



Große Cloud-Umgebungen (automatisiert) prüfen

Cloud-Umgebungen wie Microsoft Azure, Amazon Web Services (AWS) oder Google Cloud Platform (GCP) bieten eine Fülle von Einstellungen zur Konfiguration und Absicherung. Das birgt die Gefahr, notwendige Schutzmaßnahmen zu übersehen. Bei deren Aufspüren helfen Prüftools wie ScoutSuite oder Prowler.

Von Viktor Rechel

■ „Die Cloud“ ist noch immer für viele Unternehmen die erste Wahl, wenn es darum geht, ihre Systeme effizient auszulagern, Ressourcen zu sparen, auf externes Know-how zu setzen und eine sicherere Umgebung zu betreiben. Vor allem im Hinblick auf Sicherheitsaspekte vertrauen die Unternehmen dabei häufig auf die Betreiber der Cloud-Plattform und deren Know-how in Sachen sicherer Betrieb von IT-Umgebungen.

Das Shared Responsibility Model beschreibt jedoch die Problematik, dass bei Cloud-Umgebungen von Anbietern wie Microsoft Azure, Amazon Web Services (AWS), Google Cloud Platform (GCP) und anderen nicht gewährleistet ist, dass der Betreiber sein (ungewisses) Know-how auch für die Absicherung der Kundenumgebungen einbringt (siehe Artikel „Microsoft-365-Audits mit Maester“ ab Seite 14). Stattdessen stehen Unternehmen bei den eigenen Komponenten auch selbst in der Pflicht. Bei Cloud-Umge-

bungen und den darin enthaltenen Funktionen gilt nicht immer das Security-by-Default-Prinzip.

Eigenverantwortliche Prüfungen müssen sicherstellen, dass die Konfiguration ausreichend sicher gewählt ist. Die Ansprüche des Unternehmens und der Prozesse können sich unterscheiden, wodurch eine eigene Bewertung notwendig ist. Für Cloud-Umgebungen müssen

mindestens die gleichen Ansprüche wie bei internen Systemen gelten – im Zweifel aufgrund der exponierten Lage auch höhere Ansprüche.

Baseline für Prüfungen und manuelle Verifizierung

Jede Prüfung muss eine Grundlage haben. Manchmal ist das einfach die eigene Erfahrung, idealerweise richtet sich eine Prüfung aber nach einer allgemeingültigen Best Practice. Gängige Beispiele dafür sind (je nach Thema) unter anderem die CIS-Benchmarks des Center for Internet Security (CIS), die Azure Security Benchmarks oder die AWS Startup Security Baseline (diese und alle weiteren Links siehe ix.de/zp5z).

Diese Best Practices lassen sich grundsätzlich auch ohne Automatisierung und ohne spezielle Tools heranziehen. Wenn beispielsweise spezifische Komponenten der eigenen Umgebung betrachtet oder gehärtet werden sollen, kann auch eine manuelle Nutzung von Best Practices sinnvoll sein. Sobald mehrere unterschiedliche Komponenten oder auch viele gleiche Komponenten ins Spiel kommen, wird die Automatisierung schnell unerlässlich.

Automatisierte Prüfung

Je nach Umgebung können spezifische Tools wie das PowerShell-Framework Maester bei M365 und Azure zum Einsatz kommen (siehe Artikel „Microsoft-365-Audits mit Maester“ ab Seite 14), nicht jede Umgebung erfordert aber ein eigenes Tool. Anwendungen wie ScoutSuite und Prowler eignen sich auch für Prüfungen auf verschiedenen Cloud-Plattformen. Anhand einer vorbereiteten AWS-Beispielumgebung werden beide Werkzeuge nachfolgend einzeln vorgestellt und die Vorgehensweise wird verglichen. Zur Vereinfachung ist die Umgebung bewusst klein gehalten, sie enthält für die Prüfungen lediglich eine Datenbank, eine

X-TRACT

- Unternehmen müssen trotz der Sicherheitsmaßnahmen von Cloud-Anbietern wie AWS, Azure oder GCP selbst Verantwortung für die Absicherung ihrer Cloud-Umgebungen übernehmen.
- Automatisierung ist unverzichtbar. Tools wie ScoutSuite und Prowler helfen, Schwachstellen aufzudecken.
- Während ScoutSuite für kleinere Umgebungen und erste Prüfungen geeignet ist, empfiehlt sich Prowler für komplexere Set-ups mit mehreren Cloud-Providern und regelmäßigen Audits.

Root Account without Hardware MFA

Description

The root account is the most privileged user in an account. MFA adds an extra layer of protection on top of a user name and password. With MFA enabled, when a user signs in to an AWS website, they're prompted for their user name and password and for an authentication code from their AWS MFA device.

Remediation

Enable MFA for the root account

Compliance

- CIS Amazon Web Services Foundations version 1.0.0, reference 1.13
- CIS Amazon Web Services Foundations version 1.1.0, reference 1.14
- CIS Amazon Web Services Foundations version 1.2.0, reference 1.14

References

- <https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-cis-controls.html#securityhub-cis-controls-1.13>
- https://docs.aws.amazon.com/organizations/latest/userguide/orgs_best-practices_mgmt-acct.html#best-practices_mgmt-acct_mfa

Accounts checked: 3
Accounts flagged: 1

ScoutSuite-Befunde listen stets eine Beschreibung des Problems, Möglichkeiten zur Behebung sowie Referenzen zu CIS-Benchmarks und Herstellerinformationen auf – wie das Beispiel „Root Account without Hardware MFA“ zeigt (Abb. 1).

EC2-Instanz, eine Anwendung über die AppRegistry, zwei IAM-Benutzer und anhängige Infrastruktur (wie etwa zwei Security Groups bei EC2). Scans liefern aber selbst in dieser kleinen Beispielumgebung schon zahlreiche Befunde, was die Notwendigkeit unterstreicht, sich eingehend mit automatisierten Sicherheitsprüfungen auseinanderzusetzen.

ScoutSuite

ScoutSuite ist Open Source und steht damit kostenlos zur Verfügung. Lizenztechnisch steht einem Einsatz zur Verifizierung der eigenen Umgebung somit nichts im Wege. Auch aus sicherheitstechnischer Sicht besteht kein erhöhtes Risiko, da sämtliche bei den Scans erhobenen Daten ausschließlich lokal verarbeitet werden und nicht nach außen gelangen. Entwickelt und gepflegt wird ScoutSuite durch das unter anderem im Bereich Cybersecurity aktive Unternehmen NCC Group.

Mit ScoutSuite lassen sich Umgebungen in AWS, GCP, Azure, Alibaba Cloud, Oracle Cloud Infrastructure, Kubernetes und DigitalOcean prüfen. Die initiale Installation erfolgt gemäß den Angaben im zugehörigen GitHub-Repository auf einem Linux-System, alternativ lässt sich auch das Windows-Subsystem für Linux (WSL) nutzen. Nach der Installation steht ScoutSuite inklusive aller notwendigen Abhängigkeiten als Python-Skript zur Verfügung.

Zur Verwendung benötigt ScoutSuite noch die erforderlichen Zugangsdaten für die Umgebung. Ähnlich wie bei Maester gilt auch hier: „Nur gucken, nicht anfassen!“ ScoutSuite führt Prüfungen durch,

Root Account without Hardware MFA

Show all

<root_account>

<root_account>

Credentials Report

Creation Date: Wed Aug 06 2025 09:50:29 GMT+0200 (Central European Summer Time)
Last Used Date: Tue Aug 26 2025 23:48:50 GMT+0200 (Central European Summer Time)
Password Enabled: true
Password Last Used: Tue Aug 26 2025 23:48:50 GMT+0200 (Central European Summer Time)
Password Last Changed: Mon Aug 11 2025 19:44:12 GMT+0200 (Central European Summer Time)
MFA Active: true
Hardware MFA Active: **false**
Access Key 1 Active: false
Access Key 1 Last Used: No date available
Access Key 1 Last Rotated: No date available
Access Key 2 Active: false
Access Key 2 Last Used: No date available
Access Key 2 Last Rotated: No date available
Signing Cert 1 Active: false
Signing Cert 2 Active: false

Informationen zu einer betroffenen Ressource beim Testfall „Root Account without Hardware MFA“. Je Testfall lassen sich alle betroffenen Ressourcen inklusive ihrer – hinsichtlich des Tests relevanten – Konfigurationen einsehen (Abb. 2).

ändert aber die Konfigurationen nicht eigenständig. Trotzdem sollten nur die (lesenden) Zugriffe eingerichtet werden, die notwendig sind. Für ein Proof of Concept lassen sich (in AWS) die Rollen „ReadOnly Access“ und „SecurityAudit“ nutzen. Für den dauerhaften Einsatz ist hingegen eine Einschränkung auf die je nach Umgebung explizit notwendigen Lese-rechte empfehlenswerter, da die zentralen Rollen oft übermäßig umfangreiche Berechtigungen beinhalten.

Die konkrete Authentisierung unterscheidet sich je nach Umgebung. Bei AWS lassen sich beispielsweise Parameter im CLI-Befehl, über Umgebungsvariablen oder die AWS-Profilen nutzen. In Azure und GCP wird das zugehörige Schlüsselmaterial entweder per CLI oder per Datei übergeben.

Um beim Beispiel AWS zu bleiben: Die einfachste Variante ist die Übergabe beim

Aufruf von ScoutSuite (mit den Parametern `--access-key-id` und `--secret-access-key`). Die sicherere Variante, um auch Schlüsselmaterial unter anderem in Skripten zu verhindern, wären Umgebungsvariablen. Dazu dienen die Befehle `export AWS_ACCESS_KEY_ID=` und `export AWS_SECRET_ACCESS_KEY=`. ScoutSuite verwendet die Umgebungsvariablen dann automatisch. Eine dritte Möglichkeit sind Dateien, die AWS-Profilen darstellen und sich auch für die AWS-CLI nutzen lassen. Profile sind immer dann notwendig, wenn für die Scans mehrere Benutzer mit unterschiedlichen Berechtigungen erforderlich sind.

Vor dem Start eines Scandurchlaufs mit der Standardkonfiguration über das Skript `python scout.py aws` sollten also die Umgebungsvariablen definiert werden. Ohne weitere Konfiguration kommt dann das Regelwerk zum Einsatz, das die

NCC Group als Standard hinterlegt hat. ScoutSuite prüft daraufhin verschiedene Maßnahmen, die Sicherheitsprobleme verursachen oder begünstigen. Alternativ lassen sich auch andere vordefinierte Regelwerke nutzen, beispielsweise auf Basis des zugehörigen CIS-Benchmarks. Enthält das Regelwerk Prüfungen für Komponenten, die in der untersuchten Umgebung nicht enthalten sind, führt dies zwar zu Fehlermeldungen, der Scan läuft aber weiter. Nach dem Scandurchlauf generiert ScoutSuite automatisch einen Bericht im HTML-Format, legt ihn im ScoutSuite-Verzeichnis im Ordner scoutsuite-report

ab und öffnet ihn. Der Bericht enthält alle durchgeführten Tests. In der kleinen Testumgebung sind es insgesamt 1.045 Einzelprüfungen.

Die Ergebnisse sind nach den Komponentenarten der entsprechenden Umgebung strukturiert. Die Einzelergebnisse der Kategorien lassen sich in spezifischen Dashboards aufrufen, entweder durch Wahl des Service in der Top-Navigation der ScoutSuite oder per Klick auf den Service im allgemeinen Dashboard.

Beispielsweise findet sich im IAM-Bereich (im Header unter Security) ein Befund, dass der Root-Account nicht mit

einem Hardwaretoken geschützt wird (Abbildung 1). Jeder Befund umfasst eine Beschreibung, einen kurzen Hinweis zur Behebung, Referenzen zu den zugrunde liegenden Best Practices und Referenzen zu weiteren Informationen (beispielsweise von AWS). Zur weiteren Verifizierung von Scanergebnissen stellt ScoutSuite auch detailliertere Informationen zu den betroffenen Komponenten sowie zur Konfiguration der Komponente zum Zeitpunkt des Scans bereit (Abbildung 2). Bei Bedarf kann man die Liste der betroffenen Ressourcen in der Übersicht auch als CSV- oder JSON-Datei exportieren, um sie gegebenenfalls zum leichteren Nachverfolgen von Maßnahmen in andere Anwendungen wie etwa Ticketsysteme zu importieren.

Bei den Referenzen auf die CIS-Benchmarks fällt eine Problematik auf. Der gezeigte Testfall verweist auf die Version 1.2.0 als den aktuellen Benchmark. Diese Version stammt noch von 2018. Die aktuelle Version des Benchmarks ist 5.0.0 (aus dem März 2025). Das Problem der fehlenden Aktualität (zum CIS-Benchmark) zeigt sich auch im Regelwerk `cis-1.2.0.json`, das eigentlich auch auf die Konformität mit dem CIS-Benchmark prüfen sollte. Auch dieses referenziert nur die Version 1.2.0. Das ist ein Problem, wenn eine explizite Konformitätsprüfung durchgeführt werden soll. Es spielt aber bei den eigentlichen Scans keine Rolle, wenn diese nicht aus Compliancegründen gemacht werden sollen. Die Testfälle und damit der konkrete Inhalt sind aktuell und relevant.

Wem das Regelwerk zu umfangreich erscheint, der kann es individuell anpassen und irrelevante Regeln außer Kraft setzen. Hierfür empfiehlt es sich, eine Kopie des Standardregelwerks zu erstellen und diese zu bearbeiten. Die Regelwerke finden sich in Unterordnern von ScoutSuite (beispielsweise `/home/<Benutzer>/ScoutSuite/ScoutSuite/providers/aws/rules/rulesets/`). Im Regelwerk sind alle definierten Testfälle beschrieben, die bei einem Scan durchgeführt werden. Pro

Listing 1: Das von ScoutSuite verwendete Regelwerk

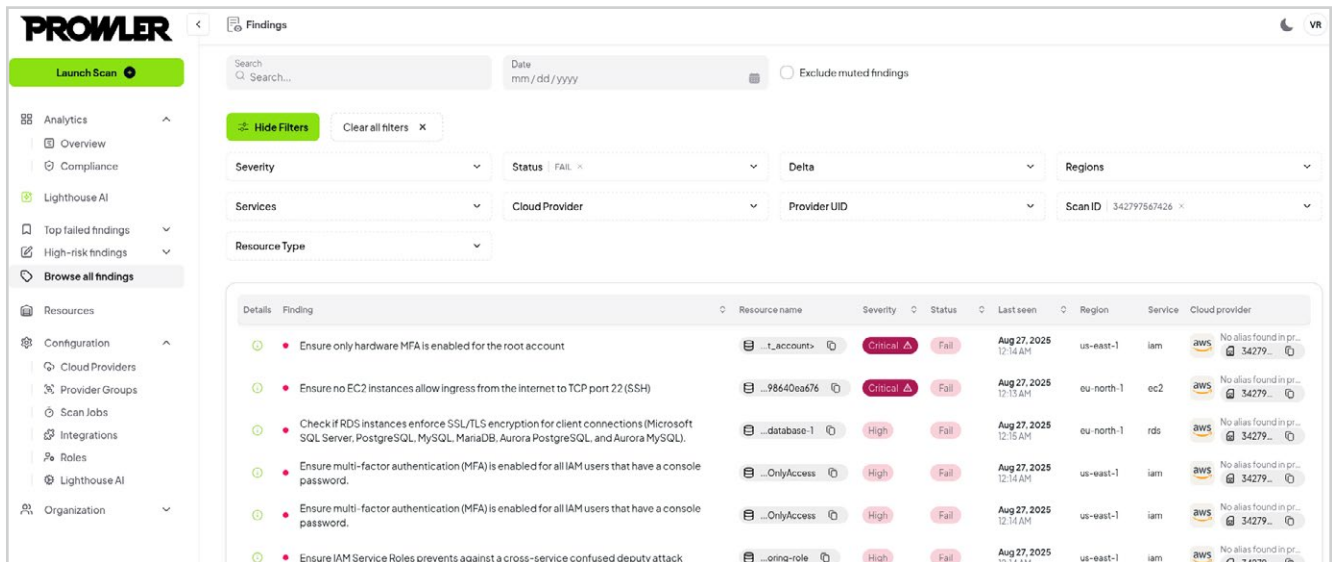
```
{
  "about": "This ruleset consists of numerous rules that are considered a
  standard by NCC Group. The rules enabled range from violations of well-known
  security best practices to gaps resulting from less-known security
  implications of provider-specific mechanisms. Additional rules exist,
  some of them requiring extra-parameters to be configured, and some of
  them being applicable to a limited number of users.",
  "rules": {
    "acm-certificate-with-close-expiration-date.json": [
      {
        "args": [
          "7"
        ],
        "enabled": true,
        "level": "warning"
      }
    ],
    "acm-certificate-with-transparency-logging-disabled.json": [
      {
        "enabled": true,
        "level": "warning"
      }
    ],
    "cloudformation-stack-with-role.json": [
      {
        "enabled": true,
        "level": "danger"
      }
    ],
    "cloudfront-distribution-clear-text-origin.json": [
      {
        "enabled": true,
        "level": "warning"
      }
    ]
  }
}
```

Listing 2: Vereinfachte Regel als Beispiel einer selbst definierten Regel

```
{
  "description": "Test: Minimum Password Length Too Short",
  "rationale": "Test: The password policy did not enforce a minimum of
  10 characters. As a result, password complexity requirements were
  not in line with security best practice.",
  "dashboard_name": "Password policy",
  "path": "iam.password_policy.MinimumPasswordLength",
  "conditions": [
    "or",
    [
      "this",
      "lessThan",
      "10"
    ]
  ]
}
```

Listing 3: Kopiertes Regelwerk mit einem einzigen Testfall

```
{
  "about": "This is a PoC ruleset",
  "rules": {
    "iam-testrule.json": [
      {
        "enabled": true,
        "level": "warning"
      }
    ]
  }
}
```



Prowler listet standardmäßig die fehlgeschlagenen Testfälle für alle Scans. Die Ansicht lässt sich über Filter anpassen (Abb. 3).

Testfall legt der Parameter `enabled` fest, ob er durchgeführt (`true`) oder ignoriert wird (`false`). Mit der Eigenschaft `level` lässt sich definieren, welche Einstufung der Befund hat, wenn der Testfall in der eigenen Umgebung fehlschlägt. Hier ist zwischen `warning` und `danger` zu unterscheiden, wie in Listing 1 dargestellt.

Die Parameter lassen sich in der kopierten Version des Regelwerks (im Ordner `rulesets`) beliebig bearbeiten. Wel-

ches Regelwerk zum Einsatz kommen soll, lässt sich beim Python-Aufruf mit dem Parameter `--ruleset` festlegen – beispielsweise also `python scoutsuite aws --ruleset test.json`.

Wer sich nicht nur darauf beschränken möchte, vordefinierte Regeln zu aktivieren oder zu deaktivieren, der kann auch eigene Regeln hinzufügen. Die Regeln sind nach der bekannten booleschen Algebra aufgebaut und lassen sich über `and`, `or`

und ähnliche Verknüpfungen verbinden (Listing 2). Kerninhalt sind die zu prüfenden Parameter, die automatisiert abgefragt werden sollen. Die Parameter, die Verknüpfungen und die Beschreibungen zum Testfall sind bei ScoutSuite in JSON-Dateien abgelegt. Die Testfälle liegen unter <Ordner von ScoutSuite>/ScoutSuite/providers/aws/rules/findings/.

Ein individueller, neuer Testfall lässt sich durch Erstellen einer neuen JSON-

Datei oder Kopieren und Anpassen eines bestehenden Testfalls generieren. Damit der Testfall auch angewendet wird, muss er in einem Regelwerk hinterlegt sein. Die dafür notwendige Anpassung zeigt Listing 3. Um den neuen Testfall vorab testen zu können, empfiehlt es sich, eigens dafür ein separates Regelwerk mit ausschließlich dem neu erzeugten Testfall zu erstellen. Damit ScoutSuite nicht bei jedem Test die Daten neu von AWS abrufen muss, lässt sich über den Parameter `--local` der Zugriff auf die gecachten Daten einrichten, was den Prozess beschleunigt.

Prowler

Wer es etwas bunter, professioneller und insbesondere im Hinblick auf die CIS-Benchmarks aktueller möchte, dem steht Prowler zur Auswahl. Hinter dem Open-Source-Werkzeug steht nicht nur die Community, sondern die ProwlerPro, Inc. mit Sitz in den USA. Die Nutzung von Prowler als eigene Instanz ist kostenlos. Für die alternative SaaS-Lösung Prowler Cloud fallen hingegen Gebühren an. Die

nachfolgende Beschreibung zum Einsatz von Prowler beschränkt sich auf die selbst betriebene Open-Source-Variante.

Prowler arbeitet neben AWS, Azure, GCP, Kubernetes und Microsoft 365 auch mit GitHub zusammen. Alle unterstützten Umgebungen können auf Fehlkonfigurationen oder Verbesserungspotenziale untersucht werden. In GitHub wird beispielsweise auf die Absicherung der Repositories, Branches und allgemeinen Prozesse geachtet. Nachfolgend stehen aber wieder Cloud-Umgebungen am Beispiel von AWS im Fokus.

Am einfachsten gelingt die Erstinstallation von Prowler aus dem entsprechenden Docker-Container – gemäß den Angaben im Prowler-GitHub-Repository. Nach der Installation stehen zwei Prowler-Varianten zur Verfügung. Das Tool lässt sich wie ScoutSuite über die CLI direkt ansprechen, bietet aber einen größeren Funktionsumfang. Sind Zugangsdaten hinterlegt (auch hier über Umgebungsvariablen möglich), lassen sich Scans direkt über die CLI ausführen. Die erzeugten Berichte liegen standardmäßig im HTML-Format vor und zusätzlich als CSV

und JSON. Die Nutzung der CLI hat prinzipbedingt allerdings ihre Grenzen. Das regelmäßige Prüfen von Cloud-Umgebungen – insbesondere mehreren parallelen – gelingt einfacher mit der zweiten Variante: dem Prowler-Webportal.

Prowler bringt eine eigene Webanwendung mit, die das Verwenden und Steuern deutlich vereinfacht. Die Webanwendung übernimmt die Kommunikation mit der CLI (über eine zwischengelagerte API), was den Aufwand für Benutzer senkt. Die Weboberfläche von Prowler ist standardmäßig über `localhost:3000` erreichbar. Nach der Installation ist das Dashboard zunächst noch leer. Relevante Informationen zu durchgeführten Scans, geprüften Ressourcen und den aktuellen Befunden stellt das Dashboard erst dar, nachdem alle Cloud-Provider mit ihren jeweiligen Zugangsdaten über den Button „Add Provider“ hinzugefügt und konfiguriert sind.

Bei der Konfiguration eines Cloud-Providers hinterlegt man wieder entsprechendes Schlüsselmaterial. Im Fall von AWS kann der in Prowler hinterlegte Benutzer entweder dauerhaft die Berechtigung

Ensure only hardware MFA is enabled for the root account

FAIL

Finding Details

aws	34279...	iam	us-east-1	Aug 27, 2025 12:14 AM	New	Critical
-----	----------	-----	-----------	-----------------------	-----	----------

Finding ID

0198e872-8697-7862-8d02-926cdd2a2de6

Check ID

iam_root_hardware_mfa_enabled

Finding UID

prowler-aws-iam_root_hardware_mfa_enabled-342797567426-us-east-1-<root_account>

Risk

The root account is the most privileged user in an AWS account. MFA adds an extra layer of protection on top of a user name and password. With MFA enabled when a user signs in to an AWS website they will be prompted for their user name and password as well as for an authentication code from their AWS MFA device. For Level 2 it is recommended that the root account be protected with only a hardware MFA.

Description

Ensure only hardware MFA is enabled for the root account

Status Extended

Root account has a virtual MFA instead of a hardware MFA device enabled.

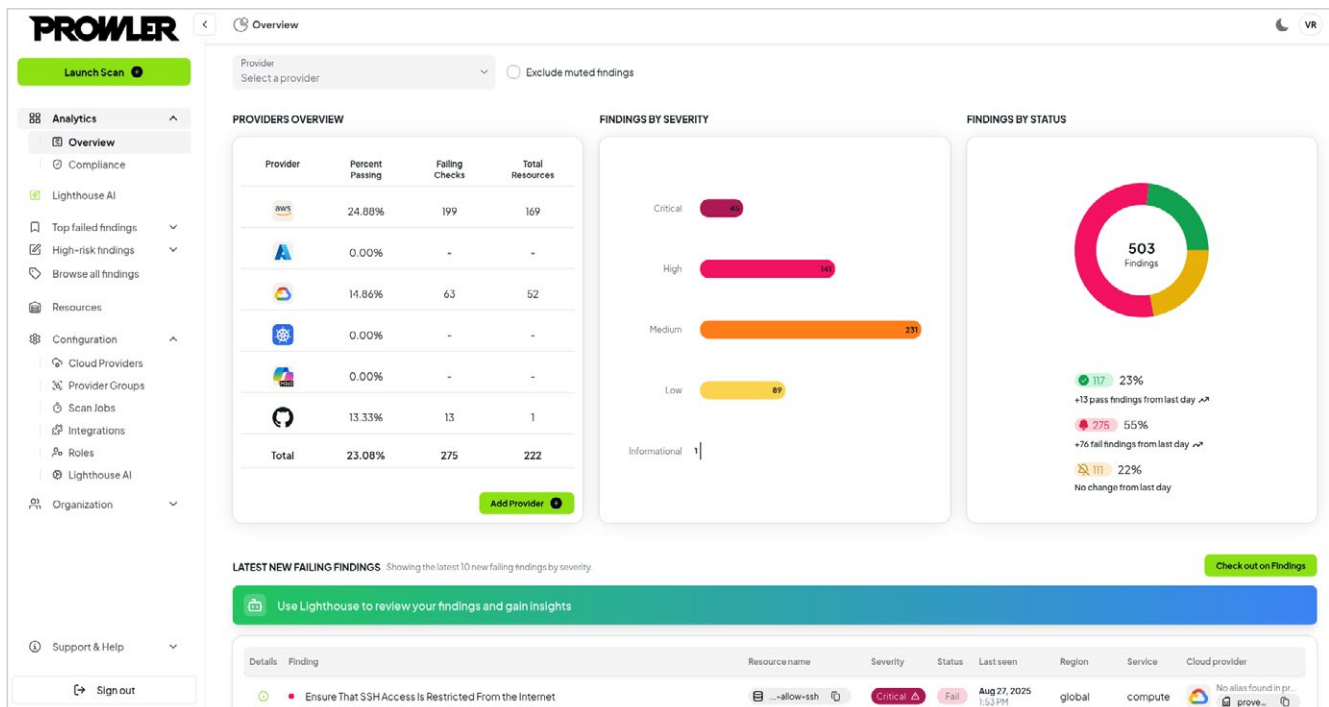
Remediation Details

Recommendation

Using IAM console navigate to Dashboard and expand Activate MFA on your root account. If using AWS Organizations, consider enabling Centralized Root Management and removing individual root credentials.

Learn more

Beschreibung zum Befund eines AWS-Scans, dass der Root-Account keine Multi-Faktor-Authentifizierung aktiviert hat. Prowler und ScoutSuite liefern hierzu das gleiche Ergebnis (Abb. 4).



Nach einem Scan präsentiert das Dashboard von Prowler alle relevanten Informationen übersichtlich strukturiert (Abb. 5).

gungen haben oder – aus Sicherheitsaspekten die bessere Lösung – der Benutzer lässt sich die Rollen automatisch (für den Zeitraum der Prüfungen) zuweisen. Für den zweiten Fall muss die AWS-Umgebung entsprechend vorbereitet werden. Hierzu legt man in der Managementkonsole eine Rolle an, die (wie für ScoutSuite) die Berechtigungen „ReadOnly Access“ und „SecurityAudit“ erhält. Zusätzlich muss ein Benutzer angelegt werden, den Prowler verwenden soll. Damit die automatische Zuweisung der Rolle funktioniert, muss der Amazon Resource Name (ARN) des Benutzers in der erstellten Rolle als Principal in einer Vertrauensbeziehung hinterlegt sein.

Die Erstellung des Cloud-Providers endet direkt mit einem Scan, der auch gleich als regelmäßiger Scan konfiguriert werden kann. Soll ein Scan regelmäßig ausgeführt werden, so wird er standard-

mäßig alle 24 Stunden durchlaufen. Diese Regelmäßigkeit kann nicht einfach über die Webanwendung angepasst werden. Ist ein Scan beendet, erhält man eine Übersicht über alle Ergebnisse, indem man über das Menü auf die Übersicht aller Befunde oder zuerst auf die Übersicht der Scan-Jobs und dann auf die Übersicht der spezifischen Befunde wechselt. In beiden Fällen landet man auf einer (wahrscheinlich nicht zu kurzen) Liste von Ergebnissen, die beim Scan identifiziert wurden (Abbildung 3). Die Befunde werden auch bei Prowler jeweils einzeln beschrieben. Die Einstufung orientiert sich hier aber an der in Sicherheitsprüfungen üblichen Unterteilung informational, low, medium, high und critical statt der groben, dreistufigen Unterscheidung von ScoutSuite.

Auch bei Prowler erhält man zu jedem Befund eine kurze Beschreibung und

eine Empfehlung zur Behebung sowie die Zuordnung zu den betroffenen Komponenten. In der Regel wird bei den Behebungshinweisen auch direkt auf die Dokumentation von AWS verwiesen, was die Behandlung vereinfacht. Prowler hebt zusätzlich die Risikoeinschätzung und damit das eigentliche Problem noch einmal gesondert hervor (Abbildung 4). Bei der Übersicht der Ergebnisse zeigt sich ein Problem von Prowler: Wo ScoutSuite Ergebnisse grundsätzlich konsolidiert und fehlgeschlagene Regeln als einzelne Fehler darstellt, zeigt Prowler mehrfache Befunde auch mehrfach an. Dies erschwert die Übersicht, vor allem nach einem initialen Scan, bei dem eher mehr als weniger Befunde auftreten.

Zur effizienten Behandlung erkannter Problemstellen lässt sich theoretisch auch eine Besonderheit der Prowler-CLI verwenden: der Prowler Fixer. Zu den stan-

Configure Mutelist

- This Mutelist configuration will take effect on the next scan.
- Mutelist configuration can be modified at anytime on the Providers and Scans pages.
- Learn more about configuring the Mutelist [here](#).
- A default Mutelist is used, to exclude certain predefined resources, if no Mutelist is provided.

Mutelist Configuration

```

Mutelist:
  Accounts:
    '*':
    Checks:
      ec2_*:
        Regions:
          - '*'
        Resources:
          - '*'
        Description: Mute all EC2-related findings across all accounts and
regions
      iam_root_hardware_mfa_enabled:
        Regions:
          - '*'
        Resources:
          - '*'
        Description: Mute hardware MFA finding

```

Cancel
Update

Delete Configuration

Mutelist des AWS-Cloud-Providers, die alle Checks bei EC2-Instanzen sowie den Check auf Hardware-MFA beim Root-Account mutet (Abb. 6).

dardmäßigen Regeln sind stellenweise Festlegungen zur automatisierten Behebung des Problems hinterlegt. Diese können (wenn gewünscht) über den Parameter `--fixer` automatisiert angewendet werden. Zum Zeitpunkt der Entstehung dieses Artikels war es jedoch nicht möglich, die Funktion zu nutzen, auch wenn die neueste CLI-Version (Version 3.11.3) zum Einsatz kam.

Ein Vorteil von Prowler ist, dass sich mehrere Cloud-Instanzen gleichzeitig verwalten lassen (Abbildung 5), wenn in der Praxis beispielsweise Office-Anwendungen auf M365 und der Systembetrieb auf AWS laufen. Mit Prowler lassen sich beide Umgebungen regelmäßig prüfen oder auch mehrere AWS-Umgebungen gleichzeitig im Blick behalten, ohne dafür unterschiedliche Tools einführen zu müssen.

Sind die ersten Versuche in Prowler geglückt und die gewonnenen Informationen erweisen sich als nützlich, lassen sich über die Organisation in Prowler weitere Benutzerinnen und Benutzer hinzufügen. Standardmäßig ist nur die mit umfassenden Rechten ausgestattete Adminrolle vorkonfiguriert. Soll ein Benutzer jedoch lediglich Scans ausführen und Befunde einsehen können, bietet sich die Defi-

nition einer neuen Rolle an, die nur für den Lesezugriff eingerichtet ist. Als passende Berechtigung für diese Rolle ist „Unlimited Visibility“ zu wählen.

Und zum Abschluss auch hier der Hinweis zur Anpassungsfähigkeit: Es besteht keine Möglichkeit zur Anpassung des Regelwerks, ohne direkt in den Code einzugreifen. Zumindest kann man aber irrelevante Testfälle stumm schalten und pro Cloud-Provider eine Mutelist anlegen, die festlegt, welche Ressourcen und Aspekte nicht mehr gelistet werden sollen. Eine beispielhafte Mutelist, die einen spezifischen Testfall sowie alle Testfälle bezüglich EC2-Instanzen ausblendet, ist in Abbildung 6 dargestellt. Ist sie beim AWS-Cloud-Provider hinterlegt, lassen sich entsprechende Befunde ausblenden. Stumm geschaltete Checks führt Prowler aber dennoch durch. Daher lassen sich auch in der Vergangenheit ausgeblendete Checks bei Bedarf jederzeit noch überprüfen.

Das Erstellen eigener Testfälle gestaltet sich in Prowler nicht ganz so einfach wie in ScoutSuite. Es ist zwar grundsätzlich möglich, erfordert jedoch jeweils ein vollständiges Python-Skript zum Testfall (siehe Listing 4 im GitHub-Repository unter ix.de/zp5z). Damit lässt sich einer-

seits mehr und übersichtlichere Logik in die Testfälle integrieren, andererseits sind aber auch tiefer gehende Kenntnisse in Python notwendig. Ob das ein Problem ist, hängt stark von der eigenen IT ab.

Fazit

Bei Cloud-Plattformen geht es nicht mehr um die Frage, wer sich kümmert, sondern schlicht um die Frage, wann und wie man sich selbst kümmert. Die Testumgebung, die nur um einzelne Komponenten erweitert wurde, aber keineswegs absichtlich geschwächt wurde, enthielt viele Probleme und Schwachstellen. Um diese Probleme kann sich ein Unternehmen nur selbst kümmern. Die Standardeinstellungen der Cloud-Umgebungen übernehmen das nicht von selbst. Die schiere Menge an Themen lässt sich allerdings auch nur automatisiert bewältigen.

Die Wahl des geeigneten Tools ist eine Frage der individuellen Ansprüche sowie des vorhandenen Know-hows. ScoutSuite empfiehlt sich als ressourcenschonendere Variante, die Unternehmen für den einfachen und raschen Einstieg nutzen können, um erste Sicherheitsprüfungen in den verwendeten Cloud-Umgebungen einzurichten. Zwar sind die Compliance-checks des Tools nicht topaktuell, individuelle Anpassungen lassen sich in ScoutSuite jedoch recht einfach umsetzen.

Wenn man Testfälle und Regeln nicht oder nur selten ändert, aber trotzdem die aktuellen Best Practices referenzieren möchte, dann ist die Prowler-CLI eine gute Lösung – vor allem wenn nur einzelne Umgebungen geprüft werden sollen. Nutzt ein Unternehmen mehrere Cloud-Umgebungen parallel und will diese überprüfen, ist Prowler als Web-App die sinnvollere Wahl. Das gilt insbesondere dann, wenn die Überprüfungen längerfristig und vor allem regelmäßig geplant sind. (map@ix.de)

Quellen

Weiterführende Informationen zu den Tools und die Quelltexte der Listings sind unter ix.de/zp5z zu finden.

VIKTOR RECHEL

ist leitender Cyber-sicherheitsberater bei der secuvera GmbH. Im Rahmen der Sicherheitsberatung und seinen Penetrationstests beschäftigt er sich unter anderem mit Cloud-Umgebungen.



