JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 26, No. 7: 1047~1059, Jul. 2022

STRIDE 위협 모델링에 기반한 클라우드 컴퓨팅의 쿠버네티스(Kubernetes)의 보안 요구사항에 관한 연구

이승욱[1] · 이재우[2*]

Kubernetes of cloud computing based on STRIDE threat modeling

Seungwook Lee[1] · Jaewoo Lee[2*]

[1]Graduate Student, Department of Convergence Security, Chung-Ang University, Seoul, 06974 Korea
[2*]Assistant Professor, Department of Industrial Security, Chung-Ang University, Seoul, 06974 Korea

요 약

클라우드 컴퓨팅 기술의 발전으로 가상 환경을 기반으로 서비스를 제공하는 컨테이너 기술 또한 발전하고 있다. 컨테이너 오케스트레이션 기술은 클라우드 서비스를 위한 핵심적인 요소이며, 대규모로 구성된 컨테이너를 빌드, 배포, 테스트하는데 자동화로 관리하기 위한 중요한 핵심 기술이 되었다. 최초 구글에 의해 설계되었고, 현재 리눅스 재단에 의해 관리되고 있는 쿠버네티스는 컨테이너 오케스트레이션 중에 하나이며 사실상 표준으로 자리 매김을 하고 있다. 하지만 컨테이너 오케스트레이션 중 쿠버네티스의 사용이 증가하고 있음에도 불구하고, 보안 취약점에 의한 사고사례도 또한 증가하고 있다. 이에 본 논문에서는 쿠버네티스의 취약점을 연구하고, 위협 분석을 통해 개발 초기 또는 설계 단계에서부터 보안을 고려할 수 있는 보안 정책을 제안한다. 특히, STRIDE 위협 모델링을 적용하여 보안 위협을 분류함으로써 구체적인 보안 가이드를 제시하고자 한다.

ABSTRACT

With the development of cloud computing technology, container technology that provides services based on a virtual environment is also developing. Container orchestration technology is a key element for cloud services, and it has become an important core technology for building, deploying, and testing large-scale containers with automation. Originally designed by Google and now managed by the Linux Foundation, Kubernetes is one of the container orchestrations and has become the de facto standard. However, despite the increasing use of Kubernetes in container orchestration, the number of incidents due to security vulnerabilities is also increasing. Therefore, in this paper, we study the vulnerabilities of Kubernetes and propose a security policy that can consider security from the initial development or design stage through threat analysis. In particular, we intend to present a specific security guide by classifying security threats by applying STRIDE threat modeling.

키워드 : 클라우드 네이티브, 컨테이너 오케스트레이션, 쿠버네티스, 스트라이드 위협 모델링

Keywords : Cloud Native, Container Orchestration, Kubernetes, STRIDE threat modeling

Received 26 April 2022, **Revised** 9 May 2022, **Accepted** 27 May 2022

*** Corresponding Author** Jaewoo Lee(E-mail: jaewoolee@cau.ac.kr, Tel: +82-2-820-5935)
Assistant Professor, Department of Industrial Security, Chung-Ang University, Seoul, 06974 Korea

Open Access http://doi.org/10.6109/jkiice.2022.26.7.1047 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

쿠버네티스(Kubernetes)는 컨테이너화된 애플리케이션의 배포, 확장 등과 같은 관리를 자동화하기 위한 플랫폼(컨테이너 오케스트레이션 엔진)이다. 시스디그(Sysdig, Inc)가 공개한 'Sysdig 2019 container usage report'에 따르면, 컨테이너 오케스트레이션 엔진 시장 점유율에 쿠버네티스가 약 77%, 오픈시프트(Openshift)가 약 9%, 도커 스웜(Docker Swarm)이 약 5%를 차지하고 있다[1]. 쿠버네티스는 컨테이너 오케스트레이션 엔진 분야에서 사실상 표준이 되었다고 할 수 있으며, 앞으로 클라우드 네이티브 환경에서의 개발이 더욱 보편화되면서 반드시 사용해야 하는 플랫폼이 될 것이다[2].

개방과 공유, 수많은 컨테이너의 유기적 결합, 마이크로서비스 아키텍처, 워크로드의 자동배포 및 증설 등, 클라우드 네이티브 컴퓨팅을 정의하는 수많은 수식어들을 다른 시각, 즉 사이버 보안의 시각에서 바라보면 이런 수식어 자체가 엄청난 잠재적 보안 위협 요소들을 내포하고 있음을 짐작할 수 있다. 개방과 공유정신에 반하는 악의적 개발자의 백도어(back door) 가능성, 오염된 컨테이너를 통한 전체 시스템에 대한 위협, 자동 배포 단계에서의 작동 오류 등이 바로 이러한 위협들이다 [3].

이에 본 논문에서는 쿠버네티스의 보안 취약점을 연구하고, STRIDE 위협 모델링을 통해 인증(S : 위장(Spoofing)), 무결성(T : 데이터 변조(Tampering with data)), 부인 방지(R : 부인(Repudiation)), 기밀성(I : 정보유출(Information Disclosure)), 가용성(D : 서비스 거부(Denial of Service)), 인가(E : 권한상승(Elevation of Privilege))에 따라 위험을 분류하고, Attack Tree를 구성하여 쿠버네티스의 자산 또는 대상이 어떻게 공격을 받을 수 있는지 분석하여 대응 방안을 제시한다. 이를 통해 클라우드 컴퓨팅 환경에서 컨테이너 오케스트레이션을 안전하게 이용할 수 있도록 기여하고자 한다.

세부적인 연구 목표는 첫째, 클라우드 네이티브(Cloud Native) 보안 4C(클라우드(Cloud), 컨테이너(Container), 클러스터(Cluster), 코드(Code))에 대해 STRIDE 위협 분석 모델링을 적용하여 위협을 식별 및 대응 방안을 제시하고, 둘째, 최종적으로 쿠버네티스에 대한 보안 요구사항 및 대응방안을 도출하는 것을 목표로 한다.

Ⅱ. 선행 연구

2.1. 쿠버네티스 개념 정리

쿠버네티스는 컨테이너화된 애플리케이션의 자동 디플로이(Deploy), 스케줄링(Scheduling) 등을 제공하는 관리시스템으로, 오픈소스 기반이며, 구글에 의해 설계되었고 현재 리눅스 재단이 관리하고 있다. 목적은 여러 클러스터의 호스트 간에 애플리케이션 컨테이너의 배치, 스케줄링, 운영을 자동화하기 위한 플랫폼을 제공하기 위함이며, 도커를 포함하여 일련의 컨테이너 도구들과 함께 동작하는 것을 말한다 [4].

컨테이너화된 애플리케이션을 자동 배포, 스케줄링 등을 제공하며, 유연성과 확장성이 용이하고, 멀티 하이브리드 클라우드 플랫폼(Multi-hybrid Cloud Platform)의 기반이라고 볼 수 있다.

쿠버네티스 공식문서에 따른 쿠버네티스 아키텍처 및 클러스터 구성 요소는 그림 1과 같으며, 주요 구성 요소로는 표 1과 같다. 쿠버네티스는 크게 마스터 노드(Master Node)와 일반 노드(Worker Node)로 구성되어 있다. 마스터 노드는 쿠버네티스 전체를 관리하는 역할을 한다. 노드에서 발생하는 다양한 이벤트를 감지하고 응답하는 등 의사결정을 수행한다. 마스터 노드는 여러 개의 노드로 구성될 수 있으므로 단일 마스터가 아닌 이중화 또는 삼중화 형태로도 구성할 수 있다. 다음과 같이 마스터 노드는 API server, controller manager, scheduler, etcd 등으로 구성된다. 일반 노드는 컨테이너가 실행되는 VM 또는 실제 머신을 의미한다. kubelet이라는 에이전트를 통해 마스터 노드와 통신하며, 실제 컨테이너인 파드(pod)가 생성되는 곳이다. 일반 노드 안에는 kubelet, kube-proxy 등 구성되어 있다 [5].

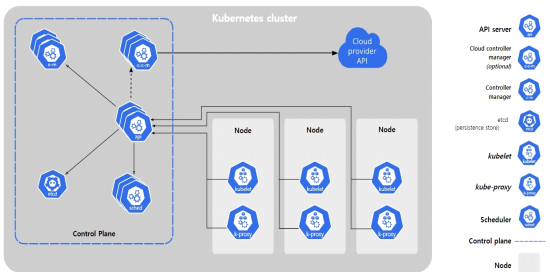


Fig. 1 Kubernetes Architecture

Table. 1 Kubernetes main components

Division		Detail
Master Node	API Server	• A control plane component that exposes the Kubernetes API.
	Controller manager	• The control plane component that runs the controller process
	Scheduler	• A control plane component that detects newly created Pods that have no nodes assigned to them and selects which nodes to run on.
	etcd	• Consistency/high availability key-value store used as data store for all cluster data
Worker Node	Kubelet	• Agents running on each node in the cluster • Manage containers to reliably operate in Pods
	kube -proxy	• A network proxy running on each node in the cluster to maintain the node's network rules

2.2. 클라우드 네이티브 보안

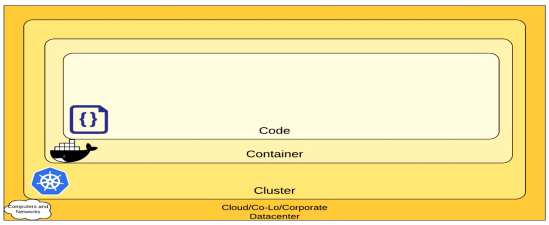


Fig. 2 Cloud Native Security 4Cs

쿠버네티스 공식문서에 따른 클라우드 네이티브 보안의 4C는 그림 2와 같으며, 4개의 계층으로 클라우드, 클러스터, 컨테이너와 코드로 구분할 수 있다 [6].

첫번째, 클라우드 계층에서는 클라우드 계층이 취약하거나 취약한 방식으로 구성된 경우 이 기반 위에서 구축된 구성 요소가 안전하다는 보장은 없다. 각 클라우드 공급자는 해당 환경에서 워크로드를 안전하게 실행하기 위한 보안 권장사항을 제안해야 하며, 인프라스트럭처(Infrastructure) 보안에서 고려해야 할 부분으로는 API 서버에 대한 네트워크 접근(컨트롤 플레인), 노드에 대한 네트워크 접근(노드), 클라우드 공급자 API에 대한 쿠버네티스 접근, etcd에 대한 접근, etcd 암호화가 있다.

두번째, 클러스터 계층에서는 설정 가능한 클러스터 컴포넌트의 보안과 클러스터에서 실행되는 애플리케이션의 보안으로 나뉠 수 있으며, 워크노드 보안에서 고려해야할 부분으로는 RBAC 인증(쿠버네티스 API에 대한 접근), 인증, 애플리케이션 시크릿 관리(및 유휴 상태에서의 etcd 암호화 등), 파드가 Pod Security Policy를 만족하는지 확인하기, 서비스 품질(및 클러스터 리소스 관리), 네트워크 정책, 쿠버네티스 인그레스를 위한 TLS가 있다.

세번째, 컨테이너 계층에서는 컨테이너 취약점 스캔 및 OS에 종속적인 보안, 이미지 서명 및 시행, 권한 있는 사용자의 비허용, 더 강력한 격리로 컨테이너 런타임 사용에 대해 고려해야 한다.

마지막으로 코드 계층의 경우엔 가장 많은 제어를 할 수 있는 주요 공격 영역 중 하나로 TLS를 통한 접근, 통신 포트 범위 제한, 타사 종속성 보안, 정적 코드 분석, 동적 탐지 공격에 대해 고려해야 한다.

쿠버네티스는 기본적으로 보안을 지원하지 않는다. 다시말해 보안 설정을 할 수 있는 기능은 제공하지만 관리자가 설정을 하지 않으면 보안에 취약할 수 밖에 없다. 예를 들면, API 서버에 기본적으로 암호화되지 않은 base64로 인코딩된 문자열로 Secret을 저장되므로 액세스 권한이 있는 모든 사용자는 검색할 수 있기 때문에 TLS 암호화 설정 또는 RBAC을 구현해야 한다. 또한, 클러스터 리소스의 접근에 대한 사용자 인증은 Kubernetes의 기본 제공 기능이 아니므로, 관리자 및 일반 사용자 계정과 서비스 계정에 대해 인증 및 권한 부여 매커니즘을 구현해야하며, Kubelet는 기본적으로 Master의 APIServer에서 전달되는 요청에 대해 권한 검사 없이 모두 허용하므로, Kubelet용 RBAC을 구현해야하는 등 쿠버네티스에 대해 정확하게 이해하고 구축 및 설계 단계에서부터 보안적인 측면을 고려해야 한다.

2.3. 쿠버네티스에 대한 기존 연구 분석

강혜민 외 6명은 '도커 쿠버네티스 환경에서의 웹 서버 보안 정책 연구'[7]에서 도커 및 쿠버네티스의 취약점을 연구하고, 이를 보완할 수 있는 보안설정에 기초한 보안 정책을 제안하였으며, 특히 정보보호 핵심 요소인 CIA를 기반으로 새로운 보안 가이드를 제시하였다. 쿠버네티스에 대해 Kubernetes Infrastructure security, Kubernetes Cluster Security, Kubernetes Container Security, Kubernetes Code Security 영역으로 분류하였으며, SK Infosec의 클라우드 보안가이드, Kubernetes 공식문서의 정책을 참고하여 CIA를 분류하였으나, 인증, 인가 및 부인방지에 따른 보안 위협(위장, 권한상승,

부인)에 대한 분석도 요구되어 진다.

Ⅲ. STRIDE 위협 모델링 기반 쿠버네티스 위협 분석

3.1. 위협 모델링 방법론

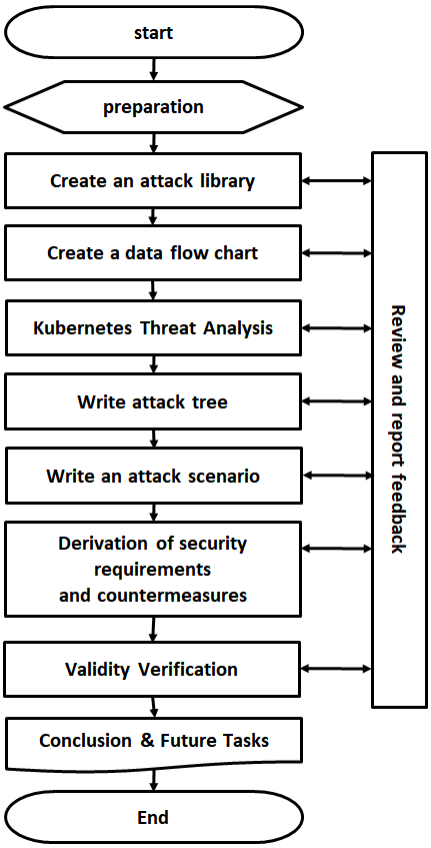


Fig. 3 Threat modeling process

쿠버네티스에 적용하는 위협 모델링 절차는 그림 3과 같은 형태로 진행된다. Attack Library는 분석 대상에서 발생 가능한 공격들의 리스트로 신뢰할 수 있는 기관의 쿠버네티스 관련 공식문서 또는 기술 자료, 선행 연구, 컨퍼런스 발표, 알려진 취약점(CVE) 등을 통해 수집한다. 이를 이용하여 데이터 흐름 상의 위협 요소들을 식별하며, 위협 식별 과정에서 중복되는 부분을 줄이고, Attack Tree를 통해 쿠버네티스를 공격 할 수 있는 방법을 구체화 시킨다. 이후 공격 벡터를 선정하여 취약점 분석을 통해 취약점 체크리스트를 작성하고, 이에 대한 보안 요구사항 도출 및 대응 방안을 제시하며, 도출된 취약점 체크리스트에 대한 실효성을 검증하여 효과성을 입증 한다. 끝으로 위협 모델링 절차에 따른 결론 및 향후 과제를 제시함으로써 마무리 한다.

3.2. 공격 라이브러리

쿠버네티스 공격 라이브러리는 크게 논문, CVE, 쿠버네티스 관련 공식 문서를 수집하였다. 총 5개의 논문 및 총 3개의 쿠버네티스 관련 공식문서와 총 12개의 CVE를 수집하였고, 아래의 표 2, 표 3, 표 4와 같다.

CVE와 같은 경우는 2015년부터 발견된 취약점은 총 140개이며, 본 논문에서는 2022년에 공개된 취약점 13개를 대상으로 연구하였다.

Table. 2 Attack Library: Papers

Vulnerability Analysis Paper			
Author	Title	Year	Ref
Kang, Hye-Min	A Study on Web Server Security Policy in Docker Kubernetes Environment	2020	[7]
Mytilinakis Panagiotis	Attack methods and defenses on Kubernetes	2020	[8]
Nadin Habbal	Enhancing Availability of Microservice Architecture	2020	[9]
Tuomas Autio	Securing a Kubernetes Cluster on GoogleCloudPlatform	2021	[10]
Thomas Fowley	Security of Virtual Infrastructures	2021	[11]

Table. 3 Attack library: Official document

Official Document			
Source	Title	Year	Ref
SK Infosec	Cloude Security Guide (Container Security) - Docker, Kubernetes	2019	[12]
Kubernetes	Cloud Native Security Overview	2021	[13]
NSC/CISA	Kubernetes Hardening Guidance	2021	[14]

Table. 4 Attack library: CVE

CVE		
No.	Type	Ref
CVE -2022 -27211	• CWE-862 : authentication missing • Vulnerability Type: Confidentiality	[15]

CVE		
No.	Type	Ref
CVE -2022 -27210	• CWE-352 : CSRF • Vulnerability type: confidentiality, access complexity	[16]
CVE -2022 -27209	• CWE-862 : authentication missing • Vulnerability Type: Confidentiality	[17]
CVE -2022 -27208	• CWE-22: Improper restriction of pathnames for restricted directories (path traversal) • Vulnerability Type: Confidentiality	[18]
CVE -2022 -26311	• Exposure of sensitive information to unauthorized actors • Vulnerability Type: Confidentiality	[19]
CVE -2022 -24768	• CWE-200: Exposing sensitive information to unauthorized actors • Vulnerability types: confidentiality, integrity, availability	[20]
CVE -2022 -24731 CVE -2022 -24730	• CWE-22: Improper restriction of path names for restricted directories (path traversal) • CWE-284: Improper access control • Vulnerability Type: Confidentiality	[21]
CVE -2022 -23652	• CWE-287 : Inappropriate authentication • Vulnerability types: confidentiality, integrity, availability	[22]
CVE -2022 -23648	• CWE-200: Exposing sensitive information to unauthorized actors • Vulnerability Type: Confidentiality	[23]
CVE -2022 -21701	• CWE-863: Invalid Permission Granted • Vulnerability types: confidentiality, integrity, availability	[24]
CVE -2022 -0811	• CWE-94: Inadequate control of code generation (code injection) • Vulnerability types: confidentiality, integrity, availability	[25]
CVE -2022 -0270	• CWE-284: Improper access control • Vulnerability types: confidentiality, integrity, availability	[26]

3.3. DFD 도출

본 장에서는 STRIDE 위협 모델링을 진행하기 위해 클라우드 기반 쿠버네티스 서비스 아키텍처를 바탕으로 데이터 흐름 다이어그램을 도출한다.

3.3.1. 최상위 레벨 DFD 도출

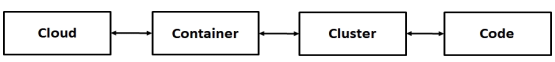


Fig. 4 Top-level data flow diagram

클라우드 기반 쿠버네티스 서비스의 데이터 흐름 다이어그램의 도출은 그림 4의 아키텍처 기반으로 진행하였다. 데이터 흐름 다이어그램의 도출 과정은 현재 쿠버네티스 공식문서의 클라우드 네이티브 보안의 4C(클라우드, 클러스터, 컨테이너, 코드)를 도식화로부터 시작하여, 각 요소 내부의 구성을 고려하여 세분화하는 과정으로 진행하였다.

클라우드 기반 쿠버네티스 서비스 아키텍처는 컨테이너화된 애플리케이션을 효율적으로 빌드, 배포 및 테스트 하기위해 설계된 오픈소스 플랫폼으로 클라우드 네이티브 보안의 핵심 요소인 클라우드, 클러스터, 컨테이너, 코드에 대해 각 요소별 보안에 대한 심층 방어 접근 방식으로 관리적, 물리적, 기술적으로 고려되어야 한다. 이러한 데이터와 서비스의 흐름을 바탕으로 클라우드, 컨테이너, 클러스터, 코드를 각각 하나의 외부 엔티티로 두어 그림 4와 같이 최상위 레벨의 데이터 흐름 다이어그램을 생성하였다.

3.3.2. 클라우드 인프라스트럭처(Cloud Infrastructure)의 세부 데이터 흐름 다이어그램 도출

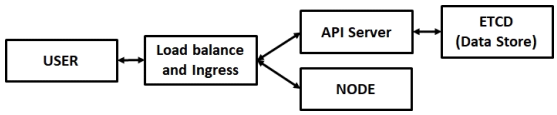


Fig. 5 Detailed data flow diagram of cloud infrastructure

최상위 레벨 다이어그램에서 클라우드 인프라스트럭처 보안은 사용자(User)가 로드밸런스(Load Balance)와 인그레스(Ingress)를 통해 쿠버네티스의 주요 구성 요소인 API 서버, etcd 서버, 노드(Node)에 접근하는데 있어 인프라 측면을 고려하여 세분화 하였다. API 서버는 쿠버네티스 Control Plane의 핵심으로, 클러스터에 속한 노드 혹은 외부의 사용자로부터 HTTP API 요청을 받아 클러스터의 상태를 관리하는 기능을 수행하며, etcd 서버는 데이터 저장소로서, 클러스터의 데이터(설정 데이터, 클러스터 상태, 메타데이터 등)를 관리하는 용도로 사용한다. 또한, 사용자는 컨테이너를 파드 내에 배치하고 노드에서 실행함으로 워크로드를 구동하고, 각 노드는 컨트롤 플레인에 의해 관리되며 파드를 실행하는 데 필요한 서비스를 포함한다. 이러한 작업의 흐름을 바탕으로 사용자를 외부 엔티티로 두며, 사용자가 인프라 측면에서 쿠버네티스에 접근하는 주요 구성요소로 API

서버, etcd 서버, 노드들과의 통신을 데이터 흐름으로 분류하여 그림 5와 같이 클라우드 인프라 스트럭처의 세부 데이터 흐름 다이어그램을 도출하였다.

3.3.3. 클러스터(Cluster)의 세부 데이터 흐름 다이어그램 도출

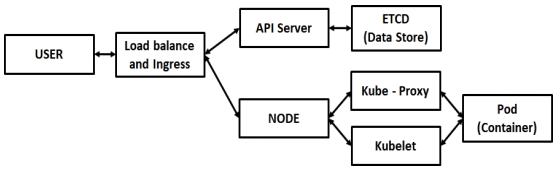


Fig. 6 Detailed data flow diagram of the cluster

최상위 레벨 다이어그램의 클러스터는 크게 컨트롤 플레인(Control Plane)과 노드(Node)로 구분할 수 있다. 컨트롤 플레인은 쿠버네티스 클러스터의 핵심 요소로서 워커노드와 클러스터 내 파드를 관리하고 클러스터에 관한 전반적인 결정을 수행하며, 구성 요소로는 쿠버네티스 API를 노출하는 API server, 데이터 저장소인 etcd, 노드 배정 및 선택을 하는 Scheduler, 컨트롤러 프로세스를 실행하는 Controller manager로 구성되어 있다. 노드는 동작 중인 파드를 유지시키고 쿠버네티스 런타임 환경을 제공하여, 모든 노드 상에서 동작한다. 구성 요소로는 kubelet, kube-proxy, Container Runtime, Add-On, DNS, 웹 UI(대시보드), 컨테이너 리소스 모니터링(Container Resource Mornituling), 클러스터-레벨 로깅(Cluster-level Logging)으로 구성되어 있다.

컨트롤 플레인에서 노드로의 통신, 노드에서 컨트롤 플레인으로의 통신 및 API 서버에서 노드, 파드, 서비스로의 통신 간에 발생되어지는 데이터의 흐름에 따라 컨트롤 플레인에서는 API 서버와 etcd를 노드에서는 파드 및 주요 통신 경로인 Kublet과 Kube-Proxy를 데이터 흐름으로 분류하여 그림 6과 같이 클러스터의 세부 데이터 흐름 다이어그램을 도출하였다.

3.3.4. 컨테이너(Container)의 세부 데이터 흐름 다이어그램 도출

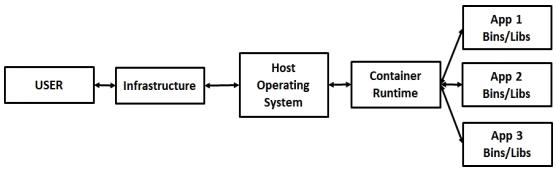


Fig. 7 Detailed data flow diagram of container

최상위 레벨 데이터 흐름 다이어그램의 컨테이너는 Host OS와 컨테이너 런타임을 통한 컨테이너 내 애플리케이션들을 구동하는 것으로 세분화 하였다. 컨테이너 이미지를 빌드, 배포 및 테스트를 하기위해 컨테이너의 취약점을 스캔하고 OS의 종속적인 보안을 강화해야 한다. 또한, 사용되어지는 컨테이너 이미지에 대해 데이터의 무결성을 보장하기 위해 이미지 서명를 통해 수행해야하며, 컨테이너 런타임을 사용하여 더 강력한 격리를 통해 컨테이너의 권한 제어를 강화해야 한다. 이러한 흐름을 바탕으로 Host OS, 컨테이너 런타임, Application들을 엔티티로 분류하여 그림 7과 같이 컨테이너의 세부 데이터 흐름 다이어그램을 도출하였다.

3.3.5. 코드(Code)의 세부 데이터 흐름 다이어그램 도출

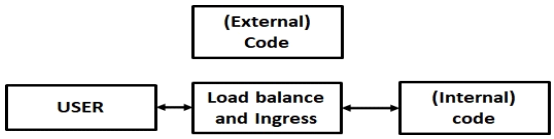


Fig. 8 Detailed data flow diagram of the code

최상위 레벨 데이터 흐름 다이어그램의 코드는 쿠버네티스의 코드에 접근하기 위해 암호화된 통신인 TLS를 통한 접근이 되어야 하며, 통신간 사용되어지는 포트 범위를 제한하여야 한다. 또한, 타사 제품과 호환시 종속적인 보안 측면도 고려해야하며, 안전하지 않은 코딩 방법에 대해 정적 코드 분석 및 알려진 공격 중 일부를 서비스에 대해 동적 탐지를 통해 취약점이 있는지 확인 및 조치를 해야 한다. 이러한 흐름을 바탕으로 내부에서 사용되는 코드와 외부로부터 유입되는 코드를 엔티티로 분류하여 그림 8과 같이 코드의 세부 데이터 흐름 다이어그램을 도출하였다.

3.3.6. 클라우드 기반 쿠버네티스의 전체 데이터 흐름 다이어그램 도출

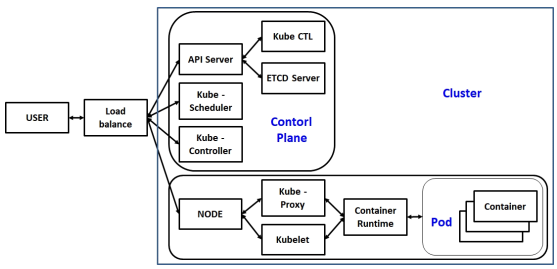


Fig. 9 Full data flow diagram in Kubernetes

앞서 도출된 각 엔티티의 세부 데이터 흐름 다이어그램을 하나로 통합하여 쿠버네티스 시스템 전체 구성을 하나의 다이어그램으로 확인할 수 있도록 하였다.

전체 데이터 흐름 다이어그램은 쿠버네티스에서 새로운 파드를 생성하는 과정에서 쿠버네티스의 동작 흐름에 따라 필요로 하는 요청과 응답, 데이터 교환 등의 작업을 수행한다. 동작 흐름 과정에서 각각의 컴포넌트들이 서로 개별적으로 통신하지 않고, API 서버를 통해 통신하며, 컴포넌트들은 각기 현재 상태를 체크하고 독립적으로 동작하게 된다. 위 과정을 통해 도출된 클라우드 기반 쿠버네티스 서비스의 전체적인 데이터 흐름 다이어그램은 그림 9와 같다.

3.4. 쿠버네티스 위협 분석

그림 10은 앞선 공격 라이브러리를 통해 위협 분석 후 쿠버네티스의 아키텍처 및 구성 요소들의 동작원리에 따라 이미지의 빌드, 배포에 따른 데이터 흐름을 기반으로 Microsoft Threat Modeling Tool 2016을 이용하여 자동 위협식별 및 도출을 진행한다. Microsoft Threat Modeling Tool은 STRIDE 접근 방식을 사용하여 보안 결함을 파악하는데 사용되는 툴이다. 그림 10과 같이 쿠버네티스 위협 모델을 구성 후 위협분석에 따라 각 위협 요소별 분류한 결과로 표 5와 같이 총 49개의 위험을 도출되었으며, 쿠버네티스의 DFD에서 워크노드는 사용자의 요구사항으로 증가할 수 있는 부분으로 워크노드가 증가하면 할수록 위협요소 또한 증가할 수 있다.

표 6에서 보듯이, 스푸핑으로 인한 무단 액세스의 위험, 안전하지 않은 통신 채널의 남용으로 인한 정보 노출, 서비스 거부 공격에 의한 가용성 문제 등이 있다.

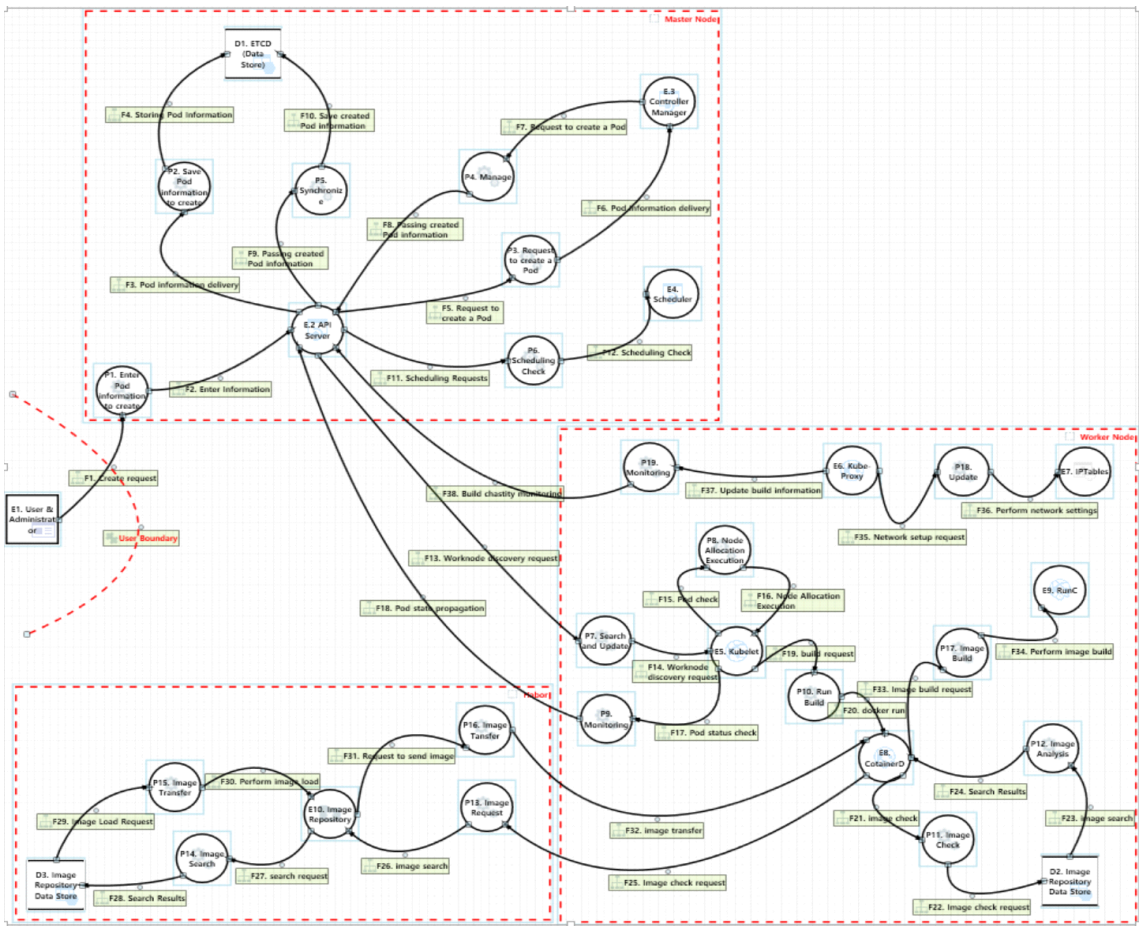


Fig. 10 Kubernetes data flow diagram

Table. 5 STRIDE by process in Kubernetes

Process No.	name	STRIDE classification	Threat number	Threat Description
P1	Enter Pod information to create	S	T1	Threats of unauthorized access through spoofing
		T	T2	Threat of inserting malicious code through malicious input
		R	T3	Threats to deny access and connections by malicious behavior
		I	T4	Threats to hijack and steal sensitive data (accounts, passwords) through sniffing
		E	T5	Threats to gain unauthorized access to web APIs due to incorrect access control
P2	Save Pod information to create	R	T6	Accountability is difficult to trace without proper auditing and logging controls.
		E	T7	Threats that could cause general-purpose users to attempt elevation of privilege through race conditions
P3	Request to create a Pod	D	T8	Threats from denial of service attacks
		E	T9	Threats to gain access through lack of request control or compromised application identity
P4	Manage	S	T10	Threats of unauthorized access through spoofing
		T	T11	Threat of inserting malicious code through malicious input
		R	T12	Threats to deny access and connections by malicious behavior
		I	T13	Threats to hijack and steal sensitive data (accounts, passwords) through sniffing
		E	T14	Threats to gain unauthorized access to web APIs due to incorrect access control
P5	Synchronize	R	T15	Difficulty in Traceability Without Appropriate Auditing and Logging Controls
		E	T16	Threats of gaining privileges due to lack of account management and appropriate access controls
P6	Scheduling check	D	T17	Threats from denial of service attacks
		E	T18	Threats to gain access through lack of request control or compromised application identity
P7	Search and Update	S	T19	Threats to gain access if an administrator's credentials are compromised
		D	T20	Threats from denial of service attacks
		E	T21	Threats to gain access through malicious internal users, lack of control over requests, or compromised application identity
P8	Node Allocation Execution	D	T22	Threats from denial of service attacks
		E	T23	Threats to gain access through lack of request control or compromised application identity
P9	Monitoring	S	T24	Threats of unauthorized access through spoofing
		T	T25	Threat of inserting malicious code through malicious input
		R	T26	Threats to deny access and connections by malicious behavior
		I	T27	Threats from obtaining sensitive data through sniffing, obtaining sensitive data through error messages, and obtaining sensitive data within configuration files
		E	T28	Threats to gain privileges from malicious insiders or lack of adequate access controls

Process No.	name	STRIDE classification	Threat number	Threat Description
P11	image check	S	T29	Threats to gain access to sensitive data if anonymous access is inadvertently provided
		R	T30	Threats to deny access and connections by malicious behavior
		E	T31	Threats of gaining privileges due to lack of account management and appropriate access controls
P13	Image Request	S	T32	Threats posed by attackers bypassing authentication due to non-standard Identity Server authentication schemes, improper logout from Identity Server allowing attackers to access user sessions, and abuse of unmanaged Identity Server signing keys
		I	T33	Threats to intercept and steal sensitive data through sniffing
		D	T34	Threats from denial of service attacks
P14	Image Check	S	T35	Threats to gain access to sensitive data if anonymous access is inadvertently provided
		R	T36	Threats to deny access and connections by malicious behavior
		E	T37	Threats of gaining privileges due to lack of account management and appropriate access controls
P15	Image Transfer	S	T38	Threats posed by attackers bypassing authentication due to non-standard Identity Server authentication schemes, improper logout from Identity Server allowing attackers to access user sessions, and abuse of unmanaged Identity Server signing keys
		I	T39	Threats to intercept and steal sensitive data through sniffing
		D	T40	Threats from denial of service attacks
P18	Update	S	T41	Threats to configure incorrect TLS parameters, store and transmit credentials in clear text, obtain sensitive information, and gain unauthorized access through spoofing
		T	T42	Remote code execution through XSLT scripting, the threat of injecting malicious code through malicious input
		R	T43	Threats to deny access and connections by malicious behavior
		I	T44	Threats from configuring weak encryption, obtaining sensitive data through sniffing, and obtaining sensitive data through error messages
P19	Monitoring	S	T45	Lack of proper authentication, threats to gain access if administrator's credentials are compromised
		T	T46	Threat of inserting malicious code through malicious input
		R	T47	Threats to deny access and connections by malicious behavior
		I	T48	Threats from obtaining sensitive data through sniffing, obtaining sensitive data through error messages, and obtaining sensitive data within configuration files
		E	T49	Threats of gaining privileges due to lack of account management and appropriate access controls

Table. 6 MS Tool threat analysis result

STRIDE	Spoofing	Tempering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege	Total
Number	22	11	9	17	6	34	99

3.5. 공격 트리(AttackTree) 작성

공격트리는 자산 또는 대상이 공격자로부터 어떻게 공격을 받을 수 있는지 보여주는 개념도로 자산 또는 대상에 대한 위협을 이해하는데 사용된다[27]. 앞선 쿠버네티스의 구성 요소들에 대해 STRIDE 분류를 통해 도출된 위협을 바탕으로 표 7과 같이 작성되었다. 공격 대상인 쿠버네티스를 루트 노드로 하고 하위 노드는 클라우드 네이티브 보안의 4C인 Cloud, Cluster, Container, Code로 구성하였으며, 최하위 노드로는 각 세부 구성요소들에 대해 STRIDE를 통해 도출된 위협 사항들이 연결되는 Attack Tree를 작성하였다. 표 7은 STRIDE 위협 분석을 통해 식별된 위협과 작성된 Attack Tree 사이의 연관성을 보여주며, 쿠버네티스를 대상으로 한 공격방법들을 구체화하는데 의의가 있다.

Ⅳ. 보안요구사항 및 대응방안

4장에서는 식별한 쿠버네티스 위협을 기반으로 분석한 결과를 통해 표 7과 같이 보안요구사항을 도출하

Table. 7 Attack Tree

Attack Tree				Threats
1	Enter, store, and manage Pod information in the cluster			
OR	1.1	Enter and save pod information		
	OR	1.1.1	Gaining access through spoofing	T1
	OR	1.1.2	Elevate privileges after gaining access	T5, T7, T16
	OR	1.1.3	information leak	T4
	OR	1.1.4	Inserting malicious code	T2
	OR	1.1.5	Denial of Attack (Traceability of Responsibility)	T3, T6, T15
OR	1.2	Create and manage pod build information		
	OR	1.1.1	Gaining access through spoofing	T10
	OR	1.1.2	Elevate privileges after gaining access	T9, T14, T18
	OR	1.1.3	information leak	T13
	OR	1.1.4	Inserting malicious code	T11
	OR	1.1.5	attack denial	T12
	OR	1.1.6	Denial of Service	T8, T17
2	Creating Pods in Worknodes			
OR	2.1	Pod creation and monitoring		
	OR	2.1.1	Gaining access through spoofing	T19, T24, T29, T41, T45
	OR	2.1.2	elevation of privilege	T21, T23, T28, T31, T49
	OR	2.1.3	Sniffing	T27, T44, T48
	OR	2.1.4	Inserting malicious code	T25, T42, T46
	OR	2.1.5	attack denial	T26, T30, T43, T47
	OR	2.1.6	Denial of Service	T20, T22
3	Build images within Worknode			
OR	3.1	Image request and check		
	OR	3.1.1	Bypass authentication and gain access	T32, T35, T37
	OR	3.1.2	information leak	T33
	OR	3.1.3	attack denial	T36
	OR	3.1.4	Denial of Service	T34
OR	3.2	image transfer		
	OR	3.2.1	Bypass authentication and gain access	T38
	OR	3.2.2	information leak	T39
	OR	3.2.3	Denial of Service	T40

였다. 쿠버네티스 공식문서에 따른 '클라우드 네이티브 보안의 4C'인 클라우드, 클러스터, 컨테이너와 코드를 보안요구사항의 범위로 설정하였으며, 도출된 보안 요구사항에 따른 대응 방안까지 제시하였다.

보안 요구사항과 대응 방안은 STRIDE 위협 모델링을 통해 도출된 위협 분석 내용에 대해 쿠버네티스 관련 공식 문서인 SK Infosec의 '클라우드보안가이드(컨테이너 보안) - Docker, Kubernetes', Kubernetes 공식 문서인 'Cloud Native Security Overview', NSC/CISA (미국 국가안보국/사이버 보안 및 인프라 보안 기관)의 'Kubernetes Hardening Guidance'를 참고하여 세부적으로 작성하였다.

Table. 8 MS Tool threat analysis result

Category	Surface	Detail	Countermeasures
Cloud Infrastructure Security	API Server Authentication Control	A secure authentication method must be implemented to prevent access through spoofing.	User Account Management, Implementation of Secure Authentication Method (RBAC)
	API Server Permission Control	Elevation of privileges can affect or destroy the operations of other containers.	Network Isolation, Network Exposure Prevention
	Admission Control Plugin	Attacks such as network sniffing can be used for other attacks, such as inserting malicious code due to exposure of key information.	Admission Control applied, PodSecurity Policy access control. Webhook based security function
	Log management	Logging, monitoring, and warning systems should be set up for access and connection by an attacker's malicious behavior.	Log settings and management, service mesh platform configuration
	etcd access control	secure authentication method must be implemented for access to etcd (data store) containing key information and secrets.	RBAC enforcement, HTTPS enforcement, and TLS certificate implementation
	etcd encryption	Encryption and communication section encryption should be performed with a verified encryption algorithm for etcd containing key information.	Use of verified encryption algorithms (secret box, kms)
	RBAC Authentication (API)	Security settings should be made so that sensitive information is not exposed due to incorrect access control and privilege escalation.	Apply RBAC
	Cluster authentication	To prevent access through spoofing, you need to add authentication to the cluster.	ServiceAccount Authorization (Token Authorization Authorization Controller)
	Secret management	All secrets are encrypted and managed to prevent theft and abuse of sensitive data through sniffing.	RBAC enforcement, HTTPS enforcement, and TLS certificate implementation
	Controlling Container Permissions	If your application is compromised, you must ensure that the container privileges cannot be further compromised by an attacker by collecting the container privileges.	Implement PodSecurityPolicy, use security context
Cluster Security	network policy	Since there is a threat that a specific application may attack a neighboring application, it is restricted through network rules to prevent the attack.	Implement PodSecurityPolicy
	TLS for Ingress	Data encryption and encrypted communication must be configured to prevent the attacker from acquiring sensitive data.	Configuration file encryption with sensitive data, HTTPS encrypted communication (SSL/TLS)
	Kubelet Authentication Control	Security settings must be made to prevent unauthorized access by attackers.	Authentication setting in kublet service file or Config file, HTTPS encrypted communication (SSL/TLS)
	Kubelet Permission Control	Kubelet allows all requests without permission checks, so you need to do permission validation.	RBAC implementation for kubelet
	Kubelet parameter settings	If the default value of the parameter is different from the security policy, you must prevent Pods with unwanted kernel features from running.	Security function applied to parameter tuning

Category	Surface	Detail	Countermeasures
Container Security	Vulnerability scanning and OS security	If you do not take measures against known vulnerabilities in containers, security devices may be bypassed or the container and the host OS may be attacked.	Checking containers for known vulnerabilities
	Image Signing and Enforcement	Image management is very important, and security issues can occur due to image vulnerabilities, configuration flaws, and malicious code injection.	Implement image inspection and acceptance controllers
	Disallowance of privileged users	When configuring the container, it should be configured so that no user has more privileges than necessary.	Create a user with the least necessary privileges in the container
	Controlling Container Permissions	For a specific container, the least privilege principle or the least privilege principle should be set to allow access to only those that are absolutely necessary.	Implement PodSecurityPolicy, configure container runtime class (optional)
Code Security	Access via TLS	In the same way as network sniffing, code information can be exposed and used for other attacks.	Implement Mutual TLS Authentication (mTLS)
	Port range limits	Only necessary ports should be allowed to prevent intrusion by an attacker using unnecessary service ports.	Whenever possible, expose only ports for services that are essential for communication or metric collection. (recommend)
	Third-Party Dependency Security	An attacker can use a supply chain of weak dependencies, so it must be checked periodically to prevent malicious behavior.	Regularly scan your application's third-party libraries for any currently known vulnerabilities.

Ⅴ. 결 론

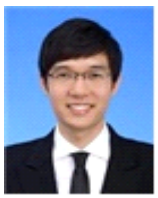
본 논문에서는 STRIDE 위협 모델링의 위장(Spooping Identity, 인증 관련), 무결성(T : 변조(Tampering with data)), 부인 방지(R : 부인(Repudiation)), 기밀성(I : 정보 유출(Information Disclosure)), 가용성(D : 서비스 거부(Denial of Service)), 인가(E : 권한상승(Elevation of Privilege))을 기반으로 웹 서비스를 쿠버네티스 기반 배포 시 참고할 수 있는 새로운 분류의 보안 가이드를 제시하였다. 기존의 보안 가이드 및 논문 자료를 분석한 결과 SK Infosec의 클라우드 보안 안내서는 Infrastructure 관련 내용을 상세히 다뤘고, 쿠버네티스 공식 문서에서는 주로 Cluster, Container, Code 보안 관련 내용을 상세히 다루 졌으며, NCS/CISA의 쿠버네티스 강화 지침에서는 포드 보안, 네트워크 분리 및 강화, 인증 및 권한 부여, 로그 감사, 업그레이드 및 애플리케이션 보안 관련 내용을 다루고 있음을 알 수 있었으며, 기존 선행 논문[7]에서는 보안 가이드들을 보안 3대 요소인 CIA 측면만 고려하여 논문이 작성 되었음을 알 수 있었다. 기존 논문을 기반으로 각각 쿠버네티스 취약점별로 재구분하고, 그것을 다시 STRIDE 위협모델링으로 분류를 진행하였으며, 이에 따른 Attack Tree를 구성하여 보안 위협의 대응방안에 대한 체크리스트를 작성까지 진행하였다. 쿠버네티스에 대한 다양한 가이드별로 명확한 기준 없이 정보가 나열되어 있어 사용자 입장에서 보안 방법을 파악하고 위협 모델링을 통해 사전에 소프트웨어의 설계 및 개발 단계에서 보안 요소를 고려할 수 있도록 한 것이 본 논문의 기여점이다.

쿠버네티스 기반의 클라우드 컴퓨팅을 STRIDE 위협 모델링에 적용하여 새로이 보안 요구사항을 도출하였지만, 향후 개인정보와 관련하여 LINDDUN 위협 모델링 [28]으로 논문을 작성하고자 한다. LINDDUN 위협 모델링은 정확성, 완전성, 생산성을 이용하여 평가함으로서 개인정보의 위협에 대해 자세히 분석할 수 있으며, 개인정보 보호(연결성, 식별성, 부인 방지, 탐지성, 정보 유출, 내용 인지, 부적절한 정책과 동의)에 초점을 맞추어 논문을 작성할 수 있다. 이를 통해 다양한 관점에서의 쿠버네티스에 대한 위협 분석을 통해 사전에 보안 요구사항을 도출하여 소프트웨어의 설계 및 개발 단계에서부터 보안 위협을 제거하고자 한다.

REFERENCES

[1] Sysdig, Sysdig 2019 Container Usage Report: New Kubernetes and security insights [Internet]. Available: https://sysdig.com/blog/sysdig-2019-container-usage-report/.

[2] M. Aoyama, *Kubernetes Perfect Guide*, 1st ed. Seoul, Korea, Ltd. Gilbot, 2021.
[3] Cloudstore Ceart, "Main Cloud Computing Trends in 2020," Ceart Issue Report, vol. 2, pp. 12, Jan. 2020.
[4] Kubernetes [Internet]. Available: https://en.wikipedia.org/wiki/Kubernetes.
[5] K. Kim, G. Lee, T. Kim, J. Choi, S. Ha, Y. Jeong, and S. Jin, "Kubernetes Architecture for Cloud Services," *Information & Communications Magazine*, vol. 35, no 11, pp. 11-19, Oct. 2018.
[6] Kubernetes, Cloud native security overview [Internet]. Available:https://Kubernetes.io/ko/docs/concepts/security/overview/.
[7] H. Kang, S. Park, H. Yoon, and E. Lee, "A Study on Web Server Security Policy in Docker Kubernetes Environment," in *Korea Society of IT Services 2020 Fall Conference*, Seoul, Korea, pp, 632-637, 2020.
[8] M. Panagiotis, "Attack methods and defenses on Kubernetes," Bachelor's dissertation, University of Piraeu, Pireas, Greece, 2020.
[9] N. Habbal, "Enhancing Availability of Microservice Architecture," A Case Study on Kubernetess Security Configurations, Bachelor's dissertation, Luleå University of Technology, Luleå, Seden, 2020.
[10] T. Autio, "Securing a Kubernetes Cluster on Google Cloud Platform," Bachelor's dissertation, Metropolia University of Applied Sciences, Vantaa, Finland, 2021.
[11] T. Fowley, "Security of Virtual Infrastructures : Assessing Kubernetes Attack Automation," M. S. dissertation, Trinity College Dublin, Dublin, Ireland, 2021.
[12] H. Kim, "Cloud Security Guide (Container Security) - Docker, Kubernetes", SK infosec, Technical Report, Korea, Jun. 2019.
[13] Kubernetes, The 4Cs of Cloud Native Security [Internet]. Available: https://Kubernetes.io/ko/docs/concepts/security/overview/.
[14] NSC/CISA, "Kubernetes Hardening Guidance," NSC/CISA, Washington D.C, USA, Technical Report PP-21-1104, Ver 1.0, Aug. 2021.
[15] The MITRE Corp., CVE-2022-27211 [Internet]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-27211.
[16] The MITRE Corp., CVE-2022-27210 [Internet]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-27210.
[17] The MITRE Corp., CVE-2022-27209 [Internet]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-27209.
[18] The MITRE Corp., CVE-2022-27208 [Internet]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-27208.
[19] The MITRE Corp., CVE-2022-26311 [Internet]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-26311.
[20] The MITRE Corp., CVE-2022-24768 [Internet]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-24768.
[21] The MITRE Corp., CVE-2022-24731, CVE-2022-24730 [Internet]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-24731,https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-24730.
[22] The MITRE Corp., CVE-2022-23652 [Internet]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-23652.
[23] The MITRE Corp., CVE-2022-23648 [Internet]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-23648.
[24] The MITRE Corp., CVE-2022-21701 [Internet]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-21701.
[25] The MITRE Corp., CVE-2022-0811 [Internet]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-0811.
[26] The MITRE Corp., CVE-2022-0270 [Internet]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-0270.
[27] WIKIPEDIA. Attack Tree [Internet]. Available: https://en.wikipedia.org/wiki/Attack_tree.
[28] LINDDUN, Privacy threat modeling(LINDDUN) [Internet]. Available: https://www.linddun.org/.



이승욱(Seungwook Lee)

2006년 2월 육군3사관학교 전자공학과 학사
2019년 3월 ~ 현재 중앙대학교 융합보안학과 정보보안 전공 석사과정
※관심분야 : 클라우드, 쿠버네티스, 위협 모델링

이재우(Jaewoo Lee)

2006년 2월 서울대학교 컴퓨터공학부 학사
2008년 2월 서울대학교 컴퓨터공학부 석사
2017년 8월 University of Pennsylvania, Ph.D in Computer and Information Science
2018년 3월~현재 중앙대학교 산업보안학과 조교수
※관심분야 : Cyber Physical System Security