마이크로서비스 아키텍처에서 효율적인 이상 징후 탐지를 위한 공격 진원지 추적형 컨테이너 스냅샷 기법 설계

Design of a Attack Provenance Trackable Container Snapshot Scheme for Efficient Anomaly Detection in Microservices Architecture

이병용, 박기웅[1)]
Byeong-Yong Yi, Ki-Woong Park

(05006) 서울특별시 광진구 능동로 209 세종대학교 정보보호학과 시스템보안 연구실, 정보보호학과
yibyeongyong@sju.ac.kr woongbak@sejong.ac.kr

요 약

마이크로서비스 아키텍처는 소프트웨어 개발에서 주로 사용하는 분산 소프트웨어 아키텍처이다. 그러나, 마이크로서비스 간 통신을 위해 추가하는 요소들로 인해 1) 구조적 복잡성이 증가하는 이슈와 2) 공격 표면이 증가하는 이슈가 발생한다. 구조적 복잡성 증가 이슈를 해결하기 위해 분산 추적 기법이 제안되었으며, 최근엔 이러한 분산 추적 기법을 응용하여 보안성 향상 목적의 관련 연구가 수행되고 있다. 그러나, 분산 추적 기술은 본래 서비스 성능 관리 및 디버깅 수행이 목적이므로, 보안성 향상 목적으로 활용할 수 있는 범위가 제한적이다. 본 논문은 이러한 문제를 해결하기 위해 진원지 추적형 컨테이너 스냅샷 기법을 제안한다. 진원지 추적형 컨테이너 스냅샷 기법은 분산 추적 기법의 인과 관계 정보를 활용하여, 사이버 위협이 발생한 관련 마이크로서비스만을 선별적으로 컨테이너 스냅샷을 수행한다. 이를 통해, 컨테이너 스냅샷 과정을 효율적으로 수행할 수 있다. 제안하는 방안의 실용성을 검증하기 위해, 임의의 마이크로서비스 환경을 구성하여 스냅샷 과정에서 발생하는 오버헤드 및 지연 시간을 측정하였다. 실험 결과, 컨테이너 스냅샷 과정에서 0.7%의 메모리 오버헤드가 발생했으며, 평균 172ms를 필요로 하였다. 본 논문에서 제안하는 진원지 추적형 컨테이너 스냅샷 기법은 마이크로서비스 환경에서 발생한 이상 징후 탐지에 다양하게 활용될 수 있으며, 이를 통해 마이크로서비스 환경의 보안성을 향상할 수 있다.

Abstract

Microservices architecture is a distributed software architecture mainly used in software development. However, due to the elements added for communication between microservices, 1) an issue of increasing structural complexity and 2) an issue of increasing attack surface arises. A distributed tracing scheme has been proposed to solve the issue of increasing structural complexity,

1) 교신저자

논문투고일자 : 2021. 04. 08 심사일자 : 2021. 04. 08 게재확정일자 : 2021. 04. 23

and recently, related research for the purpose of improving security has been conducted by applying this distributed tracing scheme. However, since the distributed tracing scheme is originally intended to manage service performance and perform debugging, the scope that can be used for the purpose of improving security is limited. To solve this problem, this paper proposes a container snapshot method for tracking the attack provenance. The attack provenance trackable container snapshot scheme utilizes the causal relationship information of the distributed tracing scheme to selectively perform container snapshots of only related microservices in which cyber threats have occurred. Through this, it is possible to efficiently perform the container snapshot process. To verify the practicality of the proposed scheme, an arbitrary microservices environment was configured to measure the overhead and delay time incurred during the snapshot process. As a result of the experiment, a memory overhead of 0.7% occurred during the container snapshot process, and an average of 172ms was required. The attack provenance trackable container snapshot scheme proposed in this paper can be used in various ways to detect abnormal signs occurring in microservices environments, and through this, the security of the microservices environment can be improved.

키워드: 마이크로서비스, 분산 추적, 공격 지원지 추적, 컨테이너 스냅샷
Keyword: Microservices, Distributed Tracing, Attack Provenance Tracking, Container Snapshot

1. 서론

마이크로서비스 아키텍처는 최근 소프트웨어 개발에 있어 대중적으로 선택되는 분산 소프트웨어 아키텍처이다[1]. 이러한 선택에는 컨테이너와 같은 가상화 기술의 발전과 신속한 소프트웨어 개발 및 배포를 수행하고자 하는 DevOps[2]가 영향을 미쳤다. 마이크로서비스 아키텍처를 사용하는 상용 서비스 분야는 퍼블릭 클라우드, 웹 검색 및 소셜 네트워크, 빅 데이터 분석, 대규모 기계학습, 마이크로서비스 및 서버리스 등 매우 광범위하다[3].

마이크로서비스 아키텍처를 구성하는 서비스들은 컨테이너 기술을 기반으로 하고 있으므로, 개발 및 배포와 서비스 확장에 특화된 장점을 갖고 있다. 단일 호스트로 동작하는 단일 구조 소프트웨어와 비교했을 때, 빌드 시간 및 테스트 시간이 짧으며, 개발 언어에 종속적이지 않다. 또한, 새로운 기능 및 서비스의 추가나 변경이 간편하며, 프로젝트의 규모 확장 과정에서도 단일 구조 소프트웨어는 전체 프로젝트를 복사해야 하는 번거로움이 있으나, 마이크로서비스는 규모를 확장하고자 하는 특정 서비스만을 선택적으로 확장하는 것이 가능하다.

그러나 마이크로서비스 아키텍처가 장점만을 가지고 있는 것은 아니다. 마이크로서비스 아키텍처는 1) 구조적 복잡성 증가와 2) 공격 표면이 증가하는 이슈를 갖고 있다. 먼저 마이크로서비스 아키텍처에서 구조적 복잡성은 관리 대상의 증가 및 서비스 간 의존성 증가로 발생한다. 마이크로서비스 아키텍처는 기본적으로 가상화 기술이 기반이므로 이와 관련된 컨테이너 기술을 필요로 하다. 또한, 하나의 기능을 수행하기 위해선 네트워크에 걸쳐 분포한 마이크로서비스 요소 간 통신을 주고는 과정이 필요한데, 이 과정에서 메시지 브로커(Message Broker) 등의 외부 서비스가 필요하다. 결국, 마이크로서비스 아키텍처의 사용하는 분산 소프트웨어는 일반적으로 구조가 복잡해진다. 마이크로서비스 개념을 정의한 Martin Fowler[4]는 동시에 마이크로서비스 아키텍처의 구조적 복잡성 문제를 지적하였다[5].

마이크로서비스 아키텍처의 공격 표면 증가 이슈는 마이크로서비스의 구성 요소 간 통신을 위해 별도의 API 및 포트를 사용하기 때문에 발생한다[6]. 이는 보안 업무 담당 인력에 관리 대상을 증가시켜 결국 보안 업무 효율성을 떨어뜨리고, 전체 시스템의 취약성을 증가시킬 수 있다[7].

분산 추적 기법은 마이크로서비스 아키텍처에서 서비스 성능 관리 및 디버깅을 수행하기 위해 제안되었다. 분산 추적 기술은 서비스 간 정확한 인과관계를 제공하므로 서비스 간 발생한 일련의 통신 흐름을 한눈에 확인할 수 있으며, 문제의 근본 원인을 파악할 수 있다. 이처럼 분산 추적 기법은 본래 성능 관리 및 디버깅을 목적으로 제안되었으나, 최근에는 분산 추적 기법이 제공하는 서비스 간 인과관계 정보, 서비스별 가용성 정보 및 지연 시간 정보를 보안에도 활용하는 관련 연구가 진행되었다[8]. 그러나, 분산 추적 플랫폼은 본래 서비스 성능 관리 및 디버깅을 위해 제안되었으므로 보안 분야에서 활용 가능한 정보가 서비스 성능 관리에 관련된 정보로 한정되어, 보안 목적으로 활용 가능한 정보가 제한적이다.

따라서 본 논문은 분산 추적 플랫폼을 보안 분야에 적극적으로 활용 할 수 있도록 공격 진원지 추적형 컨테이너 스냅샷 기법을 제안한다. 제안하는 방안은 보안 목적으로 활용 가능한 공격 진원지 추적형 컨테이너 스냅샷 기법을 기존의 분산 추적 플랫폼에 적용한다. 공격 진원지 추적형 컨테이너 스냅샷은 분산 추적 기법이 제공하는 인과 관계 정보를 활용하며, 운용 중인 마이크로서비스 아키텍처에서 사이버 위협이 발생하면, 해당 통신과 연관된 서비스만을 대상으로 컨테이너 스냅샷을 수행한다. 공격 진원지 추적형 컨테이너 스냅샷을 통해 확보한 컨테이너 스냅샷 데이터는 마이크로서비스 아키텍처의 보안성 향상에 활용될 수 있다. 또한, 사이버 위협이 발생한 서비스와 인과 관계가 있는 서비스만을 대상으로 컨테이너 스냅샷을 수행하므로 효율적으로 작동하며, 담당 보안 인력의 수고를 덜어 보안 업무의 효율성을 높일 수 있다.

본 논문은 다음과 같이 구성되어 있다. 2장은 본 논문의 연구 범위인 마이크로서비스 아키텍처에 대해서 살펴보고, 이와 관련된 사이버 위협의 종류에 대해서 살펴봄으로써, 마이크로서비스의 보안 취약점을 확인한다. 3장에서는 본 논문에서 제안하는 공격 진원지 추적형 컨테이너 스냅샷 기법의 구성 요소 및 공격 진원지 추적형 컨테이너 스냅샷 기법이 분산 추적 플랫폼 적용되었을 때, 작동 과정 살펴본다. 4장에서는 분산 추적 및 분산 추적 기법의 보안 분야 적용에 대한 관련 연구에 관해 기술한다. 마지막으로 5장에서는 본 논문의 결론 및 추후 연구를 기술한다.

2. 배경지식

2.1 마이크로서비스 아키텍처

마이크로서비스 아키텍처는 시스템 내부에서 사용하는 소프트웨어 기능을 하나의 서비스로 개념화한다. 또한, 서비스 간 공통의 통신언어를 사용하는 서비스 인터페이스를 사용하여 서비스 간 통신을 수행하는 특징을 갖고 있다. 이러한 특징은 마이크로서비스 아키텍처가 단일 소프트웨어 아키텍처보다 신속한 개발 및 배포, 편리한 성능 확장, 상대적으로 뛰어난 가용성 유지를 가능케 한다. 따라서, 웹 검색, 미디어 서비스, 소셜 네트워크, 빅 데이터 분석, 기계학습 등 다양한 분야에서 마이크로서비스 아키텍처를 활용하고 있다.

하지만, 마이크로서비스 아키텍처는 서비스 간 통신을 수행하고, 구조적 유연성을 갖추기 위해 추가 요소가 필요하다. 예를 들어, 마이크로서비스 아키텍처는 단일 소프트웨어와 달리, 서비스들을 구동하기 위해 컨테이너와 같은 가상화 기술을 필요로 한다. 또한, 수많은 마이크로서비스 개체들을 효율적으로 운영하기 위해 Kubernetes[9] 같은 마이크로서비스 관리 플랫폼 또한 필요하다. 이에 더해, 마이크로서비스 간 통신을 위한 원격 프로시저 호출(Remote Procedure Call, RPC) 등의 외부 요소가 필요하며[10], 안정적인 서비스 운영을 위

〈표 1〉 마이크로서비스 환경 대상 사이버 위협 분류[16]

<table>
<tr><th colspan="4">Issues During the Services Communication</th><th>Other Issues</th></tr>
<tr><td>Container Issues</td><td>Data Issues</td><td>Permission Issues</td><td>Network Issues</td><td rowspan="2">SQL Injection</td></tr>
<tr><td>Kernel Exploit</td><td>Data Intercept</td><td>Identity Spoofing</td><td>SDN Issues</td></tr>
<tr><td>Malicious Process</td><td rowspan="2">Secret Leak</td><td rowspan="2">Access Illegal</td><td rowspan="2">DOS Attack</td><td></td></tr>
<tr><td>Escapes from a Container</td><td rowspan="3">OWASP TOP 10</td></tr>
<tr><td>Poisoned Images</td><td>Database Destroy</td><td>Password Leak</td><td>MITM Attack</td></tr>
<tr><td>Network Sharing</td><td>Data Tamper</td><td>Replay Attack</td><td>ARP Spoofing</td></tr>
</table>

해서 사용하는 로드 밸런싱, 오토 스케일링과 같은 기술을 사용하기 위해서도 외부 요소가 필요하다. 이러한 추가 요소들로 인해 마이크로서비스 아키텍처는 구조적으로 복잡해지는 경향이 있으며, 복잡한 구조는 서비스 성능 관리 및 디버깅 방해한다. 단일 소프트웨어 환경에서 D-trace[11]와 같은 도구는 성능 관리 및 디버깅에 유용한 정보를 제공하였다. 그러나, 마이크로서비스 아키텍처 같은 분산 소프트웨어 환경에선 서비스 요청 간 직관적인 인과 관계를 제공하지 못한다[12]. 따라서, 서비스 간 요청 로그가 네트워크에 걸쳐 여러 호스트에 분산되어있더라도, 인과 관계를 제공할 수 있는 분산 추적 기술이 제안되었다.

2.2 마이크로서비스 대상 사이버 위협

마이크로서비스 아키텍처는 원활한 동작을 위해 추가적인 요소가 더해지는 과정에서 전체적인 공격 표면이 증가한다. 마이크로서비스 아키텍처는 단일 소프트웨어 아키텍처와 비교하면 전체적인 관점에서 외부와 통신 가능한 엔드 포인트가 증가하고, 마이크로서비스 간의 통신을 위해 외부 API 사용, 포트 개방 등을 수행하므로 결국 사이버 위협에 대해서 취약해진다. 본 절에선 마이크로서비스 아키텍처를 대상으로 수행되는 사이버 위협을 살펴봄으로써 마이크로서비스 아키텍처에서 보안성 향상의 필요성을 확인한다.

2.2.1 마이크로서비스 대상 사이버 위협의 분류

마이크로서비스는 아키텍처에 대한 주목은 컨테이너와 같은 가상화 기술의 발전으로 가상 자원 격리, 오토스케일링, 이미지 기반의 편리한 배포가 가능한 장점이 있기 때문이다. 이러한 특징은 마이크로서비스의 장점으로써 긍정적인 영향을 미치지만, 보안 측면에서는 새로운 이슈를 발생시킨다[13]. 예를 들어 가상화 기술의 자원 격리의 허점을 노린 취약점을 발생하기도 하며[14], 컨테이너 기술에서의 접근 권한의 허점을 활용한 취약점이 발견되고 있다[15].

2.2.1 마이크로서비스 대상 사이버 위협의 상세

Yu, D et al[16]. 연구진에 따르면, 마이크로서비스 아키텍처는 단일 소프트웨어와 비교해 보안에 취약하다. Yu, D et al. 연구진은 마이크로서비스 아키텍처를 대상으로 한 사이버 위협의 범주를 크게 컨테이너, 데이터, 네트워크 등으로 나누었으며, 각 항목은 <표 1>과 같다. 그중 마이크로서비스의 핵심 기반 기술인 컨테이너와 네트워크와 관련된 정보를 살펴본다.

컨테이너 대상 사이버 위협 컨테이너는 마이크로서비스의 기반기술이다. 컨테이너 기술의 핵심은 자원 격리 기능이다. 컨테이너 기술을 활용하면 단일 호스트에서 여러 컨테이너를 구동하더라도 각 컨테이너에 격리된 환경을 제공함으로써 다수의 컨테이너가 단일 호스트의 자원을 나누어 사용할 수 있다. 컨테이너 기술은 cgroups와 네임 스페이스 등의 격리 기능을 활용하여, 접근 권한이 없는 컨테이너에 접근하는 것을 방지할 수 있다.

그러나 컨테이너 기술은 커널 가상화까지 수행하지는 않으므로, 가상 머신(Virtual Machine, VM)에 비해서 완벽한 격리를 보장하지 않아, 상대적으로 보안성이 취약하다는 단점을 가지고 있다[17].

컨테이너를 대상으로 가능한 공격은 자원 격리 우회, 권한 탈취 등의 컨테이너 자체 취약점에 직접적인 공격 및 도커 허브 등의 컨테이너 배포 플랫폼에 오염된 이미지를 배포하는 것과 같은 간접적인 공격이 발생할 수 있다.

네트워크 대상 사이버 위협 마이크로서비스는 특정 요청을 수행하기 위해서 여러 마이크로서비스를 거쳐서 수행되어야 하므로, 상대적으로 단일 소프트웨어보다 공격 표면이 증가한다. 네트워크와 관련된 마이크로서비스 환경 대상 사이버 위협은 서비스 거부 공격, ARP 스푸핑, 중간자 공격 등이 있다.

2.3 분산 추적 기법의 인과 관계 형성

분산 추적 기법은 마이크로서비스 환경에서 서로 관련 있는 내부 요청 로그의 인과 관계를 형성하는 기법이다. 분산 추적 기법은 추적 과정에서 메타데이터 전파기법을 사용한다. 메타데이터 전파기법은 Jaeger[18], Zipkin[19] 등 다수의 분산 추적 도구에서 사용된다. 메타데이터 전파란 추적에 활용할 메타데이터 정보를 내부 서비스 간 요청 패킷에 지속해서 전파 시키는 기법을 의미한다. 추적점(Trace point)은 추적 대상 서비스에 구현되어 메타데이터를 생성한다. 외부 사용자로부터 추적점이 구현된 서비스를 대상으로 요청이 발생하면, 추적점의 추적 코드는 요청 간 인과 관계 형성을 위한 고유 식별자를 요청 패킷의 헤더 영역에 기록한다. 메타데이터는 다음 서비스로 메타데이터를 전달하고, 이것이 반복되어 전체 서비스에서 서로 관련 있는 요청을 한 눈에 파악할 수 있게 된다.

추적데이터는 Trace와 Span 개념으로 구성된다. Trace는 외부 사용자에서 전송된 요청과 응답 과정으로 인해 생성되는 전체 추적 흐름이며, Span은 Trace를 이루는 논리적인 작업단위에 대한 추적데이터이다. 추적 대상 서비스에서 메타데이터가 기록된 요청 패킷을 수신하면 Span 데이터가 생성되며, 타임스탬프, 작업 명칭 등을 기록하여 분산 추적 백엔드로 전송한다. 분산 추적 백엔드는 Trace와 Span 데이터를 기반으로 요청 간 인과 관계를 재구성한다. 이러한 과정을 거쳐, 마이크로서비스 환경에서 지연 발생 지점 같은 정보를 파악할 수 있다. 메타이터 전파기법은 추적 대상이 되는 모든 마이크로서비스에 추적점을 사전 구현해야 하는 번거로움이 있으나, 최근에는 이러한 추적점을 자동으로 구현하는 공개 라이브러리를 활용하여 이러한 수고를 줄일 수 있다. 또한, 블랙박스 추론, 스키마 기반과 같은 다른 추적 기법과 비교하여 정확한 인과 관계 정보를 제공하므로, 현재 사용되는 대다수의 분산 추적 도구가 사용하는 분산 추적 기법이다[19-22].

3. 공격 진원지 추적형 컨테이너 스냅샷

본 장에서는 제안하는 공격 진원지 추적형 컨테이너 스냅샷 기법의 구조, 작동 과정을 설명한다.

3.1 분산 추적 플랫폼에 적용된 공격 진원지 추적형 컨테이너 스냅샷 기법의 구조

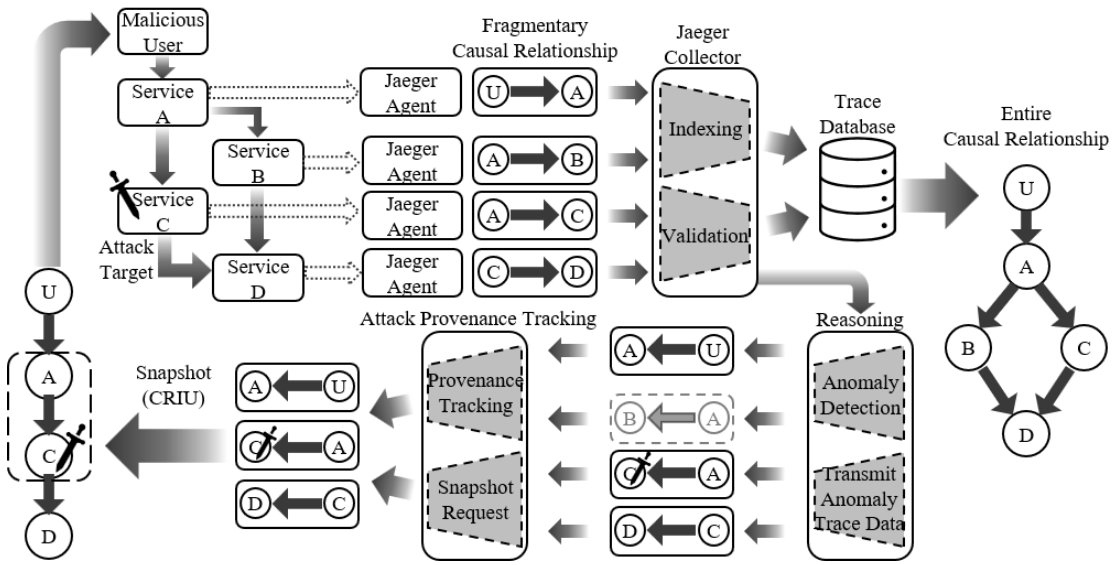
본 절에서는 분산 추적 플랫폼과 공격 진원지 추적형 컨테이너 스냅샷 기법의 구성 요소에 관해서 설명한다. 예시에 사용한 분산 추적 플랫폼은 오픈소스 분산 추적 플랫폼인 Jaeger이다.

공격 진원지 추적형 컨테이너 스냅샷 기법이 적용된 분산 추적 플랫폼은 (그림 1)과 같다. 전체 구조에 대한 간략한 내용은 3.1절에서 살펴보며, 3.2절에서는 공격 진원지 추적형 컨테이너 스냅샷 기법이 분산 추적 플랫폼에서 원활하게 사용하기 위한 요구 사항을 설명한다. 분산 추적 플랫폼과 함께 공격 진원지 추적형 스냅샷 기법의 상세한 작동 방식은 3.3 절에서 설명한다.

3.1.1 분산 추적 플랫폼 구조

Jaeger 분산 추적 플랫폼의 기본적인 구성 요소는 (그림 1)에서 Jaeger Client, Jaeger Agent, Jaeger Collector이다.

Jaeger Client는 추적 대상 마이크로서비스에 이



(그림 1) 공격 진원지 추적형 컨테이너 스냅샷 기법이 적용된 분산 추적 플랫폼 전체 구조

식되어 추적데이터 생성에 필요한 라이브러리를 제공한다. 생성하는 추적데이터는 요청 간 구분을 위한 고유 식별자인 Trace ID, 서비스별 논리적 작업 단위를 구분하기 위한 고유 식별자인 Span ID, 서비스 지연 시간 정보인 타임스탬프로 구성되어 있다. 추적데이터의 Trace ID는 요청 패킷의 헤더에서 읽어온 Trace ID를 사용하며, 만일 Trace ID가 없는 요청이라면 새로운 Trace ID를 생성한다. 같은 외부 요청으로 인해 발생한 모든 추적데이터에는 같은 Trace ID가 기록되므로, 서로 다른 외부 요청 관련 로그가 뒤섞인 데이터베이스에서도, 관련 있는 정보만을 추출할 수 있다.

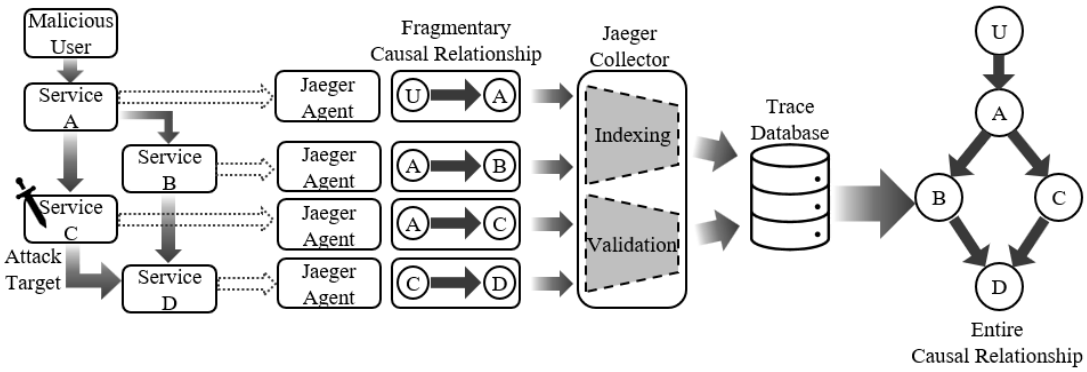
Jaeger Agent는 분산 추적 플랫폼 구동 시, 추적 대상 마이크로서비스와 1:1로 대응되어 구동된다. Jaeger Client로부터 추적데이터를 전송받아 버퍼에 잠시 저장하며, Jaeger Collector까지 전송할 수 있도록 라우팅한다. 버퍼에 저장된 추적데이터는 일정 간격으로 Jaeger Collector로 전송된다.

Jaeger Collector 또한 버퍼로 구성되어 있어, Jaeger Agent로부터 전송받은 추적데이터를 버퍼에 잠시 저장한다. Jaeger Collector는 버퍼에 저장된 추적데이터에 대해서 유효성을 검증하고, 데이터베이스에 적합한 형태로 변환한 데이터베이스로 전송한다.

3.1.2 공격 진원지 추적형 컨테이너 스냅샷 기법의 구조

공격 진원지 추적형 컨테이너 스냅샷 기법은 분산 추적 플랫폼에 적용되며, Reasoning Engine, Attack Provenance Tracking Engine, CRIU[23]로 구성된다. 이러한 구성 요소 중 컨테이너 스냅샷을 직접 수행하는 CRIU를 제외하면, 컨테이너 형태로 구동된다.

Reasoning Engine은 Jaeger Collector의 버퍼에서 추적데이터가 머무르는 동안 추적데이터를 파싱하여 이상 행위를 탐지한다. Reasoning Engine은 이상 행위 탐지를 위해, 사전에 설정한 지연 시간 임계치 정보를 활용한다. 예를 들어 추적데이터에서 확보한 마이크로서비스의 지연 시간이 사전에 설정한 임계치를 넘어서면 이와 관련된 서비스를 이상 행위가 발생한 서비스로 판단한다. 이후 이상 행위와 관련된 추적 정보를 Attack Provenance Tracking Engine로 전송한다. Reasoning Engine은 도커 컨테이너 형태로 구동하므로, 새로운 이상



(그림 2) 분산 추적 플랫폼의 요청 패킷과 추적데이터 흐름

행위 탐지 기법 적용이 필요하면 교체할 수 있다.

Attack Provenance Tracking Engine은 Reasoning Engine로부터 전송받은 추적데이터에서 문제가 발생한 Server 주소, 이상 행위가 발생한 마이크로서비스의 컨테이너 PID 정보를 전송받아 CRIU에 Container Snapshot 명령을 내린다.

3.2 분산 추적 규격에 대한 요구 사항

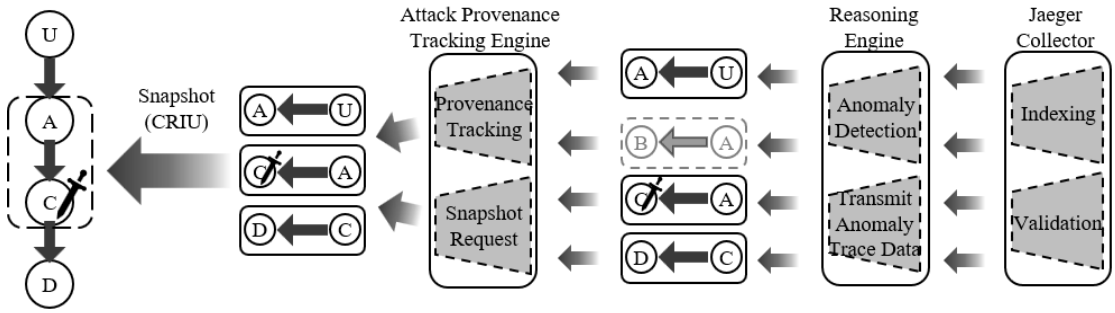
본 절에선 제안하는 공격 진원지 추적형 컨테이너 스냅샷 기법이 분산 추적 플랫폼이 원활하게 구동될 수 있도록 기존의 분산 추적 플랫폼에서 추가로 필요한 요구 사항을 나열한다.

먼저 공격 진원지 추적형 컨테이너 스냅샷 기법을 분산 추적 플랫폼에서 활용하기 위해선, 원격지 호스트에 존재하는 특정 마이크로서비스 컨테이너를 지목할 수 있는 정보가 필요하며, 이러한 정보를 추적데이터에 담아 전송해야 한다. 공격 진원지 추적형 컨테이너 스냅샷 기법은 원격지 호스트에서 컨테이너로 구동되며, 컨테이너 스냅샷을 수행해야 하는 대상 컨테이너는 원격지 호스트에서 구동된다. 따라서 Attack Provenance Tracking Engine이 원격지 호스트에 컨테이너 스냅샷 명령을 내리기 위해선, 컨테이너 스냅샷 대상 마이크로서비스의 컨테이너 위치를 특정 지을 수 있는 호스트 IP 정보 및 컨테이너 PID 정보 등의 추가 정보가 필요하다.

3.3 분산 추적 플랫폼에 적용된 공격 진원지 추적형 컨테이너 스냅샷 작동 과정

본 절에서는 가상의 마이크로서비스 환경을 상정하여 공격 진원지 추적형 컨테이너 스냅샷 기법과 분산 추적 플랫폼의 작동 과정에 관해 일련의 가상 작동 시나리오를 설명한다.

1) (그림 2)는 Jaeger 분산 추적 플랫폼을 사용하는 마이크로서비스 환경을 나타낸다. 각 서버에는 마이크로서비스가 구동되고 있으며, 각 마이크로서비스가 모두 추적 대상이라는 가정하에, Jaeger Client가 구현되어 있다. Jaeger 분산 추적 플랫폼 구동 시, Jaeger Agent는 추적 대상 마이크로서비스와 1:1로 대응되어 구동하며, Jaeger Collector는 Jaeger Agent와 1:N로 대응되어 구동된다. 공격 진원지 추적형 컨테이너 스냅샷 기법은 Jaeger 분산 추적 플랫폼과 함께 백그라운드에서 작동한다. 외부 사용자의 요청을 처리하는 과정에서 요청 패킷이 추적 대상 마이크로서비스를 경유 하면, Jaeger Client는 추적데이터를 생성한다. 마이크로서비스가 외부 사용자로부터 요청 패킷을 수신하면, 서비스 작업의 고유 식별자를 생성하고, 요청 패킷의 헤더에 기록한다. 헤더에는 고유 식별자를 기록하여 요청 간 인과 관계 형성 시 사용한다. 생성된 추적데이터는 Jaeger Agent로 전송되며, 추적데이터에는 요청 간 기록되어 있다. 생성된 추적데이터는



(그림 3) 공격 진원지 추적 과정

Jaeger Agent를 거쳐 Jaeger Collector로 전송된다. Jaeger Collector는 추적데이터에 대하여 인덱싱, 데이터 유효성 검사를 수행한 뒤, 데이터베이스에 저장된다.

2) (그림 3)은 공격 진원지 추적형 컨테이너 스냅샷 기법이 추적데이터 중 이상 행위 정보를 판단하고, 컨테이너 스냅샷을 수행하는 과정이다. 먼저 Reasoning Engine은 이상 행위 정보를 포함하고 있는 추적데이터를 선별하기 위해 Jaeger Collector의 버퍼에 머무르는 추적데이터를 파싱한다. Reasoning Engine은 서비스 지연 시간 데이터가 사전에 설정한 임계치 초과 여부를 기준으로 이상 행위를 데이터를 결정한다. 이상 행위로 판별된 데이터에 대해선 컨테이너 스냅샷 결정을 내린다. 마이크로서비스 간 인과 관계 정보 및 스냅샷 대상 마이크로서비스 컨테이너가 구동되는 호스트 주소 정보, 스냅샷 대상 마이크로서비스의 PID 정보를 Attack Provenance Tracking Engine로 전송한다. 이후 Attack Provenance Tracking Engine은 Reasoning Engine로부터 전송받은 인과 관계 정보를 기반으로 요청의 흐름을 재구성하여 전체 인과 관계 정보를 파악한다. 재구성한 요청 정보와 이상 행위가 발생한 컨테이너의 원격지 호스트 정보를 기반으로 원격지 호스트의 CRIU에 컨테이너 스냅샷 명령을 내린다. CRIU는 추적 대상 호스트 환경에 사전에 이식되어 있음을 가정한다.

4. 관련 연구

본 장에서는 분산 추적 기법 관련 연구와 분산 추적 기술을 응용하여 보안 분야에 활용한 연구를 살펴본다.

4.1 분산 추적 기법 관련 연구

분산 추적 기법과 관련된 주요 연구 주제는 분산 추적데이터의 인과 관계 형성 요소 선정 및 형성 방안에 관한 연구, 분산 추적데이터를 선별 및 저장하는 과정에서 발생하는 오버헤드 경량화 방안에 관한 연구, 분산 추적데이터의 효과적인 시각화 방안에 관한 연구로 분류할 수 있다[24]. 분산 추적데이터의 인과 관계 형성 방안 및 형성 요소에 관하여 메타데이터 전파기법, 스키마 기반 기법, 블랙박스 추론 기법이 제안되었다.

Barham, Paul et al[25]. 연구진이 제안한 Magpie는 추적 대상 시스템의 커널 레벨 모니터링 데이터를 활용한 스키마 기반의 분산 추적 툴체인(Tool chain)이다. Magpie는 서비스 요청 데이터의 인과 관계뿐만 아니라 자원 사용량 정보 또한 제공하여, 시스템에 대한 폭넓은 이해를 제공한다. Magpie는 추적 대상 시스템의 모든 워크로드 모니터링 데이터에서 사전에 정의한 이벤트 스키마를 기준으로 서비스 간 요청 정보를 추출하고, 이러한 데이터를 정규화한 뒤, 관련 있는 데이터만을 클러스터링하는 방식으로 요청 간 인과 관계를 형성한다. 그러나, Magpie는 추적 대상 시스템의 모든 데

이터를 로깅 해야 하므로 샘플링 기법을 사용하는 기타 추적 도구보다 상대적으로 많은 양의 추적데이터 저장공간이 필요하다. 또한, 추적데이터의 인과 관계 형성에 필요한 이벤트 스키마는 사전 정의될 필요가 있으므로 규모 및 서비스의 변화가 빈번한 환경에서는 사용에 한계가 있다.

〈표 2〉 실험 환경

Evaluation Environment	
CPU Cores	8
Memory	16GB
Operating System	Ubuntu 18.04.5 LTS
Kernel Version	5.12.0-051200-generic
Container Environment	Docker version 20.10.6, build 370c289
Snapshot Tool	CRIU 3.15 "Titanium Falcon"
Container Image	Busybox(32)
Number of containers	1000

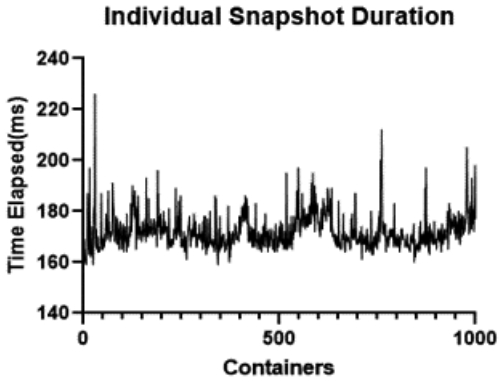
Sigelman et al[26]. 연구진이 제안한 Dapper 분산 추적 도구는 분산 추적 과정을 추적의 전체 흐름인 Trace와 서비스 동작의 논리적 단위인 Span이라는 세부 개념으로 나누어 수행하였다. Dapper는 Trace와 Span에 고유 식별자를 부여하고 이를 기반으로 인과 관계를 형성함으로써, 기존에 제안된 블랙박스 추론, 스키마 기반의 방법보다 정확한 인과 관계를 형성할 수 있다. 또한, 분산 추적데이터의 수집 과정에 있어 모든 서비스 간 요청을 수집하는 것이 아닌 수집 정책에 의해 데이터를 선별적으로 수집하는 샘플링 기법을 사용하였다. 이러한 샘플링 기법을 통해 연산 오버헤드를 줄일 수 있었으며, 디스크 저장 공간을 효율적으로 사용할 수 있었다. 그러나, 추적을 위해 사용하는 Trace와 Span 정보를 생성하기 위해선 마이크로서비스에 추적 관련 코드가 사전에 구현될 필요가 있다. 또한, Dapper가 추적을 위해 사용하는 메타데이터는 요청 패킷에 기록되어 전송되는데, 이 과정에서 오버헤드가 읽기, 쓰기 오버헤드가 발생한다.

Mace, Jonathan, et al[27]. 연구진은 동적 코드 삽입 및 메타데이터 전파를 동시에 수행하는 Pivot Tracing을 제안하였다. Pivot Tracing은 두 가지 작업을 동시에 수행함으로써, 이기종 장비로 구성된 분산 시스템 환경에서 추적 대상의 특징에 적합한 추적데이터를 수집한다. 이를 통해 사전 정의된 추적데이터만을 수집할 수 있는 기존의 분산 추적 기법 한계를 극복할 수 있었다.

이 외에도 Las-Casas, Pedro, et al[28]. 연구진은 Sifter를 통해 추적데이터 저장과정에서 발생하는 연산 오버헤드 및 저장공간 이슈를 해결하고자 하였다. Sifter는 전체 추적데이터에서 이상 행위 데이터를 포함하고 있는 유의미한 데이터는 극히 일부라는 점에 착안하여 제안되었다. Borkin, Michelle A., et al[29]. 연구진은 인과 관계 정보를 갖는 데이터를 효과적으로 시각화하기 위해 방사형 트리 레이아웃을 제공하는 Inprov를 제안하였다.

4.2 분산 추적 기법의 보안 관련 연구

Nedelkoski et al[30]. 연구진은 자동화된 인프라 유지 관리에 초점을 맞추어 연구를 수행하였다. 이를 위해 분산 추적 기술을 활용한 이상 행위 탐지를 수행하였으며, 서비스 지연 시간 정보 및 가용성 정보를 기반으로 딥 러닝을 수행하여 이상 행위 탐지 모델을 생성하였다.



(그림 4) 공격 진원지 파악을 위한 개별 컨테이너 스냅샷에 필요한 시간 분포

Jacob et al[31]. 연구진은 마이크로서비스 환경에서 분산 추적 기법을 활용하여 사이버 위협에 대한 보안을 수행하는 연구를 진행하였다. 이상 행위를 탐지하기 위해 일반적인 통신 데이터로부터 획득한 추적데이터의 빈도 분포를 기준으로 이상 행위를 탐지하였으며, 이러한 이상 행위 탐지 기법의 정확도를 검증하기 위해 Password Guessing Attack을 활용하였다. 또한, 단순한 분산 추적데이터만으로는 NoSQL과 같은 공격에 대해서 탐지하는 데 한계가 있음을 보였다.

5. 실험

본 장에선 제안하는 공격 진원지 추적형 컨테이너 스냅샷 기법의 실용성을 검증하기 위해, 가상의 마이크로서비스 환경을 대상으로 컨테이너 스냅샷 수행의 소요시간을 측정하였다. 실험에 사용된 컨테이너 스냅샷 도구는 CRIU이며, 실험 환경은 다음과 같다. 실험은 가상 머신(Virtual Machine, VM)에서 수행되었으며, 상세한 사항은 다음 <표 2>와 같다.

성능 측정 시, 마이크로서비스와 유사한 환경에서 측정하기 위해 1000개의 컨테이너를 구동한 뒤, 스냅샷 수행에 걸린 시간을 측정하였다. 스냅샷 과정에선 약 0.7%의 오버헤드가 발생하였으며, 단일 컨테이너에 대해서 스냅샷을 수행하는데 평균 172ms가 필요하였다. 인과 관계 추적을 가정하여, 연속적 스냅샷을 수행하더라도, 지연 시간은 선형적으로 증가하여 100개의 컨테이너에 대해서 스냅샷을 수행하는데 약 172초가 필요하였다. 각 컨테이너에 대해서 스냅샷을 수행하는데 필요한 시간 분포는 (그림 4)와 같다.

6. 결론

본 논문은 마이크로서비스 환경에서 서비스 성능 관리 및 디버깅을 수행하기 위해 사용하는 분산 추적 플랫폼을 보안 목적에 다방면으로 활용할 수 있도록, 공격 진원지 추적형 컨테이너 스냅샷 기법을 제안하였다. 제안하는 기법은 분산 추적 기법을 통해 확보한 인과 관계 정보를 기반으로 효율적인 컨테이너 스냅샷을 수행하도록 설계하였다. 이와 같은 설계를 통해 사이버 위협과 관련된 마이크로서비스 컨테이너만을 대상으로 컨테이너 스냅샷을 수행할 수 있으며, 스냅샷 데이터 분석의 효율을 높일 수 있다. 추후 연구에선 이상 행위 탐지, 추적데이터 파싱과 관련된 성능 측정 실험을 추가로 수행할 예정이다. 또한, 컨테이너 스냅샷 수행에 필요한 시간 외에도, 스냅샷 과정에서 발생하는 서비스 중단 시간, 스냅샷 이미지의 용량 등의 성능 관련 부분을 추가로 검증할 예정이다.

Acknowledgment

본 연구는 2019년도 과학기술정보통신부의 재원으로 정보통신기획평가원의 지원(No. 2019-0-00273, No.2019-0-00426) 및 한국연구재단 연구과제(NRF-2020R1A2C4002737)의 지원을 받아 수행된 연구임

참고문헌

[1] Yu, Dongjin, et al. "A survey on security issues in services communication of Microservices enabled fog applications." Concurrency and Computation: Practice and Experience 31.22 (2019): e4436.

[2] Ebert, Christof, et al. "DevOps." Ieee Software 33.3 (2016): 94-100.

[3] Mace, Jonathan, and Rodrigo Fonseca. "Universal context propagation for distributed system instrumentation." Proceedings of the Thirteenth EuroSys Conference. 2018.

[4] Microservices, https://martinfowler.com/articles/microservices.html

[5] MicroservicePremium, https://martinfowler.com/bliki/MicroservicePremium.html

[6] Microservices Security Probably Not What You

Think It Is, https://thenewstack.io/microservices-security-probably-not-what-you-think-it-is/
〔7〕 Obermeier, Sebastian, et al. "Automatic 공격 표면 reduction in next-generation industrial control systems." 2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS). IEEE, 2014.
〔8〕 Nedelkoski, Sasho, Jorge Cardoso, and Odej Kao. "Anomaly detection and classification using distributed tracing and deep learning." 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). IEEE, 2019.
〔9〕 Kubernetes, https://kubernetes.io/
〔10〕 Shkuro, Yuri. Mastering Distributed Tracing: Analyzing performance in microservices and complex systems. Packt Publishing Ltd, 2019.
〔11〕 Gregg, Brendan, and Jim Mauro. DTrace: Dynamic Tracing in Oracle Solaris, Mac OS X, and FreeBSD. Prentice Hall Professional, 2011.
〔12〕 Bailis, Peter, Peter Alvaro, and Sumit Gulwani. "Research for practice: tracing and debugging distributed systems; programming by examples." Communications of the ACM 60.7 (2017): 46-49.
〔13〕 Mohsen Ahmadvand and Amjad Ibrahim. 2016. Requirements reconciliation for scalable and secure microservice (de) composition. In Requirements Engineering Conference Workshop s (REW), IEEE International. IEEE, 68 - 73.
〔14〕 NIST, https://nvd.nist.gov/vuln/detail/CVE-2019-5736
〔15〕 NIST, https://nvd.nist.gov/vuln/detail/CVE-2021-21284
〔16〕 Yu, Dongjin, et al. "A survey on security issues in services communication of Microservices enabled fog applications." Concurrency and Computation: Practice and Experience 31.22 (2019): e4436.
〔17〕 Gupta, Udit. "Comparison between security majors in virtual machine and linux containers." arXiv preprint arXiv:1507.07816 (2015).
〔18〕 Jaeger, https://www.jaegertracing.io/
〔19〕 Zipkin, https://zipkin.io/
〔20〕 Operations, https://cloud.google.com/products/operations
〔21〕 X-ray, https://aws.amazon.com/ko/xray/
〔22〕 Light-step, https://lightstep.com/
〔23〕 CRIU, https://criu.org/Docker
〔24〕 Sambasivan, Raja R., et al. "So, you want to trace your distributed system? Key design insights from years of practical experience." Parallel Data Lab., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-PDL 14 (2014).
〔25〕 Barham, Paul, et al. "Using Magpie for request extraction and workload modelling." OSDI. Vol. 4. 2004.
〔26〕 Sigelman, Benjamin H., et al. "Dapper, a large-scale distributed systems tracing infrastructure." (2010).
〔27〕 Mace, Jonathan, Ryan Roelke, and Rodrigo Fonseca. "Pivot tracing: Dynamic causal monitoring for distributed systems." Proceedings of the 25th Symposium on Operating Systems Principles. 2015.
〔28〕 Las-Casas, Pedro, et al. "Sifter: Scalable sampling for distributed traces, without feature engineering." Proceedings of the ACM Symposium on Cloud Computing. 2019.
〔29〕 Borkin, Michelle A., et al. "Evaluation of filesystem provenance visualization tools." IEEE transactions on visualization and

computer graphics 19.12 (2013): 2476-2485.
〔30〕 Nedelkoski, Sasho, Jorge Cardoso, and Odej Kao. "Anomaly detection and classification using distributed tracing and deep learning." 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). IEEE, 2019.
〔31〕 Jacob, Stephen, Yuansong Qiao, and Brian Lee. "Detecting Cyber Security Attacks against a Microservices Application using Distributed Tracing." (2021).
〔32〕 Busybox, https://hub.docker.com/_/busybox

저자소개

◆ 이병용

- 2018년 대전대학교 정보보안학과 학사
- 2019년 세종대학교 정보보호학과 석사 과정
- 관심분야: 클라우드, 가상화, 분산 추적

◆ 박기웅

- 2005년 연세대학교 Computer Science 학사
- 2007년 KAIST Electrical Engineering 석사
- 2012년 KAIST Electrical Engineering 박사
- 2008년 Microsoft Research Asia, Wireless and Networking Group, Research Intern
- 2009년 Microsoft Research, Network Research Group, Graduate Research Fellow
- 2012년 국가보안기술연구소 연구원
- 2012년~2016년 대전대학교 정보보안학과 교수
- 2016~현재 세종대학교 정보보호학과 교수
- 관심분야: 시스템보안, 모바일-클라우드 컴퓨팅, 보안 프로토콜, 디지털 포렌식