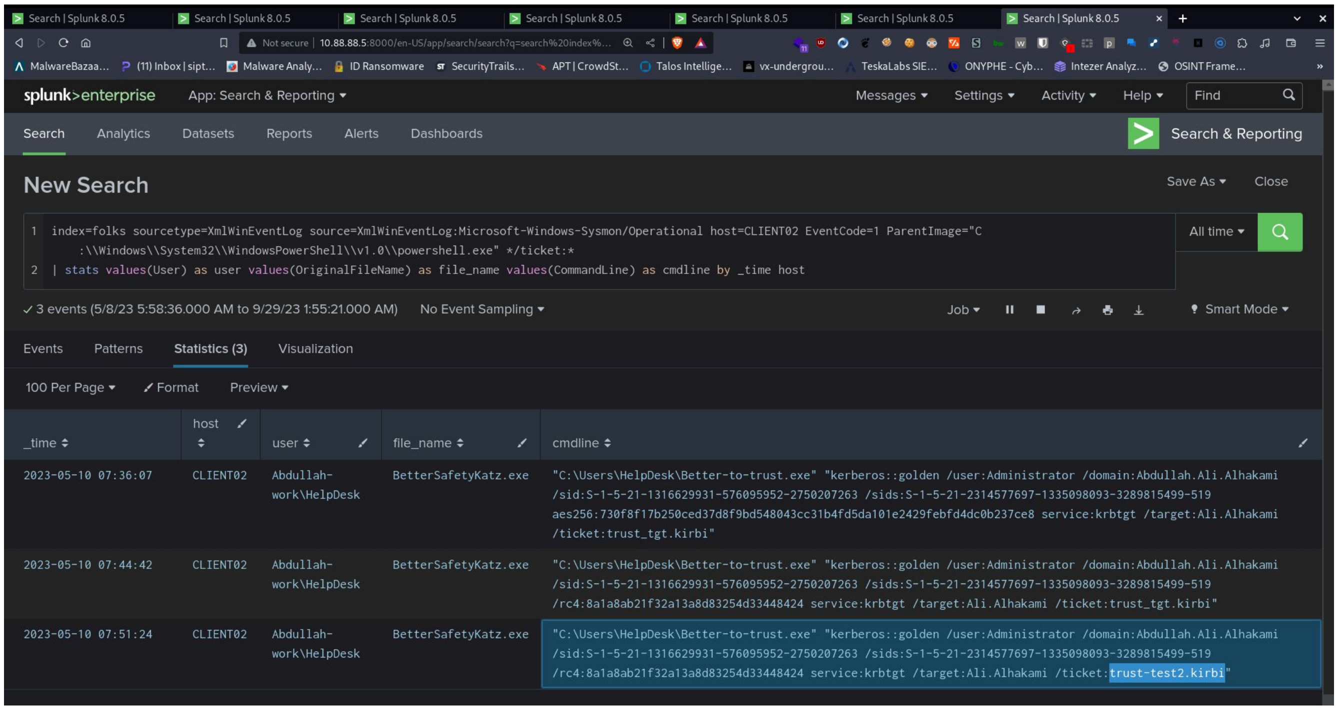
Looking back at results for the query from in up for PowerShell command line activity, we can see that the threat actor is using mimikatz to create golden tickets.

Mimikatz : https://book.hacktricks.xyz/windows-hardening/stealing-credentials/credentials-mimikatz

A Golden Ticket is a TGT using the KRBTGT NTLM password hash to encrypt and sign.
A Golden Ticket (GT) can be created to impersonate any user (real or imagined) in the domain as a member of any group in the domain (providing a virtually unlimited amount of rights) to any and every resource in the domain.

The Mimikatz command to create a golden ticket includes the parameter “/ticket (optional)”, which provides a path and name for saving the Golden Ticket file to for later use. The attacker could also use /ptt to immediately inject the golden ticket into memory. The SPL query below is used to filter on this parameter and checking through the results, I can see the name of the ticket.

Search Splunk :
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=1 ParentImage="C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe" */ticket:*
| stats values(User) as user values(OriginalFileName) as file_name values(CommandLine) as cmdline by _time host



```
index=folks sourcetype=XmlWinEventLog source=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational host=CLIENT02 EventCode=1 ParentImage="C
    :\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe" */ticket:*
| stats values(User) as user values(OriginalFileName) as file_name values(CommandLine) as cmdline by _time host
```

✓ 3 events (5/8/23 5:58:36.000 AM to 9/29/23 1:55:21.000 AM)

_time	host	user	file_name	cmdline
2023-05-10 07:36:07	CLIENT02	Abdullah-work\HelpDesk	BetterSafetyKatz.exe	"C:\Users\HelpDesk\Better-to-trust.exe" "kerberos::golden /user:Administrator /domain:Abdullah.Ali.Alhakami /sid:S-1-5-21-1316629931-576095952-2750207263 /sids:S-1-5-21-2314577697-1335098093-3289815499-519 aes256:730f8f17b250ced37d8f9bd548043cc31b4fd5da101e2429febfd4dc0b237ce8 service:krbtgt /target:Ali.Alhakami /ticket:trust_tgt.kirbi"
2023-05-10 07:44:42	CLIENT02	Abdullah-work\HelpDesk	BetterSafetyKatz.exe	"C:\Users\HelpDesk\Better-to-trust.exe" "kerberos::golden /user:Administrator /domain:Abdullah.Ali.Alhakami /sid:S-1-5-21-1316629931-576095952-2750207263 /sids:S-1-5-21-2314577697-1335098093-3289815499-519 /rc4:8a1a8ab21f32a13a8d83254d33448424 service:krbtgt /target:Ali.Alhakami /ticket:trust_tgt.kirbi"
2023-05-10 07:51:24	CLIENT02	Abdullah-work\HelpDesk	BetterSafetyKatz.exe	"C:\Users\HelpDesk\Better-to-trust.exe" "kerberos::golden /user:Administrator /domain:Abdullah.Ali.Alhakami /sid:S-1-5-21-1316629931-576095952-2750207263 /sids:S-1-5-21-2314577697-1335098093-3289815499-519 /rc4:8a1a8ab21f32a13a8d83254d33448424 service:krbtgt /target:Ali.Alhakami /ticket:trust-test2.kirbi"

What ticket name did the attacker generate to access the parent DC as Administrator : **trust-test2.kirbi**

Closing Remarks:
I really enjoyed working through this challenge and getting the opportunity to learn more about investigating incidents using Splunk. The challenge provides opportunities to learn about different log sources (e.g. Sysmon, Windows Events, etc.), different adversary techniques and threat hunting with Splunk. Thank you for reading till the end.

Regards,

M Arif