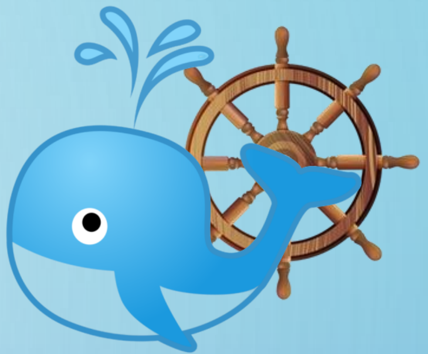
DEFENDING CONTAINERS LIKE A NINJA:
A WALK THROUGH THE ADVANCED SECURITY FEATURES OF DOCKER AND KUBERNETES

Sheila A. Berta (@UnaPibaGeek)
Dreamlab Technologies



DREAMLAB TECHNOLOGIES

black hat USA 2020

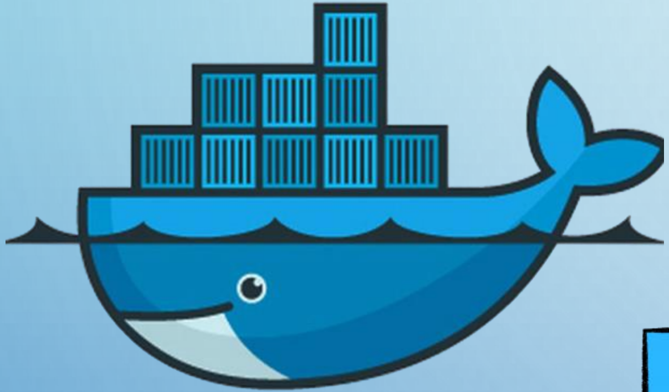
WHO AM I?

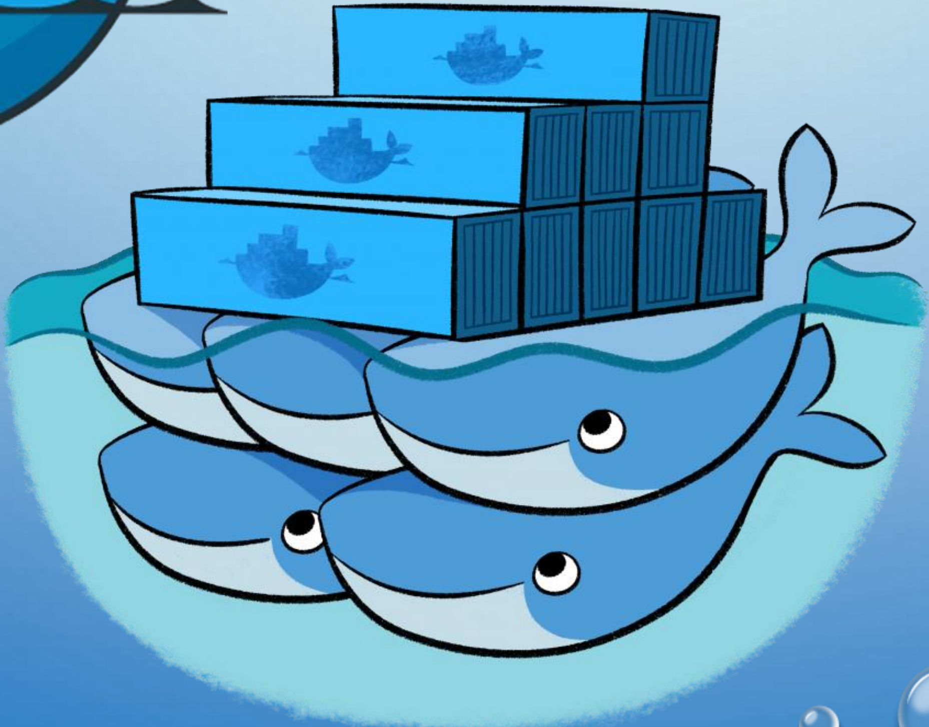
Sheila A. Berta - @UnaPibaGeek
Head of Research at Dreamlab Technologies

A little bit more:

- Developer in ASM (Microcontrollers & Microprocessors x86/x64), C/C++, Go & Python.
- Speaker at Black Hat (x3), DEF CON (x2), Ekoparty (x4), HITB, SCSD, IEEE… & more.
- Docker Captain! (DCA Certified)

COMPLEXITY IS THE WORST ENEMY OF SECURITY...

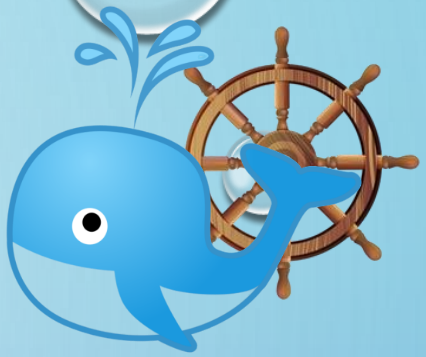




DREAMLAB TECHNOLOGIES

black hat USA 2020

SECURING DOCKER DAEMON AND CORE COMPONENTS



DAEMON ROOTLESS MODE

Prerequisites: https://docs.docker.com/engine/security/rootless/#distribution-specific-hint

Step 1:

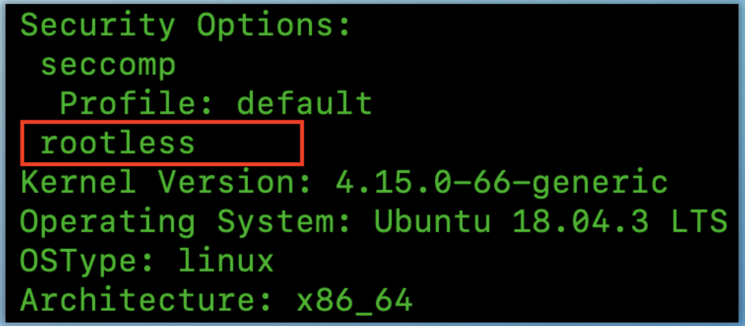
```
$ curl -fsSL https://get.docker.com/rootless | sh
```

Step 2:

```
export XDG_RUNTIME_DIR=/tmp/docker-1000
export PATH=/home/<user>/bin:$PATH
export DOCKER_HOST=unix:///tmp/docker-1000/docker.sock
```

Step 3:

```
$ /home/<user>/bin/dockerd-rootless.sh --experimental --storage-driver vfs &
```

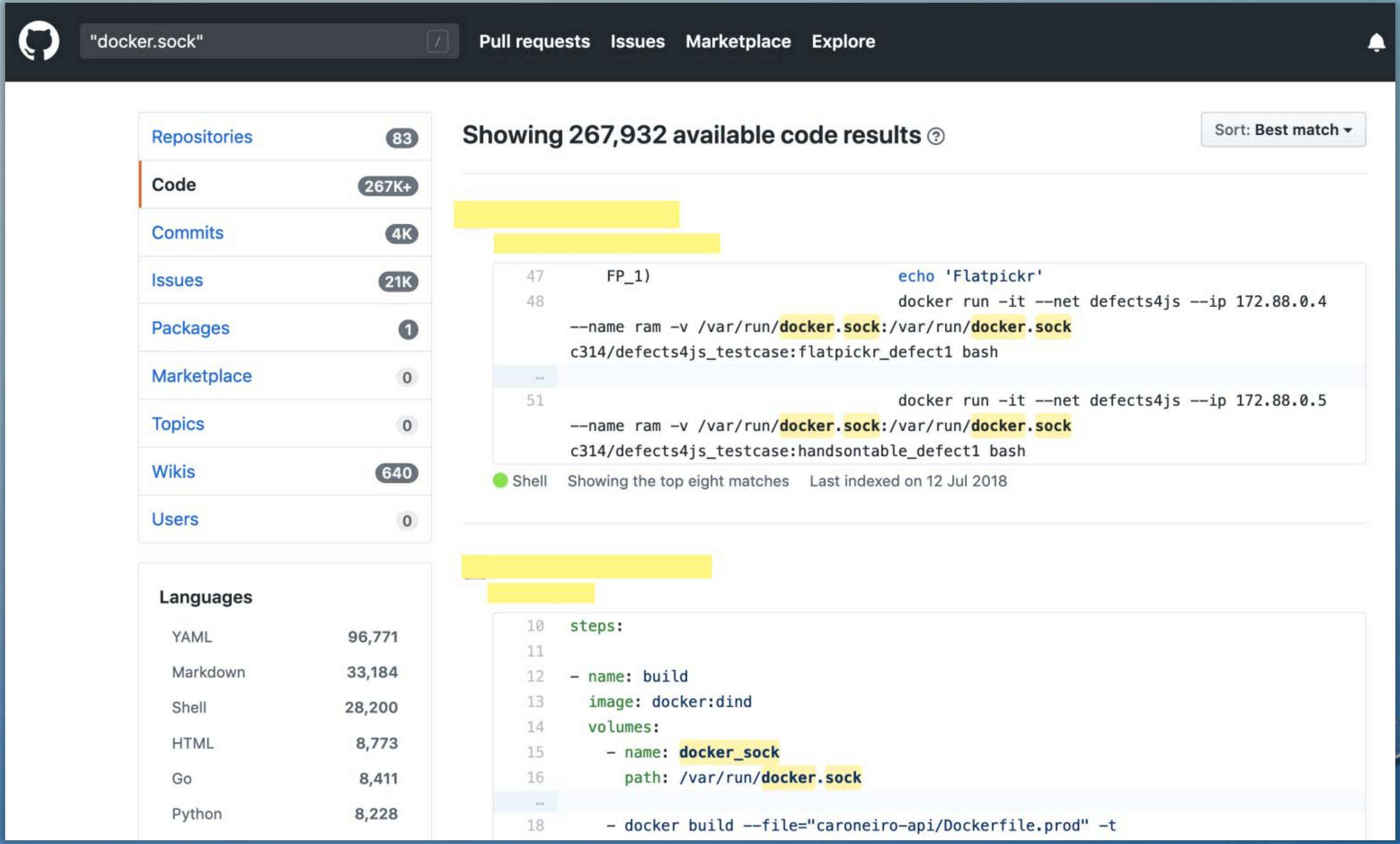


```
Security Options:
 seccomp
  Profile: default
 rootless
Kernel Version: 4.15.0-66-generic
Operating System: Ubuntu 18.04.3 LTS
OSType: linux
Architecture: x86_64
```

DOCKER SOCKET

- ❑ UNIX
- ❑ TCP
- ❑ FD

UNIX socket
/var/run/docker.sock



"docker.sock" Pull requests Issues Marketplace Explore

Repositories	83
Code	267K+
Commits	4K
Issues	21K
Packages	1
Marketplace	0
Topics	0
Wikis	640
Users	0

Languages

YAML	96,771
Markdown	33,184
Shell	28,200
HTML	8,773
Go	8,411
Python	8,228

Showing 267,932 available code results

Sort: Best match

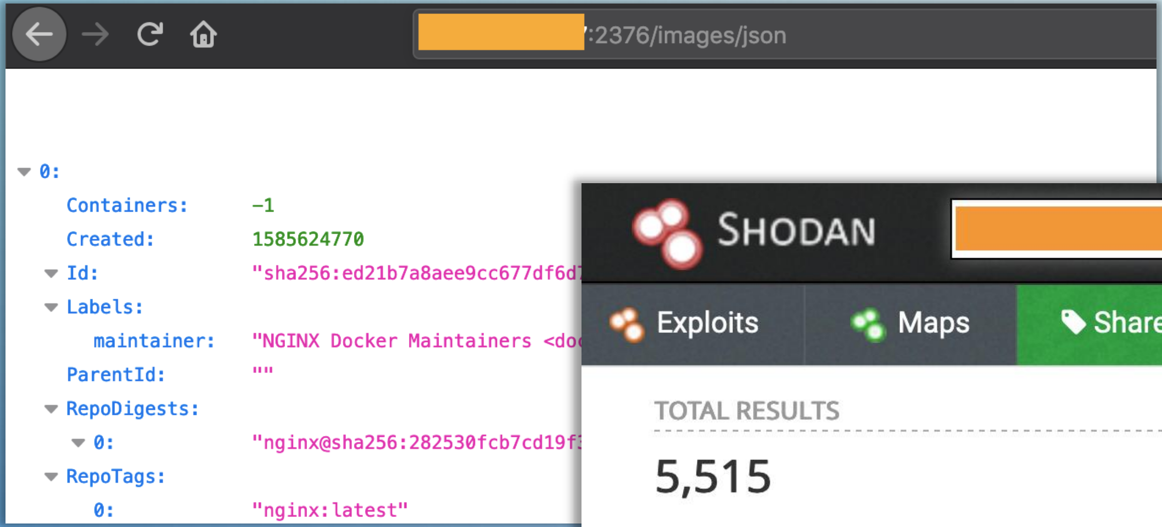
```
    FP_1)                           echo 'Flatpickr'
                                    docker run -it --net defects4js --ip 172.88.0.4
--name ram -v /var/run/docker.sock:/var/run/docker.sock
c314/defects4js_testcase:flatpickr_defect1 bash
...
                                    docker run -it --net defects4js --ip 172.88.0.5
--name ram -v /var/run/docker.sock:/var/run/docker.sock
c314/defects4js_testcase:handsontable_defect1 bash
```

● Shell Showing the top eight matches Last indexed on 12 Jul 2018

```
steps:

- name: build
  image: docker:dind
  volumes:
    - name: docker_sock
      path: /var/run/docker.sock
...
    - docker build --file="caroneiro-api/Dockerfile.prod" -t
```

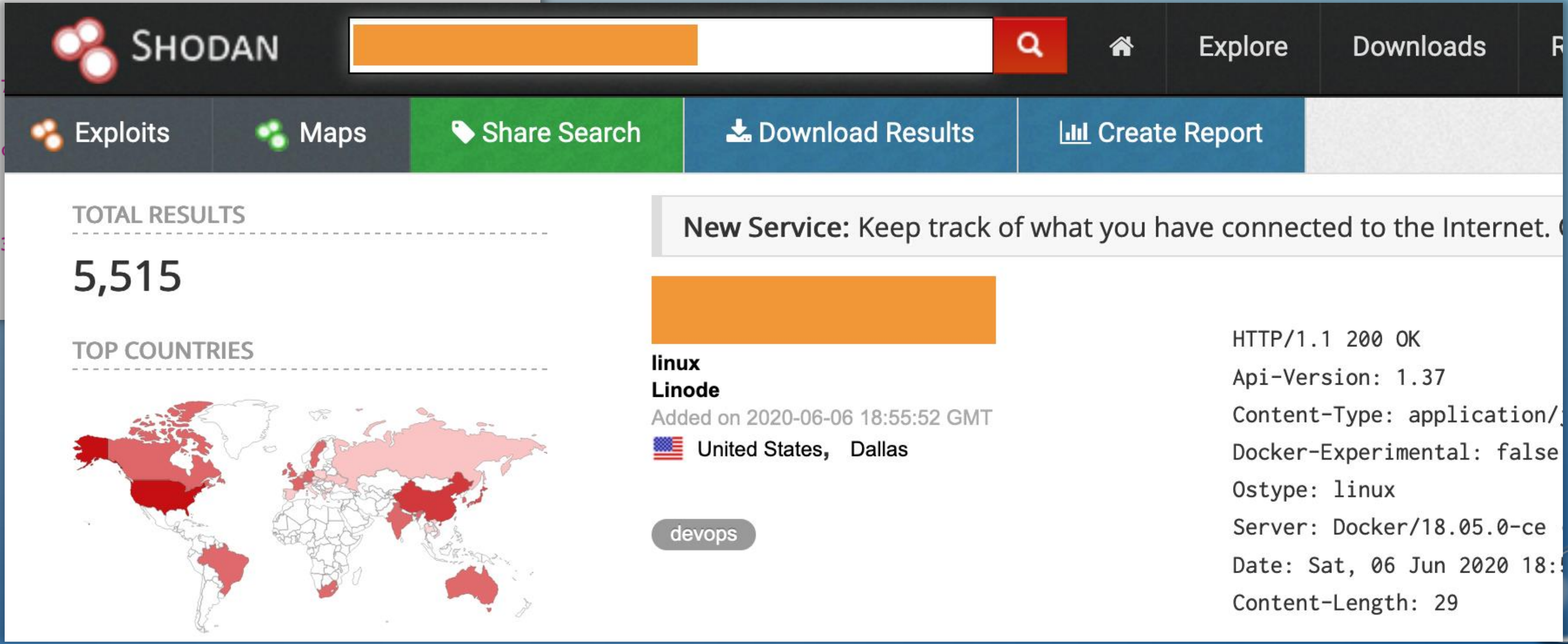
DOCKER SOCKET

TCP socket



[redacted]:2376/images/json

```
▼ 0:
    Containers:      -1
    Created:         1585624770
  ▼ Id:              "sha256:ed21b7a8aee9cc677df6d
  ▼ Labels:
      maintainer:    "NGINX Docker Maintainers <do
    ParentId:        ""
  ▼ RepoDigests:
    ▼ 0:             "nginx@sha256:282530fcb7cd19f
  ▼ RepoTags:
      0:             "nginx:latest"
```



[redacted]

TOTAL RESULTS

5,515

TOP COUNTRIES

New Service: Keep track of what you have connected to the Internet.

[redacted]

linux
Linode
Added on 2020-06-06 18:55:52 GMT
United States, Dallas

devops

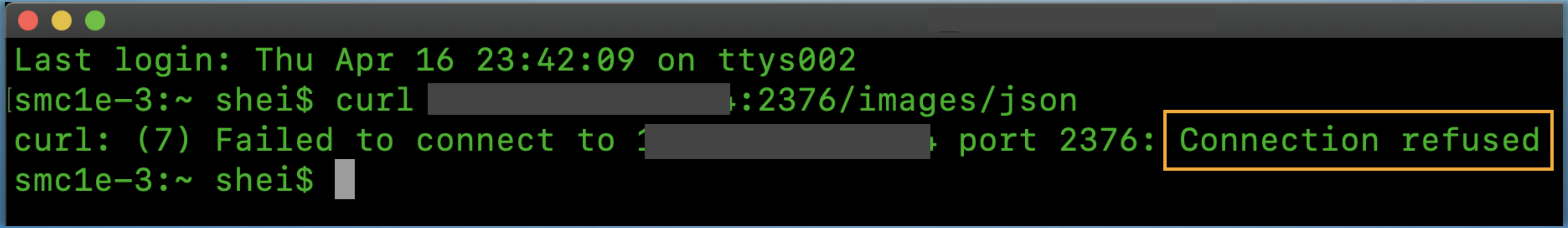
```
HTTP/1.1 200 OK
Api-Version: 1.37
Content-Type: application/
Docker-Experimental: false
Ostype: linux
Server: Docker/18.05.0-ce
Date: Sat, 06 Jun 2020 18:
Content-Length: 29
```

DOCKER SOCKET

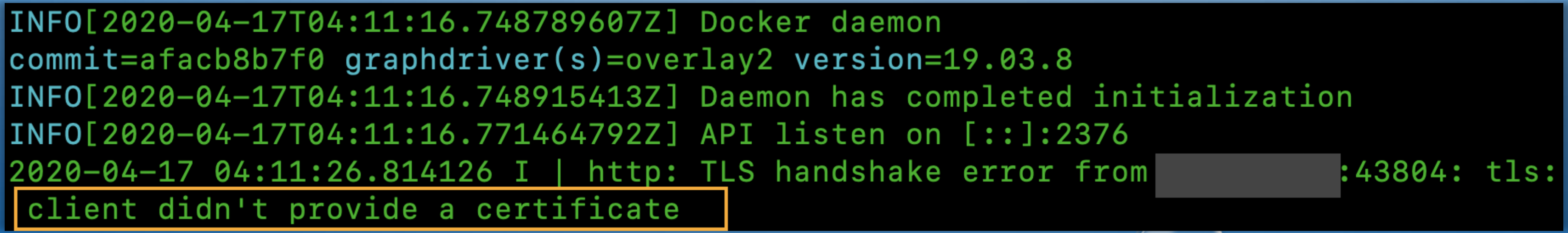
TCP socket – Built-in HTTPS encrypted socket

- ❑ Create a CA and server keys using OpenSSL
- ❑ Run the Docker daemon with the TLS certificates.

```
$ dockerd --tlsverify --tlscacert=ca.pem --tlscert=server-cert.pem --tlskey=server-key.pem -H=0.0.0.0:2376
```



```
Last login: Thu Apr 16 23:42:09 on ttys002
smc1e-3:~ shei$ curl [redacted]:2376/images/json
curl: (7) Failed to connect to [redacted] port 2376: Connection refused
smc1e-3:~ shei$
```



```
INFO[2020-04-17T04:11:16.748789607Z] Docker daemon
commit=afacb8b7f0 graphdriver(s)=overlay2 version=19.03.8
INFO[2020-04-17T04:11:16.748915413Z] Daemon has completed initialization
INFO[2020-04-17T04:11:16.771464792Z] API listen on [::]:2376
2020-04-17 04:11:26.814126 I | http: TLS handshake error from [redacted]:43804: tls:
client didn't provide a certificate
```

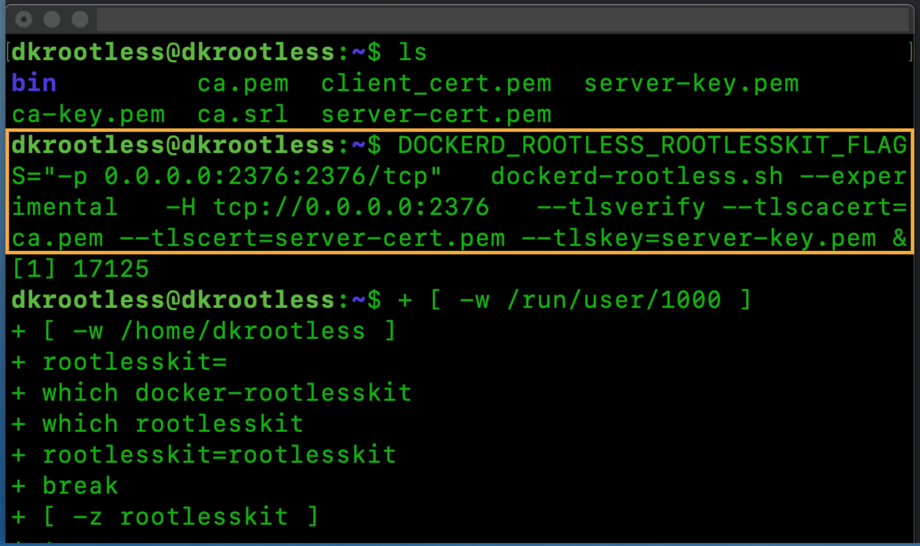
API ENDPOINT – CLIENT AUTHENTICATION

- ❑ Create the client TLS certificates and sign them with the CA.
- ❑ Connect to the remote API endpoint providing the certificates.

$ docker --tlsverify --tlscacert=ca.pem --tlscert=client-cert.pem --tlskey=client-key.pem -H=<host>:2376 version

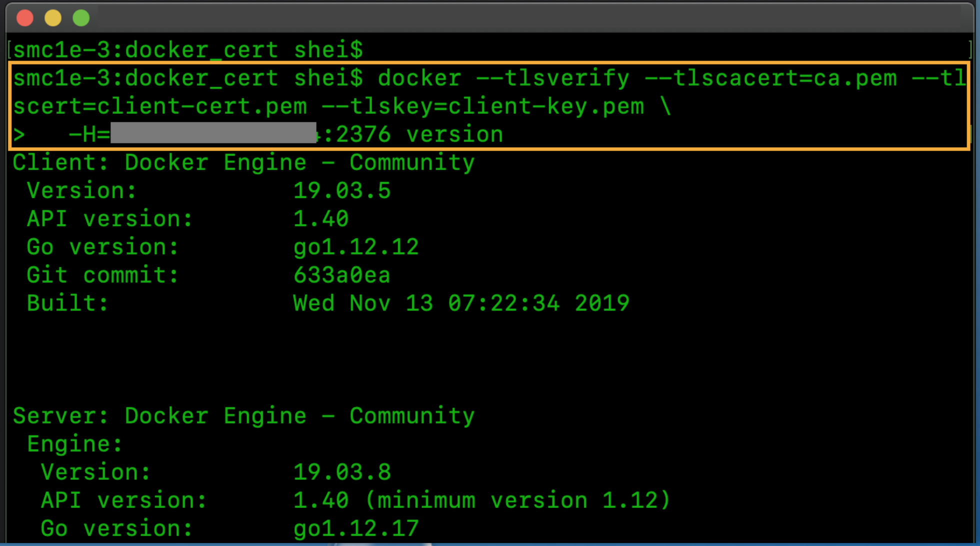
$ export DOCKER_HOST=tcp://<host>:2376 DOCKER_TLS_VERIFY=1

DOCKER DAEMON'S HOST



```
dkrootless@dkrootless:~$ ls
bin            ca.pem  client_cert.pem  server-key.pem
ca-key.pem   ca.srl  server-cert.pem
dkrootless@dkrootless:~$ DOCKERD_ROOTLESS_ROOTLESSKIT_FLAG
S="-p 0.0.0.0:2376:2376/tcp"   dockerd-rootless.sh --exper
imental   -H tcp://0.0.0.0:2376   --tlsverify --tlscacert=
ca.pem --tlscert=server-cert.pem --tlskey=server-key.pem &
[1] 17125
dkrootless@dkrootless:~$ + [ -w /run/user/1000 ]
+ [ -w /home/dkrootless ]
+ rootlesskit=
+ which docker-rootlesskit
+ which rootlesskit
+ rootlesskit=rootlesskit
+ break
+ [ -z rootlesskit ]
```

CLIENT'S MACHINE

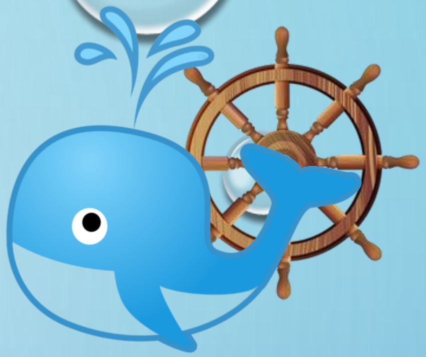


```
smc1e-3:docker_cert shei$
smc1e-3:docker_cert shei$ docker --tlsverify --tlscacert=ca.pem --tl
scert=client-cert.pem --tlskey=client-key.pem \
>    -H=[redacted]:2376 version
Client: Docker Engine - Community
 Version:           19.03.5
 API version:       1.40
 Go version:        go1.12.12
 Git commit:        633a0ea
 Built:             Wed Nov 13 07:22:34 2019


Server: Docker Engine - Community
 Engine:
  Version:          19.03.8
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.17
```

DREAMLAB TECHNOLOGIES

black hat USA 2020

SECURING DOCKER CONTAINERS



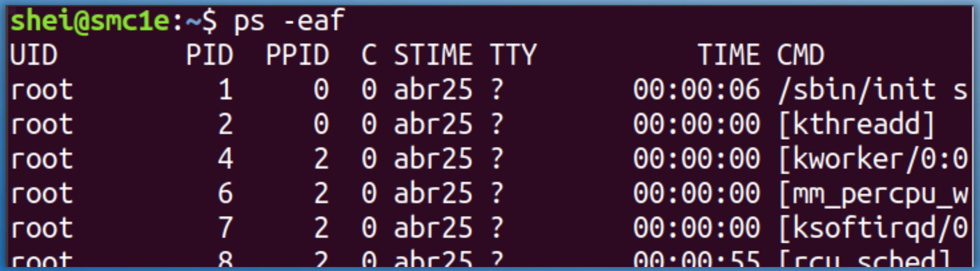
DREAMLAB TECHNOLOGIES

black hat USA 2020

KERNEL NAMESPACES

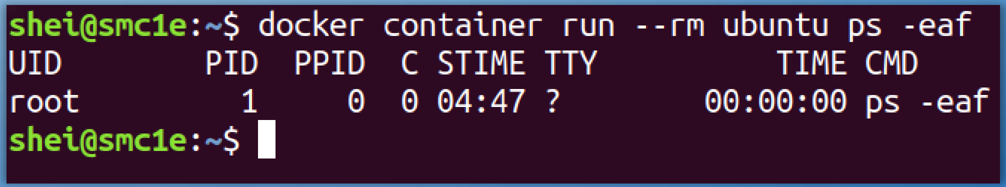
- ❑ UTS namespace: isolates system identifiers.
- ❑ PID namespace: isolates the PID space.
- ❑ IPC namespace: isolates IPC resources.
- ❑ NET namespace: isolates network interfaces.
- ❑ USER namespace: isolates user and group ID spaces (disabled by default).
- ❑ Mount namespace: isolates the set of filesystem mount points.

Docker Host



```
shei@smc1e:~$ ps -eaf
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 abr25 ?        00:00:06 /sbin/init s
root         2     0  0 abr25 ?        00:00:00 [kthreadd]
root         4     2  0 abr25 ?        00:00:00 [kworker/0:0
root         6     2  0 abr25 ?        00:00:00 [mm_percpu_w
root         7     2  0 abr25 ?        00:00:00 [ksoftirqd/0
root         8     2  0 abr25 ?        00:00:55 [rcu_sched]
```

Docker Container



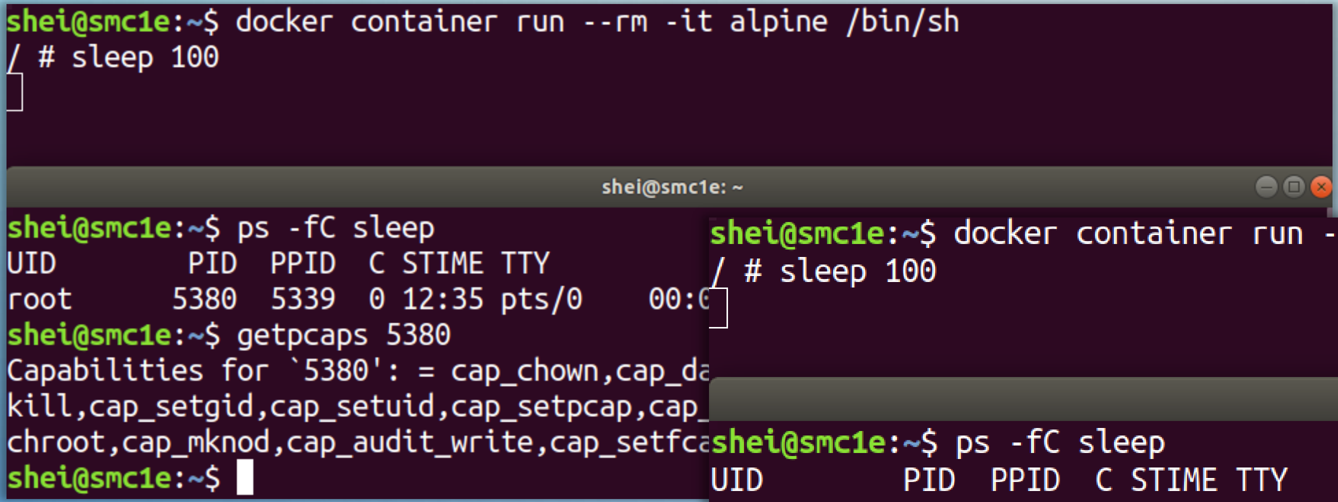
```
shei@smc1e:~$ docker container run --rm ubuntu ps -eaf
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 04:47 ?        00:00:00 ps -eaf
shei@smc1e:~$ 
```

DREAMLAB TECHNOLOGIES

black hat USA 2020

KERNEL CAPABILITIES

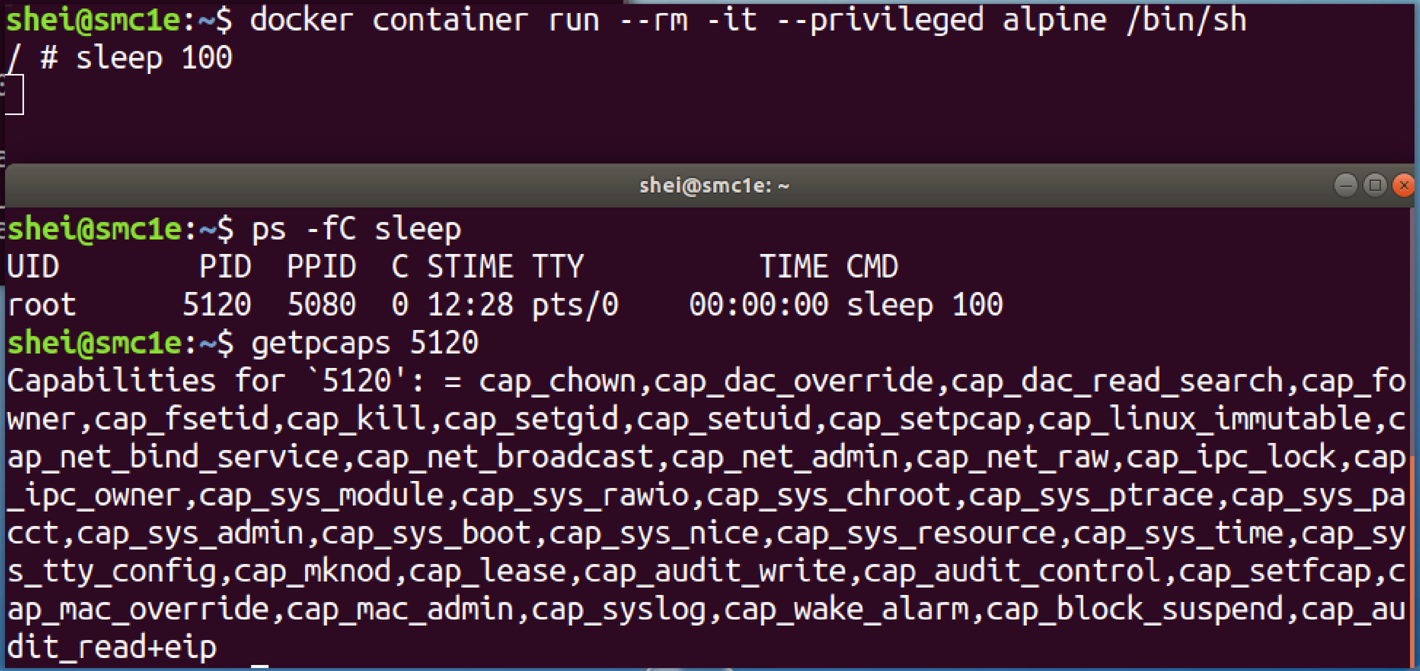
Default Capabilities: https://github.com/moby/moby/blob/master/oci/caps/defaults.go



```
shei@smc1e:~$ docker container run --rm -it alpine /bin/sh
/ # sleep 100
```

shei@smc1e: ~

```
shei@smc1e:~$ ps -fC sleep
UID         PID  PPID  C STIME TTY          00:0
root       5380  5339  0 12:35 pts/0
shei@smc1e:~$ getpcaps 5380
Capabilities for `5380': = cap_chown,cap_da
kill,cap_setgid,cap_setuid,cap_setpcap,cap_
chroot,cap_mknod,cap_audit_write,cap_setfca
shei@smc1e:~$
```



```
shei@smc1e:~$ docker container run --rm -it --privileged alpine /bin/sh
/ # sleep 100
```

shei@smc1e: ~

```
shei@smc1e:~$ ps -fC sleep
UID         PID  PPID  C STIME TTY          TIME CMD
root       5120  5080  0 12:28 pts/0    00:00:00 sleep 100
shei@smc1e:~$ getpcaps 5120
Capabilities for `5120': = cap_chown,cap_dac_override,cap_dac_read_search,cap_fo
wner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_linux_immutable,c
ap_net_bind_service,cap_net_broadcast,cap_net_admin,cap_net_raw,cap_ipc_lock,cap
_ipc_owner,cap_sys_module,cap_sys_rawio,cap_sys_chroot,cap_sys_ptrace,cap_sys_pa
cct,cap_sys_admin,cap_sys_boot,cap_sys_nice,cap_sys_resource,cap_sys_time,cap_sy
s_tty_config,cap_mknod,cap_lease,cap_audit_write,cap_audit_control,cap_setfcap,c
ap_mac_override,cap_mac_admin,cap_syslog,cap_wake_alarm,cap_block_suspend,cap_au
dit_read+eip
```

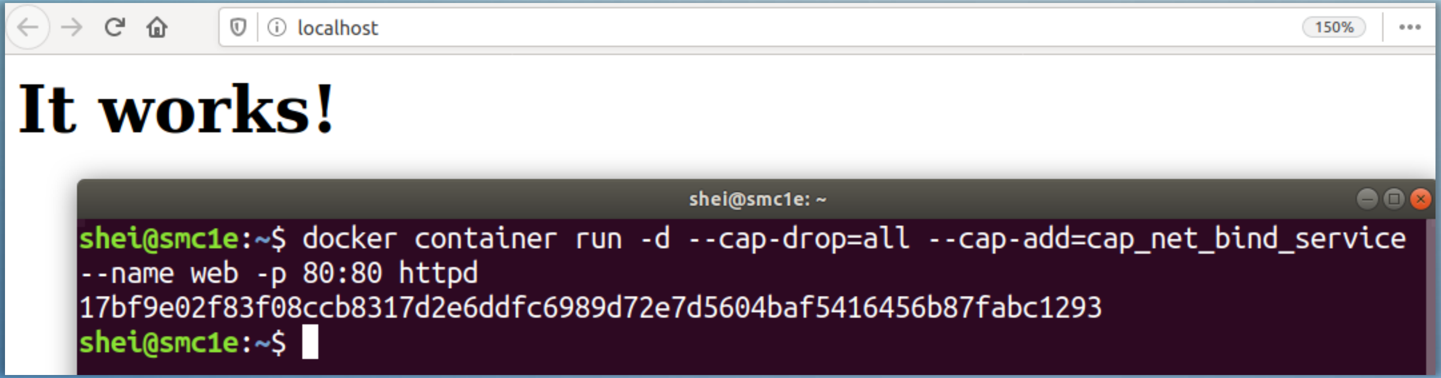
DREAMLAB TECHNOLOGIES

black hat USA 2020

KERNEL CAPABILITIES

Fine-grained access control system

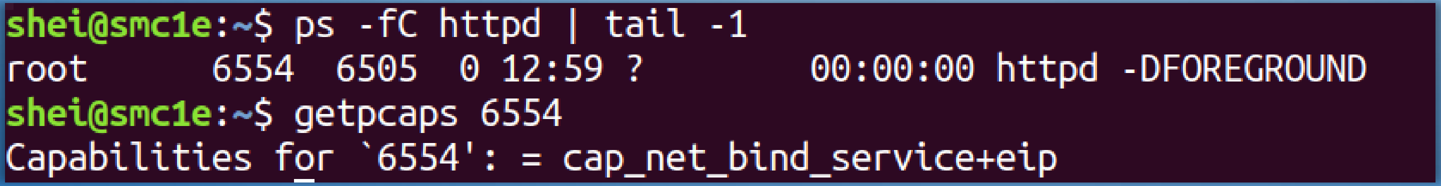
```
$ docker container run --cap-drop=all --cap-add=cap_net_bind_service -p 80:80 httpd
```



localhost

It works!

```
shei@smc1e:~$ docker container run -d --cap-drop=all --cap-add=cap_net_bind_service
--name web -p 80:80 httpd
17bf9e02f83f08ccb8317d2e6ddfc6989d72e7d5604baf5416456b87fabc1293
shei@smc1e:~$
```



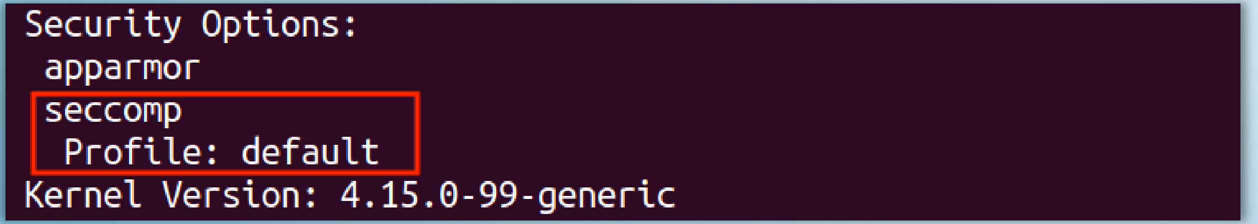
```
shei@smc1e:~$ ps -fC httpd | tail -1
root      6554  6505  0 12:59 ?        00:00:00 httpd -DFOREGROUND
shei@smc1e:~$ getpcaps 6554
Capabilities for `6554': = cap_net_bind_service+eip
```

DREAMLAB TECHNOLOGIES

black hat USA 2020

SYSTEM CALLS RESTRICTION

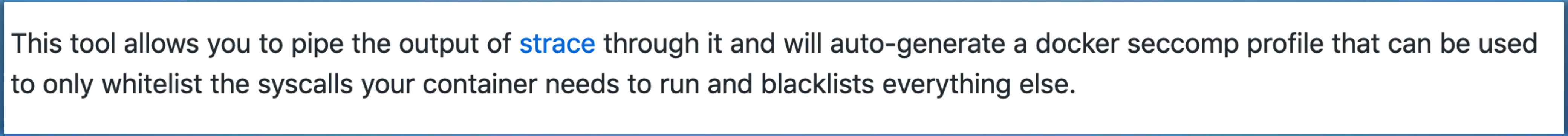
Seccomp Profiles



```
Security Options:
 apparmor
 seccomp
  Profile: default
Kernel Version: 4.15.0-99-generic
```

$ docker container run --security-opt seccomp=/path/to/seccomp/profile.json myapp

https://github.com/blacktop/seccomp-gen



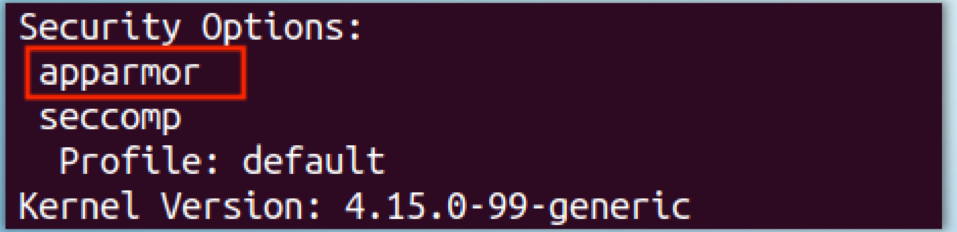
This tool allows you to pipe the output of strace through it and will auto-generate a docker seccomp profile that can be used to only whitelist the syscalls your container needs to run and blacklists everything else.

DREAMLAB TECHNOLOGIES

black hat USA 2020

MANDATORY ACCESS CONTROL

AppArmor / SELinux Profiles

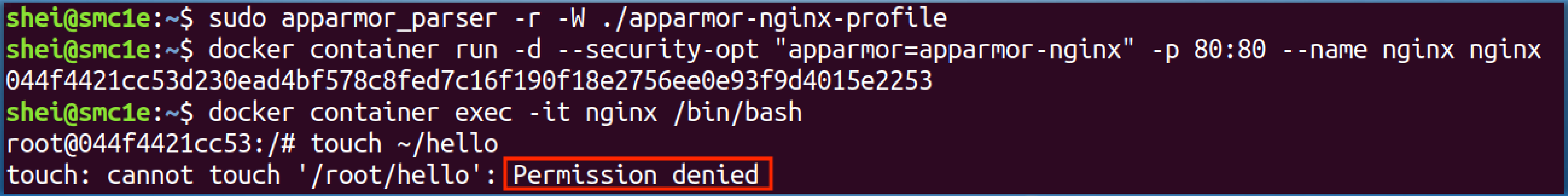


```
Security Options:
 apparmor
 seccomp
  Profile: default
Kernel Version: 4.15.0-99-generic
```

https://github.com/genuinetools/bane

$ sudo apparmor_parser -r -W /path/to/your/apparmor-nginx-profile

$ docker run -d --security-opt "apparmor=apparmor-profile-name" -p 80:80 nginx

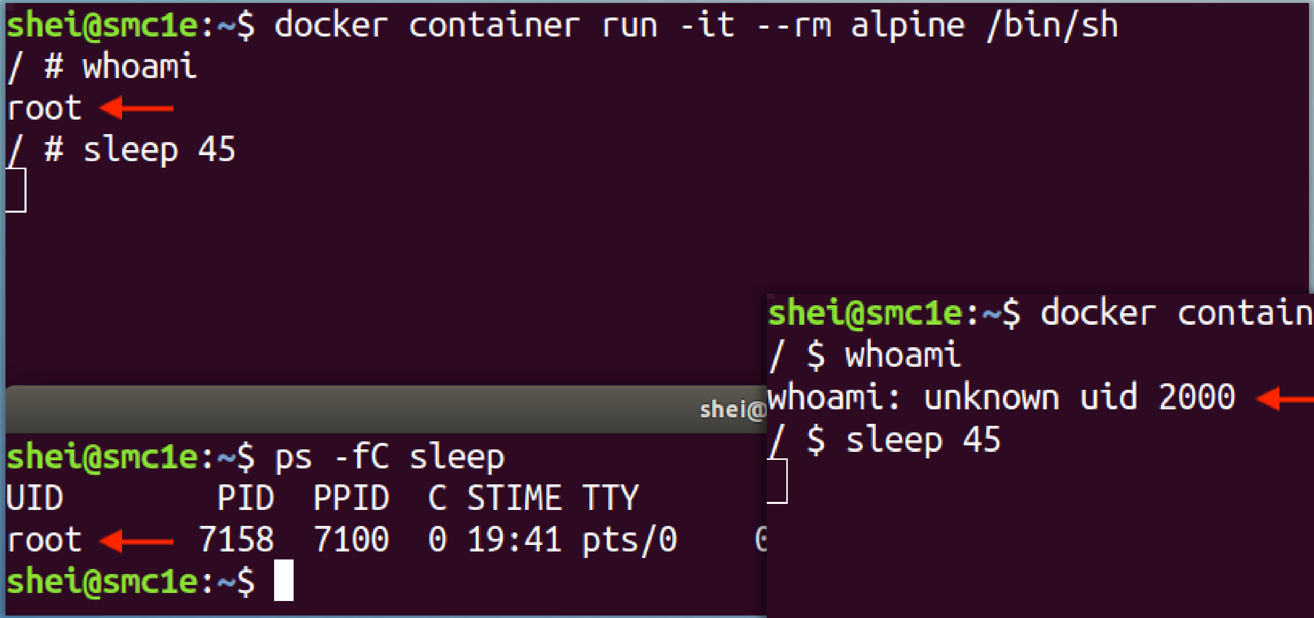


```
shei@smc1e:~$ sudo apparmor_parser -r -W ./apparmor-nginx-profile
shei@smc1e:~$ docker container run -d --security-opt "apparmor=apparmor-nginx" -p 80:80 --name nginx nginx
044f4421cc53d230ead4bf578c8fed7c16f190f18e2756ee0e93f9d4015e2253
shei@smc1e:~$ docker container exec -it nginx /bin/bash
root@044f4421cc53:/# touch ~/hello
touch: cannot touch '/root/hello': Permission denied
```

DREAMLAB TECHNOLOGIES

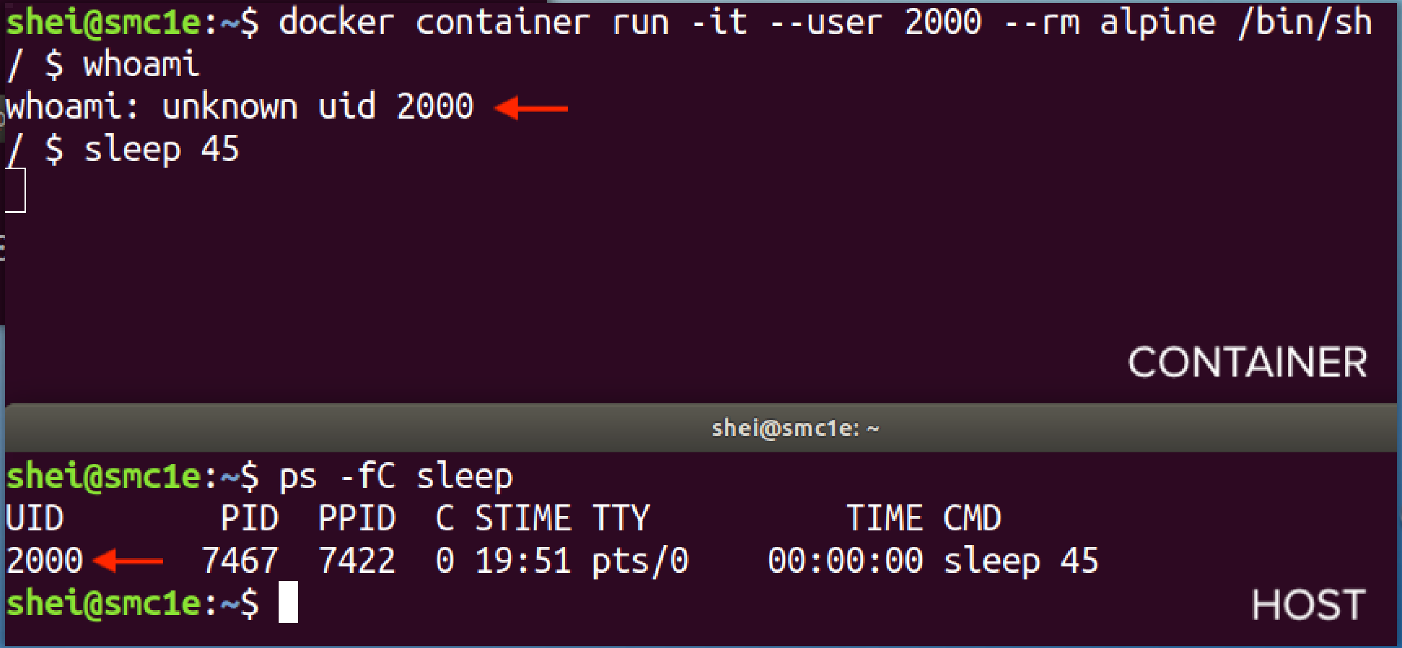
black hat USA 2020

CONTAINER UID & GID MANAGEMENT



```
shei@smc1e:~$ docker container run -it --rm alpine /bin/sh
/ # whoami
root
/ # sleep 45
```

```
shei@smc1e:~$ ps -fC sleep
UID          PID  PPID  C STIME TTY
root        7158  7100  0 19:41 pts/0
shei@smc1e:~$
```



```
shei@smc1e:~$ docker container run -it --user 2000 --rm alpine /bin/sh
/ $ whoami
whoami: unknown uid 2000
/ $ sleep 45
```

CONTAINER

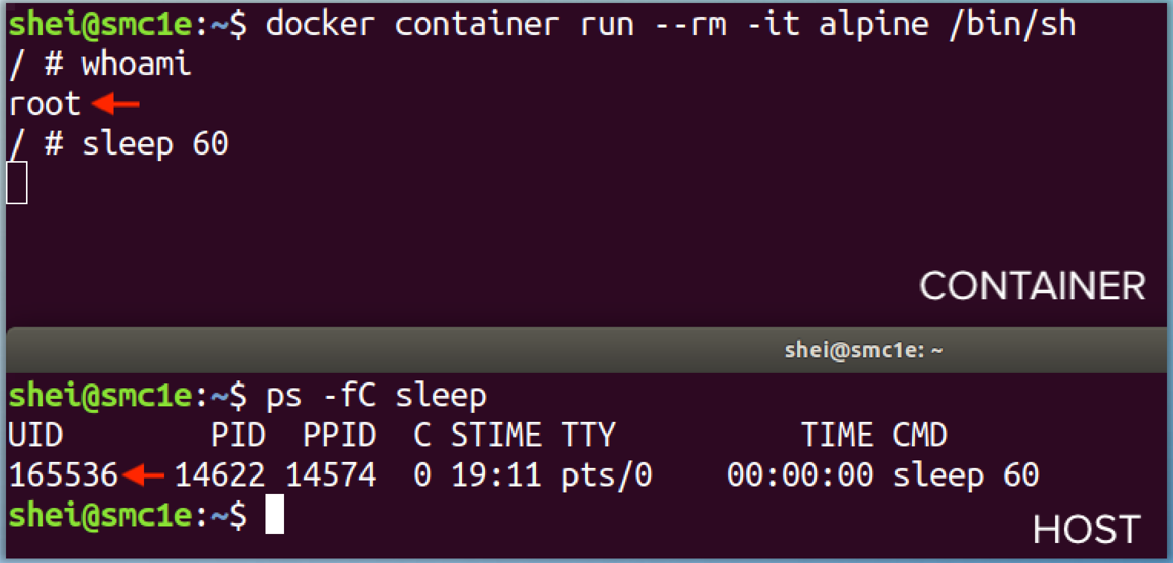
shei@smc1e: ~

```
shei@smc1e:~$ ps -fC sleep
UID          PID  PPID  C STIME TTY          TIME CMD
2000        7467  7422  0 19:51 pts/0    00:00:00 sleep 45
shei@smc1e:~$
```

HOST

USER NAMESPACE REMAP

```
/etc/docker/daemon.json

{
  "userns-remap": "default"
}
```



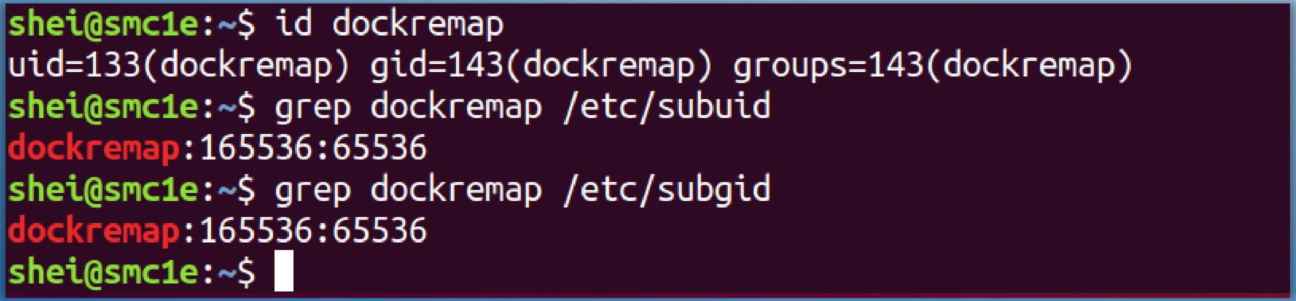
```
shei@smc1e:~$ docker container run --rm -it alpine /bin/sh
/ # whoami
root
/ # sleep 60
```

CONTAINER

shei@smc1e: ~

```
shei@smc1e:~$ ps -fC sleep
UID          PID  PPID  C STIME TTY          TIME CMD
165536     14622 14574  0 19:11 pts/0    00:00:00 sleep 60
shei@smc1e:~$
```

HOST



```
shei@smc1e:~$ id dockremap
uid=133(dockremap) gid=143(dockremap) groups=143(dockremap)
shei@smc1e:~$ grep dockremap /etc/subuid
dockremap:165536:65536
shei@smc1e:~$ grep dockremap /etc/subgid
dockremap:165536:65536
shei@smc1e:~$
```

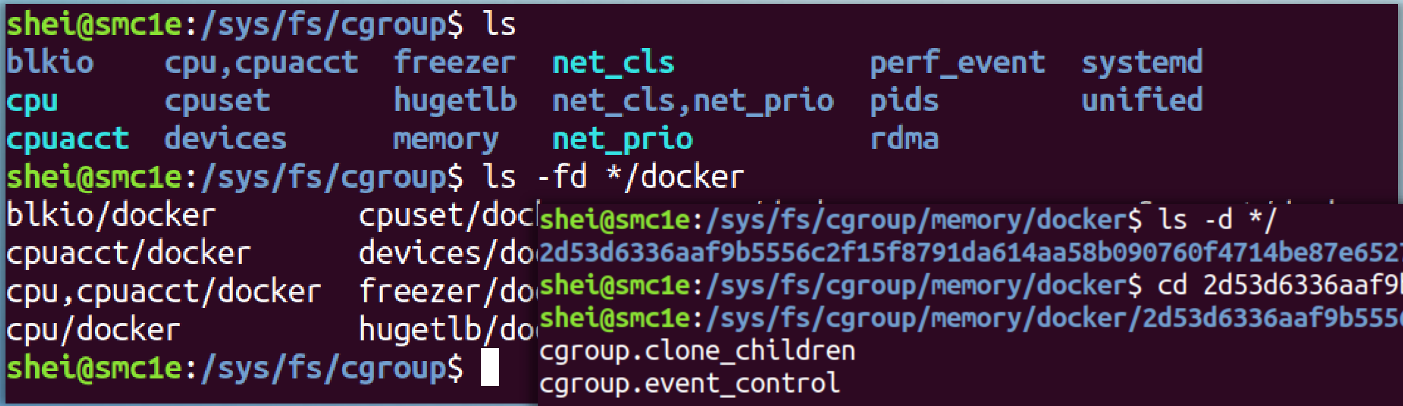
DEMO TIME!

DREAMLAB TECHNOLOGIES

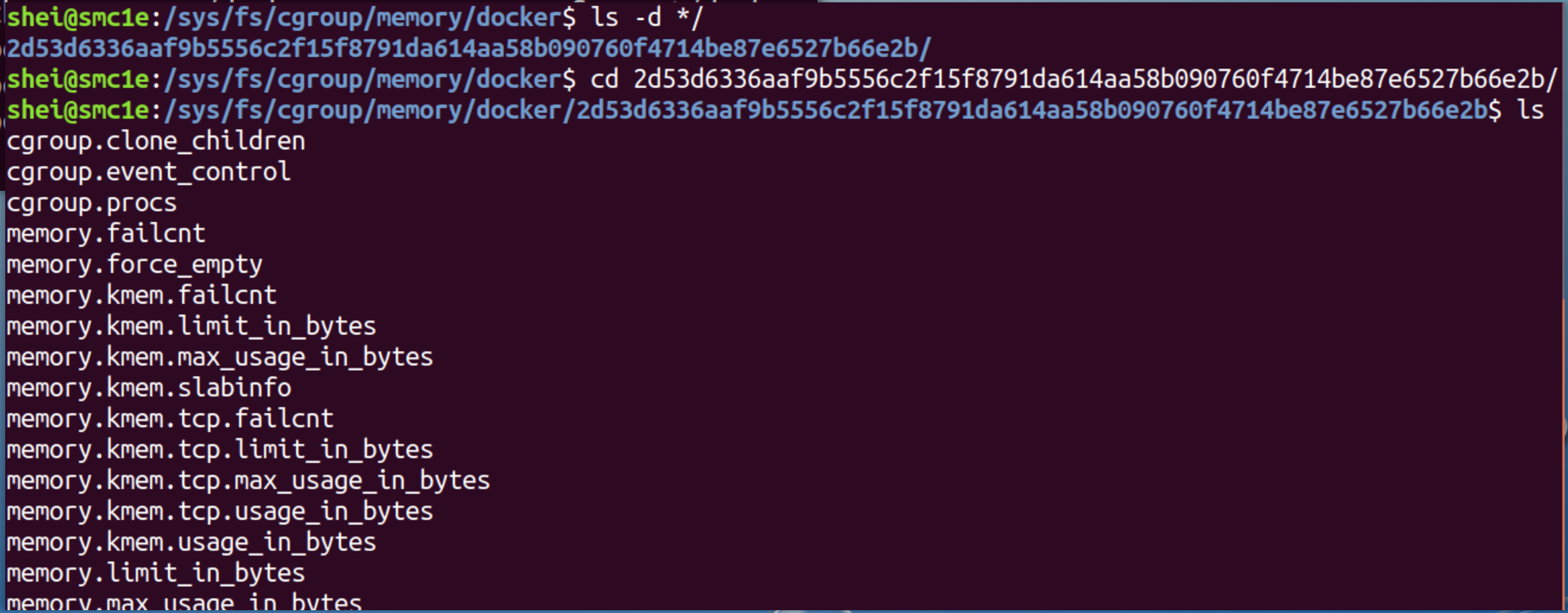
black hat USA 2020

CONTROL GROUPS – RESOURCE LIMITATION

❑ CPU ❑ Disk I/O ❑ Memory ❑ Hardware Resources

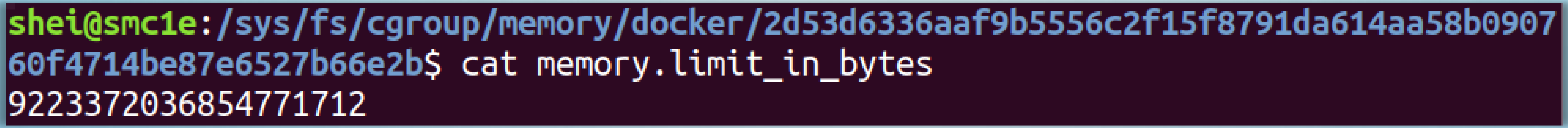


```
shei@smc1e:/sys/fs/cgroup$ ls
blkio    cpu,cpuacct  freezer  net_cls           perf_event  systemd
cpu      cpuset       hugetlb  net_cls,net_prio  pids        unified
cpuacct  devices      memory   net_prio          rdma
shei@smc1e:/sys/fs/cgroup$ ls -fd */docker
blkio/docker         cpuset/doc
cpuacct/docker       devices/do
cpu,cpuacct/docker   freezer/do
cpu/docker           hugetlb/do
shei@smc1e:/sys/fs/cgroup$ 
```



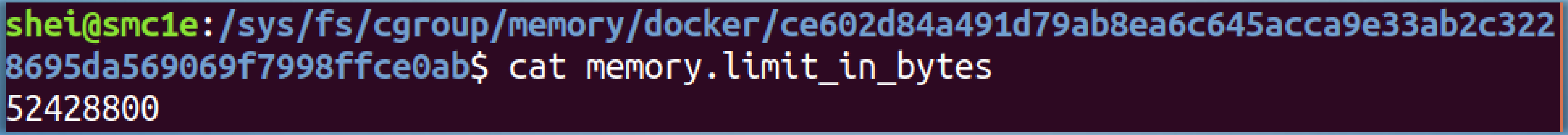
```
shei@smc1e:/sys/fs/cgroup/memory/docker$ ls -d */
2d53d6336aaf9b5556c2f15f8791da614aa58b090760f4714be87e6527b66e2b/
shei@smc1e:/sys/fs/cgroup/memory/docker$ cd 2d53d6336aaf9b5556c2f15f8791da614aa58b090760f4714be87e6527b66e2b/
shei@smc1e:/sys/fs/cgroup/memory/docker/2d53d6336aaf9b5556c2f15f8791da614aa58b090760f4714be87e6527b66e2b$ ls
cgroup.clone_children
cgroup.event_control
cgroup.procs
memory.failcnt
memory.force_empty
memory.kmem.failcnt
memory.kmem.limit_in_bytes
memory.kmem.max_usage_in_bytes
memory.kmem.slabinfo
memory.kmem.tcp.failcnt
memory.kmem.tcp.limit_in_bytes
memory.kmem.tcp.max_usage_in_bytes
memory.kmem.tcp.usage_in_bytes
memory.kmem.usage_in_bytes
memory.limit_in_bytes
memory.max_usage_in_bytes
```

CONTROL GROUPS – RESOURCE LIMITATION



```
shei@smc1e:/sys/fs/cgroup/memory/docker/2d53d6336aaf9b5556c2f15f8791da614aa58b0907
60f4714be87e6527b66e2b$ cat memory.limit_in_bytes
9223372036854771712
```

```
$ docker container run --rm --memory 50M -it alpine /bin/sh
```

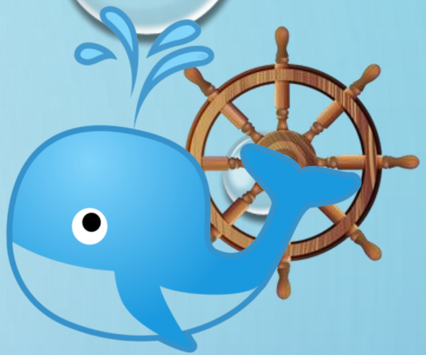


```
shei@smc1e:/sys/fs/cgroup/memory/docker/ce602d84a491d79ab8ea6c645acca9e33ab2c322
8695da569069f7998ffce0ab$ cat memory.limit_in_bytes
52428800
```

--memory-swap	--cpu-quota
--cpus	--cpu-period

https://docs.docker.com/engine/reference/commandline/run/

SECURING DOCKER IMAGES



DISTROLESS – MULTI-STAGE BUILDS

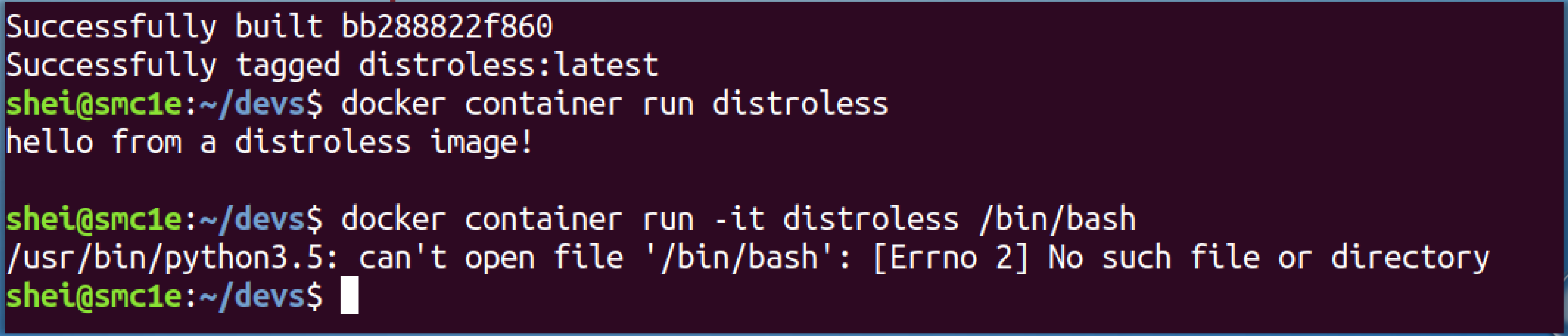
DISTROLESS BASE IMAGES AND MULTI-STAGE BUILDS

```
FROM python:3-slim AS build-env
ADD . /app
WORKDIR /app
```

Build Stage

```
FROM gcr.io/distroless/python3
COPY --from=build-env /app /app
WORKDIR /app
CMD ["hello.py"]
```

Final Image



```
Successfully built bb288822f860
Successfully tagged distroless:latest
shei@smc1e:~/devs$ docker container run distroless
hello from a distroless image!

shei@smc1e:~/devs$ docker container run -it distroless /bin/bash
/usr/bin/python3.5: can't open file '/bin/bash': [Errno 2] No such file or directory
shei@smc1e:~/devs$
```

DREAMLAB TECHNOLOGIES

black hat USA 2020

DOCKER CONTENT TRUST

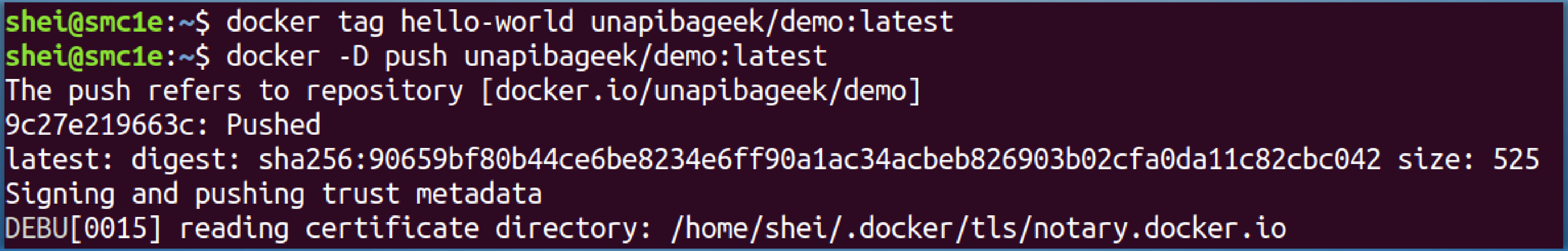
DOCKER CONTENT TRUST – SIGNED IMAGES

Image Publisher side:

Step 1: $ DOCKER_CONTENT_TRUST=1

Step 2: $ docker trust key generate <your_name>

Step 3: $ docker trust signer add --key <your-key.pub> <your-name> <your-repo>



```
shei@smc1e:~$ docker tag hello-world unapibageek/demo:latest
shei@smc1e:~$ docker -D push unapibageek/demo:latest
The push refers to repository [docker.io/unapibageek/demo]
9c27e219663c: Pushed
latest: digest: sha256:90659bf80b44ce6be8234e6ff90a1ac34acbeb826903b02cfa0da11c82cbc042 size: 525
Signing and pushing trust metadata
DEBU[0015] reading certificate directory: /home/shei/.docker/tls/notary.docker.io
```

DREAMLAB TECHNOLOGIES

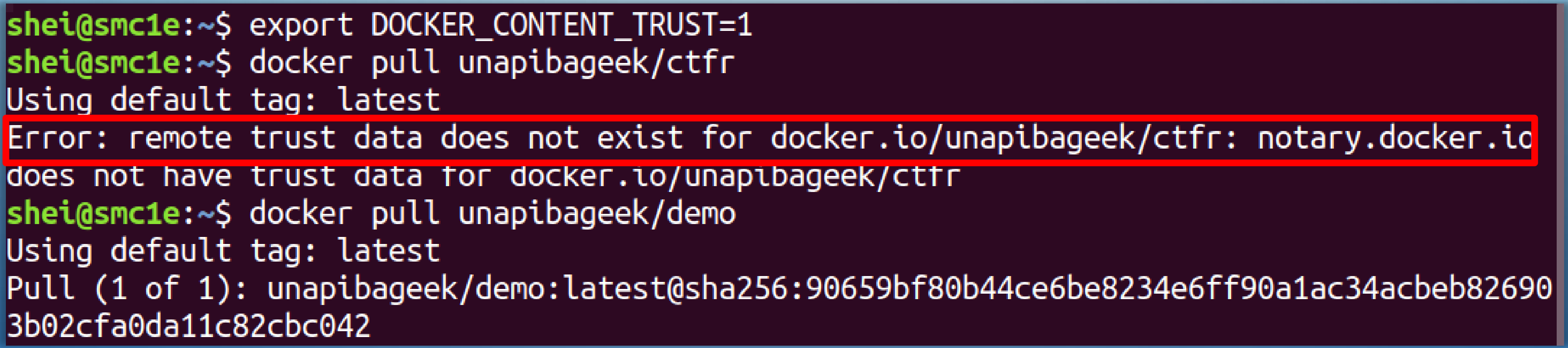
black hat USA 2020

DOCKER CONTENT TRUST

DOCKER CONTENT TRUST – SIGNED IMAGES

Image Consumer side:

$ DOCKER_CONTENT_TRUST=1

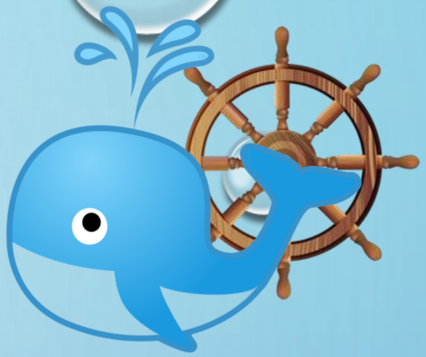


```
shei@smc1e:~$ export DOCKER_CONTENT_TRUST=1
shei@smc1e:~$ docker pull unapibageek/ctfr
Using default tag: latest
Error: remote trust data does not exist for docker.io/unapibageek/ctfr: notary.docker.io
does not have trust data for docker.io/unapibageek/ctfr
shei@smc1e:~$ docker pull unapibageek/demo
Using default tag: latest
Pull (1 of 1): unapibageek/demo:latest@sha256:90659bf80b44ce6be8234e6ff90a1ac34acbeb82690
3b02cfa0da11c82cbc042
```

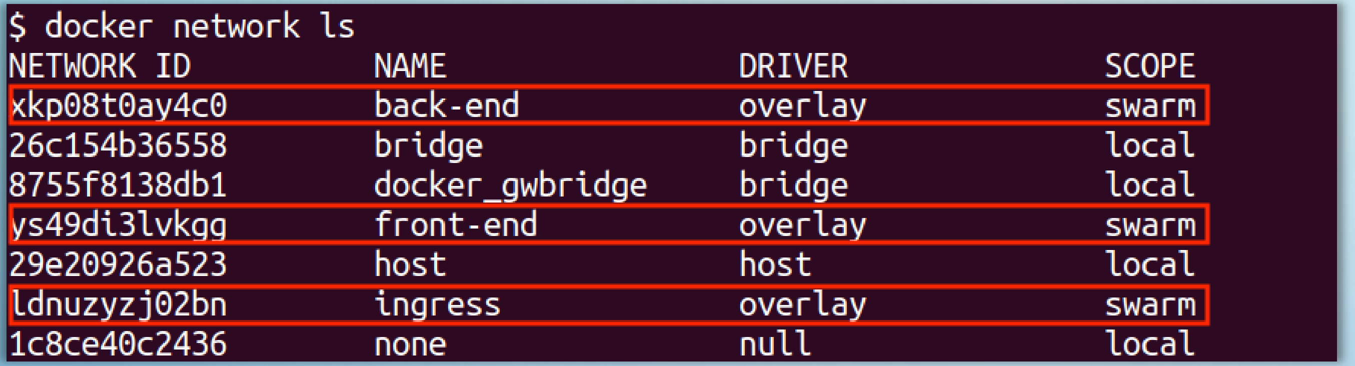
DREAMLAB
TECHNOLOGIES

black hat
USA 2020

SECURING DOCKER SWARM ENVIRONMENTS



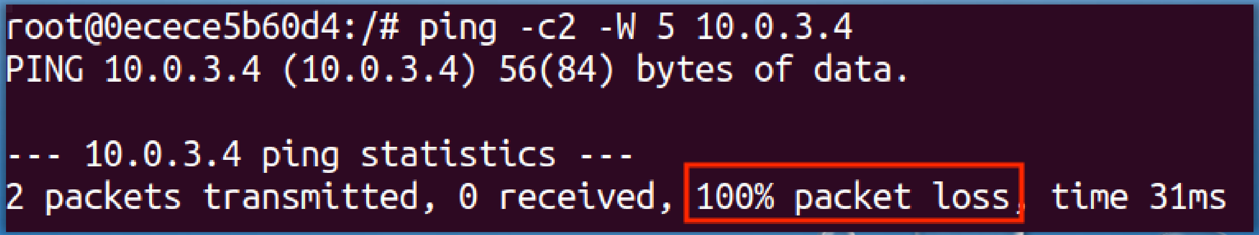
NETWORK ISOLATION



```
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
xkp08t0ay4c0        back-end            overlay             swarm
26c154b36558        bridge              bridge              local
8755f8138db1        docker_gwbridge     bridge              local
ys49di3lvkgg        front-end           overlay             swarm
29e20926a523        host                host                local
ldnuzyzj02bn        ingress             overlay             swarm
1c8ce40c2436        none                null                local
```

```
$ docker network create -d overlay back-end
$ docker network create -d overlay front-end
```

```
$ docker service create -d --network back-end --name redis redis
$ docker service create -d --network front-end --name nginx nginx
```



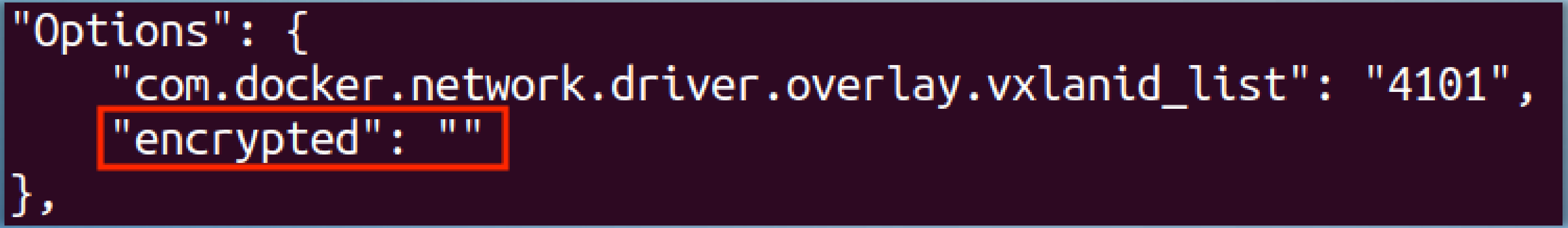
```
root@0ecece5b60d4:/# ping -c2 -W 5 10.0.3.4
PING 10.0.3.4 (10.0.3.4) 56(84) bytes of data.

--- 10.0.3.4 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 31ms
```

COMMUNICATION ENCRYPTION

$ docker network create -d overlay --opt encrypted encrypted-net

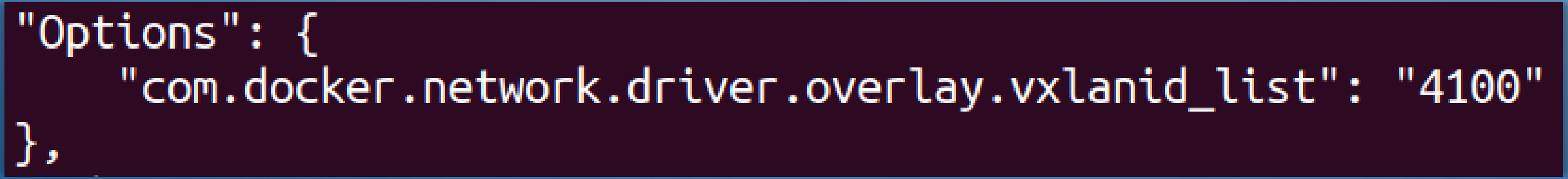
$ docker network inspect encrypted-net

Encrypted network:



```
"Options": {
    "com.docker.network.driver.overlay.vxlanid_list": "4101",
    "encrypted": ""
},
```

Non-encrypted network:

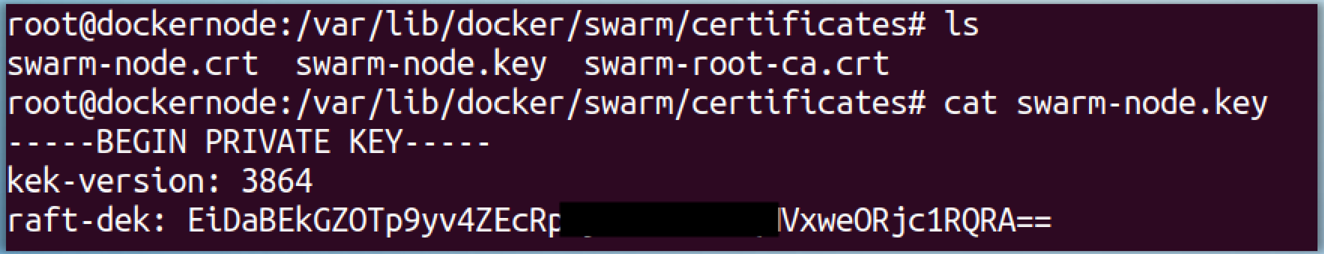


```
"Options": {
    "com.docker.network.driver.overlay.vxlanid_list": "4100"
},
```

RAFT-LOGS KEY ENCRYPTION

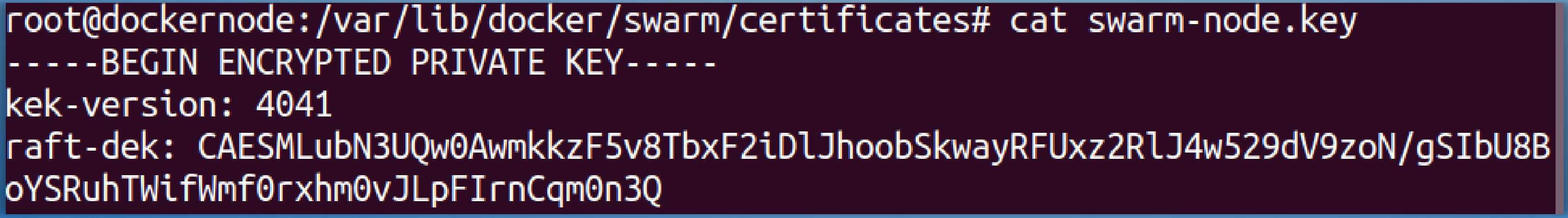
/var/lib/docker/swarm/raft/

/var/lib/docker/swarm/certificates/swarm-node.key



```
root@dockernode:/var/lib/docker/swarm/certificates# ls
swarm-node.crt  swarm-node.key  swarm-root-ca.crt
root@dockernode:/var/lib/docker/swarm/certificates# cat swarm-node.key
-----BEGIN PRIVATE KEY-----
kek-version: 3864
raft-dek: EiDaBEkGZOTp9yv4ZEcRp[redacted]VxweORjc1RQRA==
```

$ docker swarm update --autolock=true

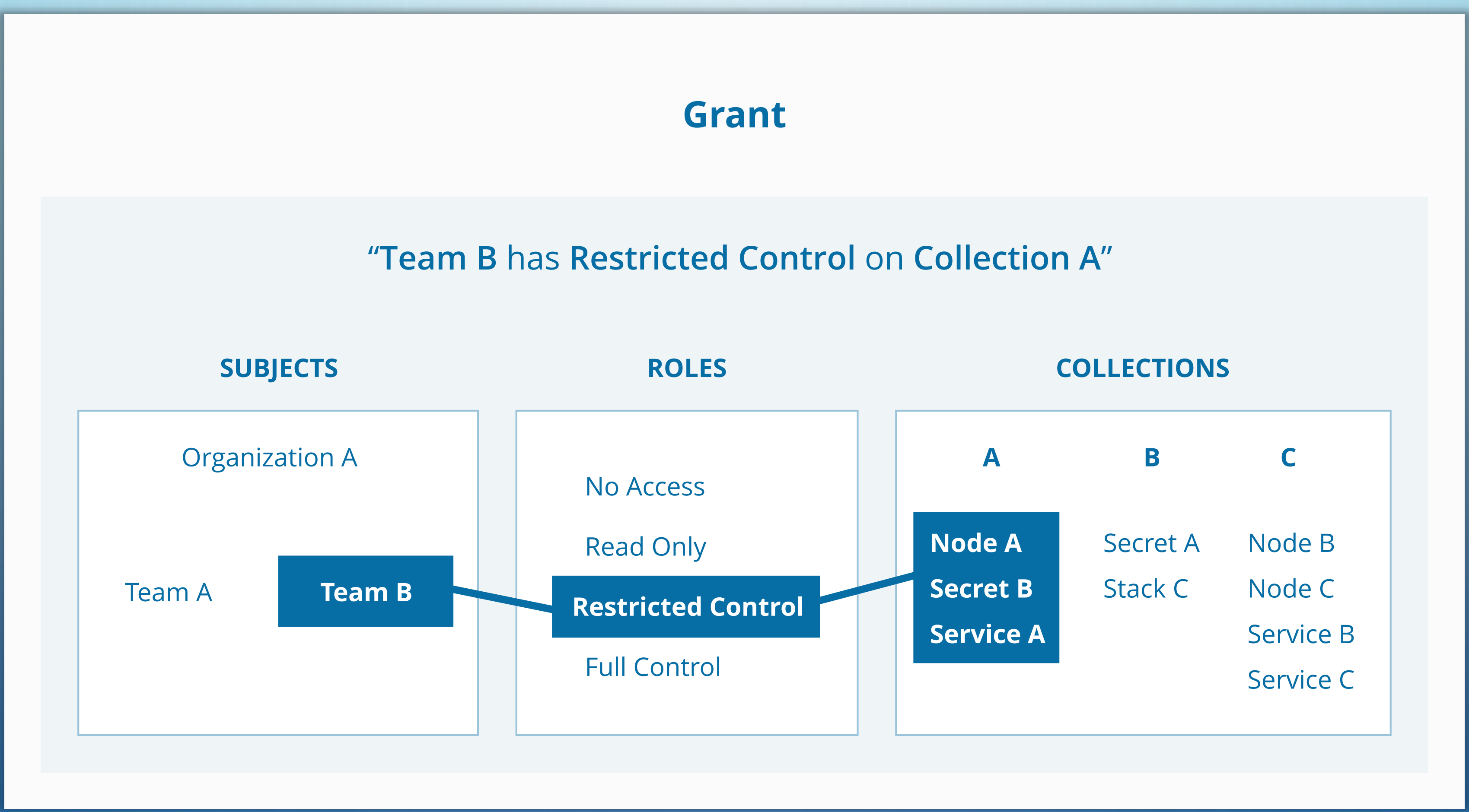


```
root@dockernode:/var/lib/docker/swarm/certificates# cat swarm-node.key
-----BEGIN ENCRYPTED PRIVATE KEY-----
kek-version: 4041
raft-dek: CAESMLubN3UQw0AwmkkzF5v8TbxF2iDlJhoobSkwayRFUxz2RlJ4w529dV9zoN/gSIbU8B
oYSRuhTWifWmf0rxhm0vJLpFIrnCqm0n3Q
```

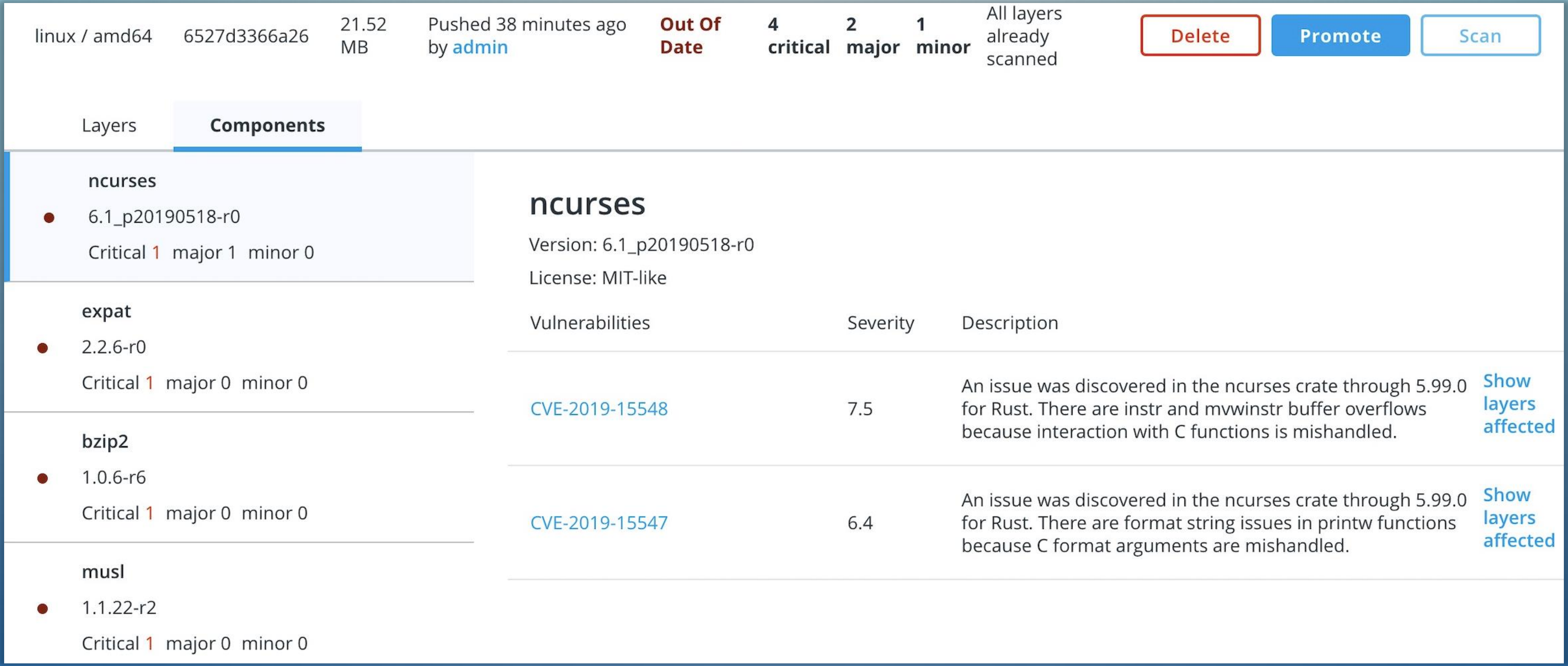
DREAMLAB TECHNOLOGIES

black hat USA 2020

UCP SECURITY – RBAC



DTR SECURITY



linux / amd64 | 6527d3366a26 | 21.52 MB | Pushed 38 minutes ago by admin | Out Of Date | 4 critical | 2 major | 1 minor | All layers already scanned

● **ncurses** 6.1_p20190518-r0 Critical 1 major 1 minor 0

● **expat** 2.2.6-r0 Critical 1 major 0 minor 0

● **bzip2** 1.0.6-r6 Critical 1 major 0 minor 0

● **musl** 1.1.22-r2 Critical 1 major 0 minor 0

ncurses

Version: 6.1_p20190518-r0

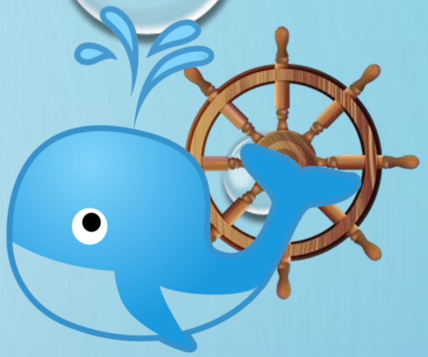
License: MIT-like

Vulnerabilities	Severity	Description	
CVE-2019-15548	7.5	An issue was discovered in the ncurses crate through 5.99.0 for Rust. There are instr and mvwinstr buffer overflows because interaction with C functions is mishandled.	Show layers affected
CVE-2019-15547	6.4	An issue was discovered in the ncurses crate through 5.99.0 for Rust. There are format string issues in printw functions because C format arguments are mishandled.	Show layers affected

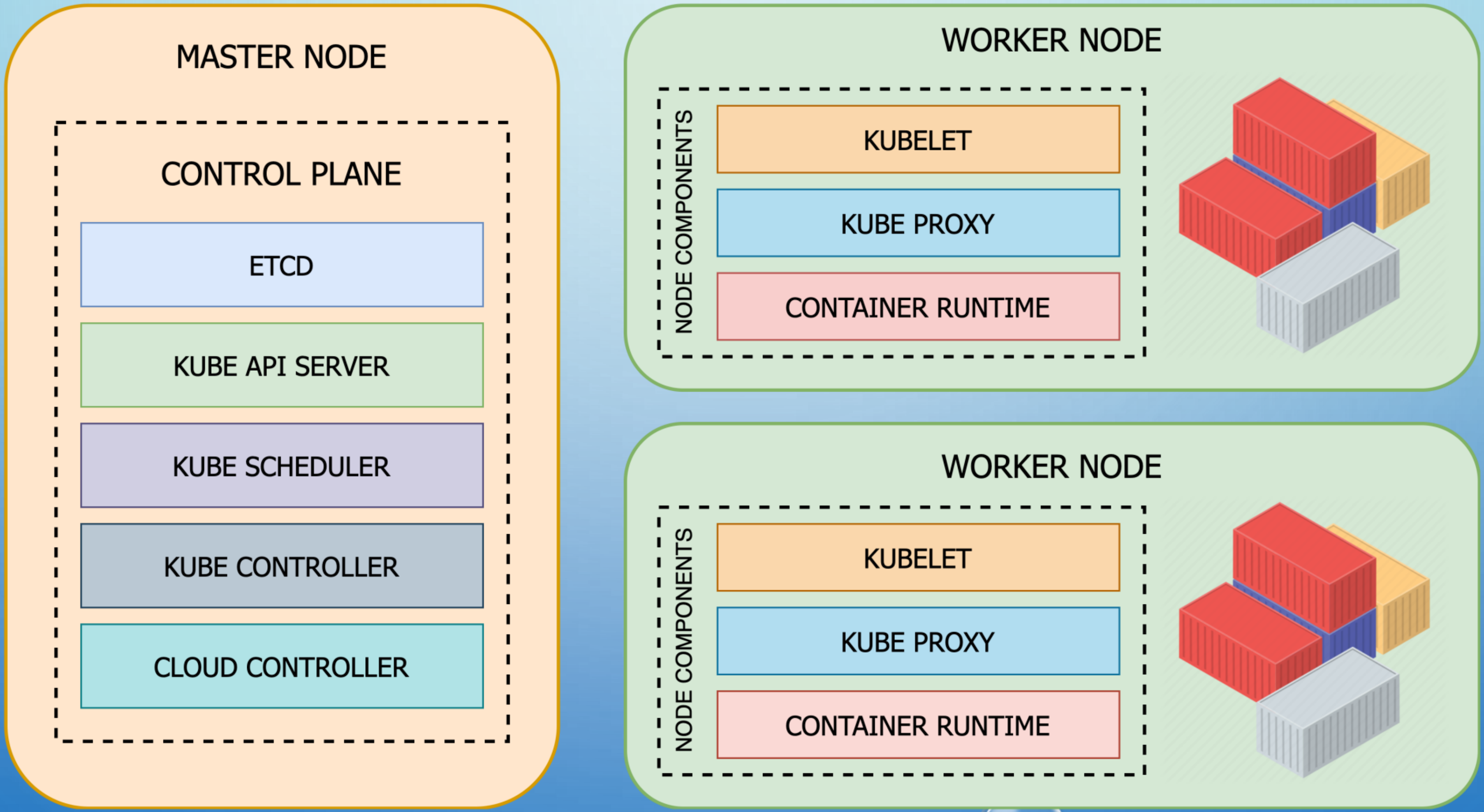
DREAMLAB TECHNOLOGIES

black hat USA 2020

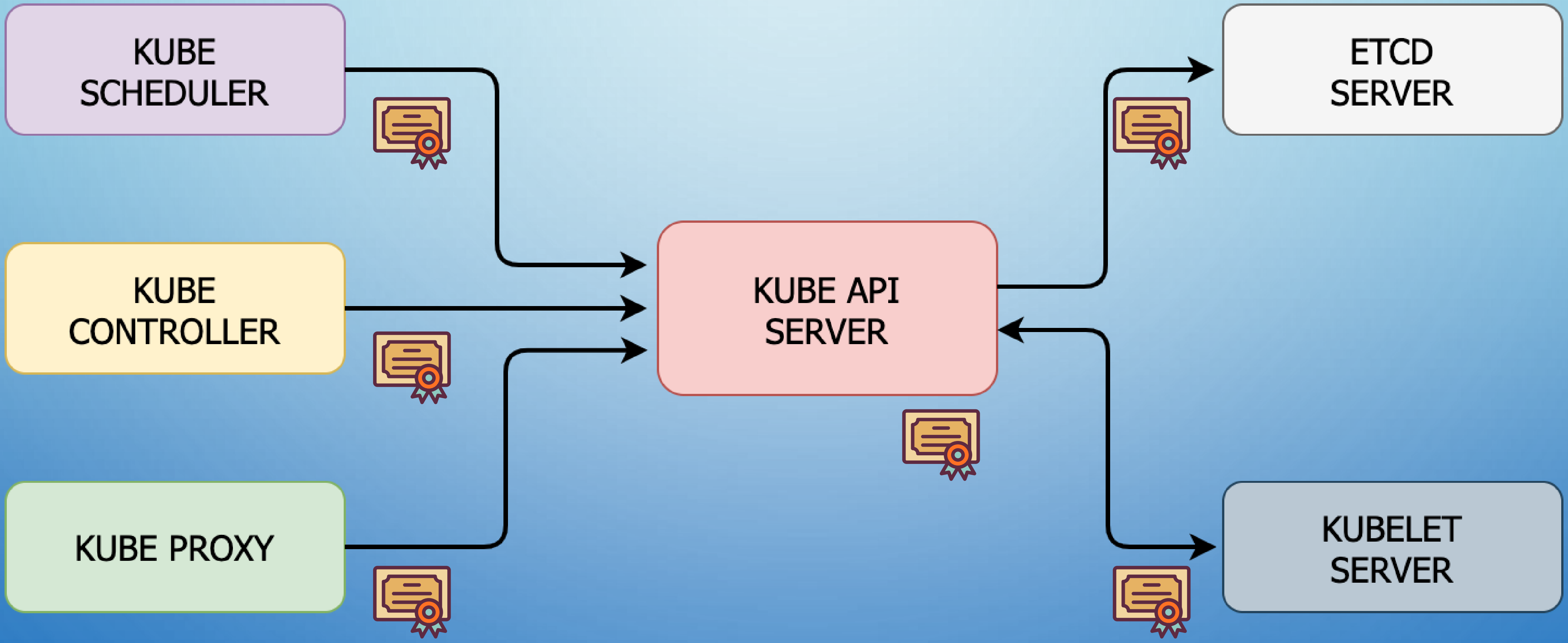
SECURING KUBERNETES ENVIRONMENTS



KUBERNETES ARCHITECTURE



SECURING COMPONENTS COMMUNICATION



SECURING COMPONENTS COMMUNICATION

Kubernetes cluster's CAs:

- ❑ Etcd CA
- ❑ Kubernetes CA

Server Certificates:

- ❑ Etcd
- ❑ Kube-API server
- ❑ Kubelet

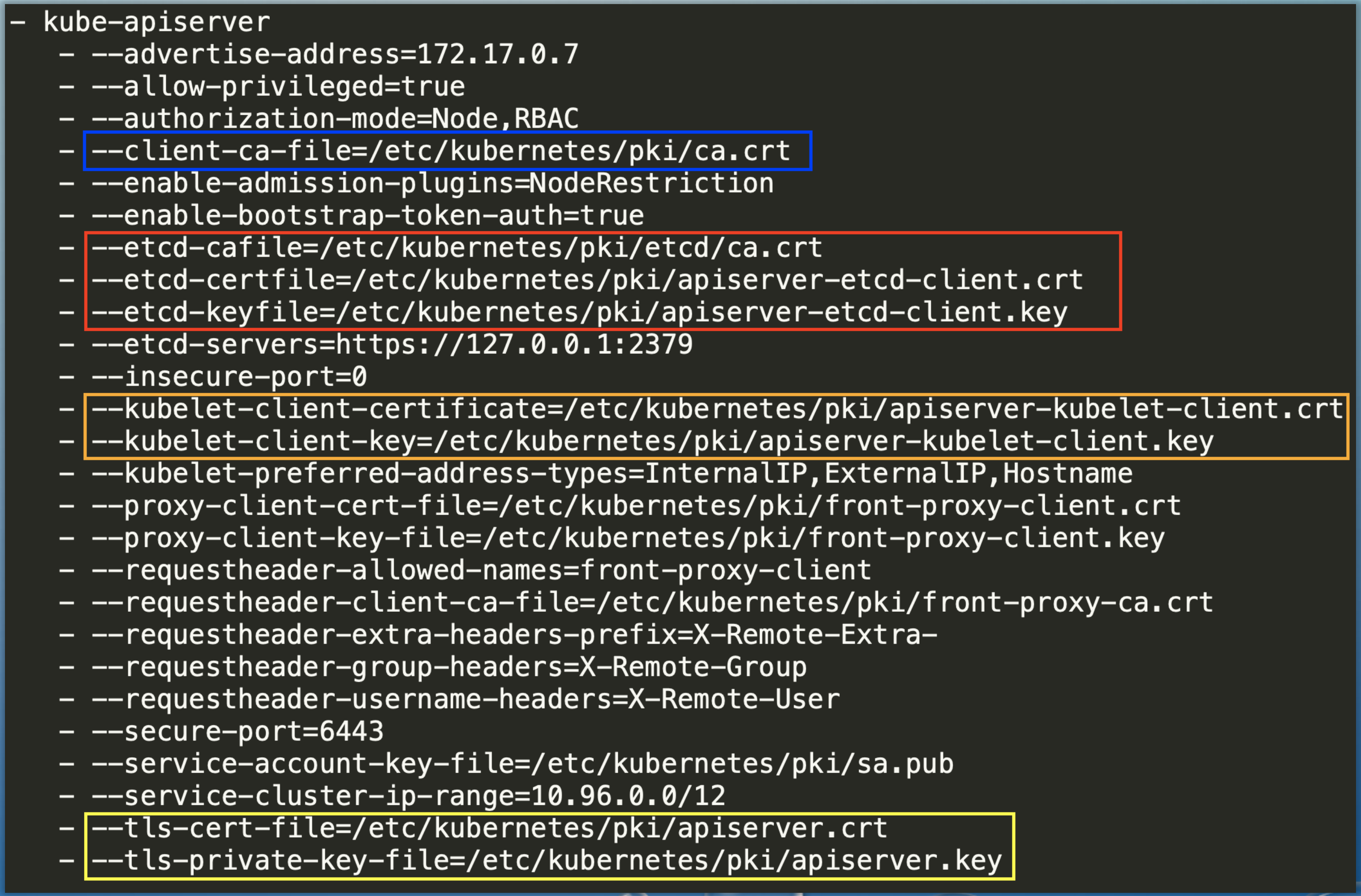
Client Certificates:

- ❑ Kube-scheduler
- ❑ Kube-controller
- ❑ Kube-proxy
- ❑ Kube-API server (etcd client)
- ❑ Kube-API server (kubelet client)
- ❑ Kubelet (API server client)

SECURING COMPONENTS COMMUNICATION

Kube-API Certificates:

- ❑ Etcd client
- ❑ Kubelet client
- ❑ Kube-api server



```
- kube-apiserver
    - --advertise-address=172.17.0.7
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
    - --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
    - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
    - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
    - --requestheader-allowed-names=front-proxy-client
    - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
    - --requestheader-extra-headers-prefix=X-Remote-Extra-
    - --requestheader-group-headers=X-Remote-Group
    - --requestheader-username-headers=X-Remote-User
    - --secure-port=6443
    - --service-account-key-file=/etc/kubernetes/pki/sa.pub
    - --service-cluster-ip-range=10.96.0.0/12
    - --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
    - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
```

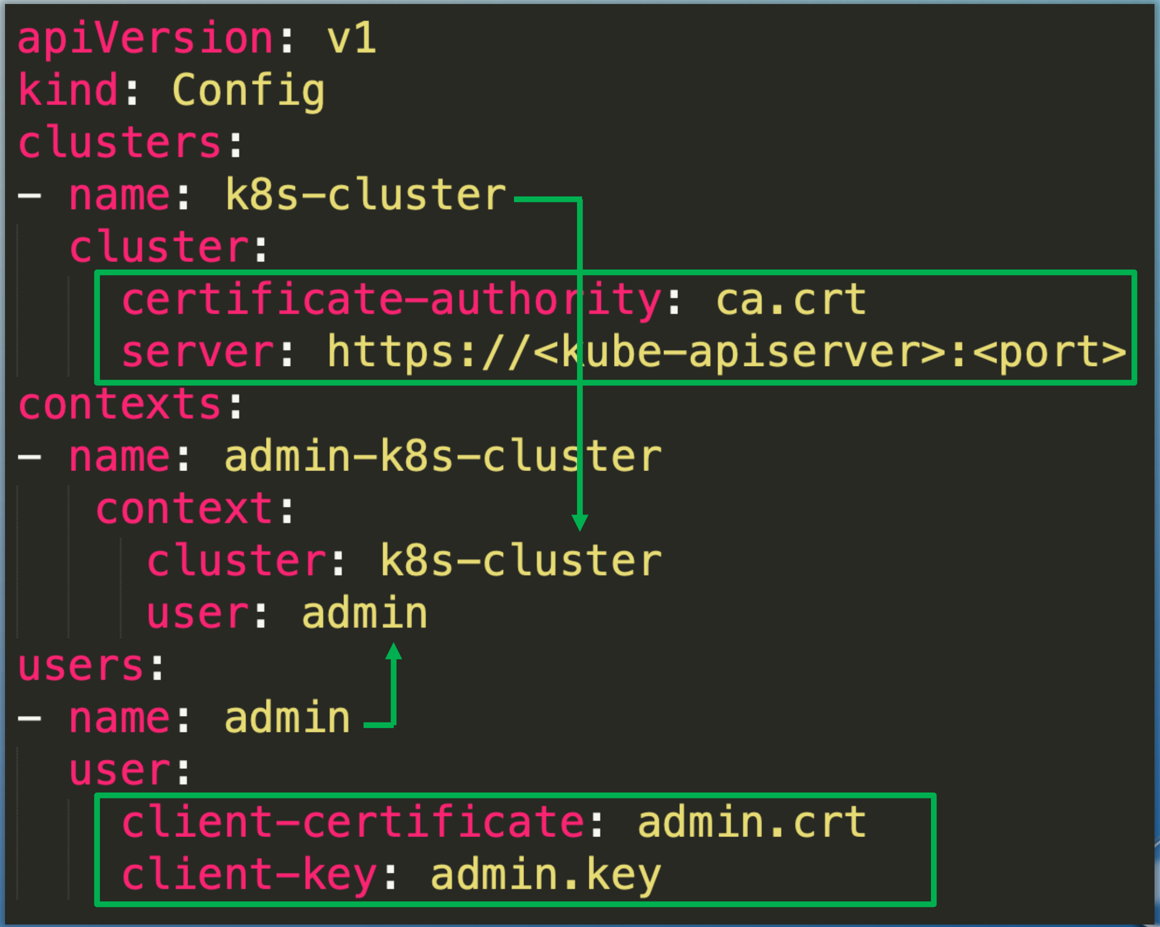
API AUTHENTICATION

Kubernetes authentication mechanisms:

- ❑ Basic (user/password or token).
- ❑ TLS Certificates.
- ❑ LDAP, Kerberos, etc.

```
$ openssl genrsa -out admin.key 2048

$ openssl req -new -key admin.key -subj
"CN=admin/O=system:masters" \ -out
admin.csr
$ openssl x509 -req -in admin.csr -CA ca.crt \
-CAkey ca.key -out admin.crt
```

$HOME/.kube/config



```
apiVersion: v1
kind: Config
clusters:
- name: k8s-cluster
  cluster:
    certificate-authority: ca.crt
    server: https://<kube-apiserver>:<port>
contexts:
- name: admin-k8s-cluster
   context:
     cluster: k8s-cluster
     user: admin
users:
- name: admin
  user:
    client-certificate: admin.crt
    client-key: admin.key
```

DREAMLAB TECHNOLOGIES

black hat USA 2020

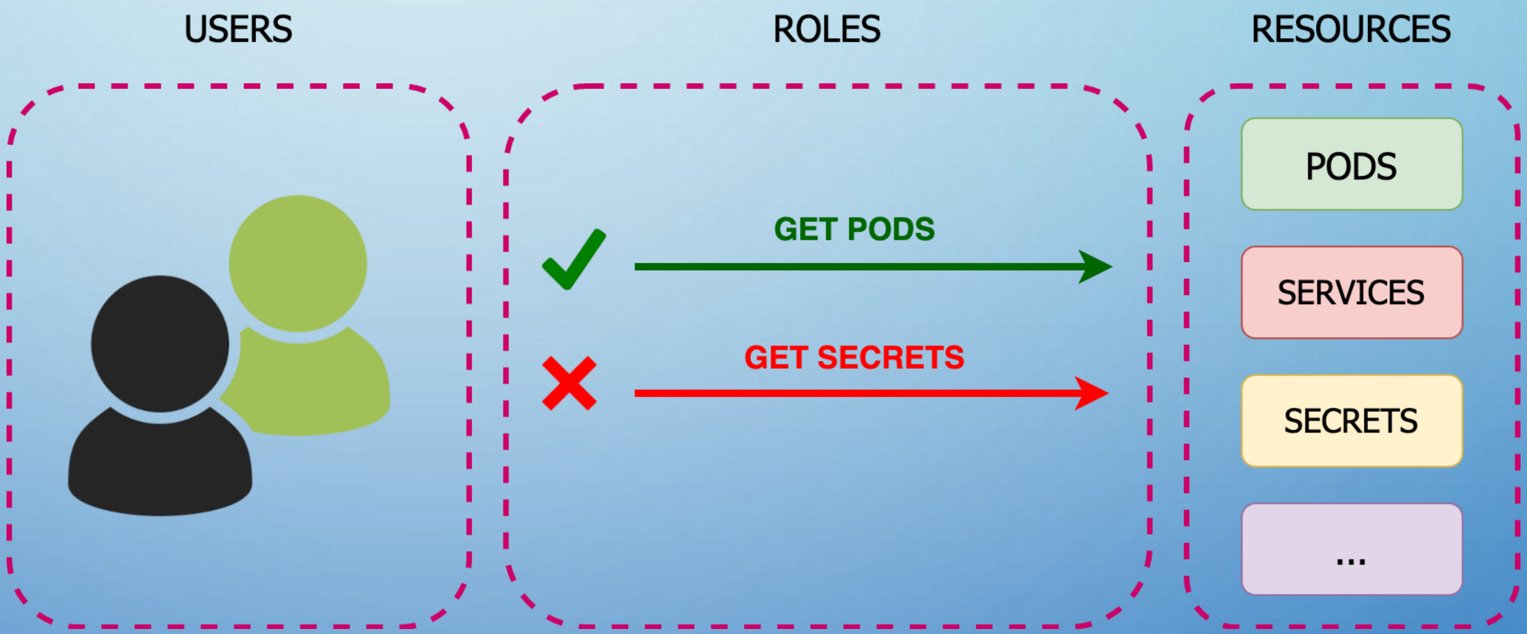
API AUTHORIZATION

Kubernetes authorization mechanisms:

- ❑ Node
- ❑ ABAC
- ❑ RBAC
- ❑ WebHook

RBAC objects:

- ❑ Role
- ❑ Role Binding
- ❑ Cluster Role
- ❑ Cluster Role Binding



API AUTHORIZATION – RBAC

Role & Role Binding example

role-definition.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: pod-reader-role
  namespace: backend
rules:
  - apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list"]
```

role-binding-definition.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: pod-reader-binding
  namespace: backend
subjects:
- kind: ServiceAccount
  name: sa-token
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader-role
  apiGroup: rbac.authorization.k8s.io
```

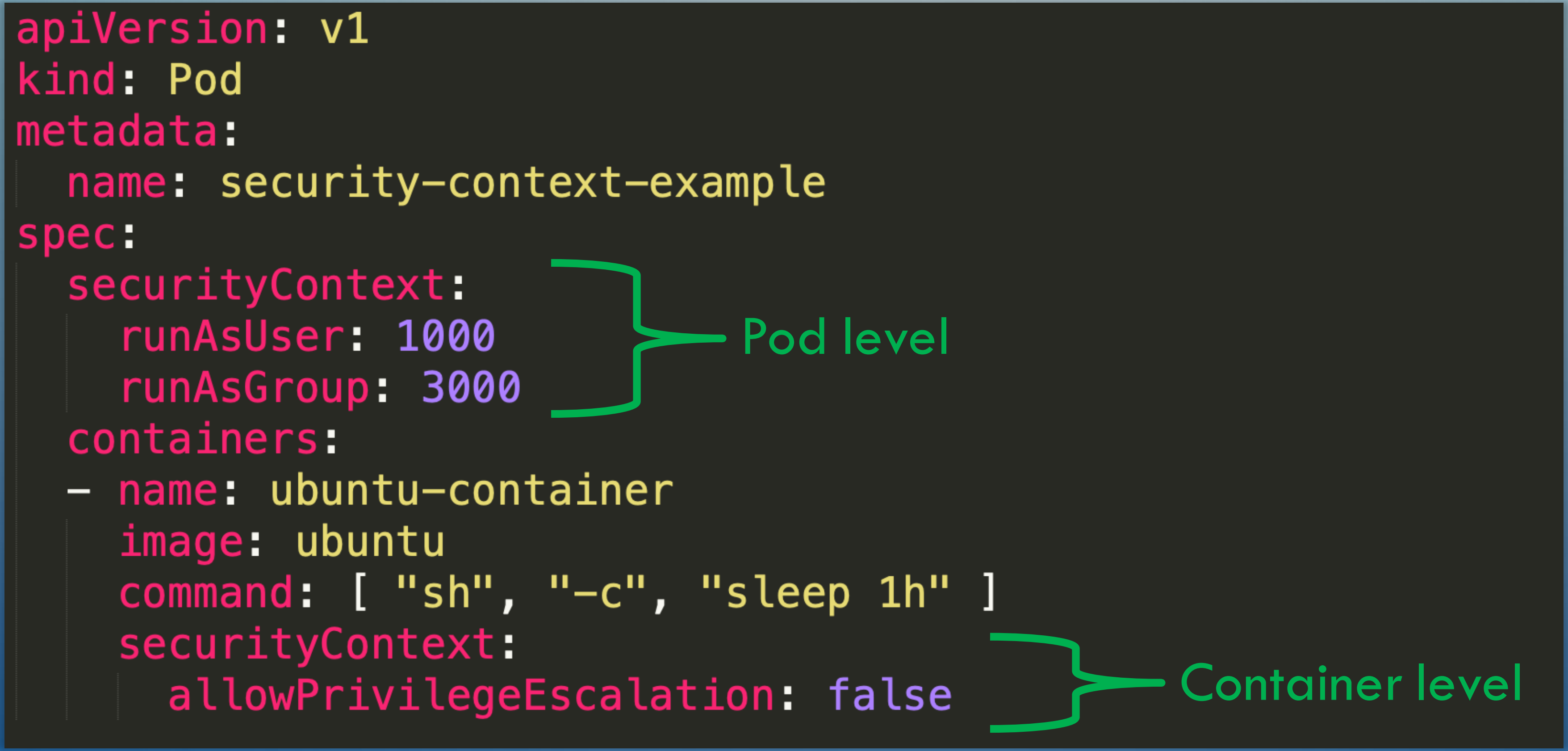
DEMO TIME!

DREAMLAB TECHNOLOGIES

black hat USA 2020

SECURITY CONTEXT

Security Context example



```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-example
spec:
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
  containers:
  - name: ubuntu-container
    image: ubuntu
    command: [ "sh", "-c", "sleep 1h" ]
    securityContext:
      allowPrivilegeEscalation: false
```

NETWORK POLICIES

Network Policies examples

default-deny-all.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-all
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress
```

namespace-isolation.yaml

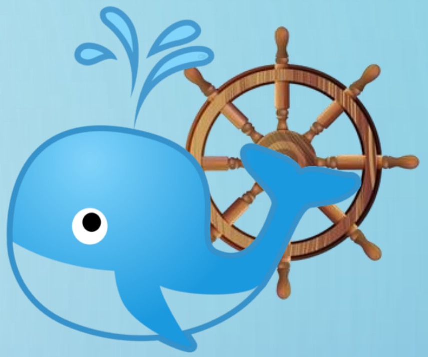
```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-same-namespace
  namespace: backend
spec:
  podSelector: {}
  ingress:
    - from:
      - namespaceSelector:
          matchLabels:
            name: backend
  egress:
    - to:
      - namespaceSelector:
          matchLabels:
            name: backend
```

WHITE PAPERS DOWNLOAD

Defending Docker, Swarm and Kubernetes white papers:

https://dreamlab.net/blackhat-whitepapers

- docs.docker.com
- kubernetes.io/docs



THANK YOU!

Sheila A. Berta (@UnaPibaGeek)
Dreamlab Technologies

sheila.berta@dreamlab.net
www.dreamlab.net