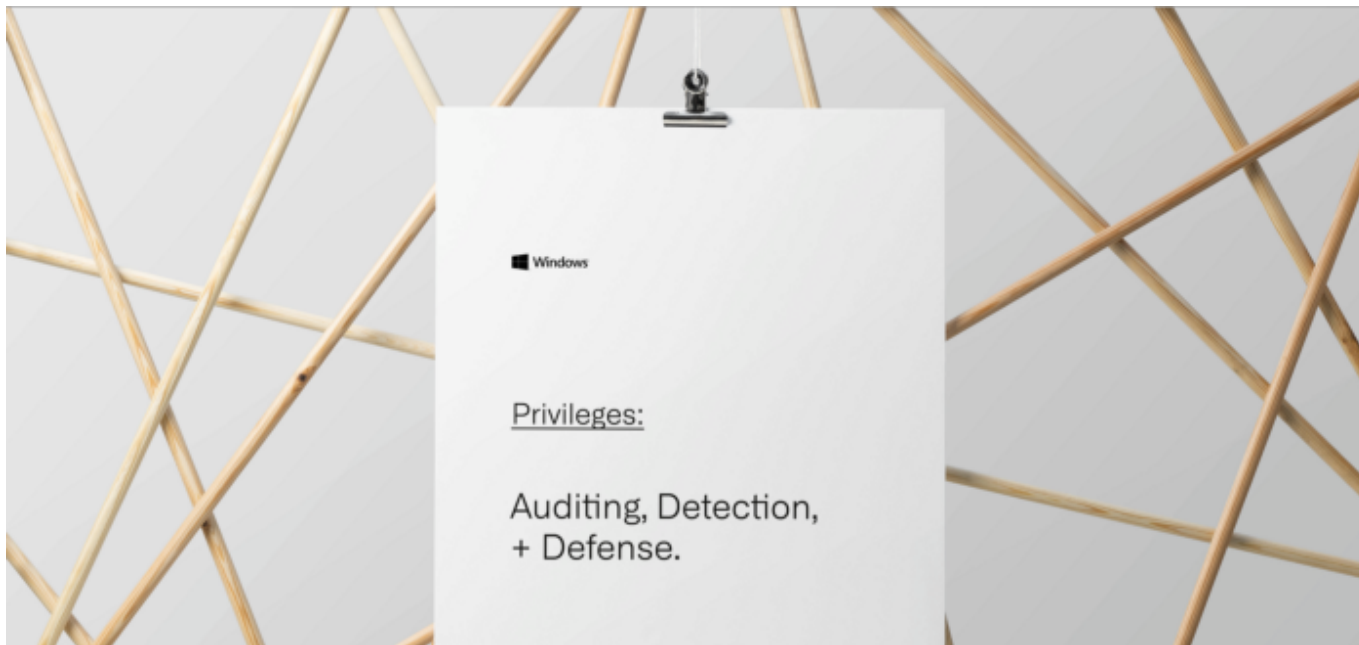


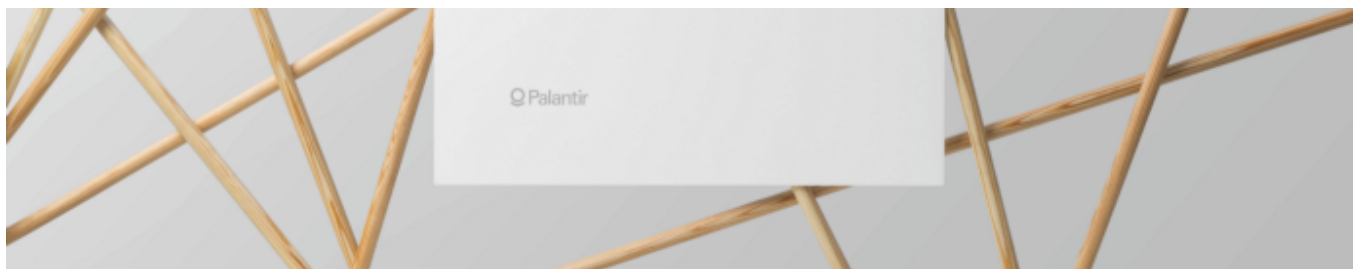
Windows Privilege Abuse: Auditing, Detection, and Defense



Palantir [Follow](#)

Jan 29 · 10 min read





Privileges are an important native security control in Windows. As the name suggests, privileges grant rights for accounts to perform privileged operations within the operating system: debugging, impersonation, etc. Defenders who understand privileges and how attackers may abuse them can enhance their detection and attack surface reduction capabilities.

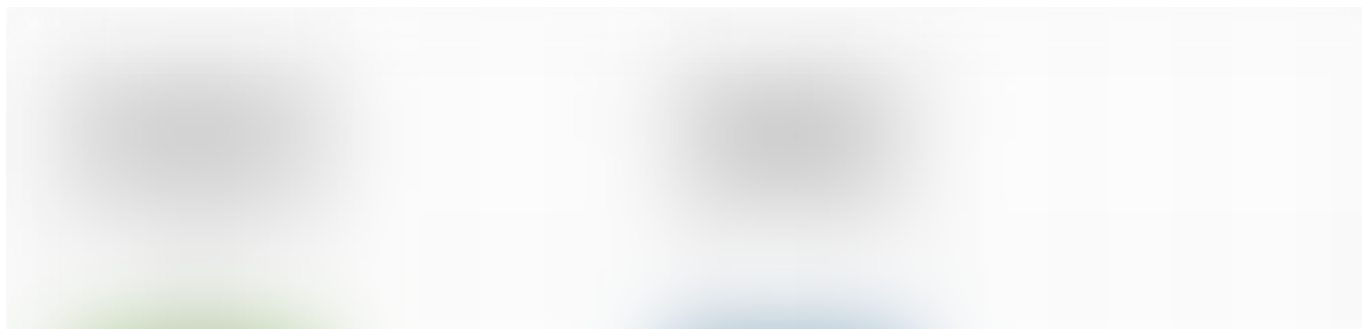
In this blog post, we give a brief introduction to privileges and share our recommendations for detecting and preventing their abuse. We walk through the key concepts a defender needs to understand to protect privileges, and provide an example on how to improve security through auditing, detection strategies, and targeted privilege removal.

Introduction to Windows privileges

A privilege is a right granted to an account to perform privileged operations within the operating system. It's important to distinguish between privileges (which apply to system-related resources) and access rights (which apply to securable objects). Microsoft provides a detailed explanation of Windows privileges in their [Access Control documentation](#). Below, we walk through the most important concepts to understand if you want to better defend against abuse.

Access tokens

Access tokens are the foundation of all authorization decisions for securable resources hosted on the operating system. They are granted to authorized users by the Local Security Authority (LSA). The access token includes the user's security identifier (SID), group SIDs, privileges, integrity level, and other security-relevant information.





User Access Token and a Securable Object. Reference: [Microsoft Security Principals Documentation](#)

Every process or thread created by a user inherits a copy of their token. This token is used by to perform access checks when accessing securable objects or performing privileged actions within the operating system.

Access tokens may exist as *primary tokens* or *impersonation tokens*. Primary tokens function as described and are used to present the default security

information for a process or thread.

Impersonation allows for a thread to perform an operation using an access token from another user or client. Impersonation tokens are typically used in client/server communication. For example, when a user accesses an SMB file share, the server needs a copy of the user's token to validate that the user has sufficient permissions. The executing server-side thread includes an impersonation token for the user in addition to the thread's primary token, and uses the impersonation token to perform access checks for the user's actions.

Restricted access tokens

Restricted tokens (also known as a *filtered admin token*) are a subset of primary or impersonation tokens that have been modified to control privileges or permissions. Restricted access tokens allow the system to remove privileges, add deny-only access control entries, or perform other access rights changes.

Assuming User Account Control (UAC) is running during the initial token creation process, LSA will attempt to identify if the user is a member of a

privileged group or has been granted a sensitive privilege using functionality similar to the IsTokenRestricted function. Presence of a restricted SID will result in a call to produce a new access token with reduced privileges.

An example of the restricted access token can be seen in the following screenshot:



Even though the user in question is a local administrator, the unelevated `cmd.exe` shell carries a token restricted to only a handful of privileges. When elevated to run as administrator, the process carries the user's primary token with a larger list of privileges:



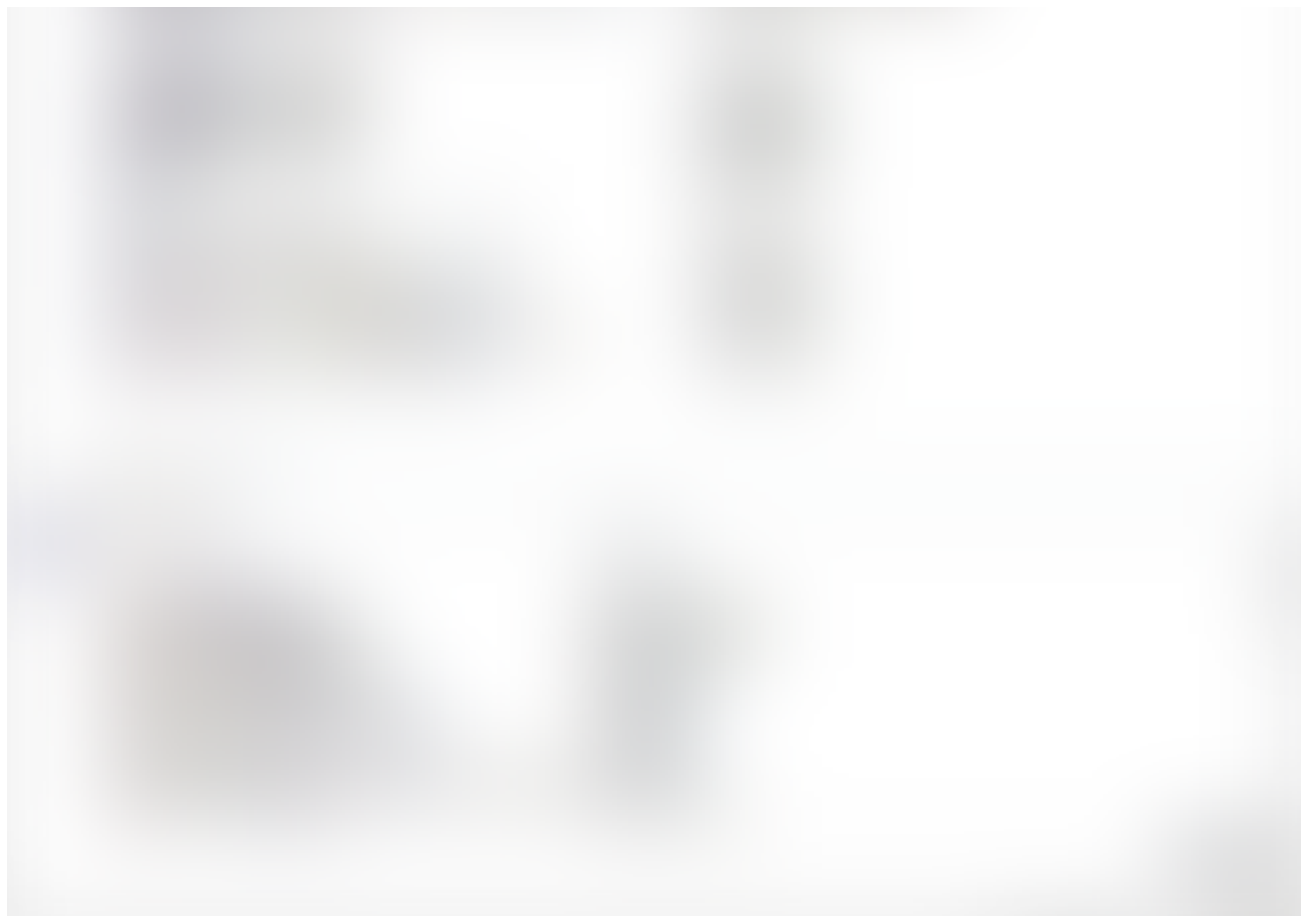
The primary token can also be inspected with Process Explorer. The following screenshot shows the restricted access token attached to the unelevated process.





The following screenshot shows the primary access token attached to the elevated process:





Commonly abused privileges

Microsoft provides [documentation outlining the privilege constants in Windows](#). These privileges can be assigned directly to a user or inherited via group membership. While many of these privileges can be abused, the

following are the most commonly abused privilege constants in malicious software and attacker tradecraft:

1. **SeBackupPrivilege**

Description: This privilege causes the system to grant all read access control to any file, regardless of the access control list (ACL) specified for the file.

Attacker Tradecraft: Collection.

2. **SeCreateTokenPrivilege**

Description: Required to create a primary token.

Attacker Tradecraft: Privilege Escalation

3. **SeDebugPrivilege**

Description: Required to debug and adjust the memory of a process owned by another account.

Attacker Tradecraft: Privilege Escalation; Defense Evasion; Credential Access

4. **SeLoadDriverPrivilege**

Description: Required to load or unload a device driver.

Attacker Tradecraft: Persistence; Defense Evasion

5. **SeRestorePrivilege**

Description: Required to perform restore operations. This privilege causes the system to grant all write access control to any file, regardless of the ACL specified for the file.

Attacker Tradecraft: Persistence; Defense Evasion

6. **SeTakeOwnershipPrivilege**

Description: Required to take ownership of an object without being granted discretionary access.

Attacker Tradecraft: Persistence; Defense Evasion; Collection

7. **SeTcbPrivilege**

Description: This privilege identifies its holder as part of the trusted computer base. Some trusted protected subsystems are granted this privilege.

Attacker Tradecraft: Privilege Escalation

The “Abusing Token Privileges for LPE” whitepaper provides a comprehensive reference of privilege abuse techniques, refer to section “3.1 — Exploitable Privileges” for more information.

Privilege auditing and removal

Now that we've laid out some key concepts of privileges, let's walk through a representative example: identifying and mitigating abuse of the **debug programs** privilege (SeDebugPrivilege).

SeDebugPrivilege allows a process to inspect and adjust the memory of other processes, and has long been a security concern. SeDebugPrivilege allows the token bearer to access any process or thread, regardless of security descriptors. The Windows credential harvesting tool Lsadump uses this technique to provide processes with read access to the memory space of the Local System Authority (LSASS). Malware also abuses this privilege to perform code injection into otherwise trustworthy processes, because it permits the creation of new remote threads in a target process.

SeDebugPrivilege does have many legitimate use cases. Many administrative tools need to inspect the memory of other processes for troubleshooting or profiling. Likewise, many commercial applications that inject their own code into running processes on a system require SeDebugPrivilege for legitimate reasons. (For example, see this article that explains how Symantec Endpoint Protection relies on SeDebugPrivilege.)

Additional context on SeDebugPrivilege and its usage in malware can be found in several books and publications. Some of those we referenced were [The Art of Memory Forensics](#) (pages: 173, 186, 197–199), [Malware Analysts Cookbook](#) (pages: 58, 231, 589) and [Windows Malware Analysis Essentials](#) (page: 143).

Enabling privilege auditing

Let's now look at auditing as a technique for collecting the events necessary to identify potential privilege abuse. At Palantir, we use native Windows Event Forwarding (WEF) in order to collect audit logs in a central location. If you want to deploy WEF, please see [our prior blog post](#) and [GitHub repository](#) for configuration and management details.

The native event logging facilities in Windows 10 and Server 2016 support auditing privilege use within the operating system. Auditing of both [sensitive privilege use](#) and [non-sensitive privilege use](#) can be enabled via Group Policy Object (GPO) and collected via WEF subscriptions. Additionally, it's valuable to audit [special privileges assigned to new logons](#) to identify where privileged access tokens are being created.

In most environments we recommend that you collect only events related to sensitive privilege use and disable auditing of the use of backup and restore privileges. While these techniques can be used by a malicious actor as part of collection, persistence, and defense evasion techniques, they create a prohibitively large number of events.

With the correct audit GPO applied, we collect usage of the following privileges:

- Act as part of the operating system
- Create a token object
- Debug programs
- Enable computer and user accounts to be trusted for delegation
- Generate security audits
- Impersonate a client after authentication
- Load and unload device drivers
- Manage auditing and security log

- Modify firmware environment values
- Replace a process-level token
- Take ownership of files or other objects

Identifying privilege usage

Now that event logs have been collected into a centralized location, we can identify potentially abusable privilege primitives through targeted searches.

As we are collecting events with event code 4672 (*Special privileges assigned to new logon*), we can perform searches across our fleet to identify where user tokens with the SeDebugPrivilege are generated. An example event:

```
LogName=Security
SourceName=Microsoft Windows security auditing.
EventCode=4672
EventType=0
Type=Information
ComputerName=dane
TaskCategory=Special Logon
OpCode=Info
RecordNumber=17946067
Keywords=Audit Success
Message=Special privileges assigned to new logon.
```



```
Subject:
  Security ID:
  Account Name:      dane
  Account Domain:
  Logon ID:          0x5623BE0

Privileges:          SeSecurityPrivilege
                    SeTakeOwnershipPrivilege
                    SeLoadDriverPrivilege
                    SeBackupPrivilege
                    SeRestorePrivilege
                    SeDebugPrivilege
                    SeSystemEnvironmentPrivilege
                    SeImpersonatePrivilege
                    SeDelegateSessionUserImpersonatePrivilege
```

In this instance, the user account was granted the SeDebugPrivilege as part of a logon event. This indicates the user token generated on this machine may be targeted and abused by a malicious actor with system access.

If Authorization Policy Change auditing is enabled, we can additionally receive event notifications when token privileges are enabled or disabled. An example of the 4703 event (*A user right was adjusted*):

```
LogName=Security
SourceName=Microsoft Windows security auditing.
```

EventCode=4703
EventType=0
Type=Information
ComputerName=dane
TaskCategory=Authorization Policy Change
OpCode=Info
RecordNumber=161204239
Keywords=Audit Success
Message=A user right was adjusted.

Subject:

Security ID:
Account Name: dane
Account Domain:
Logon ID: 0x3E7

Target Account:

Security ID:
Account Name: dane
Account Domain:
Logon ID: 0x3E7

Process Information:

Process ID: 0xa64
Process Name:
C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe

Enabled Privileges:

SeDebugPrivilege

Disabled Privileges:

-

In this instance, the user account token was modified to enable the SeDebugPrivilege. While not inherently malicious, this could be indicative of adversary activity using the PowerShell binary to perform code injection or protected credential access.

Finally, event IDs 4673 (*A privileged service was called*) and 4674 (*An operation was attempted on a privileged object*) may contain additional context or other privilege calls. An example of the 4673 event:

```
LogName=Security
SourceName=Microsoft Windows security auditing.
EventCode=4673
EventType=0
Type=Information
ComputerName=dane
TaskCategory=Sensitive Privilege Use
OpCode=Info
RecordNumber=93434404
Keywords=Audit Failure
Message=A privileged service was called.
```

Subject:

```
Security ID:
Account Name:      dane
Account Domain:
Logon ID:          0xADF23180D
```

```
Service:
  Server:      Security
  Service Name: -

Process:
  Process ID:   0xf818
  Process Name:
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Service Request Information:
  Privileges:      SeTcbPrivilege
```

In this instance, the privilege `SeTcbPrivilege` was invoked by the PowerShell binary as a normal user. Adversaries can abuse the `SeTcbPrivilege` to generate a new token with additional privileges or features that are then used with impersonation.

Removing privileges across the fleet

Now that we've analyzed the `SeDebugPrivilege` event logs and validated they can be removed safely, we perform removal to ensure that only the users who need this privilege have it.

First, we create a security group in Active Directory (*SeDebug-Exceptions-sg*). Any users added to the security group can continue using the SeDebugPrivilege on their systems (e.g., administrators performing system-level debugging), while any other user loses the privilege (e.g., recruiting, help desk).

Next, we generate a Group Policy Object (GPO) and configure it to only assign the privileges for “Debug Programs” to users in the SeDebug-Exceptions-sg group. The setting can be configured at: **Computer Configuration\Windows Settings\Security Settings\Local Policies\User Rights Assignment**.

We then deploy the GPO to test machines and users throughout the fleet with security filtering.

Testing and validation

Once deployed to the test fleet, we conduct testing and validation exercises to identify any adverse impact or issues. Using data derived from Windows security, we conduct a granular whitelisting of potentially impacted user accounts. At the end of the test phase, not a single report or issue was

identified or attributed to the change. We can then apply the privilege removal GPO to the remainder of the fleet.

The image below is an administrative prompt from one of our machines. Note that the SeDebugPrivilege is no longer present in the token, even when associated with an elevated cmd.exe process:



Using a combination of Windows events and host-based scripts, we continue monitoring and tracking of the SeDebugPrivilege until we can validate the fleet had received the change and is stable.

Issues and limitations

Finally, let's discuss the limitations of the discussed privilege removal technique.

Firstly, not all privileges that are vulnerable to abuse can be removed (e.g. SeBackupPrivilege, SeImpersonatePrivilege, etc.) This technique should thus be considered one of many layers of a defense-in-depth strategy, not a panacea.

Secondly, modifying privileges does not restrict system-level accounts. In order for the operating system and associated tooling to function, these privileges are required and cannot be revoked. An example of this is the following screenshot of privileges associated with a primary access token for the SYSTEM user. Such an action would be captured in detection and alerting, but it's important to mention that there's no hard stop on obtaining the privilege on the system.



In this instance, an administrator user executed psexec to spawn a cmd.exe shell as NT AUTHORITY\SYSTEM. Note the presence of the SeDebugPrivilege in the associated privileges table for the token. If users are granted administrator rights to their machines, there are multiple mechanisms to bypass this security control.

Conclusion

While the presented technique will not by itself stop a determined attacker in their tracks, it is a valuable defense-in-depth control that can shut down automated malware functionality and break some out-of-the-box attacker tooling. Armed with an understanding of privileges and how attackers may abuse them, defenders can develop and implement enhanced detection and attack surface reduction capabilities for their fleets.

Further reading

- [Security Principals](#)
- [Enumerating remote access policies through GPO](#)
- [Abusing Token Privileges for LPE](#)

Authors

Chad D., Dane S., Tyler B.

Security

Palantirtech

Infosec

Information Security



191 claps



WRITTEN BY

Palantir

Follow



Palantir Blog

Palantir Blog

Follow

See responses (1)

More From Medium


More from Palantir Blog

Related reads

Related reads





Code Review Best Practices

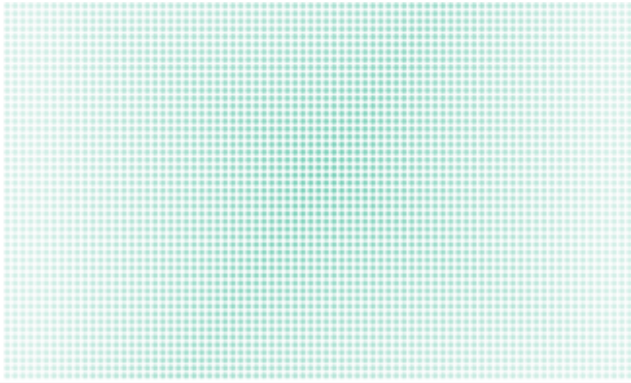


Palantir in Palantir...


Mar 4, 2018 · 12 min...

 11.2K







59 Hosts to Glory — Passing the OSCP

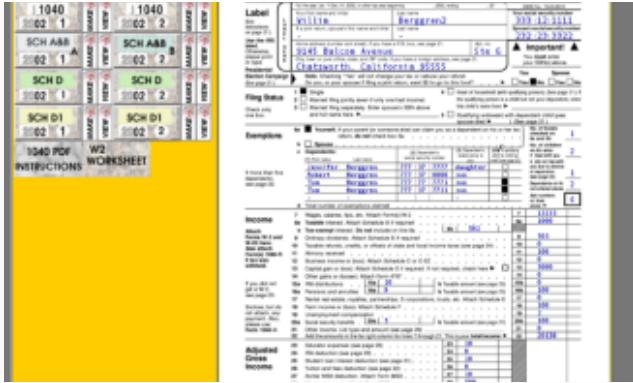


Tib3rius

Apr 30 · 8 min read ★

 485





VulnHub — Kioptrix: Level 5



Mike Bond

Oct 21, 2018 · 13 min...

 353

