SDN 환경에서 보안서비스 제공을 위한 시스템 구축 방안 연구

최성호*, 곽진**
*순천향대학교 정보보호학과 정보보호응용및보증연구실
**순천향대학교 정보보호학과
e-mail: shchoi@sch.ac.kr

A Study on Construction Method for Provide Security Service in SDN Environment

Seong-Ho Choi*, Jin Kwak**
*ISAA Lab, Department of Information Security Engineering, Soonchunhyang University.
**Department of Information Security Engineering, Soonchunhyang University.

요 약

SDN(Software Defined Network)은 차세대 네트워크 기술로써 기존의 네트워크 기술의 문제점을 해결할 수 있는 하나의 수단으로 주목받고 있다. SDN은 기존 환경과는 다르게 장비를 통한 네트워크 연결이 아닌 소프트웨어를 통해 네트워크를 연결하고 통신을 할 수 있는 환경으로 현재 OpenFlow 기술이 활발히 연구되고 있다. 이 기술은 물리적인 네트워크 장치를 단순화하고 하나의 컨트롤러를 통해 네트워크 물리 장치를 제어함으로써 네트워크 환경을 구성한다. 이러한 기능을 통해 보안 서비스 제공 또한 소프트웨어를 통해 제공 될 수 있으나, 오버헤드 관리 및 가용성, 유지보수가 용이해야한다. 따라서 본 논문에서는 오버헤드 발생 및 유지보수, 가용성에 대한 요구사항을 만족하는 시스템 구성하고 간단한 IDS를 구현하여 테스트한다.

1. 서론

기존 네트워크 환경은 여러 네트워크 장치로부터 발생되는 다양한 패킷 및 가변적인 트래픽 등을 효율적으로 처리하기에는 한계가 있다[1]. 다양화 되는 패킷 및 가변적인 트래픽을 처리하기 위해 다양한 고가의 네트워크 장치를 구입하고 설치함으로써 네트워크 환경이 복잡해지고 있으며, 네트워크 벤더에 의존적으로 변화하고 있다. 이러한 문제점을 해결할 수 있는 하나의 수단으로 SDN 기술이 주목받고 있다[2]. SDN은 소프트웨어 정의 네트워크로써, 제어평면과 전달평면이 분리된 형태로 소프트웨어를 구현하여 여러 네트워크 장비를 제어할 수 있는 특성이 있다[3][4].

SDN의 특성을 통해 IDS/IPS 등과 같은 네트워크 보안 서비스 또한 소프트웨어로 제공될 수 있으며, 구현에 따라 다양한 보안 정책을 실행할 수 있다. SDN 환경에서 보안서비스를 제공하기 위해서는 기본적으로 오버헤드 문제 해결과 유지보수, 가용성이 용이해야 한다. 오버헤드가 급증할 경우 시스템 마비로 인한 네트워크 제어 기능을 상실할 수 있으며, 보안서비스 업데이트 및 오류 발생 시 유지보수가 용이해야한다. 또한 새로운 장치가 연결될 경우 기존에 제공받는 보안서비스를 동일하게 제공받을 수 있어야 한다.

본 논문에서는 SDN 환경에서 보안서비스 제공시 필요한 요구사항을 만족할 수 있는 시스템을 구성하고 간단한 IDS 서비스 모듈을 구현하여 테스트를 진행한다. 테스트는 단일로 구성된 보안 서비스모듈시스템과 Thrift로 분리된 보안서비스모듈시스템의 CPU 성능 차이 및 요구사항에 대하여 비교 분석한다.

본 논문의 구성은 다음과 같다. 2장에서 보안서비스 제공시 필요한 요구사항에 대하여 분석하고 3장에서 실제 시스템을 구성하고 테스트해 본다. 4장에서 요구사항 비교 분석하고, 5장에서 결론을 맺는다.

2. 보안서비스 제공의 요구사항

2.1 오버헤드 문제 해결

SDN은 컨트롤러의 소프트웨어를 통해 컨트롤러와 연결된 네트워크 장치의 통신제어가 가능하다.

이러한 시스템 환경에서 보안서비스를 제공할 경우, 컨트롤러와 연결되는 네트워크 장치가 증가할수록 패킷을 수집하고 분석하는 양이 많아지게 된다. 이 경우 컨트롤러 시스템의 오버헤드를 증가시킬 수 있다. 오버헤드가 증가될 경우 컨트롤러 시스템이 마비 될 수 있으며 네트워크 제어를 할 수 없게 되는 문제가 발생할 수 있다. 따라서 보안서비스 제공시 오버헤드를 감소시키는 방향에 대한

이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2012R1A2A2A01010886).

연구가 필요하다.

2.2 유지보수

SDN 환경에서 보안서비스 제공시 패킷 수집 및 분석 또는 IDS/IPS 등과 같은 네트워크 보안 서비스에 대한 새로운 기능을 프로그래밍으로 추가할 경우 컨트롤러에 최소한의 영향만 주어야 한다. 프로그래밍 특성상 기능이 추가될 경우 적용을 위해 재 컴파일을 실시하거나 재부팅을 실시할 수 있다. 또한 보안서비스 모듈의 오류로 인해 컨트롤러 시스템에 영향을 끼칠 경우 컨트롤러 시스템에 대한 마비 및 속도 저하 등과 같은 문제가 발생할 수 있기 때문에 컨트롤러 시스템에 영향을 받지 않도록 보안서비스를 제공해야 한다.

2.3 가용성

SDN 환경에서 새로운 네트워크 장치가 연결될 경우 기존의 네트워크 장치가 제공받는 보안서비스 기능을 동일하게 제공해야 한다.

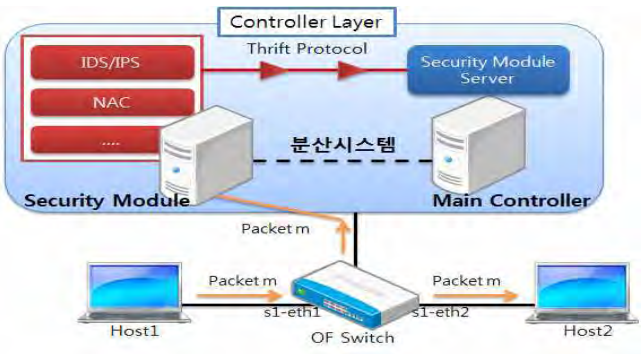
3. 구현 및 테스트

본 장에서는 SDN 환경에서 보안 서비스 제공 시 컨트롤러 시스템의 오버헤드를 증가시키지 않고 가용성과 유지보수가 용이하도록 컨트롤러 시스템을 구성한다. 구성한 시스템에 대한 결과를 호가인하기 위해 간단한 IDS 기능을 구현하고 테스트를 진행한다.

3.1 시스템 구성

본 논문에서 보안서비스 제공의 요구사항을 충족하기 위해 Thrift를 활용하여 컨트롤러 시스템으로부터 분산된 시스템을 구성한다. Thrift는 별도의 프로토콜을 통해 서버와 클라이언트로 구성될 수 있으며 다양한 언어를 하나의 프로그램으로 동작시킬 수 있는 기능이 있다[5]. 이러한 특징을 활용하여 메인 컨트롤러에 탐지결과 모듈을 구성하여 메인 컨트롤러와 병행하여 동작할 수 있는 분리된 보안 모듈 서비스 시스템을 구성한다.

(그림 1)은 Thrift를 활용한 보안 서비스 제공을 위한 시스템 구축 방안이다.



(그림 1) 보안서비스 제공을 위한 시스템 구축방안

보안 서비스 모듈을 분산시스템으로 구성할 경우 네트워크 제어를 담당하는 컨트롤러 시스템에 대한 오버헤드 발생을 감소시킬 수 있다. 보안 분석을 위한 패킷 메시지는 별도로 분리된 모듈 시스템으로 전송되기 때문에 컨트롤러에는 패킷 수집 및 분석으로 인한 오버헤드에 대한 문제를 해결할 수 있다. 또한 보안서비스 모듈에 대한 오류나 업데이트 사항이 생길 경우 컨트롤러 시스템을 재부팅 하거나 정지시킬 필요가 없다.

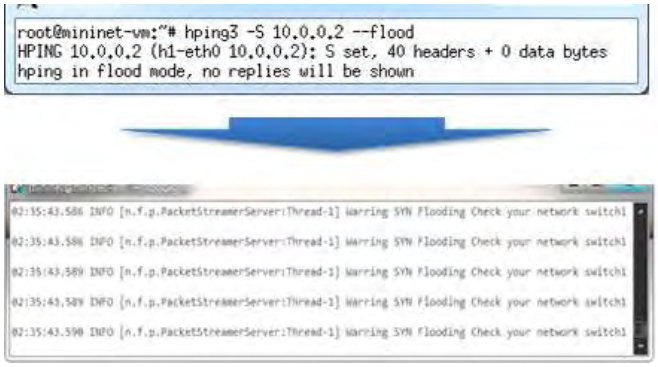
3.2 보안 서비스 모듈 구현 및 테스트

본 논문에서는 보안서비스 모듈 제공의 요구사항을 충족하기 위해 간단한 IDS 서비스 모듈을 구성하고 테스트를 진행한다. 테스트를 위해 가상의 네트워크 환경을 구성하고, TCPDUMP를 활용하여 패킷을 수집하였다. <표 1>은 IDS 서비스 모듈을 구현하고 테스트하기 위한 환경 구성이다.

<표 1> 테스트 환경

가상머신	Oracle VM Virtual BOX
CPU	Intel(R) Xeon(R) CPU E3-1230 v3 1/8
메모리	2G
운영체제	Ubuntu12.04 64bit
컨트롤러	Floodlight 0.9.0
네트워크 환경	Mininet 2.0

테스트는 (그림 1)과 같이 IDS 모듈을 구현하고, 컨트롤러 상에 연결된 스위치 1개, 스위치 5개, 스위치 10개를 기준으로, 각각에 호스트를 2개 생성하여 SYN Floodling 공격을 수행하였다. 이 과정에서 패킷을 수집하고 공격탐지를 위해 분석하는 동안 발생하는 CPU 점유율을 분석하였다. (그림 2)는 IDS 서비스 모듈에 대한 동작 화면이다.



(그림 2) IDS 모듈 서비스 동작 결과

3.3 테스트 결과

본 논문에서는 분산된 IDS 서비스 모듈을 구현하고 스위치를 연결시켜 테스트를 진행하였다. 진행결과 컨트롤러 자체에서 패킷을 수집하고 분석하여 네트워크 공격을 탐

지할 경우 IDS 서비스 모듈을 실행시키지 않은 상태보다 CPU 점유율이 크게 증가하였다. 반면 분산된 서비스 모듈로 구현하였을 때에는 컨트롤러 시스템이 단일 시스템으로 구성된 IDS 서비스 모듈의 CPU 점유율보다 안정적이었다. <표 2>는 테스트 결과를 나타낸다.

<표 2> CPU 점유율 비교 결과

Attack name	Switch	HOST	IDS 단일 모듈 구성	IDS 분산 모듈 구성
			CPU 점유율	CPU 점유율
SYN Flooding	1	2	1.7%	1.0%
	5	2	3.0%	1.4%
	10	2	6.1%	2.3%

4. 비교분석

본 논문에서는 보안서비스 제공을 위한 구성방안으로 Thrift를 활용하여 분산 모듈 시스템을 구성하였으며 간단한IDS 기능을 구현하고 테스트를 진행하였다. <표 3>은 요구사항에 대한 단일로 구성된 IDS 모듈 시스템과 비교한 결과이다.

<표 3> 요구사항 비교 분석 결과

요구사항	단일 모듈 구성	분산된 모듈 구성
오버헤드	· 높음	· 낮음
유지보수	· 유지 보수 시 컨트롤러 시스템 정지 및 재부팅	· 유지 보수 시 IDS 서비스 모듈 재부팅 · 단일 모듈 구성보다 용이함
가용성	· 새로운 장치에 대한 동일한 서비스 제공	· 새로운 장치에 대한 동일한 서비스 제공

❑ 오버헤드 관리

본 논문에서는 분산된 IDS 서비스 모듈 시스템을 구성하고 테스트를 진행하였다. 단일 시스템으로 구성할 경우 패킷 수집 및 분석 과정에서 CPU 점유율이 증가함에 따라 네트워크에 연결된 네트워크 장치들이 늘어날수록 오버헤드가 점점 증가하는 것을 확인하였다. 반면 IDS 서비스 모듈을 분산 모듈로 구성하였을 때에는 기존 IDS를 적용하지 않은 CPU 점유율과 비슷하게 측정되었다. 따라서 오버헤드를 증가시키지 않음으로써 단일로 구성된 시스템보다 안정적인 보안서비스 모듈을 제공할 수 있을 것이다.

❑ 유지보수

IDS 서비스 모듈을 단일 시스템상에서 구현하였을 때에는 오류 및 새로운 기능을 추가하기 위해 컨트롤러 시스템을 정지시키고 재시작과 같은 번거로움이 존재하였다. 하지만 Thrift를 활용하여 분산 모듈 시스템으로 구성한 결과 IDS 서비스 중 필요한 모듈만 재부팅하면 되기 때문에 네트워크를 제어하는 컨트롤러에 아무런 영향을 끼치지 않아 유지보수가 용이하였다.

❑ 가용성

IDS 서비스 모듈을 단일로 구성한 것과 분산된 서비스 모듈로 구성된 것 모두 새로운 스위치를 연결하였을 때에는 기존의 컨트롤러에서 제공하는 플로우 서비스를 동일하게 제공하는 것과 마찬가지로 새로운 연결 장치에 대한 IDS 서비스를 제공한다.

5. 결론

본 논문에서는 SDN 환경에서 보안서비스를 제공할 경우 오버헤드 문제 해결 및 유지보수, 가용성에 대한 요구사항을 충족하도록 Thfrit를 활용하여 IDS 서비스 모듈을 분산된 모듈로 구성하였다. 테스트 결과 단일 시스템으로 구성된 IDS 서비스 모듈보다 CPU 점유율이 안정적이었다. 요구사항에 대한 결과로는 단일로 구성한 IDS서비스 모듈보다 오버헤드가 낮고 유지보수가 용이한 결과가 나왔으며, 장치에 대한 동일한 네트워크 보안 서비스를 제공하였다.

결과적으로 보안서비스 모듈을 제공할 경우 Thrift를 활용하여 각각의 최적에 알맞은 언어로 서비스를 구현함과 동시에 오버헤드 감소, 유지보수, 가용성에 대한 기능을 제공함으로써 보안서비스 제공이 용이하였다. 이러한 결과를 통해 향후 안정적인 보안서비스를 제공하기 위해 시스템을 구성할 경우 하나의 방안으로 사용될 수 있을 것이다.

참고문헌;

[1] 유재형, 김우성, 윤찬현 “SDN/OpenFlow 기술 동향 및 전망” KNOM Review, 2014, pp.1-22.

[2] Kim, H. and Feamster, N. “Improving Network Management with Software Defined Networking” Communications Magazine. IEEE.51, 2013, pp.114-119.

[3] ONF Software-Defined Networking: “The New Norm for Networks” ONF White Paper.4, 2012, pp.3-12.

[4] Fernando, N. N. Farias, Joao J. Salvatti, Eduardo Cerqueira, and Antonio Jorge Gomes Abelem “Management of the Existing Network Environment Using Openflow Control Plane” IEEE NOMS. 2012, pp.1143-1150.

[5] Thrift, http://www.thrift.apache.org