Sicherheitskonfigurationsrichtlinien effizient verwalten und umsetzen: Der Scapolite-Ansatz

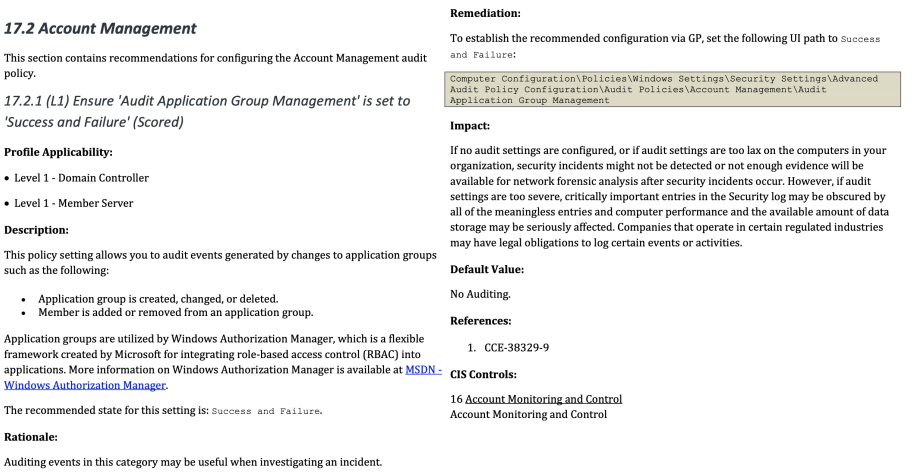
Patrick Stöckle
TUM
patrick.stoeckle@tum.de

Bernd Grobauer
Siemens AG
bernd.grobauer@siemens.com

Alexander Pretschner
Technische Universität München (TUM)
alexander.pretschner@tum.de

Zusammenfassung

Um Systeme vor Angriffen zu schützen, müssen Administratoren alle sicherheitsrelevanten Einstellungen ihrer Systeme kennen und sichere Werte wählen; hierbei setzen sie Sicherheitskonfigurationsrichtlinien um. In vielen Fällen wollen Unternehmen jedoch nicht nur externe Richtlinien verwenden, sondern basierend auf diesen eigene erstellen und verwalten. Der aktuelle Prozess zur Administration solcher Richtlinien ist überwiegend auf manuelle Übertragungen und Überprüfungen angewiesen, wodurch er fehleranfällig und teuer wird. Um den Prozess von der Erstellung bis hin zur Überprüfung und Umsetzung von Richtlinien auf den Zielsystemen schneller und effizienter zu machen, haben wir den Scapolite-Ansatz entwickelt. Hierbei nutzten wir bereits bekannte Methoden aus der Softwaretechnik wie beispielsweise Versionsverwaltung oder Continuous Integration, um Richtlinien, ähnlich wie Code, zu verwalten und zu testen. Unseren Ansatz haben wir in einer qualitativen Fallstudie bei der Siemens AG erfolgreich umgesetzt. Hierbei konnten wir zeigen, dass mithilfe des Scapolite-Ansatzes Fehler in den Richtlinien deutlich früher gefunden werden können, wodurch die Systemhärtung vereinfacht wird. Zudem konnten wir Schlüsselfaktoren identifizieren, die Aufschluss darüber geben, ob ein Unternehmen seine Richtlinien eher traditionell oder mit dem Scapolite-Ansatz verwalten sollte. Wir hoffen, dass die jeweiligen Administratoren durch die effizientere Anwendung der Richtlinien mehr Systeme sicher konfigurieren und so besser vor Angriffen schützen können.



17.2 Account Management

This section contains recommendations for configuring the Account Management audit policy.

17.2.1 (L1) Ensure 'Audit Application Group Management' is set to 'Success and Failure' (Scored)

Profile Applicability:

- Level 1 - Domain Controller
- Level 1 - Member Server

Description:

This policy setting allows you to audit events generated by changes to application groups such as the following:

- Application group is created, changed, or deleted.
- Member is added or removed from an application group.

Application groups are utilized by Windows Authorization Manager, which is a flexible framework created by Microsoft for integrating role-based access control (RBAC) into applications. More information on Windows Authorization Manager is available at MSDN - Windows Authorization Manager.

The recommended state for this setting is: `Success and Failure`.

Rationale:

Auditing events in this category may be useful when investigating an incident.

Remediation:

To establish the recommended configuration via GP, set the following UI path to `Success and Failure`:

```
Computer Configuration\Policies\Windows Settings\Security Settings\Advanced Audit Policy Configuration\Audit Policies\Account Management\Audit Application Group Management
```

Impact:

If no audit settings are configured, or if audit settings are too lax on the computers in your organization, security incidents might not be detected or not enough evidence will be available for network forensic analysis after security incidents occur. However, if audit settings are too severe, critically important entries in the Security log may be obscured by all of the meaningless entries and computer performance and the available amount of data storage may be seriously affected. Companies that operate in certain regulated industries may have legal obligations to log certain events or activities.

Default Value:

No Auditing.

References:

1. CCE-38329-9

CIS Controls:

16 Account Monitoring and Control
Account Monitoring and Control

Abbildung 1: Beispiel einer Regel aus einer Richtlinie des CIS.

1 Systemhärtung mit Sicherheitskonfigurationsrichtlinien: Eine Einführung

Fehlkonfigurationen verringern die Sicherheit eines Systems. Durch sie entstehen Schwachstellen, die schwer aufzuspüren sind. Die meisten Angriffe auf Softwaresysteme sind durch Fehlkonfigurationen deutlich leichter auszuführen oder werden dadurch erst ermöglicht. Gerade in großen vernetzten Systemen ist es den Administratoren nicht möglich, sämtliche sicherheitsrelevanten Einstellungen für alle verwendeten Software-Lösungen zu kennen. Dieser Ansicht ist auch ein Großteil der Administratoren. Sie selbst nennen eigene Wissenslücken als einen Hauptgrund für Fehlkonfigurationen. [3]

Im Allgemeinen werden von Sicherheitsexperten erarbeitete Sicherheitskonfigurationsrichtlinien (im Folgenden Richtlinien) für die Systemhärtung (im Folgenden Härtung) verwendet, um den Risikofaktor *mangelndes Wissen* zu minimieren. Diese Richtlinien bestehen aus textbasierten Regeln, die Anweisungen für die Härtung eines bestimmten Softwaresystems geben, wie beispielsweise Windows 10. Jede Regel setzt sich aus verschiedenen Bestandteilen zusammen (siehe Abbildung 1).

Für die Administratoren ist im Prozess der Härtung vor allem der Regelabschnitt *Umsetzung* hilfreich: Hier ist definiert, welche von den verfügbaren Einstellungsoptionen die Sicherheit des Systems erhöhen. Internationale Her-

ausgeber von Richtlinien sind zum Beispiel das CIS[1] oder die DISA[2]. Auf nationaler Ebene wurde in Deutschland ein solches Projekt zur Herausgabe von Richtlinien mit dem Namen [2] im BSI durchgeführt. Leider wurde dieses im Jahr 2021 eingestellt. Es gibt jedoch z.B. beim bayerischen LSI[3] Ideen für eine landesweit einheitlich geltende Richtlinie. So wird es für Organisationen wie die TUM und Unternehmen wie der Siemens AG möglich, externe Richtlinien zu nutzen, um ihre Systeme bestmöglich zu härten.

Nach der Veröffentlichung einer Richtlinie durchlaufen Richtlinien drei Stationen innerhalb eines Unternehmens. Diese sind

1. **Dokumentation**. Nutzergruppe: Sicherheitsexperten der Firma.
2. **Automatisierte Überprüfung**. Nutzergruppe: Administratoren.
3. **Automatisierte Umsetzung**. Nutzergruppe: Administratoren.

Im Folgenden beschreiben wir die drei Stationen genauer.

1. Dokumentation: In diesem Dokument erklären die Herausgeber der Richtlinien, was die Aufgabe einer Einstellung ist, warum sie sicherheitsrelevant ist, was sichere Einstellungsoptionen sind und welche potenziellen Auswirkungen durch eine Änderung der Einstellung auf das Systemverhalten zu erwarten sind. Die Sicherheitsexperten in Unternehmen und Institutionen können auf Basis der öffentlichen Richtlinien in einem zweiten Schritt unternehmenseigene Richtlinien erstellen. Bei der Anpassung der allgemeinen Richtlinien an die Bedürfnisse des Unternehmens ist vor allem wichtig, wann eine Regel in der eigenen IT nicht umgesetzt werden kann, weil z. B. Altsysteme dann nicht mehr zugreifbar wären. Auch kann es vorkommen, dass in der unternehmenseigenen IT höhere Sicherheitsanforderungen gelten, die Änderungen an den Regeln oder zusätzliche Regeln erfordern. Für die Dokumentation verwenden die Herausgeber üblicherweise allgemeine Dokumentenformate wie PDF, Tabellenformate wie EXCEL oder das für diesen Anwendungsfall standardisierte XCCDF[4] [9]; XCCDF ist Teil des umfassenderen SCAP[5]-Standards [8].

Am Ende dieser Station steht eine unternehmenseigene Richtlinie, die auf die Sicherheitsanforderungen des jeweiligen Unternehmens eingeht.

[1] Center for Internet Security
[2] Defense Information Systems Agency
[3] Landesamt für Sicherheit in der Informationstechnik
[4] Extensible Configuration Checklist Description Format
[5] Security Content Automation Protocol

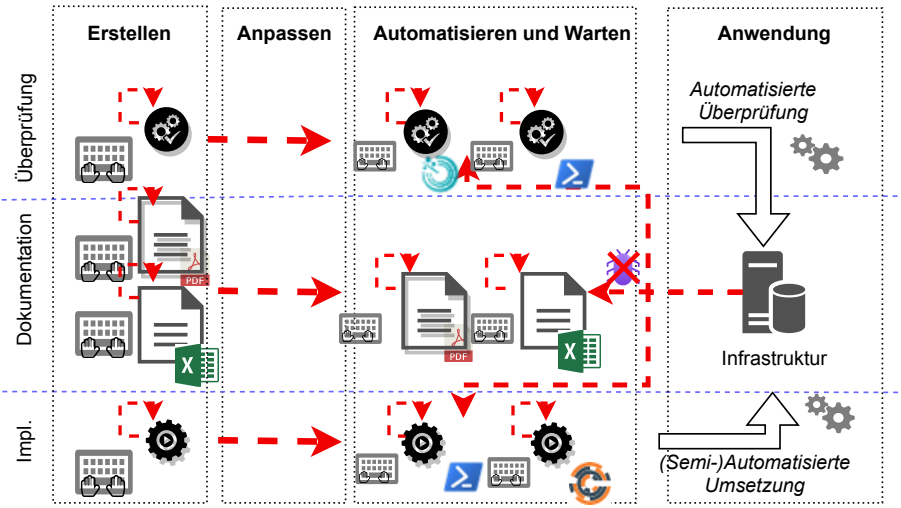


Abbildung 2: Aktueller Prozess um Richtlinien zur Systemhärtung in einer Firma oder Institution umzusetzen. Die rot-gestrichelten Pfeile symbolisieren händische Prozesse.

2. Automatisierte Überprüfung: Das im ersten Schritt entstandene Dokument ist für die Administratoren in der konkreten Umsetzung der Regeln nur bedingt hilfreich, da es für die meisten Systeme ökonomisch nicht sinnvoll wäre, hunderte Regeln einzeln händisch zu überprüfen. Bei Unsicherheiten kann das PDF jedoch zurate gezogen werden. Aufgrund dessen erstellen die Sicherheitsexperten des Unternehmens Dateien, mithilfe derer ein Administrator überprüfen kann, inwieweit ein System im Sinne der Richtlinie bereits sicher konfiguriert ist. Sie können damit die Systeme, für die sie verantwortlich sind, auf eventuelle Schwachstellen prüfen und diese dann gegebenenfalls schließen. Hierbei können die Sicherheitsexperten die Dateien entweder komplett selbst schreiben oder existierende Überprüfungsdateien, die von den Herausgebern der Richtlinien bereitgestellt werden, anpassen. Die existierenden Überprüfungsdateien sind in diesem Fall allgemeine Skriptformate wie Bash- oder PowerShell-Skripte, spezielle Scanner-Formate wie für Tenable Nessus[6] oder Qualys[7] oder die ebenfalls im SCAP definierte OVAL[8] [1].

Nach dieser Station existiert eine Möglichkeit, um Konfigurationslücken in den eigenen Systemen zu entdecken.

[6] tenable.com/products/nessus
[7] qualys.com
[8] Open Vulnerability and Assessment Language

3. Automatisierte Umsetzung: Für eine effiziente Umsetzung der Richtlinie benötigen die Administratoren Dateien, mit denen sie die Regeln der Richtlinie ausführen können. Auch hier gilt: Eine *händische* Umsetzung mit den Dokumentationsformaten ist zwar möglich, aber sehr kosten- und zeitintensiv. Ähnlich wie bei Station 2 erstellen unternehmenseigene Sicherheitsexperten diese Umsetzungsdateien neu oder auf Basis von Dateien, die die Herausgeber bereitstellen. Für die automatisierte Umsetzung verwenden sie meist ebenfalls PowerShell, proprietäre Formate wie Gruppenrichtlinien-Exporte oder Konfigurationsmanagement-Formate wie Ansible[9] oder Chef[10]; ein standardisiertes Format für die Umsetzung gibt es nicht. Am Ende dieser Station können die Administratoren die Richtlinien über die Umsetzungsdateien ausführen.
Der bisherige Prozess ist in Abbildung 2 dargestellt. Die Autoren der Sicherheitsrichtlinie erstellen alle Dokumente getrennt und übertragen auch die Änderungen in alle Artefakte getrennt. Falls die Administratoren eigene Tools zur Umsetzung oder Überprüfung verwenden, erstellen diese dann auf Basis der Richtlinie eigene Überprüfungen/Umsetzungen. Die Administratoren überprüfen damit die Systeme in ihrer Infrastruktur und härten diese bei Bedarf. Sollten Anpassungen nötig sein, z. B. weil das System nicht mehr funktioniert, wenn Regel X angewendet wird, so muss dies in alle Artefakte übertragen werden, um Verwirrung zu vermeiden.

2 Probleme der bisherigen Vorgehensweise

Während des oben beschriebenen Prozesses mit den drei Stationen sind bei der Siemens AG verschiedene Probleme aufgetreten:

A. Fehler beim Anpassen der Richtlinien: Beim Anpassen der Richtlinien kam es immer wieder zu kleineren oder größeren Fehlern. Kleinere Fehler waren beispielsweise Rechtschreibfehler in der Dokumentation, bei größeren Fehlern sollte eine Einstellung konfiguriert werden, die es nicht gab, oder es sollte eine Einstellungsoption gesetzt werden, die nicht gültig war. Um derartige Fehler zu finden und zu vermeiden, gab es bei der Siemens AG zwar Qualitätssicherungsverfahren, diese waren allerdings sehr zeit- und kostenaufwändig.

B. Inkonsistenzen zwischen den Dokumenten: Für die fehlerfreie Umsetzung einer Richtlinie ist es vonnöten, dass in den Dokumenten, die das Ergebnis der jeweiligen Station (Dokumentation, automatisierte Überprüfung

[9] ansible.com/products/automation-platform
[10] chef.io

und automatisierte Umsetzung) sind, die gleiche Einstellungsoption für eine Einstellung definiert ist. Hier kam es nicht nur bei der Siemens AG immer wieder zu Inkonsistenzen zwischen den einzelnen Dokumenten. Besonders gravierend sind diese Fehler, wenn das entsprechende Unternehmen sie nicht durch interne Qualitätssicherungsmaßnahmen entdeckt und die Richtlinien so produktiv einsetzt oder die publiziert. Vorkommnisse wie diese betreffen nicht nur das Beispielunternehmen als Nutzer solcher Richtlinien, sondern, wie wir aus Gesprächen mit dem CIS wissen, auch die Herausgeber.

Ein Beispiel für eine solche Inkonsistenz, die nicht von der Internen QA gefunden wurde, war die Regel *1.8.7.4* der CIS Office Word 2016 Richtlinie. In den publizierten PDF und XCCDF Dokumenten fordert das CIS, dass `Turn off file validation` auf `Disabled` gesetzt werden soll, in der automatische Überprüfung wurde aber der Wert `Enabled` überprüft.

Gerade Anpassungen an den Richtlinien haben immer wieder dazu geführt, dass die Dokumentation einen anderen Wert spezifizierte als die Umsetzung einstellte oder die Überprüfung checkte. Dabei kam es zu zwei unterschiedlichen Problemen: Im ersten Fall kam es dazu, dass im Zuge der Überprüfung der Test einer Einstellung fehlschlug, obwohl bereits die richtige Einstellungsoption für eine sichere Konfiguration gesetzt war (false negative). Durch die händische Überprüfung entstand ein unnötiger Mehraufwand für die Administratoren. In Beispiel der Regel *1.8.7.4* hieß dies, dass die automatische Überprüfung einen falschen Wert für die Einstellung anzeigte. Wenn der verantwortliche Administrator die Einstellung auf den Wert setzen wollte, fand er schon den in den Dokumenten geforderten Wert vor. Solche Erlebnisse sorgen für Frust bei den Administratoren und schmälern das Vertrauen in die Überprüfungsergebnisse.

Im zweiten Fall trat das Problem im umgekehrter Version auf. Manche Überprüfungen waren erfolgreich, obwohl noch eine unsichere Einstellungsoption gesetzt war (false positive). Diese Situation ist problematischer, da Sicherheitslücken nicht aufgedeckt wurden. Im Beispiel wäre die Überprüfung korrekt, obwohl der Wert für die File Validation Einstellung auf Enabled ist. Dies könnte einem Angreifer eine Attacke ermöglichen oder zumindest vereinfachen.

Die Problematik verschärft sich, falls in einem Unternehmen verschiedene Werkzeuge zur Umsetzung oder Überprüfung unterstützt werden müssen. Falls das Unternehmen z. B. sowohl PowerShell als auch Nessus für die Überprüfung der Richtlinie nutzt, müssen die Nessus-, die PowerShell- und die PDF-Dateien zwingend die gleichen Werte enthalten. Die verschiedenen Formate potenzieren den Mehraufwand, der nötig ist, um Inkonsistenzen aufzudecken.

C. Regeln nur über grafische Nutzeroberfläche umsetzbar: Auf manchen Systemen lassen sich die Regeln nur über eine grafische Nutzeroberfläche (GUI) umsetzen. Dies führt zu einem Handhabungsproblem. Bei einer Richtlinie von hunderten von Regeln bedeutet eine Umsetzung über die GUI, dass der Administrator sich durch zahlreiche Menüs klicken und in jedem Menü die richtige Einstellungsoption wählen muss. Aufgrund der hohen Anzahl an Regeln kam es immer wieder dazu, dass Regeln vergessen oder falsche Einstellungsoptionen für Regeln gesetzt wurden. Vor allem im Windows Kontext ist das der Fall, weshalb es hier zu zahlreichen Fehlern kam.

D. Ganz oder gar nicht: In vielen Systemen lassen sich Regeln zwar mithilfe von Programmen automatisiert umsetzen, jedoch nur komplette Richtlinien und nicht einzelne Regeln oder Gruppen von Regeln. Die Einschränkung ist im Anpassungsprozess hinderlich, bei dem die Sicherheitsexperten die für die eigene IT problematischen Regeln herausfiltern müssen. Da im Regelfall immer Anpassungen vonnöten sind, ist eine komplette Umsetzung der Richtlinie nur selten sinnvoll. Administratoren müssen problematische Regeln manuell wieder entfernen, was meist ebenso mühsam ist wie die gewünschten Regeln von Hand anzuwenden. Die angebotene Automatisierung ist somit nicht hilfreich.

E. Nachvollziehbarkeit von Änderungen: Die Herausgeber aktualisieren ihre Richtlinien im Regelfall alle sechs Monate. Wenn ein Herausgeber eine neue Version einer Richtlinie veröffentlicht, müssen sich die Sicherheitsexperten auf das mitgelieferte *Changelog* verlassen, um die Aktualisierungen auch auf die unternehmenseigene Richtlinie zu übertragen. Bei fehlendem oder schlecht geführtem Changelog, müssen die Sicherheitsexperten die aktualisierte, allgemeine Richtlinie manuell mit der vorherigen Version abgleichen, um Änderungen zu finden. In anderen Kontexten würde man hier auf ein *Diff* zurückgreifen, wie ihn die meisten Versionsverwaltungswerkzeuge anbieten. Das ist auf den PDF-, XLSX- oder GPO-Dateien aber in der Regel nicht sinnvoll, weil Änderungen in diesen Formaten nicht kleinteilig angezeigt werden können. Daher sind Aktualisierungen von Richtlinien schwer nachzuvollziehen und die Anpassung der unternehmenseigene Richtlinie schwer.

3 Lösungsansatz

Zur Lösung der eben beschriebenen Problemfelder A-E haben die TUM und die Siemens AG den sogenannten **Scapolite**-Ansatz entwickelt[11]:

Automatisierte Überprüfung der Regeln auf Semantische Korrektheit: Um den bei Problem A beschriebenen Qualitätssicherungsaufwand zu reduzieren, werden geänderte Regeln automatisiert auf ihre semantische Korrektheit überprüft. Im Zuge dessen kontrolliert unser Werkzeug erstens, ob die Einstellung, auf die die geänderte Regel sich bezieht, existiert. Zweitens gleicht es die geforderte Einstellungsoption mit den möglichen Einstellungsoptionen für diese Einstellung ab. Wir greifen dazu auf Definitionen von vorhandenen Einstellungen und legalen Einstellungsoptionen zurück. Diese extrahiert automatisiert das Werkzeug z. B. aus den administrativen Vorlagen von Microsoft. Unsere Vorarbeiten [7] in diesem Bereich liefern für diesen Schritt die wissenschaftliche Grundlage. Darüber hinaus haben wir eine Continuous Integration (CI) Pipeline eingerichtet, die jede Änderung automatisiert auf ihre semantische Korrektheit überprüft. Die zeitaufwändige Überprüfung der Regeln durch die Sicherheitsexperten wird so durch eine automatisierte Variante ersetzt.

Regel-zentrierte Verwaltung: Um Inkonsistenzen vorzubeugen (Problem B), haben wir eine verbesserte Methode zur Verwaltung der Regeln entwickelt. Anstatt die Verwaltung nach den Stationen zu organisieren, wird jeder Regel bei uns ihr eigenes Dokument zugewiesen. In diesem Dokument befinden sich dann alle Informationen über die Dokumentation, die automatisierte Überprüfung und die automatisierte Umsetzung der Regel gebündelt. Man erhält so alle Informationen zu einer bestimmten Regel auf einen Blick, wodurch die Wahrscheinlichkeit für Inkonsistenzen sinkt. Um dennoch die einzelnen Formate zur Verfügung zu stellen, generiert unsere CI Pipeline zusätzlich aus den einzelnen Regeldateien die üblichen Dokumentations-, die Umsetzungs- und die Überprüfungsdateien in den jeweiligen Formaten.

Fokus auf Automatisierung: Mit unserem Werkzeug können wir alle Regeln anders als bisher mit einfachen Kommandozeilen-Befehlen umsetzen (`Apply-Rule XYZ`). Somit entfällt die Navigation in der GUI (Problem C). Außerdem ist unser Werkzeug in der Lage, einzelne Regeln, Listen von Regeln oder ganze Richtlinien als Eingabe zu verarbeiten. Hierdurch entlasten wir die

[11] Der Anfang (Scap-) bezieht sich auf den SCAP-Standard, von dem wir aber nur einen Teil berücksichtigen, daher das Suffix (-lite). Dass Scapolite im Englischen das harte Mineral Skapolith bezeichnet, ist im Kontext der (System-)Härtung ein amüsanter Zufall.

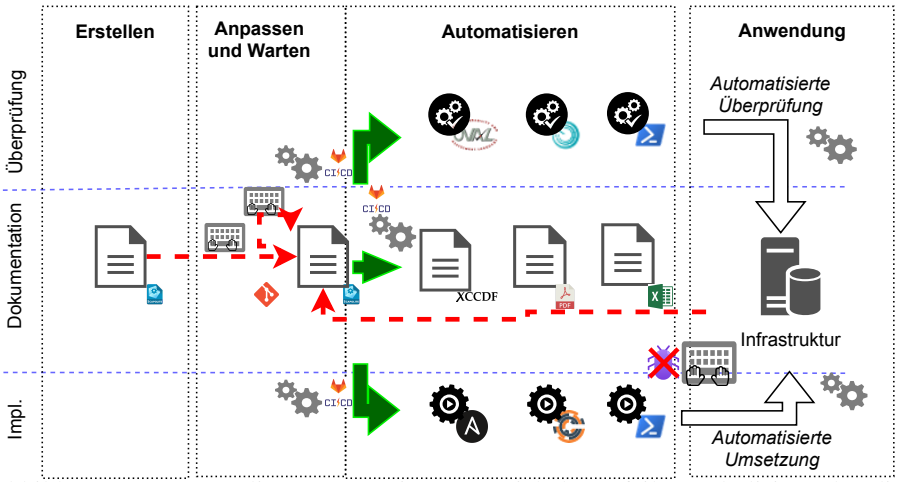


Abbildung 3: Der Scapolite-Ansatz. Die rot-gestrichelten Pfeile symbolisieren händische Prozesse, die grünen Pfeile die automatisierten CI Schritte.

Administratoren beim Umsetzen der Regeln und die Befehle erleichtern das Aufspüren von Regeln, die in produktiven Systemen zu Problemen führen.

Richtlinien-as-Code: Um Änderungen einfach nachvollziehen zu können, werden im Scapolite-Ansatz alle Richtlinien wie Code verwaltet, d.h. alle Regeldateien werden mit dem Versionsverwaltungssystem *git* versioniert. So können wir Änderungen über Diffs nachvollziehen und Updates sowie verschiedene Versionen in verschiedenen Zweigen abbilden. Die bereits genannte CI Pipeline erzeugt, wie oben bereits erwähnt, für jeden Zweig die aktuellsten Artefakte und stellt diese in allen benötigten Formaten den Administratoren bereit. Einerseits bekommen die Administratoren in unserem Ansatz neue Regeln so schnell wie möglich, andererseits können die Administratoren so Fehler auch für alte Versionen der Richtlinien nachvollziehen.

Abbildung 3 zeigt den Scapolite-Ansatz. Die Autoren erstellen die Scapolite-Dateien, die die Dokumentation, Umsetzung und Überprüfung enthalten und machen alle Anpassungen auf diesen Dateien. Sie verwalten die Dokumente in git und können parallel an den Richtlinien arbeiten. Die GitLab-CI generiert mit unseren Tools für jede Änderung die Überprüfungs- und Umsetzungsskripte sowie die Dokumentation, z. B. in PDF. Damit können nun die Administratoren ihre Systeme überprüfen und härten mit den Werkzeugen, die sie benötigen.

Die in Abbildung 2 noch händischen Prozessschritte sind im Scapolite-Ansatz durch eine automatisierte Vorgehensweise ersetzt worden. Durch die Automatisierung werden häufige Fehlerquellen eliminiert.

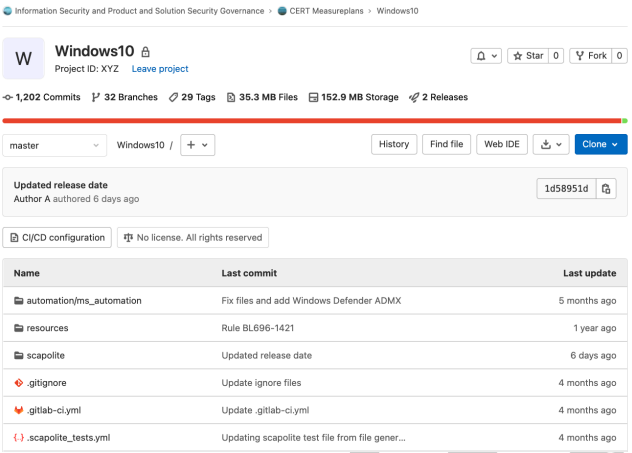


Abbildung 4: Die Siemens Windows 10 Richtlinie in der GitLab-Ansicht.

4 Fallstudie

Im Folgenden stellen wir dar, wie der Scapolite-Ansatz bei der Siemens AG umgesetzt wurde und beleuchten, welche Erkenntnisse wir im Austausch mit den Administratoren und Autoren der Richtlinien gewonnen haben. Zunächst erläutern wir die Verwaltung der Richtlinie. Im darauffolgenden Abschnitt gehen wir kurz auf die Verwendung der CI ein, bevor wir dann den Aufbau der Richtlinie näher beschreiben. Zuletzt stellen wir unsere lessons learned vor.

Im Rahmen dieser Fallstudie wurde die siemensinterne Windows 10 (W10) Richtlinie mit dem Scapolite-Ansatz umgesetzt. Die Arbeiten begannen 2017 und laufen kontinuierlich weiter, um einerseits akute Gefahren abzuwehren und andererseits auf Richtlinien-Änderungen zu reagieren.

Wir verwalten Richtlinien mit der Versionsverwaltungssoftware git[12] und nutzen eine bei Siemens gehostete GitLab[13]-Instanz als zentrales Repository für alle internen Richtlinien. Die Figur Abbildung 4 zeigt einen Screenshot der Weboberfläche zu der W10 Richtlinie. Durch die Abbildung wird deutlich, dass die Entwicklung von Richtlinien im Scapolite-Prozess äquivalent zur Softwareentwicklung ist, d.h. die Autoren bearbeiten die Richtlinie über Commits, die sie lokal oder über die GitLab Weboberfläche durchführen können. Durch die Commits kann auch im Nachhinein nachvollzogen werden, zu welchem Zeitpunkt eine Änderung eingeführt wurde. Durch Branches wird den Autoren ermöglicht,

[12] git-scm.com
[13] about.gitlab.com

parallel an einer Richtlinie zu arbeiten. In Siemens werden die Branches auch genutzt, um verschiedene Versionen von Betriebssystemen anzupassen (W10 1809, 1909) oder über Feature-Branches neue Releases vorzubereiten.

Die GitLab CI wird genutzt, um bei jeder Änderung der Richtlinie die Regeln auf Fehler zu überprüfen und für jeden Commit die Artefakte, z. B. die Skripte zur Umsetzung, zu generieren. Innerhalb von vier Jahren haben so 14 Autoren in 1202 Commits Rechtschreibfehler in der Dokumentation entfernt, Regeln hinzugefügt oder geändert und auf diese Weise Fehler behoben. Insgesamt liefen während dieser Zeit 136 Pipelines; in Abbildung 5 sehen Sie ein Beispiel. Im Rahmen einer Pipeline führt die CI verschiedene Aufgaben durch, z.B. wird automatisiert überprüft, ob die angegebenen Gruppenrichtlinien-Pfade existieren, ob die Werte für diese Pfade legal sind und es werden die Werte aus der Oberfläche in Registry-Werte übersetzt (Abbildung 5 *Enrich*-Phase). Als Kernaufgabe startet die CI VMs, die anschließend auf Regelkonformität überprüfen werden. Sodann wird die Richtlinie angewendet und abschließend nochmals überprüft (Abbildung 5, *Test_execution*-Phase). Der zuletzt durchgeführte Dreischritt wird automatisiert mit älteren Ergebnissen abgeglichen, um unerwünschte Änderungen identifizieren zu können. In ca. 32% der Fälle stellten unsere Werkzeuge Fehler fest, die von den Autoren erst behoben werden mussten, bevor die nächste CI die Skripte wieder generieren konnte.

Unsere Richtlinien bestehen aus vielen Regeln (z. B. über 500 Regeln für W10). Jede Regel besteht aus einer Markdown-Datei mit einer YAML-Präambel. In der Präambel speichern wir strukturierte Daten über die Regel, im Markdownteil können die Autoren die Dokumentation formatieren. Eine Beispiel-Regel finden Sie in Listing 1. Nachfolgend stellen wir einige unserer lessons learned vor.

Unveränderliche IDs vor semantischer Information: Eine wichtige Erfahrung, die wir beim Verwalten von Richtlinien bei Siemens gemacht haben, betrifft die ID *BL696-0616* (Z.5). Die CIS benennt ihre IDs nach Richtlinien-Struktur (R1.1.1, R1.2.1). Diese Struktur macht das CIS abhängig von den Pfaden der Einstellungen. Die Pfade sind weder einheitlich für verschiedene W10-Versionen noch sind sie flexibel wenn beispielsweise eine Gruppe hinzugefügt wird. In diesem Fall kommt es entweder zu Inkonsistenzen in der Benennung oder viele IDs müssen geändert werden. Aufgrund dessen ändert das CIS seine IDs ständig, wodurch das Vergleichen von verschiedenen Versionen einer Richtlinie und der darauf basierenden Überprüfungsergebnisse sehr aufwändig ist. Aus diesem Grund haben wir uns für zufällige IDs (hier *0616*)

```yaml
---
scapolite:
  class: rule
  version: '0.51'
id: BL696-0616
id_namespace:
↪ com.siemens.seg.policy_framework.rule
title: Enable Pattern Update Check before Scans
rule: <see below>
rationale: <see below>
description: <see below>
applicability:
- system: siemens.acp
  c: '123'
  i: '123'
  a: '123'
- system: siemens.scapolite.target_audience
  roles:
  - asset_manager
implementations:
- relative_id: '01'
  description: <see below>
  automations:
  - system: org.scapolite.implementation.win_gpo
    ui_path: Computer
    ↪ Configuration\Policies\Administrative
    ↪ Templates\Windows Components\Windows
    ↪ Defender Antivirus\Scan\Check for the
    ↪ latest virus and spyware security
    ↪ intelligence before running a scheduled
    ↪ scan
    value: Enabled
    verification_status: Checked.
...
history:
- version: '2.5'
  date: '2020-03-12'
  action: revised
  hide: true
  description: Added automation information to
  ↪ Yaml
  internal_comment: ''
...
---
## /rule
Set the _Check for the latest virus and spyware
↪ security intelligence before running a
↪ scheduled scan_ setting to the value
↪ _Enabled_.
## /rationale
As malware may spread very rapidly, the latest
↪ security intelligence data should be used
↪ when running a scan.
## /description
This policy enabled the check for new antivirus
↪ pattern updates before running a scheduled
↪ scan.
## /implementations/0/description
To set the protection level to the desired state
↪ set the following Group Policy setting to
↪ `Enabled`
`Computer Configuration\Policies\Administrative
↪ Templates\Windows Components\Windows
↪ Defender Antivirus\Scan\Check for the latest
↪ virus and spyware security intelligence
↪ before running a scheduled scan`.
```

Listing 1: Eine Regel aus der Siemens Windows 10 Richtlinie.

mit dem Richtliniennamen als Präfix (hier *BL696*) entschieden. Es lassen sich zwar anhand der ID keine Aussagen mehr über die Regel treffen, jedoch ist keine ständige Umbenennung mehr nötig.

Eine Richtlinie, viele Varianten: Bei manchen Regeln muss zwischen Sicherheitsgewinn und den entstehenden Einschränkungen abgewogen werden. Im Zuge einer Richtlinienerstellung können wir entweder jede Regel, die nur entfernt sicherheitsrelevant ist, einschließen, oder wir wählen lediglich die sicherheitsrelevantesten Regeln aus. Möglich ist es auch, einen Kompromiss aus beidem zu verfolgen. Indem wir die Richtlinie in einem zentralen Verzeichnis verwalten, wird es möglich, auf Basis der *Applicabilities* verschiedene Spezialisierungen dieser Richtlinie, je nach Nutzerwunsch, bereitzustellen. Über eine Applicability kann der Autor einer Regel festlegen, für welche Sicherheitseigenschaft (wir verwenden hier die klassischen Eigenschaften *confidentiality (c)*, *integrity (i)*, und *availability (a)*) und für welches jeweilige Sicherheitsniveau (wir verwenden hier 3 Level) eine Regel gelten soll. `c:123` bedeutet so, dass eine Regel für alle Vertraulichkeitsstufen gilt, während `c:3` dafür sorgt, dass eine Regel nur bei der höchsten Vertraulichkeitsstufe verwendet wird. Auf Systemseite gibt der Administrator die entsprechende Sicherheitsstufe des Systems

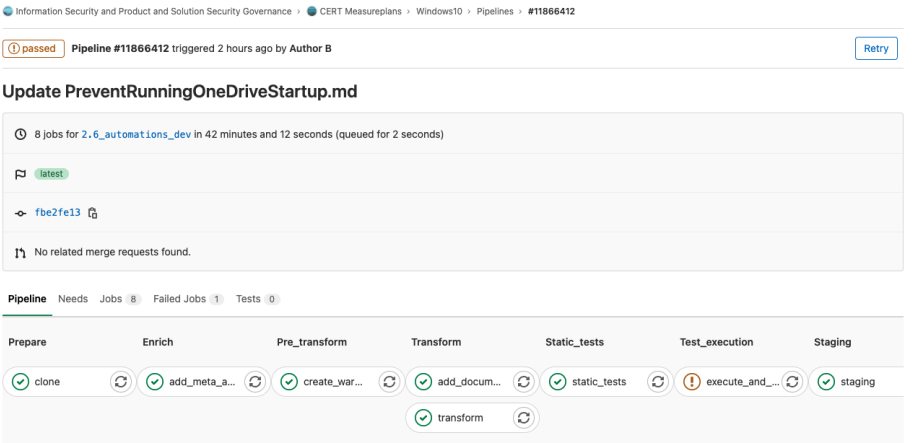


Abbildung 5: Eine Pipeline im Scapolite-Prozess.

an, z. B. niedrige Vertraulichkeit `c:1`, mittlere Integrität `i: 2` und hohe Verfügbarkeit `a:3`. Unsere Werkzeuge setzen dann für das System nur die Regeln um, auf die diese Sicherheitsbeschreibungen zutreffen. So können die Regeln bei Siemens einerseits weiterhin einfach verwaltet werden und andererseits werden lediglich die Regeln umgesetzt, die für das jeweilige System angemessen sind.

Fehler früh finden Unsere Annahme war, dass Sicherheitsexperten und Administratoren durch die semantische Überprüfung, die Regel-zentrierte Verwaltung und unsere CI Pipeline Fehler in den Regeln früher erkennen und effektiver beheben können, da nicht mehr verschiedene Dokumente überblickt und angepasst werden müssen. Somit lässt sich einerseits Zeit in der Qualitätssicherung einsparen und andererseits die Qualität von Richtlinien erhöhen. In unserer Fallstudie konnten wir zeigen, dass unsere Überprüfungen zusammen mit der CI tatsächlich dazu führen, dass wir die Fehler schon deutlich früher finden als zuvor. Vorher konnten Fehler erst gefunden werden, wenn die Administratoren vorläufige Versionen einer Richtlinie umsetzen wollten und dabei auf die Fehler aufmerksam wurden. Jetzt können viele Fehler meist schon kurz nach dem Erstellen entdeckt und behoben werden. Ein Beispiel hierfür ist die Regel *BL696-4537*. Ein Autor hat diese Regel durch einen Commit vom 17. Juni 2021, 13:48 hinzugefügt. Hierbei sollte die Einstellung *Configure detection for potentially unwanted applications* auf den Wert *Audit* gesetzt werden. Das Problem: *Audit* ist kein legaler Wert für diese Einstellung, der gewünschte Wert ist *Audit Mode*. Dies wird von unseren Werkzeugen erkannt, die Regel bleibt weiterhin *Unchecked* – der Status wird direkt in der Datei gespeichert

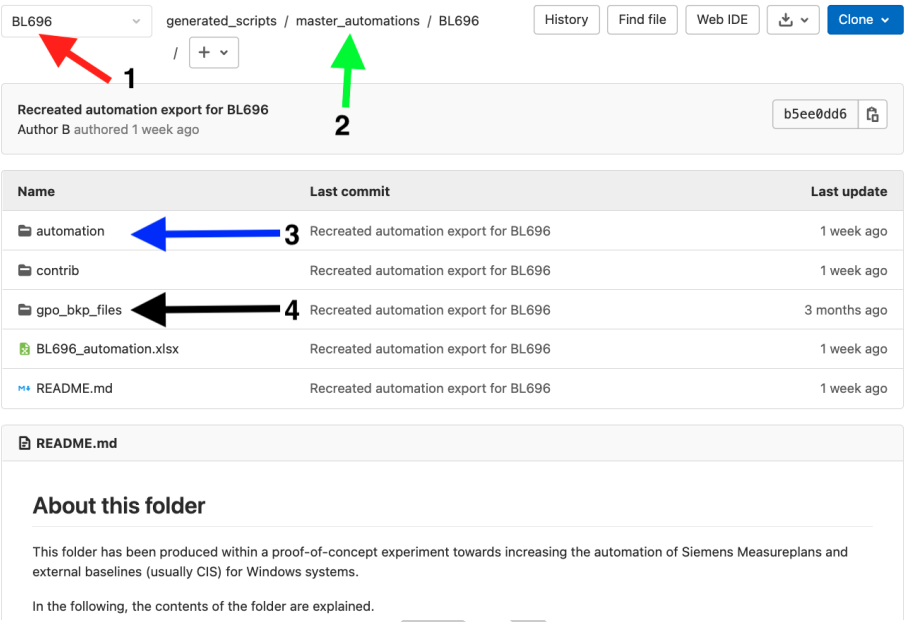


Abbildung 6: Ein Commit in unserem neu eingeführtem Artefakte-Repository.

(s. Listing 1, Z.26) – und unsere Werkzeuge zeigen die möglichen Legalen werte, in diesem Falle *Block*, *Disable* und *Audit Mode*, in der Datei an. Aus dem Commitlog können wir nachvollziehen, dass dies im Falle unseres Beispiels so gut geklappt hat, dass die Regel schon um 14:56 von einer anderen Autorin gefixt wurde. Dieses anekdotische Beispiel zeigt, wie unser Überprüfungsansatz einfache Fehler mit wenig Aufwand schon innerhalb einer knappen Stunde beheben kann, die sonst erst nach Wochen behoben wurden.

Versionierung für Quellen *und* Artefakte: Ursprünglich hatten wir nur unsere Regel-Dateien mit git versioniert. Im Laufe unserer Fallstudie erkannten wir aber immer mehr, dass es genauso sinnvoll ist, unsere generierten Skripte in einem getrennten Repository zu versionieren. In der aktuellen Version erstellt die Pipeline einen Commit mit den aktuellen Artefakten und schiebt diesen in das Artefakt-Repository (Abbildung 5, *Staging*-Phase). Abbildung 6 zeigt dieses Repository. Der Branch (Pfeil 1) bezeichnet das Quell-Repository, für das diese Artefakte erzeugt wurden. Der Ordner (Pfeil 2) zeigt an, für welchen Branch des Quell-Repositorys diese Artefakte erzeugt wurden. Unter *automation* (Pfeil 3) finden die Administratoren die Skripte zur Umsetzung und Überprüfung und unter *gpo_bkp_files* (Pfeil 3) generierte GPO Backups, die

sie einspielen können. Über die Oberfläche können Administratoren und wir verschiedene Versionen der Artefakte direkt vergleichen. Die Administratoren können somit die Auswirkungen von Richtlinienänderungen abschätzen und wir konnten Änderungen in unseren Werkzeugen besser validieren.

Je einfacher die Härtung, desto mehr Systeme werden gehärtet: In unserer Fallstudie hat sich gezeigt, dass ein einfacherer Härtungsprozess dazu führt, dass die Administratoren Härtungen häufiger und früher durchführen; Administratoren innerhalb von Siemens meldeten uns dies in Gesprächen zurück. Vorher war die Härtung vor allem ein Schritt, der vor der Auslieferung durchgeführt wurde, da der Mehraufwand durch die mühsame, händische Härtung so auf ein Minimum reduziert wurde. Wenn aber aufgrund der Härtungsmaßnahmen nun Funktionen nicht mehr gegeben waren, dann war die Fehlersuche an dieser Stelle sehr mühsam. Durch die Automatisierung konnte die Härtung jetzt aber schon früher, z. B. während des Testens, umgesetzt werden. Um dies noch weiter zu vereinfachen haben wir in Absprache mit unseren Nutzern eingeführt, dass man lokal eine Liste von Regeln definieren kann, die nicht umgesetzt werden. Dadurch können Regeln, die auf den meistens virtuellen Testsystemen keinen Sinn ergeben, ausgeklammert werden und führen während des Testens weder zu Fehlern noch zu unnötigen Aufwand, um zu false negatives durchzuschauen. In unserer Fallstudie bei Siemens haben Administratoren z.B. Regeln für das TPM entfernt, da dies bei den Test-VMs nur zu Problemen führen würde. Außerdem entfernten die Administratoren Regeln, die PowerShell-Skripte oder Remote Desktop Services verboten, da die VMs darüber verwaltet wurden. Die Administratoren können diese Listen in JSON-Dateien speichern. So kann schon früh das Produkt auf einer gehärteten VM getestet werden. Vor der Auslieferung setzen die Administratoren dann das gesamte Profil um. Sollte es dann zu Problemen kommen, müssen nur die problematischen Regeln überprüft werden. Vor allem aber die Möglichkeit, Windows-Richtlinien einfach per Skript umzusetzen, hat dafür gesorgt, dass die Härtung breitere Akzeptanz gefunden hat, z. B. hat eine Gruppe von Nutzern die Härtung direkt in die Erstellung von Basis-VMs eingebaut. Diese gehärteten VMs können dann von anderen Nutzern z. B. im Cloud-Kontext als Grundlage für ihre Zwecke verwendet werden. Auch die Möglichkeit, Gruppen von Regeln umsetzen und Regeln einfach zurücknehmen zu können, wurde in unserer Fallstudie sehr positiv bewertet.

Klassifikation von störungsanfälligen Regeln und der Umgang damit: Ein Problem, dessen Wichtigkeit wir erst während unserer Fallstudie erkannt haben, sind die *disruptiven* oder brechenden Regeln. Wir bezeichnen so Regeln,

die dafür sorgen, dass eine gewisse Funktion des Systems nicht mehr gegeben ist; im schlimmsten Fall kann man auf das System überhaupt nicht mehr zugreifen. Manche Regeln sind nur für wenige Spezialanwendungen problematisch und daher nur für wenige Administratoren relevant. Die Administratoren dieser Systeme kennen die Probleme meistens, können die entsprechende Regel daher einfach bestimmen und diese für ihr System über die eben schon erwähnte Liste bei der Umsetzung ausklammern. Andere Regeln sind aber auf vielen Systemen disruptive, z. B. die Regel *BL968-0591*, wodurch man sich nur noch mit einer Smart Card anmelden kann. Der Ablauf hier war häufig folgender: Ein Administrator wollte die Umsetzung unserer Richtlinie testen und führte die Skripte auf einem Testsystem aus. Anschließend hatte er keinen Zugriff mehr auf sein System, setzte sein Testsystem zurück und ging durch die Regeln, um die problematischen Regeln zu finden; oder er wandte sich an die Autoren oder an uns, um uns zu fragen, was hier die Ursache für die Probleme sein könnte. Auf diese Weise ging viel Zeit für die Fehlersuche verloren, die man hätte sparen können. Um das Problem zu lösen, markierten wir die besonders disruptiven Regeln direkt in der Richtlinie. Diese Information exportieren wir dann auch in die Skripte zur Umsetzung und Überprüfung. Unser erster Ansatz war dann, dass diese Regeln nur umgesetzt werden, wenn der ausführende Administrator die Variable hierfür aktiv auf `true` setzt. Hierdurch konnten wir zwar die Anfragen zu den häufigen Regeln reduzieren, doch dies führte dazu, dass diese Regeln z.T. auch auf den produktiven Systemen vergessen wurden.

Durch diese Erfahrung änderten wir unseren Ansatz: Wenn ein Administrator Regeln anwendet, geben wir ihm die Liste der disruptiven Regeln innerhalb dieser und deren Problembeschreibung an. Der Administrator muss aktiv *ja* angeben, um diese Regeln umzusetzen. Erfahrene Administratoren können diese Überprüfung durch einen Kommandozeilenparameter überspringen. Auf diese Weise haben wir erreicht, dass vor allem Erstnutzer unserer Skripte weniger Zeit in Probleme investieren, die bereits gelöst waren, und sorgen dennoch dafür, dass die Regeln auch überall umgesetzt werden.

5 Diskussion

Basierend auf unseren Erfahrungen bei Siemens haben wir verschiedene Faktoren erarbeitet, die Entscheidern in Unternehmen und Gemeinden helfen können. Sind diese Faktoren in einem Unternehmen gegeben, empfiehlt sich ein Ansatz wie der Scapolite Ansatz für das Härten. In der Diskussion ist für uns der

Vergleichspunkt die traditionelle Herangehensweise an die Systemhärtung wie sie in Abbildung 2 abgebildet ist.

Reine Verwendung: In vielen Anwendungsfällen müssen wir nachweisen, dass wir unser System nach externen Standards gesichert haben. Hier ist der zusätzliche Aufwand für den Scapolite-Ansatz nicht gerechtfertigt, da wir die Richtlinie ja nicht anpassen wollen, d. h. weder die Nachvollziehbarkeit noch das gemeinsame Arbeiten an einer Richtlinie noch das Überprüfen von Änderungen sind relevante Punkte. Der externe Anbieter liefert meist die Skripte für die automatische Überprüfung, manchmal sogar die Skripte für die Umsetzung.

Keine Varianten: Die Komplexität verschiedener Varianten und Profile ist eine Motivation für den Scapolite-Ansatz. Gerade kleinere Firmen oder Unternehmen umgehen diese Komplexität aber, in dem immer nur eine Version unterstützt wird und es nur ein Profil gibt. Falls Sie also planen sollten, eine Richtlinie zu erstellen, die nur für ein kleines Unternehmen gültig ist, ist der Scapolite-Ansatz unnötiger Mehraufwand.

Ein Autor pro Richtlinie: Bei Siemens arbeiten viele verschiedene Autoren an einer Richtlinie, bei der W10 Richtlinie waren es z. B. 14 Autoren. Daher müssen sich die Autoren untereinander abstimmen und es werden Werkzeuge wie git notwendig. Bei vielen Firmen gibt es aber kein Sicherheitsteam, sondern nur einen Sicherheitsbeauftragten. In diesem Falle gibt es keinen Grund, die Dokumente im Scapolite-Format zu verwalten und der Autor kann die Werkzeuge, die im Scapolite-Ansatz meistens in der CI laufen, lokal installieren und ausführen, bevor er eine neue Version der Richtlinie freigibt.

Keine Wartung: Das Weiterentwickeln einer Richtlinie war im Siemens-Kontext sehr wichtig, da wir auf Feedback der Administratoren eingehen wollten. Manche Administratoren bringen ihr praktisches Wissen ein und schlagen Regeln vor oder verweisen auf neue Angriffe, die zusätzliche Regeln erfordern. Andererseits können Sie auch zurückmelden, dass eine Regel zu streng ist für gewisse Standardumgebungen und wir sie daher entweder lockern, z. B. durch einen Wert, der nicht mehr so streng ist, entfernen oder in ein Profil mit höherer Sicherheitsanforderung verschieben sollten. Manche Situation erfordern aber nur, dass wir ein Produkt härten und anschließend ist die Verantwortung nicht mehr bei uns. In diesem Falle interessieren uns weder die Nachverfolgbarkeit von Änderungen noch die gute Verwaltbarkeit von Scapolite Dokumenten. Daher können wir auch diesem Falle den traditionellen Prozess empfehlen.

Homogene Werkzeuge: Die Administratoren bei Siemens hatten uns früh zurückgemeldet, dass sie verschiedene Formate für die Dokumentation, Über-

prüfung und Umsetzung verwenden. Manche Administratoren wollen ein PDF Dokument zum Nachschlagen, andere eine EXCEL Liste. Die einen überprüfen ihre Systeme mit Scannern wie Tenable Nessus[14], andere verlangen nach Checks im OVAL-Format. Andere wollen die Regeln über die vorinstallierte PowerShell überprüfen. Für die Umsetzung ergaben sich vor allem die Varianten Active Directory (AD) oder Einzelsystem. Im AD Fall setzen die Administratoren alle Einstellungen auf dem Controller und diese werden dann an alle Rechner im AD geschickt. Bei Einzelsystemen können unsere Administratoren die Regeln per PowerShell Befehl umsetzen. Viele Unternehmen reduzieren diesen Aufwand, indem sie auf die Dokumentation und die Checks verzichten; die Regeln werden im Controller gesetzt, aber es gibt keine Dokumentation dazu. Oder man einigt sich auf ein Format pro Anwendungsfall; das Verwalten dreier Dokumente ist zwar mühsam, aber für die meisten dennoch einfacher zu meistern, als einen neuen Prozess einzuführen.

Folgerung: Im Umkehrschluss ergeben sich für uns die Faktoren, die für eine Anwendung des Scapolite Ansatzes sprechen. Je mehr Faktoren auf ihr Unternehmen oder ihre Strategie zur Systemhärtung zutreffen, desto eher ist der Scapolite-Ansatz für Sie geeignet. Die Faktoren sind: **Anpassungen:** Sie erstellen eine Richtlinie oder passen eine existierende Richtlinie an anstatt nur die Richtlinie einer externen Instanz zu verwenden. **Teamarbeit:** Mehrere Autoren arbeiten an Ihrer Richtlinie. Eventuell sind die Autoren sogar gar nicht Teil Ihres Sicherheitsteams, sondern Administratoren, die Änderungsvorschläge an der Richtlinie auf Basis von praktischen Erfahrungen einbringen möchten. **Kontinuierliche Verwaltung:** Sie arbeiten kontinuierlich an Ihrer Richtlinie, anstatt sie einmal zu erstellen. Hierbei wollen Sie Änderungen möglichst direkt auf semantische Fehler untersuchen anstatt diese erst später im Prozess beheben zu müssen. **Heterogene Umgebungen:** Sie müssen auf verschiedene Umgebungen oder Sicherheitsstufen eingehen. Zudem stellen Sie die Richtlinie für verschiedene Abteilungen Ihrer Firma zur Verfügung, ihrerseits die die Richtlinie leicht anpassen, aber auch Updates für die Richtlinie allgemein erhalten wollen. **Heterogene Werkzeuge:** Sie müssen den Administratoren verschiedene Möglichkeiten zur Überprüfung oder Umsetzung bieten.

Bei Siemens waren alle Faktoren gegeben und Autoren und Administratoren nahmen den neue Ansatz positiv. Wir verwalten jetzt auch andere Siemens-Richtlinien im Scapolite-Ansatz, z. B. zu WS2019 oder Office16. Seit neustem übertragen wir den Ansatz auch auf Linux-basierte Systeme und verwalten

[14] `https://www.tenable.com/products/nessus`

so die Richtlinien zu SUSE 15, Debian 10 und Red Hat 8. Zusätzlich importieren wir auch CIS oder DISA Richtlinien in das Scapolite-Format und speichern diese in eigenen Repositories. Obwohl wir hier nichts anpassen, erleichtert dies das Vergleichen von verschiedenen Versionen einer CIS Richtlinie enorm. So können wir unsere eigenen Richtlinien noch schneller als früher an neue Erkenntnisse anpassen und unsere Systeme immer besser sichern. Da der Scapolite-Ansatz mittlerweile in großem Maße bei Siemens eingesetzt wird, sehen wir diese qualitative Fallstudie als Erfolg.

6 Verwandte Arbeiten

Das OpenScap-Projekt von Red Hat kommt unserem Ansatz am nächsten [4, 5]. Red Hat verwaltet seine Richtlinien für verschiedene Linux-Systeme in einem git-Repository, in dem jede Regel durch eine Datei repräsentiert wird. Die Datei enthält Verweise auf andere Dateien, die Artefakte für die automatische Implementierung und Überprüfung enthalten. Allerdings gibt es keine OpenScap-Richtlinien für Windows und keine Pläne, dies zu ändern. Außerdem sind OpenScap Regeln auf YAML beschränkt, während wir auf YAML für strukturierte Daten und Markdown für formatierten Text zurückgreifen. Im OpenScap-Ansatz verlinken die einzelnen Regeln auf die Skripte zur Umsetzung oder Überprüfung. Dies macht zwar die Wiederverwendung einfacher, erhöht aber das Risiko von Inkonsistenzen.

Für das Härten von W10 ist vor allem das HardeningKitty [6] relevant. Administratoren können mit diesem Werkzeug Microsoft-, CIS- oder DISA-Richtlinien überprüfen und umsetzen. Das HardeningKitty ist wie unser Ansatz aus dem Wunsch entstanden, Systeme nach gegebenen Richtlinien zu konfigurieren. Da das HardeningKitty erst am Ende unseres Prozesses ansetzt, kann man es als Ergänzung zum Scapolite-Ansatz sehen, z. B. indem wir in Regeln hinterlegen, welche HardeningKitty ID dieser Regel entspricht. Damit können wir dann beim Generieren HardeningKitty-kompatible Skripte als Alternative zu unseren Skripten generieren.

7 Ausblick

Wir sehen die Zukunft der Richtlinien als Open-Source-Projekte. Die bereitstellende Stelle, z. B. das CIS, agiert hierbei als Moderator und verwaltet die Regeln. Auf Basis der Scapolite-Dateien generieren wir mit einer CI die jeweiligen Artefakte. Ist nur die Verwendung gewünscht, können diese heruntergeladen werden. Möchten Unternehmen ihre eigene Richtlinie erstellen, erstellen sie einen Fork. Auf diesem können sie eigenen Änderungen einbringen, aber immer wieder Updates der Hauptrichtlinie zu sich nachziehen. Falls eine Firma eine Änderung, z. B. eine zusätzliche Regel, in die Hauptrichtlinie integrieren will, erstellt sie lässt einen Merge Request. Die verantwortliche Stelle kann die Änderungen genehmigen oder ablehnen. Administratoren können Probleme als Issues erstellen oder in den vorhandenen nach Lösungen suchen. Idealerweise kombinieren wir diesen Ansatz mit offener Software, da wir so überprüfen können, dass kein Bug verhindert, dass die Einstellung den gewünschten Effekt hat. Bei offener Software können wir im Code überprüfen, ob und wie die Einstellung verwendet wird. Auf diese Weise können wir dafür sorgen, dass immer mehr Systeme sicher konfiguriert sind und wenigstens diese Angriffsflanke schließen.

8 Schlussfolgerung

Wenn ein Unternehmen eine sichere Infrastruktur wünscht, ist eine sichere Konfiguration der Systeme unabdingbar. Wie wir oben dargestellt haben, ergeben sich aber beim bisherigen Prozess zur sicheren Konfiguration diverse Probleme. Um diese Probleme anzugehen, haben wir den Scapolite-Ansatz entwickelt. Wir konnten diesen Ansatz im Rahmen unserer Fallstudie bei der Siemens AG erfolgreich von der Theorie in die Praxis übertragen. Durch unsere Werkzeuge und Prozesse ist es den Administratoren bei Siemens nun möglich, ihre Systeme schneller und effektiver zu härten. Um zu prüfen, ob der Scapolite-Ansatz für die eigene IT-Infrastruktur sinnvoll ist, wurden im Verlauf der Fallstudie Faktoren ermittelt, die Aufschluss darüber geben.

Aufgrund der beschriebenen Erfahrungen sind wir überzeugt, dass der Scapolite-Ansatz auch in anderen Institutionen erfolgreich einzusetzen ist. Wie wir im Ausblick bereits skizziert haben, sehen wir die Zukunft in öffentlichen und automatisierten Richtlinien. Wir sind der Ansicht, dass die IT-Sicherheit in vernetzten Systemen auf diese Weise deutlich gestärkt wird.

Literatur

[1] BAKER, J., HANSBURY, M., AND HAYNES, D. Open Vulnerability and Assessment Language, MITRE CORPORATION. Available from http://oval.mitre.org/language/version5.11/oval-language-specification_12-18-2014.pdf.

[2] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK (BSI). Studie zu Systemintegrität, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10. https://www.bsi.bund.de/DE/Service-Navi/Publikationen/Studien/SiSyPHuS_Win10/SiSyPHuS_node.html.

[3] DIETRICH, C., KROMBHOLZ, K., BORGOLTE, K., AND FIEBIG, T. Investigating system operators' perspective on security misconfigurations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2018), CCS '18, ACM, pp. 1272–1289.

[4] PREISLER, M., AND HAICMAN, M. Security Automation for Containers and VMs with OpenSCAP. In *USENIX LISA* (Washington, 2018). Available from martin.preisler.me/wp-content/uploads/2018/03/LISA17-Tutorial-Security-Compliance-for-Containers-and-VMs-with-OpenSCAP.pdf.

[5] RED HAT, INC. OpenSCAP. https://www.open-scap.org, 2010. Accessed: 2021-11-17.

[6] SCHNEIDER, M. HardeningKitty. https://github.com/0x6d69636b/windows_hardening, 2017. Accessed: 2021-11-17.

[7] STÖCKLE, P., GROBAUER, B., AND PRETSCHNER, A. Automated Implementation of Windows-Related Security-Configuration Guides. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering* (New York, NY, USA, 2020), ASE '20, Association for Computing Machinery, p. 598–610.

[8] WALTERMIRE, D., QUINN, S., BOOTH, H., AND SCARFONE, K. The Technical Specification for the Security Content Automation Protocol (SCAP): Scap version 1.3. Tech. rep., NIST, 2018. Available from https://csrc.nist.gov/publications/detail/sp/800-126/rev-3/final.

[9] WALTERMIRE, D., SCHMIDT, C., SCARFONE, K., AND ZIRING, N. Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2. Tech. rep., NIST, 2012. Available from https://csrc.nist.gov/CSRC/media/Publications/nistir/7275/rev-4/final/documents/nistir-7275r4_updated-march-2012_clean.pdf.