Wiretapping Pods and Nodes - Lawful Interception in Kubernetes

Daniel Spiekermann, Jörg Keller

[1] daniel.spiekermann@polizei.niedersachsen.de
Polizeiakademie Niedersachsen
[2] joerg.keller@fernuni-hagen.de
FernUniversität in Hagen

Abstract: Nowadays IT infrastructures have to supply a flexible and dynamic platform for the provision of modern applications. Kubernetes is one of the most notable environments for the provisioning of small and independently running microservices used by modern applications. With Kubernetes, these microservices can be developed, deployed, updated and scaled in a continuous process. This flexibility is a huge advantage to older and more static environments. But whereas these old infrastructures lack in dynamics, necessary digital investigation are easier to accomplish. This need is still existing in modern environments, hence this paper presents a novel approach for the lawful interception of network packets in a Kubernetes cluster. The approach improves the static capture processes by monitoring involved devices assigned to a defined application without hampering the environment or capturing unwanted network packets.

Keywords: Kubernetes, network forensics, lawful interception, virtual network

1 Introduction

The provision of applications is a crucial task in modern infrastructures. In the past, computer programs were mostly monolithic, using a single process. The use of multi-threading in combination with virtualisation helps to spread modern application across a flexible number of services. This improves the resource usage, runtime behaviour and availability of the processes, but it impedes the update and upgrade process, which is a critical aspect in the life-cycle of a software application.

To support and improve this process, agile methods like CI/CD (continuous integration/continuous delivery) are used [Luk18]. The need for supporting technologies led to the evolution of orchestration tools like Kubernetes. Kubernetes is an open-source implementation for flexible and on-demand provisioning of applications. Its main benefits are the improved scaling and management of a huge number of applications running simultaneously in the environment. Kubernetes relies heavily on virtualisation techniques like containers and virtual networks. By decoupling the applications from the underlying hardware, Kubernetes is able to provide tens of thousands of applications at the same time, each isolated in a special environment, but connected to other systems if needed. Administrators are able to define different configurations, which manage the life-cycle of an application, e. g. the number of running instances, network parameters and connections between involved systems.

Forensic investigation in these highly virtualized environments is a complex task, faced with a number of different challenges [SE16]. Whereas the underlying systems in a Kubernetes cluster use either bare-metal-platforms or virtual machines, the investigation of these systems is well researched [RUS20]. In contrast to this, advanced techniques like network forensics are hampered because of the inherent dynamic. Containers are started and stopped on different systems, which results in an ongoing change in the involved network structure. The need for such an investigations is crucial, because the number of adversarial attacks increases and advanced security mechanisms are increasing [IHG16]. Network forensic investigation helps to examine these attacks by capturing and investigating relevant network traffic [JP16]. We present a novel approach for the lawful interception of network packets in a Kubernetes cluster, improving over a static capture process, but without hampering the network environment or capturing unwanted packets.

2 Network forensic investigation in Kubernetes environments

To analyse network traffic transferred in Kubernetes cluster demands a previous packet capture process, which manages the collection of all relevant network packets. The correct position of packet capturing is the most critical part. If the position is wrong, not all relevant network packets traverse this position. E. g., a capture process running on a uplink port results in a huge number of irrelevant packets in combination with a number of missing relevant packets, which are not transmitted via this uplink.

Kubernetes provides a huge number of involved components to fulfil the necessary requirements of a flexible and agile basement. The most relevant components for network forensic investigations are:

- **Pod**
  A pod is a container or a group of containers. It is the basic unit in the Kubernetes environment and provides a secure environment with its own network information like a unique IP-address by using a virtual network interface (vNIC).

- **Node**
  A node in a Kubernetes environment is a host, where pods are deployed. A node may be a physical or a virtual system and provides all services to run the the pods. A node uses an internal IP-address to communicate with systems inside the cluster. If the node has to communicate with external systems, a dedicated external IP-address is assigned.

The design of Kubernetes provides different positions of packet intercepting. Figure 1 shows the different positions in detail.

- **NIC of node (1)**
  At this position a process is able to capture all network packets traversing this position, e. g. all incoming and outgoing network packets of the containers, which communicate with external systems. This might result in capture files containing some relevant network traffic of the suspicious application, but on the other hand the number of irrelevant

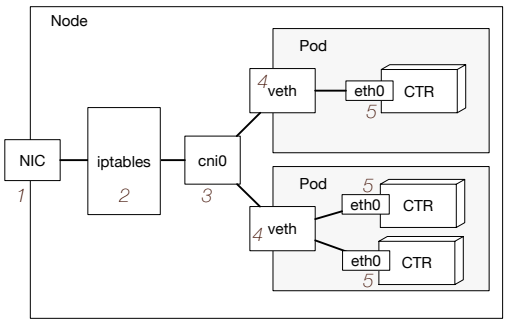

Figure 1: Positions of packet capture

network packets, especially of pods providing another application on the same node or cluster management traffic, will increase.

- **iptables (2)**
  Kubernetes uses additional firewall rules like KUBE-FIREWALL and KUBE-FORWARD based on *netfilter/iptables* to manage the network packets. On the one hand, these rules isolate containers and, on the other hand, expose the provided services to the network. Using *iptables* to intercept the packets is possible, but it requires additional extensions like TEE.

- **cni0 (3)**
  The internal use of the container network interface (cni0) is the default connection to the pods running on a node. This interface implements an inter-pod communication, thus a capture process would capture the needed network packets of the involved pod, but will additionally capture all traffic of the other pods running on the node.

- **vNIC of pod (4)**
  Each pod uses a virtual network interface card (vNIC) to establish the connection to cni. At this position, all relevant network packets are available.

- **vNIC of container (5)**
  The last position of accessing the network packets is the vNIC of the involved container. At this point all packets regarding the involved container are accessible. Whereas a capture process would be valid in common container-based environments, the restriction might miss some relevant packets, i. e. when the sidecar pattern is used [Luk18].

To capture all relevant network packets, a capture process has to be implemented at each pod providing the suspicious application. To reduce the number of irrelevant packets, this process should capture at the vNIC of the involved pods. This position is similar to the Distance-0-Capture defined in [SKE18].

3 Discussion

To validate the results, a PoC based on Python 3 is implemented. It monitors the environment and captures the network traffic of all pods running the relevant application by performing different steps:

- Identification of the involved nodes and pods
- Creation of a packet capture on these systems
- Monitoring the environment to detect changes
- Adaptation of the capture process as a reaction to the changes

To verify and evaluate the PoC, a testbed of four nodes running Kubernetes 1.20.4 is created. This environment runs different applications on a number of pods, after starting the PoC with the name of one application, the environment is analysed, and a packet capture process is started on every involved system. Possible changes are detected and the process adapted:

```
>>python3 k8cap.py nginx1
Creating capture process at  172.16.40.24 and 172.16.40.25
Start monitoring...
Change detected - adapt capture process
New pod: nginx1-6c46465cc6-dgfmp on node n3 -> 172.16.40.26
Creating capture process at 172.16.40.26
```

The PoC is able to monitor the environment and adapt the capture process, which facilitates the ongoing wiretapping of the suspicious application. So, a dynamic solution for wiretapping Kubernetes environments exists.

Bibliography

[IHG16] A. S. Ibrahim, J. Hamlyn-Harris, J. Grundy. Emerging security challenges of cloud virtual infrastructure. *arXiv preprint arXiv:1612.09059*, 2016.

[JP16] R. Joshi, E. S. Pilli. *Fundamentals of Network Forensics*. Springer, 2016.

[Luk18] M. Luksa. *Kubernetes in Action*. Manning Publications, 2018.

[RUS20] I. Riadi, R. Umar, A. Sugandi. Web Forensic On Kubernetes Cluster Services Using Grr Rapid Response Framework. *International Journal of Scientific & Technology Research* 9:3484–3488, 2020.

[SE16] D. Spiekermann, T. Eggendorfer. Challenges of network forensic investigation in virtual networks. *Journal of Cyber Security and Mobility* vol. 9:15–46, 2016.

[SKE18] D. Spiekermann, J. Keller, T. Eggendorfer. Improving lawful interception in virtual datacenters. In *Proceedings of the Central European Cybersecurity Conference 2018*. Pp. 1–6. 2018.