AIRLOCK®
SECURE ACCESS HUB

WHITEPAPER

Auf zu DevSecOps

Best Practices für Microgateway Deployments

Partner: vshn.ch

Inhalt

Überblick über den Inhalt

Zielgruppe

Gewichtung	Zielgruppe
+	Security Officer
	Business
	CIO
++	Developer / Projektleiter
+	Architekt
++	DevOps-Ingenieur
	Einkauf

Einführung

Was ist DevOps und was sind die Vorteile?

In einer Welt, in der Software immer wichtiger wird, hängt der Erfolg eines Unternehmens davon ab, wie schnell Software entwickelt und ausgerollt werden kann. Technologie entwickelt sich schnell und ist für alle zugänglich. Heute ist der entscheidende Faktor, wie gut ein Unternehmen seine Zielgruppe versteht. Das ermöglicht es ihm, Kundenbedürfnisse besser zu bedienen und schneller als die Wettbewerber zu sein.

DevOps steht für eine Kultur der funktionsübergreifenden und kollaborativen Zusammenarbeit. Traditionell haben Entwicklung und Betrieb unterschiedliche Ziele verfolgt: Softwareentwicklung muss agil, kreativ und auf dem neusten technischen Stand sein, um ständig neue Features liefern zu können. Im Gegensatz dazu ist IT Operations auf Stabilität, Sicherheit und Zuverlässigkeit ausgelegt. DevOps vereint diesen scheinbaren Widerspruch zwischen Flexibilität und Stabilität. Als logischen nächsten Schritt agiler Softwareentwicklung versucht DevOps die gesamte Wertschöpfungskette von der Softwareentwicklung bis zum Betrieb auf interdisziplinäre Weise zu kombinieren. Das bricht Silodenken auf, indem es die Gräben zwischen den Silos überbrückt, und richtet die Organisation auf das gemeinsame Ziel der schnellen Lieferung neuer Funktionen in stabilen, sicheren Schritten aus.

DevOps besteht aus Prozessen, Werkzeugen und Kulturen, die vor allem von den Menschen abhängen. Eine DevOps-Kultur muss gelebt werden. DevOps kann nicht mit Werkzeugen, neuen Prozessen oder einem DevOps-Ingenieur „gekauft“ werden. DevOps ist interdisziplinäre Zusammenarbeit nicht nur von Entwicklern, Ingenieuren und Betreibern, sondern von allen am Produktlebenszyklus beteiligten Parteien wie Product Owner, Scrum Master, Testing und Security.

DevSecOps

Traditionell war Security ein „Gatekeeper“ beim Abschluss der Softwareentwicklung, ganz ähnlich wie Operations vor DevOps. Das „Sec“ in „DevSecOps“ macht die Zusammenarbeit und die Verschiebung der Verantwortung explizit.

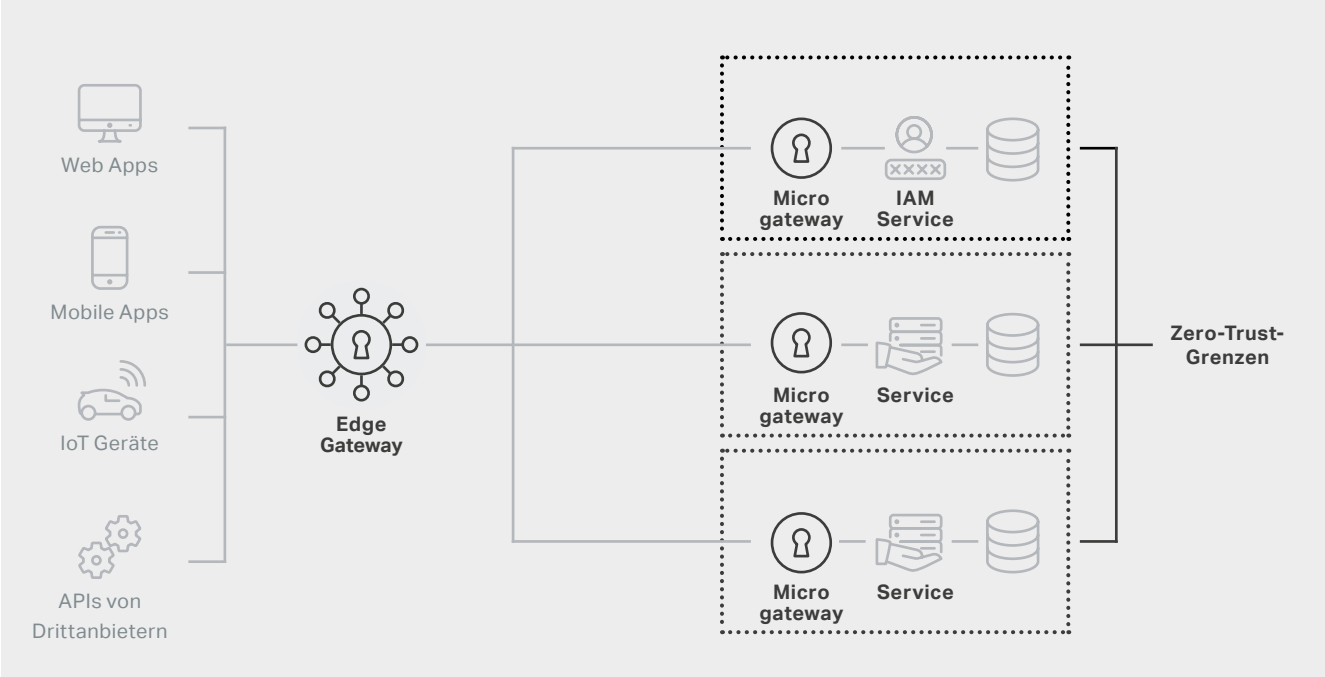
In einer DevSecOps-Kultur hat jedes agile Team einen Sicherheitsexperten. Er erfüllt nichtfunktionale Anforderungen (wie Datenklassifizierung und andere Teile der Risikoanalyse), sodass der Product Owner im Entwicklungsplan auch Security berücksichtigt. Er bietet auch Unterstützung und Werkzeuge (idealerweise mit allen Produktentwicklungsteams geteilt) für sichere Entwicklungs- und Betriebspraktiken.

Dieser proaktive Ansatz ermöglicht es Teams, die Gesamtverantwortung für den Umfang ihrer Services übernehmen. Er stellt ihnen verlässliche Dienste bereit und wird von Operational Security ergänzt, um die Compliance zu garantieren (zum Beispiel durch Scanning und Penetration Testing). Das erfüllt die Anforderungen eines Security Officers auf der Unternehmensebene und erlaubt gleichzeitig Agilität auf der Ebene einzelner Softwareprodukte.

Wir lernen von Softwarearchitekturen, dass synchrone externe Service Calls einen Prozess blockieren können. Wenn man diese Erkenntnis auf DevSecOps überträgt, soll Security asynchron mit der Produktentwicklung gekoppelt werden. Das ermöglicht dem Product Owner, Geschwindigkeit und Sicherheit zu steuern und weder das Eine noch das Andere zu opfern. Eine agile Entwicklung benötigt agile Infrastrukturen und agile Sicherheit.

Sicherheit ist Pflicht

Der Art, wie wir Software gegen externe und interne Gefahren absichern, entwickelt sich als Antwort auf die Bedrohungslage, der wir ausgesetzt sind. Der neuste Schritt in dieser Evolutionskette ist die Entwicklung zu Zero-Trust-Architekturen.



Zero-Trust-Architektur

Zero-Trust ist ein Paradigmenwechsel. Der Gedanke hinter Perimeter-Sicherheits-Architekturen bestand in der Trennung eines unsicheren externen Netz (d.h. dem Internet) von einem sicheren internen Netz, Am Perimeter wurde der ganze Verkehr inspiziert und der potentiell gefährliche Teil blockiert. Mit Zero-Trust-Architekturen werden Inspektion und Blockierung vom Perimeter direkt zu den Diensten selbst verlegt. Anders gesagt: Jeder Dienst untersucht seinen eigenen Traffic und lässt nur den als sicher erkannten zu.

Die Implementierung einer Zero-Trust-Architektur erfordert Technologien, die denen am Perimeter ähneln, jedoch in kleinerer und resourcenschonender Form. Daraus leitete sich die Idee des Microgateways ab.

Zero-Trust-Architekturen eignen sich ideal zur Umsetzung durch DevSecOps-Organisationen. Ein DevSecOps-Team, versteht seinen eigenen Service perfekt und ist ideal geeignet, um die Security Policy zum Schutz dieses Dienstes zu definieren und zu steuern. Wenn Time-to-Market und schnelle Entwicklung wichtig sind, kann das DevSecOps-Team schneller für Sicherheit sorgen und diesen Prozess auch automatisieren.

Durch die Verschiebung der Zuständigkeit für die Sicherheit vom zentralen Perimeter in dezentrale Dienste wird die Komplexität reduziert und die Abwehr vereinfacht. Das wirkt der schnell steigenden Komplexität entgegen, da immer mehr Dienste ausserhalb des internen Netzwerks bereitgestellt werden müssen.

Auch wenn die Dezentralisierung viele Vorteile hat, müssen doch manche Funktionen zentral bleiben. Eine dieser Funktionen ist das Identitäts- und Zugangsmanagement (IAM). Ein nahtloses Single-Sign-on-Benutzererlebnis kann am besten mit zentralen Authentifizierungs- und Identitätsmanagementdiensten erreicht werden. Das Edge Gateway agiert als Policy Enforcement Point, wobei die eigentlichen Entscheidungen durch den IAM-Dienst getroffen werden (siehe obiges Diagramm). Die Prüfung der Identitätsinformationen und die Autorisierung der Benutzer wird von den individuellen Diensten durchgeführt.

Zero-Trust-Architektur gewährleistet, dass jeder Dienst nicht nur sich selbst gegen unerwünschten Traffic schützt, sondern jede Anfrage validiert, um zu garantieren, dass nur korrekt authentifizierte Benutzer Zugang zu den Services und Daten haben, für die sie autorisiert sind.

Gewonnene Erkenntnisse

Unsere Partner von VSHN – The DevOps Company - teilen ihre Erfahrungen mit der praktischen Umsetzung von DevSecOps und Zero-Trust-Architekturen:

Teile und herrsche

Der Bedarf an Geschwindigkeit und Unternehmensflexibilität hat dazu geführt, dass Softwareentwicklungsteams Grössen erreicht haben, die nicht mehr effektiv an einem grossen monolithischen Softwareprojekt arbeiten können. Das hat die Nutzung von Microservice-Architekturen mit standardisierten Interfaces (API) für jeden Microservice vorangetrieben. Als Ergebnis sind sowohl Software- als auch Entwicklungsteams in Einheiten mit einer effizienteren Grösse aufgeteilt. Solange die Schnittstellen rückwärtskompatibel bleiben, kann jedes Team seine Dienste unabhängig von anderen Teams entwickeln und deployen. Mit geringerem Zeitaufwand für die Koordination der Deployments haben die Teams die Kapazität, den Code öfter zu aktualisieren, so dass kleinere Änderungen ausgerollt werden, sobald sie verfügbar sind. Das führt zu weniger Codeänderungen bei jedem einzelnem Deployment, geringerem Fehlerrisiko und schnellerem Rollback sowie höherer Gesamtstabilität.

In der Folge werden massiv mehr Services entwickelt und ausgerollt, was es immer schwieriger macht, mit der Prüfung und Genehmigung jeder Änderung Schritt zu halten, insbesondere mit einer einzelnen zentralen Security-Funktion. Das wird in der DevSecOps-Kultur durch einen „Shift Left" gelöst, was eine Verlagerung der Sicherheitsverantwortung zum Entwicklungsteam bedeutet. Das Entwicklungsteam wird asynchron unterstützt, um Sicherheitsherausforderungen zu bewältigen, bevor Dritte sie im produktiven Betrieb finden.

Automatisierung

Automatisierung ist der Schlüssel, um die „Digitalisierung der Entwicklungs- und Sicherheitsverfahren“ in einer Umgebung mit Dutzenden oder Hunderten täglichen Service-Deployments zu schaffen. Das automatische Ausführen von Tests bei jeder Codeänderung (Continuous Integration) ist schnell zu einer bewährten Methode der Softwareentwicklung geworden. Security-Fachleute automatisieren Teile der Code-Reviews (White Box Testing) mit Tools für Codequalität, Testabdeckung, statische Codeanalyse, Dynamic Application Security Testing, Dependency- und Container-Scans, Lizenz-Compliance-Checks, Secret Embedding Detection und Fuzzing. Diese Werkzeuge und Prozesse können für den gesamten Servicebestand ausgerollt werden, um eine solide Sicherheitsbasis und Frühwarnungen bei Schwachstellen in bestimmten Bibliotheken zu bekommen.

Services wirksam nutzen

DevSecOps macht die Teams für die Stabilität und Sicherheit ihrer Dienste verantwortlich. Die Teams können Erwartungen einfacher erfüllen, wenn der Funktionsumfang ihres Dienstes minimiert wird. Das wird oft durch die Nutzung anderer Services erreicht, die das Team nicht selbst bauen und betreiben muss.

Containerplattform

Als ausgezeichnete Abstraktionsschicht für Operations kann eine Containerplattform Services für Softwareentwicklungsteams bereitstellen. Die Containertechnologie führt zu erhöhter Stabilität und Sicherheit des Dienstes durch:

- Hochverfügbarkeit, Failover, Automatische Neustarts, Rollendes Deployment
- einen zentralen Einstiegspunkt für Auditing und Administrations-Zugang
- Standardisierung der Containertechnologie für Build, Deployment, Scanning, Monitoring, Backup, CMDB, Configuration and Secrets Management usw.
- Unabhängigkeit in Bezug auf die Infrastruktur: Cloud, standortbasiert oder hybrid
- standardmässige Trennung von Nutzern/Diensten (RBAC auf Namespaces, Storage, Netzwerk usw.)

Infrastruktur zur Authentifizierung und Autorisierung

Zentrales Identity and Access Management (IAM) nutzt Standardprotokolle wie OAuth 2.0, OpenID Connect oder JWT. Das erlaubt es dem Entwicklungsteam, Standardfunktionen ihrer Entwicklungsframeworks für diese Services zu nutzen, anstatt Benutzer- und Rechtemanagement für jeden Dienst einer Anwendung neu erfinden zu müssen.

Validierte Basis-Images

Validierte Container-Images für Entwicklungstechnologien und interne Services helfen dem Team, sich auf die Softwareentwicklung zu konzentrieren. Idealerweise wird das von Container-Scanning begleitet, um auch Compliance durchzusetzen.

Das Airlock Microgateway

Das Verfolgen eines DevSecOps-Paradigmas bringt eine Reihe von Vorteilen mit sich. Ein glücklicher CISO und ein agiler Deployment-Prozess, der wiederum zu einer kürzeren Time-to-Market von Innovationen führt. Aber all das in der Praxis umzusetzen, ist nicht trivial, selbst wenn man nur eine einzige Umgebung hat. Beim Entwickeln von Security-Produkten zur Unterstützung von DevSecOps besteht die grösste Herausforderung darin, flexibel genug zu sein, um alle möglichen Lösungen gleich gut zu unterstützen. Manche Kunden möchten einen ungewöhnlichen Weg gehen oder einen proprietären Technologie-Stack nutzen. Das Produkt sollte generische Prinzipien implementieren und nicht von bestimmten Werkzeugen abhängig sein.

Die Entwicklung eines Sicherheitsprodukts stellt zwar eine Herausforderung dar, ermöglicht aber die Bereitstellung von Richtlinien und Best Practices für Kunden. Kunden die nur wenig Erfahrung mit DevOps und Container-Plattformen haben, werden durch die vom Produkt angebotenen Best Practices unterstützt. In diesem Kapitel werden verschiedene Design-Entscheidungen diskutiert, die im Airlock Microgateway getroffen wurden.

Den guten Wein bewahren

Die Zeiten ändern sich schnell. Das gilt besonders für Trends in der Informationstechnologie (https://www.airlock.com/en/zero-trust-is-a-journey). Es ist aber nicht alles schon nach zwei Jahren überholt. In den letzten 20 Jahren ist viel Security-Expertise in das Airlock Gateway geflossen. Ein kurzer Blick auf die OWASP Top 10 Risikoliste bestätigt, dass viele der Probleme der ersten Ausgabe von 2003 immer noch bestehen. Das gilt auch für die kürzlich veröffentlichte OWASP API Security Top 10 Liste (https://www.airlock.com/en/secure-access-hub/components/api-gateway).

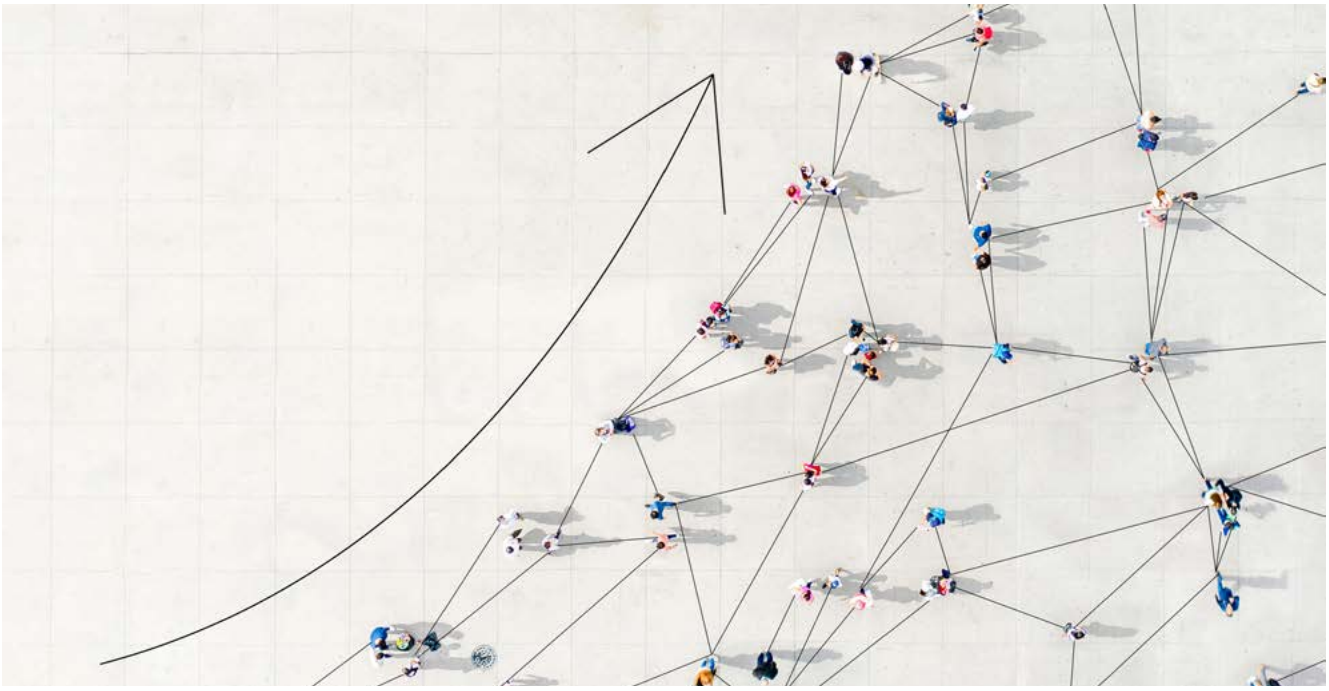
Es ist schwierig, die aktuellsten Angriffe und die neusten Umgehungsmechanismen zu verhindern, aber die eigentliche Herausforderung stellt die Feinjustierung der Regeln für wenige False Positives dar. Viele Regel-Sets erdrücken die Ingenieure geradezu mit einer Auswahl von Tausenden von Regeln. Aber wenn es um tatsächliche Deployments geht, erfordert jeder falsche Alarm eine manuelle Interaktion, was dem Hauptziel von DevOps zuwiderläuft: Automatisierung. Die Haupterausforderung liegt also darin, den „guten Wein", der über viele Jahre gereift ist, zu bewahren und in neue Schläuche zu füllen.

In neuen Schläuchen

Was sind die neuen Weinschläuche in der derzeitigen Ära? Docker ist ohne Frage das Standard Containerformat für das Deployment von Sicherheitskomponenten. Das grafische Benutzerinterface (GUI) bildete in den letzten Jahrzehnten den Mittelpunkt jeder neuen Software, aber die zwingende Anforderung der Automatisierung setzt der Nützlichkeit von Konfigurations-GUIs heutzutage Grenzen. Auch wenn GUIs dabei helfen können, schnell Konfigurations-Templates zu entwickeln, so können sie trotzdem einfach kein Teil einer automatisierten Pipeline sein. Die Bedienung der GUIs ist Handarbeit und erfordert Menschen. Im Vergleich zu automatisierten Skripten sind diese Interaktionen langsam, blockieren den gesamten Prozess und erfolgen nicht jeden Sonntag um 2 Uhr in der Frühe.

Wenn eine GUI von Menschen benutzt wird, um die Konfiguration zu bearbeiten, wird ein technisches Low-Level-Format generiert und gespeichert. Diese generierte Konfiguration wird dann von den Produktkomponenten gelesen. Die naheliegende Frage ist, ob es DevSecOps-Ingenieuren möglich ist, das technische Low-Level-Format mit ihren automatisierten Skripten direkt zu bearbeiten. Leider wurden die technischen Low-Level-Interfaces der Produktkomponenten nicht für diese Nutzung konzipiert. Sie verwenden mitunter interne Datenstrukturen, technische Identifikatoren, Redundanz, willkürliche Sortierregeln usw. Wenn man diese Dateien in die Quellcode-Versionsverwaltung einpflegt, ist es oftmals unmöglich, die relevanten Unterschiede zwischen zwei Versionen zu finden.

Wenn man die User Experience (UX) in Zeiten von DevOps bedenkt, dann ist der User ein Ingenieur und weiss, wie man programmiert. Das Hauptinterface zum Benutzer ist die Konfigurationsdatei, nicht das GUI. Dementsprechend ist das Wichtigste am Design von Produkten anhand der Bedürfnisse der Benutzer die Bereitstellung einer von Menschen lesbaren, gut formatierten und sauberen Syntax der Konfigurationsdatei. Das ist der Hauptgrund dafür, dass das Airlock Microgateway eine optimierte domänenspezifische Sprache (DSL) als Konfigurationsinterface verwendet. Die DSL baut auf YAML auf und wurde für DevOps-Ingenieure entwickelt: Sie ist leicht diffbar, bearbeitbar und mit Skripten modifizierbar.



Die Lücke mit formalen Spezifikationen überwinden

OpenAPI ist ein viel genutzter Standard zur Dokumentation moderner APIs. Häufig werden OpenAPI-Spezifikation automatisch in der Build-Pipeline generiert und sind für die Entwickler leicht zugänglich. Diese Spezifikationen beschreiben die öffentliche API, ihre Ressourcen, ihre Objektstrukturen und ihre Attribute präzise. Airlock Microgateway unterstützt die OpenAPI-Spezifikationen, um den Traffic zum geschützten Dienst zu filtern, und führt damit zu einer Win-Win-Situation für Sicherheit und DevOps.

Ein Sieg für die Sicherheit
OpenAPI-Spezifikationen stellen eine strenge Whitelist-Definition der API zur Verfügung. Wenn Attribute korrekt typisiert sind, reduziert dies Injection-Angriffe mit unerwarteten Inputdaten enorm. Im Kontext der Sicherheit steht „unerwartet" beinahe als Synonym für „nicht korrekt verarbeitet". Demzufolge bewirkt das automatische Beseitigen von möglichst vielen unerwarteten Situationen einen grossen Sicherheitsvorteil. Ausserdem kann die API nicht mehr nach undokumentierten Ressourcen und Attributen durchsucht werden, weil alle Aufrufe der formalen Spezifikation entsprechen müssen.

Ein Sieg für DevOps
Das automatische Durchsetzen einer Spezifikation in einem Microgateway führt DevOps und Security zusammen. Ein einzelner Commit kann zu einer aktualisierten OpenAPI-Spezifikation führen, die automatisch hochgeladen und vom Microgateway durchgesetzt wird. Echtes DevSecOps in Aktion!

Think Globally, Act Locally

„Global denken, lokal handeln“ ist ein Motto vieler Umweltschützer, die genau wissen, dass einzelne Personen nicht die ganze Welt retten können, aber sie können ihre unmittelbare Umgebung lenken. Bei der Entwicklung eines Microgateways können ähnliche Überlegungen einbezogen werden. Ein Microgateway wurde entwickelt, um einen einzelnen Service zu schützen, und kann dementsprechend nicht für die Sicherheit einer ganzen Umgebung zuständig sein. Aber wenn man an die Aufgaben des Microgateways denkt, ist klar, dass auch ein Bewusstsein für die Umgebung einbezogen werden muss:

- Was ist die Aufgabe des Microgateways?
- Welche Funktionen sind Out-of-Scope?
- Welche Services werden von der Plattform bereitgestellt?
- Welche Arten von Infrastrukturkomponenten wird das Microgateway nutzen?

Wir haben gründlich über diese Fragen nachgedacht und sowohl ein Microgateway als auch ein Edge Gateway in unserem Airlock-Produktportfolio. Wie können diese beiden Komponenten Aufgaben und Zuständigkeiten untereinander aufteilen? Die folgende Tabelle zeigt die wichtigsten Use Cases, für die die beiden Komponenten entwickelt wurden:

Frage	Airlock Gateway	Airlock Microgateway
Standort	Edge/Netzwerkperimeter	Sehr nah am geschützten Dienst
Hauptaufgabe	Veröffentlichen aller APIs	Absichern individueller APIs
Architektur	Reverse-Proxy Appliance mit Hardened OS	Container
Hohe Verfügbarkeit	Load Balancing und Failover für Backend-Service	Von Containerplattform verwaltet, mit Endpunkten für Health Checks
Authentifizierung/ Autorisierung	Policy Enforcement Point Proxy für Single-Sign-on (SSO) Injektion des internen Zugriffstokens	Prüfen des internen Zugriffstokens
Sicherheit	Basissicherheit	Strenge Sicherheit
Serviceintegration	Allgemeine schlüsselfertige Konfiguration	Detaillierte Serviceintegration einschliesslich Ausnahmen usw.
API-Schlüssel API-Nutzungspläne	Policy Enforcement Point	Prüfen des internen Zugriffstokens

Das Platzieren eines Edge Gateways vor den Microgateways ist keine technische Anforderung. Es ist eine Design-Entscheidung aufgrund der Tatsache, dass manche Aufgaben besser auf einem Edge-orientierten Gerät implementiert werden können und andere eng mit dem jeweiligen Service verbunden sein müssen.

Das Edge-orientierte Airlock Gateway hat eine eher generische Konfiguration, die die schnelle und nahtlose Integration neuer, von Microgateway-Instanzen geschützter Dienste ermöglicht. Der für das Microgateway verantwortliche Ingenieur hat detaillierte Kenntnisse über den geschützten Dienst. Typische Integrationsaufgaben, wie das Hinzufügen von Regelausnahmen oder URL Rewriting, werden auf dem Microgateway erledigt. So befinden sich alle Service-spezifischen Details in der Nähe des Services selbst, was Rollenkonflikte und Handover-Probleme mit dem Edge Gateway minimiert.

Bezüglich des Access Managements wurde das Microgateway so entworfen, dass es signierte Zugangstoken akzeptiert. Diese Zugangstoken werden von einem internen Autorisierungsserver (z.B. Airlock IAM) ausgegeben. Das Edge Gateway agiert als Policy Enforcement Point und interagiert mit dem Autorisierungsserver.

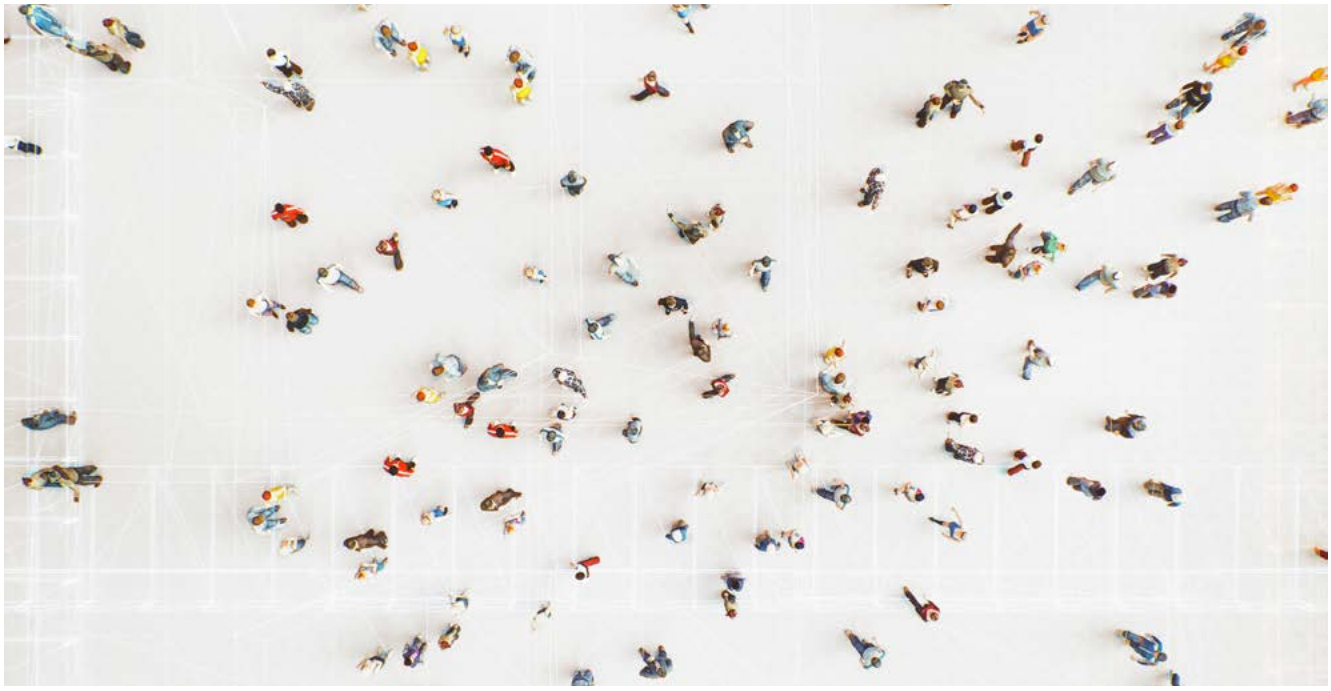
Unterstützung von Standard-Tools

Auch wenn es immer möglich ist, Artefakte von Herstellerwebseiten herunterzuladen und selbst zusammenzufügen, bevorzugen Ingenieure hierfür Standards.

Die Bereitstellung eines offiziellen Docker Hub Repositories für die Microgateway Artefakte erleichtert die Integration in die jeweilige Umgebung. Sie befreit die Ingenieure davon, sich mit der Versionsverwaltung, z.B. für die Verfügbarkeit der neusten Versionen, zu beschäftigen. Ebenso ist die Integration der Komponenten mit Kubernetes erheblich vereinfacht, wenn der Anbieter Helm Charts bereitstellt.

Staging der Konfiguration

Das Deployment von Microgateways in verschiedenen Staging-Umgebungen (Test, Integration, Pre-Production, Produktion) wirft die Frage auf, wie mit den Unterschieden zwischen diesen Umgebungen umgegangen wird. Es ist möglich, die Konfigurationsdatei in jeder Umgebung direkt zu patchen. Es ist aber sehr viel besser, in allen Umgebungen die gleiche Konfiguration zu verwenden. Das Airlock Microgateway unterstützt Platzhalter für Umgebungsvariablen in der DSL. Das ermöglicht das Einfügen Stage-spezifischer Konfigurationswerte von aussen in die Konfiguration hinein.



Darüber hinaus kann das Airlock Microgateway Hooks zum Ausführen benutzerdefinierter Skripte während des Deployments bereitstellen. Das kann für komplexere Anpassungsanforderungen nützlich sein, z.B. um OpenAPI-Dateien beim Deployment von einem zentralen Repository zu lesen.

Session Handling

Geschützte Dienste sind nicht immer komplett zustandslos. Das macht es schwierig, die Last auf mehrere Microgateway-Instanzen zu verteilen. Deswegen unterstützt das Airlock Microgateway die Integration eines gemeinsamen Session Stores mit Redis. Mit dem Session Store können zustandsbehaftete Serviceinteraktionen über mehrere Microgateway-Instanzen verteilt werden, ohne den Kontext zu verlieren. Praktischerweise ist die Integration der Redis-Datenbank Teil der Microgateway Helm Charts.

Logging und Reporting

„Unzureichendes Logging und Monitoring“ ist ein Problem, das auf beiden OWASP Top 10 Listen auftaucht, der traditionellen sowie der für API-Security. Der Airlock Secure Access Hub hat einen integrierten Logging- und Reporting-Stack auf der Basis von Elasticsearch und Kibana (ELK). Der Stack steht Out-of-the-Box mit der Airlock Gateway Appliance zur Verfügung, da die Appliance mit einem vollständigen Betriebssystem und diversen weiteren Komponenten geliefert wird. Das ist beim Microgateway keine Option, da es klein, leicht und an seine Hauptaufgaben angepasst sein muss.

Es gibt verschiedene Wege, um Microgateway-Logs zu integrieren. So kann zum Beispiel der Airlock Reporting Stack separat und losgelöst von der Appliance deployed werden. Die Microgateway-Logs können dann an die externe Elasticsearch-Datenbank gesendet werden und alle vordefinierten Airlock Dashboards sind direkt verfügbar. Im Falle eines bestehenden Elasticsearch Clusters können die Logs optional auch dort gespeichert werden. Dashboards müssten bei Bedarf separat importiert werden.

Microgateway Logs sind gut strukturiert und nutzen Key-Value-Paare für einfaches Parsing, ganz ähnlich wie die Konfigurationsdateien. So können sie auch einfach an andere Log Storages wie Splunk weitergeleitet werden. Zur einfachen Integration mit SIEM-Systemen wurden Microgateway Logs von ArcSight zur Einhaltung des CEF-Format (Common Event Format) zertifiziert.

Damit es funktioniert

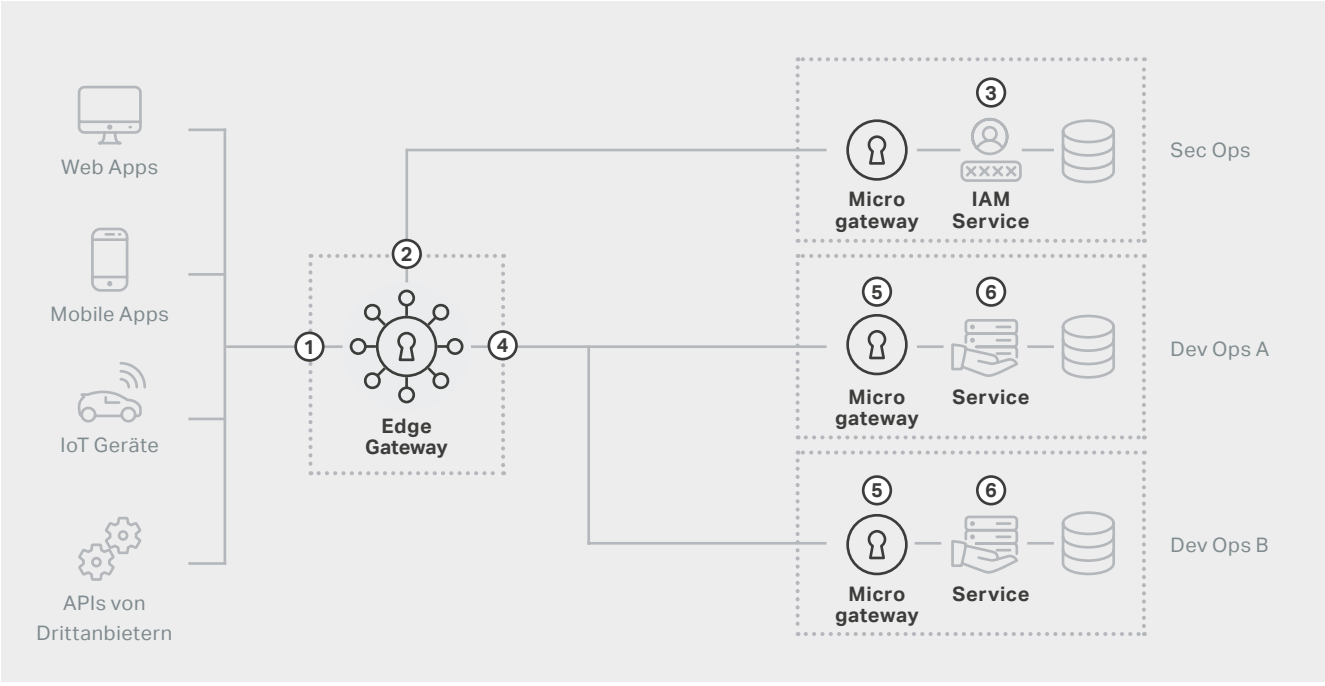
Zusammenarbeit der Komponenten

Microgateways müssen mit den Diensten, die sie schützen, und in den meisten Fällen auch mit einem Identity and Access Management System integriert sein. Die folgende Liste zeigt alle Komponenten, die für ein Produktionssetup benötigt werden.

- **Edge Gateway:** Das (Edge) Gateway sitzt an der Grenze des Netzwerks und ist die erste Komponente, die eine Anfrage entgegennimmt. Es wird vom zentralen SecOps-Team verwaltet.
-
- **Identity and access management (IAM) service:** Identitätsprovider und Access Management Service. Er wird vom zentralen SecOps-Team verwaltet.

- **Microgateway:** Schlanke Sicherheitskomponente, die einen bestimmten Service schützt. Es wird vom DevOps-Team des geschützten Dienstes verwaltet.

- **Service/Dienst:** Business-Anwendung oder -API, die vom gleichen DevOps-Team wie das Microgateway verwaltet wird.



Schritte einer Anfrage in der Microgateway-Architektur

Das obige Diagramm zeigt alle relevanten Komponenten und wie sie interagieren.
Es folgen ihre Aktivitäten im Detail:

① **Edge Gateway**
- empfängt die eingehende Anfrage vom Client
- beendet die TLS-Session
- blockiert Bots und DoS-Angriffe
- prüft API-Schlüssel und setzt gegebenenfalls Quotas und Rate Limits durch
- nutzt generische Sicherheitsfilter, um OWASP und andere Angriffe zu blockieren
- erzeugt zur einfacheren Nachverfolgbarkeit entlang der Call Chain eine einzigartige Request-ID und Session-ID

② Falls nicht authentifiziert:
Edge Gateway
- setzt Authentifizierung für alle nichtöffentlichen Dienste durch
- prüft, dass alle notwendigen Rollen vorhanden sind
- delegiert die eigentliche Arbeit an einen IAM-Dienst

③ **IAM-Dienst**
- verifiziert die Identität des Callers (z.B. durch Prüfen der Anmeldedaten)
- sammelt Identitätsattribute aus einer Nutzerdatenbank oder einem Verzeichnis (einschliesslich Identitätsattributen wie Organisation, Rollen und Zugriffsrechte)
- erzeugt einen *signierten Zugangstoken* mit einigen dieser Identitätsattribute des Benutzers und sendet ihn zurück an das Edge Gateway

④ **Edge Gateway**
- nach erfolgreicher Authentifizierung:
- leitet die Anfrage zusammen mit dem Zugangstoken, der Request-ID und der Session-ID an den Downstream weiter

⑤ **Microgateway**
- prüft die Signatur des Zugriffstokens und extrahiert daraus die Identitätsattribute
- prüft die Anfrage:
  – Hält sie die Default-Regeln ein?
  – Erfüllt sie die Anforderungen der OpenAPI-Spezifikation?
- leitet die Anfrage an den Dienst weiter.

⑥ **Service/Dienst**
- extrahiert Identitätsattribute aus dem Zugangstoken
- überprüft die Autorisierung des Callers erneut (auf Funktions- und Objektebene)
- loggt die Request-ID und Session-ID mit jeder Nachricht, zur Nachvollziehbarkeit entlang der gesamten Call Chain.
- schliesst Zugangstoken und Session-/Request-ID ein, wenn andere Services aufgerufen werden.

Vorteile der Microgateway-Architektur

Microgateways sind ein wichtiges Werkzeug, um den Weg der DevOps-Teams bei der Umsetzung von Zero-Trust-Architekturen zu unterstützen und so zu DevSecOps-Teams zu werden. Microgateways helfen auf vielfache Weise:

- **Agilität:**
  Mehrere unabhängige Entwicklungsteams profitieren von der bestehenden Infrastruktur. Da die Konfiguration des Microgateways vom Entwicklungsteam gewartet wird, erfordert eine neue Serviceversion wenig bis keine Koordinierung mit dem Gateway-Administrator.

- **Skalierbarkeit und Verfügbarkeit:**
  Microgateways werden direkt mit ihren Services deployt und skalieren mit ihnen. Die Funktionen der Microgateways stellen sicher, dass Session-Informationen unabhängig davon zur Verfügung stehen, welche Microgateway-Instanz die Anfrage bearbeitet.

- **Time-to-Market:**
  Microgateways erzwingen die Authentifizierung, bevor der Zugriff auf den Dienst erlaubt wird. Das beseitigt die Notwendigkeit, diese Funktionen in jeden einzelnen Dienst einzuarbeiten. Da diese kritischen Sicherheitsaufgaben von der standardisierten Infrastrukturkomponente übernommen werden, können die Entwickler mehr Zeit in Business Features investieren.

- **Massgeschneiderte Sicherheit:**
  Microgateways haben einen sehr kleinen Fussabdruck, sodass die Entwickler sie während des gesamten Entwicklungsprozesses nutzen können. Das stellt sicher, dass der Service mit dem Microgateway funktioniert und ermöglicht es dem Entwickler, die Filterregeln für optimale Sicherheit zu konfigurieren. Integrationsfehler und Sicherheitsprobleme werden viel früher im Entwicklungszyklus bemerkt, lange bevor der Dienst live geht.

Falls Sie mehr über Microgateways erfahren möchten, diskutieren Sie das mit unseren Spezialisten und sehen Sie, wie wir Ihnen auf Ihrem Weg zu DevSecOps helfen können.

Autoren

Aarno Aukio

CTO & Partner VSHN AG

Traumberuf Software Operations Engineering – damit hat Aarno gestartet und daraus in 6 Jahren die VSHN (ausgesprochen vɪʒn wie «vision») mit 45 Mitarbeitern aufgebaut. Als erster Schweizer Kubernetes Certified Service Provider und führender Partner für DevOps, Docker, Kubernetes, OpenShift, Rancher & 24/7 Cloud Operations. Als Open Source Unternehmen sind nicht nur Sourcecode sondern auch Servicedefinitionen öffentlich verfügbar und verkörpern die Kernwerte des Unternehmens: Offenheit, Kollaboration und Verantwortung übernehmen (wie im öffentlichen Mitarbeiterhandbuch unter https://handbook.vshn.ch/hb/values.html dokumentiert).

Michael Doujak

Product Manager Airlock

Michael Doujak hat nach seinem Studium an der ETH Zürich verschiedene Stationen durchlaufen. Seit 20 Jahren hat er verschiedene Projekte und Lösungen im Bereich von Identity Management geplant und umgesetzt. Besonders herauszustreichen ist der Aufbau von SwissSign als Herausgeber von qualifizierten Zertifikaten in der Schweiz, der Aufbau der Patientendossier Plattform «MonDossierMedical», der Aufbau der EPD Infrastruktur und der Zuweiser Plattform Lösung der Schweizerischen Post. Er ist ein profunder Kenner der Materie, von der tiefen Technik bis in die geschäftlichen Anwendungsfälle hinauf. Michael Doujak ist heute als Product Manager für Airlock bei Ergon Informatik tätig.