Tags:

Thales Cyber Solutions Belgium

application security

Luxembourg

05 September 2024

Discovery of Kubernetes Native applications from an application security perspective

Disclaimer

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

🤝 If it 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.

Resources used

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

- The book "Understanding Kubernetes in a visual way: Learn and discover Kubernetes in sketchnotes"
- The following training from the company WE4IT: Containers & Kubernetes Security
- Hand-on practice, against a local lab, based on Minikube.
- Several posts and resources about the "distroless" concepts:
  - Getting Started with Distroless Images
  - Is Your Container Image Really Distroless?
  - Why distroless containers aren't the security solution you think they are?
  - Language focused docker images, minus the operating system.

Context of the blog post

💡 This post is the continuation of my study regarding Cloud Native applications. The content of this study is summarized into this post: "Discovery of Cloud Native applications from an application security perspective".

🎯 This time, I focused my work on the discovery of the type of application named **Kubernetes Native** application, in order to understand the impact of such topology from an application security perspective (defensive and offensive point of views).

📍 From here a:

- Kubernetes Native application will be called a KNA.
- Cloud Native application will be called a CNA.

🤓 I assume that the reader has a technical foundation about Kubernetes.

What is considered a KNA?

🤔 A KNA is an application that is designed to be deployed and operated on a Kubernetes cluster. To be more precise, it shares **the same properties** than a CNA with the difference that, instead of *"leveraging the maximum of services provided by a Cloud provider"*, it leverages the *"maximum of features provided by Kubernetes"*.

📍 This post give insight about why companies are focusing on KNA.

KNA from an application security perspective: Defensive point of view

📄 The challenges faced by a KNA are the same than for a CNA, refer to the related post for details. Therefore, the idea to leverage the *common layer* (Kubernetes here) shared by all modules composing the KNA, is still applicable.

🕵️ To validate this statement from a technical perspective, I decided to identify which Kubernetes features (including third party components) can be used to handle a security area of an KNA. For the list of security area, the points from the "OWASP Proactive Controls" were used to be consistent with the work performed for the CNA.

📖 The table below provides a view of the results:

Area	Features
Authentication	Istio authentication features can be used.
Authorization	• Istio authorization features can be used for the authorization that are enforced at the endpoint level. • For the part of authorization that are enforced at data level then it will be handled into the app code business logic.
Input data validation and output data encoding.	• Strict input validation must be performed into the app code business logic because the app knows what a "valid" data is. Modern web framework provided feature to perform content validation. • Same regarding the output escaping. This is automatically handled by modern web framework. • It is possible to add a Web Application Firewall in front of the Kubernetes cluster in case of need. Refer to here for insight.

Error handling and technical information disclosure prevention.	• Istio Virtual Service features can be used to add and remove HTTP response headers (like Security headers ones).
	• Regarding the rewrite of the content of the response, two pending issues, on Istio and Kubernetes side, were found.
	  ◦ https://github.com/istio/istio/issues/15540
	  ◦ https://github.com/kubernetes-sigs/gateway-api/issues/1998
	🚩 Note that usage of an envoy filter, like this one, must be used with care because any incorrect configuration can destabilize the entire mesh, see here for details.
	💡 However, modern framework support returning a generic error, so this point is easy to handle at the app code level as well as into any reverse proxy/WAF in front of the Kubernetes cluster.

Protection of data in transit and at rest.	• Istio Mutual TLS features can be used. • Hashicorp Vault integration with Kubernetes and app for data ciphering.
Secure credentials/secret handling and storage. Security logging and monitoring.	• Hashicorp Vault integration with Kubernetes and app for secret storage and access. • Kiali features can be used. • Falco features can be used.
Monitoring of third party components used. Application security perspective to prevent using vulnerable ones.	• Generate SBOM (Software Bill of Materials) descriptor with Syft during the image builds and ingest it into a tool, like " OWASP Dependency Track ", to cartography the external dependencies of the image. • Regularly scan image with Grype.

✅ So, several security areas can be hardened using Kubernetes features, allowing to define a "minimum security posture" whatever the security posture of the application code/configuration itself.

📦 Regarding the hardening of the context, in which the KNA is deployed, the following features can be leveraged:

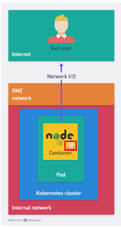
- Kubernetes Network Policies to define network access rules between components.
- Kyverno to enforce security constraints and prevent the deployment of components using insecure configuration or permissions.
- KubiScan to identify risky permissions at the cluster level.

KNA from an application security perspective: Offensive point of view

🤔 For this perspective, I focused on the following scenario:

"Which attack vectors are available if a module of an KNA is compromised, in the way of a Remote Code Execution vulnerability?"

📍 This scenario is often called a *"container breakout"*, below is a simplified overview of such scenario:



📖 This post provide a good explanation about this scenario.

👀 Based on such scenario and the different resources studied, I identified the following attack vectors:

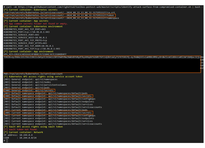
📍 Naturally, the capability of the attacker to leverage the different attack vectors above depend on the content of the image used by the comprised container as well as its runtime configuration. Indeed, the more system tools/complex libraries it contains, the more attack vectors can be leveraged. On the contrary, if the image, for example, does not contain any tools to perform network I/O, then the attacker will need to import them. If, in addition, the container is into readOnly mode, then it will be more difficult to import such tools. Refer to the distroless resources mentioned for technical insights about to use image with only required content.

💡 It is interesting to note that, some HTTP response headers, can disclose that the app is deployed on a Kubernetes cluster. Indeed, header starting with **X-Envoy** for example, indicate the presence of Istio. Same thing regarding header starting with **X-Kubernetes** for the presence of a Kubernetes cluster. Refer to this gist for a list of such headers.

🔎 Example of search on Shodan for such headers:

🤓 Like for the defensive point of view, to validate the above statement from a technical perspective, I decided to create a POC using my Kubernetes lab to see available attack vectors. This POC led to the creation of this script.

👀 Example of execution of the script in "a compromised container" deployed into my Kubernetes cluster where the app was deployed on the default namespace:



✅ So, such script and approach can be leveraged to gather information about the *"current landing context"* and then to guide, for example, the following operations:

- The tentative of escalation of privilege to the entire Kubernetes cluster.
- The compromising of other business service exposed by the Kubernetes cluster.
- And so on.

Conclusion

Kubernetes, and its ecosystem, provide a collection of useful features to apply a "default security posture" to any application deployed, so it represents an interesting platform to deploy and operate applications. It requires to be correctly configured and hardened to prevent than a security issue, in a container, affect the whole cluster, but it is the same for every platform on which applications are deployed. However, many tools and guidelines exist to achieve a good Kubernetes setup.

Author

Dominique Nyheitz

Contact us | Contactez-nous