AIRLOCK®
SECURE ACCESS HUB

heise Business Services

Sicherheit für cloud-native Anwendungen

Apps in Kubernetes

In Kooperation mit unserem Partner

ISOVALENT
now part of cisco

www.airlock.com

Inhalt

Wie es Unternehmen gelingt, die Sicherheit von Web-Applikationen und APIs in Kubernetes zu gewährleisten.

—

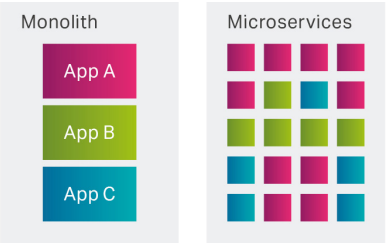
Moderne Anwendungen entstehen zunehmend auf der Basis von Microservices in Kubernetes-Umgebungen. Diese Methode, eine Anwendung in kleine Module zu zerlegen, erfordert eine Anpassung der Anwendungssicherheit.
Der Sicherheit von Kubernetes kommt dabei eine besondere Bedeutung zu, da Kubernetes die Container orchestriert, also für deren Bereitstellung und Verwaltung verantwortlich ist. Dieses Dokument beschreibt die erforderliche Umstellung von traditionellen Sicherheitsansätzen und stellt zeitgemässe Sicherheitspraktiken für Kubernetes vor.

Das Security-Dilemma für CISOs: Agilität und Sicherheit dürfen nicht im Widerspruch stehen

DevOps und Kubernetes sind die Basis der notwendigen Agilität und Innovation

Viele bestehende Web-Applikationen erfüllen die heutigen Anforderungen nicht mehr. Unternehmen benötigen Anwendungen, mit denen sie ihre Innovationen vorantreiben, ihre geschäftliche Agilität erhöhen und ihre Widerstandsfähigkeit steigern können, so die IDC-Studie „Cloud in Deutschland 2023“[1].

Der Markt wird immer dynamischer, der Wettbewerb immer härter. Neue digitale Dienste müssen schnell verfügbar sein, Unternehmen müssen flexibel auf neue Anforderungen reagieren können. Da der Faktor Zeit eine entscheidende Rolle spielt, ist es wichtig, dass sich neue Anwendungen schnell und mit geringem Aufwand herstellen und weiterentwickeln lassen. Deshalb gehen immer mehr Unternehmen weg vom monolithischen Ansatz, der eine Anwendung als grosse Einheit betrachtet. Stattdessen erfolgt die Aufteilung in sogenannte Microservices. Dabei handelt es sich um kleine, eigenständige Einheiten, die jeweils einen Teil der Anwendungslogik bereitstellen und über definierte Schnittstellen (APIs) miteinander kommunizieren. Dieser Ansatz verkürzt die Entwicklungszeiten, erhöht die Agilität und Flexibilität in der Software-Entwicklung und erlaubt die Nutzung der Cloud-Eigenschaften als Reaktion auf Phasen hoher Last.

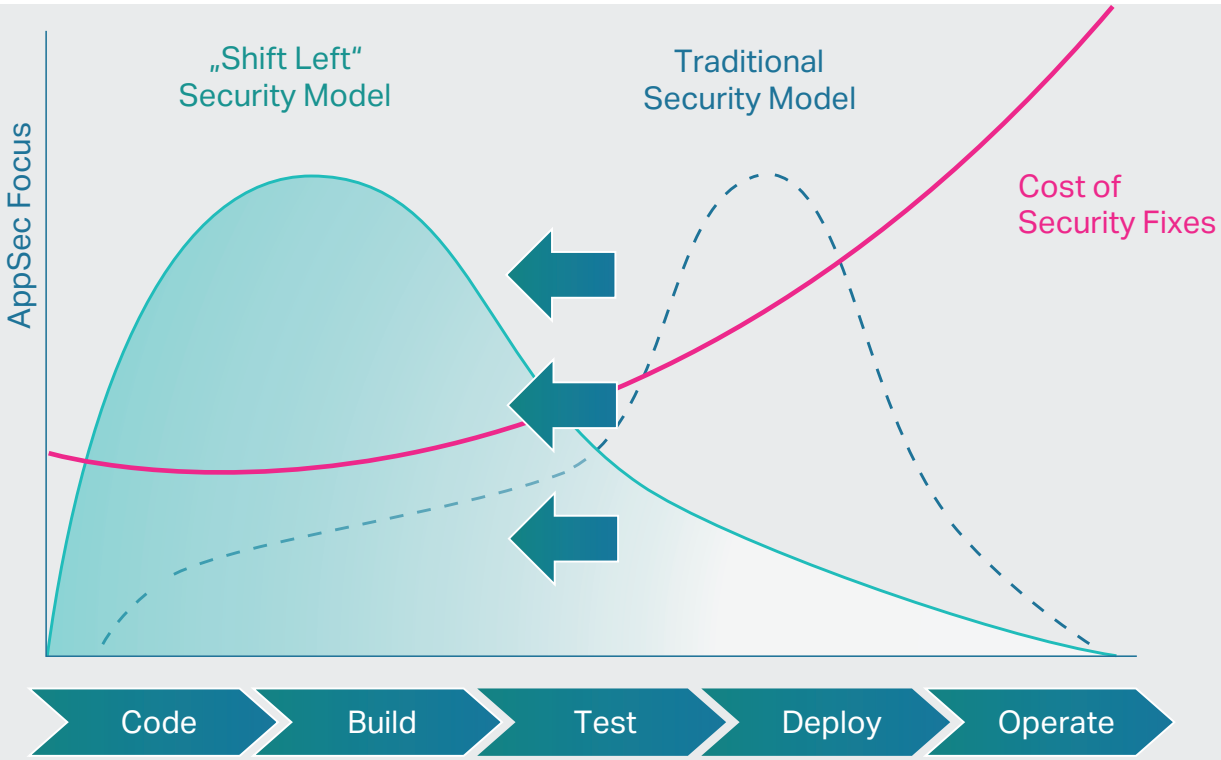


Jeder Microservice wird dabei mit seiner Laufzeitumgebung und allen notwendigen Abhängigkeiten in einem Container gekapselt. Damit die Container optimal zusammenarbeiten, kommt eine Orchestrierungs-Software zum Einsatz. Als De-facto-Standard hat sich die Open-Source-Software Kubernetes[2] etabliert. Diese automatisiert die Bereitstellung sowie die bedarfsabhängige Skalierung und Verwaltung von containerisierten Anwendungen. Kubernetes sorgt für eine hohe Verfügbarkeit, indem es den Status aller Microservices überwacht, um bei Ausfall einen automatischen Neustart, bei Fehlern eine umgehende Reaktion und bei Updates eine unterbrechungsfreie Aktualisierung zu ermöglichen.

Laut Gartner-Report „The CTO's Guide to Containers and Kubernetes“[3] werden bis 2027 weltweit mehr als 90 Prozent der Firmen containerisierte Anwendungen in der Produktion einsetzen. Dabei empfinden 71 Prozent der Fortune-100-Unternehmen Kubernetes als wichtigstes Container-Orchestrierungstool, so die Cloud Native Computing Foundation (CNCF)[4]. Das Marktforschungshaus Gartner sieht in Kubernetes den De-facto-Standard für die Container-Orchestrierung[5].

---

[1] https://www.idc.com/getdoc.jsp?containerId=prEUR150529923
[2] https://kubernetes.io/de/
[3] https://www.gartner.com/en/documents/3988026
[4] https://www.cncf.io/reports/kubernetes-project-journey-report/
[5] https://www.gartner.com/interactive/hc/4526999



Die Anwendungssicherheit darf nicht zum Hemmschuh werden

Die für die hohe Innovationskraft nötigen kurzen Entwicklungszyklen erzeugen Stress, wenn jede Änderung am Code vor der Auslieferung manuell überprüft und abgesegnet werden muss. Es frustriert die Entwicklerteams schnell, wenn sich Aktualisierungen verzögern und nur stockend live gehen. Darum sollte eine zum agilen Arbeiten passende Sicherheitslösung den Entwicklungsprozess möglichst früh begleiten, möglichst automatisiert arbeiten und trotzdem manuelle Übersteuerungen erlauben.

Die Veränderungen in der Entwicklung hin zu Kubernetes müssen auch auf den verschiedenen technischen Ebenen Berücksichtigung finden, denn Kubernetes sorgt nicht automatisch für eine umfassende Sicherheit der Microservices oder der daraus zusammengesetzten Anwendungen. Vielmehr bedarf es weiterer Sicherheitsmassnahmen beim Einsatz von Containern[6] und Kubernetes[7], um eine durchgehende Anwendungssicherheit zu erreichen – von den Microservices über die Container und die Container-Orchestrierung bis hin zur Applikation.

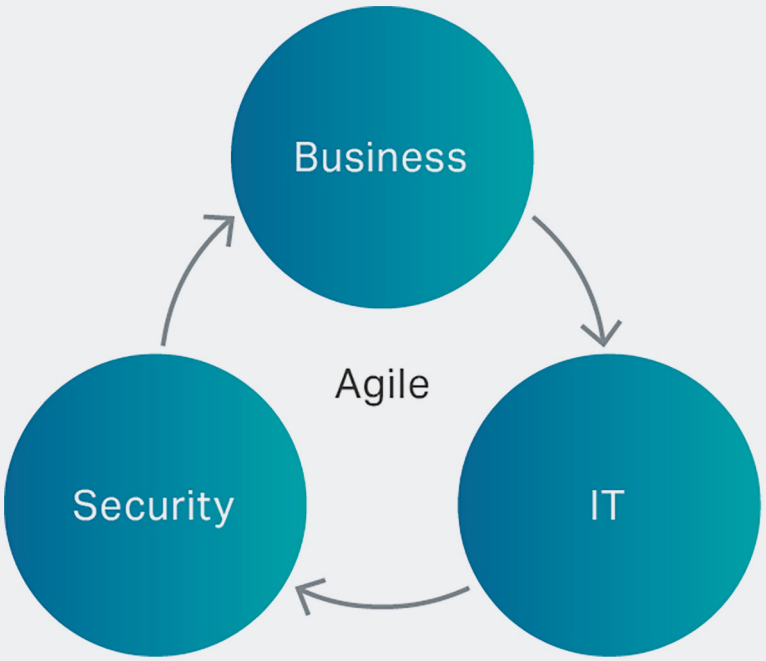
Der Bedarf für neue Sicherheitsmassnahmen bei Web-Apps und APIs steigt: 83 Prozent der Firmen haben mehr als zehn schutzbedürftige APIs/Web-Applikationen im Einsatz, wie die Studie „Application und API-Security im Container-Umfeld 2022“[8] zeigt. 21 Prozent der Unternehmen sagen sogar, mangelnde Sicherheit bei ihren Web-Apps könnte ihre Existenz gefährden. Deshalb kommt der sicheren Nutzung von Kubernetes eine entscheidende Bedeutung bei der Anwendungsmodernisierung zu. Sicherheitsverantwortliche sollten sich mit der Sicherheit von Kubernetes und Containern genauer befassen, denn dies spielt eine wichtige Rolle bei der modernen Applikationssicherheit.

[6] https://docs.docker.com/develop/security-best-practices/
[7] https://kubernetes.io/docs/concepts/security/
[8] https://www.airlock.com/insights/airlock-blog/business-blog/studie-application-und-api-security-im-container-umfeld-2022

Die Lösung des Dilemmas: Die richtige Kubernetes-Sicherheit verbindet Agilität und Security

DevSecOps: Die Sicherheit für Kubernetes muss selbst agil sein

Wenn man die DevOps-Prozesse der Entwicklerinnen und Entwickler in agil arbeitenden Teams nicht behindern und gleichzeitig ein hohes Mass an Sicherheit herstellen möchte, müssen neue Lösungsansätze her. Die Anwendungssicherheit muss sich transformieren, um mit der Entwicklung Schritt halten zu können.



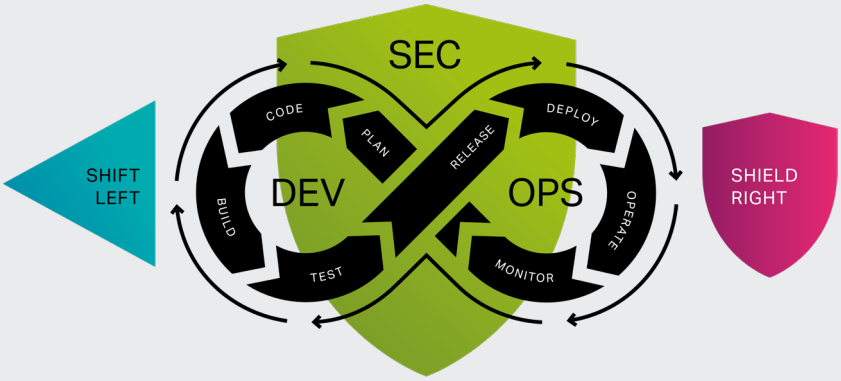
Wie der „State of Kubernetes security report 2023"[9] von Red Hat zeigt, kämpfen viele Unternehmen mit spezifischen Sicherheitsrisiken bei der cloudnativen Entwicklung. Es fällt auf, dass 67 Prozent der Befragten ihre cloudnativen Vorhaben aufgrund von Sicherheitsbedenken verlangsamen mussten. Dabei soll die Anwendungsmodernisierung über den Container-Kubernetes-Ansatz eigentlich die Entwicklungsarbeiten beschleunigen. Sicherheit bleibt so der Hemmschuh der Innovation. Der Bericht von Red Hat kommt zu dem Schluss, dass cloudnative Lösungen auch cloudnative Sicherheitslösungen erfordern, um die Agilität, Flexibilität und Geschwindigkeit bei DevOps beibehalten zu können. Klassische Security-Ansätze können dem verteilten Charakter und der Dynamik von cloudnativem Computing nicht gerecht werden.

[9] https://www.redhat.com/en/resources/state-kubernetes-security-report-2023

Cloudnative Security ist im Gegensatz zum klassischen Ansatz direkt in die DevOps-Prozesse integriert und kommt nicht erst zu einem späteren Zeitpunkt als Add-on hinzu. Sicherheit darf die cloudnative Entwicklung nicht ausbremsen, sondern muss ein Bestandteil von ihr sein. DevOps verwandelt sich dadurch in DevSecOps, Sicherheit fliesst also in die Entwicklung und den Betriebslebenszyklus ein.

„Shift left" und „Shield right": Microservices benötigen einen durchgehenden Schutz wie die Web-Applikationen

Umfassende Anwendungssicherheit erstreckt sich von der Web-App bis hin zu den Microservices und verbindet die Konzepte „Shift left" und „Shield right". „Shift left" steht für die Praxis, Sicherheit, Qualität und Leistung in eine frühe Phase des Entwicklungslebenszyklus zu integrieren. Traditionell finden Sicherheitstests und Validierung erst am Ende des Entwicklungszyklus statt, nachdem die Anwendung bereits existiert. Dieser Ansatz kann jedoch dazu führen, dass Sicherheitslücken erst spät entdeckt werden, was ihre Behebung teurer und zeitaufwendiger macht.



Durch „Shift left" fliessen Sicherheitstests, die Integration von Sicherheitslösungen und Validierungen zu einem früheren Zeitpunkt in den Entwicklungsprozess ein und beeinflussen auch die Codeentwicklung, -überprüfung und die automatisierten Tests. Dies ermöglicht es Entwicklerinnen und Entwicklern, Sicherheitsprobleme früher im Entwicklungszyklus zu erkennen und zu beheben. Es reduziert das Risiko, dass sich Sicherheitslücken in das Endprodukt einschleichen oder sich die Inbetriebnahme aufgrund bekannter Sicherheitsprobleme verzögert.

Unter „Shield right" versteht man hingegen die Praxis der fortlaufenden Überwachung und des Schutzes von Anwendungen und Systemen, nachdem diese in der Produktion bereitgestellt wurden. Um „Shield right" zu erreichen, können Unternehmen zum Beispiel SIEM-Tools (Security Information and Event Management), Web Application and API Protection (WAAP) sowie Identity and Access Management (IAM) mit starker Authentifizierung einsetzen.

„Shift left" und „Shield right" ergänzen einander und bieten einen umfassenden Anwendungsschutz in allen Phasen des Software-Entwicklungszyklus. Eine Lösung wie Airlock Microgateway[10] hilft dabei, die Lücke zwischen den beiden Ansätzen zu schliessen: Sie ermöglicht die Integration des Laufzeitschutzes bereits zur Entwicklungs- oder Build-Zeit.

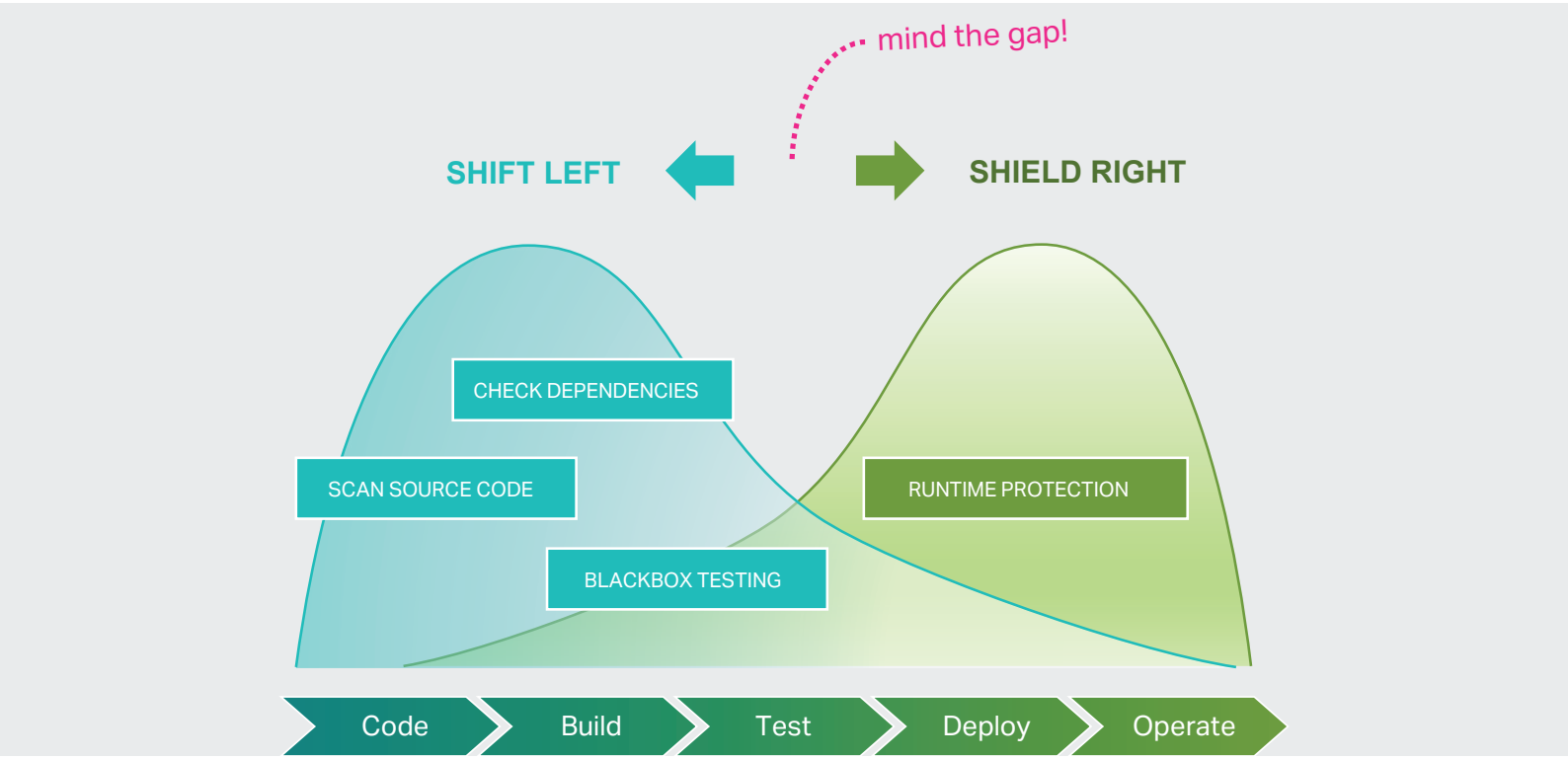
[10] https://www.airlock.com/microgateway

Microgateways bilden die Basis von „Shield right“

Vorstellen kann man sich Microgateways als cloudnative WAAPs. Dabei versteht man unter WAAP (Web App and API Protection) ein spezielles Werkzeug zum Schutz von Web-Anwendungen und APIs. Es umfasst eine Web Application Firewall (WAF), wehrt Attacken auf interne und öffentliche APIs ab und schützt vor HTTP-Attacken, bösartigen Bots sowie DDoS-Angriffen auf Anwendungsebene (Layer 7). Das Tool arbeitet mit einem IAM-System (Identity and Access Management) zusammen, um eine kontinuierliche, risikoabhängige Authentifizierung (Continuous Adaptive Trust) zu ermöglichen.

Ein Microgateway arbeitet unmittelbar vor dem Microservice, kein Zugriffsversuch kommt daran vorbei. Bei jedem Aufruf findet eine erneute Berechtigungsprüfung statt, ganz im Sinne einer Zero-Trust-Architektur. Microgateways erzwingen die Authentifizierung, bevor der Zugriff auf den Service erfolgen kann. Das beseitigt die Notwendigkeit, diese Funktionen in jeden einzelnen Service einzuarbeiten. Die Microgateways kommen bereits während der Entwicklung und bei den Tests zum Einsatz. Entwicklungsteams agieren so unabhängiger von der Security-Abteilung und der IT-Administration, sie können die Sicherheit agil in ihre DevOps-Prozesse integrieren. Entwicklungsteams kennen ihre Services am besten, entsprechend können sie die Sicherheitsregeln selbst definieren, und zwar im von den Sicherheitsverantwortlichen vorgegebenen Rahmen. Entwickler können die Regeln auch selbst durchsetzen, denn die Konfiguration per Kubernetes Custom Resource (YAML-Datei) erlaubt die Automatisierung und Integration in die DevSecOps-Prozesse. Die Servicespezifikation im OpenAPI-Format[11] deklariert die erlaubten Aufrufe und dient sowohl als Dokumentation als auch als Sicherheits-Allow-List.

Unter dem Strich gelingt es, firmenweite Sicherheitsrichtlinien durchzusetzen, gleichzeitig sind anwendungsspezifische Regeln durch die Entwicklungsteams möglich. So entfallen manuelle Übergaben und die Koordination mit SecOps weitgehend. Da die



[11] https://www.openapis.org/

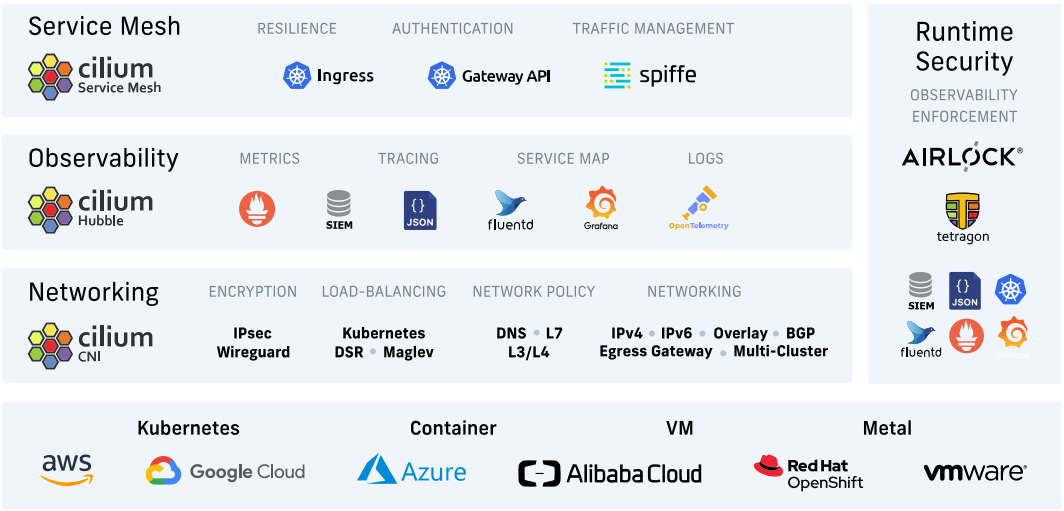
Entwickler die Konfiguration des Microgateways warten, erfordert eine neue Serviceversion wenig bis keine Koordinierung mit dem Administrator. Die Security-Abteilung ist nicht mehr der gefürchtete Bremsklotz, sondern Partner und Unterstützer.

Die Sicherheit bekommt mit Cilium ein weiteres cloudnatives Fundament

Ein Microgateway wie Airlock Microgateway kann zusammen mit einem vollständigen Toolstack wie Cilium betrieben werden, der sich perfekt in Kubernetes integrieren lässt. Die Cilium-Funktionen können so um den Schutz für Web-Applikationen und APIs durch das Microgateway erweitert werden.

Cilium[12] liefert mit Funktionen wie transparenter Verschlüsselung, gegenseitiger Authentifizierung, Sicherheitsmonitoring, erweiterten Netzwerkrichtlinien, Egress-Gateway und Laufzeitdurchsetzung cloudnative Sicherheit. Das Open-Source-Projekt Cilium bietet Vernetzung, Sicherheit und Beobachtbarkeit für Kubernetes-Cluster und andere Containerumgebungen. Cilium basiert auf einer Technologie namens eBPF[13] (extended Berkeley Packet Filter), die Netzwerksteuerungslogik, Sicherheitskontrollen und Observability-Funktionen direkt in den Kernel integrieren kann. Cilium nutzt eBPF, um leistungsstarke Netzwerk-, Multi-Cluster- und Multi-Cloud-Funktionen, Verschlüsselung, Lastausgleich sowie Netzwerksicherheitsfunktionen bereitzustellen.

Die meisten Cilium-Funktionen basieren auf dem Cilium-Container-Networking-Interface-Plug-in. Kubernetes stellt Container in Einheiten namens Pods bereit. Ein Pod enthält einen oder mehrere Container, die Zugriff über eine einzelne IP-Adresse bieten. In Cilium erhält jeder Pod seine eigene IP-Adresse vom Knotenpräfix des Knotens, auf dem der Pod ausgeführt wird. Cilium verwendet für seinen Schutz aber nicht nur IP-Adressen, sondern nutzt einen identitätsbasierten Ansatz, der weitere Merkmale von Pods und Containern berücksichtigt. Mit Cilium gelingt es, eine dedizierte Netzwerksicherheitsrichtlinie zu definieren. Das stellt sicher, dass Pods nur mit den Pods kommunizieren können, auf die sie auch Zugriff benötigen. Dadurch entsteht ein gezielter Zugriffsschutz. Cilium kommt zum Einsatz, um seine Funktionen für die unteren OSI-Schichten (wie Routing, Observability) mit eBPF bereitzustellen.



[12] https://docs.cilium.io/en/latest/security/
[13] https://ebpf.io/

Wenn das Microgateway sich mit Cilium austauscht, fördert das die Sicherheit insgesamt. Denn das Gateway kann dann zum Beispiel Informationen über den Netzwerkverkehr dazu nutzen, um Verhaltensänderungen der Anwendungen zu erkennen und darauf zu reagieren. Weil Cilium Sichtbarkeit auch auf Layer 7 (Anwendungsschicht) hat, kann es den API-Kontext verstehen und darauf basierend Regeln durchsetzen. Dadurch ist es dem Microgateway möglich, fein abgestufte Sicherheits- und Routing-Richtlinien auf der Grundlage von API-Anfragen anzuwenden.

Die Vorteile des Airlock Microgateways

Airlock Microgateway wurde für moderne Kubernetes-Architekturen konzipiert und lässt sich einfach in GitOps-Prozesse (codebasierte Infrastruktur und Betriebsabläufe, die sich auf Git als Quellkontrollsystem stützen) einbinden. Es hilft DevOps-Engineers und Applikationsteams, ihre Services mit wenig Aufwand vor unerlaubten oder bösartigen Zugriffen zu schützen.

Airlock Microgateway zeichnet sich aus durch:

- Agilen Schutz von Microservices dank einfacher Einbindung in moderne Servicearchitekturen.
- Engmaschige Serviceabsicherung: Nur was explizit deklariert ist, wird akzeptiert.
- Mehrstufige Sicherheitsfilter als Schutz vor bekannten Angriffen (wie OWASP Top 10[16]) und Zero-Day-Exploits wie Log4Shell.
- Ein bewährtes Regelwerk von Experten, das sich für alle Arten von Web-Apps und APIs eignet.
- Die Eignung für den Einsatz in allen wichtigen Kubernetes-Distributionen (inkl. OpenShift).
- Die sichere Durchsetzung der Richtlinien durch Unterstützung von Gatekeeper[17] (Policy Controller für Kubernetes) und Kyverno[18] (Kubernetes Native Policy Management).
- Das Zusammenspiel mit Kubernetes (Istio-Service-Mesh[19] und Cilium-Unterstützung, Operator und CRDs (Custom Resource Definitions[20]), automatische Sidecar-Installation[21] und Hot Reload.
- Eine modulare Konfiguration, die das Templating mit Kustomize[22] oder Helm[23] erlaubt.
- Helm-Charts für die einfache Provisionierung.
- Die erleichterte Überwachung und Analyse dank Metriken, die in Prometheus[24], gesammelt werden können, sowie strukturierte Logs im ECS-Format (Elastic-Common-Schema).
- Plug-ins für moderne IDEs (integrierte Entwicklungsumgebungen) zur automatischen Validierung, Code-Vervollständigung und Tooltips beim Editieren der Konfiguration.
- Die Unterstützung aller führenden Cloud-Plattformen (cloudagnostisch).
- Starke Verschlüsselungsmethoden und -protokolle.
- Die Einhaltung der Compliance-Vorgaben (PSD2, DSGVO, FINMA und BaFin-Regularien) durch strikte Datenhoheit (keine Datenübermittlung an Dritte).
- Die sichere Zugriffsverwaltung im Zusammenspiel mit Airlock IAM[25] (u. a. OAuth, OIDC, SAML).
- Den integrierten Authentifizierungs-Check (u. a. JWT-Token-Validierung).
- Load Balancing (Verteilung der Last auf mehrere Serviceinstanzen).

Praxisbeispiele für den Einsatz von Airlock Microgateways:

- „IT-Security für über 200 Millionen Kunden" – dacadoo und Airlock Microgateway[26]
- „Agile Sicherheit für das Asset-Management" – AssetMetrix und Airlock[27]

[16] https://owasp.org/www-project-top-ten/
[17] https://github.com/open-policy-agent/gatekeeper
[18] https://kyverno.io/
[19] https://istio.io/latest/about/service-mesh/
[20] https://kubernetes.io/docs/tasks/extend-kubernetes/custom-resources/custom-resource-definitions/
[21] https://kubernetes.io/docs/concepts/workloads/pods/sidecar-containers/
[22] https://kustomize.io/
[23] https://helm.sh/docs/chart_template_guide/
[24] https://prometheus.io/
[25] https://www.airlock.com/secure-access-hub/komponenten/iam
[26] https://www.airlock.com/fileadmin/content/07_Airlock-PDFs/201125_AIR_Referenz_Dacadoo_de_v02-01.pdf
[27] https://www.airlock.com/fileadmin/content/07_Airlock-PDFs/211222_AIR_Referenz_AssetMetrix_de_v03-01.pdf

Die moderne Anwendungs-sicherheit ist agil und innovativ

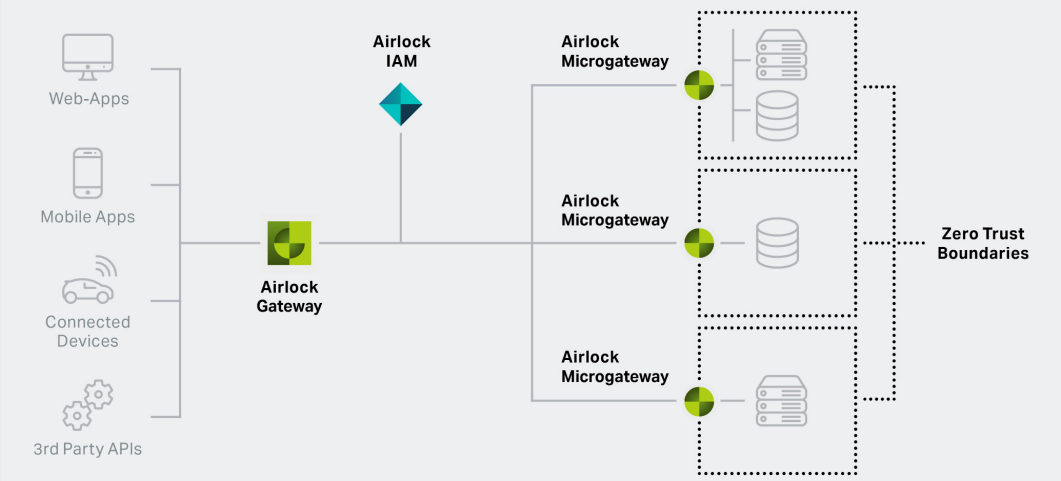
Microgateways, WAAP und IAM arbeiten Hand in Hand

Microgateways und die WAAP-Technologie auf Applikationsebene ergänzen sich: Bei einer Kombination aus Microgateways und zentralen Gateways profitiert jede Applikation von den zentralen Sicherheitskomponenten (API-Gateway, Web Application Firewall und Access Management). Zusätzlich geniessen die aus Microservices gebildeten Applikationen den Schutz durch die vor den Microservices geschalteten Microgateways. Mithilfe von Microgateways wird nicht nur der externe Netzwerkverkehr (North-South-Traffic), sondern auch der Datenverkehr zwischen den Containern innerhalb der Cloud-Umgebung (East-West-Traffic) gefiltert.

Gleichzeitig senken die Microgateways die Komplexität eines zentralen Gateways. Statt nur auf eine zentrale Filterung zu setzen, erhält jede Anwendung oder API zusätzlich ein eigenes kleines Schutzsystem. Das Anwendungsteam ist selbst eine Art WAAP-Betreiber und erhält damit mehr Autonomie bei den Sicherheitsregeln. Diese dezentralisierte WAAP-Ausprägung ermöglicht eine Zero-Trust-Architektur, bei der jeder Zugriff vor Ort nochmals überprüft und autorisiert wird.

Auch das Identity and Access Management (IAM) unterstützt die moderne Applikationssicherheit, da es alle externen Identitätsnachweise (Token) prüft und anschliessend einen einzigen, intern gültigen Token ausstellt. Die Microgateways müssen dann nur jeweils diesen internen Token prüfen. Im IAM finden neue Identity-Provider Berücksichtigung, was die schlanken Microgateways entlastet.

Microgateway, WAAP und IAM arbeiten so gemeinsam für die umfassende Applikationssicherheit. Moderne Sicherheitskonzepte wie Zero Trust („Vertraue niemandem, überprüfe alles“) lassen sich durch das Zusammenspiel von IAM, WAAP und Microgateways bis auf die Ebene der Microservices durchsetzen.



Gateway: Das Gateway sitzt an der Grenze des Netzwerks und nimmt als erste Komponente eine Anfrage entgegen. Es wird vom zentralen SecOps-Team verwaltet.
Identity and Access Management (IAM): Identitätsprovider und Access-Management-Service, verwaltet vom zentralen SecOps-Team.
Microgateway: Schlanke Sicherheitskomponente, die einen bestimmten Service schützt und vom DevOps-Team des geschützten Dienstes verwaltet wird.
Service/Dienst: Business-Anwendung oder -API, die dem gleichen DevOps-Team wie das Microgateway untersteht.

Fazit: Das sind die Empfehlungen für die Sicherheit von Apps in Kubernetes

Die Sicherheit von Kubernetes erfordert eine umfassende Strategie und die Umsetzung bewährter Sicherheitspraktiken. Durch die Implementierung dieser Massnahmen können Sie die Integrität und Verfügbarkeit Ihrer containerisierten Anwendungen gewährleisten. Dabei helfen Ihnen diese Empfehlungen:

- Moderne Anwendungen entstehen agil auf Basis von Microservices, Containern und Kubernetes. Die Security muss Teil dieses Entwicklungsprozesses sein und sich nahtlos in die DevOps-Prozesse einfügen.
- Verwenden Sie einen Technik-Toolstack, bei dem die einzelnen Werkzeuge von vornherein auf Zusammenarbeit abgestimmt sind, um den Konfigurationsaufwand zu minimieren.
- Anstatt Zugriffe auf Applikationen und Microservices erst nachträglich einzuschränken, sollte die Zugriffskontrolle von Beginn an im Sinne einer hohen Anfangssicherheit verfügbar sein.
- Die Sicherheitsfunktionen sollten sich nicht auf einzelne Clouds beschränken, sondern plattformübergreifend zur Verfügung stehen.
- Die Sicherheit für Microservices, Container und Kubernetes entscheidet über die Einhaltung von Compliance-Vorgaben, sie muss aber auch selbst compliancegerecht sein. Dazu gehört insbesondere, die Übermittlung vertraulicher Daten an Dritte auszuschliessen.
- Für die Kommunikation auf Ebene der Microservices und der Applikationen sollten sichere Verschlüsselungsmethoden zum Einsatz kommen.
- Starke Authentisierung und die Umsetzung von Zero Trust sollte schon auf der Ebene der Microservices beginnen.
- Die Sicherheit muss von der Entwicklung bis in die Produktionsphase kontinuierlich durch die Kombination der Prinzipien „Shift left" und „Shield right" gewährleistet werden, indem Microgateways, zentrale Gateways (WAAPs) und IAM zusammenarbeiten. Bereits im Entwicklungs- und Testumfeld wirken die gleichen Schutzmechanismen wie später in der Produktion. Integrationsfehler und Sicherheitsverstösse fallen früh in der Entwicklung auf.
- Verantwortliche für die Kubernetes-Sicherheit müssen dabei auch stets die technologischen Entwicklungen im Auge behalten (Machine Learning, künstliche Intelligenz[14] und die Post-Quanten-Kryptografie[15]).

[14] https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Kuenstliche-Intelligenz/kuenstliche-intelligenz_node.html
[15] https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Quantentechnologien-und-Post-Quanten-Kryptografie/Post-Quanten-Kryptografie/post-quanten-kryptografie_node.html

Airlocks Microgateway bietet dafür besondere Vorteile:

- **Continuous Deployment:** Das DevOps-Team kann das Deployment automatisieren, sodass jede API und jeder Microservice mit einem Microgateway ausgerollt wird.
- **Security as Code:** Alle Microgateway-Konfigurationen (wie auch die OpenAPI-Spec) sind als „Security as Code" konzipiert. Sie können also wie der Anwendungs-Code versioniert im Repository abgelegt werden. Die Änderungshistorie ist damit beispielsweise jederzeit einsehbar.
- **OWASP Top 10:** WAF- und API-Schutzfunktionen halten Angreifer und böswillige Anfragen von der Applikation fern.
- **Zero Trust:** Das Microgateway sorgt für eine Netzwerkarchitektur nach dem Zero-Trust-Prinzip, indem es die Autorisierung jedes Zugriffs prüft (u. a. JWT, JWKS), bevor er die Anwendung erreicht. Dabei wird nicht nur North-South-, sondern auch East-West-Traffic gefiltert.

Die wichtigsten Sicherheitsempfehlungen für Kubernetes

Verifizierte Images: Verwenden Sie nur vertrauenswürdige Quellen oder offizielle Repositorys für Container-Images und scannen Sie diese auf Schwachstellen.

Ressourcenbeschränkungen: Legen Sie Memory- und CPU-Limits für jeden Container fest.

Least Privilege: Geben Sie Containern nur die notwendigen Berechtigungen und vermeiden Sie den privilegierten Modus (--privileged Flag), um unzureichende Container-Isolierung zu vermeiden.

Aktualität: Sorgen Sie für regelmässige Updates des Kubernetes-Daemons und der Images (gegebenenfalls über Watchtower).

Sicherheitsprofile: Implementieren Sie AppArmor oder SELinux Profile.

RBAC: Implementieren Sie Role-Based Access Control (RBAC), um Zugriffe einzuschränken.

Netzwerk-Policies: Definieren Sie, wie Pods miteinander kommunizieren dürfen.

Pod-Security-Policies: Legen Sie die Berechtigungen und Ressourcen fest, welche Pods zum Einsatz kommen dürfen.

Sicherheits-Policies: Filtern Sie die ein- und ausgehenden Anfragen und Antworten.

API-Zugriff: Begrenzen Sie den Zugriff auf die Kubernetes-API und verwenden Sie sichere Authentifizierungsmethoden.

Konfigurationssicherheit: Nutzen Sie Kubernetes Secrets oder spezialisierte Tools wie HashiCorp Vault, um Konfigurationen zu speichern.

Audit-Logging: Aktivieren Sie das Audit-Logging, um Aktivitäten im Kubernetes-Cluster zu überwachen.

Sicherheitsscans: Scannen und überwachen Sie Container-Images sowie Cluster-Konfigurationen auf Schwachstellen und Sicherheitslücken.

Versionierung: Verwendung Sie „GitOps"-Praktiken, um Infrastruktur-Code in Versionen zu verwalten und Änderungen nachvollziehbar zu machen.

AIRLOCK®
SECURE ACCESS HUB

Über uns:

Der Airlock Secure Access Hub vereint seit 20 Jahren die kritischen IT-Sicherheitsthemen der Filterung und Authentisierung zu einem gut abgestimmten Gesamtpaket, das Massstäbe in Sachen Bedienbarkeit und Services setzt. Der Secure Access Hub deckt alle wichtigen Funktionen der modernen Applikationssicherheit ab: von einer durch Fachjournalisten ausgezeichneten Web Application and API Protection (WAAP) über ein Microgateway für Sicherheit in Container-Umgebungen bis hin zu einem Identitäts- und Zugriffsmanagement (IAM) mit integrierter starker Authentifizierung. Die IT-Sicherheitslösung Airlock schützt mehr als 20 Millionen aktive digitale Identitäten und 30.000 Backends von über 600 Kunden auf der ganzen Welt.
Weitere Informationen unter https://www.airlock.com.

Airlock ist eine Security-Innovation des Schweizer Softwareunternehmens Ergon Informatik AG. Das Unternehmen wurde 1984 gegründet, zählt rund 450 Mitarbeiter und wurde wiederholt als einer der beliebtesten Arbeitgeber der Schweiz ausgezeichnet.

Kontaktadresse:
Ergon Informatik AG
Merkurstrasse 43
8032 Zürich

Telefon +41 268 87 00
info@airlock.com
www.airlock.com

Airlock® – Security Innovation by
Ergon Informatik AG

Das Whitepaper wurde in Kooperation mit unserem Partner Isovalent erstellt.

ISOVALENT
now part of cisco

Isovalent, der Entwickler von Cilium, hat sich dem Ausbau und der Pflege von Cilium in Zusammenarbeit mit der Cloud-Native-Community verschrieben. Isovalent vertreibt eine Enterprise-Variante von Cilium mit Support.