DevSec-Leitfaden zu Infrastructure-as-Code

DevSecOps bietet Teams die Möglichkeit, Sicherheitsmaßnahmen zu automatisieren und in den DevOps-Lebenszyklus einzubinden. In diesem Leitfaden betrachten wir die Herausforderungen beim Einsatz von DevSecOps für die Cloud-Sicherheit und die Vorteile von Infrastructure-as-Code in diesen Szenarien.

Inhalt

Einführung in Infrastructure-as-Code

Infrastructure-as-Code bezeichnet die Technologien und Prozesse, mit denen eine Infrastruktur mithilfe von Code bereitgestellt und verwaltet wird. Sie unterstützen DevOps-Prozesse wie Versionskontrollen, Peer-Reviews, automatische Tests, Tagging sowie Continuous-Integration- und Continuous-Development-Verfahren.

Hintergrund

Infrastructure-as-Code (oder kurz IaC) wurde 2009 vom DevOps-Unternehmen Puppet als Alternative zu den traditionellen Methoden für die Bereitstellung und Verwaltung von Infrastrukturen eingeführt. Puppet erklärte dazu:

> „Die bisherigen Methoden für das Infrastrukturmanagement – manuelle Prozesse und Dokumentation, anfällige Skripte für einen einzigen Anwendungsfall und GUI-basierte Tools – haben in der Vergangenheit ihren Zweck erfüllt. Heutzutage hat die Skalierbarkeit jedoch hohe Priorität und es werden kurzzeitig eingesetzte Infrastrukturen und komplexere Anwendungssysteme genutzt. Für diese neuen Bedingungen benötigen wir auch neue Kontrollmechanismen.“

Inzwischen ist IaC das Kerngeschäft verschiedener Unternehmen, zum Beispiel Ansible, Chef und Salt. In den letzten Jahren wurde IaC vor allem durch Terraform bekannt, ein beliebtes Open-Source-IaC-Framework von HashiCorp, mit dem vorrangig Ressourcen in Diensten in öffentlichen Clouds definiert werden. Dank Terraform ist IaC unbegrenzt anpassbar und zugänglich, sodass ein gesamtes IaC-Netzwerk möglich wird.

Außerdem haben Cloud-Anbieter eigene Konfigurationsframeworks entwickelt, um die Orchestrierung und Verwaltung der Infrastrukturen zu vereinfachen und zu automatisieren. Mit AWS CloudFormation, Azure Resource Manager (ARM) und Google Cloud Deployment Manager können Infrastrukturexperten wesentlich einfacher reproduzierbare Umgebungen aufsetzen.

Funktionsweise

IaC kann entweder deklarativ (Festlegung der Ziele) oder imperativ sein (Festlegung der einzelnen Schritte). Terraform und CloudFormation sind Beispiele für das deklarative Modell, AWS Cloud Development Kit (CDK) hingegen ist ein imperatives IaC-Framework. Kubernetes ist ebenfalls eng mit IaC verbunden, da seine Konfiguration als Code bereitgestellt werden kann.

Jedes Framework hat eigene Konventionen und eine eigene Syntax, aber im Allgemeinen besteht IaC aus Ressourcendeklarationen, Eingabevariablen, Ausgabewerten, Konfigurationseinstellungen und anderen Parametern. Meistens basiert es auf JSON, HCL oder YAML und umfasst alle Konfigurationen zur Einrichtung der Infrastruktur, zum Beispiel für Rechenressourcen, Netzwerk, Speicher, Sicherheit und IAM (Identity and Access Management).

```
module "s3_bucket" {
  source = "terraform-aws-modules/s3-bucket/aws"

  bucket = "my-s3-bucket"
  acl    = "private"

  versioning = {
    enabled = true
  }

}
```

Beispiel für ein Terraform-Modul, das einen privaten S3-Bucket mit aktivierter Versionsverwaltung erstellt

Vorteile von Infrastructure-as-Code

Da die Ressourcenanforderungen bei IaC im Code festgehalten werden, kann die Cloud-Bereitstellung automatisiert, skaliert und zuverlässig wiederholt werden.

Automatisierung

Moderne Unternehmen stellen jeden Tag zahlreiche Anwendungen bereit und die Anforderungen an die Infrastruktur ändern sich daher ständig.

IaC vereinfacht die Cloud-Bereitstellung, indem alle Konfigurationen als Codevorlagen zur Verfügung gestellt werden. Da die bisher manuell vorgenommenen Infrastrukturkonfigurationen als maschinell lesbare Vorlagen verfügbar sind, müssen Entwickler dann die Infrastruktur nicht mehr selbst bereitstellen und verwalten. Sie können stattdessen automatisierte Arbeitsabläufe nutzen, um neue Infrastrukturen zu entwickeln, zu testen und bereitzustellen.

Skalierbarkeit

Mit IaC können Teams wesentlich einfacher zahlreiche Cloud-Ressourcen konfigurieren und gleichzeitig ohne größeren Zeit- und Ressourcenaufwand das Risiko von Fehlkonfigurationen minimieren. Die Automatisierung und Codekonfiguration erleichtern die konsistente Bereitstellung der Cloud-Services.

Außerdem kann die Infrastruktur einfacher aufgelöst werden, wenn sie nicht benötigt wird. Dadurch lassen sich die Kosten für die Rechenkapazitäten und die Wartung reduzieren.

Wiederholbarkeit

Konsistenz ist in einer Cloud-Infrastruktur von entscheidender Bedeutung. Mit IaC werden die Rechenkapazitäten, Speicher und Netzwerkservices immer auf dieselbe Weise bereitgestellt, sodass sie für Ressourcen und sogar in Multi-Cloud-Umgebungen konsistent sind. So lassen sich nicht nur menschliche Fehler minimieren, sondern auch eine umfassende Versionsverwaltung und Protokollierung unterstützen. Dank dieser Wiederholbarkeit können Sie mit geringerem Aufwand mehr Ressourcen bereitstellen und gleichzeitig für eine hohe Qualität, Best Practices zur Sicherheit und die Einhaltung von Branchenbenchmarks sorgen.

Sicherheit

IaC bietet wichtige Funktionen für die Zusammenarbeit diverser Teams. Da die Cloud-Ressourcen in verschiedenen Umgebungen und Clouds in einer einheitlichen Sprache bereitgestellt werden, können sich Entwickler und Ops-Teams besser absprechen und enger zusammenarbeiten, um cloudnative Anwendungen zu schützen.

Zusammenfassung

IaC spart Zeit und Ressourcen:

- **Die Automatisierung sorgt für eine schnelle und skalierbare Ressourcenbereitstellung.**
- **Durch die konsistente und planbare Infrastrukturbereitstellung werden menschliche Fehler minimiert.**
- **Dank der besseren Zusammenarbeit der Teams und dem kodifizierten Wissen besteht eine geringere Gefahr zukünftiger Risiken.**

Risiken bei Infrastructure-as-Code

Doch trotz aller Vorteile gibt es auch bei IaC einige Risiken, die Teams beachten sollten – insbesondere in Bezug auf die Sicherheit und Compliance.

Probleme bei der Einführung

Ähnlich wie eine neue Open-Source-Bibliothek oder SaaS-Plattform bietet auch IaC das Potenzial, die Effizienz zu steigern, allerdings müssen dafür die Unterstützung gesichert und potenzielle Probleme berücksichtigt werden. Da IaC noch relativ neu ist, besteht eine der größten Herausforderungen für die Teams in der richtigen Integration der neuen Frameworks in die bestehende Infrastruktur.

Dabei entstehen unter Umständen komplexere Systeme und es kann für Verwirrung sorgen, wenn nicht festgelegt wurde, wie die Ressourcen bereitgestellt, verwaltet und geschützt werden.

Konfigurationsabweichungen

Bei IaC wird die Infrastruktur nicht in einer zentralen Konsole verwaltet, sondern über ein weiteres Framework, das als verbindliche Datenquelle dient. Werden Änderungen an der Infrastruktur nicht im Bereitstellungscode vorgenommen, entstehen Abweichungen zwischen den aktiven Ressourcen und der zugrunde liegenden IaC-Konfiguration.

Ohne entsprechende Tools besteht daher die Gefahr, dass sich Fehlkonfigurationen einschleichen oder andere Risiken die Umgebung bedrohen.

Nachlässigkeit bei der Sicherheit

Fehlkonfigurationen gehören zu den Hauptursachen für Datenverletzungen in der Cloud. Einigen Angaben zufolge werden 99 Prozent der Fehlkonfigurationen in Unternehmen nicht bemerkt.[1]

Cloud-Sicherheitstools ermöglichen zwar den notwendigen Überblick und eine Überwachung, doch das Feedback ist nicht immer hilfreich für IaC.

Werden Fehlkonfigurationen nach der Zusammenführung und Bereitstellung gefunden, kann dies zu Konflikten zwischen den Teams führen und zusätzliche Arbeit verursachen, da die Probleme untersucht, priorisiert und letztendlich behoben werden müssen.

1. McAfee, „What is Cloud Security?“, letzter Aufruf: 9. November 2021, https://www.mcafee.com/enterprise/en-us/security-awareness/cloud.html.

Die beste Lösung für diesen Fall ist eine Shift-Left-Strategie für die Cloud-Sicherheit, damit nicht die Laufzeit der Infrastruktur überwacht werden muss, sondern Fehlkonfigurationen im Code gesucht werden können. Doch die Behebung von Sicherheitsproblemen auf der IaC-Ebene ist bisher vernachlässigt worden. Dieses Problem werden wir im nächsten Abschnitt genauer betrachten.

Zusammenfassung

IaC kann Konflikte und Risiken verursachen:

- Wenn IaC nicht umfassend integriert und vollständig angenommen wird, entstehen in den Umgebungen, Arbeitsabläufen und Teams Konflikte.
- Unter Umständen wird nicht deutlich kommuniziert, wie und wo die Infrastruktur verwaltet und die Richtlinien durchgesetzt werden.
- Cloud-Sicherheitstools liefern kein proaktives, aussagekräftiges Feedback.

Sicherheitsherausforderungen bei Open-Source-IaC

IaC unterstützt zwar vorkonfigurierte Open-Source-Vorlagen oder -Module, doch diese lassen sich nicht mit einem sicherheitsorientierten Ansatz vereinbaren.

Open-Source-IaC wird sowohl in GitHub als auch in speziellen Repositorys wie der Terraform Registry und im Artifact Hub zunehmend unterstützt. Mit Open-Source-IaC können Entwickler Cloud-Services wesentlich schneller und einfacher bereitstellen, doch dabei wird häufig die Sicherheit außer Acht gelassen.

Etwa die Hälfte aller Open-Source-Module in der Terraform Registry und aller Helm-Charts im Artifact Hub enthielten Fehlkonfigurationen. Diese Untersuchungsergebnisse verdeutlichen die Probleme in Bezug auf die Umsetzung der Cloud-Sicherheit. Sie zeigen, wie stark die Sicherheit vernachlässigt wurde und welche entscheidende Rolle DevSecOps bei dem Schutz der Cloud-Infrastruktur spielen kann.

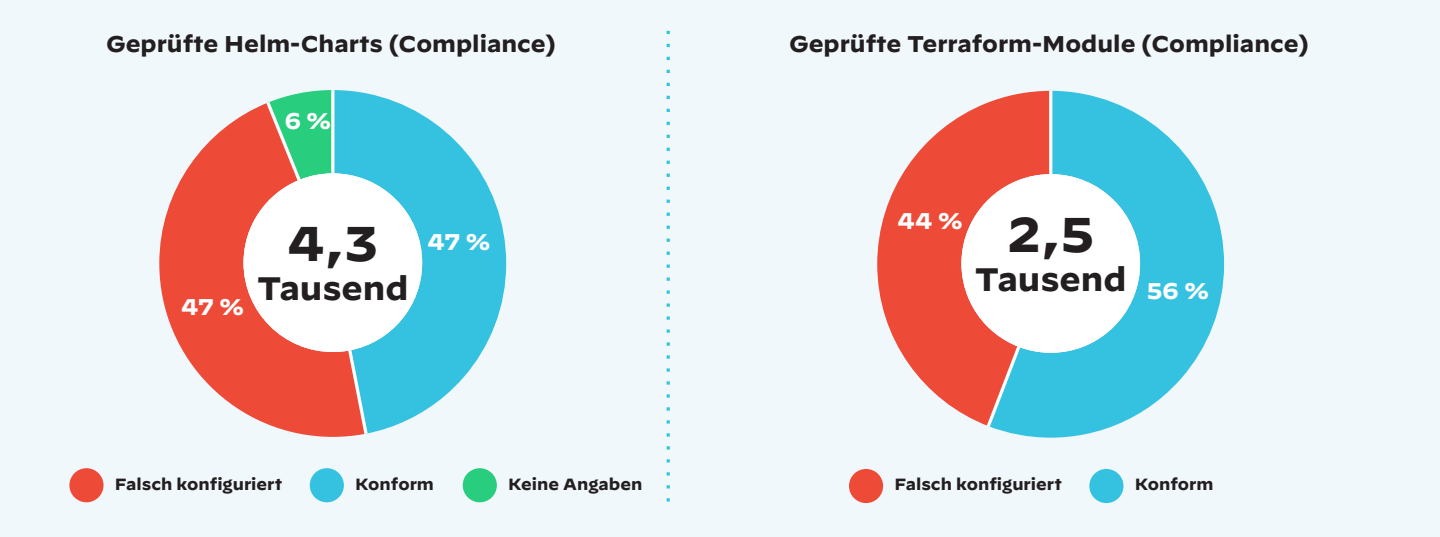

Abbildung 1: Ergebnisse der Studie von Bridgecrew zur Sicherheit bei Open-Source-Helm-Charts (links), Ergebnisse des Berichts von Bridgecrew „State of Open Source Terraform Security“ (rechts)

Herausforderungen mit DevSecOps in IaC

Ohne die richtige Strategie, Methoden und Tools kann die Einführung von DevSecOps für die Cloud-Sicherheit schnell zu noch größeren Engpässen und Konflikten zwischen Teams führen.

Entwickler und Sicherheitsteams verfolgen bekanntlich häufig recht unterschiedliche Ziele. DevOps-Teams möchten möglichst schnell und iterativ arbeiten, doch das nachträgliche Feedback, das Sicherheitsteams außerhalb der Sprintphasen bereitstellen, verhindert dies.

Mit DevSecOps sollen Sicherheitsmaßnahmen in die DevOps-Prozesse eingebunden werden, um Fehlkonfigurationen und anfällige Implementierungen cloudnativer Anwendungen zu vermeiden. Doch ohne den richtigen Ansatz kann die Einführung einer DevSecOps-Strategie ebenfalls zu einer Herausforderung werden, insbesondere in Bezug auf die Cloud-Sicherheit.

Das hat vor allem zwei Ursachen. Erstens verändert sich die Infrastruktur sehr schnell und die Unternehmen verfügen eventuell nicht über die notwendige Erfahrung und das erforderliche Know-how, um damit Schritt zu halten. Zweitens wurden viele der vorhandenen Prozesse und Tools nicht für die Cloud-Sicherheit konzipiert und verursachen daher Engpässe für cloudnative Entwickler und Sicherheitsteams.

Inkonsistente Governance

DevSecOps ist in der Cloud schwieriger umzusetzen, da es wenig Prozesse für die Entwicklung einer skalierbaren Infrastruktur gibt. IaC ist die Lösung für das Problem. Damit lassen sich die Herausforderungen in Bezug auf Leistung und Kosten einer großen Infrastruktur bewältigen.

Die Richtlinienverwaltung hat bei IaC höchste Priorität, da zusätzliche Cloud-Infrastrukturen entstehen, die von diversen Teams mit unterschiedlichen Arbeitsabläufen verwaltet werden. Diese Arbeitsabläufe sorgen für eine größere Komplexität, sodass nicht immer eindeutig ist, wo und wann Richtlinien durchgesetzt werden. Im besten Fall entstehen dadurch nur Redundanzen, im schlimmsten Fall erhöhen sich jedoch die Risiken.

Wie sieht das in der Praxis aus?

- Das IT-Team geht davon aus, dass die Sicherheits- und Compliancerichtlinien von den Tools des Cloud-Anbieters oder anderen Sicherheitslösungen auf Cloud-Ebene durchgesetzt werden.
- Das Sicherheitsteam hat keinen Überblick über die diversen Bereitstellungsframeworks und weiß daher nicht, ob für die bereitgestellten Ressourcen die korrekten Richtlinien festgelegt wurden.

Aus diesem Grund muss unbedingt ein Arbeitsablauf einrichtet werden, der genau diese Probleme behebt, damit keine Cloud-Ressourcen mit falschem Code bereitgestellt werden und dann für zusätzlichen Arbeitsaufwand sorgen.

Tipp: Fehlerbehebung im Quellcode

Wenn eine Fehlkonfiguration nur in der Laufzeit von IaC korrigiert wird, besteht laut unseren Erkenntnissen eine Wahrscheinlichkeit von 70 Prozent, dass sie erneut auftritt. Wird hingegen der Fehler direkt im Quellcode behoben, treten die Fehlkonfigurationen in Zukunft nicht mehr auf.

Fehlende Fachkenntnisse und Zugriffsrechte

Automatisierte Tools können ganz unterschiedliche Probleme aufdecken – von kritischen Sicherheitsverletzungen bis zu Verstößen gegen Best Practices. Doch auch die neuesten und besten Tools sind kein Ersatz für zuverlässige DevSecOps-Prozesse und -Arbeitsabläufe in der Cloud.

Denn wenn ein Tool eine Fehlkonfiguration erkennt, müssen anschließend der Kontext analysiert, die Dringlichkeit ermittelt und der Fehler behoben werden. Es ist aber oft nicht einfach, die Sicherheits- und Entwicklerteams zur gemeinsamen Problembehebung zu bewegen. Die Herausforderung besteht darin, dass Sicherheits- und Complianceteams oft keinen Zugriff auf Coderepositorys oder Cloud-Konsolen haben, um die notwendigen Korrekturen vorzunehmen. Und selbst wenn sie über die Rechte verfügen, haben sie eventuell nicht die Fachkenntnisse und den Überblick über den gesamten Anwendungskontext, sodass sie die Fehlkonfigurationen nicht selbst beheben können. Den Entwicklern fehlen wiederum meist die notwendigen Sicherheitskenntnisse, um die Problembehebungen zu priorisieren. Und selbst mit dem erforderlichen Know-how kann es schwierig werden, Korrekturen vorzunehmen.

An dem folgenden Beispiel wird deutlich, wie durch die Automatisierung diese Situation sogar noch verschärft werden könnte:

- Das Sicherheitsteam erstellt ein Ticket und weist es einem Teamleiter zu, dessen Team eventuell für dieses Problem zuständig ist.
- Nach einigen Runden Jira-Roulette wird die Fehlerbehebung für einen der nächsten Sprints eingeplant.
- In der Zwischenzeit wurden zehn weitere „PO“-Tickets erstellt.
- Bei automatisierten Scans kann dies schnell zu Hunderten neuen Tickets führen und der gesamte Prozess wird endlos wiederholt.

Der Fachkräftemangel verschärft die Lage zusätzlich, da Sicherheits- und Complianceprobleme den Entwicklern übertragen werden, die sich oft nicht ausreichend damit auskennen.

Doch unabhängig von den verfügbaren Sicherheitsressourcen ist es unrealistisch, zu erwarten, dass Entwickler sich im Detail mit allen Konfigurationsproblemen auskennen, die bei der Einrichtung einer Cloud-Infrastruktur auftreten können. Aus diesem Grund schleichen sich recht schnell Fehlkonfigurationen ein. Fehlkonfigurationen sind nicht nur ein Risiko. Während der Einrichtung kann ein Entwickler das Problem relativ schnell beheben, doch je länger der Fehler im Umlauf ist, desto kostspieliger und aufwendiger wird auch die Behebung.

Zusammenfassung

DevSecOps kann unter Umständen größere Konflikte verursachen:

- Wenn die Richtlinienverwaltung zwischen Infrastrukturcode und bereitgestellten Cloud-Ressourcen nicht festgelegt wurde oder inkonsistent ist
- Wenn Fachkenntnisse oder Zugriffsrechte fehlen, um potenziell gefährliche Fehlkonfigurationen zu identifizieren und zu beheben

Tipps für die erfolgreiche Einführung von cloudbasiertem DevSecOps und IaC

Bisher haben wir nur die potenziellen Herausforderungen von DevSecOps in der Cloud betrachtet. Mit diesen drei Tipps lassen sich diese Herausforderungen bewältigen:

1. **Umfassende Automatisierung**

   Ihre IT-Teams haben vermutlich schon einen Großteil der Softwaretests automatisiert – von Komponententests bis zu Abhängigkeitsprüfungen. Cloud-Sicherheit sollte da keine Ausnahme bilden. Auch die erfahrensten Sicherheitsexperten können nicht über alle Sicherheitsempfehlungen für alle Cloud-Anbieter, Serviceebenen, Orchestrierungsplattformen, Ressourcen und ähnliches informiert sein und diese Faktoren bei ihrer spezifischen Architektur berücksichtigen.

   Automatisierte Scans sind die einzige Möglichkeit, Sicherheits- und Compliancekontrollen umfassend und programmatisch zu implementieren, ohne dass sich Mitarbeiter erst lange in diverse Dokumentationen einlesen müssen. Ohne aussagekräftiges Feedback kann die Automatisierung allerdings zu Konflikten für die IT-Teams führen und den Arbeitsaufwand sogar erhöhen. Aus diesem Grund benötigen Sie die richtigen Tools – entweder Standardlösungen oder Open-Source-Angebote wie Checkov –, um sowohl die Aufdeckung als auch die Behebung von Problemen zu automatisieren.

2. **Einbindung in bestehende Prozesse**

   Werden Informationen zum falschen Zeitpunkt oder am falschen Ort angezeigt, kann auch aussagekräftiges Feedback stören. DevSecOps wird am besten angenommen, wenn es in die Tools und Prozesse eingebunden wird, die Entwickler bereits jeden Tag nutzen. Glücklicherweise verfügen Sie also bereits über die Grundlage für die Automatisierung und Security-as-Code.

   Wo und wie Sie die Kontrollen einbinden, hängt von verschiedenen Faktoren ab, unter anderem von den vorhandenen Tools, den Unternehmenszielen und dem Reifegrad der Sicherheitsmaßnahmen.

3. **Sicherheit im Code**

   Viele der DevSecOps-Konflikte treten auf, weil in der Vergangenheit meist eine reaktive Sicherheitsstrategie verfolgt wurde, die auf die Suche nach Fehlern statt auf deren Verhinderung setzt. Mithilfe von IaC können Sie einen proaktiven Ansatz einführen. Ganz gleich, ob es sich um Policy-as-Code oder Security-as-Code handelt – in beiden Fällen ist die Governance-Automatisierung allgemein nachvollziehbar, sodass Sicherheitsexperten und DevOps-Mitarbeiter zeitnah Feedback und praxisrelevante Ergebnisse erhalten.

   Da die Richtlinien auf Code-Ebene durchgesetzt werden, sind die Cloud-Sicherheitsmaßnahmen konsistent und können nach und nach für die gesamte Umgebung übernommen werden. Außerdem sparen Sie so langfristig, da die Behebung eines Softwarefehlers in der Produktionsumgebung 100-mal teurer als die Korrektur im Code sein kann.[2] Es stimmt allerdings, dass in den frühen Lebenszyklusphasen nur wenige Informationen zur Infrastruktur verfügbar sind und sich Fehlkonfigurationen daher nur schwer voraussehen lassen.

   Daher ist es so wichtig, die Sicherheit der Infrastruktur in jeder Phase des DevOps-Lebenszyklus zu berücksichtigen.

2. Mukesh Soni, *Defect Prevention: Reducing Cost and Enhancing Quality*, https://www.isixsigma.com/industries/software-it/defect-prevention-reducing-costs-and-enhancing-quality/.

Cloud-Sicherheit im DevOps-Lebenszyklus

Es gibt zwar keine Universallösung, doch die Durchsetzung von Kontrollfunktionen in der IaC und die Bereitstellung von aussagekräftigem Feedback während des gesamten DevOps-Lebenszyklus vereinfachen die Einführung von cloudbasiertem DevSecOps.

Abbildung 2: Die Vor- und Nachteile der Identifizierung von Cloud-Fehlkonfigurationen für jede Phase des Entwicklungszyklus

IDE-Scans

Die frühzeitige Meldung von Problemen ist einer der Grundpfeiler der Shift-Left-Sicherheitsstrategie. Idealerweise findet dies bereits bei der Programmierung durch die Entwickler statt.

Bei der Shift-Left-Strategie werden Sicherheitsmaßnahmen schon während der Design- und Planungsphase berücksichtigt und Kontrollfunktionen in die IDE (Integrated Development Environment) integriert. Meist wird dazu ein Plug-In oder eine Erweiterung wie Checkov VS Code verwendet. Da kaum Kontextwechsel notwendig sind, sind IDE-Scans die kostengünstigste und sicherste Methode, um Probleme aufzudecken.

Pre-Commit-Hooks

Pre-Commit-Komponenten- und -Integrationstests gehören allgemein zu den Best Practices. In Zukunft sollten auch Pre-Commit-IaC-Sicherheitsscans dazu zählen.

Die lokale Suche in IaC nach Fehlkonfigurationen ist die beste Methode, um Fehler in einer sicheren Arbeitsumgebung zu korrigieren, bevor der Code in gemeinsam genutzte Repositorys integriert wird. Ein weiterer großer Vorteil von Pre-Commit-Scans ist die Zeitersparnis, da keine Fehler in Builds auftreten oder Pull-Anfragen geprüft werden müssen.

Ganz gleich, ob Sie die Änderungen aus dem Feedback sofort implementieren oder die IDE-Empfehlungen als Hinweise nutzen, um eine sichere Infrastruktur zu erstellen – diese Methode hat überwiegend Vorteile. Der einzige Nachteil besteht darin, dass der Entwickler diese lokalen Scans ausführen und die Änderungen vornehmen muss.

Mit dem richtigen Gleichgewicht zwischen passiven Analysen und praxisrelevantem Feedback können Sie Sicherheitsmaßnahmen besser durchsetzen, ohne dass dazu ein größerer Aufwand erforderlich ist.

Prüfungen von Pull/Merge-Anfragen

Für Teams, die sich ganz auf ihr VCS (Version Control System) verlassen, hat die Einbindung der Sicherheitsmaßnahmen in genehmigte Prozesse für den Codereview zahlreiche Vorteile. Je nachdem, wie die CI/CD-Prozesse gestartet werden, kann diese Methode in Kombination mit oder anstelle von CI/CD-Scans genutzt werden.

Das VCS umfasst eventuell auch spezielle Kontrollfunktionen. Die drei bekanntesten Plattformen (GitHub, GitLab und Bitbucket) bieten Versionskontrollen, Duplizierung (Branching), eingebettete Codereviews und Autorisierungskontrollen, mit denen Entwickler Tests ohne Beeinträchtigung der Produktionssysteme durchführen und Änderungen am Code noch vor der Zusammenführung vornehmen können.

Tipp: Zusätzliche Sicherheitskontrollen

Es empfiehlt sich, in den Codereview-Einstellungen Zusammenführungen zu blockieren, wenn Prüfungen negativ ausfallen. So verhindern Sie, dass falsch konfigurierte IaC in den Master-Branch gelangen.

CI/CD-Jobs

CI/CD-Pipelines sind ein wichtiger Faktor bei der Einrichtung einer Infrastruktur und dem Testen von kompiliertem Code vor der Bereitstellung. Scans bieten die Möglichkeit, Fehlkonfigurationen in den zugrunde liegenden Ressourcen, Variablen und abhängigen Modulen aufzudecken, bevor diese implementiert werden.

Die Einbindung der Infrastruktursicherheitsmaßnahmen in die CI/CD-Pipeline hat zudem den Vorteil, dass sie automatisiert und gezielt an den Arbeitsablauf angepasst werden können. Sie können festlegen, bei welchen Prüfergebnissen Builds überarbeitet werden müssen, und Feedback direkt in der CI/CD-Lösung anzeigen. Dieser Prozess eignet sich für das gesamte Team, das damit Änderungen prüfen, ablehnen oder genehmigen kann.

Unabhängig davon, ob die Scans in der CI/CD-Lösung oder im VCS durchgeführt werden, fördern Sie so die Zusammenarbeit und die Zugriffsrechte der Entwickler.

Sicherheit in der Laufzeit

Die Verlagerung der Sicherheitsmaßnahmen in die Entwicklung bedeutet nicht zwangsläufig, dass traditionelle Cloud-Sicherheitsmethoden wie die Überwachung der Cloud-Ressourcen auf Sicherheits- und Compliancefehlkonfigurationen ausgeschlossen werden.

Auch bei einer umfassenden IaC können weiterhin manuelle Änderungen vorgenommen werden, wodurch eventuell unerwünschte Abweichungen zwischen Code und Cloud-Ressourcen auftreten. Bei der Verwaltung von Cloud-Abweichungen erhalten Sie einen Überblick über diese Ressourcen und können potenziell riskante Lücken aufdecken.

Wenn Sie sich hingegen darauf verlassen, Fehler in den Builds zu erkennen, und diese nicht dem Laufzeitstatus zuordnen, können Konfigurationskonflikte entstehen. Da sich die Laufzeitscans an dem jeweiligen Konfigurationsstatus orientieren, ist dies der einzige Weg, um Konfigurationsänderungen im Zeitverlauf zu überprüfen, wenn die Konfiguration mit unterschiedlichen Methoden verwaltet wird. Außerdem werden so die Vorgaben der Complianceaudits erfüllt, die kontinuierliche Änderungskontrollen und -nachweise erfordern.

Zusammenfassung

Ein umfassendes cloudbasiertes DevSecOps-Programm sollte die folgenden Kriterien erfüllen:

- **Den Entwicklern sollten Tools zur Verfügung stehen, mit denen sie frühzeitig und kostengünstig Feedback erhalten.**
- **Es sollten umfassende Kontrollfunktionen im VCS oder in der CI/CD-Pipeline verfügbar sein.**
- **Es ist ein umfassender und kontinuierlicher Überblick über die Laufzeitressourcen erforderlich, um Abweichungen durch manuelle Konfigurationsänderungen erkennen und beheben zu können.**

Fazit

Mit IaC ist die Bereitstellung und Verwaltung einer Infrastruktur häufig effizienter und sie ist eine Grundvoraussetzung für cloudbasiertes DevSecOps. Trotz gewisser Herausforderungen kann sie zur Unterstützung der Infrastruktursicherheit genutzt werden.

Werden IaC-Scans und Security-as-Code-Fehlerkorrekturen während des gesamten DevOps-Lebenszyklus vorgenommen, ist eine modernere Cloud-Sicherheitsstrategie möglich.

Wie bei allen neuen Technologien werden auch mit IaC eventuell die Komplexität und Risiken erhöht, insbesondere, wenn Teams unterschiedliche Frameworks nutzen. Da jedoch parallel auch eine manuelle Cloud-Orchestrierung möglich ist, muss für IaC ein umfassender Überblick gewährleistet sein. Andernfalls können Lücken bei der Bereitstellung und vor allem auch der Sicherheit der Ressourcen entstehen.

Die Vorteile von IaC übertreffen jedoch die Kosten, da sie Automatisierung, Skalierbarkeit und Wiederholbarkeit in der Cloud-Bereitstellung und -Verwaltung ermöglicht.

Für das Prisma Cloud-Team ist IaC außerdem das notwendige Bindeglied zwischen Infrastrukturentwicklung, DevOps und Sicherheit. Dank der Vorteile dieser Technologie können Teams die Cloud-Sicherheitsmaßnahmen automatisieren und in den DevOps-Lebenszyklus einbinden. Unsere codebasierte Cloud-Sicherheitsplattform unterstützt eine solche Zusammenarbeit, da IaC-Scans und Security-as-Code-Fehlerkorrekturen in die bereits vorhandenen Tools und Arbeitsabläufe der Entwickler eingebunden werden.

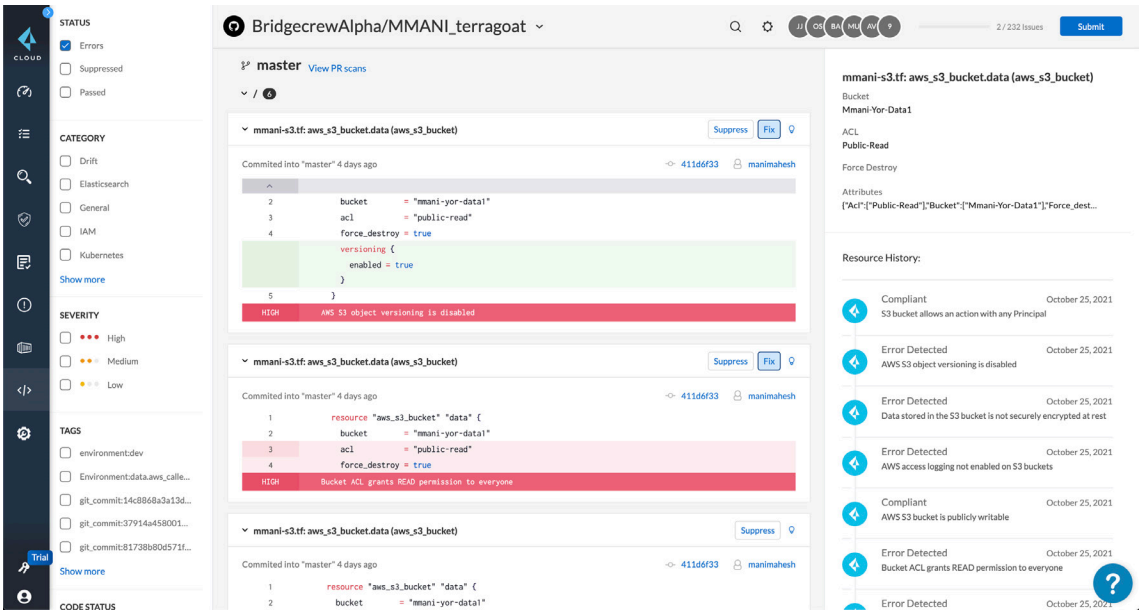

Abbildung 3: IaC-Sicherheitsfunktionen in Prisma Cloud

Mit Prisma Cloud können Sie

- Fehlkonfigurationen in Cloud-Ressourcen und IaC finden und beheben.
- Hunderte integrierte Richtlinien diverser Sicherheits- und Compliancebenchmarks durchsetzen.
- Kontrollfunktionen über IDE-Plug-Ins, Pre-Commit-Hooks und native VCS- und CI/CD-Integrationen einbetten.

Testversion anfordern

paloalto NETWORKS

Oval Tower, De Entrée 99–197
1101 HE Amsterdam, Niederlande

Telefon: +31 20 888 1883
Vertrieb: +800 7239771
Support: +31 20 808 4600

www.paloaltonetworks.de