radware

Securing Cloud-Native Kubernetes Applications

WEB APPLICATION FIREWALL
TECHNICAL WHITE PAPER

SHARE THIS WHITE PAPER

TABLE OF CONTENTS

Overview

Application security has historically taken a back seat to application delivery. Traditional IT security teams view themselves as gatekeepers; they must do their jobs correctly or their organizations face increased risk. They incorporate high security standards into every operation, but achieving these standards takes time, testing and iterations. Development teams fret because this slows application development and often does not ensure comprehensive protection.

When businesses look to optimize and accelerate application development life cycles and deliver their applications in public clouds (typically referred to as cloud-native applications), security becomes a greater challenge. Cloud-native apps usually run in new architectures that provide unprecedented efficiency, flexibility and cost-effectiveness. Development and operations (DevOps) groups emerged to take an active role in application delivery by selecting the tools and budget to set the development and deployment pace. Automation platforms, powerful orchestration frameworks, open-source toolsets and visibility solutions all play a major role in these environments.

Development, security and operations (DevSecOps) represent the security component of DevOps and seek to fit security due diligence into processes that drive speed, agility and continuous delivery. However, DevSecOps may find it challenging to deliver security to continuous delivery processes if they lack automation, orchestration tools and visibility and have inferior application protection.

Containers, Microservices and Service Meshes

Transitioning from a monolithic architecture to a microservice architecture allows applications to be deployed more frequently. Different microservices can be independently delivered in a more reliable manner. In parallel, container technology emerged, which is a perfect match to microservices. Each microservice is deployed across multiple containers to allow for quick rollout and scalability, improving quality and reducing time to market.

A service mesh is a layer that handles the interservice communication in a microservice architecture. Its purpose is to reduce the complexity associated with a microservice architecture by providing scalability through load balancing, telemetry, traffic routing, health checks, etc.

One of the challenges with the transition to a microservice architecture has to do with the scale of objects provisioned. In the monolithic era, few monolith web instances were load balanced. Today, there are thousands of containers which need to be automatically spawned.

Kubernetes Orchestration

With the growth of the microservice architecture, a new space of containerized application orchestration frameworks has evolved, and Kubernetes is one of its largest components, given its adoption by most organizations.

Kubernetes is an open-source platform for managing containerized workloads and services and facilitates automating application deployment, scaling and management.

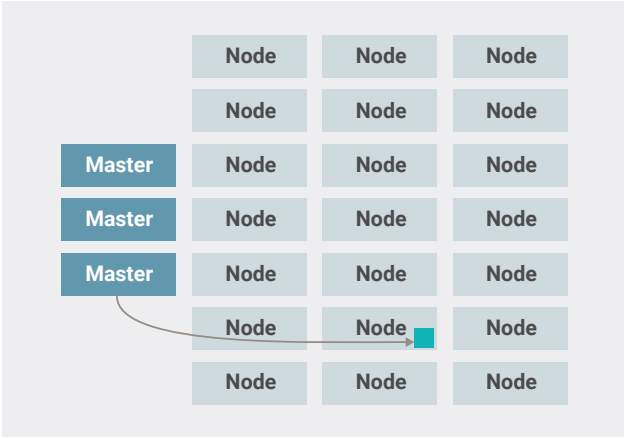
Kubernetes defines a set of objects and building blocks to orchestrate the containerized applications.

Container

- Programs running on Kubernetes are packaged as containers
- Limit of one process per container

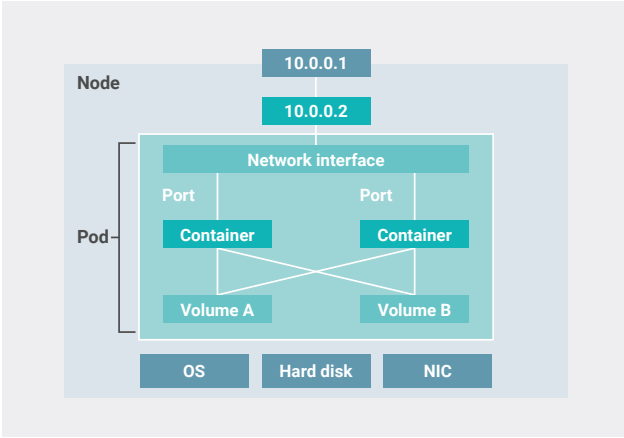
Cluster

- One master machine and multiple worker machines (nodes)
- A master coordinator between all nodes



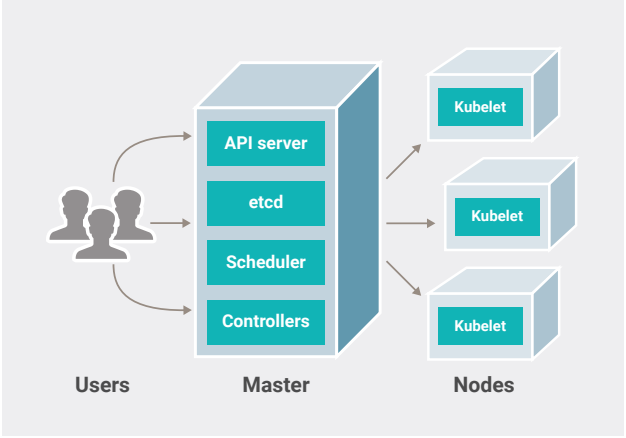
Pod

- An instance of the microservice
- The smallest unit of a cluster
- Represents a running process on a cluster
- Runs on a single container or multiple containers
- Different containers of a single pod communicate with each other using “localhost”
- Has an IP address



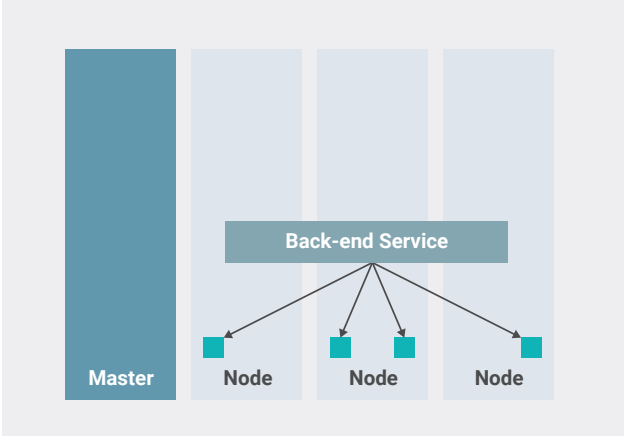
Node (Minion)

- A worker machine (cloud workload, virtual machine or physical machine) that contains services to run a pod
- Has a host name
- Has an IP address



Service

- Exposes an application running on a set of pods as a network service



Cloud-Native Application Security Challenges

With all the power that microservice architectures and service mesh infrastructures provide, they do not address application and data security challenges. In addition to the already known application security challenges, which are as relevant now as in the past, the distributed nature of microservice apps introduces latencies, topology changes and challenges with managing many microservices.

The Good Old OWASP Top 10 for Web Applications

The Open Web Application Security Project (OWASP) provides security professionals with an overview of the most critical web application security risks. Injections, broken authentication, cross-site scripting (XSS) and sensitive data exposure are just a few examples of risks that are as relevant for cloud-native apps as they are for monolithic applications.

The API Role in Modern Applications

Modern apps make intensive use of APIs across different use cases: internet of things (IoT) applications, machine-to-machine communication, event-driven web applications, automated actions in web frameworks, function-as-a-service (FaaS) apps, mobile apps, etc. All of these use cases refer to north-south communication, which is client-to-app traffic. Most of the APIs serving these use cases are REST APIs with JSON bodies (REST-JSON). A smaller portion of the APIs are simple object access protocol (SOAP) based with XML structured data formats.

As these APIs are running on top of the HTTP protocol, most of the web app security risks are as relevant for APIs as they are for web apps. Additionally, APIs introduce additional security challenges, mostly around authorization and access control, as the APIs may be served independently.

The New OWASP Top 10 for APIs

With the growing adoption of APIs, the OWASP organization has produced the API Security Project under which it has published the OWASP API Security Top 10 Release Candidate. Examples of the API security risks referred to in the project include broken object level authorization, excessive data exposure, lack of resources and rate limiting, and injection, all of which may lead to data theft or service disruption.

DevSecOps Requirements

By their nature, cloud-native apps are running in cloud environments, leveraging and consuming the different services which the public cloud vendors offer, such as workloads, storage, Kubernetes orchestration services, content delivery networks (CDNs), etc. These services provide simple delivery while using industry standard open-source projects or public cloud technology.

However, the quality of the security services offered by the public cloud vendors is inadequate for different reasons. The complexity of the security operation, the continuous research investment required and the constantly changing threat landscape are three of the primary reasons.

DevSecOps seek to address this gap in quality of security offering while looking for a security solution that will fit into their ecosystem. The solution must be behind the cloud vendor CDN service, be scalable, offer complete support for automation to integrate into the Kubernetes orchestration system, provide detailed security visibility and reporting, keep the footprint low, maintain low false-positive posture, and address the most critical security risks associated with APIs and mobile apps.

Radware's Kubernetes Web Application Firewall (WAF)

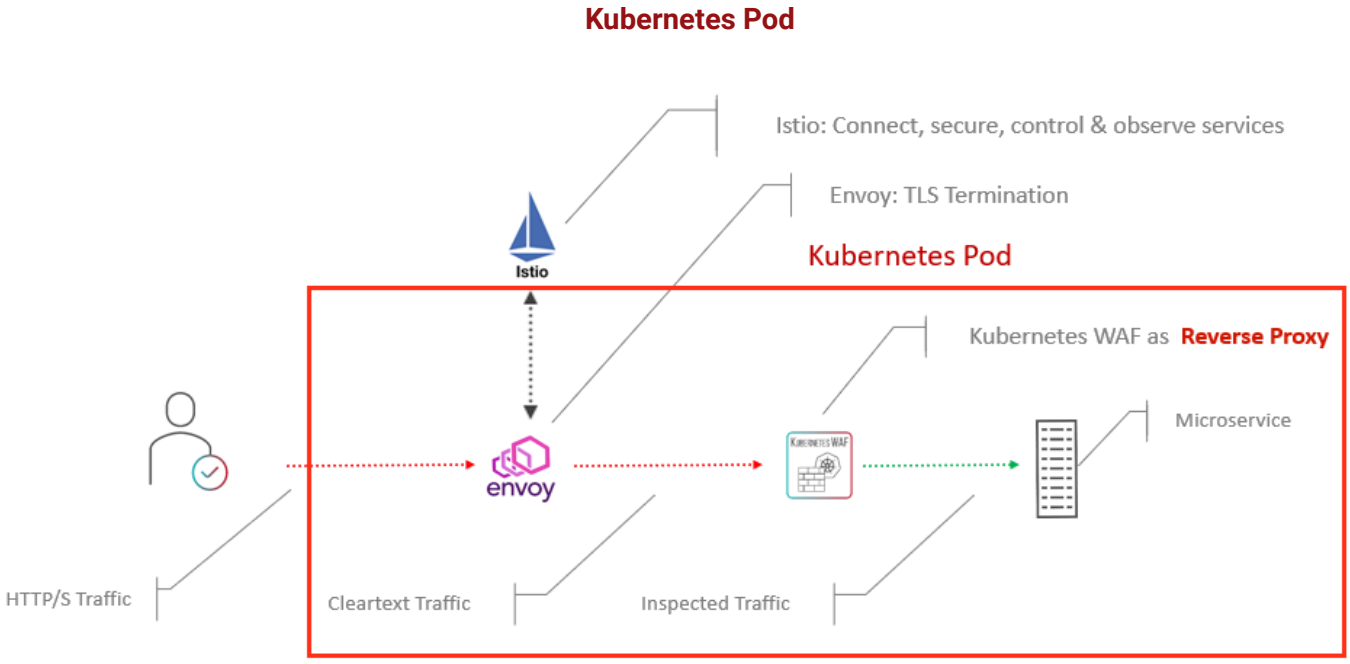
Radware's Kubernetes WAF is designed to fit into the Kubernetes orchestration system in a service mesh architecture to provide the best of both worlds: market-leading application security with advanced automation and the elasticity required by today's DevSecOps teams. It offers its own management portal, integrating common visibility and reporting platforms such as Elastic and Prometheus. The security offering includes web and API security for applications and microservices running in Kubernetes.

The Architecture — Designed for Kubernetes

Radware's Kubernetes WAF consists of a distributed architecture with security sidecars deployed at the pod level in the data plan and a management back end running on the same Kubernetes cluster for the control plane. It is based on four primary components.

Data Plane Policy Enforcer

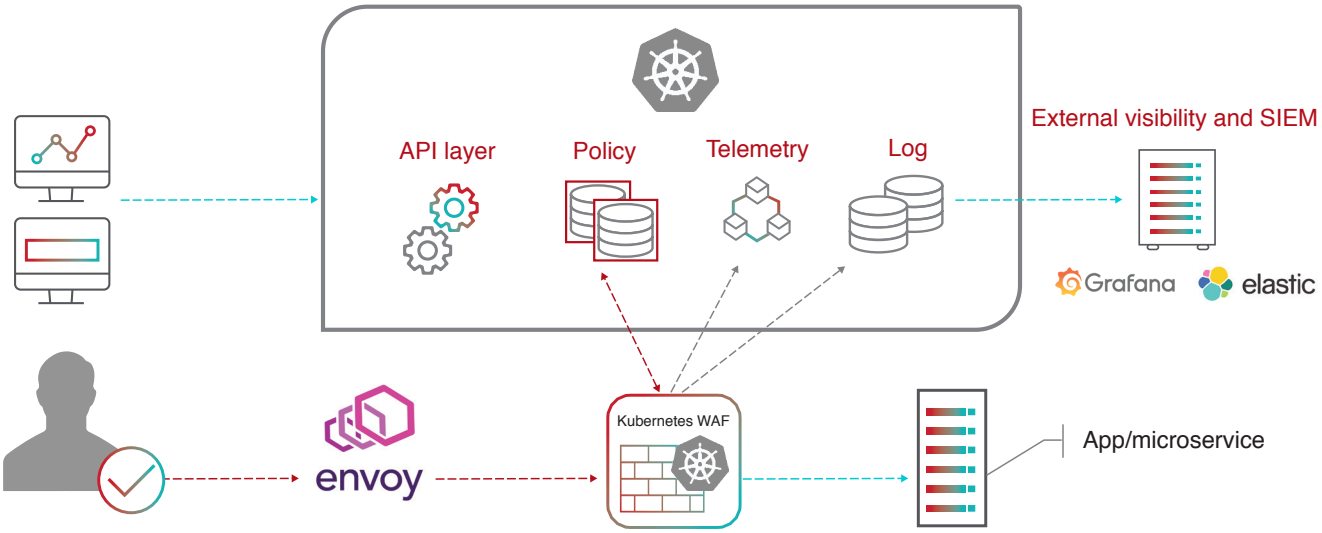
The Kubernetes WAF data plane policy enforcer runs as a security sidecar in the same pod as the microservice. It functions as a reverse proxy before the microservice and can work after either a sidecar proxy (e.g., Istio/Envoy) or any other ingress method outside of the pod (e.g., NGINX ingress controller). In both cases, SSL termination is accomplished before the enforcement, by either Istio or the Kubernetes cluster's ingress controller, to allow the enforcer to handle clear text traffic. In any case, all reverse proxies are supported. The solution enables single termination of TLS traffic only at the host level and thus eliminates the need to manage multiple certificates across different parties.



Controller

The Kubernetes WAF centralized control plane back end is running in the same Kubernetes cluster as the application. It provides centralized administration, management, reporting and forensics, via either APIs or the management portal GUI. Kubernetes will provision the data plane enforcer to allow seamless scalability and elasticity. Since the policy is centrally stored and managed in the controller, any policy change that is applied manually by the administrator or automatically generated by the machine learning modules will be synchronized automatically across all data path policy enforcers. Telemetry information is collected from the policy enforcer and pushed to the controller for analytics and auto-policy processes.

Logs are sent from the enforcer to a centralized logging module in the controller to allow centralized forensics and analytics and allow forwarding of the logs to external visibility and security information and event management (SIEM) systems such as Elastic and Prometheus.

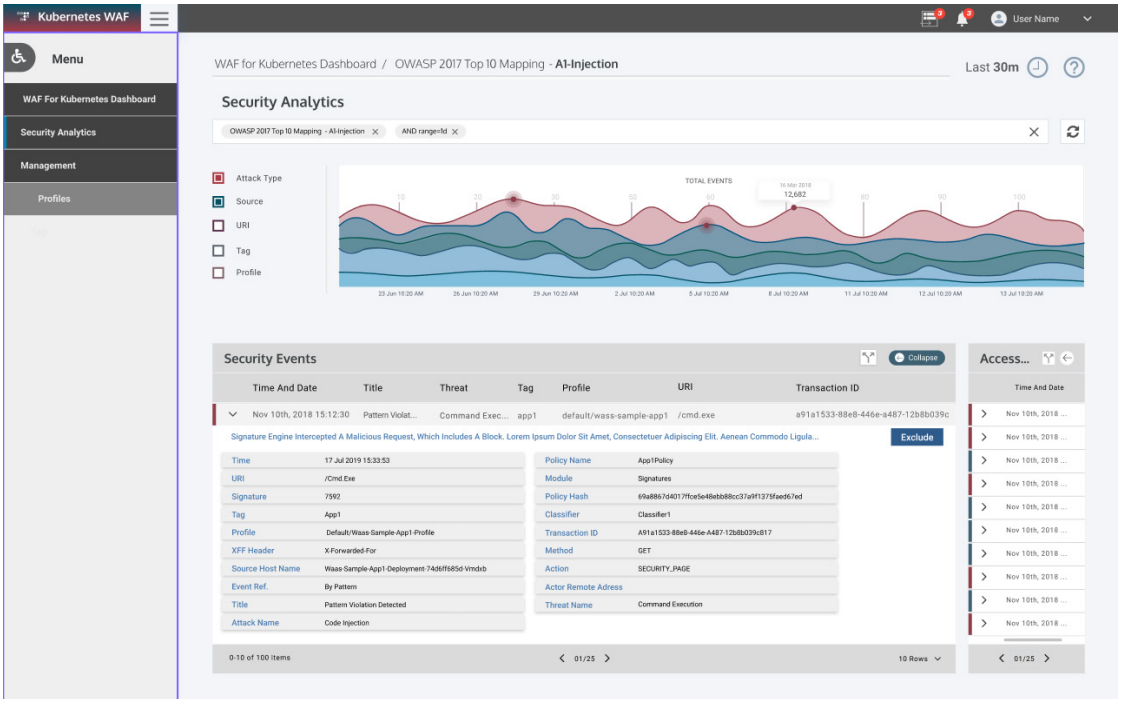


An essential part of the control plane is the auto-policy generation engine. Machine learning algorithms map the application resources, analyze potential threats and then apply and tune the security policy. This process runs automatically whenever a new microservice is introduced or a modification is being pushed.

Management Portal

The Kubernetes WAF management portal provides a GUI to monitor and manage Kubernetes WAF security. It consists of:

- **Dashboards** which provide visualization of real-time telemetry and security events tailored for the DevOps and SecOps users.
- **Forensics** which provide detailed security event reports with drill-down analysis, analytics and exception handling. Each security event can be excluded from future inspection if deemed legitimate traffic. Additionally, full transaction logs are visible under the access logs.
- **Security profiles** which provide security configuration settings for the system. Users can define flexible classifiers based on client traffic attributes (HTTP header values), and the compatible policy will be applied.



Management APIs

Every configuration and operation on the Kubernetes WAF can be performed without the GUI via Kubernetes compatible APIs.

Security profiles are defined as Kubernetes custom resource definitions (CRDs), and they are managed in the same way as other Kubernetes resources. This approach allows for full management and configuration of security profiles via native Kubernetes APIs without using the management portal.

Telemetry and log collection are accomplished with Prometheus to allow external visibility with commonly integrated tools like Grafana, including traffic metrics, statistics, Kubernetes WAF resource usage and security event data.

Security logs can be pushed to an external Elastic platform for detailed visibility and reporting using Kibana dashboards.

SECURITY — ROBUST APPLICATION AND DATA PROTECTION

Active Protection and Report-Only Mode

Radware's Kubernetes WAF provides protection modules originated from Radware's WAF, AppWall®, but tailored to deliver WAF protection to Kubernetes-aligned security solutions. Each module can be set to a report-only mode, which generates security logs for violations but will not block traffic or activate a protection mode.

Access Control

The access control module allows the defining of web resources and APIs that should be accessible. Requests to nonlisted resources will be blocked or logged, depending on the policy settings. Policy settings can be defined at a full path level or file extension level or by a regular expression definition.

Expressions

The expressions security module is based on regular expression and logical rules for detection of known types of attacks, such as SQL and non-SQL injections.

Signatures

The signatures security module is based on a string match engine to detect attack patterns in HTTP requests. Signature rules detect known types of attacks such as XSS, predictable resource locations, directory traversal, etc.

JSON and XML Parsing

Both JSON and XML bodies are parsed, JSON/XML validity checks are applied, and key values are extracted for further inspection by the other protection modules, such as signatures and expressions. This allows detection of common API attacks such as XML bombs, manipulation of APIs and detection of embedded attacks.

Data Leakage Protection

The data leakage protection module identifies sensitive information in application responses, allowing the masking of sensitive data. Examples of sensitive data are credit card numbers, Social Security numbers, server error messages, etc.

Source Blocking

This module provides per-source correlation of security violations for the blocking of persistent attack agents.

Conclusion

With its advanced automation, policy generation and robust security, Radware's Kubernetes WAF provides DevSecOps with the flexibility and automation needed to allow for the timely development and deployment of secure applications across hybrid computing environments.

About Radware

Radware® (NASDAQ: RDWR) is a global leader of cybersecurity and application delivery solutions for physical, cloud and software-defined data centers. Its award-winning solutions portfolio secures the digital experience by providing infrastructure, application and corporate IT protection and availability services to enterprises globally. Radware's solutions empower more than 12,500 enterprise and carrier customers worldwide to adapt quickly to market challenges, maintain business continuity and achieve maximum productivity while keeping costs down. For more information, please visit www.radware.com.

Radware encourages you to join our community and follow us on: Radware Blog, LinkedIn, Facebook, Twitter, YouTube, Radware Connect app for iPhone® and our security center DDoSWarriors.com that provides a comprehensive analysis of DDoS attack tools, trends and threats.