소프트웨어 정의 네트워킹 환경에서의 자원 고갈 공격에 대한 효과적인 대응 방안에 대한 연구

남재현†, 양준석, 김연근, 신승원‡

정보보호대학원,
전산학부, 한국과학기술원

Defense Mechanism against Resource Consumption Attacks in Software-Defined Networking Environments

Jaehyun Nam†, Joonseok Yang, Yeonkeon Kim, Seungwon Shin‡

Graduate School of Information Security,
School of Computing, KAIST

요 약

소프트웨어 정의 네트워킹의 등장은 기존 네트워크 환경의 새로운 패러다임을 가져왔다. 하지만 새로운 네트워킹 기술의 적용은 또한 새로운 공격 벡터를 열어주게 되었다. 본 논문에서는 여러 공격 벡터 중 소프트웨어 정의 네트워킹의 특징을 이용한 자원 고갈 공격에 주목한다. 그리고 공격자가 적은 자원만으로도 대상 네트워크 전체를 마비시킬 수 있는 자원 고갈 공격을 일회용 플로우 룰, 동적 타임아웃 메커니즘, 임시 및 정적 플로우 테이블을 통해 효과적으로 대응할 수 있는 방법을 제시하고자 한다.

I. 서론

소프트웨어 정의 네트워킹(Software-Defined Networking: SDN) 기술은 차세대 네트워킹 기술 중 하나로 기존 네트워크 환경에서의 제어평면과 데이터평면을 분리함으로써, 네트워크의 유연성을 향상시켰을 뿐만 아니라 중앙집중형 관리구조를 통해 전체 네트워크를 손쉽게 관리할 수 있도록 하였다. 소프트웨어 정의 네트워킹 기술에서는 분리된 각각의 평면을 연결하기 위해 오픈플로우(OpenFlow) 프로토콜 [1]을 널리 사용한다. 오픈플로우 프로토콜을 통해 제어평면에서는 데이터평면 내 패킷들에 대한 다양한 액션(패킷 포워딩, 리다이렉션, 로드벨런싱 등)을 취할 수 있게 되었다.

하지만 소프트웨어 정의 네트워킹을 통해 위와 같이 다양한 액션을 표현하기 위해서는 기존 네트워크에서의 단순 포워딩이나 라우팅과는 달리 더 많은 정보들을 필요로 하게 된다. 이와 같이 각각의 네트워크 플로우에 대한 다양한 액션을 정의하기 위해서는 많은 공간이 필요하게 되며, 이 점을 이용한 것이 바로 자원 고갈 공격 [2]이다. 소프트웨어 정의 네트워킹에서의 자원 고갈 공격은 소프트웨어 정의 네트워킹의 기본 특성을 이용한 공격으로 적은 수의 새로운 네트워크 플로우들을 데이터평면 내로 전송한다. 여기서 전송되는 네트워크 플로우란 중복되지 않는 독립적인 네트워크 플로우를 의미한다. 데이터평면 내로 전송된 공격자의 네트워크 플로우들은 데이터평면 내 네트워크 장비의 플로우 저장 공간을 무의미하게 차지하게 되며, 네트워크 장비들은 더 이상 새로운 플로우에 대한 정보를 유지할 수 없게 된다. 결과

- 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [B0190-15-2011, 한·미 SDN/NFV WAN 네트워크 안전성 기술연구 및 Test bed 구축]

†주저자: namjh@kaist.ac.kr

‡교신저자: claude@kaist.ac.kr (corresponding author)

적으로 공격자의 무의미한 플로우 생성으로 인하여 네트워크 장비들은 본래의 역할을 수행할 수 없게 되며, 최악의 경우 전체 네트워크가 마비될 수 있다.

본 논문에서는 세 가지 기법(일회용 플로우 룰, 동적 타임아웃 메커니즘, 임시 및 정적 플로우 테이블)을 통해 각각의 네트워크 플로우에 대한 액션들이 데이터평면 내 유지되는 시간을 동적으로 변화시켜 자원 고갈 공격을 효과적으로 대응할 수 있는 방법을 제시하고자 한다.

II. SDN 환경에서의 자원 고갈 공격

기존 네트워크 장비들의 경우 네트워크 플로우들에 대하여 포워딩 혹은 라우팅과 같은 기본적인 액션을 취하기 위해 단순히 MAC 주소 또는 IP 주소 정보만 유지하면 된다. 반면, 오픈플로우를 이용하여 다양한 액션을 취하기 위해서는 프로토콜 정보부터 MAC 주소, IP 주소, 포트 번호, MPLS, VLAN 정보 등 더 많은 정보를 유지해야 한다. 이처럼 네트워크 장비들의 고정된 저장 공간 내에 각 플로우마다 더 많은 정보를 유지해야하기 때문에 장비마다 최대로 유지할 수 있는 플로우 개수에 제약이 발생하게 된다. 기존 연구 [3]에 따르면 일반적인 상용 스위치들의 경우 최대 1,500 ~ 2,000개정도의 플로우들을 관리할 수 있다고 한다.

따라서 이러한 제약 때문에 자원 고갈 공격 [2]이 가능하게 된다. 공격자가 임의의 네트워크 플로우를 생성함으로써 네트워크 장비 내 플로우들을 위한 가용 저장 공간들이 무의미한 플로우들로 채워지고, 실제 정상적인 플로우가 생성되었을 때 가용공간의 부족으로 인하여 해당 플로우에 대한 처리를 할 수 없게 된다. 결과적으로 네트워크 장비들이 정상적으로 동작할 수 없게 되면서 전체 네트워크의 기능을 상실하게 되는 것이다.

III. 자원 고갈 공격에 대한 대응 방안

자원 고갈 공격의 경우 플로우들에 대한 네트워크 장비들의 가용 공간을 소모시키는 것이 주된 목적이기에 불필요한 플로우들에 대한 정보들을 빠르게 네트워크 장비 내에서 제거함으로써 자원 고갈 공격에 대응할 수 있다. 본 논문에서는 불필요한 플로우들을 제거하기 위해 세 가지의 대응 메커니즘(일회용 플로우 룰, 동적 타임아웃 메커니즘, 임시 및 정적 플로우 테이블)을 활용한 대응 방안을 제안하고자 한다.

3.1 일회용 플로우 룰

새로운 플로우가 발생하였을 때, 제어평면에서는 해당 플로우가 정상적인지 아닌지 판단하는 기준이 없다. 따라서 해당 플로우의 정상 유무를 판단하기 위해 우선적으로 해당 플로우에 대해서 일회용 플로우 룰을 적용한다. 일회용 플로우 룰이란 단어가 의미하는 그대로 어떠한 액션에 대해서 한 번만 수행하기 위한 룰로써, 만약 일회성 액션이 수행된 이후에 해당 플로우에 대한 액션 요청이 데이터평면에서 제어평면으로 다시 이루어진다면 해당 플로우에 대해서 정상적인 플로우일 가능성이 높다고 판단할 수 있게 된다.

오픈플로우 프로토콜을 이용할 경우, 기존 환경에서는 FLOW MOD를 이용하여 해당 플로우에 대한 액션을 적용하지만, 일회용 플로우 룰은 PACKET OUT 또는 FLOW MOD의 타임아웃을 최소값(오픈플로우 프로토콜 명세 [5]에서는 1초를 최소값으로 둔다)으로 설정하여 해당 플로우에 대한 액션을 적용한다. 여기서 두 가지 형태의 일회용 플로우 룰을 제공하는 이유는 네트워크 내 트래픽 양에 따라 제어평면의 부하를 조절할 수 있게 하기 위함이다.

3.2 동적 타임아웃 메커니즘

SDN 환경에서는 기본적으로 일정한 타임아웃을 기반으로 데이터평면 내에 플로우들에 대한 액션 정보를 유지한다. 제어평면 내 컨트롤러의 종류에 따라 다소 차이는 있으나, 일반적으로 10초 ~ 30초 정도의 타임아웃을 둔다. 즉, 타임아웃이 발생될 때까지 해당 플로우가 이용되지 않을 경우 해당 플로우가 종료되었거나 현재 유휴 상태라고 간주하고 해당 플로우에 대한 정보를 네트워크 장비 내 저장 공간에서 제거한다.

하지만 기존 연구결과 [4] 에 따르면, 일반적인 네트워크 플로우의 경우 수 초 내에 플로우들이 종료된다. 따라서 플로우들에 대한 액션 정보를 유지하는 시간을 우선적으로 최대한 짧게 정의하고, 특정 플로우에 대한 액션 요청이 지속될수록 점차 유지시간을 증가시킴으로써, 데이터평면 내에서 유지하고 있는 종료된 또는 유휴 상태인 플로우들에 대한 정보를 최소화 할 수 있게 된다.

3.3 임시 및 정적 플로우 관리 테이블

앞서 언급한 일회용 플로우 룰과 동적 타임아웃 메커니즘의 경우, 주기적으로 발생되는 네트워크 플로우들에 대해서는 역으로 추가적인 네트워크 오버헤드를 발생시킬 수 있다. 따라서 주기적으로 발생되는 네트워크 플로우들은 별도로 관리할 필요가 있다. 이를 위하여 어떠한 플로우가 새로 발생하였을 경우, 우선적으로 임시 플로우 테이블 내에서 해당 플로우를 관리하며, 임시 플로우 테이블 내에 존재하는 플로우들에 대해서는 일회용 플로우 룰과 동적 타임아웃 메커니즘을 적용한다.

특정 플로우의 발생 빈도가 임의의 임계치를 넘게 되면 해당 플로우는 정적 플로우 테이블 내에서 관리되며, 정적 플로우 테이블 내에 존재하는 플로우들은 기존 SDN 환경과 동일하게 일반적인 타임아웃 기반 플로우 처리방법을 따라 처리된다. 정적 플로우 테이블 내에서 관리되는 플로우들의 경우, 일정 시간이 지나면 정적 플로우 테이블에서 제거되고, 해당 플로우가 다시 발생하였을 때에는 다시 임시 플로우 테이블에서부터 관리된다.

IV. 실험결과

본 논문에서 제시한 대응 방안의 효과를 확인하기 위하여 자원 고갈 공격에 대한 대응 메커니즘 적용 전과 후를 비교해보았다. 비교실험을 위한 테스트 환경은 두 대의 Intel i5-3470 CPU @ 3.20GHz 머신을 사용하였으며, 하나의 머신 내에는 네트워크 에뮬레이터인 Mininet [6] 을 활용하여 단일 네트워크 토폴로지를 생성하였다. 또한 데이터평면 내 최대 유지 가능한 플로우 정보의 개수를 일반적인 네트워크 장비들과 유사하게 맞추기 위하여 Open vSwitch [7] 내 가용공간을 플로우 2,000개로 한정하였다. 다른 머신 내에는 제어평면 내 액션을 정의하기 위하여 POX [8] 컨트롤러와 기본적으로 제공하는 레이어-2 포워딩 어플리케이션을 사용하였다. POX 컨트롤러의 경우 Python 형태로 구현되어 있어, 다양한 연구에서 프로토타입을 실험할 때 많이 사용된다. 본 논문에서 제시한 각각의 메커니즘은 POX 컨트롤러 내 자원 고갈 공격 대응 어플리케이션 형태로 구현하였다.

5.1 대응 메커니즘 적용 전

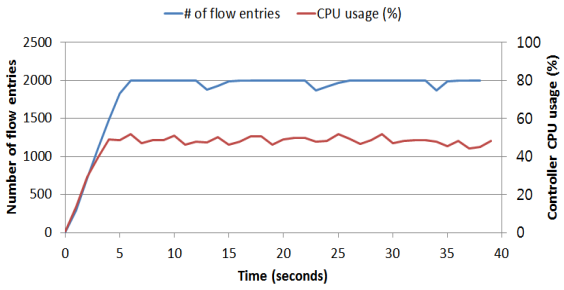

Fig. 1. Before applying the defense mechanism against the resource consumption attacks

Fig. 1에서 볼 수 있듯이 대응 메커니즘을 적용하기 전에는 일정한 타임아웃을 가진 FLOW MOD를 이용하여 각각의 플로우들을 관리하기 때문에 임의의 네트워크 플로우들을 생성하였을 경우 공격자에 의한 네트워크 플로우들이 가용 공간을 빠르게 고갈 시킬 수 있게 된다. 또한 가용 공간 내 유지될 수 있는 네트워크 플로우의 수가 한정되어 있기 때문에, 공격자는 초당 (가용 공간 내 유지될 수 있는 네트워크 플로우 수) / (기본적으로 주어지는 타임아웃 시간) 개의 새로운 네트워크 플로우를 지속적으로 생성함으로써 매우 적은 양의 트래픽으로도 가용 공간을 고갈 시킬 수 있음을 확인할 수 있다.

제어 평면 측면에서도 공격자의 네트워크 플로우를 처리하기 위해 CPU 사용량 역시 증가함을 확인할 수 있다. 이는 공격자의 발생시키는 무의미한 플로우들로 인한 제어평면의 자원 고갈 공격 역시 가능함을 보여준다.

5.2 대응 메커니즘 적용 후

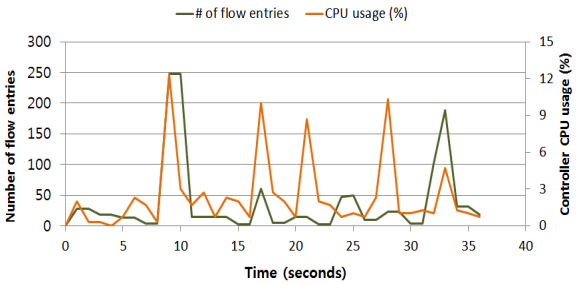

Fig. 2. After applying the defense mechanism against the resource consumption attacks

반면 Fig. 2에서 볼 수 있듯이 앞선 실험과 달리 대응 메커니즘을 적용한 후에는 데이터평면 내 공격자의 네트워크 플로우가 빠르게 제거됨을 확인할 수 있다. 또한 CPU 사용량 역시 대응 메커니즘 적용 전과 달리 상당히 낮은 수준을 유지함을 확인 할 수 있다.

이번 실험을 통해 일회용 플로우 룰에 의해 1차적으로 공격자의 네트워크 플로우가 필터링되고 공격자에 의해 동일 플로우가 다시 발생하더라도 동적 타임아웃 메커니즘에 의해 공격자의 네트워크 플로우가 빠른 시간 내에 네트워크 장비 내 저장 공간에서 제거될 수 있음을 확인할 수 있다. 또한 임시 및 정적 플로우 테이블을 통해 정상적인 네트워크 플로우를 안정적으로 처리함과 동시에 공격자의 네트워크 플로우를 분별할 수 있게 된다. 따라서 본 논문에서 제시하는 세 가지 기법을 통해 자원 고갈 공격에 대한 효과적인 대응이 가능함을 알 수 있다.

V. 결론

본 논문에서는 소프트웨어 정의 네트워킹의 환경적인 특성을 이용한 자원 고갈 공격에 대응하는 효과적인 방안을 제시하였다. 실제 네트워크와 유사한 환경에서 대응 방안의 실용성을 검증하였으며, 실험 결과를 통해 일회용 플로우 룰, 동적 타임아웃 메커니즘, 그리고 임시 및 정적 플로우 관리 테이블을 활용한 방법으로 자원 고갈 공격에 효과적으로 대응할 수 있음을 확인할 수 있었다.

[참고문헌]

[1] McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74.

[2] Shin, Seungwon, and Guofei Gu. "Attacking software-defined networks: A first feasibility study." Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, 2013.

[3] Curtis, Andrew R., et al. "DevoFlow: Scaling flow management for high-performance networks." ACM SIGCOMM Computer Communication Review. Vol. 41. No. 4. ACM, 2011.

[4] Woo, Shinae, et al. "Comparison of caching strategies in modern cellular backhaul networks." Proceeding of the 11th annual international conference on Mobile systems, applications, and services. ACM, 2013.

[5] OpenFlow specification, https://www.opennetworking.org/sdn-resources/openflow

[6] Lantz, Bob, Brandon Heller, and Nick McKeown. "A network in a laptop: rapid prototyping for software-defined networks." Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. ACM, 2010.

[7] Pfaff, Ben, et al. “Extending Networking into the Virtualization Layer.” Proceedings of the 8th ACM SIGCOMM Workshop on Hot Topics in Networks. ACM, 2009.

[8] POX, http://www.noxrepo.org/pox/