https://doi.org/10.33778/kcsa.2023.23.5.081

# 쿠버네티스 환경에서의 강화학습 기반 자원 고갈 탐지 및 대응 기술에 관한 연구★

김 리 영*, 김 성 민**


## 요 약

쿠버네티스는 컨테이너 통합 관리를 위한 대표적인 오픈소스 기반 소프트웨어로, 컨테이너에 할당된 자원을 모니터링하고 관리하는 핵심적인 역할을 한다. 컨테이너 환경이 보편화됨에 따라 컨테이너를 대상으로 한 보안 위협이 지속적으로 증가하고 있으며, 대표적인 공격으로는 자원 고갈 공격이 있다. 이는 악성 크립토마이닝 소프트웨어를 컨테이너 형태로 배포하여 자원을 탈취함으로써, 자원을 공유하는 호스트 및 다른 컨테이너의 동작에 영향을 끼친다. 선행 연구는 자원 고갈 공격의 탐지에 초점이 맞춰져 있어 공격 발생 시 대응하는 기술은 부족한 실정이다. 본 논문은 쿠버네티스 환경에서 구동되는 컨테이너를 대상으로 한 자원 고갈 공격 및 악성 컨테이너를 탐지하고 대응하기 위한 강화학습 기반 동적 자원 관리 프레임워크를 제안한다. 이를 위해, 자원 고갈 공격 대응 관점에서의 강화학습 적용을 위한 환경의 상태, 행동, 보상을 정의하였다. 제안한 방법론을 통해, 컨테이너 환경에서의 자원 고갈 공격에 강인한 환경을 구축하는 데 기여할 것으로 기대한다.


# Reinforcement Learning-Based Resource exhaustion attack detection and response in Kubernetes

Ri-Yeong Kim*, Seongmin Kim**


## ABSTRACT

Kubernetes is a representative open-source software for container orchestration, playing a crucial role in monitoring a nd managing resources allocated to containers. As container environments become prevalent, security threats targeting con tainers continue to rise, with resource exhaustion attacks being a prominent example. These attacks involve distributing m alicious crypto-mining software in containerized form to hijack computing resources, thereby affecting the operation of the host and other containers that share resources. Previous research has focused on detecting resource depletion attacks, so t echnology to respond when attacks occur is lacking. This paper proposes a reinforcement learning-based dynamic resource management framework for detecting and responding to resource exhaustion attacks and malicious containers running in Kubernetes environments. To achieve this, we define the environment's state, actions, and rewards from the perspective of responding to resource exhaustion attacks using reinforcement learning. It is expected that the proposed methodology will contribute to establishing a robust defense against resource exhaustion attacks in container environments


**Key words : kubernetes, reinforcement learning, autoscaling, resource exhaustion attack**

접수일(2023년 09월 30일), 게재확정일(2023년 10월 16일)

★ 본 논문은 2023년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원(NRF-2021R1G1A1006326l1), 산업통상자원부의 재원으로 한국산업기술진흥원의 지원(P0008703, 2022년 산업혁신인재성장지원사업) , 과학기술정보통신부 및 정보통신기획평가원의 ICT혁신인재4.0 사업(IITP-2022-RS-2022-00156310)의 연구결과로 수행되었음.

* 성신여자대학교 미래융합기술공학과 (주저자)

** 성신여자대학교 융합보안공학과 교수 (교신저자)

# 1. 서 론

도커(Docker)로 대표되는 컨테이너 기술은 뛰어난 이식성과 확장성으로 인해 클라우드 환경에서 기존 가상화 시스템을 대체하는 주요 기술로 주목받고 있다. 사용자들은 다수의 컨테이너를 효율적으로 관리하기 위해 컨테이너 통합 관리 도구를 사용하는데, 대표적으로는 쿠버네티스(Kubernetes)가 있다. 이는 컨테이너화된 애플리케이션의 배포, 확장 및 관리에 대한 자동화를 가능케 한다. 특히, 자원 할당과 모니터링, 스케일링과 같은 일반 사용자가 직접 운영 및 설정하기 어려운 기능을 자동화해줌으로 인해, 국내외 다양한 서비스에서 클라우드 네이티브 어플리케이션 호스팅을 위한 핵심적인 소프트웨어로 활용되고 있다[1].

쿠버네티스는 자원 할당을 위해 HPA(Horizontal Pod Auto-scaler)를 통해 사용자가 정의한 임계값을 기반으로 오토스케일링(auto-scaling)을 수행한다[2]. 하지만 현재의 고정된 임계값 기반 오토스케일링은 워크로드 특성에 따라 자원 사용량이 동적으로 다이나믹하게 변하는 환경에 효과적으로 대응하기 어렵다[5, 6, 7, 8, 9, 10]. 이로 인한 비효율성은 자원 낭비, 가용성 침해, SLA(Service Level Agreement) 위반 등의 문제를 야기한다. 이를 해결하기 위해 선행 연구에서는 컨테이너 기반 애플리케이션에 환경 적응적인 강화학습 기반 오토스케일링 정책을 제안하였다[5, 6, 9, 10]. 에이전트가 환경과 상호작용하며 보상을 통해 최적의 행동을 학습하는 강화학습의 적용을 통해, 제안한 방법론은 변동하는 자원 요청과 서비스의 QoS(Quality of Service) 등을 동적으로 학습 후 인스턴스 수 및 내부 자원 할당을 동적으로 조절하여 자원 사용을 최적화하였다.

한편, 컨테이너의 사용이 증가함에 따라 컨테이너 환경을 대상으로 한 보안 위협이 증가하고 있다. 클라우드 인프라가 제공하는 대규모 컴퓨팅 자원을 오용하기 위한 목적으로 인스턴스를 감염시켜 암호화폐 채굴 및 악성 크립토마이닝 소프트웨어 배포에 활용하는 사례가 대표적이다. Google Cloud의 조사에 따르면 컨테이너 환경에 대한 공격 중 86%가 크립토마이닝 악성 컨테이너에 의해 발생했다[3]. 자원 탈취를 목적으로 하는 크립토마이닝 악성 컨테이너는 컨테이너의 권한을 넘어 호스트 자원 탈취까지 이어질 수 있다. 컨테이너 환경은 컨테이너 간 호스트 커널을 공유하기 때문에, 완전한 격리를 보장하는 전통적인 가상화 환경과 달리 컨테이너 안에서 발생하는 공격이 다른 컨테이너나 호스트에 영향을 미칠 수 있다. 선행 연구에서는 기본 자원 설정을 사용한 멀티 테넌시 클라우드 환경에서 악성 컨테이너가 지나치게 많은 CPU와 메모리 자원을 사용하여 다른 컨테이너의 성능이 감소함을 보였다[12]. 따라서, 악성 컨테이너 유입이나 컨테이너 자원에 대한 제한 설정 미비 등으로 인해 발생하는 자원 고갈 공격 탐지 및 대응 기술에 관한 연구가 시급하다.

본 연구에서는 쿠버네티스 환경에서 구동되는 컨테이너를 대상으로 한 자원 고갈 공격 및 악성 컨테이너를 탐지하고 대응하기 위한 강화학습 기반 동적 자원 관리 프레임워크를 제안한다. 자원 효율성 향상을 위한 강화학습의 적용에 초점을 맞춘 선행 연구와 달리, 본 연구에서는 자원 고갈 공격에 대응하기 위한 강화학습의 활용 방안을 제안한다. 쿠버네티스 계층에서 수집 가능한 자원 고갈 공격 관련 메트릭들을 정의하고, 수집된 정보를 바탕으로 한 기계 학습 기반 악성 행위 탐지 모듈을 제안한다. 또한, 탐지 모듈의 결과를 바탕으로 자원 고갈 공격 대응을 위한 강화학습 서브 모듈을 설계하였다. 이때, 강화학습 적용을 위한 상태, 행동 및 보상을 기존 쿠버네티스 시스템에 연동할 수 있도록 정의하였다.

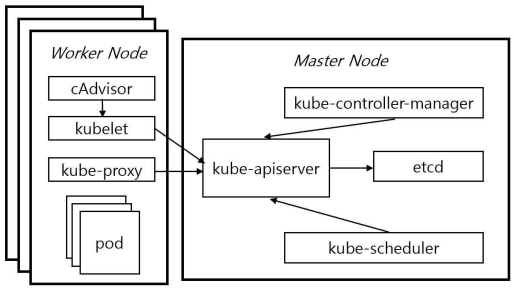
# 2. 배경 지식 및 관련 연구

## 2.1 쿠버네티스

마이크로서비스는 클라우드 컴퓨팅을 활용하여 기업들이 빠르게 애플리케이션을 개발하고 배포할 수 있도록 하며, 이로 인해 더 큰 유연성과 확장성 확보를 가능케 한다. 마이크로서비스 아키텍처는 하나의 애플리케이션을 독립적으로 배포가 가능한 여러 개의 작은 서비스로 구성된다. 컨테이

너는 운영체제 수준의 가상화 기술로, 기존의 하이퍼바이저 기반의 가상화 기술과 비교하여 가볍고 쉽게 확장 가능하므로 마이크로서비스 아키텍처 구현에 적합한 실행 환경을 제공한다. 많은 IT 기업들이 컨테이너를 채택함에 따라 CNCF 조사에 따르면 2016년부터 2020년까지 컨테이너의 사용이 300% 증가했다[4].

이에 따라 대규모의 컨테이너를 효율적으로 배치하고 관리하기 위해 컨테이너 통합 관리 도구인 쿠버네티스가 개발되었다. 쿠버네티스는 애플리케이션을 배포하고, 관리하기 위해 pod라는 기본 단위를 사용한다. pod는 하나 또는 다수의 컨테이너로 구성되며, 해당 pod 내의 컨테이너들은 모든 자원을 공유한다. 쿠버네티스 클러스터는 마스터 노드와 워커 노드로 구성된다. 마스터 노드는 kube-apiserver, etcd, kube-controller-manager, kube-scheduler를 통해 노드의 상태를 감지하고, 전체 클러스터를 관리한다. 워커 노드는 마스터 노드의 명령에 따라 실제 pod를 생성하며, kublet, kube-proxy, 컨테이너 런타임, cAvisor로 구성된다.




(Figure 1) Kubernetes Architecture

쿠버네티스는 효율적인 자원 관리, 최적의 컨테이너 배치, 오토스케일링 등 컨테이너 오케스트레이션을 수행하기 위해 모니터링을 통해 관련 메트릭을 수집한다. Figure 1은 쿠버네티스 아키텍처 구성도 및 동작 흐름을 나타낸다. cAdvisor은 각 노드에서 컨테이너의 CPU 및 메모리 사용량과 같은 정보를 수집하여 kubelet으로 전달한다. metrics-server는 각 노드의 kubelet으로 수집된 메트릭을 주기적으로 스크랩하여 통합하고, 마스터 노드의 kube-apiserver로 전송한다. metrics-server를 통해 수집되는 메트릭은 각 pod와 노드의 CPU 및 메모리 사용량으로 이를 리소스 메트릭이라 한다. 이러한 리소스 메트릭은 오토스케일링를 수행할 때 사용된다. 또한, 쿠버네티스에서는 HPA를 디폴트 오토스케일러로 사용한다. HPA는 pod의 리소스 메트릭을 수집하여 임계값을 기반으로 pod 수를 수평적으로 확장시킨다. HPA의 임계값은 애플리케이션의 워크로드 특성을 고려하여 적절하게 설정하는 것이 중요하다.

## 2.2 클라우드에서의 강화학습 활용

강화학습은 기계 학습 분야의 한 기법으로, 주어진 환경에서 반복적인 시행착오를 통해 바람직한 행동 패턴을 학습한다. 강화학습 모델은 에이전트, 환경의 상태(state), 행동(action) 그리고 보상(reward)으로 정의된다. 에이전트는 원하는 목표 달성을 위해 현재 상태에서 특정 행동을 선택하고, 행동이 목표 달성에 미치는 영향에 따라 보상받는다. 에이전트는 보상을 최대화하기 위해 학습하며, 훈련된 강화학습 에이전트는 불확실한 환경에서 자동 의사결정을 통해 최적의 행동을 스스로 습득할 수 있다.

동적인 클라우드 환경의 변화에 적응적인 오토스케일링 정책을 최적화하기 위해 강화학습을 활용하는 연구가 활발히 진행되고 있다. Table 1은 관련 선행 연구를 특성에 따라 분류하여 나타낸 것이다. 선행 연구에서는 응답 시간, 자원 효율성 및 처리율(throughput)과 같은 메트릭을 최적화하기 위해 Q-learning, SARSA, Dyna-Q 등의 강화학습 알고리즘을 적용하였다. 제안 방법론들은 가상화의 세분성 (granularity) 측면에서 가상 머신 레벨 또는 컨테이너 레벨로 구분되며, 각 레이어 수준에서의 수평적 및 수직적 스케일링을 통해 보상 지표를 최적화하기 위해 강화학습을 적용하였다.

Khaleq 외 1인은 강화학습을 활용하여 자원 요구 사항 및 QoS에 따라 인스턴스 수를 동적으로 조정하고[5], Horovitz 외 1인은 에이전트가 HPA를 위해 사용되는 임계값을 동적으로 학습하여 애플리케이션의 응답 시간을 향상시켰다[6]. 또한 [7], [8]의 연구는 보상을 응답 시간, 처리율 및 S

<Table 1> Related works on RL based Cloud Auto-scaling

| Related work | State | Action | Reward metrics | Virtualization | RL Algorithnm |
|---|---|---|---|---|---|
| Khaleq et al [5] | Number of pods (min, max, current), Resource utilization, Response time | Horizontal | Response time | Microservices, Container | Q-learning, SARSA, DQN etc. |
| Horovitz et al [6] | Number of resources allocated to the application (VMs, Containers) | Threshold adjustment | Response time, CPU Utilization | VMs, Container | Q-learning |
| Veni et al [7] | Number of CPUs, CPU time, Memory size | Vertical | Response time, Resource Utilization, Throughput, SLA violation | VMs | Q-Learning + Neuro-Fuzzy function approximation |
| Ebbal Benifa et al [8] | Number of user requests, VM utilization, Response time, Throughput | Horizontal | Response time, Resource Utilization, Throughput, SLA violation | VMs | SARSA |
| Rossi etl al [9] | Number of containers, CPU utilization, CPU share | Horizontal + Vertical | Availability, Response time, Resource Utilization | Container | Q-learning, Dyna-Q, Model-based |
| Zhang et al [10] | Concurrency limit, CPU utilization, Memory utilization | Concurrency limit | Throughput | Container | Q-learning |

*Horizontal : scale-out/in , Vertical : scale-up/down

LA 위반과 같은 성능 지표와 자원 활용 간의 관계에 기반하여 자원을 동적으로 구성함으로써 최소한의 자원으로 최대 시스템 성능을 극대화하는 방법을 제안했다. Rossi 외 2인은 애플리케이션의 서비스 배포 목적에 따라 HPA 및 VPA (Vertical Pod Auto-scaler) 탄력성을 효율적으로 제어하기 위해 강화학습을 적용했다[9]. Zhang 외 3인은 강화학습 기반의 모델을 사용하여 다양한 워크로드에 대해 런타임 동안 최적의 동시성 수준을 최적화하는 방법을 제안했다[10]. Knative 환경에서 Q-learning을 활용하여 최적의 동시성 수준이 처리량 측면에서 우수한 성능을 보여줄 수 있음을 입증하였다.

선행 연구에서는 컨테이너 환경에서 실행되는 워크로드의 동적인 특성에 적응적인 강화학습 기반 오토스케일링 최적화를 통해 효율적인 자원 관리를 달성하고자 하였다. 본 연구에서는 이러한 접근과 달리, 악성 컨테이너 및 자원 고갈 공격 대응의 관점에서 강화학습을 적용하고자 한다는 측면에서 차별성을 가진다. 본 연구를 통해 제안한 프레임워크는 자원 효율성이 아닌 보안 관점에서 악의적인 컨테이너의 동작을 강화학습을 통해 제어함으로써, 자원을 공유하는 컨테이너 및 호스트를 보호하는 것을 목표로 한다.

### 2.3 컨테이너 환경에서의 자원 고갈 공격

시스템 자원 고갈 공격은 CPU 및 메모리 등의 자원을 고갈시켜 정상적인 서비스 운영을 방해하는 공격 형태이다. 현대의 멀티 테넌시(Multi-tenancy) 클라우드 환경은 클라우드 서버 리소스를 여러 사용자가 나누어 사용하며, 사용자는 논리적으로 격리된 가상화 환경을 할당받아 자신의 서비스

와 애플리케이션을 실행시킨다. 구체적으로, 컨테이너는 namespace, cgroup과 같은 리눅스 커널의 기능을 사용하여 독립적인 실행 환경을 제공한다. 하지만 이는 논리적으로 격리된 환경이기 때문에 다른 컨테이너의 프로세스에 영향을 끼칠 수 있다[11]. 즉, 컨테이너 환경에서 발생하는 자원 고갈 공격은 이러한 호스트 OS 커널을 공유하는 구조적 취약점으로 인해 다른 컨테이너와 더불어 호스트 서버의 다양한 리소스에 대한 가용성을 침해하고 성능에 영향을 끼치는 일련의 행위를 포괄한다. 악성 크립토마이닝 소프트웨어를 컨테이너에서 구동시키거나 컨테이너에서 구동되는 애플리케이션에 악의적인 요청을 보내 호스트의 시스템 자원을 고갈시키는 공격이 대표적인 예시이다.

컨테이너 환경에서의 자원 고갈 공격을 탐지하고 대응하기 위한 선행 연구들이 존재한다. 이준희 외 2인은 기본 자원 설정을 사용한 멀티 테넌트 클라우드 환경에서 악성 컨테이너가 과도한 CPU 및 메모리 자원을 소비하여 다른 컨테이너의 성능이 저하됨을 보였다[12]. 공격자는 특정 애플리케이션의 취약점을 악용한 악성 페이로드를 사용하여 시스템 자원을 과도하게 소모하고, 이로 인해 정상 사용자 및 다른 애플리케이션의 동작을 간섭한다.

공격자는 암호화폐 채굴과 같이 CPU 집약적인 악성 컨테이너 이미지를 배포하여 서버 자원을 탈취하고, 정상 컨테이너가 자원을 활용하는 것을 방해할 수 있다. 크립토하이재킹은 인스턴스 소유자의 승인 없이 자원을 남용하여 가상 화폐를 채굴하는 공격으로, CPU를 집약적으로 사용하므로 CPU 사용량을 기반으로 공격을 탐지하는 연구가 존재한다[13, 14, 15]. 그러나 CPU를 집약적으로 사용하는 정상 컨테이너에 CPU 사용량의 변화가 크게 나타날 경우에는 잘못된 탐지가 발생할 가능성이 있으며, 악성 컨테이너가 CPU 사용량을 조작하여 탐지를 회피할 수 있다. 컨테이너와 호스트 커널 간의 시스템 호출을 모니터링하여 컨테이너의 동작을 추적하고, 이를 통해 컨테이너의 악성 활동을 감지할 수 있다. 크립토마이닝은 암호화폐 기반의 작업 증명 알고리즘을 반복적으로 실행하므로 이러한 반복된 패턴은 특정 시스템 호출의 패턴을 생성한다. Karn 외 4인은 컨테이너의 시스템 호출 패턴과 CPU 사용량을 기반으로 악성 컨테이너를 탐지하는 방법을 제안한다[16].

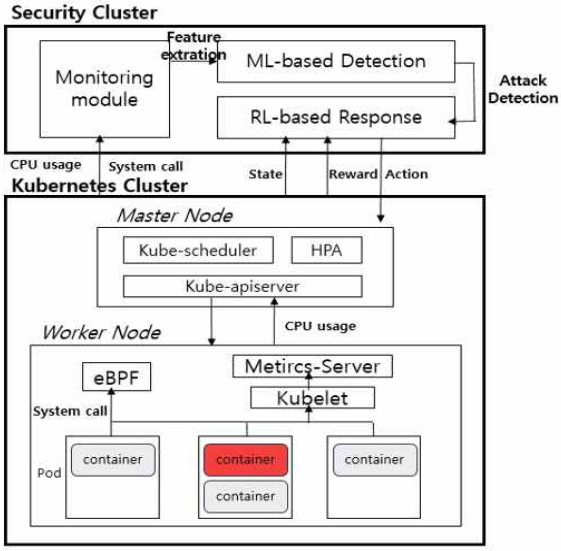
Coda[17]는 컨테이너 내의 애플리케이션 계층에서 발생하는 CPU 자원 고갈 공격을 감지하기 위해 eBPF(extended Berkeley Packet Filter)를 통해 컨테이너의 시스템 호출을 분석하고 각 연결이 소비하는 CPU 시간을 추적하였다. 연결이 소비하는 CPU 시간을 정상적인 연결과 비교하여 통계적으로 공격을 식별한다. 또한 Baarzi 외 3인은 컨테이너의 CPU 및 메모리 사용량을 예측하여 공격을 탐지한다[18]. 이러한 공격에 대응하기 위해 공격자를 격리 노드에 격리시켜 정상 사용자의 자원 고갈을 완화한다. Castro 외 2인은 컨테이너의 자원 사용량, 소켓 수, 스레드 수 등 메트릭을 기반으로 기계 학습을 사용하여 DoS 공격을 탐지한다[19].

컨테이너 환경에서의 자원 고갈 공격에 관한 기존 선행 연구는 공격의 탐지에 초점이 맞추어져 있다. 현재 대응 관점에서 어떻게 악성 컨테이너의 동작을 제어할 것인지에 대한 연구는 부족한 실정이며, 악성으로 판단되는 컨테이너의 동작을 즉각적으로 종료시키는 형태의 대응은 오탐이 발생했을 때 취약하다는 한계점을 가진다. 따라서, 본 연구에서는 강화학습의 활용을 통해 점진적으로 자원 고갈 공격을 야기시키는 악성 컨테이너의 동작을 조절(throttle)하는 대응 방안을 제안한다.

## 3. 프레임워크 디자인 설계 및 구성

본 장에서는 제안하는 쿠버네티스 환경에서의 악성 컨테이너 탐지 및 대응 프레임워크의 구성 요소 및 동작 흐름에 대해 기술한다. 먼저, 공격 여부 탐지를 위한 데이터 확보를 위해 모니터링 모듈에서 관련 메트릭을 수집한다. 이후 수집된 정보를 전처리 및 특징 추출을 통해 재가공 후 기계학습 모델을 학습시킨다. 학습을 통해 획득한 모델을 기반으로 구동 중인 컨테이너의 데이터를 모니터링 모듈이 수집한 뒤 추론 단계를 통해 자원 고갈 공격 발생 여부를 탐지한다. 마지막으로 강화학습 기반의 동적 자원 관리를 통해 모델이 악

성으로 판단한 컨테이너에 한정하여 사용 가능한 자원을 점진적으로 제한함으로써 정상 컨테이너에게 미치는 영향을 최소화한다. Figure 2는 전체 프레임워크 구성도 및 동작 흐름을 나타낸다.




(Figure 2) Architecture of the proposed framework

### 3.1 컨테이너 환경에서의 자원 고갈 공격

모니터링 모듈은 주기적으로 클러스터 내 컨테이너의 CPU 사용량과 시스템 호출 시퀀스를 수집한다. CPU 사용률은 kubelet 내에 내장된 컨테이너 모니터링 도구인 cAdvisor에서 수집되는 메트릭을 기반으로 한다. 시스템 호출을 모니터링하기 위해 워커 노드 내에서 eBPF(extended Barkeley Packet Filter)가 실행된다. eBPF[20]는 리눅스 커널에서 동작하여 사용자가 정의한 코드를 커널 내에서 실행시킬 수 있는 도구로, 이를 통해 서버 전체의 시스템 호출 추적이 가능하다. kprobe를 활용하여 시스템 호출이 커널 내에서 실행되기 직전에 발생하는 sys_enter 이벤트에 eBPF 프로그램을 연결함으로써 해당 이벤트가 발생할 때 시스템 호출을 추적한다. 이를 통해 eBPF는 시스템 호출이 커널 내에서 실행되기 직전에 실행되며, 모니터링 모듈이 시스템 호출의 시퀀스 정보를 수집할 수 있다. 모니터링 모듈에서 수집한 메트릭은 이후 공격 탐지에 활용되는 기계 학습 모델에 적합한 데이터 세트로 변환되어 탐지 모델 학습에 활용된다.

### 3.2 기계 학습 기반 공격 탐지

모델 학습을 위한 라벨링 된 데이터 세트를 생성하기 위해 실제 컨테이너 배치 및 구동 전 정상 컨테이너와 악성 크립토마이닝 컨테이너를 실행하여 데이터를 수집한다. 수집된 데이터는 CPU 사용량과 시스템 호출 시퀀스 번호를 시계열 형태로 포함하며, 이러한 원시 데이터를 모델 학습을 위해 전처리한다. CPU 사용량은 일정한 시간 간격으로 샘플링하고, 해당 시간 간격 내의 CPU 사용량 평균 및 최대 사용량을 계산한다. 시스템 호출에 대한 특성은 시스템 호출의 빈도수와 n-gram을 활용한 연속적인 시퀀스 특징을 고려한다. 여기서 n-gram은 연속된 n개의 시스템 호출을 하나의 패턴으로 간주하여 모델 학습에 시스템 호출 시퀀스의 구조적 정보를 제공한다. 이때, 서로 다른 n값에 대한 모델 성능 평가를 통해 적절한 윈도우 크기를 선택해야 한다.

선행 연구에서는 크립토마이닝 악성 소프트웨어를 탐지하기 위해 다양한 기계 학습 기반 모델을 활용했다. 탐지 모듈의 경우, 규칙 및 통계적 접근 방식을 기반으로 Decision Tree[16], XgBoost[16], Random Forest[21, 22], SVM[23]과 같은 기계 학습 모델을 활용하거나, LSTM, ATT-LSTM, CNN과 같은 딥러닝 모델을 활용하는 방법이 존재한다. 이중, [24]의 연구는 LSTM, ATT-LSTM, CNN과 같은 딥러닝 모델을 활용하여 시스템 호출의 시퀀스적 특성을 고려한 악성 크립토마이닝 컨테이너 탐지 모델을 제안했다. 이와 같은 학습 모델을 바탕으로, 본 연구에서 제안한 컨테이너 모니터링 모듈을 통해 수집한 시스템 호출의 시퀀스적 특성과 cAdvisor로부터 수집된 CPU 사용률 정보를 결합시켜 강화학습 기반 공격 대응을 위한 사전적인 학습 탐지 모듈을 구축한다. 이때, 오탐을 최소화하기 위해 적용 가능한 모델들의 성능을 평가하여 적절한 모델 선정 및 레이어 구성, 하이퍼파라미터 설정을 통한 최적화 과정이 필요하며, 이는 후속 연구로 진행할 예정이다.

### 3.3 강화학습 기반 공격 대응

공격 대응의 주요 목표는 공격이 탐지된 컨테이너에 대해 강화학습 기반의 동적인 자원 관리를 통해 정상 컨테이너의 자원 고갈 및 성능 저하에 대한 영향을 최소화하는 것이다. Table 2는 강화학습 기반 자원 고갈 공격 대응을 위한 상태, 행동, 보상에 대한 정의를 보여준다.

<Table 2> Metrics for RL environment configuration

| State | State - Action | | Reward |
|---|---|---|---|
| numInstances, maxInstances, CPU Usage, Attack Detection | Attack Detection = 0 | HPA | Resource usage, Number of Active Instances |
| | Attack Detection = 1 | Resource Limit | |

Table 3은 제안 프레임워크에서 상태 공간의 구조를 나타낸다. 시간 $i$에서 상태는 numInstances, maxInstances, CPU Usage, Attack Detection로 정의된다. numInstances는 현재 배포된 인스턴스 수를 나타낸다. 인스턴스 수는 1부터 maxInstances까지의 범위를 가진다. maxInstances는 설정된 최대 인스턴스 수를 나타낸다. CPU Usage는 인스턴스의 CPU 평균 사용량을 나타낸다. 자원 사용량은 실수 값일 수 있기 때문에 이산값으로 변환된다. Attack Detection는 공격 탐지 에이전트의 결과값으로 공격이 감지되지 않은 경우 0으로 표시되며, 공격이 감지된 경우 1로 표시되어 이진 형태로 표현된다.

각 상태에서 실행 가능한 에이전트의 행동은 HPA(-1, 0, +1)와 Resource Limit으로 총 4가지로 정의된다. HPA는 현재 상태에 따라 인스턴스 수를 동적으로 조절하며, Resource Limit는 해당 인스턴스에 대한 자원 제한을 설정한다. 수행 가능한 에이전트의 행동은 Attack Detection의 상태에 따라 구분된다. Attack Detection이 0인 경우에는 선택 가능한 행동 집합은 HPA로, HPA를 통해 현재 상태에 적절한 오토스케일링 정책을 학습할 수 있다. 반면에 Attack Detection이 1인 경우에는 선택 가능한 행동이 Resource Limit로, 공격이 탐지된 악성 컨테이너의 자원 소비를 제한한다. 즉 Attack Detection의 상태에 따라 행동 집합을 구분함으로써 공격이 감지되지 않았을 때 강화학습 에이전트는 환경에 적응적인 오토스케일링 정책을 적용하고, 공격이 탐지되었을 때 악성 컨테이너에 대한 오토스케일링을 중단한 후 자원에 대한 엄격한 제한을 수행할 수 있다. 따라서 악성 컨테이너에 자원 제한을 설정함으로써 지정된 자원 이상을 사용하지 못하도록 격리하여 다른 컨테이너나 호스트 자원에 영향을 미치는 것을 방지할 수 있다.

마지막으로, 에이전트의 행동에 따라 자원 사용량 및 활성화된 인스턴스 수를 고려하여 보상 함수를 정의한다. 실행 중인 인스턴스 수와 자원 사용량에 비례한 패널티를 부과하여 비용을 최소화하는 것을 목적으로 한다. 또한 최대 인스턴스 수를 초과하는 경우에 추가적인 패널티를 부과한다.

## 4. 결 론

본 논문은 자원 사용량 모니터링이 필요한 자원 고갈 공격 대응이라는 시나리오에 대해 강화학습 기반의 자원 관리를 적용하기 위한 범용성을 갖춘 프레임워크를 제시하였다. 자원 고갈 공격이 의심되는 컨

<Table 3> Metrics composing the RL State Space

| Metic | Description | Unit of Measurement | Type |
|---|---|---|---|
| numInstances | Number of deployed instances (containers/pods) | Number | Instance Status |
| CPU Usage | Average CPU usage for deployed instances (containers/pods) | Percentage | Resource usage |
| Attack Detection | Detection status based on the attack detection module | Number | Threat Status |

테이너를 즉각적으로 종료하지 않고 강화학습을 통해 점진적으로 자원 사용 측면에서의 영향을 약화시킴으로써, 오탐으로 인한 컨테이너의 비정상적인 종료를 피할 수 있다. 이를 위해, 자원 고갈 공격 대응 관점에서의 강화학습 적용을 위한 환경의 상태, 행동, 보상을 정의하였다.

본 논문에서 제시한 강화학습 기반의 공격 대응 접근 방식은 악성 컨테이너에 대한 자원 제한에 중점을 두고 있지만, 이 방식은 기존 연구에서 쿠버네티스 환경에서 자원 사용의 효율성을 향상시키는 것을 목표로 하는 동적 자원 관리 접근 방식과 유사성을 갖고 있다. 따라서 두 시나리오의 강화학습 환경 정의에 대한 메트릭이 유사하므로, 제안한 프레임워크를 구현하여 보안 관점에서 자원 효율성 최적화를 위한 목적으로 활용했을 때의 성능 평가를 수행할 것이다. 더불어, 자원 고갈 공격에 대한 강화학습 기반의 공격 대응 방식을 기존 알고리즘 및 시스템과 비교하여 평가하는 후속 연구를 수행할 예정이다.

## 참고문헌


[1] 박재현, “황금기 맞이한 국내 쿠버네티스 시장,” http://www.itdaily.kr/news/articleView.html?idxno=212840, 2023.

[2] Kubernetes - Horizontal Pod Autoscaler, https://kubernetes.io/ko/docs/tasks/run-application/horizontal-pod-autoscale/, 접속 : 2023-09-27.

[3] Google Cloud, “Threat Horizons Cloud Threat Intelligence 2021,” https://bit.ly/41THxbT, 2021

[4] C. N. C. Foundation, “Cloud native survey 2020,” https://www.cncf.io/wp-content/uploads/2020/11/CNCF_Survey_Report_2020.pdf, 2020.

[5] A. A. Khaleq and I. Ra, "Intelligent Autoscaling of Microservices in the Cloud for Real-Time Applications," in IEEE Access, vol. 9, pp. 35464-35476, 2021.

[6] S. Horovitz and Y. Arian, "Efficient Cloud Auto-Scaling with SLA Objective Using Q-Learning," 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 85-92, 2018.

[7] T. Veni and S. Mary Saira Bhanu, "Auto-scale: automatic scaling of virtualised resources using neuro-fuzzy reinforcement learning approach," International Journal of Big Data Intelligence 3.3, pp. 145-153, 2016.

[8] J. V. Bibal Benifa and D. Dejey, "Rlpas: Reinforcement learning-based proactive auto-scaler for resource provisioning in cloud environment," Mobile Networks and Applications 24, pp. 1348-1363, 2019.

[9] F. Rossi, M. Nardelli and V. Cardellini, "Horizontal and Vertical Scaling of Container-Based Applications Using Reinforcement Learning," 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), pp. 329-338, 2019.

[10] Z. Zhang, T. Wang, A. Li and W. Zhang, "Adaptive Auto-Scaling of Delay-Sensitive Serverless Services with Reinforcement Learning," 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 866-871, 2022.

[11] C. Priebe, D. Muthukumaran, D. O'Keeffe, D. Eyers, B. Shand, R. Kapitza and P. Pietzuch, "Cloudsafetynet: Detecting data leakage between cloud tenants," Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security, pp. 117-128, 2014.

[12] 이준희, 남재현, and 김진우, "컨테이너 환경에서의 호스트 자원 고갈 공격 영향 분석," 정보보호학회논문지 33.1, pp. 87-97, 2023.

[13] M. Musch, C. Wressnegger, M. Johns and K. Rieck, "Thieves in the browser: Web-based cryptojacking in the wild," Proceedings of the 14th International Conference on Availability, Reliability and Security, pp. 1-10, 2019.

[14] F. Gomes and M. Correia, "Cryptojacking Detection with CPU Usage Metrics," 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), pp. 1-10, 2020.

[15] A. D. Yulianto, P. Sukarno, A. A. Warrdana and M. A. Makky, "Mitigation of Cryptojacking Attacks Using Taint Analysis," 2019 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), pp. 234-238, 2019.

[16] R. R. Karn, P. Kudva, H. Huang, S. Suneja and I. M. Elfadel, "Cryptomining Detection in Container Clouds Using System Calls and Explainable Machine Learning," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 3, pp. 674-691, 2021.

[17] M. Zhan, Y. Li, H. Yang, G. Yu, B. Li and W. Wang, "Coda: Runtime Detection of Application-Layer CPU-Exhaustion DoS Attacks in Containers," in IEEE Transactions on Services Computing, vol. 16, no. 3, pp. 1686-1697, 2023.

[18] A. F. Baarzi, G. Kesidis, D. Fleck and A. Stavrou, "Microservices made attack-resilient using unsupervised service fissioning," Proceedings of the 13th European workshop on Systems Security, pp. 31-36, 2020.

[19] J. Castro, N. Laranjeiro and M. Vieira, "Detecting DoS Attacks in Microservice Applications: Approach and Case Study," Proceedings of the 11th Latin-American Symposium on Dependable Computing, pp. 73-78, 2022.

[20] eBPF, https://ebpf.io/, 접속 : 2023-09-27.

[21] M. Caprolu, S. Raponi, G. Oligeri and R. Di Pietro, "Cryptomining makes noise: Detecting cryptojacking via machine learning," Computer Communications 171, pp. 126-139, 2021.

[22] H. N. C. Neto, M. A. Lopez, N. C. Fernandes and D. M. Mattos, "Minecap: super incremental learning for detecting and blocking cryptocurrency mining on software-defined networking," Annals of Telecommunications 75, pp. 121-131, 2020.

[23] A. Gangwal, S. G. Piazzetta, G. Lain and M. Conti, "Detecting covert cryptomining using hpc," Cryptology and Network Security: 19th International Conference, pp. 344-364, 2020.

[24] H. Darabian, S. Homayounoot, A. Dehghantanha, S. Hashemi, H. Karimipour, R. M. Parizi and K. K. R. Choo, "Detecting cryptomining malware: a deep learning approach for static and dynamic analysis," Journal of Grid Computing 18, pp. 293-303, 2020.

## 〔저자소개〕

김 리 영 (Ri-Yeong Kim)
2023년 2월: 성신여자대학교 융합보안공학과 졸업
2023년 3월~현재: 성신여자대학교 미래융합기술공학과 석사
email : 220236036@sungshin.ac.kr

김 성 민 (Seong-min Kim)
2012년 2월: 한국과학기술원 전기 및 전자공학과 졸업
2014년 2월: 한국과학기술원 전기 및 전자공학과 석사
2019년 2월: 한국과학기술원 정보보호대학원 박사
2019년 9월~2020년 8월: 삼성전자 삼성리서치 Staff Engineer
2020년 9월~현재: 성신여자대학교 융합보안공학과 조교수
email : sm.kim@sungshin.ac.kr