Netz-Weise
Lernen von den Besten.

Erweitertes Windows Auditing einrichten

von Holger Voges

BETA

Version 1.0

Freundallee 13 a
30173 Hannover
www.netz-weise.de

Inhalt

Einführung

Auditing ist die Protokollierung von sicherheitsrelevanten Ereignissen, die auf einem Computer aufgetreten sind. Es hat hauptsächlich einen Zweck – bösartiges Verhalten innerhalb Ihres Netzwerks so schnell wie möglich aufzudecken und Gegenmaßnahmen einzuleiten. Denn so gut Sie Ihr Netzwerk auch schützen – so lange Computersysteme über ein Netzwerk erreichbar sind, ist es faktisch nicht möglich, sie zu 100% abzusichern. Umso wichtiger ist es, ungewöhnliches Verhalten, wie es z.B. durch einen Eindringling oder eine Malware erzeugt wird, frühzeitig zu entdecken und einen Überblick über den entstandenen Schaden zu bekommen sowie die Einfallstore zu schließen.

Windows stellt für das Auditing einen Überwachungsmechanismus bereit, der direkt in die LSA (Local Security Authority) integriert ist, sowie das Windows Sicherheitsprotokoll, in dem Ereignisse dauerhaft gespeichert werden können.

Die Überwachung kann an drei Stellen aktiviert werden – per Kommandozeile mit dem Werkzeug auditpol.exe, per lokaler Sicherheitsrichtlinie und per Gruppenrichtlinie. Seit Windows Vista und Windows Server 2008 haben Sie hierfür sogar zwei verschiedene Überwachungssysteme zur Verfügung – die klassische (alte) Überwachung und die erweiterte Überwachung.

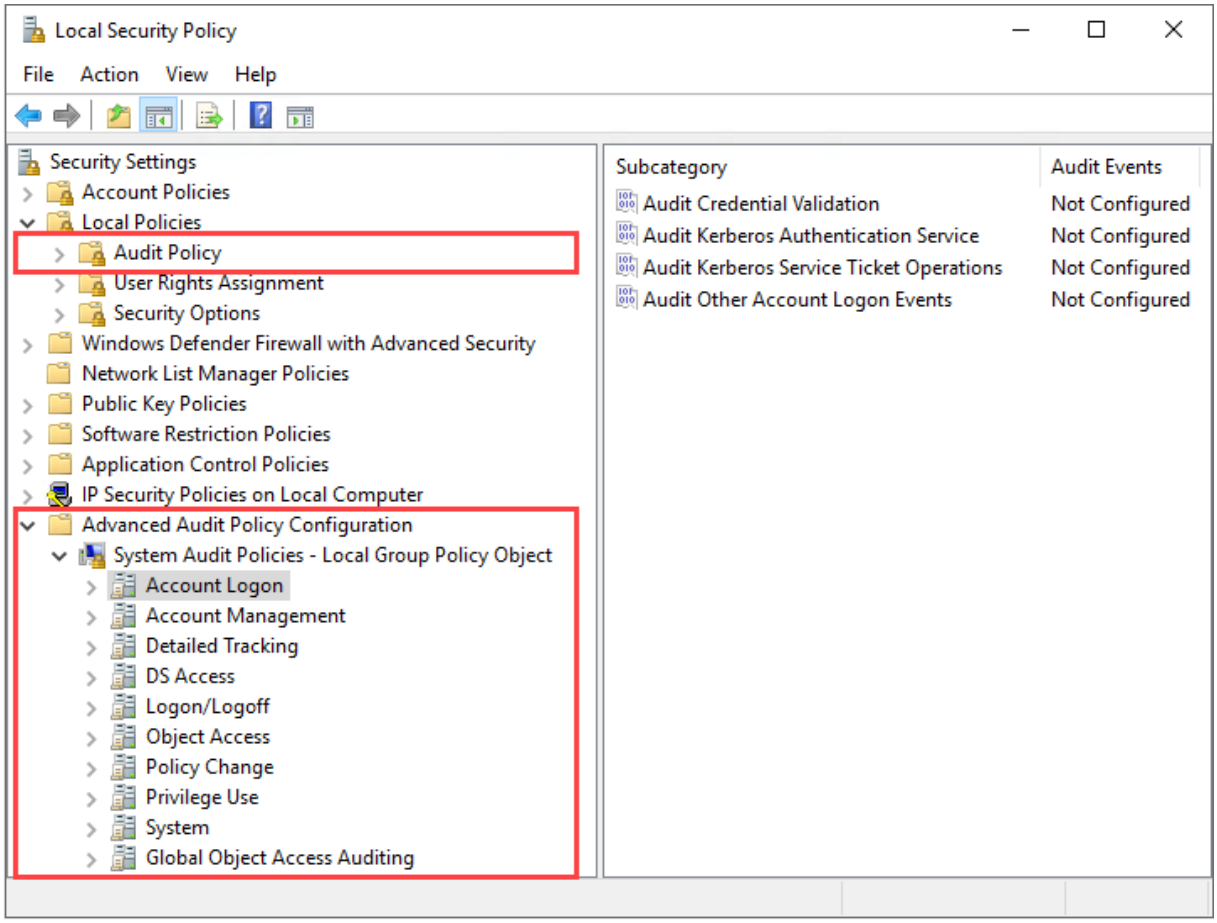


Bild 1 - Die alte und neue Auditing-Konfiguration in den lokalen Sicherheitsrichtlinien

Die erweiterte Überwachung war ursprünglich nicht in den Richtlinien, sondern ausschließlich über die Kommandozeile mit auditpol.exe konfigurierbar. Erst mit Windows 7 und Windows Server 2008 R2 hat Microsoft die erweiterten Überwachungsrichtlinien eingeführt (s. Bild 1 - Die alte und neue Auditing-Konfiguration in den lokalen Sicherheitsrichtlinien).

Der Hauptvorteil der erweiterten Überwachung liegt in der feineren Steuerungsmöglichkeit. Während die alte Überwachung 9 verschiedene Überwachungskategorien kennt (s. Bild 2), kennt die erweiterte Überwachung für jede dieser Kategorien noch einmal Unterkategorien, durch die die Menge der aufgezeichneten Ereignisse eingeschränkt werden kann (s. Bild 1). Für beide Systeme gilt, dass man die Protokollierung erfolgreicher und fehlgeschlagene Aktionen getrennt aktivieren kann.

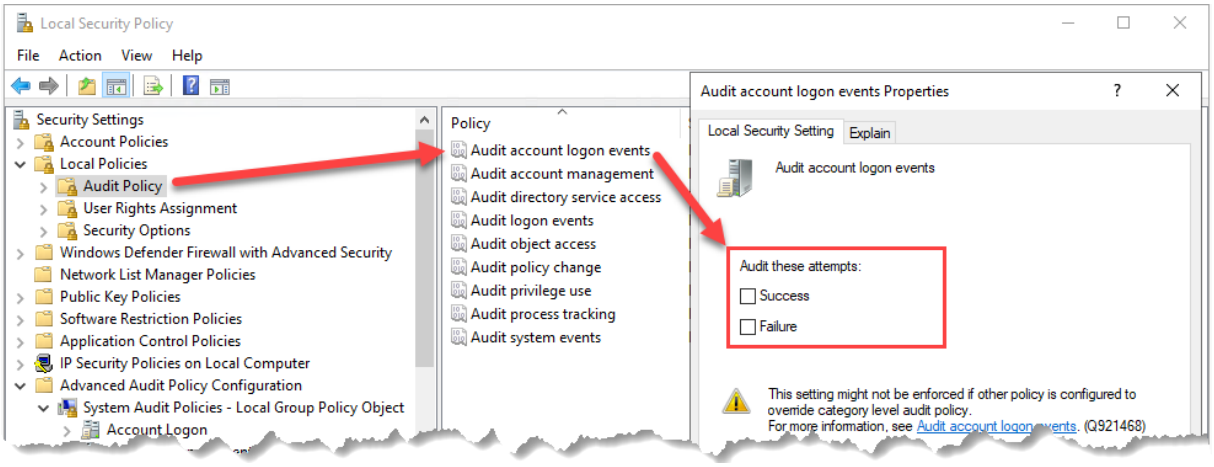


Bild 2 - die alte Überwachung kennt 9 Kategorien

Weniger zur protokollieren erscheint auf den ersten Blick erst einmal nicht unbedingt von Vorteil, denn man erhält letztlich mit der erweiterten Überwachung weniger Informationen. Das große Problem ist aber, die relevanten aus der Unmenge von protokollierten Ereignissen herauszufiltern, da die Überwachung sehr, sehr viele Daten generiert, und diese sich mit der Anzahl der Systeme multiplizieren, die Sie überwachen. Datensparsamkeit ist also auf jeden Fall hilfreich, um den Heuhaufen, in dem sich die Nadel versteckt, möglichst klein zu halten.

Alle überwachten Ereignisse werden von Windows im Sicherheitsprotokoll gespeichert. Es kann über den Eventviewer (Eventvwr.msc) oder mit Hilfe von Powershell angezeigt werden.

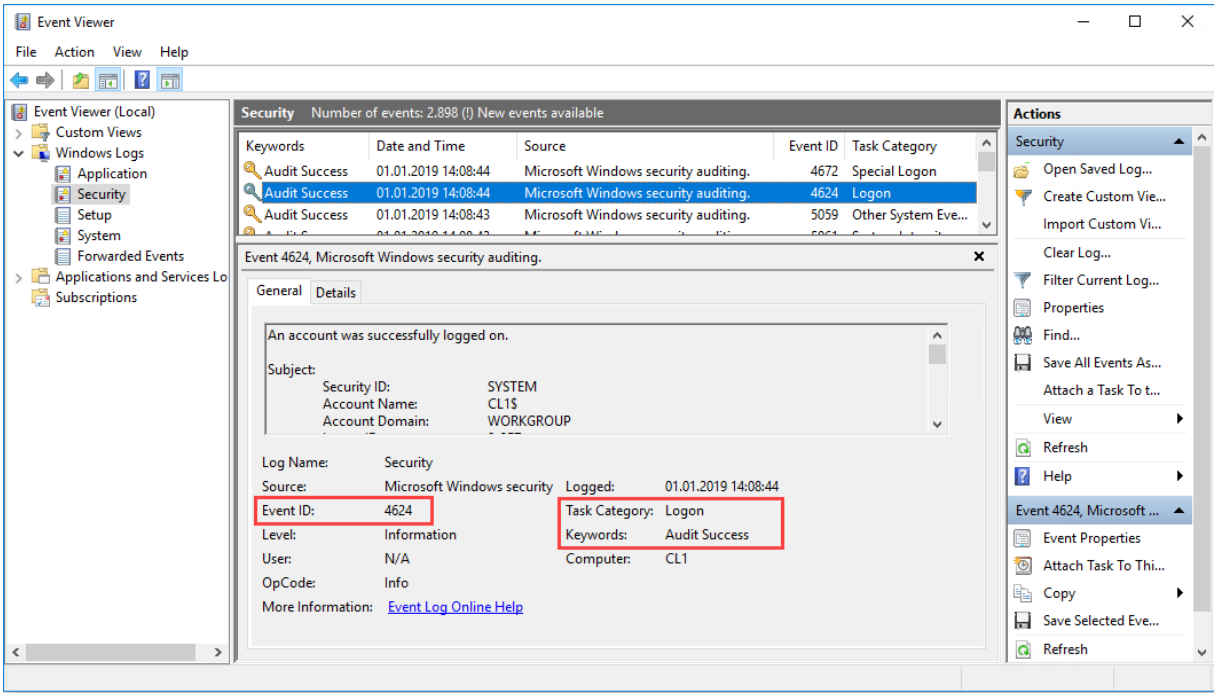


Bild 3 - Eine im Eventlog protokollierte Anmeldung unter Windows 10

Der Eventviewer bietet eine mehrere Möglichkeiten, um Ereignisse zu filtern. Besser ist es aber, die Suche nach auffälligen Ereignissen zu automatisieren. Dies können Sie z.B. mit den Powershell-Cmdlets *Get-Eventlog* und *Get-WinEvent* machen, wobei *Get-Winevent* für komplexe Suchen besser geeignet ist.

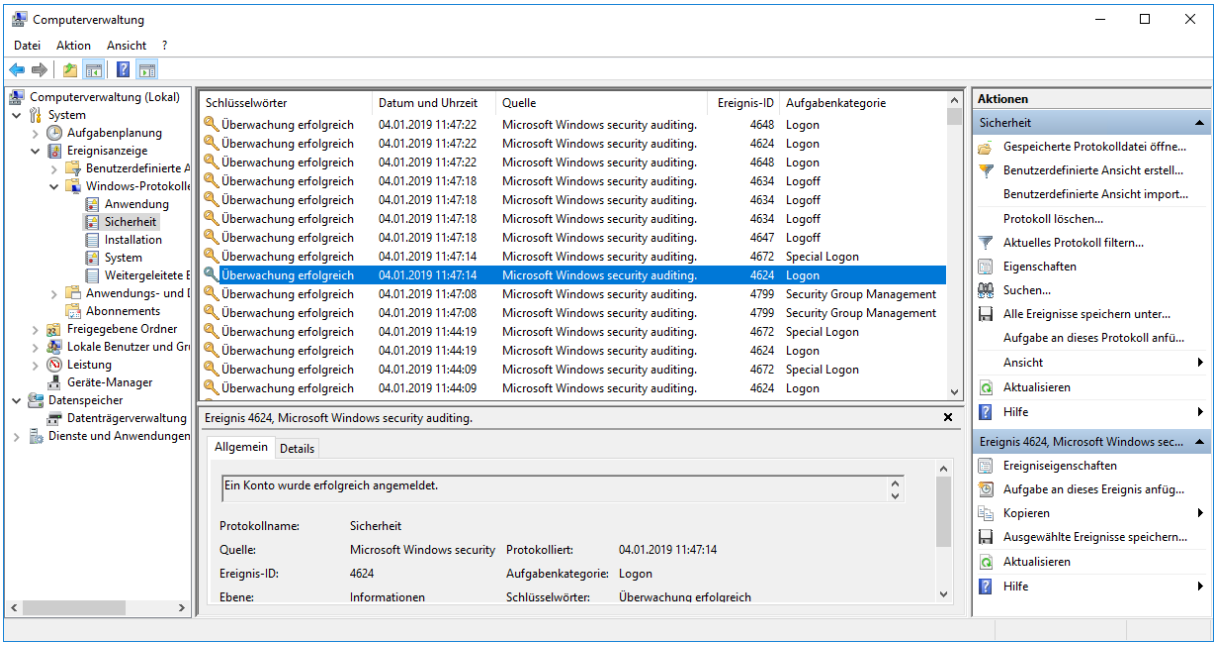


Bild 4 - Das Sicherheitsprotokoll speichert alle Überwachungsereignisse

Zugriff auf das Sicherheitsprotokoll haben standardmäßig nur Administratoren. Hat ein Angreifer sich aber erst einmal administrative Berechtigungen auf einem System verschafft, kann er seine Spuren sehr einfach verwischen, indem er das lokale Sicherheitsprotokoll löscht. Um das zu verhindern, sollten Sie einen Mechanismus implementieren, der alle Ereignisse zentral sammelt. Dadurch wird es auch bedeutend einfacher, die Protokolle zu analysieren. Windows stellt Ihnen hierfür die Ereignisweiterleitung (Event Log Fowarding) zur Verfügung. Alternativ können Sie aber auch auf Fremdherstellertools zurückgreifen, die mit Hilfe von Clients neben dem Ereignisprotokoll auch andere Logs zentral speichern können.

Die große Kunst der Überwachung liegt darin, die gesammelten Daten regelmäßig auszuwerten und auffälliges Verhalten frühzeitig zu erkennen. Das können Sie mit Hilfe von Skripten selbst machen, oder Sie können sich auf Fremdherstellertools verlassen, die Ihnen helfen, verdächtige Ereignisse ausfindig zu machen und automatische Warnungen zu generieren. Man spricht in diesem Zusammenhang von SIEM-Systemen – Security Information and Event Management.

In dieser Dokumentation werde ich zuerst darauf eingehen, wie das Auditing konfiguriert wird. Anschließend wird beschrieben, wie Sie Ereignisse aus dem Eventlog filtern können. Im darauffolgenden Abschnitt wird beschrieben, wie Sie das Eventlog-Forwarding einrichten, um dann einen kurzen Überblick über die wichtigsten Ereignisse zu liefern.

Das erweitert Auditing einrichten

Das erweiterte Auditing hat Microsoft mit Vista eingeführt. Es löst das alte Auditing-System ab und ist parallel dazu implementiert – Sie können also nach wie vor entweder das alte oder das neue Auditing verwenden. Sobald das neue Auditing aktiviert ist, werden alle Einstellungen des alten Auditing ignoriert. Das macht Sinn, denn eigentlich ist das alte Auditing nur noch aus Kompatibilitätsgründen in Windows enthalten.

Das erweiterte Auditing unterscheidet sich vom alten Auditing in erster Linie dadurch, dass deutlich feiner gesteuert werden kann, welche Ereignisse protokolliert werden sollen. Dafür stehen 9 Überwachungs-Kategorien zur Verfügung, die sich in insgesamt 59 Unterkategorien aufteilen. Jede einzelne Unterkategorie kann einzeln gesteuert werden. Im alten Auditing-System konnten nur die Hauptkategorien aktiviert oder deaktiviert werden. Durch die granulare Steuerung ist es deutlich besser möglich, die unwichtige Ereignisse erst gar nicht zu protokollieren und die generierte Datenmenge überschaubar zu halten.

In Vista ist das erweiterte Auditing nur lokal über das Kommandozeilen-Programm auditpol.exe steuerbar, was letztlich gegenüber dem alten Auditing ein Rückschritt war. Erst seit Windows 7 und Windows Server 2008 R2 kann man es wieder über Gruppenrichtlinien steuern. Allerdings ist diese Funktion nur "angebaut" worden. Eine wirklich detaillierte Steuerung und Auswertung, welche Ereignisse überwacht werden, können Sie nur über die Kommandozeile durchführen. Daher ist es wichtig, dass Sie sich mit der Funktionsweise von auditpol.exe vertraut machen. Im Folgenden werde ich zuerst zeigen, wie Sie das Auditing mit Richtlinien konfigurieren können, um dann die auf die Konfiguration mit Auditpol einzugehen.

Überwachungsrichtlinien

Welche Ereignisse überwacht werden sollen, können Sie über die lokale Sicherheitsrichtlinie oder über Gruppenrichtlinien konfigurieren. Da die Einstellungen identisch sind, gehe ich hier primär auf die Konfiguration per Gruppenrichtlinien ein – sämtliche Einstellungen sind auf die lokalen Sicherheitsrichtlinien übertragbar. Die Verwaltungskonsole für lokale Sicherheitsrichtlinien können Sie über *Secpol.msc* aus der Kommandozeile heraus starten, oder indem Sie "lokale Sicherheitsrichtlinie" im Windows Startmenü eingeben (s. Bild 1).

Um die Sicherheitsrichtlinien per Gruppenrichtlinie zu konfigurieren, legen Sie ein neues Gruppenrichtlinienobjekt in der Gruppenrichtlinienverwaltungskonsole an. Anschließend öffnen Sie in der Computerkonfiguration den Knoten *Richtlinien > Windows Einstellungen > Sicherheitseinstellungen > Erweiterte Überwachungsrichtlinienkonfiguration* und wählen *Überwachungsrichtlinien* aus. Unterhalb von Überwachungsrichtlinien werden Ihnen die 9 Überwachungskategorien sowie die *Globale Objektzugriffsüberwachung* angezeigt.

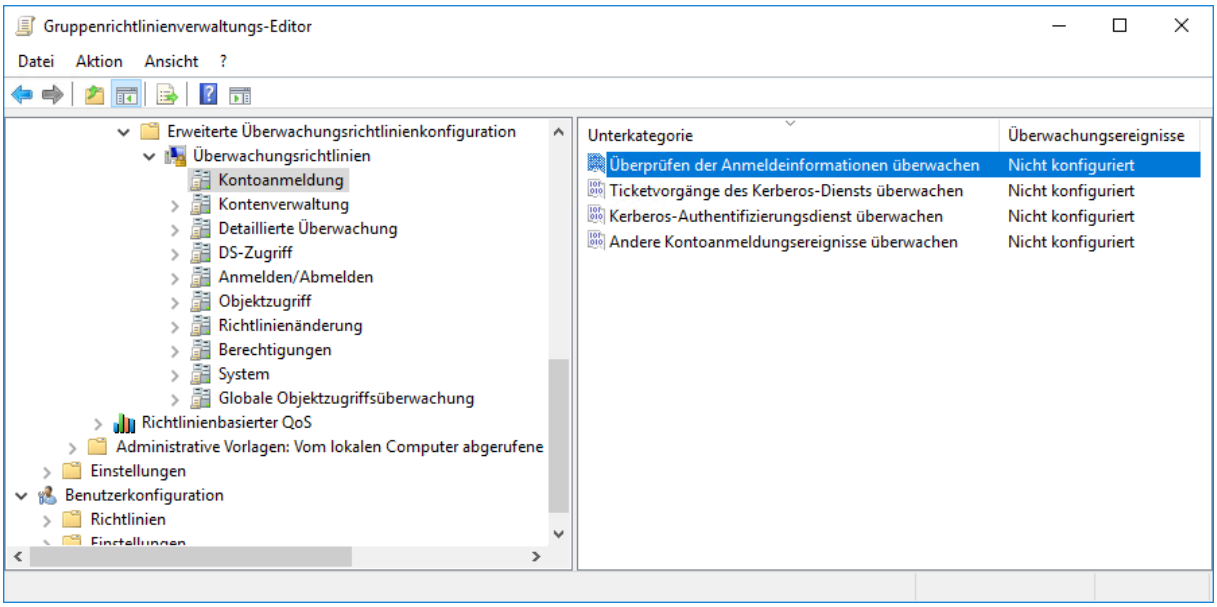


Bild 5 - die erweiterte Überwachungsrichtlinienkonfiguration in der GPMC

Wenn Sie eine Kategorie auswählen, werden im rechten Fenster die Unterkategorien angezeigt. Mit einem Doppelklick können Sie die Einstellungen öffnen.

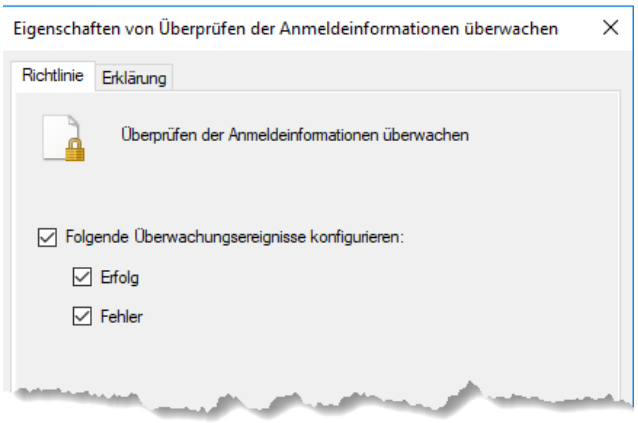


Bild 6 - legen Sie fest, ob erfolgreiche oder fehlgeschlagene Ereignisse protokolliert werden sollen

Sie können beim Aktivieren festlegen, ob Sie nur erfolgreiche, nur fehlgeschlagene oder beide Typen von Ereignissen überwachen wollen. Welche Ereignisse wirklich interessant sind, ist von der Überwachungskategorie abhängig.

Standardmäßig ist in keiner Gruppenrichtlinie die Überwachung aktiviert. Trotzdem werden Sie auf jedem Computer eine ganze Reihe von Überwachungsereignissen vorfinden, wenn Sie das Sicherheitsprotokoll öffnen (s. Bild 4). Diese Logs stammen von Standardüberwachungen, die über die lokalen Überwachungseinstellungen aktiviert sind. Die lokalen Überwachungseinstellungen können Sie über Gruppenrichtlinien nicht deaktivieren, sondern nur ergänzen.

Die Überwachungseinstellungen teilen sich in 9 Kategorien auf:

Kategoriename	Funktion
Kontoanmeldung (Account Logon)	Kontoanmeldungen protokollieren die Authentifizierung eines Kontos gegen die Benutzerdatenbank. Auf einem Domänencontroller aktiviert, werden die Authentifizierungen von Domänenkonten am DC protokolliert. Auf allen anderen Computern aktivieren Sie die

	Protokollierung der Authentifizierung mit lokalen Konten am Security Accounts Manager (SAM).
Anmelden/Abmelden (Logon/Logoff)	Im Gegensatz zur Kontoanmeldung, die die Authentifizierung (quasi die Kennwortprüfung) protokolliert, wird mit dieser Kategorie der An- und Abmeldevorgang überwacht. Der Anmeldevorgang an sich findet statt, nachdem der Benutzer sich authentifiziert hat und wird immer auf dem System gespeichert, auf das zugegriffen werden soll.
Kontenverwaltung (Account Management)	Mit Hilfe der Kontenverwaltung können Sie die das Anlegen, Löschen und Bearbeiten von Benutzer, Computer und Gruppen überwachen. Wieder gilt: Auf einem DC werden die Zugriffe auf das AD überwacht, auf allen anderen Systemen der Zugriff auf die lokale Benutzerdatenbank.
DS-Zugriff (DS Access)	Mit Hilfe des Verzeichnisdienstzugriffs werden Lesezugriffe auf (Verzeichnisdienstzugriff) und Änderungen am (Verzeichnisdienständerungen) Active Directory gespeichert. Diese Überwachungseinstellungen wirken nur auf Domänencontrollern.
Richtlinienänderung (Policy Change)	Richtlinienänderungen bezieht sich auf die lokalen Sicherheitsrichtlinien, die Sie mit SecPol.msc bearbeiten können, und auf die Überwachungsrichtlinien selbst.
Detaillierte Überwachung (Detailed Tracking)	Mit Hilfe der detaillierten Überwachung können Sie die Ausführung von Programmen protokollieren. Die Log-Einträge enthalten einen Verweis auf die Sitzung (Anmelden/Abmelden-Kategorie), in der das Programm gestartet wurde.
Objektzugriff (Object Access)	Die Kategorie Objektzugriffe fasst den Zugriff auf alles im System zusammen, was über ACLs (Access Control Lists = Berechtigungslisten) verfügt. Sie müssen den Objektzugriff aktivieren, um Änderungen am Dateisystem, der Systemregistrierung, Druckern oder Zertifikatsservern zu protokollieren. Eine Aktivierung des Objektzugriffs aktiviert in diesem Falle die Protokollierung nicht sofort, sondern auf den zu Objekten muss die Überwachung erst noch explizit aktiviert werden.
Berechtigungen (Privilege Use)	Berechtigungen protokolliert das Anwenden von Systemrechten (Privileges) wie das Sichern von Dateien oder das Übernehmen des Besitzes an Daten. Benutzerrechte können in der lokalen Sicherheitsrichtlinie unter *Zuweisen von Benutzerrechen* eingesehen und geändert werden.
System (System)	System fasst alle sicherheitsrelevanten Ereignisse zusammen, die sich nicht in die obigen Kategorien einordnen lassen, wie den Systemstart, den IPSec-Dienst oder Ereignisse der Windows-Firewall.

Tabelle 1 - Auditing-Kategorien

Neben den Kategorien finden Sie in den Richtlinien-Einstellungen noch einen weiteren Eintrag, Globale Objektzugriffsüberwachung. Mit seiner Hilfe können Sie Objektzugriffe über Richtlinien aktivieren.

Wenn Sie Zugriffe auf Dateien oder die Systemregistrierung überwachen wollen, müssen Sie zusätzlich zur Aktivierung des Objektzugriffs die Objekte angeben, die überwacht werden sollen. Das geschieht über SACLs (System Access Control Lists). Die SACL finden Sie, wenn Sie die Eigenschaften eines über den Objektzugriff überwachten Objekts und auf der Registerkarte Sicherheit die erweiterten Einstellungen öffnen. Im Bild 6 sehen die Eigenschaften eines Ordners mit dem Namen *Top Secret*.

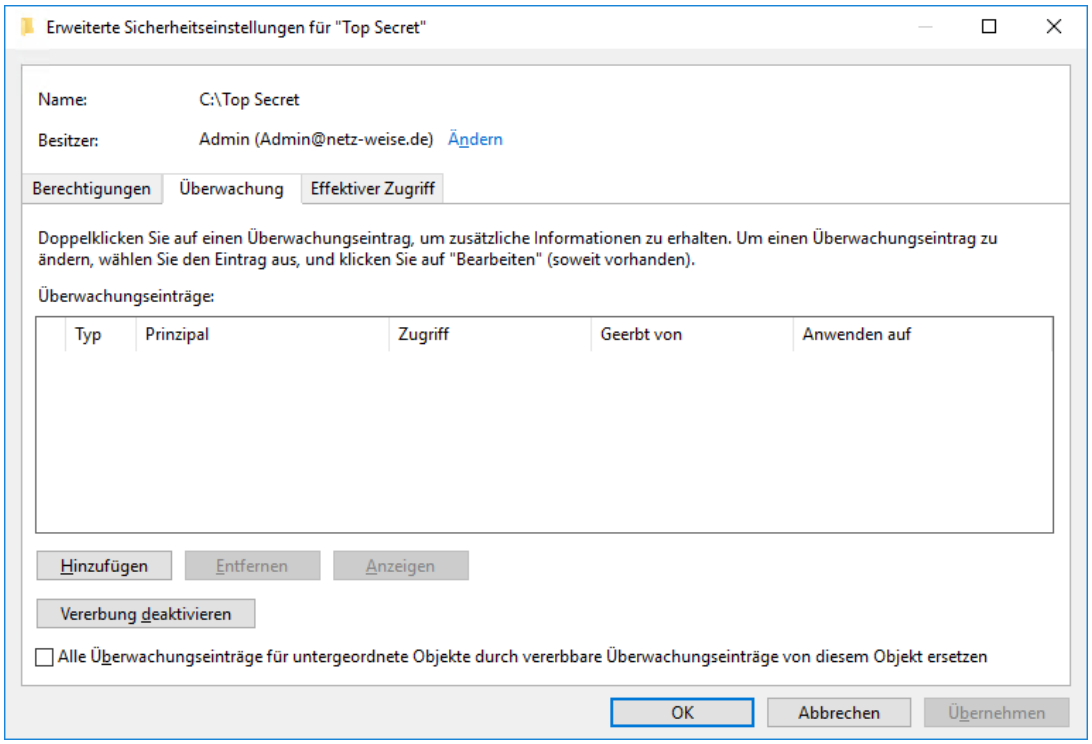


Bild 7 - Objekte werden erst überwacht, nachdem man die SACL konfiguriert hat

SACLs werden ähnlich gepflegt wie Berechtigungslisten[1]. Um eine Überwachungskonfiguration hinzuzufügen, geben Sie an, für welche Benutzer oder Gruppen eine Protokollierung erfolgen soll, und bei welchen Zugriffen.

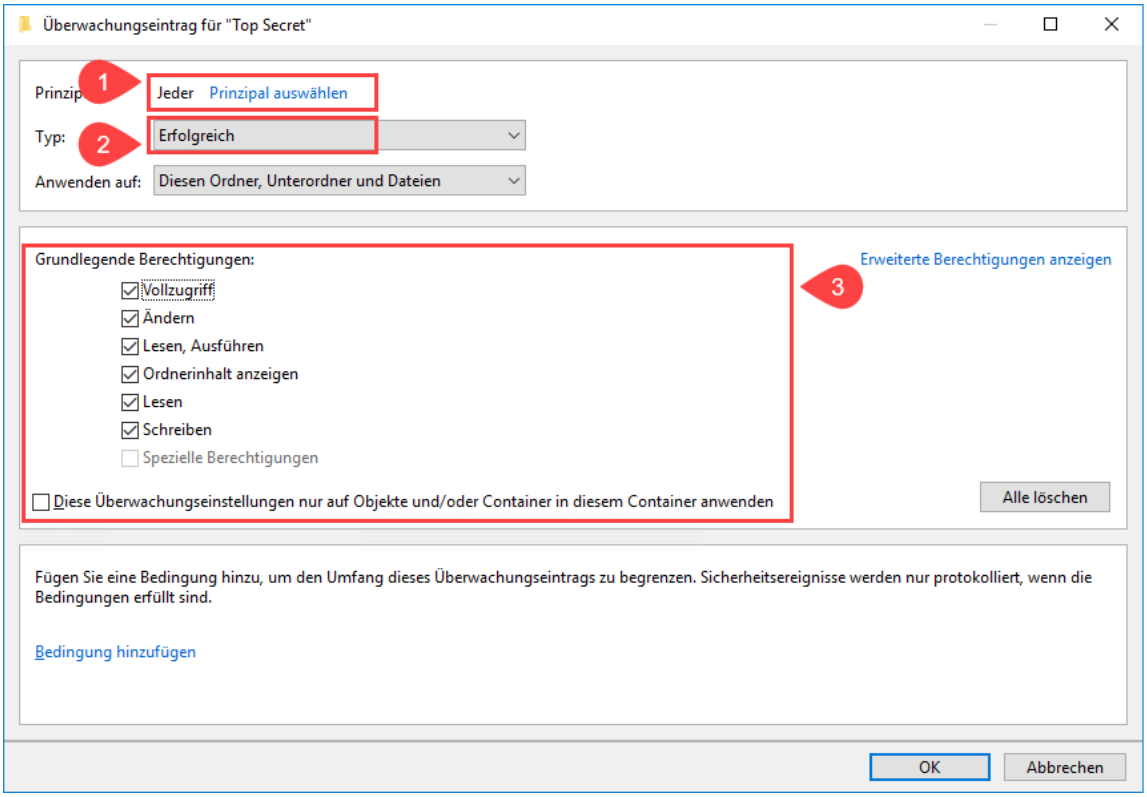


Bild 8 - Die Objektüberwachung wird erst mit einem Überwachungseinstrag aktiv

[1] Berechtigungslisten werden als DACLs oder Discretionary Access Control Lists bezeichnet.

Wählen Sie dazu *Hinzufügen* aus, geben Sie unter (1) das Konto oder die Gruppe (den "Prinzipal") an, ob Erfolgreiche oder Fehlgeschlagene Zugriffe überwacht werden sollen (2) und bei welchen Zugriffen die Protokollierung erfolgen soll (3). Anschließend übernehmen Sie die Einstellungen mit OK.

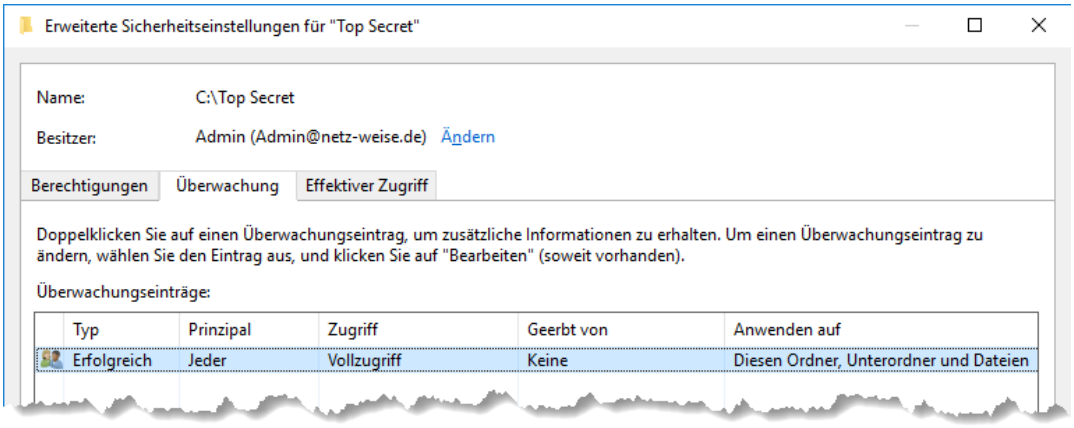


Bild 9 - Die SACL überwacht nun alle erfolgreichen Zugriff auf die en Ordner Top Secret

Sie müssen für jeden Benutzer und jede Gruppe, die Sie überwachen wollen, einen Überwachungseintrag hinzufügen. Je mehr Zugriffe Sie protokollieren, desto schwieriger ist das Auswerten der Daten. Daher sollten Sie versuchen, so selektiv wie möglich zu überwachen. Das gilt speziell für Datei- und Registrierungszugriffe, von denen allein das Betriebssystem tausende pro Sekund durchführt. Sehr schön kann man das mit dem Process-Monitor aus den Sysinternals-Tools sehen. Sie können ihn unter https://docs.microsoft.com/en-us/sysinternals/downloads/procmon oder kurz https://bit.ly/2uA4B4h herunterladen.

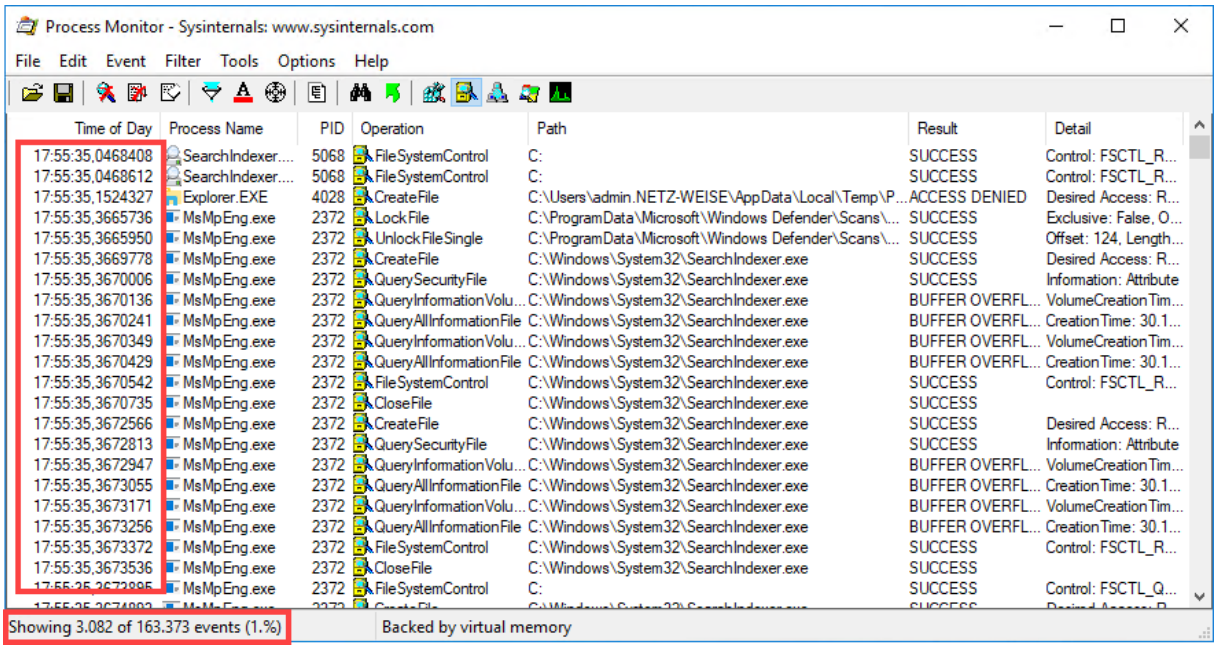


Bild 10 - Alle Dateizugriff auf einem untätigen Windows 10 in Procmon

Die Objektüberwachung auf vielen Systemen zu aktivieren ist sehr aufwändig. Deswegen bringt die erweiterte Überwachungsrichtlinienkonfiguration die *Globale Objektzugriffsüberwachung* mit. Hier können Sie die Zugriffsprotokollierung der Systemregistrierung und des Dateisystems direkt über eine Gruppenrichtlinie aktivieren. Global heißt die Einstellung, weil Sie hier keine Ordner oder Registrierungsschlüssel angegeben können, sondern nur die zu protokollierenden Zugriffe.

Öffnen Sie zum Einrichten den Eintrag *Globale Objektzugriffsüberwachung* in der erweiterten Überwachungsrichtlinienkonfiguration und wählen Sie die Ressource aus, für die die Objektüberwachung eingeschaltet werden soll. Aktivieren Sie den Haken *Richtlinieneinstellungen definieren* und bearbeiten Sie anschließend die SACL-Einträge über *Konfigurieren*.

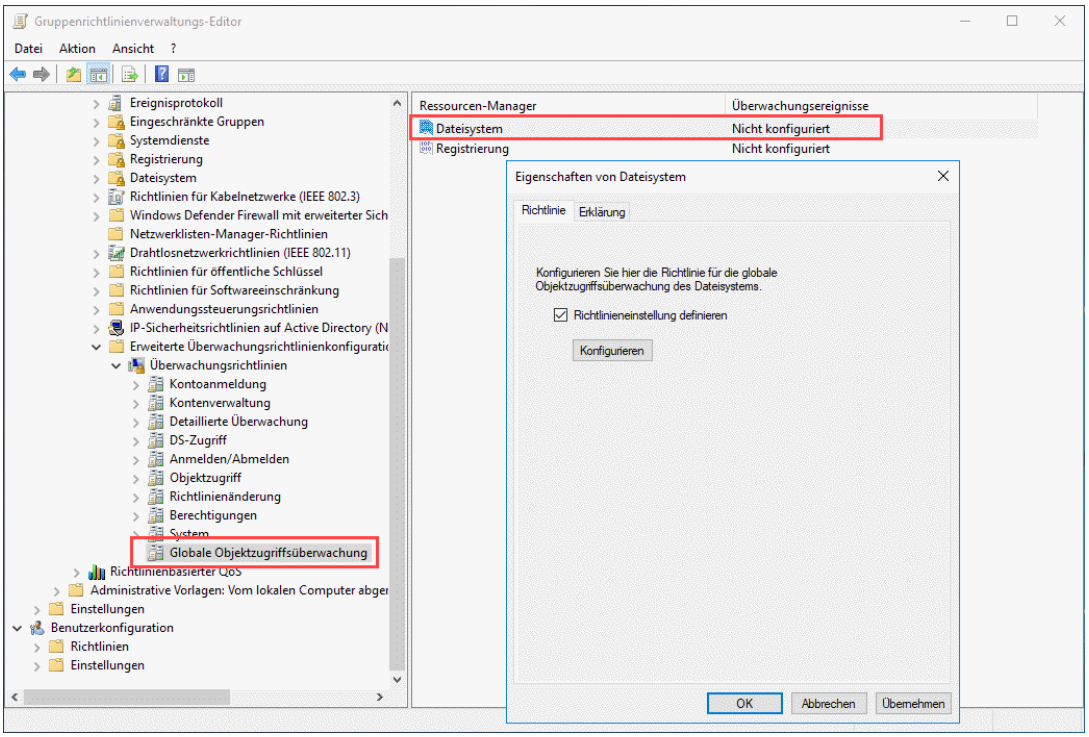


Bild 11 - Wählen Sie die Ressource aus, für die die Objektüberwachung aktiviert werden soll

Im nächsten Schritt müssen Sie die SACLs hinzufügen. Wählen Sie in den erweiterten Sicherheitseinstellungen *Hinzufügen* und tragen Sie ein, welche Prinzipale (Konten und Gruppen) überwacht, und welche Zugriffe protokolliert werden sollen. Die Konfiguration ist identisch mit dem Hinzufügen von SACLs am Objekt.

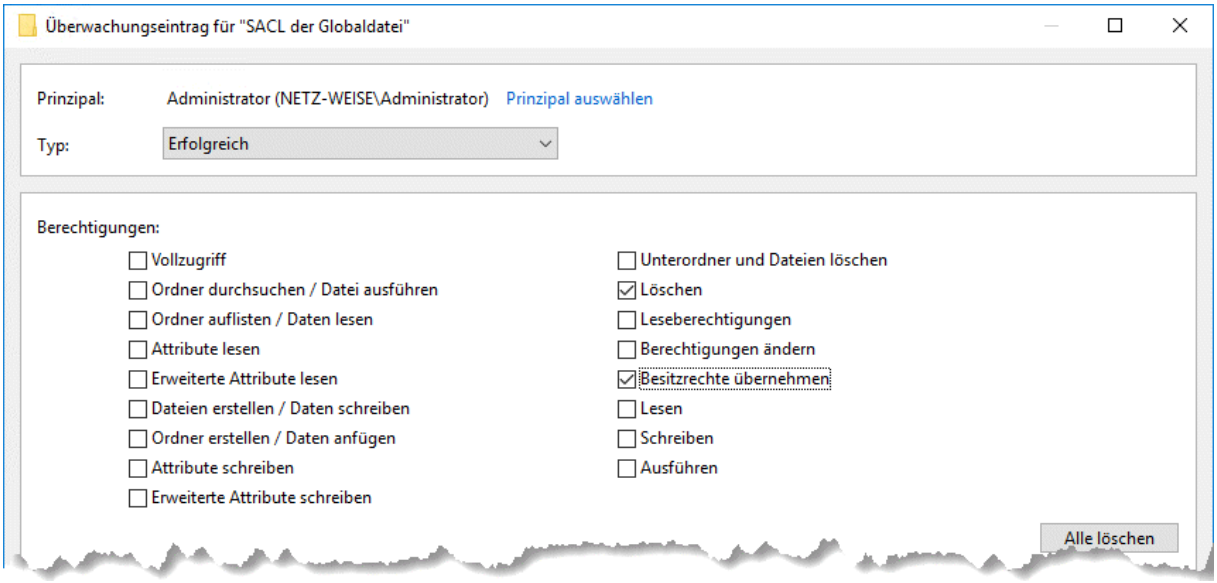


Bild 12 - Es soll protokolliert werden, wenn ein Administrator etwas löscht oder den Besitz an Daten übernimmt.

Nachdem Sie alle Prinzipale zur SACL hinzugefügt haben, übernehmen Sie die Einstellungen.

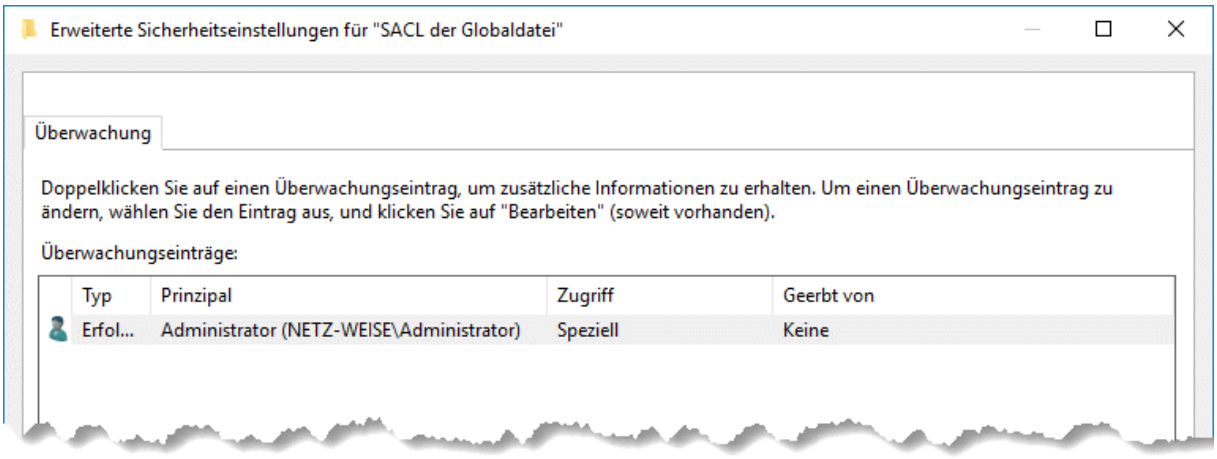


Bild 13 - Die Administratoren sind jetzt der SACL hinzugefügt.

Die globale Objektüberwachung ist sehr hilfreich, wenn es darum geht, eine allgemeine Überwachung des Dateisystems oder der Systemregistrierung zu aktivieren. Seien Sie sich aber bewusst, dass Sie die Überwachung immer für alle Objekte aktivieren – Sie können keine Ordner oder einzelne Systemregistrierungsschlüssel angeben.

Auditpol

Auditpol.exe steht Ihnen auf jedem Windows-System zur Verfügung und erlaubt Ihnen das Auslesen und Konfigurieren der lokalen Überwachungseinstellungen. Wenn Sie *auditpol.exe* ohne Parameter aufrufen, wird Ihnen eine Liste der Optionen angezeigt.

```
PS > auditpol
Syntax: Befehl AuditPol [<Unterbefehl><Optionen>]

Befehle (nur ein Befehl pro Ausführung zulässig)
  /?              Hilfe (kontextabhängig)
  /get            Zeigt die aktuelle Überwachungsrichtlinie an.
  /set            Legt die Überwachungsrichtlinie fest.
  /list           Zeigt die auswählbaren Elemente der Richtlinie an.
  /backup         Speichert Überwachungsrichtlinien in einer Datei.
  /restore        Stellt eine Überwachungsrichtlinie aus einer Datei wieder her.
  /clear          Löscht die Überwachungsrichtlinie.
  /remove         Entfernt die Einzelbenutzer-Überwachungsrichtlinie für
                  einen angegebenen Benutzer.
  /resourceSACL   Konfiguriert SACLs für globale Ressourcen
```

Um die aktuellen Einstellungen zu sichern, erstellen Sie zuerst eine Sicherung. Verwenden Sie hierzu den Parameter */backup* zusammen mit dem Parameter */File*.

```
PS > auditpol /backup /file:C:\temp\DefaultAuditSettings.csv
The command was successfully executed.
```

Auditpol exportiert die aktuelle Konfiguration in Form einer csv-Datei. Sie können die Konfiguration nun jederzeit mit dem Parameter */restore* wiederherstellen.

```
PS > auditpol /restore /file: C:\temp\DefaultAuditSettings.csv
The command was successfully executed.
```

Verwenden Sie den Parameter */list*, um sich alle verfügbaren Kategorien und Unterkategorien anzeigen zu lassen. Mit den Parametern /Category und /Subcategory legen Sie fest, ob Sie nur die Hauptkategorien oder auch die untergeordneten Kategorien angezeigt bekommen:

```
PS > auditpol /list /Category
```

```
Kategorie/Unterkategorie
Account Logon
Account Management
Detailed Tracking
DS Access
Logon/Logoff
Object Access
Policy Change
Privilege Use

PS > auditpol /list /Subcategory:*
Kategorie/Unterkategorie
System
  Security State Change
  Security System Extension
  System Integrity
  IPsec Driver
  Other System Events
Logon/Logoff
  Logon
  Logoff
  Account Lockout
[…]
```

Die Ausgabe kann auch lokalisiert sein. Wenn Sie mit verschiedensprachigen Systemen arbeiten, können Sie auditpol.exe mit dem Parameter */v* dazu überreden, Ihnen auch die sprachunabhängige GUIDs (eindeutige ID) auszugeben:

```
PS > auditpol /list /category /v

Kategorie/Unterkategorie                GUID
An-/Abmeldung                           {69979849-797A-11D9-BED3-505054503030}
Berechtigungen                          {6997984B-797A-11D9-BED3-505054503030}
Detaillierte Nachverfolgung             {6997984C-797A-11D9-BED3-505054503030}
DS-Zugriff                              {6997984F-797A-11D9-BED3-505054503030}
Kontenverwaltung                        {6997984E-797A-11D9-BED3-505054503030}
Kontoanmeldung                          {69979850-797A-11D9-BED3-505054503030}
Objektzugriff                           {6997984A-797A-11D9-BED3-505054503030}
Richtlinienänderung                     {6997984D-797A-11D9-BED3-505054503030}
System                                  {69979848-797A-11D9-BED3-505054503030}
```

Wenn Sie die Ausgabe mit Powershell weiterverarbeiten oder ein Excel-lesbares Format umwandeln wollen, hilft Ihnen der Parameter */r* weiter, der die Ausgabe im CSV-Format erzwingt. In Powershell können Sie mit Hilfe des Cmdlets *ConvertFrom-Csv* alle Daten in Objekte umwandeln.

```
PS > auditpol /list /category /v /r | ConvertFrom-Csv

Kategorie/Unterkategorie GUID
------------------------ ----
Account Logon            {69979850-797A-11D9-BED3-505054503030}
Account Management       {6997984E-797A-11D9-BED3-505054503030}
Detailed Tracking        {6997984C-797A-11D9-BED3-505054503030}
DS Access                {6997984F-797A-11D9-BED3-505054503030}
Logon/Logoff             {69979849-797A-11D9-BED3-505054503030}
Object Access            {6997984A-797A-11D9-BED3-505054503030}
Policy Change            {6997984D-797A-11D9-BED3-505054503030}
Privilege Use            {6997984B-797A-11D9-BED3-505054503030}
System                   {69979848-797A-11D9-BED3-505054503030}
```

Um die effektive Überwachungseinstellung auszulesen, verwenden Sie den Parameter */get*. Sie können wieder mit den gleichen Filtern arbeiten wie mit List, also */Category* und */Subcategory*.

```
PS > auditpol /get /category:*
System audit policy
Category/Subcategory                    Setting
System
  Security System Extension             No Auditing
```

```
  System Integrity                        Success and Failure
  IPsec Driver                            No Auditing
  Other System Events                     Success and Failure
  Security State Change                   Success
Logon/Logoff
  Logon                                   Success
  Logoff                                  Success
  Account Lockout                         Success
  IPsec Main Mode                         No Auditing
  IPsec Quick Mode                        No Auditing
  IPsec Extended Mode                     No Auditing
  Special Logon                           Success
  Other Logon/Logoff Events               No Auditing
[…]
```

Die Ausgabe wurde auf einem frisch installierten Windows 10 1803 ausgeführt. Wie sie sehen können, werden bereits standardmäßig eine ganze Reihe von Ereignissen protokolliert. Diese Einstellungen sind direkt im Auditing konfiguriert. Der Beispielclient, auf dem die Konfiguration ausgelesen wurde, ist nicht Mitglied einer Domäne, hat also keine Auditing-Konfiguration aus Gruppenrichtlinien enthalten. Tatsächlich gibt es auch keine Einstellungen in den lokalen Sicherheitsrichtlinien.

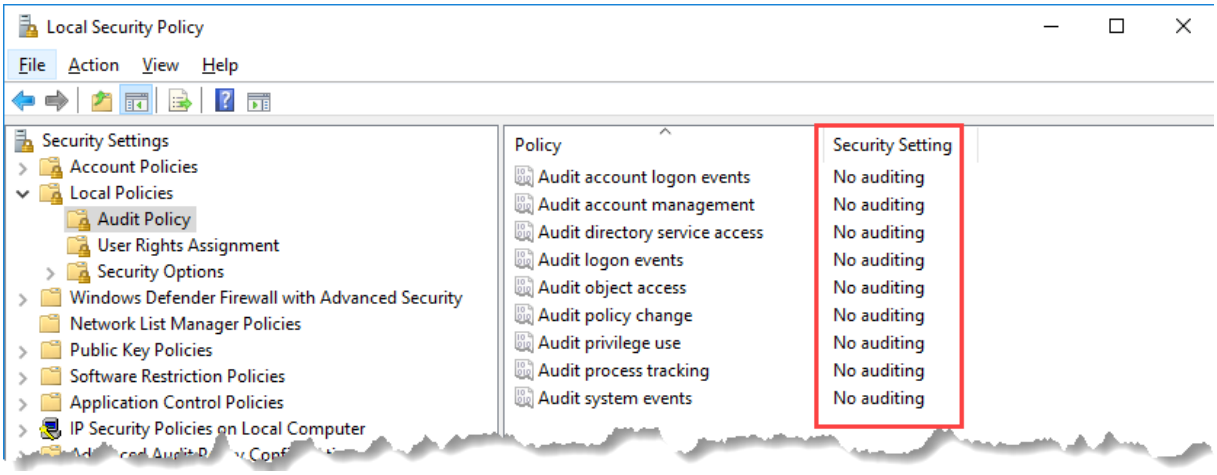


Bild 14 - die alte Überwachungsrichtlinie zeigt keine Überwachungseinstellungen an…

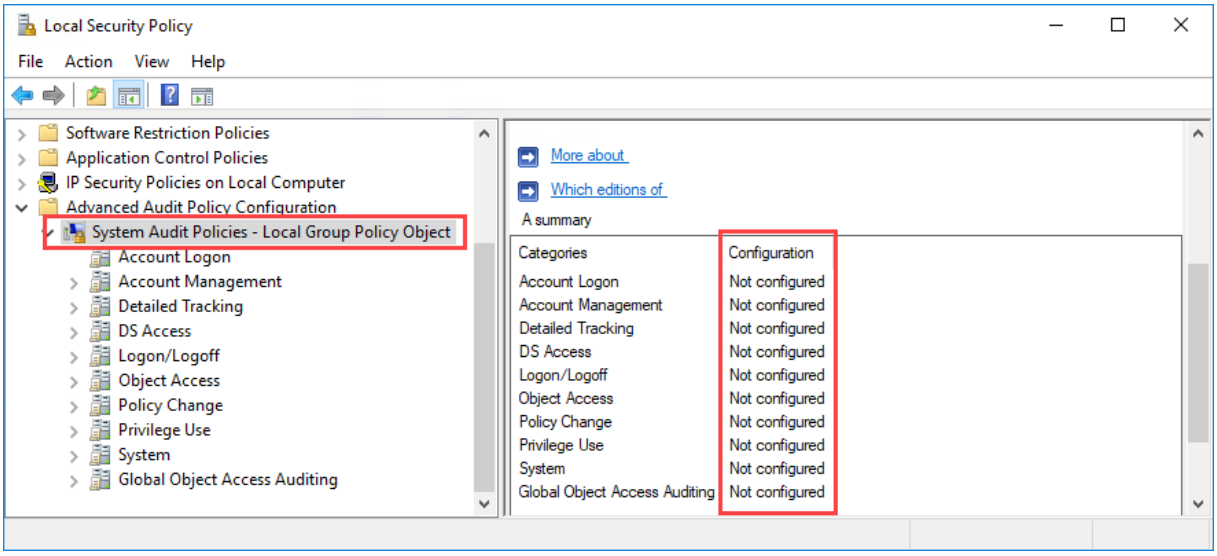


Bild 15 - und die neue auch nicht

Wenn Sie die **effektiven Überwachungseinstellungen** eines Clients anzeigen lassen wollen, müssen Sie immer *auditpol /get* verwenden, da die effektiven Überwachungsrichtlinien kumulativ sind. Sie setzen sich aus den Einstellungen der Gruppenrichtlinien, den Einstellungen der lokalen Richtlinien und den lokalen Überwachungseinstellungen zusammen. Mit *gpresult.exe* werden nur die Einstellungen der Gruppenrichtlinien angezeigt.

Um einzelne Unterkategorien zu überprüfen, können Sie den Parameter */subcategory* verwenden un die Unterkategorien kommasepariert angeben:

```
PS > auditpol /get /subcategory:"Security System Extension","Logon","Logoff"
System audit policy
Category/Subcategory                      Setting
System
  Security System Extension               No Auditing
Logon/Logoff
  Logoff                                  Success
  Logon                                   Success
```

Mit dem Parameter */Set* wird die Überwachung für ganze Kategorien oder Unterkategorien konfiguriert. Mit */success* und */failure* geben Sie an, ob Sie erfolgreiche oder nicht erfolgreiche Aktionen überwachen wollen. Um sowohl im Erfolgs- wie auch im Misserfolgsfall einen Überwachungseintrag zu generieren, geben Sie beide Parameter an.

Im folgenden Beispiel wird zuerst die Überwachung von fehlgeschlagenen Ereignissen der gesamten Kategorie "Logon/Logoff" aktiviert, danach die Überwachung der Unterkategorie "File Share" für beide Ereignistypen.

```
PS > auditpol /set /category:Logon/Logoff /failure:Enable
The command was successfully executed.

PS > auditpol /set /SubCategory:"File Share" /failure:Enable /success:Enable
The command was successfully executed.

PS > auditpol /get /Category:Logon/Logoff /SubCategory:"File Share"
System audit policy
Category/Subcategory                      Setting
Logon/Logoff
  Logon                                   Success and Failure
  Logoff                                  Success and Failure
  Account Lockout                         Success and Failure
  IPsec Main Mode                         Failure
  IPsec Quick Mode                        Failure
  IPsec Extended Mode                     Failure
  Special Logon                           Success and Failure
  Other Logon/Logoff Events               Failure
  Network Policy Server                   Success and Failure
  User / Device Claims                    Failure
  Group Membership                        Failure
Object Access
  File Share                              Success and Failure
```

Die */set*-Option kennt noch drei erweiterte Funktionen, die Sie so nur über auditpol setzen können. Zum einen können Sie Überwachungen auf einem System auch Anwenderspezifisch aktivieren. Verwenden Sie hierfür den Parameter */user* und geben Sie den Benutzer an, für den eine Überwachung aktiviert werden soll.

```
PS > auditpol /set /user:Cl1\student /subcategory:"File System" /success:enable
The command was successfully executed.
```

Tatsächlich können Sie mit der benutzerspezifischen Überwachung sogar einzelne Benutzer aus der Überwachung ausnehmen.

```
PS > auditpol /set /user:CL1\Admin /category:* /exclude
The command was successfully executed.
```

Neben dem Parameter */exclude* gibt es auch noch einen Parameter */include*. Da /include der Standard ist, müssen Sie ihn aber normalerweise nicht mit angeben.

Wenn Sie Einstellungen für einzelne Benutzer entfernen wollen, hilft Ihnen der Parameter */remove*. Er entfernt alle Einstellungen, so dass die im vorigen Beispiel gesetzte Ausnahme für den admin schnell wieder zurückgesetzt ist. Anschließende können Sie mit den Parametern */get /user* prüfen, ob die Überwachungseinstellungen tatsächlich entfernt worden sind.

```
PS > auditpol /remove /user:cl1\admin
The command was successfully executed.

PS > auditpol /get /user:CL1\admin /Category:*
No audit policy is defined for the user account.
```

Um alle Benutzerspezifischen Einstellungen zu entfernen, benutzen Sie den Parameter */allusers*.

```
PS > auditpol /remove /allusers
The command was successfully executed.
```

Statt des Benutzernamens kann auditpol auch die SID des Benutzers verarbeiten.

```
PS > auditpol /set /user:"{S-1-5-21-2658506762-1918846799-3508967985-1001}"
/category:*
The command was successfully executed.
```

Mit dem Parameter */resourceSACL* können Sie die globale Objektzugriffsüberwachung aktivieren (s. hierzu auch die Globale Objektzugriffsüberwachung in den Gruppenrichtlinien). Für die Konfiguration müssen Sie mit */Type* den Objekttyp angeben, den Sie überwachen wollen, mit User den zu überwachenden Benutzer und mit */success* bzw. */failure* erfolgreiche bzw. fehlgeschlagene Zugriffe. Der Parameter /Access gibt an, welche Zugriff Sie überwachen wollen. Wenn er ausgelassen wird, werden alle Zugriffe überwacht.

Achtung! Der Parameter */type* ist Case-Sensitiv. Sie müssen *File* oder *Key* mit großem F/K angeben, ansonsten bekommen Sie die Fehlermeldung "falscher Parameter"!

```
PS > auditpol /resourceSACL /set /type:Key /user:cl1\student /success
The command was successfully executed.

PS > auditpol /resourceSACL /set /type:File /user:"netz-weise\administrator"
/success /access:FW
The command was successfully executed.
```

Die Argumente für den Parameter /access können Sie sich von auditpol über die Hilfe anzeigen lassen oder folgender Tabelle entnehmen:

Berechtigungsmaske für globalen Dateizugriff	
Allgemeine Zugriffsrechte – universell für Dateisystem und Registrierung	
GA	Alle Zugriffe (GENERIC ALL)
GR	Alle Lesezugriffe (GENERIC READ)
GW	Alle Änderungen (GENERIC WRITE)
GX	Ausführen (GENERIC EXECUTE) – nur Dateien
Zugriffsrechte für Dateien	
FA	Alle Dateizugriffe (FILE ALL ACCESS)
FR	Lesender Dateizugriff (FILE GENERIC READ)
FW	Änderungen an Dateien (FILE GENERIC WRITE)
FX	Ausführen von Dateien (FILE GENERIC EXECUTE)

Zugriffsrechte für Registrierungsschlüssel	
KA	Alle Registry-Zugriffe (KEY ALL ACCESS)
KR	Registry lesen (KEY READ)
KW	Registry ändern (KEY WRITE)

Um die gesetzten globalen SACLs anzuzeigen, verwenden Sie den Parameter */view*, zusammen mit dem Überwachungstyp.

```
PS > auditpol /ResourceSacl /view /type:File
Eintrag:            1
Ressourcentyp:    File
Benutzer:             NETZ-WEISE\domain admins
Kennzeichen:            Erfolgreich
Bedingung:        (null)
Zugriffe:
    FILE_GENERIC_WRITE

Der Befehl wurde erfolgreich ausgeführt.
```

Zum Entfernen eines Eintrags können Sie den Parameter /remove verwenden.

```
PS > auditpol /ResourceSacl /remove /type:File /User:"netz-weise\Domain Admins"
```

Zum Entfernen aller globalen Überwachungseinträge eines Typs verwenden Sie */clear*.

```
PS > auditpol /ResourceSacl /clear /type:File
Der Befehl wurde erfolgreich ausgeführt.
```

Die Ereignisanzeige konfigurieren

Mit Windows Vista hat Microsoft die Ereignisprotokolle in das Windows Event Tracing (ETW) integriert. Dadurch haben sich die Performance und Funktionalität der Ereignisverwaltung deutlich verbessert.

Eine Ereignisprotokolldatei kann bis 16GB groß werden, wobei Microsoft eine maximale Dateigröße von 4GB empfiehlt. Dabei kann das System ohne Probleme 10.000 Ereignisse und mehr pro Sekunde speichern. Vermutlich sind auf aktuellen Systemen auch deutlich mehr Ereignisse ohne große Auswirkungen auf die Systemperformance möglich, aber es gibt dazu keine offiziellen Tests. Microsoft hat die empfohlenen Größen-Einstellungen in einem Knowledgebase-Artikel zusammengefasst. Sie finden Ihn unter https://support.microsoft.com/en-us/help/957662/recommended-settings-for-event-log-sizes-in-windows oder kurz https://bit.ly/2h2QiPa. Eine Beschreibung der Architektur finden Sie unter https://blogs.msdn.microsoft.com/ericfitz/2009/08/10/auditing-system-impact-on-performance oder kurz https://bitly.com/.

Die Konfiguration der Ereignisprotokolle kann an verschiedenen Stellen angepasst werden. Am einfachsten geht es über die Ereignisanzeige, indem Sie aus dem Kontextmenü des Protokolls die Eigenschaften aufrufen.

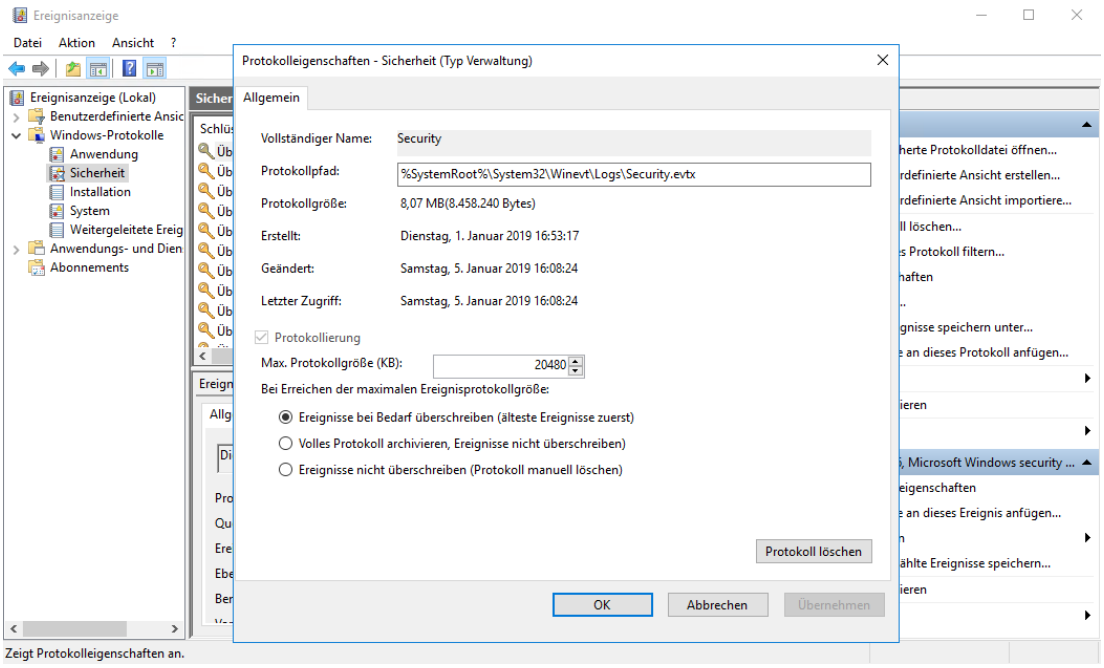


Bild 16 - In den Eigenschaften des Protokolls können Sie die Größe und das Überlaufverhalten konfigurieren

Hier finden Sie den Ablagepfad des Protokolls, die aktuelle und maximale Größe und das Überlaufverhalten. Wenn Sie die Sicherheitsüberwachung aktivieren, ist es empfehlenswert, die maximale Größe des Sicherheitsprotokolls anzupassen.

Wenn ein Protokoll die maximale Größe erreicht hat, bestimmt das Überlaufverhalten, was mit weiteren Ereignissen passiert.

Einstellung	Verhalten
Ereignisse bei Bedarf überschreiben	Überschreibt das Ereignisprotokoll, beginnend bei den ältesten Ereignissen. Mit dieser Einstellung ist es einem Angreifer möglich, sein

	Verhalten zu verschleiern, indem er so viele Sicherheitsereignisse generiert, dass die interessanten Einträge überschrieben werden.
Ereignisse nicht überschreiben	Wenn das Ereignisprotokoll voll ist, gehen alle weiteren Ereignisse verloren. Ein Angreifer kann jetzt zuerst das Protokoll mit sinnlosen Ereignissen vollschreiben, bevor er seine Arbeit beginnt.
Volles Protokoll archivieren (ab Vista)	Das Ereignisprotokoll wird im Ordner *%WinDir%\System32\winevt\Logs* unter dem Namen *Archive-<LogName>- <Datum>- <Zeit>.evtx* archiviert und ein neues Log angelegt, sobald die Datei voll ist.

Sicherheitsprotokollierung bringt im Normalfall wenig, wenn durch geschicktes Ausnutzen der Grenzwerte die Protokollierung umgangen werden kann. Auf kritischen Systemen können Sie daher in der lokalen Sicherheitsrichtlinie unter *Sicherheitseinstellungen > Lokale Richtlinien > Sicherheitsoptionen* die *Einstellung Überwachung: System sofort herunterfahren, wenn Sicherheitsüberprüfungen nicht protokolliert werden können* aktivieren. Anders als der Titel es vermuten lässt, fährt Windows mit dieser Einstellung nicht kontrolliert herunter, sondern erzeugt einen Bluescreen. Sie müssen den Rechner dann im abgesicherten Modus starten und das Ereignisprotokoll löschen, bevor Windows wieder normal hochfährt. Wenn Sie diese Einstellung nutzen wollen, sollten Sie auf jeden Fall die Größe des Sicherheitsprotokolls anpassen und überwachen, um zu verhindern, dass Ihr System wegen fehlender Wartung vollläuft. Eine genaue Beschreibung der Funktion finden Sie unter https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/audit-shut-down-system-immediately-if-unable-to-log-security-audits oder kurz https://bit.ly/2scjuGf.

Alternativ können Sie Eventlogs auch von der Kommandozeile aus konfigurieren. Hierfür stehen Ihnen eine Reihe von Powershell-Cmdlets zur Verfügung, die allerdings nur mit den klassischen Eventlogs funktionieren.

- Limit-Eventlog
- Clear-Eventlog
- New-Eventlog
- Write-Eventlog
- Remove-Eventlog

Mit *Limit-Eventlog* können Sie oben beschriebenen Überlaufeinstellungen anpassen, allerdings mit Einschränkungen. *Limit-Eventlog* kann den Archivmodus nicht aktivieren! Dafür ist es allerdings möglich festzulegen, wie lange Ereignisse nicht überschrieben werden dürfen, was in der Ereignisanzeige (nicht mehr) geht.

```
PS > Limit-EventLog -LogName Security -RetentionDays 14 -OverflowAction
OverwriteOlder -MaximumSize 200MB
```

Mit Clear-Eventlog können Sie ein Eventlog löschen.

```
PS > Clear-EventLog -LogName Security
```

Immer wenn eine Eventlog gelöscht wird, wird als erstes Ereignis die Löschung mit der EventID 1102 protokolliert.

New-Eventlog, Write-Eventlog und Remove-Eventlog habe ich hier nur der Vollständigkeit halber aufgeführt. Mit Ihnen können Sie aus Skripten heraus Ereignisprotokoll-Einträge generieren und eigene Ereignisprotokolle anlegen sowie löschen. Sie sind für die Überwachung aber unerheblich.

Die bessere Alternative zu den Powershell-Cmdlets ist das Kommandozeilenwerkzeug *wevtutil.exe*. *Wevtutil.exe* ist das Schweizer Taschenmesser der Ereignisprotokollverwaltung. Mit *wevtutil.exe /?* können Sie sich alle Parameter anzeigen lassen. Die Hauptfunktionen werden direkt hinter dem Programmaufruf angegeben, gefolgt von Unterparametern, die mit einem Schrägstrich angeführt werden. Die Argumente für die Unterparameter werden direkt mit einem Doppelpunkt eingeführt: */enabled:true*. Alle Parameter haben eine lange und eine abgekürzte Schreibform. Das wird in der Hilfe durch den senkrechten Strick kenntlich gemacht. Sie können zum konfigurieren des Protokolls also *Wevtutil.exe sl* oder *Wevtutil.exe set-log* schreiben. Wenn Sie sich die Funktionsweise einer Hauptfunktion anzeigen lassen wollen, geben Sie die Hauptparameter ein, gefolgt von einem /?.

```
PS > wevtutil.exe sl /?
Modify the configuration of a log.

Usage:

wevtutil { sl | set-log } <LOG_NAME> [/OPTION:VALUE [/OPTION:VALUE] ...]

<LOG_NAME>
String that uniquely identifies a log. If option /c is specified, <LOG_NAME>
should not be specified since it is read from the config file.

Options:

You can use either the short (for example, /e) or long (for example, /enabled)
version of the option names. Options and their values are not case-sensitive.

/{e | enabled}:[true|false]
Enable or disable a log.

/{q | quiet}:[true|false]
Quiet display option. No prompts or messages are displayed to the user. If not
specified, the default is true.

[…]
```

Ich möchte an dieser Stelle nur auf die Konfiguration des Ereignisprotokolls eingehen. Im folgenden Beispiel wird das Sicherheits-Protokoll auf 1GB maximale Größe gesetzt, und die automatische Archivierung wird aktiviert. Der zweite Befehl ruft die Konfiguration des Sicherheits-Protokolls ab.

```
PS > wevtutil.exe sl Security /maxsize:1073741824 /retention:true /autobackup:true

PS > wevtutil.exe gl security
name: security
enabled: true
type: Admin
owningPublisher:
isolation: Custom
channelAccess: O:BAG:SYD:(A;;0xf0005;;;SY)(A;;0x5;;;BA)(A;;0x1;;;S-1-5-32-573)
logging:
  logFileName: %SystemRoot%\System32\Winevt\Logs\security.evtx
  retention: true
  autoBackup: true
  maxSize: 1073741824
publishing:
  fileMax: 1
```

Der merkwürdige String hinter *channelAccess* ist ein sogenannter Security-Descriptor. Es handelt sich um eine textuelle Darstellung der Berechtigungen und beschreibt, wer das Ereignisprotokoll öffnen kann. Im Falle des Sicherheitsprotokolls sind das normalerweise nur das System und Administratoren. Benutzer haben explizit keinen Zugriff auf das Sicherheitsprotokoll. Wie Sie den Security-Descriptor mit Hilfe von Powershell in eine lesbare Form umwandeln können, erfahren Sie weiter unten.

Wenn Sie auf einer großen Anzahl an Systemen die Ereignisprotokolle konfigurieren wollen, verwenden Sie hierfür am Besten Gruppenrichtlinien. Öffnen sie hierzu ein GPO oder legen Sie ein neues an. Für die Verwaltung der Ereignisprotokolle stellen die Windows gleich an zwei Stellen Gruppenrichtlinien zur Verfügung, und zwar einmal in den Sicherheitseinstellungen des Computers unter *Richtlinien – Windows Einstellungen – Sicherheitseinstellungen – Ereignisprotokoll*, und einmal in den Administrativen Vorlagen unter *Windows-Komponenten – Ereignisprotokolldienst*. Da Sie unter den administrativen Vorlagen mehr Konfigurationsmöglichkeiten haben, werde ich nur die diese Einstellungen eingehen.

Sie finden unterhalb der Kategorie Ereignisprotokolldienst 4 Unterkategorien. Jede bezieht sich auf ein Ereignisprotokoll und seine Einstellungen. Die Einstellungen sind bei allen Protokollen fast identisch.

Einstellung	Bedeutung
Speicherort der Protokolldatei steuern	Gibt den Ordner an, an dem die .evtx-Datei abgelegt werden soll. Standardmäßig werden alle Protokolle unter *%SYSTEMROOT%\System32\winevt\Logs* abgelegt.

Maximale Protokolldateigröße (KB) angeben	Passen Sie hier die maximale Größe des Logs an. Die Dateigröße muss ein Vielfaches von 64KB und mindestens 1MB groß sein. Zur schnellen Umrechnung in KB können Sie Powershell verwenden, indem Sie in die Konsole einfach die gewünschte Größe mit Einheit angeben und dann durch 1KB teilen, also z.B.:
	`PS > 1GB/1KB`
	`1048576`

Volles Protokoll automatisch sichern	Hiermit aktivieren Sie die Protokollarchivierung. Volle Protokolle werden automatisch in eine Archivdatei verschoben (s. oben)
Protokollzugriff konfigurieren	Legen Sie fest, wer Zugriff auf das Ereignisprotokoll haben soll. Sie müssen hier einen Security Descriptor-String hinterlegen. Am besten kopieren Sie den aus einem konfigurieren System, indem Sie sich mit wevtutil.exe den String ausgeben lassen.
Protokollzugriff konfigurieren (Legacy)	Für Systeme vor Vista wurden die Berechtigungen anders gesetzt. Hier geben Sie den Security-Descriptor für Windows Server 2003 / XP und älter an.
Verhalten des Ereignisprotokolls steuern, wenn die Protokolldatei ihre maximale Größe erreicht	Wenn Sie diese Richtlinie aktivieren, kann das Ereignisprotokoll nicht überschrieben werden. Das entspricht der manuellen Konfiguration *Ereignisse nicht überschreiben*. Deaktivieren Sie diese Richtlinie, entspricht das der Einstellung *Ereignisse bei Bedarf überschreiben*.
Protokollierung aktivieren (nur im Setup-Protokoll)	Schalten Sie die Ereignisprotokollierung komplett aus.

Das Ereignisprotokoll sichten

Die Daten aller Überwachungen werden im Ereignisprotokoll gespeichert, das sich mit der Ereignisanzeige oder Powershell filtern lässt. Im Folgenden werde ich zuerst die Möglichkeiten der Ereignisanzeige beschreiben, um dann auf die erweiterten Möglichkeiten des Cmdlets *Get-Winevent* einzugehen.

Die Ereignisanzeige, die Sie mit Eventvwr.msc oder unter Windows 10 aus dem Kontextmenü des Startbuttons aufrufen können, bietet verschiedene Möglichkeiten, Protokolle zu filtern. Die Filter können auch rechts in der Aktions-Anzeige oder im Kontextmenü des Logs konfiguriert werden.

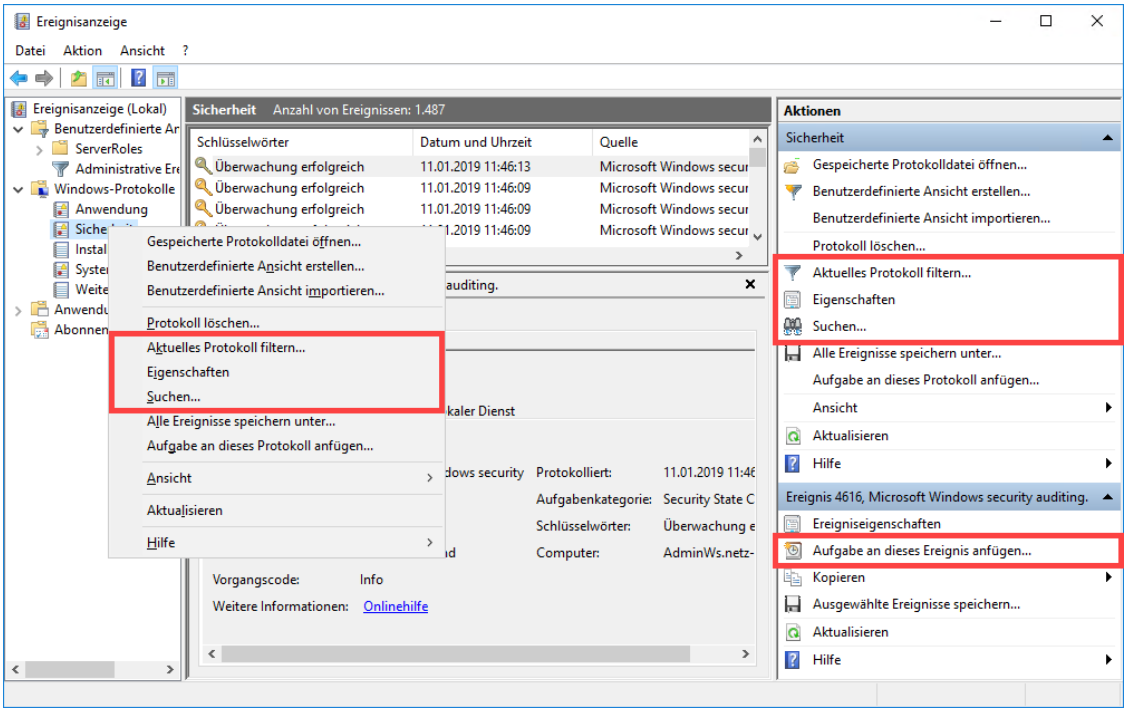


Bild 17 - Das Ereignisprotokoll bietet verschiedene Möglichkeiten, das Log zu sichten

Für eine Volltextsuche über alle Events verwenden Sie *Suchen…* in der Aktions-Leiste. Es öffnet sich ein Eingabefenster, in das Sie den Begriff eingeben können, der in der Beschreibung des Ereignisses enthalten sein soll. Die Ereignisanzeige stoppt beim ersten Ereignis, das den Begriff enthält.

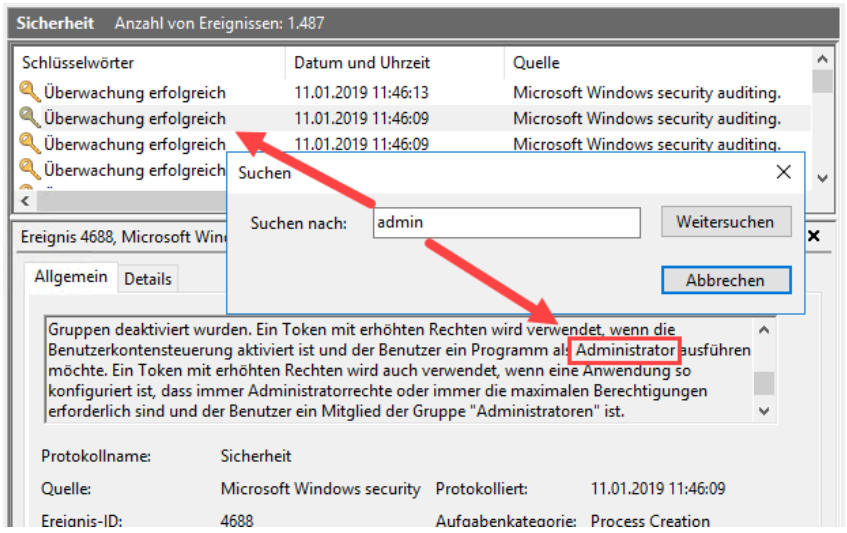


Bild 18 - Der Treffer wird ausgewählt und angezeigt

Das gefundene Ereignis wird in der Ereignisliste ausgewählt. In Bild 18 können Sie sehen, dass die Suche auch nach Teilstrings filtert – der Suchbegriff Admin hat auch Administrator gefunden. Um weitere Ereignisse zu finden, müssen Sie *Weitersuchen* auswählen. Das kann bei einem umfangreichen Protokoll sehr mühsam werden.

Alternativ können Sie einen Protokollfilter verwenden. Wählen Sie dafür im Aktionsfenster *Aktuelles Protokoll filtern* aus.

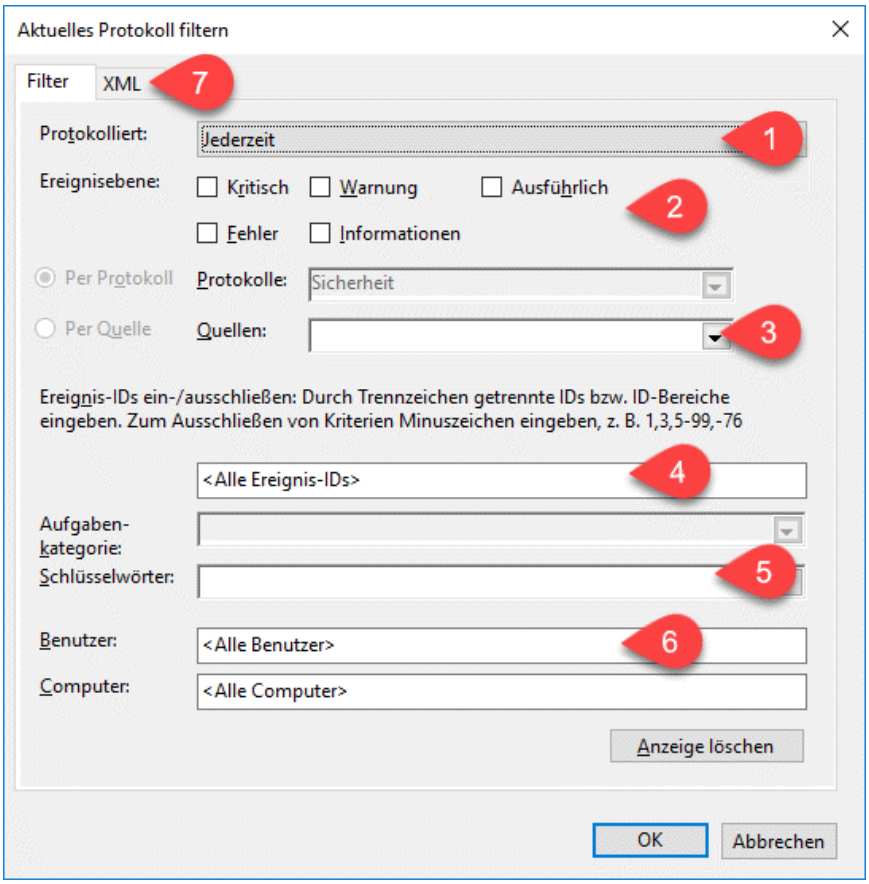


Bild 19 - Die allgemeinen Protokollfilter bieten wenig Möglichkeiten

Leider sind die Möglichkeiten, die Sicherheitsanzeige zu filtern, eingeschränkt. Die hilfreichsten Filter für Sicherheitsrelevante Ereignisse sind die Zeit, die Sie über das Menü *Protokolliert* (1) einschränken, und die Ereignis-ID, die als Zahlenwert unter (4) angegeben wird. Hier können Sie auch mehrere IDs kommasepariert eintragen, und sogar Bereiche von Zahlen, indem Sie den Start- und Endwert mit einem Bindestrich angeben. Die *Ereignisebene* (2) ist unerheblich, da sie für Sicherheitseinträge immer *Information* ist. Die Quelle (Programm), der das Ereignis zuzuordnen ist, und die unter (3) angegeben werden kann, ist im Sicherheitsprotokoll immer *Microsoft Windows Security*. Schlüsselwörter (5) ist leider kein Freitextfeld, sondern es kann nur nach den vorgegebenen Schlüsselwörtern gefiltert werden.

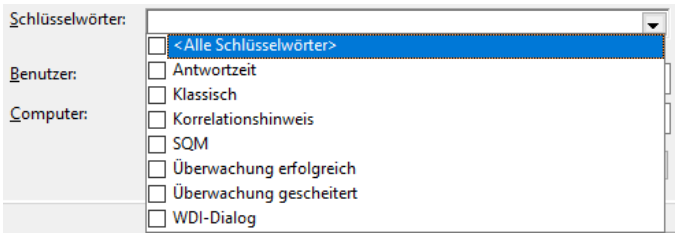


Bild 20 - Die Schlüsselwörter sind nur eingeschränkt nutzbar

Da die Schlüsselwörter vorgegeben sind, kann man mit Ihnen meistens nicht wirklich etwas anfangen. Bestenfalls interessant ist die Filterung nach erfolgreichen und gescheiterten Anmeldeereignissen. Eine Beschreibung der übrigen Schlüsselwörter finden Sie unter https://alistg.wordpress.com/2011/07/04/using-custom-views-in-windows-event-viewer-part-1/ oder kurz https://bit.ly/2AGHbuK.

Die XML-Ansicht von Ereignis-Einträgen

Um nach beliebigen Schlüsselwörtern zu suchen, können Sie mit Powershell arbeiten. Das Cmdlet *Get-Eventlog* stellt Ihnen mit dem Parameter *-Message* die Möglichkeit zur Verfügung, mit Hilfe von Wildcards nach einzelnen Strings zu suchen.

```
Get-EventLog -LogName Security -Message *Membership*
```

Das funktioniert sehr gut, solange das Ereignis, dass Sie suchen, sich durch ein einzelnes Schlüsselwort im der Ereignisbeschreibung identifizieren lässt. Sicherheitsevents sind aber normalerweise ziemlich komplex und bestehen im Beschreibungstext selbst wieder aus mehreren Feldern (s. Bild 21).

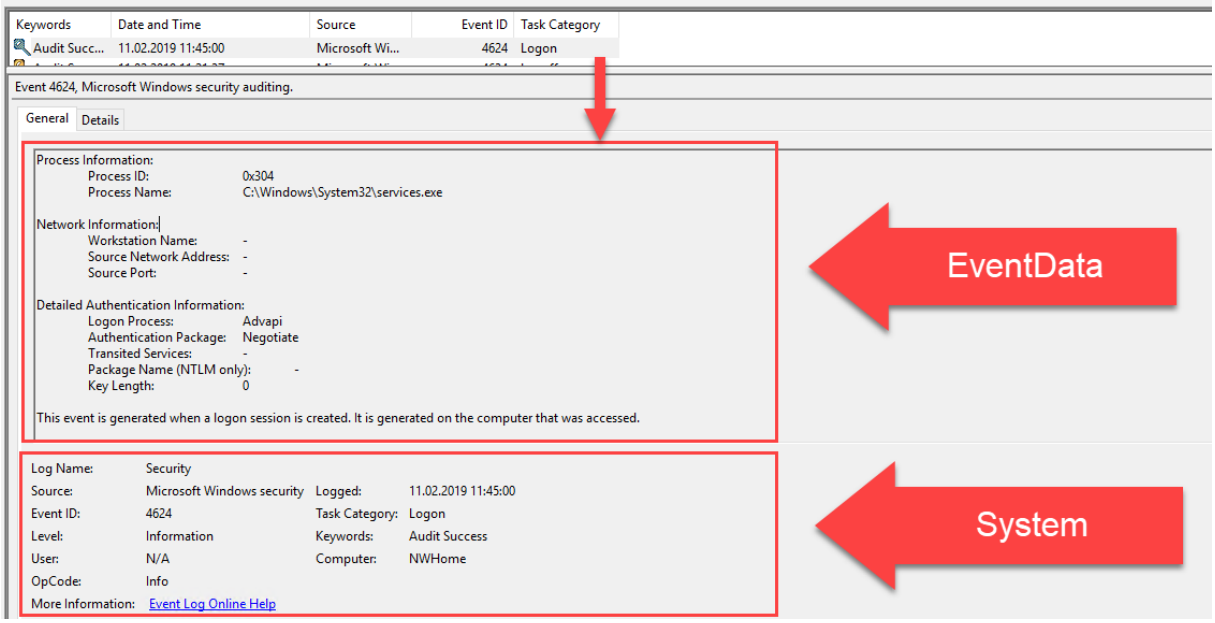


Bild 21 - Die Beschreibung (Eventdata) besteht selbst aus Datenfeldern

Sie können das Ereignisprotokoll auch auf die Inhalte dieser Datenfelder filtern, allerdings müssen Sie hierfür ein bisschen tiefer in die Trickkiste greifen.

Windows erlaubt Ihnen, alle Ereignisse auch in einer XML-Ansicht darzustellen. Öffnen Sie dazu ein beliebiges Ereignis und wählen Sie die Registerkarte Details. Hier schalten Sie auf die XML-Ansicht um (Bild 22).

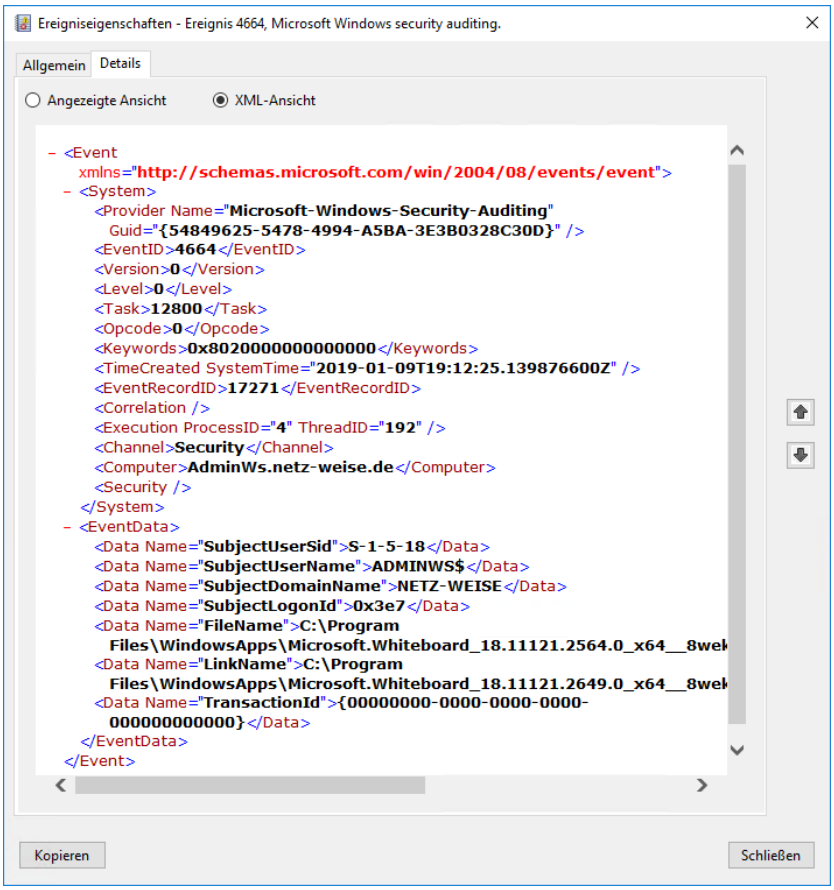


Bild 22 - Alle Ereignisse werden im XML-Format abgelegt

XML ist ein Datenformat, dass jeden Datensatz zwischen zwei Tags (Marker) einschließt. Zu jedem öffnenden Tag gehört auch ein schließendes Tag, bei dem vor dem Tag-Namen noch einen Schrägstrich (/) steht. Tags sind in spitze Klammern eingeschlossen.

Die EventID befindet sich z.B. zwischen dem öffnenden Tag *<EventID>* und dem schließenden Tag *</EventID>*. Achten Sie darauf, dass XML Groß-Kleinschreibesensitiv ist. *<eventid>* ist also nicht das gleiche wie *<EVENTID>*.

Mit XML können Hierarchien abgebildet werden, da sich ein Tag in einem anderen Tag befinden kann. Das Tag *<System>* schließt z.B. *<EventID>* oder *<TimeCreated>* ein, steht aber selbst innerhalb von <Event>.

Wenn hinter einem Tag-Namen ein Leerzeichen steht, gefolgt von weiteren Werten, die als Schlüssel-Wert-Paar angegeben sind, wie *<Data Name="SubjectUserSid">,* handelt es sich um Attribute. Der Name des Tags endet mit dem Leerzeichen, ist also *Data*, und *Name="SubjectUserSid"* ist das Attribut mit dem Namen *Name* und dem Wert "*SubjectUserSid*". Attribute können selbst Daten darstellen, oder für die Beschreibung des Tags verwendet werden. Im Eventlog werden Daten nicht in Attributen gespeichert, sondern immer im Tag.

XML wird verwendet, um Daten ähnlich einer Datenbank strukturiert abzulegen. Und genau wie bei einer Datenbank gibt es eine Abfragesprache, die als XPath bezeichnet wird. Wenn Sie das Ereignisprotokoll mit komplexen Abfragen filtern wollen, müssen Sie das mit XPath tun.

Windows unterstützt für die Abfrage von Ereignisprotokollen einen eingeschränkten Teil der XPath 1.0 Spezifikation. Da macht Abfragen relativ einfach, da die Menge der Sprachelemente eingeschränkt ist.

XML-Filter und XPath-Abfragen

Um XPath-basierte Filter zu definieren, öffnen Sie die Registerkarte XML in den Protokollfilter-Einstellungen (s. (7) in Bild 19) und aktivieren die Checkbox *Manuell bearbeiten*. Die Warnung, dass Sie den Filter dann nicht mehr mit der GUI konfigurieren können, bestätigen Sie mit ja.

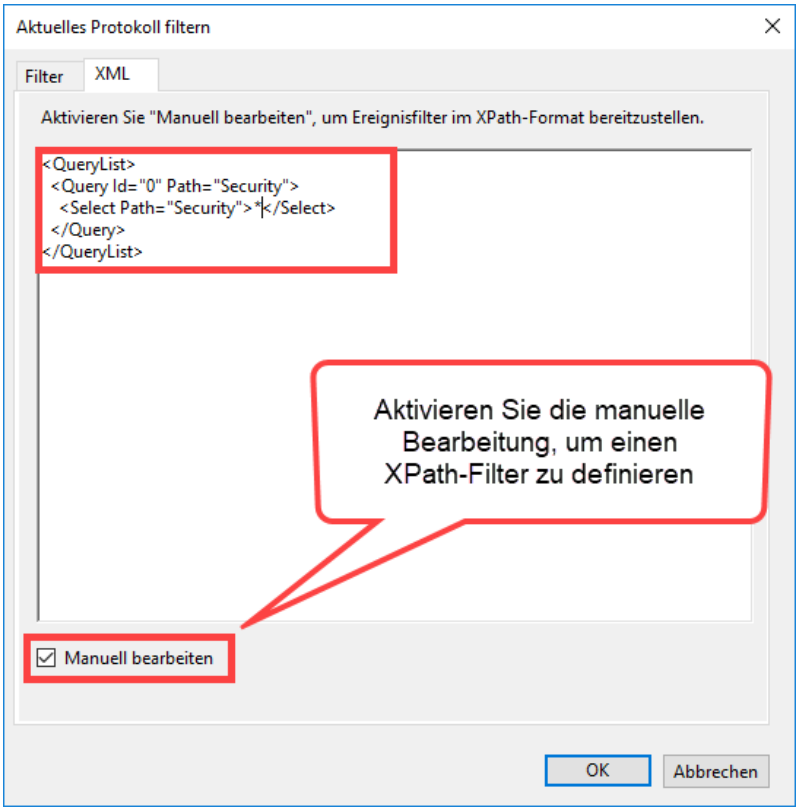


Bild 23 - der XML-Filter erlaubt eigene XPath-Abfrageen zu definieren.

Hier können Sie einen XML-Filter hinterlegen. Ein XML-Filter besteht aus einer oder mehreren XPath-Abfragen, die gemeinsam ausgewertet und selbst in XML dargestellt werden. Wenn Sie mehrere XPath-Abfragen kombinieren, werden diese and-Verknüpft, müssen also alle gemeinsam wahr sein, damit ein Ereignis durch den Filter erfasst wird. Die Abfragen stehen im *<Query>*-Tag, das wiederum durch das *<QueryList>*-Tag umfasst ist.

Die Standard-Abfrage

```
<Select Path="Security">*</Select>
```

erfasst alle Ereignisse des Sicherheits-Protokolls. Das Protokoll wird über das Attribut Path des *<Select>*-Tags angegeben. Innerhalb des *<Select>*-Tags folgt die XPath-Abfrage, die in diesem Fall nur aus einem * besteht. * ist ein Platzhalter für beliebige Elemente. Alternativ können Sie statt des * Event eingeben, da alle Wurzelelemente vom Typ Event sind, so dass folgende Abfrage das gleiche Ergebnis zeigt:

```
<Select Path="Security">Event</Select>
```

Wenn Sie nach bestimmten Informationen suchen, müssen Sie zuerst herausfinden, wo in der XML-Darstellung des Events diese Information hinterlegt ist. Jeder Logeintrag speichert im Tag *<System>* allgemeine Informationen über das Ereignis. Sicherheitseinträge verwalten die Detailinformationen im Tag *<EventData>*. In der grafischen Darstellung der Registerkarte *Allgemein* der Events entspricht *<EventData>* den Datenfeldern des Meldungstextes, während *<System>* die Metainformationen zusammenfasst.

```
<System>
        <Provider Name="Microsoft-Windows-Security-Auditing" Guid="{…}" />
        <EventID>4624</EventID>
        <Version>2</Version>
        <Level>0</Level>
        <Task>12544</Task>
        <Opcode>0</Opcode>
        <Keywords>0x8020000000000000</Keywords>
        <TimeCreated SystemTime="2019-02-11T10:45:00.061240700Z" />
        <EventRecordID>191537</EventRecordID>
        <Correlation ActivityID="{207935A6-BB48-0002-B635-792048BBD401}" />
        <Execution ProcessID="780" ThreadID="4932" />
        <Channel>Security</Channel>
        <Computer>Server1</Computer>
<Security  />
<EventData>
        <Data Name="SubjectUserSid">S-1-5-18</Data>
        <Data Name="SubjectUserName">NWHOME$</Data>
        <Data Name="SubjectDomainName">WORKGROUP</Data>
        <Data Name="SubjectLogonId">0x3e7</Data>
        <Data Name="TargetUserSid">S-1-5-18</Data>
        <Data Name="TargetUserName">SYSTEM</Data>
        <Data Name="TargetDomainName">NT AUTHORITY</Data>
        <Data Name="TargetLogonId">0x3e7</Data>
        <Data Name="LogonType">5</Data>
        <Data Name="LogonProcessName">Advapi</Data>
        <Data Name="AuthenticationPackageName">Negotiate</Data>
        <Data Name="WorkstationName">-</Data>
        <Data Name="LogonGuid">{00000000-0000-0000-0000-000000000000}</Data>
        <Data Name="TransmittedServices">-</Data>
        <Data Name="LmPackageName">-</Data>
        <Data Name="KeyLength">0</Data>
        <Data Name="ProcessId">0x304</Data>
        <Data Name="ProcessName">C:\Windows\System32\services.exe</Data>
        <Data Name="IpAddress">-</Data>
        <Data Name="IpPort">-</Data>
        <Data Name="ImpersonationLevel">%%1833</Data>
        <Data Name="RestrictedAdminMode">-</Data>
        <Data Name="TargetOutboundUserName">-</Data>
        <Data Name="TargetOutboundDomainName">-</Data>
        <Data Name="VirtualAccount">%%1843</Data>
        <Data Name="TargetLinkedLogonId">0x0</Data>
        <Data Name="ElevatedToken">%%1842</Data>
</EventData>
```

Um mit XPath nach einem bestimmten Tag zu filtern, geben Sie den Pfad zum gesuchten Element an. Sie können die einzelnen Tags hierfür in eckige Klammern schreiben, oder Sie können, wie in XPath normalerweise üblich, den Schrägstrich verwenden. Um alle Events anzuzeigen, die die Eigenschaft EventID haben, können Sie also eine der folgenden beiden Schreibweisen verwenden.

```
<Select Path="Security">*[System[EventID]]
```

Vor jedem Element steht eine öffnende Klammer. Am Ende werden alle Klammern geschlossen.

Alternativ verwenden Sie die Xpath-Syntax.

```
<Select Path="Security">*/System/EventID</Select>
```

Dieses Beispiel ist ziemlich sinnlos, da alle Events eine EventID haben. Stattdessen werden Sie vermutlich eher nach einer bestimmten EventID suchen. Hierzu geben Sie hinter dem gesuchten Element, getrennt durch ein Gleichheitszeichen, die gesuchte ID an.

```
<Select Path="Security">*[System[EventID=4624]]
```

Auch hier können Sie wieder die XPath-Schreibweise Klammern verwenden.

```
<Select Path="Security">*/System/EventID=4624</Select>
```

Mehrere Bedingungen können über *and* bzw. *or* verknüpft werden. Eine and-Verknüpfung liefert nur dann Daten zurück, wenn alle angegebenen Bedingungen wahr sind, während bei einer or-Verknüpfung nur eine der angegebenen Suchbedingungen wahr sein muss. Wenn Sie nach zwei EventIDs suchen, werden Sie immer die or-Verknüpfung verwenden, da ein Event nie zwei unterschiedliche EventIDs gleichzeitig haben kann.

Da die folgenden Abfragen etwas länger werde, gebe ich ab jetzt nur noch die Abfrage an und lassen das *<Select>*-Tag aus.

```
*[System[EventID=4798 or EventID=4797]]
```

In der XPath-Schreibweise sieht das so aus.

```
*/System/EventID=4798 or */System/EventID=4797
```

Hier sehen Sie den wesentlichen Unterschied zwischen den beiden Schreibweisen. Verwenden Sie eckige Klammern, um Elemente zu trennen, können Sie für ein Element innerhalb der Klammern mehrere Bedingungen angeben, während Sie bei der XPath-Schreibweise immer den vollständigen Pfad angeben müssen.

Eckige Klammern	XPath
'*[System[EventID=4798 or EventID=4797]]'	'*/System/EventID=4798 or */System/EventID=4797'

Da die XPath-Schreibweise bei komplexen Bedingungen deutlich länger wird, werde ich bei den folgenden Beispielen auf die XPath-Schreibweise verzichten.

Attribute eines Tags werden durch ein @ gekennzeichnet. Wenn Sie z.B. alle Ereignisse suchen, bei denen das Attribut Activity-ID den Wert *{207935A6-BB48-0002-B635-792048BBD401}* hat, filtern Sie auf *@ActivityID={207935A6-BB48-0002-B635-792048BBD401}.* Die ActivityID ist ein Attribut des Elements Correlation.

```
*[System[Correlation[@ActivityID="{207935A6-BB48-0002-B635-792048BBD401}"]]]
```

Die Attribute sind speziell für den Eventdata-Zweig relevant, da für die Ereignisse des Sicherheitsprotokolls hier alle Daten noch einmal in Form von einzelnen Tags aufbereitet sind, so dass es mit Hilfe von XPath sehr einfach möglich ist, nach einzelnen Schlüsselwerten wie z.B. einem Benutzernamen oder einem Anmeldetyp zu suchen. Alle Daten sind in Data-Tags abgelegt, deren Inhalt sich erst aus dem Attribut erschließt. Hierfür hat jedes Data-Tag ein Attribut *Name*, das den hinterlegten Wert benennt.

```
<EventData>
        <Data Name="SubjectUserSid">S-1-5-18</Data>
        <Data Name="SubjectUserName">NWHOME$</Data>
        <Data Name="SubjectDomainName">WORKGROUP</Data>
        <Data Name="SubjectLogonId">0x3e7</Data>
        <Data Name="TargetUserSid">S-1-5-21-2426476874-3228847160-2382446698-1000</Data>
        <Data Name="TargetUserName">admin</Data>
        <Data Name="TargetDomainName">NWHOME</Data>
        <Data Name="TargetLogonId">0x6cc0aed</Data>
        <Data Name="LogonType">10</Data>
        ...
</EventData>
```

Um nach allen Events zu suchen, die ein Datenfeld mit dem Attribut *Name="TargetUserName"* besitzen und bei denen der gespeicherte Wert *"Administrator"* ist, müssen Sie zwei Bedingungen kombinieren - *@Name="SubjectUserName"* und *Data="Administrator"*.

```
*[EventData[Data[@Name="SubjectUserName"] and Data="Admin"]]
```

Sie können auch weitere Bedingungen zur Abfrage hinzufügen. Für Anmeldeversuche filtern Sie auf die EventID 4264.

```
*[System[EventID=4624]] and *[EventData[Data[@Name="TargetUserName"] and
Data="Admin"]]
```

Prinzipiell ist es dadurch möglich, beliebig komplexe Abfragen zu erstellen.

Das letzte Sprachelement, dass Ihnen zur Verfügung steht, sind Funktionen. Funktionen können innerhalb des XPath-Ausdrucks Berechnungen ausführen. Hier machen sich die Einschränkungen von XPath am stärksten bemerkbar, denn aus der Menge der in XPath verfügbaren Funktionen werden nur drei unterstützt. Achten Sie bei der Erstellung von Skripten außerdem darauf, dass Funktionsnamen klein geschrieben sein müssen!

Funktion	Beschreibung
position()	Bestimmt die Position (Tiefe) eines Elements innerhalb eines XML-Baums. Position kann nur auf Blatt-Objekte angewendet werden.
band()	Binary AND. Das binäre Und verknüpft zwei Integer-Werte bitweise. Ist das Ergebnis ungleich Null, liefert die Funktion True zurück, ansonsten False. band() kann Beispielsweise verwendet werden, um auf das Element *Schlüsselwörter* zu filtern (s. Bild 19, (5)). Schlüsselwörter werden in der XML-Darstellung als Binärmuster dargestellt. <Keywords>0x8020000000000000</Keywords>
timediff()	Die timediff-Funktion berechnet den Unterschied in Millisekunden zwischen zwei Zeitwerten im Filetime-Format. Das Ergebnis ist positiv, wenn der zweite Wert einen späteren Zeitwert darstellt. Wird timediff() nur ein Argument übergeben, berechnet er die Zeitdifferenz zur aktuellen Systemzeit.

Während ich keinen sinnvollen Anwendungszweck für die position()-Funktion in Verbindung mit Ereignisprotokollen kenne, kann man mit band() nach Schlüsselwörtern filtern. Es gibt, je nach Protokoll, eine Reihe von vorgegebenen Schlüsselwörtern. Jedes Schlüsselwort wird durch ein Bit innerhalb eines 64 Bit langen Bitmusters repräsentiert. Wenn Sie alle Events filtern wollen, bei denen die Schlüsselwörter *Überwachung erfolgreich* oder *Überwachung fehlgeschlagen* gesetzt sind, müssen Sie nach 13510798882111488 suchen.

Um die Bitrepräsentation zu erhalten, können Sie die angegebene Zahl einfach in den Windows-Taschenrechner einfügen und auf die Programmieransicht umstellen. Das hat keine Relevanz, da Sie in der Xpath-Abfrage immer die Integer-Repräsentation angeben, die Ihnen die Ereignisanzeige automatisch erzeugen kann, aber es ist interessant zu sehen, wie sich aus der offenbar völlig sinnlosen Integer-Zahl plötzlich ein logisches Muster ergibt.

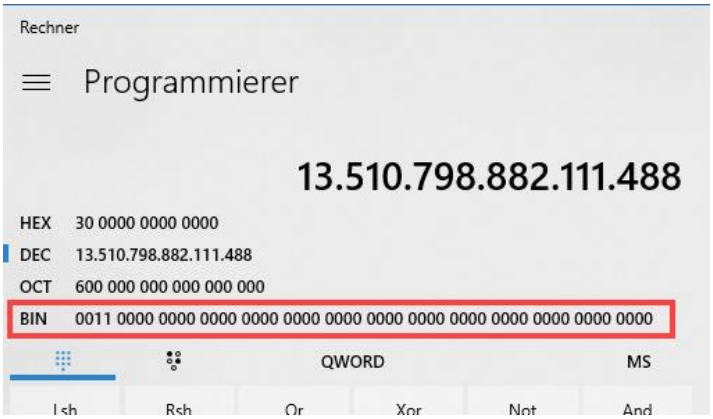


Bild 24 - Die Dezimalzahl in Bit umgewandelt

Sie sehen, dass das 61ste und 62ste Bit (von hinten gezählt) gesetzt sind, was den gesuchten Kategorien entspricht. Wenn weitere Kategorien ausgewählt wären, wären weitere Bits gesetzt.

Woher weiß man jetzt, welches Zahl welchen Schlüsselwörtern entspricht? Ganz einfach, gehen Sie in die Ereignisanzeige, erstellen Sie in der GUI einen neuen Filter auf ein Protokoll, wählen Sie die Schlüsselwörter aus, die Sie filtern möchten, und wählen Sie anschließend die Registerkarte XML aus.

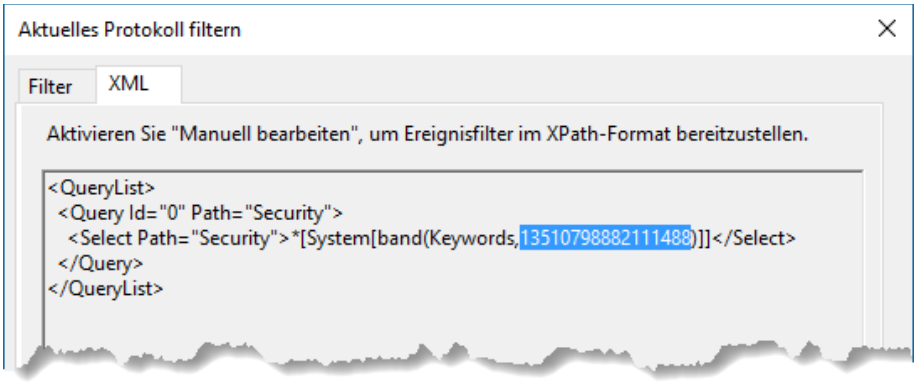


Bild 25 - Windows erzeugt das Bitmuster (und die band-Abfrage) für uns

Tatsächlich können Sie sich den Filter auf diese Art und Weise sehr einfach automatisch erzeugen.

timediff() können Sie verwenden, um Ereignisse innerhalb einem bestimmten Zeitraum zu filtern. Wenn Sie alle Anmeldeereignisse der letzten 24 Stunden ausgeben wollen, müssen Sie das Attribut Systemtime aus dem Element TimeCreated filtern. Übergeben Sie timediff() nur einen Datumswert, wird die Differenz zur Systemzeit in Millisekunden ausgerechnet. Sie können das Ergebnis dann mit 86400000 vergleichen, was den Millisekunden in 24 Stunden entspricht.

```
*[System[EventID=4624 and TimeCreated[timediff(@SystemTime) <= 86400000]]]
```

Um die Anzahl von Millisekunden eines Zeitwertes zu berechnen, können Sie das Powershell-Cmdlet *New-Timespan* bemühen. Es berechnet Ihnen Zeitdifferenzen zwischen zwei Zeiten, wenn Sie die Parameter *-Start* und *-End* verwenden, oder rechnet Ihnen Zeitwerte um, wenn Sie die Parameter *-Days*, *-Hours*, *-Minutes* oder *-Seconds* benutzen.

```
PS > (New-TimeSpan -End (get-date).AddDays(1)).TotalMilliSeconds
86400000

PS > (New-TimeSpan -Days 1).TotalMilliseconds
86400000
```

Wie Sie an der timediff()-Funktion sehen können, gibt es noch mehr Vergleichsoperatoren als =. Aus den in XPath 1.0 definierten Operatoren stehen Ihnen folgende zur Verfügung:

Operator	Bedeutung
OR	Oder-Verknüpfung, mind. eine der angegebenen Bedingungen muss wahr sein
AND	Und-Verknüpfung, alle angegebenen Bedingungen müssen wahr sein
=	Gleich
!=	Ungleich
<=	Der Wert Links muss kleiner oder gleich dem rechten Wert sein
<	Der Wert links muss kleiner dem rechten Wert sein
>=	Der Wert links muss größer oder gleich dem Wert rechts sein
>	Der Wert links muss größer dem Wert rechts sein

Was hier fehlt ist ein Like-Operator, der nach Mustern sucht. Das ist eine ziemlich starke Einschränkung, die Sie mit Hilfe von Powershell aber umgehen können.

Mehrere XPath-Abfragen in einem XML-Filter kombinieren

Wenn Sie einen Eventfilter erstellen, geben Sie immer einen XML-Filter an. Der Filter besteht meistens nur aus einer einzigen XPath-Abfrage, kann aber mehrere Abfragen zusammenfassen. Das ist vor allem dann interessant, wenn man bestimmte Events aus dem Ergebnis ausschließen möchte.

Um mehrere XPath-Abfragen zu kombinieren, legen Sie innerhalb des <Query>-Tags mehrere <Select>-Tags an. In jedem <Select>-Tag können Sie eine eigenständige Abfrage definieren. So ist es auch möglich, über mehrere Protokolle gleichzeitig zu filtern.

```
<QueryList>
  <Query Id="0">
    <Select Path="Application">
        *[System[(Level <= 3) and
        TimeCreated[timediff(@SystemTime) <= 86400000]]]
    </Select>
    <Suppress Path="Application">
        *[System[(Level = 2)]]
    </Suppress>
    <Select Path="System">
        *[System[(Level=1  or Level=2 or Level=3) and
        TimeCreated[timediff(@SystemTime) <= 86400000]]]
    </Select>
  </Query>
</QueryList>
```

Interessant ist hier aber vor allem das <Suppress>-Tag, dass Sie anstatt des <Select>-Tags einsetzen können. Alle Ereignisse, die dem Filter des Suppress-Tags entsprechen, werden explizit aus dem Suchergebnis ausgeschlossen. Dadurch können Sie Ihre XPath-Abfragen deutlich vereinfachen.

Das Eventlog mit Powershell abfragen

Statt der Ereignisanzeige können sie Powershell verwenden, um sich die Ereignisprotokolle anzeigen zu lassen. Powershell kennt zwei Cmdlets zum Abrufen der Ereignisse, *Get-Eventlog* und *Get-WinEvent*. Während *Get-Eventlog* einfacher zu handhaben ist, kann *Get-WinEvent* XML-Filter verwenden, um Ereignisprotokolle zu filtern. Außerdem kann *Get-Eventlog* nur die klassischen Ereignisprotokolle abfragen. Um sich anzeigen zu lassen, welche Logs *Get-Eventlog* anzeigen kann, rufen Sie *Get-Eventlog -List* auf.

```
PS > Get-Eventlog -List

  Max(K) Retain OverflowAction        Entries Log
  ------ ------ --------------        ------- ---
  20.480      0 OverwriteAsNeeded       2.740 Application
  20.480      0 OverwriteAsNeeded           0 HardwareEvents
     512      7 OverwriteOlder              0 Internet Explorer
  20.480      0 OverwriteAsNeeded           0 Key Management Service
  10.240     -1 DoNotOverwrite          5.604 Security
  20.480      0 OverwriteAsNeeded       2.480 System
  15.360      0 OverwriteAsNeeded         542 Windows PowerShell
```

Get-WinEvent verwendet stattdessen den Parameter -Listlog, der aber ein Argument benötigt. Verwenden Sie *, um sich alle Logs anzeigen zu lassen, oder geben Sie einen Filter ein, um nur Logs anzeigen zu lassen, deren Namen dem angegebenen Muster entsprechen.

```
PS > Get-WinEvent -ListLog *Powershell*

LogMode   MaximumSizeInBytes RecordCount LogName
-------   ------------------ ----------- -------
Circular            15728640         542 Windows PowerShell
Circular             1052672           0 Microsoft-Windows-PowerShell-...
Retain            1048985600           0 Microsoft-Windows-PowerShell/Admin
Circular            15728640         144 Microsoft-Windows-PowerShell/Operational
```

In beiden Fällen bekommen Sie nicht nur die Namen der verfügbaren Logs angezeigt, sondern auch deren Konfiguration, wie z.B. die maximale Größe oder die Anzahl der Einträge.

Da *Get-WinEvent* das deutlich mächtigere Kommando und *Get-Eventlog* auch sehr gut in der Hilfe beschrieben ist, werde ich mich hier nur auf *Get-WinEvent* konzentrieren.

Um sich alle Events des Security-Logs anzeigen zu lassen, verwenden Sie den Parameter -Logname und geben Sie das Protokoll an, dass Sie ausgeben wollen. Wenn Sie die Daten direkt am Bildschirm auswerten, leiten Sie die Ausgabe direkt in ein Gridview um.

```
Get-WinEvent -LogName Security | Out-GridView
```

Out-Gridview stellt Ihnen die Ausgabe von Get-WinEvent in einem durchsuchbaren Fenster dar.

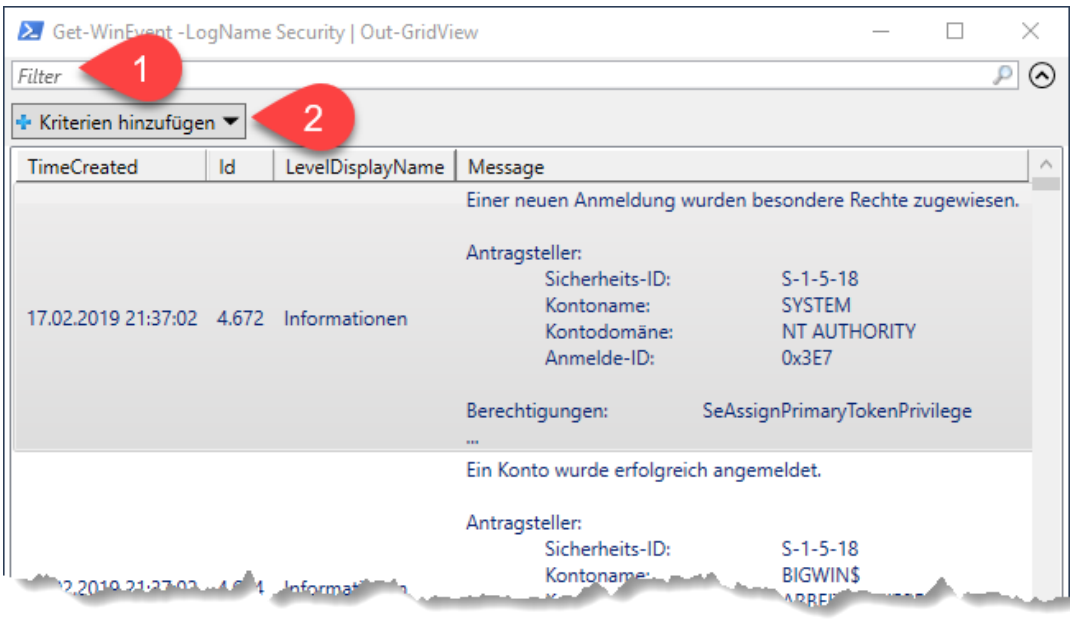


Bild 26 - Die Ausgabe läßt sich im Gridview besser auswerten

Das schöne am Gridview ist, dass Sie alle wichtigen Daten direkt im Fenster einsehen können. Anders als im Eventviewer ist hier die Nachricht jedes einzelnen Ereignisses direkt in einer eigenen Spalte angezeigt. Sie können das gesamte Gridview über das Filter-Feld (1) auch direkt durchsuchen. Geben Sie einen Suchbegriff ein, wird das Gridview sofort auf die Ereignisse eingeschränkt, in denen der Suchbegriff vorkommt. Dabei sucht der Filter standardmäßig über alle Spalten. Suchen Sie nach dem Wort "Error", werden also alle Spalten durchsucht. Alternativ können Sie aber auch den Spaltennamen, gefolgt von einem Doppelpunkt und dem Suchwort angeben, um die Suche auf eine Spalte einzuschränken. Wenn der Suchbegriff Leerzeichen enthält, müssen Sie ihn in Anführungszeichen angeben.

```
Message:"Ein Konto"
```

Sie können neben dem Volltextfilter aber auch noch einzelne Spaltenfilter hinzufügen. Dafür verwenden Sie den Schalter *Kriterien hinzufügen* (2).

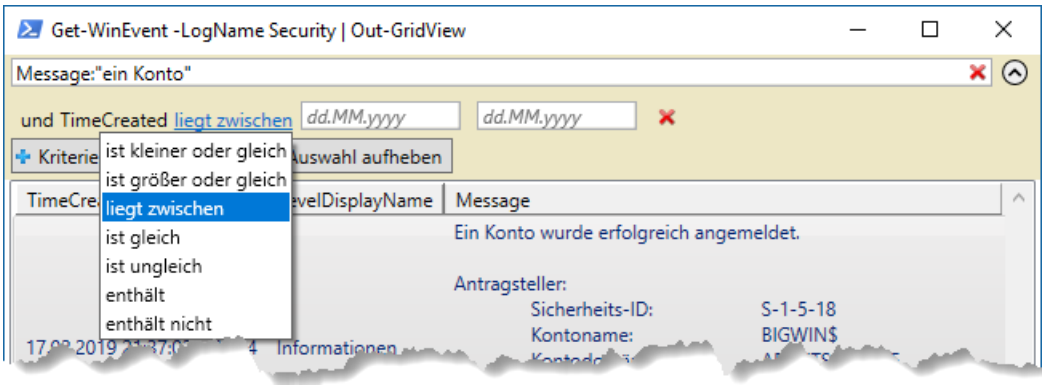


Bild 27 - Über Kriterien hinzufügen fügen Sie Spaltenfilter hinzu

Der Spaltenfilter ist abhängig vom Datentyp. Wenn Sie einen Filter für die Spalte TimeCreated hinzufügen, können Sie u.a. Datensätze suchen, die älter als ein eingegebenes Datum sind, neuer oder aber zwischen zwei Datumswerten liegen.

Sie können die Daten mit Get-WinEvent aber auch vorfiltern. Dafür stehen Ihnen drei verschiedene Parameter zur Verfügung.

Parameter	Beschreibung
Filterhashtable	Eine Hashtable ist eine Liste von Schlüssel-Wertepaaren, ähnlich einer ini-Datei. Sie können hier eine Liste von Werten übergeben, nach denen Sie filtern wollen. Mögliche Werte sind: LogName = Das zu durchsuchende Eventlog ProviderName = Der Anbieter (Provider) Path = Der Pfad - wenn evtx-Dateien durchsucht werden sollen Keywords= Schlüsselwort als in Int konvertiertes Bitmuster (s.o.) ID = Eine oder mehrere Event-IDs Level = ein Int, codiert den Typ (Information, Warnung, Fehler, Ausführlich) StartTime = Events ab dieser Zeit EndTime = bis zu dieser Zeit, muss nach der Starttime liegen UserID = eine Benutzer-SID oder ein Benutzername Data = Daten aus dem Datenfeld, gilt für klassische Eventlogs *= Der Platzhalter steht für ein Datenfeld, hinter dem Gleichheitszeichen folgt der Suchgebriff. Für die neuen Eventlogs
FilterXPath	Sie können direkt einen XPath-Filter als Suchkriterium angeben
FilterXml	Es können auch XML-Filter als Suchkriterium übergeben werden

FilterHashTable und FilterXml können nicht in Verbindung mit dem Parameter -Logname verwendet werden. Geben Sie das Log direkt an, können Sie also nur noch XPath-Filter verwenden. Sowohl beim Filterhashtable als auch bei XML-Filter geben Sie das oder die zu durchsuchenden Logs in der Filterabfrage an.

Ein Filterhashtable ist relativ einfach zu erstellen, weil Sie sich nicht mit XPath-Abfragen herumschlagen müssen. Ein Hashtable ist aufgebaut wie eine ini-Datei, mit einem Schlüsselnamen, der festgelegt ist und den Sie der obigen Tabelle entnehmen können, und einem Wert, den Sie als Suchkriterium übergeben können. Das besondere an einem Hashtable ist, dass die Schlüssel-Wertepaare von einer geschweiften Klammer mit einem führen @ umschlossen sind.

```
@{
   Logname="Security"
   Keywords="13510798882111488"
   StartTime='2019-02-01'
   EndTime='2019-02-15'
   UserID='S-1-5-21-3349445500-2401538872-2285400615-1103'
}
```

Sie haben zwei Möglichkeiten, die Filterhashtable zu übergeben – als Variable oder direkt als Argument des Parameters. Achten Sie darauf, dass Sie die einzelnen Wertepaare mit einem Semikolon trennen müssen, wenn sie nicht durch einen Zeilenumbruch getrennt sind.

```
$Filter = @{
   Logname="Security"
   Keywords="13510798882111488"
   StartTime='2019-02-01'
   EndTime='2019-02-15'
   UserID='S-1-5-21-3349445500-2401538872-2285400615-1103'
}
Get-WinEvent -FilterHashtable $filter

# und im Parameter, hier gekürzt auf 2 Datenfelder

Get-WinEvent -FilterHashtable @{ Logname="Security"; ID="4624" }
```

Um nach Daten im Nachrichtenfeld zu suchen, können Sie über Data direkt Daten angeben, die Sie suchen, oder Sie können das Datenfeld angeben, dass durchsucht werden soll. Die erste Methode funktioniert nur für klassische Protokolle, die zweite nur für die neuen Protokolle. Die Datenfelder können Sie sich am einfachsten über die Ereignisanzeige auflisten lassen, wenn Sie die *Angezeigt Ansicht* (im Original Friendly View) auswählen.

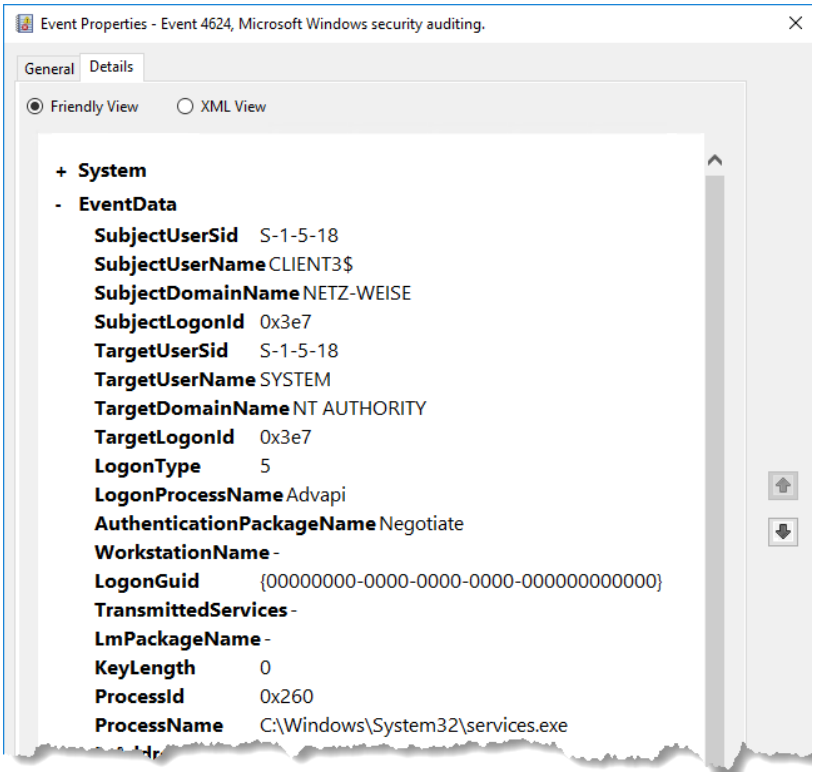


Bild 28 - die fettgedruckten Namen unter Eventdata entsprechen den Datenfeld-Namen

Wenn Sie im Sicherheits-Protokoll alle Ereignisse anzeigen lassen wollen, die den Computernamen Client3$ in irgend einem Datenfeld tragen, suchen Sie also über den Schlüssel Data, da es sich beim Sicherheits-Log um ein klassisches Protokoll handelt.

```
Get-WinEvent -FilterHashtable @{Logname="Security";Data="Client3$"}
```

Wenn Sie nur die Ereignisse filtern wollen, bei denen der Computername im Datenfeld *SubjectUserName* steht, müssen Sie einen XPath-Filter verwenden. Den können Sie *Get-WinEvent* auch direkt übergeben.

```
$XPathFilter = '*[EventData[Data[@Name="SubjectUserName"] and Data="Client3$"]]'
Get-WinEvent -LogName Security -FilterXPath $XPathFilter
```

Alternativ können Sie auch einen XML-Filter übergeben.

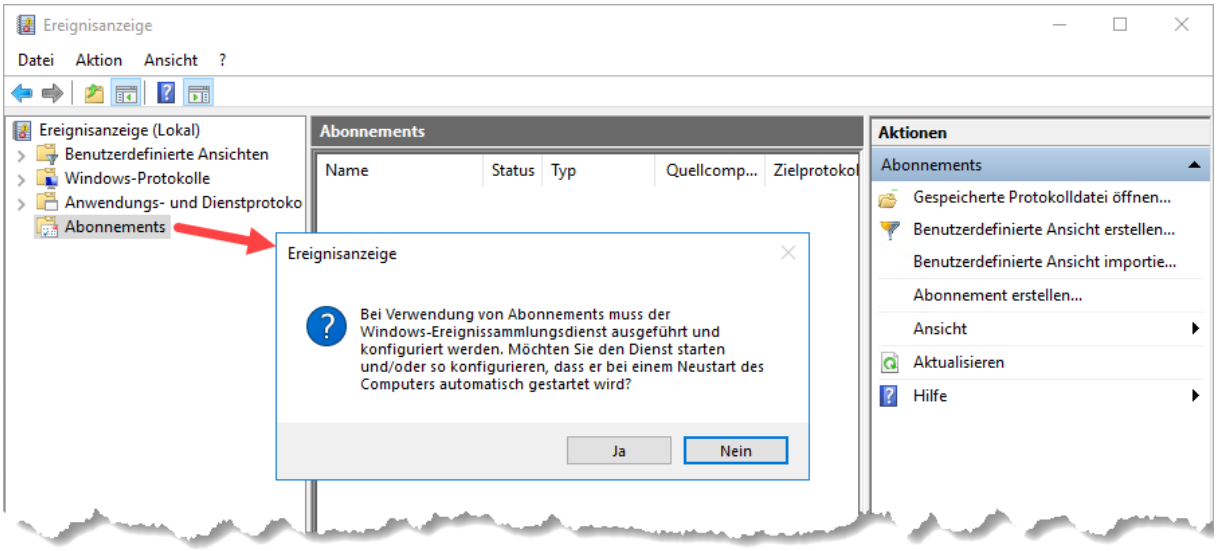
```
$XMLFilter = @'
<QueryList>
  <Query Id="0" Path="Security">
    <Select Path="Security">
      *[EventData[Data[@Name="SubjectUserName"] and Data="Client3$"]]
    </Select>
  </Query>
</QueryList>
'@
Get-WinEvent -FilterXml $XMLFilter
```

Das Ergebnis ist identisch, allerdings müssen Sie bei einem XML-Filter die zu durchsuchenden Logs immer im Filter angeben. Eine Besonderheit dieses Skripts muss noch erwähnt werden – das @' am Anfand und Ende des Filters. Es handelt sich hierbei um einen sogenannten Here-String. Technisch gesehen ist es einfach nur ein String (Text), allerdings bleibt seine Struktur komplett erhalten. Powershell versucht also nicht mehr, die Formatierung zu interpretieren. Der Here-String wird mit einem @' eingeleitet, direkt gefolgt von einem Zeilenumbruch. In der Zeile hinter dem @' darf kein Code mehr stehen! Das gleich gilt für das Ende des Here-String. Das '@ muss direkt am Anfang einer Zeile stehen.

Zwei weitere interessante Parameter zur Ausgabe von Ereignissen sind der Parameter -oldest, der die ältesten Events zuerst ausgibt, und -MaxEvents, der die Menge der zurückgegebenen Ereignisse einschränkt.

```
Get-WinEvent -LogName Security -MaxEvents 10 -Oldest
```

Ereignisprotokoll-Weiterleitung einrichten



Wichtige Ereignisse

Anmeldetyp	Beschreibung
2	Interactive Dieses Ereignis wird aufgezeichnet, wenn der Benutzer sich direkt an der Konsole, also am physikalischen Rechner anmeldet. Dieses Anmeldeereignis wird sowohl für lokale als auch Domänenbenutzer protokolliert, es sei denn, der Domänencontroller ist nicht erreichbar und die Anmeldung wir mit zwischengespeicherten Anmeldeinformationen durchgeführt. In diesem Fall ist der Anmeldetyp 11.
3	Network Auch beim Zugriff über das Netzwerk, z.B. beim Verbinden mit einer Freigabe, muss der Remote-Benutzer sich an dem Rechner anmelden, der die Ressourcen zur Verfügung stellt. Für die Remote-Anmeldung wird der Anmeldetyp 3 protokolliert.
4	Batch / Batch Der Anmeldetyp 4 wird für geplante Aufgaben verwendet. Wenn eine geplante Aufgabe im Kontext des angemeldeten Benutzers gestartet wird, kommt es zu keiner erneuten Anmeldung, und es wird auch kein Anmeldeereignis protokolliert.
5	Service / Dienst
7	Entsperren / Unlock
8	NetworkCleartext
9	NewCredentials
10	RemoteInteractive
11	CachedInteractive

Powershell-Logging

Powershell ist ein großartiges Automatisierungswerkzeug. Mit Powershell Remoting, dass auf WinRM (Windows Remote Management) aufsetzt, kann man Befehle und ganze Skripte sogar auf einem oder mehren anderen Computern ausführen lassen. Durch seine Fähigkeiten ist Powershell aber nicht nur bei Administratoren beliebt, sondern auch bei Angreifern.

Hier gleich ein Wort der Warnung – nur weil sich mit Powershell bösartige Skripte ausführen lassen, ist Powershell selbst keine Schwachstelle. Powershell ist beliebt, weil es ein auf Windows-Systemen allseits verfügbares, einfach einzusetzendes Werkzeug ist. Die Lücken, durch die Angreifer auf das System zugreifen können, sind aber nicht Bestandteil von Powershell. Man kann Sie auch mit jeder anderen Sprache ausnutzen. Powershell zu deaktivieren ist also keine sicherheitsfördernde Maßnahme, sondern im Gegenteil hat Microsoft sehr viel Aufwand betrieben, Powershell mit jeder Version sicherer zu machen. Standardmäßig verwendet Powershell Remoting zwischen Systemen innerhalb einer Domäne Kerberos für die Authentifizierung und AES256, um den gesamten Datenverkehr zu verschlüsseln. Außerdem hat Microsoft seit Powershell 3.0 angefangen, Protokoll-Funktionen in Powershell zu integrieren. Ab der Version 5 (Standard auf allen Windows 10 und Windows Server 2016-Sytemen) stehen zwei relevante zur Verfügung:

- Over the Shoulder Transcription
- Deep Script Block logging

Achten Sie darauf, dass Sie unter Windows 8.1 und Windows Server 2012 R2 neben Powershell 5.0 noch den Patch 3000850 installiert haben müssen.

Over the Shoulder Transcription

Over the Shoulder Transcription erstellt für jede Powershell-Sitzung ein Transkript – eine Klartext-Logdatei -, das alle ausgeführten Befehle und Ausgaben protokolliert. Die Transkriptions-Fähigkeiten beherrscht Powershell schon lange, aber bis Version 5 kann nur die Ausgabe der Standard-Ausgabe mitgeschnitten werden, so dass viele Informationen nicht ins Transkript aufgenommen werden. Mit Powershell 5 werden nicht nur die aufgerufenen Befehle mitgeschnitten, sondern auch Metainformationen über den Aufruf.

```
**********************
nStart der Windows PowerShell-Aufzeichnung
Startzeit: 20190110183559
Benutzername: ADMINWS\Admin
RunAs-Benutzer: ADMINWS\Admin
Konfigurationsname:
Computer: ADMINWS (Microsoft Windows NT 10.0.17134.0)
Hostanwendung: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Prozess-ID: 6228
PSVersion: 5.1.17134.407
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.17134.407
BuildVersion: 10.0.17134.407
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
PS C:\Windows\system32> Get-ADDomain
PS C:\Windows\system32> TerminatingError(Get-ADDomain): "Unter "DC=netz-
weise,DC=de" kann kein Objekt mit der ID "ADMINWS" gefunden werden."
```

Im Header des Transkripts finden Sie die Startzeit, den Benutzer, der die Sitzung gestartet hat, und das Benutzerkonto, unter dem das Skript ausgeführt wurde. Es wird der Powershell-Host angezeigt, die Prozess-ID und noch eine Reihe von weiteren Informationen. Danach folgen alle Ein- und Ausgaben, die auch der Benutzer sieht, sowie zusätzliche Prozessinformationen.

Over the Shoulder Transcription heißt so, weil man dem Anwender bei Ausführen quasi über die Schulter schauen kann. Sie können es als Gruppenrichtlinie sowohl für Benutzer als auch für Computer aktivieren, und zwar in den administrativen Einstellungen unter *Windows-Komponenten – Powershell*. Hier finden Sie sowohl die Transkription unter dem Namen *Powershell-Aufzeichnung aktivieren* als auch das Skriptblock-Logging unter *Protokollierung von PowerShell-Skriptblöcken aktivieren*.

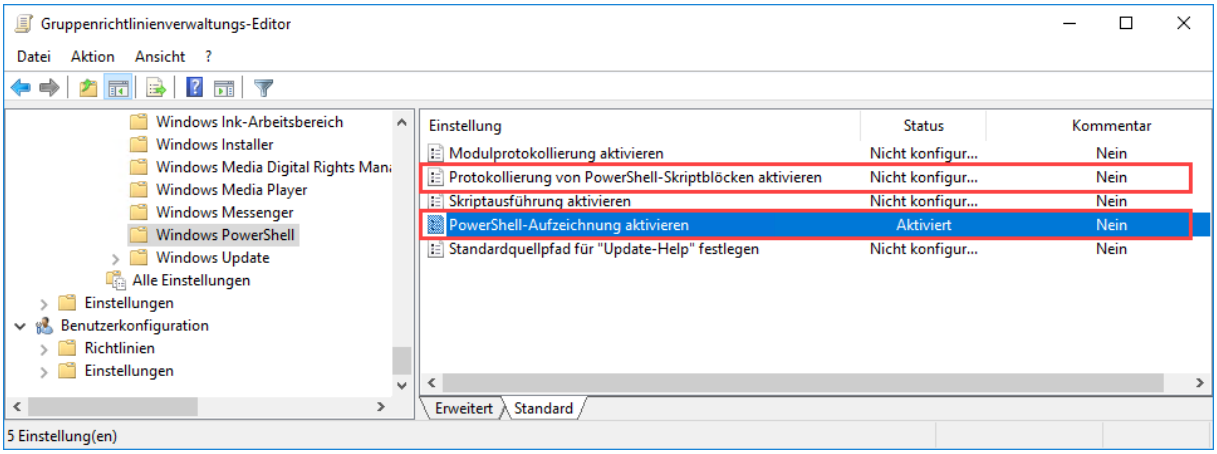


Bild 29 - Sie können die Protokollierung über Gruppenrichtlinien aktivieren

Aktivieren Sie die Powershell-Aufzeichnung, legt Windows die Sitzungsprotokolle standardmäßig im Dokumente-Ordner des Benutzers ab. Die Protokolle werden jeden Tag in einem neuen Ordner abgelegt, der nach dem Schema *JahrMonatTag* erstellt wird, also z.B. *20190110* für den 10.1.2019. Innerhalb des Ordners wird pro Powershell-Sitzung ein Protokoll mit dem Namen *Powershell_transcript.<Computername>.JahrMonatTagStundeMinuteSekunde.<8stelliger Zufallsstring>.txt* erzeugt.

Der Zielordner kann über die Gruppenrichtlinie angepasst werden. Geben Sie dafür den Pfad in das Feld *Verzeichnis der Aufzeichnungsausgabe* an.

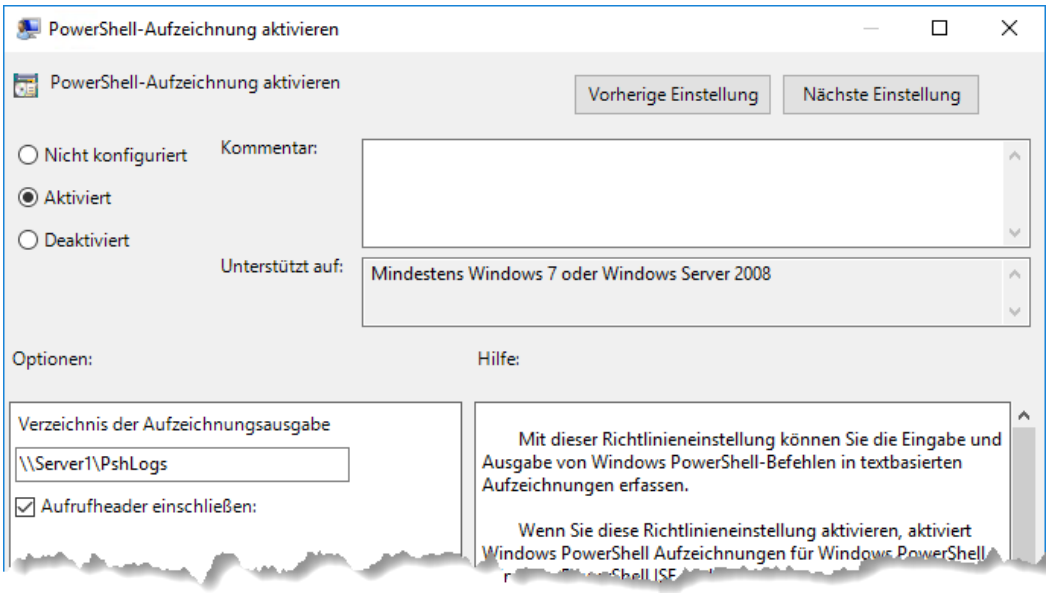


Bild 30 - Der Zielpfad für das Protokoll kann angepasst werden

Sie können auch einen Netzwerkpfad angeben, aber leider keine Umgebungsvariablen verwenden. Die Eingabe von *%ProgramData%\PshLogs* erstellt stattdessen einen neuen Ordner %Programdata% im Dokumente-Ordner des Benutzers.

Wenn Sie die Checkbox *Aufrufheader einschließen* aktivieren, wird im Transkript beim Aufruf eines Kommandos jeweils die Startzeit protokolliert.

```
**********************
Command start time: 20190110222809
**********************
PS C:\Users\admin> get-command -Module activedirectory
**********************
Command start time: 20190110222814
**********************
PS C:\Users\admin> Get-NetIPAddress
```

Bei jedem Kommando-Aufruf wird die Startzeit angegeben.

Zumindest in Windows 10 1803 ist mir ein Bug aufgefallen, der den Transkript-Header zweimal ausgegeben hat. Unter Windows Server 2019 und Windows 10 2019 trat das Phänomen nicht auf.

Aus verschiedenen Gründen ist die Transkription nicht unbedingt empfehlenswert. Damit Sie das Logging für die Überwachung verwenden können, müssen Sie die Protokolle zentral ablegen. Die Logs werden dann im Benutzerkontext in die Freigabe geschrieben, so dass alle Benutzer Schreibrechte für die Freigabe brauchen. Da der Log-Prozess die Datei offen hält, benötigt der Benutzer auch Leserechte. Das bedeutet, dass die Logs der Freigabe für alle Benutzer offen sichtbar sind, inklusive aller eventuell empfindlichen Daten. Noch schlimmer aber wiegt, dass ein böswilliger Benutzer die Logs jederzeit einfach löschen kann, wenn er Schreibrechte auf das Log hat. Für ein Sicherheitsüberwachung also kaum einsetzbar. Zu guter Letzt generiert der Prozess auch noch eine hohe Last auf dem System, wenn viele Protokolle geschrieben werden müssen.

Scriptblock-Logging

Im Gegensatz zur Transkription werden beim Scriptblock-Logging die ausgeführten Befehle in die Ereignisanzeige unter *Anwendungs- und Dienstprotokolle – Microsoft – Windows – Powershell - Operational* geschrieben. Das hat die bereits oben erwähnten Vorteile – Sie können den Zugriff auf das Protokoll besser schützen, die Protokollierung ist performant und ressourcenschonend und kann

per Weiterleitung zentral gesammelt werden. Außerdem kann das Protokoll nicht gelöscht werden, ohne dass zumindest das Löschen protokolliert wird, und der Code wird kompakter protokolliert. Powershell speichert dabei den ausgeführten Code, und nicht die Kommandozeile. Das ist wichtig, weil ein beliebtes Mittel von Angreifern ist, Code Base64-codiert aufzurufen, was die Erkennung des tatsächlich ausgeführten Befehls sehr schwer macht.

Das Deep Skriptblock-Logging hat Microsoft mit Powershell 5 eingeführt. Es ist nach der Aktualisierung auf Powershell 5.1 auch in Windows 7 / Server 2008 R2 und später verfügbar. Ist Skriptblock-Logging aktiviert, werden alle ausgeführten Powershell-Befehle im Ereignisprotokoll gespeichert.

Der Name weist bereits darauf hin, dass die Protokollierung Skriptblockweise stattfindet. Ein Skriptblock ist in Powershell gemeinsam ausgeführter Code, der üblicherweise in geschweiften Klammern zusammengefasst ist. Powershell-Skripte sind Skriptblöcke, aber auch ein einzeln ausgeführtes Powershell-Kommando (Funktion oder Cmdlet) wird als Skriptblock gespeichert.

Wird ein Skriptblock zum ersten Mal ausgeführt, wird der gesamte Code des Skriptblocks protokolliert. Dem Block wird eine GUID (eine eindeutige ID) zugewiesen, die im Ereignis gespeichert wird. Jede weitere Ausführung des Skriptblocks wird nicht mehr ins Ereignisprotokoll aufgenommen. Sie können aber festlegen, bei jedem wiederholten Starten und Beenden ein weiteres Ereignis angelegt wird, das aber nur die GUID referenziert, ohne den Code erneut zu speichern. Dadurch wird die Protokollierung quasi dedupliziert. Grundsätzlich ist davon aber abzuraten, da die Menge der protokollierten Daten sich vervielfachen kann. Es geht beim Skriptblocklogging ja nicht darum, zu überprüfen, was ein Benutzer wann gemacht hat, sondern nach einem Angriff herauszufinden, welchen Code der Angreifer ausgeführt hat. Dafür reicht es aus, den Code einmal zu sichern.

Ist der Code zu lang, um ihn in einem Ereignis zu speichern, wird der Skriptblock auf mehrere Ereignisse aufgeteilt (s. Bild 31). Um den Code wieder zusammenführen zu können, wird in jede Ereignisbeschreibung eingetragen, auf wie viele Ereignisse der Skriptblock aufgeteilt wurde und welchen Zähler das angezeigt Event hat (2). Außerdem bekommen alle zum Skriptblock gehörenden Events die gleiche ActivityID (1), eine weitere eindeutige GUID, durch die zusammengehörige Events identifiziert werden können.

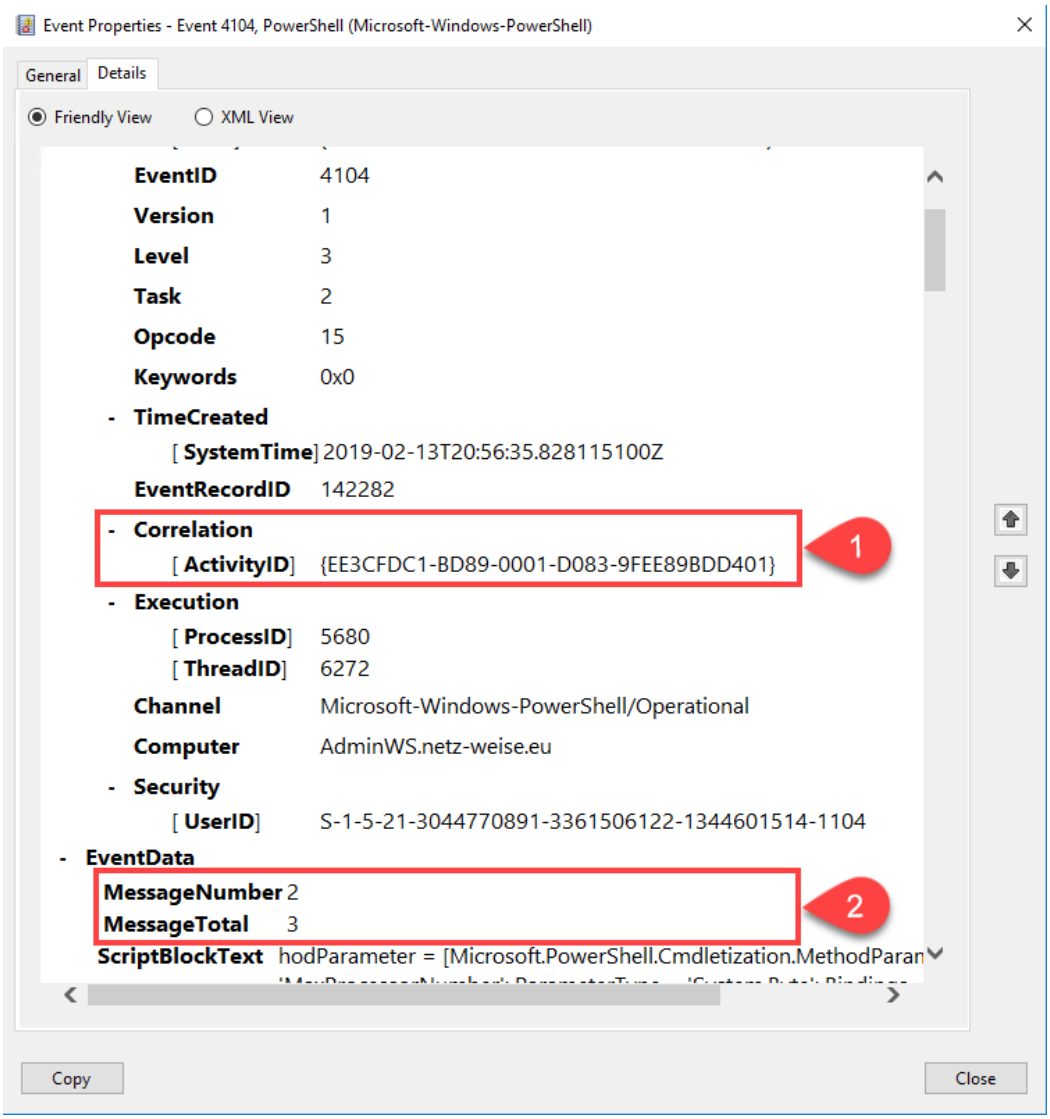


Bild 31 - Der zweite Teil eines auf mehrere Ereignisse verteilen Skriptblocks

Im Powershell Operational Log landen nicht nur die aufgezeichneten Skriptblöcke, sondern auch z.B. das Öffnen einer Powershell-Konsole oder das Starten eines Skriptes aus der ISE. Alle protokollierten Skriptblöcke erhalten daher zur eindeutigen Identifizierung die EventID 4104. Dabei werden zwei unterschiedliche Informationsebenen verwendet – Ausführlich und Warnung. Warnungen sind Skriptblöcke, die potentiell schädlichen Code enthalten. Das heißt aber nicht, dass Code, der nicht als Warnung protokolliert ist, unbedenklich ist. Welche Kriterien Powershell anwendet, um gefährlichen Code zu erkennen, ist nicht dokumentiert.

Scriptblöcke, die die Kriterien für mutmaßlich gefährlichen Code erfüllen, werden automatisch protokolliert, auch wenn Sie das Skriptblocklogging nicht aktiviert haben. Sie können das einfach ausprobieren, indem Sie *Get-NetIPAddress* eingeben. *Get-NetIPAddress* ist eine Funktion, die im Hintergrund weiteren Code aufruft. Sie bekommen jede Menge neue Protokolleinträge als Warnung angezeigt, unabhängig davon, ob Sie die Protokollierung aktiviert haben oder nicht (s. Bild 32).

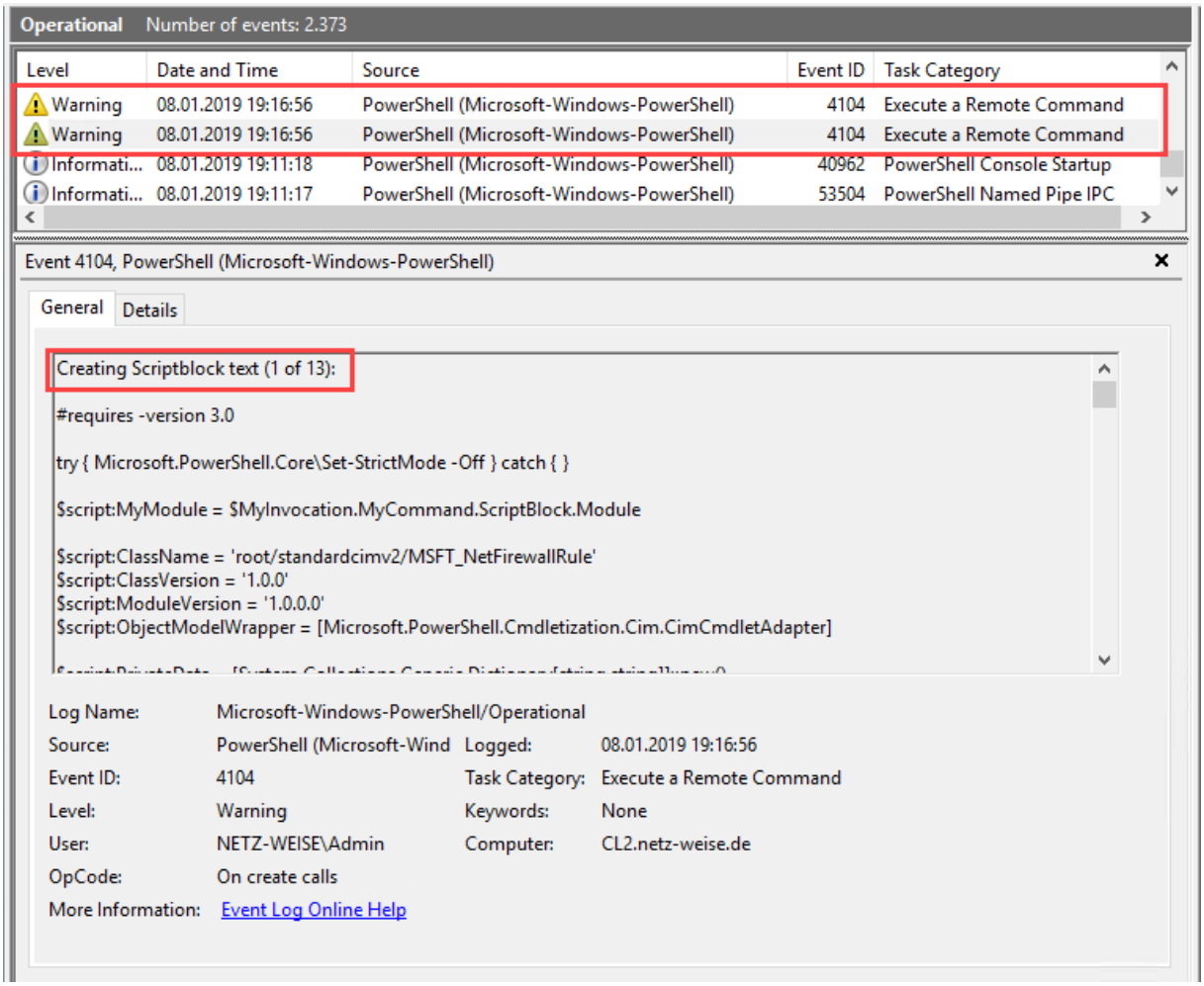


Bild 32 - potentiell gefährlicher Code wird als Warning gespeichert und auch ohne aktivieres Logging protokolliert

Skriptblock-Logging aktivieren

Skriptblock-Logging wird über die Systemregistrierung aktiviert. Erstellen Sie hierfür folgenden Wert:

Schlüssel: HKLM\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging\
WertName: EnableScriptBlockLogging
Typ: Dword (Int32)
Wert: 1(aktiv), 0(inaktiv)

Wenn Sie den Schlüssel auf 1 setzen, aktivieren Sie das Logging aller Skriptblöcke. Wenn Sie den Wert auf 0 setzen, deaktivieren Sie das Skriptblocklogging vollständig. Vollständig bedeutet, dass auch keine Warnungen mehr protokolliert werden. Tatsächlich hat das Skriptblocklogging also drei Zustände: nichts protokollieren (0), alles protokollieren (1) oder den Standard, nur Warnungen protokollieren (undefiniert).

Wie Sie bereits am Registry-Pfad (HKLM\Software**Policies**) sehen können, handelt es sich um eine Richtlinie, die auch zentral über die Domäne gesetzt werden kann. Öffnen Sie hierfür ein Gruppenrichtlinienobjekt. Sie finden die Gruppenrichtlinie sowohl in den Benutzer wie auch in den Computereinstellungen in den administrativen Einstellungen unter *Windows-Komponenten – Powershell* (s. Bild 29) unter dem Namen *Protokollierung von Powershell-Skriptblöcken aktivieren*.

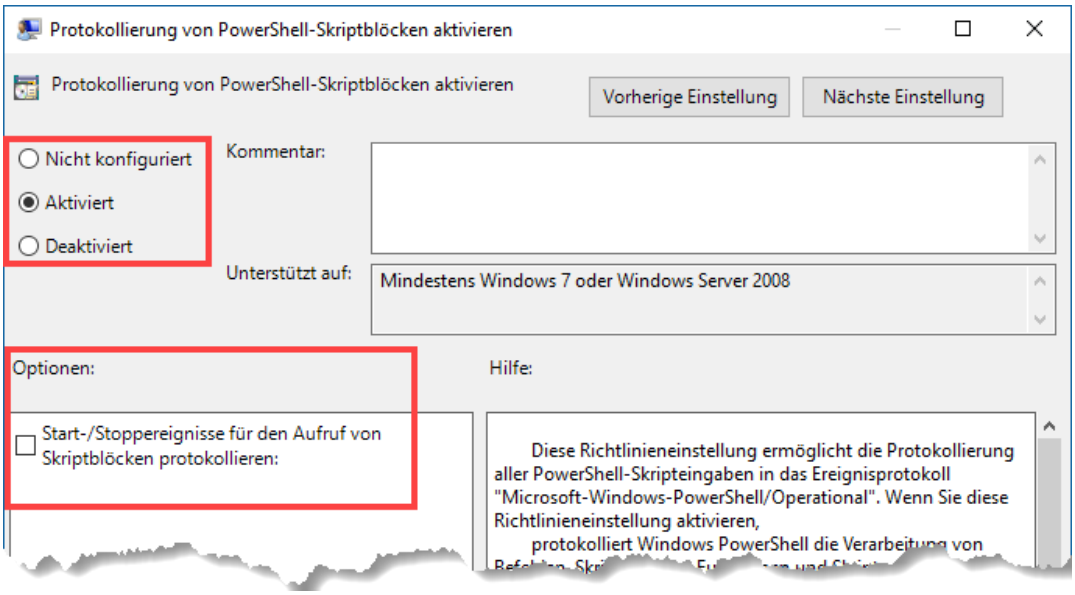


Bild 33 - Aktivieren des Skriptblocklogging per Gruppenrichtlinie

Achten Sie darauf, dass das Deaktivieren der Richtlinie das Skriptblocklogging vollständig deaktiviert. N*icht konfiguriert* ist die Standardkonfiguration und bedeutet, dass nur potentiell gefährliche Skriptblöcke als Warnungen geschrieben werden. Zusätzlich können Sie über die Checkbox "Start-/Stoppereignisse für den Aufruf von Skriptblöcken protokollieren" festlegen, dass ein Skriptblock bei jedem Start minimal geloggt wird. Für das Starten eines bereits protokollierten Blocks wird die EventID 4105 verwenden, für das Beenden die EventID 4106. Obwohl die Start- und Stoppevents sehr kompakt sind, sollte diese Einstellung vermieden werden, da das Ereignisprotokoll sehr schnell mit sehr vielen Ereignissen geflutet werden kann.

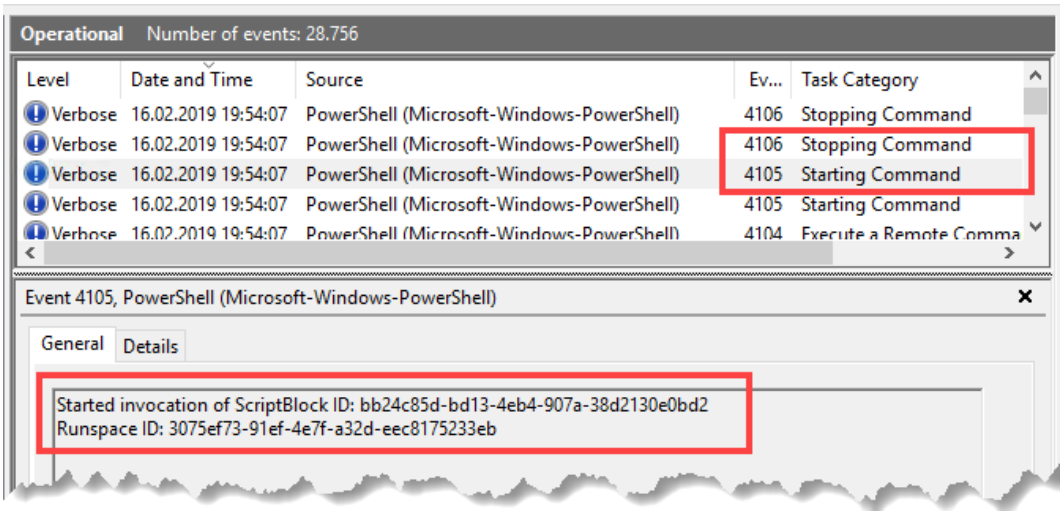


Bild 34 - Start-und Stopereignisse sind kompakt, können dafür aber sehr oft auftreten

Sie können den Status des Skriptblocklogging auch mit Powershell direkt in der Registry prüfen und ändern. Führen Sie dazu folgendes Kommando aus.

```
$Key = HKLM:\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging\'
$Value = 'EnableScriptBlockLogging'
Try
{
    $Logging = Get-ItemPropertyValue -Path $Key -Name $Value -ErrorAction Stop
    If ( $Logging )
    {
```

```
        'SkriptblockLogging ist aktiviert'
    }
    Else
    {
        'SkriptblockLogging ist nicht aktiviert'
    }
}
Catch
{
    "SkriptblockLogging ist nicht definiert"
}
```

Das geht auch noch deutlich schöner, aber zum Glück hat uns Dr. Tobias Weltner mit dem Model ScriptBlockLoggingAnalyzer die Arbeit bereits abgenommen. Sie können es aus der Powershell Gallery herunterladen. Geben Sie hierfür auf einem mit dem Internet verbunden Computer den folgenden Befehl in die Powershell ein.

```
Install-Module -Name ScriptBlockLoggingAnalyzer -Scope CurrentUser
```

Wenn Sie *Install-Module* zum ersten Mal ausführen, muss zuerst NuGet installiert werden. Das ist ungefährlich und kann mit [Y]es bestätigt werden. Anschließend weist Sie Powershell darauf hin, dass das Modul aus einem nicht-vertrauenswürdigen Repository (der Powershell Gallery) stammt. Bestätigen Sie ein weiteres mal mit [Y]es. Das Modul finden Sie anschließend in Ihrem Dokumente-Ordner unter *\WindowsPowershell\Modules\SkriptBlockLoggingAnaylzer*. Von hier aus können Sie es auch auf andere Computer kopieren.

Der ScriptBlockLoggingAnalyzer bringt eine Reihe von Cmdlets zur einfacheren Verwaltung und Auswertung der Eventlogs mit. Mit dem Cmdlet *Get-SBLStatus* können Sie prüfen, ob Skriptblock-Logging aktiviert ist, mit *Enable-SBL* schalten Sie es ein und mit *Disable-SBL* deaktivieren Sie es. Die Cmdlet tun dabei auch nichts anderes, als das, was obenstehender Skriptcode auch tut – die Registry abfragen oder ändern.

```
PS > Get-SBLStatus
EnableScriptBlockLogging EnableScriptBlockInvocationLogging SettingsKeyExists SBLActive
------------------------ ---------------------------------- ----------------- ----------
                   False                              False             False       True
PS > Enable-SBL
```

Das Eventlog auswerten

Das Modul hat aber noch eine Reihe von weiteren nützlichen Cmdlets. Sie können sich alle Cmdlets eines Moduls anzeigen lassen, indem Sie *Get-Command* -mit dem Parameter *-Module* aufrufen.

```
PS > get-command -Module ScriptBlockLoggingAnalyzer

CommandType Name                Version Source
----------- ----                ------- ------
Function    Clear-SBLLog        1.2     scriptblockloggінganalyzer
Function    Convert-SIDToUser   1.2     scriptblockloggінganalyzer
Function    Disable-SBL         1.2     scriptblockloggінganalyzer
Function    Enable-SBL          1.2     scriptblockloggінganalyzer
Function    Get-SBLCodeStatistic 1.2    scriptblockloggінganalyzer
Function    Get-SBLEvent        1.2     scriptblockloggінganalyzer
Function    Get-SBLLogSize      1.2     scriptblockloggінganalyzer
Function    Get-SBLStatus       1.2     scriptblockloggінganalyzer
Function    Set-SBLLogSize      1.2     scriptblockloggінganalyzer
```

Um sich alle ausgeführen Skriptblöcke anzeigen zu lassen, benutzen Sie *Get-SBLEvent*. Das Cmdlet hat keine weiteren Parameter. Es benutzt das Cmdlet *Get-Winevent*, das weiter oben bereits

beschrieben wurde, um die ausgeführten Skriptblöcke wieder in einen Eintrag zusammenzuführen und die wesentlichen Metadaten gleich mit aufzulösen.

```
PS> Get-SBLEvent | Sort-Object -Property TimeCreated

TimeCreated  : 14.02.2019 17:31:28
Name         : [from memory]
Code         : ipconfig
Path         : [from memory]
UserName     : NETZ-WEISE\admin
ComputerName : Client3.netz-weise.de
ProcessId    : 4768
ThreadId     : 5544
Sid          : S-1-5-21-3349445500-2401538872-2285400615-1103
TotalParts   : 1
CodeId       : 0d7d9077-8a8e-4cfd-b3f5-0d29e387c70c

TimeCreated  : 14.02.2019 17:31:28
Name         : [from memory]
Code         : prompt
Path         : [from memory]
UserName     : NETZ-WEISE\admin
ComputerName : Client3.netz-weise.de
ProcessId    : 4768
ThreadId     : 5544
Sid          : S-1-5-21-3349445500-2401538872-2285400615-1103
TotalParts   : 1
CodeId       : f9ec7d8b-b56f-4ee7-b310-56af9b5b5b78
```

Das Cmdlet gibt Ihnen die Zeit zurück, zu der das Skript ausgeführt wurde, in der Eigenschaft *Code* die ausgeführten Befehle, und in der Eigenschaft *Totalparts* die Anzahl Events, in die der Skriptblock aufgeteilt wurde. *Path* gibt den Pfad zum ausgeführten Skript an, soweit der Code als Skript gestartet wurde. Kommandos, die als Cmdlet oder Funktion direkt in Powershell direkt zur Verfügung stehen, werden hier als *[from memory]* bezeichnet. Der Benutzer, in dessen Kontext der Code gestartet wurde, wird in der Eigenschaft *UserName* angegeben.

Wenn Sie die Events selbst auswerten wollen, ist es interessant, sich XML-Darstellung des Ipconfig-Logs noch einmal anzusehen.

```
<System>
 <Provider Name="Microsoft-Windows-PowerShell" Guid="{A0C1853B-5C40-4B15-8766-3CF1C58F985A}" />
 <EventID>4104</EventID>
 <Version>1</Version>
 <Level>5</Level>
 <Task>2</Task>
 <Opcode>15</Opcode>
 <Keywords>0x0</Keywords>
 <TimeCreated SystemTime="2019-02-14T16:31:28.347255500Z" />
 <EventRecordID>2027</EventRecordID>
 <Correlation ActivityID="{C29D1547-C47D-0001-9062-9DC27DC4D401}" />
 <Execution ProcessID="4768" ThreadID="5544" />
 <Channel>Microsoft-Windows-PowerShell/Operational</Channel>
 <Computer>Client3.netz-weise.de</Computer>
 <Security UserID="S-1-5-21-3349445500-2401538872-2285400615-1103" />
 </System>
<EventData>
 <Data Name="MessageNumber">1</Data>
 <Data Name="MessageTotal">1</Data>
 <Data Name="ScriptBlockText">ipconfig</Data>
 <Data Name="ScriptBlockId">0d7d9077-8a8e-4cfd-b3f5-0d29e387c70c</Data>
 <Data Name="Path" />
</EventData>
```

Sie können hier sehen, dass der Pfad unter *<Eventdata>* im Datenfeld mit dem Attributnamen *Path* gespeichert ist. Wird Code direkt aus dem Speicher gestartet – und das betrifft auch Funktionen –, bleibt der Pfad leer. Nur wenn ein Skript direkt aus dem Dateisystem gestartet wurde, wird er Pfad angegeben.

```
<EventData>
  <Data Name="MessageNumber">1</Data>
  <Data Name="MessageTotal">1</Data>
  <Data Name="ScriptBlockText">Write-Output -InputObject "Ich bin ein böser Schadcode"</Data>
  <Data Name="ScriptBlockId">532168fe-042d-4037-a368-9b361fe586ee</Data>
  <Data Name="Path">C:\Temp\get-evil.ps1</Data>
</EventData>
```

Sie finden im Ereignis nicht den Namen des ausführenden Benutzers, sondern nur dessen Security-ID (SID). Das Modul ScriptBlockLoggingAnalyzer bringt ein eigenes Cmdlet mit, um die SID in einen Namen aufzulösen. Es heißt *Convert-SidToUser*. Um allen Events vor der Ausgabe einen Benutzernamen hinzuzufügen, können Sie folgendes Powershell-Codefragment benutzen.

```
Get-WinEvent -LogName Microsoft-Windows-PowerShell/Operational |
 Foreach-Object {
   Add-Member -InputObject $_ -MemberType NoteProperty -Name UserName -Value (
     Convert-SIDToUser -SID $_.UserID )
   $_
 }
```

Der Code liest das Powershell Operational Log aus und fügt jedem Ereignis eine neue Eigenschaft Username hinzu. Der Wert der Eigenschaft wird mit der Funktion Convert-SIDToUser direkt aus der SID berechnet.

Konfigurieren des Protokolls

Wenn Sie das Skriptblocklogging aus Sicherheitsgründen aktivieren, sollten Sie sich darüber im Klaren sein, dass alle in Powershell eingegebenen Befehle protokolliert werden. Jeder ausgeführte Skriptcode ist in Protokoll einsehbar. Das betrifft auch Kennwörter, die in Skripten gespeichert sind, oder sonstige vertrauliche Informationen. Dummerweise ist das Powershell Operational Ereignisprotokoll anders als das Sicherheits-Protokoll auch für normale Benutzer zugänglich. Prinzipiell kann also jeder Benutzer auf allen Systemen, auf denen er Zugriffsrechte hat, sämtliche ausgeführten Befehle einsehen. Wenn Sie die Ereignisprotokoll-Weiterleitung verwenden, gehen die Protokolle außerdem auch ungeschützt über das Netzwerk.

Sie können zwei Sicherheitsmaßnahmen treffen, um die Protokolle besser zu schützen. Zum einen sollten Sie Benutzern den Zugriff auf das Powershell Operational Log verweigern. Am einfachsten verwenden Sie die gleichen Berechtigungen, die den Zugriff auf das Sicherheits-Ereignisprotokoll sichern. Sie können sich die Berechtigungen in Form des Security-Descriptors mit *Get-Winevent* ganz einfach ausgeben lassen und mit *wevtutil.exe set-log* setzen.

```
$PoshLog = 'Microsoft-Windows-PowerShell/Operational'
(Get-WinEvent -ListLog $PoshLog ).SecurityDescriptor | out-file C:\temp\Backup.log
$SecurityLogPerms = (Get-WinEvent -ListLog Security).SecurityDescriptor
.\wevtutil.exe sl $PoshLog /ca:$SecurityLogPerms
(Get-WinEvent -ListLog $PoshLog ).SecurityDescriptor
```

Die zweite Codezeile sichert die Original-Berechtigungen in der Datei c:\temp\Backup.log, ist also für die Funktion des Skripts nicht wirklich notwendig. Die dritte Zeile liest die Berechtigungen des Sicherheitsprotokolls aus, die dann in der vierten Zeile mit wevtutil.exe auf dem Powershell-Operational Protokoll gesetzt werden. Prinzipiell können Sie auch beliebige Benutzer und Gruppen

auf das Eventlog berechtigen, allerdings müssen Sie dann einen entsprechenden Security-Descriptor von Hand erstellen.

Weiterhin kann Powershell beim Protokollieren der Skriptblöcke sämtliche Einträge verschlüsseln. Die Verschlüsselung findet dabei mit Public Private Key-Verschlüsselung nach dem CMS-Standard (Cryptographic Message Syntax) statt. Für die Verschlüsselung benötigen Sie einen öffentlichen Schlüssel in Form eines Zertifikats und einen privaten Schlüssel auf dem Rechner, auf dem die Entschlüsselung stattfindet. Der private Schlüssel sollte nicht im Netzwerk verteilt werden. Daher macht die Verschlüsselung nur dann Sinn, wenn man die Ereignisprotokolle auch z.B. per Ereignisprotokollweiterleitung zentral sammelt.

Die Konfiguration ist relativ simpel. Das Zertifikat für die Verschlüsselung wird am Besten über eine Gruppenrichtlinie verteilt, und da Powershell den Herausgeber des Zertifikats nicht prüft, können Sie ein selbstsigniertes Zertifikat verwenden. Das können Sie sich (auf einem administrativen Rechner) selbst erstellen. Achten Sie darauf, dass der private Schlüssel zusammen mit dem Zertifikat generiert wird und auf dem Computer abgelegt ist, auf dem Sie das Zertifikat erstellt haben. Der private Schlüssel darf niemals zusammen mit dem Zertifikat verteilt werden und sollte sicher aufbewahrt werden.

Das Zertifikat können Sie sich mit dem Powershell-Cmdlet *New-SelfSignedCertificate* erzeugen. An das Zertifikat werden folgende Anforderungen gestellt:

- Das Zertifikat muss als erweiterte Schlüsselverwendung Dokumentenverschlüsselung aktiviert haben
- Es müssen entweder Datenverschlüsselung oder Schlüssel-Verschlüsselung aktiviert sein
- Zusätzlich sollten Sie das Zertifikat mit einem sprechenden Namen versehen.

Danach müssen Sie das Zertifikat Base64-kodiert (Binärdaten als Text) speichern. Das geht direkt über die Convert-Klasse des .net-Framework. Den zurückgegebenen Code können Sie in eine Datei schreiben oder ins Clipboard kopieren. Das Cmdlet Tee-Object macht das in einem Rutsch – es speichert den Code in einer Textdatei und gibt ihn gleichzeitig in die Pipeline weiter, von wo er ins Clipboard kopiert wird.

```
$Cert = New-SelfSignedCertificate -Subject "ScriptBlockVerschlüsselung" `
  -KeyUsage DataEncipherment -Type DocumentEncryptionCert
$CertBase64 = ( [convert]::toBase64String($cert.RawData,'InsertLineBreaks' ))
"-----BEGIN CERTIFICATE-----`n" + $CertBase64 + "`n-----END CERTIFICATE-----" |
    Tee-Object -FilePath c:\temp\EncryptionCert.txt | clip.exe
```

Anschließend aktivieren Sie die Protokollverschlüsselung über die Gruppenrichtlinie *Geschütze Ereignisprotokollierung*, die Sie in der Computerkonfiguration unter *Richtlinien – Adminsitrative Vorlagen – Windows Kompontenen – Ereignisprotokollierung* finden. Wenn Sie die Richtlinie aktivieren, können Sie im Optionsfeld das zu verwendenden Zertifikat hinterlegen (s. Bild 35).

Grundsätzlich haben Sie mehrere Möglichkeiten, auf das Zertifikat zu referenzieren. Statt des Base64-Codes können Sie das Zertifikat auch in einer .cer-Datei speichern und auf den Pfad der Datei referenzieren. Diese muss dann im Dateisystem aber immer verfügbar sein. Alternativ können Sie auch den Thumbprint (eine eindeutige Zertifikats-ID, genau genommen ein Hash) des Zertifikats angeben. Das Zertifikat muss sich dann aber auf allen Zielrechnern im Zertifikatsspeicher befinden. Wenn Sie den Base64-Code in der Richtlinie hinterlegen, verteilen Sie das Zertifikat mit der Richtlinie, und Sie müssen sich nicht mehr darum kümmern, ob das Zertifikat vom Zielrechner aus erreichbar ist. Achten Sie beim Einfügen des Zertifikats in das Optionsfeld darauf, dass keine Umbrüche oder Leerzeichen mehr hinter dem Text stehen dürfen. Das codierte Zertifikat muss außerdem mit dem

Starttext -----*BEGIN CERTIFICATE*----- beginnen und mit dem Endtext -----*END CERTIFICATE*----- abgeschlossen werden.

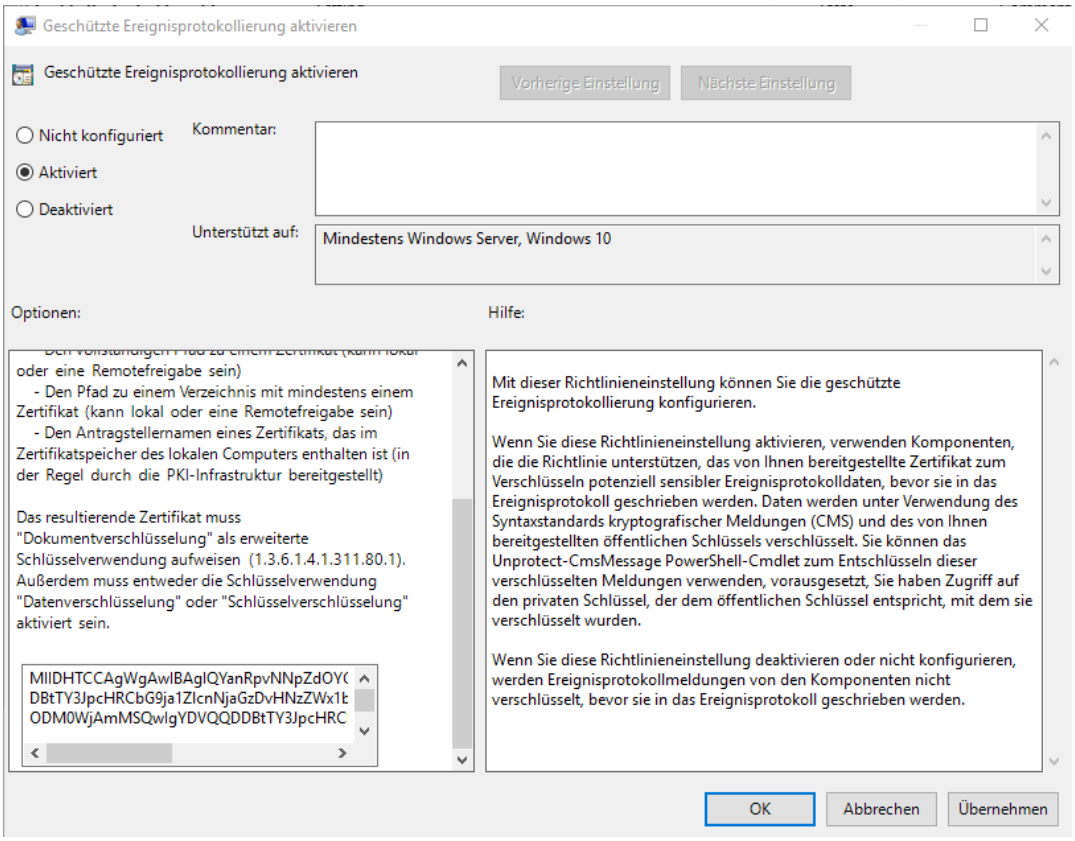


Bild 35 - Aktivieren Sie die geschützte Protokollierung, indem Sie das Base64-kodierte Zertifikat in das Optionsfeld einfügen

Sobald die Gruppenrichtlinie auf den Zielsystemen angewendet wurde, werden alle vom Skriptblock-Logging erstellen Einträge im Ereignisprotokoll verschlüsselt.

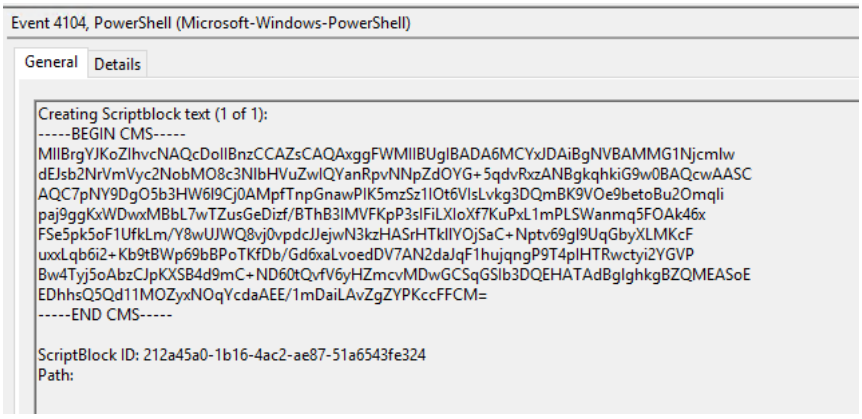


Bild 36 - Der gesamte Text ist verschlüsselt

Um die verschlüsselten Einträge zu entschlüsseln, verwenden Sie das Cmdlet Unprotect-CMSMessage.

Weiterführende Links

Spotting the Adversary with Windows Event Log Monitoring (version 2)
https://apps.nsa.gov/iaarchive/library/reports/spotting-the-adversary-with-windows-event-log-monitoring.cfm

Über den Autor

Holger Voges ist IT-Trainer und Consultant. Seine IT-Karriere begann mit einem Atari 520 ST+ Mitte der 80er Jahre. Seine ersten Erfahrungen mit großen Netzwerken hat er im Systembetrieb der Volkswagen Financial Services 1999 gewonnen. Ab dem Jahr 2000 war er dann als freiberuflicher IT-Trainer für verschiedene Schulungsunternehmen im Bereich Braunschweig und Hannover tätig, bis er 2002 mit zwei Mitstreitern sein erstes Schulungsunternehmen LayerDrei in Braunschweig gründete. Nach seinem Ausstieg bei LayerDrei war er dann mehrere Jahre als freiberuflicher Consultant vor allem im SQL-Server Umfeld u.a. für T-Home Entertain, e.on und Hewlett-Packard unterwegs. 2012 hat er das Schulungsunternehmen Netz-Weise IT-Training gegründet.

Im Dezember 2016 erschien sein Buch „Gruppenrichtlinien in Windows Server 2016, 2012 und 2008 R2“, dass er in der 3. Auflage übernommen hat, und das im Dezember 2018 in 4. Auflage komplett überarbeitet erschienen ist. Außerdem ist er regelmäßiger Sprecher z.B. auf der europäischen Powershell-Konferenz und auf verschiedenen anderen Veranstaltungen.

Netz-Weise IT-Training hat sich auf Firmenschulungen im professionellen IT-Umfeld spezialisiert und bietet Schulungen u.a. im Bereich Microsoft, VMware, Linux und Oracle an.