docker
취약점 진단 프로젝트

KISA
Korea Internet & Security Agency

K-Shield Jr.

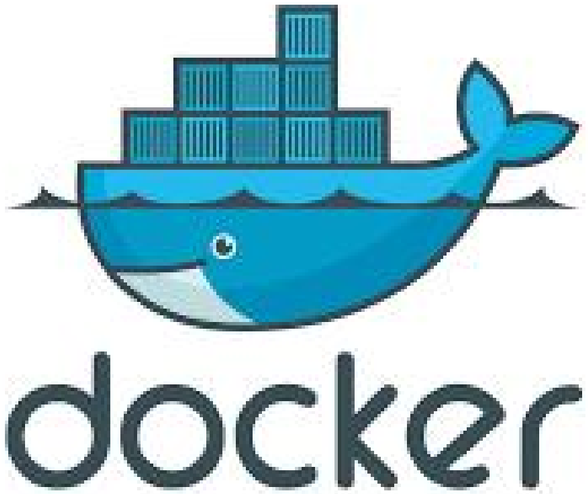
E-01
정보시스템 취약점 진단

표재경 (팀장)
김서연
김효민
전동현
황유림

목 차

1. Docker 소개

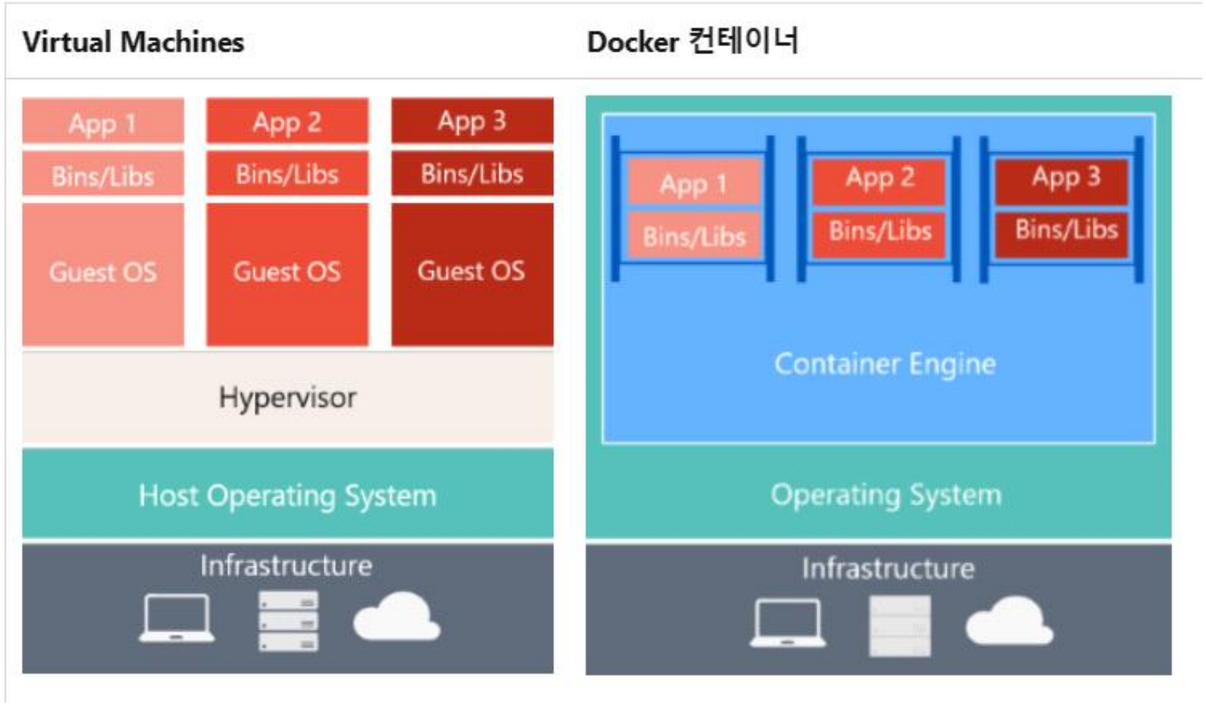
Docker란?



1. 컨테이너 기반의 오픈소스 가상화 플랫폼

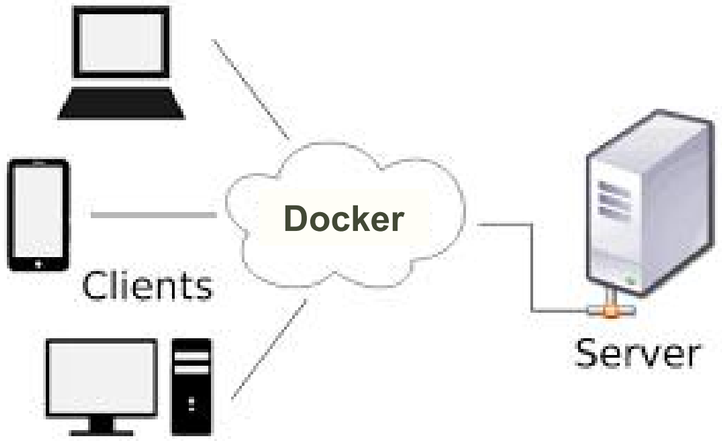
2. 리눅스 컨테이너에 리눅스 어플리케이션을
컨테이너로 프로세스 격리하여
실행 및 관리하는 프로젝트로 시작

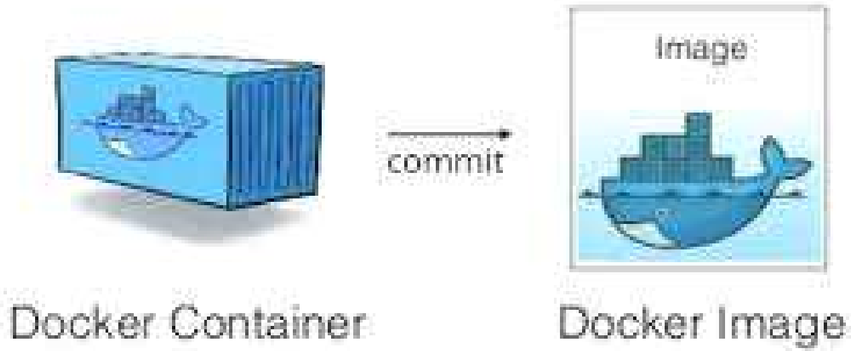
Docker의 장점



가상머신	도커
게스트 운영체제 사용을 위한 라이브 커널 포함 (큰 용량 배포) 가상화된 공간이 하이퍼바이저를 거치며 호스트에 비해 성능 손실 O	커널을 공유해서 사용하기 때문에 컨테이너에는 라이브러리 및 실행 파일만 존재 (작은 용량 배포) 리눅스 자체 기능을 사용하여 프로세스 단위의 격리 환경을 생성, 성능 손실 X

도커의 장점





1. 여러 개의 서비스를 1개의 서버에서 구동 용이

2. 대상 컨테이너 재현 용이

도커 사용량 추세

18 million +
developers

7 million +
applications

13 billion +
monthly image downloads

기업의 도커 이용


See who uses Docker
Adobe
AT&T
Blue Apron
INTEGRATE
Lucent Health
paloalto NETWORKS
PathFactory
PayPal
pipedrive
Segment
snowflake
spiceworks
splunk>
stripe
TARGET
The Container Store
UNIVERSITY OF CALGARY
verizon
VIRGINIA TECH

개인의 도커 이용

K-shield Jr 10기 취약점 진단 분반 프로젝트 <도커 사용 경험에 대한 설문조사> 폼

안녕하세요! 저희는 K-Shield Jr. 10기를 수강중인 학생들입니다😄

현재 저희는 docker 취약점 진단 프로젝트를 진행하고 있습니다. 프로젝트 진행 중 docker 사용 경험에 대해 자료 수집이 필요한 부분이 있어 설문을 진행하려고 합니다.
설문대상은 도커 사용경험이 있는 IT 관련 종사자~경험자로, 개인, 팀 혹은 회사 프로젝트에서 도커를 사용하신 분들이라면 더욱 감사할 것 같습니다🙇

✓K-Shield Jr.란?
향후 정보보호 분야 취업 시 현장 실무를 즉각 수행할 수 있는 사이버 보안 전문 주니어 인력 양성을 목표로 하는 프로그램입니다.

✓설문조사에서 제공해주신 개인정보(선택사항)는 프로젝트 종료 후 6개월 이내에 일괄 폐기될 예정입니다.

많은 참여 부탁드립니다‼️

[회사 프로젝트]
redis, kafka, MariaDB, MongoDB, ES 등을 구축할 때 이용.
spring boot 및 flask 무중단 서비스 운영을 위해 사용.
서버의 운영체제 버전과 프로젝트의 호환성이 충돌하여 docker image를 통해 서버 배포

[회사 프로젝트] Deep learning 서비스를 위해 일정한 환경구축용 사용

[회사 프로젝트] 쿠버네티스에서 사용할 컨테이너 이미지 개발
[개인 프로젝트] CTF문제 제작
[팀 프로젝트] K Shield Jr 프로젝트 장고 서버 운영

개인 채굴 프로젝트

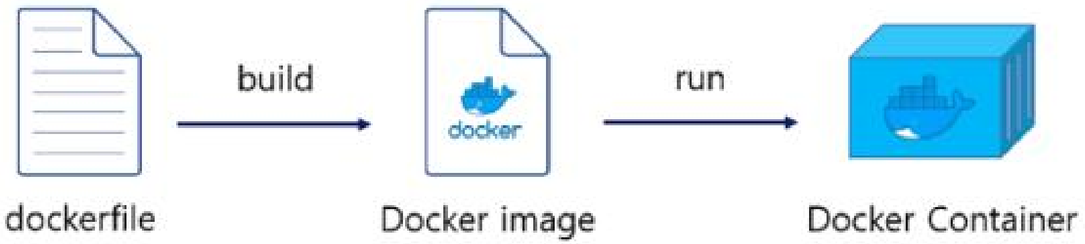
개인, 팀

회사프로젝트 도커진단, 개인프로젝트 테스트 및 공부용도

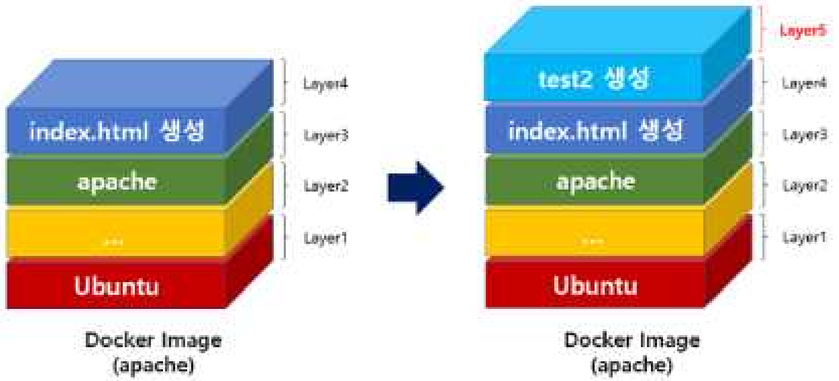
[개인 프로젝트] 랩실에서 알파고 돌리기위해서 사용

2. 악성 Docker 이미지 배포 및 Docker 취약성 사례

악성 Docker 이미지 배포

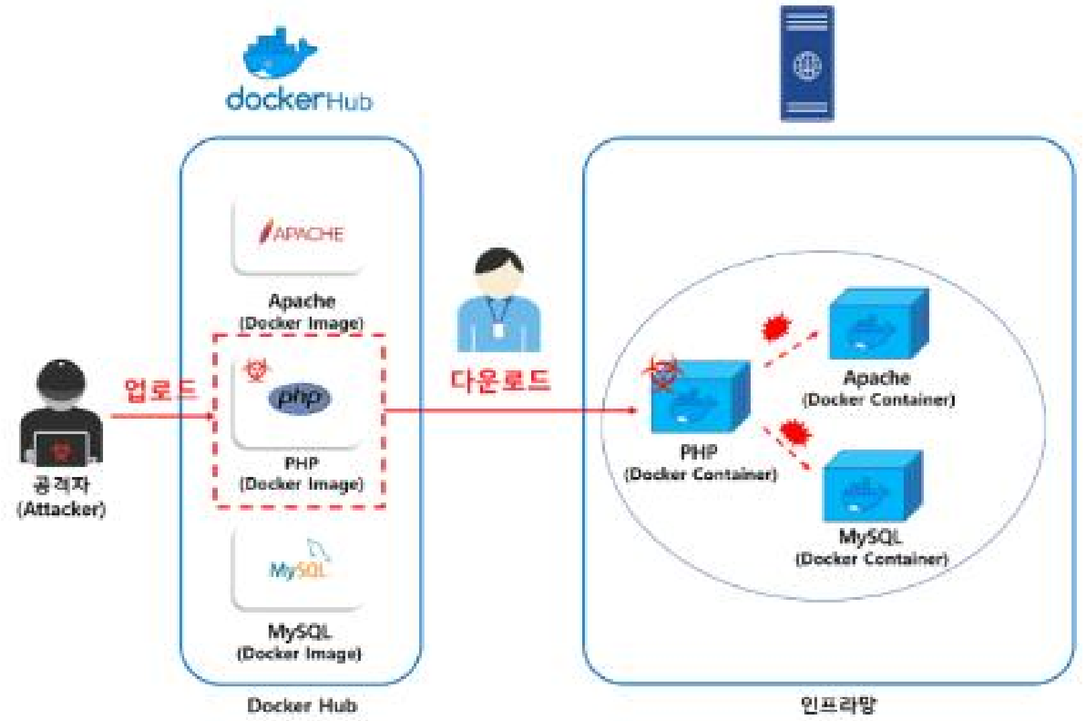


1. 도커 파일을 빌드하여 도커 이미지 생성



2. 기존 도커 이미지에 컨테이너에서 작업한 내역을 추가하여 새로운 도커 이미지 생성

악성 Docker 이미지 배포



Docker 취약성 사례

보안뉴스 | 전체기사 | SECURITY | IT | SAFETY | Security World

Home > 전체기사

도커허브에서 1650개 이상의 악성 컨테이너 발견돼

좋아요 5개 | 입력 : 2022-11-25 11:37

도커허브의 높은 신뢰도 노린 공급망 공격...

요약 : 도커 이미지들의 공공 저장소이자
악성 컨테이너가 발견됐다고 IT 외신 블리
암호화폐 채굴, 백도어, DNS 하이재킹, 웹
시스디그(Sysdig)가 25만 개의 이미지들을 검사하여 얻어낸 결과다. 이 컨테이너들을 의심 없이 받아 로컬 시스템에 설치 후 사용하면 해당 시스템은 자동으로 암호화폐 채굴 기계가 되거나, 백도어 및 DNS 하이재킹 기능이 설치된 감시 기계가 된다.

데일리시큐 2020.12.06.

도커 허브 컨테이너 **이미지** 50% 이상, 심각한 취약점 포함...잠재...

보안 전문가는 **도커** 허브(Docker Hub)에 호스팅 된 400만 개의 공개 **도커** 컨테이너 **이미지**를 분석 한 결과 그 중 절반에 심각한 결함이 있음을 발견했다. 시큐리...

Docker 취약성 사례

프로젝트 진행 과정

Docker 진단 항목 수립

KISA
클라우드 취약점 점검 가이드

SK infosec
클라우드 보안가이드
Docker, Kubernetes

수동 진단

Doker 진단 가이드의 모든 항목에 대하여 수동 진단 후 ‘도커 취약점 수동점검 결과 리포트’ 작성

자동 스크립트 진단

Docker 진단 가이드 '상' 항목에 대해 자동화 스크립트 작성

Docker 취약점 상세보고서

수동 진단 및 자동 진단 결과 보고서 작성

3. Docker 진단 항목 수립

Docker 진단 항목 수립

KISA 클라우드 취약점 점검 가이드

SK infosec 클라우드 보안 가이드

Docker 진단 항목 수립

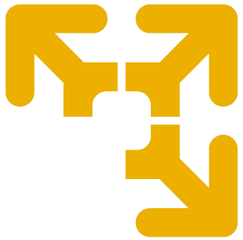
목 차

총 41개 항목

4. Docker 진단 환경 소개

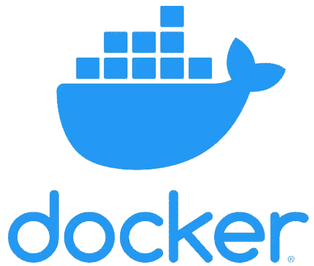
Docker 진단 환경

VMWare



Ubuntu 18.04

docker 23.0.6



MariaDB

5. 수동 진단

수동 진단

DO-13. docker.service 소유권 설정

양호 : docker.service 파일의 소유자 및 소유그룹이 root:root인 경우

취약 : docker.service 파일의 소유자 및 소유그룹이 root:root가 아닌 경우

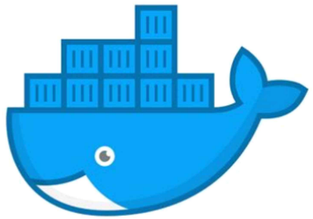
※ 이 파일은 시스템에 없을 수 있다. 그런 경우에는 권장 사항이 적용되지 않음.

```
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
```

=> 소유자 및 소유그룹이 root:root 이므로 양호

Docker 진단 가이드 모든 항목에 대해 수동진단 진행

수동 진단



K-Shield Jr 10기

도커 취약점
수동 점검 결과 리포트

취약점 진단 E-01조

표재경
김서연
김효민
전동현
황유림

진단코드	D-09	참고	kisa D0-13 / 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 D0-14 / 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.

6. 자동스크립트 진단

자동 스크립트 진단

구분	진단코드	진단 항목	취약도
	D-01	도커최신 패치 적용 (kisa(DO-01), sk(5.3))	상
	D-02	Docker daemon audit 설정 (kisa(DO-03), sk(2.2))	상
	D-03	/var/lib/docker audit 설정 (kisa(DO-04), sk(2.2))	상
	D-04	/etc/docker audit 설정 (kisa(DO-05), sk(2.2))	상
	D-05	docker.service audit 설정 (kisa(DO-06), sk(2.2))	상
	D-06	docker.socket audit 설정 (kisa(DO-07), sk(2.2))	상
	D-07	/etc/default/docker audit 설정 (kisa(DO-08), sk(2.2))	상
	D-08	default bridge를 통한 컨테이너 간 네트워크 트래픽 제한 (kisa(DO-09), sk(1.5))	상
	D-09	docker.service 소유권 설정 (kisa(DO-13), sk(2.1))	상
	D-10	docker.service 파일 접근권한 설정 (kisa(DO-14), sk(2.1))	상
	D-11	docker.socket 소유권 설정 (kisa(DO-16), sk(2.1))	상
	D-12	docker.socket 파일 접근권한 설정 (kisa(DO-18), sk(2.1))	상
	D-13	/etc/docker 디렉터리 소유권 설정 (kisa(DO-17), sk(2.1))	상
	D-14	/etc/docker 디렉터리 접근권한 설정 (kisa(DO-18), sk(2.1))	상
	D-15	/var/run/docker.sock 파일 소유권 설정 (kisa(DO-19), sk(2.1))	상
	D-16	/var/run/docker.sock 접근 권한 설정 (kisa(DO-20), sk(2.1))	상
	D-17	daemon.json 파일 소유권 설정 (kisa(DO-21), sk(2.1))	상
	D-18	daemon.json 접근 권한 설정 (kisa(DO-22), sk(2.1))	상
	D-19	/etc/default/docker 파일 소유권 설정 (kisa(DO-23), sk(2.1))	상
	D-20	/etc/default/docker 접근 권한 설정 (kisa(DO-24), sk(2.1))	상
	D-21	컨테이너에서 ssh 사용 금지 (kisa(DO-25), sk(1.5))	상
	D-22	호스트 OS 주요 자원 접근 제어 (sk(1.1))	상
	D-23	인증-권한 제어 (sk(1.2))	상
	D-24	SSL/TLS 적용 (sk(1.3))	상
	D-25	컨테이너 권한 제어 (sk(1.7))	상
	D-26	인증제어 (sk(4.1))	상
	D-27	SSL/TLS 적용 (sk(4.2))	상

취약도 상에 해당하는 27개 항목 선정

자동 스크립트 진단

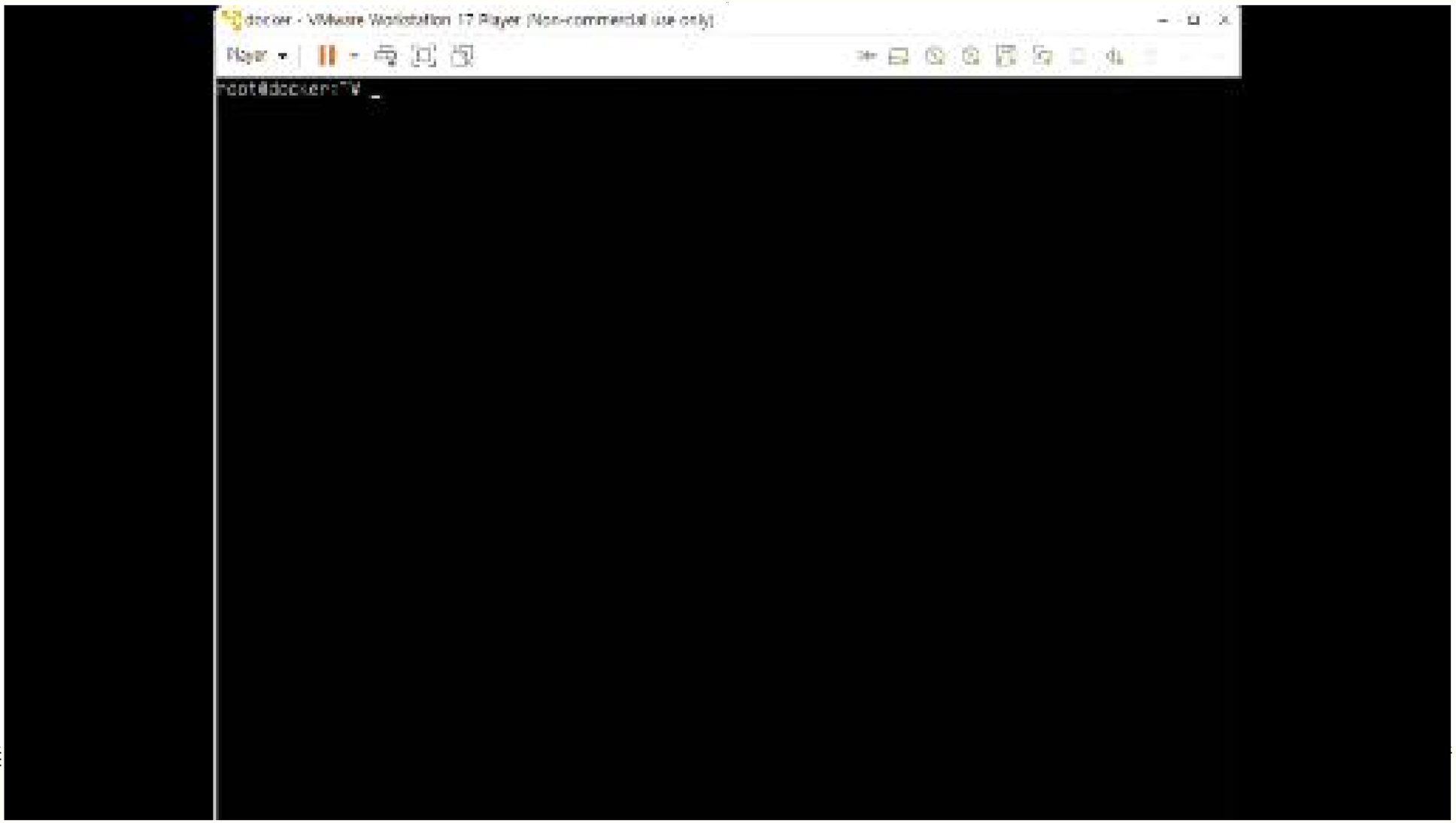
리눅스 쉘 스크립트

```
echo "[ DO-16 ]: Cehck"
echo "========================[ DO-16 START ]" >> $RESULT_FILE 2>&1
echo "" >> $RESULT_FILE 2>&1

DOCKER_SOCKET_FILE=$(systemctl show -p FragmentPath docker.socket | awk -F= '{print $2}')

if [ -e "$DOCKER_SOCKET_FILE" ]; then
        ls -l $DOCKER_SOCKET_FILE >> $RESULT_FILE 2>&1
        permission_val=`stat -c '%a' $DOCKER_SOCKET_FILE`
        owner_perm_val=`echo "$permission_val" | awk '{ print substr($0, 1, 1) }'`
        group_perm_val=`echo "$permission_val" | awk '{ print substr($0, 2, 1) }'`
        other_perm_val=`echo "$permission_val" | awk '{ print substr($0, 3, 1) }'`
        if [ "$owner_perm_val" -le 6 ] && [ "$group_perm_val" -le 4 ] && [ "$other_perm_val" -le 4 ]
; then
                echo "Result: Good" >> $RESULT_FILE 2>&1
        else
                echo "Result: Vulnerable" >> $RESULT_FILE 2>&1
        fi
else
        echo "docker.socket file not found." >> $RESULT_FILE 2>&1
        echo "Result: Review" >> $RESULT_FILE 2>&1
fi

echo "" >> $RESULT_FILE 2>&1
echo "========================[ DO-16 END ]" >> $RESULT_FILE 2>&1
echo "" >> $RESULT_FILE 2>&1
```

쉘 스크립트 진단 결과 파일

```
========================[ DO-16 START ]

-rw-r--r-- 1 root root 295 May  5 21:17 /lib/systemd/system/docker.socket
Result: Good

========================[ DO-16 END ]

========================[ DO-17 START ]

drwxr-xr-x 2 root root 4096 May  5 21:17 /etc/docker
Result: Good

========================[ DO-17 END ]

========================[ DO-18 START ]

drwxr-xr-x 2 root root 4096 May  5 21:17 /etc/docker
Result: Good

========================[ DO-18 END ]

========================[ DO-19 START ]

srw-rw---- 1 root docker 0 May 14 14:24 /var/run/docker.sock
Result: Good

========================[ DO-19 END ]

========================[ DO-20 START ]

srw-rw---- 1 root docker 0 May 14 14:24 /var/run/docker.sock
Result: Good
```

자동 스크립트 진단 실행 영상

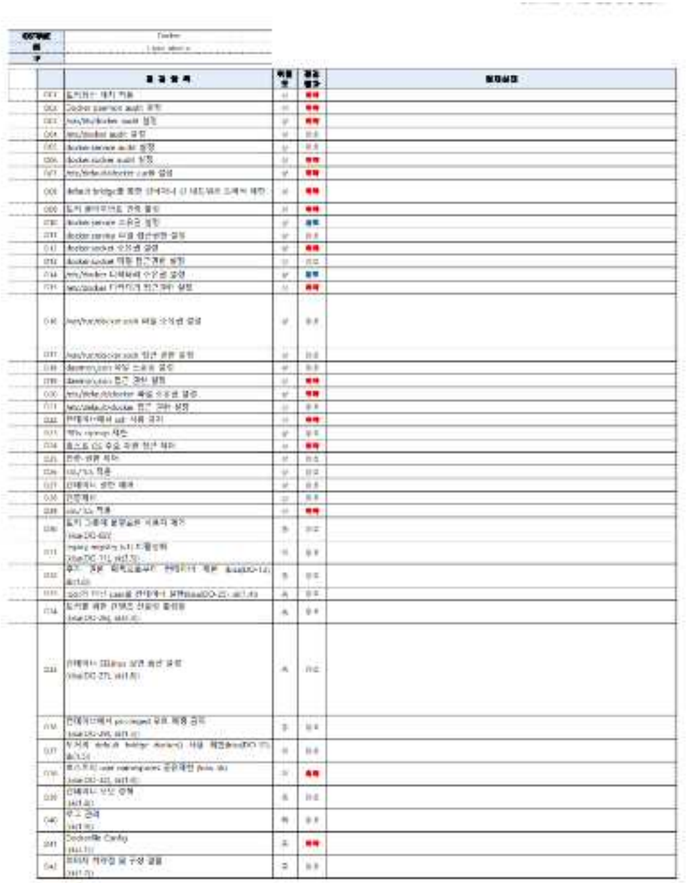


7. Docker 취약점 상세 보고서

Docker 취약점 상세 보고서

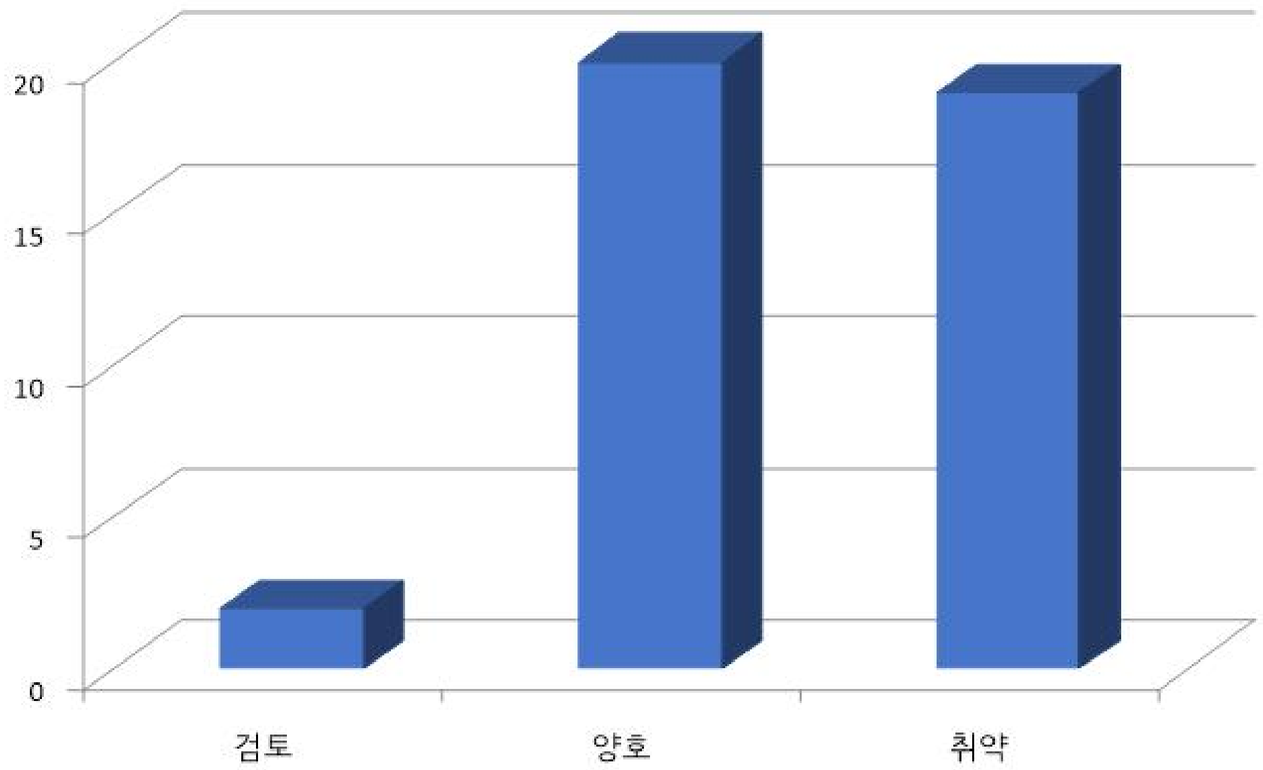
Docker 취약점 점검 상세 보고서

2023. 05. 17



진단 결과

각 항목별 진단 결과



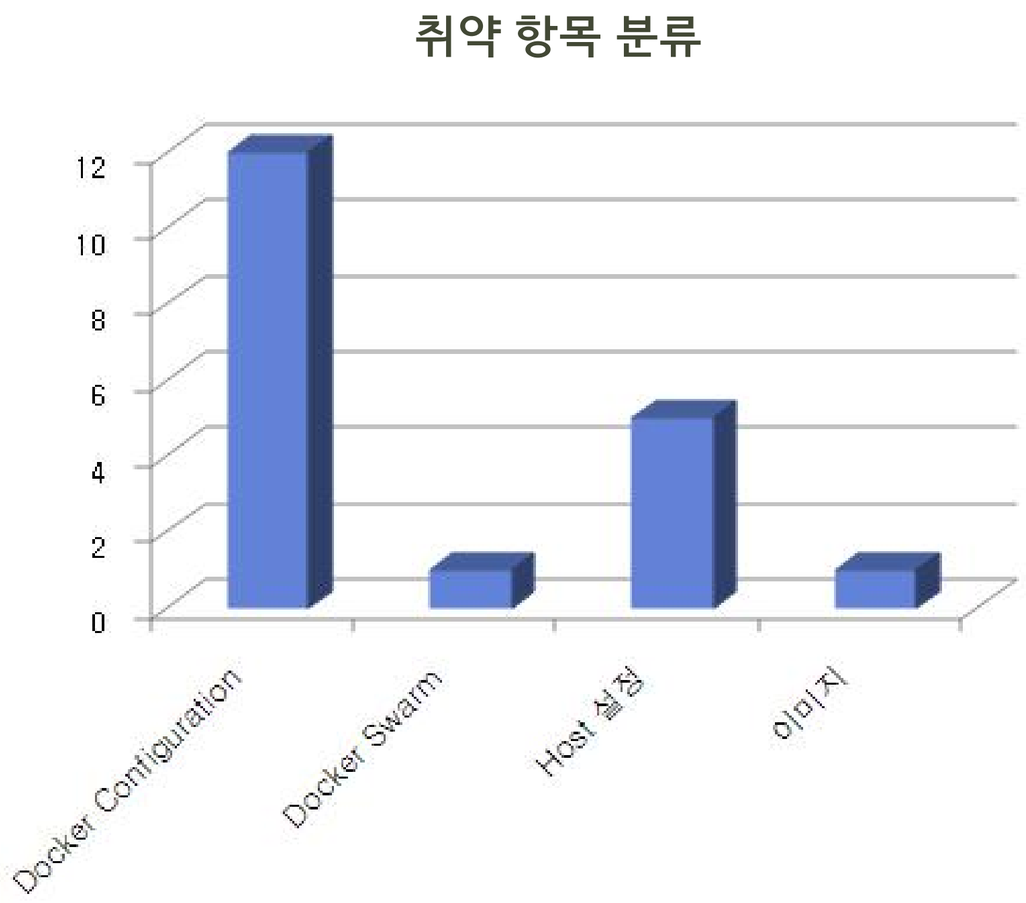
총 41개 항목

양호: 20개

취약: 19개

검토: 2개

취약 항목



총 19개 항목

Docker Configuration: 12개

Docker Swarm: 1개

Host 설정: 5개

이미지: 1개

취약 항목 보호 대책

Docker Configuration	Docker Swarm	Host 설정	이미지
도커 컨테이너 각 설정 파일 내부 설정을 알맞게 변경한 후 서비스 재실행 필요	**SSL/TLS** 적용 필요	**docker** 감사 파일 설정 추가 필요	**dockerfile** 내 **ADD** 명령어 대신 **COPY** 명령어 사용 필요

8. 기대효과

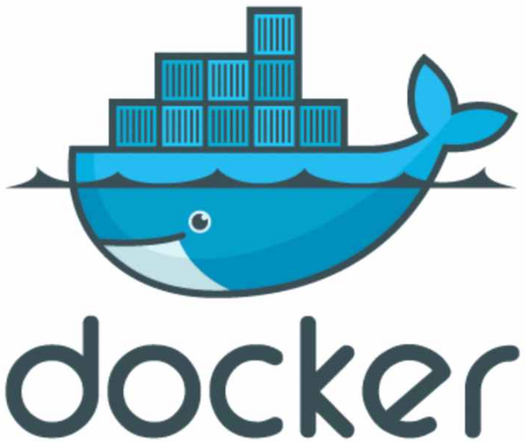
DOCKER 취약점 진단 필요성

DOCKER

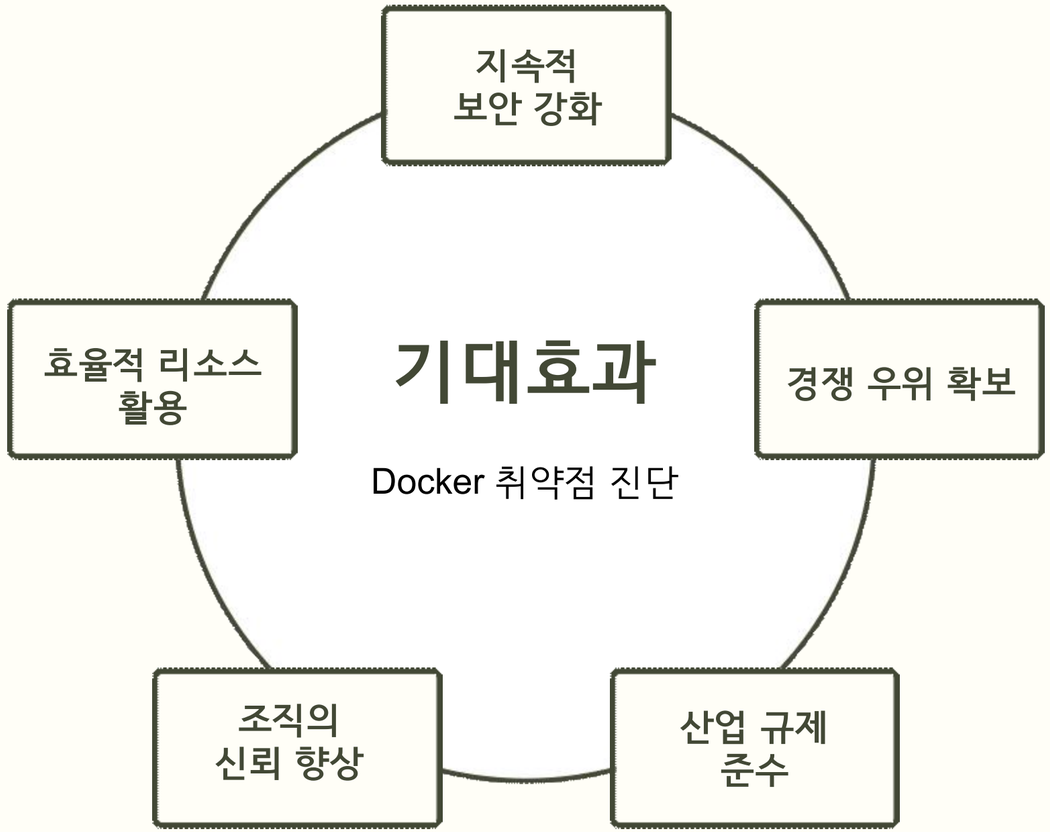
- 현대 애플리케이션의 개발과 배포는
  컨테이너화 기술인 Docker의 사용으로 큰 변화 O

- Docker 환경에서의 보안 취약점은 여전히 중요한 문제 존재

- 전체적인 취약점 식별과 보안요소 체크
  ⇒ 애플리케이션 및 시스템의 안전성 보장



Docker 취약점 진단 기대효과



Docker 환경에서 발생 가능한 보안 취약점 식별 & 조치

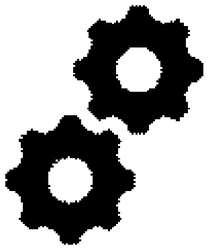
→ **시스템 보안 수준 향상**

→ **잠재적인 데이터 누출 위험 감소**

DOCKER 취약점 진단 기대효과

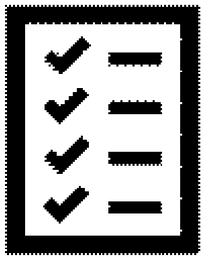
지속적 보안 강화

효율적 리소스 활용



조직의 신뢰 강화
& 경쟁우위 확보

산업 규제 준수



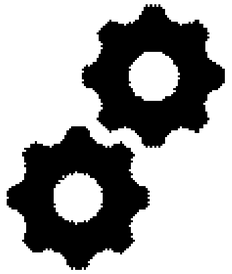
DOCKER 취약점 진단 기대효과

지속적 보안 강화

- 단기적 사용이 아닌 지속적 보안 강화를 위한 도구로 활용
- 새로운 취약점 발견 & 보안 요구사항 변경 시
  ⇒ 스크립트 업데이트를 통한 최신 보안 취약점을 식별 및 대응 가능
- 조직의 보안 위협에 대응하고 시스템의 지속적인 보안 강화 목표 달성 가능

DOCKER 취약점 진단 기대효과

효율적 리소스 활용



- 대량의 컨테이너 및 이미지를 효율적으로 분석 가능
  (Docker 취약점 진단 스크립트는 자동화된 방식으로 작동)
- 보안 전문가 및 개발자의 시간과 노력을 절약하고, 업무 효율성을 향상 가능
- 스크립트의 결과 및 보안 취약점 보고서를 통해 리소스 할당 및 우선순위 설정에 도움

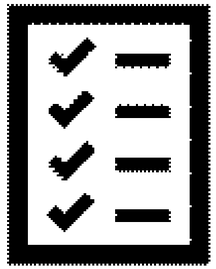
DOCKER 취약점 진단 기대효과

조직의 신뢰 강화 & 경쟁 우위 확보

- 조직의 보안인식을 강조하고 고객, 파트너, 규제 기관 등과의 신뢰 강화에 기여
- 보안에 대한 책임감을 보여주고 고객 정보와 기업 자산을 안전하게 보호 가능
- 조직이 경쟁 환경에서 우위를 점하는 데 기여
  ⇒ 보안 취약점으로인한 시스템 다운타임 & 데이터 유출 시시
  ⇒ 심각한 비용과 이미지 손상
- 스크립트를 사용하여 취약점을 사전에 발견하고 대응
  ⇒ 조직의 안정성과 신뢰성을 강조하며 고객과 파트너들에게 더 큰 신뢰를 줄 수 있음

DOCKER 취약점 진단 기대효과

산업 규제 준수



- 다양한 산업 분야에서 보안 규제 및 규정 준수는 필수적
- Docker 취약점 진단 스크립트를 사용하여 조직은 규제 요구 사항 확인 가능
  ⇒ 보안 관련 감사 및 인증 절차를 통과 가능
- 지속적 보안 취약점진단을 통해 조직의 비즈니스 확장과 고객의 신뢰 유지에 기여

9. 인사이트

INSIGHT

- 클라우드 보안 취약점 진단 가이드에서 취약도 ‘상’ 기준의 항목만 체크
  → 향후 진단 스크립트를 발전시켜 나머지 ‘중’, ‘하’ 항목에 대해서도 적용 필요

- 다양한 조건에서의 취약점 진단 여부 확인 미흡
  → 정책적인 요소를 통한 접근, 컨테이너 내부의 악성 이미지 환경에서의 취약점 진단 부족

- **Docker** 취약점 진단 스크립트의 지속적 개선과 혁신 필요
  → 보안 커뮤니케이션을 통한 스크립트 지속 개선과 취약점 식별의 중요성 이해
  → 새로운 보안 조치 및 정책 개발에 대한 인사이트 제공을 통한 보안 역량 강화 기대

감사합니다