Google

Secure Pods

Sandboxing workloads in Kubernetes

Tim Allclair < tallclair@google.com >

Software Engineer, Google

@tallclair

What is a secure pod?

Google Cloud

Threats
from the *outside*

Keeping the attackers out:

- Application Security
- Firewall
- Integrity Checks
- Intrusion Detection
- ...

Threats
from the *inside*

Keeping the attackers in.

Why do we care?

Who put the attackers there in the first place?

1. Untrusted Code
2. Containment

How do we protect from internal threats?

02

Google Cloud

Attack Surfaces

- Kernel
- Storage
- Network
- Daemons

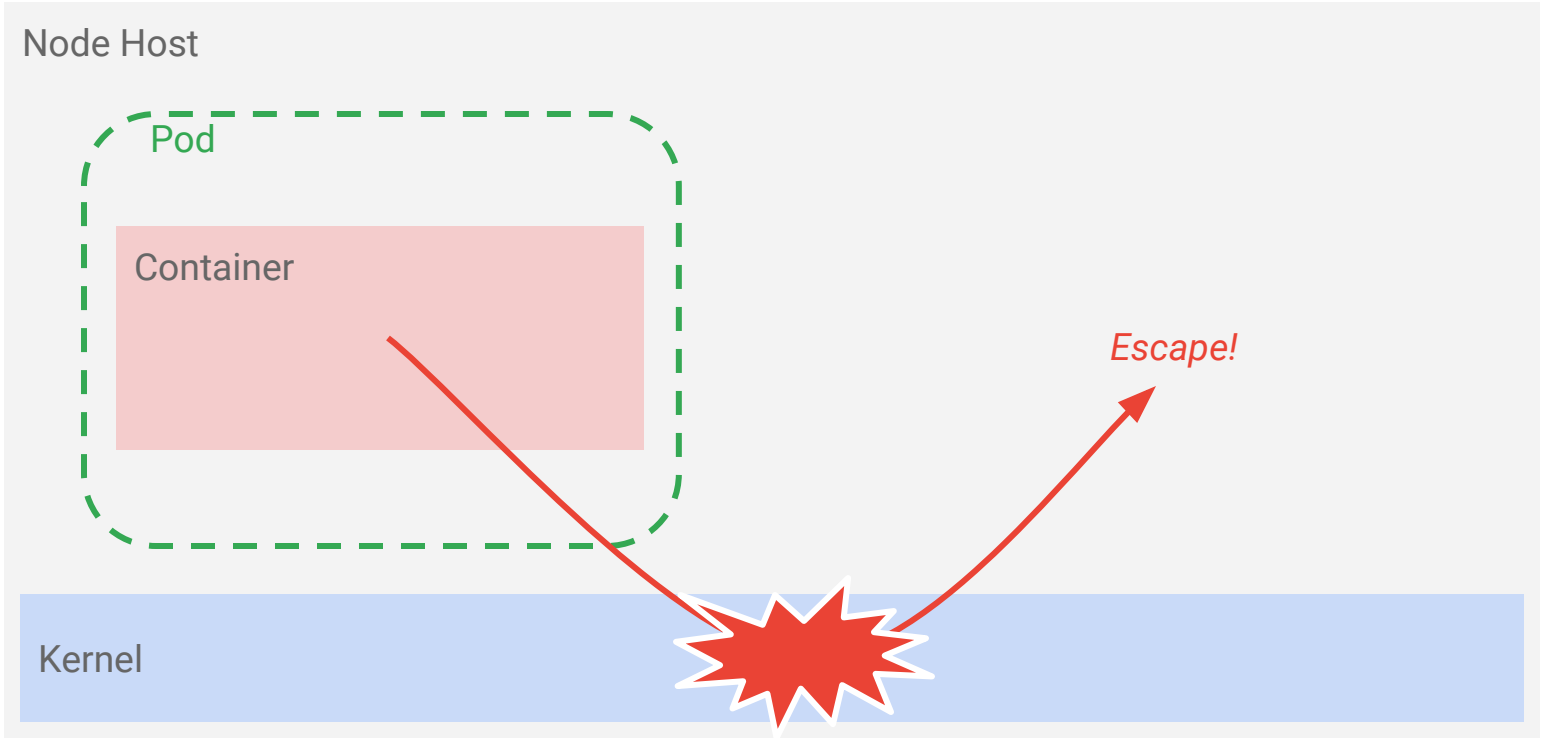
  Logging, monitoring, ...
- Hardware
- ...

Attack Surfaces

- **Kernel**
- Storage
- Network
- Daemons

  Logging, monitoring, ...
- Hardware
- ...

Attacks via the **Kernel**



Linux » Linux Kernel : Security Vulnerabilities Published In 2017 (Execute Code)

2017 : January February March April May June July August September October November December CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Total number of vulnerabilities : 169 Page : 1 (This Page) 2 3 4

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2016-10229	358		Exec Code	2017-04-04	2017-09-19	10.0	None	Remote	Low	Not required	Complete	Complete	Complete
udp.c in the Linux kernel before 4.5 allows remote attackers to execute arbitrary code via UDP traffic that triggers an unsafe second checksum calculation during execution of a recv system call with the MSG_PEEK flag.														
2	CVE-2017-0561	264		Exec Code	2017-04-07	2017-08-15	10.0	None	Remote	Low	Not required	Complete	Complete	Complete
A remote code execution vulnerability in the Broadcom Wi-Fi firmware could enable a remote attacker to execute arbitrary code within the context of the Wi-Fi SoC. This issue is rated as Critical due to the possibility of remote code execution in the context of the Wi-Fi SoC. Product: Android. Versions: Kernel-3.10, Kernel-3.18. Android ID: A-34199105. References: B-RB#110814.														
3	CVE-2017-13715	20		DoS Exec Code	2017-08-28	2017-09-08	10.0	None	Remote	Low	Not required	Complete	Complete	Complete
The __skb_flow_dissect function in net/core/flow_dissector.c in the Linux kernel before 4.3 does not ensure that n_proto, ip_proto, and thoff are initialized, which allows remote attackers to cause a denial of service (system crash) or possibly execute arbitrary code via a single crafted MPLS packet.														
4	CVE-2016-6758	284		Exec Code +Priv	2017-01-12	2017-01-19	9.3	None	Remote	Medium	Not required	Complete	Complete	Complete
An elevation of privilege vulnerability in Qualcomm media codecs could enable a local malicious application to execute arbitrary code within the context of a privileged process. This issue is rated as High because it could be used to gain local access to elevated capabilities, which are not normally accessible to a third-party application. Product: Android. Versions: Kernel-3.10, Kernel-3.18. Android ID: A-30148882. References: QC-CR#1071731.														
5	CVE-2016-6759	284		Exec Code +Priv	2017-01-12	2017-01-19	9.3	None	Remote	Medium	Not required	Complete	Complete	Complete
An elevation of privilege vulnerability in Qualcomm media codecs could enable a local malicious application to execute arbitrary code within the context of a privileged process. This issue is rated as High because it could be used to gain local access to elevated capabilities, which are not normally accessible to a third-party application. Product: Android. Versions: Kernel-3.10, Kernel-3.18. Android ID: A-29982686. References: QC-CR#1055766.														
6	CVE-2016-6760	284		Exec Code +Priv	2017-01-12	2017-01-19	9.3	None	Remote	Medium	Not required	Complete	Complete	Complete
An elevation of privilege vulnerability in Qualcomm media codecs could enable a local malicious application to execute arbitrary code within the context of a privileged process. This issue is rated as High because it could be used to gain local access to elevated capabilities, which are not normally accessible to a third-party application. Product: Android. Versions: Kernel-3.10, Kernel-3.18. Android ID: A-29617572. References: QC-CR#1055783.														
7	CVE-2016-6761	284		Exec Code +Priv	2017-01-12	2017-01-19	9.3	None	Remote	Medium	Not required	Complete	Complete	Complete
An elevation of privilege vulnerability in Qualcomm media codecs could enable a local malicious application to execute arbitrary code within the context of a privileged process. This issue is rated as High because it could be used to gain local access to elevated capabilities, which are not normally accessible to a third-party application. Product: Android. Versions: Kernel-3.10, Kernel-3.18. Android ID: A-29421682. References: QC-CR#1055792.														

Source: https://www.cvedetails.com/vulnerability-list/vendor_id-33/product_id-47/year-2017/opec-1/Linux-Linux-Kernel.html

Today: kernel isolation features

```
apiVersion: v1
kind: Pod
metadata:
  name:   restricted-pod
  annotations:
    seccomp.security.alpha.kubernetes.io/pod: docker/default
    apparmor.security.beta.kubernetes.io/pod: runtime/default
spec:
  securityContext:
    runAsUser:    1234
    runAsNonRoot: true
  containers:
    - name:  untrusted-container
      image: sketchy:v1
      securityContext:
        allowPrivilegeEscalation: false
```

Today: kernel isolation features

```
apiVersion: v1
kind: Pod
metadata:
  name:   restricted-pod
  annotations:
    seccomp.security.alpha.kubernetes.io/pod: docker/default
    apparmor.security.beta.kubernetes.io/pod: runtime/default
spec:
  securityContext:
    runAsUser:    1234
    runAsNonRoot: true
  containers:
    - name:  untrusted-container
      image: sketchy:v1
      securityContext:
        allowPrivilegeEscalation: false
```

Today: kernel isolation features

```
apiVersion: v1
kind: Pod
metadata:
  name:   restricted-pod
  annotations:
    seccomp.security.alpha.kubernetes.io/pod: docker/default
    apparmor.security.beta.kubernetes.io/pod: runtime/default
spec:
  securityContext:
    runAsUser:    1234
    runAsNonRoot: true
  containers:
    - name:  untrusted-container
      image: sketchy:v1
      securityContext:
        allowPrivilegeEscalation: false
```

Today: kernel isolation features

```
apiVersion: v1
kind: Pod
metadata:
  name:   restricted-pod
  annotations:
    seccomp.security.alpha.kubernetes.io/pod: docker/default
    apparmor.security.beta.kubernetes.io/pod: runtime/default
spec:
  securityContext:
    runAsUser:    1234
    runAsNonRoot: true
  containers:
    - name:  untrusted-container
      image: sketchy:v1
      securityContext:
        allowPrivilegeEscalation: false
```

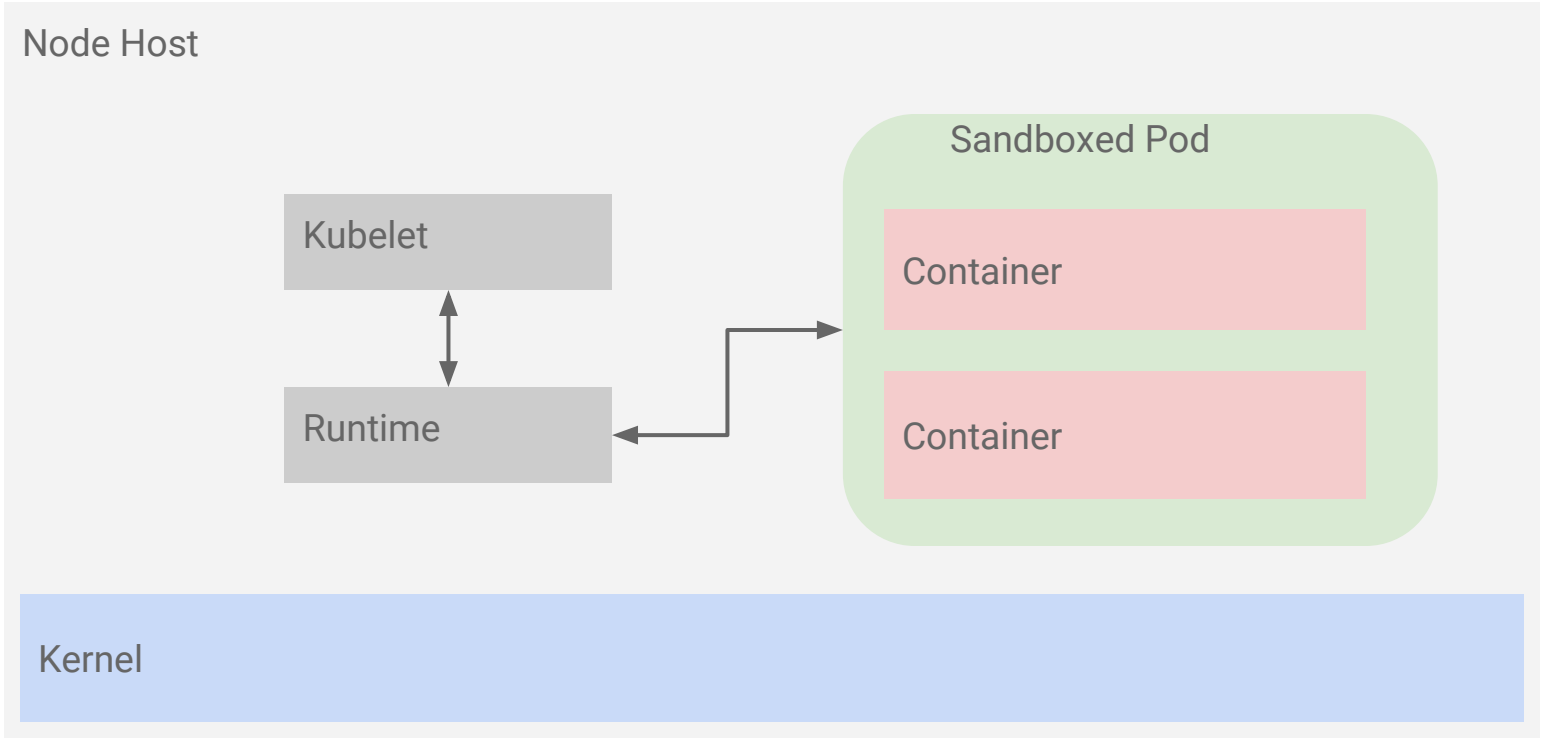
Dirty COW

CVE-2016-5195

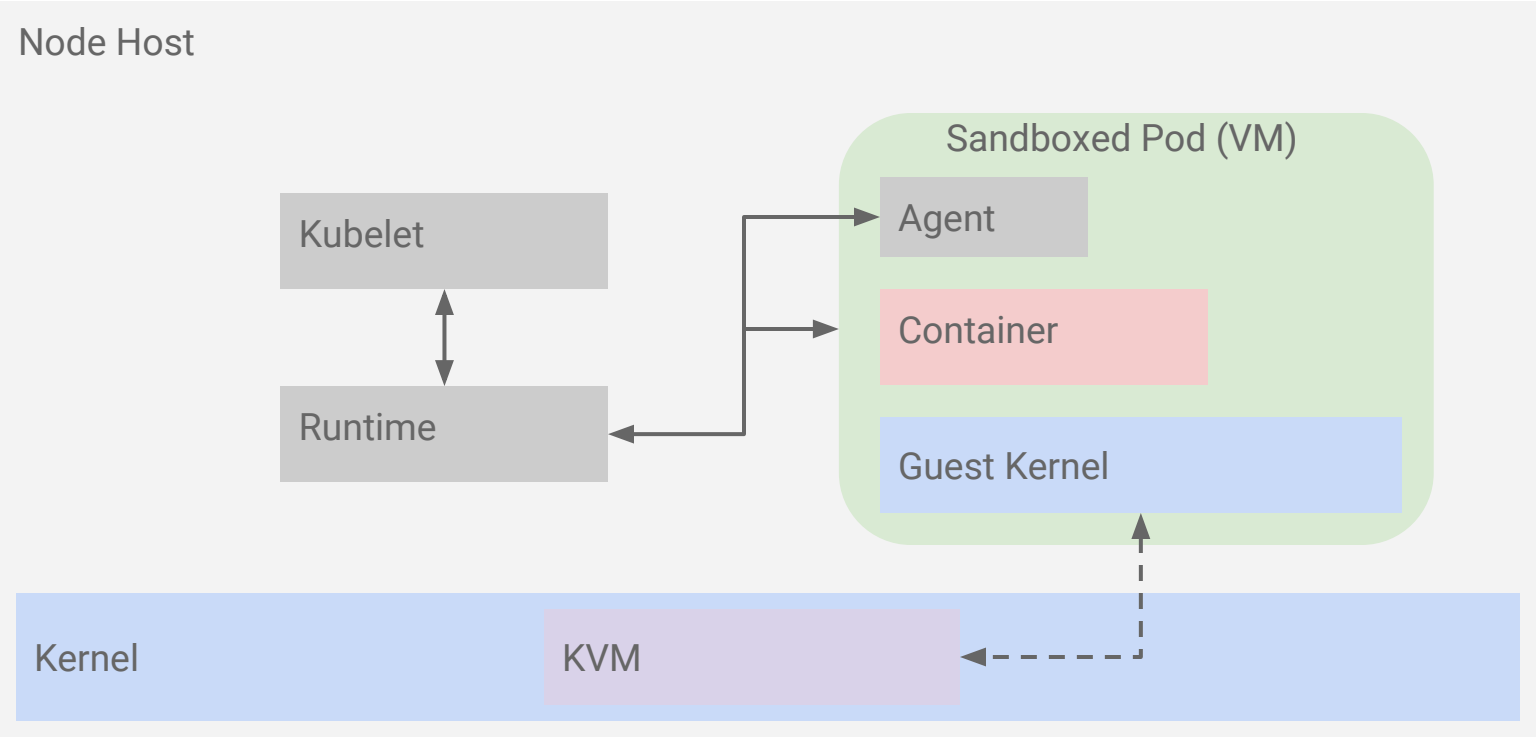
PoC escape demonstrated via the vDSO by @scumjr

Image credit: Ian Paterson / CC BY-SA 2.0

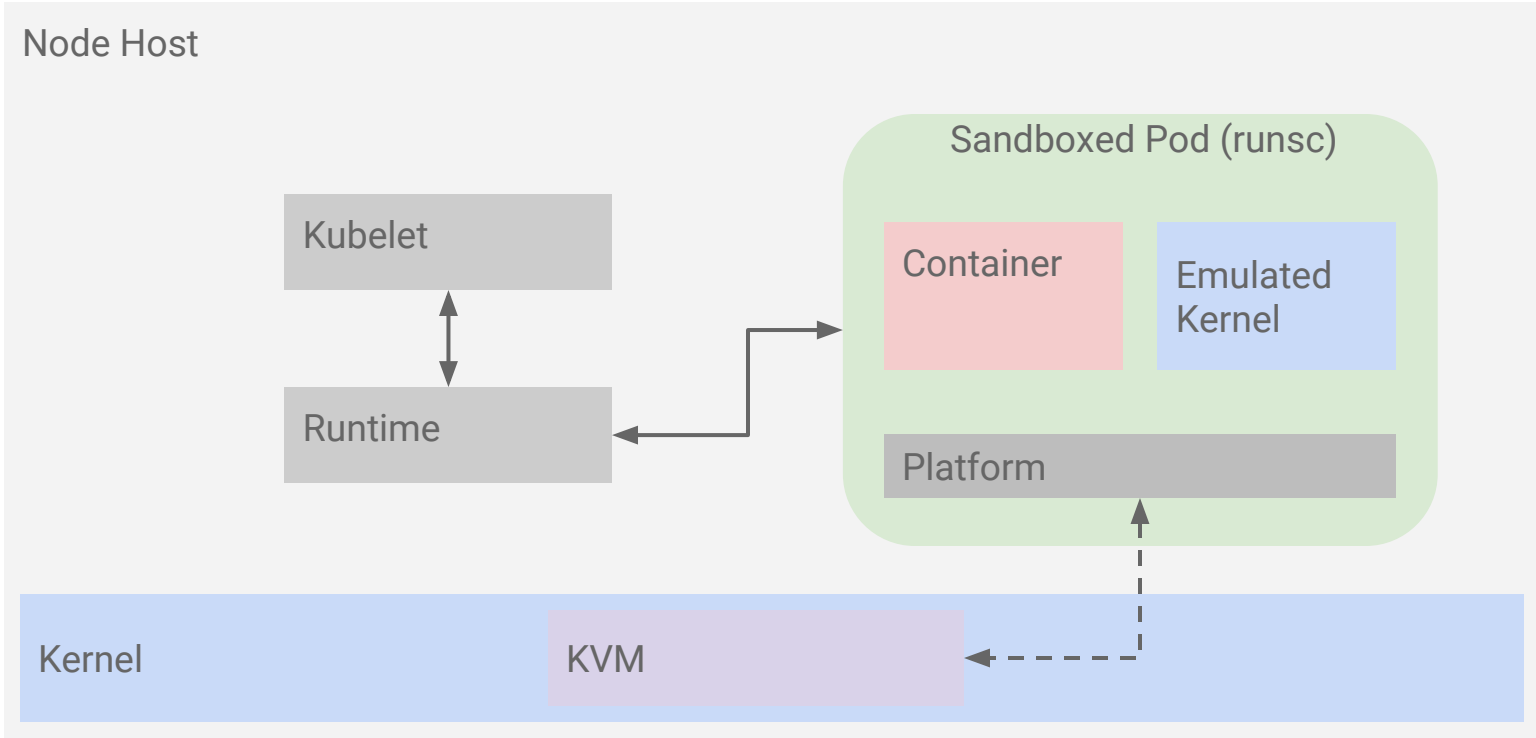
Sandboxes


Node Host
Sandboxed Pod
Kubelet
Container
Runtime
Container
Kernel

Sandboxes - kata containers



Sandboxes - gVisor

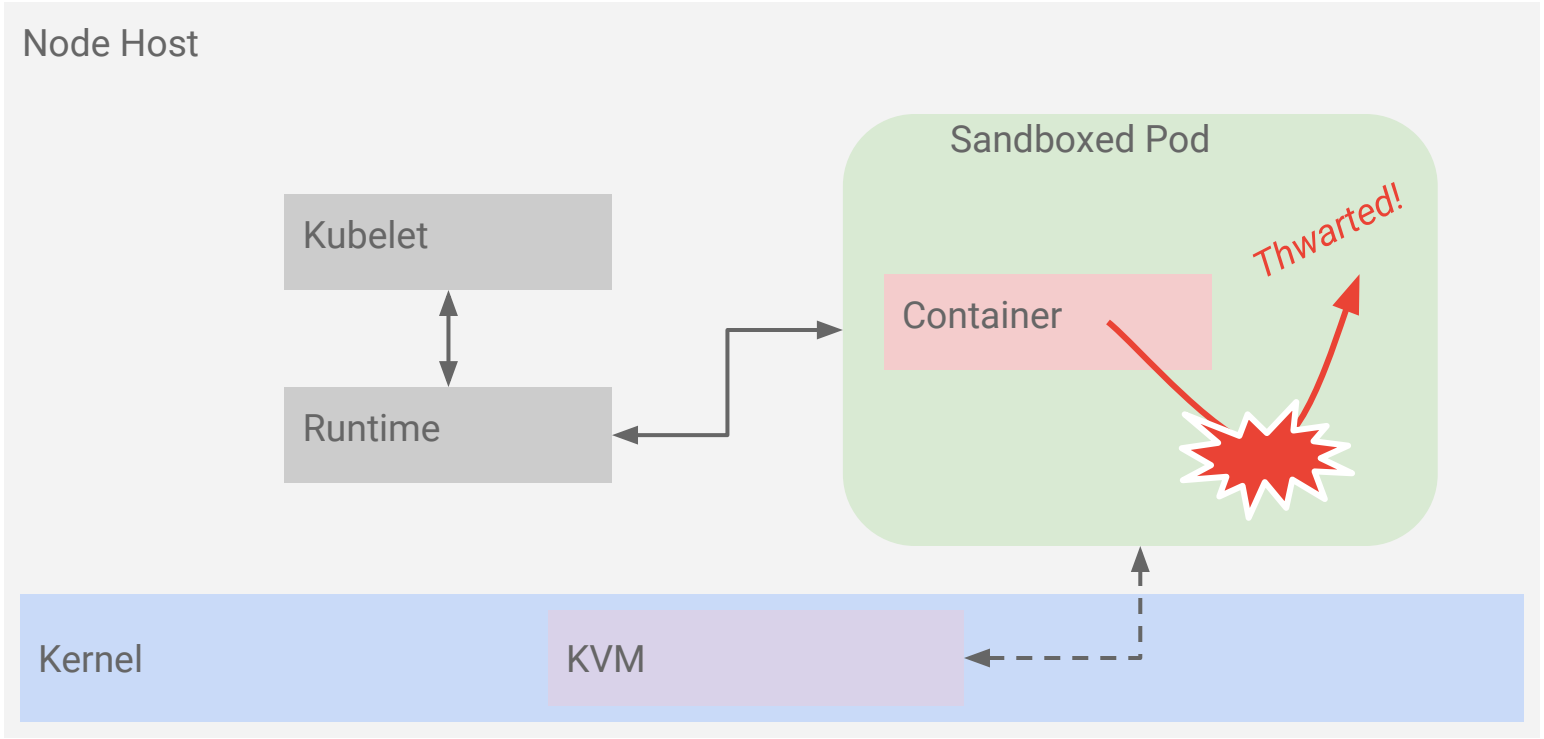


Future: sandboxed

API still under discussion

```
apiVersion: v1
kind: Pod
metadata:
  name:   sandboxed-pod
spec:
  securityContext:
    sandboxed: true
  containers:
    - name:  untrusted-container
      image: sketchy:v1
```

Sandboxes


Node Host
Sandboxed Pod
Kubelet
Runtime
Container
Thwarted!
Kernel
KVM

What's the catch?

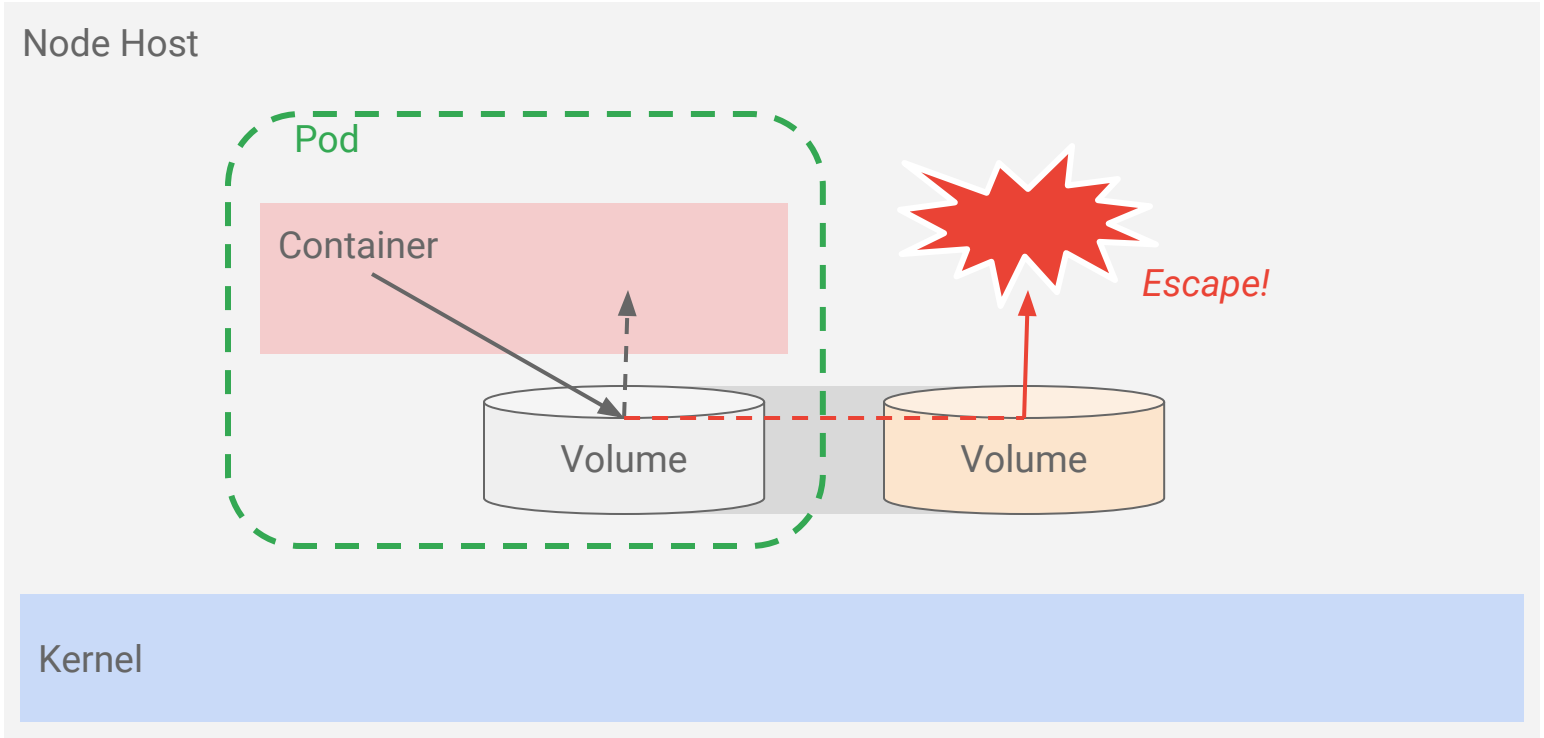
1. Cost & Performance
   But getting better!
2. Not 100% compatible
   Close though
3. Incomplete solution
   Requires network hardening

Attack Surfaces

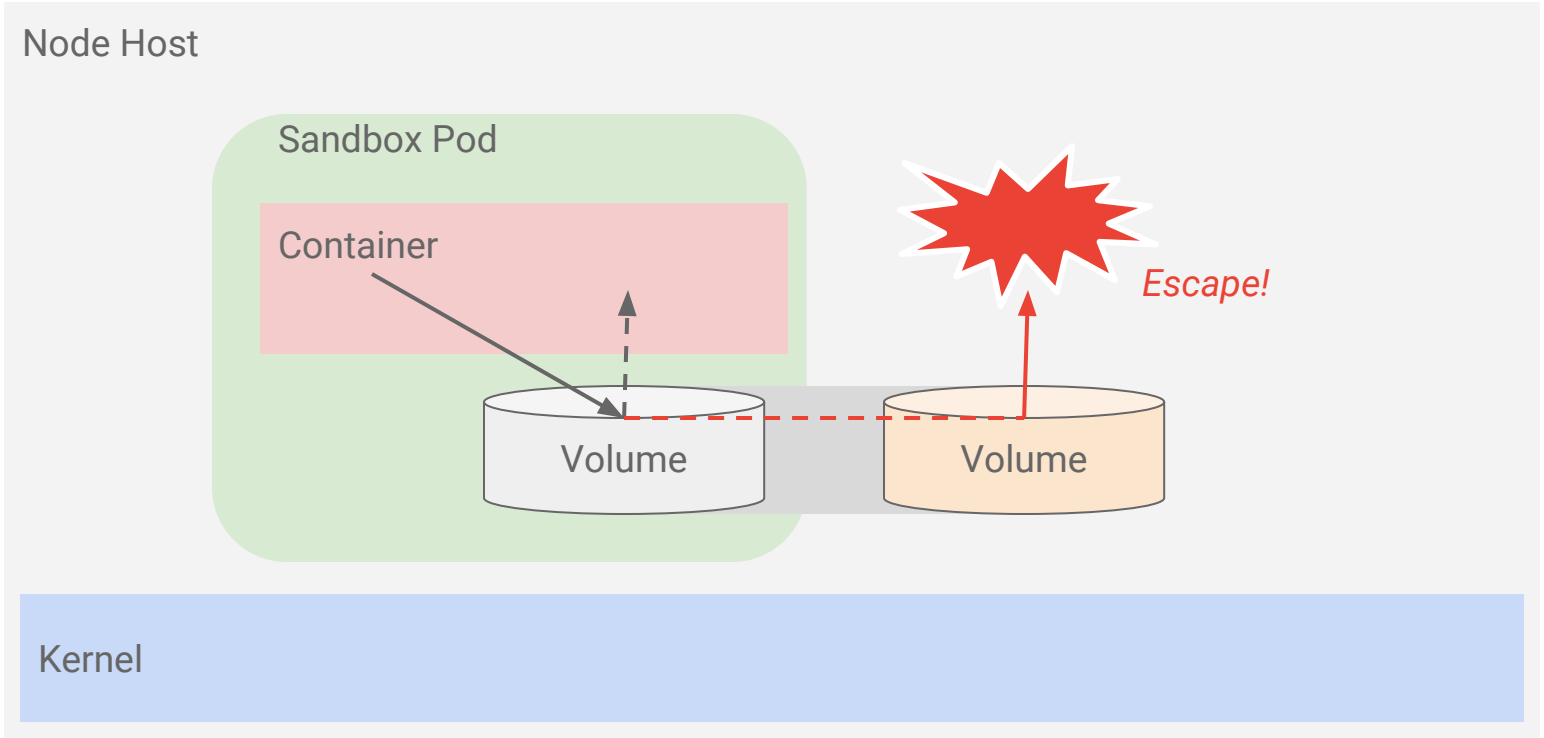
- Kernel
- **Storage**
- Network
- Daemons

  Logging, monitoring, ...
- Hardware
- ...

CVE-2017-1002101: Host-resolved **symlinks**



CVE-2017-1002101: Host-resolved **symlinks**



TODO: Sandboxed storage

1. Readonly storage via **readonly protocols**
2. Ephemeral storage **opaque to host**
3. **Direct access** block volumes
4. Sandboxed persistent filesystems **???**

Image credit: Charles J Sharp

Attack Surfaces

- Kernel
- Storage
- **Network**
- Daemons

  Logging, monitoring, ...
- Hardware
- ...

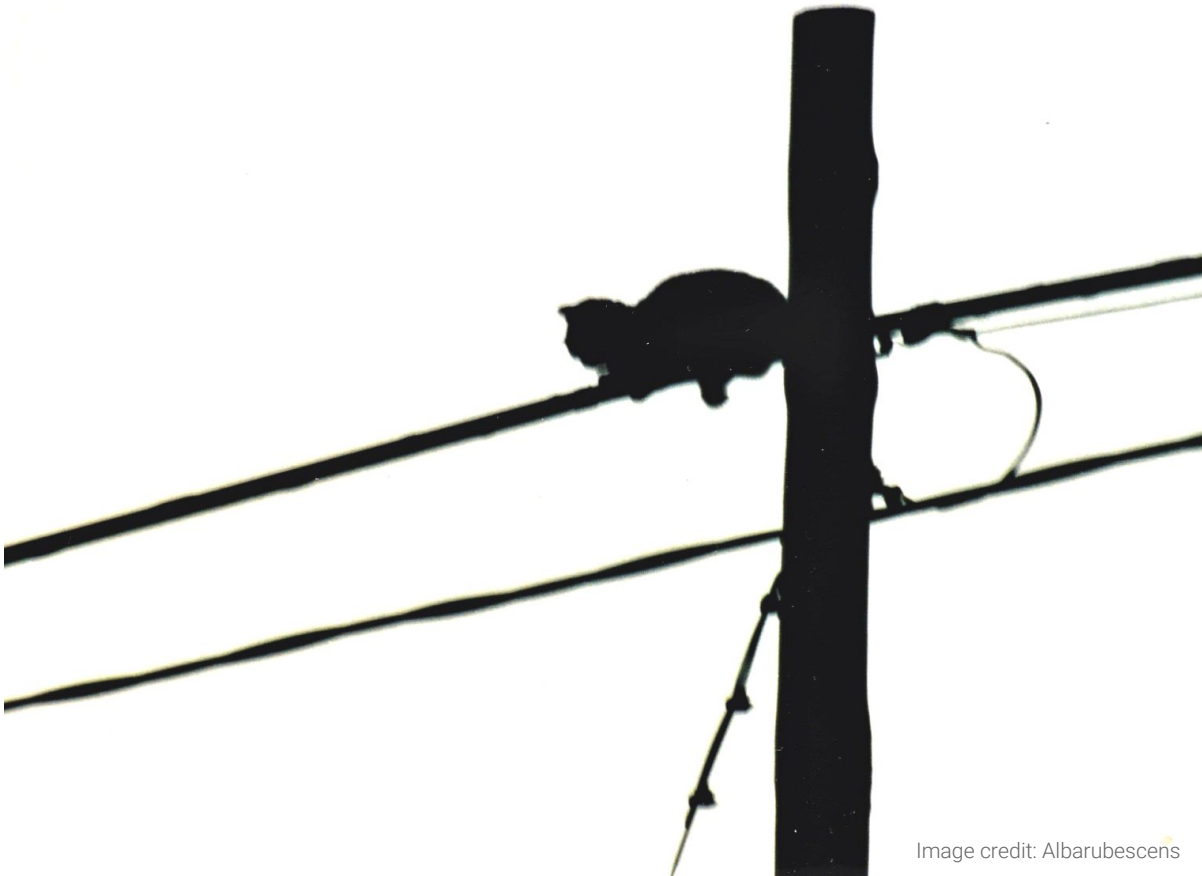
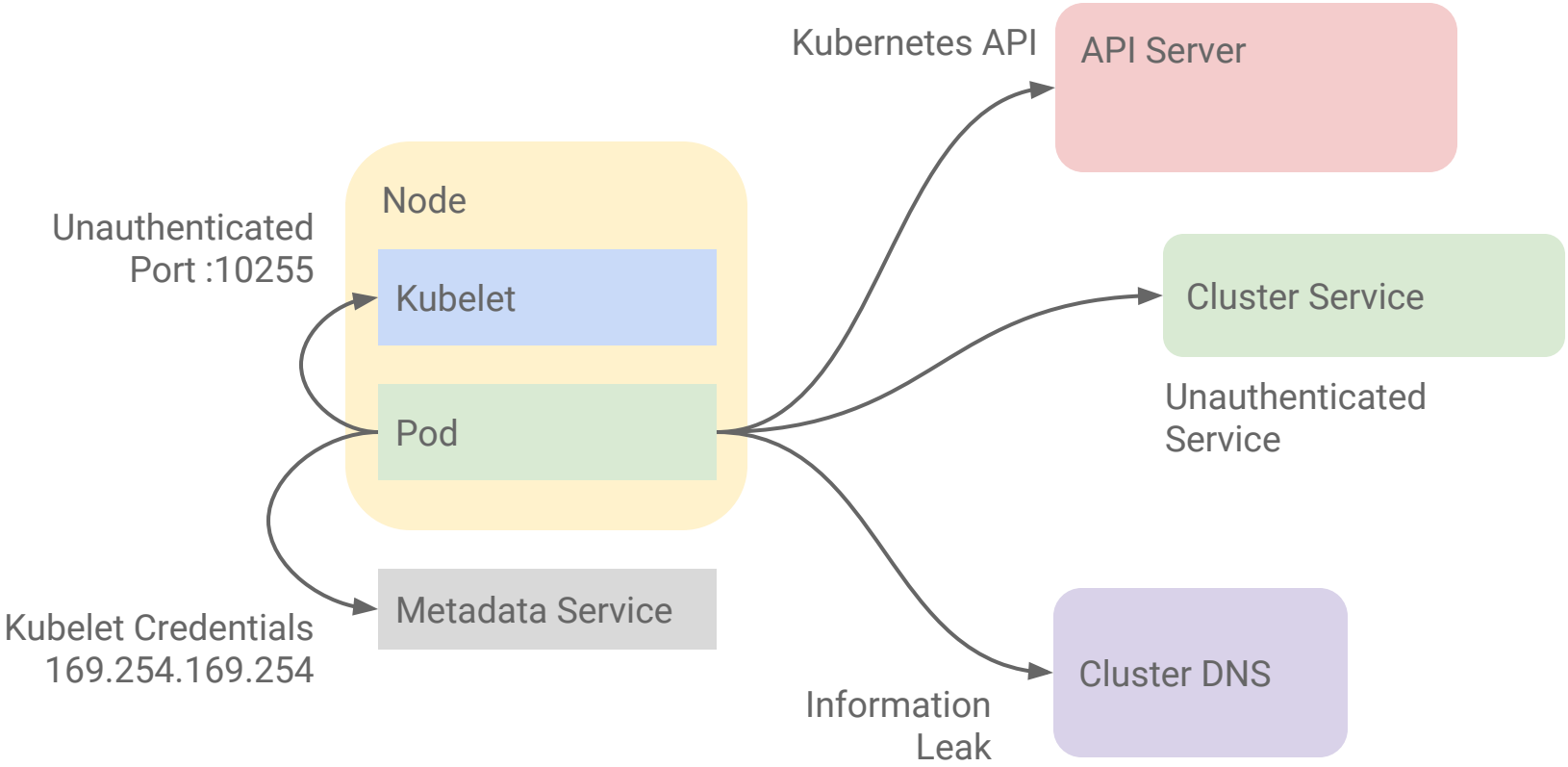


Image credit: Albarubescens

Attacks over the **network**



Network Policy

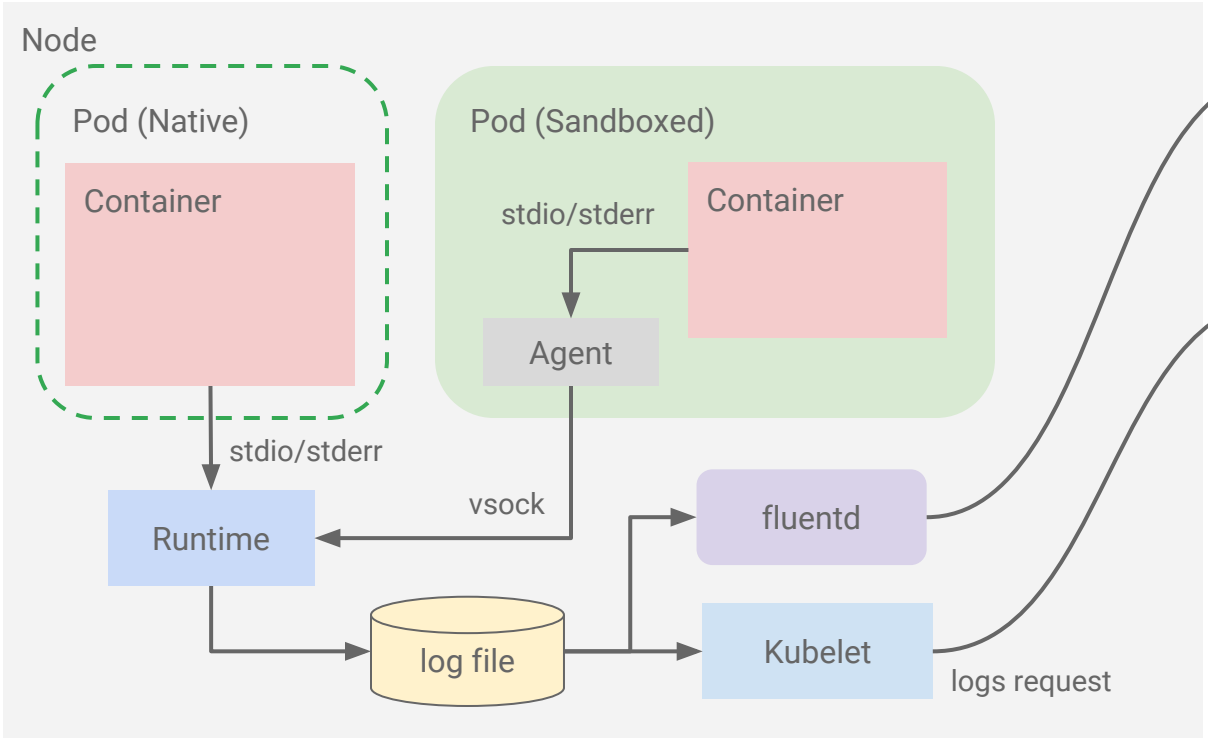
```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata: { ... }
spec:
  podSelector:
    matchLabels:
      sandboxed: true
  policyTypes:
  - Egress
  egress:
  - to:
    - ipBlock:
        cidr: 0.0.0.0/0
        except:
        - 10.0.0.0/8
        - 172.16.0.0/12
        - 192.168.0.0/16
```

Attack Surfaces

- Kernel
- Storage
- Network
- Daemons

  Logging, *monitoring, ...*
- Hardware
- ...

Attacks via system **logs**



Google

Attack Surfaces

- Kernel
- Storage
- Network
- Daemons

  Logging, monitoring, ...
- **Hardware**
- ...

Image credit: Jonas Löwgren

Attacks via the **Hardware**

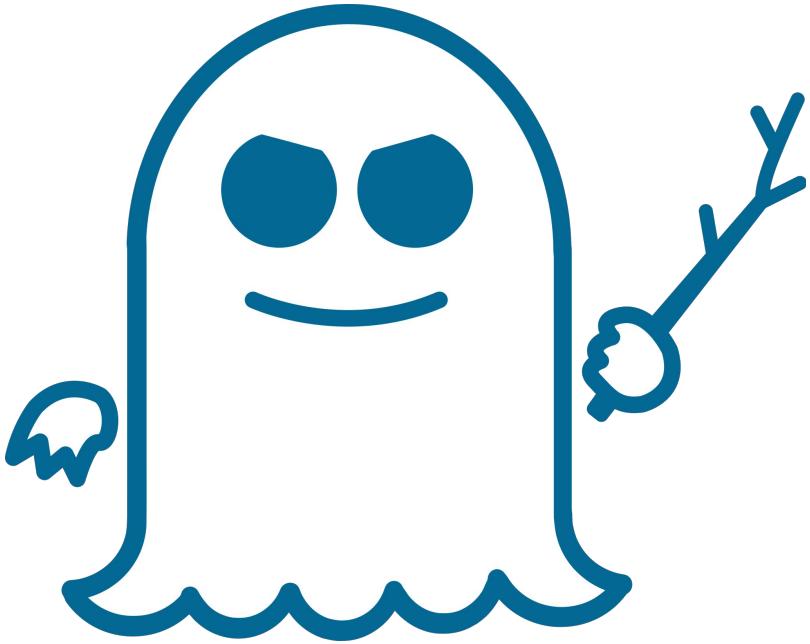


Image credit: William Warby

Summary

- Kubernetes is a **complex system** with many layers of **attack surfaces** exposed to **internal threats**
- **Sandboxes** is an upcoming feature to mitigate many of those threats
  i. Leverage hypervisor isolation
  ii. Deeper Kubernetes integration for enhanced protection

Roadmap

- **Experimental** today!
  i. Annotations in CRI-O & containerd
- **Alpha** in 1.12
  i. Kubernetes & CRI API
  ii. Basic Kata & gVisor implementation
  iii. Improved resource management
- **Beta**
  i. Hardened storage interfaces
  ii. Hardened logging & monitoring

Image credit: William Warby

Get Involved!

Join the Conversation

Sandboxes: https://goo.gl/eQHuqo

SIG-Node: https://github.com/kubernetes/community/tree/master/sig-node

Contribute

Kata Containers: https://katacontainers.io/

gVisor: https://github.com/google/gvisor

Thank you!

Learn More:

Sandboxes: https://goo.gl/eQHuqo

Docker non-events: https://docs.docker.com/engine/security/non-events/

Dirty Cow: https://github.com/scumjr/dirtycow-vdso
https://blog.paranoidsoftware.com/dirty-cow-cve-2016-5195-docker-container-escape/

Symlink vulnerability: https://kubernetes.io/blog/2018/04/04/fixing-subpath-volume-vulnerability/

Network Policy: https://kubernetes.io/docs/concepts/services-networking/network-policies/

Tim Allclair <tallclair@google.com>