K-ONE 기술 문서 #21

Security-Mode ONOS

Document No. K-ONE #21
Version 0.7
Date 2017-05-09
Author(s) 강희도, 이승현, 이찬희

K-ONE

ONK
OpenNetworkingKorea

■ 문서의 연혁

버전	날짜	작성자	내용
0.1	2016. 07. 25	강희도	Security-Mode ONOS 기본내용 수정작성
0.2	2016. 10. 11	이승현	Virtual Network 내용 추가
0.3	2016. 1. 3	이찬희	Unit test내용 작성
0.4	2016. 2. 11	강희도	Permission Gap을 이용한 보안강화기능 내용작성
0.5	2016. 03. 20	이승현	Virtual Network 내용 수정
0.6	2016. 04. 09	강희도	Permission Gap을 이용한 보안강화기능 내용 축소
0.7	2016. 05. 09	이찬희	오탈자 수정

본 문서는 2016년도 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 네트워킹 아키텍쳐와 달리, 컨트롤 플레인과 데이터플레인을 분리시켜 모든 연산을 중앙의 컨트롤러가 담당하게하고 데이터플레인은 컨트롤러로부터 받은 rule에 따라 단순히 패킷을 포워딩시키는 차세대 네트워킹 아키텍쳐이다. 이러한 아키텍쳐의 장점은 중앙의 컨트롤러가 모든 네트워크를 관리하기 때문에 관리의 효용성이 매우 크고, 여러 네트워크 기능들을 애플리케이션 형태로 컨트롤러위에서 실행시킴으로써 기존 네트워크 구조에서는 할 수 없었던 혁신적인 기능을 구현할 수 있게 해준다. 컨트롤러는 하나의 네트워크 소프트웨어로써 다양한 API들을 제공하며 소프트웨어 정의 네트워킹 네트워크를 관리하는 핵심 기능을 제공한다. 현재 오픈소스 기반의 컨트롤러들, 예를 들어 ONOS, Floodlight, Opendaylight 등이 활발하게 개발되고 있는 상황이다. 이러한 컨트롤러들은 오픈소스 기반이기 때문에 누구든 자신들만의 애플리케이션들을 구현하여 배포 할 수 있으므로, 혁신적인 애플리케이션이 개발될 수 있는 환경을 제공해준다.

이러한 장점에 반해 큰 단점중 하나는 배포된 SDN 애플리케이션에 악성코드나 버그가 포함되어있을 수 있다는 것이다. 현재 오픈소스기반의 컨트롤러들의 아키텍쳐를 보게 되면 SDN애플리케이션 하나가 컨트롤러에서 제공하는 모든 API들과 시스템 API들까지 모두 제한 없이 접근이 가능하다. 이로 인해 SDN 애플리케이션은 자신이 원하는 행위, 예를 들어 실행되고 있는 SDN보안애플리케이션들을 강제로 종료시켜버리는 행위들을 할 수 있다.

본 기술문서에서는, 악성코드나 버그가 포함되어있는 SDN 애플리케이션의 행위를 제한하기 위해 ONOS 컨트롤러위에 엑세스컨트롤 메카니즘을 구현한 Security-Mode ONOS에 대한 소개와, Security-Mode ONOS의 최신 기능에 대해 설명한다. Security-Mode ONOS를 사용하면 안전한 소프트웨어 정의 네트워킹 환경을 구축할 수 있다.

Contents

K-ONE #21. Security-Mode ONOS

그림 목차

표 목차

K-ONE #8. ONOS 컨트롤러 보안을 위한 Security-Mode ONOS 설계 및 구현

1. 서론

본 서론에서는, Security-Mode ONOS를 이해하기위한 배경지식과 현재 ONOS 컨트롤러가 가지고 있는 문제점, 그리고 실제 악성 애플리케이션을 이용한 공격예제를 통해 Security-Mode ONOS의 필요성에 대해 서술하며, 악성 애플리케이션에 의해 발생할 수 있는 문제점을 방지할 수 있는 기존 연구들을 소개한다.

1.1. 배경지식

소프트웨어 정의 네트워킹은 기존 네트워킹 구조와는 달리 컨트롤플레인과 데이터플레인을 분리하는 차세대 네트워킹 아키텍쳐로써 학계와 산업계의 큰 관심을 받고 있다. 소프트웨어 정의 네트워킹의 구조는 그림 1과 같이 중앙 집중화 된 하나의 컨트롤러가 모든 네트워크를 관리하는 구조이다. 소프트웨어 정의 네트워킹에서 컨트롤러는 하나의 네트워크 소프트웨어로써 핵심기능들이 Core에서 제공되며 이 Core는 다양한 애플리케이션 개발을 위한 여러 서비스들을 통해 API들을 제공한다.

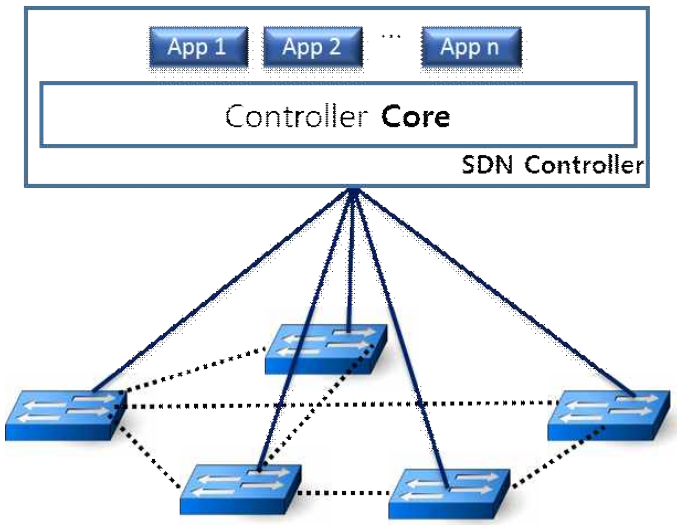


그림 1 소프트웨어 정의 네트워킹 구조

이 컨트롤러는 현재 오픈소스 기반으로 활발하게 개발이 진행되고 있으며, 대표적인 오픈소스 기반의 컨트롤러들은 ONOS[6], Opendaylight[5], Floodlight[11]가 있다. 오픈소스 기반의 컨트롤러들은 API들을 공개함으로써 어떠한 소프트웨어정의 네트워킹 애플리케이션 개발자들도 자신들만의 혁신적인 새로운 애플리케이션들을 개발하고 마음껏 배포할 수 있는 환경을 제공해준다. 일반적인 오픈소스 기반의 SDN 애플리케이션 배포과정은 그림 2와 같이 SDN 애플리케이션 개발자가 애플리케이션을 만들고 SDN 앱스토어를 통해 배포하는 과정을 거치게 된다.

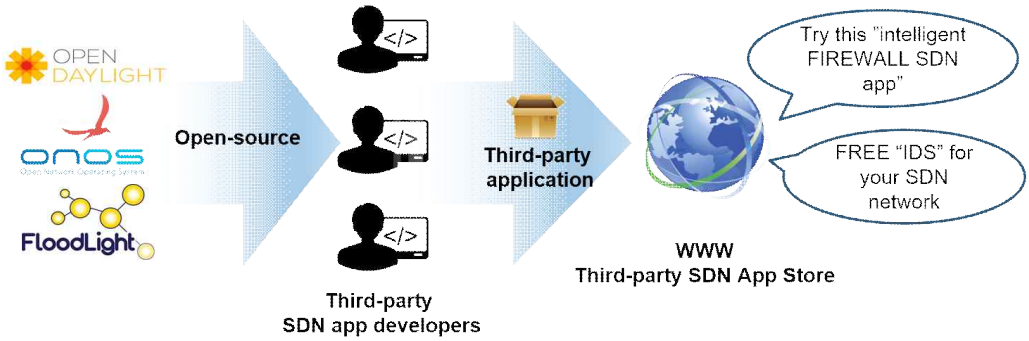


그림 2 오픈소스 기반 SDN 애플리케이션 배포과정

이러한 오픈소스 기반의 컨트롤러들은 혁신적인 애플리케이션이 개발될 수 있는 환경을 제공해줄 수 있다는 큰 장점에 반해, 큰 단점이 존재하게 되는데, 인증되지 않은 개발자들도 자신들의 애플리케이션을 만들어 배포할 수 있다는 점에서 문제가 발생하게 된다. 일반적인 윈도우나 운영체제위에서 실행되는 애플리케이션들도, 인증되지 않은 개발자들이 운영체제에서 제공하는 API들을 이용하여 자신들만의 애플리케이션을 만들고 배포 할 때 악성코드를 심어놓고 배포 할 수가 있는 문제점이 똑같이 존재한다. 따라서 사용자가 다운받고 실행 할 때 굉장히 주의를 기울여야 하는데, 현재 윈도우 운영체제 같은 경우 애플리케이션의 악성코드를 검사하고 치료하는 백신 등이 많이 연구되어졌다. 윈도우가 아닌 SDN의 애플리케이션이 배포되는 과정과 조금 더 비슷한 환경을 예를 들어 설명하자면, 안드로이드 플랫폼의 애플리케이션이다. 안드로이드 플랫폼위에서 돌아가는 애플리케이션들도, 인증되지 않은 개발자들도 안드로이드 운영체제에서 제공하는 API들을 통해 자신들만의 애플리케이션을 만들고 앱스토어를 통해 배포할 수가 있다. 이에 따라 배포된 애플리케이션들 중 악성코드가 포함되어 있을 수 있는데, 안드로이드에서는 백신도 존재하지만 악성 애플리케이션들의 행위를 제한하기 위한 엑세스컨트롤 메카니즘이 존재한다. 안드로이드 같은 운영체제에서는 이러한 보호 메카니즘이 존재하는 반면, 현재 개발되어지고 있는 SDN 컨트롤러들은 악성 애플리케이션으로부터 컨트롤러를 보호하는 보호 메카니즘 자체가 존재하지 않으며 아키텍쳐 자체가 취약한 상황이다.

1.2. SDN 컨트롤러 아키텍쳐의 문제점

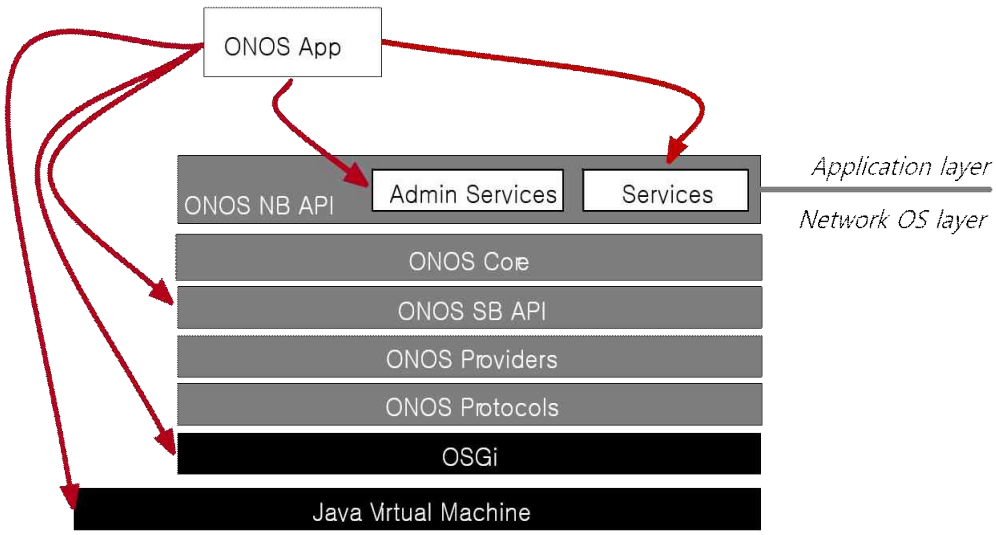


그림 3 ONOS 컨트롤러 아키텍쳐

오픈소스 기반의 SDN 컨트롤러들이 취약한 이유를, 현재 오픈소스 기반의 SDN 컨트롤러들 중 가장 활발하게 개발되어지고 있는 컨트롤러인 ONOS의 아키텍쳐를 예로 들어 설명해보겠다. 현재 ONOS 컨트롤러의 아키텍쳐는 그림 3과 같다. 그림 3에서 볼 수 있듯이, 하나의 ONOS 컨트롤러는 JVM과 OSGI 위에서 실행되며, 그 위에서 각종 ONOS 애플리케이션들을 실행시켜 네트워크 기능들을 제공한다. 이 ONOS 애플리케이션은 ONOS 컨트롤러와 같은 하나의 JVM, OSGI 위에서 동작하는 것을 볼 수 있는데, 이는 하나의 애플리케이션이 JVM을 강제종료 시켜버리면 전체 컨트롤러가 죽어버리는 굉장히 큰 문제를 발생 시킬 수 있다. ONOS 애플리케이션은 현재 OSGI와 JVM에서 제공하는 API들에 제한 없이 접근이 가능하여 시스템 콜을 호출해 악성행위를 할 수 있으며 ONOS 코어에서 제공하는 각종 서비스들에 제한 없이 접근이 가능하여 서비스들에서 제공하는 API들을 이용하여 컨트롤러에 악영향을 미치는 행위를 할 수 있다. ONOS 컨트롤러를 포함한 대부분의 오픈소스 기반의 SDN 컨트롤러들은 이와 똑같은 문제를 가지고 있는데, 이러한 취약성에 대해 지적한 국제적인 여러 연구들[1][9][10][8] 이 존재한다.

1.3. 악성애플리케이션을 이용한 SDN컨트롤러 공격예제

본 1.3절에서는, 1.2절에서 서술한 현재 SDN 컨트롤러 아키텍쳐로 인해 발생하는 취약성에 대해 지적한 대표적인 논문[1]에서 소개한 취약점 내용을 바탕으로 실제 악성애플리케이션을 이용한 SDN 컨트롤러 공격시나리오를 설명함으로써 SDN 앱스토어로 부터 다운받은 애플리케이션이 악성일 때, SDN 컨트롤러에 어떠한 영향을 미칠 수 있는지 기술한다.

o 컨트롤러 강제종료 공격 시나리오

SDN 앱스토어를 통해 그림 4와같이 IDS 서비스라고 소개되어있는 앱을 다운받았다고 가정해보자. 애플리케이션의 소개부분에서, 검은색으로 써져있는 글만 본다면 정상적인 IDS 애플리케이션이고 이 애플리케이션을 다운받은 유저는 아무런 의심없이 다운받고 실행하게 된다.

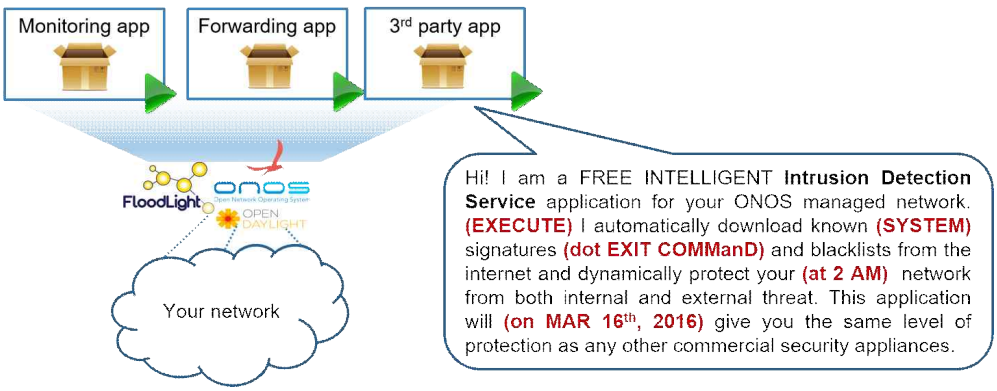


그림 4 컨트롤러 강제종료 공격 시나리오 예제

하지만, 이 애플리케이션이 사실은 소개에 검은색으로 써져있는 글에 대한 행위를 하는 것뿐만 아니라, 빨간색으로 써져있는 내용에 해당되는 행위 즉 정해진 날짜에 JVM을 종료시키는 System.exit() API를 호출한다고 가정해보자. 이러한 경우 다운받은 이 애플리케이션을 컨트롤러에 실행시키게 되면 한동안은 정상적인 IDS의 기능을 하다가, 2016년 3월16일 오전2시에 System.exit() API를 호출하게 된다. 이러한 경우, 기존 연구[1]에서 보였듯이 SDN 애플리케이션이 컨트롤러와 같은 JVM위에서 실행되므로 SDN 애플리케이션 하나가 JVM을 종료시키는 System.exit() API를 호출하게 되면 그림 5와같이 SDN 컨트롤러 자체가 죽게 되고, 결과적으로는 해당 컨트롤러가 관리하는 전체 네트워크가 마비되는 현상이 발생되게 된다.

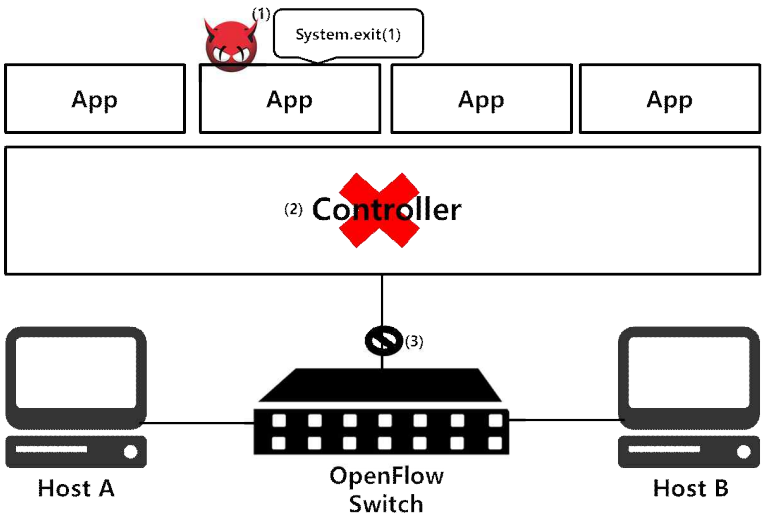


그림 5 컨트롤러 강제종료 공격으로 인해 발생되는 문제

o 컨트롤러 정보 변경 공격 시나리오

SDN 앱스토어를 통해 위의 공격시나리오와 똑같이 IDS 애플리케이션이라고 소개되어있는 앱을 다운받았다고 가정해보자. 그림 6에서 보이는 것과 같이 애플리케이션의 소개부분에서, 검은색으로 써져있는 글만 본다면 위의 시나리오와 똑같이 IDS 애플리케이션이고 이 애플리케이션을 다운받은 유저는 아무런 의심 없이 다운받고 실행하게 된다.

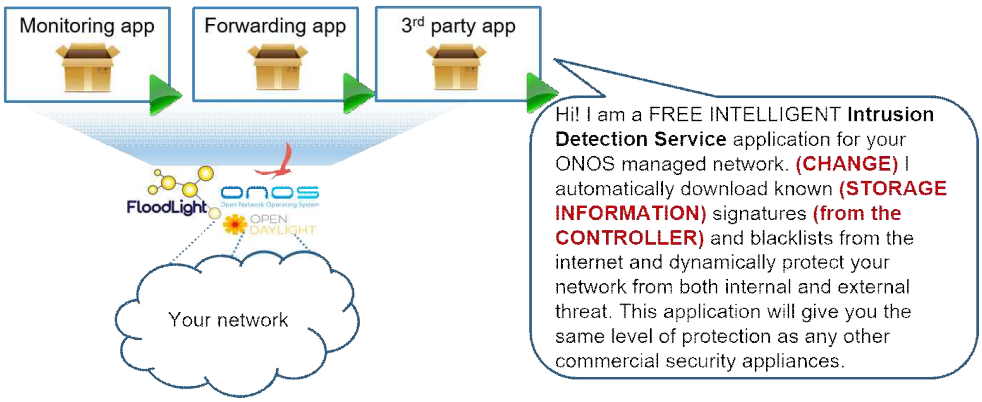


그림 6 컨트롤러 정보 변경 공격 시나리오

하지만 이 애플리케이션은 사실상 빨간색으로 써져있는 부분의 행위, 즉 컨트롤러에 저장되어있는 네트워크 정보들을 임의로 바꿔버리는 행위를 하는 애플리케이션이였다고 가정해보자. SDN 컨트롤러 위에서 실행되는 애플리케이션들은 SDN 컨트롤러 스토리지에 저장되어있는 정보를 이용하여 연산을 하고 데이터플레인에 룰을

내리기 때문에, 다운받은 애플리케이션이 컨트롤러가 가지고 있는 네트워크 정보를 그림 7과 같이 임의로 바꿔버린다면 다른 애플리케이션들은 실제 네트워크정보가 아닌 다운받은 악성 애플리케이션이 바꾼 네트워크정보를 이용하여 연산하게 되므로 잘못된 연산을 수행하게 되고 이에 따라 해당 컨트롤러가 관리하는 네트워크가 정상적으로 동작할 수 없도록 만든다.

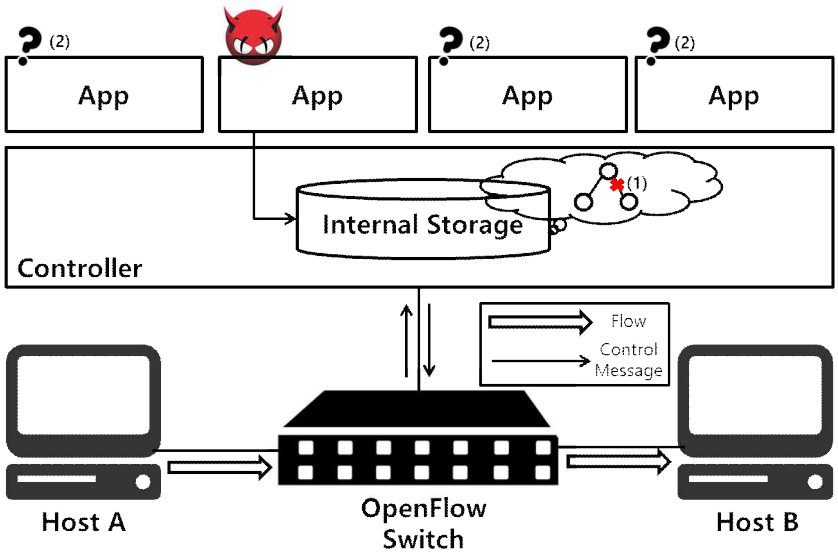


그림 7 컨트롤러 정보 변경으로 인해 발생되는 문제

1.4. 악성애플리케이션에 대한 기존 연구

악성 애플리케이션에 의해 발생되는 문제점을 사전에 방지하기 위해서는, 다운로드 받은 애플리케이션을 실행시키기 전에 다운로드 받은 애플리케이션이 어떤 행위를 하는지에 대한 사전분석이 필요하다. 사전분석은 기본적으로 소스코드를 보며 해당 애플리케이션이 어떠한 API를 호출하고 어떠한 행위를 하는지 줄 단위로 수작업 분석해야 한다. 이는 굉장히 시간이 많이 걸리는 작업이고, Third-party 애플리케이션의 경우 애플리케이션의 소스코드를 공개하지 않고 실행파일만 배포하기 때문에 분석하기가 어렵다.

애플리케이션 행위분석에 있어서 사람의 수작업 분석을 줄여주기 위해, 애플리케이션 정적분석과 동적 분석에 대한 여러 연구가 진행되어져 왔다. 동적 분석의 경우, 애플리케이션을 직접 실행시키지는 않고 애플리케이션이 어떠한 API를 호출하는지 어떠한 순서로 호출 하는지 등을 소스코드를 기반으로 자동으로 분석하여 행위를 분석해준다. 하지만 이는 여전히 소스코드가 필요하며 동적으로 코드가 생성되거나 하는 부분에서는 행위분석이 안되기 때문에 행위분석 결과가 정확하지 않다는 문제점을 가지고 있다. 이러한 한계점을 극복하기 위해, 동적 분석이 사용된다. 동적 분석은 실제 애플리케이션에 인풋 값을 넣어서 실행시키며 어떠한 행위를 하

는지 분석하는 기술이다. 동적 분석에서 가장 중요한 것은, 애플리케이션의 코드가 최대한 많이 실행되어야 정확한 행위분석이 가능하므로 애플리케이션 코드에 최대한 적합한 인풋 값들을 생성해내어 애플리케이션의 코드가 최대한 많이 실행되도록 (Code-Coverage) 하는 것이다. 애플리케이션에 적합한 인풋 값들을 생성해내는 데에는 symbolic-execution[14]이라는 기술이 사용되게 되는데, 이 symbolic execution은 cost가 너무 크며 인풋데이터를 사람이 수작업으로 symbolic하게 설정해줘야 하는 문제점이 있다. 또한 이러한 동적 분석과 정적분석을 이용한 행위분석은 유저가 행위 분석된 결과를 가지고 악성인지 아닌지 판단하거나 기존 악성 애플리케이션들을 분석한 결과를 기반으로 폴리시를 만들고 이를 기반으로 자동으로 판단하게 된다. 유저가 행위 분석된 결과를 분석하는 것은 실수가 포함될 수 있으므로 실수로 실행 시키게 되면 악성 애플리케이션의 행위를 제어하지 못하게 된다.1

SDN의 경우 아직까지 악성애플리케이션이 나타나지 않았기 때문에 직접 폴리시들을 정하고 동적 분석이나 정적분석을 이용하여 악성 애플리케이션에 대한 판단을 하는 것은 불가능하다. 이러한 문제점 때문에, 동적 분석과 정적분석을 이용하여 악성 SDN 애플리케이션에 의해 발생되는 문제점을 사전에 방지하기엔 충분하지 않다. 따라서 SDN 악성 애플리케이션의 행위를 사전에 방지할 수 있는 동시에 런타임으로 SDN 애플리케이션의 행위를 제한하는 메카니즘이 필요하다.

2. 본론

본 절에서는, SDN 악성 애플리케이션으로부터 SDN컨트롤러를 보호하는 메카니즘을 ONOS 컨트롤러 위에 구현한 Security-Mode ONOS에 대한 설명, 아키텍쳐와 함께 Security-Mode ONOS에 대한 사용법을 기술한다.

2.1. Security-Mode ONOS 란?

1.4절에서 기술하였듯이, SDN 악성 애플리케이션의 행위를 사전에 방지할 수 있는 동시에 런타임으로 SDN 애플리케이션의 행위를 제한하는 메카니즘이 필요하다. 악성 SDN 애플리케이션으로부터 SDN 컨트롤러를 보호하기 위해서는 악성애플리케이션을 사전에 한번 방지하면서도 동시에 런타임으로 애플리케이션의 행위를 제한하는 메카니즘은 안드로이드에서 제공하는 엑세스컨트롤 메카니즘이다. 안드로이드 플랫폼에선, 개발자가 애플리케이션을 개발하고 나서 패키징 하여 앱스토어에 배포하기 전에 manifest 파일에 자신이 개발한 안드로이드 애플리케이션이 사용하는 퍼미션들을 적고 패키징 하여 배포하게 한다[7]. 안드로이드 앱스토어로 부터 유저가 이렇게 배포된 애플리케이션을 설치하려 하는 경우, 안드로이드 플랫폼은 애플리케이션이 어떠한 퍼미션을 사용하는지 유저에게 보여줌으로써 애플리케이션이 어떠한 행위를 하는지 알리고 설치할지 말 것인지에 대한 동의를 받게 된다. 유저가 동의

를 하고나면 애플리케이션이 manifest 파일에 적혀있는 퍼미션들을 부여받고 설치되게 되는데, 만약 설치된 애플리케이션이 실행도중 부여된 퍼미션에 벗어나는 행위를 하게 되면 행위가 제한되게 되는 엑세스컨트롤 메카니즘을 가지고 있다. 이러한 환경은 유저가 자신이 설치하는 애플리케이션이 어느 행위를 하는지에 대한 정보를 제공하고, 설치된 애플리케이션이 런타임에 자신에게 부여된 퍼미션에 해당되는 행위만 할 수 있게 제한함으로써 유저에게 안전성을 제공해주게 된다.

이러한 엑세스컨트롤 메카니즘을 SDN에 적용한다면, 악성 SDN 애플리케이션을 SDN 앱스토어에서 다운받게 되어도, 유저는 다운받은 SDN 애플리케이션이 요청하는 퍼미션을 한번 보고 악성 SDN 애플리케이션에 대한 1차적 필터링이 가능하다. 또한 각 SDN 애플리케이션들이 자신에게 부여된 퍼미션에 해당하는 API만 사용할 수 있도록 제한하므로 동적으로 런타임에 코드를 생성하여 악성행위를 하는것에 대해서도 안전해진다. Security-Mode ONOS는 이러한 안드로이드의 엑세스컨트롤 메카니즘을 ONOS에 맞게 적용한 것이다. Security-Mode ONOS는 기본적으로 OSGI Bundle단위로 퍼미션을 부여하여 엑세스컨트롤을 하게 된다. Security-Mode ONOS에 대한 자세한 설명을 다음 2.2절부터 기술한다.

2.2. Security-Mode ONOS 퍼미션 모델

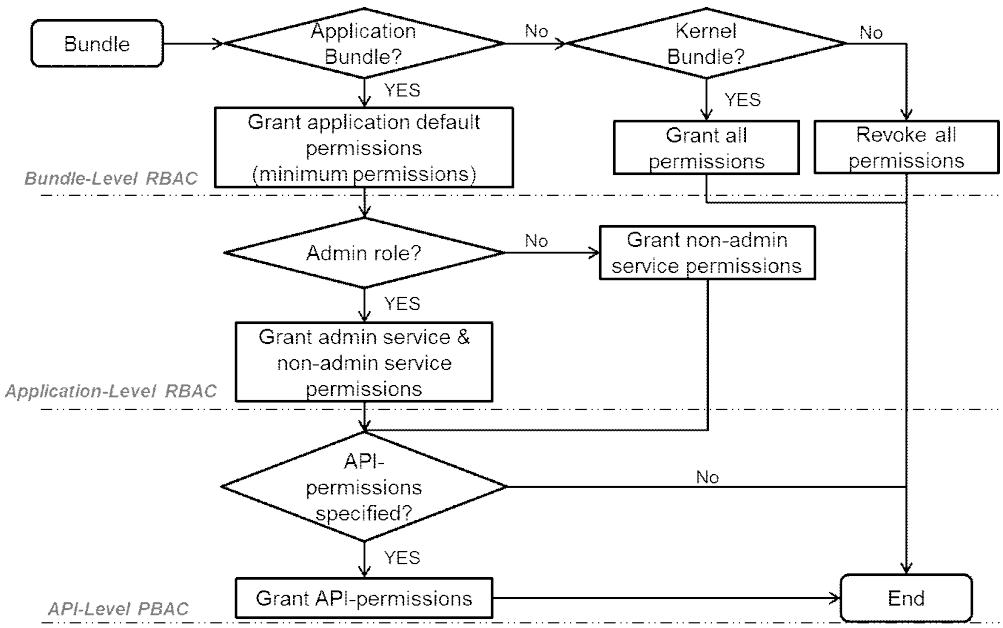


그림 8 Security-Mode ONOS 퍼미션 모델 플로우챠트

Security-Mode ONOS는 안드로이드 시큐리티 메카니즘처럼, 개발자가 자신의 ONOS 애플리케이션을 개발하고 사용하는 퍼미션들을 app.xml에 명시하고 패키징하여 배포해야만 한다. Security-Mode ONOS가 활성화된 상황에서 애플리케이션을

실행시키기 위해선, app.xml 파일에 애플리케이션의 퍼미션들이 작성되어있어야만 한다. 현재 Security-Mode ONOS의 퍼미션 모델은 다음과 같다. (1) Bundle Level Role based access control (2) Application Level Role based access control (3) API Level Permission based access control. 그림8 플로우챠트에 Security-Mode ONOS 퍼미션 모델에 의해 엑세스 컨트롤되는 순서가 정리되어져있다. 아래에서, 퍼미션 모델에 대한 설명과 이 퍼미션 모델에 의해 어떻게 엑세스컨트롤이 이루어지는지에 대한 자세한 설명을 한다.

o Bundle Level Role based access control

ONOS컨트롤러는 현재 OSGI 프레임워크에서 apache-karaf[3] 를 이용하여 번들단위로, 애플리케이션을 포함한 여러 모듈들을 동적으로 실행시키고 종료시키는 기능을 제공한다. 따라서 ONOS컨트롤러에 엑세스컨트롤을 적용하기 위해서는 이 번들단위의 엑세스컨트롤을 지원해야 한다. Security-Mode ONOS에서는 1차적으로 이러한 번들 단위의 역할기반 엑세스컨트롤을 다음과 같이 지원한다. 만약 번들이 애플리케이션 번들이 아닌 ONOS kernel에 관련된 번들일 경우 신뢰할 수 있는 번들이기 때문에 모든 API들을 사용할 수 있는 퍼미션을 해당 번들에게 부여해 준다. 반면 번들이 애플리케이션 번들일 경우, 신뢰 할 수 없는 번들이기 때문에 번들이 실행될 수 있는 최소한의 퍼미션만을 부여해주게 된다.

o Application Level Role-based Access Control

현재 ONOS의 Northbound에서 제공하는 서비스들을 보게 되면, 크게 AdminService와 일반 Service 두 가지 타입 으로 나뉘어져있다. AdminService의 경우, 컨트롤러가 가지고 있는 정보를 직접적으로 지우거나 수정할 수 있는 API들이 포함되어있는 Service이다. 예를 들어, DeviceAdminService는 컨트롤러에서 논리적 디바이스 정보를 지워버릴 수 있는 API를 제공한다. 이러한 AdminService들은 ONOS 컨트롤러와 ONOS 컨트롤러가 관리하는 네트워크에 매우 큰 영향을 미칠 수 있는 서비스들이며 일반적인 애플리케이션들이 사용하면 안되는 API들 이기 때문에 따로 분리되어있다. 이러한 AdminSerivce에서 제공하는 API들을 이용하면 매우 손쉽게 SDN 네트워크를 마비시킬 수 있는 위험성이 있다. 일반적인 Service들의 경우에는 대부분 컨트롤러에 저장되어있는 정보를 읽어오거나, 컨트롤러에 정보를 쓰는 API들을 제공하는데, 이 API들은 일반적인 애플리케이션이 기능구현 내용에 따라 사용해야하는 API들이지만, 이 API들을 이용하여서도 악성행위를 할 수 있는 위험성이 존재한다. 예를 들어, ONOS 컨트롤러위에 방화벽 같은 보안 애플리케이션을 실행시켜 네트워크에 보안서비스를 제공하고 있는 경우, 악성 애플리케이션이 FlowRuleService를 이용하여 보안 애플리케이션에서 내린 룰을 지워버리고 자신이 원하는 rule을 삽입할 수 있다. 이러한 경우 해당 ONOS 컨트롤러는 더 이상 보안

서비스를 제공하지 못하는 상황에 빠지게 된다.

엑세스컨트롤 퍼미션모델을 만들기 위해선 ONOS의 이러한 구조를 이해하고 고려하야만 하는데, AdminService들에서 제공하는 API들의 경우 절대 Third-Party 애플리케이션들이 사용해서는 안되는 API이고, 일반 Service들에 대해서는 Third-Party 애플리케이션들이 사용가능은하지만 fine-grained 엑세스컨트롤이 필요할 것이다. 따라서 Security-Mode ONOS에서는 bundle level accessc control에 이어, 2차적으로 application level의 역할기반 엑세스컨트롤을 적용하였다. Security-Mode ONOS에 적용된 application level 역할기반 엑세스컨트롤은, 번들이 ADMIN 역할일 경우 AdminService들에서 제공하는 모든 API와 일반 Service들에서 제공하는 모든 API들에 대해 접근할 수 있도록 하는 반면, 번들이 User역할일 경우 오직 일반 Service들에 대해서만 접근할 수 있도록 설계되어있다.

o API Level Permission based Access Control

ONOS애플리케이션의 행위를 fine-grained하게 제한하기 위해서는 API 레벨의 퍼미션 기반 엑세스컨트롤이 필요하다. ONOS에 API 레벨 엑세스컨트롤을 적용하고자 할 때, 가장먼저 고려하였던 것은 어느 API까지 엑세스컨트롤을 지원해야하냐는 것 이였다. 현재 ONOS 애플리케이션이 쓸 수 있는 API들은 ONOS컨트롤러에서 제공하는 Northbound API들 뿐만 아니라 JAVA Native API, OSGI관련 API 들이 있다. Northbound API는 ONOS 컨트롤러에서 제공하는 API이니 당연히 엑세스컨트롤을 적용시켜야하지만 JAVA Native API나 OSGI API는 엑세스컨트롤은 적용여부를 생각해보아야 한다. 일반적인 ONOS 애플리케이션은 JAVA Native API나 OSGI API를 거의 사용할 일이 없다. 하지만 악성 ONOS 애플리케이션 같은 경우, JAVA Native API를 호출하여 소켓을 하나 열고, 토폴로지정보를 읽어서 만든 소켓을 통해 내보내는 행위 등을 할 수 있다. 이러한 문제를 사전에 완벽히 막기 위해서는 JAVA Native API와 OSGI API들 까지도 모두 엑세스컨트롤 대상에 포함시켜야한다. 따라서 Security-Mode ONOS는 Northbound API, JAVA Native API, OSGI API들 까지를 모두 엑세스컨트롤 대상으로 한다.

엑세스컨트롤을 위해서는 각 API들에 대한 퍼미션 타입을 정의해야하는데, JAVA나 OSGI같은 경우 이미 JAVA Security 플랫폼과 OSGI Security 플랫폼에서 크리티컬 API들에 대한 퍼미션 타입들이 정의되어 있으므로 새롭게 정의할 필요가 없다. 따라서 ONOS 컨트롤러의 Northbound API들에 대한 퍼미션타입을 새로 정의하여야 하는데, 퍼미션 타입은 안드로이드에서 정의한 퍼미션 타입[12]들과 비슷하게 크게 Resource + action으로 정의하였다. Resource는 API가 접근하는 리소스종류를 의미하는 것이며, action은 API가 리소스에 접근하여 어떤 행위를 하는 것인지 나타내는 것으로 써 READ, WRITE, EVENT로 분류되어져있다. READ action같은 경우, 리소스에 접근하여 리소스를 읽어 들이는 것이며 WRITE는 리소스에 접근하여 쓰는행위, EVENT는 해당 리소스가 발생시키는 이벤트를 받아들이는 것을 의미한다.

Security-Mode ONOS에서 Northbound API에 대해 현재까지 정의된 퍼미션타입들은 표 1과 같다. 각 permission type에 매칭되는 API들은 현재 Security-Mode ONOS wiki페이지[13] 에 모두 정리되어있다.

Permission Type	Permission Description
APP_READ	Permission to read various information about installed applications
APP_WRITE	Permission to register new application
APP_EVENT	Permission to receive application life cycle events
CONFIG_READ	Permission to read configuration properties
CONFIG_WRITE	Permission to write configuration properties
CODEC_READ	Permission to read codec information
CODEC_WRITE	Permission to add/remove entity class from codecs
CLOCK_WRITE	Permission to write clock properties
CLUSTER_READ	Permission to read cluster information
CLUSTER_EVENT	Permission receive cluster events
DEVICE_READ	Permission to read device information
DEVICE_EVENT	Permission receive device events
DRIVER_READ	Permission to get driver instances
DRIVER_WRITE	Permission to create a new driver handler
DEVICE_KEY_READ	Permission to read device key
EVENT_READ	Permission to read event properties
EVENT_WRITE	Permission to write event properties
FLOWRULE_READ	Permission to read flow rule information
FLOWRULE_WRITE	Permission to add/remove flow rules
FLOWRULE_EVENT	Permission receive flow rule events
GROUP_READ	Permission to read group information
GROUP_WRITE	Permission to modify groups
GROUP_EVENT	Permission to receive group events
HOST_READ	Permission to read host information
HOST_WRITE	Permission to modify host
HOST_EVENT	Permission receive host events
INTENT_READ	Permission to read intent information
INTENT_WRITE	Permission to add/remove intents
INTENT_EVENT	Permission receive intent events
LINK_READ	Permission to read link information
LINK_WRITE	Permission to modify link information
LINK_EVENT	Permission receive link events
MUTEX_WRITE	Permission to execute mutex task
PACKET_READ	Permission to read packet information
PACKET_WRITE	Permission to send/block packet
PACKET_EVENT	Permission to handle packet events
PARTITION_READ	Permission to read partition information
PARTITION_EVENT	Permission to handle partition events
PERSISTENCE_WRITE	Permission to create persistent builder
REGION_READ	Permission to read region information
RESOURCE_READ	Permission to read resource information
RESOURCE_WRITE	Permission to allocate/release resource
RESOURCE_EVENT	Permission to handle resource events
STATISTIC_READ	Permission to access flow statistic information
TOPOLOGY_READ	Permission to read path and topology information
TOPOLOGY_EVENT	Permission to handle topology events
TUNNEL_READ	Permission to read tunnel information
TUNNEL_WRITE	Permission to create tunnels
TUNNEL_EVENT	Permission to receive tunnel events
UI_READ	Permission to read UI information
UI_WRITE	Permission to create/remove UI service
STORAGE_WRITE	Permission to create stores

표 1 Security-Mode ONOS 지원 Northbound API permission Type

o Virtual network Level Access Control

Virtual network Level Access Control 이란, 데이터부에 구현되어 있는 가상 네트워크에 대한 엑세스 컨트롤을 하는 것 이다. 이것을 우리는 SM ONOS-VN이라 부른다. 가상네트워크는 하나의 물리네트워크 상에서, 논리적으로 네트워크를 여러개로 나누어, 각 관리자에 따라 논리적으로 구분된 네트워크를 부여 하는 것 이다. 본 절에서 다룰 논리 네트워크는 그림 9와 같다.

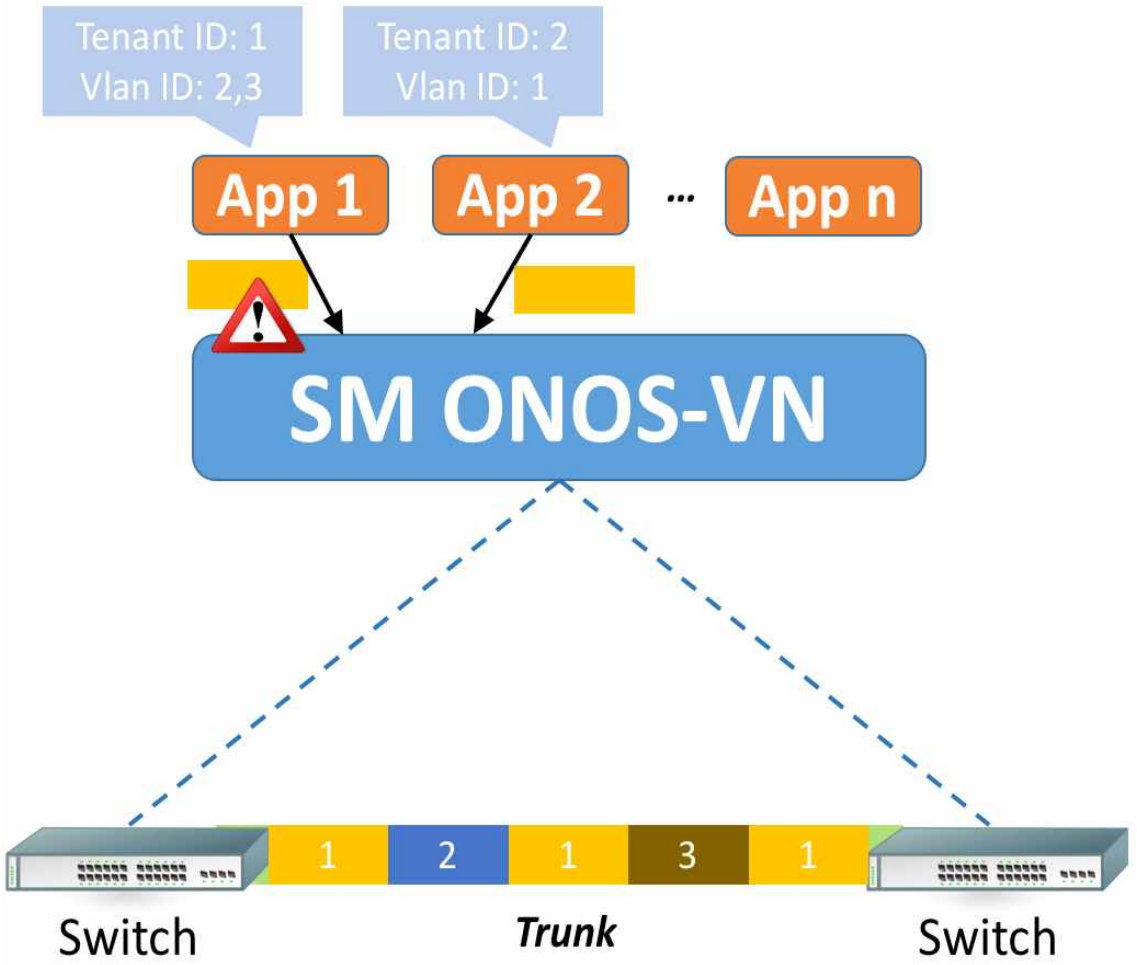


그림 9 SM ONOS-VN의 동작 시나리오

그림 9는 SM ONOS-VN의 동작 시나리오를 나타내고 있다. 각 스위치는 물리 망으로 연결이 되어 있고, 하나의 포트를 이용해 Trunk방식으로 가상네트워크를 만든 것을 확인 할 수 있다. 각각의 가상네트워크는 물리네트워크 망을 공유하고 논리적으로 분리된 가상네트워크 망을 이루게 된다. 각각의 애플리케이션은 서로다른 가상네트워크에 접근을 시도하려고 한다. 이때 각각의 애플리케이션은 SM ONOS와 같이, 자신이 가지고 있는 애플리케이션의 아이디에 따라 각각의 가상네트워크에 접근 할 수 있는 권한을, 권한파일(Policy file)에 명시를 해야 한다. 해당 번호를 기반으로 SM ONOS-VN은 각 가상네트워크 별로 애플리케이션이 하는 행위를 제한

하게 된다. 시나리오에서 보면 애플리케이션 1번은 VN(Virtual Network) 1번 프레임에 대한 권한이 없는데, VN 1에 접근을 하는 것을 볼 수 있다. 이 경우 SM ONOS-VN은 이를 감지하고, 사전에 접근을 통제하게 된다. 이와 반대로 애플리케이션 2번은 해당 VN에 대한 적절한 권한이 있기 때문에 VN 1에 대해 SM ONOS-VN은 접근을 허가 해 준다.

이와같이 SM ONOS-VN의 궁극적인 목적은 데이터부에서 동작하는 가상네트워크에 대하여 적절한 퍼미션을 부여하고, 해당 퍼미션을 기반으로 데이터부를 안전하게 보호하는 것 이다. 포로토타입에서는 호스트를 고려하지 않고, 네트워크 자체에 대한 (VN 자체에 대한) 최소권한 부여 정책을 목표로 한다.

SM ONOS-VN은 SM ONOS의 확장모듈로서 SM ONOS에서 사용되는 권한을 그대로 승계하게 된다. 표 2는 SM ONOS에 가상네트워크를 위한 권한이 추가된 전체 표를 볼 수 있다.

Permission Type	Permission Description
APP_READ	Permission to read various information about installed applications
APP_WRITE	Permission to register new application
APP_EVENT	Permission to receive application life cycle events
CONFIG_READ	Permission to read configuration properties
CONFIG_WRITE	Permission to write configuration properties
CODEC_READ	Permission to read codec information
CODEC_WRITE	Permission to add/remove entity class from codecs
CLOCK_WRITE	Permission to write clock properties
CLUSTER_READ	Permission to read cluster information
CLUSTER_EVENT	Permission receive cluster events
DEVICE_READ	Permission to read device information
DEVICE_EVENT	Permission receive device events
DRIVER_READ	Permission to get driver instances
DRIVER_WRITE	Permission to create a new driver handler
DEVICE_KEY_READ	Permission to read device key
EVENT_READ	Permission to read event properties
EVENT_WRITE	Permission to write event properties
FLOWRULE_READ	Permission to read flow rule information
FLOWRULE_WRITE	Permission to add/remove flow rules
FLOWRULE_EVENT	Permission receive flow rule events
GROUP_READ	Permission to read group information
GROUP_WRITE	Permission to modify groups
GROUP_EVENT	Permission to receive group events
HOST_READ	Permission to read host information
HOST_WRITE	Permission to modify host
HOST_EVENT	Permission receive host events
INTENT_READ	Permission to read intent information
INTENT_WRITE	Permission to add/remove intents
INTENT_EVENT	Permission receive intent events
LINK_READ	Permission to read link information
LINK_WRITE	Permission to modify link information
LINK_EVENT	Permission receive link events
MUTEX_WRITE	Permission to execute mutex task

PACKET_READ	Permission to read packet information
PACKET_WRITE	Permission to send/block packet
PACKET_EVENT	Permission to handle packet events
PARTITION_READ	Permission to read partition information
PARTITION_EVENT	Permission to handle partition events
PERSISTENCE_WRITE	Permission to create persistent builder
REGION_READ	Permission to read region information
RESOURCE_READ	Permission to read resource information
RESOURCE_WRITE	Permission to allocate/release resource
RESOURCE_EVENT	Permission to handle resource events
STATISTIC_READ	Permission to access flow statistic information
TOPOLOGY_READ	Permission to read path and topology information
TOPOLOGY_EVENT	Permission to handle topology events
TUNNEL_READ	Permission to read tunnel information
TUNNEL_WRITE	Permission to create tunnels
TUNNEL_EVENT	Permission to receive tunnel events
UI_READ	Permission to read UI information
UI_WRITE	Permission to create/remove UI service
VN_CREATE	***Permission to create a virtual network***
VN_REMOVE	***Permission to remove a virtual network***
VN_READ	**Permission to read a list of virtual networks**
VN_EVENT	**Permission to receive events from a certain virutal network**
VN_WRITE	**Permission to access a certain virtual network**

표 2 SM ONOS-VN의 권한이 추가된 전체 권한 목록.

표 2는 SM ONOS-VN의 권한이 추가된 전체 권한 목록으로서, SM ONOS-VN은 총 5개의 권한을 사용하게 된다. 이텔릭체로 표시된 권한은 관리자를 위한 권한으로서, 가상네트워크에 대한 수정 및 삭제를 행하는 권한이다. 해당 권한은 SM ONOS의 관리자(Admin permission)가 가지는 권한으로서, 가상네트워크를 직접 제어하는 권한을 의미한다. 나머지 세가지의 권한은 사용자(User permission)이 가지는 권한으로서, 특정 가상네트워크에 대한 접근권한이다. VN_READ는 가상네트워크에 대한 정보를 열람할 수 있는 지에 대한 유무를 표현 한 것이다. 해당 권한은 데이터부에 직접적인 영향은 줄 수 없지만, 가상네트워크를 구성하는 다양한 정보를 열람 할 수 있기 때문에 해당 권한의 유무로, 가상네트워크 정보의 유출을 방지할 수 있다. VN_EVENT는 특정 가상네트워크에서 발생되는 다양한 네트워크 정보들 (Packet_IN, Flow statistics, ...)에 대한 제어메시지를 받을 수 있는지에 대한 여부이다. 예를들어 특정 가상네트워크에 대한 권한이 있는 경우, SM ONOS-VN은 발생하는 이벤트가 어느 가상네트워크 상에서 발생되어 있는지를 확인 한 후, 적절한 권한이 있는 애플리케이션에게 해당 정보를 넘겨주게 된다. VN_WRITE권한은 데이터부에 직접적으로 영향을 줄 수 있는 권한으로서, 특정 가상네트워크에 대하여 SM ONOS에서 정의한 데이터부에 영향을 줄 수 있는 권한을 사용 할 수 있는 권한이다. 예를 들어 특정네트워크에 대하여 해당 권한을 가진 경우에는, 애플리케이션이 대상 가상네트워크에 네트워크 룰도 내릴 수 있고, 물리네트워크를 관리할 때처럼 다양한 행위를 할 수 있다.

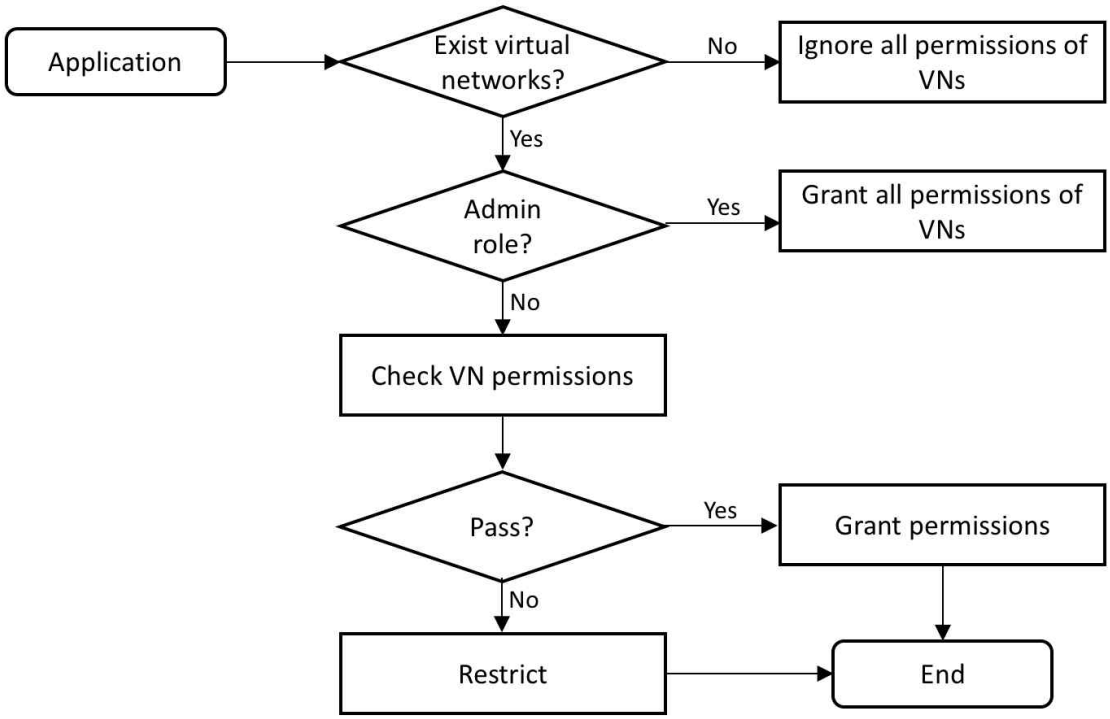


그림 10 SM ONOS-VN 퍼미션 모델 플로우 차트

그림 10은 SM ONOS-VN의 퍼미션 모델 플로우 차트를 도식화 한 그림이다. 특정 이벤트 (가상네트워크 접근, 제어, 등등)가 발생 한 경우 먼저 SM ONOS-VN은 현재 가상 네트워크가 있는지 없는지를 먼저 판단 한 후, 없는 경우에는 가상네트워크에 대한 모든 권한을 무시하게 된다. 그 다음 관리자역할인지 확인을 한 후 만약 관리자 권한을 가진다면 모든 권한을 부여하게 된다. 관리자 권한이 없는 경우에는 해당 애플리케이션은 최소 권한 전략 (Fine-grained access control mechanism)을 취하고 있는 것이므로, 권한을 상세히 확인하는 프로세스를 진행하게 된다. SM ONOS-VN의 권한파일에 정의 되어 있는 다양한 권한을 확인 한 뒤, 만약 해당 권한을 애플리케이션이 가지고 있으면, 권한을 주고 아니면 권한을 부여하지 않는다.

기본적인 SM ONOS-VN의 퍼미션 플로우 차트는 기존 SM ONOS 퍼미션모델과 비슷하다. 차이점은 대상이 제어부에 대한 최소 권한 전략에서 데이터부를 위한 최소권한 전략으로 바뀐게 가장 큰 차이점이다.

그림 11은 SM ONOS-VN의 권한이 추가된 권한 형석을 나타내고 있다. 기본적으로 SM ONOS-VN의 권한파일은 SM ONOS에서 제공하는 권한파일을 그대로 계승하고, 가상네트워크에 대한 권한이 추가된 점을 알 수 있다. SM ONOS에서 사용되는 role도 동일하게 적용이 된다. User 역할 일 경우, 가상네트워크에 대한 관리 권한

및 사용 권한은 최소 권한 부여 전략 (Fine-grained permission grant)을 취할 수 있고, 가상네트워크 관리 자체에 대한 권한과, 가상 네트워크 별 권한을 부여 할 수

```
<security>
    <role>USER</role>

    <VNs>
        <VN>
            <VNID>5</VNID>
            <VN-perm>VN_READ</VN-perm>
        </VN>
        <VN>
            <VNID>2</VNID>
            <VN-perm>VN_READ</VN-perm>
            <VN-perm>VN_EVENT</VN-perm>
            <VN-perm>VN_WRITE</VN-perm>
        </VN>
    </VNs>

    <permissions>
        <app-perm>VN_CREATE</app-perm>
        <app-perm>VN_REMOVE</app-perm>
        <app-perm>DEVICE_READ</app-perm>
        <app-perm>PACKET_WRITE</app-perm>
        <app-perm>HOST_READ</app-perm>
        <java-perm>
            <classname>org.osgi.framework.AdminPermission</classname>
            <name>*</name>
            <actions>metadata></actions>
        </java-perm>
        <java-perm>
            <classname>org.lang.RuntimePermission</classname>
            <name>modifyThread</name>
        </java-perm>
    </permissions>
</security>
```

Permissions per VN

SM ONOS permissions

그림 11 SM ONOS-VN이 추가된 권한 파일 형식

있다. 만약 Admin인 경우에는 전체 가상네트워크에 대한 모든 권한을 가질 수 있다. 예를들어 Admin권한을 가진 애플리케이션은 모든 가상네트워크에 대한 VN_READ, VN_EVENT, VN_WRITE권한을 가지게 되고, 관리자로서 모든 가상네트워크에 대한 삭제 및 생성을 할 수 있다.

VN테그는 새롭게 추가된 권한으로서, 특정 가상네트워크에 대한 최소 권한을 부여 할 수 있는 태그이다. 각 가상 네트워크에 따라 서로다른 권한을 부여 할 수 있다. 특히 VN_WRITE권한은 SM ONOS에서 정의 한 데이터부에 영향을 줄 수 있는 권한들을 app-perm테그에 정의되어 있는 권한에 한해 수행 할 수 있도록 한다. 마찬가지로 VN_READ는 가상네트워크에 대한 전반적인 정보와 해당 가상네트워크에 대한 정보를 열람 할 수 있다. 이 경우에는 데이터부에 직접적인 영향을 주지 않으므로 VN_WRITE보다 덜 중요한 권한이 된다. VN_EVENT같은 경우에는, 해당 가상네트워크에서 오는 이벤트정보를 받을 수 있는 권한이다. app-perm테그에는 가상네트워크 관리에 대한 권한을 부여 할 수 있다. 가상네트워크 자체에 대한 권한은 가상네트워크를 생성하거나 삭제할 수 있는 권한으로서 Admin권한에 속하지만, 최소권한 정책으로 좀더 상세히 정의 할 수 있다.

이와 같이 SM ONOS-VN은 SM ONOS 권한 파일을 그대로 승계하여, 가상네트워크에 대한 부분을 추가하였고, 하위호환성을 고려하여 디자인되었다. 하지만 아직 ONOS의 virtual network가 구현완료 되지 않았기에 개발은 진행하지 않았다. ONOS의 virtual network가 구현완료되어 배포 되는대로 SM ONOS-VN도 구현하여 배포할 계획이다.

2.3. Security-Mode ONOS 디자인

Security-Mode ONOS는 현재 그림 12와같이 디자인 되어져 있다. 그림 12에서 볼 수 있듯이, Security-Mode ONOS의 ONOS security subsystem은 ONOS의 kernel과 애플리케이션을 논리적으로 분리시켜 각 ONOS에 설치된 애플리케이션들을 sandboxing 시킨다. 또한, ONOS 애플리케이션들이 ONOS core와 그 밑의 레이어들에 대한 접근을 감시하고 중재하기 위해, ONOS Security subsystem은 ONOS kernel 번들들과는 다른 독립된 번들로써 동작한다. 이 ONOS security subsystem은 애플리케이션이 로딩될 때, 애플리케이션 번들 내에 포함된 policy파일을 읽어와서 파싱하고, 해당 퍼미션들을 애플리케이션 번들에게 부여한다. 애플리케이션번들에게 퍼미션이 부여된 이후에는, 애플리케이션이 부여받은 퍼미션 범위를 넘어서는 API들을 호출하는지 실시간 체크를 하게 되는데 OSGI API, Northobund API, JAVA native API들에 대한 실시간 체크는 OSGI security layer, NB-API permission checker layer, JAVA security layer에서 수행하게 디자인 되어져 있다. 좀 더 자세히 설명하자면 그림 12에서, OSGI security layer는 OSGI에서 제공하는 행위들에 관련된, 예를 들어 서비스를 요청한다던지 metadata를 변경하는 행위에 대한 제어를 수행하며, SM-ONOS NB-API permission checker는 애플리케이션이 ONOS에서 제공하는 northbound api를 호출 할 때마다, 해당 애플리케이션이 사용하려는 api에 대한 퍼미션을 가지고 있는지 아닌지를 확인한다. JAVA security는 JAVA native에서 제공하는 행위들, 예를 들면 자바의 소켓퍼미션이나 파일퍼미션을 필요로 하는 행위들에 대한 접근제어를 수행한다.

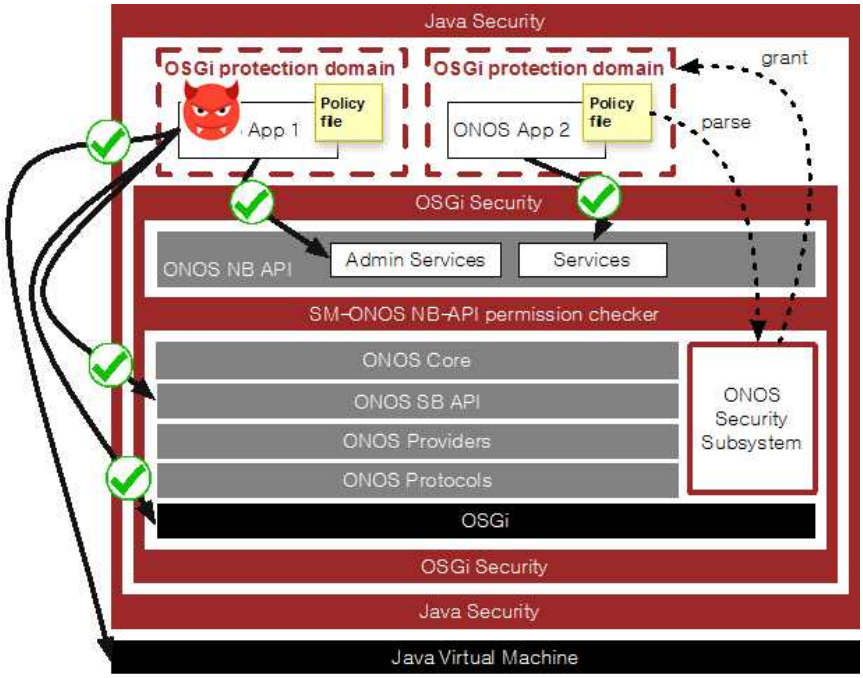


그림 12 Security-Mode ONOS 디자인

2.4. Security-Mode ONOS 구현

현재 Security-Mode ONOS의 구현에서 OSGI와 JAVA native API들에 대한 access control은 Felix OSGI security[4]를 확장, JavaSE 1.2 security를 이용하여 구현이 되어져 있다. ONOS Northbound API에 대한 access control은 2.2에서 기술한 것과 같이 새로운 ONOS Northbound API들에 따른 새로운 퍼미션 타입을 만들고 각 ONOS Northbound API들이 호출 될 때 마다, 해당 API가 실행되기 이전에 ONOS Security Subsystem내에 있는 퍼미션체킹 함수를 먼저 호출하여 API를 호출한 애플리케이션이 접근할 수 있는 퍼미션을 가지고 있는지 없는지를 검사한다.

Security-Mode ONOS는 현재 계속 구현진행중이며, ONOS 컨트롤러의 Emu(1.4) version부터 컨트롤러 내에 포함되어 배포되어지고 있다.

2.5. Security-Mode ONOS 추가기능

2.5.1 Security-Mode ONOS의 보안강화기능

Security-Mode ONOS와 Android같은 퍼미션 모델과 같이, 애플리케이션 개발자가 애플리케이션이 필요한 퍼미션들을 선언하고 배포하여 end user가 그것을 다운받아 애플리케이션이 어떠한 권한이 필요한지 살펴보고 설치하는 경우, permission gap 이라는 문제가 존재한다. Permission gap은 실제 애플리케이션이 사용하는 퍼미션보다 더 많은 퍼미션들이 xml파일에 선언되어있음에도 불구하고 end-user는 해당 퍼미션이 실제 애플리케이션이 사용하는 퍼미션인지 아닌지 모르기 때문에 더 많은 퍼미션들을 애플리케이션에게 부여하는 문제이다. 이러한 문제는 Android쪽에서 많

은 연구가 진행되어져 왔다. Android쪽에서는 대부분 이러한 permission gap의 문제에 대해 실제 Android market에서 배포 되고있는 애플리케이션들을 다운로드받아 얼마나 많은 애플리케이션들이 더 많은 권한을 요청하고 있는지를 분석 연구하였다. 하지만, permission gap을 어떻게 해결해야 하는지에 대한 문제해결방법은 제시된 적이 없다. SM-ONOS는 permission gap의 문제를 해결하기위해, permission을 network operator가 review 할 때, application이 어떠한 권한을 실제로 사용하는지에 대해 정적분석 기법을 이용하여 분석하고, 선언된 permission과의 차이를 비교하여 어떠한 permission gap이 존재하며 이것이 차후 어떠한 문제를 발생시킬 수 있는지에 대해 알려줌으로써 보안을 강화시켰다. 그림 13이 pemission gap을 이용한 보안강화기능의 결과화면이며, 이 부분에 대한 자세한 설명은 차후에 공개할 예정이다.

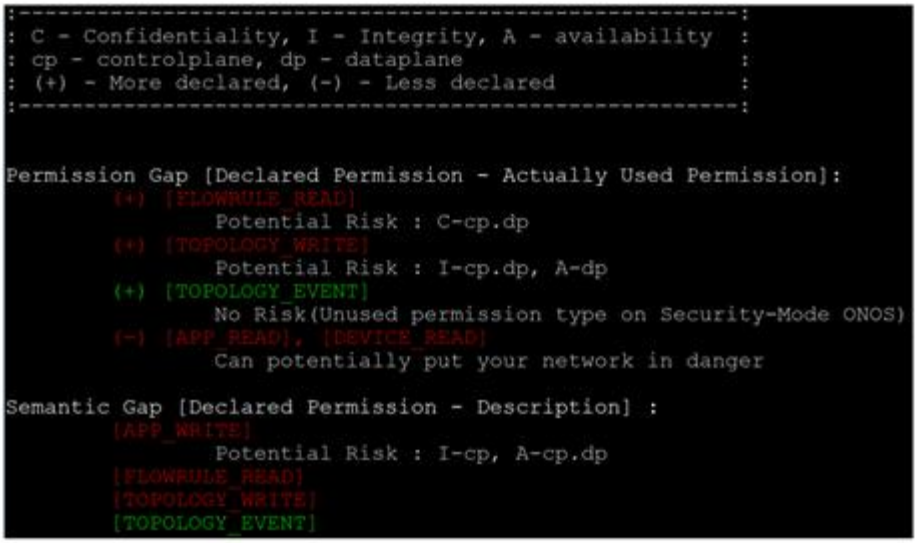


그림 13 Permission gap을 활용한 보안강화기능 결과화면

2.5.2 Unit test

Security-Mode ONOS의 디자인에 대한 구현을 점검하기 위해 unit test를 수행할 수 있다. 즉, 구현된 Security-Mode ONOS 각 모듈의 동작을 테스트함으로써, 디자인에 따른 정확한 구현을 확인할 수 있다. unit test는 기본적으로 프로그램의 오류를 발견하여 시스템의 안정성을 제공한다. 예를 들어, 잘못된 자료형이나 논리 연산자의 사용, 수식 및 알고리즘 계산 오류, 무한 루프 등의 시스템 오류를 점검한다. 또한, ONOS의 일부분으로서의 Security-Mode ONOS의 안전한 실행도 함께 점검할 수 있다. unit test를 통해 Security-Mode ONOS를 ONOS에서 고립 시켜서 테스트를 수행함으로써, Security-Mode ONOS의 불확실성을 제거하고, 전체 ONOS 모듈들과의 통합성을 향상시킨다.

Security-Mode ONOS는 다른 ONOS 모듈들과 마찬가지로, JUnit 프레임워크 기반의 unit test를 제공한다. JUnit은 컴파일 과정에서 JAR 파일 형태로 연결되어 동작하며, 본 시스템의 동작에는 영향을 끼치지 않고, 별개의 모듈로서 구동된다. 표 3

은 JUnit에서 자주 사용하는 주석(annotation)들의 예시들을 나타내며, 이러한 주석들을 사용하여 테스트 케이스를 구성할 수 있다.

Annotation	Description
@Test public void method()	해당 메소드를 테스트 메소드로 선언한다.
@Test (expected = Exception.class)	메소드가 정의된 예외 클래스를 발생시키지 않을 경우, 테스트 실패를 리턴한다.
@Test (timeout=100)	메소드의 실행이 정의된 시간 (100 milliseconds)을 초과할 경우, 테스트 실패를 리턴한다.
@Before public void method()	해당 메소드를 테스트 시작 전에 실행한다. 보통 인풋 데이터를 읽거나, 변수 초기화 등의 테스트 환경을 구성하는데 사용한다.
@After public void method()	해당 메소드를 테스트 종료 후에 실행한다. 보통 임시 데이터를 삭제하거나, 초기값 복구 등의 테스트 환경을 정리하는데 사용한다.
@BeforeClass public static void method()	모든 테스트를 시작하기 전에 해당 정적 메소드를 실행한다. 주로 데이터베이스 연결과 같은 작업을 수행한다.
@AfterClass public static void method()	모든 테스트가 종료된 후에 해당 정적 메소드를 실행한다. 주로 데이터베이스 연결 종료와 같은 환경 정리 작업을 수행한다.
@Ignore	테스트 메소드를 생략한다. 코드가 수정 중이거나, 테스트 케이스가 적용되지 않은 경우, 혹은 테스트 실행에 많은 시간이 소요될 때 유용하다.

표 3 JUnit 기본 주석(Annotation) 예시

o unit test 케이스 구현 예시

```
/**
 * Test of AppGuard.
 */
public class AppGuardTest {

    @Test(expected = AccessControlException.class)
    public void testCheckPermission() throws Exception {
        SecurityManager sm = new SecurityManager();
        sm.checkPermission(new AppPermission((AppPermission.Type.APP_EVENT)));
    }

}
```

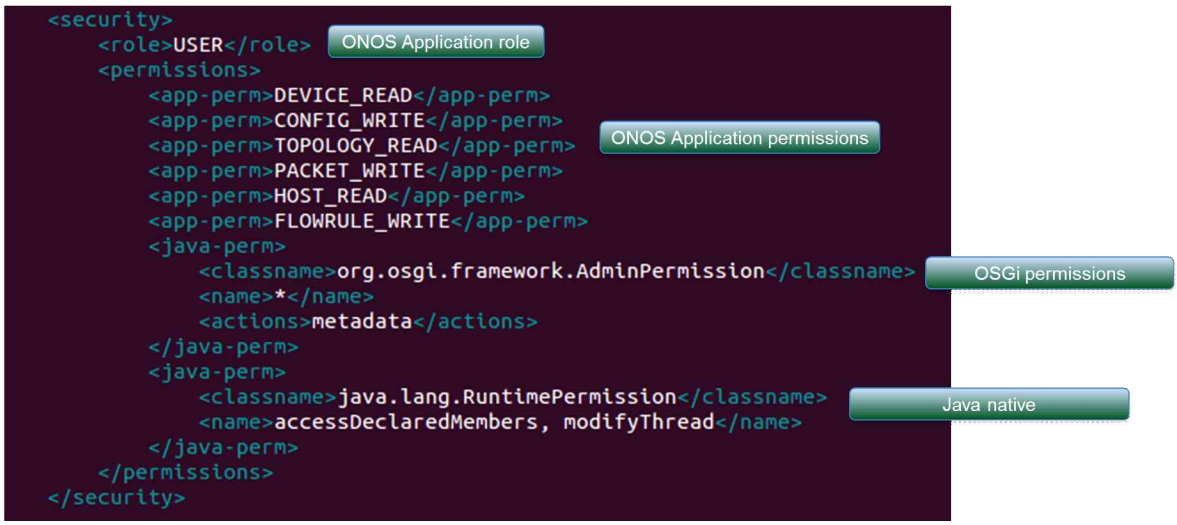
그림 14 Security-Mode ONOS unit test 케이스 예시

그림 14는 작성된 Security-Mode ONOS unit test 케이스의예시를보여준다. 해당

테스트 케이스는 Security-Mode ONOS의 AppGuard 클래스의 checkPermission 메소드에 대한 테스트를 수행한다. 테스트 실행 동안 AccessControlException 클래스의 예외가 발생하지 않으면, 해당 테스트 케이스는 실패로 결과를 리턴한다. 이를 통해, checkPermission 메소드가 정상적으로 퍼미션을 검증할 수 있는지를 테스트 할 수 있다.

2.6. Security-Mode ONOS 사용법

o 개발자가 작성해야하는 policy 파일 형식




```
<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 15 Policy 파일 형식 예제

Security-Mode ONOS에서 애플리케이션이 실행될 수 있게 하기 위해서는, 개발자는 자신의 ONOS 애플리케이션을 구현한 후, 어떠한 API를 사용하는지 분석하여 그에 맞는 policy 파일을 app.xml로 만들어서 패키징하고 배포해야한다. 개발자가 작성해야 하는 policy 파일의 형식은 그림 15와 같이 Application의 role을 USER나 ADMIN으로 작성해야하며 ONOS Northbound API들에 대한 퍼미션리스트, OSGI 퍼미션, 그리고 자바 Native 퍼미션들을 차례대로 작성해주면 된다.

o Security-Mode ONOS를 활성화시키는 방법

ONOS가 설치되어있는 로컬에서, shell에 아래와 같이 입력한면 자동으로 Security-Mode ONOS를 활성화시킨 환경이 구축된다.

```
$ onos-setup-karaf secure
$ onos-package -s -t
```

o Security-Mode ONOS에서 애플리케이션을 활성화 시키는 방법

Security-Mode ONOS는 2.3절에서 기술하였듯이, 애플리케이션이 어떠한 퍼미션을 필요로 하는지 검토하고 애플리케이션에게 해당 퍼미션을 부여해야만 애플리케이션이 활성화 될 수 있게 된다. 만약 검토 과정을 거치지 않는다면 그림 16과 같이 경고가 뜨며 애플리케이션이 활성화되지 않는다.

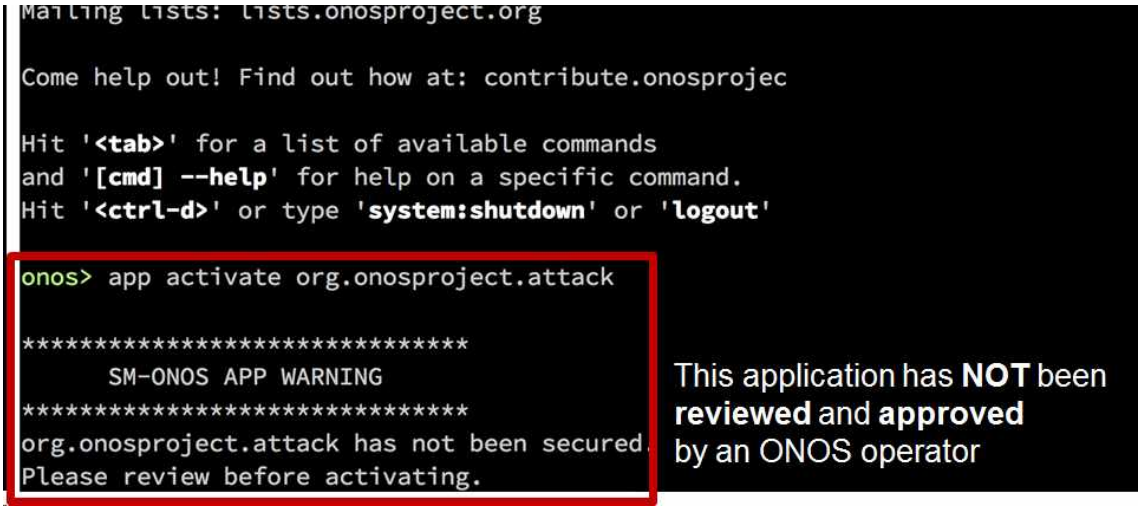


그림 16 퍼미션 부여되지 않은 애플리케이션을 활성화시키려는 경우

애플리케이션을 활성화시키기 위한 과정은, 먼저 그림 17과 같이 review [app-name] 명령어를 통해 애플리케이션이 요구하는 퍼미션들을 검토해야 한다.

그림 17 Security-Mode ONOS review 명령어 사용법

퍼미션이 해당 애플리케이션이 가질만한 퍼미션 이라면, 그림 18과 같이 review

[app-name] accept 명령어를 통해 해당 애플리케이션에게 퍼미션을 부여해준다. 이렇게 퍼미션을 부여해주고 난 이후에는, ONOS에서 애플리케이션을 활성화시키는 명령어인 app activate [app-name]을 통해 애플리케이션을 활성화 시킬 수 있다.

그림 18 Security-Mode ONOS accept 명령어 사용법

o unit test 실행 방법

ONOS가 설치된 환경에서, Shell 명령어를 통해 작성된 테스트 케이스들을 실행시킬 수 있다. Maven 빌드를 사용할 경우, 아래와 같이 maven 명령어로 테스트를 실행한다.

$ mvn clean test

최신 버전의 ONOS는 오픈 소스 빌드 도구인 Buck을 사용하는 빌드를 제공한다. Buck을 활용하여 ONOS의 unit test는 다음의 명령어를 통해 실행할 수 있다.

$ ONOS_ROOT/tools/build/onos-buck test

3. 결론

현재 SDN 컨트롤러들은 오픈소스 기반으로 활발하게 개발되어지고 있지만, ONOS를 포함한 대부분의 JAVA기반의 오픈소스 SDN 컨트롤러들이 보안을 고려하지 않고 디자인되었다. 이에 따라 하나의 JVM위에 컨트롤러와 모든 애플리케이션이 동작하고, 하나의 애플리케이션이 SDN컨트롤러가 제공하는 모든 서비스들에 접근가능하며 JVM에도 직접적으로 접근이 가능한, 굉장히 큰 권한을 가지는 취약한

구조를 가지고 있다. 이러한 취약한 구조를 보완하기 위해서는, SDN 애플리케이션의 안전성을 테스트하는 효율적인 SDN application security-instrumentation 이나, 엑세스 컨트롤 메카니즘이 필요하다.

본 기술문서에서 설명한 Security-Mode ONOS는 보다 안전한 SDN 환경을 만들기 위해 ONOS 컨트롤러위에 엑세스컨트롤 메카니즘을 구현한 것으로, 네트워크 관리자가 애플리케이션을 컨트롤러위에 활성화시키기 전에 애플리케이션의 행위를 퍼미션 기반으로 이해하고 활성화 시킬지 말지 결정하는 1차 필터링 메카니즘, 그리고 런타임에 애플리케이션이 부여받은 퍼미션 이외의 행위를 하는지 검사하고 이외의 행위에 대해서는 제한하는 메카니즘을 제공한다. 이러한 Northbound쪽의 접근제어 뿐만이 아니라, data-plane쪽에 관련된 virtual network의 접근제어 기능도 구현중에 있으며, 추가기능으로써 unit test와 permission gap을 활용한 보안강화기능도 제공한다. Security-Mode ONOS는 안전한 SDN 환경을 만들기 위한 첫걸음으로, 앞으로는 이 기능을 확장하여 ONOS 컨트롤러 위에 더 다양한 보안기능 뿐만 아니라 통합된 보안 프레임워크를 만들 계획이다.

References

[1] SHIN, S., SONG, Y., LEE, T., LEE, S., CHUNG, J., PORRAS, P., YEGNESWARAN, V., NOH, J., AND KANG, B. B. Rosemary: A robust, secure, and high-performance network operating system. In Proceedings of the 21th ACM Conference on Computer and Communications Security (CCS'14)

[2] THE APACHE SOFTWARE FOUNDATION. Apache Felix. http://felix.apache.org.

[3] THE APACHE SOFTWARE FOUNDATION. Apache Karaf. http://karaf.apache.org.

[4] THE APACHE SOFTWARE FOUNDATION. Apache felix framework security. http://felix.apache.org/documentation/subprojects/apache-felix-framework-security.html.

[5] A LINUX FOUNDATION COLLABORATIVE PROJECT. Open-Daylight SDN Controller. http://www.opendaylight.org.

[6] BERDE, P., GEROLA, M., HART, J., HIGUCHI, Y., KOBAYASHI, M., KOIDE, T., LANTZ, B., O'CONNOR, B., RADOSLAVOV, P., SNOW, W., ET AL. ONOS: towards an open, distributed SDN OS. In Proceedings of the third workshop on Hot topics in software defined networking (2014), ACM, pp. 1–6.

[7] ANDROID OPEN SOURCE PROJECT. App Manifest. http://developer.android.com/guide/topics/manifest/manifest-intro.html.

[8] K.Hong, L. Xu, H. Wang, and G. Gu. Poisoning network visibility in software-defined networks: New attacks and countermeasures. In Proceedings of the 22nd Annual Network and Distributed System Security Symposium(NDSS15), Feburary 2015.

[9] C. Ropke and T. Holz. Sdn rootkits: Subverting network operating systems of software-defined networks. In Research in Attacks, Intrusions, and Defenses, pages 339–356. Springer, 2015.

[10] 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.

[11] FloodLight. Open sdn controller. http://floodlight.openflowhub.org
[12] Android permission. http://developer.android.com/
reference/android/Manifest.permission.html.
[13] Security-Mode ONOS permission.
https://wiki.onosproject.org/display/ONOS/ONOS+Application+Permissions
[14] Symbolic execution.
https://en.wikipedia.org/wiki/Symbolic_execution

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: 2017/05