K-ONE 기술 문서 #10

Security-Mode ONOS 기반 자동 퍼미션 추출

Automatic Policy Generation on Security-Mode ONOS

Document No. K-ONE #10
Version 1.0
Date 2016-04-30

Author(s) 김진우

K-ONE

ONK
OpenNetworkingKorea

■ 문서의 연혁

버전	날짜	작성자	내용
초안 - 0.1	2016. 04. 01	김진우	초안 작성
0.5	2016. 04. 07	김진우	내용 추가
0.8	2016. 04. 15	김진우	내용 추가
1.0	2016. 04. 30	김진우	작성 완료

본 문서는 2015년도 정부(미래창조과학부)의 재원으로 정보통신 기술진흥센터의 지원을 받아 수행된 연구임 (No. B0190-15-2012, 글로벌 SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발)

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. B0190-15-2012, Global SDN/NFV OpenSource Software Core Module/Function Development)

기술문서 요약

소프트웨어 정의 네트워킹은 기존 네트워크의 제어평면과 데이터평면을 분리하고, 하나의 통합된 제어평면으로 네트워크 관리자에게 통합된 토폴로지 뷰 및 프로그래머블한 환경을 제공한다. 이는 기존 네트워크에 비해 관리자가 수동적으로 네트워크 디바이스에 대한 설정을 할 필요가 없고, 더욱 더 유연하게 비즈니스 로직을 구현할 수 있게끔 한다. 제어 평면은 컨트롤러 또는 Network Operating System (NOS) 라고 불리우는데, 지금까지 오픈 소스 기반의 여러 SDN 컨트롤러들이 개발되었다. 그러나 최근에는, 이러한 SDN 컨트롤러들의 보안 취약점이 상당수 발견되었는데, SDN 환경에서 컨트롤러가 전체 네트워크 환경을 통제하므로, 컨트롤러의 보안성을 향상시키기 위한 연구가 불가피해졌다.

SDN 컨트롤러 중 ONOS는 분산 컨트롤러로써 거대한 네트워크 환경에서 높은 가용성 및 고성능, 확장성을 보장해주며, 이러한 점 때문에 최근에는 산업계 및 학회에서도 ONOS를 주목하고 있다. SDN 컨트롤러의 보안 취약점 연구가 진행됨에 따라, ONOS는 안전한 컨트롤러를 만들기 위해 Security Mode ONOS (SM ONOS)를 제안하였다. SM ONOS는 ONOS의 API에 따른 퍼미션을 정의하고 런타임에 애플리케이션의 API 호출에 따라 이를 감시하고 접근 제어를 한다. 개발자가 애플리케이션을 배포하기 전에, 프로그램에 사용되는 API에 따라 퍼미션 파일에 이러한 퍼미션들을 정의하면, SM ONOS가 이를 참조하여 제어하는 형식이다. 그러나 지금까지의 SM ONOS는 개발자가 일일이 사용되는 API를 보고 퍼미션 파일을 직접 작성해야 한다는 한계점이 있다. 이는 곧 사람이 하는 해야하는 일이므로 실수를 유발할 수가 있으며 시간적인 비용을 초래한다.

따라서 본 기술문서에서는 SM ONOS 기반 자동 퍼미션 추출 프레임워크 ONOS-ApSM (Automatic Policy Generation on Security-Mode ONOS)를 제안한다. 본 ONOS-ApSM은 ONOS 애플리케이션을 정적 분석하여 사용되는 API를 찾아내고, 각 API에 따라 매칭 테이블을 참조하여 사용되는 퍼미션들을 추출해내도록 한다.

Contents

K-ONE #10. Security-Mode ONOS 기반 자동 퍼미션 추출
Automatic Policy Generation on Security-Mode ONOS

그림 목차

표 목차

K-ONE #10. Security-Mode ONOS 기반 자동 퍼미션 추출

Automatic Policy Generation on Security-Mode ONOS

1. 서론

서론에서는 본 기술 문서를 이해하기 위한 배경 지식과 SDN 컨트롤러의 문제점, 그리고 본 연구가 해결하려고 하는 Security-Mode ONSO의 한계점에 대하여 정의하고 마지막으로 관련 연구를 소개하도록 한다.

1.1 배경지식

본 절에서는 소프트웨어 정의 네트워킹과 SDN 컨트롤러의 보안 문제 및 이를 해결하기 위한 Security-Mode ONOS (이하 SM ONOS)에 대해 간략히 다루도록 한다.

1.1.1 소프트웨어 정의 네트워킹 (Software Defined Networking)

오늘날 소프트웨어 정의 네트워킹은 차세대 네트워킹 아키텍처로써 주목 받고 있다. 제어평면과 데이터평면을 분리함으로써 데이터 평면은 간소화 되며 벤더 의존적이었던 컨트롤 로직이 하나의 SDN 컨트롤러로 집중되었다. 이를 통해 네트워크 관리자는 SDN 컨트롤러로부터 API를 제공받고, 이를 이용하여 비즈니스 로직을 SDN 애플리케이션으로 편리하게 구현함으로써, 데이터 평면을 효과적으로 통제 할 수 있게 되었다.

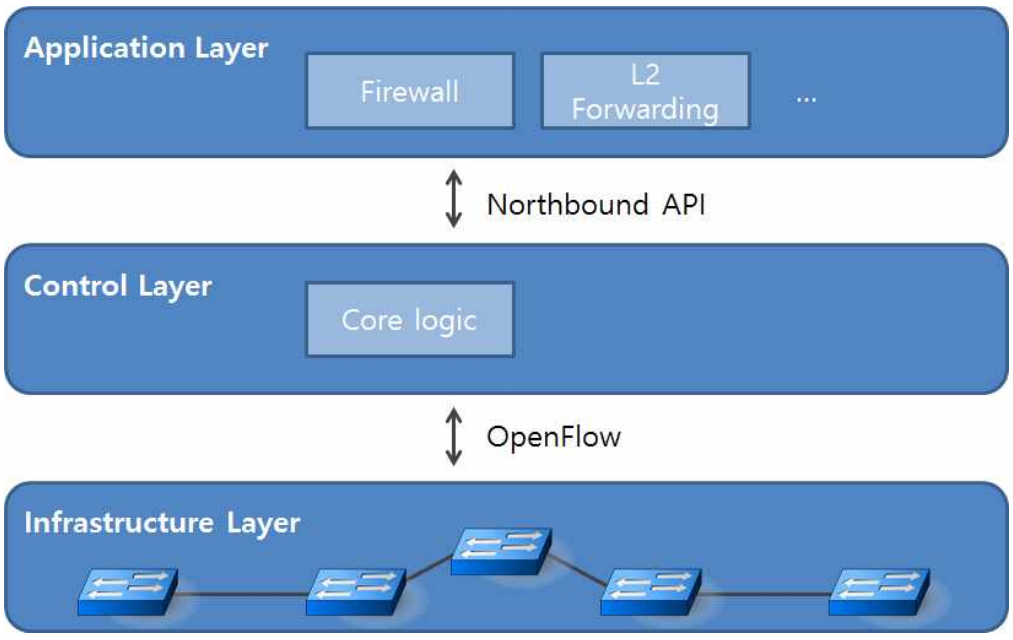


그림 1 소프트웨어 정의 네트워킹 구조

그림 1은 소프트웨어 정의 네트워킹의 전체적인 구조를 나타낸다. 크게 애플리케이션 레이어, 컨트롤 레이어, 그리고 인프라스트럭쳐 레이어 세가지로 나뉜다. 기존에 벤더 의존적이어서 제각각 달랐던 컨트롤 레이어가 SDN 환경에서는 하나의 컨

트롤 레이어로 통합되었다. 그리고 컨트롤 레이어는 위로는 애플리케이션 레이어에 Northbound API를 제공하며, 애플리케이션 레이어는 이를 이용하여 방화벽, L2 포워딩 같은 네트워킹 로직을 쉽게 구현할 수 있게 된다. 아래로는 인프라스트럭쳐 레이어와 OpenFlow 프로토콜 [7]을 이용하여 통신하게 된다. OpenFlow는 SDN의 표준 프로토콜로써, 컨트롤 레이어는 OpenFlow를 통해 스위치에 플로우 룰을 설치하여 네트워크 라우팅을 제어하거나, 스위치로부터 통계정보를 수집하여 인프라스트럭쳐를 모니터링 할 수 있다.

1.1.2 Open Network Operating System (ONOS)

SDN 환경에서는 네트워킹 매니지먼트가 컨트롤 레이어로 중앙 집중됨에 따라 컨트롤 레이어가 매우 중요한 역할을 맡게 되었다. 일반적으로 SDN 컨트롤러라는 명칭으로 불리게 되며 현재까지 NOX, POX, Beacon 같은 오픈 소스 기반의 SDN 컨트롤러가 활발하게 개발 되었다. OpenDayLight[3] 와 ONOS[8] 는 거대 엔터프라이즈 네트워크 환경에서 확장성과 높은 가용성을 보장하기 위해 개발된 분산 SDN 컨트롤러이다. 특히 ONOS는 현재 오픈소스 SDN 컨트롤러 중 가장 활발하게 릴리즈 되어지고 있는데, 미국의 거대 통신회사인 AT&T도 ONOS와 협업 프로젝트를 진행하고 있는 현황이다[6].

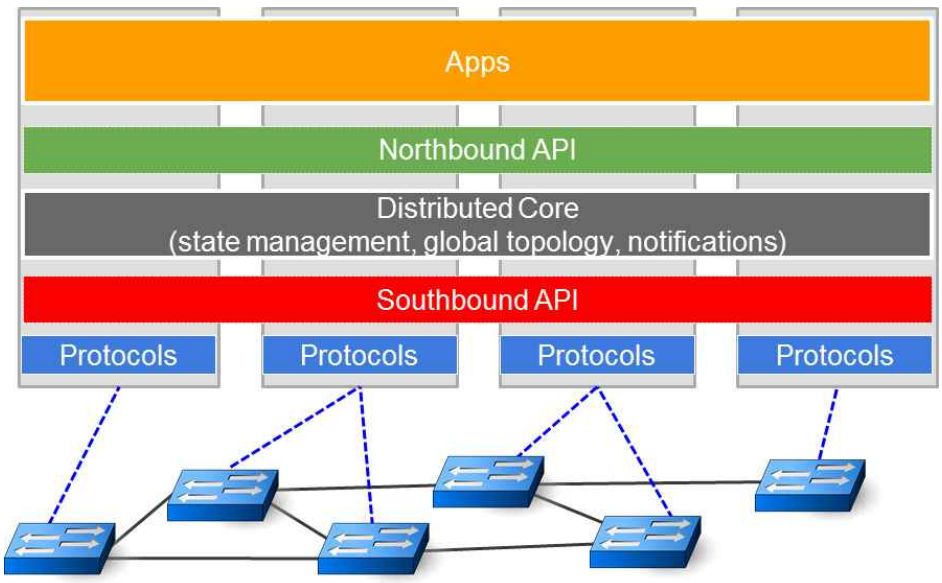


그림 2 ONOS 아키텍쳐

그림 2는 ONOS의 아키텍쳐를 나타낸다. ONOS 역시 분산 코어가 Northbound API를 애플리케이션 레이어에 제공하며 이를 이용하여 네트워킹 로직을 구현할 수 있다. 분산 코어는 각 분산 물리 인스턴스간의 heartbeat 메시지 교환 및 토폴로지 정보 저장 등을 수행한다. ONOS에서는 각 물리 인스턴스들은 전체 토폴로지의 부

분 집합 스위치에 대해 마스터 쉽을 가지게 되고 이를 통하여 확장성을 보장 할 수 있다.

1.1.3 SDN 컨트롤러의 보안 취약점

종래에는 SDN 컨트롤러의 취약점에 관한 여러 연구가 진행됨에 따라[9][10][11], SDN 컨트롤러가 결코 안전하지 않다는 것이 입증되었다. 현재 존재하는 SDN 컨트롤러들은 애플리케이션의 Northbound API 사용에 대한 접근 제어 (Access Control)를 하지 않는다. 따라서 악성 애플리케이션이 컨트롤러에 설치된다면 API를 남용하여 컨트롤러에 악성행위를 할 가능성이 존재한다. 다가올 미래에 SDN이 널리 보급된다면, 써드 파티 SDN 애플리케이션은 급격하게 증가할 것이다. 이미 HP에서는 SDN 앱 스토어를 운영 중이며, 이러한 흐름에 안드로이드 애플리케이션 스토어 같은 거대한 SDN 앱 마켓이 미래에 열릴 것으로 예측 된다[1]. 이에 따라 안드로이드 애플리케이션처럼 악성 행위를 하는 애플리케이션들이 마켓에 존재할 것이고, 이들이 설치되고 활성화되기 전에 애플리케이션이 어떤 행위를 하는지에 대해 알아야 할 필요가 생긴다.

1.1.4 Security-Mode ONOS (SM ONOS)

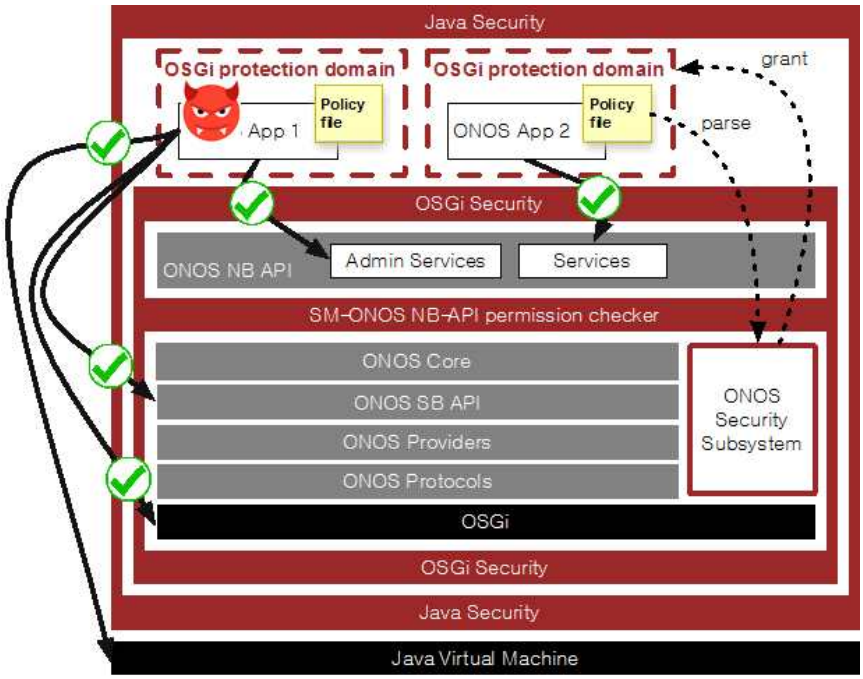


그림 3 Security-Mode ONOS 디자인

KAIST와 SRI International 에서는 현재 기존 SDN 컨트롤러의 취약점이 발견됨에 따라 보다 안전한 ONOS를 만들기 위한 새로운 피쳐를 제안하였는데 그것이 바

로 SM ONOS 이다[5]. 그림 3은 SM ONOS의 전체적인 디자인을 나타낸다. SM ONOS는 독립된 Security Subsystem 으로써 동작하는데, 이로써 애플리케이션의 API 호출에 대해 실시간으로 감시가 가능해진다. 현재 ONOS 아키텍쳐에서 ONOS 애플리케이션은 ONOS Core가 애플리케이션 레이어에 제공하는 Northbound API 뿐만 아니라 OSGi 및 Java API에 대해서도 접근이 가능하다. 애플리케이션이 그림 1과 같이 Northboud, OSGi, 또는 Java API를 호출하는 경우에 먼저 SM ONOS는 ONOS 애플리케이션의 퍼미션 파일을 파싱한 후, 해당 애플리케이션이 요청하는 퍼미션들을 읽어들인다. 만약 퍼미션 파일에 명시된 API를 호출하는 경우에는 이를 승인하며, 그렇지 않은 경우에는 API를 호출하지 못하도록 한다.

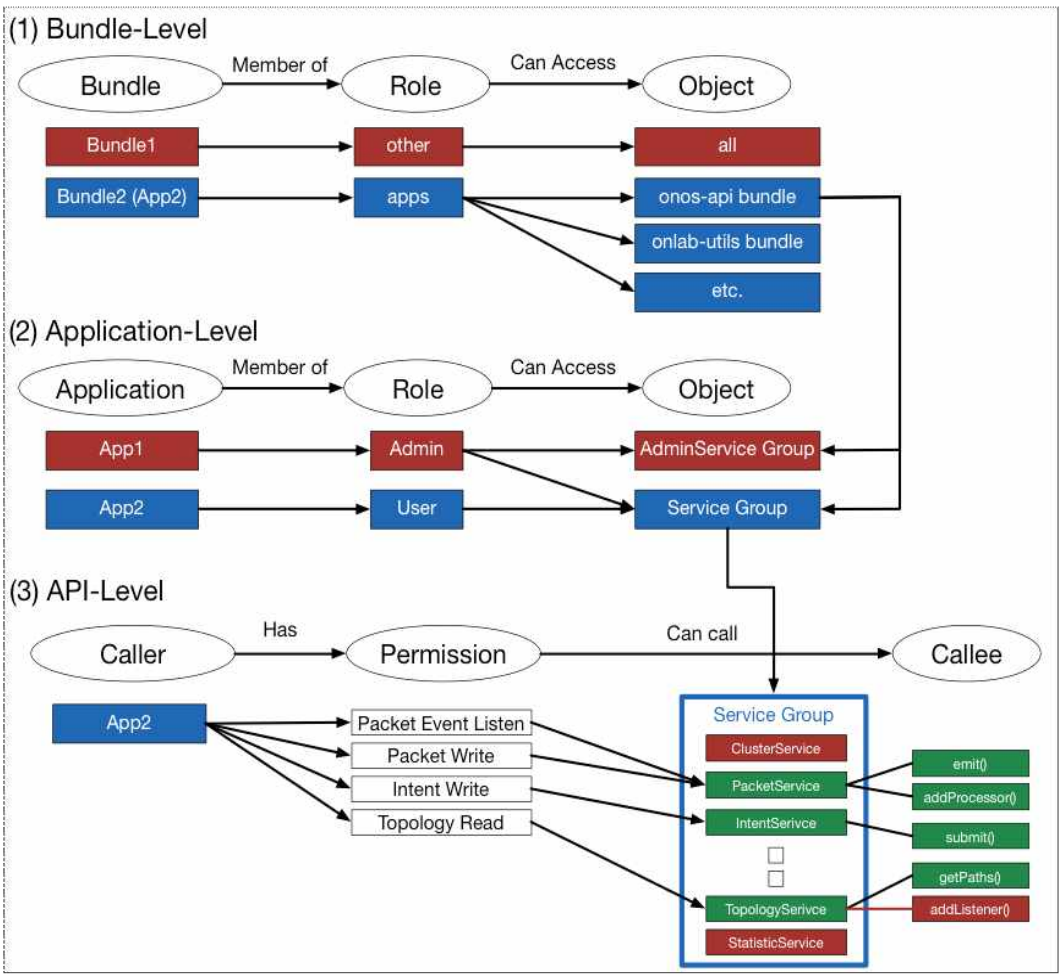
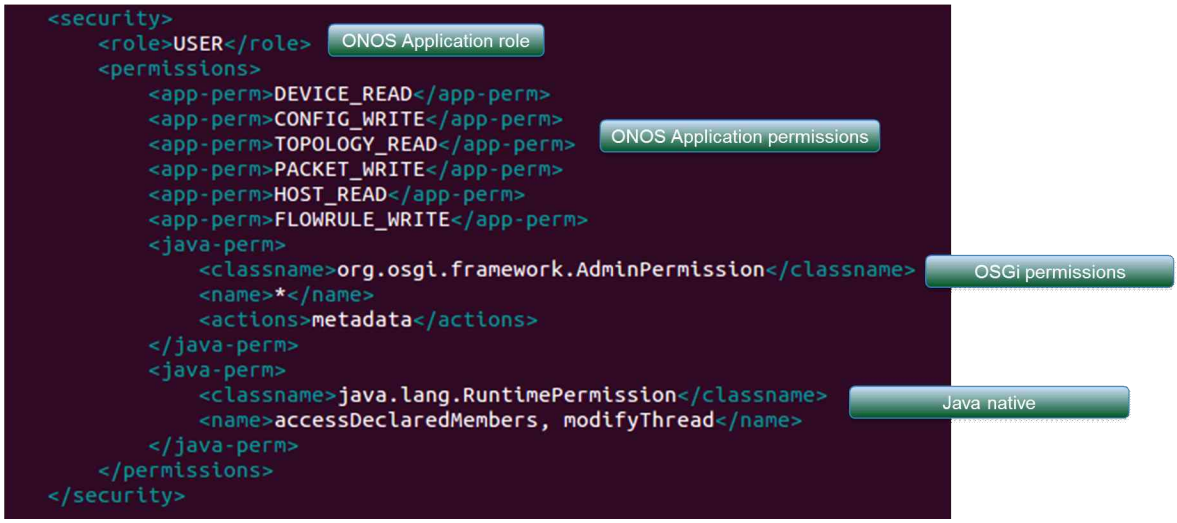


그림 4 Security-Mode ONOS 퍼미션 모델 [5]

SM ONOS 에서는 또한 애플리케이션이 컨트롤러로부터 제공받을 수 있는 API에 따라 퍼미션을 그림 4와 같이 정의하였다. 크게 Bundle-level, Application-level, 그리고 API-level로 나뉘며, 애플리케이션은 API를 호출 할 때 각 레벨에 맞는 퍼미션을 가지고 있는지 체크되고 퍼미션이 있으면 API 호출을 허용한다. 이에 개발자는

애플리케이션을 제작할 때 해당 앱이 사용하는 퍼미션을 그림 5와 같이 퍼미션 파일로 명시하여야 한다. 퍼미션 파일에는 ONOS의 Northbound API 뿐만 아니라 OSGi와 Java Native API 까지 명시가 가능하다. SM ONOS는 해당 애플리케이션이 Activate 될 때, 먼저 퍼미션 파일을 읽어서 네트워크 관리자에게 보여주도록 한다. 애플리케이션이 요청하는 퍼미션 등급 및 종류를 나타내 줌으로써, 관리자는 이를 보고 해당 앱을 Activate 시킬지 말지를 결정할 수 있다.




```
<security>
    <role>USER</role>
    <permissions>
        <app-perm>DEVICE_READ</app-perm>
        <app-perm>CONFIG_WRITE</app-perm>
        <app-perm>TOPOLOGY_READ</app-perm>
        <app-perm>PACKET_WRITE</app-perm>
        <app-perm>HOST_READ</app-perm>
        <app-perm>FLOWRULE_WRITE</app-perm>
        <java-perm>
            <classname>org.osgi.framework.AdminPermission</classname>
            <name>*</name>
            <actions>metadata</actions>
        </java-perm>
        <java-perm>
            <classname>java.lang.RuntimePermission</classname>
            <name>accessDeclaredMembers, modifyThread</name>
        </java-perm>
    </permissions>
</security>
```

그림 5 Policy 파일 형식 예제

1.2 문제정의

현재의 SM ONOS 에서는 애플리케이션 개발자가 퍼미션을 수동적으로 정의해줘야 한다는 한계점이 존재한다. 즉, 개발자는 코드를 보면서 일일이 호출되는 API를 체크하여 퍼미션 파일에 정의해줘야만 한다. 이는 사람이 직접 보고 해야 하는 작업이므로, 개발자가 미처 확인하지 못해서 필요한 퍼미션을 정의하지 못하는 일이 발생할 수 있다. 이러한 결과로 실제 애플리케이션이 요청하는 퍼미션과 개발자가 파일에 명시한 퍼미션간의 차이 (Permission gap)가 발생할 수 있다[13]. 이러한 한계를 극복하기 위해서, 본 연구에서는 ONOS 애플리케이션을 분석하여 해당 앱이 사용하는 API를 분석하고 자동으로 퍼미션을 추출, 퍼미션 파일을 작성해주는 프레임워크를 제안한다.

1.3 관련 연구

일반적으로 프로그램 분석을 위해서는 정적 분석 및 동적 분석이 사용된다. 정적 분석은 프로그램의 실행 없이 소스 코드를 분석하는 기법으로, 이를 이용하여 자바 애플리케이션의 바이트 코드를 분석하여 사용되는 API 리스트들을 알아 낼 수 있

다. 동적 분석은 정적 분석과 대비되는 방법으로써, 실제로 프로그램을 임의의 입력 값과 함께 실행함으로써 애플리케이션의 행위를 분석하는 방법이다. 동적 분석은 애플리케이션의 행위를 보다 정교하게 분석할 수 있다는 장점이 있는 반면에, 애플리케이션의 입력 가능한 경우의 수는 매우 다양하므로 높은 비용 및 시간을 요한다. 이러한 이유로 애플리케이션을 분석하는 경우에는 대개 정적 분석을 이용한다.

정적 분석은 악성 애플리케이션이 여럿 존재하는 안드로이드 플랫폼에서 이를 분석하기 위해 자주 사용되는 기법이다. Wu 등[12] 은 애플리케이션들의 정적 정보 (e.g., 퍼미션 정보, API 호출, intent messgae 등) mainifest나 바이트 코드를 분석함으로써 추출해 내는 프레임워크 DriodMat을 제안하였다. 이러한 정보를 정상적인 앱과 악성 앱으로부터 추출한 뒤에 이를 바탕으로 머신 러닝 알고리즘인 K-means와 EM (Expectation-maximization) 알고리즘을 사용하여 모델을 만들었다. Bartel 등[13] 은 안드로이드 환경에서 애플리케이션이 필요로 하는 퍼미션을 추출하고, 이를 manifest 파일에 선언된 퍼미션과 비교하여 permission gap 을 알아내는 방법을 제안하였다. 해당 논문에서는 안드로이드 환경에서 애플리케이션들이 manifest에 명시되어 있는 퍼미션들에 비해 더 많은 퍼미션을 부여 받아서 격차(gap)가 생기는 것을 permission gap 이라고 정의하였다. 따라서 이러한 격차를 해소하기 위해 바이트 코드를 정적 분석하여, API와 그에 따른 퍼미션으로 매핑 테이블을 작성하고, 애플리케이션이 실제 사용하는 퍼미션 (Inferred permission)만을 추출하였다. 이 후에는 manifest 파일에 선언된 퍼미션과 비교하여 permission gap을 알아낸다.

2. 본론

본론에서는 본 연구가 제안하는 프레임워크 ONOS-ApSM의 전체적인 디자인과 동작 흐름을 설명하고, 구현 및 검증에 관해 다루도록 한다.

2.1 ONOS-ApSM 디자인

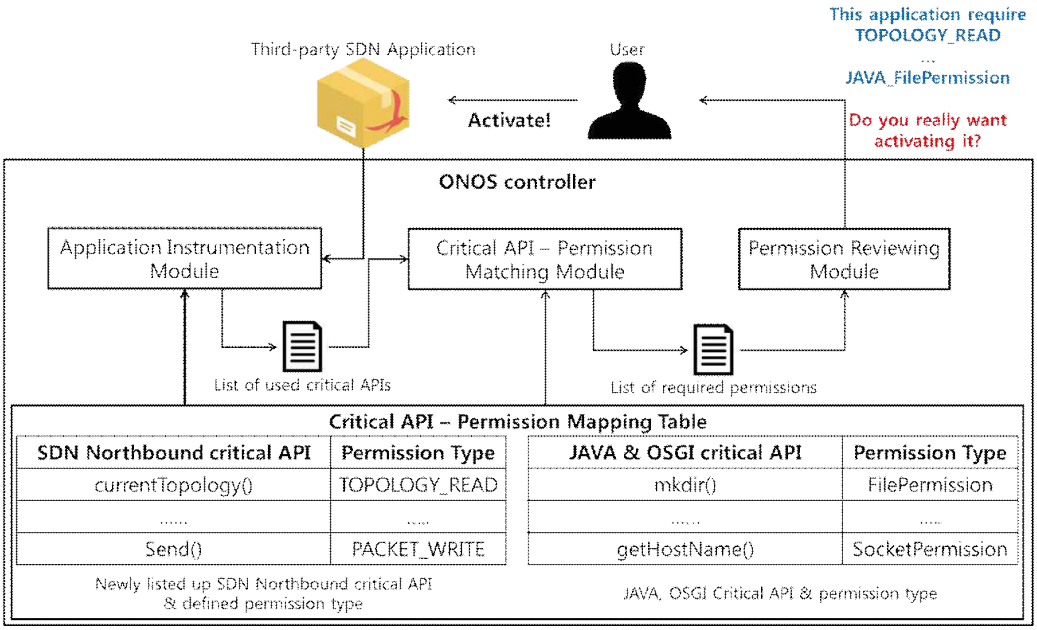


그림 6 자동 퍼미션 추출 프레임워크

본 연구에서는 1.3에서 제시한 안드로이드 기반 애플리케이션 정적 분석 방법을 ONOS 애플리케이션에 적용함으로써, 필요한 퍼미션을 자동 추출하는 프레임워크 ONOS-ApSM를 제안한다. 그림 6은 본 ONOS-ApSM의 전체적인 동작 시나리오 및 구성 모듈을 나타낸다. 사용자(User) 가 써드파티 SDN 애플리케이션(Third-party SDN Application)을 Activate하게 되면, 본 프레임워크의 애플리케이션 검사 모듈 (Application Instrumentation Module) 이 애플리케이션의 코드를 정적 분석 한 뒤, 호출되는 중요 API의 리스트를 추출해 낸다. 이후 퍼미션 매칭 모듈 (Permission Matching Module)이 퍼미션 매핑 테이블 (Permission Mapping Table)을 참조하여 애플리케이션이 호출하는 ONOS Northbound, Java 그리고 OSGi API에 따라 해당 퍼미션 타입을 매치하여 요구되는 퍼미션들을 추출해낸다. 퍼미션 매칭 모듈은 애플리케이션의 필요 퍼미션 리스트들을 추출해 낸 뒤 이를 퍼미션 리뷰 (Permission Reviewing Module) 에게 넘겨주며 이 후 퍼미션 파일을 작성하고 필요 퍼미션들을 개발자에게 보여주도록 한다.

2.2 ONOS-ApSM 구현

본 프레임워크는 BCEL 라이브러리 [5]를 확장하여 구현할 수 있을 것으로 기대된다. BCEL 라이브러리는 바이트코드 엔지니어링 라이브러리로써, 자바 바이트 코드를 읽어 정적분석 할 수 있도록 해준다. 본 프레임워크에서는 각 API에 따른 퍼미션을 대조하여 미리 매핑 테이블을 작성한 뒤에, 이후 바이트 코드를 분석하면서 정규표현식으로 중요한 API가 매칭되면, 그에 맞는 퍼미션 타입을 매핑 테이블을 참조하여 추출해 내도록 한다.

2.2 검증 (Plan)

본 절에서는 본 프레임워크를 이용하여 실제로 ONOS 애플리케이션을 분석하고 퍼미션을 추출하는 유즈케이스를 보이며, 각 ONOS 애플리케이션 마다 퍼미션 추출하는 시간을 측정하여 성능을 검증하도록 한다. 또한, 본 프레임워크가 각 중요 API에 따른 퍼미션을 얼마나 정확하게 추출해내는 지에 대해 검증하도록 한다.

2.2.1 유즈케이스

본 항목에서는 ONOS-ApSM를 이용하는 워킹 시나리오를 보이고, 그림 6과 같이 각 애플리케이션에서 어떤 퍼미션을 추출하였는지 정리하도록 한다.

Type	Applications	Description	Extracted Permissions
Benign	bgprouter	BGP router application	*TBD*
	calendar	Simple caledaring REST interface for intents	*TBD*
	election	Leadership election test app	*TBD*
	fwd	Simple reactive forwarding app	*TBD*
	...		
Malicious	deviceRemover	Simple device information remover app	*TBD*
	fakeLinkInjector	Simple fake link information injector app	*TBD*
	sysCmdExecutor	Simple system command execution app	*TBD*
	...		

표 1 애플리케이션별 추출된 퍼미션

2.2.2 성능

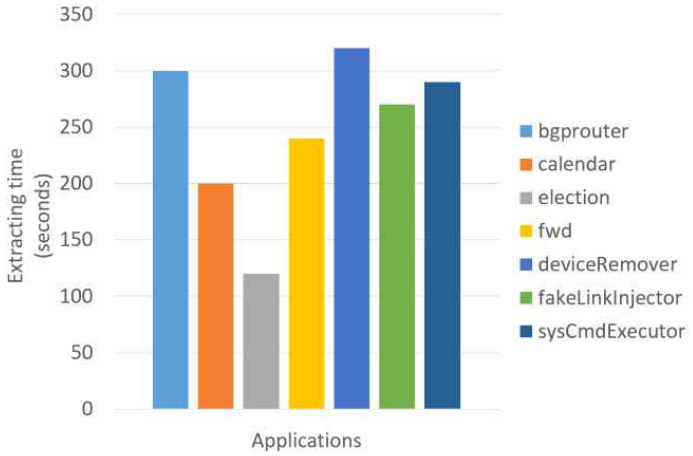


그림 7 애플리케이션별 퍼미션 추출 시간 (예제)

본 항목에서는 프레임워크가 각 애플리케이션의 퍼미션을 추출할 때에 소요되는 시간이 얼마나 되는지에 관해서 측정하여 그림 7과 같은 결과로 성능을 평가하도록 한다.

2.2.3 정확성

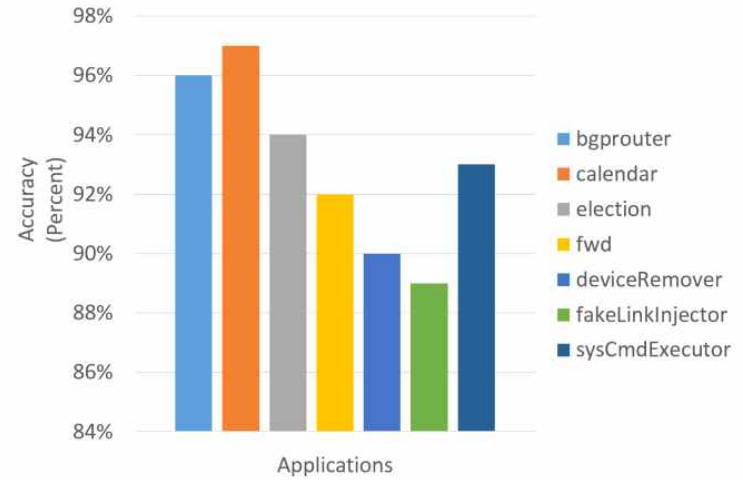


그림 8 애플리케이션별 퍼미션 추출 정확도 (예제)

본 항목에서는 프레임워크의 자동 퍼미션 추출이 사람이 직접 보고 수동으로 하는 것에 비해서 얼마나 정확한지에 대해 그림 8 과 같이 평가하도록 한다.

3. 결론

SDN 컨트롤러의 취약점이 발견됨에 따라 윈도우 및 리눅스 같은 운영체제처럼 NOS 의 보안성에 관한 연구는 필수불가결하다는 것이 입증 되었다. SM ONOS는 안전한 NOS를 만들기 위한 보안 연구의 시작 중 하나이며, ONOS API에 따른 퍼미션 모델 정의 및 런타임 API 감시를 수행하여 안전한 ONOS를 만들도록 하였다. 현재 한계점 중 하나는 애플리케이션 개발자가 직접 수동으로 퍼미션 파일을 정의해줘야 한다는 것으로, 이는 곧 실수를 유발하거나 비용을 초래할 수 밖에 없었다.

본 기술문서에서는 SM ONOS의 기능을 확장하여 ONOS 애플리케이션을 정적 분석하여 자동으로 퍼미션을 추출해주는 프레임워크를 제안하였다. 본 프레임워크는 현재 구현 및 검증이 진행 중이며 결과에 따라 본 기술문서는 지속적으로 수정 및 추가 될 계획이다.

References

[1] HP SDN App Store, https://saas.hpe.com/marketplace/sdn
[2] FloodLight. Open sdn controller. http://floodlight.openflowhub.org
[3] OpenDayLight, https://www.opendaylight.org/
[4] Bcel. https://commons.apache.org/proper/commons-bcel/
[5] https://wiki.onosproject.org/display/ONOS/Security-Mode+ONOS
[6] CORD: Central Office Reimagined as a Datacenter, https://wiki.onosproject.org/pages/viewpage.action?pageId=3441030
[7] Openflow Switch Specification 1.3.0, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf
[8] Berde, Pankaj, et al. "ONOS: towards an open, distributed SDN OS." Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014.
[9] S. Lee, C. Yoon, and S. Shin. The smaller, the shrewder: A simple malicious application can kill an entire sdn environment. In Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, pages 23–28. ACM, 2016.
[10] Hong, Sungmin, et al. "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures." NDSS. 2015.
[11] Shin, Seungwon, et al. "Rosemary: A robust, secure, and high-performance network operating system." Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. ACM, 2014.
[12] D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.-P. Wu. Droidmat: Android malware detection through manifest and api calls tracing. In Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on, pages 62.69. IEEE, 2012
[13] Bartel, Alexandre, et al. "Automatically securing permission-based software by reducing the attack surface: An application to android." Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. ACM, 2012.

K-ONE 기술 문서

- 이 문서의 기술적인 내용은 프로젝트의 진행과 함께 별도의 예고 없이 변경될 수 있습니다.

- 본 문서와 관련된 문의 사항은 아래의 정보를 참조하시길 바랍니다.
  (Homepage: http://opennetworking.kr/projects/k-one-collaboration-project/wiki, E-mail: k1@opennetworking.kr)

작성기관: K-ONE Consortium
작성년월: 2016/04