Discovery of Cloud Native applications from an application PLICATION security perspective

his article is based on my understanding of what is a Cloud Native application. This knowledge is mainly based on the reading of the following sources:

Context of the blog post

• Book: "Cloud Native: Using Containers, Functions, and Data to Build Next-Generation Applications". • Book: "Understanding Kubernetes in a visual way: Learn and discover Kubernetes in sketchnotes"

• Regular research on the topic from Abhay Bhargav and folks from his company WE45.

My goal was to try to identify which aspects of the security of an application change when an application is intended to be Cloud Native. From here, a Cloud Native Application will be called a CNA.

Disclaimer

wrong. If this is the case, then, feel free to reach out to me to discuss together allowing me to enhance my

understanding, fix my mistake and teach the right information.

🐜 As always for me, it is possible that some of my understanding/idea/hypothesis/insights were partially or totally

What is considered a Cloud Native application? An application, intended to be a CNA, will have the following characteristics:

• Leverage the maximum of services provided by a Cloud provider to focus on the added value aspect to the

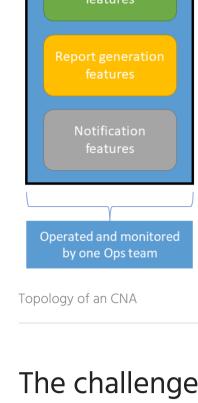
• It is composed of several dedicated parts, each one leveraging the more effective Cloud feature according to its

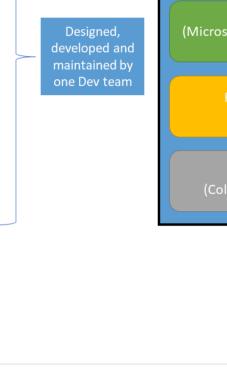
example, short network disruption with built-in "retry" mechanisms.

- business/technical objective: A part can be a serverless Function or a collection of Microservices hosted in a container managed by a container orchestrator. • It is a distributed system by design. • It is stateless to allow easy horizontal scaling in and out. • Its design and implementation anticipate the kind of issues that can occur in Cloud-based environment, like for
- The schema below shows the difference, in terms of topology, between a traditional application and a CNA: Traditional app Cloud native app
- features

ull managed by a User management User management features DevOps team (Delegated to Cloud identity provider) dedicated to this Data import features

Data import Designed, (Microservices in a container deployed in developed and a Container as a Service) one Dev team







DevOps team

security in mind according to its business/technical context. This includes the following aspects:

Authentication.

• Protection of data in transit and at rest. • Secure credentials/secret handling and storage. Security logging and monitoring. • Monitoring of third-party components used from a security perspective to prevent vulnerable ones.

controls, libraries, framework to ensure that the security in the aspect mentioned above are handled in the same rigorous way.

Service A Version 1.0

Service C

Version 1.0

Service D Version 1.3

Note about the importance of the cartography in a CNA

Cloud native app Data import features Consume the (Microservices in a container deployed

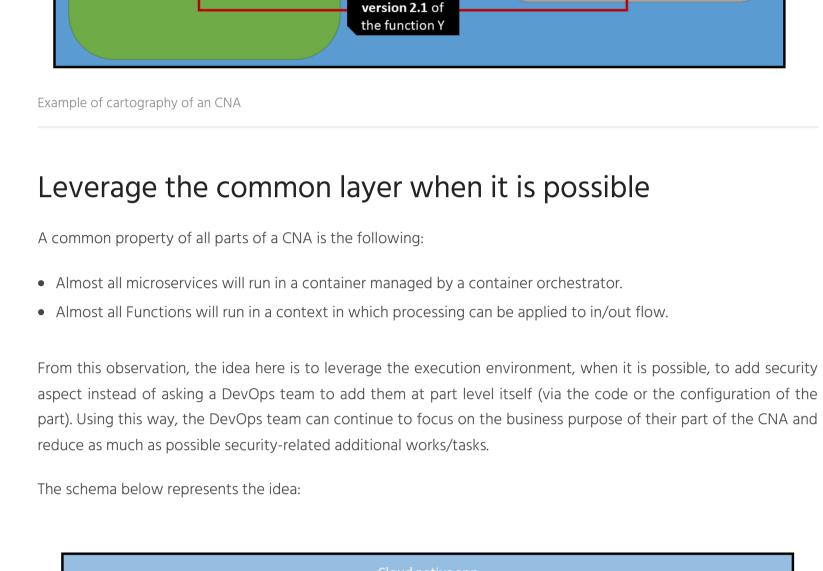
Consume the version 1.7 of Service B the function Version 1.1 Consume the

version 2.0 of

the function X

Consume the

version 1.0 of the function



Data import features (Microservices in a container deployed

Security processing layer

Usage of the common layer

• 🏭 Container orchestrator level.

Application level (code or CI/CD pipeline).

Input data validation and output data encoding.

Protection of data in transit and at rest.

Secure credentials/secret handling and storage.

Error handling and technical information disclosure prevention.

Service mesh level.

leveraged to use a maximum of built-in security feature offered by modern technology stack: Legend:

Modern container service mesh software (like Istio) and modern web application framework (like Spring Boot, ASP.Net Core, etc.) have built-in security features. Therefore, the proposed following distribution of the role can be

External systems

Security processing layer

data level

For app part

For data at rest

For data in transit

For service mesh related part

Handled at which level? Area Authentication. lly I When authorization is at service level only Authorization.

Security logging and monitoring. Monitoring of third-party components used from a security ■ In CI pipeline perspective to prevent using vulnerable one. Example of leveraging the service mesh security features To validate the elements proposed from a technical/pragmatic perspective, I put together the following proof of concept (POC): App1 container App2 container Sidecar container Sidecar container (Envoy proxy via Istio) (Envoy proxy via Istio) External systems

 Authentication via a token JWT. • Authorization via the claims of the token JWT. in This YAML file was describing the Kubernetes deployment plan for both applications.

App 1 ### Deployment

selector:

template: metadata: labels:

spec:

apiVersion: v1

References used:

Authentication rules

namespace: my-poc

apiVersion: security.istio.io/v1 kind: RequestAuthentication

name: "my-app1-authenticationrule"

app: my-app1

app: my-app1

- name: my-app1

containers:

Objective of the POC

] Create Deployment & Service for both app.. ployment.apps/my-appl-deployment created

rvice/my-appl-service created eployment.apps/my-app2-deployment created ervice/my-app2-service created ress.networking.k8s.io/my-app-ingress created Let all pods be initialized...

ngress IP is 192.168.49.2 >>> http://192.168.49.2/app1

gress IP is 192.168.49.2 >> http://192.168.49.2/app1 >>> http://192.168.49.2/app2 TTP/1.1 403 Forbidden
poc ./poc-deploy.sh

Provisioning of the POC

2. Execution traces.

Steps in *red circles* in the image above:

1. Command line used to run the script.

] Status - Pod/Deployments/Service ..

-app1-deployment-688c947759-mjgkw 2/2 -app2-deployment-69dfc4bd8c-tl5tx 2/2

IThis script was used to deploy the POC components:

TTP/1.1 200 OK

>>> http://192.168.49.2/app2

TTP/1.1 200 OK

+] Add authentication/authorization rules...
equestauthentication.security.istio.io/my-app1-authenticationrule created

equestauthentication.security.istio.io/my-app2-authenticationnule created uthorizationpolicy.security.istio.io/my-app2-authorizationrule created uthorizationpolicy.security.istio.io/my-app2-authorizationrule created +] Let all policies be applied...
+] ReTest access to both app...

Services
AME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR
y-appl-service NodePort 10.102.71.117 <none> 8080:31571/TCP 21s app=my-app1
y-app2-service NodePort 10.103.19.248 <none> 8080:30674/TCP 21s app=my-app2
+] Get Ingress IP address & test access to both app before adding authentication/authorization rules.

 READY
 STATUS
 RESTARTS
 AGE
 IP
 NODE
 NOMINAT

 2/2
 Running
 0
 17s
 10.244.0.206
 minikube
 <none>

 2/2
 Running
 0
 17s
 10.244.0.205
 minikube
 <none>

NOMINATED NODE READINESS GATE

kicbase/echo-server:latest app=my-app1 kicbase/echo-server:latest app=my-app2

The objective of the POC was to achieve the following security aspect, only using the Kubernetes or Istio

apiVersion: apps/v1 kind: Deployment metadata: name: my-app1-deployment namespace: my-poc

image: "kicbase/echo-server:latest"

https://kubernetes.io/docs/concepts/services-networking/service/

Learn how to to create a an access rule with Istio

- containerPort: 8080

security features, nothing implemented at application level:

kind: Service POC definition file This YAML file was describing the Istio deployment plan for the following authentication and authorization rules:

https://istio.io/latest/docs/tasks/security/authorization/authz-jwt/

https://istio.io/latest/docs/reference/config/security/jwt/

https://istio.io/latest/docs/reference/config/security/request_authentication/

Clarification between AuthorizationPolicy and RequestAuthentication rules: https://stackoverflow.com/a/62417272

selector: matchLabels: app: my-app1 jwtRules: - issuer: "excellium-ias" jwksUri: "https://raw.githubusercontent.com/ExcelliumSA/CloudNativeApplications-Study/main/poc/rsa-2048-public.jwks.json apiVersion: security.istio.io/v1 kind: RequestAuthentication name: "my-app2-authenticationrule" namespace: my-poc selector: matchLabels: app: my-app2 jwtRules: - issuer: "excellium-ias" Security provisioning • Authentication (*RequestAuthentication entries*): • For both app, the JWT token provided must: • Been issued by "excellium-ias" issuer. • Been signed with the RSA private key associated to this public key. • Authorization (*AuthorizationPolicy entries*): For App1: • A valid JWT token must be provided. • The audience claim of the token must be intended for app1. For App2: • A valid JWT token must be provided. • The audience claim of the token must be intended for app2. The custom claim named ispartner must be set to Yes.

This script was testing that the rules were effective:

P 200 ⇒ Request served by my-app1-deployment-688c947759-mjgkw

TP 403 → RBAC: access denied | Call appl with a JWT token having the wrong issuer but the correct audience and signer

TP 403 => RBAC: access denied -] Call appl with a JWT token having the correct issuer, correct signer and correct audience

TP 401 ⇒ Jwt issuer is not configured] Call app1 with a JWT token having the correct issuer and correct signer but a wrong audience (audience set to app2)

TP 403 ⇒ RBAC: access denied] Call app2 with a JWT token having the correct issuer, correct signer, correct audience but without the ispartner claim

.] Call app2 with a JWT token having the correct issuer, correct signer, correct audience and correct ispartner claim value TP 200 ⇒ Request served by my-app2-deployment-69dfc4bd8c-tl5tx

of them having its proper lifecycle, technology stack, team, and security maturity.

Cloud Native applications? Contact our experts!

TP 403 ⇒ RBAC: access denied

| Call app2 with a JWT token having the correct visuer, correct signer, correct audience but with the ispartner claim having a correct value using a wrong cas

Cloud Native Applications change the core structure of what is an "application" by exploding it in several parts. Each

Traditional ways of including security in an application can still be applied. However, it will imply significantly more

[+] Load signing key from JWKS Loaded.

] Call app1 without a JWT token

Test of correct application of the policies

Conclusion

common layer to make the security level consistent and transparent for Dev and Ops teams. All materials of the POC are stored in this GitHub public repository and the POC can be fully reproduced. Author

Name ³ First

Email *

Dominique Righetto

Company * **Phone Number ***

Comment or Message * Submit Application Security, Appsec, Cloud, Cloud Native, Cna, Cybersecurity, Excellium Services, Excellium Services Belgium, Excellium Services Luxembourg, IT Security, Native

≺ Share

Exercises The Art of Password Spraying: A

The Necessity of Cyber Crisis

Recent Posts

Common client-side vulnerabilities of web applications

Comprehensive Analysis

Consulting Cyber Blog Post General Newsletter The Cyber Blog Times Uncategorized

Categories

Advice

Search Search ...

Q

effort/time/cost resources. Therefore, it is important to leverage the new security features provided by the Do you have any questions? Would you like to know more about

CONTACT US

Application Security Network & Security Infrastructure CERT-XLM

Let's deliver the right solution for your business Meet Success Services About Us Eyeguard Career Opportunities Where to meet us Intrusion Tests - Red Team Privacy Notice Contact Us

Eyetools Training

Mentions légales/Privacy Notice

Information Security Governance

© 2021 Excellium Services Group 5 rue Goell L-5326 Contern Luxembourg +352 26 20 39 64

Follow us

f y in

The challenge from an application security perspective? The main characteristic of a CNA is that it is composed of several independent parts. Therefore, to have a globally secure CNA, all parts of the CNA require the same security level. Same point regarding the business/technical event logging across all parts. Indeed, from a security perspective, it is important to be able to track a communication flow through all parts of the CNA. To achieve this goal, it is required that all parts log information in the same way (format, content, etc.), with the same granularity, and have unique correlation identifiers to identify an event. Here comes a big challenge! When we talk about a secure application (CNA or traditional one), we refer to the fact that it was designed with Authoirzation. Input data validation and output data encoding. • Error handling and technical information disclosure prevention. Source: OWASP Top 10 Proactive Controls Why is it a challenge? As the CNA parts will be developed and operated by different teams, we need to define standards, security This, even if all parts are using different web technologies (or even Cloud providers) and the different teams do not have the same maturity and knowledge in terms of application security as well as the development velocity/timeline. In theory, its sound possible but unfortunately, we (like you dear reader) know what the reality in software development industry is, especially when it comes to its security aspect 😉 Therefore, let's be honest, pragmatic, and humble. As the focus of this article is modern applications topology using modern technologies, let's see how such "modern" material can help us in our challenge. As CNA were "born" to allow company to create and deliver software quicker to follow their clients' needs (and before their competitor). Let's use this as foundation (statement learned from the book). Nowadays, most of the development team already uses Agile methodologies combined with DevOps approach/mindset to create and deliver effective software in a high veloce way. Therefore, the objective is to avoid slowing them down when we add "security" related elements. Because CNA is composed of several parts, each collection of parts will evolve in a different timeline or velocity. The direct implication is the need to manage different versions of a part to ensure a functional whole CNA. Thus, developing a CNA will imply that a well-planned management and cartography of the dependencies between all parts in place. The schema below represents the idea: