Detecting Common User/Domain Recon

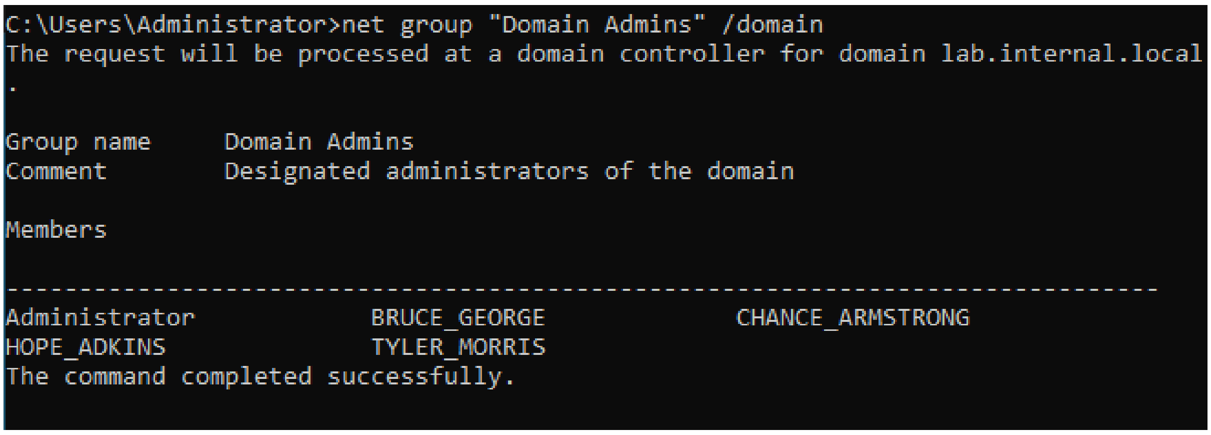
Domain Reconnaissance

Active Directory (AD) domain reconnaissance represents a pivotal stage in the cyberattack lifecycle. During this phase, adversaries endeavor to gather information about the target environment, seeking to comprehend its architecture, network topology, security measures, and potential vulnerabilities.

While conducting AD domain reconnaissance, attackers focus on identifying crucial components such as Domain Controllers, user accounts, groups, trust relationships, organizational units (OUs), group policies, and other vital objects. By gaining insights into the AD environment, attackers can potentially pinpoint high-value targets, escalate their privileges, and move laterally within the network.

User/Domain Reconnaissance Using Native Windows Executables

An example of AD domain reconnaissance is when an adversary executes the net group command to obtain a list of Domain Administrators.



```
C:\Users\Administrator>net group "Domain Admins" /domain
The request will be processed at a domain controller for domain lab.internal.local
.

Group name     Domain Admins
Comment        Designated administrators of the domain

Members

-------------------------------------------------------------------------------
Administrator            BRUCE_GEORGE             CHANCE_ARMSTRONG
HOPE_ADKINS              TYLER_MORRIS
The command completed successfully.
```

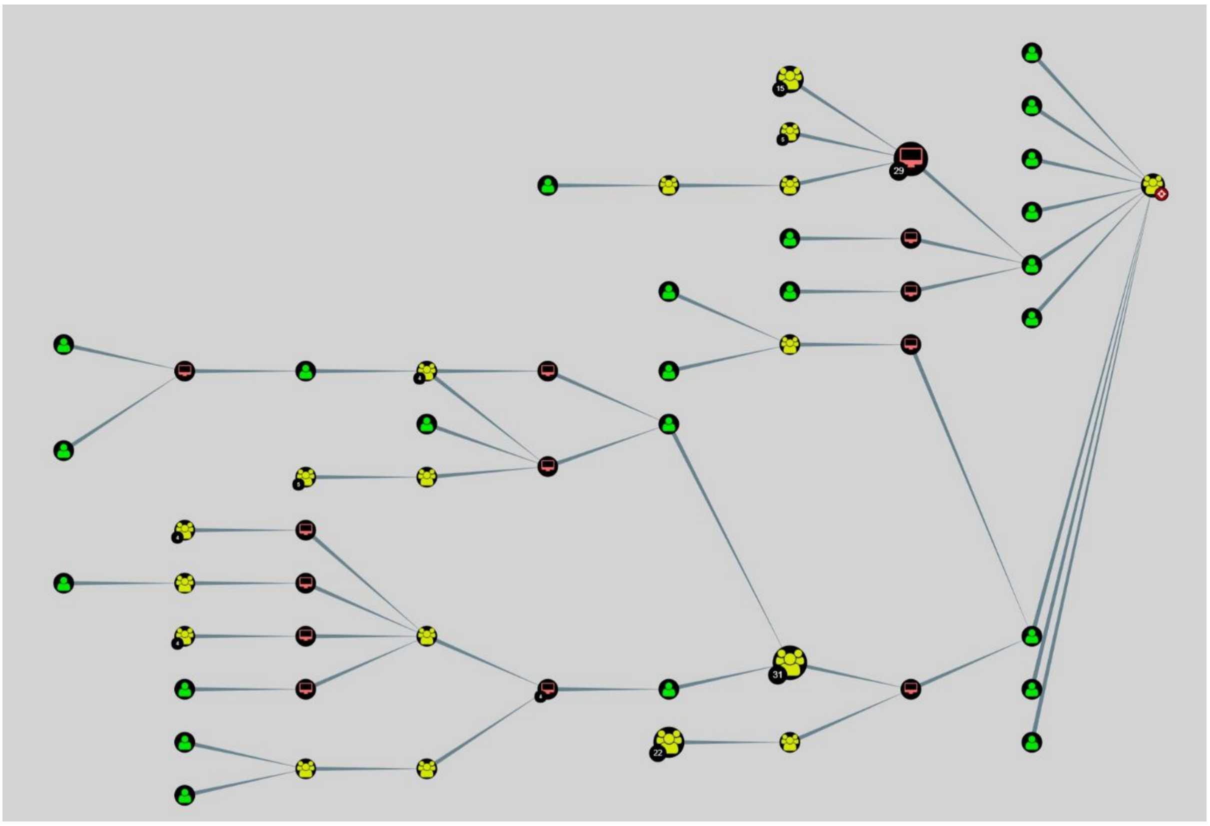
Common native tools/commands utilized for domain reconnaissance include:

- whoami /all
- wmic computersystem get domain
- net user /domain
- net group "Domain Admins" /domain
- arp -a
- nltest /domain_trusts

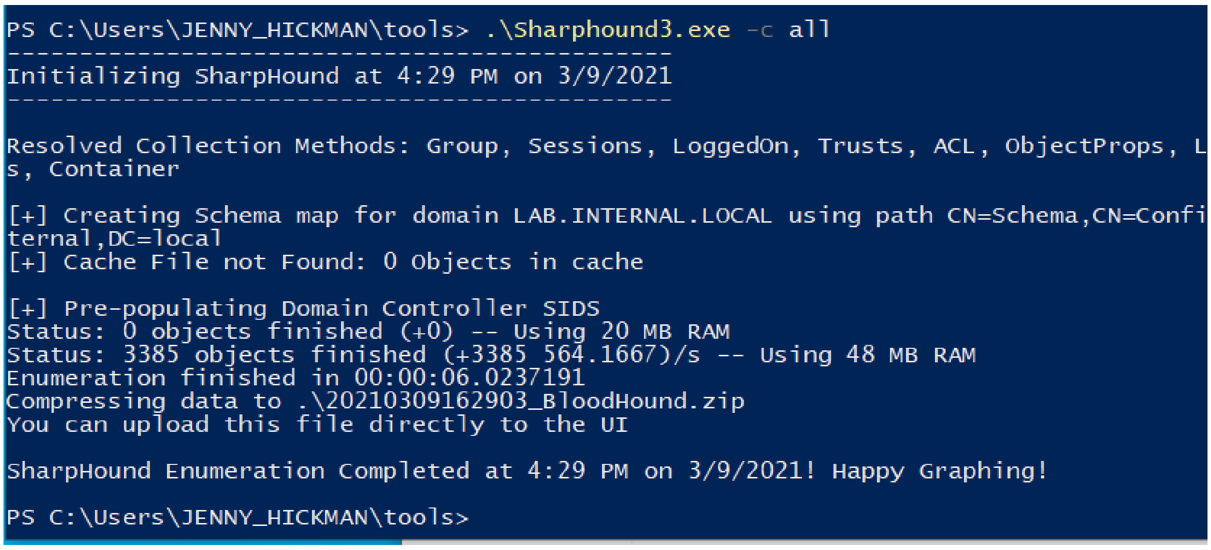
For detection, administrators can employ PowerShell to monitor for unusual scripts or cmdlets and process command-line monitoring.

User/Domain Reconnaissance Using BloodHound/SharpHound

BloodHound is an open-source domain reconnaissance tool created to analyze and visualize the Active Directory (AD) environment. It is frequently employed by attackers to discern attack paths and potential security risks within an organization's AD infrastructure. BloodHound leverages graph theory and relationship mapping to elucidate trust relationships, permissions, and group memberships within the AD domain.



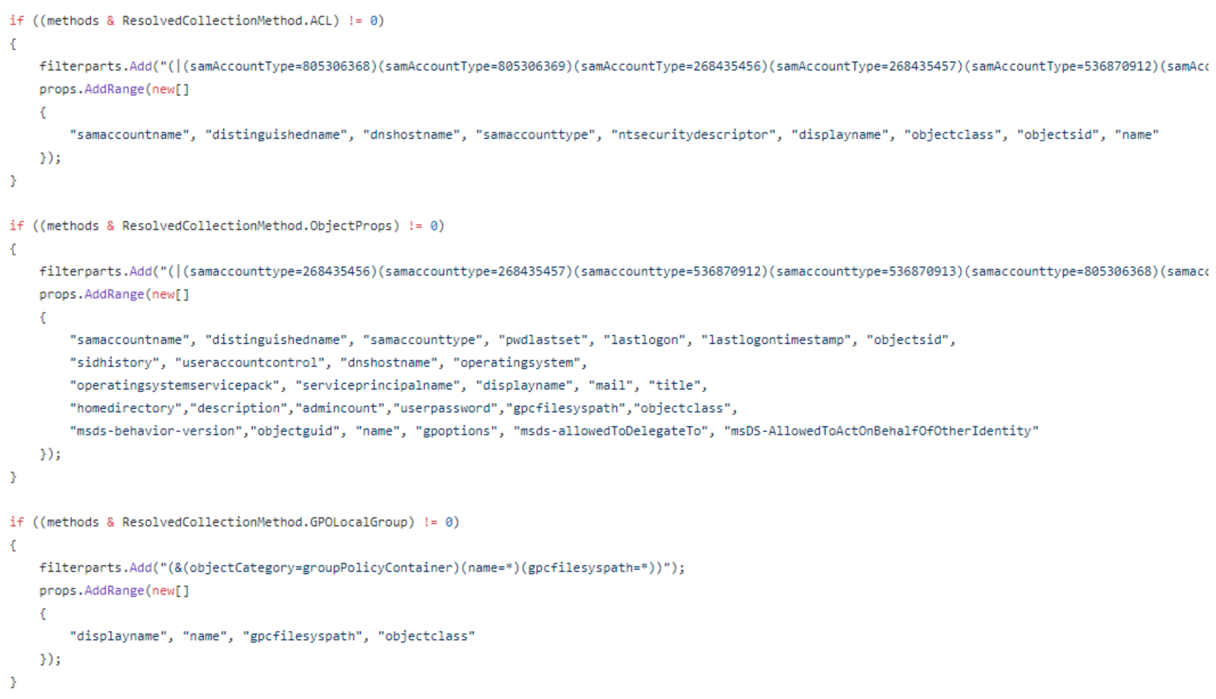
Sharphound is a C# data collector for BloodHound. An example of usage includes an adversary running Sharphound with all collection methods (-c all).



```
PS C:\Users\JENNY_HICKMAN\tools> .\Sharphound3.exe -c all
-----------------------------------------------
Initializing SharpHound at 4:29 PM on 3/9/2021
-----------------------------------------------

Resolved Collection Methods: Group, Sessions, LoggedOn, Trusts, ACL, ObjectProps, L
s, Container

[+] Creating Schema map for domain LAB.INTERNAL.LOCAL using path CN=Schema,CN=Confi
ternal,DC=local
[+] Cache File not Found: 0 Objects in cache

[+] Pre-populating Domain Controller SIDS
Status: 0 objects finished (+0) -- Using 20 MB RAM
Status: 3385 objects finished (+3385 564.1667)/s -- Using 48 MB RAM
Enumeration finished in 00:00:06.0237191
Compressing data to .\20210309162903_BloodHound.zip
You can upload this file directly to the UI

SharpHound Enumeration Completed at 4:29 PM on 3/9/2021! Happy Graphing!

PS C:\Users\JENNY_HICKMAN\tools>
```

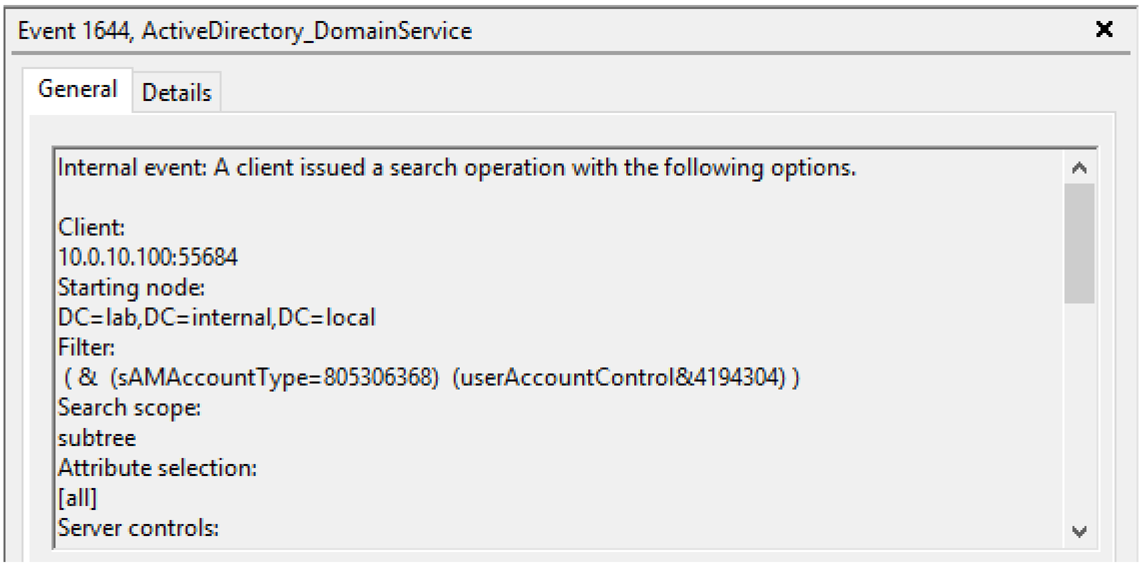
BloodHound Detection Opportunities

Under the hood, the BloodHound collector executes numerous LDAP queries directed at the Domain Controller, aiming to amass information about the domain.



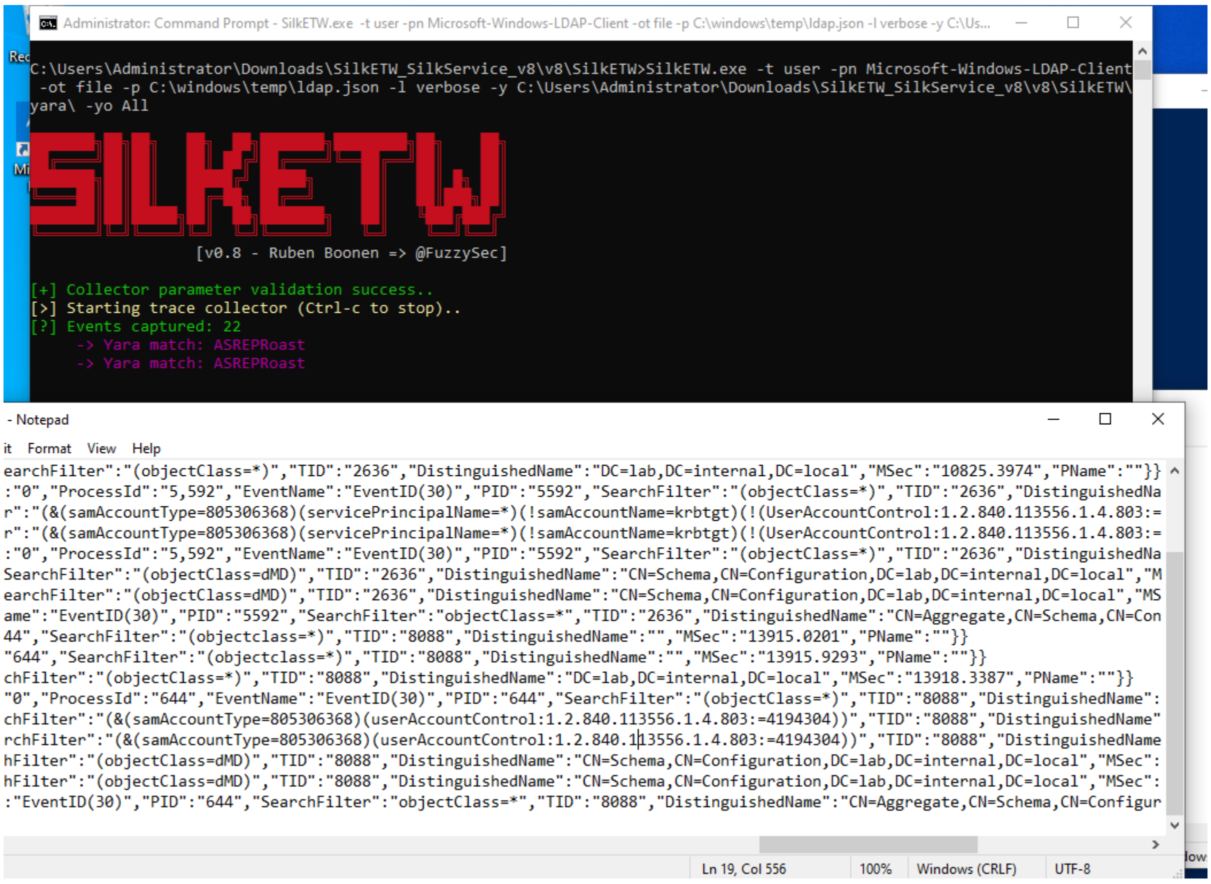
```
if ((methods & ResolvedCollectionMethod.ACL) != 0)
{
    filterparts.Add("(|(samAccountType=805306368)(samAccountType=805306369)(samAccountType=268435456)(samAccountType=268435457)(samAccountType=536870912)(samAcc
    props.AddRange(new[]
    {
        "samaccountname", "distinguishedname", "dnshostname", "samaccounttype", "ntsecuritydescriptor", "displayname", "objectclass", "objectsid", "name"
    });
}

if ((methods & ResolvedCollectionMethod.ObjectProps) != 0)
{
    filterparts.Add("(|(samaccounttype=268435456)(samaccounttype=268435457)(samaccounttype=536870912)(samaccounttype=536870913)(samaccounttype=805306368)(samacc
    props.AddRange(new[]
    {
        "samaccountname", "distinguishedname", "samaccounttype", "pwdlastset", "lastlogon", "lastlogontimestamp", "objectsid",
        "sidhistory", "useraccountcontrol", "dnshostname", "operatingsystem",
        "operatingsystemservicepack", "serviceprincipalname", "displayname", "mail", "title",
        "homedirectory","description","admincount","userpassword","gpcfilesyspath","objectclass",
        "msds-behavior-version","objectguid", "name", "gpoptions", "msds-allowedToDelegateTo", "msDS-AllowedToActOnBehalfOfOtherIdentity"
    });
}

if ((methods & ResolvedCollectionMethod.GPOLocalGroup) != 0)
{
    filterparts.Add("(&(objectCategory=groupPolicyContainer)(name=*)(gpcfilesyspath=*))");
    props.AddRange(new[]
    {
        "displayname", "name", "gpcfilesyspath", "objectclass"
    });
}
```

However, monitoring LDAP queries can be a challenge. By default, the Windows Event Log does not record them. The best option Windows can suggest is employing Event 1644 - the LDAP performance monitoring log. Even with it enabled, BloodHound may not generate many of the expected events.



```
Internal event: A client issued a search operation with the following options.

Client:
10.0.10.100:55684
Starting node:
DC=lab,DC=internal,DC=local
Filter:
 ( & (sAMAccountType=805306368) (userAccountControl&4194304) )
Search scope:
subtree
Attribute selection:
[all]
Server controls:
```

A more reliable approach is to utilize the Windows ETW provider Microsoft-Windows-LDAP-Client. As showcased previously in the SOC Analyst path, [SilkETW & SilkService] are versatile C# wrappers for ETW, designed to simplify the intricacies of ETW, providing an accessible interface for research and introspection. SilkService supports output to the Windows Event Log, which streamlines log digestion. Another useful feature is the ability to employ Yara rules for hunting suspicious LDAP queries.


Administrator: Command Prompt - SilkETW.exe -t user -pn Microsoft-Windows-LDAP-Client -ot file -p C:\windows\temp\ldap.json -l verbose -y C:\Us...
SILKETW
- Notepad
it Format View Help
Ln 19, Col 556
100%
Windows (CRLF)
UTF-8

```
C:\Users\Administrator\Downloads\SilkETW_SilkService_v8\v8\SilkETW>SilkETW.exe -t user -pn Microsoft-Windows-LDAP-Client
 -ot file -p C:\windows\temp\ldap.json -l verbose -y C:\Users\Administrator\Downloads\SilkETW_SilkService_v8\v8\SilkETW\
yara\ -yo All

                  [v0.8 - Ruben Boonen => @FuzzySec]

[+] Collector parameter validation success..
[>] Starting trace collector (Ctrl-c to stop)..
[?] Events captured: 22
     -> Yara match: ASREPRoast
     -> Yara match: ASREPRoast
```

```
earchFilter":"(objectClass=*)","TID":"2636","DistinguishedName":"DC=lab,DC=internal,DC=local","MSec":"10825.3974","PName":""}}
:"0","ProcessId":"5,592","EventName":"EventID(30)","PID":"5592","SearchFilter":"(objectClass=*)","TID":"2636","DistinguishedNa
r":"(&(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)(!(UserAccountControl:1.2.840.113556.1.4.803:=
r":"(&(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)(!(UserAccountControl:1.2.840.113556.1.4.803:=
:"0","ProcessId":"5,592","EventName":"EventID(30)","PID":"5592","SearchFilter":"(objectClass=*)","TID":"2636","DistinguishedNa
SearchFilter":"(objectClass=dMD)","TID":"2636","DistinguishedName":"CN=Schema,CN=Configuration,DC=lab,DC=internal,DC=local","M
earchFilter":"(objectClass=dMD)","TID":"2636","DistinguishedName":"CN=Schema,CN=Configuration,DC=lab,DC=internal,DC=local","MS
ame":"EventID(30)","PID":"5592","SearchFilter":"objectClass=*","TID":"2636","DistinguishedName":"CN=Aggregate,CN=Schema,CN=Con
44","SearchFilter":"(objectclass=*)","TID":"8088","DistinguishedName":"","MSec":"13915.0201","PName":""}}
"644","SearchFilter":"(objectclass=*)","TID":"8088","DistinguishedName":"","MSec":"13915.9293","PName":""}}
chFilter":"(objectClass=*)","TID":"8088","DistinguishedName":"DC=lab,DC=internal,DC=local","MSec":"13918.3387","PName":""}}
"0","ProcessId":"644","EventName":"EventID(30)","PID":"644","SearchFilter":"(objectClass=*)","TID":"8088","DistinguishedName":
chFilter":"(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304))","TID":"8088","DistinguishedName"
rchFilter":"(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304))","TID":"8088","DistinguishedName
hFilter":"(objectClass=dMD)","TID":"8088","DistinguishedName":"CN=Schema,CN=Configuration,DC=lab,DC=internal,DC=local","MSec":
hFilter":"(objectClass=dMD)","TID":"8088","DistinguishedName":"CN=Schema,CN=Configuration,DC=lab,DC=internal,DC=local","MSec":
:"EventID(30)","PID":"644","SearchFilter":"objectClass=*","TID":"8088","DistinguishedName":"CN=Aggregate,CN=Schema,CN=Configur
```

In addition, Microsoft's ATP team has compiled a list of LDAP filters frequently used by reconnaissance tools.

Recon tool	**Filter**
enum_ad_user_comments (Metasploit)	(&(&(objectCategory=person)(objectClass=user))(|(description=*pass*)(comment=*pass*)))
enum_ad_computers (Metasploit)	(&(objectCategory=computer)(operatingSystem=*server*))
enum_ad_groups (Metasploit)	(&(objectClass=group))
enum_ad_managedby_groups (Metasploit)	(&(objectClass=group)(managedBy=*)), (&(objectClass=group)(managedBy=*)(groupType:1.2.840.113556.1.4.803:=2147483648))
Get-NetComputer (PowerView)	(&(sAMAccountType=805306369)(dnshostname=*))
Get-NetUser - Users (Powerview)	(&(samAccountType=805306368)(samAccountName=*)
Get-NetUser - SPNs (Powerview)	(&(samAccountType=805306368)(servicePrincipalName=*)
Get-DFSshareV2 (Powerview)	(&(objectClass=msDFS-Linkv2))
Get-NetOU (PowerView)	(&(objectCategory =organizationalUnit)(name=*))
Get-DomainSearcher (Empire)	(samAccountType=805306368)

Armed with this list of LDAP filters, BloodHound activity can be detected more efficiently.

Then, access the Splunk interface at http://[Target IP]:8000 and launch the Search & Reporting Splunk application. The vast majority of searches covered from this point up to end of this section can be replicated inside the target, offering a more comprehensive grasp of the topics presented.

Detecting User/Domain Recon With Splunk

You'll observe that a specific timeframe is given when identifying each attack. This is done to concentrate on the relevant events, avoiding the overwhelming volume of unrelated events.

Now let's explore how we can identify the recon techniques previously discussed, using Splunk.

Detecting Recon By Targeting Native Windows Executables

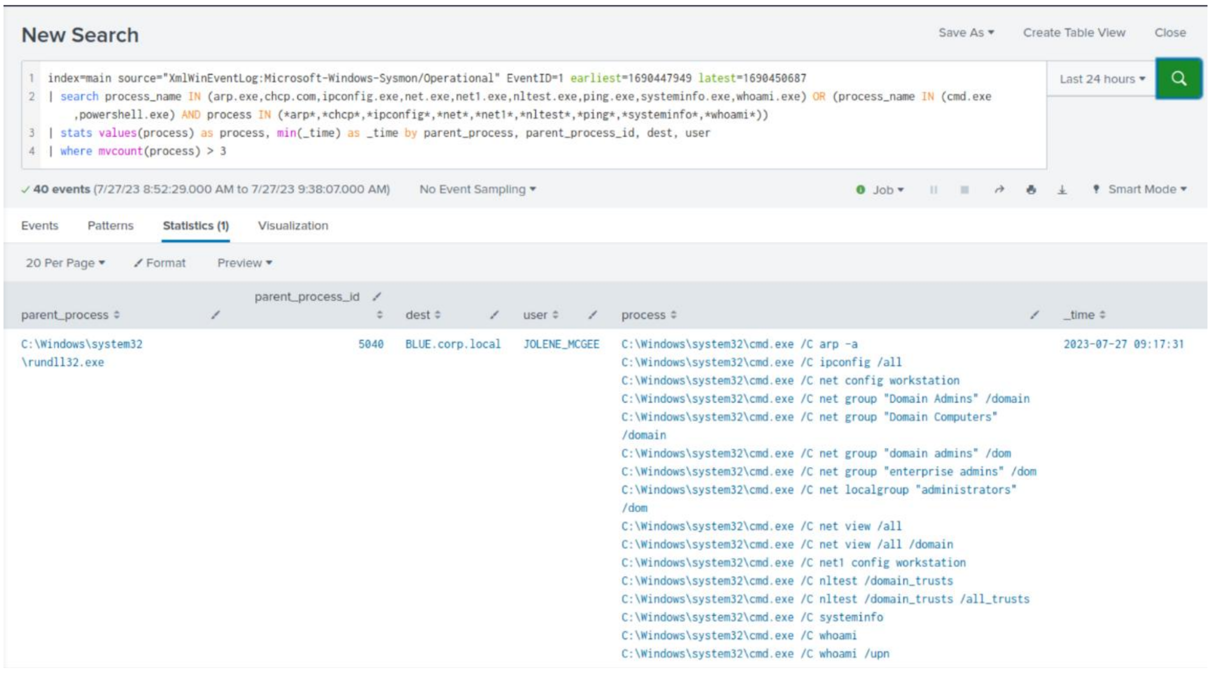
Timeframe: earliest=1690447949 latest=1690450687

index=main source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational" EventID=1 earliest=1690447949 latest=1690450687

| search process_name IN (arp.exe,chcp.com,ipconfig.exe,net.exe,net1.exe,nltest.exe,ping.exe,systeminfo.exe,whoami.exe) OR (process_name IN (cmd.exe,powershell.exe) AND process IN (*arp*,*chcp*,*ipconfig*,*net*,*net1*,*nltest*,*ping*,*systeminfo*,*whoami*))

| stats values(process) as process, min(_time) as _time by parent_process, parent_process_id, dest, user

| where mvcount(process) > 3



```
index=main source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational" EventID=1 earliest=1690447949 latest=1690450687
| search process_name IN (arp.exe,chcp.com,ipconfig.exe,net.exe,net1.exe,nltest.exe,ping.exe,systeminfo.exe,whoami.exe) OR (process_name IN (cmd.exe
    ,powershell.exe) AND process IN (*arp*,*chcp*,*ipconfig*,*net*,*net1*,*nltest*,*ping*,*systeminfo*,*whoami*))
| stats values(process) as process, min(_time) as _time by parent_process, parent_process_id, dest, user
| where mvcount(process) > 3
```

✓ 40 events (7/27/23 8:52:29.000 AM to 7/27/23 9:38:07.000 AM)

parent_process	parent_process_id	dest	user	process	_time
C:\Windows\system32\rundll32.exe	5040	BLUE.corp.local	JOLENE_MCGEE	C:\Windows\system32\cmd.exe /C arp -a C:\Windows\system32\cmd.exe /C ipconfig /all C:\Windows\system32\cmd.exe /C net config workstation C:\Windows\system32\cmd.exe /C net group "Domain Admins" /domain C:\Windows\system32\cmd.exe /C net group "Domain Computers" /domain C:\Windows\system32\cmd.exe /C net group "domain admins" /dom C:\Windows\system32\cmd.exe /C net group "enterprise admins" /dom C:\Windows\system32\cmd.exe /C net localgroup "administrators" /dom C:\Windows\system32\cmd.exe /C net view /all C:\Windows\system32\cmd.exe /C net view /all /domain C:\Windows\system32\cmd.exe /C net1 config workstation C:\Windows\system32\cmd.exe /C nltest /domain_trusts C:\Windows\system32\cmd.exe /C nltest /domain_trusts /all_trusts C:\Windows\system32\cmd.exe /C systeminfo C:\Windows\system32\cmd.exe /C whoami C:\Windows\system32\cmd.exe /C whoami /upn	2023-07-27 09:17:31

Search Breakdown:

- Filtering by Index and Source: The search begins by selecting events from the main index where the source is XmlWinEventLog:Microsoft-Windows-Sysmon/Operational, which is the XML-formatted Windows Event Log for Sysmon (System Monitor) events. Sysmon is a service and device driver that logs system activity to the event log.
- EventID Filter: The search is further filtered to only select events with an Event ID of 1. In Sysmon, Event ID 1 corresponds to Process Creation events, which log data about newly created processes.
- Time Range Filter: The search restricts the time range of events to those occurring between the Unix timestamps 1690447949 and 1690450687. These timestamps represent the earliest and latest times in which the events occurred.
- Process Name Filter: The search then filters events to only include those where the process_name field is one of a list of specific process names

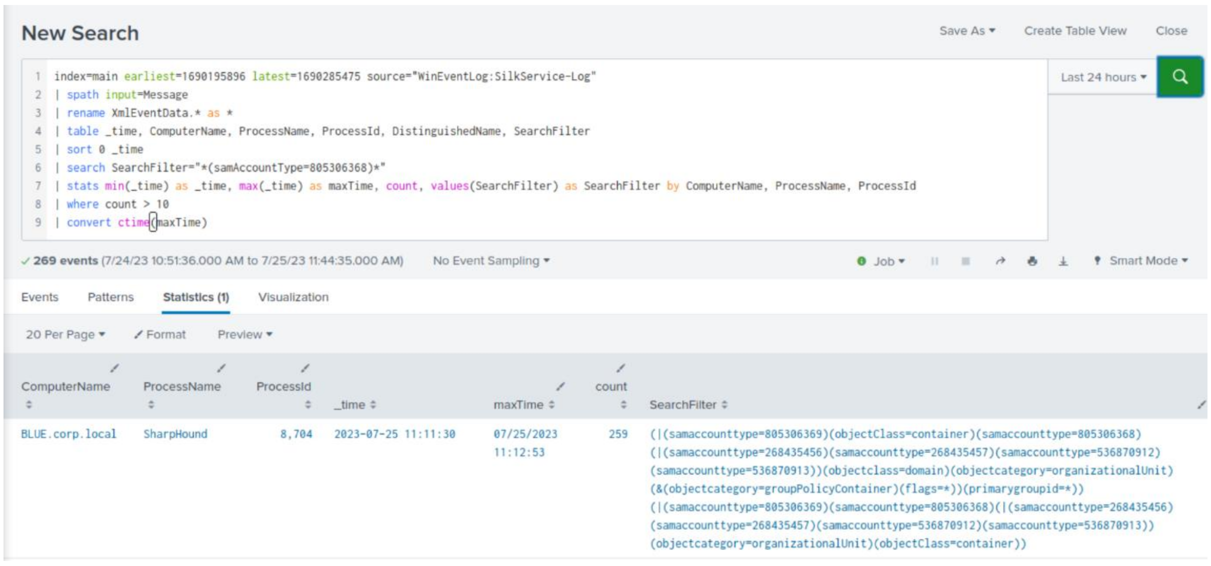
(e.g., arp.exe, chcp.com, ipconfig.exe, etc.) or where the process_name field is cmd.exe or powershell.exe and the process field contains certain substrings. This step is looking for events that involve certain system or network-related commands, as well as events where these commands were run from a Command Prompt or PowerShell session.

- Statistics: The stats command is used to aggregate events based on the fields parent_process, parent_process_id, dest, and user. For each unique combination of these fields, the search calculates the following statistics:
  - values(process) as process: This captures all unique values of the process field as a multivalue field named process.
  - min(_time) as _time: This captures the earliest time (_time) that an event occurred within each group.
- Filtering by Process Count: The where command is used to filter the results to only include those where the count of the process field is greater than 3. This step is looking for instances where multiple processes (more than three) were executed by the same parent process.

Detecting Recon By Targeting BloodHound

Timeframe: earliest=1690195896 latest=1690285475

```
index=main earliest=1690195896 latest=1690285475 source="WinEventLog:SilkService-Log"

| spath input=Message

| rename XmlEventData.* as *

| table _time, ComputerName, ProcessName, ProcessId, DistinguishedName, SearchFilter

| sort 0 _time

| search SearchFilter="*(samAccountType=805306368)*"

| stats min(_time) as _time, max(_time) as maxTime, count, values(SearchFilter) as SearchFilter
by ComputerName, ProcessName, ProcessId

| where count > 10

| convert ctime(maxTime)
```



```
index=main earliest=1690195896 latest=1690285475 source="WinEventLog:SilkService-Log"
| spath input=Message
| rename XmlEventData.* as *
| table _time, ComputerName, ProcessName, ProcessId, DistinguishedName, SearchFilter
| sort 0 _time
| search SearchFilter="*(samAccountType=805306368)*"
| stats min(_time) as _time, max(_time) as maxTime, count, values(SearchFilter) as SearchFilter by ComputerName, ProcessName, ProcessId
| where count > 10
| convert ctime(maxTime)
```

✓ 269 events (7/24/23 10:51:36.000 AM to 7/25/23 11:44:35.000 AM)   No Event Sampling ▾   Job ▾   Smart Mode ▾

ComputerName	ProcessName	ProcessId	_time	maxTime	count	SearchFilter
BLUE.corp.local	SharpHound	8,704	2023-07-25 11:11:30	07/25/2023 11:12:53	259	(|(samaccounttype=805306369)(objectClass=container)(samaccounttype=805306368)(|(samaccounttype=268435456)(samaccounttype=268435457)(samaccounttype=536870912)(samaccounttype=536870913))(objectclass=domain)(objectcategory=organizationalUnit)(&(objectcategory=groupPolicyContainer)(flags=*))(primarygroupid=*))(|(samaccounttype=805306369)(samaccounttype=805306368)(|(samaccounttype=268435456)(samaccounttype=268435457)(samaccounttype=536870912)(samaccounttype=536870913))(objectcategory=organizationalUnit)(objectClass=container))

Search Breakdown:

- Filtering by Index and Source: The search starts by selecting events from the main index where the source is WinEventLog:SilkService-Log. This source represents Windows Event Log data gathered by SilkETW.
- Time Range Filter: The search restricts the time range of events to those occurring between the Unix timestamps 1690195896 and 1690285475. These timestamps represent the earliest and latest times in which the events occurred.
- Path Extraction: The spath command is used to extract fields from the Message field, which likely contains structured data such as XML or JSON. The spath command automatically identifies and extracts fields based on the data structure.
- Field Renaming: The rename command is used to rename fields that start with XmlEventData. to the equivalent field names without the XmlEventData. prefix. This is done for easier reference to the fields in later stages of the search.
- Tabulating Results: The table command is used to display the results in a tabular format with the following columns: _time, ComputerName, ProcessName, ProcessId, DistinguishedName, and SearchFilter. The table command only includes these fields in the output.
- Sorting: The sort command is used to sort the results based on the _time field in ascending order (from oldest to newest). The 0 argument means that there is no limit on the number of results to sort.
- Search Filter: The search command is used to filter the results to only include events where the SearchFilter field contains the string *(samAccountType=805306368)*. This step is looking for events related to LDAP queries with a specific filter condition.
- Statistics: The stats command is used to aggregate events based on the fields ComputerName, ProcessName, and ProcessId. For each unique combination of these fields, the search calculates the following statistics:
    - min(_time) as _time: The earliest time (_time) that an event occurred within each group.

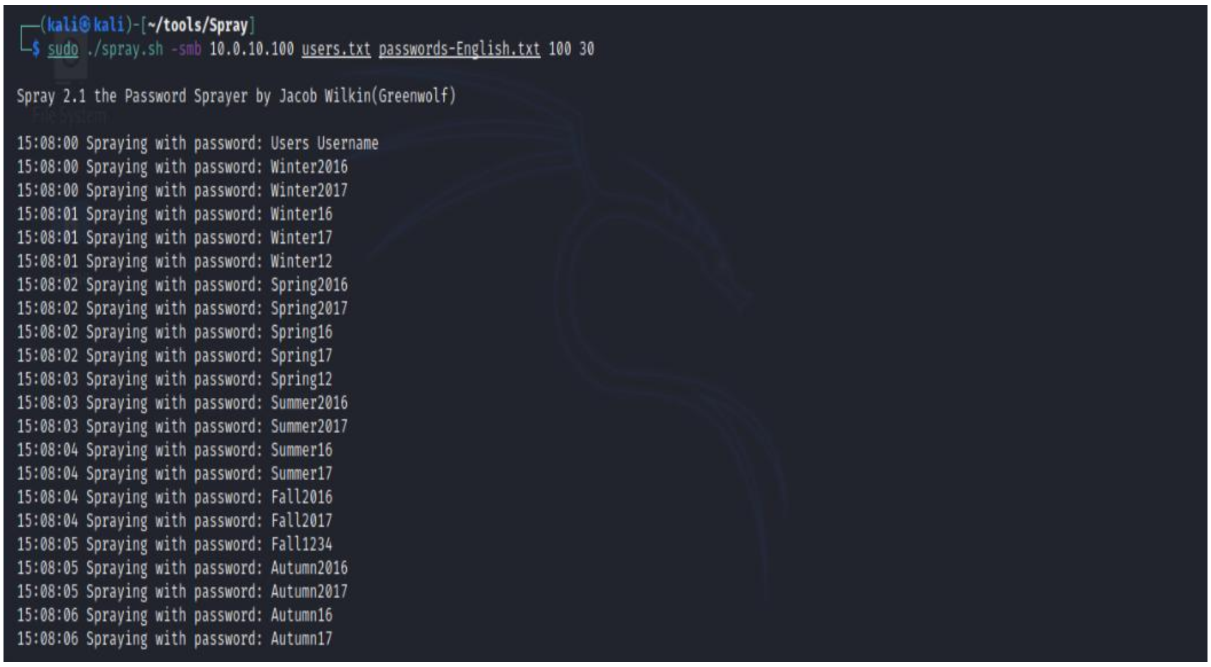
    - max(_time) as maxTime: The latest time (_time) that an event occurred within each group.
    - count: The number of events within each group.
    - values(SearchFilter) as SearchFilter: All unique values of the SearchFilter field within each group.
- Filtering by Event Count: The where command is used to filter the results to only include those where the count field is greater than 10. This step is looking for instances where the same process on the same computer made more than ten search queries with the specified filter condition.
- Time Conversion: The convert command is used to convert the maxTime field from Unix timestamp format to human-readable format (ctime).

Detecting Password Spraying

Password Spraying

Unlike traditional brute-force attacks, where an attacker tries numerous passwords for a single user account, password spraying distributes the attack across multiple accounts using a limited set of commonly used or easily guessable passwords. The primary goal is to evade account lockout policies typically instituted by organizations. These policies usually lock an account after a specified number of unsuccessful login attempts to thwart brute-force attacks on individual accounts. However, password spraying lowers the chance of triggering account lockouts, as each user account receives only a few password attempts, making the attack less noticeable.

An example of password spraying using the **Spray** tool can be seen below.



```
┌──(kali㉿kali)-[~/tools/Spray]
└─$ sudo ./spray.sh -smb 10.0.10.100 users.txt passwords-English.txt 100 30

Spray 2.1 the Password Sprayer by Jacob Wilkin(Greenwolf)

15:08:00 Spraying with password: Users Username
15:08:00 Spraying with password: Winter2016
15:08:00 Spraying with password: Winter2017
15:08:01 Spraying with password: Winter16
15:08:01 Spraying with password: Winter17
15:08:01 Spraying with password: Winter12
15:08:02 Spraying with password: Spring2016
15:08:02 Spraying with password: Spring2017
15:08:02 Spraying with password: Spring16
15:08:02 Spraying with password: Spring17
15:08:03 Spraying with password: Spring12
15:08:03 Spraying with password: Summer2016
15:08:03 Spraying with password: Summer2017
15:08:04 Spraying with password: Summer16
15:08:04 Spraying with password: Summer17
15:08:04 Spraying with password: Fall2016
15:08:04 Spraying with password: Fall2017
15:08:05 Spraying with password: Fall1234
15:08:05 Spraying with password: Autumn2016
15:08:05 Spraying with password: Autumn2017
15:08:06 Spraying with password: Autumn16
15:08:06 Spraying with password: Autumn17
```

Password Spraying Detection Opportunities

Detecting password spraying through Windows logs involves the analysis and monitoring of specific event logs to identify patterns and anomalies indicative of such an attack. A common pattern is multiple failed logon attempts with Event ID 4625 - Failed Logon from different user accounts but originating from the same source IP address within a short time frame.

Other event logs that may aid in password spraying detection include:

- 4768 and ErrorCode 0x6 - Kerberos Invalid Users
- 4768 and ErrorCode 0x12 - Kerberos Disabled Users
- 4776 and ErrorCode 0xC000006A - NTLM Invalid Users
- 4776 and ErrorCode 0xC0000064 - NTLM Wrong Password
- 4648 - Authenticate Using Explicit Credentials
- 4771 - Kerberos Pre-Authentication Failed

Then, access the Splunk interface at http://[Target IP]:8000 and launch the Search & Reporting Splunk application. The vast majority of searches covered from this point up to end of this section can be replicated inside the target, offering a more comprehensive grasp of the topics presented.

Detecting Password Spraying With Splunk

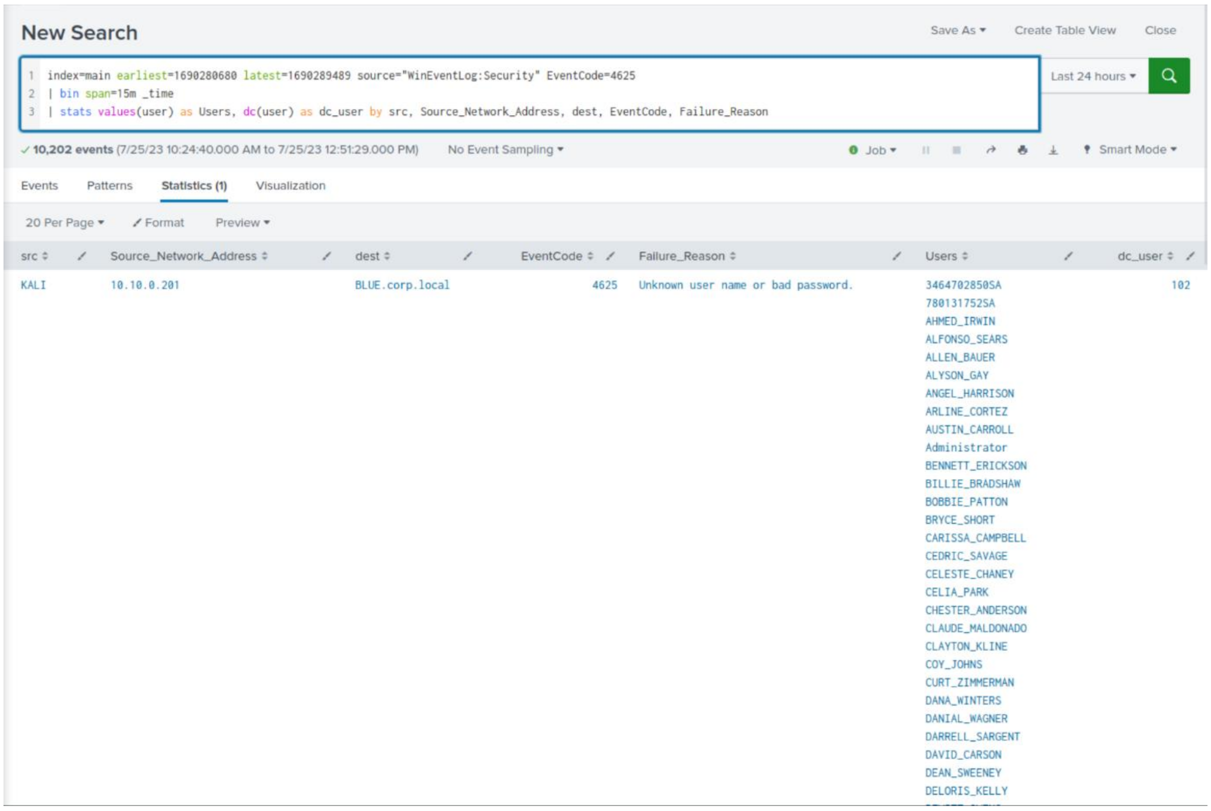
Now let's explore how we can identify password spraying attempts, using Splunk.

Timeframe: earliest=1690280680 latest=1690289489

index=main earliest=1690280680 latest=1690289489 source="WinEventLog:Security" EventCode=4625

| bin span=15m _time

| stats values(user) as Users, dc(user) as dc_user by src, Source_Network_Address, dest, EventCode, Failure_Reason



```
index=main earliest=1690280680 latest=1690289489 source="WinEventLog:Security" EventCode=4625
| bin span=15m _time
| stats values(user) as Users, dc(user) as dc_user by src, Source_Network_Address, dest, EventCode, Failure_Reason
```

✓ 10,202 events (7/25/23 10:24:40.000 AM to 7/25/23 12:51:29.000 PM)

src	Source_Network_Address	dest	EventCode	Failure_Reason	Users	dc_user
KALI	10.10.0.201	BLUE.corp.local	4625	Unknown user name or bad password.	3464702850SA 780131752SA AHMED_IRWIN ALFONSO_SEARS ALLEN_BAUER ALYSON_GAY ANGEL_HARRISON ARLINE_CORTEZ AUSTIN_CARROLL Administrator BENNETT_ERICKSON BILLIE_BRADSHAW BOBBIE_PATTON BRYCE_SHORT CARISSA_CAMPBELL CEDRIC_SAVAGE CELESTE_CHANEY CELIA_PARK CHESTER_ANDERSON CLAUDE_MALDONADO CLAYTON_KLINE COY_JOHNS CURT_ZIMMERMAN DANA_WINTERS DANIAL_WAGNER DARRELL_SARGENT DAVID_CARSON DEAN_SWEENEY DELORIS_KELLY	102

Search Breakdown:

- Filtering by Index, Source, and EventCode: The search starts by selecting events from the main index where the source is WinEventLog:Security and the EventCode is 4625. This EventCode represents failed logon attempts in the Windows Security Event Log.
- Time Range Filter: The search restricts the time range of events to those occurring between the Unix timestamps 1690280680 and 1690289489. These timestamps represent the earliest and latest times in which the events occurred.
- Time Binning: The bin command is used to create time buckets of 15 minutes duration for each event based on the _time field. This step groups the events into 15-minute intervals, which can be useful for analyzing patterns or trends over time.
- Statistics: The stats command is used to aggregate events based on the fields src, Source_Network_Address, dest, EventCode, and Failure_Reason. For each unique combination of these fields, the search calculates the following statistics:
    - values(user) as Users: All unique values of the user field within each group.
    - dc(user) as dc_user: The distinct count of unique values of the user field within each group. This represents the number of different users associated with the failed logon attempts in each group.

Detecting Responder-like Attacks

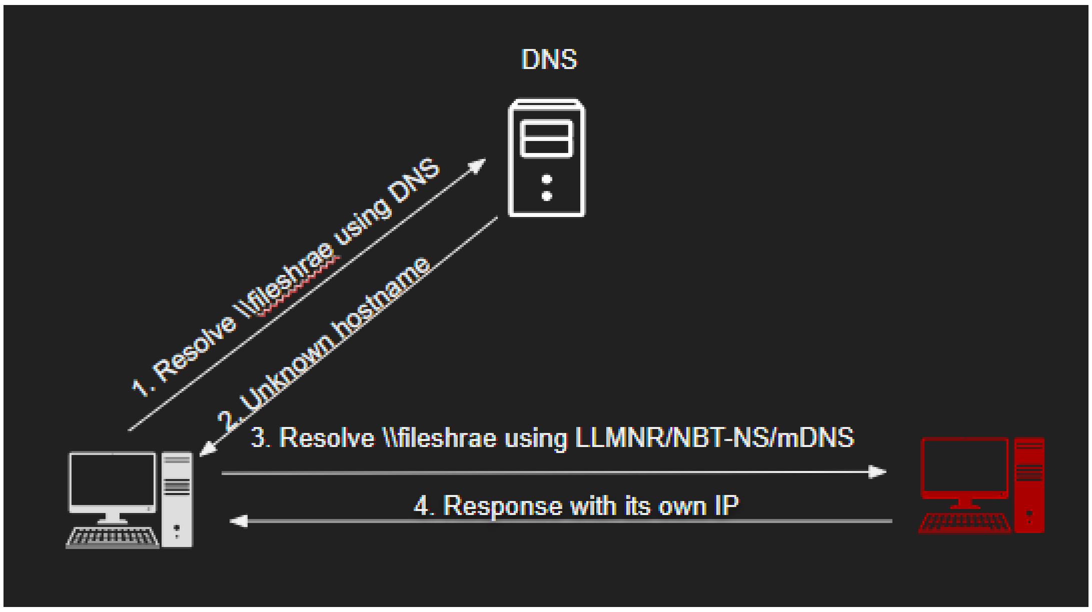
LLMNR/NBT-NS/mDNS Poisoning

LLMNR (Link-Local Multicast Name Resolution) and NBT-NS (NetBIOS Name Service) poisoning, also referred to as NBNS spoofing, are network-level attacks that exploit inefficiencies in these name resolution protocols. Both LLMNR and NBT-NS are used to resolve hostnames to IP addresses on local networks when the fully qualified domain name (FQDN) resolution fails. However, their lack of built-in security mechanisms renders them susceptible to spoofing and poisoning attacks.

Typically, attackers employ the Responder tool to execute LLMNR, NBT-NS, or mDNS poisoning.

Attack Steps:

- A victim device sends a name resolution query for a mistyped hostname (e.g., fileshrae).
- DNS fails to resolve the mistyped hostname.
- The victim device sends a name resolution query for the mistyped hostname using LLMNR/NBT-NS.
- The attacker's host responds to the LLMNR (UDP 5355)/NBT-NS (UDP 137) traffic, pretending to know the identity of the requested host. This effectively poisons the service, directing the victim to communicate with the adversary-controlled system.

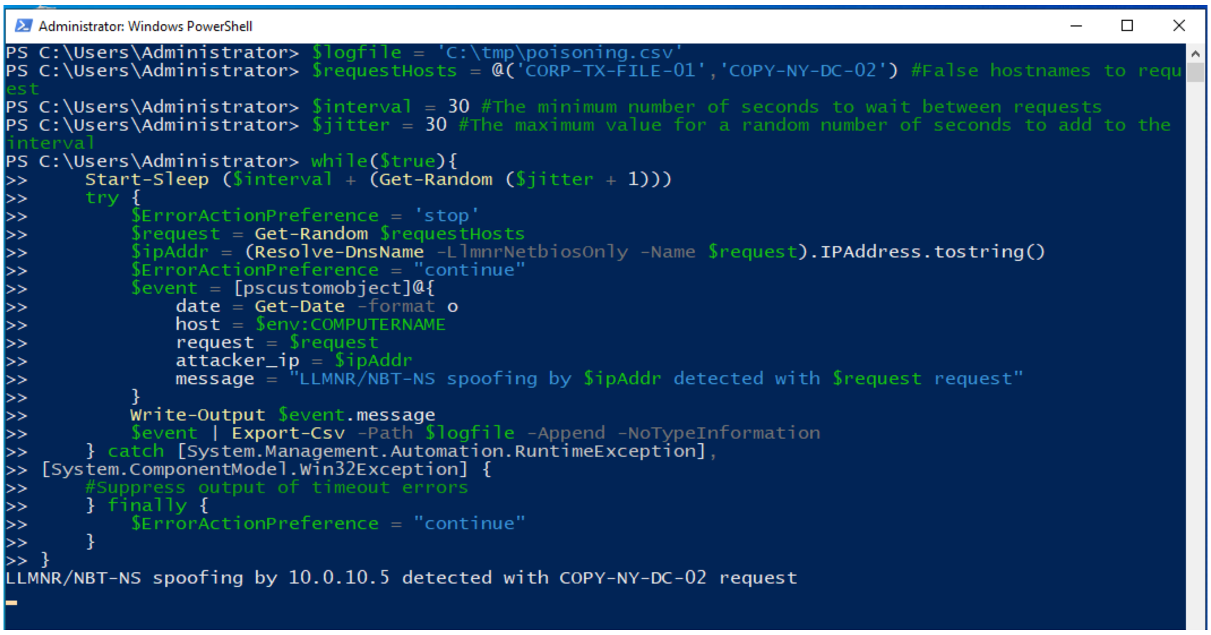

DNS
1. Resolve \\fileshrae using DNS
2. Unknown hostname
3. Resolve \\fileshrae using LLMNR/NBT-NS/mDNS
4. Response with its own IP

The result of a successful attack is the acquisition of the victim's NetNTLM hash, which can be either cracked or relayed in an attempt to gain access to systems where these credentials are valid.

Responder Detection Opportunities

Detecting LLMNR, NBT-NS, and mDNS poisoning can be challenging. However, organizations can mitigate the risk by implementing the following measures:

- Deploy network monitoring solutions to detect unusual LLMNR and NBT-NS traffic patterns, such as an elevated volume of name resolution requests from a single source.
- Employ a honeypot approach - name resolution for non-existent hosts should fail. If an attacker is present and spoofing LLMNR/NBT-NS/mDNS responses, name resolution will succeed. https://www.praetorian.com/blog/a-simple-and-effective-way-to-detect-broadcast-name-resolution-poisoning-bnrp/



```
PS C:\Users\Administrator> $logfile = 'C:\tmp\poisoning.csv'
PS C:\Users\Administrator> $requestHosts = @('CORP-TX-FILE-01','COPY-NY-DC-02') #False hostnames to requ
est
PS C:\Users\Administrator> $interval = 30 #The minimum number of seconds to wait between requests
PS C:\Users\Administrator> $jitter = 30 #The maximum value for a random number of seconds to add to the
interval
PS C:\Users\Administrator> while($true){
>>     Start-Sleep ($interval + (Get-Random ($jitter + 1)))
>>     try {
>>         $ErrorActionPreference = 'stop'
>>         $request = Get-Random $requestHosts
>>         $ipAddr = (Resolve-DnsName -LlmnrNetbiosOnly -Name $request).IPAddress.tostring()
>>         $ErrorActionPreference = "continue"
>>         $event = [pscustomobject]@{
>>             date = Get-Date -format o
>>             host = $env:COMPUTERNAME
>>             request = $request
>>             attacker_ip = $ipAddr
>>             message = "LLMNR/NBT-NS spoofing by $ipAddr detected with $request request"
>>         }
>>         Write-Output $event.message
>>         $event | Export-Csv -Path $logfile -Append -NoTypeInformation
>>     } catch [System.Management.Automation.RuntimeException],
>> [System.ComponentModel.Win32Exception] {
>>     #Suppress output of timeout errors
>>     } finally {
>>         $ErrorActionPreference = "continue"
>>     }
>> }
LLMNR/NBT-NS spoofing by 10.0.10.5 detected with COPY-NY-DC-02 request
```

A PowerShell script similar to the above can be automated to run as a scheduled task to aid in detection. Logging this activity might pose a challenge, but the New-EventLog PowerShell cmdlet can be used.

PS C:\Users\Administrator> New-EventLog -LogName Application -Source LLMNRDetection

To create an event, the Write-EventLog cmdlet should be used:

PS C:\Users\Administrator> Write-EventLog -LogName Application -Source LLMNRDetection -EventId 19001 -Message $msg -EntryType Warning

Then, access the Splunk interface at http://[Target IP]:8000 and launch the Search & Reporting Splunk application. The vast majority of searches covered from this point up to end of this section can be replicated inside the target, offering a more comprehensive grasp of the topics presented.

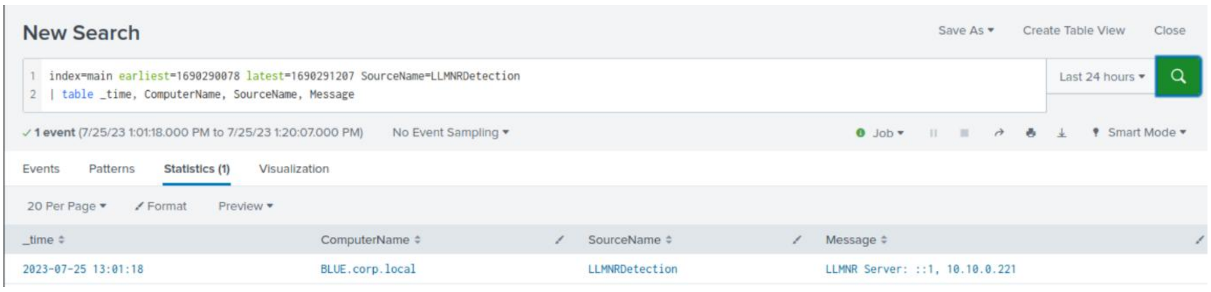
Detecting Responder-like Attacks With Splunk

Now let's explore how we can identify the Responder-like attacks previously discussed, using Splunk and logs from a PowerShell script similar to the one above.

Timeframe: earliest=1690290078 latest=1690291207

index=main earliest=1690290078 latest=1690291207 SourceName=LLMNRDetection

| table _time, ComputerName, SourceName, Message



```
index=main earliest=1690290078 latest=1690291207 SourceName=LLMNRDetection
| table _time, ComputerName, SourceName, Message
```

✓ 1 event (7/25/23 1:01:18.000 PM to 7/25/23 1:20:07.000 PM)

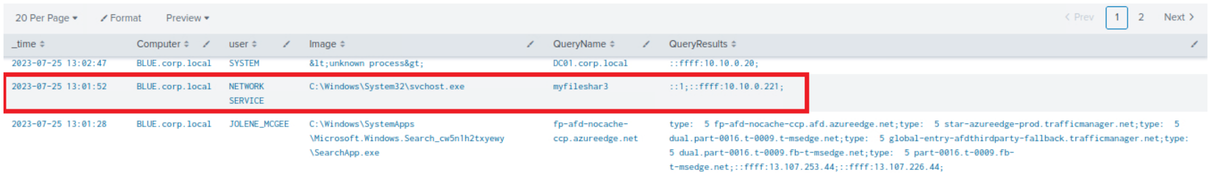
_time	ComputerName	SourceName	Message
2023-07-25 13:01:18	BLUE.corp.local	LLMNRDetection	LLMNR Server: ::1, 10.10.0.221

Sysmon Event ID 22 can also be utilized to track DNS queries associated with non-existent/mistyped file shares.

Timeframe: earliest=1690290078 latest=1690291207

index=main earliest=1690290078 latest=1690291207 EventCode=22

| table _time, Computer, user, Image, QueryName, QueryResults



_time	Computer	user	Image	QueryName	QueryResults
2023-07-25 13:02:47	BLUE.corp.local	SYSTEM	<unknown process>	DC01.corp.local	::ffff:10.10.0.20;
2023-07-25 13:01:52	BLUE.corp.local	NETWORK SERVICE	C:\Windows\System32\svchost.exe	myfileshar3	::1;::ffff:10.10.0.221;
2023-07-25 13:01:28	BLUE.corp.local	JOLENE_MCGEE	C:\Windows\SystemApps \Microsoft.Windows.Search_cw5n1h2txyewy \SearchApp.exe	fp-afd-nocache-ccp.azureedge.net	type: 5 fp-afd-nocache-ccp.afd.azureedge.net;type: 5 star-azureedge-prod.trafficmanager.net;type: 5 dual.part-0016.t-0009.t-msedge.net;type: 5 global-entry-afdthirdparty-fallback.trafficmanager.net;type: 5 dual.part-0016.t-0009.fb-t-msedge.net;type: 5 part-0016.t-0009.fb-t-msedge.net;::ffff:13.107.253.44;::ffff:13.107.226.44;

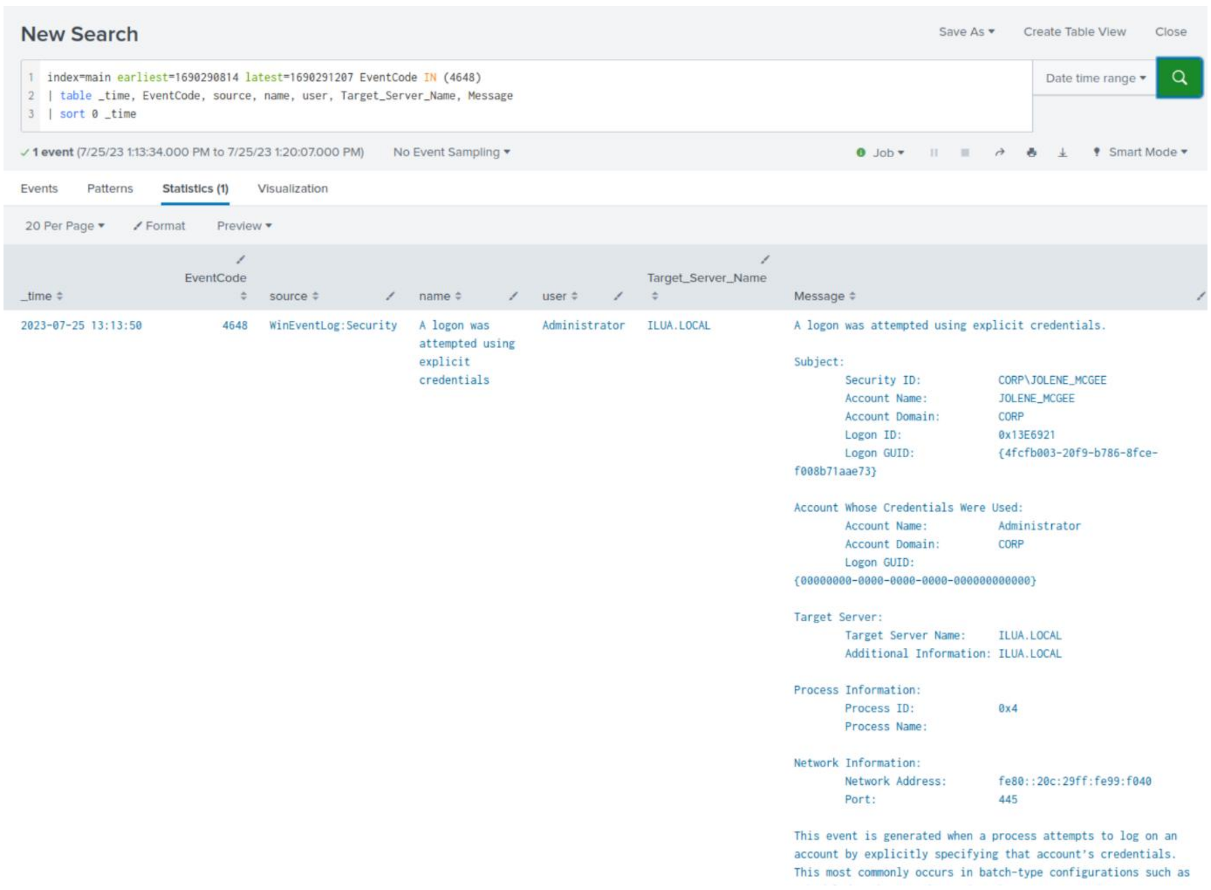
Additionally, remember that Event 4648 can be used to detect explicit logons to rogue file shares which attackers might use to gather legitimate user credentials.

Timeframe: earliest=1690290814 latest=1690291207

index=main earliest=1690290814 latest=1690291207 EventCode IN (4648)

| table _time, EventCode, source, name, user, Target_Server_Name, Message

| sort 0 _time



```
index=main earliest=1690290814 latest=1690291207 EventCode IN (4648)
| table _time, EventCode, source, name, user, Target_Server_Name, Message
| sort 0 _time
```

✓ **1 event** (7/25/23 1:13:34.000 PM to 7/25/23 1:20:07.000 PM)

_time	EventCode	source	name	user	Target_Server_Name	Message
2023-07-25 13:13:50	4648	WinEventLog:Security	A logon was attempted using explicit credentials	Administrator	ILUA.LOCAL	A logon was attempted using explicit credentials. Subject: Security ID: CORP\JOLENE_MCGEE Account Name: JOLENE_MCGEE Account Domain: CORP Logon ID: 0x13E6921 Logon GUID: {4fcfb003-20f9-b786-8fce-f008b71aae73} Account Whose Credentials Were Used: Account Name: Administrator Account Domain: CORP Logon GUID: {00000000-0000-0000-0000-000000000000} Target Server: Target Server Name: ILUA.LOCAL Additional Information: ILUA.LOCAL Process Information: Process ID: 0x4 Process Name: Network Information: Network Address: fe80::20c:29ff:fe99:f040 Port: 445 This event is generated when a process attempts to log on an account by explicitly specifying that account's credentials. This most commonly occurs in batch-type configurations such as