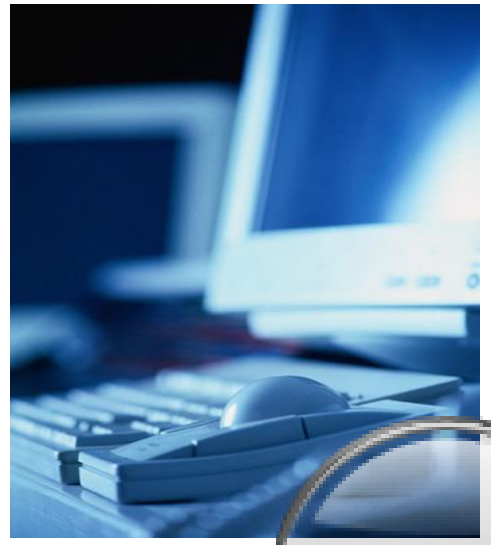
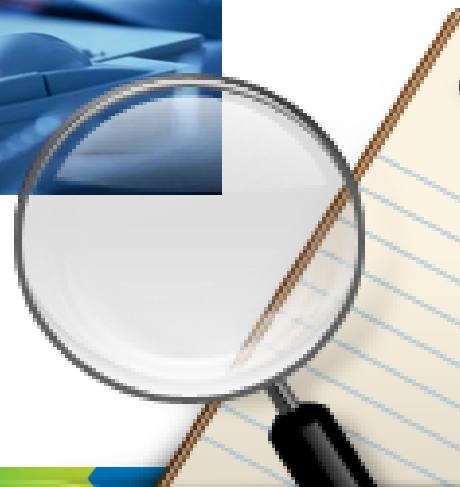
쿠버네티스 보안

- 시큐리티 콘텍스트
- 네트워크 정책 적용
- 쿠버네티스 감사(Audit) 기능 활성화
- Trivy를 활용한 컨테이너 취약점 진단
- kube-bench를 활용한 쿠버네티스 보안 점검
- falco를 활용한 쿠버네티스 컨테이너 보안 모니터링





시큐리티 콘텍스트

시큐리티 콘텍스트

보안 컨텍스트

- 서버 침해사고 발생 시 침해사고를 당한 권한을 최대한 축소하여 그 사고에 대한 확대를 방지
- 침해사고를 당한 서비스가 모든 권한(root나 노드 커널 기능)으로 동작하는 경우 서비스를 탈취한 공격자는 그대로 컨테이너의 권한을 사용할 수 있음
- 최소 권한 정책에 따른 취약점 감소

- 보안 컨텍스트 설정에는 다음 사항을 포함
  - **권한 상승 가능 여부**
  - **프로세스 기본 UID/GID를 활용한 파일 등의 오브젝트의 액세스 제어**
  - **Linux Capabilities를 활용한 커널 기능 추가**
  - 오브젝트에 보안 레이블을 지정하는 SELinux (Security Enhanced Linux) 기능
  - AppArmor : 프로그램 프로필을 사용하여 개별 프로그램의 기능 제한
  - Seccomp : 프로세스의 시스템 호출을 필터링

시큐리티 콘텍스트

파드에 시큐리티 콘텍스트 설정하기

- 파드에 UID를 설정하면 모든 컨테이너에 적용됨
  - 컨테이너 UID/GID 설정하기
  - 권한상승 제한하기

```
$ kubectl exec -it security-context-demo /bin/bash
/ $ id
uid=1000 gid=3000 groups=2000
/ $ ps -eaf
PID   USER     TIME  COMMAND
    1 1000      0:00 sleep 1h
   10 1000      0:00 sh
   16 1000      0:00 ps -eaf
```

security-context.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo
spec:
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: sec-ctx-vol
    emptyDir: {}
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: sec-ctx-vol
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

시큐리티 콘텍스트

컨테이너에 시큐리티 콘텍스트 설정하기

- **컨테이너에 내부에 새로운 유저를 사용하면 그 설정이 우선됨**
  - 컨테이너 UID/GID 설정하기

security-context-2.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo-2
spec:
  securityContext:
    runAsUser: 1000
  containers:
  - name: sec-ctx-demo-2
    image: gcr.io/google-samples/node-hello:1.0
    securityContext:
      runAsUser: 2000
      allowPrivilegeEscalation: false
```

시큐리티 콘텍스트

캐퍼빌리티 적용

- 캐퍼빌리티를 적용하면 리눅스 커널에서 사용할 수 있는 권한을 추가 설정 가능
- 캐퍼빌리티를 활용한 리눅스 커널 권한 변수
  - https://github.com/torvalds/linux/blob/master/include/uapi/linux/capability.h
- date +%T -s "12:00:00" 명령은 SYS_TIME 커널 권한이 있어야만 가능

security-context-4.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo-4
spec:
  containers:
  - name: sec-ctx-4
    image: gcr.io/google-samples/node-hello:1.0
    securityContext:
      capabilities:
        add: ["NET_ADMIN", "SYS_TIME"]
```

시큐리티 콘텍스트

SELinux 권한

- 모든 프로세스와 객체에 시큐리티콘텍스트(또는 레이블)을 달아 관리하는 형태
- 넷필터와 더불어 리눅스의 핵심 보안 기능을 담당
- 그러나 실무에서는 복잡도가 높아 오히려 소외…
- SELinux에 대한 자세한 정보 사이트
  - https://www.lesstif.com/ws/selinux/selinux

- 다음과 같은 형태로 취급

```
...
securityContext:
  seLinuxOptions:
    level: "s0:c123,c456"
```

자료출처: https://www.lesstif.com/ws/selinux/selinux

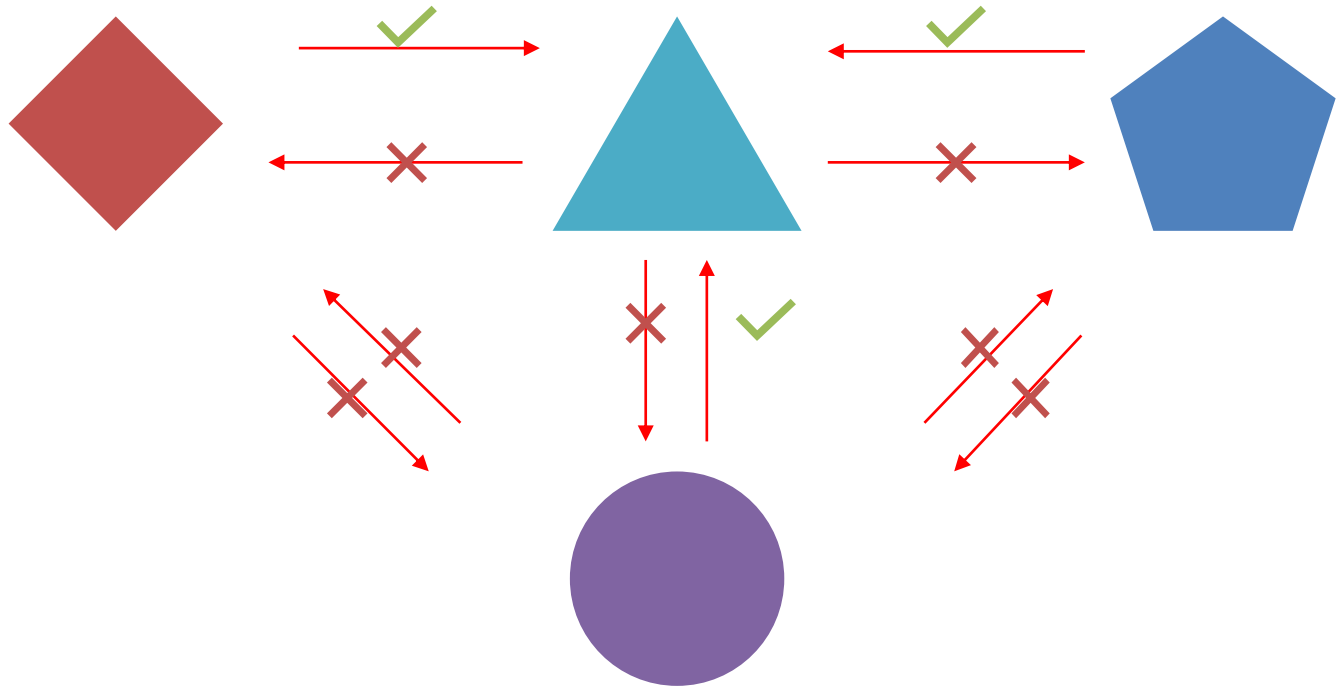
요소	설명
사용자	시스템의 사용자와는 별도의 SELinux 사용자로 역할이나 레벨과 연계하여 접근 권한을 관리하는데 사용.
역할(Role)	하나 혹은 그 이상의 타입과 연결되어 SELinux 의 사용자의 접근을 허용할 지 결정하는데 사용.
타입(Type)	Type Enforcement의 속성중 하나로 프로세스의 도메인이나 파일의 타입을 지정하고 이를 기반으로 접근 통제를 수행.
레벨(Label)	레벨은 MLS(Multi Level System)에 필요하며 강제 접근 통제보다 더 강력한 보안이 필요할 때 사용하는 기능으로 정부나 군대등 최고의 기밀을 취급하는 곳이 아니라면 사용하지 않습니다.

네트워크 정책 적용

네트워크 정책 적용

네트워크 정책(Network Policy)

- 파드 그룹이 서로 및 다른 네트워크 끝점과 통신하는 방법을 지정
- NetworkPolicy 리소스는 레이블을 사용하여 파드를 선택
- 선택한 파드에 허용되는 트래픽을 지정하는 규칙을 정의
- 특정 파드를 선택하는 네임 스페이스에 NetworkPolicy가 있으면 해당 파드는 NetworkPolicy에서 허용하지 않는 연결을 거부



네트워크 정책 적용

이그레스

- 선택된 파드에서 나가는 트래픽에 대한 정책 설정
- ipBlock을 통해 허용하고자 하는 ip 대역 설정 가능
- Ports에는 어떤 포트를 허용하는지 명시

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Egress
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

네트워크 정책 적용

인그레스

- 선택된 파드로 들어오는 트래픽에 대한 정책 설정
- IpBlock 을 통해 허용하고자 하는 ip 대역 설정 가능
- Except를 사용하여 예외 항목 설정 가능
- NamespaceSelector와 podSelector를 사용
  - 그룹별 더욱 상세한 정책 설정 가능

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
```

네트워크 정책 적용

인그레스 정책 만들어보기

● 앱 생성

```
kubectl run hello-web --labels app=hello \
  --image=gcr.io/google-samples/hello-app:1.0 --port 8080 --expose
```

● 네트워크 폴리시 적용

```
# hello-allow-from-foo
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: hello-allow-from-foo
spec:
  policyTypes:
  - Ingress
  podSelector:
    matchLabels:
      app: hello
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: foo
```

네트워크 정책 적용

인그레스 정책 만들어보기

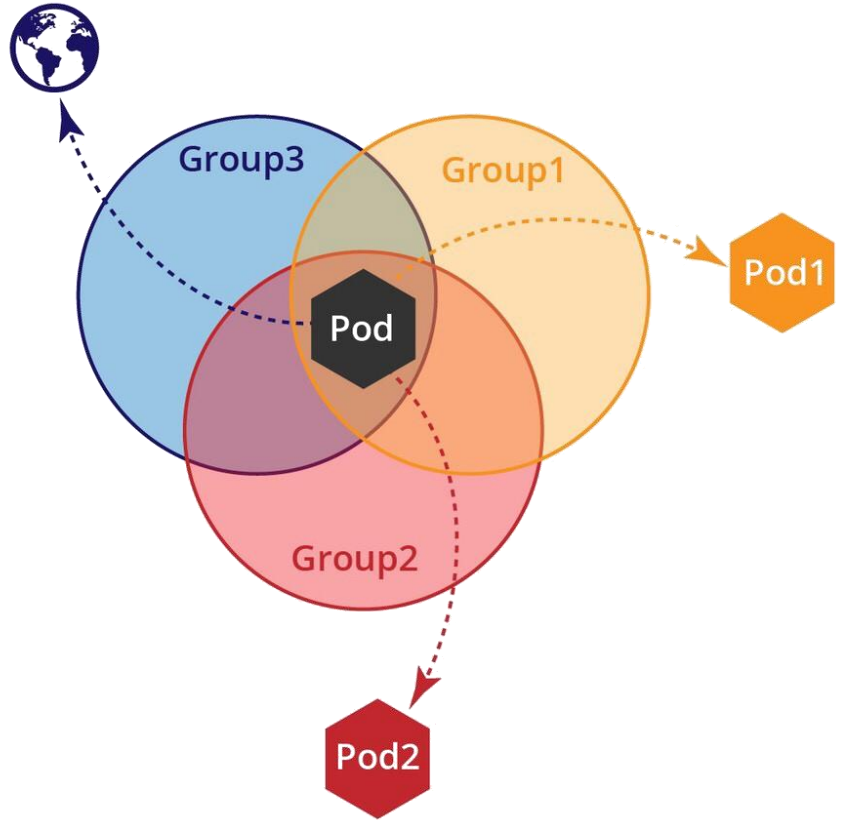
- 서로 다른 레이블을 가진 파드를 만들고 네트워크 폴리시 동작 확인

```
kubectl run -l app=foo --image=alpine --restart=Never --rm -i -t test-1
wget -qO- --timeout=2 http://hello-web:8080

kubectl run -l app=other --image=alpine --restart=Never --rm -i -t test-1
wget -qO- --timeout=2 http://hello-web:8080
```

네트워크 정책 적용

여러 개의 정책이 겹치면 어떻게 될까?

- OR 연산을 수행해 하나라도 허용하면 허용

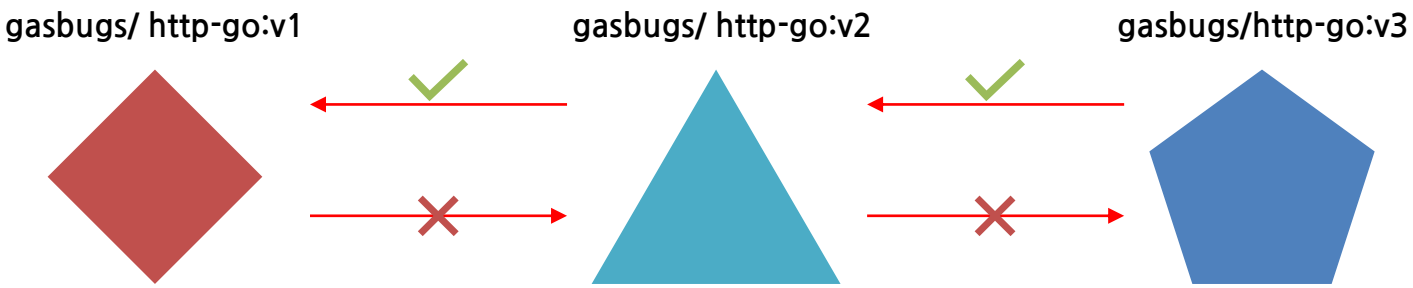


이미지 출처: https://medium.com/@reuvenharrison/an-introduction-to-kubernetes-network-policies-for-security-people-ba92dd4c809d

네트워크 정책 적용

연습 문제

- 다음과 같이 통신이 가능하도록 구성하라(v 표시는 통신 가능, x 표시는 통신 불가).
- 실습 환경 구성 yaml 파일: https://bit.ly/2Jla4k4



쿠버네티스 감사(Audit) 기능 활성화

쿠버네티스 감사(Audit) 기능 활성화

쿠버네티스 감사 기능

- 클러스터에서 발생하는 다양한 작업들에 대해 관리자가 모니터링을 하기 위해 audit 기능을 활성화
- 쿠버네티스 audit 기능을 사용해 실시간으로 각종 유저들이 사용하는 API 정보를 확인 가능
- 쿠버네티스 audit는 보안 관련 레코드 세트를 시간순으로 저장
- 클러스터는 사용자 정보와 Kubernetes API를 사용하는 애플리케이션 및 컨트롤플레인 자체에서 발생하는 활동을 감사

감사 기능을 통해 클러스터 관리자가 확인할 수 있는 정보들

- 무슨 일이 일어났는가?
- 언제 일어났는가?
- 누가 그것을 시작했는가?
- 과거에 무슨 일이 있었는가?
- 어디에서 관찰되었는가?
- 어디에서 시작되었는가?
- 결과는 어디로 흘러가는가?

쿠버네티스 감사(Audit) 기능 활성화

kube-apiserver를 통해 실행

- 감사 레코드는 kube-apiserver를 통해서 실행
- 실행의 각 단계에 대한 각 요청은 감사 이벤트를 생성하고
- 특정 정책에 따라 사전 처리되고 백엔드에 기록
- 각 요청은 연결된 단계에 따라 기록하는 정보가 다름

- 요청에 따른 기록 정보

단계	설명
RequestReceived	감사 핸들러가 요청을 받는 즉시 그리고 핸들러 체인 아래로 위임되기 전에 생성된 이벤트에 대한 단계다.
ResponseStarted	응답 헤더가 전송되었으나 응답 본문이 전송되기 전 상태다. 이 단계는 장기 실행 요청(예: watch)에 대해서만 생성된다.
ResponseComplete	응답 본문이 완료되었으며 더 이상 바이트가 전송되지 않는 단계다.
Panic	패닉이 발생했을 때 생성되는 이벤트다.

쿠버네티스 감사(Audit) 기능 활성화

감사 정책

- 감사 정책은 기록해야 하는 이벤트와 포함해야 하는 데이터에 대한 규칙을 정의
- 감사 정책 객체 구조는 audit.k8s.io API 그룹에 정의
- 이벤트가 처리되면 규칙 목록과 순서대로 비교
- 첫 번째로 일치하는 규칙은 이벤트의 감사 수준을 설정한다

- 정의된 감사 수준

감사 수준	설명
None	이 규칙과 일치하는 이벤트를 기록하지 않습니다.
Metadata	요청 메타데이터(요청 사용자, 타임스탬프, 리소스, 동사 등)를 기록하지만 요청 또는 응답 본문은 기록하지 않습니다.
Request	이벤트 메타데이터 및 요청 본문을 기록하지만 응답 본문은 기록하지 않습니다. 리소스가 아닌 요청에는 적용되지 않습니다.
RequestResponse	이벤트 메타데이터, 요청 및 응답 본문을 기록합니다. 리소스가 아닌 요청에는 적용되지 않습니다.

쿠버네티스 감사(Audit) 기능 활성화

정책 파일 구성하기

- 다음은 감사 정책 파일의 예시: https://blog.naver.com/isc0304/222509921722
- 일을 마스터 노드의 /etc/kubernetes/audit-policy.yaml에 작성
- 각 룰에 대한 설명은 주석을 참조

```
apiVersion: audit.k8s.io/v1 # This is required.
kind: Policy
# RequestReceived 단계의 모든 요청에 대해 감사 이벤트를 생성하지 말아야 한다.
omitStages:
  - "RequestReceived"
rules:
  # RequestResponse 수준에서 포드 변경 사항 기록
  - level: RequestResponse
    resources:
    - group: ""
      # 리소스 "포드"는 RBAC 정책과 일치하는 포드의 하위 리소스에 대한 요청과 일치하지 않습니다
      resources: ["pods"]
  # 메타데이터 수준에서 "pods/log", "pods/status"를 기록합니다.
  - level: Metadata
    resources:
    - group: ""
      resources: ["pods/log", "pods/status"]
```

쿠버네티스 감사(Audit) 기능 활성화

kube-apiserver 파드 설정 변경과 재시작

- 마스터 노드에 --audit-policy-file 플래그를 설정
- 플래그를 생략하면 이벤트가 기록되지 않음
- /etc/kubernetes/manifests/kube-apiserver.yaml 파일을 열고 다음 작업을 진행
- .spec.containers[0].command에 다음 인자를 추가

```
- --audit-policy-file=/etc/kubernetes/pki/audit.conf
- --audit-log-path=/var/log/audit.log
```

- .spec.container[0].volumeMounts와 .spec.volumes에 다음 내용 추가

```
volumeMounts:
- mountPath: /etc/kubernetes/audit-policy.yaml
  name: audit
  readOnly: true
- mountPath: /var/log/audit.log
  name: audit-log
  readOnly: false
```

```
volumes:
- name: audit
  hostPath:
    path: /etc/kubernetes/audit-policy.yaml
    type: File
- name: audit-log
  hostPath:
    path: /var/log/audit.log
    type: FileOrCreate
```

쿠버네티스 감사(Audit) 기능 활성화

감사 로그 확인

- api 서버가 잘 올라오는지 확인
  - (오류 발생 시 트러블슈팅 필요)

```
$ kubectl get pod -n kube-system
```

- 마스터 노드에서 /etc/logs/audit.log 로그 확인

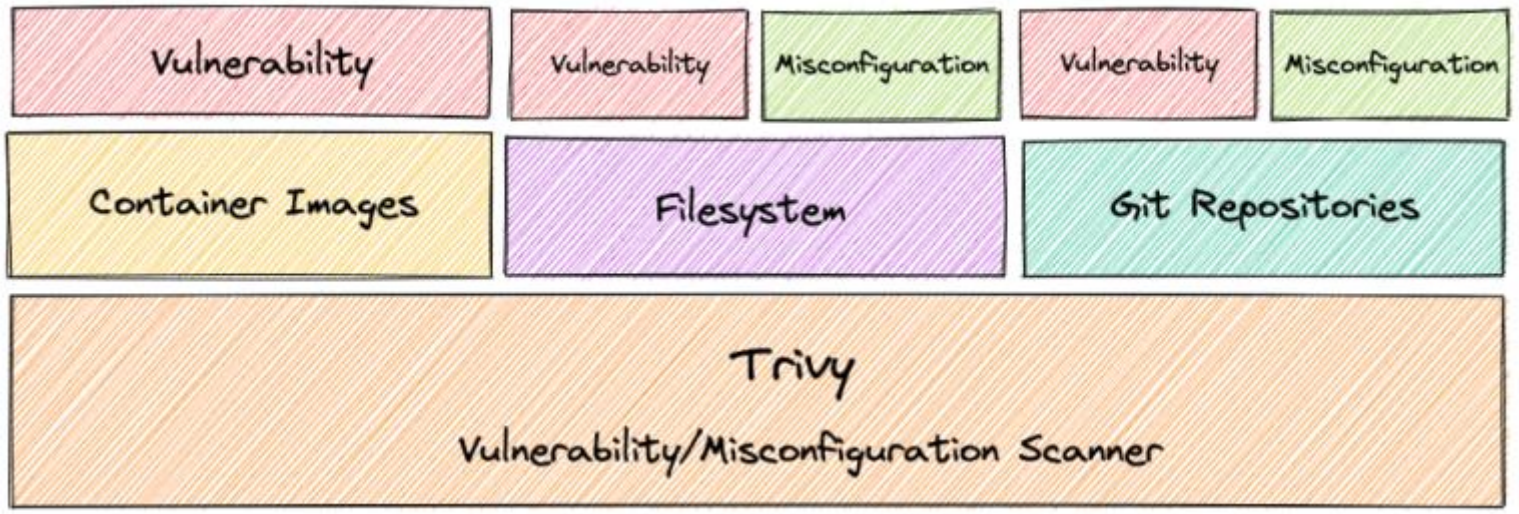

```
{
   "kind":"Event",
   "apiVersion":"audit.k8s.io/v1",
   "level":"Request", // 로그 수준, 요청을 기록
   "auditID":"d8d2a777-d6f9-4b29-b8f4-70cba6b98986",
   "stage":"ResponseStarted",
   "requestURI":"/api/v1/services?allowWatchBookmarks=true\u0026resourceVersion=59106
   "verb":"watch", // api 실행
   "user":{
      "username":"system:kube-scheduler", // 요청을 수행한 유저
      "groups":[
         "system:authenticated" # 유저가 속한 그룹
      ]
   },
   "sourceIPs":[
      "10.0.2.15" // 요청한 IP
   ],
   "userAgent":"kube-scheduler/v1.22.1 (linux/amd64) kubernetes/632ed30/scheduler",
   "objectRef":{
      "resource":"services",
      "apiVersion":"v1"
   },
   "responseStatus":{
      "metadata":{

      },
      "status":"Success",
      "message":"Connection closed early",
      "code":200 // 응답 코드
   },
   "requestReceivedTimestamp":"2021-09-02T03:39:34.261603Z",
   "stageTimestamp":"2021-09-02T03:45:11.263737Z",
   "annotations":{
      "authorization.k8s.io/decision":"allow",
      "authorization.k8s.io/reason":"RBAC: allowed by ClusterRoleBinding \"system:kub
   }
}
```

Trivy를 활용한 컨테이너 취약점 진단

Trivy를 활용한 컨테이너 취약점 진단

Trivy 소개

- 취약점을 간단히 스캔할 수 있는 도구
- 컨테이너 이미지, 파일 시스템 및 Git 리포지토리의 취약점, 미설정 구성 문제에 대한 다양한 진단
- OS 패키지(알파인, RHEL, CentOS 등)와 언어별 패키지(Bundler, Composer, npm, yarn 등)의 취약점을 탐지
- 또한 테라폼, Dockerfile 및 Kubernetes와 같은 IaC(코드) 파일로 인프라를 스캔하여 배포를 공격 위험에 노출시키는 잠재적 구성 문제를 검색하는 기능
- 바이너리를 설치하거나 컨테이너를 사용하는 등 다양하고도 간단한 방법으로 취약점을 진단



Trivy를 활용한 컨테이너 취약점 진단

도커 컨테이너를 활용한 Trivy 설치

- 컨테이너를 사용해서 설치하면 호환성 등의 문제 없이 1분내로 Trivy를 구성
- -v 옵션을 사용해 캐시 디렉토리를 마운트
- socket을 공유해 현재 호스트에 구성된 도커 소켓을 통해 이미지를 컨트롤
- --rm 옵션이 설정되어 있어 실행 후에는 컨테이너가 자동으로 삭제
- 다음 명령을 사용해 다운로드 후에 trivy가 자동으로 실행되며 도움말이 출력

```
$ docker run --rm -v trivy-cache:/root/.cache/ \
-v /var/run/docker.sock:/var/run/docker.sock aquasec/trivy:latest
```

```
NAME:
   trivy - A simple and comprehensive vulnerability scanner for containers

USAGE:
   trivy command [command options] target

COMMANDS:
   image, i          scan an image
   filesystem, fs    scan local filesystem
   repository, repo  scan remote repository
   client, c         client mode
   server, s         server mode
   config, conf      scan config files
   plugin, p         manage plugins
```

Trivy를 활용한 컨테이너 취약점 진단

Trivy를 활용한 도커 이미지 취약점 진단

- Trivy에 image 명령을 사용하면 도커 이미지의 취약점을 진단
- 알려진 컴포넌트에 대한 버전별 취약점 DB를 다운로드
- 현재 구성되어 있는 다양한 모듈들의 취약점을 도출

```
$ docker run --rm -v trivy-cache:/root/.cache/ \
-v /var/run/docker.sock:/var/run/docker.sock \
aquasec/trivy:latest \
image gasbugs/http-go
```

Trivy를 활용한 컨테이너 취약점 진단

Trivy 결과 확인

```
2021-09-19T05:57:18.809Z    [34mINFO[0m    Detected OS: debian
2021-09-19T05:57:18.809Z    [34mINFO[0m    Detecting Debian vulnerabilities...
2021-09-19T05:57:18.855Z    [34mINFO[0m    Number of language-specific files: 0

gasbugs/http-go (debian 10.0)
=============================
Total: 1217 (UNKNOWN: 5, LOW: 65, MEDIUM: 644, HIGH: 451, CRITICAL: 52)

+---------------------------+------------------+----------+-------------------+-----------------------+-----------------------------------------------------------+
|          LIBRARY          | VULNERABILITY ID | SEVERITY | INSTALLED VERSION |     FIXED VERSION     |                           TITLE                           |
+---------------------------+------------------+----------+-------------------+-----------------------+-----------------------------------------------------------+
| apt                       | CVE-2020-27350   | MEDIUM   | 1.8.2             | 1.8.2.2               | apt: integer overflows and underflows                     |
|                           |                  |          |                   |                       | while parsing .deb packages                               |
|                           |                  |          |                   |                       | -->avd.aquasec.com/nvd/cve-2020-27350                     |
+                           +------------------+          +                   +-----------------------+-----------------------------------------------------------+
|                           | CVE-2020-3810    |          |                   | 1.8.2.1               | Missing input validation in                               |
|                           |                  |          |                   |                       | the ar/tar implementations of                             |
|                           |                  |          |                   |                       | APT before version 2.1.2...                               |
|                           |                  |          |                   |                       | -->avd.aquasec.com/nvd/cve-2020-3810                      |
+                           +------------------+----------+                   +-----------------------+-----------------------------------------------------------+
|                           | CVE-2011-3374    | LOW      |                   |                       | It was found that apt-key in apt,                         |
|                           |                  |          |                   |                       | all versions, do not correctly...                         |
|                           |                  |          |                   |                       | -->avd.aquasec.com/nvd/cve-2011-3374                      |
+---------------------------+------------------+----------+-------------------+-----------------------+-----------------------------------------------------------+
| bash                      | CVE-2019-18276   | HIGH     | 5.0-4             |                       | bash: when effective UID is not                           |
|                           |                  |          |                   |                       | equal to its real UID the...                              |
|                           |                  |          |                   |                       | -->avd.aquasec.com/nvd/cve-2019-18276                     |
+---------------------------+------------------+----------+-------------------+-----------------------+-----------------------------------------------------------+
```

kube-bench를 활용한 쿠버네티스 보안 점검

kube-bench를 활용한 쿠버네티스 보안 점검

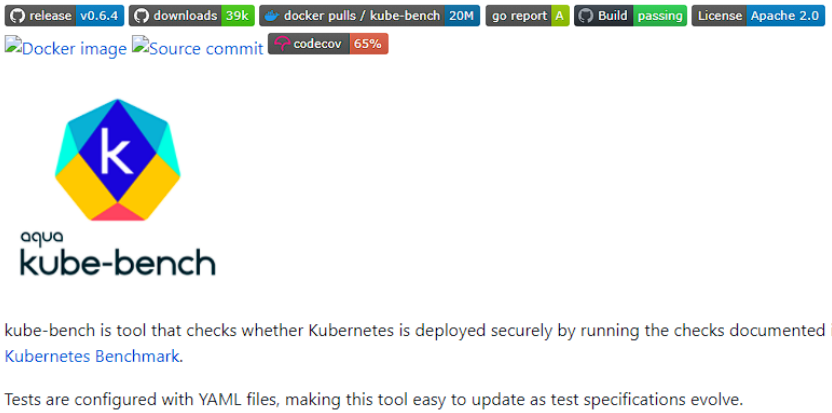
쿠버네티스 CIS 벤치마크 소개

- CIS에서는 클라우드에서 준수해야하는 다양한 보안 점검 리스트를 benchmark로 제공한다. 다음 사이트를 통해 상세히 명시된 pdf 파일을 다운로드할 수 있다. (개인정보 입력 필요)
- https://www.cisecurity.org/benchmark/kubernetes/

kube-bench를 활용한 쿠버네티스 보안 점검

kube-bench?

- aquasecurity에서는 쿠버네티스에서 사용할 수 있는 kube-bench라는 오픈소스를 구성해 많은 사용자들이 쿠버네티스에 설정 취약성을 파악할 수 있도록 구성
- 벤치마크에서 모든 부분을 구현하지는 않았지만 그래도 상당한 부분을 자동화
- 깃헙의 릴리즈 페이지를 통해 다양한 플랫폼에서 컴파일된 파일을 구성
- https://github.com/aquasecurity/kube-bench/releases




```
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 API Server
[FAIL] 1.1.1 Ensure that the --allow-privileged argument is set to false (Scored)
[FAIL] 1.1.2 Ensure that the --anonymous-auth argument is set to false (Scored)
[PASS] 1.1.3 Ensure that the --basic-auth-file argument is not set (Scored)
[PASS] 1.1.4 Ensure that the --insecure-allow-any-token argument is not set (Scored)
[FAIL] 1.1.5 Ensure that the --kubelet-https argument is set to true (Scored)
[PASS] 1.1.6 Ensure that the --insecure-bind-address argument is not set (Scored)
[PASS] 1.1.7 Ensure that the --insecure-port argument is set to 0 (Scored)
[PASS] 1.1.8 Ensure that the --secure-port argument is not set to 0 (Scored)
[FAIL] 1.1.9 Ensure that the --profiling argument is set to false (Scored)
[FAIL] 1.1.10 Ensure that the --repair-malformed-updates argument is set to false (Scored)
[PASS] 1.1.11 Ensure that the admission control policy is not set to AlwaysAdmit (Scored)
[FAIL] 1.1.12 Ensure that the admission control policy is set to AlwaysPullImages (Scored)
[FAIL] 1.1.13 Ensure that the admission control policy is set to DenyEscalatingExec (Scored)
[FAIL] 1.1.14 Ensure that the admission control policy is set to SecurityContextDeny (Scored)
[PASS] 1.1.15 Ensure that the admission control policy is set to NamespaceLifecycle (Scored)
[FAIL] 1.1.16 Ensure that the --audit-log-path argument is set as appropriate (Scored)
[FAIL] 1.1.17 Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Scored)
[FAIL] 1.1.18 Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Scored)
[FAIL] 1.1.19 Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Scored)
[PASS] 1.1.20 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)
[PASS] 1.1.21 Ensure that the --token-auth-file parameter is not set (Scored)
[FAIL] 1.1.22 Ensure that the --kubelet-certificate-authority argument is set as appropriate (Scored)
```

kube-bench를 활용한 쿠버네티스 보안 점검

클러스터 진단

- kubectl이 실행한 가능한 환경에서 kube-bench를 설치
- 릴리즈로부터 파일을 다운로드 받고 실행

```
# 릴리즈 파일 다운로드
wget https://github.com/aquasecurity/kube-bench/releases/download/v0.6.3/kube-bench_0.6.3_linux_amd64.tar.gz
tar  - xf kube-bench_0.6.3_linux_amd64.tar.gz
sudo mv kube-bench /usr/bin/ # 설치 완료

# 깃헙 프로젝트 다운로드
git clone https://github.com/aquasecurity/kube-bench
cd kube-bench
kube-bench --config-dir `pwd`/cfg --config `pwd`/cfg/config.yaml
```

```
gasbugs21c@cloudshell:~/kube-bench (gkesecurity)$ kube-bench --config-dir `pwd`/cfg --config `pwd`/cfg/config.yaml
[INFO] 4 Worker Node Security Configuration
[INFO] 4.1 Worker Node Configuration Files
[FAIL] 4.1.1 Ensure that the kubelet service file permissions are set to 644 or more restrictive (Automated)
[FAIL] 4.1.2 Ensure that the kubelet service file ownership is set to root:root (Automated)
[PASS] 4.1.3 If proxy kubeconfig file exists ensure permissions are set to 644 or more restrictive (Manual)
[PASS] 4.1.4 Ensure that the proxy kubeconfig file ownership is set to root:root (Manual)
[FAIL] 4.1.5 Ensure that the --kubeconfig kubelet.conf file permissions are set to 644 or more restrictive (Automated)
[WARN] 4.1.6 Ensure that the --kubeconfig kubelet.conf file ownership is set to root:root (Manual)
[WARN] 4.1.7 Ensure that the certificate authorities file permissions are set to 644 or more restrictive (Manual)
[WARN] 4.1.8 Ensure that the client certificate authorities file ownership is set to root:root (Manual)
[FAIL] 4.1.9 Ensure that the kubelet --config configuration file has permissions set to 644 or more restrictive (Automated)
[FAIL] 4.1.10 Ensure that the kubelet --config configuration file ownership is set to root:root (Automated)
[INFO] 4.2 Kubelet
[FAIL] 4.2.1 Ensure that the anonymous-auth argument is set to false (Automated)
[FAIL] 4.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)
[FAIL] 4.2.3 Ensure that the --client-ca-file argument is set as appropriate (Automated)
[WARN] 4.2.4 Ensure that the --read-only-port argument is set to 0 (Manual)
[WARN] 4.2.5 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Manual)
[FAIL] 4.2.6 Ensure that the --protect-kernel-defaults argument is set to true (Automated)
[FAIL] 4.2.7 Ensure that the --make-iptables-util-chains argument is set to true (Automated)
[WARN] 4.2.8 Ensure that the --hostname-override argument is not set (Manual)
[WARN] 4.2.9 Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture (Manual)
[WARN] 4.2.10 Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Manual)
[WARN] 4.2.11 Ensure that the --rotate-certificates argument is not set to false (Manual)
[WARN] 4.2.12 Verify that the RotateKubeletServerCertificate argument is set to true (Manual)

== Summary total ==
2 checks PASS
10 checks FAIL
35 checks WARN
0 checks INFO
```

kube-bench를 활용한 쿠버네티스 보안 점검

노드 진단

- GKE에서 실행하는 경우 SSH 로 접속해서 실행
- 클라우드 쿠버네티스에서는 COS를 사용하므로 읽기 전용 구간이 많음
- GKE에서는 /home/Kubernetes/bin에 설정이 가능

```
# 릴리즈 파일 다운로드
wget https://github.com/aquasecurity/kube-bench/releases/download/v0.6.3/kube-bench_0.6.3_linux_amd64.tar.gz
tar -xf kube-bench_0.6.3_linux_amd64.tar.gz
sudo mv kube-bench /home/kubernetes/bin

# 깃헙 프로젝트 다운로드
git clone https://github.com/aquasecurity/kube-bench
cd kube-bench
kube-bench --config-dir `pwd`/cfg --config `pwd`/cfg/config.yaml
```

```
== Summary total ==
9 checks PASS
5 checks FAIL
33 checks WARN
0 checks INFO
```

falco를 활용한 쿠버네티스 컨테이너 보안 모니터링

falco를 활용한 쿠버네티스 컨테이너 보안 모니터링

falco 개요

- 클라우드 네이티브 런타임 보안 프로젝트인 Falco는 사실상 Kubernetes 위협 탐지 엔진
- 2016년 Sysdig에 의해 만들어졌으며 인큐베이션 수준 프로젝트
- CNCF에 합류한 최초의 런타임 보안 프로젝트
- Falco는 예기치 않은 애플리케이션 동작을 감지하고 런타임 시 위협에 대해 경고
- CKS 시험에서도 나올 정도로 CNCF에서 밀고 있는 프로그램

Falco, the cloud-native runtime security project, is the de facto **Kubernetes threat detection engine**

Falco was created by Sysdig in 2016 and is the first runtime security project to join CNCF as an incubation-level project. Falco detects unexpected application behavior and alerts on threats at runtime.

https://falco.org/

falco를 활용한 쿠버네티스 컨테이너 보안 모니터링

호스트에 falco 설정 정보 확인

- log_syslog: true 옵션은 /var/log/syslog에 falco 관련 로그를 함께 로깅한다는 것을 의미
- rules_file 섹션에는 실행할 때 참조하는 룰의 위치가 명시됨

```
$ sudo vim /etc/falco/falco.yaml
# 룰 파일 목록
rules_file:
  - /etc/falco/falco_rules.yaml
  - /etc/falco/falco_rules.local.yaml
  - /etc/falco/k8s_audit_rules.yaml
  - /etc/falco/rules.d


...
# 로그 저장 형식
log_stderr: true
log_syslog: true
```

falco 로그

- 다음 명령으로 /var/log/syslog에서 확인

```
$ sudo cat /var/log/syslog | grep falco
Sep 22 03:33:40 node01 kernel: [11782.615640] falco: loading out-of-tree module taints
Sep 22 03:33:40 node01 kernel: [11782.616590] falco: module verification failed: signat
Sep 22 03:33:40 node01 kernel: [11782.618203] falco: driver loading, falco 17f5df52a7d9
Sep 22 03:33:40 node01 falco: Falco version 0.29.1 (driver version 17f5df52a7d9ed6bb12c
Sep 22 03:33:40 node01 falco[85088]: Wed Sep 22 03:33:40 2021: Falco version 0.29.1 (dr
Sep 22 03:33:40 node01 falco: Falco initialized with configuration file /etc/falco/falc
Sep 22 03:33:40 node01 falco[85088]: Wed Sep 22 03:33:40 2021: Falco initialized with c
Sep 22 03:33:40 node01 falco: Loading rules from file /etc/falco/falco_rules.yaml:
Sep 22 03:33:40 node01 falco[85088]: Wed Sep 22 03:33:40 2021: Loading rules from file
Sep 22 03:33:40 node01 falco: Loading rules from file /etc/falco/falco_rules.local.yaml
Sep 22 03:33:40 node01 falco[85088]: Wed Sep 22 03:33:40 2021: Loading rules from file
Sep 22 03:33:40 node01 falco: Loading rules from file /etc/falco/k8s_audit_rules.yaml:
Sep 22 03:33:40 node01 falco[85088]: Wed Sep 22 03:33:40 2021: Loading rules from file
```

falco에서 지원하는 필드

- 생성하는 로그를 어떤 정보를 매칭해 어떤 데이터를 저장할 지 필드
- 필드의 개수가 워낙 많고, 많은 정보를 내포
- 너무 많아 모든 것을 다 외우기는 어려움

Name	Type	Description
`evt.num`	UINT64	event number.
`evt.time`	CHARBUF	event timestamp as a time string that includes the nanosecond part.
`evt.time.s`	CHARBUF	event timestamp as a time string with no nanoseconds.
`evt.time.iso8601`	CHARBUF	event timestamp in ISO 8601 format, including nanoseconds and time zone offset (in UTC).
`evt.datetime`	CHARBUF	event timestamp as a time string that includes the date.
`evt.rawtime`	ABSTIME	absolute event timestamp, i.e. nanoseconds from epoch.
`evt.rawtime.s`	ABSTIME	integer part of the event timestamp (e.g. seconds since epoch).
`evt.rawtime.ns`	ABSTIME	fractional part of the absolute event timestamp.
`evt.reltime`	RELTIME	number of nanoseconds from the beginning of the capture.
`evt.reltime.s`	RELTIME	number of seconds from the beginning of the capture.

다양한 클래스를 지원

Create child page
Create documentation issue
Create project issue

System Calls (source `syscall`)
- Field Class: evt
- Field Class: process
- Field Class: user
- Field Class: group
- Field Class: container
- Field Class: fd
- Field Class: syslog
- Field Class: fdlist
- Field Class: k8s
- Field Class: mesos
- Field Class: span
- Field Class: evtin

Kubernetes Audit Events (source `k8s_audit`)

클래스 내부에는 다수의 필드를 포함

falco 룰 예제

- falco 사이트에 몇가지 예제를 제공
- https://falco.org/docs/examples/

룰 조건
1) container and spawned_process and: 앞서 생성한 두 macro는 참이어야 하고
2) proc.name = bash and: 프로세스 이름은 bash여야 하며,
3) proc.pname exists and: 부모 프로세스 이름이 있어야하고
4) not proc.pname in (bash, docker): 부모 프로세스 이름은 bash나 docker여서는 안된다.

A shell is run in a container Rule

```
- macro: container
  condition: container.id != host

- macro: spawned_process
  condition: evt.type = execve and evt.dir=<

- rule: run_shell_in_container
  desc: a shell was spawned by a non-shell program in a container. Container entrypoints are excluded.
  condition: container and proc.name = bash and spawned_process and proc.pname exists and not proc.pname in (bash,
docker)
  output: "Shell spawned in a container other than entrypoint (user=%user.name container_id=%container.id
container_name=%container.name shell=%proc.name parent=%proc.pname cmdline=%proc.cmdline) “
  priority: WARNING
```

간단한 로그 생성과 탐지

- python 이미지 컨테이너를 구성하고 패키지 매니저를 통해 vim을 설치

```
kubectl run py --image=python:3.7  -- sleep infinity
kubectl exec py -- apt update
kubectl exec py -- apt install -y vim
```

- syslog에 falco를 grep하면 관련 로그 확인 가능

```
# cat /var/log/syslog ¦ grep falco
Sep 22 05:44:13 node01 falco[123799]: 05:44:13.306534965: Error Package management process launched
in container (user=root user_loginuid=-1 command=apt update container_id=4831e292a3c8
container_name=k8s_py_py_default_13877cb8-d825-4831-a8ca-f21a9d79b7f0_0 image=python:3.7)
Sep 22 05:44:13 node01 falco: 05:44:13.306534965: Error Package management process launched in
container (user=root user_loginuid=-1 command=apt update container_id=4831e292a3c8
container_name=k8s_py_py_default_13877cb8-d825-4831-a8ca-f21a9d79b7f0_0 image=python:3.7)
Sep 22 05:45:02 node01 falco[123799]: 05:45:02.431342562: Error Package management process launched
in container (user=root user_loginuid=-1 command=apt install -y vim container_id=48f80132919f
container_name=k8s_py_py_default_2293f44d-dfd3-4508-b001-9fc0d43a4c9e_0 image=python:3.7)
Sep 22 05:45:02 node01 falco: 05:45:02.431342562: Error Package management process launched in
container (user=root user_loginuid=-1 command=apt install -y vim container_id=48f80132919f
container_name=k8s_py_py_default_2293f44d-dfd3-4508-b001-9fc0d43a4c9e_0 image=python:3.7)
```