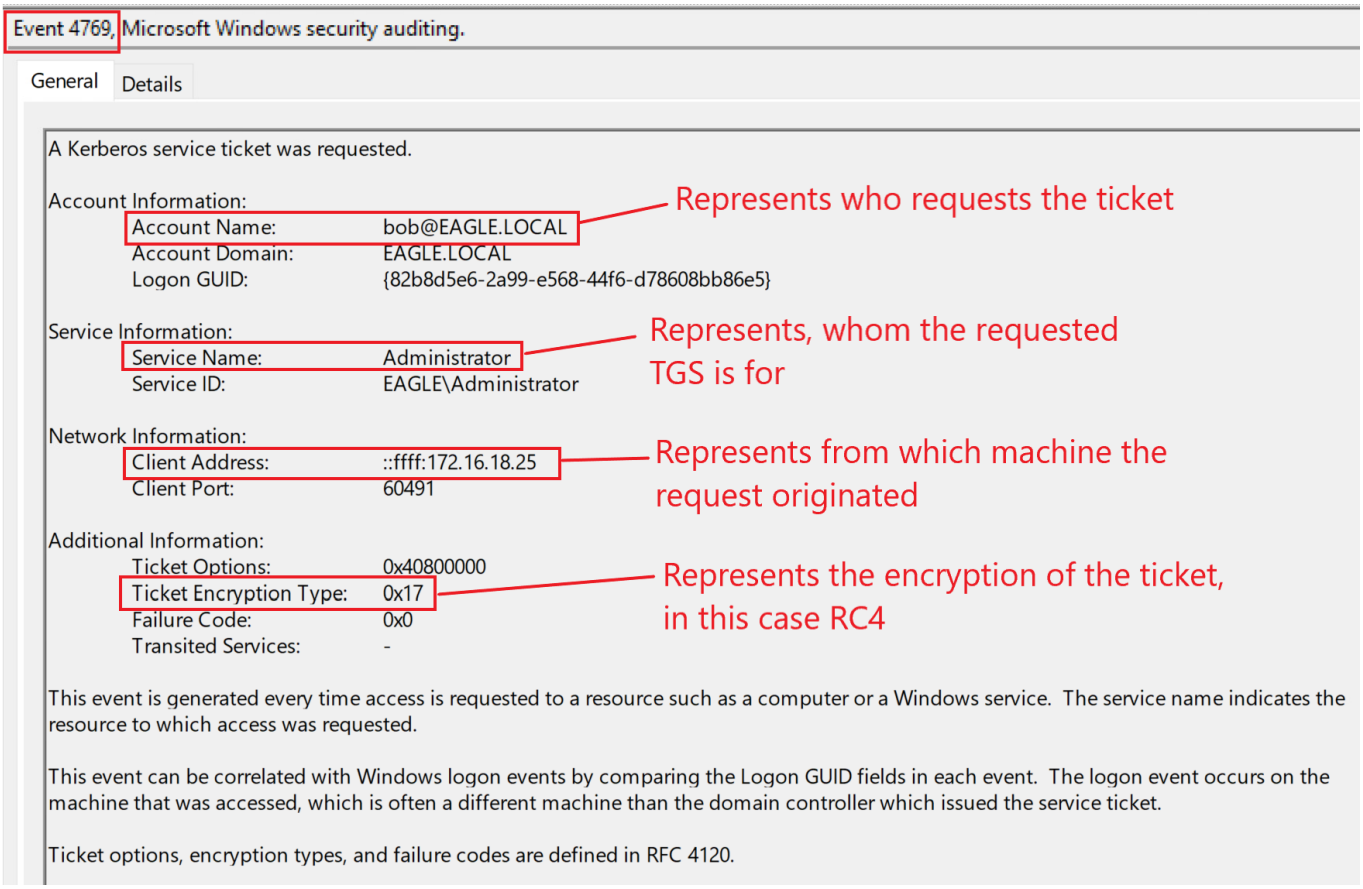
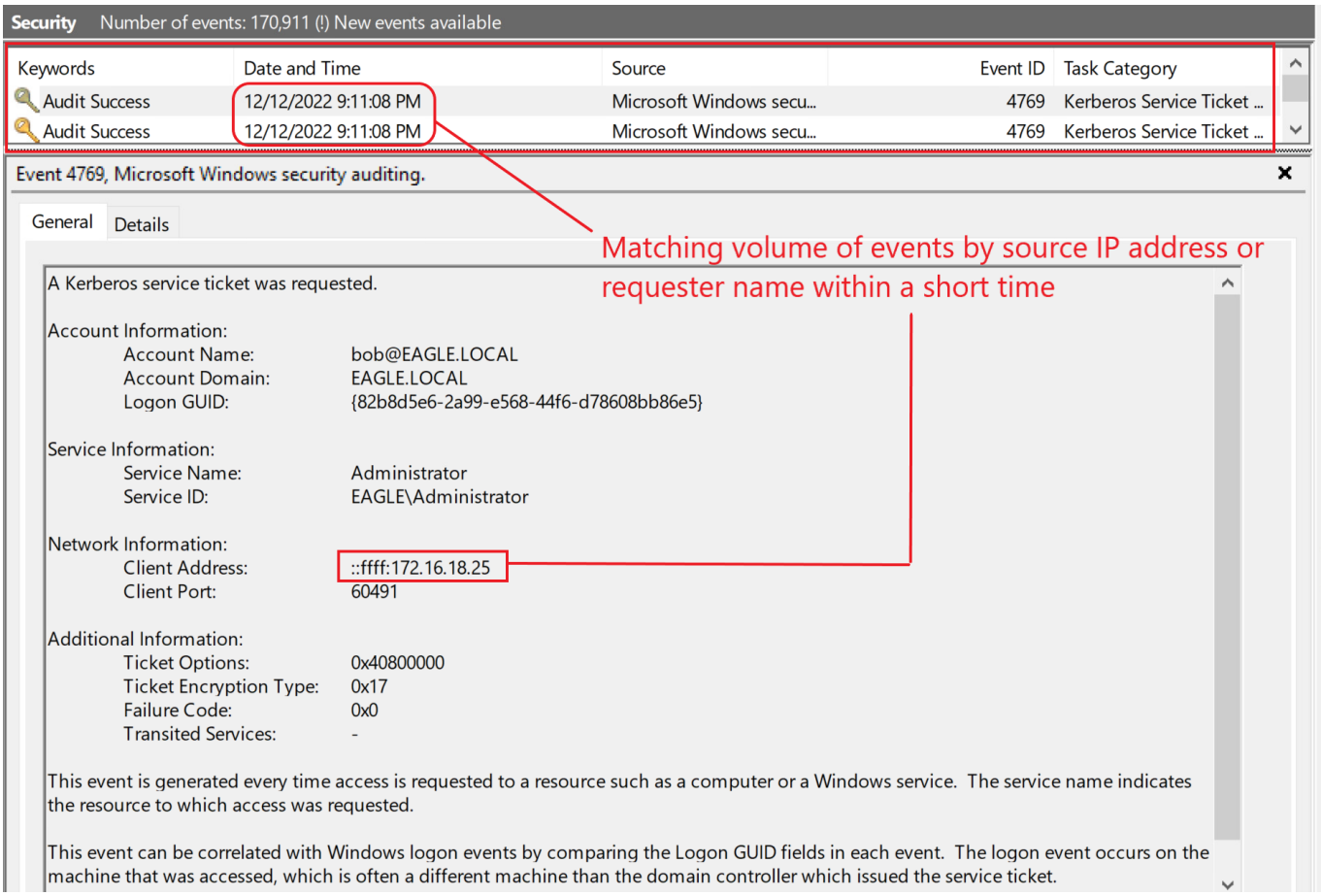
Kerberoasting

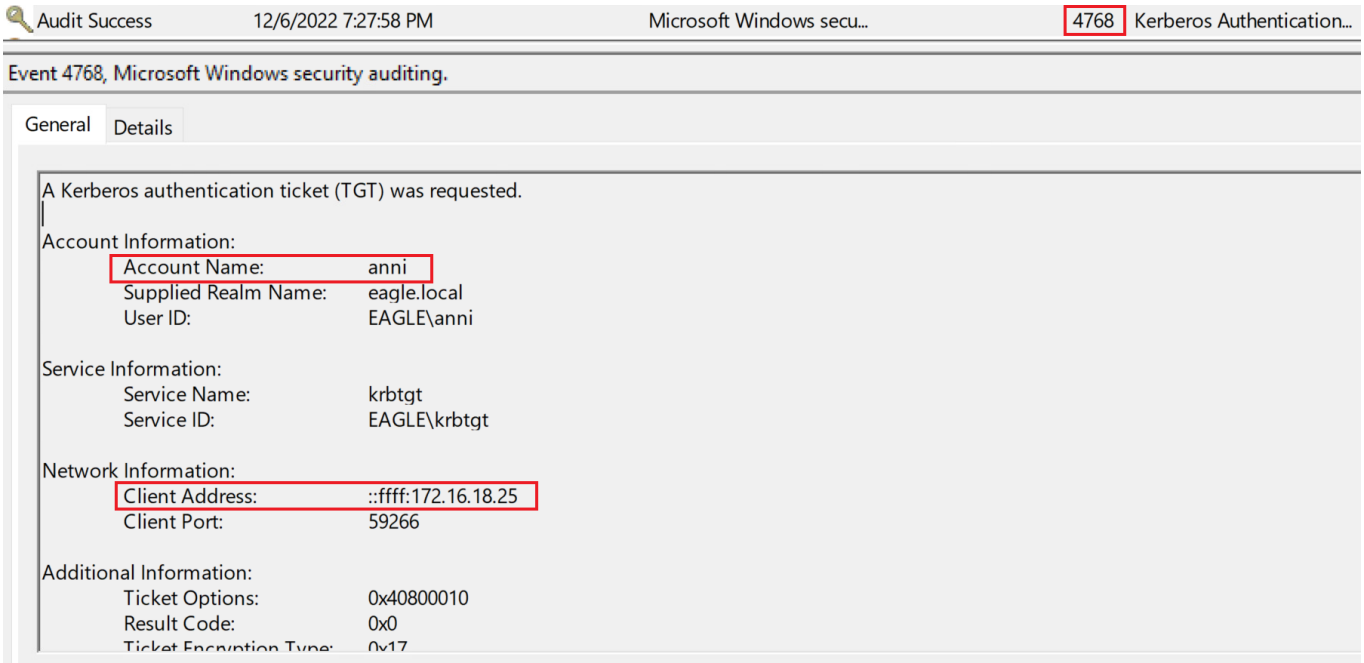
When a `TGS` is requested, an event log with ID `4769` is generated. However, AD also generates the same event ID whenever a user attempts to connect to a service, which means that the volume of this event is gigantic, and relying on it alone is virtually impossible to use as a detection method. If we happen to be in an environment where all applications support AES and only AES tickets are generated, then it would be an excellent indicator to alert on event ID `4769`. If the ticket options is set for `RC4`, that is, if `RC4` tickets are generated in the AD environment (which is not the default configuration), then we should alert and follow up on it. Here is what was logged when we requested the ticket to perform this attack:





AS-REProasting

When we executed Rubeus, an Event with ID `4768` was generated, signaling that a `Kerberos Authentication ticket` was generated:

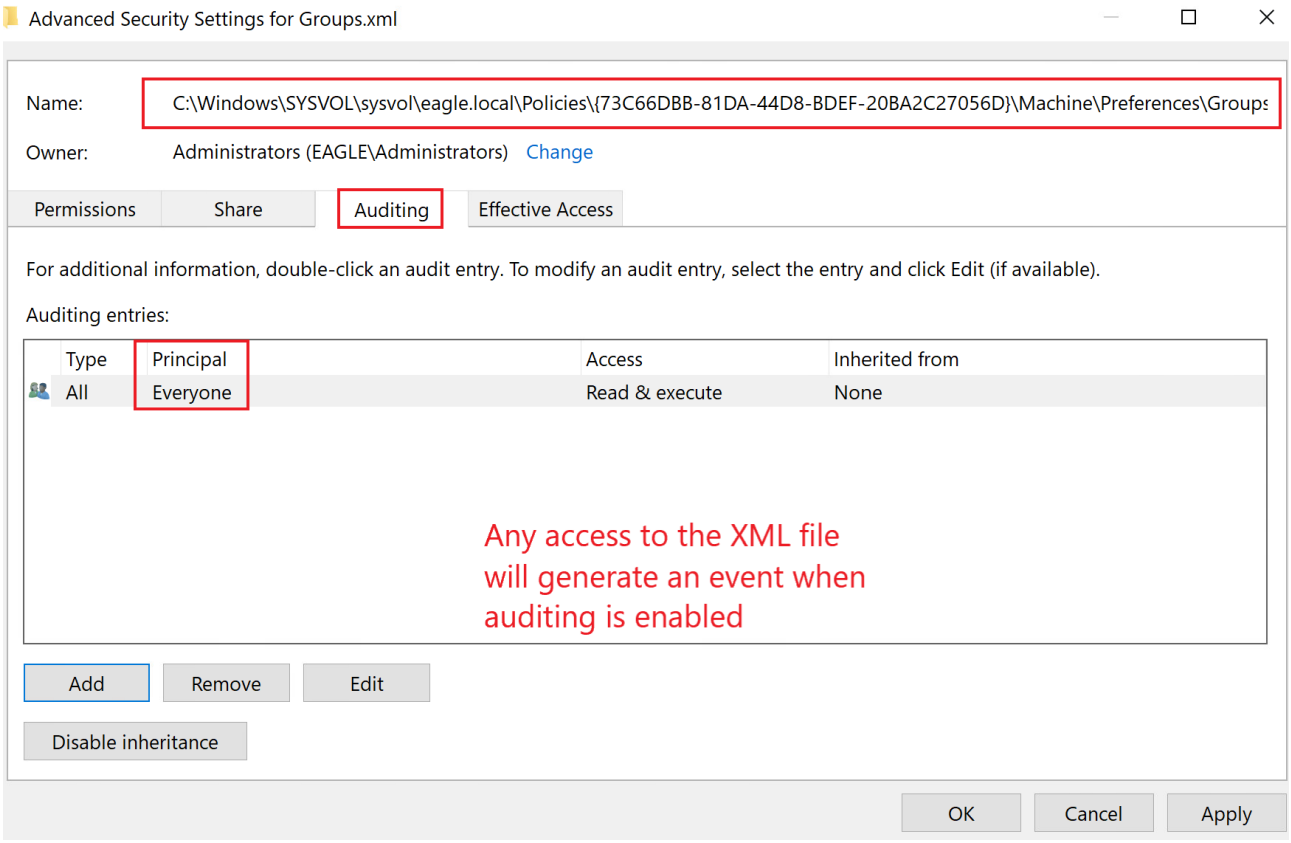


The caveat is that AD generates this event for every user that authenticates with Kerberos to any device; therefore, the presence of this event is very abundant. However, it is possible to know where the user authenticated from, which we can then use to correlate known good logins against potential malicious hash extractions. It may be hard to inspect specific IP addresses, especially if a user moves around office locations. However, it is possible to scrutinize the particular VLAN and alert on anything outside it.

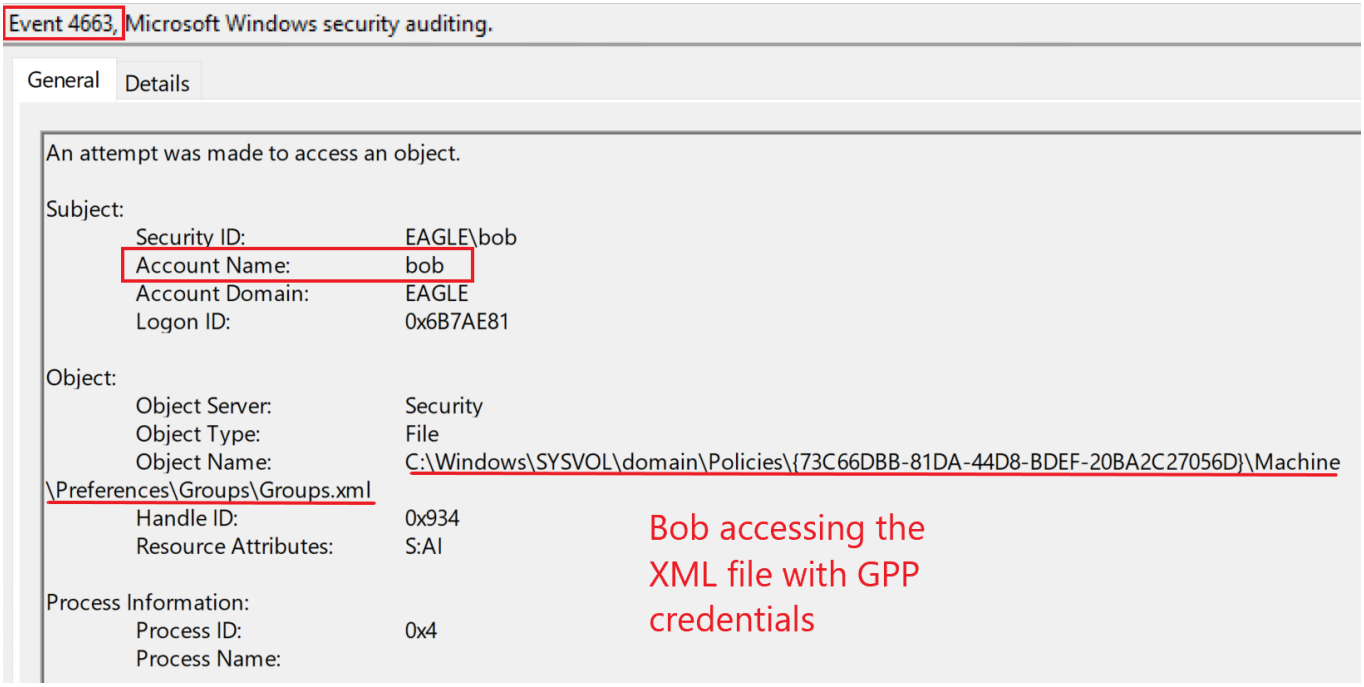
GPP Passwords

There are two detection techniques for this attack:

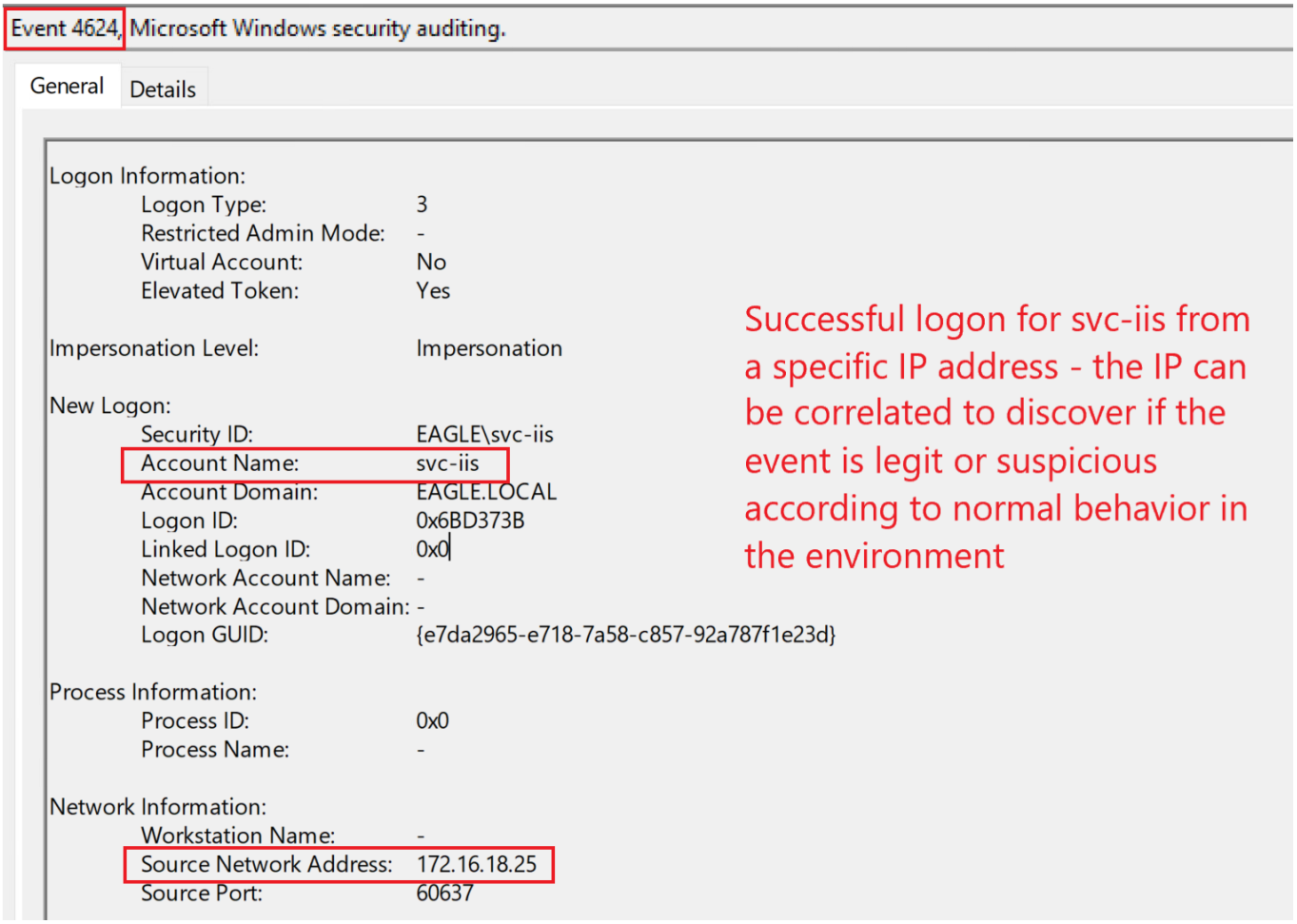
- Accessing the XML file containing the credentials should be a red flag if we are auditing file access; this is more realistic (due to volume otherwise) regarding detection if it is a dummy XML file, not associated with any GPO. In this case, there will be no reason for anyone to touch this file, and any attempt is likely suspicious. As demonstrated by `Get-GPPPasswords`, it parses all of the XML files in the Policies folder. For auditing, we can generate an event whenever a user reads the file:



Once auditing is enabled, any access to the file will generate an Event with the ID `4663`:



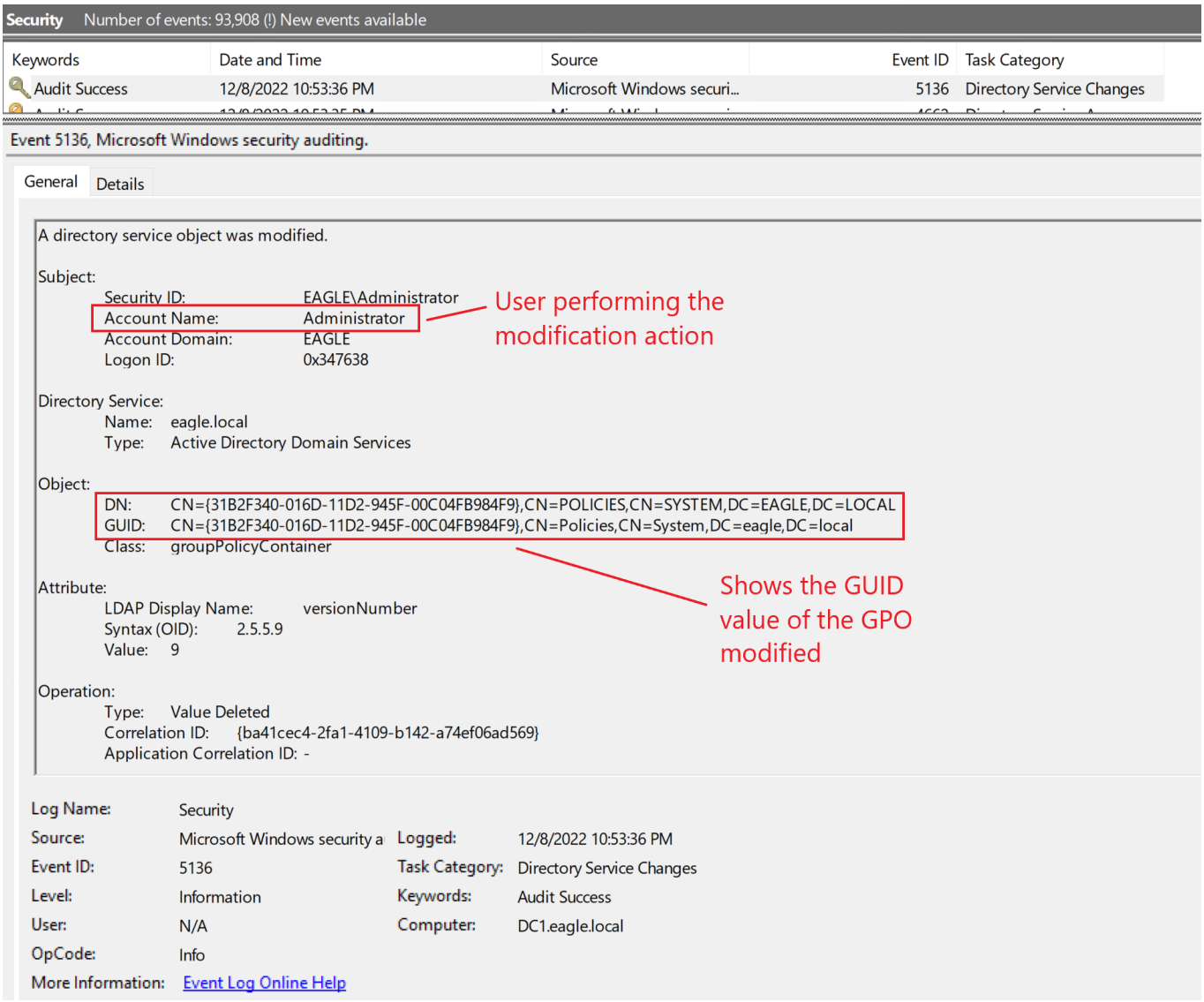
Logon attempts (failed or successful, depending on whether the password is up to date) of the user whose credentials are exposed is another way of detecting the abuse of this attack; this should generate one of the events `4624` (`successful logon`), `4625` (`failed logon`), or `4768` (`TGT requested`). A successful logon with the account from our attack scenario would generate the following event on the Domain Controller:



In the case of a service account, we may correlate logon attempts with the device from which the authentication attempt originates, as this should be easy to detect, assuming we know where certain accounts are used (primarily if the logon originated from a workstation, which is abnormal behavior for a service account).

GPO Permissions/GPO Files

Fortunately, it is straightforward to detect when a GPO is modified. If Directory Service Changes auditing is enabled, then the event ID `5136` will be generated:

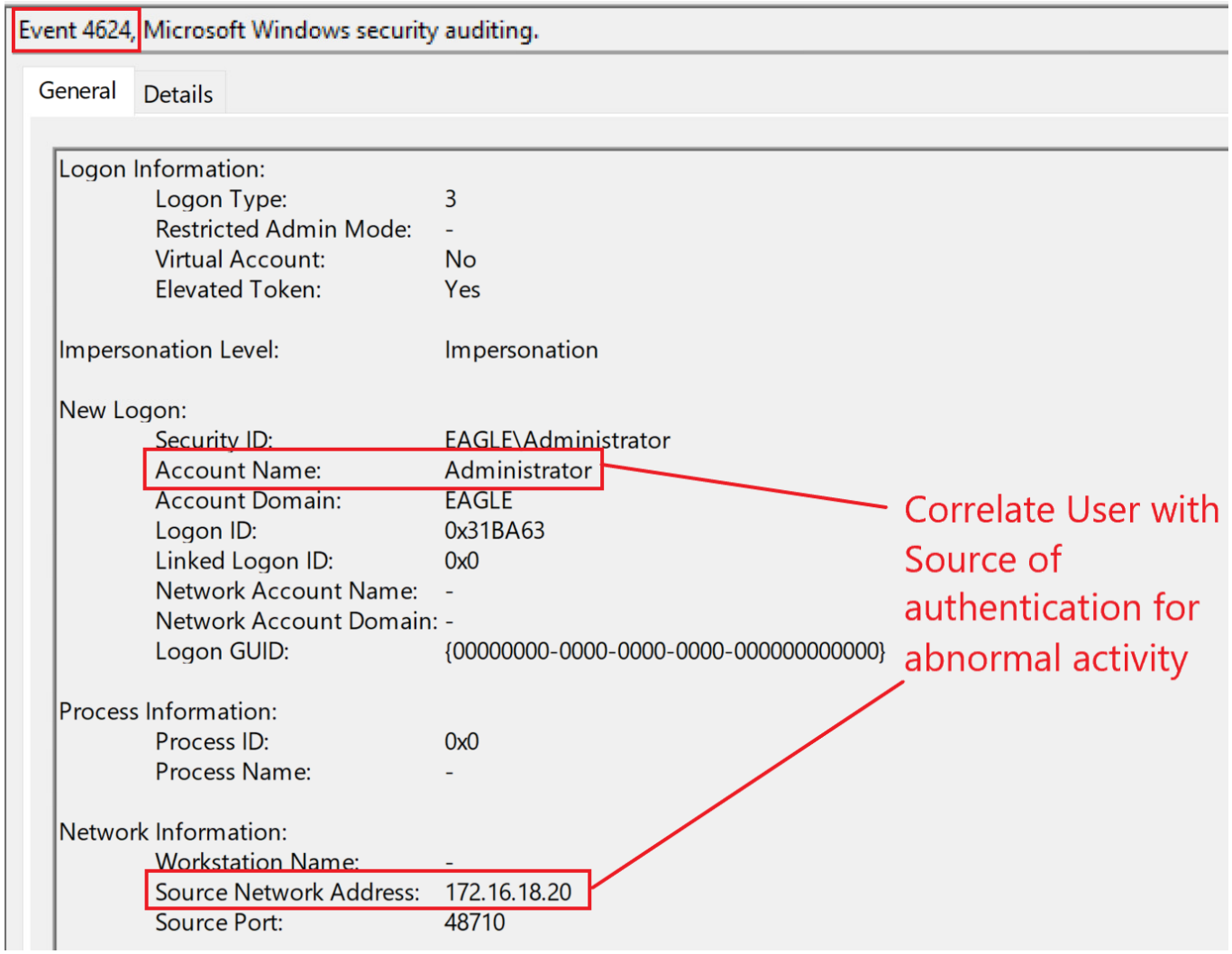


From a defensive point of view, if a user who is `not` expected to have the right to modify a GPO suddenly appears here, then a red flag should be raised.

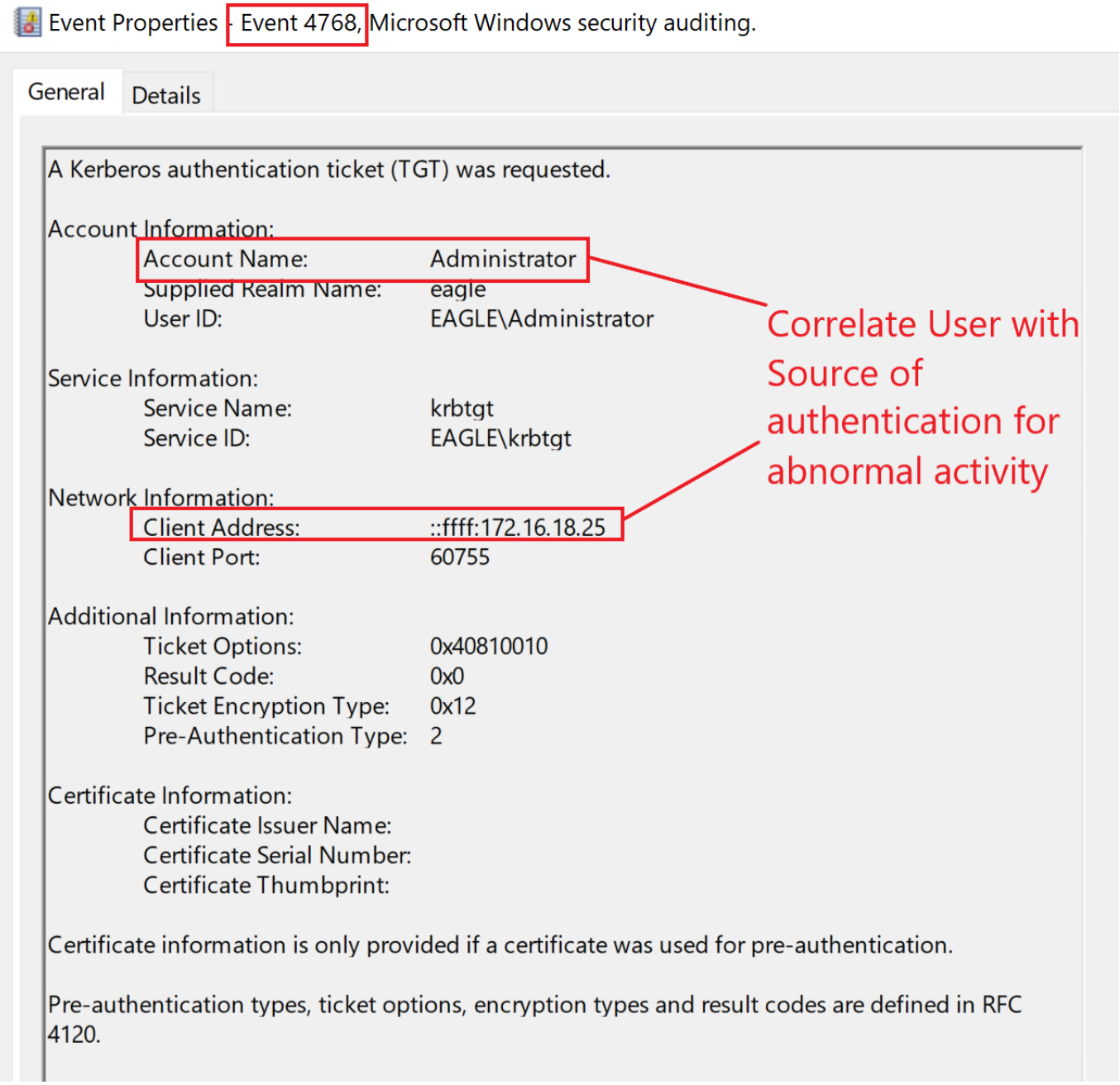
Credentials in Shares

Understanding and analyzing users' behavior is the best detection technique for abusing discovered credentials in shares. Suppose we know the time and location of users' login via data analysis. In that case, it will be effortless to alert on seemingly suspicious behaviors—for example, the discovered account 'Administrator' in the attack described above. If we were a mature organization that used `Privileged Access Workstation`, we would be alert to privileged users not authenticating from those machines. These would be alerts on event IDs `4624` / `4625` (failed and successful logon) and `4768` (Kerberos TGT requested).

Below is an example of a successful logon with event ID `4624` for the Administrator account:


Event 4624, Microsoft Windows security auditing.
General
Details
Logon Information:
Logon Type: 3
Restricted Admin Mode: -
Virtual Account: No
Elevated Token: Yes
Impersonation Level: Impersonation
New Logon:
Security ID: EAGLE\Administrator
Account Name: Administrator
Account Domain: EAGLE
Logon ID: 0x31BA63
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: -
Source Network Address: 172.16.18.20
Source Port: 48710
Correlate User with Source of authentication for abnormal activity

Similarly, if Kerberos were used for authentication, event ID `4768` would be generated:

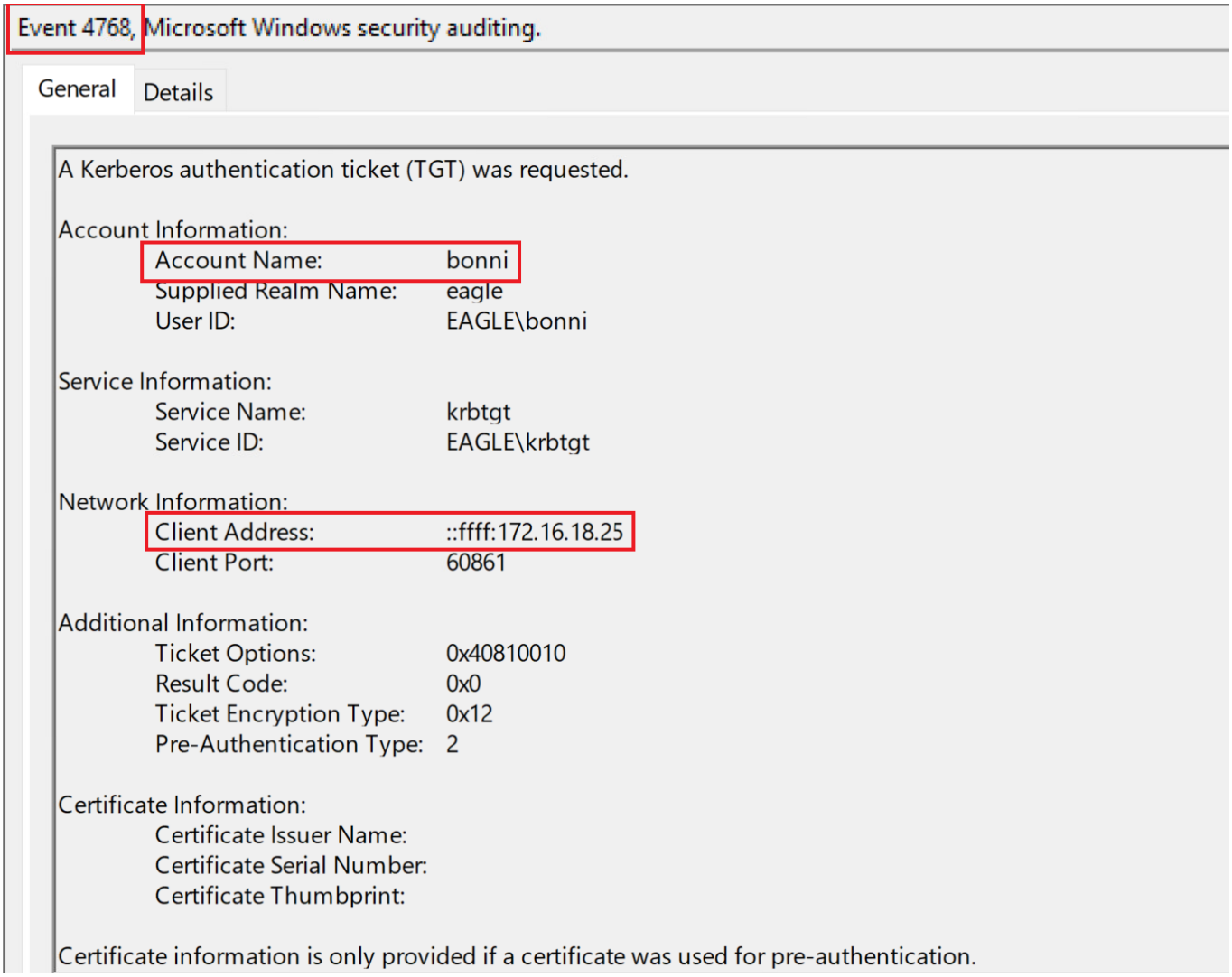


Another detection technique is discovering the `one-to-many` connections, for example, when `Invoke-ShareFinder` scans every domain device to obtain a list of its network shares. It would be abnormal for a workstation to connect to 100s or even 1000s of other devices simultaneously.

Credentials in Object Properties

Baselining users' behavior is the best technique for detecting abuse of exposed credentials in properties of objects. Although this can be tricky for regular user accounts, triggering an alert for

administrators/service accounts whose behavior can be understood and baselined is easier. Automated tools that monitor user behavior have shown increased success in detecting abnormal logons. In the example above, assuming that the provided credentials are up to date, we would expect events with event ID `4624` / `4625` (failed and successful logon) and `4768` (Kerberos TGT requested). Below is an example of event ID `4768`:

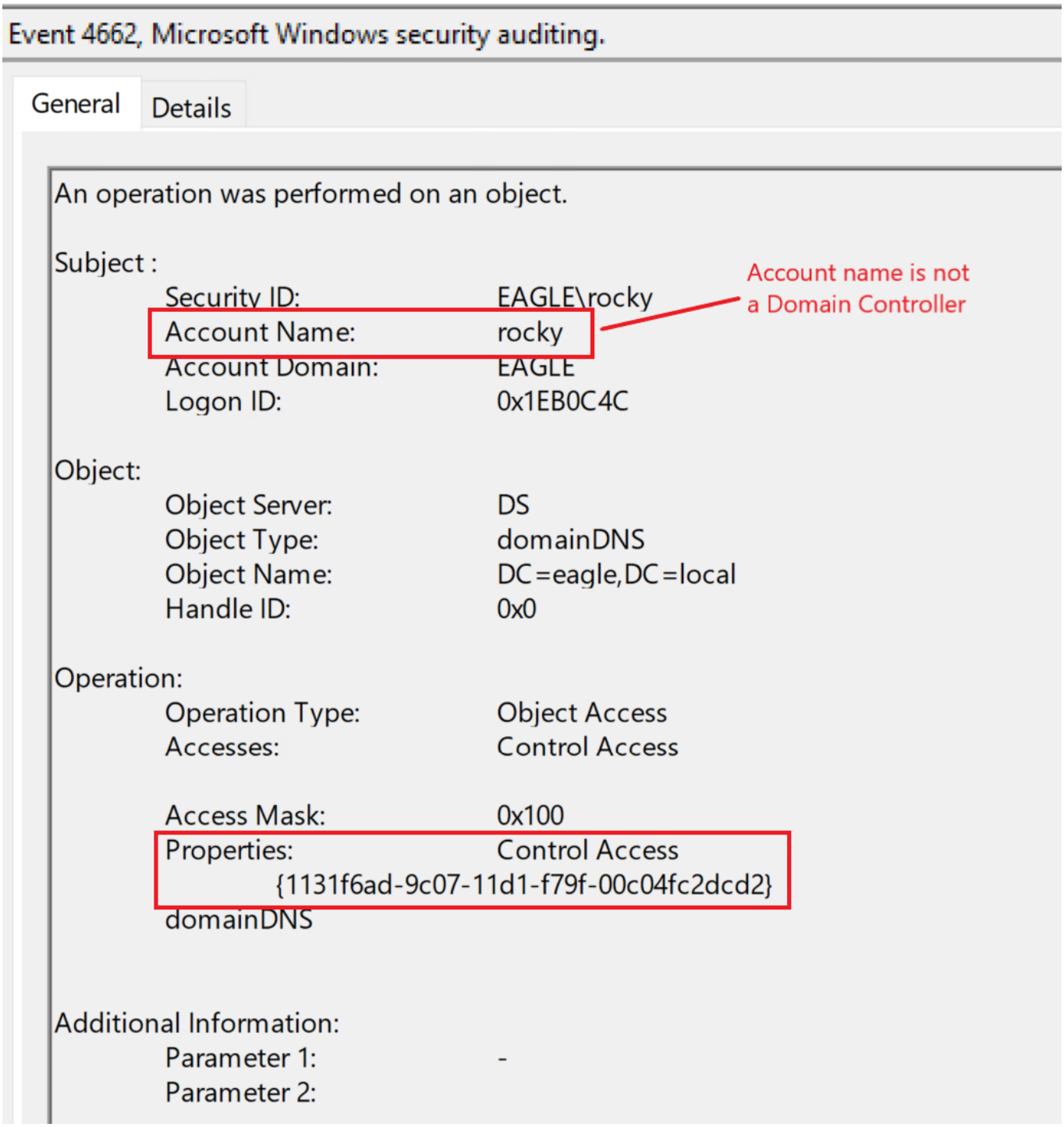


Unfortunately, the event ID `4738` generated when a user object is modified does not show the specific property that was altered, nor does it provide the new values of properties. Therefore, we cannot use this event to detect if administrators add credentials to the properties of objects.

DCSync

Detecting DCSync is easy because each Domain Controller replication generates an event with the ID `4662`. We can pick up abnormal requests immediately by monitoring for this event ID and checking whether the initiator account is a Domain Controller. Here's the event generated

from earlier when we ran `Mimikatz`; it serves as a flag that a user account is performing this replication attempt:



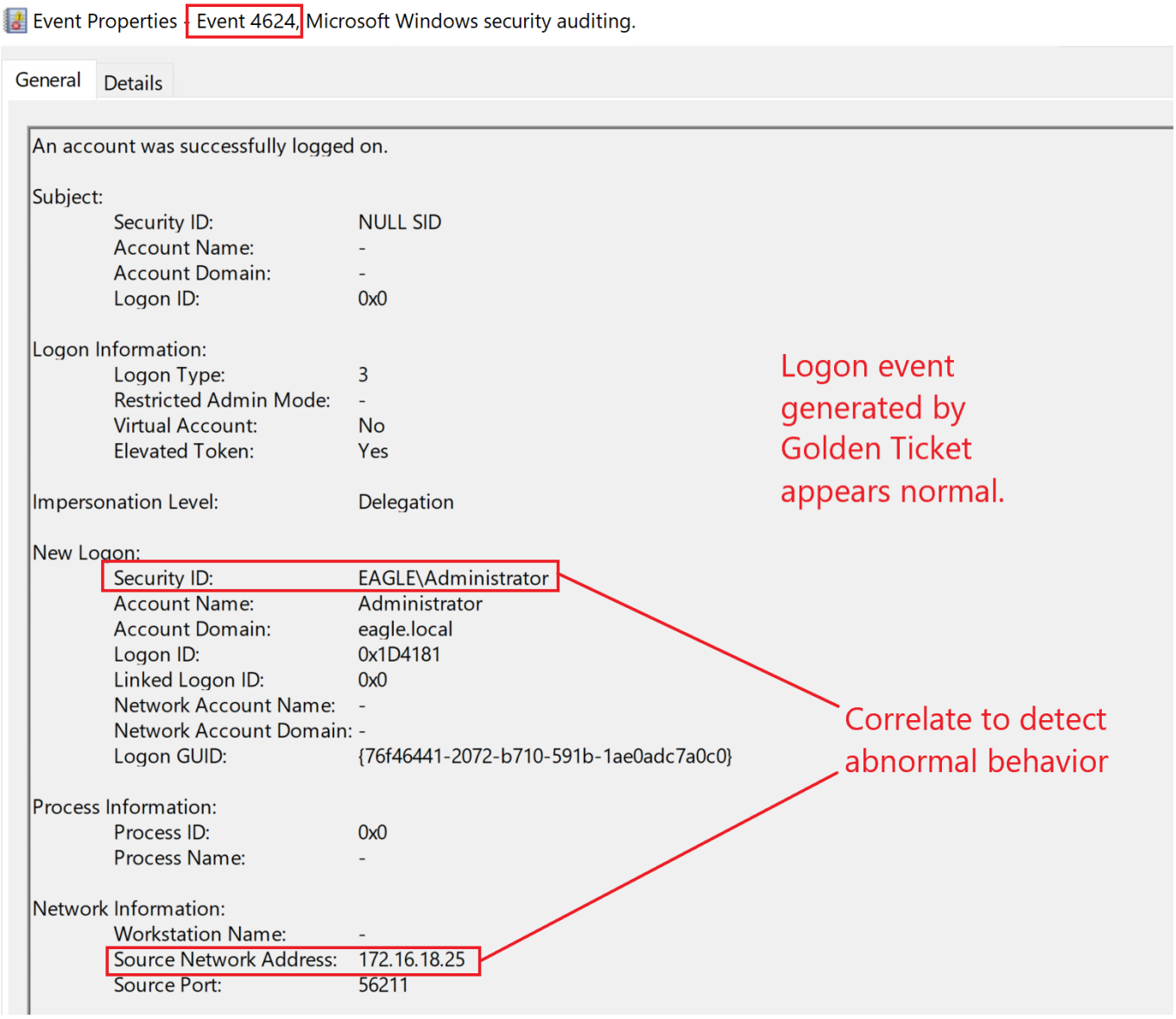
Since replications occur constantly, we can avoid false positives by ensuring the followings:

- Either the property `1131f6aa-9c07-11d1-f79f-00c04fc2dcd2` or `1131f6ad-9c07-11d1-f79f-00c04fc2dcd2` is present in the event.
- Whitelisting systems/accounts with a (valid) business reason for replicating, such as `Azure AD Connect` (this service constantly replicates Domain Controllers and sends the obtained password hashes to Azure AD).

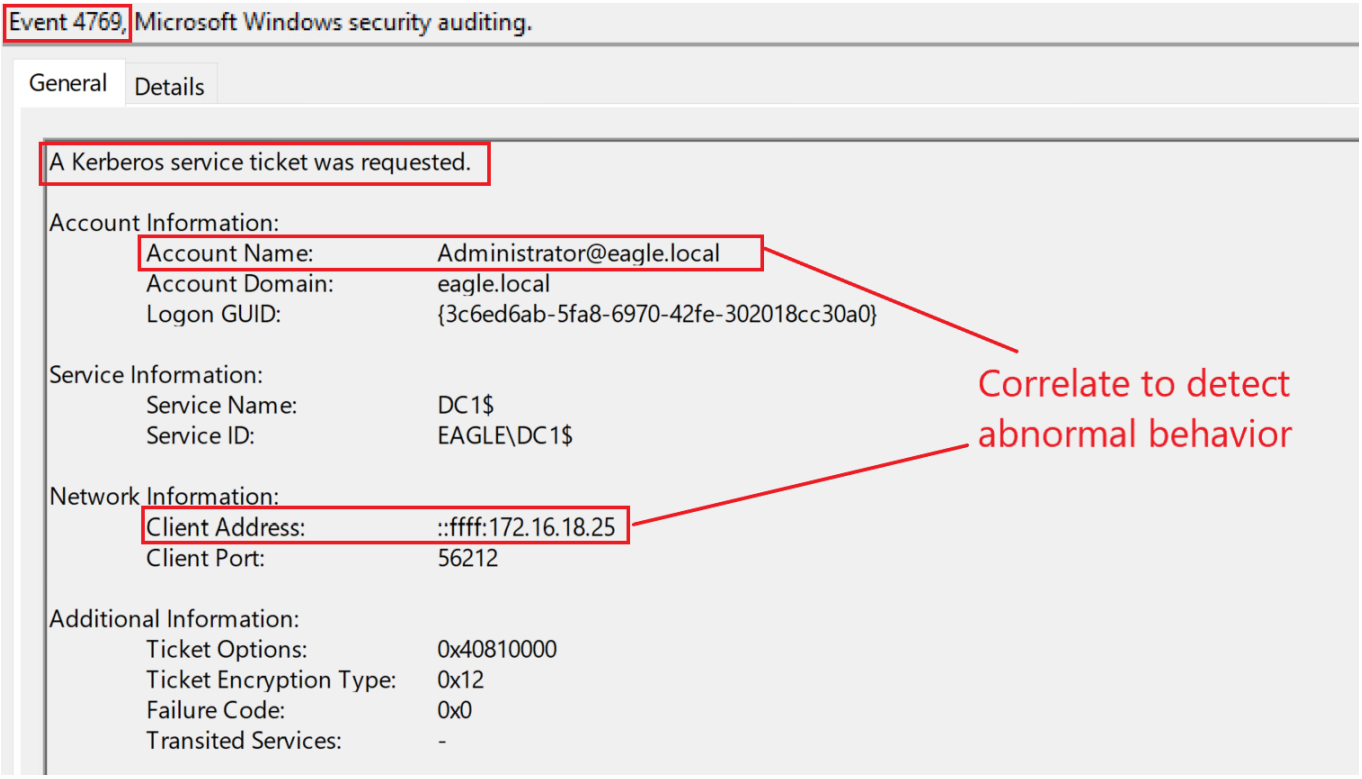
Golden Ticket

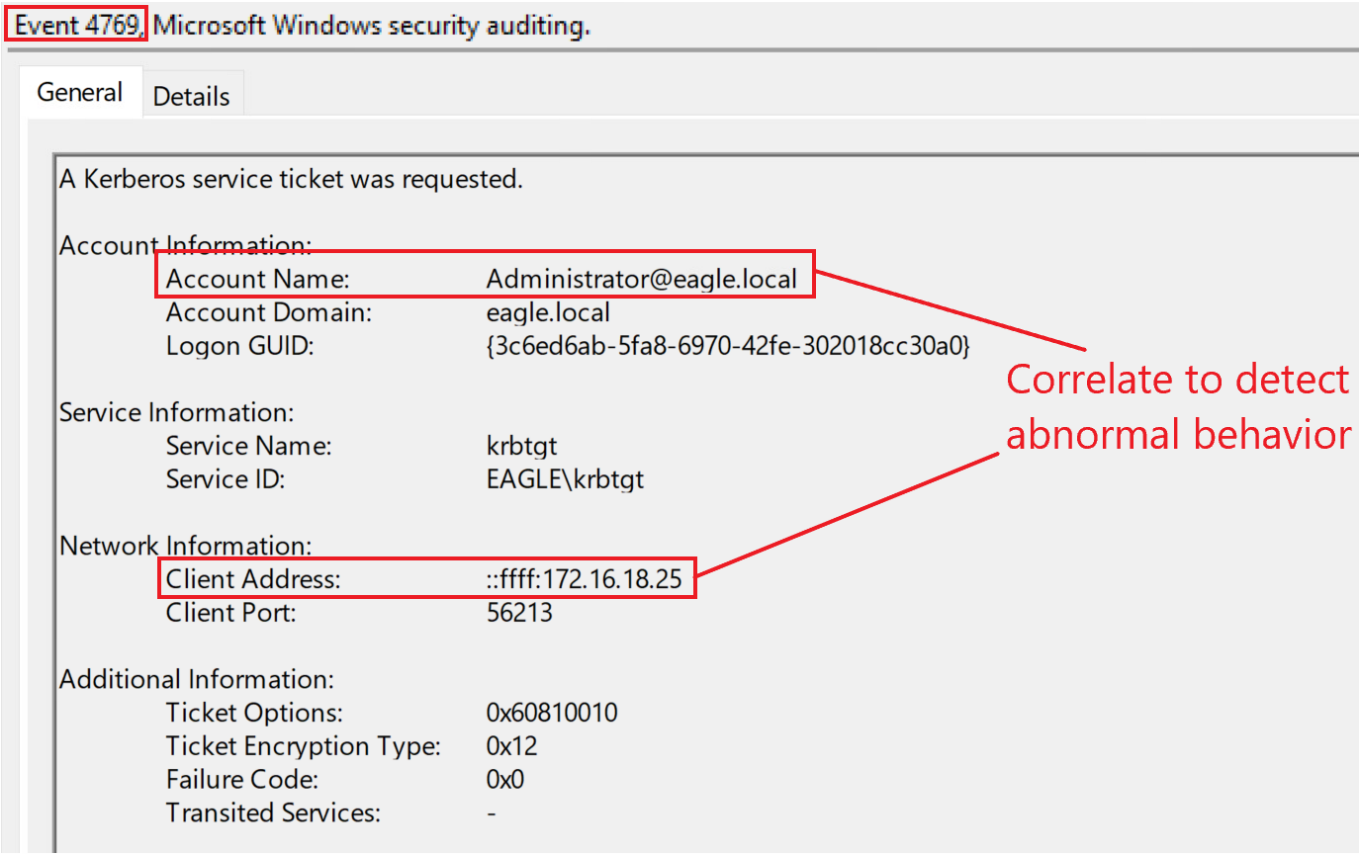
Correlating users' behavior is the best technique to detect abuse of forged tickets. Suppose we know the location and time a user regularly uses to log in. In that case, it will be easy to alert on other (suspicious) behaviors—for example, consider the account 'Administrator' in the attack described above. If a mature organization uses `Privileged Access Workstations` (`PAWs`), they should be alert to any privileged users not authenticating from those machines, proactively monitoring events with the ID `4624` and `4625` (successful and failed logon).

Domain Controllers will not log events when a threat agent forges a `Golden Ticket` from a compromised machine. However, when attempting to access another system(s), we will see events for successful logon originating from the compromised machine:



Another detection point could be a TGS service requested for a user without a previous TGT. However, this can be a tedious task due to the sheer volume of tickets (and many other factors). If we go back to the attack scenario, by running `dir \\dc1\c$` at the end, we generated two TGS tickets on the Domain Controller:





The only difference between the tickets is the service. However, they are ordinary compared to the same events not associated with the `Golden Ticket`.

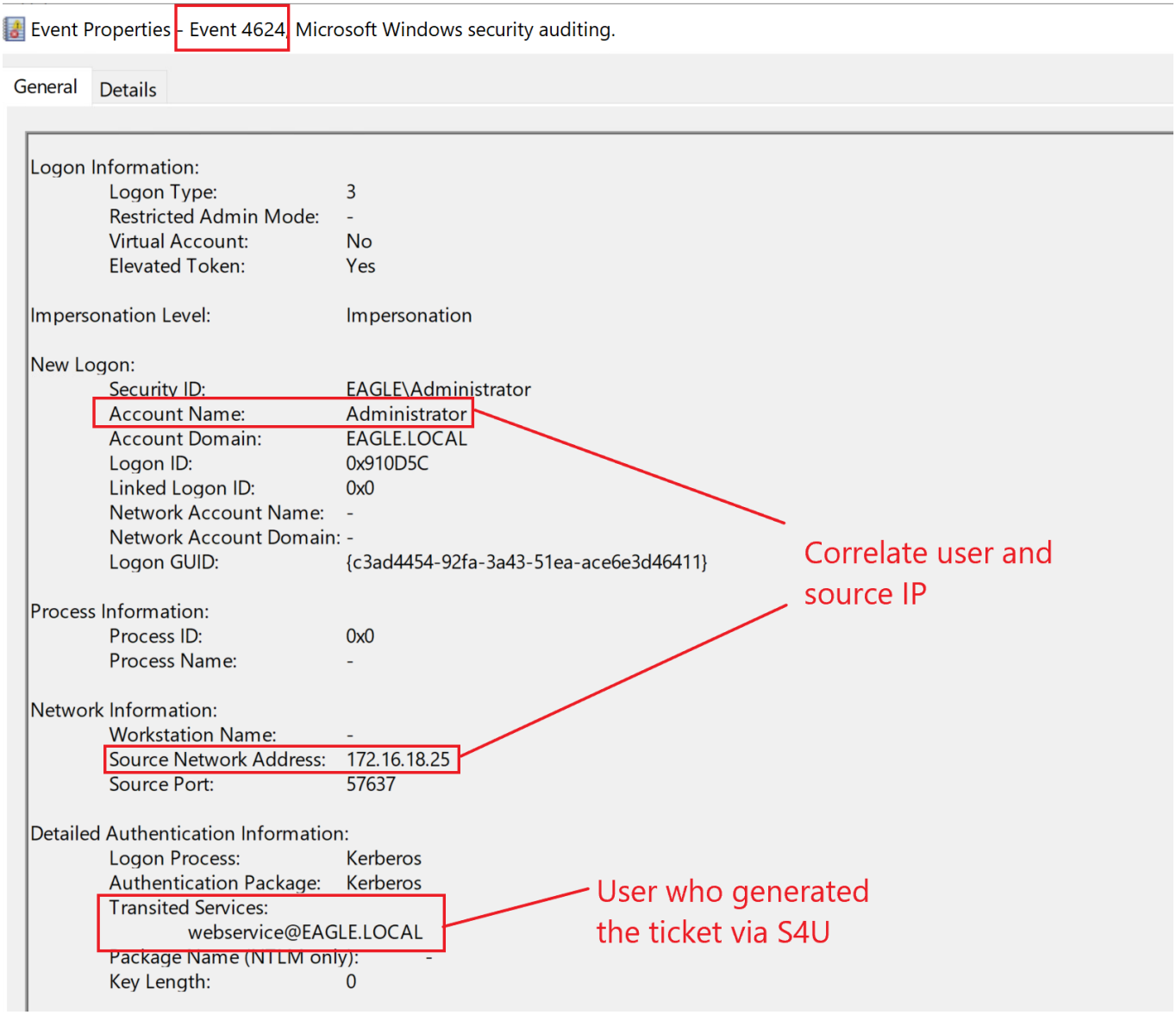
If `SID filtering` is enabled, we will get alerts with the event ID `4675` during cross-domain escalation.

Kerberos Constrained Delegation

Correlating users' behavior is the best technique to detect `constrained delegation` abuse. Suppose we know the location and time a user regularly uses to log in. In that case, it will be easy to alert on other (suspicious) behaviors—for example, consider the account 'Administrator' in the attack described above. If a mature organization uses Privileged Access Workstations (PAWs), they should be alert to any privileged users not authenticating from those machines, proactively monitoring events with the ID `4624` (successful logon).

In some occasions, a successful logon attempt with a delegated ticket will contain information about the ticket's issuer under the `Transited Services` attribute in the events log. This attribute is normally populated if the logon resulted from an `S4U` (`Service For User`) logon process.

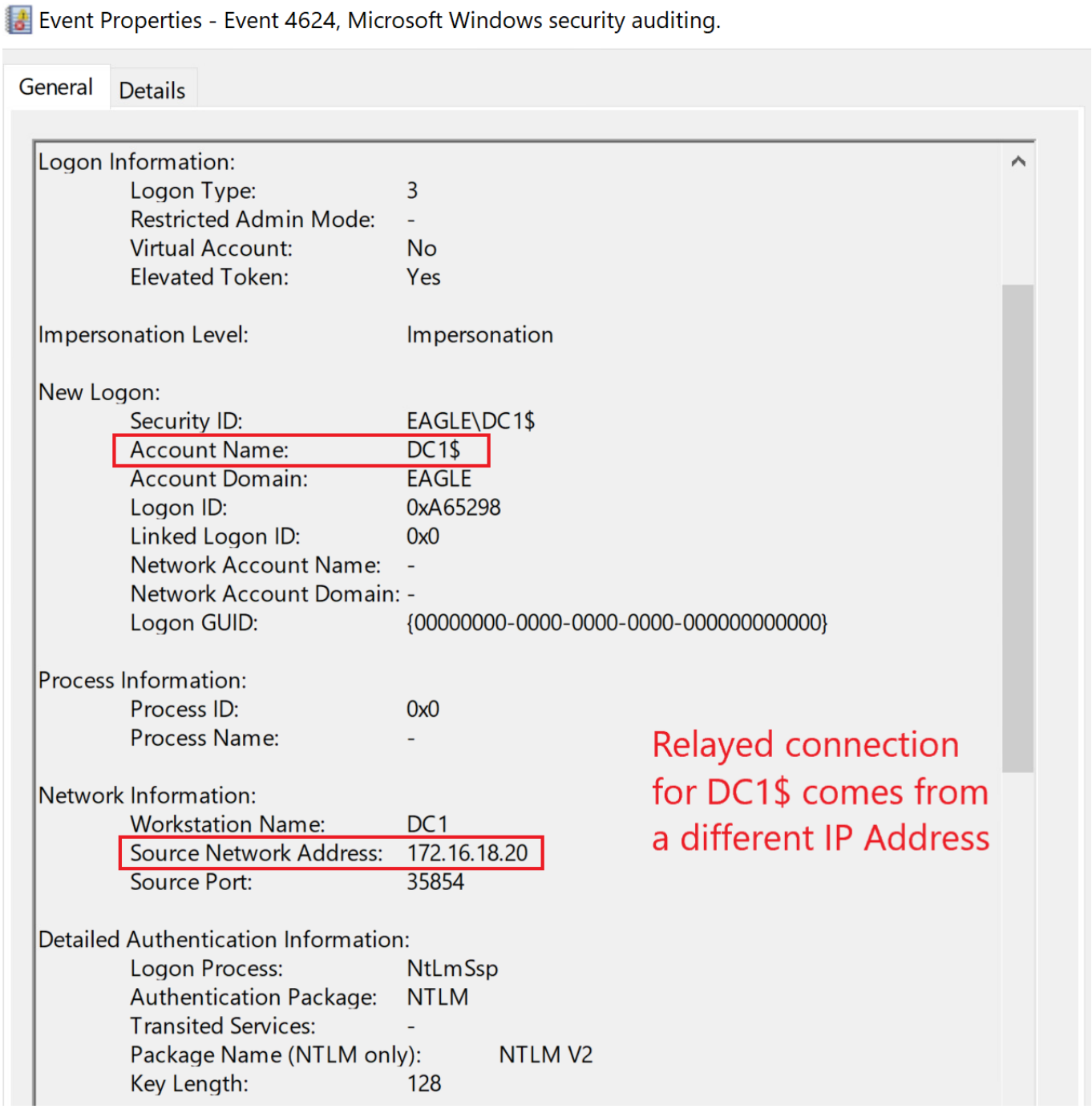
S4U is a Microsoft extension to the Kerberos protocol that allows an application service to obtain a Kerberos service ticket on behalf of a user; if we recall from the attack flow when utilizing `Rubeus`, we specified this `S4U` extension. Here is an example logon event by using the web service to generate a ticket for the user Administrator, which then was used to connect to the Domain Controller (precisely as the attack path above):



Print Spooler & NTLM Relaying

Exploiting the `PrinterBug` will leave traces of network connections toward the Domain Controller; however, they are too generic to be used as a detection mechanism.

In the case of using `NTLMRelayx` to perform DCSync, no event ID `4662` is generated (as mentioned in the DCSync section); however, to obtain the hashes as DC1 from DC2, there will be a successful logon event for DC1. This event originates from the IP address of the Kali machine, not the Domain Controller, as we can see below:



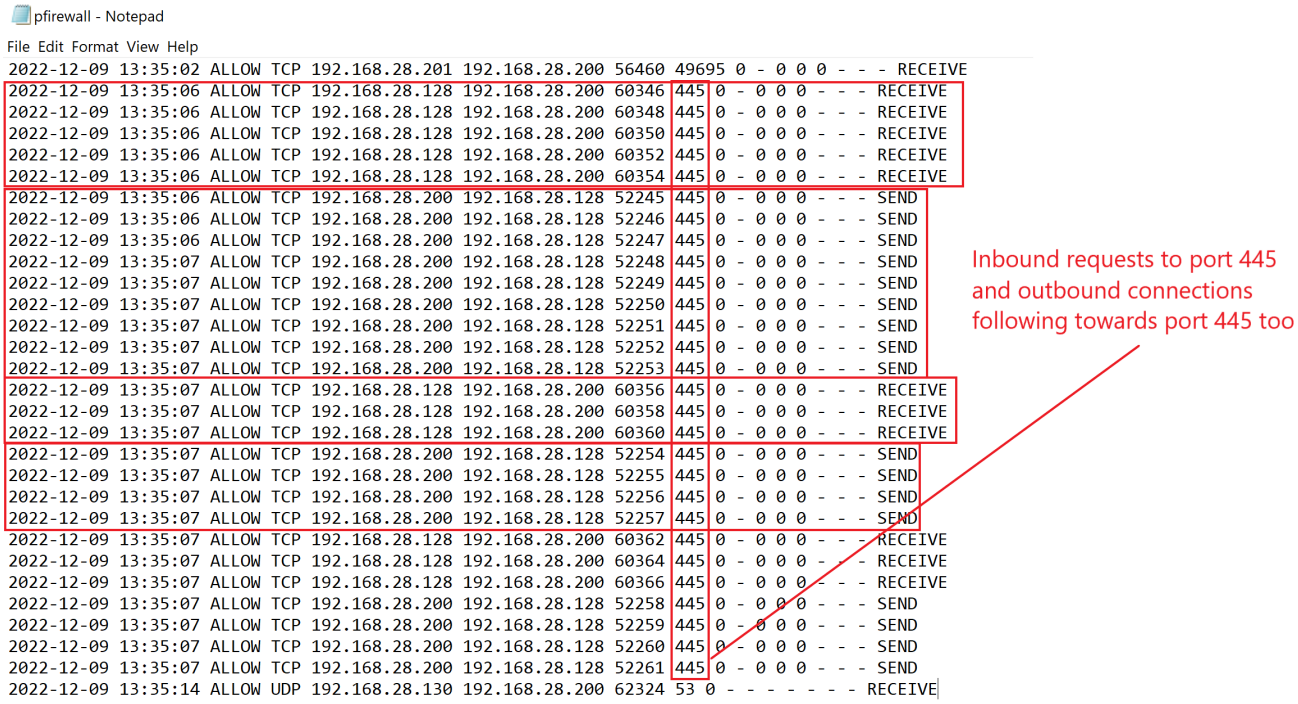
A suitable detection mechanism always correlates all logon attempts from core infrastructure servers to their respective IP addresses (which should be static and known).

Coercing Attacks & Unconstrained Delegation

As mentioned, Windows does not provide an out-of-the-box solution for monitoring RPC activity. The RPC Firewall from zero networks is an excellent method of detecting the abuse of these functions and can indicate immediate signs of compromise; however, if we follow the general

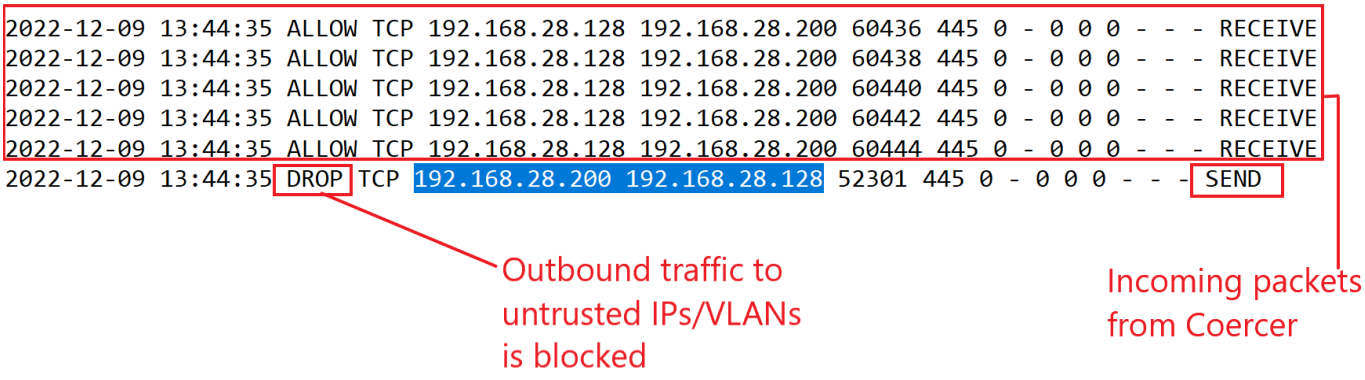
recommendations to not install third-party software on Domain Controllers then firewall logs are our best chance.

A successful coercing attack with Coercer will result in the following host firewall log, where the machine at `.128` is the attacker machine and the `.200` is the Domain Controller:



We can see plenty of incoming connections to the DC, followed up by outbound connections from the DC to the attacker machine; this process repeats a few times as Coercer goes through several different functions. All of the outbound traffic is destined for port 445.

If we go forward and block outbound traffic to port 445, then we will observe the following behavior:

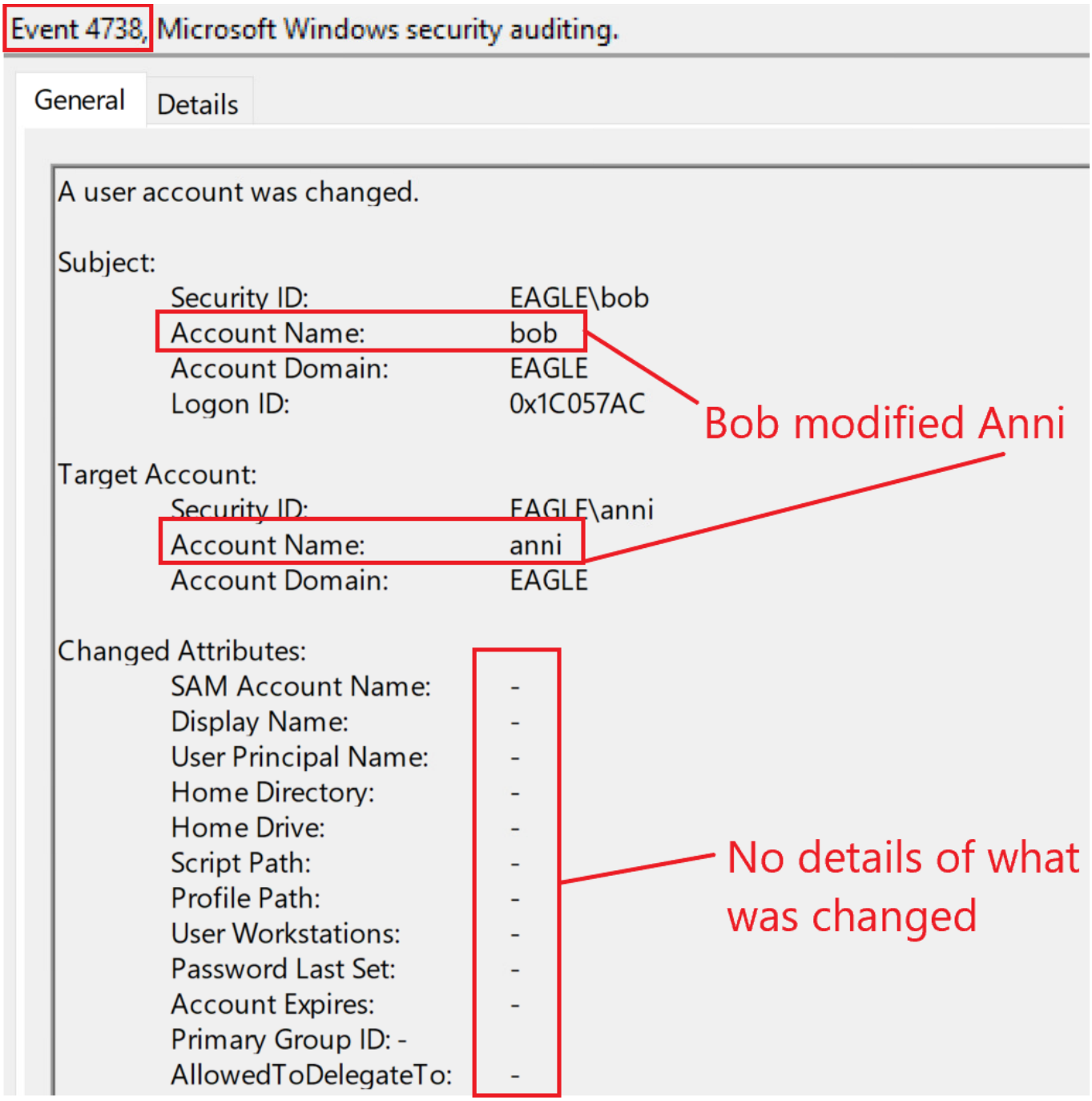


Now we can see that even though the inbound connection is successful, the firewall drops the outbound one, and consequently, the attacker does not receive any coerced TGTs. Sometimes, when port 445 is blocked, the machine will attempt to connect to port 139 instead, so blocking both ports `139` and `445` is recommended.

The above can also be used for detection, as any unexpected dropped traffic to ports `139` or `445` is suspicious.

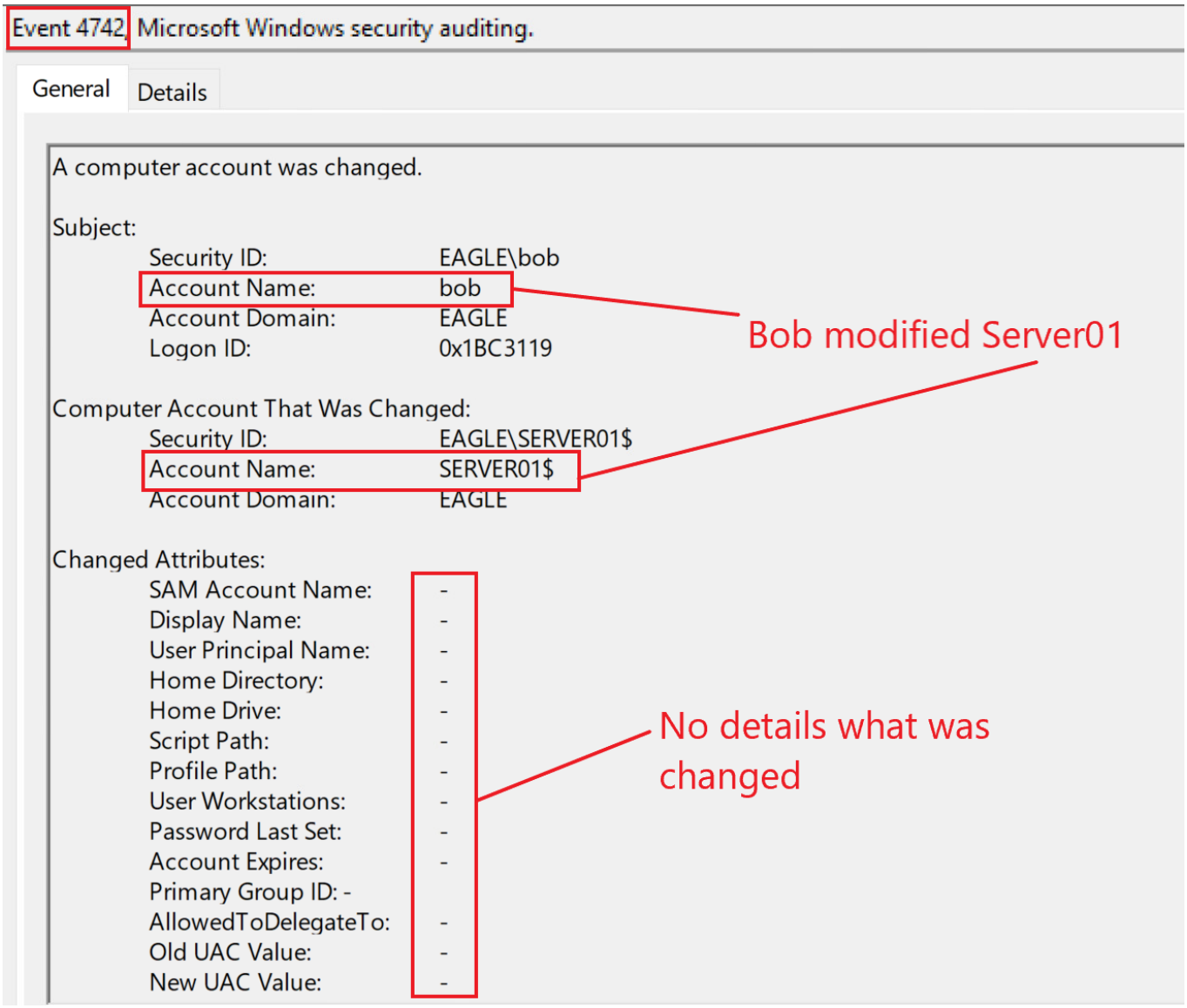
Object ACLs

Fortunately, we have several ways to detect if AD objects are modified. Unfortunately, the events generated for modified objects are incomplete, as they do not provide granular visibility over what was changed. For example, in the first case described above, Bob modified Anni by adding an SPN value. By doing so, Bob will have the means to perform Kerberoasting against Anni. When the SPN value gets added, an event with the ID `4738`, "A user account was changed", is generated. However, this event does not demonstrate all modified user properties, including the SPN. Therefore, the event only notifies about the modification of a user but does not specify what exactly was changed (although it does have a fair amount of fields that can be useful). Below is the event that will be generated if Bob adds any bogus SPN value to Anni's User Object:



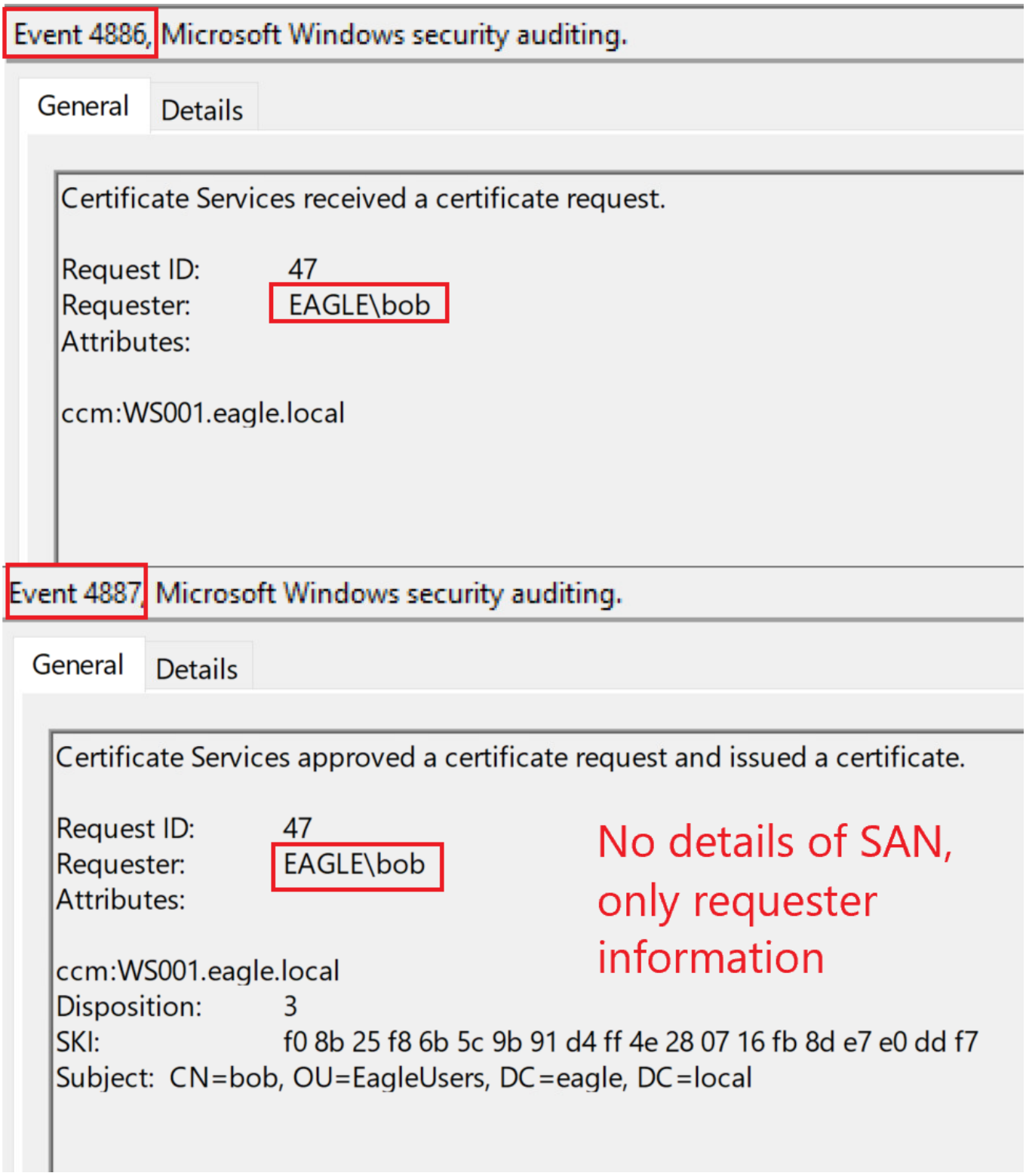
However, using this event, we can tell if a non-privileged user performs privileged actions on another user. If, for example, all privileged users have a naming convention that begins with "adminxxxx", then any change not associated with "adminxxxx" is suspicious. If an ACL abuse leads to a password reset, the event ID `4724` will be logged.

Similarly, if Bob were to perform the second scenario, an event with ID `4742` would be generated, which is also unfortunately limited in the information it can provide; however, it notifies about the action that the user account Bob is compromised and used maliciously. The following was the event ID `4742` generated when Bob modified Server01:



PKI - ESC1

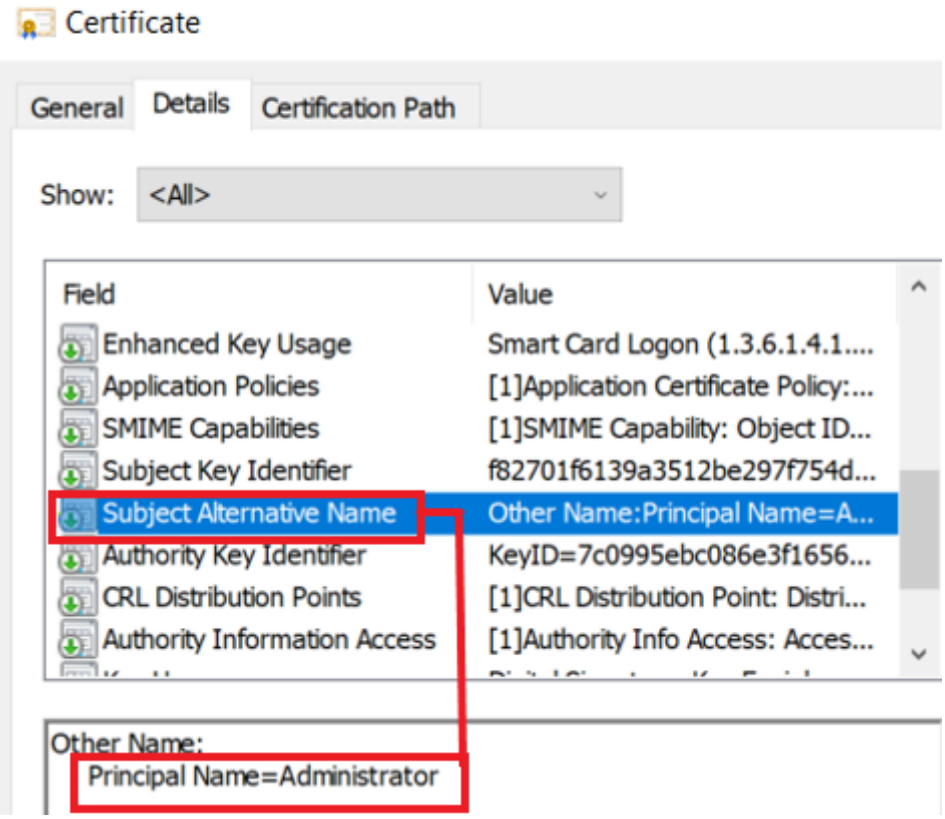
When the CA generates the certificate, two events will be logged, one for the received request and one for the issued certificate, if it succeeds. Those events have the IDs of `4886` and `4887` as shown below:



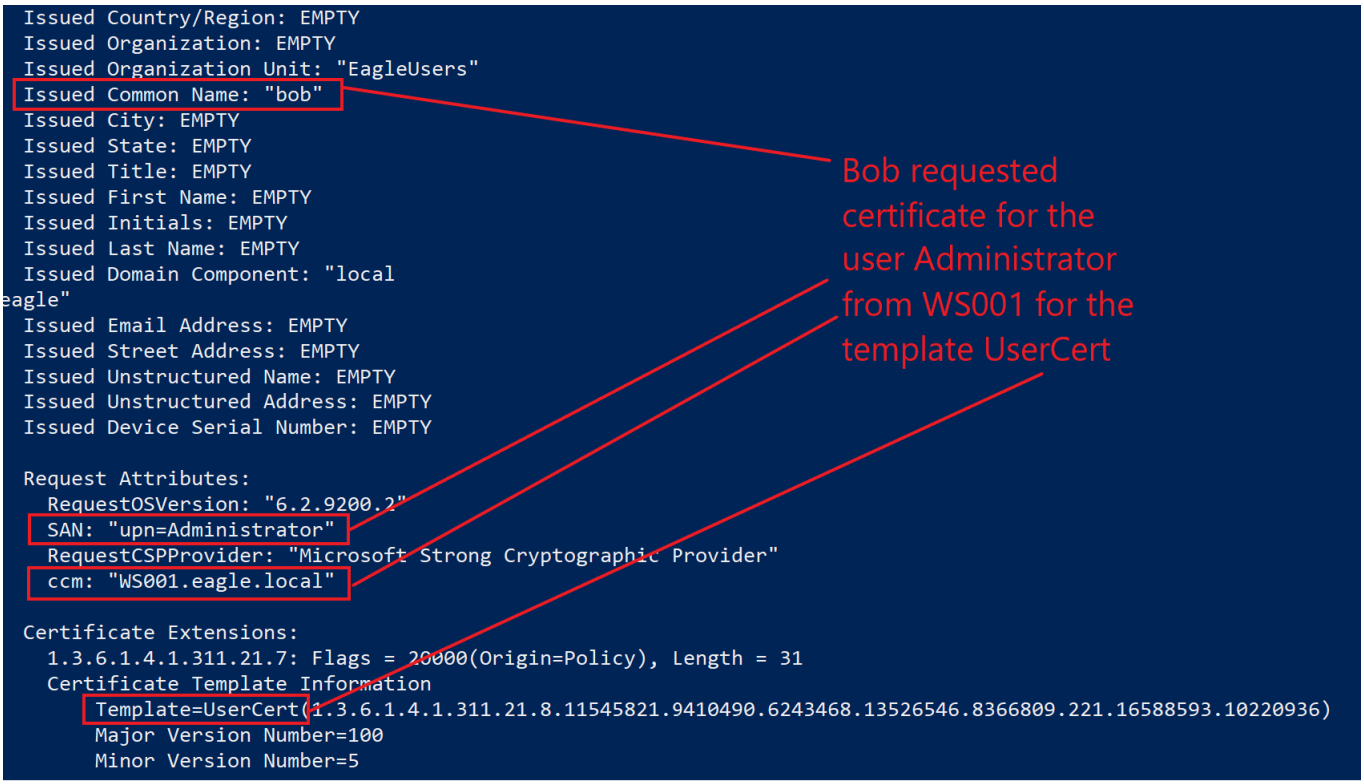
Unfortunately, we can only tell that Bob requested a certificate from WS001; we cannot know if the request specified the SAN.

The CA contains a list of all issued certificates, so if we look there, we will see the request for certificate ID 36 (the one from the attack scenario above):

The general overview of the GUI tool does not display the SAN either, but we can tell that a certificate was issued via the vulnerable template. If we want to find the SAN information, we'll need to open the certificate itself:

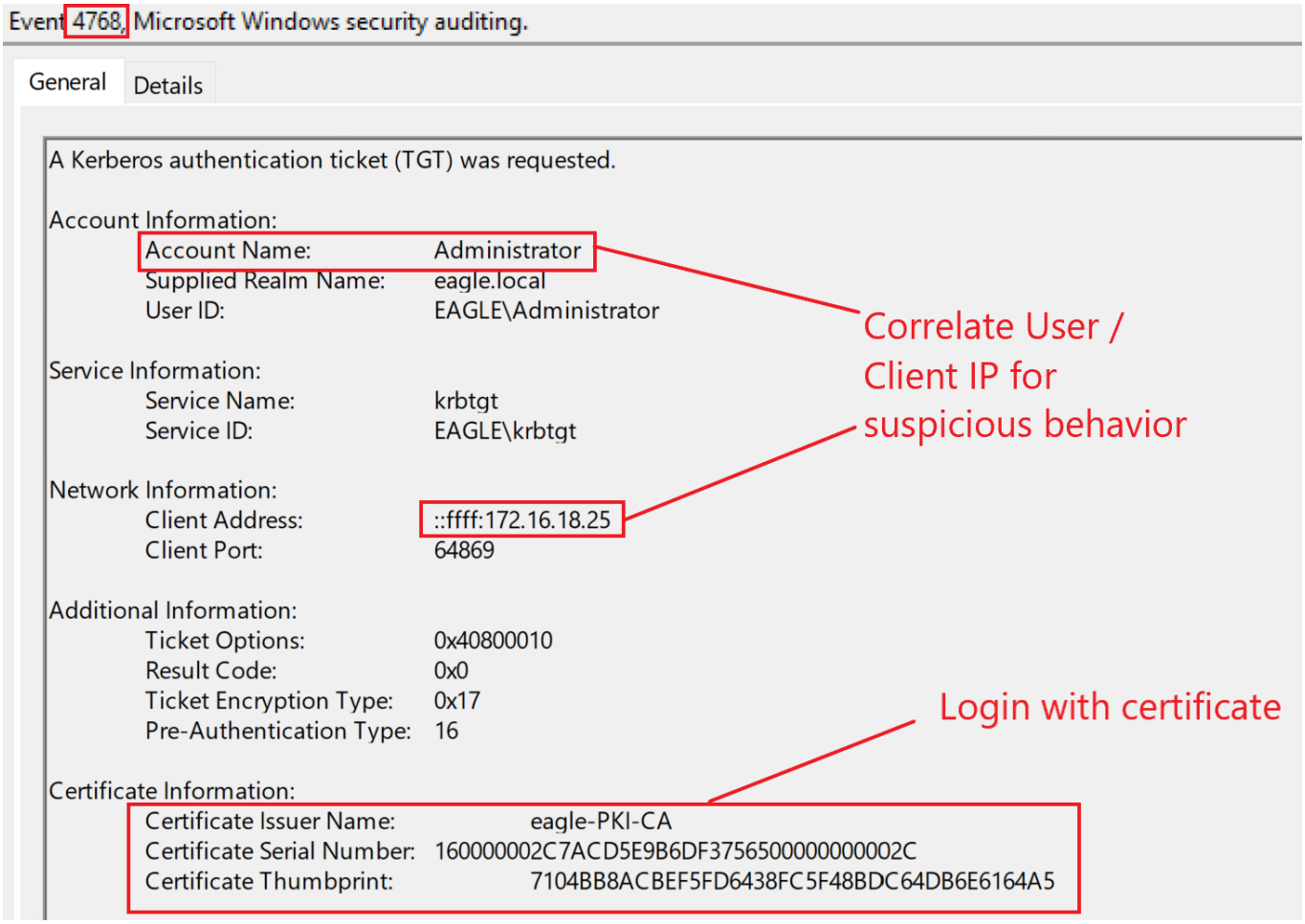


There is also the possibility to view that programmatically: the command `certutil -view` will dump everything on the CA with all of the information about each certificate (this can be massive in a large environment):



With some scripting, we can automate parsing and discovery of abused vulnerable templates by threat agents.

Finally, if you recall, in the attack, we used the obtained certificate for authentication and obtained a TGT; AD will log this request with the event ID `4768`, which will specifically have information about the logon attempt with a certificate:



Note that events `4886` and `4887` will be generated on the machine issuing the certificate rather than the domain controller. If GUI access is not available, we can use PSSession to interact with the PKI machine, and the `Get-WinEvent` cmdlet to search for the events:

```
C:\Users\bob\Downloads>runas /user:eagle\htb-student powershell

Enter the password for eagle\htb-student:
Attempting to start powershell as user "eagle\htb-student" ...
```