Container Security and Sandboxing

Yiying Zhang

Lec6: Security Implications of Containers

TABLE 1. Threat model specifications for apps, containers, and host for the studied use cases. *'Semi'* refers to semi-honest. Apps in semi-honest/malicious containers can be semi-honest or malicious too.

Use Case	Apps can be			Containers can be			Host can be		
	honest	semi	malicious	honest	semi	malicious	honest	semi	malicious
(I) Protect container from applications	✓	✓	✓	✓	-	-	✓	-	-
(II) Inter-container protection	✓	-	-	✓	✓	✓	✓	-	-
(III) Protect host from containers	✓	✓	✓	✓	✓	✓	✓	-	-
(IV) Protect containers from host	✓	-	-	✓	-	-	✓	✓	✓

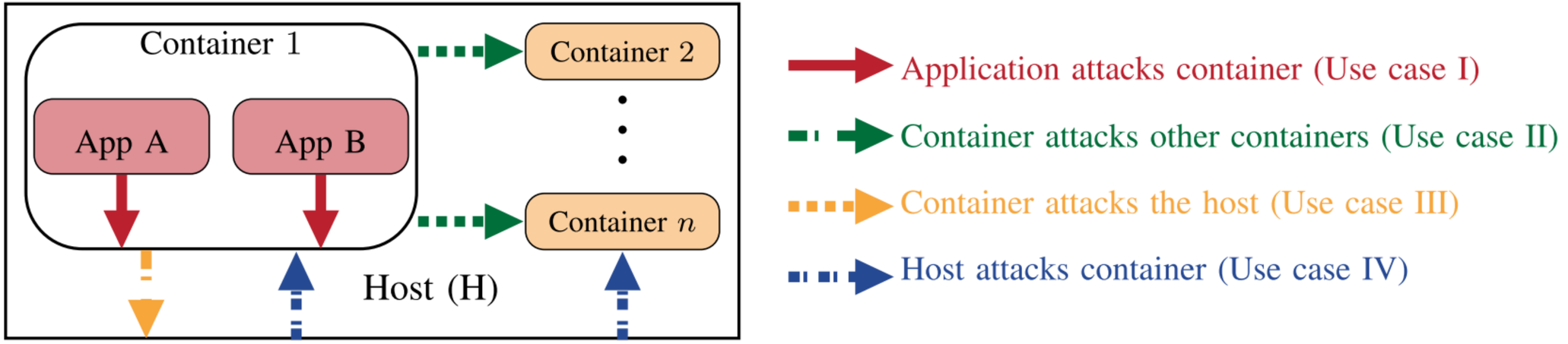


FIGURE 3. Overview of security protection requirements in containers.

Source: S. Sultan et al.: Container Security: Issues, Challenges, and the Road Ahead

Lec6: Threats of Containers

- Unlike VMs whose interface is hardware instructions, containers' interface is OS system calls
- More difficult to protect syscalls
  - Involve large amount of code in the OS
  - And there are many syscalls

Lec6: Threats of Container Images

- Difficult to understand the source/provenance of images

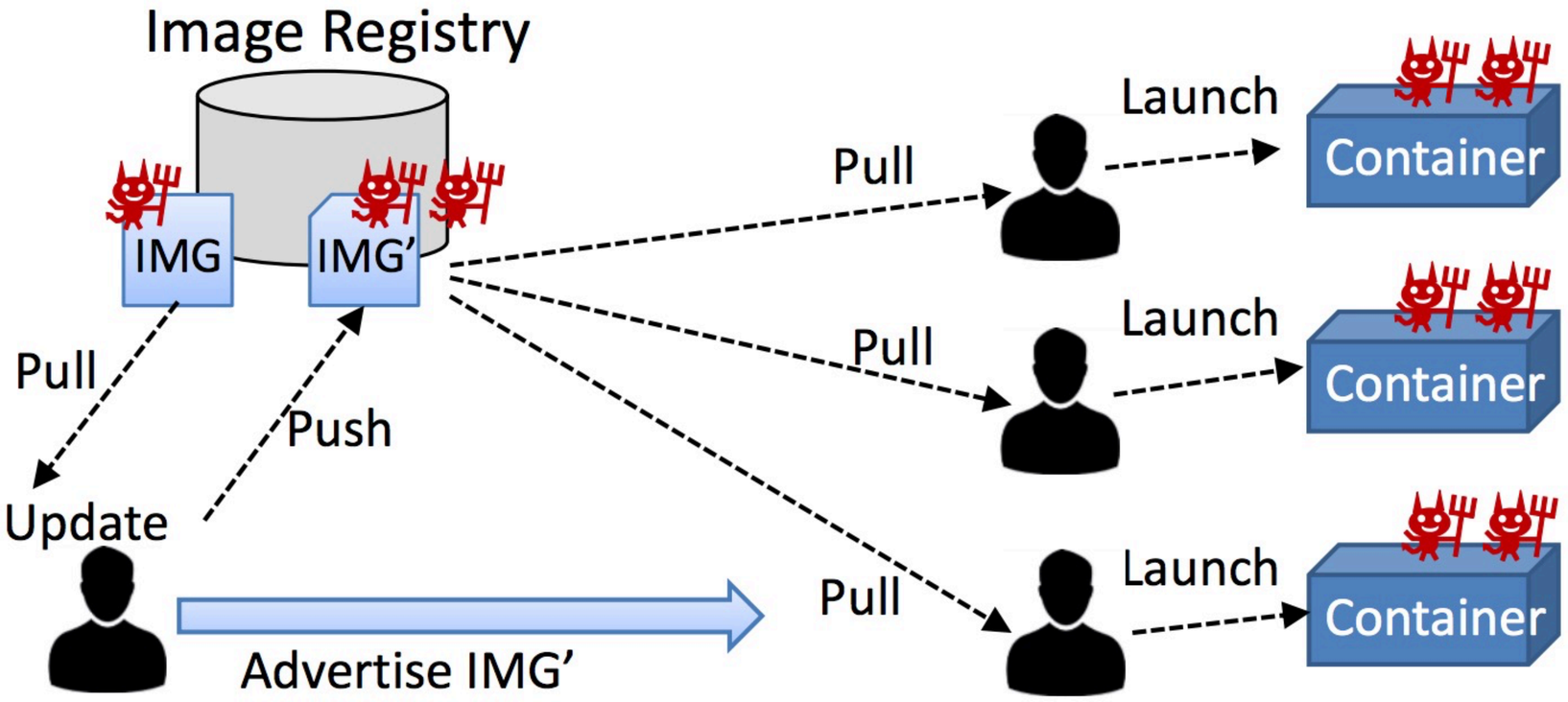


Figure 1: Scenario of vulnerability spread

Source: B. Tak et al.: Understanding Security Implications of Using Containers in the Cloud

Google

Secure Pods

Sandboxing workloads in Kubernetes

Tim Allclair < tallclair@google.com >

Software Engineer, Google

@tallclair

What is a secure pod?

01

Threats
from the *outside*

Keeping the attackers out:

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

Google

Threats
from the *inside*

Keeping the attackers in.

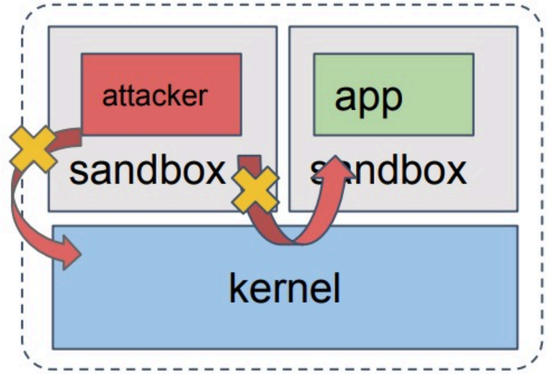
How do we protect from internal threats?

02

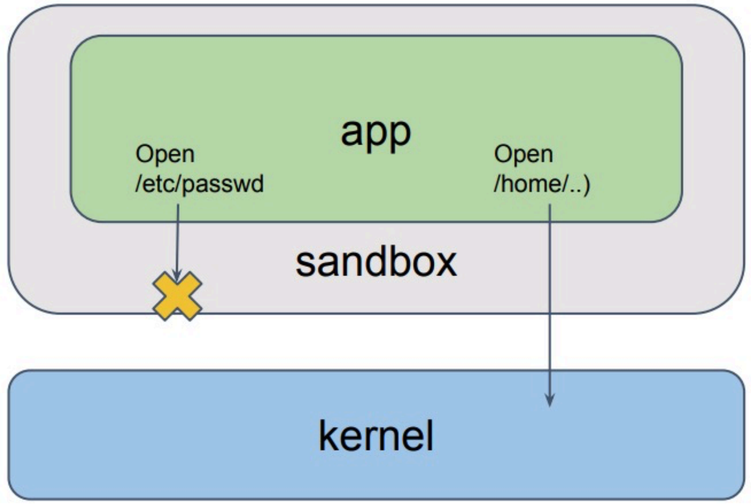
Google Cloud

Lec7: Sandboxing

- Rule-based sandboxing: reduce the attack surface by restricting what applications can access
  - e.g., AppArmor, SELinux, Seccomp-bpf
  - Rules can be fragile (not properly capture threats) and can’t prevent side channel attacks



host

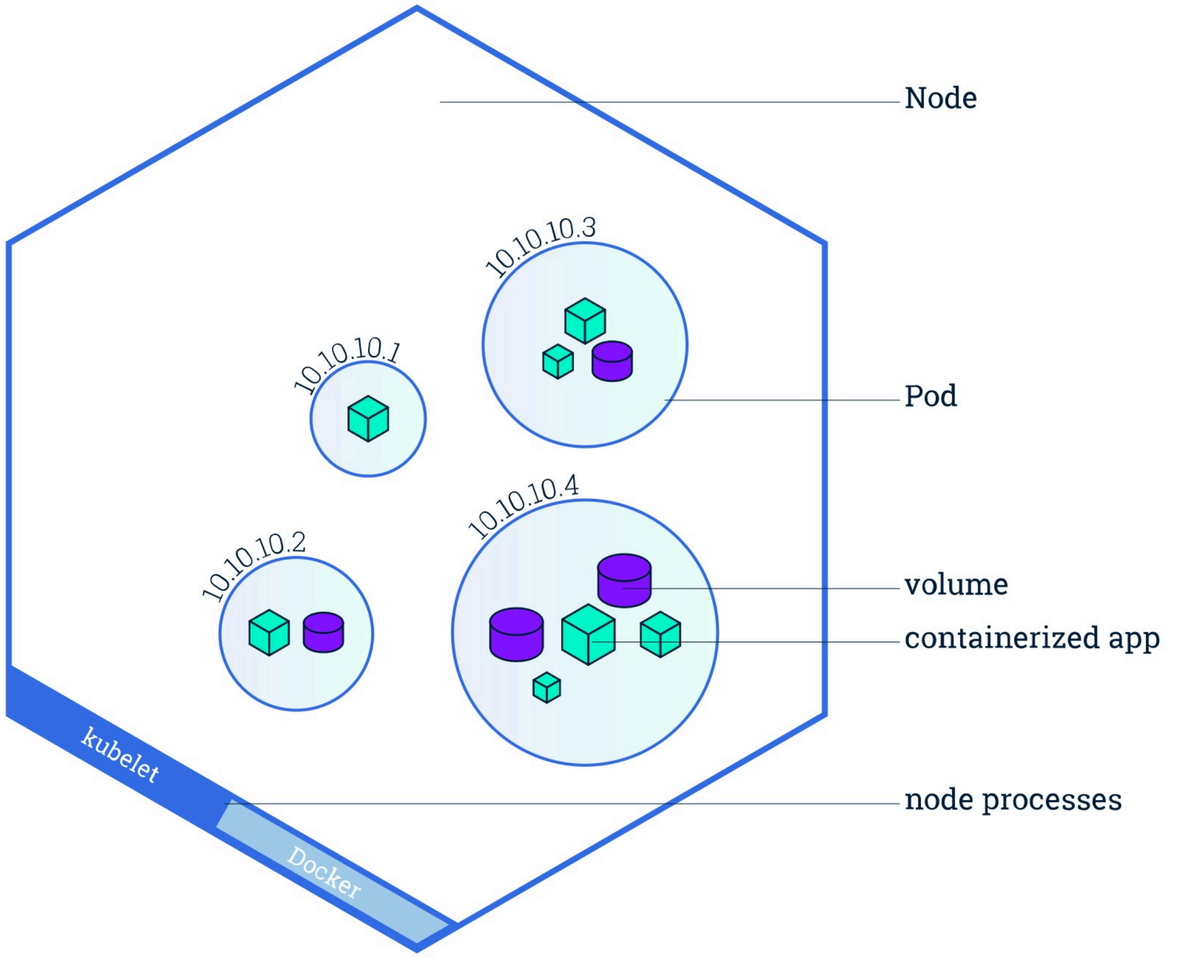


Sandbox Use Cases

- Sandbox vulnerable code
- Sandbox untrusted code
- Provide max defense in depth
- Sandbox multitenant code
- Sandbox multitenant services
- Mutually untrusted users want to share a cluster
- Sidecar container ha distinct privileges

Sandbox Layer

- Container
- A subset of containers in a pod
- Both container and pod isolation
- Pod
- A set of pots in a sandbox
- Node



- Pros and cons?

Attack Surfaces

- Kernel
- Storage
- Network
- Daemons

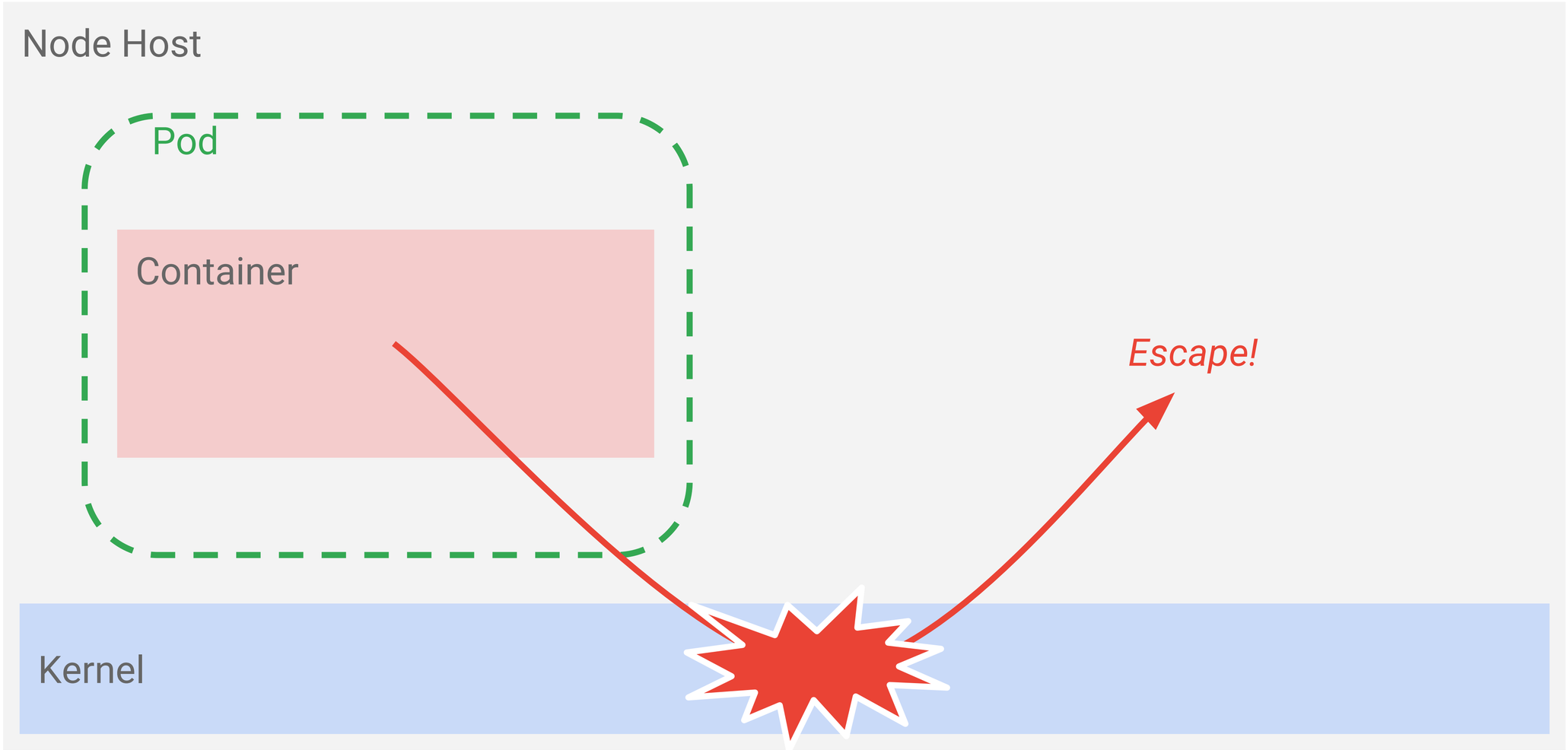
  Logging, monitoring, ...
- Hardware
- ...

Attack Surfaces

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

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

Attacks via the **Kernel**



Vulnerability Trends Over Time

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
1999	19	7		3						1		2			
2000	5	3										1			
2001	22	6								4		3			
2002	15	3		1						1	1				
2003	19	8		2						1	3	4			
2004	51	20	5	12							5	12			
2005	133	90	19	19	1					6	5	7			
2006	90	61	5	7	7			2		5	3	3			
2007	62	41	2	8						3	8	7			
2008	71	43	3	17	4					4	6	12			
2009	102	64	2	21	6					7	11	21			5
2010	123	67	3	16	7					7	30	14			5
2011	83	62	1	21	10					1	21	9			1
2012	115	83	4	25	10					6	19	11			
2013	189	101	6	41	13					11	57	26			7
2014	130	88	8	20	9					11	28	20			10
2015	86	55	6	15	4					11	10	17			
2016	217	153	5	38	18					12	35	52			1
2017	454	147	169	54	26			1		17	89	36			
2018	177	84	3	28	9					4	20	3			
2019	170	44	5	25	6					4	18	1			
Total	2333	1230	246	373	130			3		116	369	261			29
% Of All		52.7	10.5	16.0	5.6	0.0	0.0	0.1	0.0	5.0	15.8	11.2	0.0	0.0	

source: https://www.cvedetails.com/product/47/Linux-Linux-Kernel.html?vendor_id=33

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
```

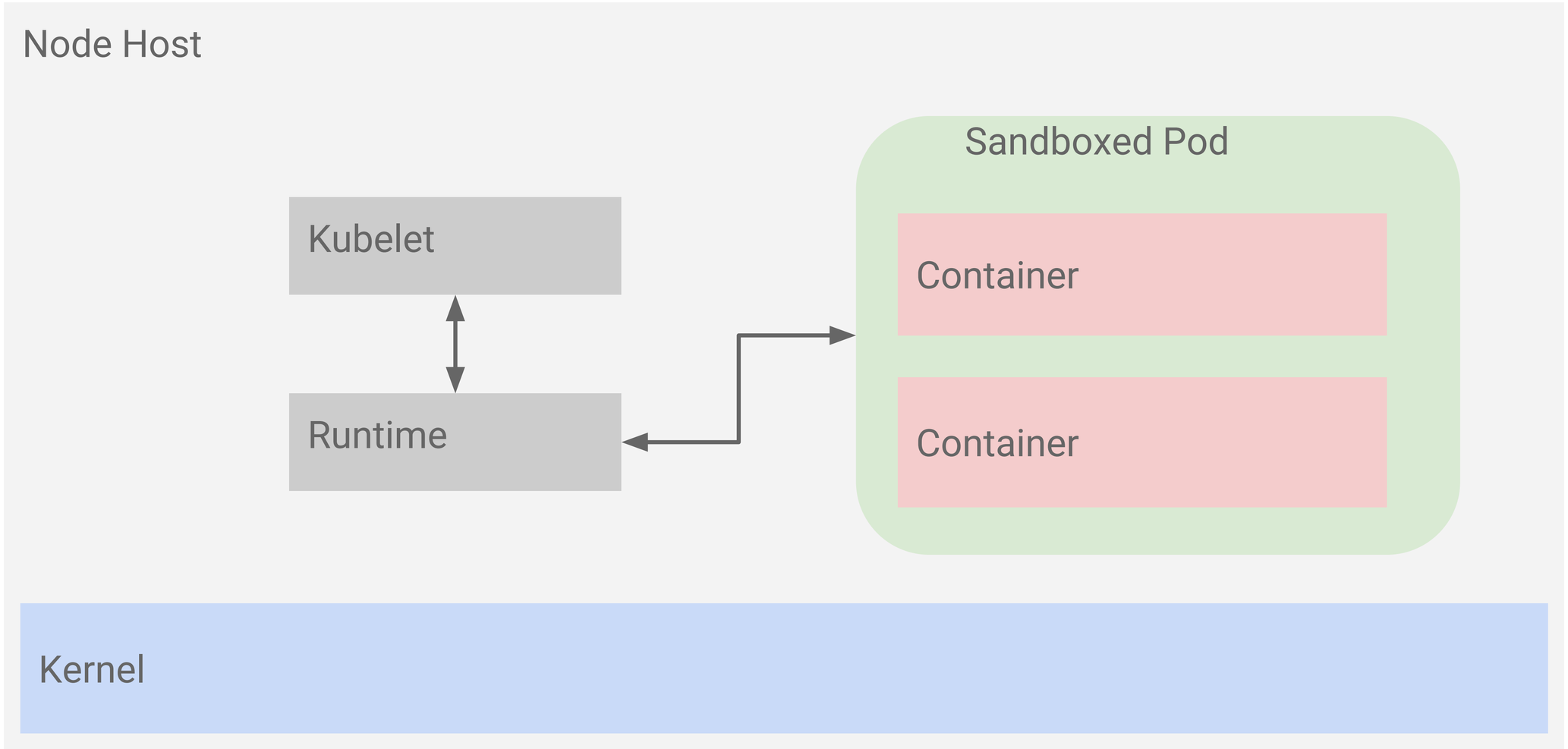
Today: kernel isolation features

```
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
```

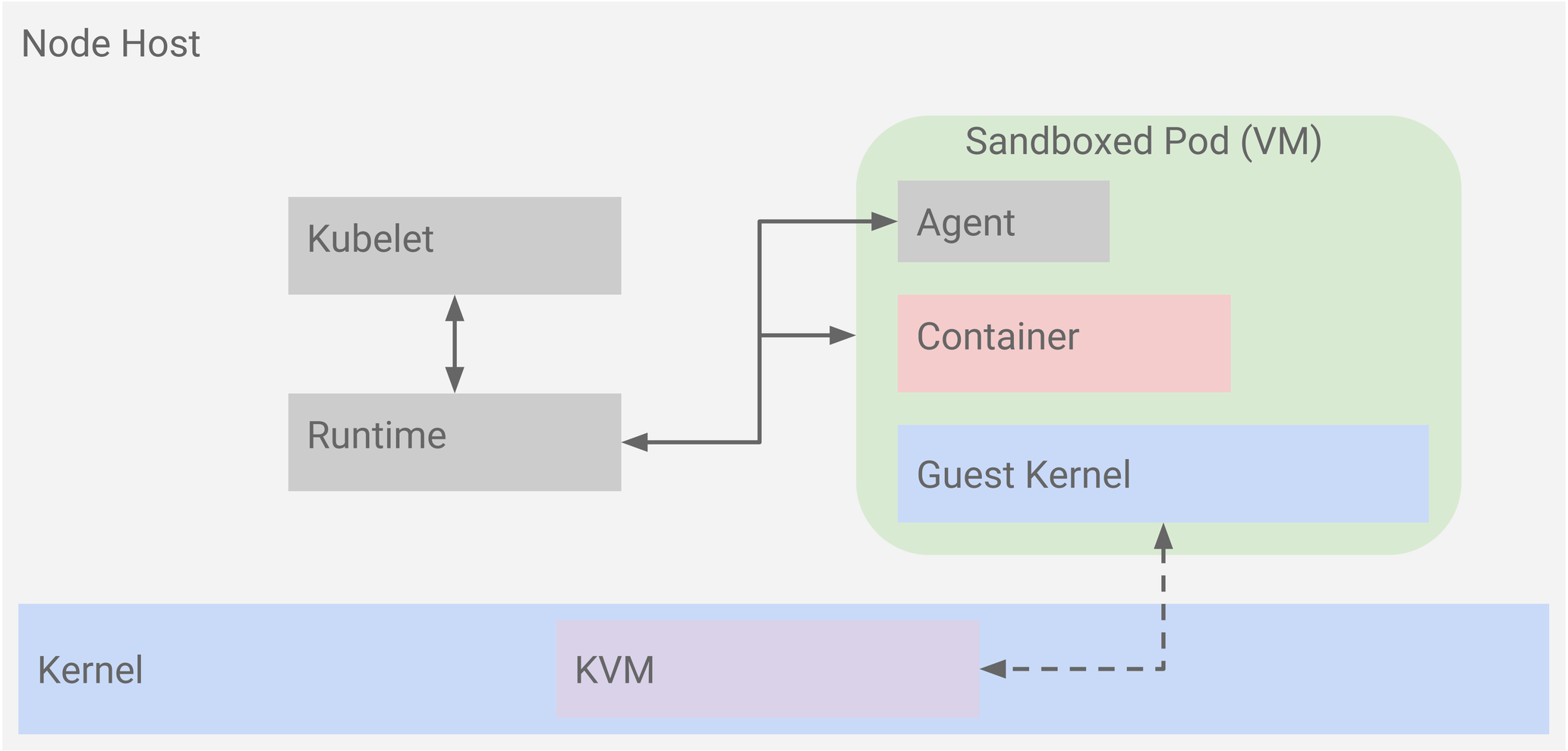
Today: kernel isolation features

```
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
```

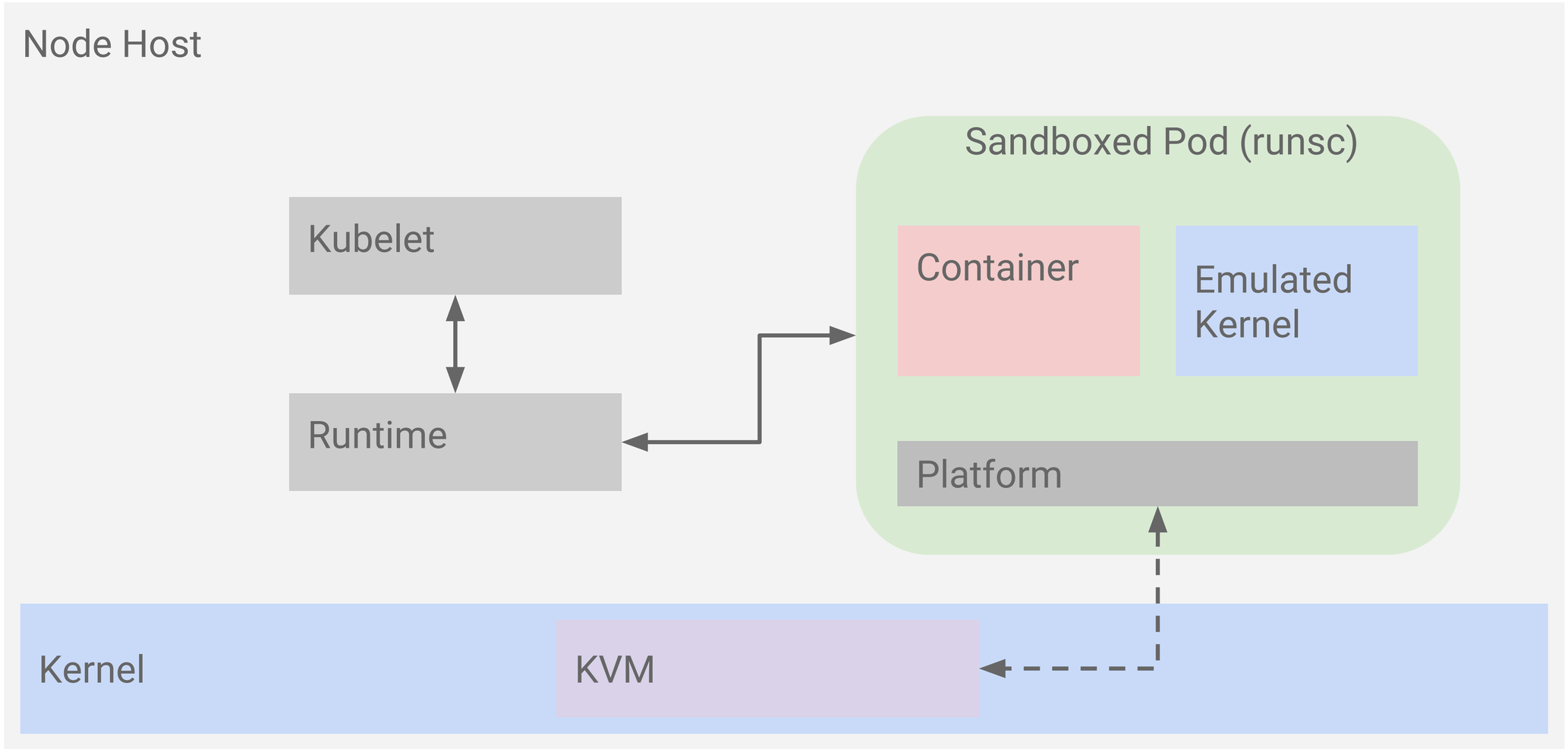
Sandboxes


Node Host
Sandboxed Pod
Kubelet
Container
Runtime
Container
Kernel

Sandboxes - kata containers



Sandboxes - gVisor

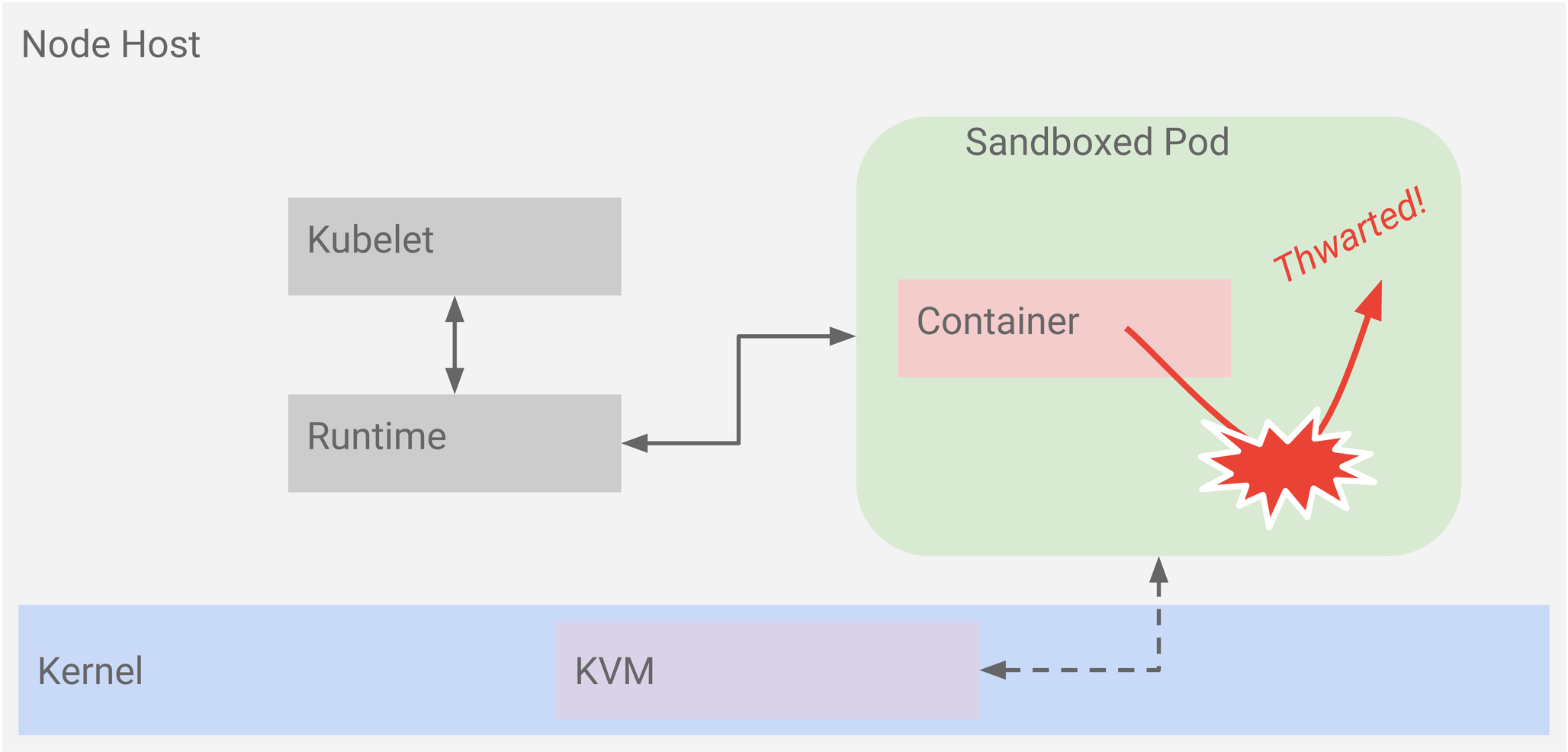


Future: sandboxed

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?

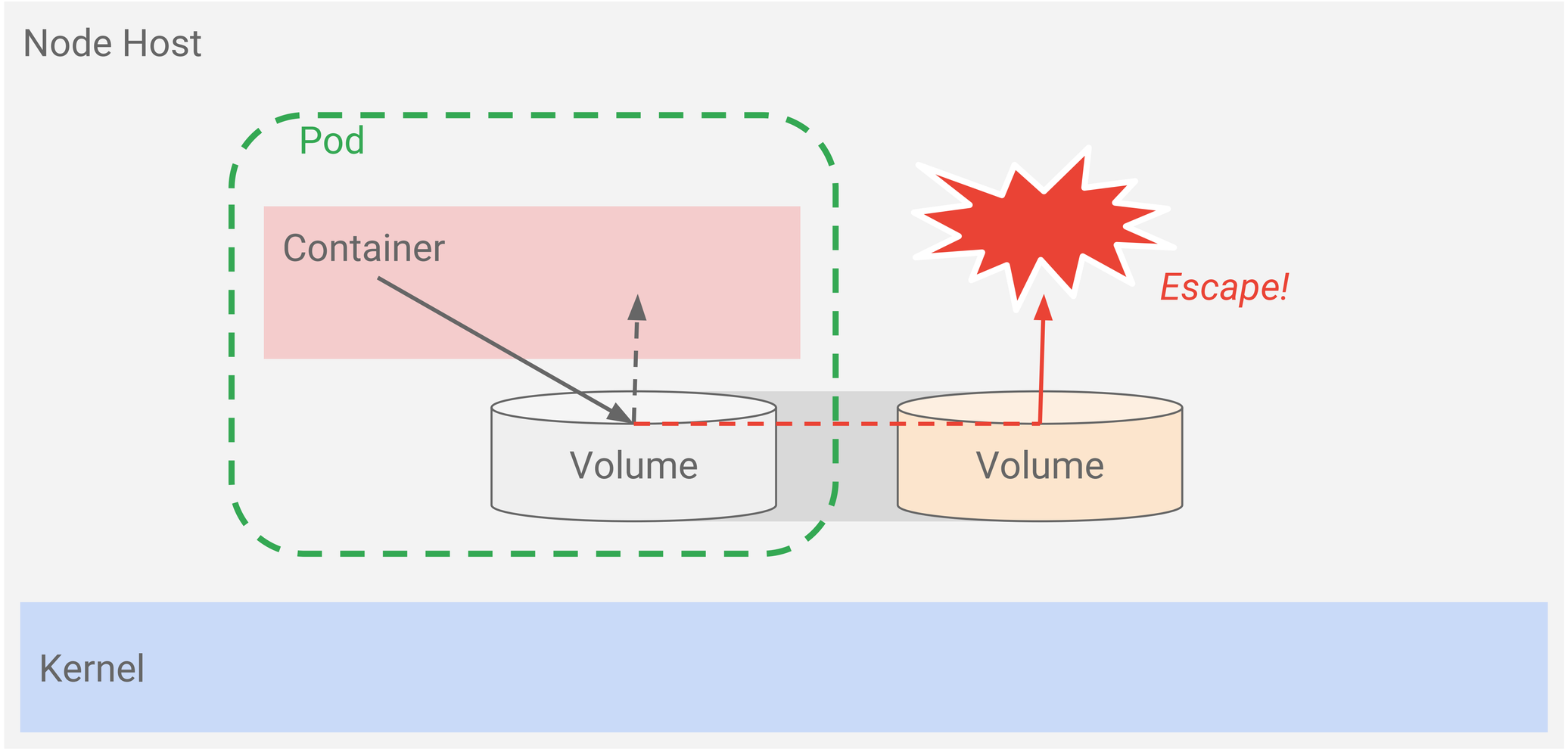
1. Cost & Performance
   But getting better!
2. Not 100% compatible
   Close though
3. Incomplete solution
   Requires network hardening

Attack Surfaces

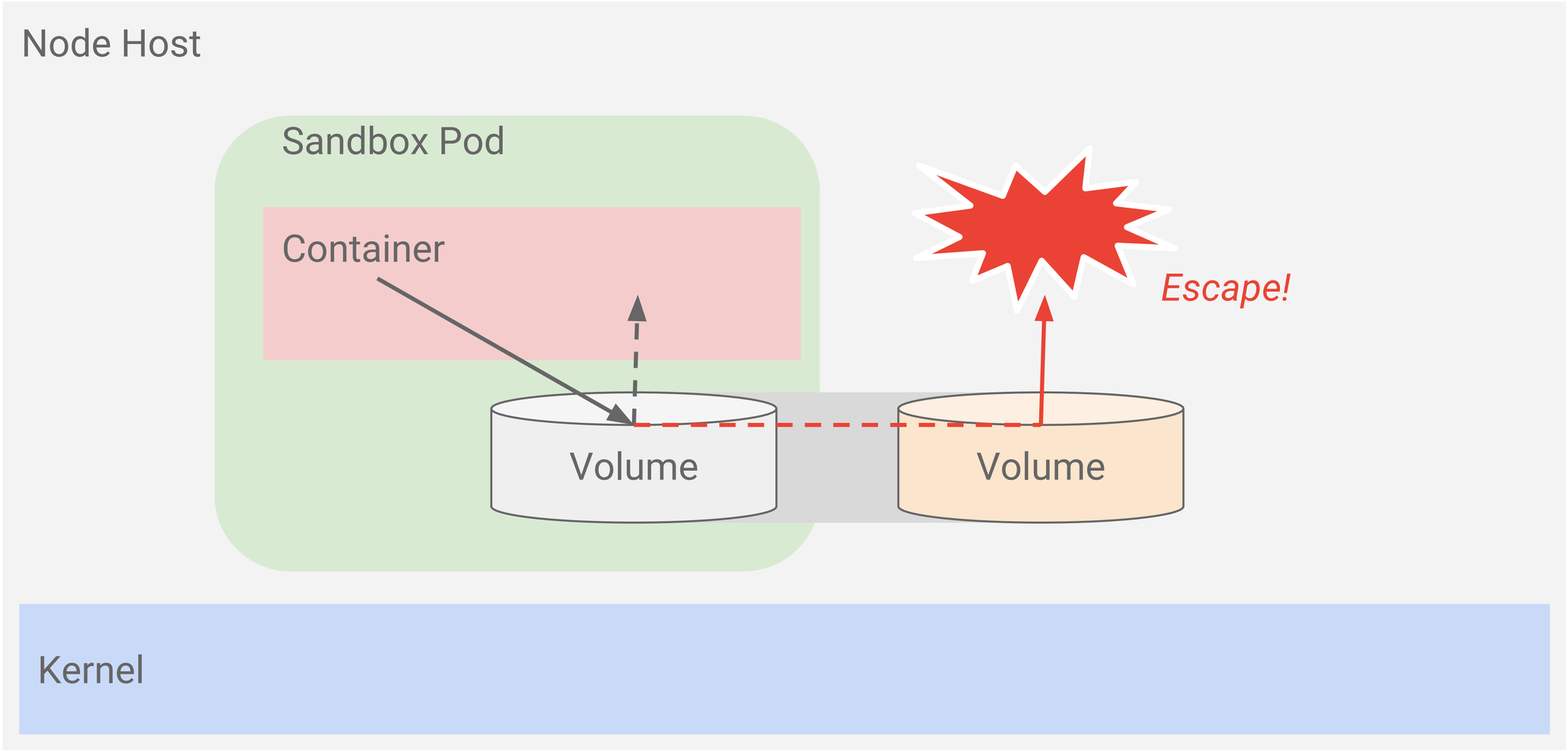
- Kernel
- **Storage**
- Network
- Daemons

  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

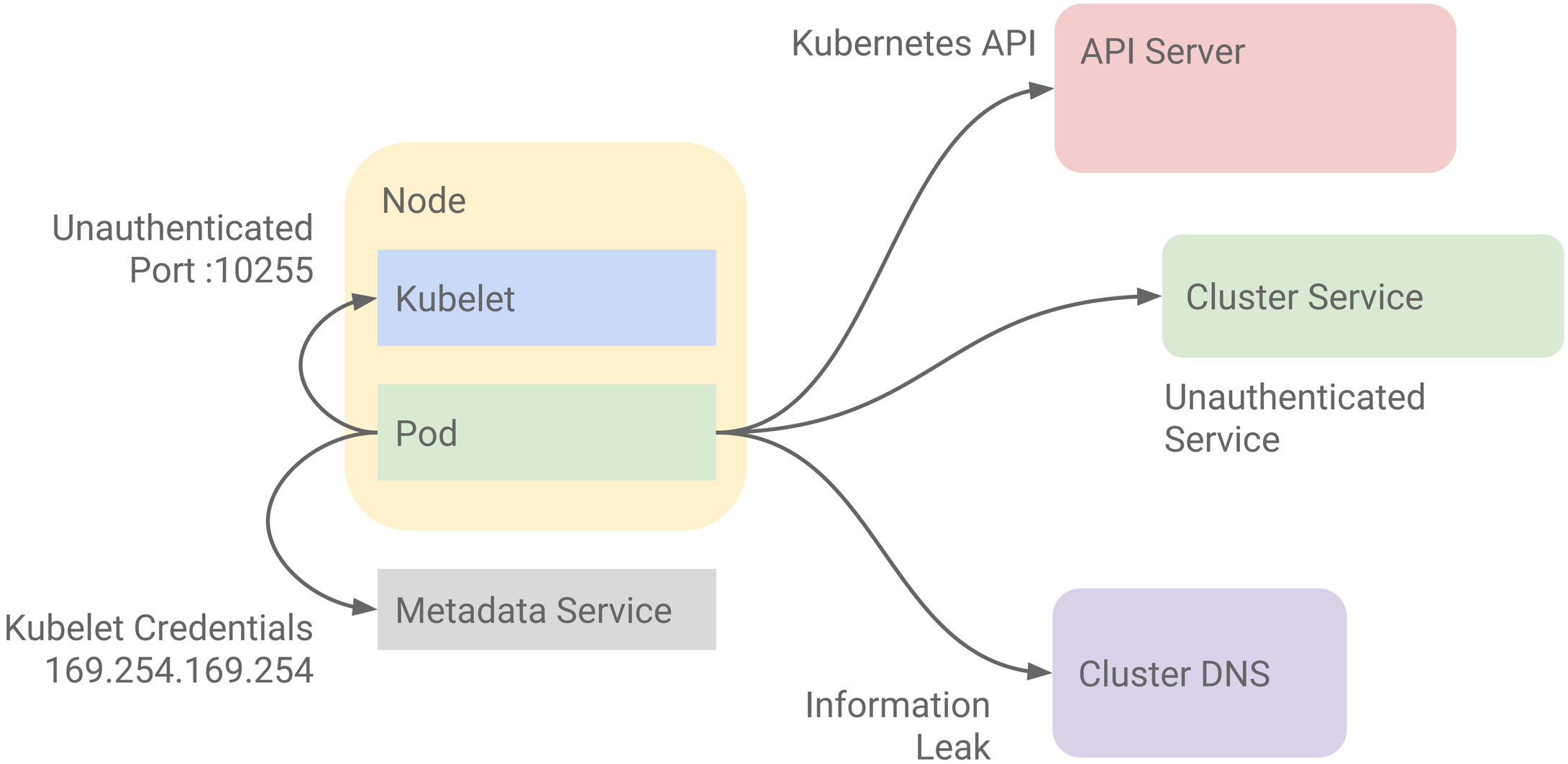
- Kernel
- Storage
- **Network**
- Daemons

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

Image credit: Albarubescens

Attacks over the **network**

Possible solution: enforce iptables rules at host



Network Policy

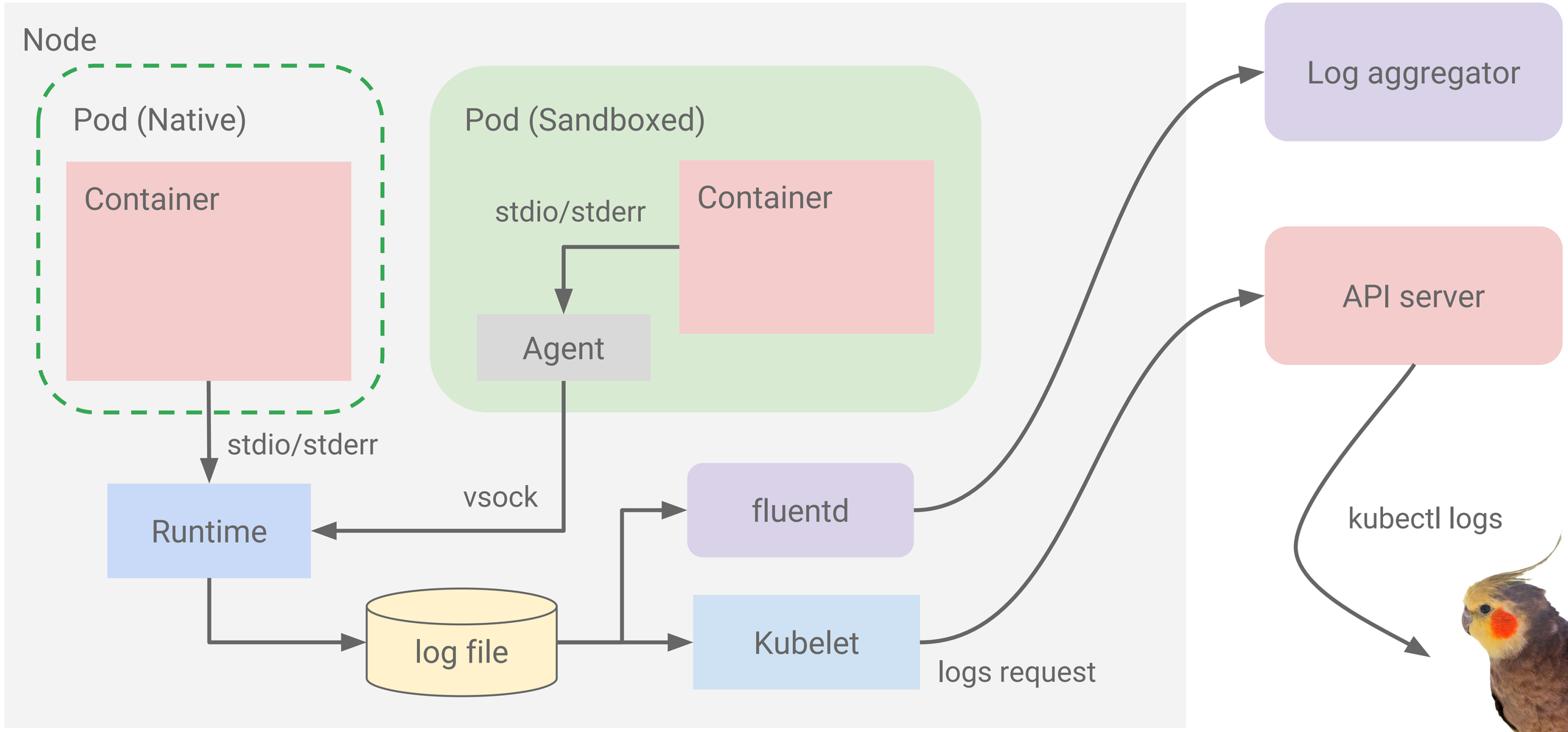
```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata: { ... }
spec:
  podSelector:
    matchLabels:
      sandboxed: true
  policyTypes:
  - Egress
  egress:
  - to:
    - ipBlock:
        cidr: 0.0.0.0/0
        except:
        - 10.0.0.0/8
        - 172.16.0.0/12
        - 192.168.0.0/16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

Attack Surfaces

- Kernel
- Storage
- Network
- Daemons

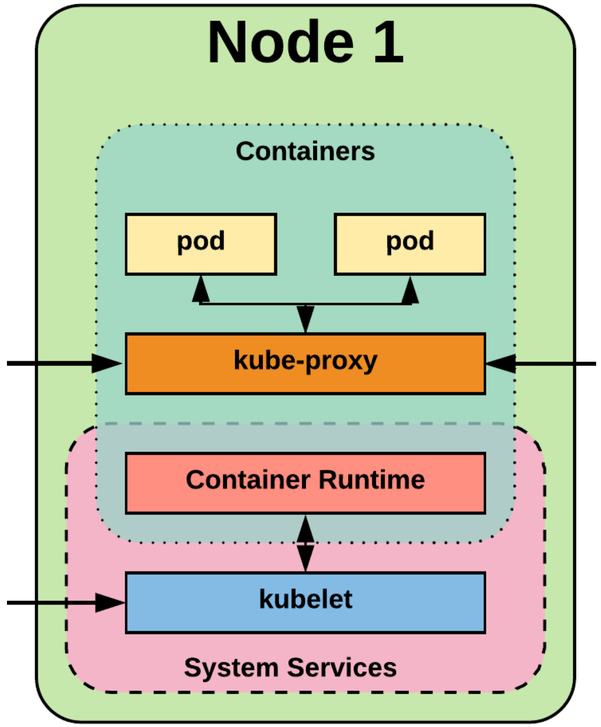
  ***Logging**, monitoring, ...*
- Hardware
- ...

Attacks via system **logs**



Threats Related to Other Daemons

- Kubelet
  - CVE-2018-1002100: Kubectl copy doesn’t check for paths outside of its destination directory
- Container runtime
  - Spoofed container status, fake container stats
- Processes not specified by the user that run in sandbox
- Kube-proxy



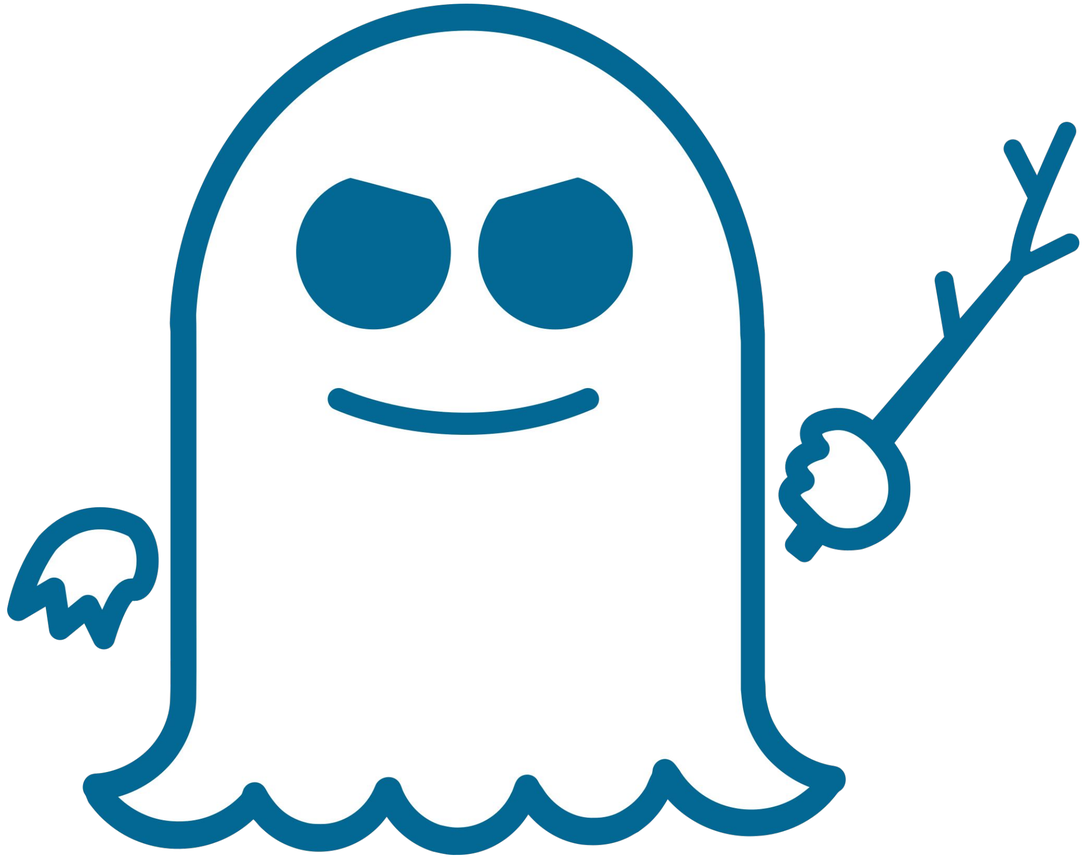
Attack Surfaces

- Kernel
- Storage
- Network
- Daemons

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

Image credit: Jonas Löwgren

Attacks via the **Hardware**



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

Image credit: William Warby