4 Zugriffstoken

Das *Sicherheitszugriffstoken*, kurz *Zugriffstoken* oder *Token* genannt, ist das Herzstück der Sicherheit in Windows. Der Sicherheitsreferenzmonitor (SRM) verwendet Token, um Identitäten zu repräsentieren, wie beispielsweise Benutzerkonten, und gewährt oder verweigert ihnen dann den Zugriff auf Ressourcen. Windows stellt Token durch `Token`-Kernel-Objekte dar, die mindestens die spezifische Identität, die sie repräsentieren, alle Sicherheitsgruppen, denen die Identität angehört, und die besonderen Privilegien, die der Identität gewährt wurden, enthalten.

Wie andere Kernel-Objekte unterstützen Token die `Query`- und `Set`-Systemaufrufe, mit denen der Benutzer die Eigenschaften eines Tokens überprüfen und bestimmte Eigenschaften festlegen kann. Obwohl sie weniger häufig verwendet werden, stellen einige Win32-API-Funktionen diese `Set`- und `Query`-Systemaufrufe ebenfalls zur Verfügung: zum Beispiel `GetTokenInformation` und `SetTokenInformation`.

Beginnen wir mit einem Überblick über die beiden Haupttypen von Token, auf die Sie bei der Analyse der Sicherheit eines Windows-Systems stoßen werden: primäre Token und Token für Identitätswechsel. Dann werden wir viele der wichtigen Eigenschaften eines Tokens detailliert beschreiben. Diese müssen Sie verstehen, bevor wir in Kapitel 7 die Zugriffsüberprüfung besprechen können.

4.1 Primäre Token

Jeder Prozess hat ein zugewiesenes Token, das seine Identität für jeden Ressourcenzugriffsvorgang beschreibt. Wenn der SRM eine Zugriffsprüfung durchführt, fragt er das Token des Prozesses ab und verwendet es, um zu bestimmen, welche Art von Zugriff gewährt werden soll. Wenn ein Token für einen Prozess verwendet wird, nennt man es ein primäres Token.

Sie können das Token eines Prozesses mit dem Systemaufruf `NtOpenProcessToken` öffnen, der ein Handle zurückgibt, mit dem Sie Token-Informationen abfragen können.

Da das Token-Objekt eine sichere Ressource ist, muss der Aufrufer eine Zugriffsprüfung bestehen, um das Handle zu erhalten. Beachten Sie, dass Sie auch ein Handle für den Prozess mit `QueryLimitedInformation`-Zugriff benötigen, um das Token abfragen zu können.

Wenn Sie ein Token-Objekt öffnen, können Sie die folgenden Zugriffsrechte anfordern:

- **`AssignPrimary`** Erforderlich, um ein primäres Token an einen Prozess anzuhängen
- **`Duplicate`** Erforderlich, um ein Token-Objekt zu duplizieren
- **`Impersonate`** Erforderlich, um einem Prozess ein Identitätswechseltoken anzufügen
- **`Query`** Erforderlich, um die Eigenschaften des Token-Objekts abzufragen, z. B. seine Gruppen und Berechtigungen
- **`QuerySource`** Erforderlich, um die Quelle des Token-Objekts abzufragen
- **`AdjustPrivileges`** Erforderlich, um die Berechtigungen in einem Token-Objekt anzupassen
- **`AdjustGroup`** Erforderlich, um die Gruppenliste eines Token-Objekts anzupassen
- **`AdjustDefault`** Erforderlich, um die Eigenschaften eines Token-Objekts anzupassen, die nicht durch die anderen Zugriffsrechte abgedeckt sind
- **`AdjustSessionId`** Erforderlich, um die Sitzungs-ID des Token-Objekts anzupassen

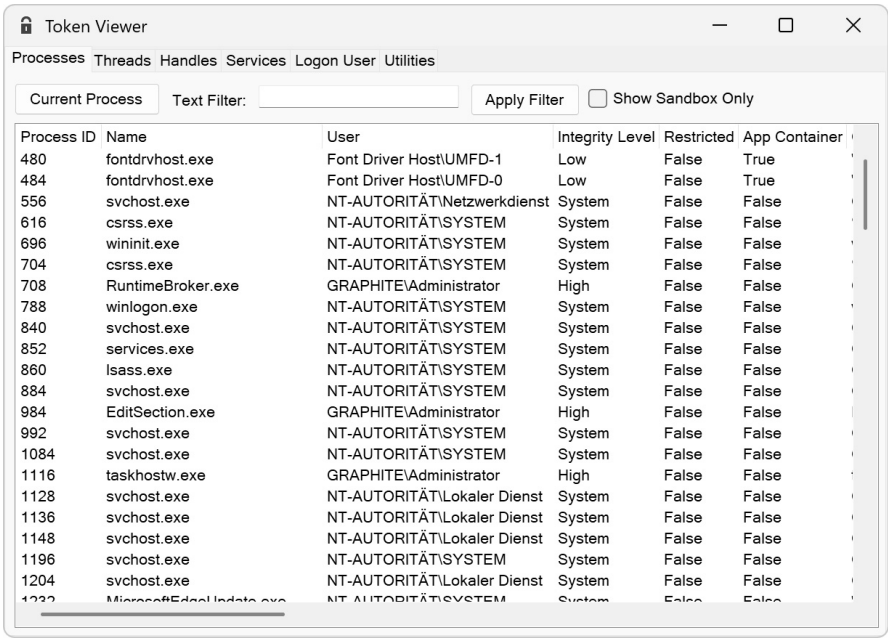

Abb. 4.1 *Der Token Viewer listet alle zugänglichen Prozesse und ihre Token auf.*

Sie können eine Liste der zugänglichen Prozesse und ihrer Token anzeigen, indem Sie den PowerShell-Befehl `Show-NtToken -All` ausführen. Dadurch wird die Anwendung Token Viewer geöffnet, wie in Abbildung 4.1 dargestellt.

Die Listenansicht bietet nur einen einfachen Überblick über die verfügbaren Token. Wenn Sie mehr Informationen sehen möchten, doppelklicken Sie auf einen der Prozesseinträge, um eine detaillierte Ansicht des Tokens aufzurufen, wie in Abbildung 4.2 dargestellt.

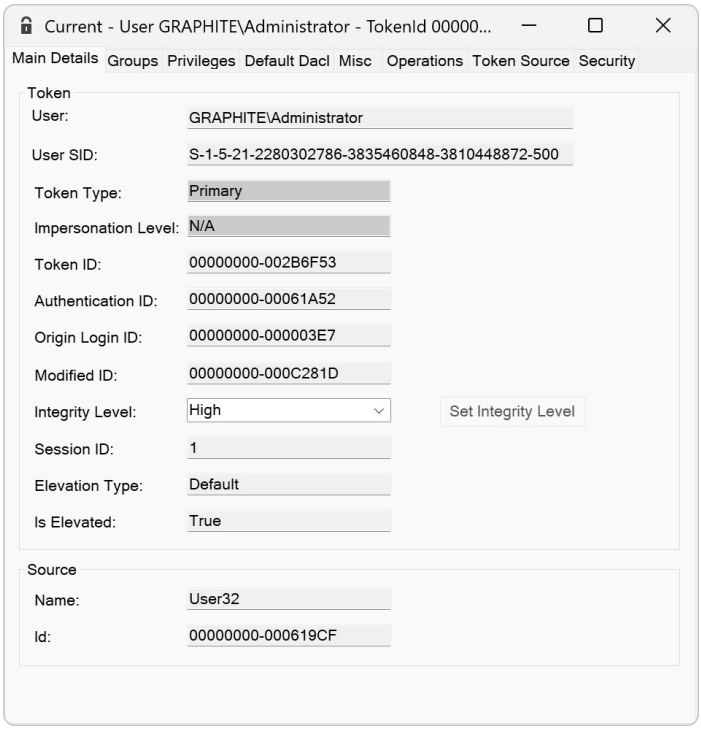

Abb. 4.2 *Die Detailansicht für das Token-Objekt eines Prozesses*

Lassen Sie uns ein paar wichtige Informationen in dieser Ansicht hervorheben. Ganz oben stehen der Name und die SID des Benutzers. Das Token-Objekt speichert nur die SID, aber die Token-Ansicht zeigt den Namen an, wenn er verfügbar ist. Das nächste Feld gibt den Typ des Tokens an. Da es sich um ein primäres Token handelt, ist der Typ auf `Primary` gesetzt. Die darunter liegende Ebene des Identitätswechsels (Impersonation Level) wird nur für Identitätswechseltoken verwendet, die wir im nächsten Abschnitt besprechen werden. Für primäre Token wird sie nicht benötigt, daher ist sie auf N/A gesetzt.

In der Mitte des Dialogfelds befindet sich eine Liste mit vier 64-Bit-Ganzzahlbezeichnern:

- **`Token ID`** Ein eindeutiger Wert, der bei der Erstellung des Token-Objekts zugewiesen wurde
- **`Authentication ID`** Ein Wert, der angibt, zu welcher Anmeldesitzung das Token gehört

- **`Origin Login ID`** Der Authentifizierungsbezeichner der übergeordneten Anmeldesitzung
- **`Modified ID`** Ein eindeutiger Wert, der aktualisiert wird, wenn bestimmte Token-Werte geändert werden

LSASS erstellt eine *Anmeldesitzung*, wenn sich ein Benutzer an einem Windows-Rechner authentifiziert. Die Anmeldesitzung verfolgt authentifizierungsbezogene Ressourcen für einen Benutzer; sie speichert zum Beispiel eine Kopie der Anmeldeinformationen des Benutzers, damit sie wiederverwendet werden können. Während des Erstellungsprozesses der Anmeldesitzung generiert der SRM eine eindeutige Authentifizierungs-ID, die zur Referenzierung der Sitzung verwendet werden kann. Daher haben für eine bestimmte Anmeldesitzung alle Benutzertoken dieselbe Authentifizierungs-ID. Wenn sich ein Benutzer zweimal an demselben Rechner authentifiziert, generiert der SRM unterschiedliche Authentifizierungs-IDs.

Die ursprüngliche Anmeldekennung (Origin Login ID) gibt an, wer die Anmeldesitzung des Tokens erstellt hat. Wenn Sie einen anderen Benutzer auf Ihrem Desktop authentifizieren (indem Sie z.B. die `LogonUser`-API mit einem Benutzernamen und einem Kennwort aufrufen), dann dient die ursprüngliche Anmeldekennung als Authentifizierungs-ID des aufrufenden Tokens.

Beachten Sie, dass dieses Feld in Abbildung 4.2 den Wert `00000000-000003E7` anzeigt. Dies ist eine von vier festen Authentifizierungskennungen, die vom SRM definiert wurden und in diesem Fall die SYSTEM-Anmeldesitzung bezeichnen. Tabelle 4.1 zeigt die vier festen Werte zusammen mit den SIDs für die mit den Sitzungen verbundenen Benutzerkonten.

Authentifizierungs-ID	Benutzer-SID	Benutzername der Anmeldesitzung
00000000-000003E4	S-1-5-20	NT-AUTORITÄT\NETZWERKDIENST
00000000-000003E5	S-1-5-19	NT-AUTORITÄT\LOKALER DIENST
00000000-000003E6	S-1-5-7	NT-AUTORITÄT\ANONYMOUS-ANMELDUNG
00000000-000003E7	S-1-5-18	NT-AUTORITÄT\SYSTEM

Tab. 4.1 *Authentifizierungs-IDs und Benutzer-SIDs für feste Anmeldesitzungen*

Nach den IDs in der Detailansicht befindet sich ein Feld, das die Integritätsebene des Tokens angibt (Integrity Level). Windows Vista fügte die Integritätsebene erstmals hinzu, um einen einfachen obligatorischen Zugriffskontrollmechanismus zu implementieren, bei dem systemweite Richtlinien den Zugriff auf Ressourcen erzwingen, anstatt einer einzelnen Ressource zu erlauben, ihren Zugriff zu spezifizieren. Wir werden die Integritätsebenen im Abschnitt 4.5 besprechen.

Danach folgt die Sitzungs-ID (Session ID), eine Nummer, die der Konsolensitzung zugewiesen wird, mit der der Prozess verbunden ist. Auch wenn die Konsolensitzung Bestandteil des Prozesses ist, wird der Wert im Token des Prozesses angegeben.

Lokal eindeutige Bezeichner

Ich habe erwähnt, dass die Bezeichner eines Tokens 64-Bit-Ganzzahlen sind. Technisch gesehen, sind sie Strukturen des Typs lokal eindeutige Bezeichner *(Locally Unique Identifier, LUID),* die zwei 32-Bit-Werte enthalten. LUIDs sind ein gängiger Systemtyp, und der SRM verwendet sie, wenn er einen eindeutigen Wert benötigt. Sie werden beispielsweise zur eindeutigen Identifizierung von Berechtigungswerten verwendet.

Sie können Ihre eigene LUID zuweisen, indem Sie den Systemaufruf `NtAllocateLocallyUniqueId` aufrufen oder den PowerShell-Befehl `Get-NtLocallyUniqueId` ausführen. Wenn Sie einen Systemaufruf verwenden, stellt Windows sicher, dass es für das Erzeugen der nächsten eindeutigen ID eine zentrale Autorität gibt. Dies ist wichtig, da die Wiederverwendung eines Wertes katastrophale Folgen haben kann. Wenn beispielsweise eine LUID als Authentifizierungs-ID für ein Token wiederverwendet würde, könnte sie sich mit einem der in Tabelle 4.1 definierten Bezeichner überschneiden. Dies könnte das System austricksen, sodass es davon ausgeht, dass ein privilegierterer Benutzer auf eine Ressource zugreift, was zu einer Rechteausweitung führt.

Die grafische Benutzeroberfläche von Token Viewer eignet sich hervorragend, wenn Sie die Informationen eines Tokens manuell überprüfen möchten. Für programmatischen Zugriff können Sie in PowerShell mit dem Befehl `Get-NtToken` ein Token-Objekt öffnen. Verwenden Sie den folgenden Befehl, um das Token des aktuellen Prozesses abzurufen:

```
PS> $token = Get-NtToken
```

Wenn Sie das Token für einen bestimmten Prozess öffnen möchten, können Sie diesen Befehl verwenden, wobei Sie *<PID>* durch die Prozess-ID des Zielprozesses ersetzen:

```
PS> $token = Get-NtToken -ProcessId <PID>
```

Das Ergebnis des Befehls **`Get-NtToken`** ist ein *Token*-Objekt, dessen Eigenschaften Sie abfragen können. Sie können zum Beispiel den Benutzer des Tokens anzeigen, wie in Listing 4.1 gezeigt.

```
PS> $token.User
Name           Attributes
----           ----------
GRAPHITE\user  None
```

Listing 4.1 *Anzeige des Benutzers über die Eigenschaften eines* `Token`*-Objekts*

Verwenden Sie den Befehl `Format-NtToken`, um grundlegende Informationen auf der Konsole auszugeben, wie in Listing 4.2 gezeigt.

```
PS> Format-NtToken $token -All
USER INFORMATION
----------------
Name           Attributes
----           ----------
GRAPHITE\user  None
```

```
GROUP SID INFORMATION
---------------------
Name           Attributes
----           ----------
GRAPHITE\None  Mandatory, EnabledByDefault
Jeder          Mandatory, EnabledByDefault
--gekürzt--
```

Listing 4.2 *Eigenschaften eines Tokens mit* `Format-NtToken` *anzeigen*

Sie können das geöffnete Token-Objekt an `Show-NtToken` übergeben, um die in Listing 4.2 gezeigte Benutzeroberfläche anzuzeigen.

4.2 Identitätswechseltoken

Der andere Token-Typ, den Sie kennen sollten, ist das *Identitätswechseltoken (Impersonation Token)*. Identitätswechseltoken sind vor allem für Systemdienste wichtig, da sie es einem Prozess mit einer Identität ermöglichen, sich zum Zweck einer Zugriffsprüfung vorübergehend als eine andere Identität auszugeben. Zum Beispiel könnte ein Dienst eine Datei öffnen müssen, die einem anderen Benutzer gehört, während er eine Operation durchführt. Indem das System dem Dienst erlaubt, sich als der aufrufende Benutzer auszugeben, gewährt es ihm Zugriff auf die Datei, auch wenn der Dienst die Datei nicht direkt öffnen könnte.

Identitätswechseltoken werden Threads zugewiesen, nicht Prozessen. Das bedeutet, dass nur der Code, der in diesem Thread läuft, die verkörperte Identität annimmt. Es gibt drei Möglichkeiten, wie ein Identitätswechseltoken einem Thread zugewiesen werden kann:

- indem einem Token-Objekt explizit `Impersonate`-Zugriff und einem `Thread`-Objekt `SetThreadToken`-Zugriff gewährt wird
- indem Sie einem `Thread`-Objekt explizit `DirectImpersonation`-Zugriff gewähren
- implizit, indem Sie einen Identitätswechsel für eine RPC-Anfrage vornehmen

Eine implizite Token-Zuweisung ist am wahrscheinlichsten, da dies der häufigste Fall bei Systemdiensten ist, die RPC-Mechanismen bereitstellen. Wenn ein Dienst beispielsweise einen Named-Pipe-Server erstellt, kann er Clients, die sich mit der Pipe verbinden, über die `ImpersonateNamedPipe`-API mit einer anderen Identität ausstatten. Bei einem Aufruf der Named Pipe erfasst der Kernel einen Identitätswechselkontext, der auf dem aufrufenden Thread und Prozess basiert. Dieser Identitätswechselkontext wird verwendet, um dem Thread, der `ImpersonateNamedPipe` aufruft, ein Identitätswechseltoken zuzuweisen. Der Identitätswechselkontext kann entweder auf einem bestehenden Identitätswechseltoken des Threads oder einer Kopie des primären Tokens des Prozesses basieren.

4.2.1 Sicherheitsdienstqualität

Was ist, wenn Sie dem Dienst nicht die Möglichkeit geben wollen, sich als Ihre Identität auszugeben? Der SRM unterstützt eine Funktion namens *Sicherheitsdienstqualität (Security Quality of Service, SQoS),* mit der Sie dies steuern können. Wenn Sie eine benannte Pipe über die Dateisystem-APIs öffnen, können Sie im Feld `SecurityQualityOfService` der `OBJECT_ATTRIBUTES`-Struktur eine `SECURITY_QUALITY_OF_SERVICE`-Struktur übergeben. Die SQoS-Struktur enthält drei Konfigurationswerte: die Identitätswechselebene, den Nachverfolgungsmodus und den effektiven Token-Modus.

Die *Identitätswechselebene* im SQoS ist das wichtigste Feld, um zu steuern, was ein Dienst mit Ihrer Identität tun kann. Sie definiert die Zugriffsebene, die dem Dienst gewährt wird, wenn er sich implizit als der Anrufer ausgibt. Die Ebene kann einen von vier Werten annehmen, in aufsteigender Reihenfolge der Berechtigung:

1. **Anonym:** verhindert, dass der Dienst das `Token`-Objekt öffnet und die Identität des Benutzers abfragt. Dies ist die niedrigste Ebene; nur eine begrenzte Anzahl von Diensten würde funktionieren, wenn der Aufrufer diese Ebene angibt.
2. **Identifizieren (Identification):** ermöglicht es dem Dienst, das `Token`-Objekt zu öffnen und die Identität, Gruppen und Berechtigungen des Benutzers abzufragen. Der Thread kann jedoch keine gesicherten Ressourcen öffnen, während er die Identität des Benutzers verkörpert.
3. **Identität annehmen (Impersonation):** ermöglicht es dem Dienst, die Identität des Benutzers auf dem lokalen System vollständig auszuüben. Der Dienst kann die vom Benutzer gesicherten lokalen Ressourcen öffnen und sie manipulieren. Er kann auch auf Remote-Ressourcen für den Benutzer zugreifen, wenn sich der Benutzer lokal am System authentifiziert hat. Wenn sich der Benutzer jedoch über eine Netzwerkverbindung authentifiziert hat, z. B. über das SMB-Protokoll (Server Message Block), kann der Dienst das `Token`-Objekt nicht für den Zugriff auf entfernte Ressourcen verwenden.
4. **Delegieren (Delegation):** ermöglicht es dem Dienst, alle lokalen und entfernten Ressourcen so zu öffnen, als wäre er der Benutzer. Dies ist die höchste Ebene. Für den Zugriff auf eine Remote-Ressource durch netzwerkauthentifizierte Benutzer reicht es jedoch nicht aus, diese Identitätswechselebene zu haben. Die Windows-Domäne muss auch so konfiguriert sein, dass sie dies zulässt. In Kapitel 14, Kerberos-Authentifizierung, werden wir uns mit dieser Identitätswechselebene näher befassen.

Sie können die Identitätswechselebene im SQoS entweder beim Aufruf eines Dienstes oder beim Erstellen einer Kopie eines vorhandenen Tokens angeben. Um einzuschränken, was ein Dienst tun kann, geben Sie die Ebene Identifizieren oder Anonym an. Dadurch wird dem Dienst der Zugriff auf Ressourcen verwehrt, ob-

wohl der Dienst auf der Identitätswechselebene immer noch in der Lage ist, auf das Token zuzugreifen und Operationen im Namen des Anrufers durchzuführen.

Lassen Sie uns einen Test mit dem PowerShell-Befehl `Invoke-NtToken` ausführen. In Listing 4.3 übernehmen wir die Identität eines Tokens auf zwei verschiedenen Identitätswechselebenen und versuchen, ein Skript auszuführen, das eine gesicherte Ressource öffnet. Wir geben die Identitätswechselebene mit der Eigenschaft `ImpersonationLevel` an.

```
PS> $token = Get-NtToken
PS> Invoke-NtToken $token {
    Get-NtDirectory -Path "\"
} -ImpersonationLevel Impersonation
Name NtTypeName
---- ----------
     Directory

PS> Invoke-NtToken $token {
    Get-NtDirectory -Path "\"
} -ImpersonationLevel Identification
Get-NtDirectory : (0xC00000A5) - A specified impersonation level is invalid.
--gekürzt--
```

Listing 4.3 *Die Identität eines Tokens auf verschiedenen Ebenen übernehmen und eine gesicherte Ressource öffnen*

Mit dem ersten Befehl, den wir ausführen, erhalten wir einen Zugriff auf das primäre Token des aktuellen Prozesses. Anschließend rufen wir `Invoke-NtToken` auf, um für das Token auf der Ebene *Impersonation* den Identitätswechsel durchzuführen. Wir führen dann ein Skript aus, das `Get-NtDirectory` aufruft, um das OMNS-Stammverzeichnis zu öffnen. Das Öffnen ist erfolgreich, und wir geben das Verzeichnisobjekt auf der Konsole aus.

Wir versuchen dann, den Vorgang auf der Ebene *Identification* zu wiederholen, aber dieses Mal erhalten wir den Fehler `STATUS_BAD_IMPERSONATION_LEVEL`. (Wenn Sie diesen Fehler bei der Entwicklung einer Anwendung oder bei der Verwendung des Systems sehen, kennen Sie jetzt den Grund dafür!) Beachten Sie, dass der Vorgang Öffnen keinen »Zugriff verweigert«-Fehler zurückgibt, da der SRM nicht weit genug geht, um zu prüfen, ob der Nutzer der Identität auf die Ressource zugreifen kann.

Anonyme Benutzer

Die Angabe der Identitätswechselebene *Anonym* ist nicht dasselbe wie die Ausführung als Benutzer ANONYMOUS-ANMELDUNG, der in Listing 4.1 erwähnt wird. Es ist möglich, dass Sie etwas mit einer anonymen Benutzeridentität ausführen und im Rahmen der Zugriffsprüfung der Zugriff auf eine Ressource gewährt wird. Jedoch kann ein Token der Identitätswechselebene Anonym keine Zugriffsprüfung bestehen, und zwar unabhängig davon, wie die Sicherheit der Ressource konfiguriert ist.

→

Der Kernel implementiert den Systemaufruf `NtImpersonateAnonymousToken`, mit dem der angegebene Thread die Identität des anonymen Anmeldetokens des Systems annehmen kann. Sie können auch auf das Token des anonymen Benutzers mit `Get-NtToken` zugreifen:

```
PS> Get-NtToken -Anonymous | Format-NtToken
NT-AUTORITÄT\ANONYMOUS-ANMELDUNG
```

Die beiden anderen Felder im SQoS werden weniger häufig verwendet, sind aber dennoch wichtig. Der Modus für die *Kontextnachverfolgung* legt fest, ob die Identität des Benutzers statisch erfasst werden soll, wenn eine Verbindung zum Dienst hergestellt wird. Wenn die Identität nicht statisch erfasst wird und der Aufrufer dann die Identität eines anderen Benutzers annimmt, bevor er den Dienst aufruft, wird die neue übernommene Identität für den Dienst verfügbar, nicht für die Prozessidentität. Beachten Sie, dass die übernommene Identität nur dann an den Dienst weitergegeben werden kann, wenn sie die Ebene Identitätswechsel oder Delegierung verwendet. Befindet sich das übernommene Token auf der Ebene Identifizierung oder Anonym, erzeugt der SRM einen Sicherheitsfehler und lehnt den Identitätswechsel ab.

Der *effektive Token-Modus* ändert das an den Server übergebene Token auf andere Weise. Es ist möglich, Gruppen und Berechtigungen vor einem Anruf zu deaktivieren. Wenn der effektive Token-Modus deaktiviert ist, kann der Server diese Gruppen und Berechtigungen wieder aktivieren und verwenden. Wenn jedoch der effektive Token-Modus aktiviert ist, entfernt der SRM die Gruppen und Berechtigungen, sodass der Server diese nicht wieder aktivieren oder verwenden kann.

Wenn beim Öffnen des IPC-Kanals, des Kanals zur prozessübergreifenden Kommunikation (Inter Process Communication), keine SQoS-Struktur angegeben wird, wird standardmäßig die Ebene Identitätswechsel mit statischer Nachverfolgung und einem nicht effektiven Token verwendet. Wenn ein Identitätswechselkontext erfasst wird und der Aufrufer bereits eine andere Identität angenommen hat, muss der Grad des Identitätswechsels des Thread-Tokens größer oder gleich sein als der Grad des Identitätswechsels; andernfalls wird die Nachverfolgung fehlschlagen. Dies gilt auch dann, wenn der SQoS die Ebene Identifizierung anfordert. Hierbei handelt es sich um ein wichtiges Sicherheitsmerkmal; es verhindert, dass ein Aufrufer auf der Ebene Identifizierung Aufrufe über einen RPC-Kanal vornimmt und vorgibt, ein anderer Benutzer zu sein.

Hinweis Ich habe beschrieben, wie SQoS auf der Ebene der nativen Systemaufrufe spezifiziert wird, da die Struktur `SECURITY_QUALITY_OF_SERVICE` von den Win32-APIs nicht direkt angezeigt wird. Stattdessen wird sie in der Regel mit zusätzlichen Flags angegeben. `CreateFile` stellt SQoS beispielsweise durch das Flag `SECURITY_SQOS_PRESENT` bereit.

4.2.2 Expliziter Token-Identitätswechsel

Es gibt zwei Möglichkeiten, um explizit einen Token-Identitätswechsel durchzuführen. Wenn Sie ein Handle auf Token-Objekt mit Identitätswechselrechten haben, können Sie es mit dem Systemaufruf `NtSetInformationThread` und der Informationsklasse `ThreadImpersonationToken` einem Thread zuweisen.

Wenn Sie stattdessen einen Thread haben, für den Sie mit dem Zugriff `Direct Impersonation` einen Identitätswechsel durchführen wollen, können Sie den anderen Mechanismus verwenden. Mit dem Handle zu einem Quell-Thread können Sie den Systemaufruf `NtImpersonateThread` aufrufen und einem anderen Thread ein Impersonation-Token zuweisen. Die Verwendung von `NtImpersonateThread` ist eine Mischung aus explizitem und impliziten Identitätswechsel. Der Kernel erfasst einen Identitätswechselkontext, als ob der Quell-Thread den Aufruf über eine benannte Pipe vorgenommen hätte. Sie können sogar die SQoS-Struktur für den Systemaufruf angeben.

Sie denken vielleicht, dass Identitätswechsel eine riesige Sicherheits-Hintertür öffnet. Wenn ich meine eigene benannte Pipe einrichte und einen privilegierten Prozess dazu bringe, sich mit mir zu verbinden, und der Aufrufer SQoS nicht so einstellt, dass der Zugriff beschränkt wird, kann ich dann nicht erhöhte Privilegien erlangen? Wir werden im Abschnitt 4.12 darauf zurückkommen, wie dies verhindert wird.

4.3 Typ des Tokens konvertieren

Sie können zwischen den beiden Token-Typen konvertieren, indem Sie das Duplizieren verwenden. Wenn Sie ein Token duplizieren, erstellt der Kernel ein neues Token-Objekt und fertigt eine detaillierte Kopie aller Eigenschaften des Objekts an. Während das Token dupliziert wird, können Sie seinen Typ ändern.

Diese Duplizierung unterscheidet sich von der Handle-Duplizierung, die wir in Kapitel 3 besprochen haben, da die Duplizierung eines Handles für ein Token lediglich ein neues Handle erzeugen würde, das auf dasselbe Token-Objekt zeigt. Um das eigentliche `Token`-Objekt zu duplizieren, müssen Sie für das Handle über das Zugriffsrecht `Duplicate` verfügen.

Sie können dann entweder den Systemaufruf `NtDuplicateToken` oder den PowerShell-Befehl `Copy-NtToken` verwenden, um das Token zu duplizieren. Verwenden Sie beispielsweise das Skript in Listing 4.4, um auf Basis eines vorhandenen Tokens ein Identitätswechseltoken auf der Ebene Delegieren zu erstellen.

```
PS> $imp_token = Copy-NtToken -Token $token -ImpersonationLevel ↵
    Delegation
PS> $imp_token.ImpersonationLevel
Delegation

PS> $imp_token.TokenType
Impersonation
```

Listing 4.4 *Duplizieren eines Tokens zur Erstellung eines Identitätswechseltokens*

Sie können das Identitätswechseltoken wieder in ein primäres Token umwandeln, indem Sie erneut `Copy-NtToken` verwenden, wie in Listing 4.5 gezeigt.

```
PS> $pri_token = Copy-NtToken -Token $imp_token -Primary
PS> $pri_token.TokenType
Primary

PS> $pri_token.ImpersonationLevel
Delegation
```

Listing 4.5 *Ein Identitätswechseltoken in ein primäres Token umwandeln*

Interessant an der Ausgabe ist, dass das neue primäre Token die gleiche Identitätswechselebene besitzt wie das ursprüngliche Token. Das liegt daran, dass der SRM nur die Eigenschaft `TokenType` berücksichtigt; wenn das Token ein primäres Token ist, wird die Identitätswechselebene ignoriert.

Da wir ein Identitätswechseltoken wieder in ein primäres Token umwandeln können, stellen Sie sich vielleicht die Frage: Könnten wir ein Token der Ebene Identifizieren oder Anonym wieder in ein primäres Token umwandeln, einen neuen Prozess erstellen und die SQoS-Einstellungen umgehen? Versuchen wir das in Listing 4.6.

```
PS> $imp_token = Copy-NtToken -Token $token -ImpersonationLevel ↵
Identification
PS> $pri_token = Copy-NtToken -Token $imp_token -Primary
Ausnahme beim Aufrufen von "DuplicateToken" mit 5 Argument(en):  "(0xC00000A5)
- Eine angegebene Ebene für Identitätswechsel ist ungültig."
```

Listing 4.6 *Ein Token der Ebene Identifizieren wieder zu einem primären Token duplizieren*

Dieses Listing zeigt, dass wir ein Token der Ebene Identifizieren nicht in ein primäres Token duplizieren können. Die zweite Zeile löst eine Ausnahme aus, da der Vorgang eine Sicherheitsgarantie des SRM verletzen würde (nämlich die, dass der SQoS dem Aufrufer die Kontrolle über die Verwendung seiner Identität erlaubt).

Ein letzter Hinweis: Wenn Sie ein Token mit `Get-NtToken` öffnen, können Sie den Duplizierungsvorgang in einem Schritt durchführen, indem Sie den Parameter `Duplicate` angeben.

4.4 Pseudo-Token-Handle

Um auf ein Token zuzugreifen, müssen Sie ein Handle auf das `Token`-Objekt öffnen und dann daran denken, das Handle nach der Verwendung wieder zu schließen. Windows 10 hat drei *Pseudo-Handles* eingeführt, mit denen Sie Token-Informationen abfragen können, ohne ein vollständiges Handle auf ein Kernel-Objekt zu öffnen. Hier sind diese drei Handles, mit ihren Handle-Werten in Klammern:

- **Primary** (**-4**) Das primäre Token für den aktuellen Prozess
- **Impersonation** (**-5**) Das Identitätswechseltoken für den aktuellen Thread; schlägt fehl, wenn der Thread keine andere Identität angenommen hat
- **Effective** (**-6**) Das Identitätswechseltoken für den aktuellen Thread, wenn er keine andere Identität angenommen hat; ansonsten das primäre Token

Im Gegensatz zu den Pseudo-Handles für den aktuellen Prozess und den aktuellen Thread können Sie diese Token-Handles nicht duplizieren; Sie können sie nur für bestimmte eingeschränkte Zwecke verwenden, z.B. zur Abfrage von Informationen oder zur Durchführung von Zugriffsprüfungen. Der Befehl `Get-NtToken` kann diese Handles zurückgeben, wenn Sie den Parameter `Pseudo` angeben, wie in Listing 4.7 gezeigt.

```
PS> Invoke-NtToken -Anonymous ↵
    {Get-NtToken -Pseudo -Primary | Get-NtTokenSid}
Name          Sid
----          ---
GRAPHITE\user S-1-4-21-2318445812-3516008893-216915059-1002 ❶

PS> Invoke-NtToken -Anonymous ↵
    {Get-NtToken -Pseudo -Impersonation | Get-NtTokenSid}
Name                            Sid
----                            ---
NT-AUTORITÄT\ANONYMOUS-ANMELDUNG S-1-4-7 ❷

PS> Invoke-NtToken -Anonymous ↵
    {Get-NtToken -Pseudo -Effective | Get-NtTokenSid}
Name                            Sid
----                            ---
NT-AUTORITÄT\ANONYMOUS-ANMELDUNG S-1-4-7 ❸

PS> Invoke-NtToken -Anonymous ↵
    {Get-NtToken -Pseudo -Effective} | Get-NtTokenSid
Name          Sid
----          ---
GRAPHITE\user S-1-4-21-2318445812-3516008893-216915059-1002 ❹
```

Listing 4.7 *Abfrage von Pseudo-Token*

Hier werden die drei Arten von Pseudo-Token abgefragt, während wir die Identität des anonymen Benutzers angenommen haben. Der erste Befehl fragt das primäre Token ab und extrahiert dessen Benutzer-SID ❶. Der nächste Befehl fragt das Identitätswechseltoken ab, das die SID des anonymen Benutzers zurückgibt ❷. Dann wird das effektive Token abgefragt, das, da wir den anonymen Benutzer verkörpern, auch die SID des anonymen Benutzers zurückgibt ❸. Schließlich wird das effektive Token erneut abgefragt, wobei diesmal gewartet wird, bis der Skriptblock ausgeführt wurde, um die Benutzer-SID zu extrahieren. Dieser Vorgang gibt die Benutzer-SID ❹ des primären Tokens zurück, was zeigt, dass das Pseudo-Token kontextsensitiv ist.

4.5 Token-Gruppen

Müssten Administratoren jede Ressource für jeden möglichen Benutzer sichern, wäre die Identitätssicherheit zu schwerfällig zu verwalten. *Gruppen* ermöglichen es Benutzern, eine breitere Sicherheitsidentität zu teilen. Die meisten Operationen für die Zugriffsprüfungen unter Windows gewähren den Zugriff auf Gruppen und nicht auf einzelne Benutzer.

Aus der Sicht des SRM ist eine Gruppe nur eine weitere SID, die möglicherweise den Zugriff auf eine Ressource definiert. Wir können die Gruppen in der PowerShell-Konsole mit dem Befehl `Get-NtTokenGroup` anzeigen, wie in Listing 4.8 gezeigt.

```
PS> Get-NtTokenGroup $token
Name                            Attributes
----                            ----------
GRAPHITE\None                   Mandatory, EnabledByDefault, Enabled
Jeder                           Mandatory, EnabledByDefault, Enabled
VORDEFINIERT\Administratoren    UseForDenyOnly
VORDEFINIERT\Benutzer           Mandatory, EnabledByDefault, Enabled
NT-AUTORITÄT\INTERAKTIV         Mandatory, EnabledByDefault, Enabled
--gekürzt--
```

Listing 4.8 *Abfrage der Gruppen des aktuellen Tokens*

Wir können `Get-NtTokenGroup` auch verwenden, um nach bestimmten Attribut-Flags zu filtern, indem wir den Parameter `Attributes` angeben. Tabelle 4.2 zeigt die möglichen Attribut-Flags, die an den Befehl übergeben werden können.

Attributname im SDK	Attributname in PowerShell
SE_GROUP_ENABLED	Enabled
SE_GROUP_ENABLED_BY_DEFAULT	EnabledByDefault
SE_GROUP_MANDATORY	Mandatory
SE_GROUP_LOGON_ID	LogonId
SE_GROUP_OWNER	Owner
SE_GROUP_USE_FOR_DENY_ONLY	UseForDenyOnly
SE_GROUP_INTEGRITY	Integrity
SE_GROUP_INTEGRITY_ENABLED	IntegrityEnabled
SE_GROUP_RESOURCE	Resource

Tab. 4.2 *Gruppenattribute im SDK- und PowerShell-Format*

In den folgenden Abschnitten wird beschrieben, was die einzelnen Flags bedeuten.

4.5.1 Enabled, EnabledByDefault und Mandatory

Das wichtigste Flag ist `Enabled`. Wenn es gesetzt ist, berücksichtigt der SRM die Gruppe bei der Zugriffsprüfung; andernfalls wird die Gruppe ignoriert. Jede Gruppe, bei der das Attribut `EnabledByDefault` gesetzt ist, wird automatisch aktiviert.

Es ist möglich, eine Gruppe mit dem Systemaufruf `NtAdjustGroupsToken` zu deaktivieren (d.h. von der Zugriffsprüfung auszuschließen), wenn Sie auf das Token-Handle über `AdjustGroups`-Zugriffsrechte verfügen; der PowerShell-Befehl `Set-NtTokenGroup` stellt diesen Systemaufruf zur Verfügung. Sie können jedoch keine Gruppen deaktivieren, für die das Flag `Mandatory` gesetzt ist. Dieses Flag ist für alle Gruppen im Token eines normalen Benutzers gesetzt, aber bestimmte Systemtoken haben nicht obligatorische Gruppen. Wenn eine Gruppe deaktiviert wird, wenn Sie ein Identitätswechseltoken über RPC übergeben und das Flag für den effektiven Token-Modus im SQoS gesetzt ist, wird das Identitätswechseltoken die Gruppe löschen.

4.5.2 LogonId

Das Flag `LogonId` identifiziert jede SID, die allen Token auf demselben Desktop zugewiesen wird. Wenn Sie beispielsweise einen Prozess mit dem Dienstprogramm *runas* unter einem anderen Benutzer ausführen, hat das Token des neuen Prozesses dieselbe Anmelde-SID wie der Aufrufer, auch wenn es sich um eine andere Identität handelt. Dieses Verhalten ermöglicht es dem SRM, den Zugriff auf sitzungsspezifische Ressourcen, wie beispielsweise das Sitzungsobjektverzeichnis, zu gewähren.

Die SID hat immer das Format `S-1-4-4-X-Y`, wobei X und Y die beiden 32-Bit-Werte der LUID sind, die bei der Erstellung der Authentifizierungssitzung vergeben wurde. Wir werden im nächsten Kapitel auf die Anmelde-SID zurückkommen und darauf, wo sie Anwendung findet.

4.5.3 Owner

Alle sicherbaren Ressourcen im System gehören entweder zu einer Gruppen-SID oder einer Benutzer-SID. Token besitzen die Eigenschaft `Owner` (Besitzer), die eine SID enthält, die bei der Erstellung einer Ressource als Standardbesitzer verwendet wird. Der SRM lässt nur eine bestimmte Gruppe von Benutzer-SIDs zu, die in der Eigenschaft `Owner` angegeben werden können: entweder die SID des Benutzers oder eine Gruppen-SID, die mit dem `Owner`-Flag gekennzeichnet ist.

Sie können die aktuelle `Owner`-Eigenschaft des Tokens abrufen oder festlegen, indem Sie die `Get-NtTokenSid` oder `Set-NtTokenSid` verwenden. In Listing 4.9 wird beispielsweise die SID des Besitzers aus dem aktuellen Token ermittelt und dann versucht, den Eigentümer zu setzen.

```
PS> Get-NtTokenSid $token -Owner
Name         Sid
----         ---
GRAPHITE\rgh S-1-4-21-818064984-378290696-2985406761-1002

PS> Set-NtTokenSid -Owner -Sid "S-1-2-3-4"
Ausnahme beim Festlegen von "Owner": "(0xC000005A) - Zeigt an, dass eine
bestimmte Sicherheitskennung möglicherweise nicht als Besitzer eines Objekts
zugewiesen wurde."
```

Listing 4.9 *SID des Token-Besitzers abrufen und setzen*

In diesem Fall schlägt der Versuch, die Eigenschaft `Owner` auf die SID `S-1-2-3-4` zu setzen, mit einer Ausnahme fehl, da diese nicht unsere aktuelle Benutzer-SID ist und sie auch nicht in unserer Liste der Gruppen enthalten ist.

4.5.4 UseForDenyOnly

Die Zugriffsprüfung des SRM gewährt oder verweigert den Zugriff auf eine SID. Wenn eine SID jedoch deaktiviert wird, nimmt sie nicht mehr an den Zugriffsgenehmigungs- oder -verweigerungsprüfungen teil, was zu einer falschen Zugriffsprüfung führen kann.

Lassen Sie uns ein einfaches Beispiel betrachten. Stellen Sie sich vor, es gibt zwei Gruppen: *Mitarbeiter* und *RemoteZugriff*. Ein Benutzer erstellt ein Dokument, das alle Mitarbeiter lesen können sollen, mit Ausnahme derjenigen, die remote auf das System zugreifen, da der Inhalt des Dokuments vertraulich ist und der Benutzer nicht möchte, dass er nach außen dringt. Das Dokument ist so konfiguriert, dass alle Mitglieder der Gruppe *Mitarbeiter* darauf zugreifen können, den Benutzern der Gruppe *RemoteZugriff* jedoch der Zugriff verweigert wird.

Stellen Sie sich nun vor, dass ein Benutzer, der beiden Gruppen angehört, beim Zugriff auf eine Ressource eine Gruppe deaktivieren könnte. Er könnte einfach den Remotezugriff deaktivieren, um auf der Grundlage seiner Zugehörigkeit zur Gruppe *Mitarbeiter* Zugriff auf das Dokument zu erhalten und so die Zugriffsbeschränkungen auf triviale Weise zu umgehen.

Aus diesem Grund wird es einem Benutzer nur selten erlaubt sein, Gruppen zu deaktivieren. In bestimmten Fällen, wie beispielsweise beim Sandboxing, möchten Sie jedoch die Möglichkeit haben, eine Gruppe zu deaktivieren, damit sie nicht für den Zugriff auf eine Ressource verwendet werden kann. Das Flag `UseForDenyOnly` löst dieses Problem. Wenn eine SID mit diesem Flag markiert ist, wird sie bei der Überprüfung des erlaubten Zugriffs nicht berücksichtigt, wohl aber bei der Überprüfung der Zugriffsverweigerung. Ein Benutzer kann seine eigenen Gruppen als `UseForDenyOnly` markieren, indem er sein Token filtert und es zum Erstellen eines neuen Prozesses verwendet. Die Filterung von Token wird bei der Betrachtung von eingeschränkten Token im Abschnitt 4.7 behandelt.

4.5.5 Integrity und IntegrityEnabled

Die Attribut-Flags `Integrity` und `IntegrityEnabled` zeigen an, dass eine SID die Integritätsebene (Verbindlichkeitsstufe) des Tokens repräsentiert und aktiviert ist. Gruppen-SIDs, die mit dem Attribut-Flag `Integrity` gekennzeichnet sind, speichern diese Integritätsebene als 32-Bit-Zahl in ihrer endgültigen RID. Die RID kann ein beliebiger Wert sein; im SDK gibt es jedoch sieben vordefinierte Ebenen, die Tabelle 4.3 zeigt. Nur die ersten sechs sind gebräuchlich und von einem Benutzerprozess aus zugänglich. Zur Angabe einer Integritäts-SID verwendet der SRM die Sicherheitsautorität `MandatoryLabel` (die den Wert 16 hat).

Integritätsebene (Wert)	Name im SDK	Name in PowerShell
0	SECURITY_MANDATORY_UNTRUSTED_RID	Untrusted
4096	SECURITY_MANDATORY_LOW_RID	Low
8192	SECURITY_MANDATORY_MEDIUM_RID	Medium
8448	SECURITY_MANDATORY_MEDIUM_PLUS_RID	MediumPlus
12288	SECURITY_MANDATORY_HIGH_RID	High
16384	SECURITY_MANDATORY_SYSTEM_RID	System
20480	SECURITY_MANDATORY_PROTECTED_PROCESS_RID	ProtectedProcess

Tab. 4.3 *Vordefinierte Werte für die Integritätsebene*

Die Standardstufe für einen Benutzer ist `Medium`. Administratoren wird in der Regel die Stufe `High` zugewiesen, und Dienste erhalten die Stufe `System`. Wir können die Integritäts-SID eines Tokens mit `Get-NtTokenSid` abfragen, wie in Listing 4.10 gezeigt.

```
PS> Get-NtTokenSid $token -Integrity
Name                                                    Sid
----                                                    ---
Verbindliche Beschriftung\Mittlere Verbindlichkeitsstufe S-1-16-8192
```

Listing 4.10 *Die SID der Integritätsebene eines Tokens abrufen*

Wir können auch eine neue Token-Integritätsebene festlegen, sofern sie kleiner oder gleich dem aktuellen Wert ist. Es ist auch möglich, die Ebene zu erhöhen, aber dies erfordert besondere Rechte und die Aktivierung von `SeTcbPrivilege`.

Obwohl Sie die gesamte SID setzen können, ist es in der Regel bequemer, nur den Wert zu setzen. Das Skript in Listing 4.11 setzt zum Beispiel die Integritätsebene eines Tokens auf die Ebene `Low`.

```
PS> Set-NtTokenIntegrityLevel Low -Token $token
PS> Get-NtTokenSid $token -Integrity
Name                                                    Sid
----                                                    ---
Verbindliche Beschriftung\Niedrige Verbindlichkeitsstufe S-1-16-8192
```

Listing 4.11 *Token-Integritätsebene auf* `Low` *festlegen*

Wenn Sie dieses Skript ausführen, werden Sie möglicherweise feststellen, dass in Ihrer PowerShell-Konsole Fehler aufgrund eines blockierten Dateizugriffs angezeigt werden. Warum der Dateizugriff blockiert wird, wird in Kapitel 7 im Rahmen der obligatorischen Integritätssteuerung erläutert.

4.5.6 Resource

Das letzte Attribut-Flag verdient nur eine kurze Erwähnung. Das Flag `Resource` zeigt an, dass die Gruppen-SID eine lokale Domänen-SID ist. Wir werden auf diesen SID-Typ in Kapitel 10 zurückkommen.

4.5.7 Gerätegruppen

Ein Token kann auch eine separate Liste von Gerätegruppen haben. Diese Gruppen-SIDs werden hinzugefügt, wenn sich ein Benutzer über ein Netzwerk in einer Unternehmensumgebung bei einem Server authentifiziert, wie in Listing 4.12 gezeigt.

```
PS> Get-NtTokenGroup -Device -Token $token
Name                          Attribute
----                          ---------
VORDEFINIERT\Benutzer         Mandatory, EnabledByDefault, Enabled
AD\CLIENT1$                   Mandatory, EnabledByDefault, Enabled
AD\Domänencomputer            Mandatory, EnabledByDefault,Enabled
NT-AUTORITÄT\Anspruchswert Mandatory, EnabledByDefault,Enabled
--gekürzt--
```

Listing 4.12 *Gerätegruppen mit* `Get-NtTokenGroup` *anzeigen*

Sie können die Gruppen auf dem Token abfragen, indem Sie `Get-NtTokenGroup` verwenden und den Parameter `Device` übergeben.

4.6 Privilegien

Mit Gruppen können Systemadministratoren den Zugriff eines Benutzers auf bestimmte Ressourcen kontrollieren. *Privilegien,* die manchmal auch *Berechtigungen* oder einfach nur Rechte genannt werden, werden einem Benutzer gewährt, damit er bestimmte Sicherheitsprüfungen für alle Arten von Ressourcen umgehen kann, z.B. durch Umgehung einer Zugriffsprüfung. Ein Privileg kann sich auch auf bestimmte privilegierte Aktionen beziehen, wie das Ändern der Systemzeit. Sie können die Privilegien eines Tokens in der Konsole mit `Get-NtTokenPrivilege` (Listing 4.13) anzeigen.

```
PS> Get-NtTokenPrivilege $token
Name                          Luid              Enabled
----                          ----              -------
SeShutdownPrivilege           00000000-00000013 False
SeChangeNotifyPrivilege       00000000-00000017 True
```

```
SeUndockPrivilege              00000000-00000019  False
SeIncreaseWorkingSetPrivilege  00000000-00000021  False
SeTimeZonePrivilege            00000000-00000022  False
```

Listing 4.13 *Token-Privilegien anzeigen*

Die Ausgabe ist in drei Spalten unterteilt. Die erste Spalte ist der allgemeine Name des Privilegs. Wie bei SIDs verwendet der SRM diesen Namen nicht direkt, sondern den LUID-Wert des Privilegs, der in der zweiten Spalte zu sehen ist. Die letzte Spalte gibt an, ob das Privilegs derzeit aktiviert ist. Privilegien können aktiviert oder deaktiviert sein.

Bei jeder Prüfung auf ein Privileg sollte sichergestellt werden, dass das Privileg aktiviert und nicht nur vorhanden ist. Unter bestimmten Umständen, wie beispielsweise beim Sandboxing, kann ein Token zwar ein Privileg enthalten, aber die Sandbox-Einschränkungen können verhindern, dass es als aktiviert markiert wird. Das `Enabled`-Flag ist eigentlich ein Satz von Attribut-Flags, wie die Attribute für die Gruppen-SIDs. Wir können diese Attribute anzeigen, indem wir die Ausgabe von `Get-NtTokenPrivilege` als Liste formatieren (Listing 4.14).

```
PS> Get-NtTokenPrivilege $token -Privileges ↵
SeChangeNotifyPrivilege | Format-List
Name        : SeChangeNotifyPrivilege
Luid        : 00000000-00000017
Attributes  : EnabledByDefault, Enabled
Enabled     : True
DisplayName : Auslassen der durchsuchenden Überprüfung
```

Listing 4.14 *Alle Eigenschaften des Privilegs* `SeChangeNotifyPrivilege` *anzeigen*

In der Ausgabe sehen wir nun die Attribute, die sowohl `Enabled` als auch `EnabledByDefault` umfassen. Das Attribut `EnabledByDefault` gibt an, ob der Standardstatus des Zugriffsrechts aktiviert werden soll. Wir sehen jetzt auch eine zusätzliche `DisplayName`-Eigenschaft, die dazu dient, einem Benutzer zusätzliche Informationen zu geben.

Um den Status der Berechtigungen eines Tokens zu ändern, benötigen Sie `AdjustPrivileges`-Zugriff auf das Token-Handle; dann können Sie den Systemaufruf `NtAdjustPrivilegesToken` verwenden, um die Attribute anzupassen und ein Privileg zu aktivieren oder zu deaktivieren. Die PowerShell-Befehle `Enable-NtTokenPrivilege` und `Disable-NtTokenPrivilege` stellen diesen Systemaufruf bereit, wie in Listing 4.15 gezeigt.

```
PS> Enable-NtTokenPrivilege SeTimeZonePrivilege -Token $token -PassThru
Name                 Luid                 Enabled
----                 ----                 -------
SeTimeZonePrivilege  00000000-00000022    True

PS> Disable-NtTokenPrivilege SeTimeZonePrivilege -Token $token -PassThru
Name                 Luid                 Enabled
----                 ----                 -------
SeTimeZonePrivilege  00000000-00000022    False
```

Listing 4.15 *Das Privileg* `SeTimeZonePrivilege` *aktivieren und deaktivieren*

Mit der API NtAdjustPrivilegesToken ist es auch möglich, ein Privileg durch Angabe des Remove-Attributs vollständig zu entfernen, was Sie mit dem PowerShell-Befehl Remove-NtTokenPrivilege erreichen können. Durch das Entfernen eines Privilegs wird sichergestellt, dass das Token diese nie wieder verwenden kann. Wenn Sie das Privileg nur deaktivieren, könnte es versehentlich wieder aktiviert werden. Listing 4.16 zeigt, wie Sie ein Privileg entfernen können.

```
PS> Get-NtTokenPrivilege $token -Privileges SeTimeZonePrivilege
Name                 Luid               Enabled
----                 ----               -------
SeTimeZonePrivilege  00000000-00000022  False

PS> Remove-NtTokenPrivilege SeTimeZonePrivilege -Token $token
PS> Get-NtTokenPrivilege $token -Privileges SeTimeZonePrivilege
WARNING: Couldn't get privilege SeTimeZonePrivilege
```

Listing 4.16 *Einem Token ein Privileg entziehen*

Um Privilegien zu prüfen, kann eine Benutzeranwendung den Systemaufruf NtPrivilegeCheck verwenden, während Kernel-Code die API SePrivilegeCheck aufrufen kann. Sie fragen sich vielleicht, ob Sie nicht einfach manuell prüfen können, ob ein Privileg aktiviert ist, anstatt einen speziellen Systemaufruf zu verwenden. In diesem Fall ja, aber wenn möglich lohnt es sich immer, die Systemfunktionen zu verwenden, für den Fall, dass Sie in Ihrer Implementierung einen Fehler machen oder einen Grenzfall nicht bedacht haben. Der PowerShell-Befehl Test-NtTokenPrivilege umhüllt den Systemaufruf, wie in Listing 4.17 gezeigt.

```
PS> Enable-NtTokenPrivilege SeChangeNotifyPrivilege
PS> Disable-NtTokenPrivilege SeTimeZonePrivilege
PS> Test-NtTokenPrivilege SeChangeNotifyPrivilege
True

PS> Test-NtTokenPrivilege SeTimeZonePrivilege, ↲
SeChangeNotifyPrivilege -All
False

PS> Test-NtTokenPrivilege SeTimeZonePrivilege, ↲
SeChangeNotifyPrivilege -All -PassResult
EnabledPrivileges          AllPrivilegesHeld
-----------------          -----------------
{SeChangeNotifyPrivilege}  False
```

Listing 4.17 *Privilegienüberprüfungen durchführen*

Dieses Listing demonstriert einige Beispiel-Privilegienüberprüfungen mit Test-NtTokenPrivilege. Wir beginnen damit, SeChangeNotifyPrivilege zu aktivieren und SeTimeZonePrivilege zu deaktivieren. Dies sind allgemeine Privilegien, die allen Benutzern gewährt werden, aber Sie müssen das Beispiel möglicherweise ändern, wenn Ihr Token nicht über diese Rechte verfügt.

Dann testen wir nur auf SeChangeNotifyPrivilege; es ist aktiviert, also gibt dieser Test True zurück. Als Nächstes prüfen wir sowohl auf SeTimeZonePrivilege als auch

auf SeChangeNotifyPrivilege; wir sehen, dass wir nicht alle Berechtigungen haben, also gibt Test-NtTokenPrivilege False zurück. Schließlich führen wir denselben Befehl aus, geben aber die Option -PassResult an, um das vollständige Prüfergebnis zu erhalten. In der Spalte EnabledPrivileges sehen wir, dass nur das Recht SeChange NotifyPrivilege aktiviert ist.

Im Folgenden sind einige der auf dem System verfügbaren Privilegien aufgeführt:

- **SeChangeNotifyPrivilege** Der Name dieses Privilegs ist irreführend. Es erlaubt einem Benutzer, Benachrichtigungen über Änderungen im Dateisystem oder in der Registrierung zu erhalten, aber es wird auch verwendet, um die durchsuchende Überprüfung zu umgehen. Wir werden die durchsuchende Überprüfung in Kapitel 8 besprechen.
- **SeAssignPrimaryTokenPrivilege** und **SeImpersonatePrivilege** Diese Privilegien ermöglichen es dem Benutzer, die Prüfungen für die Zuweisung des primären Tokens bzw. des Identitätswechseltokens zu umgehen. Im Gegensatz zu den meisten Privilegien in dieser Liste müssen diese für das primäre Token des aktuellen Prozesses aktiviert werden, nicht für ein Identitätswechseltoken.
- **SeBackupPrivilege** und **SeRestorePrivilege** Diese Rechte erlauben es dem Benutzer, die Zugriffsprüfung beim Öffnen bestimmter Ressourcen wie Dateien oder Registrierungsschlüssel zu umgehen. Dadurch kann der Benutzer Ressourcen sichern und wiederherstellen, ohne dass er explizit Zugriff auf sie erhalten muss. Diese Privilegien wurden auch für andere Zwecke verwendet: Das Restore-Privileg erlaubt es dem Benutzer beispielsweise, beliebige Registry-Hives zu laden.
- **SeSecurityPrivilege** und **SeAuditPrivilege** Das erste dieser Privilegien ermöglicht es einem Benutzer, das Zugriffsrecht AccessSystemSecurity auf eine Ressource zu erhalten. Dies erlaubt dem Benutzer, die Audit-Konfiguration der Ressource zu ändern. Das Privileg SeAuditPrivilege erlaubt es einem Benutzer, beliebige Objekt-Audit-Meldungen aus einer Benutzeranwendung zu erzeugen. Wir werden Auditing in den Kapiteln 5, 6 und 9 besprechen.
- **SeCreateTokenPrivilege** Dieses Privileg sollte nur einer sehr ausgewählten Gruppe von Benutzern erteilt werden, da es die Möglichkeit bietet, mit dem Systemaufruf NtCreateToken beliebige Token zu erstellen.
- **SeDebugPrivilege** Der Name dieses Privilegs impliziert, dass es zum Debuggen von Prozessen notwendig ist. Das ist jedoch nicht wirklich der Fall, da es möglich ist, einen Prozess ohne dieses Privileg zu debuggen. Das Privileg erlaubt es dem Benutzer, auf beliebige Prozess- und Thread-Objekte zuzugreifen, und zwar unabhängig von deren Sicherheitsdeskriptor.
- **SeTcbPrivilege** Der Name dieses Rechts kommt von Trusted Computing Base (TCB), einem Begriff, der sich auf den privilegierten Kern des Windows-Betriebssystems, einschließlich des Kernels, bezieht. Es ist ein Sammelbegriff für privilegierte Operationen, die nicht durch ein spezifischeres Privileg abgedeckt sind. So können Benutzer beispielsweise die Prüfung zur Erhöhung der Integritätsebene eines Tokens (bis zur Grenze der Systemebene) umgehen,

aber auch einen Fallback-Exception-Handler für einen Prozess festlegen – zwei Operationen, die wenig miteinander zu tun haben.

- **`SeLoadDriverPrivilege`** Wir können einen neuen Kernel-Treiber über den Systemaufruf `NtLoadDriver` laden, obwohl es üblicher ist, den SCM zu verwenden. Dieses Privileg ist erforderlich, um diesen Systemaufruf erfolgreich auszuführen. Beachten Sie, dass Sie mit diesem Privileg keine Kernel-Treiber-Prüfungen, wie beispielsweise die Codesignierung, umgehen können.
- **`SeTakeOwnershipPrivilege`** und **`SeRelabelPrivilege`** Diese Privilegien haben die gleiche unmittelbare Wirkung: Sie ermöglichen es einem Benutzer, `WriteOwner`-Zugriff auf eine Ressource zu erhalten, auch wenn die normale Zugriffssteuerung dies nicht zulassen würde. `SeTakeOwnershipPrivilege` ermöglicht es einem Benutzer, das Eigentum an einer Ressource zu übernehmen, da er dafür `WriteOwner` haben muss. `SeRelabelPrivilege` umgeht die Überprüfung des obligatorischen Labels einer Ressource; normalerweise kann ein Label nur so gesetzt werden, dass es gleich oder niedriger ist als die Integritätsebene des Aufrufers. Das Setzen des obligatorischen Labels erfordert auch den `WriteOwner`-Zugriff auf ein Handle, wie wir in Kapitel 6 sehen werden.

Konkrete Beispiele für die Verwendung dieser Privilegien werden wir in späteren Kapiteln betrachten, wenn wir Sicherheitsdeskriptoren und Zugriffsprüfungen besprechen. Wenden wir uns nun den Möglichkeiten der Zugriffsbeschränkung durch Sandboxing zu.

4.7 Sandbox-Token

In unserer vernetzten Welt müssen wir eine Vielzahl von nicht vertrauenswürdigen Daten verarbeiten. Angreifer könnten die Daten für böswillige Zwecke missbrauchen, beispielsweise um eine Sicherheitslücke in einem Webbrowser oder einem Reader für Dokumente auszunutzen. Um dieser Bedrohung entgegenzuwirken, bietet Windows eine Methode zur Einschränkung der Ressourcen, auf die ein Benutzer zugreifen kann, indem alle seine Prozesse, die nicht vertrauenswürdige Daten verarbeiten, in eine Sandbox gestellt werden. Wenn der Prozess kompromittiert wird, hat der Angreifer nur einen begrenzten Blick auf das System und kann nicht auf die vertraulichen Informationen des Benutzers zugreifen. Windows implementiert Sandboxen durch drei spezielle Token-Typen: Token mit eingeschränktem Zugriff, Token mit eingeschränktem Schreibzugriff und Low-Box-Token.

4.7.1 Eingeschränkte Token

Der *eingeschränkte Token-Typ* ist das älteste Sandbox-Token in Windows. Es wurde als Funktion in Windows 2000 eingeführt, aber erst mit der Einführung des Google-Chrome-Webbrowsers in großem Umfang als Sandbox verwendet. Andere

Browser wie Firefox haben seitdem die Sandbox-Implementierung von Chrome übernommen, ebenso wie Dokumentenleseprogramme wie Adobe Reader.

Sie können ein eingeschränktes Token mit dem Systemaufruf `NtFilterToken` oder der Win32-API `CreateRestrictedToken` erstellen. In beiden Fällen können Sie eine Liste von eingeschränkten SIDs angeben, um die Ressourcen zu begrenzen, auf die das Token zugreifen darf. Die SIDs müssen nicht bereits im Token vorhanden sein. Die restriktivste Sandbox von Chrome legt beispielsweise die NULL-SID (`S-1-0-0`) als einzige eingeschränkte SID fest. Die NULL-SID wird einem Token niemals als normale Gruppe gewährt.

Jede Zugriffsprüfung muss sowohl die normale Liste der Gruppen als auch die Liste der eingeschränkten SIDs zulassen; andernfalls wird dem Benutzer der Zugriff verweigert, was wir in Kapitel 7 noch ausführlich besprechen werden. Der Systemaufruf `NtFilterToken` kann auch normale Gruppen mit dem Attribut-Flag `UseForDenyOnly` und Löschberechtigungen markieren. Wir können die Fähigkeit, ein Token mit eingeschränkten SIDs zu filtern, kombinieren oder sie allein verwenden, um ein Token mit geringeren Rechten zu erstellen, ohne dass ein umfassenderes Sandboxing erforderlich ist.

Es ist einfach, ein eingeschränktes Token zu erstellen, das auf keine Ressourcen zugreifen kann. Eine solche Einschränkung führt zu einer guten Sandbox, macht es aber auch unmöglich, das Token als primäres Token eines Prozesses zu verwenden, da der Prozess nicht gestartet werden kann. Dies schränkt die Effektivität einer Sandbox mit eingeschränktem Token stark ein. Listing 4.18 zeigt, wie man ein eingeschränktes Token erstellt und die Ergebnisse extrahiert.

```
PS> $token = Get-NtToken -Filtered -RestrictedSids RC -SidsToDisable ↵
    WD -Flags DisableMaxPrivileges
PS> Get-NtTokenGroup $token -Attributes UseForDenyOnly
Name   Attributes
----   ----------
Jeder  UseForDenyOnly

PS> Get-NtTokenGroup $token -Restricted
Name                              Attributes
----                              ----------
NT-AUTORITÄT\EINGESCHRÄNKTER[..]  Mandatory, EnabledByDefault, Enabled

PS> Get-NtTokenPrivilege $token
Name                    Luid               Enabled
----                    ----               -------
SeChangeNotifyPrivilege 00000000-00000017  True

PS> $token.Restricted
True
```

Listing 4.18 *Eingeschränktes Token erstellen und Gruppen sowie Berechtigungen anzeigen*

Wir beginnen mit der Erstellung eines eingeschränkten Tokens mit dem Befehl `Get-NtToken`. Wir geben eine eingeschränkte SID, `RC`, an, die einer speziellen SID, `NT-AUTORITÄT\EINGESCHRÄNKTER ZUGRIFF`, entspricht, die üblicherweise für Systemressourcen konfiguriert wird, um Lesezugriff zu ermöglichen. Außerdem geben wir an,

dass wir die Gruppe *Jeder* (`WD` = World) in `UseForDenyOnly` umwandeln wollen. Schließlich geben wir ein Flag an, um die maximale Anzahl von Privilegien zu deaktivieren.

Als Nächstes zeigen wir die Eigenschaften des Tokens an, beginnend mit allen normalen Gruppen, unter Verwendung des Attributs `UseForDenyOnly`. Die Ausgabe zeigt, dass nur bei der Gruppe *Jeder* das Kennzeichen gesetzt ist. Anschließend wird die Liste der eingeschränkten SIDs angezeigt, die die SID `NT-AUTORITÄT\EINGESCHRÄNKTER ZUGRIFF` enthält.

Danach zeigen wir die Privilegien an. Beachten Sie, dass das Privileg `SeChangeNotifyPrivilege` immer noch vorhanden ist, obwohl wir darum gebeten haben, die maximalen Privilegien zu deaktivieren. Dieses Privileg wird nicht gelöscht, da es ohne dieses Privileg sehr schwierig werden kann, auf Ressourcen zuzugreifen. Wenn Sie es wirklich loswerden wollen, können Sie es explizit in `NtFilterToken` angeben oder es löschen, nachdem das Token erstellt wurde.

Schließlich wird die Eigenschaft `Restricted` des Tokens abgefragt, die angibt, ob es sich um ein eingeschränktes Token handelt.

Geschützter Modus im Internet Explorer

Der erste Sandbox-Webbrowser unter Windows war Internet Explorer 7, der in Windows Vista eingeführt wurde. Internet Explorer 7 nutzte die Möglichkeit, die Integritätsstufe des Tokens eines Prozesses herabzusetzen, um die Ressourcen einzuschränken, in die der Browser schreiben konnte. Windows 8 ersetzte schließlich diese einfache Sandbox, den sogenannten geschützten Modus, durch eine neue Art von Token, das LowBox-Token-Objekt, das wir im Abschnitt 4.7.3 besprechen. Das LowBox-Token bietet eine stärkere Isolierung (erweiterter geschützter Modus genannt). Es ist interessant zu wissen, dass Microsoft keine eingeschränkten Token verwendet hat, obwohl diese bereits seit Windows 2000 verfügbar waren.

4.7.2 Schreibgeschützte Token

Ein schreibgeschütztes Token verhindert den Schreibzugriff auf eine Ressource, erlaubt aber den Lese- und Ausführungszugriff. Sie können ein schreibgeschütztes Token erstellen, indem Sie an `NtFilterToken` das Flag `WRITE_RESTRICTED` übergeben.

Dieser Token-Typ wurde in Windows XP SP2 eingeführt, um die Systemdienste abzusichern. Es ist viel einfacher, es als Sandbox zu verwenden als ein eingeschränktes Token, da Sie sich keine Sorgen machen müssen, dass das Token kritische Ressourcen wie DLLs nicht lesen kann. Allerdings wird dadurch eine weniger nützliche Sandbox geschaffen. Wenn Sie beispielsweise die Dateien eines Benutzers lesen können, könnten Sie dessen private Informationen, wie beispielsweise von einem Webbrowser gespeicherte Kennwörter, stehlen, ohne die Sandbox verlassen zu müssen.

Der Vollständigkeit halber wollen wir ein schreibgeschütztes Token erstellen und seine Eigenschaften betrachten (Listing 4.19).