

# Kubernetes 보안 취약 사례 분석 및 대응 연구

## Analysis and Countermeasures for Kubernetes Security Vulnerabilities

### 요약

이 논문은 **Kubernetes**의 보안 취약점을 조사하고 구현하며, 분석 및 대응 방법을 제공하는 한다. 이를 위해, 논문은 최소한의 **Kubernetes** 클러스터를 구축하고 필요한 지식과 기술을 습득하며, 보안 취약점을 식별, 분석 및 예방한다. 논문은 2절에서 조사한 기술과 취약점의 특성을 설명한다. 3절에서는 보안 취약점을 구현하고 그 원인을 분석한다. 마지막으로, 4절에서는 결론과 및 향후 연구 방향을 제시한다.

### 1. 서론

현대 기업에서는 컨테이너 기술의  
경량성과 확장성으로 인해 많은 기업들이  
컨테이너 기술을 도입하고 있다. 이러한  
컨테이너 기술을 보다 효율적으로  
관리하기 위해 **Kubernetes**가 많이  
사용되고 있다. 그러나 **Kubernetes**는 많은  
기능을 제공하기 때문에, 보안에 대한  
취약점이 발생할 가능성이 있다.

이에 본 논문은 **Kubernetes**를 사용하여  
보안 취약점을 조사하고 구현하여 분석 및  
대응 방법을 연구하는 것을 목표로 한다.  
구체적으로, 구축한 **Kubernetes**  
클러스터에 내부에서 발생하는 보안

취약점을 식별하고 분석하며, 예방하고  
대응하는 것을 목표로 한다.

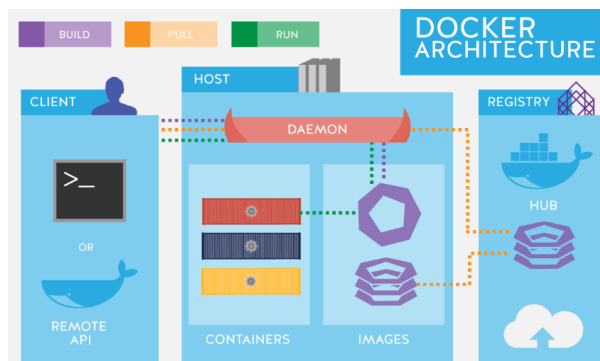
이를 위해, 간소한 클러스터를 구성하고,  
필요한 기술과 지식을 습득하여 웹  
서비스를 구축하고 보안 취약점을 해결할  
수 있도록 연구할 것이다. 본 논문은  
**Kubernetes**를 사용하는 기업들에게 많은  
도움을 줄 수 있으며, 보안 취약점에 대한  
인식을 높이고 대응 방안을 제시하는 데에  
큰 의의가 있다.

본 논문은 2절에서 연구에 필요한  
기술들과 조사한 취약점의 특징에 대해  
정리한다. 3절에서는 실제로 보안  
취약점을 구현하고 취약점의 원인에 대해

분석한다. 4절에서는 결론 및 향후 연구를 제시한다.

## 2. 연구 이론 및 관련 연구

### 2.1 Docker



[그림 1] Docker 구조

Docker는 컨테이너 가상화 기술을 이용하여 애플리케이션을 개발, 배포 및 실행할 수 있는 플랫폼이다. Docker를 사용하면 애플리케이션과 필요한 라이브러리, 환경 등을 패키징하여 컨테이너 이미지로 만들고, 이를 다른 환경에서도 동일하게 실행할 수 있다.

Docker는 다양한 운영 체제에서 사용할 수 있으며, 매우 가볍고 빠른 속도로 애플리케이션을 실행할 수 있다. 또한, Docker 이미지는 레이어(layer) 형태로 구성되어 있어서 여러 이미지를 공유하여 사용할 수 있고, 이를 이용하여

애플리케이션을 보다 쉽게 배포할 수 있다.

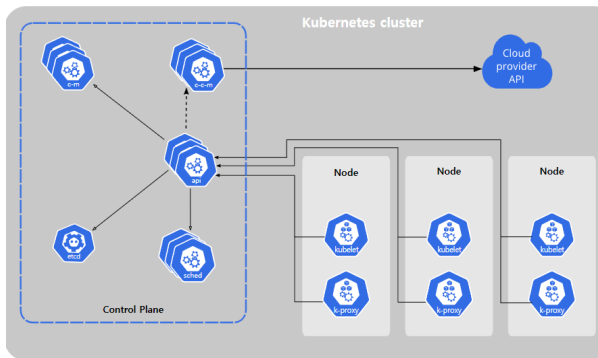
Docker는 다양한 기능을 제공하며, CLI를 통해 컨테이너를 생성, 실행, 중지 및 삭제할 수 있으며 컨테이너를 이미지로 빌드할 수 있다.

Docker는 다양한 플랫폼에서 사용되며, 애플리케이션을 빠르게 개발하고 배포하는 데 매우 유용하다. 본 논문에서는 웹 서비스 이미지를 빌드하거나 취약점 구현을 위해 Dockerfile로 빌드한 이미지를 Docker Hub를 통해 이미지를 배포한다.

### 2.2 Kubernetes

Kubernetes는 컨테이너 오케스트레이션 플랫폼으로, 컨테이너화된 애플리케이션의 배포, 확장, 관리 등을 자동화하는 도구입니다. 이는 컨테이너 애플리케이션을 보다 쉽게 관리하고, 더욱 효율적으로 배포 및 확장할 수 있도록 지원한다.

Kubernetes는 Control Plane과 Worker Node로 구성되며, 클러스터의 상태를 관리하고, 애플리케이션 컨테이너를 실행합니다. 이는 API 서버, 스케줄러, 컨트롤러 매니저, etcd 등의 다양한 컴포넌트로 구성되어 있다.



[그림 2] Kubernetes 구조

Kubernetes는 자동 배포, 스케일링, 관리 기능을 제공하여 애플리케이션을 자동으로 관리할 수 있으며, 무중단 업데이트와 롤백, 스케일링, 로드 밸런싱, 자동 복구 등의 기능도 지원한다. 또한, Kubernetes는 컨테이너 간의 네트워크, 보안, 스토리지 등의 기능을 제공하여 보다 쉬운 관리를 지원한다.

Kubernetes는 다양한 리소스 오브젝트(Resource Object)를 사용하여 애플리케이션을 관리합니다. 이러한 리소스 오브젝트에는 노드, 파드(Pod), 서비스(Service), 볼륨(Volume), 디플로이먼트(Deployment), 스테이트풀셋(StatefulSet) 등이 포함된다. 이러한 리소스 오브젝트는 YAML 파일로 정의되며, Kubernetes는 이를 이용하여 애플리케이션을 배포하고 관리한다. 본 논문에서는 주로 Deployment를 이용해 웹 서비스 Pod 및 취약점 구현 Dockerfile 이미지를 관리할 예정이다.

## 2.3 보안 취약점 정리

첫 번째 보안 취약점은

CVE-2022-0185이다. CVE-2022-0185는 리눅스 커널의 파일 시스템 컨텍스트 기능의 `legacy_parse_param` 함수에서 발견된 힙 기반 버퍼 오버플로 취약점으로, 공급된 매개변수 길이를 확인하는 방식에서 발생한다. 파일 시스템 컨텍스트 API를 지원하지 않는 파일 시스템에서 (따라서 이전 방식을 사용하는 경우) 권한이 없는 로컬 사용자가 이 취약점을 이용하여 시스템 권한을 상승시킬 수 있다. 이는 비권한 사용자 네임스페이스가 활성화된 경우 비권한, 그렇지 않은 경우는 `namespaced CAP_SYS_ADMIN` 권한이 필요하다.

CVE-2022-0185는 리눅스 커널에 대한 취약점으로 `kubernetes`와 직접적으로 관련은 없다. 하지만 `kubernetes`가 Linux 운영 체제 위에서 실행되기 때문에, `kubernetes` 클러스터가 취약한 버전의 Linux 커널에서 실행 중인 경우에는 이 취약점의 영향을 받을 수 있다. 따라서 `kubernetes`에서는 사용자가 클러스터의 기본 운영 체제에 대한 보안 패치와 업그레이드를 하여 CVE-2022-0185 취약점의 위험을 최소화한다.

두 번째로는 CVE-2021-25735이다. CVE-2021-25735는 kubernetes의 kube-apiserver에서 발견된 취약점으로 Linux 커널의 tty 서브시스템에서 발생하는 use-after-free 취약점으로 이어질 수 있는 race condition이다. 이 취약점은 다양한 버전의 Linux 커널에 영향을 미치며, 취약한 시스템에 로컬 액세스가 있는 공격자가 임의의 코드를 상승 권한으로 실행할 수 있는 가능성이 있다.

이 취약점을 해결하기 위해 Linux 커널 커뮤니티에서 패치를 발표 했고 사용자들은 공개 된 패치를 적용한다. 그렇게 사용자는 운영체제를 최신 버전으로 업그레이드 하거나 패치를 수동적으로 설치하여 취약점을 해결할 수 있다.

```
sudo apt-get update
sudo apt-get upgrade

sudo yum update
```



세 번째는 CVE-2021-25741이다. CVE-2021-25741은 인기 있는 콘텐츠 관리 시스템인 Drupal의 여러 버전에 영향을 주는 보안 취약점이다. 이 취약점은 Drupal이 특정 입력 데이터를 처리하는 방식과 관련이 있으며, 공격자가 대상 시스템에서 임의의 코드를 실행할 수 있게 한다.

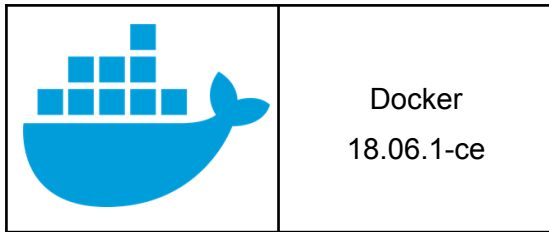
구체적으로, 이 취약점은 Drupal에서 양식을 빌드하고 처리하는 데 사용되는 Form API에 존재한다. 공격자는 취약한 Drupal 사이트로 특별히 조작된 양식을 제출함으로써 취약점을 악용할 수 있으며, 이로 인해 대상 시스템에서 임의의 코드를 실행할 수 있다.

이 취약점을 해결하기 위해서는 마찬가지로 최신 버전의 Drupal로 업그레이드하고 사용 가능한 모든 보안 패치를 적용하여 이 취약점이 악용되는 위험을 완화하는 것이다.

### 3. 취약점 구현 및 분석

구현한 취약점은 CVE-2019-5736이다. Docker CLI 툴인 runc에서 발생한 취약점이며 외부에서 희생자 PC의 root권한을 갖고 접근할 수 있게 된다. 구현 환경은 다음과 같다.

	<p>Virtual Box version: 7.0.2</p>
	<p>Ubuntu 18.04.6 (Host) 16.04.7(Attacker)</p>



[표 1] 취약점 구현 환경

Docker 설치에 공식 문서를 참고하며 취약점 구현을 위해 설치 시 다음과 같이 버전을 설정한다.

```
$sudo apt-get install
docker-ce=18.06.1~ce~3-0~ubuntu
```

공개된 POC코드를 다운로드한다.

```
git clone
https://github.com/agppp/cve-2019-5736-poc
```

run.sh 파일에 host pc에 맞는 libseccomp버전을 수정한다. stage.c 파일에 공격자로 사용할 pc ip를 기입하고 다음 라이브러리를 추가한다.

```
#include <string.h>
#include <unistd.h>
```

Host pc에서 Dockerfile로 이미지를 빌드한 뒤 컨테이너를 실행하여 루트 권한으로 run.sh을 실행한다.

공격자 pc에서는 4455 port로 listen 상태로 설정한다. 그 뒤 Host pc에서 다시

run.sh을 실행하면 공격자 pc에서 host pc의 root로 접근하는 것을 확인할 수 있다.

```
attack@attack-VirtualBox:~$ nc -nlvvp 4455
Listening on [0.0.0.0] (family 0, port 4455)
Connection from 192.168.100.10 33976 received!
bash: cannot set terminal process group (5636): Inapp
bash: no job control in this shell
<af535938cfc0c64aae74fe4250b03cf299f58db27fd9b9daa# h
hostname
control-VirtualBox
<af535938cfc0c64aae74fe4250b03cf299f58db27fd9b9daa# i
id
uid=0(root) gid=0(root) groups=0(root)
```

[그림 3] 취약점 구현

이 취약점의 원인은 runC가 사용하는 기능 중 하나인 "file-descriptor passing"에서 발생한다.

컨테이너 내에서 실행되는 프로세스들은 호스트 시스템의 파일 디스크립터를 공유하여 사용한다. 그러나 runC는 호스트 시스템에서 파일 디스크립터를 다시 열고 그것을 컨테이너 내의 프로세스에 전달하는 기능을 가지고 있다. 이 과정에서 파일 디스크립터를 전달 받은 프로세스는 그 파일 디스크립터가 소유하는 파일에 대한 제어 권한을 가지게 되는데, 이때 runC가 제공하는 파일 디스크립터 전달 방식의 버그로 인해, 컨테이너 내부의 악성 코드가 호스트 시스템의 파일 디스크립터를 이용하여 호스트 시스템에서 원하는 동작을 수행할 수 있게 된다.

#### 4. 결론 및 향후 연구

본 논문에서는 **Kubernetes**를 사용한 보안 취약점 분석 및 실제 구현을 연구하였다.

향후 연구에서는 조사한 다른 취약점들을 구현하고 더욱 복잡하고 대규모의 **Kubernetes** 클러스터를 대상으로 보안 취약점을 분석하고 대응하는 방법을 연구할 것이다. 또한, **Kubernetes** 클러스터의 보안 이슈를 해결하기 위해 개발된 보안 솔루션을 조사하고 적용하는 것을 연구할 것이다.