# Exploiting  Containers

The inside-out approach

# ESSENTIAL THEORY

# Hey Docker! What’s under the hood?

- Namespaces
- Control Groups
- Union File System
- Capabilities

# Hey Docker! What's under the hood?

- N[illegible]
- Control Groups
- Union File System
- Capabilities

The isolated workspaces we call containers

# Hey Docker! What's under the hood?

- N[illegible]
- C[illegible]ups
- Union File System
- Capabilities

The isolated workspaces we call containers

Limits a process to a specific set of resources

# Hey Docker! What's under the hood?

- N
- C
- Uni
- Capabilities

The isolated workspaces we call containers

Limits a process to a specific set of resources

Operate by creating layers

# Hey Docker! What's under the hood?

- N
- C
- Uni
- Cap

The isolated workspaces we call containers

Limits a process to a specific set of resources

Operate by creating layers

Granular root powers

# Hey Docker! What's under the hood?

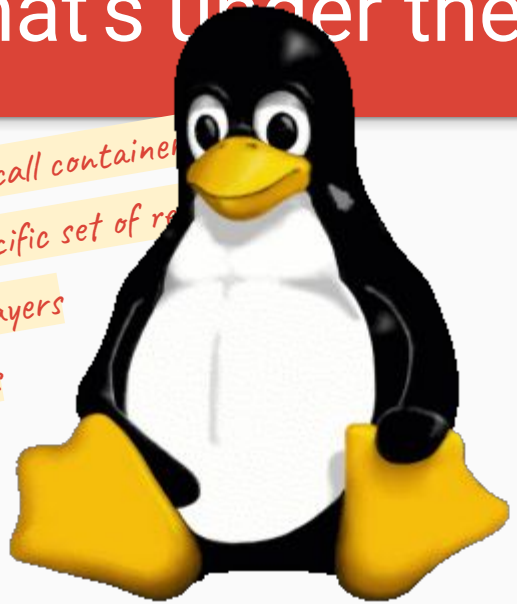
- N
- C
- Uni
- Cap

The isolated workspaces we call containe

Limits a process to a specific set of r

Operate by creating layers

Granular root powers



# Namespaces

Namespaces provide a layer of isolation. Each aspect of a container runs in a separate namespace and its access is limited to that namespace.

Docker Engine uses namespaces such as the following on Linux:

- **PID**: Process isolation
- **NET**: Managing network interfaces
- **IPC**: Managing access to IPC resources
- **MNT**: Managing filesystem mount points
- **UTS**: Isolating kernel and version identifiers

Each process is in one namespace of each type

# Control Groups (cgroups)

A cgroup limits an application to a specific set of resources. It allows Docker Engine to share available hardware resources to containers and optionally enforce limits and constraints. For example, you can limit the memory available to a specific container.

A cgroup is itself a type of namespace.

# Union File System (unionfs)

Union mounting is a way of combining multiple directories into one that appears to contain their combined contents.

Many images that are used spin up our containers are quite bulky whether. It would be quite expensive to allocate that much space every time a container is created from these images. Thanks to union filesystem, Docker only needs to create thin layer on top of the image and rest of it can be shared between all the containers.

# Capabilities

The Linux kernel is able to break down the privileges of the root user into distinct units referred to as capabilities.

This breaking down of root privileges into granular capabilities allows you to:

- Remove individual capabilities from the root user account, making it less powerful/dangerous.
- Add privileges to non-root users at a very granular level.

By default, Docker drops all capabilities except those needed using a whitelist approach.

# OFFENSIVE OPERATIONS

# Where’s me in a Docker environment?

- Inside a container
- On a host running container(s)
- In a network with other host(s) exposing Docker API
- In a code repository containing the Dockerfile/docker-compose.yml file(s)

# Where’s me in a Docker environment?

- Inside a container
- On a host running container(s)
- In a network with other host(s) exposing Docker API
- In a code repository containing the Dockerfile/docker-compose.yml file(s)

Breakout/Escape, Extract Secrets

# Where’s me in a Docker environment?

- Inside a container
- On a host running container(s)
- In a network with other host(s) exposing Docker API
- In a code repository containing the Dockerfile/docker-compose.yml file(s)

Breakout/Escape, Extract Secrets

Escalate Privileges, Extract Secrets

# Where's me in a Docker environment?

- Inside a container
- On a host running container(s)
- In a network with other host(s) running Docker API
- In a code repository containing the Dockerfile/docker-compose.yml file(s)

Breakout/Escape, Extract Secrets

Escalate Privileges, Extract Secrets

Move Laterally, Extract Secrets

# Where's me in a Docker environment?

- Inside a container
- On a host running container(s)
- In a network with other hosts running Docker API
- In a code repository containing the Dockerfile/docker-compose.yml file(s)

Breakout/Escape, Extract Secrets

Escalate Privileges, Extract Secrets

Move Laterally, Extract Secrets

Source Code Review, Extract Secrets

# Where’s me in a Docker environment?

- **Inside a container**
- On a host running container(s)
- In a network with other host(s) exposing Docker API
- In a code repository containing the Dockerfile/docker-compose.yml file(s)

# How to be sure if it’s a Docker container?

1. */.dockerenv* there?
2. */proc/1/cgroup* has */docker/* in it?
3. Few processes with PID 1 != an init system?
4. *sudo*, *ifconfig*, *netstat*, *ss* there?

# How to be sure if it’s a Docker container?

1. */.do*Default file* there?
2. */proc/1/cgroup* has */docker/* in it?
3. Few processes with PID 1 != an init system?
4. *sudo*, *ifconfig*, *netstat*, *ss* there?

# How to be sure if it’s a Docker container?

1. */.dockerenv* there?
2. */proc/1/cgroup* has *cgroup* in it?
3. Few processes with PID 1 != an init system?
4. *sudo*, *ifconfig*, *netstat*, *ss* there?

# How to be sure if it’s a Docker container?

1. */.do*Default file* there?
2. */proc/1/cgroup* has *Control Group* in it?
3. Few processes with P*PID 1 is the ENTRYPOINT process by default*
4. *sudo*, *ifconfig*, *netstat*, *ss* there?

# How to be sure if it’s a Docker container?

1. */.do* there?
2. */proc/1/cgroup* has in it?
3. Few processes with P
4. sudo, if

Default file

Control Group

PID 1 is the ENTRYPOINT process by default

Stripper down images for a smaller footprint

# What to know about a Docker container before exploiting it?

1. Are you root?
2. Which OS environment is running inside the container?
3. Which OS & kernel are running on the host?
4. Which processes are running?
5. Which capabilities do the processes in the container have?
6. Is the container privileged?
7. What volumes are mounted?
8. What's in the environmental variable?
9. Is the Docker Socket mounted?

# What to know about a Docker container before exploiting it?

1. *To escape a restricted zone, own it first*
2. Which OS environment is running inside the container?
3. Which OS & kernel are running on the host?
4. Which processes are running?
5. Which capabilities do the processes in the container have?
6. Is the container privileged?
7. What volumes are mounted?
8. What's in the environmental variable?
9. Is the Docker Socket mounted?

# What to know about a Docker container before exploiting it?

1. To escape a restricted zone, own it first
2. Which (Aids in performing tasks)s running inside the container?
3. Which OS & kernel are running on the host?
4. Which processes are running?
5. Which capabilities do the processes in the container have?
6. Is the container privileged?
7. What volumes are mounted?
8. What's in the environmental variable?
9. Is the Docker Socket mounted?

# What to know about a Docker container before exploiting it?

1. To escape a restricted zone, own it first
2. Which (Aids in performing tasks)s running inside the container?
3. Which O(Public kernel exploits for breakouts)n the host?
4. Which processes are running?
5. Which capabilities do the processes in the container have?
6. Is the container privileged?
7. What volumes are mounted?
8. What's in the environmental variable?
9. Is the Docker Socket mounted?

# What to know about a Docker container before exploiting it?

1. To escape a restricted zone, own it first
2. Which (Aids in performing tasks)s running inside the container?
3. Which O(Public kernel exploits for breakouts)n the host?
4. Which proce(What to look look for and where?)
5. Which capabilities do the processes in the container have?
6. Is the container privileged?
7. What volumes are mounted?
8. What's in the environmental variable?
9. Is the Docker Socket mounted?

# What to know about a Docker container before exploiting it?

1. To escape a restricted zone, own it first
2. Which (Aids in performing tasks)s running inside the container?
3. Which O(Public kernel exploits for breakouts)n the host?
4. Which proce(What to look look for and where?)
5. Which capabilities do th(Potential breakouts)in the container have?
6. Is the container privileged?
7. What volumes are mounted?
8. What's in the environmental variable?
9. Is the Docker Socket mounted?

# What to know about a Docker container before exploiting it?

1. To escape a restricted zone, own it first
2. Which (Aids in performing tasks)s running inside the container?
3. Which O(Public kernel exploits for breakouts)n the host?
4. Which proce(What to look look for and where?)
5. Which capabilities do th(Potential breakouts)in the container have?
6. Is the c(Potential breakouts)leged?
7. What volumes are mounted?
8. What's in the environmental variable?
9. Is the Docker Socket mounted?

# What to know about a Docker container before exploiting it?

1. To escape a restricted zone, own it first
2. Which (Aids in performing tasks)s running inside the container?
3. Which O(Public kernel exploits for breakouts)n the host?
4. Which proce(What to look look for and where?)
5. Which capabilities do th(Potential breakouts)in the container have?
6. Is the c(Potential breakouts)leged?
7. What volum(Persistent data store with sensitive information)
8. What's in the environmental variable?
9. Is the Docker Socket mounted?

# What to know about a Docker container before exploiting it?

1. To escape a restricted zone, own it first
2. Which (Aids in performing tasks)s running inside the container?
3. Which O(Public kernel exploits for breakouts)n the host?
4. Which proce(What to look look for and where?)
5. Which capabilities do th(Potential breakouts)in the container have?
6. Is the c(Potential breakouts)leged?
7. What volum(Persistent data store with sensitive information)
8. What's in the environmental varia(Secrets)
9. Is the Docker Socket mounted?

# What to know about a Docker container before exploiting it?

1. To escape a restricted zone, own it first
2. Which (Aids in performing tasks)s running inside the container?
3. Which O(Public kernel exploits for breakouts)n the host?
4. Which proce(What to look look for and where?)
5. Which capabilities do th(Potential breakouts)in the container have?
6. Is the c(Potential breakouts)leged?
7. What volum(Persistent data store with sensitive information)
8. What's in the environmental varia(Secrets)
9. Is the D(Potential breakouts) mounted?

# How to exploit a Docker Container…

1. …with CAP_SYS_ADMIN capability
2. …with CAP_DAC_READ_SEARCH capability
3. …with CAP_DAC_READ_SEARCH & CAP_DAC_OVERRIDE capabilities
4. …with CAP_SYS_PTRACE capability & shared PID namespace with host
5. …with CAP_SYS_MODULE capability
6. …which is privileged
7. …with mounted Docker Socket

?

# QUESTIONS?

Thank you