FaaSMon: 프로비넌스 GNN 기반 서버리스 침입 탐지 시스템*

정지환[1], 양정용[1], 이혜진[1], 김진우[2,†]

[1,2]광운대학교 (학부생, 교수)

FaaSMon: A Provenance GNN-Based System for Serverless Intrusion Detection*

Ji-Hwan Jung[1], Jeong-Yong Yang[1], Hye-Jin Lee[1], Jin-Woo Kim[2,†]

[1,2]Kwangwoon University(Undergraduate Student, Professor)

요 약

서버리스 컴퓨팅은 각 기능을 함수 컨테이너로 만들고 API를 통해 이를 실행하는 클라우드 네이티브 개발 및 실행 모델이다. 그러나 함수 컨테이너를 생성할 때 발생하는 오버헤드를 줄이기 위해 웜 컨테이너를 사용하면서 '컨테이너 재사용 공격'이 서버리스 환경의 큰 위험 요소가 되었다. 해당 공격은 복잡한 함수 체인을 활용하기 때문에 공격을 탐지하는 데 어려움이 있다. 본 논문에서는 기존에 제안된 서버리스 모니터링 솔루션을 분석하고, 사이드카 패턴을 활용해 기존 연구의 한계점을 개선한 FaaSMon을 소개하고자 한다.

I. 서론

서버리스 컴퓨팅은 클라우드 네이티브 개발 및 실행 모델로, Function-as-a-Service (FaaS)라고도 한다. 필요한 서비스 및 기능을 코드로 정의하고, 각 함수(function)를 사용하기 위한 API를 제공한다. 개발자는 서버 및 백엔드 인프라를 구축하고 관리할 필요가 없고, API를 사용한 만큼 이용료를 지불함으로써 인프라 구축 비용 절감 효과를 누릴 수 있다.

한편으로 서버리스 환경에서는 함수 실행 시 비용 절감을 위해 기존에 생성된 웜 컨테이너(warm container)를 사용한다. 그러나 웜 컨테이너의 남아있는 크리덴셜 등을 악용하는, 이른바 '컨테이너 재사용 공격'이 보고되었다[1, 2]. 공격자는 웜 컨테이너를 통해 다른 컨테이너로 이동하면서 공격하는 서버리스 침입(serverless intrusion)이 가능하게 된다. 이들은 APT 공격과 같이 복잡하게 구성된 함수 체인을 활용한 공격이라 탐지하는 데 큰 어려움이 있다.

이를 해결하기 위해 여러 서버리스 모니터링 솔루션[3, 4, 5]이 제안되었다. 이들은 서버리스 환경을 모니터링하여 수집된 시스템 콜이나 네트워크 트래픽으로 프로비넌스 그래프(provenance graph)를 구축하여 공격자의 행위를 분석하는 기법이다. 클라우드 관리자는 해당 그래프를 통해 여러 함수 컨테이너를 넘나드는 공격이 어디서 시작되었고 어느 컨테이너를 탐색해야 하는지 확인할 수 있다.

그러나 이들은 글로벌 프로비넌스 그래프를 생성하기 때문에 실시간으로 그래프 변화를 추적하기 어렵고, 컨테이너 재사용 공격을 통한 단순한 침투 공격 시나리오에 한정된 탐지 시스템이다. 또한, OpenFaaS의 watchdog 모듈을 직접 수정하여 구현했기 때문에 해당 시스템을 실제 서버리스 환경에 배포하기 어렵다. 마지막으로 복잡한 글로벌 프로비넌스 그래프를 사용자가 직접 눈으로 보고 공격을 파악해야 하는 불편함이 존재한다.

본 논문에서는 기존 서버리스 모니터링 솔루션의 한계점을 개선한 시스템인 FaaSMon을 제

* 이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. RS-2024-00457937)

† 교신저자(jinwookim@kw.ac.kr)

안한다. FaasMon은 사이드카 패턴(sidecar pattern)을 활용해 OpenFaaS와 기존 서버리스 인프라 코드 수정을 최소화하고, 각 컨테이너의 시스템 콜 및 네트워크 트래픽을 수집해 프로비넌스 그래프를 구축한다. 이후 GNN(Graph Neural Network)을 이용해 프로비넌스 그래프들을 학습한 분류기를 통해 서버리스 침입 여부를 탐지한다.

II. 배경지식 및 관련연구

서버리스는 함수 실행 및 종료 시 해당 함수 컨테이너를 할당하거나 회수한다. 이 과정에서 컨테이너를 활성화 하기 위해 시간이 소요되기 때문에 오버헤드가 발생한다. 해당 오버헤드를 줄이기 위해 컨테이너를 종료하지 않고 다시 해당 함수 호출에 할당하는 경우가 많아졌다. 해당 취약점을 이용한 공격인 '컨테이너 재사용 공격'은 함수 컨테이너에 악성 코드를 주입하고 다시 호출해 주입한 악성 코드를 실행하는 공격이다. 이는 여러 컨테이너가 실행되는 함수 흐름 도중에 이루어지기 때문에 기존의 단순 보안 솔루션으로는 방어하기 매우 어렵다. 따라서 이러한 공격을 탐지하기 위한 여러 연구가 제안되었다.

ALASTOR [1]: ALASTOR는 함수 인스턴스 간의 인과 관계를 바탕으로 프로비넌스 그래프를 생성한다. 각 함수 인스턴스에 상주하며 시스템 및 애플리케이션 수준의 활동을 기반으로 로컬 프로비넌스 그래프와 글로벌 프로비넌스 그래프를 생성한다.

WILL.IAM [2]: WILL.IAM 역할과 루틴에 관한 권한을 제어하는 각 접근 정책을 함수 컨테이너가 아닌 워크플로에 할당해 잘못된 IAM으로 발생할 수 있는 불필요한 자원 사용과 접근을 막는다.

CLARION [3]: CLARION은 컨테이너 내부에서 발생하는 프로세스 생성 및 파일 조작과 같은 이벤트를 추적해 프로세스를 기준으로 컨테이너 내부에서 발생하는 상호작용을 바탕으로 프로비넌스 그래프를 생성한다.

III. FaaSMon

3.1 시스템 구조와 동작 흐름

FaaSMon은 기존 연구의 한계점인 인프라 코드 및 함수 컨테이너 설정을 수정하지 않으면서 효과적인 모니터링을 위해 설계되었다. FaaSMon은 패킷 캡처 및 각 함수 컨테이너 내부 시스템 콜 추적을 통해 로그를 생성하고, 로그를 기반으로 프로비넌스 그래프를 구축한다.

Fig. 1은 FaaSMon을 연결한 OpenFaaS 플랫폼에서 요청이 들어왔을 때 각 컨테이너가 수행하는 역할을 보여준다. 먼저, 기존에 OpenFaaS Gateway로 들어가던 요청이 FaaSMon 내부 프록시를 경유한다(초록 화살표, ①). 이 과정에서 내부의 tcpdump 컨테이너가 해당 패킷을 분석해 pcap 파일로 저장한다(②). 이후 Gateway를 통과한 요청을 통해 기존에 사용자가 설계한 함수 흐름이 동작할 때, FaaSMon 내부의 strace 컨테이너를 통해 각 함수 컨테이너 내부의 시스템 콜을 수집해 로그를 생성한다(③, ④).

수집한 두 종류의 로그를 파싱하고 이를 바탕으로 프로비넌스 그래프를 생성해 FaaSMon 컨테이너 내부에 저장한다(⑤). 이후 저장된 그래프는 GNN을 사용한 그래프 분류기를 통과하며, 분류기는 사용자에게 그래프에 서버리스 침입이 존재하는지 여부를 알려준다(⑥, ⑦). 내부 프록시는 NGINX를 사용하고, 로그 생성, 파싱, 그래프 생성은 Python으로 작성되었으며, FaaSMon의 모든 컨테이너는 Docker를 사용해 관리된다.

3.2 로그 및 그래프 생성

각 함수 컨테이너는 내부의 핸들러인 index.js를 실행하는 것으로 동작한다. 이때 먼저, pgrep을 사용해 실행된 index.js를 검색한다. index.js 프로세스를 찾은 경우, 해당 프로세스의 PID를 가져온다. 가져온 PID를 바탕으로 컨테이너의 이름을 검색해 로그와 그래프에 사용한다. 또한 PID를 사용해 strace를 통한 함수 컨테이너 내부에서 execve, read, clone 등의 파일 및 스레드 제어 시스템 콜과 recvfrom, sendto 등의 통신을 위한 시스템 콜을 위주로 수집한다. 수집한 시스템

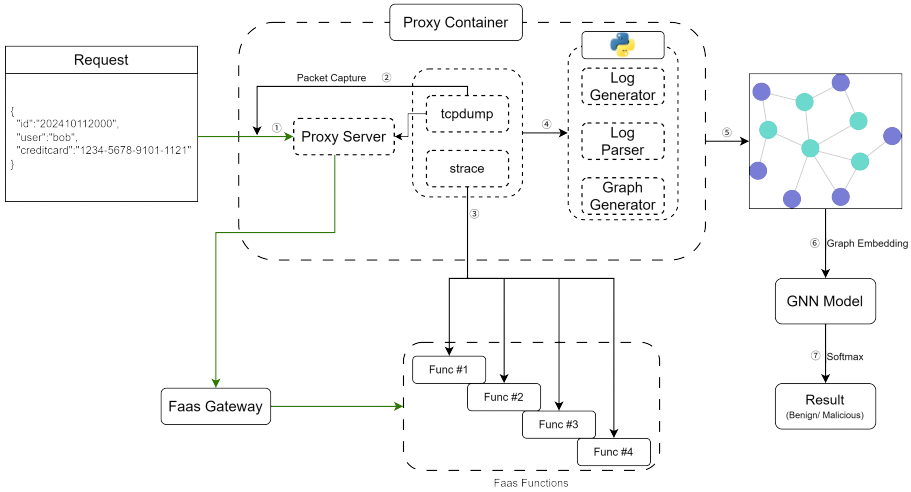

Fig. 1 FaaSMon system overview.

콜을 바탕으로 Python을 사용해 로그를 파싱하고, 그래프를 생성한다.

IV. 평가

수집할 그래프는 소매업 관련 데이터 솔루션의 예시 프로젝트인 hello-retail에서 수집한다. 공격 상황은 함수 호출 시 Attack Server에서 악성 파일을 받고 함수 컨테이너에서 실행한다. 그래프를 수집하는 단계에서는 동일한 환경에서 그래프를 구축하기 위해서 호스트에서 횟수를 지정하여 FaaSMon 및 모든 함수 컨테이너를 매번 새로 빌드하며 데이터를 수집했다. 학습을 위한 그래프는 함수 호출에 대한 정상 그래프 187개, 2가지 공격 그래프 각각 200개씩 데이터를 불러와 PyTorch Geometric을 사용해 GNN 모델을 학습시켰다. 정상 그래프, 비정상 그래프를 80:20 비율로 트레이닝 셋과 테스트 셋을 나누어 훈련시켰다. 실험 결과, F1-score는 0.9816, 테스트 정확도는 96.97%로 나타났다. Fig. 2의 Confusion matrix와 같이 정상 그래프와 비정상 그래프를 높은 정확도로 판별이 가능한 것을 볼 수 있다.

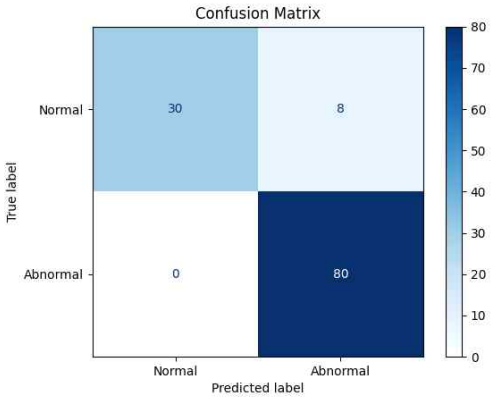

Fig. 2 GNN confusion matrix

V. 결론

본 논문에서는 사이드카 패턴을 활용해 컨테이너를 배치하여 서버리스 환경 모니터링과 프로비넌스 그래프 생성을 수행하고, 이를 학습한 GNN을 통해 서버리스 침입을 탐지하는 시스템인 FaasMon을 소개했다. 실험 결과 높은 정확도로 서버리스 침입 공격을 탐지하는 것을 보였다.

[참고문헌]

[1] Datta, Pubali, et al. "ALASTOR: Reconstructing the provenance of serverless intrusions." *31st USENIX Security Symposium (USENIX Security 22)*. 2022.

[2] Sankaran, Arnav, Pubali Datta, and Adam Bates. "Workflow integration alleviates identity and access management in serverless computing." *Proceedings of the 36th Annual Computer Security Applications Conference*. 2020.

[3] Chen, Xutong, et al. "CLARION: Sound and clear provenance tracking for microservice deployments." *30th USENIX Security Symposium (USENIX Security 21)*. 2021.