



Active Directory DFIR Guidance

Active Directory Digital Forensics and Incident Response guidance focuses on detecting, investigating, and responding to security incidents involving AD environments.

Below is a list of common attack techniques employed by threat actors, along with recommended investigation methodologies.

The attack methods and misconfigurations I cover it will include:

- 1- Kerberoasting attack detection.
- 2- AS-REP Roasting attack detection.
- 3- LLMNR poisoning attack detection.
- 4- NTLM relay attack detection.
- 5- NTDS dumping attack detection.

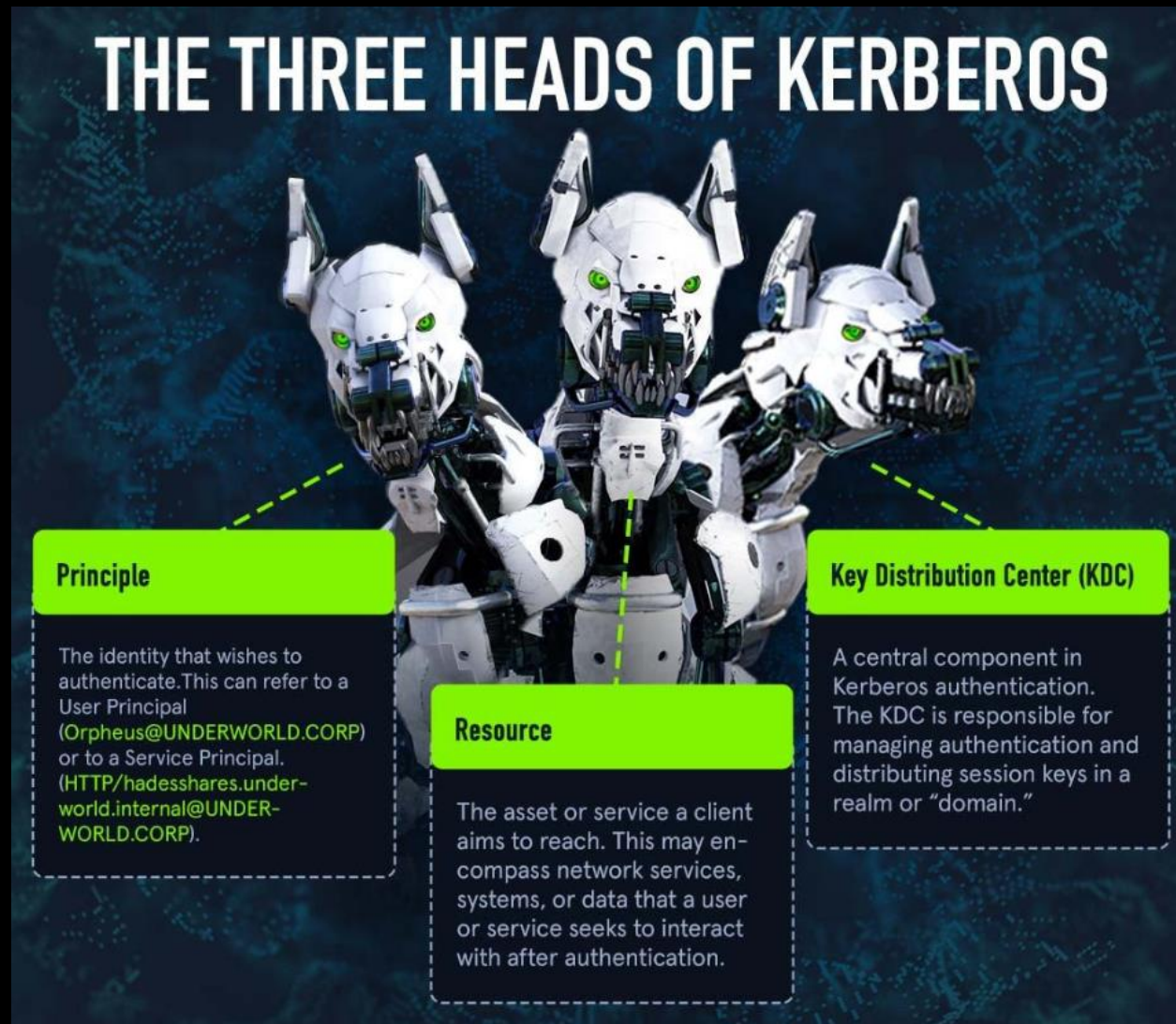
Contents

PART ONE: KERBEROASTING ATTACK DETECTION	3
PART TWO: AS-REP ROASTING ATTACK DETECTION	10
PART THREE: LLMNR POISONING ATTACK DETECTION.....	18
PART FOUR: NTLM RELAY ATTACK DETECTION	28
PART FIVE: NTDS DUMPING ATTACK DETECTION	32
REFERENCES	38

Part One: Kerberoasting attack detection

Kerberos & Active Directory

Active Directory uses the Kerberos protocol to allow users to authenticate on the network and then access services.



With Kerberos, when a user accesses a resource hosted by a Service Principal Name (SPN), a service ticket (ST) is generated by the domain controller and encrypted with that SPN password hash. The application server then decrypts and validates the ST.

When the request for the service ticket is initiated from the domain controller itself, there is no validation to ensure that the user has the necessary permissions to access the resource hosted by the SPN.

Here is where things get interesting:

If attackers know the service, as in the SPN they want to target, they can perform an ST request for it from the Domain Controller getting back an ST encrypted with the SPN's password hash.

The attacker may use Impact's [GetUserSPNs tool](#), which is often used to perform Kerberoasting attacks.

With that hash, they can go on and try to brute-force it offline to obtain the cleartext password of that service account.

Kerberos attack detection & forensic analysis:

how to detect Kerberoasting activity using domain controller logs.

As previously mentioned, Kerberoasting activity is not anything out of the ordinary, it's just regular Kerberos operations in a domain environment, except these exploit a vulnerability.

This makes it harder to detect as in corporate environments there are thousands of Kerberos events going on per minute. However, we can still find the needle in the haystack if we know what to look for.

Security Logs & events

If we go over Security Logs from a domain controller to go through detection and what kind of telemetry, we get as an aftermath of a Kerberoasting attack.

As we've already learned, Security Logs Record Event **ID 4769** on a domain controller whenever a Kerberos service ticket is requested.

AD Incident Response

Depending on the Active directory size and assets, this may be thousands of tickets per minute, and it is normal behavior.

For example, if a user accessed a file share, a service ticket would be requested by that user from the domain controller to access the service.

Since there are thousands and thousands of events occurring at a time, detecting this attack gets more difficult because it “blends in” with normal activity.

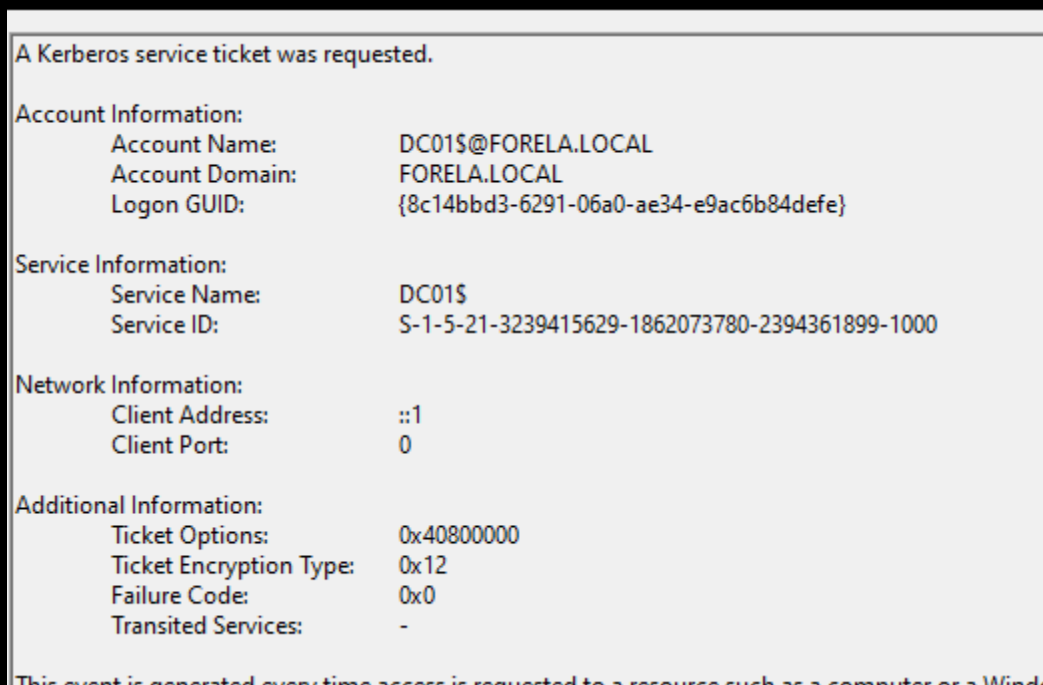
Let's open up Security Logs in Event Viewer.

SECURITY-DC_1 Number of events: 293				
Level	Date and Time	Source	Event ID	Task Category
Information	5/21/2024 8:21:25 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Information	5/21/2024 8:21:25 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Information	5/21/2024 8:21:19 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Information	5/21/2024 8:21:19 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Information	5/21/2024 8:21:16 AM	Microsoft Windows security auditing.	5140	File Share
Information	5/21/2024 8:21:15 AM	Microsoft Windows security auditing.	5140	File Share
Information	5/21/2024 8:21:13 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Information	5/21/2024 8:21:13 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Information	5/21/2024 8:21:07 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Information	5/21/2024 8:21:07 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Information	5/21/2024 8:21:01 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Information	5/21/2024 8:21:01 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Information	5/21/2024 8:21:01 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Information	5/21/2024 8:21:01 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Information	5/21/2024 8:20:55 AM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Event 4771, Microsoft Windows security auditing.				
General - Details				

We see lots of different Events, some related to Kerberos as well. To detect potential Kerberoasting activity, let's filter down for event ID 4769.

SECURITY-DC_1 Number of events: 293				
Filtered: Log file://H:\Project-Sherlock\Camp Fire 1\Triage\Domain Controller\SECURITY-DC.evtx; Source: ; Event ID: 4769. Number of events: 16				
Level	Date and Time	Source	Event ID	Task Category
Information	5/21/2024 8:20:24 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:18:51 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:18:09 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:15:12 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:13:02 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:12:05 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:12:05 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:06:15 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:06:15 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:05:54 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:05:54 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:05:54 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:05:54 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Information	5/21/2024 8:05:54 AM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
Event 4769, Microsoft Windows security auditing.				
General - Details				

There are still many events. Let's view one to understand its fields.



Here we can see that Account Name “DC01\$” requested a service ticket for service named DC01\$. In Windows names ending with \$ are typically service accounts and machine accounts. Similarly, the DC01\$ service is related to that service account.

This all belongs to normal Active Directory operations. Below that we can see an option named “Ticket Encryption type” with the value of 0x12 which equals to “AES256-CTS-HMAC-SHA1-96”.

Hint: In legitimate use cases for Kerberos ticket operations, the encryption type would be 0x12 or 0x11.

But if we see an encryption type “0x17” which is RC4 encryption, that would be a clue to look into this further, as an attacker may request a ticket in this encryption type because it allows them to crack the password.

Note: FYI All major open-source tools, like Impacket and Rubeus, request tickets in RC4 encryption type.

To further reduce the chances of false positives, we can filter out requests from other service accounts and machine accounts.

Service accounts request service tickets from domain controllers all the time; that's the nature of how service accounts work. To further reduce the events to investigate, we can filter out requests from service names starting with "\$"—they are computer accounts.

With the filters discussed above we're snooping for a 4769 event where:

1. Account name that is NOT a **service or machine account** (ending with \$), so any normal domain user account (this would be the account which is compromised and from which the attacker performed this attack.)

2. Service Names that do NOT end with \$.

3. Ticket encryption type will be **0x17** which is RC4 encryption, allowing attackers to easily crack the hash.

Now, let's use those criteria to spot the actual event that was the result of a scenario of Kerberoasting attack:

A Kerberos service ticket was requested.

Account Information:
Account Name: alonzo.spire@FORELA.LOCAL
Account Domain: FORELA.LOCAL
Logon GUID: {59f3b9b1-65ed-a449-5ac0-8ea1f68478ee}

Service Information:
Service Name: MSSQLService
Service ID: S-1-5-21-3239415629-1862073780-2394361899-1105

Network Information:
Client Address: ::ffff:172.17.79.129
Client Port: 58107

Additional Information:
Ticket Options: 0x40800000
Ticket Encryption Type: 0x17
Failure Code: 0x0
Transited Services: -

This event is generated every time access is requested to a resource such as a computer or a Windows service. T

This event can be correlated with Windows logon events by comparing the Logon GUID fields in each event. Th

This event fulfills all the conditions we set that indicate Kerberoasting attack activity.

We can see that a domain Account “alonzo.spire” requested a ticket for a service name “MSSQLService” with an encryption type of 0x17 from a workstation with IP Address 172.17.79.129.

Since if we notice that both the account name and service name do not end with \$.

The follow-up to this detection would be to:

- 1- Create a timeline of when this event was generated.
- 2- Do a forensic analysis of the machine with IP Address 172.17.79.129, and find out how the “alonzo.spire” user account got compromised.
- 3- We can use artifacts like Process Logs from Sysmon if available, prefetch, Ink files, Managed File Transfer (MFT), or registry to gain insights on what occurred around the time when Kerberoasting activity was noticed.

1- Analyzing endpoint artifacts

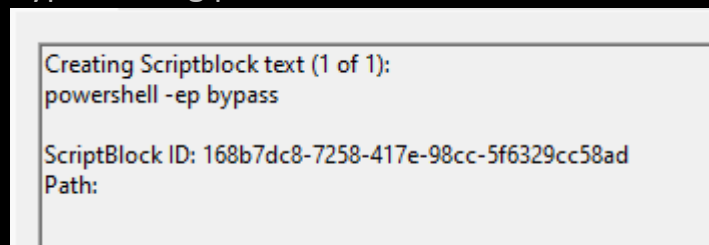
Let's expand a little bit more and explore the source endpoint artifacts from where the attack was conducted (172.17.79.129 workstation). We will analyze prefetch files and PowerShell logs.

2- PowerShell

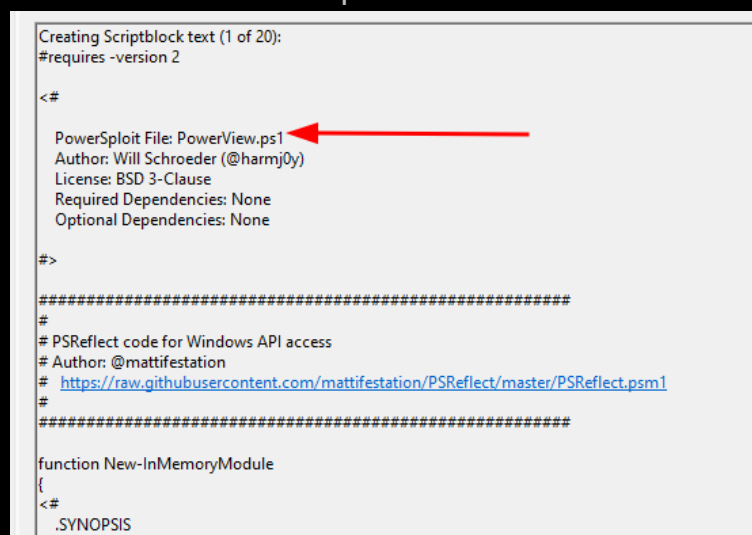
Starting with PowerShell logs, we can see executed commands/scripts by filtering for event ID 4104.

Powershell-Operational_2 Number of events: 42				
Filtered: Log: file://H:\Project-Sherlock\Camp Fire 1\Triage\Workstation\Powershell-Operational.evtx; Source: ; E				
Level	Date and Time	Source	Event ID	Task Category
Warning	5/21/2024 8:16:32 AM	PowerShell (...)	4104	Execute a Remote Command
Warning	5/21/2024 8:16:32 AM	PowerShell (...)	4104	Execute a Remote Command
Warning	5/21/2024 8:16:32 AM	PowerShell (...)	4104	Execute a Remote Command
Warning	5/21/2024 8:16:32 AM	PowerShell (...)	4104	Execute a Remote Command
Warning	5/21/2024 8:16:32 AM	PowerShell (...)	4104	Execute a Remote Command
Warning	5/21/2024 8:16:32 AM	PowerShell (...)	4104	Execute a Remote Command
Warning	5/21/2024 8:16:32 AM	PowerShell (...)	4104	Execute a Remote Command
Warning	5/21/2024 8:16:32 AM	PowerShell (...)	4104	Execute a Remote Command
Warning	5/21/2024 8:16:32 AM	PowerShell (...)	4104	Execute a Remote Command
Warning	5/21/2024 8:16:32 AM	PowerShell (...)	4104	Execute a Remote Command
Warning	5/21/2024 8:16:32 AM	PowerShell (...)	4104	Execute a Remote Command
Warning	5/21/2024 8:16:32 AM	PowerShell (...)	4104	Execute a Remote Command
Warning	5/21/2024 8:16:29 AM	PowerShell (...)	4104	Execute a Remote Command

Timestamps show 08:16, which is in my local time. Converting this in UTC, the time is 03:16 which is just two minutes before our Kerberoasting activity. Looking at the first event we see a PowerShell script execution bypass being performed.



This enables scripts to be executed in a PowerShell session. The follow-up events occurred all at the same time, which could be part of a single script, as the PowerShell script block records the full script being executed.



We find evidence that this is PowerView.ps1 script, which is an offensive PowerShell script used for AD enumeration and is used in post-exploitation activities.

Attackers can use this to find any Kerberoastable accounts.

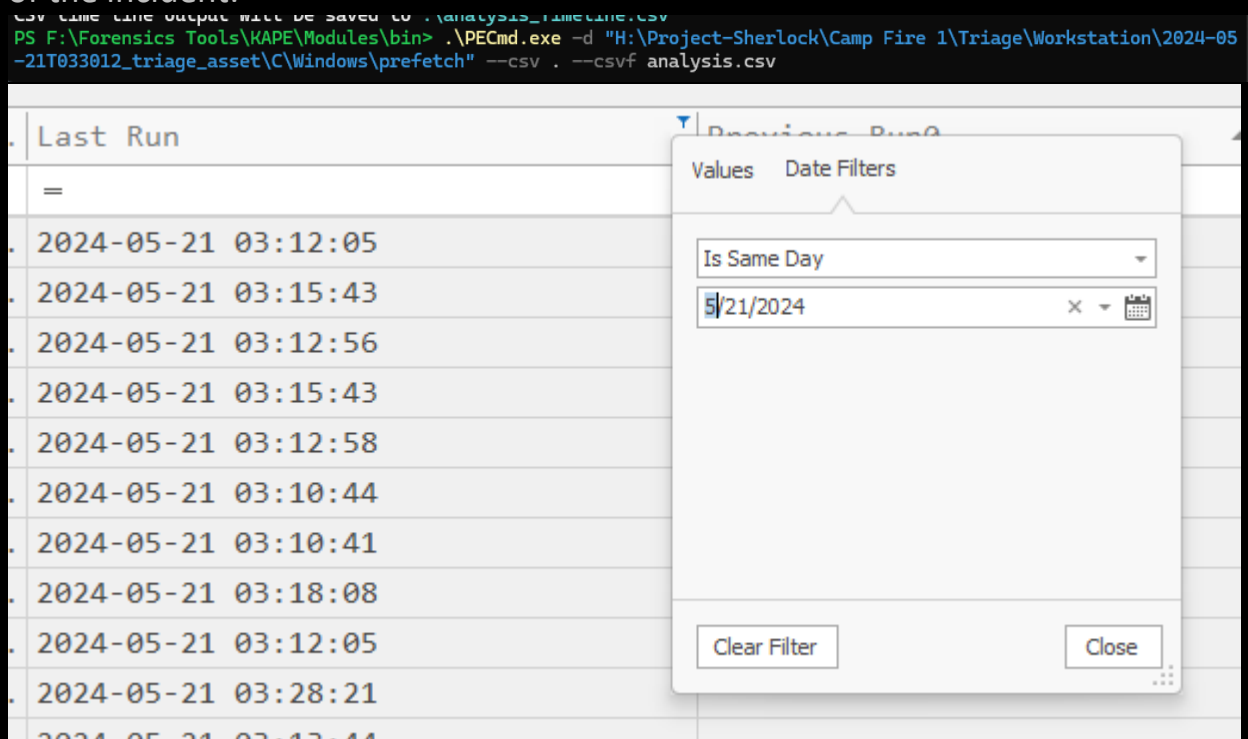
3- Prefetch files

Now let's pivot to prefetch files. We will use [PECmd by Eric Zimmerman](#) to parse the prefetch files and event viewer to go through the event logs. We used PECmd to parse the provided prefetch files and save them in the current directory with the name analysis.csv:

```
PECmd.exe -d "path-to-prefetch-files" --csv . --csvf outputfilename.csv
```

AD Incident Response

Analyzing the CSV file with Timeline Explorer, we'll need to look for any exe execution around the timeline we have established. First, filter for the date of the incident:



Looking at the last run timestamps, we find an exe was executed just a second before our malicious events were logged on the Domain Controller.

Executable Name	Run C...	Size	Last Run
Rube	=	=	=
RUBEUS.EXE	1	86612	2024-05-21 03:18:08

If search in google for RUBEUS.exe will find this tool used for Kerberos abuse tool.

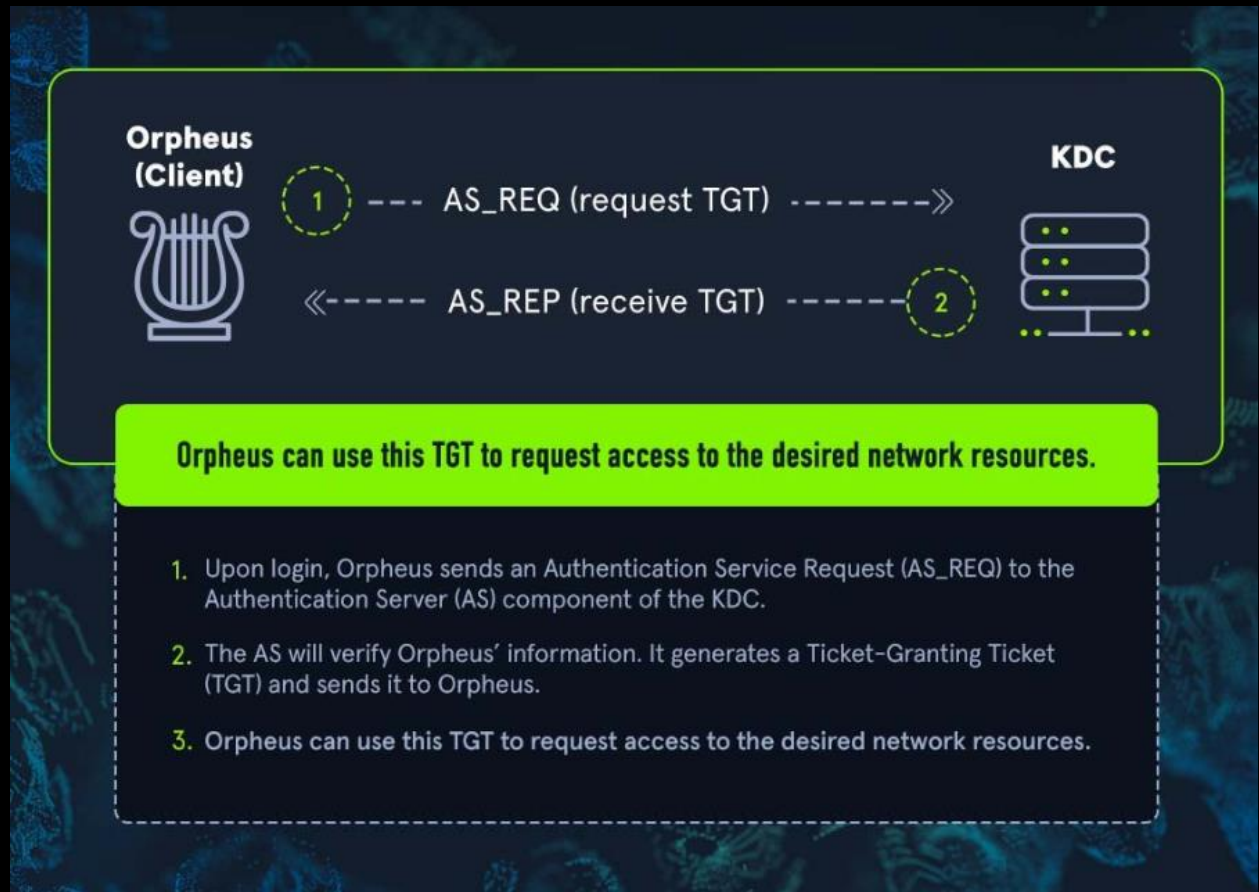
Part Two: AS-REP Roasting attack detection

How to detect AS-REP roasting attacks in part two of a critical Active Directory (AD) attack detections & misconfigurations.

When authentication occurs within Kerberos, the first thing that happens is an authentication request to the domain controller so the identity trying to authenticate can be verified.

That request is known as **Authentication Server Request (AS-REQ.)**

This process is commonly referred to as Kerberos preauthentication.



After the client's authentication is validated, the domain controller sends an Authentication Server Reply (AS-REP) to the client, containing a session key and a Ticket Granting Ticket (TGT).

The session key is encrypted using the user's password hash so that only that user can decrypt and reuse it.

Now check this out! Users within the domain can be configured to skip the preauthentication process, and that means attackers can send the AS-REQ to the domain controller on behalf of any user configured that way. (Since

the AS-REP contains the session keys encrypted with the user's password hash.)

As a result, since the AS-REP contains the session keys encrypted with the user's password hash, they can obtain the password hash of any user.

(Tools like GetNPUsers from the [Impacket tool suite](#) can simplify this process.)

Attackers can then try to brute force passwords to decrypt the session key. If the key decrypts, that means the attacker successfully guessed it and now has the user's password.

People in Cyber security will often refer to this encrypted session key as a hash, but it's really not a hash at all. Still, tools like [Hashcat](#) and John the Ripper can brute force many passwords against this "hash" in a short time to try to recover the user's plaintext password.

By default, the AD User Account Control (UAC) setting: "Do not require Kerberos preauthentication" is disabled. This means that Kerberos preauthentication is performed for all users.

But here's the kicker: This account option can be enabled manually and is seen from time to time during real-world engagements.

Now let's know what the Detection mechanisms in this attack

Detecting AS-REP Roasting

How you can detect AS-REP Roasting activity using domain controller logs.

Spotting this type of attack is easier than Kerberoasting attack detection. However, it's still complex because you need knowledge of AD and event logs to properly filter down to malicious activity.

As we mentioned in part one of this series, regular AD operations in corporate environments make it harder to detect malicious activity because there are

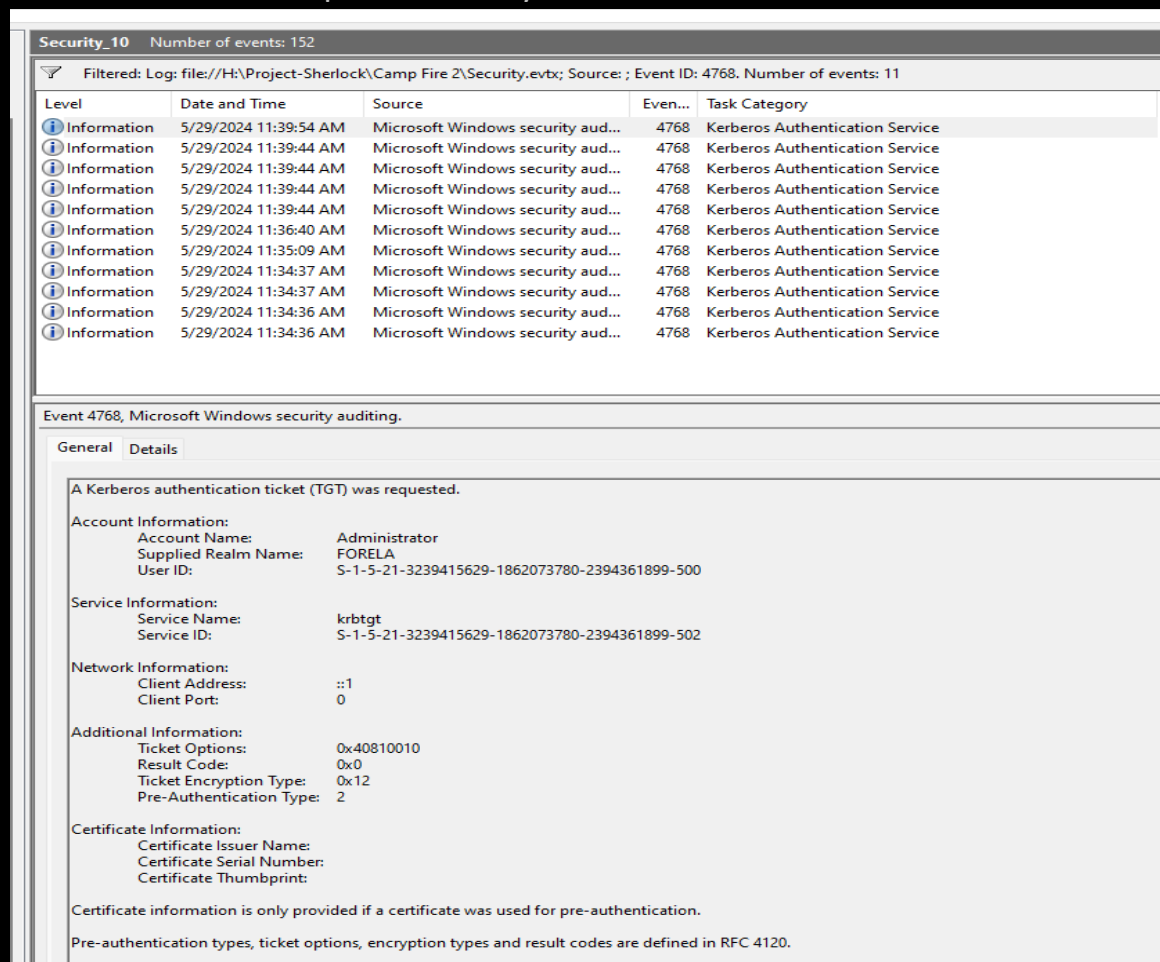
AD Incident Response

thousands of Kerberos events going on per minute. However, if we know what to look for, we can still find this needle in the haystack.

1- Filter logs by Event ID 4768

Event ID 4768 is an event ID recorded in Security Logs on the domain controller whenever a Kerberos Authentication ticket is requested.

Depending on the Active directory size and assets, this can be well over thousands of tickets per minute by different accounts in the network.



The screenshot displays the Windows Security Event Viewer interface. At the top, a filter is applied: "Filtered: Log: file://H:\Project-Sherlock\Camp Fire 2\Security.evtx; Source: ; Event ID: 4768. Number of events: 11". Below this, a table lists 11 events, all of which are Information level and have the same source and task category. The details pane for the selected event (ID 4768) is shown below the table, providing a comprehensive overview of the Kerberos authentication ticket request.

Level	Date and Time	Source	Even...	Task Category
Information	5/29/2024 11:39:54 AM	Microsoft Windows security aud...	4768	Kerberos Authentication Service
Information	5/29/2024 11:39:44 AM	Microsoft Windows security aud...	4768	Kerberos Authentication Service
Information	5/29/2024 11:39:44 AM	Microsoft Windows security aud...	4768	Kerberos Authentication Service
Information	5/29/2024 11:39:44 AM	Microsoft Windows security aud...	4768	Kerberos Authentication Service
Information	5/29/2024 11:39:44 AM	Microsoft Windows security aud...	4768	Kerberos Authentication Service
Information	5/29/2024 11:36:40 AM	Microsoft Windows security aud...	4768	Kerberos Authentication Service
Information	5/29/2024 11:35:09 AM	Microsoft Windows security aud...	4768	Kerberos Authentication Service
Information	5/29/2024 11:34:37 AM	Microsoft Windows security aud...	4768	Kerberos Authentication Service
Information	5/29/2024 11:34:37 AM	Microsoft Windows security aud...	4768	Kerberos Authentication Service
Information	5/29/2024 11:34:36 AM	Microsoft Windows security aud...	4768	Kerberos Authentication Service
Information	5/29/2024 11:34:36 AM	Microsoft Windows security aud...	4768	Kerberos Authentication Service

Event 4768, Microsoft Windows security auditing.

General Details

A Kerberos authentication ticket (TGT) was requested.

Account Information:

- Account Name: Administrator
- Supplied Realm Name: FORELA
- User ID: S-1-5-21-3239415629-1862073780-2394361899-500

Service Information:

- Service Name: krbtgt
- Service ID: S-1-5-21-3239415629-1862073780-2394361899-502

Network Information:

- Client Address: ::1
- Client Port: 0

Additional Information:

- Ticket Options: 0x40810010
- Result Code: 0x0
- Ticket Encryption Type: 0x12
- Pre-Authentication Type: 2

Certificate Information:

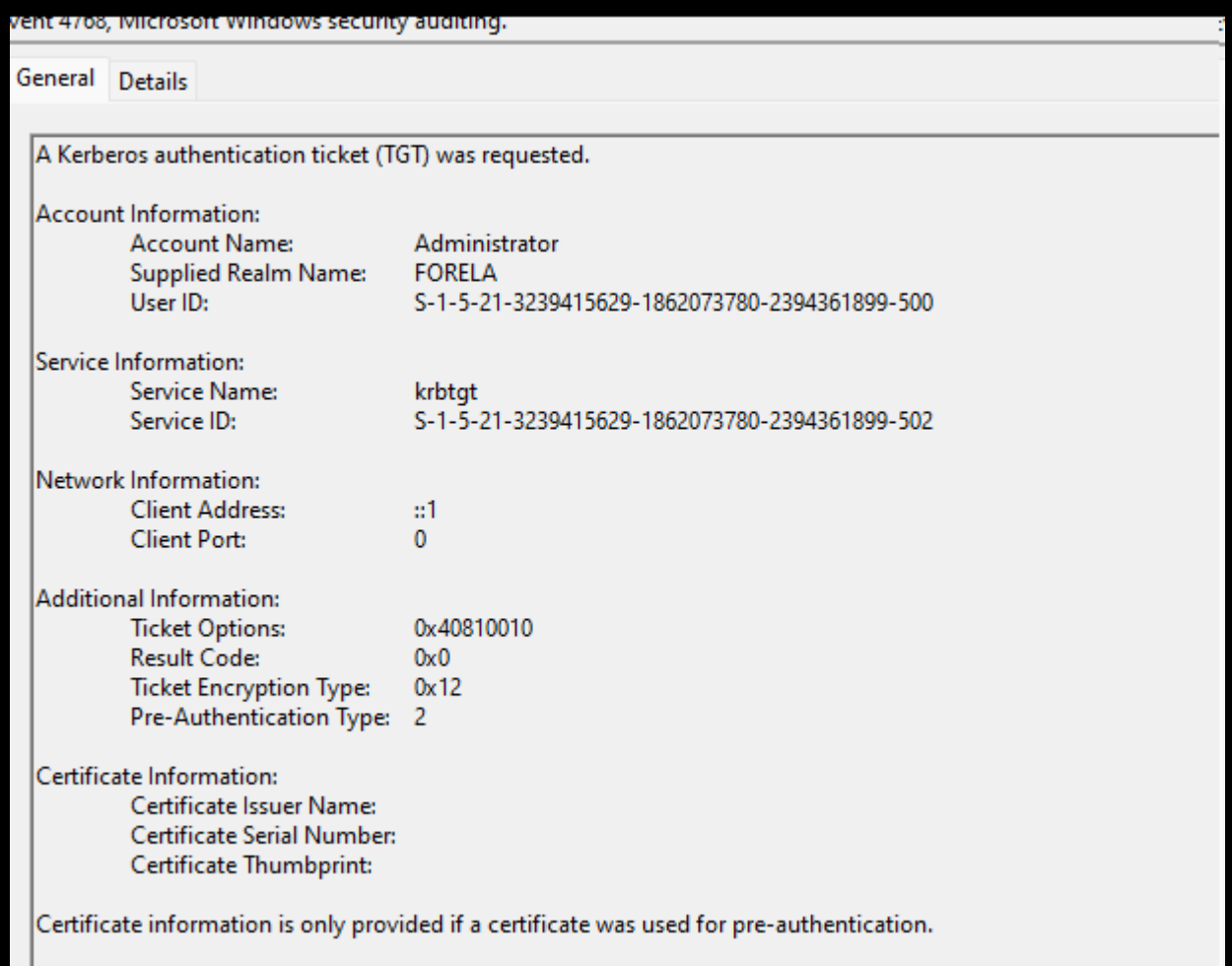
- Certificate Issuer Name:
- Certificate Serial Number:
- Certificate Thumbprint:

Certificate information is only provided if a certificate was used for pre-authentication.

Pre-authentication types, ticket options, encryption types and result codes are defined in RFC 4120.

AD Incident Response

Let's view one of the many events to understand this event's fields.



- Account Name: The user account that requested an authentication ticket from the domain controller.
- Service Name: Name of the service that handled the ticket.
- Ticket Encryption type: Depicts the Ticket encryption algorithm used (For example aes, RC4, etc).
- Pre-Authentication Type: The status code shows whether pre-authentication was disabled or enabled for the said object (The Account Name).

We can see that the administrator user requested an authentication ticket and the service name is **krbtgt**. This is regular operations and whenever an account logs in to a workstation, **krbtgt** is a universal AD service that handles Kerberos authentication.

Now let's discuss a few of the filters or conditions that would indicate a possible attack.

In legitimate use cases for Kerberos ticket operations, the encryption type would be 0x12 or 0x11.

But if we see an encryption type “0x17” which is RC4 encryption, that would be a clue to look into this further, as an attacker may request a ticket in this encryption type because it allows them to crack the password.

Note: All major open-source tools, like Impacket and Rubeus, request tickets in RC4 encryption type.

User accounts request authentication tickets from domain controllers all the time; that's the nature of how Active Directory Kerberos authentication works.

To further reduce the events to investigate, we can filter out requests from all service names other than “krbtgt”.

This is because during this attack, the attacker retrieves the authentication ticket just like a legitimate user account would, and krbtgt is a default AD Service that handles the authentication flow in Active Directory.

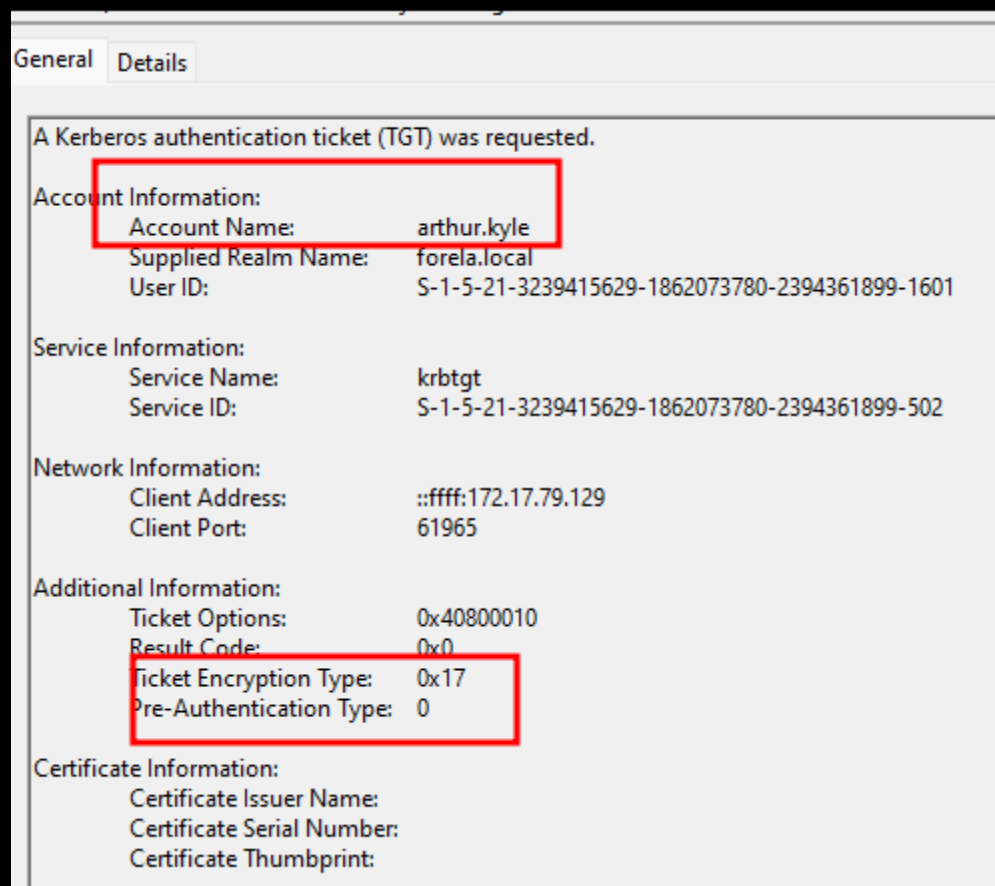
The major indicator that the AS-REP attack has been successful (the attacker managed to get the ticket, whether they cracked it or not is another case) is the pre-authentication type value in the resultant logs.

Note: A great way to threat hunt for this attack is to just look for pre-authentication type = 0, which means it is disabled. This would already remove 90 percent of the noise in the logs, leaving more granular results to go through.

With the filters discussed above we're snooping for a 4768 event where:

1. **Pre-Authentication Type is 0**, which means it is disabled. This is a major condition to be fulfilled as without this condition, the attack can't happen.
2. **Service Name should always be krbtgt**. This is also straightforward. As only krbtgt can perform authentication-related processes in AD.
3. **Ticket encryption type will be 0x17 which is RC4 encryption**, allowing attackers to easily crack the hash.

Here's an example of identifying an actual event that was the result of a AS-REP attack using the detection tips above:



This event fulfills all the conditions we set which would highly indicate AS-REP Roasting activity. We can see that a domain Account "arthur.kyle" requested an authentication ticket for a user whose pre-authentication is disabled, with an encryption type of 0x17 from a workstation with IP Address 172.17.79.129.

Correlating events to detect a compromised account:

So far we found out that the user arthur.kyle got compromised due to pre-authentication being disabled.

What we don't know is which user account was used to perform the attack.

It's important to note that while the "arthur.kyle" account is the victim here, the bad actor used another account to perform the attack.

We need to find that account, too, because it's also been compromised! And our single AS-REP incident may expand into an incident with a wider scope as we keep more compromised assets.

We have the machine's IP address from which the request originated. We will look for Kerberos service ticket events, as every domain user account requests those either during login/authentication or normal domain usage.

Filter for Event ID 4769 and look for events around the time of the malicious event.

We spotted an event about a minute later after the malicious event and it originated from User account of the "172.17.79.129" machine.

Now nothing in this event is malicious by itself. It's purely a regular operation and is not a result of any attack or exploit.

But since we already found AS-REP activity in the previous section, and we know it originated from this machine, this event caught our eye.

Here we can see that happy.grunwald was the user account logged in around the time of AS-REP Roasting attack on the source machine (machine that performed the attack).

It can be safe to assume now that this user account is compromised hence expanding the scope of the incident.

Note: you can AS-REP Roast with just a user list (i.e., if you gather it from an SMB NULL SESSION). But if you're just running a tool like GetNPUsers.py or Rubeus you need a valid user account to query the user list (which all happens in the background when you run the attack).

That means this attack can be executed without any authentication if an attacker has a user list through some means (null session, an SQLi on an AD login form, successful username enumeration using Kerbrute, etc.

Part three: LLMNR poisoning attack detection

How to detect LLMNR poisoning attacks in part three of a critical Active Directory (AD) attack detections & misconfigurations

LLMNR poisoning explained

In earlier versions of Windows networks, the Network Basic Input/output System (NetBIOS) protocol was used to perform operations across the network. A crucial component of this protocol was NetBIOS Name Service (NBT-NS), which was responsible for name registration and resolution.

Link-Local Multicast Name Resolution (LLMNR) is the successor of NBT-NS. It performs the same task as its predecessor, name resolution for hosts on the same local network.

LLMNR allows for the resolution of both IPv4 and IPv6 addresses into hostnames without the need for a DNS server on the local network. If a request to a DNS server fails (e.g., if a DNS server is not available), an LLMNR query is made across the local network to attempt to resolve that request.

So, how can this be a bad thing?

LLMNR does not require authentication to perform those name resolutions. That means that any computer on a local network can perform a LLMNR query.

If an attacker is listening on the local network, they can respond to those queries. This can lead to potential harmful behavior and attacks, such as LLMNR poisoning.

During an LLMNR poisoning attack, the attacker is listening for LLMNR requests. When a request is made across the local network, their device responds with its own IP address redirecting network traffic.

[Responder](#) is a popular tool to perform LLMNR poisoning attacks.

If an LLMNR event occurs on the network and the attacker is listening, Responder can obtain sensitive information regarding the victim such as the IP address, username, and password hash.

AD Incident Response

With the hash in their possession, attackers can attempt to crack it and obtain the password in clear text. Or they can try to relay that hash to authenticate as the victim in a particular service.

FYI: LLMNR/NBT-NS Poisoning and SMB Relay is mapped to the sub-technique T1557.001 on The MITRE ATT&CK framework.

Detecting LLMNR poisoning

Let's dive into how we can find evidence of an LLMNR poisoning attack on network traffic.

Since network traffic contains so much extra noise (all regular web traffic for example), performing network forensics to pinpoint anomalies becomes difficult due to the sheer amount of traffic in corporate environments.

So we need to know exactly where we should focus our efforts to find any evidence of malicious activity.

Just like in previous blogs, we'll include a few conditions/traces that need to be present for us to conclude that indeed the attack mentioned above did occur.

1- Analyze LLMNR traffic to find the rogue device

Start by opening the packet capture in Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
1	2024-06-24 11:17:22.462145	172.17.79.130	172.17.79.1	SSH	178	Server: Encrypted packet (len=124)
2	2024-06-24 11:17:22.469971	172.17.79.1	172.17.79.130	TCP	60	60381 → 22 [ACK] Seq=1 Ack=125 Win=4098 Len=0
3	2024-06-24 11:17:22.545625	172.17.79.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
4	2024-06-24 11:17:23.413556	172.17.79.136	34.160.236.64	TCP	60	51320 → 443 [ACK] Seq=1 Ack=1 Win=63336 Len=0
5	2024-06-24 11:17:23.413556	34.160.236.64	172.17.79.136	TCP	60	443 → 51320 [ACK] Seq=1 Ack=2 Win=64240 Len=0
6	2024-06-24 11:17:23.705126	172.17.79.136	216.58.204.142	UDP	1288	50231 → 443 Len=1246
7	2024-06-24 11:17:23.705378	172.17.79.136	216.58.204.142	UDP	771	50231 → 443 Len=729
8	2024-06-24 11:17:23.748367	172.17.79.1	224.0.0.22	IGMPv3	60	Membership Report / Leave group 224.168.100.1
9	2024-06-24 11:17:23.773462	172.17.79.136	35.208.249.213	TCP	60	51308 → 443 [ACK] Seq=1 Ack=1 Win=62852 Len=0
10	2024-06-24 11:17:23.773463	35.208.249.213	172.17.79.136	TCP	60	443 → 51308 [ACK] Seq=1 Ack=2 Win=64240 Len=0
11	2024-06-24 11:17:23.845843	216.58.204.142	172.17.79.136	UDP	69	443 → 50231 Len=27
12	2024-06-24 11:17:23.856213	172.17.79.1	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.168.100.1
13	2024-06-24 11:17:23.864200	216.58.204.142	172.17.79.136	UDP	490	443 → 50231 Len=448
14	2024-06-24 11:17:23.864201	216.58.204.142	172.17.79.136	UDP	146	443 → 50231 Len=104
15	2024-06-24 11:17:23.864778	172.17.79.136	216.58.204.142	UDP	77	50231 → 443 Len=35
16	2024-06-24 11:17:23.890873	172.17.79.136	216.58.204.142	UDP	75	50231 → 443 Len=33

LLMNR runs on **UDP port 5355** by default so we'll filter for that. Now we can only see LLMNR traffic.

	Time	Source	Destination	Protocol	Length	Info
9262	2024-06-24 11:18:30.895613	fe80::7994:1860:711:c243	ff02::1:3	LLMNR	85	Standard query 0xe708 A DCC01
9263	2024-06-24 11:18:30.895766	172.17.79.136	224.0.0.252	LLMNR	65	Standard query 0xe708 A DCC01
9264	2024-06-24 11:18:30.896012	fe80::7994:1860:711:c243	ff02::1:3	LLMNR	85	Standard query 0x2c01 AAAA DCC01
9265	2024-06-24 11:18:30.896077	172.17.79.136	224.0.0.252	LLMNR	65	Standard query 0x2c01 AAAA DCC01
9268	2024-06-24 11:18:30.900217	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	LLMNR	106	Standard query response 0xe708 A DCC01 A 17:
9269	2024-06-24 11:18:30.902155	172.17.79.135	172.17.79.136	LLMNR	86	Standard query response 0xe708 A DCC01 A 17:
9274	2024-06-24 11:18:30.903430	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	LLMNR	118	Standard query response 0x2c01 AAAA DCC01 A 17:
9277	2024-06-24 11:18:30.907809	172.17.79.135	172.17.79.136	LLMNR	98	Standard query response 0x2c01 AAAA DCC01 A 17:

Looking at source and destination IPs, we see that the IP address 172.17.79.136 performed a DNS query “DCC01” and another IP 172.17.79.135 that responds to that query.

We notice that the query in question is “DCC01” which seems like a typo for a hostname (the actual hostname is DC01 but the user wrote a typo and instead the queried hostname was DCC01).

From this, we can paint a picture that the user wanted to navigate to a file share on DC01 but instead wrote a typo “DCC01” which caused the DNS Server to fail and use LLMNR protocol for hostname resolution—from where the attacker’s rogue server acted as DC and grabbed the credentials.

In legitimate cases, the IP address responding to the queries should belong to the Domain Controller. In our case, the DC IP is 172.17.79.4 and the machine responding to the queries is different, thus prompting us to look further into it.

Then we need to confirm that the “.135” IP address is a rogue device as we currently suspect that machine to be controlled by attackers. Remove the Wireshark filter and add a filter for DHCP.

	Time	Source	Destination	Protocol	Length	Info
138	2024-06-24 11:18:30.907809	dc01.forela.local	Forela-Wkstn002.forela.local	SMB2	138	Write Response
171	2024-06-24 11:18:30.907809	Forela-Wkstn002.forela.local	dc01.forela.local	SMB2	171	Read Request Len:1024 Off:0 File: srvsvc
950	2024-06-24 11:18:30.907809	dc01.forela.local	Forela-Wkstn002.forela.local	SRVSVC	950	NetShareEnumAll response[Malformed Packet]
146	2024-06-24 11:18:30.907809	Forela-Wkstn002.forela.local	dc01.forela.local	SMB2	146	Close Request File: srvsvc
182	2024-06-24 11:18:30.907809	dc01.forela.local	Forela-Wkstn002.forela.local	SMB2	182	Close Response
174	2024-06-24 11:18:30.907809	Forela-Wkstn002.forela.local	dc01.forela.local	SMB2	174	Tree Connect Request Tree: \\DC01\DC-Confidential
138	2024-06-24 11:18:30.907809	dc01.forela.local	Forela-Wkstn002.forela.local	SMB2	138	Tree Connect Response
234	2024-06-24 11:18:30.907809	Forela-Wkstn002.forela.local	dc01.forela.local	SMB2	234	Create Request File:
298	2024-06-24 11:18:30.907809	dc01.forela.local	Forela-Wkstn002.forela.local	SMB2	298	Create Response File:
146	2024-06-24 11:18:30.907809	Forela-Wkstn002.forela.local	dc01.forela.local	SMB2	146	Close Request File:
182	2024-06-24 11:18:30.907809	dc01.forela.local	Forela-Wkstn002.forela.local	SMB2	182	Close Response
182	2024-06-24 11:18:30.907809	Forela-Wkstn002.forela.local	dc01.forela.local	SMB2	182	Create Request File: srvsvc

AD Incident Response

We see the DHCP request between the suspected IP and the DHCP server. We can see the hostname for the machine that leased the “.135” IP address.

```
> Option: (57) Maximum DHCP Message Si
  > Option: (12) Host Name
    Length: 4
    Host Name: kali
  > Option: (255) End
```

The hostname is Kali, an offensive Linux distribution used by hackers. This confirms that this machine is not any domain-joined machine and needs further investigation.

Evidence of credential stealing in SMB traffic

Now we'll analyze SMB traffic to find credentials being stolen and sent over to the attacker's Kali Linux machine.

Let's start over by adding an smb2 filter in Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
9278	2024-06-24 11:18:30.910756	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	314	Negotiate Protocol Response
9279	2024-06-24 11:18:30.911316	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	296	Negotiate Protocol Request
9280	2024-06-24 11:18:30.912373	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	314	Negotiate Protocol Response
9290	2024-06-24 11:18:30.919816	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
9291	2024-06-24 11:18:30.921154	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHAL
9292	2024-06-24 11:18:30.922052	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
9293	2024-06-24 11:18:30.925867	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	150	Session Setup Response, Error: STATUS_ACCESS_DENIED
9313	2024-06-24 11:18:30.942878	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	314	Negotiate Protocol Request
9316	2024-06-24 11:18:30.947266	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	314	Negotiate Protocol Response
9317	2024-06-24 11:18:30.948149	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
9319	2024-06-24 11:18:30.952297	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHAL
9320	2024-06-24 11:18:30.952999	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
9321	2024-06-24 11:18:30.966870	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	150	Session Setup Response, Error: STATUS_ACCESS_DENIED
9343	2024-06-24 11:18:30.980609	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	314	Negotiate Protocol Request
9347	2024-06-24 11:18:30.987381	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	314	Negotiate Protocol Response
9348	2024-06-24 11:18:30.988537	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE

We can see multiple NTLM Authentication Negotiations. To only see negotiation packets and not other SMB packets, add a filter for “ntlmssp”. We will apply this filter first and then go back to viewing full SMB traffic.

No.	Time	Source	Destination	Protocol	Length	Info
9348	2024-06-24 11:18:30.988537	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
9349	2024-06-24 11:18:30.989174	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
9350	2024-06-24 11:18:30.989862	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
9505	2024-06-24 11:18:40.778289	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
9507	2024-06-24 11:18:40.781675	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
9508	2024-06-24 11:18:40.782773	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon

- We can see that authentication is taking place for the domain-joined user called “john.deacon.”
- One other pattern we can spot is that multiple occurrences of NTLM authentications can be seen occurring within seconds of each other.

By default, we cannot see the IPs, so to view that go to View > Name resolution and enable Resolve Network Addresses. This will now display hostnames.

Source	Destination	Protocol	Length	Info
88537 Forela-Wkstn002.local	DCC01	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
89174 DCC01	Forela-Wkstn002.local	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
89862 Forela-Wkstn002.local	DCC01	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
78289 Forela-Wkstn002.local	DCC01	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
81675 DCC01	Forela-Wkstn002.local	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
82773 Forela-Wkstn002.local	DCC01	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
12200 Forela-Wkstn002.local	DCC01	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
12756 DCC01	Forela-Wkstn002.local	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
13247 Forela-Wkstn002.local	DCC01	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
40062 Forela-Wkstn002.local	DCC01	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
40367 DCC01	Forela-Wkstn002.local	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
41033 Forela-Wkstn002.local	DCC01	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
59380 Forela-Wkstn002.local	DCC01	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
61164 DCC01	Forela-Wkstn002.local	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE

One thing to note is that DCC01 is not an actual host in the network, Wireshark decoded the Attacker's Kali hostname as DCC01 in smb traffic because the victim machine (Forela-Wkstn002.local) thinks the Kali machine is the DCC01 host (the typo the victim made).

This confirms our findings so far that the credentials got stolen by the attacker from Wkstn002.

Let's again filter for smb2 traffic and see the file share the victim wanted to navigate to.

dc01.forela.local	Forela-Wkstn002.forela.local	SMB2	138	Write Response
Forela-Wkstn002.forela.local	dc01.forela.local	SMB2	171	Read Request Len:1024 Off:0 File: srvsvc
dc01.forela.local	Forela-Wkstn002.forela.local	SRVSVC	950	NetShareEnumAll response[Malformed Packet]
Forela-Wkstn002.forela.local	dc01.forela.local	SMB2	146	Close Request File: srvsvc
dc01.forela.local	Forela-Wkstn002.forela.local	SMB2	182	Close Response
Forela-Wkstn002.forela.local	dc01.forela.local	SMB2	174	Tree Connect Request Tree: \\DC01\DC-Confidential
dc01.forela.local	Forela-Wkstn002.forela.local	SMB2	138	Tree Connect Response
Forela-Wkstn002.forela.local	dc01.forela.local	SMB2	234	Create Request File:
dc01.forela.local	Forela-Wkstn002.forela.local	SMB2	298	Create Response File:
Forela-Wkstn002.forela.local	dc01.forela.local	SMB2	146	Close Request File:
dc01.forela.local	Forela-Wkstn002.forela.local	SMB2	182	Close Response
Forela-Wkstn002.forela.local	dc01.forela.local	SMB2	174	Tree Connect Request Tree: \\DC01\DC-Confidential

In smb traffic we see that the user from wkstn2 navigated to a file share on DC01. Now the whole scenario makes more sense. Let's go over again to what happened.

- The victim wanted to visit \\DC01\DC-Confidential but instead made a typo (\\DCC01\DC-Confidential) which failed the hostname resolution.
- When the DNS server was not able to resolve DCC01 (as it does not exist), Windows tried using LLMNR to resolve this.

AD Incident Response

- Our crafty attacker was running a tool called Responder in the network which acts as an LLMNR server and it responded to the LLMNR query by the victim machine.
- The victim's machine sent the hash to authenticate thinking it was a domain controller, but in reality, it was an attacker-controlled server.
- On the victim's side, nothing seems out of the ordinary. When file share does not open, victim realises they made a typo and they type in the correct path and carry on as usual. In our case victim fixed the typo and navigated to DC01 share after realizing their typo

We can see all above activity occurred in under a minute.

Recovering & cracking the user hash

This step is important because it allows us to determine if the attacker cracked the compromised user hash.

If the password policy is strict in your environment, there's a high chance the attacker failed to crack the password. The real impact of this attack is therefore reduced thanks to the complex password policy.

Go to NTLM authentication traffic (ntlmssp filter). The NTLM authentication and negotiation occur in sets of three (3) packets. Each negotiation includes an:

1. NTLMSSP_NEGOTIATE packet.
2. NTLMSSP_CHALLENGE packet.
3. NTLMSSP_AUTH packet.

Length	Info	
240	Session Setup Request, NTLMSSP_NEGOTIATE	Set 1
412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE	
717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon	
240	Session Setup Request, NTLMSSP_NEGOTIATE	Set 2
412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE	
717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon	
240	Session Setup Request, NTLMSSP_NEGOTIATE	Set 3
412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE	
717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon	

To recover the hash, we need to dissect 1 set. Let's do this on the first set. We need different values from the negotiations and to plug in the values in this format.

Format

User::Domain:ServerChallenge:NTPProofStr:NTLMv2Response(without first 16 bytes/32 characters).

To find each value:

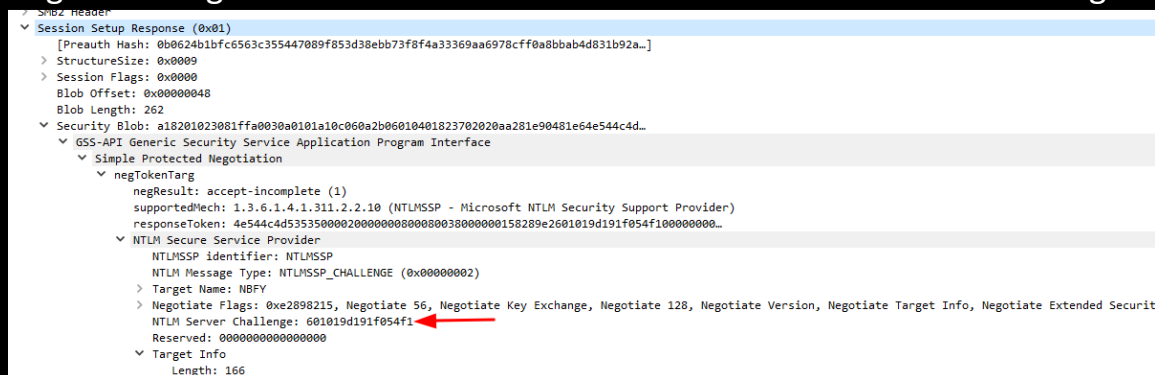
User: In details of NTLMSSP_NEGOTIATE packet expand SMB2 (Server Message Block Protocol Version 2) ->SMB2 Header -> Session ID->Account.



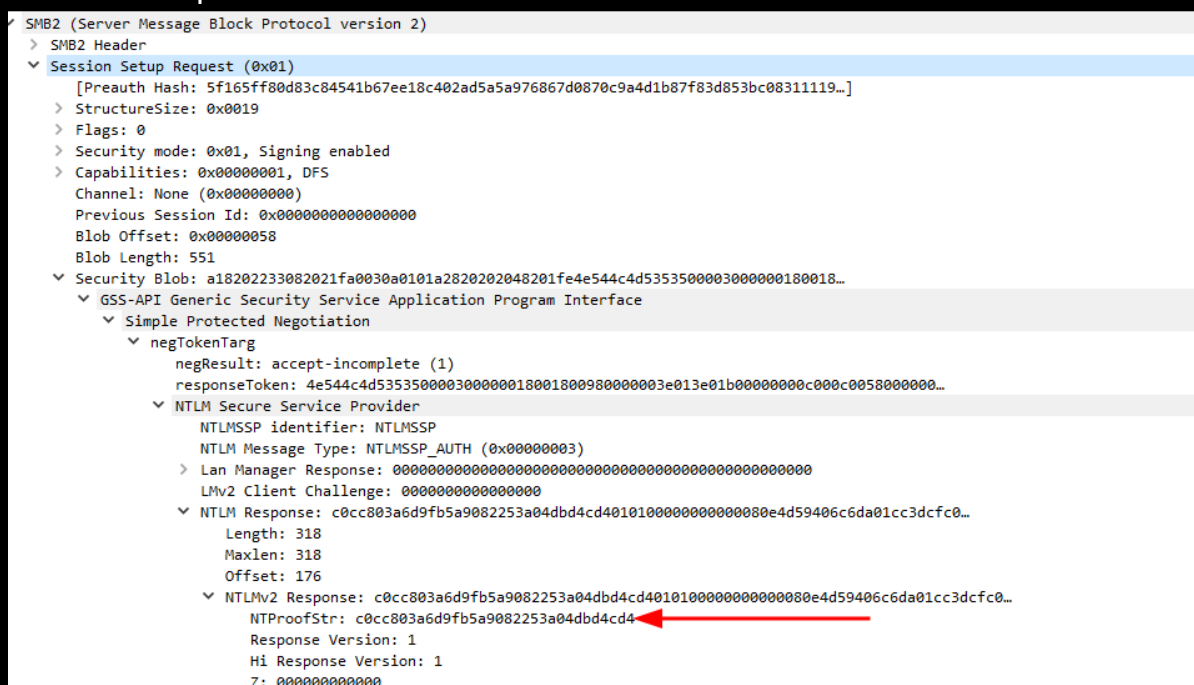
Domain: In details of NTLMSSP_NEGOTIATE packet expand SMB2 (Server Message Block Protocol Version 2) ->SMB2 Header -> Session ID->Domain.



ServerChallenge: In details of the NTLMSSP_CHALLENGE PACKET expand SMB2 (Server Message Block Protocol Version 2) -> Session Setup Response (0x1) -> Security Blob -> GSS-API Generic -> Simple Protected Negotiation -> negTokenTarg -> NTLM Secure Service Provider -> NTLM Server Challenge.



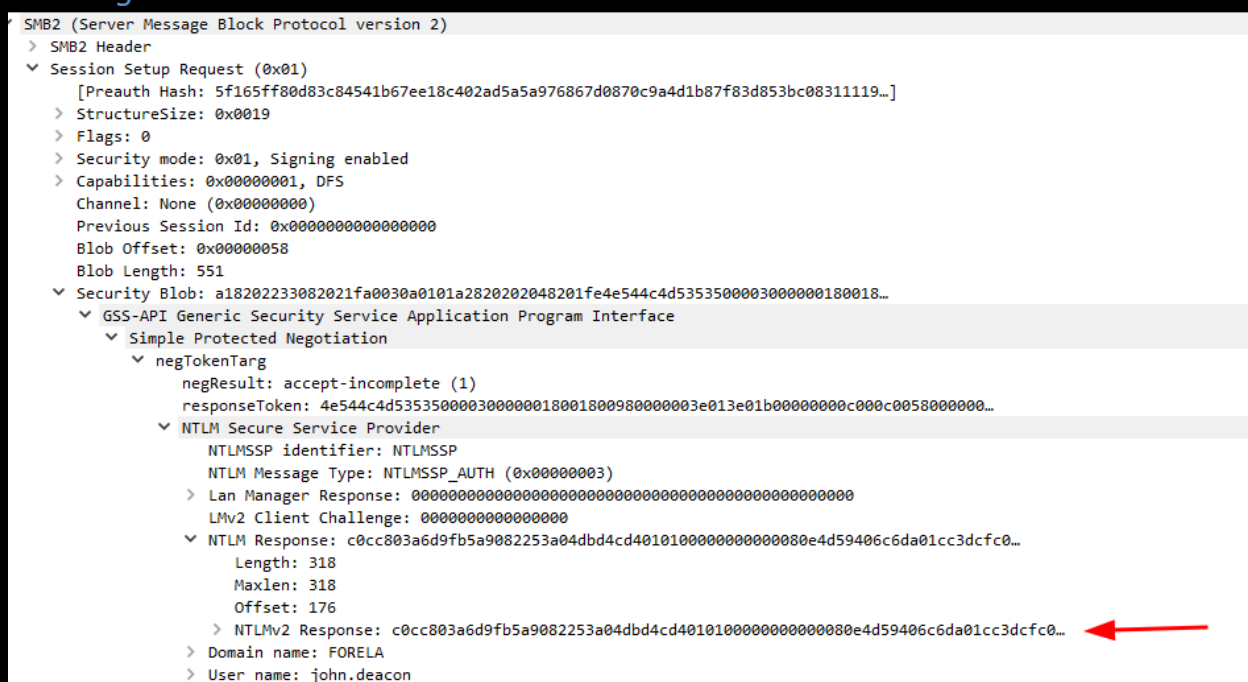
NtProofStr: In details of the NTLMSSP_AUTH Packet expand SMB2 (Server Message Block Protocol Version 2) -> Session Setup Response (0x1) -> Security Blob -> GSS-API Generic *** -> Simple Protected Negotiation -> negTokenTarg -> NTLM Secure Service Provider -> -> NTLM Response -> NTLMv2 Response -> NTPProofStr.



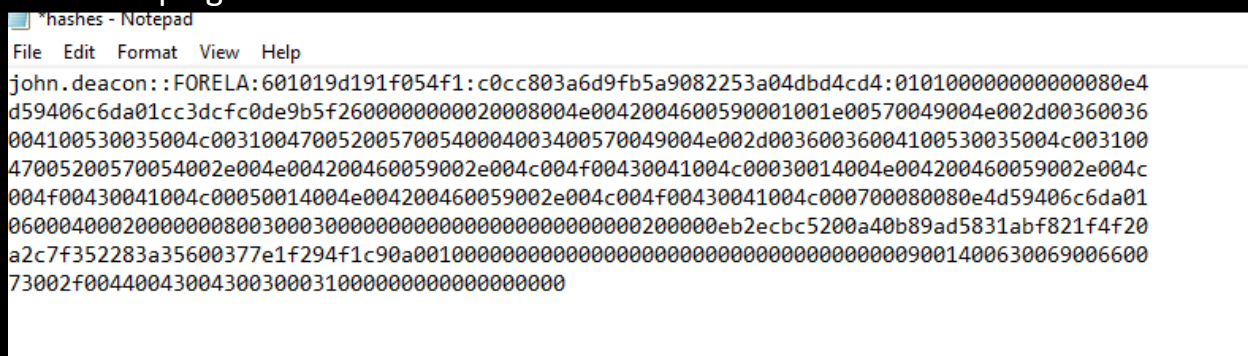
AD Incident Response

NTLMV2Response: In details of the NTLMSSP_AUTH Packet expand SMB2 (Server Message Block Protocol Version 2) -> Session Setup Response (0x1) -> Security Blob -> GSS-API Generic **** -> Simple Protected Negotiation -> negTokenTarg -> NTLM Secure Service Provider -> -> NTLM Response -> NTLMv2 Response.

Note: You have to remove first 32 characters from the value of this field because the first 32 characters are the same as NTPProofStr, which we already have acquired and is ready to be plugged in the hash format for cracking.



Now let's plug in the values in the format.



This is the final format we get and we save this a txt file.

Now let's get cracking. We will use Hashcat to crack this.

Hashcat Format: Hashcat.exe -a0 -m5600 hash.txt passwords.txt.

In this format, hash.txt contains our hash format which we plugged in and passwords.txt is our wordlist.

```
Stopped: Tue Jul 09 09:49:11 2024

C:\Users\pc\Downloads\hashcat-6.2.6>hashcat.exe -a0 -m5600 H:\Project-Sherlock\Toxic\hash.txt H:\Project-Sherlock\Toxic\password.txt
hashcat (v6.2.6) starting

Successfully initialized the NVIDIA main driver CUDA runtime library.
Failed to initialize NVIDIA RTC library.

ssion.....: hashcat
atus.....: Exhausted
sh.Mode.....: 5600 (NetNTLMv2)
sh.Target.....: JOHN.DEACON::FORELA:601019d191f054f1:c0cc803a6d9fb5...000000
me.Started.....: Tue Jul 09 09:49:43 2024 (0 secs)
me.Estimated...: Tue Jul 09 09:49:43 2024 (0 secs)
rnel.Feature...: Pure Kernel
ess.Base.....: File (H:\Project-Sherlock\Toxic\password.txt)
ess.Queue.....: 1/1 (100.00%)
eed.#1.....: 1012 H/s (0.09ms) @ Accel:1024 Loops:1 Thr:64 Vec:1
covered.....: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
ogress.....: 1/1 (100.00%)
jected.....: 0/1 (0.00%)
store.Point....: 1/1 (100.00%)
store.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
ndidate.Engine.: Device Generator
ndidates.#1...: NotMyPassword0K? -> NotMyPassword0K?
rdware.Mon.#1...: Temp: 53c Fan: 34% Util: 29% Core:1770MHz Mem:9501MHz Bus:16
```

So what we've discussed so far:

- 1- In LLMNR traffic, look for any machine responding to queries that is not a domain controller.
- 2- Look for NTLM authentication packets going towards the unidentified/unknown machine discovered from point 1.
- 3- Look for typos in LLMNR traffic, DNS Traffic, and SMB traffic.
- 4- Try cracking the hash to see how resilient the password of the compromised user is.

- 5- As a follow up, look for authentications for the compromised user in the environment, especially after the time of LLMNR poisoning, and search for logon types three (3) (network logon) and 10 (RDP Logon).

Part four: NTLM relay attack detection

How to detect NTLM relay attacks critical Active Directory (AD) attack detections & misconfigurations.

NTLM relay attacks explained

Windows New Technology LAN Manager (NTLM) is the name of a set of Active Directory protocols that provide authentication in the network. To ensure backward compatibility, NTLM is used mainly on systems that do not support Kerberos authentication.

When an attacker intercepts network traffic with an LLMNR poisoning attack, they can further attempt to relay the intercepted event to authenticate themselves to a particular service on behalf of the victim.

This is known as an NTLM relay attack. NTLM relay attacks are possible because the NTLM itself does not provide session security.

[Impacket's ntlmrelayx](#) is one of the most commonly used tools that can be leveraged to perform this attack.

Detecting NTLM relay attacks

To detect NTLM relay activity in a network we need network telemetry and logon audit logs from the endpoint.

Detecting NTLM relay attacks requires an odd approach, as we need to correlate the IP Addresses with the host names.

This means we need to have an inventory/list of IP addresses assigned to workstations. In big corporate environments with thousands of assets in the network, keeping tabs on this is difficult.

In such cases, network telemetry will help us determine the IP addresses of legitimate devices and make it easier to pinpoint the rogue machine.

Let's start off by opening a pcap file in Wireshark and listing the known IP addresses associated with domain-joined workstations and servers.

AD Incident Response

Viewing the endpoints included in the network traffic we can see few internal IP addresses.

Ethernet · 15		IPv4 · 19		IPv6 · 7		TCP · 56		UDP · 71	
Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Country		
4.241.155.66	6	425 bytes	3	245 bytes	3	180 bytes			
20.198.119.143	6	620 bytes	3	345 bytes	3	275 bytes			
34.107.243.93	3	220 bytes	2	138 bytes	1	82 bytes			
34.117.188.166	33	9 kB	17	5 kB	16	4 kB			
172.17.79.1	92	8 kB	84	7 kB	8	726 bytes			
172.17.79.2	66	9 kB	12	3 kB	54	6 kB			
172.17.79.4	656	181 kB	314	88 kB	342	93 kB			
172.17.79.129	576	148 kB	314	74 kB	262	74 kB			
172.17.79.135	197	32 kB	111	15 kB	86	16 kB			
172.17.79.136	322	61 kB	176	35 kB	146	25 kB			
172.17.79.137	1	62 bytes	0	0 bytes	1	62 bytes			
172.17.79.254	6	1 kB	3	746 bytes	3	487 bytes			
172.17.79.255	8	857 bytes	0	0 bytes	8	857 bytes			
204.79.197.239	1	60 bytes	1	60 bytes	0	0 bytes			
224.0.0.22	70	4 kB	0	0 bytes	70	4 kB			
224.0.0.251	9	689 bytes	0	0 bytes	9	689 bytes			
224.0.0.252	7	441 bytes	0	0 bytes	7	441 bytes			
239.255.255.250	19	4 kB	0	0 bytes	19	4 kB			
255.255.255.255	2	653 bytes	0	0 bytes	2	653 bytes			

The IP addresses ending with .1 and .2 are excluded because they act as gateways and are not assigned to actual workstations.

We can get host names for the IP addresses using NetBios Name service/NBNS Protocol by filter for this in Wireshark.

No.	Time	Source
3	0.001463822	172.17.79.129
28	1.499287702	172.17.79.129
294	3.014968616	172.17.79.129
579	4.471329674	172.17.79.135
580	4.471779579	172.17.79.1
585	4.472229222	172.17.79.135

And for the IP address 172.17.79.129:

[illegible]

AD Incident Response

And for IP Address 172.17.79.136:

26	172.17.79.136	172.17.79.2	NBNS	110 Refresh NB FORELA-WKSTN002<20>
3	172.17.79.136	172.17.79.2	NBNS	110 Refresh NB FORELA-WKSTN002<20>
67	172.17.79.136	172.17.79.2	NBNS	110 Refresh NB FORELA-WKSTN002<20>
62	172.17.79.136	172.17.79.2	NBNS	110 Refresh NB FORELA-WKSTN002<00>
88	172.17.79.136	172.17.79.2	NBNS	110 Refresh NB FORELA-WKSTN002<00>
2	172.17.79.136	172.17.79.2	NBNS	110 Refresh NB FORELA-WKSTN002<00>

For IP address 172.17.79.135 we can see there is no host name and instead some random data.

172.17.79.135	172.17.79.135	NBNS	199 Name query response NBSTAT
172.17.79.135	172.17.79.129	NBNS	92 Name query NBSTAT *<00><00><00><00><00><00><00><00><00><00>
172.17.79.129	172.17.79.135	NBNS	100 Name query response NBSTAT*
172.17.79.135	172.17.79.4	NBNS	92 Name query NBSTAT *<00><00><00><00><00><00><00><00><00><00>
172.17.79.4	172.17.79.135	NBNS	235 Name query response NBSTAT

So now we have a short list of hostnames:

1. Forela-Wkstn001 is assigned 172.17.79.129.
2. Forela-Wkstn002 is assigned 172.17.79.136.
3. Unknown Device is assigned 172.17.79.135.

Now, let's add a filter for smb traffic to and from the unknown device since we suspect it to be the attacker's machine acting as a Man In The Middle (MITM).



Source	Destination	Protocol	Length	Info
172.17.79.135	172.17.79.1	SMB2	250	Negotiate Protocol Request
172.17.79.1	172.17.79.135	SMB2	306	Negotiate Protocol Response
172.17.79.135	172.17.79.129	SMB2	238	Negotiate Protocol Request
172.17.79.129	172.17.79.135	SMB2	366	Negotiate Protocol Response
172.17.79.135	172.17.79.4	SMB2	250	Negotiate Protocol Request
172.17.79.4	172.17.79.135	SMB2	378	Negotiate Protocol Response
172.17.79.135	172.17.79.135	SMB2	228	Negotiate Protocol Response
172.17.79.136	172.17.79.135	SMB2	220	Session Setup Request, NTLMSSP_NEGOTIATE
172.17.79.135	172.17.79.135	SMB2	347	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
172.17.79.136	172.17.79.135	SMB2	635	Session Setup Request, NTLMSSP_AUTH, User: FORELA\arthur.kyle
172.17.79.135	172.17.79.136	SMB2	139	Session Setup Response
172.17.79.136	172.17.79.135	SMB2	146	Tree Connect Request Tree: \\D\IPC\$
172.17.79.129	172.17.79.135	SMB2	306	Negotiate Protocol Response
172.17.79.135	172.17.79.129	SMB2	164	Negotiate Protocol Request
172.17.79.129	172.17.79.135	SMB2	306	Negotiate Protocol Response
172.17.79.135	172.17.79.136	SMB2	138	Tree Connect Response, Error: STATUS_NETWORK_SESSION_EXPIRED
172.17.79.136	172.17.79.135	SMB2	186	Session Setup Request, NTLMSSP_NEGOTIATE
172.17.79.135	172.17.79.129	SMB2	186	Session Setup Request, NTLMSSP_NEGOTIATE
172.17.79.129	172.17.79.135	SMB2	380	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
172.17.79.135	172.17.79.136	SMB2	380	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
172.17.79.136	172.17.79.135	SMB2	664	Session Setup Request, NTLMSSP_AUTH, User: FORELA\arthur.kyle
172.17.79.135	172.17.79.129	SMB2	664	Session Setup Request, NTLMSSP_AUTH, User: FORELA\arthur.kyle

In the above image, we can see that user “arthur.kyle” was involved in the authentication process and the unknown device is involved in this as well.

We see the IP Address for the Wkstn002 machine as well. Seeing the traffic direction, user arthur.kyle is the user account in Wkstn002. This now screams that the credentials were stolen and relayed from the user arthur.kyle from workstation 2.

The next stream of packets now involve the same unknown device in the authentication process with the 172.17.79.129 IP address. This is another red flag as to why the same user account (arthur.kyle) is involved in authentication from two different machines (one is legitimate and the other is an unknown MITM device).

AD Incident Response

These packets tell us that:

1. Arthur Kyle on Wkstn002 had hashes stolen by the attacker. One way this can happen is if the user had a typo when navigating to file share and the attacker was running responder to intercept traffic in the environment. (We'll discuss this later.)
2. The hashes stolen by the attacker were then immediately relayed to the target system, which in our case, is Wkstn001.
3. The attacker then authenticates and logs in to the target machine (Wkstn001 using the arthur.kyle account) and then dumps the SAM Hashes from the remote machine. This is the default behavior of [ntlmrelayx](#) tool from impacket used for this purpose. After this, they can log on to the machine with the primary user account of that machine whose hash was dumped remotely.

Now let's Investigation from **Event log analysis**

So look at log events here, filtering for Event ID 4624.

Filtered: Log: file://H:\Project-Sherlock\Reaper\Security.evtx; Source: ; Event ID: 4624. Number of events: 11

Level	Date and Time	Source	Event ID	Task Category
Information	7/31/2024 10:04:39 AM	Microsoft Windows security audit...	4624	Logon
Information	7/31/2024 9:59:57 AM	Microsoft Windows security audit...	4624	Logon
Information	7/31/2024 9:59:57 AM	Microsoft Windows security audit...	4624	Logon
Information	7/31/2024 9:59:31 AM	Microsoft Windows security audit...	4624	Logon
Information	7/31/2024 9:59:31 AM	Microsoft Windows security audit...	4624	Logon
Information	7/31/2024 9:55:39 AM	Microsoft Windows security audit...	4624	Logon
Information	7/31/2024 9:55:39 AM	Microsoft Windows security audit...	4624	Logon
Information	7/31/2024 9:55:39 AM	Microsoft Windows security audit...	4624	Logon
Information	7/31/2024 9:55:16 AM	Microsoft Windows security audit...	4624	Logon
Information	7/31/2024 9:54:49 AM	Microsoft Windows security audit...	4624	Logon
Information	7/31/2024 9:54:48 AM	Microsoft Windows security audit...	4624	Logon

An account was successfully logged on.

Subject:

Security ID: NULL SID
Account Name: -
Account Domain: -
Logon ID: 0x0

Logon Information:

Logon Type: 3
Restricted Admin Mode: -
Virtual Account: No
Elevated Token: No

Impersonation Level: Impersonation

New Logon:

Security ID: S-1-5-21-3239415629-1862073780-2394361899-1601
Account Name: arthur.kyle
Account Domain: FORELA
Logon ID: 0x64A799
Linked Logon ID: 0x0
Network Account Name: -
Network Account Domain: -
Logon GUID: {00000000-0000-0000-0000-000000000000}

Process Information:

Process ID: 0x0
Process Name: -

Network Information:

Workstation Name: FORELA-WKSTN002
Source Network Address: 172.17.79.135
Source Port: 40252

Detailed Authentication Information:

Logon Process: NtLmSsp
Authentication Package: NTLM
Transited Services: -
Package Name (NTLM only): NTLM V2

In this specific event, we have multiple indicators that an NTLM relay attack did occur and authentication was conducted from the attacker's machine using stolen credentials.

The first indicator we see is that Security ID Says it is NULL SID and we have no Logon ID, LogonGUID is also null.

In network information, we can see that the source workstation says WKSTN002. In IP address we see 172.17.79.135. (Suspicious!)

Workstation002 has an IP address of 172.17.79.136, but it shows .135, which was an unknown device.

Another indicator we see is that the log-on process says it's NTLMSSP, which we confirmed from our network traffic analysis.

That's means and confirms the unknown device stole credentials from wkstn002 and used them to log on to wkstn001.

Part Five: NTDS dumping attack detection

How to detect NTDS dumping attacks of critical Active Directory (AD) attack detections & misconfigurations.

NTDS password extraction

Active Directory stores domain information in the [NTDS.dit file](#), which is located by default in `%SystemRoot%\ntds\` on the domain controller.

This file contains crucial domain information, including password hashes for users, making it a very desirable target for attackers.

To gain access to the NTDS.dit file the attacker must already have administrator access in the environment. If the attacker has access to the domain controller, they can exfiltrate the NTDS.dit file alongside the HKEY_LOCAL_MACHINE\SYSTEM registry hive, which contains all the information needed to decrypt the NTDS.dit data.

Although Active Directory locks this file while running (disallowing any copy activities), an attacker can use the [Volume Shadow Copy Service \(VSS\)](#) to copy the volume and extract the NTDS.dit file from the snapshot.

AD Incident Response

They could also make a copy using a diagnostic tool available as part of Active Directory, NTDSUTIL.exe.

Furthermore, if an attacker has a set of valid credentials, they could leverage tools like [crackmapexec](#) to dump the NTDS.dit file remotely and parse it.

After an attacker exfiltrates the NTDS.dit file and the [HKLM\SYSTEM](#) registry hive, they can perform subsequent attacks offline, and do not require access to AD.

After obtaining the password hashes from the NTDS.dit file, attackers can attempt to crack them offline to obtain the plaintext passwords. If they are unable to crack the hashes offline, they could also try using the password hashes in [pass-the-hash attacks](#) to further exploit the environment.

The process of parsing the domain information from those files can be done with tools like secretdump, which is part of the Impacket tool suite.

Note: OS Credential Dumping NTDS.dit is mapped to the [sub-technique T1003.003](#) on the MITRE ATT&CK framework.

Detection

Let's discuss how to detect NTDS.dit dumping done via NTDSUTIL.exe Utility.

NTDSUTIL is a built-in utility available for the management of the NTDS database in Windows servers, but an attacker can exploit it to gain access to this database.

However, doing this leaves few indicators of the attack that we can use to establish that NTDS.dit was dumped by a malicious entity.

Application, system & security logs

To examine this, we will use application and system logs from the Domain Controller.

We need to monitor for [Event ID 325 and 327](#) and the event source "ESENT" in application logs. These events are logged when a new database is created and when a database is detached respectively.

AD Incident Response

We will also look for event ID 7036 in the system log to correlate with our application logs findings. For our final stop, we will look for event ID 4799 in the Security logs. Let's start with the application logs.

Filter Current Log

Filter XML

Logged: Any time

Event level: ☐ Critical ☐ Warning ☐ Verbose ☐ Error ☐ Information

☒ By log Event logs: file:///H:/Project-Sherlock/CrownJewel 2/Artifa

☐ By source Event sources: ESENT

Includes/Excludes Event IDs: Enter ID numbers and/or ID ranges separated by commas. To exclude criteria, type a minus sign first. For example 1,3,5-99,-76

325,327

Task category:

Keywords:

User: <All Users>

Computer(s): <All Computers>

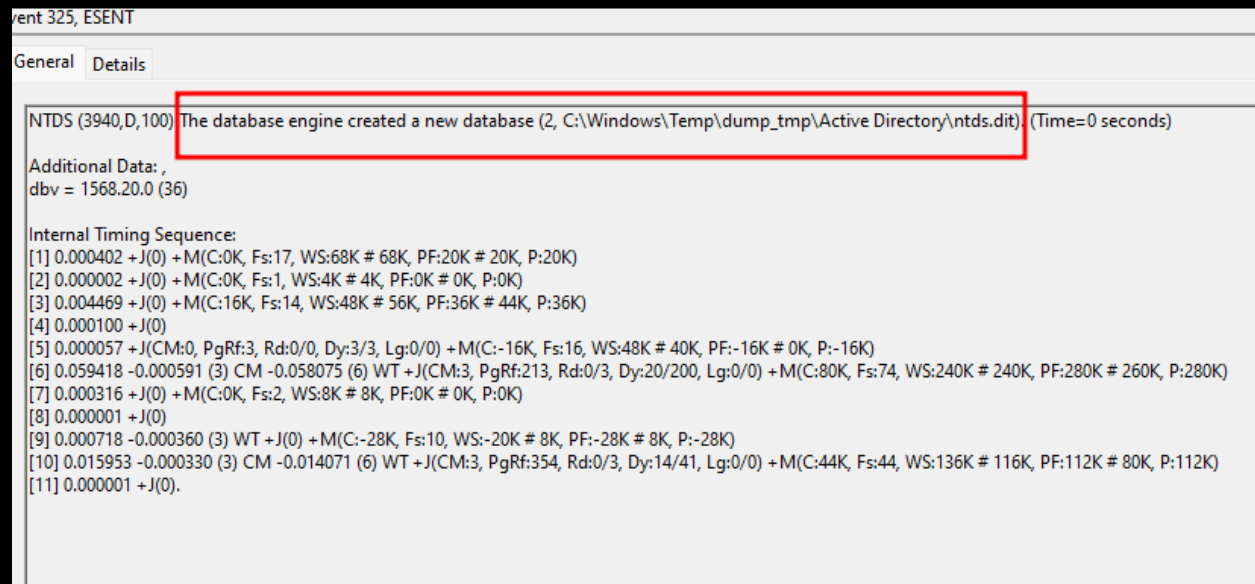
Hints of compromise include:

1- Application Log EventID 325:

We need to look for any weird file path for the newly created database (such as a dumped copy of the original NTDS.dit) besides its original path of %SystemRoot%\ntds\.

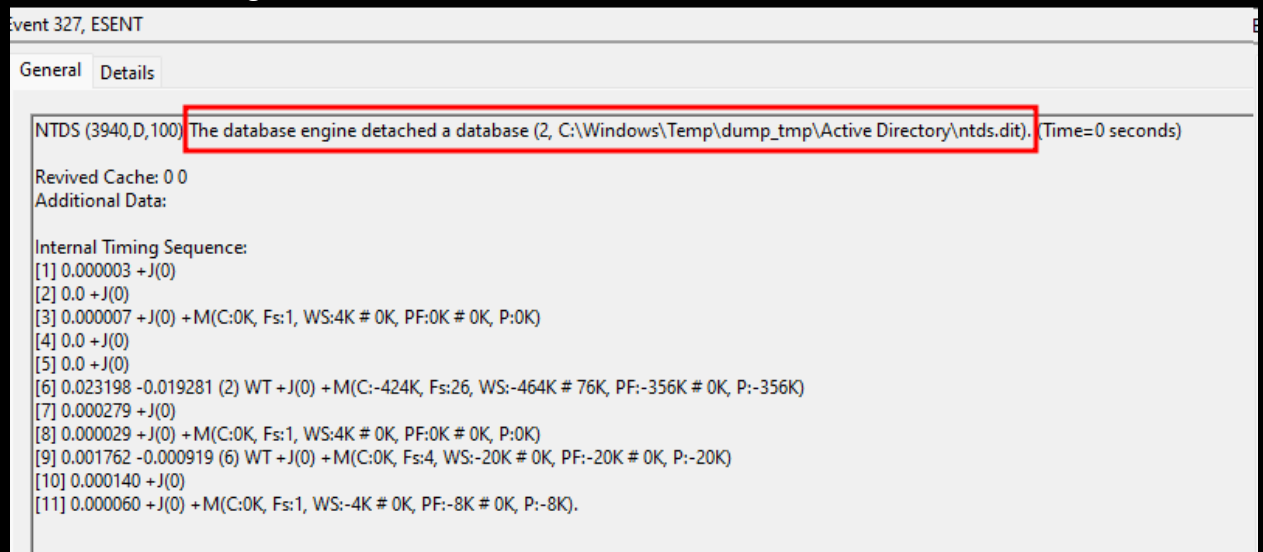
Any copy of NTDS.dit present in another location strongly indicates malicious behavior.

We can see an example of a suspicious log below:



2- Application Log EventID 327:

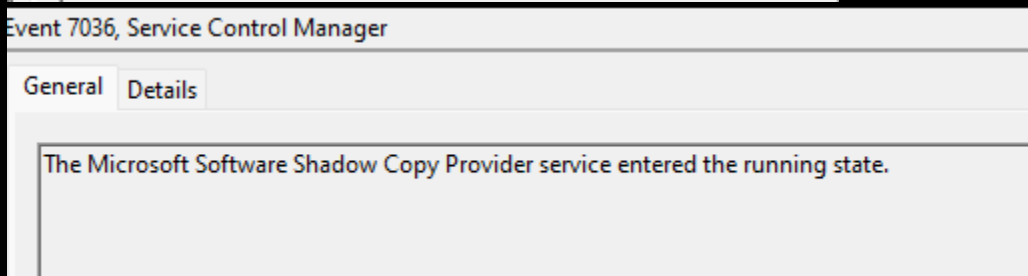
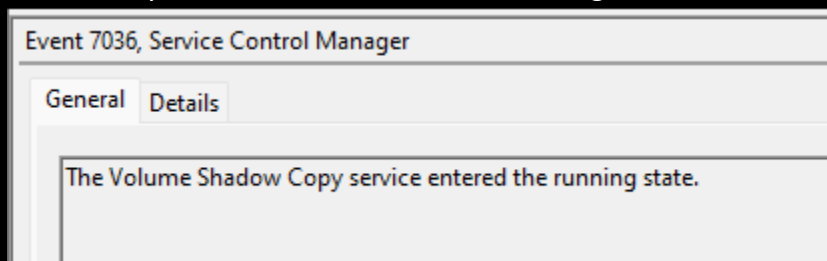
Immediately after the previous event, we would see [event 327](#), with the same NTDS.dit path and a message saying that the database was detached by the engine. From these two logs, we can create a timeline of the events.



3- System Log EventID 7036:

When we look for the 7036 events around the timestamps that we established from our application logs findings, we see two services related to Volume shadow, which started running just a second before the creation of the NTDS.dit database.

This activity would further confirm our findings so far.

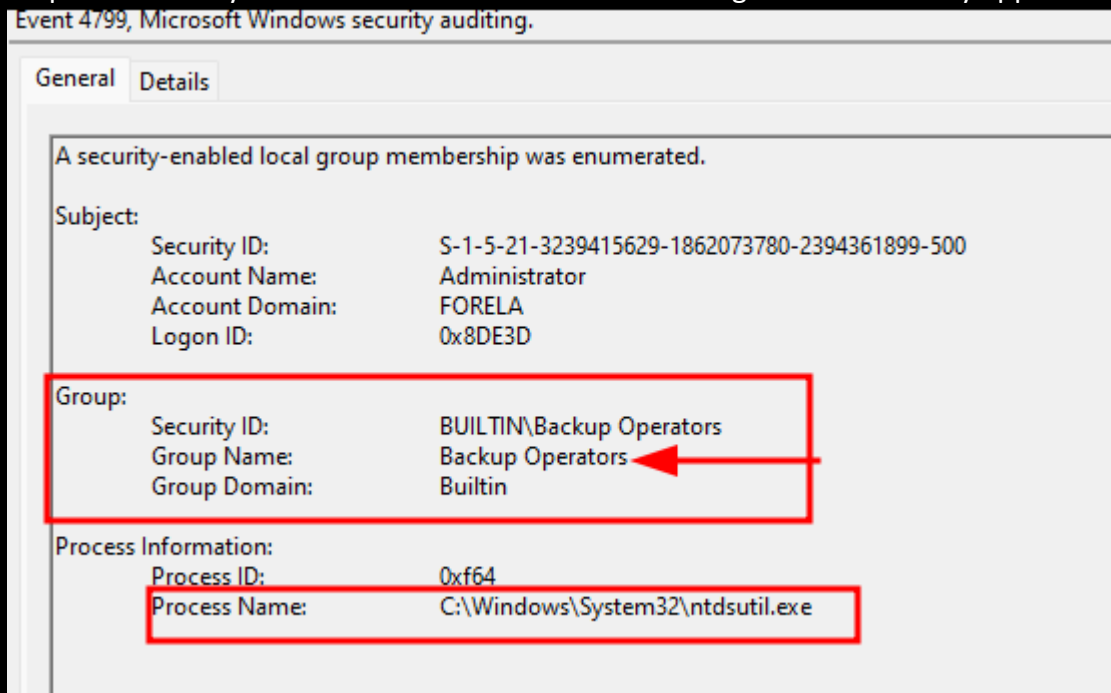


4- Security Log EventID 4799:

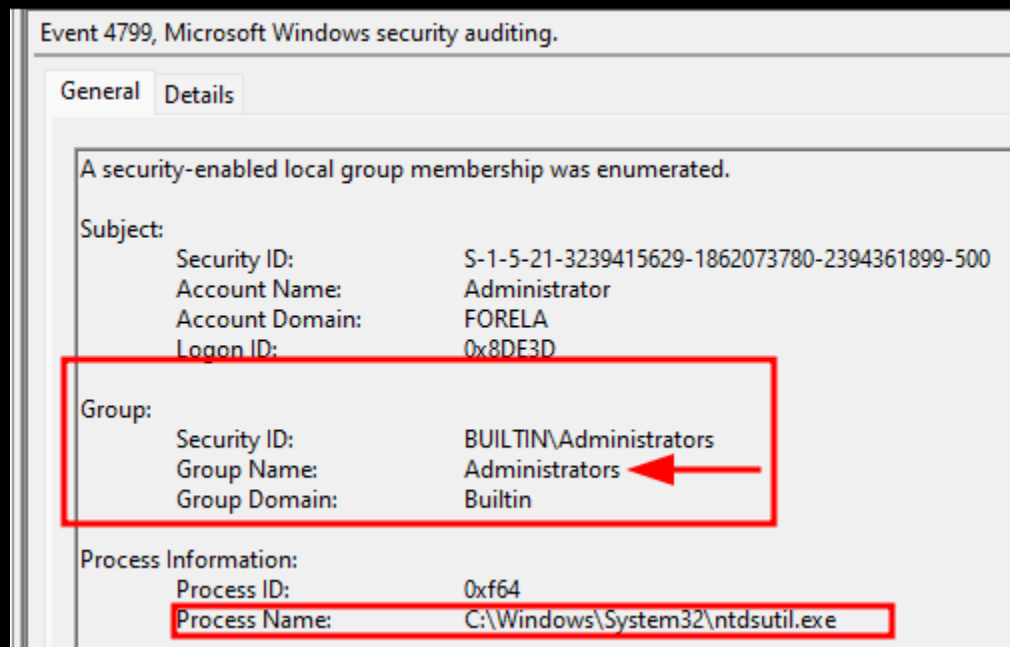
Next, filter for the event ID and look for the events in the established timeline.

We need to look for an event where two security groups (**Backup Operators** and **Administrator**) were being enumerated by the ntdsutil.exe process multiple times (around 50+ times in 1-2 seconds).

This is a strong indicator and can be used to validate our findings and establish that suspicious activity occurred. Let's take a look at some logs to see how they appear.



AD Incident Response



we can see in the figure below, there are lots of events in under a second.

[illegible]

References

- 1- <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse>
- 2- https://adsecurity.org/?page_id=4031
- 3- <https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/best-practices-for-securing-active-directory>
- 4- <https://www.blackhat.com/docs/us-15/materials/us-15-Metcalf-Red-Vs-Blue-Modern-Active-Directory-Attacks-Detection-And-Protection.pdf>