Zero Trust Network Security Model in containerized environment

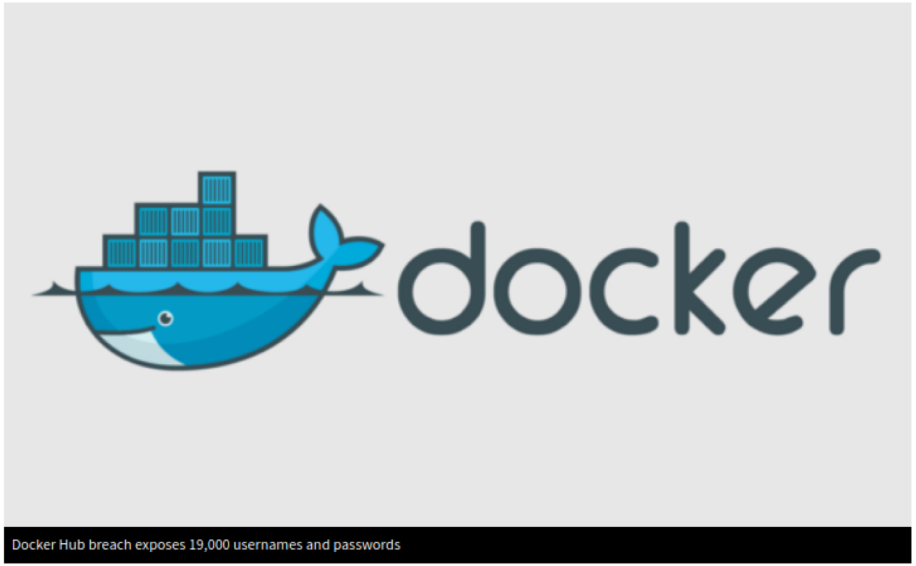
Research Project 1

Supervisor:
Jeroen Scheerder

Research project by:
Catherine de Weever
Marios Andreou

The Problem

Bitbucket and Github access tokens for Docker autobuilds also exposed



CONTAINERISATION OUTFIT Docker has fessed up to a breach of its Hub database that exposed the personal information of approximately 190,000 users.

(1)

- Deploy Container Images with Malicious Code.
- Deploy Benign Container Images and Download Malicious Payloads at Run Time.
- Deploy Malicious Payloads on the Host.
- Obtain Sensitive Information from the Docker Log.

Zero Trust

- Security Model
- Treat traffic, even inside as hostile
- Never trust, always verify
- Strategic approach

Research Question

How to implement Zero Trust for "east/west" traffic between microservices in containerized environment?

- How to regulate the "east/west" traffic flow?
- How to implement confidentiality at transit data?

Methodology

- Get to know the current setup of ON2IT
- Find out what is missing
- Literature study to find solutions
- Implement a proof of concept for viability

Related Work

- *Casimer DeCusatis et al.*
  - transport-level approach (first packet authentication)
  - protection only on layer 3/4

- *Fatima Hussain et al.*
  - API gateway/proxy-based approach (secure API service mesh)
  - Istio and Kubernetes

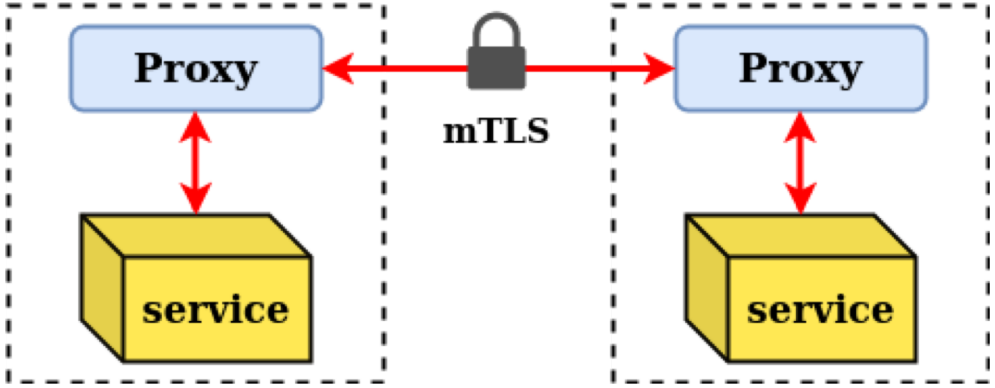
- *Zirak Zaheer et al.*
  - microservice identities (eZtrust)
  - extended Berkeley Packet Filter (eBPF)
  - Proof of concept only for visibility

ON2IT current solution

- Zero Trust approach
- Containers are segmented using Istio (sidecar)
- Data encrypted in transit using Istio
- No deep traffic visibility

Background: Istio

- Micro-segmentation
  - Envoy Sidecar proxy
- Encryption
  - mutual TLS



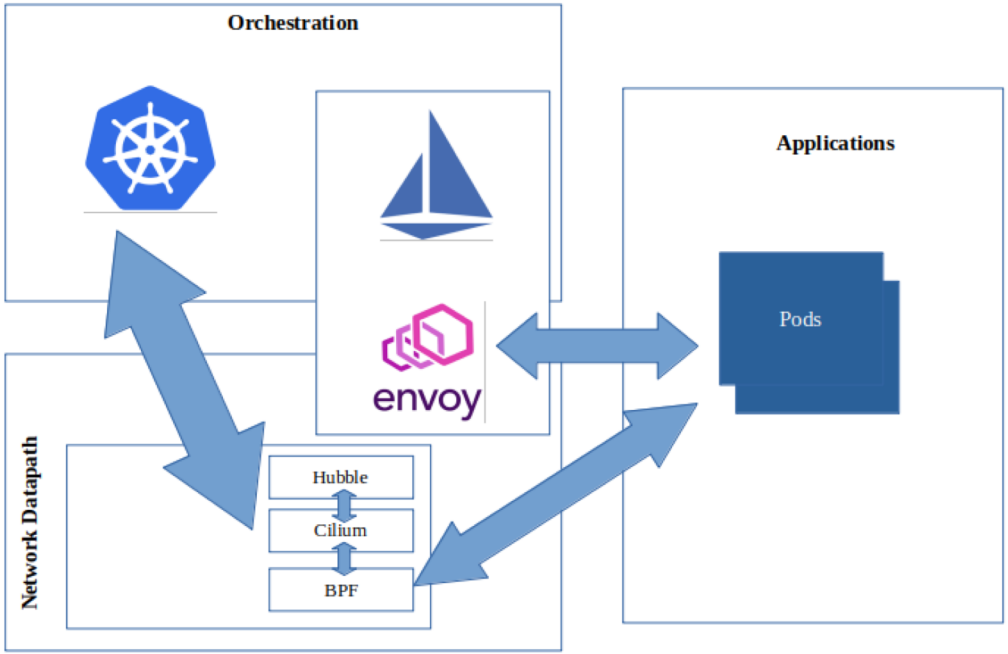
Sidecar proxy deployment

Background

- Cilium
  - Berkeley Packet Filter (BPF)
  - Security visibility and Enforcement

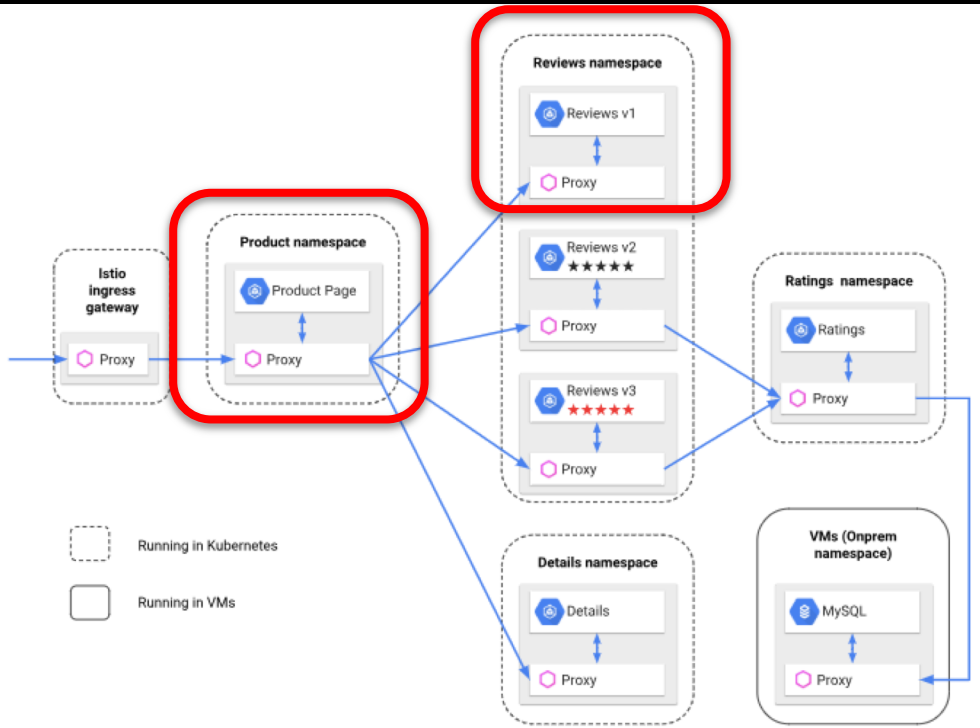
- Hubble
  - Requires Cilium and extended Berkeley Packet Filter (eBPF)
  - Deep visibility into the communication
  - TCP connections, DNS queries, HTTP requests, etc.

Setup



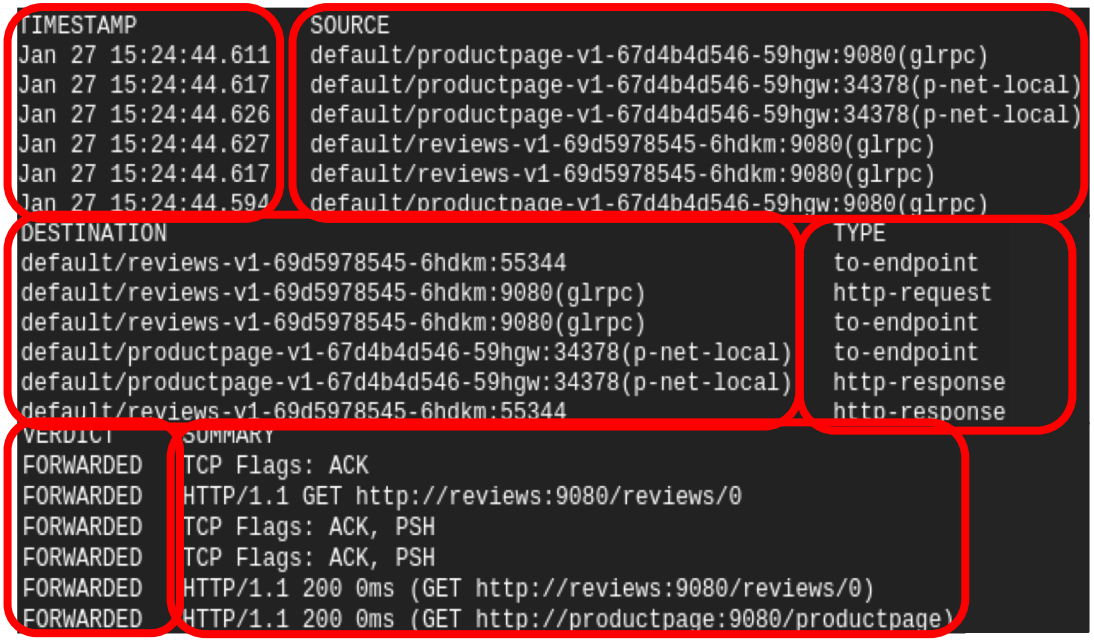
- *Google Cloud Platform*
  - *Google Kubernetes Engine*
    - *1 cluster*
    - *4 nodes*
  - *Cilium*
    - *Berkeley Packet Filter*
  - *Istio*
    - *Envoy Proxy*
    - *Built on top of Cilium*
  - *Hubble*
    - *Built on top of Istio*

Demo Application



- A demo application deployed for the purpose of having a realistic environment.
- Monitor traffic between “Product Page” proxy and “Review v1” proxy.

Proof of Concept(1)



TIMESTAMP	SOURCE	DESTINATION	TYPE	VERDICT	SUMMARY
Jan 27 15:24:44.611	default/productpage-v1-67d4b4d546-59hgw:9080(glrpc)	default/reviews-v1-69d5978545-6hdkm:55344	to-endpoint	FORWARDED	TCP Flags: ACK
Jan 27 15:24:44.617	default/productpage-v1-67d4b4d546-59hgw:34378(p-net-local)	default/reviews-v1-69d5978545-6hdkm:9080(glrpc)	http-request	FORWARDED	HTTP/1.1 GET http://reviews:9080/reviews/0
Jan 27 15:24:44.626	default/productpage-v1-67d4b4d546-59hgw:34378(p-net-local)	default/reviews-v1-69d5978545-6hdkm:9080(glrpc)	to-endpoint	FORWARDED	TCP Flags: ACK, PSH
Jan 27 15:24:44.627	default/reviews-v1-69d5978545-6hdkm:9080(glrpc)	default/productpage-v1-67d4b4d546-59hgw:34378(p-net-local)	to-endpoint	FORWARDED	TCP Flags: ACK, PSH
Jan 27 15:24:44.617	default/reviews-v1-69d5978545-6hdkm:9080(glrpc)	default/productpage-v1-67d4b4d546-59hgw:34378(p-net-local)	http-response	FORWARDED	HTTP/1.1 200 0ms (GET http://reviews:9080/reviews/0)
Jan 27 15:24:44.594	default/productpage-v1-67d4b4d546-59hgw:9080(glrpc)	default/reviews-v1-69d5978545-6hdkm:55344	http-response	FORWARDED	HTTP/1.1 200 0ms (GET http://productpage:9080/productpage)

- Hubble enables deep visibility for the following metrics:
  - DNS
  - Drop
  - TCP
  - Port-Distribution
  - ICMP
  - HTTP

Proof of Concept(2)

- Encryption

```
HOST:PORT                                  STATUS     SERVER     CLIENT     AUTHN POLICY     DESTINATION RULE
productpage.default.svc.cluster.local:9080 OK         mTLS       mTLS       default/         default/istio-system
```

```
HOST:PORT                                  STATUS     SERVER     CLIENT     AUTHN POLICY     DESTINATION RULE
reviews.default.svc.cluster.local:9080     OK         mTLS       mTLS       default/         reviews/default
```

- Micro-segmentation
  - Reviews-v1 IP → 10.56.1.112

```
root@gke-cluster1-default-pool-25fba10d-5gw8:/# curl 10.56.1.112:9080
curl: (56) Recv failure: Connection reset by peer
```

Discussion(1)

Zero Trust Operational Controls present:

- Istio:
  - SSL encryption for “east-west” and “north-south” traffic
  - Centrally managed
  - Micro-Segmentation
  - RBAC Based Controls (deprecated) → Authorization Policy
  - Restricted inbound and outbound access

Discussion(2)

Zero Trust Operational Controls present:

- Cilium:
  - Enhances network security rules/policies
- Hubble:
  - Data classification
  - Traffic-inspection
  - Behavioral analytics

Conclusion(1)

- Regulate traffic:
  - Micro-segmentation provided by Istio
  - Traffic visibility provided by Hubble in combination with Cilium and eBPF

- Confidentiality at transit data:
  - Encryption provided by Istio

Conclusion(2)

How to implement Zero Trust for "east/west" traffic between microservices in containerized environment?

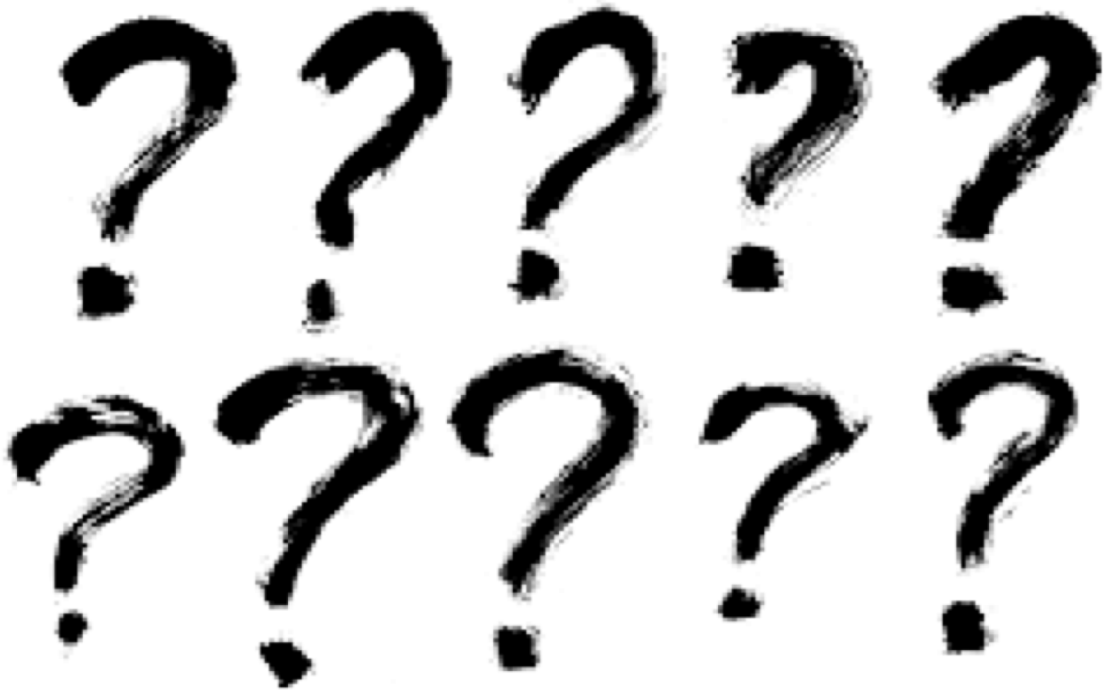
Appropriate Zero Trust Controls:

- Encryption in Transit
- Centrally managed
- Micro-Segments
- Data classification
- Traffic-inspection
- Authorization Policies

Future Work

- Data leakage detection (DLP controls)
- Content-Inspection of packets
- Behavioral analytics
- Automation
    - Logging

Questions



References

1) https://www.theinquirer.net/inquirer/news/3074793/docker-hub-breach

1) https://unit42.paloaltonetworks.com/attackers-tactics-and-techniques-in-unsecured-docker-daemons-revealed/