



K-Shield Jr 10기

도커 취약점 수동 점검 결과 리포트

취약점 진단 E-01조

표재경
김서연
김효민
전동현
황유림

목 차

진단코드	진단항목	취약도
D-01	도커최신 패치 적용	상
D-02	Docker daemon audit 설정	상
D-03	/var/lib/docker audit 설정	상
D-04	/etc/docker audit 설정	상
D-05	docker.service audit 설정	상
D-06	docker.socket audit 설정	상
D-07	/etc/default/docker audit 설정	상
D-08	default bridge를 통한 컨테이너 간 네트워크 트래픽 제한	상
D-09	docker.service 소유권 설정	상
D-10	docker.service 파일 접근권한 설정	상
D-11	docker.socket 소유권 설정	상
D-12	docker.socket 파일 접근권한 설정	상
D-13	/etc/docker 디렉터리 소유권 설정	상
D-14	/etc/docker 디렉터리 접근권한 설정	상
D-15	/var/run/docker.sock 파일 소유권 설정	상
D-16	/var/run/docker.sock 접근 권한 설정	상
D-17	daemon.json 파일 소유권 설정	상
D-18	daemon.json 접근 권한 설정	상
D-19	/etc/default/docker 파일 소유권 설정	상
D-20	/etc/default/docker 접근 권한 설정	상

D-21	컨테이너에서 ssh 사용 금지	상
D-22	호스트 OS 주요 자원 접근 제어	상
D-23	인증-권한 제어	상
D-24	SSL/TLS 적용	상
D-25	컨테이너 권한 제어	상
D-26	인증제어	상
D-27	SSL/TLS 적용	상
D-28	도커 그룹에 불필요한 사용자 제거	중
D-29	legacy registry (v1) 비활성화	하
D-30	추가 권한 획득으로부터 컨테이너 제한	중
D-31	root가 아닌 user로 컨테이너 실행	중
D-32	도커를 위한 콘텐츠 신뢰성 활성화	중
D-33	컨테이너 SELinux 보안 옵션 설정	중
D-34	컨테이너에서 privileged 포트 매핑 금지	중
D-35	도커의 default bridge docker() 사용 제한	하
D-36	호스트의 user namespaces 공유제한	하
D-37	컨테이너 보안 정책	중
D-38	로그 관리	하
D-39	Dockerfile Config	중
D-40	이미지 취약점 및 구성 결합	중
D-41	네트워크 제어	중

진단코드	D-01	참고	kisa DO-01 / sk 5.3
진단항목명	도커 최신 패치 적용	취약도	kisa 상 / sk 중
진단기준			
양호	알려진 취약점이 없는 버전을 사용하는 경우		
취약	알려진 취약점이 존재하는 버전을 사용하는 경우		
진단방법			

도커 버전 확인 : `$ docker version`

우분투 혹은 데비안의 경우 패키지 버전 확인 : `$dpkg -i | grep docker.io`

CentOS의 경우 패키지 버전 확인 : `$rpm -qa | grep docker.io`

진단결과	
<pre>C:\Users\# >docker version Client: Cloud integration: v1.0.29 Version: 20.10.20 API version: 1.41 Go version: go1.18.7 Git commit: 9fdeb9c Built: Tue Oct 18 18:28:44 2022 OS/Arch: windows/amd64 Context: default Experimental: true</pre>	<pre>Server: Docker Desktop 4.13.1 (9034b) Engine: Version: 20.10.20 API version: 1.41 (minimum version 1.12) Go version: go1.18.7 Git commit: 03df974 Built: Tue Oct 18 18:18:35 2022 OS/Arch: linux/amd64 Experimental: false containerd: Version: 1.6.8 GitCommit: 9cd3357b7fd7218e4aec3eae239db1f68a5a6ec6 runc: Version: 1.1.4 GitCommit: v1.1.4-0-g5fd4c4d docker-init: Version: 0.19.0 GitCommit: de40ad0</pre>

비고

※ Docker Release Note 사이트 :

<https://docs.docker.com/release-notes/> (Docker 공식 홈페이지)

진단코드	D-02	참고	kisa DO-03 / sk 2.2
진단항목명	docker daemon audit 설정	취약도	kisa 상 / sk 하
진단기준			
양호	/usr/bin/docker 파일의 감사 설정이 적용되어 있는 경우		
취약	/usr/bin/docker 파일의 감사 설정이 적용되어 있지 않은 경우		
진단방법			

/usr/bin/docker 감사 설정 확인 : `auditctl -l | grep /usr/bin/docker`

/etc/audit/audit.rules 파일 내용 확인 : `cat /etc/audit/audit.rules | grep /usr/bin/docker`

진단결과	
<pre>root@docker:/home/e01# auditctl -l grep /usr/bin/docker</pre>	<pre>root@docker:/home/e01# cat /etc/audit/audit.rules grep /usr/bin/docker</pre>

비고

auditd가 설치되어 있어야 함 : `$ sudo apt-get install auditd`

진단코드	D-03	참고	kisa DO-04 / sk 2.2
진단항목명	/var/lib/docker audit 설정	취약도	kisa 상 / sk 하
진단기준			
양호	/var/lib/docker 디렉터리의 감사 설정이 적용되어 있는 경우		
취약	/var/lib/docker 디렉터리의 감사 설정이 적용되어 있지 않은 경우		
진단방법			

/var/lib/docker 감사 설정 확인 : auditctl -l | grep /var/lib/docker
 /etc/audit/audit.rules 파일 내용 확인 : cat /etc/audit/audit.rules | grep /var/lib/docker

진단결과

```
root@docker:/home/e01# auditctl -l | grep /var/lib/docker
root@docker:/home/e01# cat /etc/audit/audit.rules | grep /var/lib/docker
```

비고

-

진단코드	D-04	참고	kisa DO-05 / sk 2.2
진단항목명	/etc/docker audit 설정	취약도	kisa 상 / sk 하
진단기준			
양호	/etc/docker 디렉터리의 감사 설정이 적용되어 있는 경우		
취약	/etc/docker 디렉터리의 감사 설정이 적용되어 있지 않은 경우		
진단방법			

/etc/docker 감사 설정 확인 : auditctl -l | grep /etc/docker
 /etc/audit/audit.rules 파일 내용 확인 : cat /etc/audit/audit.rules | grep /etc/docker

진단결과

```
root@docker:/home/e01# auditctl -l | grep /etc/docker
root@docker:/home/e01# cat /etc/audit/audit.rules | grep /etc/docker
```

비고

-

진단코드	D-05	참고	kisa DO-06 / sk 2.2
진단항목명	docker.service audit 설정	취약도	kisa 상 / sk 하
진단기준			
양호	docker.service 파일의 감사 설정이 적용되어 있는 경우		
취약	docker.service 파일의 감사 설정이 적용되어 있지 않은 경우		
진단방법			

docker.service 파일의 경로 확인 \$ systemctl show -p FragmentPath docker.service
 docker.service 감사 설정 확인 : auditctl -l | grep docker.service
 /etc/audit/audit.rules 파일 내용 확인 : cat /etc/audit/audit.rules | grep docker.service

진단결과

```
root@docker:/home/e01# auditctl -l | grep docker.service
root@docker:/home/e01# cat /etc/audit/audit.rules | grep /docker.service
```

비고

※ 이 파일은 시스템에 없을 수 있다. 그런 경우에는 권장 사항이 적용되지 않음.

진단코드	D-06	참고	kisa DO-07 / sk 2.2
진단항목명	docker.socket audit 설정	취약도	kisa 상 / sk 하
진단기준			
양호	docker.socket 파일의 감사 설정이 적용되어 있는 경우		
취약	docker.socket 파일의 감사 설정이 적용되어 있지 않은 경우		
진단방법			

docker.socket 파일의 경로 확인 \$ systemctl show -p FragmentPath docker.socket
 docker.socket 감사 설정 확인 : auditctl -l | grep docker.socket
 /etc/audit/audit.rules 파일 내용 확인 : cat /etc/audit/audit.rules | grep docker.socket

진단결과

```
root@docker:/home/e01# auditctl -l | grep docker.socket
root@docker:/home/e01# cat /etc/audit/audit.rules | grep /docker.socket
```

비고

※ 이 파일은 시스템에 없을 수 있다. 그런 경우에는 권장 사항이 적용되지 않음.

진단코드	D-07	참고	kisa DO-08 / sk 2.2
진단항목명	/etc/default/docker audit 설정	취약도	kisa 상 / sk 하
진단기준			
양호	/etc/default/docker 파일의 감사 설정이 적용되어 있는 경우		
취약	/etc/default/docker 파일의 감사 설정이 적용되어 있지 않은 경우		
진단방법			

/etc/default/docker 감사 설정 확인 : auditctl -l | grep /etc/default/docker
 /etc/audit/audit.rules 파일 내용 확인 : cat /etc/audit/audit.rules | grep
 /etc/default/docker

진단결과

비고

진단코드	D-08	참고	kisa DO-09 / sk 1.5
진단항목명	default bridge를 통한 컨테이너 간 네트워크 트래픽 제한	취약도	kisa 상 / sk 중
진단기준			
양호	컨테이너 간 네트워크 통신이 가능하지 않은 경우		
취약	컨테이너 간 네트워크 통신이 가능한 경우		
진단방법			

컨테이너 간 네트워크 통신 제한 옵션이 적용되어 있는지 확인 :
 \$docker network ls --quiet | xargs docker network inspect --format '{{.Name}}: {{.Options}}'

진단결과

```
root@docker:~# docker network ls --quiet | xargs docker network inspect --format '{{.Name}}: {{.Options}}'
bridge: map[com.docker.network.bridge.default_bridge:true com.docker.network.bridge.enable_icc:true com.docker.network.bridge.enable_ip_masquerade:true com.docker.network.bridge.host_binding_ipv4:0.0.0.0 com.docker.network.bridge.name:docker0 com.docker.network.driver.mtu:1500]
host: map[]
none: map[]
```

비고

/etc/default/docker를 사용할 경우 systemd를 사용하는 OS에서는 바로 적용되지 않으므로 docker.service를 수정하여 사용해야 함

진단코드	D-09	참고	kisa DO-13 / sk 2.1
진단항목명	docker.service 소유권 설정	취약도	kisa 상 / sk 하
진단기준			
양호	docker.service 파일의 소유자 및 소유그룹이 root:root인 경우		
취약	docker.service 파일의 소유자 및 소유그룹이 root:root가 아닌 경우		
진단방법			

파일 경로 확인 : `systemctl show -p FragmentPath docker.service`
 소유권 확인 : `ls -l /lib/systemd/system/docker.service`
`stat -c %U:%G /lib/systemd/system/docker.service`

진단결과

```
root@docker:~# systemctl show -p FragmentPath docker.service
FragmentPath=/lib/systemd/system/docker.service
root@docker:~# ls -l /lib/systemd/system/docker.service
-rw-r--r-- 1 root root 1730 Apr 14 10:32 /lib/systemd/system/docker.service
root@docker:~# stat -c %U:%G /lib/systemd/system/docker.service
root:root
```

비고

※ 이 파일은 시스템에 없을 수 있다. 그런 경우에는 권장 사항이 적용되지 않음.

진단코드	D-10	참고	kisa DO-14 / sk 2.1
진단항목명	docker.service 파일 접근권한 설정	취약도	kisa 상 / sk 하
진단기준			
양호	docker.service 파일의 접근 권한이 644 이하인 경우		
취약	docker.service 파일의 접근 권한이 644 초과인 경우		
진단방법			

파일 경로 확인 : `systemctl show -p FragmentPath docker.service`
 접근권한 확인 : `ls -l /lib/systemd/system/docker.service`
`stat -c %a/lib/systemd/system/docker.service`

진단결과

```
root@docker:~# ls -l /lib/systemd/system/docker.service
-rw-r--r-- 1 root root 1730 Apr 14 10:32 /lib/systemd/system/docker.service
root@docker:~# stat -c %a /lib/systemd/system/docker.service
644
```

비고

※ 이 파일은 시스템에 없을 수 있다. 그런 경우에는 권장 사항이 적용되지 않음.

진단코드	D-11	참고	kisa DO-15 / sk 2.1
진단항목명	docker.socket 소유권 설정	취약도	kisa 상 / sk 하
진단기준			
양호	docker.socket 파일의 소유자 및 소유그룹이 root:root인 경우		
취약	docker.socket 파일의 소유자 및 소유그룹이 root:root가 아닌 경우		
진단방법			

파일 경로 확인 : systemctl show -p FragmentPath docker.socket
 소유권 확인 : ls -l /lib/systemd/system/docker.socket
 stat -c %U:%G /lib/systemd/system/docker.socket

진단결과

```
root@docker:~# systemctl show -p FragmentPath docker.socket
FragmentPath=/lib/systemd/system/docker.socket
root@docker:~# ls -l /lib/systemd/system/docker.socket
-rw-r--r-- 1 root root 295 Apr 14 10:32 /lib/systemd/system/docker.socket
root@docker:~# stat -c %U:%G /lib/systemd/system/docker.socket
root:root
```

비고

-

진단코드	D-12	참고	kisa DO-16 / sk 2.1
진단항목명	docker.socket 파일 접근권한 설정	취약도	kisa 상 / sk 하
진단기준			
양호	docker.socket 파일의 접근 권한이 644 이하인 경우		
취약	docker.socket 파일의 접근 권한이 644 초과인 경우		
진단방법			

파일 경로 확인 : systemctl show -p FragmentPath docker.socket
 접근권한 확인 : ls -l /lib/systemd/system/docker.socket
 stat -c %a /lib/systemd/system/docker.socket

진단결과

```
root@docker:~# systemctl show -p FragmentPath docker.socket
FragmentPath=/lib/systemd/system/docker.socket
root@docker:~# ls -l /lib/systemd/system/docker.socket
-rw-r--r-- 1 root root 295 May 5 21:17 /lib/systemd/system/docker.socket
root@docker:~# stat -c %a /lib/systemd/system/docker.socket
644
```

비고

-

진단코드	D-13	참고	kisa DO-17 / sk 2.1
진단항목명	/etc/docker 디렉터리 소유권 설정	취약도	kisa 상 / sk 하
진단기준			
양호	/etc/docker 파일의 소유자 및 소유그룹이 root:root인 경우		
취약	/etc/docker 파일의 소유자 및 소유그룹이 root:root가 아닌 경우		
진단방법			

파일 경로 확인 : systemctl show -p FragmentPath /etc/docker
 소유권 확인 : ls--ld /etc/docker
 stat -c %U:%G /etc/docker

진단결과

```
root@docker:~# ls -ld /etc/docker
drwxr-xr-x 2 root root 4096 Apr 14 10:32 /etc/docker
root@docker:~# stat -c %U:%G /etc/docker
root:root
```

비고

-

진단코드	D-14	참고	kisa DO-18 / sk 2.1
진단항목명	/etc/docker 디렉터리 접근권한 설정	취약도	kisa 상 / sk 하
진단기준			
양호	/etc/docker 파일의 접근 권한이 755 이하인 경우		
취약	/etc/docker 파일의 접근 권한이 755 초과인 경우		
진단방법			

파일 경로 확인 : systemctl show -p FragmentPath /etc/docker
 접근권한 확인 : ls--ld /etc/docker
 stat -c %a /etc/docker

진단결과

```
root@docker:~# ls -ld /etc/docker
drwxr-xr-x 2 root root 4096 Apr 14 10:32 /etc/docker
root@docker:~# stat -c %a /etc/docker
755
```

비고

-

진단코드	D-15	참고	kisa DO-19 / sk 2.1
진단항목명	/var/run/docker.sock 파일 소유권 설정	취약도	kisa 상 / sk 하
진단기준			
양호	/var/run/docker.sock파일의 소유자 및 소유그룹이 root:root인 경우		
취약	/var/run/docker.sock파일의 소유자 및 소유그룹이 root:root가 아닌 경우		
진단방법			

파일 경로 확인 : systemctl show -p FragmentPath /var/run.docker.sock
 소유권 확인 : ls -l /var/run/docker.sock
 stat -c %U:%G /var/run/docker.sock

진단결과

```
root@docker:~# ls -l /var/run/docker.sock
srw-rw---- 1 root docker 0 Apr 23 03:05 /var/run/docker.sock
root@docker:~# stat -c %U:%G /var/run/docker.sock
root:docker
```

비고

-

진단코드	D-16	참고	kisa DO-20 / sk 2.1
진단항목명	/var/run/docker.sock 접근 권한 설정	취약도	kisa 상 / sk 하
진단기준			
양호	/var/run/docker.sock 파일의 접근 권한이 660 이하인 경우		
취약	/var/run/docker.sock 파일의 접근 권한이 660 초과인 경우		
진단방법			

파일 경로 확인 : systemctl show -p FragmentPath /var/run.docker.sock
 접근권한 확인 : ls -l /var/run/docker.sock
 stat -c %a /var/run/docker.sock

진단결과

```
root@docker:~# ls -l /var/run/docker.sock
srw-rw---- 1 root docker 0 May 15 13:40 /var/run/docker.sock
root@docker:~# stat -c %a /var/run/docker.sock
660
```

비고

-

진단코드	D-17	참고	kisa DO-21 / sk 2.1
진단항목명	daemon.json 파일 소유권 설정	취약도	kisa 상 / sk 하
진단기준			
양호	/etc/docker/daemon.json파일의 소유자 및 소유그룹이 root:root인 경우		
취약	/etc/docker/daemon.json파일의 소유자 및 소유그룹이 root:root가 아닌 경우		
진단방법			

소유권 확인 : ls -l /etc/docker/daemon.json
stat -c %U:%G /etc/docker/daemon.json

진단결과

```
root@docker:~# ls -l /etc/docker/daemon.json
ls: cannot access '/etc/docker/daemon.json': No such file or directory
```

비고

※ 이 파일은 시스템에 없을 수 있다. 그런 경우 권장 사항이 적용되지 않음

진단코드	D-18	참고	kisa DO-22 / sk 2.1
진단항목명	daemon.json 접근 권한 설정	취약도	kisa 상 / sk 하
진단기준			
양호	/etc/docker/daemon.json 파일의 접근 권한이 644 이하인 경우		
취약	/etc/docker/daemon.json 파일의 접근 권한이 644 초과인 경우		
진단방법			

소유권 확인 : ls -l /etc/docker/daemon.json
stat -c %a /etc/docker/daemon.json

진단결과

```
root@docker:~# ls -l /etc/docker/daemon.json
ls: cannot access '/etc/docker/daemon.json': No such file or directory
```

비고

※ 이 파일은 시스템에 없을 수 있다. 그런 경우 권장 사항이 적용되지 않음

진단코드	D-19	참고	kisa DO-23 / sk 2.1
진단항목명	/etc/default/docker 파일 소유권 설정	취약도	kisa 상 / sk 하
진단기준			
양호	/etc/default/docker 파일의 소유자 및 소유그룹이 root:root인 경우		
취약	/etc/default/docker 파일의 소유자 및 소유그룹이 root:root가 아닌 경우		
진단방법			

소유권 확인 : ls -l /etc/default/docker
stat -c %U:%G /etc/default/docker

진단결과

```
root@docker:~# ls -l /etc/default/docker
-rw-r--r-- 1 root root 654 Apr 14 10:30 /etc/default/docker
root@docker:~# stat -c %U:%G /etc/default/docker
root:root
```

비고

※ 이 파일은 시스템에 없을 수 있다. 그런 경우 권장 사항이 적용되지 않음

진단코드	D-20	참고	kisa DO-24 / sk 2.1
진단항목명	/etc/default/docker 접근 권한 설정	취약도	kisa 상 / sk 하
진단기준			
양호	/etc/default/docker 파일의 접근 권한이 644 이하인 경우		
취약	/etc/default/docker 파일의 접근 권한이 644 초과인 경우		
진단방법			

접근 권한 확인 : ls -l /etc/default/docker
stat -c %a /etc/default/docker

진단결과

```
root@docker:~# ls -l /etc/default/docker
-rw-r--r-- 1 root root 654 Apr 14 10:30 /etc/default/docker
root@docker:~# stat -c %a /etc/default/docker
644
```

비고

※ 이 파일은 시스템에 없을 수 있다. 그런 경우 권장 사항이 적용되지 않음

진단코드	D-21	참고	kisa DO-28 / sk 1.5
진단항목명	컨테이너에서 ssh 사용 금지	취약도	kisa 상 / sk 중
진단기준			
양호	컨테이너에 ssh가 비활성화 되어 있는 경우		
취약	컨테이너에 ssh가 활성화 되어 있는 경우		
진단방법			

실행중인 컨테이너 목록 확인: \$docker ps --quiet

실행중인 컨테이너의 활성화된 서비스 확인 : \$docker exec \$INSTANCE_ID ps -el

진단결과	
<pre>root@docker:/home/e01# docker ps --quiet 7e29df341bfe root@docker:/home/e01# docker exec 7e29df341bfe ps -el OCI runtime exec failed: exec failed: unable to start container process: exec: "ps": executable file not found in \$PATH: unknown root@docker:/home/e01# _</pre>	
비고	
-	

진단코드	D-22	참고	sk 1.1
진단항목명	호스트 OS 주요 자원 접근 제어	취약도	상
진단기준			
양호	주요 시스템 디렉터리 마운트와 호스트 장치 파일 컨테이너 직접 노출이 모두 금지되어 있을 경우		
취약	주요 시스템 디렉터리 마운트와 호스트 장치 파일 컨테이너 직접 노출이 모두 금지되어 있지 않을 경우		
진단방법			

각 컨테이너에 매핑된 디렉터리의 목록과 권한 확인 : docker ps --quiet --all | xargs docker inspect --format '{{.ID}}:Volumes={{.Mounts}};'

컨테이너 내 불필요하게 접근 가능한 호스트 장치의 존재 여부 및 필요에 의해 접근 가능한 장치에 대한 권한이 올바르게 설정되어 있는지 확인 : docker ps --quiet --all | xargs docker inspect --format '{{.Id }}: Devices={{.HostConfig.Devices }}'

진단결과	
<pre>root@docker:~# docker ps --quiet --all xargs docker inspect --format '{{.Id}}: Volumes={{.Mounts}};' e61de8ef1a41b5e56c96ebc9f9b51722b282623521e704a470e5f13699e2917c: Volumes= {} 5e20e28243975a6b86ac79964c3eb6c14a83f24c64adeccf2b95073dd72a0fe6: Volumes= {} a4fe7fb8fc56dcb5e16f3dbdc1cd9c24c5e54bc2935b847382e156ae0994c158: Volumes= {} root@docker:~# docker ps --quiet --all xargs docker inspect --format '{{.Id}}: Devices={{.HostConfig.Devices}};' e61de8ef1a41b5e56c96ebc9f9b51722b282623521e704a470e5f13699e2917c: Devices= {} 5e20e28243975a6b86ac79964c3eb6c14a83f24c64adeccf2b95073dd72a0fe6: Devices= {} a4fe7fb8fc56dcb5e16f3dbdc1cd9c24c5e54bc2935b847382e156ae0994c158: Devices= {} root@docker:~#</pre>	
비고	

※ 서비스 상 필요할 경우 예외처리 요청 / 장기 적용(적용 시 개발 및 운영자 협의)

진단코드	D-23	참고	sk 1.2
진단항목명	인증-권한 제어	취약도	상
진단기준			
양호	docker group 내 신뢰하지 않는 사용자가 존재하지 않을 경우		
취약	docker group 내 신뢰하지 않는 사용자가 존재할 경우		
진단방법			

docker group 내 사용자 확인 : getent group docker
authorization-plugin 사용 및 설정값 확인 : ps -ef | grep dockerd

진단결과

```
root@docker:~# getent group docker
docker:x:999:
root@docker:~# ps -ef | grep dockerd
root      1292      1  0 11:34 ?        00:00:05 /usr/bin/dockerd -H fd:// --containerd=/run/contai
nerd/containerd.sock
root      4045     4015  0 12:00 tty1    00:00:00 grep --color=auto dockerd
root@docker:~# _
```

비고

※ 장기 적용(적용 시 개발자 및 운영자 협의)

진단코드	D-24	참고	sk 1.3
진단항목명	SSL/TLS 적용	취약도	상
진단기준			
양호	SSL/TLS 가 적용되어 있을 경우		
취약	SSL/TLS 가 적용되어 있지 않을 경우		
진단방법			

--tlsverify --tlscacert --tlscert --tlskey 사용 여부 확인 : ps -ef | grep dockerd

진단결과

```
root@docker:~# ps -ef | grep dockerd
root      1292      1  0 11:36 ?        00:00:06 /usr/bin/dockerd -H fd:// --containerd=/run/contai
nerd/containerd.sock
root      4190     4015  0 12:05 tty1    00:00:00 grep --color=auto dockerd
```

비고

※ 중기 적용(적용 시 개발자 및 운영자 협의)

진단코드	D-25	참고	sk 1.7
진단항목명	컨테이너 권한 제어	취약도	상
진단기준			
양호	컨테이너에 불필요하게 과도한 권한이 설정되어 있지 않은 경우		
취약	컨테이너에 불필요하게 과도한 권한이 설정되어 있는 경우		
진단방법			

```

suid/sgid 제한 : ps -ef | grep dockerd
docker ps --quiet --all | xargs docker inspect --format '{{ .Id }}: SecurityOpt={{
    .HostConfig.SecurityOpt }}'
실험(Experimental) 기능 비활성화 : docker version --format '{{ .Server.Experimental }}'
cgroup 변경 금지 : ps -ef | grep dockerd
docker ps --quiet --all | xargs docker inspect --format '{{ .Id }}: CgroupParent={{
    .HostConfig.CgroupParent }}'
-privilege 컨테이너 사용 제한 : docker ps --quiet --all | xargs docker inspect --format '{{ .Id }}:
    Privileged={{ .HostConfig.Privileged }}'
exec 사용 제한 : ausearch -k docker | grep exec | grep privileged
ausearch -k docker | grep exec | grep user docker ps --quiet --all | xargs docker inspect
--format '{{ .Id }}: ReadonlyRootfs={{ .HostConfig.ReadonlyRootfs }}'
    
```

진단결과
<pre> root@docker:~# ps -ef grep dockerd root 1292 1 0 10:22 ? 00:00:51 /usr/bin/dockerd -H fd:// --containerd=/run/contai nerd/containerd.sock root 16578 4015 0 12:49 tty1 00:00:00 grep --color=auto dockerd </pre>
<pre> root@docker:~# docker ps --quiet --all xargs docker inspect --format '{{ .Id }}: SecurityOpt={{ .HostConfig.SecurityOpt }}' e61de8ef1a41b5e56c96ebc9f9b51722b282623521e704a470e5f13699e2917c: SecurityOpt=<no value> 5e20e28243975a6b86ac79964c3eb6c14a83f24c64adeccf2b95073dd72a0fe6: SecurityOpt=<no value> a4fe7fb8fc56dcb5e16f3dbdc1cd9c24c5e54bc2935b847382e156ae0994c158: SecurityOpt=<no value> </pre>
<pre> root@docker:~# docker version --format '{{ .Server.Experimental }}' false </pre>
<pre> root@docker:~# ps -ef grep dockerd root 1292 1 0 10:22 ? 00:00:51 /usr/bin/dockerd -H fd:// --containerd=/run/contai nerd/containerd.sock root 16578 4015 0 12:49 tty1 00:00:00 grep --color=auto dockerd </pre>
<pre> root@docker:~# docker ps --quiet --all xargs docker inspect --format '{{ .Id }}: CgroupParent={{ .HostConfig.CgroupParent }}' e61de8ef1a41b5e56c96ebc9f9b51722b282623521e704a470e5f13699e2917c: CgroupParent= 5e20e28243975a6b86ac79964c3eb6c14a83f24c64adeccf2b95073dd72a0fe6: CgroupParent= a4fe7fb8fc56dcb5e16f3dbdc1cd9c24c5e54bc2935b847382e156ae0994c158: CgroupParent= </pre>
<pre> root@docker:~# docker ps --quiet --all xargs docker inspect --format '{{ .Id }}: Privileged={{ .HostConfig.Privileged }}' e61de8ef1a41b5e56c96ebc9f9b51722b282623521e704a470e5f13699e2917c: Privileged=false 5e20e28243975a6b86ac79964c3eb6c14a83f24c64adeccf2b95073dd72a0fe6: Privileged=false a4fe7fb8fc56dcb5e16f3dbdc1cd9c24c5e54bc2935b847382e156ae0994c158: Privileged=false </pre>
<pre> root@docker:~# ausearch -k docker grep exec grep privileged <no matches> root@docker:~# ausearch -k docker grep exec grep user <no matches> </pre>
<pre> root@docker:~# docker ps --quiet --all xargs docker inspect --format '{{ .Id }}: ReadonlyRootfs={{ .HostConfig.ReadonlyRootfs }}' e61de8ef1a41b5e56c96ebc9f9b51722b282623521e704a470e5f13699e2917c: ReadonlyRootfs=false 5e20e28243975a6b86ac79964c3eb6c14a83f24c64adeccf2b95073dd72a0fe6: ReadonlyRootfs=false a4fe7fb8fc56dcb5e16f3dbdc1cd9c24c5e54bc2935b847382e156ae0994c158: ReadonlyRootfs=false </pre>

비고

중기 적용(적용 시 개발자 및 운영자 협의)

진단코드	D-26	참고	sk 4.1
진단항목명	인증제어	취약도	상
진단기준			
양호	불필요한 swarm 모드 및 관리자 노드가 비활성화/최소화되어 있을 경우		
취약	불필요한 swarm 모드 및 관리자 노드가 비활성화/최소화되어 있지 않을 경우		
진단방법			
swarm mode 불필요하게 활성화 금지 : docker info			
관리자 노드 최소화 : docker info --format '{{.Swarm.Managers}}'			
자동 잠금 모드 사용 : docker swarm unlock-key			
진단결과			
<pre>root@ubuntu:/home/yurim# docker info_ Swarm: inactive WARNING: NO swap limit support root@ubuntu:/home/yurim# docker info --format '{{.Swarm.Managers}}' [] root@ubuntu:/home/yurim# docker swarm unlock key could not fetch unlock key: Error response from daemon: This node is not a swarm manager. Use "docker r swarm init" or "docker swarm join" to connect this node to swarm and try again. root@ubuntu:/home/yurim# sudo docker swarm unlock-key</pre>			
비고			
장기 적용(적용 시 개발자 및 운영자 협의)			

진단코드	D-27	참고	sk 4.2
진단항목명	SSL/TLS 적용	취약도	상
진단기준			
양호	SSL/TLS이 적용되어 있으며 주기적으로 인증서가 관리되고 있을 경우		
취약	SSL/TLS이 적용되어 있으며 주기적으로 인증서가 관리되지 않는 경우		
진단방법			
SSL/TLS 적용을 통한 네트워크 구간 데이터 보호 및 사용자 인증 : docker network ls --filter driver=overlay --quiet xargs docker network inspect --format '{{.Name}} {{.Options}}'			
인증서 관리(인증서 교환주기 설정) : docker info grep "Expiry Duration"			
CA인증서 교환주기 확인 : ls -l /var/lib/docker/swarm/certificates/swarm-root-ca.crt			
진단결과			

진단결과

```
root@ubuntu:/home/yurim# docker network ls --filter driver=overlay --quiet | xargs docker network inspect --format '{{.Name}} {{.Options}}'
unknown flag: --quiet
See 'docker network --help'.

Usage: docker network COMMAND

Manage networks

Commands:
  connect    Connect a container to a network
  create     Create a network
  disconnect Disconnect a container from a network
  inspect    Display detailed information on one or more networks
  ls        List networks
  prune     Remove all unused networks
  rm        Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.

'docker network inspect' requires at least 1 argument.
See 'docker network inspect --help'.

Usage: docker network inspect [OPTIONS] NETWORK [NETWORK...]
Display detailed information on one or more networks
```

```
root@ubuntu:/home/yurim# docker info | grep "Expiry Duration"
WARNING: No swap limit support
```

```
root@ubuntu:/home/yurim# ls -l /var/lib/docker/swarm/certificates/swarm-root-ca.crt
ls: cannot access '/var/lib/docker/swarm/certificates/swarm-root-ca.crt': No such file or directory
```

비고

중기 적용(적용 시 개발자 및 운영자 협의)

진단코드	D-28	참고	kisa DO-02
진단항목명	도커 그룹에 불필요한 사용자 제거	취약도	중
진단기준			
양호	도커 그룹에 불필요한 사용자가 존재하지 않는 경우		
취약	도커 그룹에 불필요한 사용자가 존재하는 경우		
진단방법			

도커 그룹에 속한 사용자 계정 조회 : \$cat/etc/group | grep docker
 도커 그룹 이름이 dockerrott 일 경우 root 및 dockerroot 그룹에 속한 사용자 계정 동시
 조회 : \$ cat /etc/group | grep root

진단결과

```
root@docker:/home/e01# cat /etc/group | grep docker
```

```
root@docker:/home/e01# cat /etc/group | grep root
```

비고

진단코드	D-29	참고	kisa DO-11 / sk 3.3
진단항목명	legacy registry (v1) 비활성화	취약도	kisa 하 / sk 중
진단기준			
양호	legacy registry v1이 비활성화 되어 있는 경우		
취약	legacy registry v1이 비활성화 되어 있지 않은 경우		
진단방법			

--disable-legacy-registry 옵션이 적용되어 있는지 확인 : `$ps -ef | grep docker`
 /etc/default/docker 파일에서 `—disable-legacy-registry` 옵션이 적용되어 있는지 확인 :
`$cat /etc/deafulat/docker |grep —disable-legacy-registry`

진단결과

```
root@ubuntu:/home/yurim# ps -ef | grep dockerd
root      1369      1  0 15:14 ?        00:00:02 /usr/bin/dockerd -H fd:// --containerd=/run/contai
nerd/containerd.sock
root      2820    2858  0 15:36 tty1    00:00:00 grep --color=auto dockerd
```

비고

-

진단코드	D-30	참고	kisa DO-12 / sk 1.8
진단항목명	추가 권한 획득으로부터 컨테이너 제한	취약도	중
진단기준			
양호	컨테이너 추가 권한 획득 제한 설정이 적용되어 있는 경우		
취약	컨테이너 추가 권한 획득 제한 설정이 적용되어 있지 않은 경우		
진단방법			

컨테이너 목록 확인 : `$ docker ps —quiet all`
 SecurityOpt 옵션 설정 확인 : `$ docker inspect <CONTAINER_ID> | grep SecurityOpt`
 추가 권한 획득 제한 설정 적용 확인: `$ docker ps —quiet —all | xargs docker inspect`
`—format '{{ . Id}} : SecurityOpt={{ .HostConfig.SecurityOpt }}`

진단결과

```
root@docker:~# docker ps --quiet --all | xargs docker inspect --format '{{ . Id}}: SecurityOpt={{ .HostConfig.SecurityOpt}}'
e61de8ef1a41b5e56c96ebc9f9b51722b282623521e704a470e5f13699e2917c: SecurityOpt=<no value>
5e20e28243975a6b86ac79964c3eb6c14a83f24c64adeccf2b95073dd72a0fe6: SecurityOpt=<no value>
a4fe7fb8fc56dcb5e16f3dbdc1cd9c24c5e54bc2935b847382e156ae0994c158: SecurityOpt=<no value>
root@docker:~# docker info --format '{{ .SecurityOptions}}'
[name=apparmor,name=seccomp,profile=builtin]
```

비고

docker 1.11에 추가된 옵션

진단코드	D-31	참고	kisa DO-25 / sk 1.4
진단항목명	root가 아닌 user로 컨테이너 실행	취약도	중
진단기준			
양호	컨테이너가 root 계정으로 실행되지 않은 경우		
취약	컨테이너가 root 계정으로 실행되고 있는 경우		
진단방법			

컨테이너가 root 계정으로 실행되고 있는지 확인 :

```
$ docker ps -quiet -all | xargs docker inspect --format '{{.Id }}: User={{.Config.User }}
```

진단결과

```
root@docker:/home/e01# docker ps --quiet -all | xargs docker inspect --format '{{.Id }}: User={{.Config.User }}'
7e29df341bfef1860a648ddb8642bc82d703df5e6e61314d509237d589655cdd: User=
```

비고

위 명령어 실행 시 username 또는 user ID 를 반환한다.

빈칸으로 나오는 경우 root 계정으로 컨테이너가 실행되어 있다는 것을 의미한다.

진단코드	D-32	참고	kisa DO-26 / sk 3.3
진단항목명	도커를 위한 컨텐츠 신뢰성 활성화	취약도	중
진단기준			
양호	Docker 컨텐츠 신뢰성 설정이 활성화 되어 있는 경우		
취약	Docker 컨텐츠 신뢰성 설정이 비활성화 되어 있는 경우		
진단방법			

echo \$DOCKER_CONTENT_TRUST 명령어 입력했을 경우 "1"을 반환하는지 확인

진단결과

```
root@docker:/home/e01# echo $DOCKER_CONTENT_TRUST
```

비고

기본설정 값 : disabled

Bash shell 전체 적용 시 /etc/bash.bashrc(Ubuntu 계열)

또는 /etc/bashrc(CentOS계열)에 작성 후 적용

Docker Engine 1.8부터 추가된 기능

진단코드	D-33	참고	kisa DO-27 / sk 1.8
진단항목명	컨테이너 SELinux 보안 옵션 설정	취약도	중
진단기준			
양호	SELinux 보안옵션이 활성화 되어 있는 경우		
취약	SELinux 보안옵션이 비활성화 되어 있는 경우		

진단방법

프로세스 확인을 통해 SELinux 보안옵션 적용 여부 확인 : `ps -ef | grep docker | grep selinux-eabled`

SELinux 보안 옵션이 적용되어 있는지 확인 : `docker ps --quiet --all | xargs docker inspect --format "{{ .Id }}: SecurityOpt={{ .HostConfig.SecurityOpt }}"`

진단결과

```
root@docker:/home/e01# ps -ef | grep docker | grep selinux-enabled
root@docker:/home/e01# docker ps --quiet --all | xargs docker inspect --format "{{ .Id }}: SecurityOpt={{ .HostConfig.SecurityOpt }}"
7e29df341bfef1860a648ddb8642bc82d703df5e6e61314d509237d589655cdd: SecurityOpt=<no value>
root@docker:/home/e01# _
```

비고

-

진단코드	D-34	참고	kisa DO-29 / sk 1.5
진단항목명	컨테이너에서 privileged 포트 매핑 금지	취약도	중
진단기준			
양호	컨테이너 포트가 privileged 포트에 매핑되어 있지 않은 경우		
취약	컨테이너 포트가 privileged 포트에 매핑되어 있는 경우		

진단방법

컨테이너 목록 확인 : `$ docker ps --quiet --all`

각 컨테이너에 매핑된 포트 확인 : `$ docker inspect <CONTAINER ID> | grep -A 50 NetworkSettings | grep Ports`

컨테이너 전체 목록을 출력하는 옵션을 통해 매핑된 포트 확인 : `$ docker ps -a`

컨테이너에 매핑된 포트 확인 : `$ docker inspect <CONTAINER ID> | grep -A 50 NetworkSettings | grep Ports`

진단결과

```
root@docker:/home/e01# docker ps --quiet --all
7e29df341bfe
root@docker:/home/e01# docker inspect 7e29df341bfe | grep -A 50 NetworkSettings | grep Ports
"Ports": {
root@docker:/home/e01# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
7e29df341bfe   httpd     "httpd-foreground"      15 minutes ago Up 13 minutes  0.0.0.0:8080->80/tcp, :::8080->80/tcp
root@docker:/home/e01# _
```

```
root@docker:/home/e01# docker ps --quiet --all | xargs docker inspect --format "{{ .Id }}: Ports={{ .NetworkSettings.Ports }}"
7e29df341bfef1860a648ddb8642bc82d703df5e6e61314d509237d589655cdd: Ports=map[80/tcp:[map[HostIp:0.0.0.0 HostPort:8080] map[HostIp:: HostPort:8080]]]
root@docker:/home/e01# _
```

```
root@docker:/home/e01# docker ps --quiet --all | xargs docker inspect --format "{{ .Id }}: PidsLimit={{ .HostConfig.PidsLimit }}"
7e29df341bfef1860a648ddb8642bc82d703df5e6e61314d509237d589655cdd: PidsLimit=<no value>
root@docker:/home/e01#
```

비고

-

진단코드	D-35	참고	kisa DO-31 / sk 1.5
진단항목명	도커의 default bridge docker() 사용 제한	취약도	kisa 하 / sk 중
진단기준			
양호	Default bridge docker0를 사용하고 있지 않은 경우		
취약	Default bridge docker0를 사용하고 있는 경우		
진단방법			

Ifconfig 명령어를 통해 Default bridge docker0를 사용하고 있는지 확인

Docker 명령어를 통해 Default bridge docker0를 사용하고 있는지 확인 : \$ docker network ls --quiet | xargs docker network inspect --format '{{.Name }}: {{ .Options }}' | grep name

진단결과

```
root@docker:~# ifconfig | grep docker
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
docker_gwbridge: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
root@docker:~# docker network ls --quiet | xargs docker network inspect --format '{{.Name }}: {{ .Options }}' | grep name
bridge: map[com.docker.network.bridge.default_bridge:true com.docker.network.bridge.enable_icc:true com.docker.network.bridge.enable_ip_masquerade:true com.docker.network.bridge.host_binding_ipv4:0.0.0.0 com.docker.network.bridge.name:docker0 com.docker.network.driver.mtu:1500]
docker_gwbridge: map[com.docker.network.bridge.enable_icc:false com.docker.network.bridge.enable_ip_masquerade:true com.docker.network.bridge.name:docker_gwbridge]
root@docker:~# _
```

비고

-

진단코드	D-36	참고	kisa DO-32 / sk 1.4
진단항목명	호스트의 user namespaces 공유제한	취약도	kisa 하 / sk 중
진단기준			
양호	호스트의 user namespace를 컨테이너와 공유하고 있지 않은 경우		
취약	호스트의 user namespace를 컨테이너와 공유하고 있는 경우		
진단방법			

UsersnsMode 값을 반환하는지 확인 : \$ docker ps --quiet --all | xargs docker inspect --format '{{.Id }}:UsersnsMode={{ .HostConfig.UsersnsMode }}'

진단결과

```
root@docker:~# docker ps --quiet --all | xargs docker inspect --format '{{.Id }}:UsersnsMode={{ .HostConfig.UsersnsMode }}'
67b3634f3c53b262ecca0578f525ea2fbd4dc390a199f3ed889542b428f98e7a:UsersnsMode=
57f50ce4113693cfd9d9a89b068f92f7adc58cc3fad037e5d1c18e0fdc99cb03:UsersnsMode=
7e29df341bfef1860a648ddb8642bc82d703df5e6e61314d509237d589655cdd:UsersnsMode=
root@docker:~# _
```

비고

host 값을 반환하는 경우 호스트 user namespaces가 컨테이너와 공유되고 있음

진단코드	D-37	참고	sk 1.8
진단항목명	컨테이너 보안 정책	취약도	중
진단기준			
양호	호스트와 컨테이너 간 권한 상승이 이뤄지지 않을 경우		
취약	호스트와 컨테이너 간 권한 상승이 이뤄질 경우		
진단방법			
seccomp 프로파일 적용 : docker ps --quiet --all xargs docker inspect --format '{{ .Id }}: SecurityOpt={{ .HostConfig.SecurityOpt }}' docker info --format '{{ .SecurityOptions }}'			
AppArmor 프로파일 활성화 : docker ps --quiet --all xargs docker inspect --format '{{ .Id }}: SecurityOpt={{ .HostConfig.SecurityOpt }}'			
- 컨테이너 내에서 리눅스 커널 Capabilities 제한 : docker ps --quiet --all xargs docker inspect --format '{{ .Id }}: CapAdd={{ .HostConfig.CapAdd }} CapDrop={{ .HostConfig.CapDrop }}'			

진단결과
<pre> root@docker:~# docker ps --quiet --all xargs docker inspect --format '{{.Id}}: SecurityOpt={{.HostConfig.SecurityOpt}}' e61de8ef1a41b5e56c96ebc9f9b51722b282623521e704a470e5f13699e2917c: SecurityOpt=<no value> 5e20e28243975a6b86ac79964c3eb6c14a83f24c64adeccf2b95073dd72a0fe6: SecurityOpt=<no value> a4fe7fb8fc56dcb5e16f3dbdc1cd9c24c5e54bc2935b847382e156ae0994c158: SecurityOpt=<no value> root@docker:~# docker info --format '{{.SecurityOptions}}' {name=apparmor name=seccomp,profile=builtin} root@docker:~# docker ps --quiet --all xargs docker inspect --format '{{.Id}}: AppArmorProfile={{.AppArmorProfile}}' e61de8ef1a41b5e56c96ebc9f9b51722b282623521e704a470e5f13699e2917c: AppArmorProfile=docker-default 5e20e28243975a6b86ac79964c3eb6c14a83f24c64adeccf2b95073dd72a0fe6: AppArmorProfile=docker-default a4fe7fb8fc56dcb5e16f3dbdc1cd9c24c5e54bc2935b847382e156ae0994c158: AppArmorProfile= root@docker:~# docker ps --quiet --all xargs docker inspect --format '{{.Id}}: SecurityOpt={{.HostConfig.SecurityOpt}}' e61de8ef1a41b5e56c96ebc9f9b51722b282623521e704a470e5f13699e2917c: SecurityOpt=<no value> 5e20e28243975a6b86ac79964c3eb6c14a83f24c64adeccf2b95073dd72a0fe6: SecurityOpt=<no value> a4fe7fb8fc56dcb5e16f3dbdc1cd9c24c5e54bc2935b847382e156ae0994c158: SecurityOpt=<no value> root@docker:~# docker ps --quiet --all xargs docker inspect --format '{{.Id}}: CapAdd={{.HostConfig.CapAdd}} CapDrop={{.HostConfig.CapDrop}}' e61de8ef1a41b5e56c96ebc9f9b51722b282623521e704a470e5f13699e2917c: CapAdd=<no value> CapDrop=<no value> 5e20e28243975a6b86ac79964c3eb6c14a83f24c64adeccf2b95073dd72a0fe6: CapAdd=<no value> CapDrop=<no value> a4fe7fb8fc56dcb5e16f3dbdc1cd9c24c5e54bc2935b847382e156ae0994c158: CapAdd=<no value> CapDrop=<no value> </pre>

비고
-

진단코드	D-38	참고	sk 1.9
진단항목명	로그 관리	취약도	하
진단기준			
양호	로그파일이 정기적으로 확인/감독되며 백업 보관될 경우		
취약	로그파일이 정기적으로 확인/감독되며 백업 보관되지 않을 경우		
진단방법			
로그 레벨 설정 : ps -ef grep docker --log-level			
중앙 집중식 원격 로깅 구성 : ps -ef grep dockerd --log-driver			
진단결과			
<pre>root@docker:~# ps -ef grep docker root 1292 1 0 11:36 ? 00:00:06 /usr/bin/dockerd -H fd:// --containerd=/run/contai nerd/containerd.sock root 4193 4015 0 12:09 tty1 00:00:00 grep --color=auto docker</pre>			
<pre>root@docker:~# docker info --format '{{.LoggingDriver}}' json-file root@docker:~# ^C root@docker:~# ps -ef grep docker root 1292 1 0 11:36 ? 00:00:06 /usr/bin/dockerd -H fd:// --containerd=/run/contai nerd/containerd.sock root 4228 4015 0 12:15 tty1 00:00:00 grep --color=auto docker</pre>			

비고
단기 적용(적용 시 개발자 및 운영자 협의)

진단코드	D-39	참고	sk 3.1
진단항목명	Dockerfile Config	취약도	중
진단기준			
양호	이미지 내 과도한 권한 이 부여되어 있지 않을 경우		
취약	이미지 내 과도한 권한 이 부여되어 있을 경우		
진단방법			

컨테이너 사용자 지정 : `docker ps --quiet --all | xargs docker inspect --format '{{.Id}}: User={{.Config.User}}'`

Dockerfile 내 secrets 존재 여부 확인 : `docker images`
`docker history <Image_ID>`

setuid 및 setgid 권한 제거 : `docker run <Image_ID> find / -perm +6000 -type f -exec ls -ld {} \; 2> /dev/null`

ADD 대신 COPY 사용 : `docker images`
`docker history <Image_ID>`

Dockerfile을 통한 업데이트 금지 : `docker images`
`docker history <Image_ID>`

진단결과

```
root@ubuntu:/etc/docker# docker ps --quiet --all | xargs docker inspect --format '{{.Id}}:User={{.Config.User}}'
dc337d6ee6dc229249c2c863cfd2b24062627b55e228ed24fc04b20d3f106f7d:User=
0d5f0255a60b52877793dcfe045460ed2434cde3c25266393628132e7ae15db5:User=
a01bb850269e1257d3e75228f3510ca3834c751aaf6f899d8649c215a0d1b296:User=
```

```
root@ubuntu:/etc/docker# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
tomcat 9.0 c8f6d60c8268 6 days ago 484MB
root@ubuntu:/etc/docker# docker history c8f6d60c8268
IMAGE CREATED CREATED BY SIZE COMMENT
c8f6d60c8268 6 days ago /bin/sh -c #(nop) CMD ["catalina.sh" "run"] 0B
<missing> 6 days ago /bin/sh -c #(nop) EXPOSE 8080 0B
<missing> 6 days ago /bin/sh -c set -eux; nativeLines="$(catalin... 0B
<missing> 6 days ago /bin/sh -c set -eux; savedAptMark="$(apt-m... 28MB
<missing> 6 days ago /bin/sh -c #(nop) ENV TOMCAT_SHA512=0e173fc... 0B
<missing> 6 days ago /bin/sh -c #(nop) ENV TOMCAT_VERSION=9.0.74 0B
<missing> 6 days ago /bin/sh -c #(nop) ENV TOMCAT_MAJOR=9 0B
<missing> 6 days ago /bin/sh -c #(nop) ENV GPG_KEYS=48F8E69F6390... 0B
<missing> 6 days ago /bin/sh -c #(nop) ENV LD_LIBRARY_PATH=/usr/... 0B
<missing> 6 days ago /bin/sh -c #(nop) ENV TOMCAT_NATIVE_LIBDIR=... 0B
<missing> 6 days ago /bin/sh -c #(nop) WORKDIR /usr/local/tomcat 0B
<missing> 6 days ago /bin/sh -c mkdir -p "$CATALINA_HOME" 0B
<missing> 6 days ago /bin/sh -c #(nop) ENV PATH=/usr/local/tomca... 0B
<missing> 6 days ago /bin/sh -c #(nop) ENV CATALINA_HOME=/usr/lo... 0B
<missing> 6 days ago /bin/sh -c #(nop) CMD ["jshell"] 0B
<missing> 6 days ago /bin/sh -c echo Verifying install ... &&... 0B
<missing> 6 days ago /bin/sh -c set -eux; ARCH="$(dpkg --prin... 330MB
<missing> 6 days ago /bin/sh -c #(nop) ENV JAVA_VERSION=jdk-17.0... 0B
<missing> 6 weeks ago /bin/sh -c apt-get update && DEBIAN_FRON... 48.2MB
<missing> 6 weeks ago /bin/sh -c #(nop) ENV LANG=en_US.UTF-8 LANG... 0B
<missing> 6 weeks ago /bin/sh -c #(nop) ENV PATH=/opt/java/openjd... 0B
<missing> 6 weeks ago /bin/sh -c #(nop) ENV JAVA_HOME=/opt/java/o... 0B
<missing> 8 weeks ago /bin/sh -c #(nop) CMD ["/bin/bash"] 0B
<missing> 8 weeks ago /bin/sh -c #(nop) ADD file:c8ef6447752cab254... 77.8MB
<missing> 8 weeks ago /bin/sh -c #(nop) LABEL org.opencontainers... 0B
<missing> 8 weeks ago /bin/sh -c #(nop) LABEL org.opencontainers... 0B
<missing> 8 weeks ago /bin/sh -c #(nop) ARG LAUNCHPAD_BUILD_ARCH 0B
<missing> 8 weeks ago /bin/sh -c #(nop) ARG RELEASE 0B
```

```
root@ubuntu:/home/yurim# docker run c8f6d60c8268 find / -perm +6000 -type f -exec ls -ld {} \; 2> /dev/null
```

비고

진단코드	D-40	참고	sk 3.2
진단항목명	이미지 취약점 및 구성 결함	취약도	중
진단기준			
양호	컨테이너 내 이미지가 신뢰할 수 있으며 취약하지 않을 경우		
취약	컨테이너 내 이미지가 신뢰할 수 없거나 취약할 경우		
진단방법			
호스트 내 이미지 검증 : \$docker images			
\$docker history (imagename)			
컨테이너 내 패키지 검증 : \$docker ps -quiet			
\$docker exec \$INSTANCE_ID dpkg			

진단결과			
<pre> root@docke:~# docker images REPOSITORY TAG IMAGE ID CREATED SIZE mariadb latest 4af0c16be4b1 5 days ago 403MB hello-world latest 9c7a54a9a43c 12 days ago 13.3kB mysql latest 8189e588b0e8 4 weeks ago 564MB </pre>			
<pre> root@docke:~# docker history mariadb IMAGE CREATED CREATED BY SIZE COMMENT 4af0c16be4b1 5 days ago /bin/sh -c #(nop) CMD ["mariadb"] 0B <missing> 5 days ago /bin/sh -c #(nop) EXPOSE 3306 0B <missing> 5 days ago /bin/sh -c #(nop) ENTRYPOINT ["docker-entry... 0B <missing> 5 days ago /bin/sh -c #(nop) COPY file:94a071210c811522... 22.5kB <missing> 5 days ago /bin/sh -c #(nop) COPY file:cee75098a882f7d3... 8.78kB <missing> 5 days ago /bin/sh -c #(nop) VOLUME [/var/lib/mysql] 0B <missing> 5 days ago [2 GP6_KEYS=177F4010FE56CA3336300305F1656F24... 311MB <missing> 5 days ago [2 GP6_KEYS=177F4010FE56CA3336300305F1656F24... 123B <missing> 5 days ago /bin/sh -c #(nop) ARG REPOSITORY=http://arc... 0B <missing> 5 days ago /bin/sh -c #(nop) ENV MARIADB_VERSION=1:10.... 0B <missing> 5 days ago /bin/sh -c #(nop) ARG MARIADB_VERSION=1:10.... 0B <missing> 5 days ago /bin/sh -c #(nop) LABEL org.opencontainers.... 0B <missing> 13 days ago /bin/sh -c #(nop) ENV LANG=C.UTF-8 0B <missing> 13 days ago [1 GP6_KEYS=177F4010FE56CA3336300305F1656F24... 0B <missing> 13 days ago [1 GP6_KEYS=177F4010FE56CA3336300305F1656F24... 14.5MB <missing> 13 days ago /bin/sh -c #(nop) ARG GP6_KEYS=177F4010FE56... 0B <missing> 13 days ago /bin/sh -c #(nop) ENV GOSU_VERSION=1.14 0B <missing> 13 days ago /bin/sh -c groupadd -r mysql && useradd -r -m... 329kB <missing> 3 weeks ago /bin/sh -c #(nop) CMD ["/bin/bash"] 0B <missing> 3 weeks ago /bin/sh -c #(nop) ADD file:2fc6364d149ecc7f... 77.8MB <missing> 3 weeks ago /bin/sh -c #(nop) LABEL org.opencontainers.... 0B <missing> 3 weeks ago /bin/sh -c #(nop) LABEL org.opencontainers.... 0B <missing> 3 weeks ago /bin/sh -c #(nop) ARG LAUNCHPAD_BUILD_ARCH 0B <missing> 3 weeks ago /bin/sh -c #(nop) ARG RELEASE 0B </pre>			
<pre> root@docke:~# docker ps --quiet 735744ddf291 </pre>			

```

Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/half-h/conf/Half-Inst/trig-aWait/Trig-pend
| Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                  Version              Architecture Description
+++=-----+-----+-----+-----+
|| adduser                 3.110ubuntu5         all          add and remove user
s and groups
|| apt                     2.4.9                amd64        commandline package
manager
|| base-files              12ubuntu4.3          amd64        Debian base system
miscellaneous files
|| base-passwd             3.5.52build1          amd64        Debian base system
master password and group files
|| bash                    5.1-6ubuntu1          amd64        GNU Bourne Again Sh
ell
|| bsdtar                  1:2.37.2-4ubuntu3     amd64        basic utilities fro
n 4.4BSD-Lite
|| ca-certificates         20211016ubuntu0.22.04.1 all          Common CA certifica
tes
|| coreutils               8.32-4.1ubuntu1       amd64        GNU core utilities
|| dash                    0.5.11+git20210903+057cd650a4ed-3build1 amd64        POSIX-compliant she
ll
|| debconf                 1.5.79ubuntu1         all          Debian configuratio
n management system
|| debianutils             5.5-1ubuntu2          amd64        Miscellaneous utili
ties specific to Debian
|| diffutils              1:3.8-0ubuntu2        amd64        File comparison uti
lities
|| dpkg                    1.21.1ubuntu2.1       amd64        Debian package mana
gement system
|| e2fsprogs               1.46.5-2ubuntu1.1     amd64        ext2/ext3/ext4 file
system utilities
|| findutils               4.8.0-1ubuntu3        amd64        utilities for findi
ng files--find, xargs
|| galera-4                26.4.14-ubu2204       amd64        Replication framewo

```

비고

진단코드	D-41	참고	sk 4.3
진단항목명	네트워크 제어	취약도	중
진단기준			
양호	Docker Swarm 네트워크가 특정 외부 인터페이스 연결만 허용 설정되어 있을 경우		
취약	Docker Swarm 네트워크가 특정 외부 인터페이스 연결만 허용 설정되어 있을 경우		
진단방법			

Docker Swarm 네트워크 인터페이스 설정(특정 인터페이스에서만 수신 대기 중인지 확인) : `netstat -lt | grep -i 2377`

진단결과
(Docker Swarm 네트워크 인터페이스를 찾을 수 없음)
비고

장기 적용(적용 시 개발자 및 운영자 협의)



도커 취약점 수동 점검 결과 리포트