Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.6, Dec. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.6.1205

컨테이너 기술을 활용한 엣지 컴퓨팅 환경 어플리케이션 무결성 보호에 대한 연구

이 창 훈,[1†] 신 영 주[2‡]
[1,2]고려대학교 정보보호대학원 (대학원생, 교수)

A Study on Integrity Protection of Edge Computing Application Based on Container Technology

Changhoon Lee,[1†] Youngjoo Shin[2‡]
[1,2]School of Cybersecurity, Korea University (Graduate student, Professor)

요 약

엣지 컴퓨팅(Edge Computing)은 인공지능(AI)을 데이터 소스와 근접한 환경에서 수행함으로써 IoT/CPS 기기를 클라우드에 통합하는데 발생하는 네트워크 대역폭 소모로 인한 비용 문제와 전송 지연 등의 문제 해결의 방안으로 주목받고 있다. 엣지 컴퓨팅 기기는 실 세계에 위치하여 인공지능 구현 기술을 구동 가능한 수준의 향상된 연산과 네트워크 연결을 제공하므로, 인적/물적 피해를 발생할 수 있는 사이버 테러에 악용되지 않도록 어플리케이션 무결성에 대한 고려가 필요하다. 본 논문에서는 인공지능 구현 시 활용되는 파이썬(python) 과 같이 변조에 취약한 스크립트 언어로 구현된 엣지 컴퓨팅 어플리케이션을 컨테이너 이미지로 구성 후 전자서명을 하여 무결성을 보호하는 기법을 제안한다. 제안하는 기법은 오픈소스 컨테이너 기술에서 제공하는 무결성 보호기술 (Docker Contents Trust)를 기반으로하며, 엣지 컴퓨팅 기기에서 허용된 컨테이너만 구동 가능하도록 컨테이너 서명 정보에 대한 화이트리스트와 Docker Client를 개선하여 적용하는 기법을 제시한다.

ABSTRACT

Edge Computing is used as a solution to the cost problem and transmission delay problem caused by network bandwidth consumption that occurs when IoT/CPS devices are integrated into the cloud by performing artificial intelligence (AI) in an environment close to the data source. Since edge computing runs on devices that provide high-performance computation and network connectivity located in the real world, it is necessary to consider application integrity so that it is not exploited by cyber terrorism that can cause human and material damage. In this paper, we propose a technique to protect the integrity of edge computing applications implemented in a script language that is vulnerable to tampering, such as Python, which is used for implementing artificial intelligence, as container images and then digitally signed. The proposed method is based on the integrity protection technology (Docker Contents Trust) provided by the open source container technology. The Docker Client was modified and used to utilize the whitelist for container signature information so that only containers allowed on edge computing devices can be operated.

Keywords: Edge Computing, Application Integrity, Docker, Container, Docker Contents Trust

Received(10. 21. 2021), Modified(11. 10. 2021), Accepted(11. 17. 2021)

† 주저자, changhoon47lee@korea.ac.kr
‡ 교신저자, syoungjoo@korea.ac.kr(Corresponding author)

I. 서 론

엣지 컴퓨팅(Edge Computing) 기술은 증강현실 및 인공지능(AI) 기술을 데이터 소스와 근접한 환경에서 수행함으로써 사용자에게 실시간 서비스를 제공하며, IoT/CPS 기기를 클라우드에 통합시 발생하는 네트워크 대역폭 소모로 인한 비용 문제와 전송 지연문제에 대한 해결 방안으로 주목받고 있다.

엣지 컴퓨팅 기술의 적용 사례로 스마트 팩토리 기기들에 대한 유지보수/생산성 향상, 스마트 팜에서의 온/습도 자동화 처리, 자율주행 자동차의 운행, 가상현실/증강현실 기기를 활용한 새로운 인터페이스 제공을 위한 기술을 확인할 수 있다 [1][2].

엣지 컴퓨팅 어플리케이션은 스마트 팩토리와 같은 현장에서 수집되는 데이터를 기반으로 인공지능(AI) 연산을 수행하여 수집된 데이터에 대한 시각화 및 실제 환경에 대한 영상 모니터링 효율적으로 수행할 수 있도록 구성할 수 있으며, 이에 활용되는 인공지능 연산 및 기능은 파이썬과 같은 스크립트 언어를 기반으로 구현될 수 있다 [3][4].

엣지 컴퓨팅은 개념적으로 실세계와 근접한 위치에서 동작하는 기술로, 기존의 스턱스 넷이나 우크라이나 정전사태와 같은 융합보안환경(IoT/CPS)의 공격사례가 그대로 재현될 수 있는 환경으로 예상할 수 있다. 반면, 클라우드 환경에서 활용되어온 쉽게 수정 가능한 스크립트 방식의 구현언어를 그대로 엣지 컴퓨팅 환경에서 활용한 인공지능 구현이 증가할 것으로 예상되어, 향후 엣지 컴퓨팅 환경의 어플리케이션 무결성 침해로 인한 공격의 가능성이 높아 질 수 있을 것으로 예상된다.

이에 본 논문은 인공지능 구현시 사용되는 파이썬과 같이 변조에 취약할 수 있는 스크립트 언어를 활용하는 엣지 컴퓨팅 어플리케이션을 컨테이너로 구성 후 전자서명을 통해 무결성을 보호하는 기법을 제안한다. 제안하는 기법은 오픈소스 컨테이너 기술에서 제공하는 무결성 보호기술인 Docker Contents Trust를 기반으로하며, 엣지 컴퓨팅 기기에서 허용된 컨테이너만 구동 가능하도록 컨테이너 서명 정보에 대한 화이트리스트를 활용하도록 Docker Client를 개선한 기법을 추가로 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구와 오픈소스 컨테이너 기술에서 제공하는 무결성 보호를 위한 기술을 살펴보고, 3장에서는 오픈소스 컨테이너 무결성 보호 기술을 기반으로 엣지 컴퓨팅 환경에서 허용된 컨테이너만 구동되도록 제한하는 기법을 제안하며, 4장에서는 제안된 기법에 대한 구현 내용 및 검증 내용을 설명한 뒤, 마지막으로 5장에서 결론을 맺는다.

II. 관련 연구 및 배경

2.1 관련 연구

기존의 연구들은 엣지 컴퓨팅 어플리케이션의 무결성 보호 측면보다는 주로 가용성 및 배포/관리 용이성 측면에서의 컨테이너 기술들을 다루고 있다 [5][6][7].

IoT 기기의 어플리케이션 및 구성요소의 해시를 사전에 목록화하여 관리하는 기법[8]은, OS 및 시스템 프로세스에 대한 보호 기법을 제시하고 있으나, 본 연구는 컨테이너 기반 어플리케이션에 대한 보호를 제시하고 있어 그 범위가 다르다고 할 수 있다.

TPM (Trusted Platform Module)과 Linux IMA(Integrity Measurement Architecture)를 활용한 컨테이너 무결성 체크 기법에 대한 연구[9]는 컨테이너 구동 이후 변조 상황에 대한 런타임 탐지기법을 제시하고 있으나, 본 연구에서는 컨테이너 배포 시점에 대한 내용을 제시하고 있어 그 범위가 다르다고 할 수 있다.

상용 클라우드(Google) 컨테이너 저장소[11]에서 제공하는 컨테이너 배포 보호는 서버 등록정보를 이용하여 서버 환경에 컨테이너를 배포하는 시점에 컨테이너를 차단하고 있으나. 본 연구는 엣지 컴퓨팅 환경을 대상으로 기기에 사전에 배포된 정보를 이용하여, 기기에서 컨테이너 배포 후 구동 시점에 차단하고 있어, 상용 클라우드와 대상으로 하는 환경과 차단 방식이 다르며 상호 보완하여 적용할 수 있다.

2.2 배경 기술

본 연구에서는 다음의 오픈소스 컨테이너 기술들을 활용하여 연구 범위인 컨테이너 배포 후 구동 시점에 컨테이너 무결성 체크를 수행한다.

2.2.1 Docker Container

컨테이너는 코드와 종속 모듈을 함께 패키징하는 어플리케이션 레이어의 추상화이며, 여러 컨테이너가

동일한 시스템에서 OS 커널을 공유하되, 다른 컨테이너와 격리된 프로세스의 형태로 실행된다.

2.2.2 Docker Contents Trust

Docker Contents Trust (DCT) 는 컨테이너 이미지 저장소 (Registry)와 컨테이너 이미지 전자서명 서버 (Notary)를 제공한다. 개발자는 어플리케이션을 컨테이너 이미지로 구성하여 Registry에 등록 후, 등록된 이미지에 대해 전자서명을 수행할 수 있다. 사용자는 Docker Client에 대한 명령을 통해 컨테이너 이미지 저장소로부터 컨테이너 이미지를 다운로드 후 전자서명 검증을 통해 이미지의 무결성을 확인 후 컨테이너를 구동한다.

III. 제안기법

3.1 제안 배경

3.1.1 Docker DCT 고려사항

DCT는 컨테이너 이미지 저장소에 저장되는 컨테이너 이미지에 대해 전자서명 적용 기능을 제공하지만, 아래의 내용과 같은 고려사항들이 있다[10].

첫번째로, 컨테이너 이미지에 대한 전자서명 여부는 등록자의 선택사항이며, 컨테이너 이미지 저장소에는 서명된 컨테이너 이미지와 서명되지 않은 컨테이너 이미지 모두 등록될 수 있다. 또한 컨테이너 이미지 저장소에는 여러 종류의 어플리케이션에 대한 컨테이너 이미지가 등록되며, 관리를 위해 다수의 등록자가 등록될 수 있다.

두번째로, 컨테이너 이미지 전자서명 권한은 DCT의 위임(delegation) 기능을 통해 공유할 수 있으며, 위임은 공개키쌍 (public/private key pair) 생성 및 컨테이너 저장소에 위임 정보를 추가하는 단계로 진행된다. DCT에서는 공개키쌍을 생성하기 위한 명령 (docker trust key generate)과 위임 정보를 저장소에 추가하기 위한 명령 (docker trust signer add)을 제공한다. 위임을 위한 공개키쌍은 DCT에서 제공하는 명령 외에도 신뢰할 수 있는 CA를 통해 인증서를 생성하는 방법 및 openssl과 같은 일반적인 도구를 활용하여 생성 후 자체서명으로 생성하는 방법 모두 사용할 수 있다.

세번째로, 컨테이너 이미지를 사용하는 Docker 클라이언트는 기본으로 DCT 설정이 비활성화 상태로 DCT 사용을 위해서는 DCT 설정을 활성화해야 한다. DCT 설정은 일종의 필터와 같이 동작하여, 활성화시 컨테이너 이미지 저장소에서 전자서명이 적용된 컨테이너 이미지만 사용 가능하며, 비활성화 시 전자서명 여부와 무관하게 등록된 모든 이미지를 사용가능하도록 동작한다.

3.1.2 공격 모델

본 논문에서는 대상 시스템을 크게 DCT 서버와 엣지 컴퓨팅 기기로 나눌 수 있다. DCT 서버에는 컨테이너 저장소와 전자서명 서버가 설치되고 엣지 컴퓨팅 기기에는 DCT 설정이 사전에 활성화 되어 Docker 클라이언트가 설치된다.

공격자는 DCT 서버의 침해를 통해 엣지 컴퓨팅 기기에 악성 어플리케이션을 배포하는 것을 목표로 하는 것을 가정하였으며, 엣지 컴퓨팅 기기는 이미 실환경에 배포 완료된 상태로 공격자가 직접 접근하는 것이 어려운 상태인 것으로 가정하였다.

공격자는 먼저 단순히 컨테이너 이미지 저장소에 원격 접속을 확보하는 것을 가정하였다. 컨테이너 이미지 저장소는 컨테이너 이미지를 외부에 제공하기 위한 서비스로 Docker 클라이언트로 접속 가능한 위치를 확보하면, 컨테이너 이미지 저장소에 등록되어있는 이미지를 조회 및 다운로드 할 수 있다.

그 다음 공격자는 컨테이너 이미지를 다운로드 할 수 있는 환경에서 이미지에 대한 변경 및 분석을 수행할 수 있다. 이때 컨테이너 이미지에 대한 변경은 단순히 공격자의 환경 내에서만 이루어지며, 컨테이너 이미지 저장소에는 반영되지 않는 단계이다.

이후 공격자는 변경된 컨테이너 이미지를 저장소에 반영하기 위해 공격자가 DCT의 위임권한을 확보하는 것을 가정하였으며, 위임권한 확보 후 공격자는 변조된 컨테이너 이미지와 자신이 생성한 전자서명 키를 저장소에 등록 후 전자서명을 새로 수행하여 이미지의 기존 전자서명 정보를 대체할 수 있다.

제시된 공격모델과 같이 서버환경 침해는 어려운 절차일 수 있으나, 최근 스피어피싱을 통한 서버환경 침해사례들이 있으며, 과거 스턱스넷이나 우크라이나 정전사태 등 폐쇄망 환경에 대한 침해 사례들도 있어 서버환경 침해가 가능함을 예상할 수 있다.

3.1.3 공격 시나리오

오픈소스 컨테이너 기술은 DCT 기능을 제공하여, 작성된 컨테이너 이미지를 별도의 서버로 구성되는 컨테이너 이미지 저장소에 등록 후, 컨테이너 이미지에 대한 전자서명을 수행하여 컨테이너에 대한 무결성을 유지할 수 있는 기능을 제공한다. 따라서, 엣지 컴퓨팅 환경에서 활용되는 파이썬과 같은 스크립트 기반의 인공지능 어플리케이션들도 컨테이너 형태 배포를 통해 무결성 유지를 기대할 수 있다.

그러나 Fig. 1.의 공격 시나리오와 같이 공격자가 컨테이너 이미지 저장소(Registry) 및 컨테이너 이미지 전자서명 서버(Notary)에 대한 권한을 획득한 상황에서 공격자는 자신의 공격용 컨테이너 이미지를 저장소에 등록 후, 스스로 신규 생성한 전자 서명키를 이용하여 저장소의 이미지를 서명 할 수 있으며, 엣지 컴퓨팅 기기에서는 Docker Client를 이용하여 컨테이너 저장소에서 공격자의 컨테이너 이미지를 다운로드하여 실행할 수 있다. 이 시나리오에서 행위자는 정상 사용자(User)와 공격자(Attacker)로 구분할 수 있으며, 컴포넌트는 서버 컴포넌트(Registry/Notary), 엣지 컴포넌트(Docker Client) 로 구분할 수 있다.

시나리오의 내용 중 정상적으로 컨테이너 이미지를 배포/구동하는 시나리오를 각 단계별로 살펴보면 다음과 같다. 먼저 "1-1.User's container and sign key" 단계에서 정상 사용자는 파이썬과 같은 스크립트를 이용하여 엣지 컴퓨팅 어플리케이션을 개발 후 배포를 위해 컨테이너 이미지 형태로 준비한다. 또한 정상 사용자는 배포 컨테이너 이미지의 무결성 보호를 위한 전자서명 키를 생성하여 보유한다.

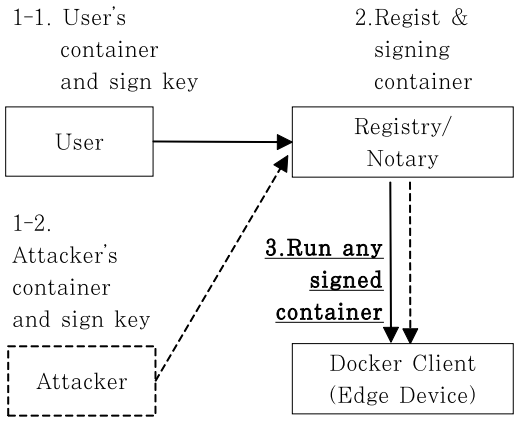

Fig. 1. Attack scenario

그 다음 단계 "2. Regist & signing container" 에서 사용자는 컨테이너 저장소(Registry)에 컨테이너를 업로드하여 등록하고, 전자 서명키를 컨테이너 저장소와 연동되어있는 컨테이너 전자서명 서버(Notary)에 등록하여 업로드된 컨테이너에 대해서 전자서명을 수행한다. 이때 사용자의 전자 서명키는 사용자가 지정한 비밀번호(Password)를 통해 보호된다.

그 다음 단계, "3. Run any signed container" 에서 엣지 클라이언트에 설치되어있는 Docker Client는 컨테이너 저장소에 등록된 컨테이너를 선택하여 다운로드 후 구동할 수 있다.

다음으로 시나리오의 내용 중 공격 시나리오에 대해서 살펴보면 다음과 같다. 이 시나리오에서 공격자는 사전에 서버 컴포넌트(Registry/Notary)에 대한 등록 권한을 사회공학적 방법이나 어플리케이션 취약점 침해를 통한 방법 등으로 이미 획득한 것을 가정하였으며, 정상 사용자가 설정한 비밀번호로 보호되는 전자서명 키에 대한 권한은 확보하지 못한 상태로 가정하였다.

공격자는 항상 구동되어 서비스를 수행하는 서버 컴포넌트에 대한 접근 시도가 상대적으로 용이하며, 서버 컴포넌트에 접근 후 내재되어있는 취약점에 대한 착취 시도를 수행할 수 있다. 서버 컴포넌트에 대한 침해가 이루어지면, 침해 범위를 엣지 컴퓨팅 환경으로 확장을 위한 정보를 수집하므로 컨테이너 전자서명 키와 같은 상세 정보는 상대적으로 공격 후반에 확보되는 것으로 공격 시나리오를 가정하였다.

공격 시나리오의 각 단계별 절차는 "1-2. Attacker's container and sign key" 단계에서 공격자가 정상 사용자의 컨테이너를 변조 또는 스스로 작성한 공격용 컨테이너를 준비하는 것으로 시작된다. 공격자는 등록 시 비밀번호로 보호되는 정상 사용자의 컨테이너 전자서명 키를 확보하지는 못한 상태이므로, 공격자 스스로 새로운 전자서명 키를 생성하여 보유한다.

그 다음 "2. Regist & signing container" 단계에서 공격자는 공격용 컨테이너를 컨테이너 저장소에 업로드 및 공격자의 전자서명 키를 전자서명 서버에 등록 후 공격용 컨테이너에 대한 전자서명을 수행한다.

마지막으로, "3. Run any signed container" 단계에서 공격용 컨테이너에 대한 구동 명령을

Docker Client가 수행하도록 함으로써 엣지 컴퓨팅 기기상에 공격용 컨테이너가 실행 될 수 있다. 이때 공격용 컨테이너에 대한 구동 명령은 엣지 컴퓨팅 기기에 대한 직접적인 침해 없이도, 정상 컨테이너와 유사한 명칭 또는 버전 정보 등을 공격용 컨테이너에 부여함으로써 정상적인 배포 절차에 혼란을 유발하여 수행될 수 있다.

엣지 컴퓨팅 기기에 설치되어 동작하는 Docker 클라이언트는 컨테이너 저장소에서 제공하는 컨테이너 이미지에 대해서 전자서명이 검증만 가능하다면 컨테이너 구동을 수행하므로, 현재의 오픈소스 컨테이너 환경은 컨테이너 저장소 및 컨테이너 전자서명 서버에 대한 침해가 발생하는 경우, 실 환경에서 구동되고 있는 다수의 엣지 컴퓨팅 기기에 악성 컨테이너 이미지가 동시에 배포되어 실행할 수 있는 위협이 존재한다고 할 수 있다.

따라서 본 연구에서는 오픈소스 컨테이너 환경의 공급망 공격 시나리오로 악용될 수 있는 위의 시나리오를 방지할 수 있는 기법을 제안하고자 한다.

3.2 제안 기법 설계

본 연구에서는 제안 배경에서 제시한 문제점을 해결하기 위해서 엣지 컴퓨팅 기기에서 구동 가능한 컨테이너 이미지의 전자서명 식별자에 대해서 화이트리스트를 설정하여 지정된 컨테이너 이미지만이 구동될 수 있도록 하는 방안을 제시하고자 한다.

본 제안 기법은 오픈소스 컨테이너 무결성 보호 기술인 DCT를 기반으로 하여, 컨테이너 생성, 전자서명 적용, 컨테이너 이미지에 대한 저장소 저장 등의 절차는 유지하여 기존 컨테이너 이미지와 서버 환경을 그대로 활용할 수 있도록 한다.

그러나 엣지 컴퓨팅 기기에는 구동 대상 컨테이너에 대한 필터링을 할 수 있는 기능이 포함된 수정된 버전의 Docker Client와 구동 대상 컨테이너 이미지에 대한 서명키 식별자 목록에 대한 화이트리스트(Whitelist)를 사전 탑재하여 실 환경에 배포하는 형태로 설계되었다. 컨테이너 이미지에 대한 메타정보(Meta data) 중 서명키 식별자를 화이트리스트 대상으로 활용하는 이유는 해당 데이터가 어플리케이션 무결성과 직접적으로 관련이 있는 데이터이며, 향후 운영 중 어플리케이션에 대한 업데이트가 필요시 기기에 사전 배포된 화이트리스트는 그대로 유지한 상태에서, 해당 전자서명 키로 신규 어플리케이션을 다시 서명하여 배포 가능하기 때문이다.

일반적으로 엣지 컴퓨팅 기기는 용도에 맞는 시스템을 구성하여 실환경에 배포되므로 수정된 버전의 Docker Client와 사전 정의된 화이트리스트를 기기에 탑재 후, 파이썬과 같은 스크립트 기반 언어를 활용한 어플리케이션을 컨테이너 형태로 배포하는 것은 실 환경에서도 적용 가능하다고 할 수 있다.

또한, 엣지 컴퓨팅 기기의 화이트리스트 및 관련 어플리케이션 코드에 대한 변조를 방지하기 위해 하드웨어 기반 암호화 기술을 적용할 수도 있다. 하드웨어 기반 암호화 는 메모리 내에서 코드 및 데이터를 격리하여 처리하는 기술로 적용 대상 환경에 따라 Intel SGX 및 Arm TrustZone과 같은 기술을 활용할 수 있다.

구동 대상 컨테이너 이미지를 필터링하기 위한 화이트리스트에 대한 처리는 Fig. 2. 에 기술된 알고리즘과 같다. 정의된 알고리즘은 컨테이너 이미지 배포 시점에 사용되며, 구동 대상 컨테이너 이미지의 서명키 식별자 (keyid)를 화이트리스트에 등록된 컨테이너 서명키 식별자 들과 비교 후, 허용된 내역이 없는 경우 오류를 발생하는 형태로 설계되었다.

제안 기법을 적용 후 기대되는 동작은 Fig. 3. 과 같이 전자서명 키 식별자를 기준으로 미리 허용된 컨테이너 이미지만 배포 가능하며, 이외의 컨테이너 이미지에 대해서는 차단되는 형태이다.

제안 기법 적용 후 시나리오에 대해 각 단계별로 살펴보면 다음과 같다. (Fig. 3. 참고)

먼저, 정상 사용자는 "1-1. User's container and sign key" 단계에서 엣지 컴퓨팅 기기로 배포

```
sign_key_id := getReceivedSignKeyId()
isPermitted := false
// Check whitelist
for item in whitelist {
    if (item == sign_key_id) {
        // Permit Container
        isPermitted = true
        break;
    }
}
// Block Container
if !isPermitted {
    error("Not permitted sign key id")
}
```

Fig. 2. Whitelist-based container checking logic

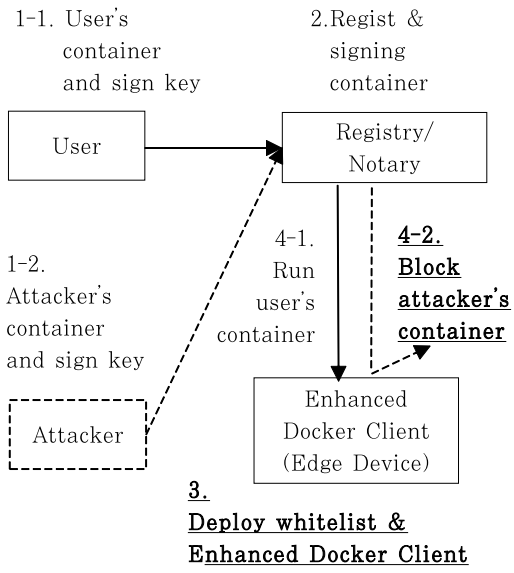

Fig. 3. Protection Scenario

하기 위한 파이썬 스크립트 기반 어플리케이션을 컨테이너 이미지 형태로 준비하고 이미지에 대한 전자서명키를 준비한다.

그 다음, 단계 "2. Regist & signing container" 에서 준비된 컨테이너 이미지와 전자서명 키를 컨테이너 이미지 저장소(Registry)와 전자서명 서버(Notary)에 각각 등록 후, 컨테이너 이미지에 대한 전자서명을 수행한다. 이 단계까지는 제안기법 적용 이전 DCT를 활용하는 단계와 동일하다.

그 다음 단계인 "3. Deploy whitelist & Enhanced Docker Client" 단계에서는 컨테이너 이미지 전자서명 서버에 서명키를 등록 후 생성되는 서명키에 대한 식별자를 화이트리스트로 구성하여 엣지 컴퓨팅 기기에 배포한다. 또한 기기에 배포된 화이트리스트를 기반으로 컨테이너 이미지 구동을 제한하는 알고리즘이 탑재된 Docker Client를 엣지 컴퓨팅 기기에 함께 배포하여 구동한다.

이후, 정상 사용자는 "4-1. Run user's container" 단계에서 정상 컨테이너 이미지를 엣지 컴퓨팅 기기에서 정상적으로 구동할 수 있다.

하지만, 공격자는 "1-2. Attacker's container and sign key" 와 "2. Regist & signing container" 단계를 수행하여 공격용 컨테이너 이미지에 대해서 전자서명까지는 수행할 수는 있으나, 엣지 컴퓨팅 기기에서 컨테이너 이미지에 대한 구동시도 시, 이전에 배포된 화이트리스트와 Docker Client에 탑재된 컨테이너 허용 알고리즘에 따라 "4-2. Block attacker's container" 단계와 같이 사전에 허용되지 않은 공격자의 컨테이너 구동을 차단한다. 위의 동작 시나리오는 서버 컴포넌트의 침해로 공격자의 컨테이너 이미지가 저장소에 등록된 상황에서도 제안 기법을 활용하여 엣지 컴퓨팅 기기로의 침해 전파를 차단할 수 있음을 보여준다.

IV. 구현 및 검증

4.1 제안기법 구현

본 논문에서는 엣지 컴퓨팅 환경 어플리케이션의 무결성 보호를 제공하기 위해, 오픈소스 컨테이너 무결성 보호기술인 DCT 를 기반 기술로 활용하여, 공격자가 변조된 이미지를 이미지 저장소 서버에 등록하여 엣지 컴퓨팅 기기로 배포하는 것을 차단할 수 있는 기법을 제안한다. 본 제안기법의 구현을 위해 아래 Table 1.과 같이 테스트 환경을 구성하였다.

제안 기법에서 제공하는 컨테이너 화이트 리스트 로직(Fig. 2. 참고)의 구현을 위해 github에 등록된 Docker Client 20.10 버전의 소스코드를 수정 및 빌드하여 구성하였다. 또한 제안기법 검증을 위한

Table 1. Test environment

Type	Specification
Server	H/W - 1 CPU, 2 GB RAM (virtual box) S/W - Ubuntu 20.04 x64 - Docker 20.10.7 - Docker Registry 2.7.1 - Docker Notary 0.7.0 - Python 3.8.10
Client	H/W - 1 CPU, 1 GB RAM (virtual box) S/W - Ubuntu 20.04 x64 - Docker Client (20.10, modified)

테스트 어플리케이션은 파이썬 3.8.10을 사용하여 작성하였으며, 작성된 어플리케이션을 정상 구동 및 변조 적용 후 구동 용도로 활용하였다.

4.2 제안기법 검증

4.2.1 정상 어플리케이션

정상 어플리케이션은 엣지 컴퓨팅 기기 환경에서 스크립트 언어 활용을 가정하여 파이썬 기반 어플리케이션을 컨테이너 형태로 배포하도록 작성하였다. 정상 어플리케이션은 구동 후 웹브라우저로 접속 시 간단한 메시지 출력을 통해 정상 여부를 확인할 수 있도록 구성하였다.

생성된 컨테이너 이미지를 Registry 서버에 등록 후, 등록된 상태 조회를 위한 명령어 (docker trust inspect) 수행 결과로 조회되는 메타정보에 같이 등록된 컨테이너 및 서명 관련 식별자 내용이 “Signer” 항목 내 “ID” 필드 값으로 포함되어있는 것을 확인 할 수 있다 (Fig. 4. 참고).

변조 어플리케이션은 앞서 작성한 정상 어플리케이션의 컨테이너에 접근(docker container exec) 후, 텍스트 편집을 통해 정상 어플리케이션의 출력 메시지를 변조 후 다시 공격용 컨테이너 이미지로 저장(docker commit)하는 방식으로 준비하였다.


```
"Name":
"registry-server:443/my-app:1.0.0",
"SignedTags": [{
    "SignedTag": "1.0.0",
    "Digest":
    "6ec77085d8ac1b3ee26fcf4a10608809
    c5110b637981ce5716a29a977880b1be",
    "Signers": [
    "my-app-signer"
    ]}],
"Signers": [{
    "Name": "my-app-signer",
    "Keys": [{
    "ID":
    "27c43278e3988ccce0e7690b949706c4
    03e16c35644e1c2282c0d059ed1301d4"
    }]}]
```


Fig. 4. Meta data for a test application

```
# docker run --name malicious-app -d -p
8080:80/tcp malicious-app
  docker: Error: remote trust data does
not exist for
docker.io/library/malicious-app:
  notary-server:4443 does not have trust
data for docker.io/library/malicious-app.
```

Fig. 5. Blocking unsigned malicious app

파이썬으로 구현된 정상 어플리케이션은 간단한 명령어를 통해 텍스트를 업데이트하는 방식으로 어플리케이션 변조 수행이 가능하였으며, 실환경 파이썬 어플리케이션에 대해서도 유사한 방법으로 중요 로직에 대한 변조가 가능함을 예상할 수 있다.

위와 같이 변조된 컨테이너 이미지를 DCT 설정을 사용하는 환경에서 구동 시 다음과 같이 전자서명 검증을 진행하지 못하는 오류 (Fig. 5. 참고)가 발생하여, 파이썬과 같은 스크립트 기반 어플리케이션을 컨테이너 형태로 구성시 DCT를 통해 기본적인 변조에 대한 대응이 가능한 것을 확인할 수 있다.

그러나, 앞서 설명한 바와 같이 공격자가 컨테이너 이미지 저장소와 컨테이너 이미지 전자서명 서버에 대한 권한 탈취 및 취약점 악용 등을 통해 컨테이너 등록 권한을 확보한 경우, 정상 어플리케이션에 대한 전자서명 키를 사용하지 않고도 공격자가 스스로 신규 생성한 전자 서명키를 사용하여 변조된 컨테이너에 대해 전자서명을 수행으로써 변조된 어플리케이션을 정상 구동 되도록 할 수 있다

4.2.2 변조 어플리케이션 구동 차단

공격자가 스스로 생성한 전자서명 키를 이용하여 컨테이너를 저장소에 등록하였기 때문에, 전자서명에 사용된 signer의 정보가 정상 어플리케이션과 다른 값으로 서버에 저장된 것을 아래와 같이 확인 할 수 있다. 정상 어플리케이션의 등록 상태 조회 시 사용하였던 명령어(docker trust inspect)를 활용하여 변조 어플리케이션에 대한 메타정보를 확인 할 수 있다 (Fig. 6. 참고).

DCT를 사용하는 Docker Client는 서버에서 컨테이너 이미지 다운로드 전 메타정보를 먼저 다운로드 후, 컨테이너 다운로드, 전자서명 검증 및 구동을 수행한다. 앞서 살펴본 바와 같이 공격자가 스스로

```
"Name":
"registry-server:443/malicious-app:1.0.0",
  "SignedTags": [{
        "SignedTag": "1.0.0",
        "Digest":
      "9100957e1b4a0f99c96849a7098d1808
      7ed10cb34afc7dcc82daba7846e9c828",
        "Signers": [
          "malicious-app-signer"
        ]}],
  "Signers": [{
        "Name": "malicious-app-signer",
        "Keys": [{
         "ID":
 "2713bcca359158c27f1ae548622b3f30d
  283047f22817ea76a64a0d9f983e5ab"
        }]}]
```

Fig. 6. Meta data for a malicious application

전자서명 키를 생성하여 변조된 컨테이너를 전자서명 후 배포하는 경우에도 관련 내용이 메타 정보에 포함되어있는 것을 확인할 수 있다.

따라서, 배포대상 정상 어플리케이션에 대한 메타정보를 기반으로 한 화이트리스트를 사전에 엣지 컴퓨팅 기기에 설정해 둠으로써 허용되지 않은 컨테이너를 서버로부터 다운로드하여 구동하는 것을 사전에 차단하는 데 활용 할 수 있다.

본 연구에서는 엣지 컴퓨팅 기기 배포 전 서버에 등록된 메타정보 중 컨테이너 전자서명 수행 시 생성

```
# docker pull
registry-server:443/malicious-app:1.0.0
  panic:
2713bcca359158c27f1ae548622b3f30d283047
f22817ea76a64a0d9f983e5ab is not
permitted key ID
  goroutine 1 [running]:
  github.com/docker/cli/vendor/github.co
m/theupdateframework/notary/tuf/signed.
VerifySignatures(0xc000090b80,
0xc000255650, 0xc0006d5240, 0x1c, 0x1,
0x0, 0x0)
```

Fig. 7. Blocking not permitted applications

되는 signer 항목의 keyId 데이터의 목록을 시스템 환경변수로 설정되도록 하였으며, 컨테이너 이미지 다운로드 시 해당 환경변수를 기준으로 메타정보를 먼저 확인하도록 수정한 버전의 Docker Client를 사용하였다. 제시한 기법을 적용하여 변조된 어플리케이션의 컨테이너 이미지에 대해서 다운로드 요청시 아래와 같이 오류를 발생시켜 허용되지 않는 컨테이너 구동을 차단할 수 있음을 확인하였다 (Fig. 7. 참고).

V. 결 론

현재 엣지 컴퓨팅 환경은 오픈소스 S/W 기술과 라즈베리파이와 같은 저비용 범용 H/W를 기반으로 손쉽게 구성할 수 있어, 향후 엣지 컴퓨팅 기술을 활용하기 위한 다양한 시도가 이루어 질 것으로 예상할 수 있다. 또한 비전문가도 상대적으로 쉽게 접근할 수 있는 파이썬과 같은 스크립트 언어를 활용하여 인공지능 기술을 엣지 컴퓨팅 어플리케이션에서 활용하는 방식이 확산됨에 따라 스마트 팩토리를 비롯하여 다양한 분야에서 혁신이 이루어질 것으로 것으로 예상된다.

반면, 엣지 컴퓨팅 기기는 현실 세계에 위치하여 인공지능 구현 기술을 구동 가능한 수준의 향상된 연산과 네트워크 연결을 제공하므로, 인적/물적 피해를 발생할 수 있는 사이버 테러에 악용되지 않도록 어플리케이션 무결성 및 다양한 보안 측면에 대한 고려가 필수적이다.

본 논문에서는 오픈소스 컨테이너 기술에서 제공하는 무결성 보호기술인 Docker Content Trust를 기반으로 하여 엣지 컴퓨팅 환경의 파이썬과 같은 스크립트 기반 어플리케이션에 대한 무결성을 유지할 수 있는 기법을 제시하였으며, 엣지 컴퓨팅 기기에서 전자서명정보에 대한 화이트리스트를 구성하여 컨테이너 저장소를 통해 의도되지 않은 엣지 컴퓨팅 어플리케이션 배포를 차단 할 수 있도록 하였다.

본 연구에서는 DCT 환경에서 제공하는 서버 컴포넌트가 침해되어 악성 컨테이너 이미지가 저장소를 통해 엣지 컴퓨팅 기기로 확산되는 최악의 상황을 방지하는데 중점을 두고 있으나, 이러한 침해상황 발생을 사전에 방지할 수 있도록 서버 컴포넌트에 대한 적절한 접근 권한 설정 등의 활동도 병행되어야 함을 알 수 있다.

향후 연구과제로는 본 연구의 내용과 함께 엣지

컴퓨팅 환경의 보안성 향상을 위해 엣지 컴퓨팅 서버 환경 및 기기에 대한 비인가 접근 시도 및 침해 시도 등 이상 행위에 대한 탐지 기법에 대한 도출, 설계 및 개발이 있다.

References

[1] J.H. Hong, K,C. Lee, and S.Y. Lee, "Trends in edge computing technology," Electronics and Telecommunications Trends, 35(6), pp. 78-87, Dec. 2020.

[2] Hyung-Sun Kim and Hong-Chul Lee, "Development of edge cloud platform for IoT based smart factory implementation," Journal of The Korea Society of Computer and Information, 24(5), pp. 49-58, May 2019.

[3] AKM, ASHIQUZZAMAN, Dongsu Lee, Seungmin Oh, Jihoon Lee, and Jinsul Kim, "A Study on deep learning-based product data visualization and intelligent monitoring technology in smart factory environment," Journal of Digital Contents Society, 20(10), pp.1933-1942, Oct. 2019.

[4] Kwihoon Kim and Bangwon Seo "Intelligent construction video management system based on edge computing using deep learning," The Journal of Korean Institute of Information Technology, 17(7), pp. 55-63. Jul. 2019.

[5] Qureshi, Basit, Kamal Kawlaq, Anis Koubaa, Basel Saeed, and Mohammad Younis, "A commodity SBC-edge cluster for smart cities," In 2019 2nd International Conference on Computer Applications & Information Security, pp. 1-6, May 2019.

[6] Tsai, Pei-Hsuan, Hua-Jun Hong, An-Chieh Cheng, and Cheng-Hsin Hsu, "Distributed analytics in fog computing platforms using tensorflow and kubernetes," In 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS). pp. 145-150, Nov. 2017.

[7] Javed, Asad, Jérémy Robert, Keijo Heljanko, and Kary Främling, "IoTEF: AW federated edge-cloud architecture for fault-tolerant IoT applications," Journal of Grid Computing, Vol. 18, pp. 1-24, Mar. 2020.

[8] Jeong, So-Won, Yu-Rim Choi, and Il-Gu Lee. "Cyber killchain based security policy utilizing hash for internet of things," Journal of Digital Convergence, 16(9), pp. 179-185, Sep. 2018.

[9] De Benedictis, Marco, and Antonio Lioy, "Integrity verification of docker containers for a lightweight cloud environment," Future Generation Computer Systems, Vol. 97, pp. 236-246, Aug. 2019.

[10] Content trust in Docker | Docker Documentation, "Content trust in docker",https://docs.docker.com/engine/security/trust/, Nov. 2021.

[11] Container Registry | Google Cloud, "Google Container registry",https://cloud.google.com/container-registry/, Nov. 2021.

〈저자소개〉

이 창 훈 (Changhoon Lee) 정회원
2006년 8월: 건국대학교 산업공학과 학사
2006년 8월~현재: 삼성SDS 근무
2020년 3월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 시스템 보안, 데이터 보안, 엣지 컴퓨팅

신 영 주 (Youngjoo Shin) 종신회원
2006년 2월: 고려대학교 컴퓨터학과 학사
2008년 2월: KAIST 전산학과 석사
2014년 8월: KAIST 전산학과 박사
2008년 4월~2017년 2월: 국가보안기술연구소 선임연구원
2017년 3월~2020년 8월: 광운대학교 컴퓨터정보공학부 조교수
2020년 9월~현재: 고려대학교 정보보호대학원 정보보호학과 조교수
〈관심분야〉 시스템 보안, CPU 마이크로아키텍처 취약점 분석, 클라우드 보안