앙상블 기반의 크립토재킹 컨테이너 탐지 프레임워크

김리영[1], 김수민[2], 유정은[2], 이수민[2], 김성민[3]
[1]성신여자대학교 미래융합기술공학과 석사과정
[2]성신여자대학교 융합보안공학과 학부생
[3]성신여자대학교 융합보안공학과 교수
220236036@sungshin.ac.kr, 20221082@sungshin.ac.kr, jjeong031121@gmail.com,
sueyeppy@gmail.com, sm.kim@sungshin.ac.kr

Ensemble-based cryptojacking container detection framework

Ri-Yeong Kim[1], Su-Min Kim[2], Jeong-Eun Ryu[2], Soo-Min Lee[2], Seongmin Kim[3]
[1]Dept. of Future Convergence Technology Enginerring, Sungshin Women's University
[2]Dept. of Convergence Security Engineering, Sungshin Women's University
[3]Dept. of Convergence Security Engineering, Sungshin Women's University

요 약

클라우드 환경에서 컨테이너 사용이 증가하면서 컨테이너 환경을 대상으로 하는 여러 보안 위협이 증가하고 있다. 대표적인 악성 컨테이너는 크립토재킹 컨테이너로, 인스턴스 소유자의 승인 없이 리소스를 탈취하여 암호화폐를 채굴하는 공격이다. 이러한 공격은 리소스 낭비를 초래할 뿐 아니라 자원을 공유하는 정상 컨테이너나 호스트 인프라에까지도 영향을 미칠 수 있다. 따라서 본 논문에서는 크립토재킹 컨테이너를 탐지하기 위한 앙상블 기반의 크립토재킹 컨테이너 탐지 프레임워크 설계를 제안한다. 또한, 앙상블 모델 학습을 위한 데이터 수집에 있어 크립토재킹 컨테이너의 동적 특징을 나타내는 시스템 콜 및 네트워크 플로우 기반의 특성 활용 가능성을 사례 연구를 통해 분석하였다.

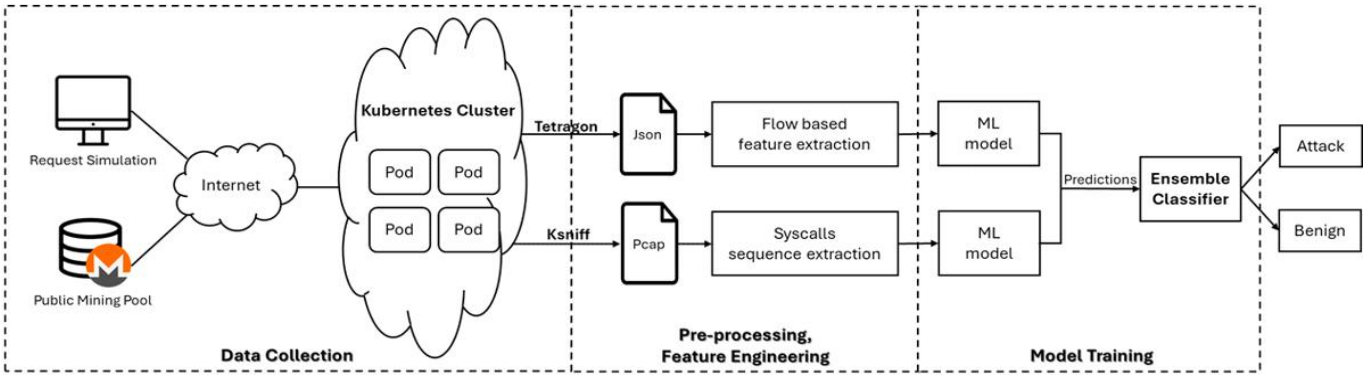
1. 서론

컨테이너는 호스트 커널과 공유하며 격리된 실행 환경을 제공하는 운영체제 수준의 가상화 기술이다. 뛰어난 이식성 및 확장성을 가진 컨테이너 기술은 클라우드 환경에 적합한 애플리케이션 배포 환경을 제공하여 지속적으로 사용량이 증가하고 있다. 대표적인 컨테이너 기반 도구로는 도커(Docker)와 쿠버네티스(Kubernetes)가 있다. 도커는 컨테이너화 기술을 기반으로 애플리케이션을 패키징하여 이미지 형태로 배포한다. 쿠버네티스는 배포된 다수의 컨테이너를 통합 관리하는 도구로, 컨테이너화된 애플리케이션의 자동화된 배포, 확장 및 관리 기능을 제공한다. 현재 대다수의 기업들은 도커 컨테이너와 이를 관리하기 위한 쿠버네티스를 채택하고 있으며, 보편적으로 두 기술을 기반으로 클라우드 네이티브 환경을 구축하고 있다.

하지만 컨테이너의 사용량이 증가하면서 컨테이너 환경을 대상으로 하는 여러 보안 위협이 증가하고 있다. 그 중, 대표적인 보안 위협으로는 악성 컨테이너를 기반으로 한 크립토재킹(Cryptojacking) 공격이 있다. 크립토재킹이란 암호화폐를 채굴하기 위해 피해자의 리소스를 승인 없이 사용하는 공격이다. 이러한 공격은 특히 대규모 컴퓨팅 리소스를 제공하는 클라우드 환경에서 더 큰 피해를 야기할 수 있다. Google Cloud의 조사에 따르면 컨테이너 환경에서 발생한 공격의 86%가 크립토재킹 컨테이너로 인한 것으로 확인되었으며[1], 2022년 Sysdig가 발표한 Cloud Native Security and Usage Report에 따르면 가장 많이 발견된 악성 컨테이너 이미지는 크립토재킹 이미지로, 두 번째로 높은 악성 컨테이너 이미지와 2배 이상의 차이로 높은 비율을 보였다[2].

크립토재킹을 통해 공격자는 컨테이너 리소스를 탈취하여 남용함으로써 정상 컨테이너의 성능을 저하시킬 수 있으며, 컨테이너는 호스트와 커널을 공유하기 때문에 컨테이너에서 발생한 공격이 호스트까지도 영향을 미칠 수 있다. 또한, 사용자들은 이미지 레지스트리인 도커 허브(Docker Hub)를 통해 각종 도커 이미지를 저장 및 공유한다. 하지만 도커 허브에는 누구나 이미지를 업로드할 수 있기 때문에 공격자들은 암호화폐 채굴을 위한 악성 컨테이너를 배포하는 수단으로 악용한다.

크립토재킹을 탐지하기 위한 선행 연구는 주로 브라



(그림 1) 앙상블 기반의 크립토재킹 컨테이너 탐지 프레임워크

우저 기반의 공격과 PC 환경에서 발생하는 악성 바이너리 형태의 호스트 기반 크립토재킹 탐지에만 중점을 두고 있다. 따라서 컨테이너 환경에서의 악성 크립토재킹 이미지 공격 탐지에 대한 연구가 부족한 실정이다. 따라서, 본 연구에서는 컨테이너 환경 내에서 크립토재킹 컨테이너를 탐지하기 위해 앙상블 기반의 탐지 프레임워크를 제안한다. 이를 위해 정상 컨테이너와 크립토재킹 컨테이너를 시스템 콜 및 네트워크 플로우 관점에서 분석하여 이를 특징으로 활용하고자 한다.

2. 관련 연구

Gomes외 1인은 브라우저 기반 크립토재킹 탐지를 위해 CPU 사용량을 기반으로 한 방안을 제안하며, CPU 메트릭 기반의 여러 특징 집합을 결합하여 머신러닝 모델을 활용한다[3]. 하지만 크립토재킹 공격이 CPU를 과도하게 사용하는 특징으로 CPU 사용량을 기반으로 공격을 탐지할 경우에는 CPU를 집약적으로 사용하는 정상 컨테이너에 대한 오탐으로 이어질 수 있다.

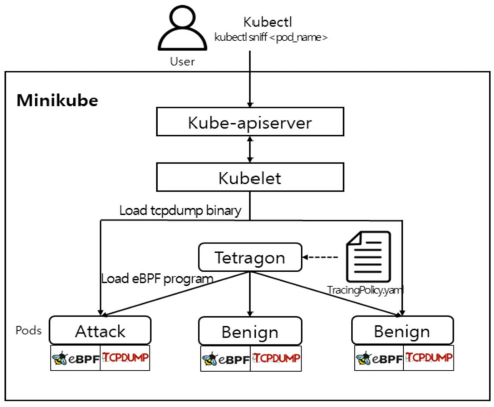
Saide 외 2인은 크립토재킹 도커 이미지를 탐지하기 위해 도커 이미지의 명령 스크립트를 데이터 셋으로 활용하는 머신러닝 모델을 구현한다[4]. 이러한 정적 분석 방법은 악성 스크립트가 난독화되거나 노출되지 않으면 탐지가 어렵다는 한계가 있다. 또한, Karn 외 4인은 컨테이너의 시스템 호출 패턴과 CPU 사용량을 기반으로 크립토재킹 컨테이너를 탐지하는 방법을 제안하였다[5].

본 연구에서는 기존 연구와 달리 크립토재킹 컨테이너의 내부 및 외부 동작을 분석하기 위해 시스템 콜과 네트워크 플로우을 함께 고려하고자 한다는 측면에서 기존 선행연구들과 차별성을 갖는다. 이 두 가지 특징을 기반으로 크립토재킹 컨테이너와 정상 컨테이너를 구별하는 앙상블 기반의 탐지 프레임워크를 제안함으로써, 탐지 성능을 향상시키고자 한다.

3. 앙상블 기반의 크립토재킹 컨테이너 탐지 프레임워크

본 장에서는 제안하는 앙상블 기반 크립토재킹 컨테이너 탐지 프레임워크의 구성 단계에 대해 기술한다. 그림 1은 제안 프레임워크의 세 단계를 보여주며, 1) 데이터 수집, 2) 전처리 및 특징 엔지니어링, 3) 모델 학습 단계로 구성된다. 이어지는 절에서는 각 구성 단계의 실행 흐름에 따라 쿠버네티스 환경에서 데이터 셋 수집을 위한 환경 구성 및 활용 도구에 대해 설명하고, 추출한 원시 데이터 셋에서 특징을 추출 및 모델 학습 방법에 대해 서술한다.

3.1 데이터 수집



(그림 2) 데이터 수집을 위한 쿠버네티스 구성도

그림 2와 같이 컨테이너 이미지를 배포하고 데이터를 수집하기 위해 쿠버네티스 클러스터를 구성한다. 쿠버네티스(Kubernetes)는 여러 컨테이너를 효율적으로 관리하기 위한 대표적인 컨테이너 오케스트레이션 도구로, 컨테이너들을 파드(Pod)라는 단위로 관리한다. 학습 데이터를 수집하기 위해 크립토재킹 컨테이너와 정상 컨테이너를 실행한다. 그림 1과 같이 크립토재킹 컨테이너는 마이닝 풀에 연결되어 채굴을 수

<표 1> 시스템 콜 호출 비율

XMRig		Nginx (streaming-server)		MariaDB		Tensorflow		Cassandra	
Syscall	Ratio (%)	Syscall	Ratio (%)	Syscall	Ratio (%)	Syscall	Ratio (%)	Syscall	Ratio (%)
epoll-wait	88	write	29	futex	56	futex	55	futex	94
mmap	3	writev	17	pwrite64	25	newfstatat	10	read	1
read	3	recvfrom	17	mprotect	8	read	9	openat	1
munmap	2	openat	17	fdatasync	4	mmap	6	newfstatat	1
writev	2	epoll-wait	14	recvfrom	3	lseek	4	close	1

행하며, 정상 컨테이너는 파드 형태로 외부로 노출시켜 외부에서 벤치마킹 툴을 실행하도록 구성한다.

제안하는 프레임워크에서는 시스템 콜과 네트워크 플로우를 기반으로 하는 두 가지 다른 데이터 셋으로부터 특징을 추출한다. Tetragon[6]은 클라우드 네이티브 환경에서 보안 관찰 및 시행을 위한 eBPF 기반 도구이다. eBPF란 리눅스 커널에서 동작하여 사용자가 정의한 코드를 커널 내에서 실행시킬 수 있는 도구로, Tetragon은 TracingPolicy yaml 파일을 통해 eBPF 기반의 추적 정책을 관리한다. 시스템 호출이 커널 내로 진입할 때 발생하는 sys_enter 이벤트 지점을 후킹하여 대상 파드에서 호출하는 모든 시스템 콜을 추적할 수 있다. 네트워크 패킷을 추출하기 위해 ksniff[7]를 활용한다. ksniff란 쿠버네티스 클러스터 내 파드에 대한 패킷 캡처 도구로, tcpdump 및 wireshark를 활용하는 kubectl 플러그인이다.

3.2 전처리 및 특징 엔지니어링

원시 Pcap 파일에서 CICFlowMeter 도구[8]를 사용하여 네트워크 패킷에서 플로우를 추출한다. 각 플로우에는 라벨이 포함된 84개의 특징이 포함된 csv 파일이 생성된다. 각 특징별 중요도를 파악하여 적합한 특징을 추출한다. Tetragon으로 시스템 콜을 추출하면 json 형식의 로그 파일이 생성된다. 로그 파일에서 시스템 콜 시퀀스를 long_arg 파라미터에 해당하는 값을 추출하는 파이썬 코트를 통해 {281, 281, 202, 41, 54 ...} 와 같은 시스템 콜 시퀀스를 추출할 수 있다. 연속적인 시퀀스 특징을 고려하기 위해 n-gram 기법을 활용한다. n-gram이란 연속된 n개의 시스템 콜을 하나의 패턴으로 간주하여 모델 학습에 시스템 콜의 구조적인 특징 정보를 제공한다. 다양한 n값에 대한 모델 평가를 통해 적절한 n값을 선택하는 것이 중요하다. 두 가지 데이터 셋을 기반으로 각각의 모델을 학습한 후, 앙상블 모델을 적용하여 최종 탐지 결과를 획득한다.

4. 사전 평가

본 장에서는 쿠버네티스 환경에 크립토재킹 컨테이너와 정상 컨테이너를 구동하여 추출한 시스템 콜 목록을 비교 및 분석한다.

4.1 실험 환경

Minikube[9]를 사용하여 Ubuntu 20.04 환경에 단일 노드 쿠버네티스 클러스터를 구축하였다. 크립토재킹 컨테이너 이미지로는 모네로(Monero)를 채굴하는 XMRig[10]를 사용하였다. 모네로는 강력하게 익명성을 보장하면서 다른 암호화폐와 달리 특수 장비 없이 CPU와 같은 일반적인 컴퓨팅 리소스로 채굴이 가능하기 때문에 크립토재킹 공격에 많이 사용된다. 정상 컨테이너는 설계된 목적에 따라 작업을 수행하며 암호화폐를 채굴하지 않는 컨테이너를 말한다. 정상 컨테이너로는 오픈 소스 관계형 데이터베이스 관리 시스템인 MariaDB[11], 분산형 NoSQL 데이터베이스 시스템인 Cassandra[12], 머신러닝 소프트웨어인 Tensorflow[13] 및 Nginx 웹 서버 기반의 미디어 스트리밍 서버 이미지[14]를 파드로 구동하였다. 시뮬레이션을 실행하기 위해 각 서버에 맞는 부하 벤치마킹 툴과 Tensorflow 파드 내부에서는 모델 학습 워크로드를 실행하였다. Tetragon을 활용하여 파드에서 호출하는 시스템 콜 로그를 추출하였다.

4.2 컨테이너 별 시스템 콜 분석

표 1은 각 파드에서 추출된 시스템 콜에서 10,000개의 시퀀스에 대한 상위 5개의 시스템 콜 호출 비율을 보여준다. 채굴 프로그램은 채굴 데이터를 메모리에 로드하고, 해시 계산을 수행한다. 또한 마이닝 풀과의 지속적인 통신을 통해 새로운 블록을 받고, 수행한 작업을 제출한다. 표 1과 같이 XMRig는 네트워트 통신과 관련된 시스템 콜을 가장 많이 호출했다. epoll_wait 시스템 콜은 마이닝 풀과의 연결

을 유지하고, 새로운 작업을 수신하는 등 마이닝 풀과의 통신을 효율적으로 관리하기 위해 사용된다. mmap 및 munmap 시스템 콜은 채굴 작업에 필요한 데이터를 메모리에 로드하고 접근하며, 작업이 끝나면 메모리를 해제하기 위해 호출된다. 또한 디스크 및 소켓에 데이터를 읽고 쓰는 작업을 수행하기 위해 파일 관련 시스템 콜인 read 및 writev가 호출됨을 확인 할 수 있다. 정상 컨테이너 중 Nginx 기반의 스트리밍 서버는 주로 네트워크 통신 관련 시스템 콜이 많이 관찰되었다. Tensorflow, MariaDB, Cassandra의 경우에는 프로세스 제어 및 파일 관련 시스템 콜이 주로 호출되었다.

결론적으로 정상 컨테이너와 달리 크립토재킹 컨테이너는 채굴과 관련된 네트워크 통신 및 메모리 관리와 관련된 시스템 콜을 더 자주 호출하는 경향을 보였다. 따라서 이러한 시스템 콜 호출 패턴을 학습 모델의 특징으로 활용할 수 있다.

시스템 콜은 컨테이너의 내부 동작을 이해하는데 중요한 정보를 제공하지만 추가적으로 네트워크 트래픽을 분석함으로써 컨테이너가 외부와 상호 작용하는 행위를 분석할 수 있다. 사전 평가를 통해 크립토재킹 컨테이너가 마이닝 풀과 통신하며 작업을 송수신하는 행위를 시스템 콜을 통해 확인할 수 있었다. 이러한 행위는 네트워크 플로우 관점에서 채굴 활동을 나타내는 특정한 통신 패턴으로도 나타날 수 있다. Muñoz 외 3인은 다양한 코인에 대한 Stratum 트래픽이 송수신별 초당 비트 수, 초당 패킷 수, 패킷 당 비트 수 측면에서 정상 트래픽과의 차이가 있음을 보였다[15]. Gomes 외 3인은 마이닝 서버에서 채굴자 간의 통신에 사용되는 Stratum 프로토콜 특성 기반의 특징 집합을 활용했다[16]. 따라서 시스템 콜 패턴과 네트워크 플로우 기반의 특징을 결합하면 컨테이너의 동작을 더 잘 반영하기 때문에 크립토재킹 컨테이너를 탐지하는데 성능 향상을 기대할 수 있다.

5. 결론

본 논문에서는 크립토재킹 컨테이너를 탐지하기 위해 시스템 콜과 네트워크 플로우를 활용한 앙상블 기반의 탐지 모델을 제안하였다. 사전 평가를 통해 크립토재킹 컨테이너는 지속적으로 네트워크 통신이 이루어 지고 있음을 확인할 수 있었다. 추후에는 컨테이너의 네트워크 플로우 특징을 분석하고, 시스템 콜 패턴과 결합한 앙상블 기반의 탐지 모델을 구현하고 평가하고자 한다.

Acknowledgement

본 논문은 2024년도 산업통상자원부 및 한국산업기술진흥원의 산업혁신인재성장지원사업 (RS-2024-00415520)과 과학기술정보통신부 및 정보통신기획평가원의 ICT혁신인재4.0 사업의 연구결과로 수행되었음 (No. IITP-2022-RS-2022-00156310)

참고문헌

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

[2] Sysdig, Cloud Native Security and Usage Report, https://sysdig.com/2022-cloud-native-security-and-usage-report/, 2022

[3] Gomes, Fábio, and Miguel Correia, "Cryptojacking detection with cpu usage metrics.", 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), 1-10, 2020.

[4] Saide, Saide Manuel, Ednilson Luis Alfredo Sarmento, and Felermino DMA Ali, "Cryptojacking Malware Detection in Docker Images Using Supervised Machine Learning.", International Conference on Intelligent and Innovative Computing Applications, 49-56, 2022.

[5] Karn, Rupesh Raj, et al, "Cryptomining detection in container clouds using system calls and explainable machine learning.", IEEE transactions on parallel and distributed systems, 32(3), 674-691, 2020.

[6] https://tetragon.io/

[7] https://github.com/eldadru/ksniff

[8] https://github.com/ahlashkari/CICFlowMeter

[9] https://minikube.sigs.k8s.io/docs/start/

[10] https://hub.docker.com/r/miningcontainers/xmrig

[11] https://hub.docker.com/_/mariadb

[12] https://github.com/parsa-epfl/cloudsuite/blob/main/docs/benchmarks/data-serving.md

[13] https://hub.docker.com/r/tensorflow/tensorflow

[14] https://github.com/parsa-epfl/cloudsuite/blob/main/docs/benchmarks/media-streaming.md

[15] i Muñoz, Jordi Zayuelas, José Suárez-Varela, and Pere Barlet-Ros, "Detecting cryptocurrency miners with NetFlow/IPFIX network measurements.", 2019 IEEE International Symposium on Measurements & Networking (M&N), 1-6, 2019.

[16] Gomes, Gilberto, Luis Dias, and Miguel Correia, "Cryingjackpot: Network flows and performance counters against cryptojacking.", 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), 1-10, 2020.