ISSN 2287-1543 (Online)

KNOM Review

Network Operations & Management

Aug. 2017 Vol.20 No.1

한국통신학회 통신망운용관리연구회

The Committee on Korean Network Operations and Management (KNOM)
Korean Institute of Communication Sciences (KICS)
http://www.knom.or.kr

목 차

knom.or.kr

✓ 서버 가상화 환경에서 트래픽 모니터링을 위한 Virtual TAP 설계 1
정세연, 홍지범, 홍원기 (포항공과대학교)

✓ SCADA 시스템의 주기성과 발생순서 정보를 반영한 Structured ACL 모델에 관한 연구 9
정우석, 윤정한, 김신규, 구영훈, 이성호, 김명섭 (고려대학교)

✓ HPC 응용의 런타임 실행 패턴에 따른 소프트웨어 정의 계산 구조 분석 17
오지선, 이효정, 김윤희 (숙명여자대학교)

✓ SDN기반 IoT 게이트웨이에서 DoS 공격 영향 평가 24
최진태, 뉘엔 신 응억, 김경백 (전남대학교)

✓ 선형 SVM기반의 안드로이드 악성코드 탐지 성능 향상 방안 34
김기현, 최미정 (강원대학교)

KNOM Review '17-01 Vol.20 No.01
https://doi.org/10.22670/knom.2017.20.1.1

서버 가상화 환경에서 트래픽 모니터링을 위한 Virtual TAP 설계

정세연◆°, 홍지범*, 홍원기*

Design of Virtual TAP for Traffic Monitoring in Server Virtualization Environment

Seyeon Jeong◆°, Jibum Hong*, James Won-Ki Hong*

요 약

오늘날 클라우드 서비스가 본격적으로 사용되어 관련 트래픽이 증가하고 높은 서비스 품질에 대한 요구사항이 증가하고 있다. 따라서, 이를 구현하고 있는 데이터센터(Datacenter) 내부 서버의 자원에 대한 효과적인 사용을 도모하는 서버 가상화 기술이 큰 관심을 끌고 있다. vTAP (Virtual Test Access Port)은 서버 가상화 환경에서 구성되는 데이터패스(Datapath)에 기존 하드웨어 TAP 장비를 배치할 수 없는 문제점을 극복하고, 가상 머신(Virtual Machine, VM) 간 전송되는 트래픽을 패킷 수준에서 모니터링 할 수 있게 한다. 그러나, 소프트웨어로 동작하는 vTAP 기능은 호스트 머신의 컴퓨팅 자원을 사용한다는 점에서 서버 가상화 환경에서의 패킷 프로세싱 성능을 저하시킨다. 본 논문에서는 이러한 문제점을 해결하기 위해 DPDK 및 eBPF와 같은 오픈소스 고속 패킷 프로세싱 기술을 활용한 vTAP 구현 설계를 제안한다. 이에 더해, 관리자가 SDN 컨트롤러 및 어플리케이션을 통해 TAP 정책을 유연하게 제어할 수 있는 시스템을 설계한다.

Key Words : Virtual TAP, Software-Defined Networking (SDN), Fast Packet Processing, Network Function Virtualization (NFV), Virtual Network Function (VNF)

ABSTRACT

Currently, virtualization technology that brings effective use of server resources to a datacenter is promising as cloud services are being prevalent with increasing traffic volumes and requirements for higher service quality. vTAP (Virtual Test Access Port) overcomes the problem that existing hardware TAP equipment cannot be deployed on a datapath composed by server virtualization, and enables to monitor traffic between virtual machines (VMs) at a packet level. This paper proposes two types of vTAP designs with open source projects for fast packet processing, such as DPDK and eBPF, to complement performance degradation induced by resource sharing to the same host machine between VM (traffic) and vTAP software. This paper also designs a system allowing a network administrator to flexibly control TAP policies through an SDN controller and application.

※ 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2015-0-00575, 글로벌 SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발)
◆ First Author : Pohang University of Science and Technology Department of Computer Science Engineering, jsy0906@postech.ac.kr
° Corresponding Author
* Pohang University of Science and Technology Department of Computer Science Engineering, {hosewq,jwkhong}@postech.ac.kr
논문번호 : KNOM2017-01-01, Received May 15, 2017; Revised June 21, 2017; Accepted August 5, 2017

Ⅰ. 서 론

오늘날 클라우드 서비스가 본격적으로 사용되어 관련 트래픽이 증가하고 높은 서비스 품질에 대한 요구사항이 증가하고 있다. 따라서, 이를 구현하고 있는 데이터센터(Datacenter) 내부 서버의 자원(컴퓨팅, 스토리지, 네트워크)에 대한 효과적인 사용을 도모하는 가상화 기술이 큰 관심을 끌고 있다. 특히 서버에 하이퍼바이저(Hypervisor)와 가상 스위치(Virtual switch)를 구축하여 그 위에 가상 머신(Virtual machine, VM)과 어플리케이션을 동작시켜 서비스를 제공하는 형태의 서버 가상화 기술이 주목 받고 있다[15]. 이러한 환경에서, 서비스 요청에 대한 처리를 위해 동일한 호스트 서버 내의 다른 VM과의 통신 또는 물리적으로 다른 서버에 위치한 VM과의 통신이 동시다발적으로 발생하게 된다. 이 과정에서 데이터센터 내부 트래픽 중 VM 간 트래픽(East-west 트래픽)이 차지하는 비중이 현저히 커지고 있는 상황이다[1].

vTAP (Virtual Test Access Port)은 기존 하드웨어 TAP 장비의 소프트웨어 버전으로써, 서버 가상화 환경에서 VM 간에 발생하는 트래픽에 대한 가시성(Visibility)을 제공하는 것을 목표로 한다[5, 6]. 기존 하드웨어 TAP 장비는 단말 혹은 스위치/라우터 사이의 링크 구간에 위치하여 통과하는 패킷을 복제(Clone)한 뒤, 복제된 패킷을 별도의 모니터링 장비로 전송하는 역할을 수행한다. 그러나 오늘날 데이터센터와 같이 서버 가상화 기술을 사용하는 환경에서 동일한 호스트 머신 내의 VM 간 링크는 가상 스위치를 통한 가상 링크(Virtual link)로 구성이 되므로, 해당 구간에 물리적인 TAP 장비를 설치하는 것이 불가능하다. 가상화 환경에서 TAP 기능의 부재는 VM 간 트래픽을 패킷 수준에서 모니터링 할 수 없음을 의미하며, 나아가 보안 및 유지보수가 어렵게 되고 패킷 분석을 통한 네트워크 관리, 최적화 등 서비스 품질을 높이기 위한 요구사항을 충족시킬 수 없게 된다. 따라서 본 논문에서는 서버 가상화 환경에서 소프트웨어 형태로 TAP 기능을 수행하여 패킷 수준에서의 모니터링을 제공하는 vTAP 설계를 제안한다. 나아가 관리자가 SDN 컨트롤러를 통해 TAP 정책을 유연하게 제어할 수 있는 시스템을 설계한다.

SDN (Software-Defined Networking) 및 NFV (Network Function Virtualization) 기술의 발전과 더불어 통신 사업자나 서비스 제공자 측에서는 자체 데이터센터에 가상화 환경을 구축하고, 기존 네트워크에서 미들박스(Middlebox) 형태로 제공되던 네트워크 기능들을 VM 위에서 동작하는 VNF (Virtualized Network Function) 형태로 전환하여 CAPEX 및 OPEX를 줄이고자 한다[2]. 대표적인 활용 사례로 vEPC (Virtual Evolved Packet Core), vIMS (Virtual IP Multimedia Subsystem) 등이 있다(그림 1.).

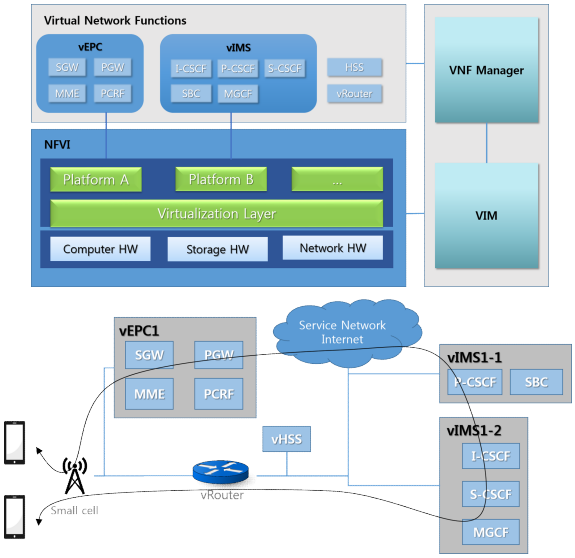


그림 1. vEPC 및 vIMS 구성과 VoLTE에의 활용
Fig. 1. A composition of vEPC and vIMS, and their application to VoLTE

기존 EPC 구성에서는 각 구성요소 간 물리적 링크 구간에 하드웨어 TAP 장비를 설치하여 전체 시스템 및 개별 구성요소의 네트워크 상태를 실시간 모니터링 한다. 그림 1의 vEPC 구성에서도 이와 같은 요구사항이 충족되어야 하지만, VNF (Virtual Network Function) 형태의 vEPC 구성요소(vSGW, vPGW 등)가 동작하는 VM 간 가상 링크에 물리적인 TAP 장비를 설치할 수 없다.

본 논문에서는 이러한 서버 가상화 환경에서 기존 TAP 장비의 역할을 대신하는 vTAP의 구현 설계와 함께, SDN의 중앙 집중적이고 유연한 제어 방식을 이용하여 관련 TAP 정책을 효과적으로 관리하는 SDN 어플리케이션의 구현 설계를 제시하고자 한다. 호스트 머신에서 소프트웨어 형태로 동작하는 vTAP의 특성상 패킷 복제 및 전달 연산에 해당 호스트 머신의 자원을 사용하므로, 같은 호스트 머신 자원을 공유하는 가상 스위치, VM 및 그 위에서

동작하는 VNF의 패킷 프로세싱 성능이 저하된다. 따라서, 본 논문에서는 DPDK (Data Plane Development Kit)[3] 및 eBPF (extended Berkley Packet Filter)[4]와 같은 고속 패킷 프로세싱 기술을 활용하여, vTAP이 한정된 자원에서 보다 효과적으로 동작하고 동시에 서버 성능에 대한 영향을 최소화함으로써 이 문제를 해결하고자 한다.

Ⅱ. 연구 배경 및 관련 연구

2.1. 연구 배경

현재 vTAP은 IXIA[5], Gigamon[6] 등 해외 업체의 가상화 환경 모니터링 솔루션의 구성요소로 포함되어 판매되고 있다. 따라서 설치 환경에 따른 구매 비용의 증가, 솔루션의 벤더 의존성 및 폐쇄성에 따른 확장과 고장 시 해결의 어려움 등의 문제가 존재한다. 외산 제품의 경우 참고 자료가 충분히 제공되지 않으나, 솔루션 매뉴얼, 홍보 자료 등을 참고했을 때, IXIA 제품의 경우 호스트 머신의 커널을 수정하여 VM 간 트래픽에 대한 vTAP을 구현한다. 또한, 복제된 패킷은 동일 호스트 내 별도의 모니터링 프로그램으로 전달되어 분석된다. Gigamon의 경우 각 VM에 vTAP 에이전트(Agent)를 설치하여 송/수신 패킷을 복제하고, 이를 Overlay 형태로 외부에 위치한 모니터링 시스템으로 전달한다. 외산 제품의 경우 높은 처리 성능을 보이지만 솔루션에 대한 종속성을 가진다. 반면, 본 논문의 vTAP 구현 설계는 Open vSwitch[16]를 기반으로 DPDK 및 eBPF와 같은 고속 패킷 프로세싱 기술을 활용하므로 오픈소스가 제공하는 개방성과 확장성의 이점을 가질 수 있다.

Open vSwitch는 VM 간의 가상 네트워크 구축 환경을 제공하는 리눅스 기반 가상 스위치로써, OpenFlow 프로토콜을 지원하여 SDN 스위치로도 사용할 수 있다. 최근 데이터센터의 서버/네트워크 가상화 환경의 필요성과 함께 내부 VM 간, 혹은 외부와의 연결성을 보장해야하므로 네트워크 구성 및 관리의 자동화 기능을 제공하는 Open vSwitch의 역할이 주목받고 있다.

DPDK는 데이터 평면(Data plane) 라이브러리와 네트워크 인터페이스 컨트롤러를 포함하는 오픈소스 프로젝트이며, EAL (Environment Abstraction Layer)을 생성하여 DPDK 기반 네트워크 응용프로그램이 서버의 하드웨어 자원을 효과적으로 사용하여 고속으로 패킷을 처리할 수 있도록 지원한다. 리눅스 환경에서 기존의 커널 네트워킹 스택 기반 패킷 처리를 수행하지 않고 유저 영역(User space)에서 CPU 인터럽트(Interrupt) 없이 고속으로 패킷을 처리할 수 있으나, 호스트 머신이 인텔(Intel) 프로세서와 DPDK를 지원하는 네트워크 카드를 사용해야 한다는 제약사항이 있다.

eBPF는 리눅스 환경에서 주로 패킷 필터링 및 모니터링 목적으로 사용되던 BPF (Berkely Packet Filter)의 성능 및 보편성을 증대시킨 확장 버전으로써, 리눅스 커널 3.15 버전부터 공식적으로 포함되어 있다. 기존 리눅스 환경에서 네트워크 응용의 커널 기반 구현은 제한적인 반면, eBPF는 C, Python 등 고급 언어로 작성된 프로그램을 바이트 코드로 변환하고, 커널에 해당 프로그램을 로드하여 이벤트 발생에 따른 핸들러를 호출하는 방식으로 동작한다. 따라서 최근에 eBPF를 통한 커널 기반 네트워크 및 시스템 모니터링, 고속 패킷 프로세싱 활용에 대한 관심이 커지고 있다.

2.2. 관련연구

학계에서는 주로 vTAP 구현 기술 자체보다는 가상화 기반 데이터센터 네트워크에서 SDN 및 OpenFlow 기술을 이용한 효과적인 네트워크 모니터링에 관한 연구가 활발하다. Plank[7]의 경우 하드웨어 OpenFlow 스위치의 포트 미러링(Port mirroring) 기능을 활용하여 특정 트래픽 플로우를 샘플링하고, 샘플링의 정확도와 수집 속도를 높이기 위한 전반적인 연구를 수행하였다. NetAlytics[8]의 경우 DPDK 및 하드웨어 OpenFlow 스위치의 포트 미러링 기능을 활용하여 VM 간 패킷의 복제본을 모니터링 엔티티(Entitiy)로 전달하는데, 호스트 머신당 하나의 VM만 동작하는 구조를 전제하고 있다.

상기 연구들은 데이터센터 네트워크 수준에서 하드웨어 스위치를 활용한 네트워크 모니터링을 다룬 반면, 본 논문에서는 호스트 머신 내부의 VM 간 패킷을 고속으로 복제하고 효과적으로 정책 제어할 수 있는 vTAP 설계 내용에 초점을 맞춘다.

Ⅲ. 시스템 디자인

그림 2는 서버 가상화 환경에서 하드웨어 TAP 장비의 한계점과 비교하여 vTAP의 동작 위치 및 기능, 가상 스위치와의 관계를 나타낸다.

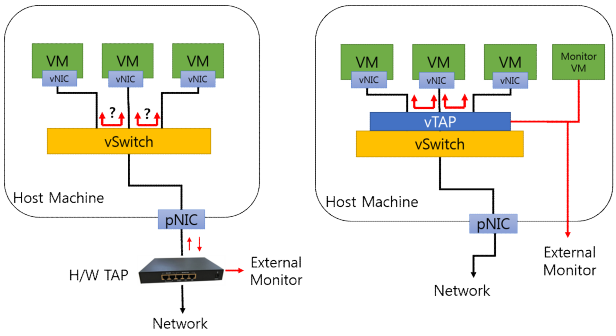


그림 2. 하드웨어 TAP과 Virtual TAP의 비교
Fig. 2. Comparison of a hardware TAP and virtual TAP

3.1. Virtual TAP 설계

vTAP 기능 구현을 위한 가상 스위치로 Open vSwitch(이하 OVS)를 사용하며 OpenFlow 프로토콜 기반의 플로우 룰(Flow rule) 제어를 통해 패킷 스위칭을 제어한다. OVS는 호스트 머신의 물리 네트워크 인터페이스와 VM의 가상 네트워크 인터페이스를 내부 가상 포트를 이용하여 브릿지(Bridge) 형태로 연결한다. 각 가상 포트에서 수신된 패킷은 해당 브릿지에 설정된 특정 플로우 룰에 매칭되어 포워딩되는데, OpenFlow의 그룹 테이블(Group table) 기능을 활용하여 해당 패킷을 실제 목적지 VM 및 가상 스위치에 연결된 별도의 모니터링 VM으로 멀티캐스트(Multicast) 할 수 있다. 이 과정에서 패킷 복제 연산이 발생하는데, 트래픽 볼륨 및 특성에 따라 가상 스위치의 스위칭 성능 저하를 야기할 수 있다. 따라서 성능 저하를 보완하기 위한 고속 패킷 프로세싱 기술로 다음 두 가지 방법 모두로 vTAP 을 구현하고 성능을 비교 분석한 뒤, 보다 적절한 방안을 선택하여 구현을 확장한다.

3.1.1. DPDK 기반 유저 스페이스 설계

DPDK는 고속 패킷 처리를 위한 데이터 평면 라이브러리와 네트워크 인터페이스 컨트롤러의 집합으로써, 인텔 프로세서와 DPDK 지원 네트워크 인터페이스 카드(Network Interface Card, NIC)로 구성된 서버 상에서 고속으로 패킷처리를 수행하는 응용프로그램의 개발을 용이하게 한다. DPDK 라이브러리는 크게 다음의 네 가지 기능을 제공한다.

1) 메모리에 DPDK 전용 오브젝트 풀을 할당하여 큐, 버퍼 등을 생성 및 관리
2) Lock-free 큐를 구현하여 다수의 DPDK 관련 쓰레드를 병렬 동작
3) 메모리에 고정 크기의 버퍼를 미리 할당받아 패킷을 저장
4) Poll Mode Driver (PMD)를 통해 CPU 인터럽트 없이 NIC 버퍼에 저장된 패킷을 메모리로 전달

OVS는 2.2 버전부터 가상 포트와 DPDK를 지원하는 물리 NIC과의 바인딩을 지원하며, 2.4 버전부터 하이퍼바이저를 통해 생성되는 VM과의 DPDK 기반 가상 인터페이스(dpdkvhostuser 포트) 연결을 지원하도록 DPDK 모드로 동작 가능하다(이하 OVS-DPDK). 일반적인 OVS가 커널 스페이스에 데이터 평면 기능을 구현하는 것과 달리, OVS-DPDK는 유저 스페이스에서 패킷 포워딩을 수행하며 OpenFlow 프로토콜 지원을 유지한다. OVS-DPDK가 1Gbps 속도로 패킷을 수신할 때, OpenFlow 그룹 테이블 기능을 이용해 패킷을 복제하여 원래 목적지 포트 및 TAP 포트로 멀티캐스팅하는 실험에서 패킷 사이즈에 관계없이 1Gbps Line rate를 보여주는 실험 결과가 있다[9].

OVS-DPDK를 구동하는 과정에서 사용자는 DPDK 코어 쓰레드가 호스트 머신의 특정 CPU 코어를 점유하도록 지정해 CPU affinity를 구현하거나, 패킷 프로세싱 목적의 2MB 또는 1GB 크기의 Hugepage를 할당하여 TLB (Translation Lookaside Buffer) miss 횟수를 감소시키는 등의 방법으로 성능을 향상 시킬 수 있다. 특히, 호스트 머신이 2개 이상의 프로세서(Processor)를 가지고 각 프로세서마다 다중 코어를 가진 형태로 NUMA (Non-Uniform Memory Access) 구조를 지원하는 경우, 하이퍼바이저와 OVS-DPDK의 CPU Affinity 기능을 활용해 대량의 트래픽을 발생시키거나 처리하는 VM의 vCPU 및 DPDK PMD 쓰레드를 동일한 NUMA 노드의 코어에 배치할 수 있다. 이는 패킷 프로세싱에 있어 해당 NUMA 노드에 Locality를 가지는 메모리를 최대한 활용할 수 있도록 한다. 이러한 환경설정을 통해, 스위칭 및 vTAP 기능을 수행하는 데 있어 OVS-DPDK가 일반(커널) 모드의 OVS 보다 더 나은 성능을 보일 것으로 예상된다[13].

3.1.2. eBPF 기반 커널 스페이스 설계

eBPF는 네트워크 모니터링 목적의 패킷 필터링 및 분석을 위한 용도로 도입된 BPF의 확장된 버전으로써, 리눅스 커널 3.16 버전부터 공식적으로 내

장되어 있다. 개발자가 eBPF 라이브러리를 통해 네트워크 응용프로그램을 C 언어 및 Python으로 작성하면, eBPF 컴파일러는 이를 바이트 코드로 변환하여 리눅스 커널에 모듈 형태로 로드시킴으로써 커널 API 콜을 통한 고속 네트워크 기능을 실현할 수 있다. 따라서 기존에 다양한 네트워크 응용프로그램(모니터링, 보안, tracing 등)의 동작에서 발생하는 커널 네트워킹 스택과 유저 스페이스 사이의 빈번한 전환(Context switching) 및 메모리 copy에 따른 성능 저하 문제를 해결 할 수 있는 방안으로 주목받고 있다.

eBPF 기반 vTAP 구현을 위해 C 언어로 작성된 프로그램과 Python으로 작성된 프로그램이 필요하다. C 프로그램은 패킷 복제 작업 및 복제된 패킷의 식별 목적으로 사용되는 VLAN 태깅 등 실제 vTAP 기능을 수행하는 로직을 작성하기 위해 BPF 라이브러리를 사용한다. 해당 API 호출은 eBPF 프레임워크에 의해 커널 수준의 기능 수행을 위한 바이트 코드로 변환된다. 유저 스페이스에서 동작하는 Python 프로그램은 해당 바이트 코드가 바인딩되어 동작할 커널 네트워킹 스택의 특정 포인트를 지정한다. 본 논문의 vTAP 기능은 리눅스 TC (Traffic Control)에 도달한 패킷에 대해 BPF API를 호출하는 것으로 구현 가능하므로 TC에 바인딩 한다(그림 3). 또한, 커널 프로그램과의 데이터 공유를 위한 Key/value 기반 Map을 생성할 수 있어, 복제된 패킷의 출력 포트 인덱스 값 등을 해당 바이트 코드에 제공할 수 있다.

가상 스위치(OVS)를 통해 전송되는 VM 간 트래픽을 작성된 eBPF 기반 vTAP 프로그램으로 우회시키기 위해 별도의 데이터패스(Datapath) 구성이 필요한데, 이는 3.3에서 다룬다.

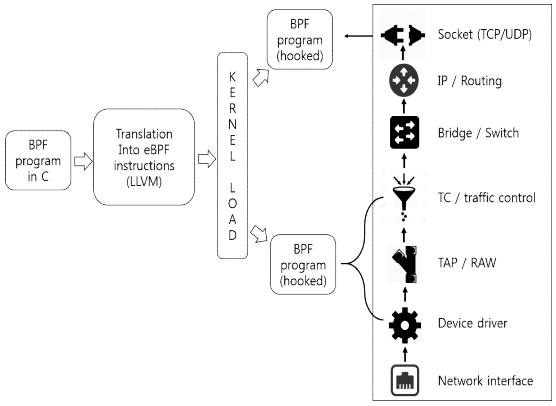


그림 3. eBPF 프레임워크
Fig. 3. eBPF Framework

3.2. TAP 정책 제어 SDN 어플리케이션 설계

본 논문의 vTAP 기능 구현에 있어 첫 번째 방안은 패킷 복제 및 복제된 패킷의 포워딩 기능을 모두 OVS-DPDK가 담당하는 것으로 요약할 수 있고, 두 번째 방안은 패킷 포워딩은 OVS가, 패킷 복제 기능은 eBPF 커널 프로그램이 담당하는 것으로 요약할 수 있다. 두 가지 방안 모두 가상 스위치로 OVS를 사용하기 때문에 OpenFlow 프로토콜을 통해 SDN 컨트롤러와 연동시킬 수 있다. 따라서, 네트워크 관리자가 SDN 어플리케이션(제어 평면)을 통해 설정한 TAP 정책을 SDN 컨트롤러가 OpenFlow 플로우 룰 형태로 변환시켜 OVS 또는 OVS-DPDK에 삽입함으로써 vTAP 기능(데이터 평면)을 제어할 수 있다. 이를 위해서 SDN 컨트롤러로 ONOS (Open Network Operating System)를 사용할 예정이며, TAP 정책 제어를 위한 관리자 인터페이스를 위해 별도의 ONOS 어플리케이션을 구현할 예정이다. 관리자는 해당 어플리케이션에서 다음의 작업을 수행할 수 있어야 한다.

1) 송신지/목적지 MAC 주소 또는 IP 주소, TCP/UDP 포트 번호, 프로토콜 타입에 대한 명세를 통한 TAP 대상 트래픽 플로우 지정

2) 설정된 TAP 플로우의 통계값 정보(플로우 룰 설치 여부, 패킷 카운트 등) 확인

3) 가상 스위치의 기본 정보(포트 번호, 연결된 VM의 식별자 등) 확인

3.3. 동작 방식 설계

본 논문에서 설계된 vTAP은 두 가지 방법으로 구현될 예정이다. 그림 4의 좌측 호스트 머신(Host1)에서는 OVS-DPDK 기반 vTAP이 구현되며, 우측 호스트 머신(Host2)에서는 OVS와 eBPF로 작성된 커널 기반 vTAP 프로그램의 연동으로 vTAP이 구현된다. 두 호스트 상의 OVS 인스턴스 모두 OpenFlow 프로토콜을 통해 ONOS 컨트롤러에 연결되며, 관리자는 ONOS의 TAP 어플리케이션을 통해 TAP 정책을 제어한다. 설정된 TAP 정책은 OpenFlow 메시지로 변환되어 각 OVS의 DP (Datapath) 컴포넌트에 플로우 룰 형태로 설치된다.

OVS-DPDK 기반 vTAP 구현의 경우(Host1), OVS를 통과하는 VM 간 트래픽 중 설정된 TAP 정책에 따른 플로우 룰에 매칭되는 패킷은 그룹 테

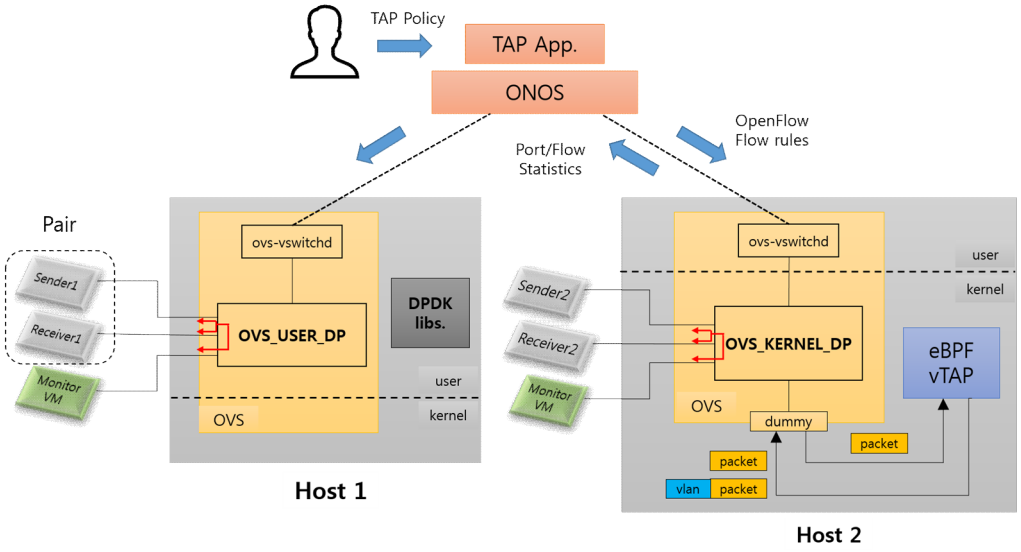


그림 4. vTAP 설계
Fig. 4. vTAP Design

이블로 전달되어 내부 버킷의 개수만큼 복제된다. 이 과정에서 패킷 복제 기능은 OVS-DPDK의 내부 구현 및 동작 성능에 의존한다. 이후 해당 패킷은 목적지 VM으로 포워딩(버킷1 액션)됨과 동시에 모니터 VM으로 포워딩(버킷2 액션)된다[14].

반면, eBPF 기반 vTAP 구현의 경우(Host2), 패킷 복제 기능을 eBPF 커널 프로그램에서 수행하므로, OVS를 통과하는 VM 간 패킷을 eBPF 프로그램으로 우회시키는 데이터패스를 구성해야 한다. 이를 위해 OVS의 더미(Dummy) 포트로 VM 간 트래픽을 전달하여 호스트 머신의 네트워킹 스택으로 진입시킨 뒤, 커널 vTAP의 동작을 수행하여 복제 패킷을 생성한다. OVS는 VLAN 태그를 통해 원본 패킷과 복제 패킷을 구분하여 각각의 목적지로 전달한다. 이를 위해 하나의 TAP 정책에 대해 2개 이상의 관련 플로우 룰이 OVS에 설치되어야 한다.

관리자는 TAP 어플리케이션에서 각 호스트 상의 OVS에서 전달하는 OpenFlow 제어 메시지를 통해 현재 설정된 TAP 정책과 관계된 통계 정보를 확인할 수 있다.

Ⅳ. 논 의

4.1. 성능 평가 방안

vTAP의 OVS-DPDK 구현과 eBPF 구현의 성능 평가를 위해 각 호스트 머신에 각각 Sender와 Receiver 역할을 수행하는 VM 인스턴스 쌍(Pair)을 배치한다(그림 4.). Monitor VM의 개수는 호스트 당 하나로 가정한다. 관리자는 TAP 어플리케이션 상에서 Sender에서 Receiver 로 향하는 트래픽에 대한 TAP 정책을 미리 설정하고, Sender VM에서 트래픽을 발생시킨다. 트래픽 발생기로는 Iperf와 D-ITG[10]를 사용하며, Receiver VM(생성 트래픽 수신) 및 Monitor VM(TAP 트래픽 수신)에서 수신된 트래픽의 Throughput을 측정한다. 또한, VM이 사용하는 호스트 머신의 자원(vCPU, 메모리 등)에 따라 트래픽 수신 과정에서 패킷 drop이 발생할 수 있으므로, VM의 커널 수준에서 실제 패킷 수신량을 확인하기 위한 pps (packet per second) 측정이 필요하다. 이 과정에서 Baseline(일반 모드 OVS의 그룹 테이블 기반 vTAP 구현)에 비해 성능이 떨어지거나 향상 수준이 미비할 경우, OVS-DPDK 기반 vTAP 구현은 DPDK 관련 커스터마이징(CPU affinity, 멀티 큐 등)을 수행하고, eBPF 기반 vTAP 구현의 경우 프로그램 로직 변경 및 OVS와 연동되는 데이터패스를 재설계한다.

확장성을 검증하기 위해 Sender/Receiver 쌍의 개수를 증가시키면서 동일한 실험을 진행하여, 트래픽 볼륨 증가에 따른 vTAP 성능 변화를 측정한다. 또한 물리적으로 이격된 VM 간 트래픽의 vTAP 성능 측정을 수행한다.

4.2. OpenStack 연동

본 연구의 확장성을 점검하고 vTAP의 실제 사용 환경에 보다 근접한 구현을 위해 NFVI (NFV Infrastructure)로 많이 사용되는 OpenStack 테스트베드와 vTAP PoC 구현을 연동할 예정이다. OpenStack은 이미 TaaS (TAP as a Service) 프로젝트를 통해 VM 인스턴스 간의 트래픽 모니터링 기능을 플러그인 형태로 제공하고 있다[11]. 그러나 해당 기능은 OpenStack 네트워킹(Neutron) 수준에서 컴퓨트 노드(Compute node)에 배치된 OVS의 포트 미러링 기능을 간접적으로 이용하고 있으므로, 사용하기 편리한 반면 성능에 대한 고려가 충분하지 않다. 따라서, OpenStack의 컴퓨트 노드에 기본적으로 설치되는 OVS를 OVS-DPDK로 대체하여 본 연구의 vTAP 구현을 적용, 사용 가능성을 검토해볼 수 있다. 이를 위해서 다음의 요구사항을 반영하도록 구현을 확장해야 한다. 1번 요구사항을 위해서 ONOS의 SONA (Simplified Overlay Network Architecture) 프로젝트를 활용할 수 있다[12].

1) OpenStack 네트워킹 제어 기능을 Neutron에서 ONOS 컨트롤러로 이전

2) OpenStack 테넌트(Tenant) 단위의 TAP 정책 제어

3) 컴퓨트 노드 간 트래픽의 TAP 처리

V. 결 론

본 논문에서는 데이터센터 등의 서버 가상화 환경에서 VM 간 트래픽을 패킷 수준에서 모니터링하기 위한 Virtual TAP(vTAP) 구현의 설계를 다루었다. 서버 가상화 환경에서 VM들은 가상 스위치와의 가상 네트워크 인터페이스를 통해 생성된 가상 링크를 통해 연결되기 때문에, 패킷 모니터링 목적으로 기존의 하드웨어 TAP 장비를 사용할 수 없다. 이러한 문제점을 극복하기 위해 호스트 머신 상에서 소프트웨어로 동작하는 vTAP을 사용할 수 있으나, vTAP 기능을 위해 사용되는 호스트 머신의 자원을 VM 및 VM 트래픽과 공유한다는 점에서 전체적인 성능 저하를 야기한다.

본 논문에서는 이러한 문제점을 최신의 오픈 소스 고속 패킷 프로세싱 기술을 활용하여 해결하기 위해 DPDK 기반의 vTAP 구현 설계 및 eBPF 기반의 vTAP 구현 설계를 다루었다. vTAP은 VM 간에 전달되는 패킷을 복제하여 원본은 본래 목적지 VM으로, 복제본은 모니터링 VM으로 포워딩 한다(데이터 평면). 또한, 가상 스위치와 SDN 컨트롤러를 OpenFlow로 연결하고, TAP 정책 제어를 위한 SDN 어플리케이션을 개발하여 관리자가 활용할 수 있게 한다(제어 평면). 설정된 TAP 정책은 SDN 컨트롤러에 의해 플로우 룰 형태로 변환, 가상스위치에 설치되어 vTAP 기능 수행을 위한 데이터패스를 반영한다.

향후 연구 방향으로는 3장에서 설계된 vTAP의 두 가지 구현 방안(DPDK 및 eBPF 기반)을 바탕으로 실제 구현하고, 그림 4의 테스트베드를 바탕으로 4장에서 다룬 성능평가를 수행한다. 이후 데이터평면의 vTAP 기능과 제어평면 상에서의 TAP 정책 제어를 위해 가상스위치와 SDN 컨트롤러를 연동하고 3.2절의 요구사항을 만족하는 어플리케이션을 구현한다. 최종적으로는 4.2절에서 다룬 바와 같이 OpenStack 환경에서도 잘 동작하도록 vTAP을 확장 구현한다.

References

[1] Gilard Shainer, 100 Gbps Headed For The Data Center (2014), Retrieved July, 20, 2017, from https://www.redbooks.ibm.com/redpapers/pdfs/redp4933-ko.pdf.

[2] Mijumbi, Rashid, et al. "Network function virtualization: State-of-the-art and research challenges." IEEE Communications Surveys & Tutorials 18.1 (2016): 236-262.

[3] DPDK, Data Plane Development Kit (2017), July, 20, 2017, from http://dpdk.org/.

[4] IO Visor Project, extended Berkeley Packet Filter (2017), July, 20, 2017, from https://www.iovisor.org/technology/ebpf.

[5] IXIA, IXIA PHANTOM VTAP WITH TAPFLOW FILTERING (2016), Retrieved July, 21, 2017, from https://www.ixiacom.com/sites/default/files/2016-07/V-DS-Phantom-vTap.pdf.

[6] Gigamon, GigaVUE-VM 데이터 시트 (2016), Retrieved July, 21, 2017, from https://www.gigamon.com/content/dam/resource-library/korean/data-sheet/ds-gigavue-vm-virtual-machine-kr.pdf.

[7] Rasley, Jeff, et al. "Planck: Millisecond-scale monitoring and control for commodity networks." ACM SIGCOMM Computer Communication Review 44.4 (2015): 407-418.

[8] Liu, Guyue, and Timothy Wood. "Cloud-scale application performance monitoring with sdn and nfv." Cloud Engineering (IC2E), 2015 IEEE International Conference on. IEEE, 2015.

[9] Shanmugalingam, Sivasothy, Adlen Ksentini, and Philippe Bertin. "DPDK Open vSwitch performance validation with mirroring feature." Telecommunications (ICT), 2016 23rd International Conference on. IEEE, 2016.

[10] Avallone, Stefano, et al. "D-ITG distributed internet traffic generator." Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings. First International Conference on the. IEEE, 2004.

[11] OpenStack, Tap-as-a-Service (2017), Retrieved July, 23, 2017, from https://github.com/openstack/tap-as-a-service.

[12] H. Moon, SONA: DC Network Virtualization (2017), Retrieved July, 2017, from https://wiki.onosproject.org/display/ONOS/SONA%3A+DC+Network+Virtualization.

[13] Ciara L. (Intel), Use the Data Plane Development Kit with Open vSwitch to Create vHost User Non-Uniform Memory Access Awareness (2016), Retrieved July, 2017, from https://software.intel.com/en-us/articles/vhost-user-numa-awareness-in-open-vswitch-with-dpdk.

[14] ONF, OpenFlow Switch Specification 1.3.1 (2012), Retrieved July, 26, 2017, from https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf.

[15] Sefraoui, Omar, Mohammed Aissaoui, and Mohsine Eleuldj. "OpenStack: toward an open-source solution for cloud computing." International Journal of Computer Applications 55.3 (2012).

[16] Pfaff, Ben, et al. "The Design and Implementation of Open vSwitch." NSDI. 2015.

정 세 연 (Seyeon Jeong)

2015 경북대학교, 컴퓨터학부 학사

2015 ~ 현재 포항공과대학교, 컴퓨터공학과 석사과정

<관심분야> SDN, OpenFlow, 네트워크 관리

홍 지 범 (Jibum Hong)

2017 한양대학교 ERICA, 컴퓨터공학과 학사

2017 ~ 현재 포항공과대학교, 컴퓨터공학과 통합과정

<관심분야> SDN, 네트워크 관리

홍 원 기 (James Won-Ki Hong)

1995 ~ 현재 포항공과대학교, 컴퓨터공학과 교수

2007 ~ 2011 포항공과대학교, 정보통신대학원장

2007 ~ 2010 포항공과대학교, 정보통신연구소 연구소장

2008 ~ 2010 포항공과대학교, 컴퓨터공학과 주임교수

2008 ~ 2012 포항공과대학교 정보전자융합공학부장

2012 ~ 2014 KT 종합기술원 원장

<관심분야> 네트워크 트래픽 모니터링, 네트워크 및 시스템 관리, SDN/NFV, IoT, 무크기반 교육

KNOM Review '17-01 Vol.20 No.01
https://doi.org/10.22670/knom.2017.20.1.9

SCADA 시스템의 주기성과 발생순서 정보를 반영한 Structured ACL 모델에 관한 연구

정 우 석◆, 윤 정 한*, 김 신 규**, 구 영 훈***, 이 성 호****, 김 명 섭°

A Study on Structured ACL Model Reflecting Periodicity and Sequence of SCADA System

Woo-Suk Jung◆, Jeong-Han Yun*, Sin-Kyu Kim**, Young-Hoon Goo, Sung-Ho Lee, Myung-Sup Kim°

요 약

SCADA 시스템은 정해진 기기들이 제한된 통신만을 반복적이고 주기적으로 사용하며 동작한다. 이러한 특징 때문에 화이트리스트 기반 보안기법이 많이 사용되고, 실제 현장에 가장 많이 적용되어 있는 것이 화이트리스트 기반 Static ACL을 이용한 접근제한 기법이다. Static ACL은 보안효과가 높은 장점이 있지만 표현력이 너무 단순하여 Dynamic Allocated Port를 사용하는 통신을 표현하는데 한계를 가진다. 또한 제어기기의 통신 특성을 전부 반영하지 못하고, 생성된 Static ACL은 분기 별로 한번 사용되더라도 1년 365일 오픈해야한다는 세 가지의 한계점을 가진다. 우리는 Static ACL에 SCADA 시스템의 기계적이고 반복적인 통신특성을 반영하기 위하여 통신별 주기성 그리고 통신간의 고정된 발생순서 정보를 확장한 Structured ACL을 제안하였다. 제안하는 시스템은 실제 SCADA 네트워크 트래픽에 적용을 통하여 타당성을 증명하였다.

Key Words : Industrial Control System, SCADA, Whitelist, Traffic Locality, Frequent Pattern Mining

ABSTRACT

The SCADA system works by repeatedly and periodically using only limited communication devices. Because of this feature, whitelist based security techniques are widely used, and access restriction method using whitelist based static ACL is most commonly applied in the field. Static ACL has an advantage in security, but their expressiveness is too simple to express communication using dynamic allocated port. In addition, it does not reflect all the communication characteristics of the control device, and the generated static ACL should always be open regardless of the frequency of use. We have proposed a structured ACL that extends the fixed generation sequence information between the communication and communication-specific periodicity to reflect the mechanical and repetitive communication characteristics of the SCADA system in the static ACL. We demonstrate the feasibility of the proposed system in this paper by applying the real SCADA network traffic.

※이 논문은 2015년도 정부(교육부)의 재원으로 한국연구재단의 기초연구사업(No.2015R1D1A3A01018057) 및 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No. 2017-0-00513)
◆ First Author : Korea University Department of Computer and Information Science, hary5832@korea.ac.kr
° Corresponding Author : Korea University Department of Computer and Information Science, tmskim@korea.ac.kr
* National Security Research Institute, dolgam@nsr.re.kr
** National Security Research Institute, skkim@nsr.re.kr
*** Korea University Department of Computer and Information Science, gyh0808@korea.ac.kr
**** Korea University Department of Computer and Information Science, gaek5@korea.ac.kr
논문번호 : KNOM2017-01-02, Received April 20, 2017; Revised May 25, 2017; Accepted July 20, 2017

Ⅰ. 서 론

제어시스템은 특정 산업현장 전체 또는 산업 단지를 전반적으로 감시하고 제어하기 위하여 다양한 기간 시설과 산업에서 사용되고 있는 컴퓨터 기반의 시스템이다. 제어시스템 중 SCADA(Supervisory Control And Data Acquisition) 시스템은 1960년대에 원격지의 시스템을 감시하고 제어하기 위하여 사용되기 시작하였다. SCADA 시스템은 폐쇄망에서 비공개 프로토콜을 사용해 동작하였기 때문에 지금까지 보안과 관련한 위협이 크지 않았다. 하지만 최근 비즈니스 시스템과의 통합으로 인해 네트워크가 확산됨에 따라 폐쇄망과 업무망 그리고 인터넷망 간의 연결이 증가하였으며, 외부 협력업체와의 협업으로 인한 연결성과 다양한 매체의 이용 등으로 인한 보안상 취약점들이 증가하고 있는 추세이다. 화이트 리스트 보안 기법은 안전이 증명된 것만을 허용하는 것으로 악의성이 입증된 것들을 차단하는 블랙리스트(blackllist) 보안 기법과 상반되는 보안 방식이다. 화이트리스트는 보안성은 높지만 편의성을 심각하게 저해할 수 있어 제한적인 영역에서만 사용되었지만, 특히 리소스가 적고 시스템 동작 패턴 또는 네트워크 통신 트래픽이 규칙적인 제어시스템 환경에서 보안을 담보할 수 있는 효율적 방안으로 주목 받고 있다.

SCADA 네트워크는 정해진 기기들이 제한된 통신만을 반복적이고 주기적으로 사용하며 동작한다. 하지만 현재 화이트리스트 보안 기법에서 일반적으로 사용되는 Static ACL은 단순한 5-tuple 정보만을 포함하므로 SCADA 시스템의 기계적이고 반복적인 통신특성을 반영하지 못한다는 한계점을 가진다.

본 논문에서는 SCADA 시스템의 기계적이고 반복적인 통신특성을 반영하기 위해서 Periodicity와 sequence의 두 가지 특징을 반영한 ACL(Access Control List) 간의 순서 연관관계를 추출한다. 추출된 순서 연관관계는 Static ACL에 추가하여 SCADA 시스템의 기계적 통신 특성이 반영된 보다 구체적인 화이트리스트인 Structured ACL를 구성한다.

2장에서는 관련 연구에 대해 서술한다. 3장에서는 기존 Static ACL이 가지는 한계점과 본 논문에서 제안하는 Structured ACL을 구성하는 다양한 ACL들에 대해 정의한다. 4장에서는 제안하는 시스템에 대해 서술하고, 5장에서는 실험과 결과를 6장에서는 결론 및 향후 연구에 대해 차례로 서술한다.

Ⅱ. 관련 연구

Static ACL은 네트워크 스위치, 방화벽 등 다양한 보안제품에서 사용된다. 국제적인 정보보호 기업인 CISCO, Juniper Networks 그리고 McAfee 등은 Static ACL을 자신들이 만든 네트워크 스위치, 방화벽 등에 적용하고 있다. 또한 무선랜 시장에서 화두가 되었던 ArubaNetworks 역시 무선랜 스위치에 Static ACL을 사용하고 있다. 이처럼 Static ACL은 제어시스템 뿐만 아니라 다양한 곳에서 활용되고 있다. 하지만 개별 장비에 적용하기 위해 사람이 Static ACL을 작성하는 것은 매우 어렵기 때문에 Static ACL을 자동으로 생성하는 방법에 대한 다양한 연구가 진행 되고 있다[2][8].

Choi Seungoh et al[2]의 연구에서는 SCADA 시스템의 TCP 핸드쉐이킹이나 공통된 포트 등의 일반적인 특성이 플로우 기반 화이트리스트를 생성하는데 불충분한 점을 문제로 제기하며, 트래픽의 지역성을 기반으로 한 플로우 화이트리스트를 자동으로 생성하는 방법을 제안했다. 해당 연구에서는 플로우의 5-tuple 정보만을 사용하여 Locally Frequently-Used Port와 Degree Centrality를 활용하여 화이트리스트를 자동으로 생성하고 이를 실제 SCADA 시스템 트래픽에 적용하여 검증하였다.

하지만 기존의 Static ACL 자동 생성에 관한 연구를 통해 생성된 Static ACL은 SCADA 네트워크의 특징을 표현하는데 있어 표현력에 한계가 존재한다. 또한 ANY-ANY 같은 규칙이 많이 생성된다는 점과 한 번만 발생한 통신이라도 규칙으로 생성한다는 등의 단점이 존재한다.

본 논문에서는 기존 Static ACL 자동 생성에 관한 연구가 가지는 Static ACL 표현력의 한계와 위에서 언급한 단점을 극복하기 위하여, Static ACL에 순서 연관관계를 추가하여 SCADA 시스템의 특징을 반영한 Structured ACL을 제안한다.

Ⅲ. Static ACL과 그 한계점

본 장에서는 기존 Static ACL 모델에 대하여 정의하고, Static ACL 모델이 가지는 세 가지 한계점에 대하여 서술한다.

1. Static ACL

일반적으로 화이트리스트를 표현하는 Static ACL은 5-tuple 기반의 단순한 네트워크 및 시스템에 대한 접근 권한 리스트이다. 따라서 단순한 5-tuple 정보만을 포함하고 있으며, ACL간의 순서나 조건을 표현 하지 못한다.

2. Static ACL의 한계점

현재의 화이트리스트를 사용한 제어시스템의 보안 기법은 대부분 Static ACL 모델을 사용하고 있다. 하지만 Static ACL 모델을 기반으로 한 화이트리스트는 세 가지 한계점을 가지고 있다.

첫 번째 한계점은 SCADA 시스템에서 사용되는 FTP나 OPC(OLE for Process Control)와 같은 Dynamically Allocated Port를 사용하는 통신의 경우 Static ACL 모델로 표현하는 방법은 ANY-ANY 규칙을 사용하는 것이 유일하다는 것이다. 하지만 ANY-ANY 규칙을 허용하게 되면 해당 IP 간의 모든 연결에 대하여 모든 포트를 항시 오픈해야한다. 이는 기존 해당 IP 간에 생성된 다른 ACL 규칙들이 무의미 해지게 되는 것을 의미한다. 특히 FTP를 사용하는 서버는 해당 서버에 접근하는 모두에게 오픈 될 가능성이 있다.

두 번째 한계점은 제어기기의 통신 특성을 전부 반영하지 못 한다는 점이다. 제어기기는 매일 계속 정해진 작업을 반복적으로 수행해야 하는 기기인데, Static ACL 모델을 사용하면 정해진 작업을 수행한다는 조건만 표현된다.

세 번째 한계점은 빈도에 상관없이 Static ACL 모델로 작성된 모든 규칙은 항시 오픈 된다는 점이다. 예를 들어 SCADA 네트워크 장비 관리 서비스에 대한 관리 단말 지정의 경우 1년에 몇 회 사용하지 않지만 1년 내내 오픈해 주어야 한다는 점이다.

Ⅳ. 제안하는 Structured ACL 모델 및 추출 시스템

본 장에서는 Static ACL이 가지는 세 가지 한계점을 극복하기 위하여 본 논문에서 제안하는 Structured ACL 모델과 이를 구성하는 다양한 ACL들에 대해서 정의한다. 또한 Static ACL 간의 연관관계를 추출하고 이를 Structured ACL로 표현하기 위한 방법에 대해 기술한다.

1. Structured ACL Model

본 논문에서는 위에서 서술한 Static ACL이 가지는 세 가지 한계점과 제어시스템 트래픽의 특징을 표현하는 표현력을 확장한 Structured ACL을 정의하고 사용한다.

Structured ACL은 제어시스템 네트워크 트래픽으로부터 연관관계를 추출하여 정의 된 순서를 가지는 ACL의 집합으로 빈번한 Static ACL들을 추출한 Frequent ACL, Static ACL들의 주기를 추출한 Periodic ACL, Static ACL들 간의 연관관계를 추출한 Sequential ACL 그리고 세 가지 ACL을 통해 추출 되지 않은 Static ACL의 4 가지 ACL을 통해 구성된다. Structured ACL과 Structured ACL을 구성하는 Static ACL, Frequent ACL, Periodic ACL, Sequential ACL은 표 1과 같이 정의된다.

표 1. ACL 정의
Table 1. Definition of ACL

용어	정의
ACL	사용자들이 디렉토리나 파일과 같은 특정 시스템에 접근 할 수 있는 권한을 설정해 놓은 리스트
Static ACL	5-tuple 기반의 단순한 네트워크 및 시스템에 대한 리스트
Frequent ACL	빈번한 Static ACL의 반복 주기를 추출한 리스트
Periodic ACL	Static ACL의 반복 주기를 추출한 리스트
Sequential ACL	Static ACL 사이의 순서나 조건 등의 연관관계를 추출한 리스트
Structured ACL	Static ACL 사이의 순서나 조건 등의 연관관계를 표현한 리스트 {Static ACL , Frequent ACL, Periodic ACL, Static ACL}

Frequent ACL과 Periodic ACL은 모두 ACL의 주기를 추출한 ACL이다. 하지만 실제 트래픽을 분석한 결과 아주 자주 사용하는 ACL에 의해 sequence가 너무 많이 추출되는 문제가 발생하였고, 불필요한 sequence를 제거하고 너무 빈번히 발생하는 ACL들이 Structured ACL로 구성되어 오히려 시스템에 부하를 발생하는 경우를 방지하기 위하여 Frequent ACL과 Periodic ACL을 구분하였다.

그림 1은 Static ACL, Frequent ACL, Periodic ACL, Sequential ACL 그리고 Structured ACL의 관계를 도식화 한 것이다. 기존 Static ACL에서

추출한 Frequent ACL, Periodic ACL 그리고 Sequential ACL과 추출되지 않은 Static ACL이 Structured ACL을 구성하게 된다.

반복 주기와 빈도수 그리고 ACL들 간의 순서 정보를 가지는 Frequent ACL, Periodic ACL 그리고 Sequential ACL의 조합으로 구성되므로 Structured ACL 역시 해당 요소들을 포함하게 된다.

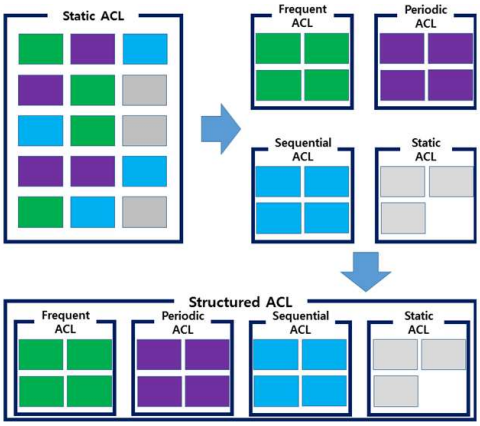


그림 1. 정의된 ACL들 간의 관계
Fig. 1. Relationship between defined ACL

2. 연관관계 추출

본 논문에서 제안한 Structured ACL은 Static ACL에 periodicity와 sequence를 추출한 순서 연관관계를 추가하여 구성된다. Structured ACL을 구성하기 위하여 Static ACL 간의 순서 연관관계를 추출하는 과정은 세 단계로 이루어진다.

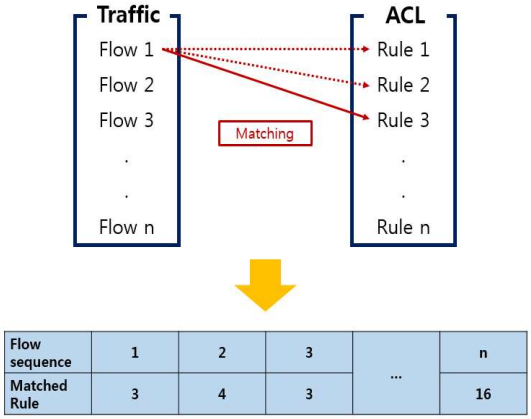


그림 2. 트래픽에 ACL을 매칭하는 과정
Fig. 2. Process of match ACL to traffic

첫 번째 단계에서는 Structured ACL 구성에 사용될 Static ACL에 각각 고유번호를 부여한다. 이는 이후 Transaction을 생성하기 위해 Static ACL을 트래픽에 매칭하여 매칭 정보를 추출하기 위해 사용된다.

두 번째 단계는 Transaction 생성 단계이다. 본 단계에서는 Static ACL과 Static ACL을 생성하는데 사용한 트래픽을 사용하여 Transaction을 생성한다. Transaction은 그림 2와 같이 트래픽을 플로우 단위로 Static ACL에 매칭시키고, 매칭되는 Static ACL의 고유 번호를 플로우의 시간 순서대로 나열한 것이다. 그림 2는 연관관계 추출의 두 번째 단계를 도식화 한 것이다. 입력 트래픽의 각 플로우에 Static ACL을 매칭하여 매칭된 ACL의 고유번호를 시간 순서대로 나열한다.

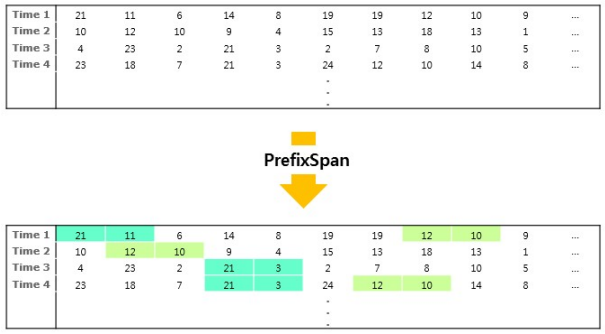


그림 3. 순서 연관관계 추출 개념
Fig. 3. Concept of sequence correlation extraction

세 번째 단계는 순서 정보 추출 단계이다. 본 단계에서는 Sequential Pattern Mining 알고리즘인 PrefixSpan 알고리즘을 사용하여 Transaction의 ACL 매칭 순서 사이의 연관관계를 추출한다. 그림 3은 PrefixSpan 알고리즘을 사용하여 Transaction으로부터 ACL 매칭 순서 사이의 연관관계를 추출하는 과정을 도식화 한 것이다. 21번 ACL 다음에는 11번 혹은 3번 ACL이 매칭되고, 12번 ACL 다음에는 10번 ACL이 매칭되는 것을 확인 할 수 있다. 그림 4는 12번 ACL 다음에 10번 ACL이 매칭된다는 순서 연관관계를 표현한 Structured ACL 예시이다. 해당 Structured ACL은 독립적인 12번과 10번 ACL을 연관관계 추출을 통하여 하나의 계층적 구조를 가지는 Structured ACL로 구성되었다. 그림 4의 Structured ACL은 계층적 구조를 통하여 12번 ACL이 매칭된 경우에만 10번 ACL을 매칭하므로 방화벽의 불필요한 탐지를 줄일 수 있다. 또한 각

ACL의 Max duration 정보를 포함하므로 해당 ACL이 매칭 된 경우에 Max duration 시간만 해당 규칙을 열어주면 되므로 모든 규칙들을 항시 열어야하는 위험을 피할 수 있다.

{(x.x.1.11, ANY <-TCP-> x.x.1.42, 382, n), (x.x.1.41, 52053 <-TCP-> x.x.1.43, ANY, n)}

그림 4. Structured ACL 모델 예시
Fig. 4. Example of structured ACL Model

3. Structured ACL 추출 시스템

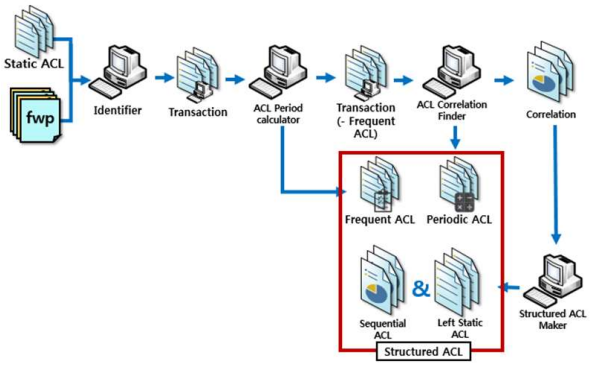


그림 5. Structured ACL 모델 추출 시스템
Fig. 5. Structured ACL model extraction system

그림 5는 Static ACL과 Static ACL을 추출하는데 사용된 트래픽을 입력으로 본 논문에서 제안하는 Structured ACL을 구성하기 위하여 Frequent ACL, Periodic ACL 그리고 Sequential ACL을 추출하는 과정을 도식화 한 것이다.

첫 번째 모듈인 Identifier는 Periodic ACL, Frequent ACL 그리고 Sequential ACL을 추출하기 위하여 Static ACL과 트래픽 파일을 매칭시켜 Transaction을 생성한다.

$$Frequency = \frac{(Total\ Input\ Time)}{(Match\ Count)}$$

그림 6. Frequency 계산식
Fig. 6. Frequency calculate formula

두 번째 모듈인 ACL Period Calculator는 1분 이하의 주기를 가지고 있는 Frequent ACL을 추출하며, 너무 빈번히 발생하는 Static ACL을 Structured ACL 구성 과정에서 제외하기 위해 사용된다. 그림 6의 수식은 총 입력 시간을 각 ACL의 매칭 횟수로 나누어 각 ACL이 얼마의 시간을 가지고 반복되는지를 계산하는 수식이다. 본 논문에서는 실험을 통해 ACL의 반복 주기가 30초 이하인 경우 빈번하다고 판단하여 해당 ACL을 연관관계 추출 과정에서 제외한다.

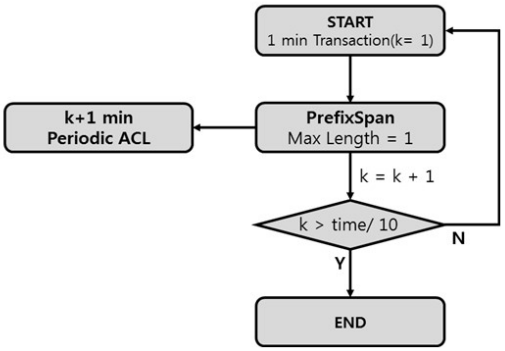


그림 7. PrefixSpan 적용을 통한 Periodic ACL 추출
Fig. 7. Periodic ACL extraction by applying the PrefixSpan

세 번째 모듈인 ACL Correlation Finder는 PrefixSpan 알고리즘을 적용하는 모듈로써 Identifier에서 생성한 Transaction에서 ACL Period Calculator에서 추출한 Frequent ACL을 제외한 내용을 입력으로 Periodic ACL을 추출한다. 그림 7은 ACL Correlation Finder에서 각 분 단위별로 반복되는 ACL을 추출하는 과정을 도식화 한 것이다. 최초로 1분 단위 Transaction을 입력으로 1분의 주기로 반복되는 Periodic ACL을 추출한다. 다시 2분 단위 Transaction을 입력으로 2분의 주기로 반복되는 Periodic ACL을 추출하며, 이 과정을 트래픽의 총 입력시간의 10분의 1이 될 때까지 반복한다. 해당 과정에서 Periodic ACL의 반복주기를 1분 단위로 설정한 것은 실험을 통해 추출된 ACL들의 반복주기가 30초를 넘어가는 경우에 대부분 1분 또는 10분 주기로 나타나는 경향이 있어, 임의로 설정하였다.

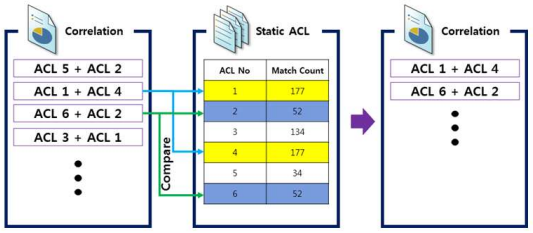


그림 8. Sequential ACL의 매칭카운트 비교
Fig. 8. Matching count comparison in Sequential ACL

네 번째 모듈인 Structured ACL Maker는 ACL

Correlation Finder를 통해 추출한 Correlation을 Match Count와 서버/클라이언트 IP의 인과관계를 기준으로 판단하여 Sequential ACL을 구성한다. 그림 8은 Match Count를 기준으로 Correlation을 판단하는 과정을 도식화 한 것이다. 첫 번째 Correlation인 ACL 5 + ACL 2는 각 ACL의 매칭 횟수가 34번과 52번으로 ACL 5번이 매칭된 이후에 반드시 ACL 2번이 매칭되었다고 볼 수 없으므로 Correlation 목록에서 제외되게 된다.

Match Count를 비교한 이후에는 Correlation으로 추출된 ACL들 간의 IP 인과관계를 확인한다. IP의 인과관계란 두 ACL들 간의 src/dst IP 비교를 통해 선행 ACL에 의해 후행 ACL이 일어났는지를 판단하는 단계이다. 본 단계에서는 선행 ACL의 src/dst IP와 후행 ACL의 src/dst IP들 간에 중복된 IP가 하나라도 있는지 확인한다. 중복된 IP가 하나 이상 있는 경우만 해당 ACL들 간에 연관관계가 존재한다고 판단한다.

최종적으로 Static ACL으로부터 추출한 Frequent ACL, Periodic ACL, Sequential ACL과 세 개의 ACL을 통해 추출되지 않은 나머지 Static ACL을 통해 Structured ACL이 구성된다.

V. 실험

본 장에서는 본 논문에서 제안한 시스템의 타당성을 증명하기 위해 실제 SCADA 네트워크 트래픽에 적용 실험한다. 표 2는 실험에 사용된 SCADA 네트워크 트래픽으로 A 사이트를 제외한 나머지 6개의 사이트 들은 모두 각 사이트에서 12시간씩 수집한 트래픽이다.

표 2는 각 사이트별로 본 논문에서 제안한 시스템을 적용한 결과를 정리한 표이다. A 사이트의 경우 전체 Static ACL의 개수는 655개이고, 추출된 Frequent ACL은 82개, 1분부터 5분까지의 Periodic ACL은 각각 3, 9, 7, 3, 5개, Sequential ACL은 30개로 중복을 제외한 ACL 원소의 개수는 18개였으며, 추출되지 않은 Static ACL은 637개이다.

표 2. 실험에 사용된 트래픽
Table 2. Test set spec

Site	Size of Traffic	No. of Flows
A	492 GB	9,198,938
B	14.6 GB	234,259
C	3.14 GB	13,729
D	10.4 GB	94,072
E	19.0 GB	1,709,279
F	2.79 GB	68,624
G	2.50 GB	637,199

실험의 결과 A 사이트 트래픽으로 추출한 총 655개의 Static ACL들 중 82개의 ACL의 Frequency가 30 이하로 추출되어 Structured ACL 추출 과정에서 제외 되었다. 또한 27개의 ACL에서 5분 이하의 반복주기를 추출 할 수 있었다. 추출된 ACL 간의 연관관계는 총 30개였으며, 중복을 제거한 고유한 ACL의 개수는 총 18개였다. 실험을 통해 전체 655개의 Static ACL들 중 18개의 ACL 간의 연관관계를 추출하여 계층적 구조를 가지는 Structured ACL로 구성할 수 있었다. 실험에 사용한 A 사이트의 경우 총 655개의 Static ACL 중 단 18개의 Static ACL들이 Sequential ACL로 구성되었지만, 해당 Static ACL들은 5분 이하의 반복주기를 가지므로 타 Static ACL들에 비하여 방화벽에서 매칭 되는 빈도수가 높다. 따라서 방화

표 3. 실험 결과
Table 3. Experiment result

Site	A	B	C	D	E	F
Static ACL	655	38	90	187	460	440
Frequent ACL	82	4	11	35	19	100
1min Periodic ACL	3	0	0	4	0	3
2min Periodic ACL	9	0	1	7	3	0
3min Periodic ACL	7	0	0	0	66	0
4min Periodic ACL	3	0	0	1	36	0
5min Periodic ACL	5	2	2	6	99	13
Sequential ACL	30(18)	2(2)	0	4(8)	3829(196)	303(16)
Left Static ACL	637	36	90	179	264	424

벽에서 1st level에서 매칭 해야 하는 ACL의 개수가 감소하고, 이에 따라 1st level에서의 search space가 줄어드는 효과를 얻을 수 있을 것으로 예상한다. 또한 ACL들 간의 순서 연관관계 추출을 통하여 제어 기기의 통신 특성을 어느 정도 반영하였다고 판단된다.

Ⅵ. 결론 및 향후 연구

본 논문에서는 Static ACL에 periodicity와 sequence를 추출한 순서 연관관계를 추가한 Structured ACL 모델을 제안하고, 이를 실제 SCADA 트래픽에 적용하여 각 규칙들 간의 순서 연관관계, Duration, S/C 정보, 반복 주기 등 다양한 정보를 포함한 Structured ACL 모델이 정상적으로 추출되는 것을 확인하였다.

본 논문에서 제안한 Structured ACL 모델이 가지는 다양한 정보들을 활용하면 기존 Static ACL 모델이 가지는 첫 번째 문제점인 Dynamic Allocated Port를 사용하는 경우에 ANY-ANY 규칙으로만 표현 가능한 문제를 해당 통신에 의해 생성된 ACL들을 발생 순서에 의해 Structured ACL로 잘 구성할 수 있다면 충분히 해결할 수 있을 것으로 생각된다. 두 번째 문제점인 제어기기의 통신 특성 반영은 ACL들 간의 발생순서 관계를 활용하여 Structured ACL을 구성함으로써 해결 하였고, 세 번째 문제점인 빈도수에 상관없이 항시 오픈해야 한다는 문제점 역시 2nd, 3rd level에 있는 규칙들은 해당 Structured ACL의 1st level에 있는 규칙이 매칭된 이후에 특정 시간동안만 오픈함으로써 해결할 수 있었다.

또한 방화벽에서 트래픽에 ACL을 매칭시에 탐색하는 1st level의 search space가 ACL들이 Structured ACL로 구성되는 비율에 따라 줄어드는 효과를 얻을 수 있을 것으로 예상된다.

향후 연구로는 플로우 내부의 비정상 탐지를 위하여 플로우 내부의 패킷 통계 정보를 모델링하여 플로우 내부의 비정상을 탐지하는 방법에 대해 연구할 계획이다.

References

[1] Yun, Jeong-Han, et al. "Burst-based anomaly detection on the DNP3 protocol." International Journal of Control and Automation 6.2 (2013): 313-324.

[2] Choi, Seungoh, et al. "Traffic-Locality-Based Creation of Flow Whitelists for SCADA Networks." Critical Infrastructure Protection IX. Springer International Publishing, 2015. 87-102.

[3] 유형욱, 윤정한, 손태식. "제어시스템 보안을 위한 whitelist 기반 이상징후 탐지 기법." 한국통신학회논문지 38.8 (2013): 641-653

[4] Lim, Y. H., Yoo, H., & Shon, T. (2013). IEC 61850 변전소 네트워크에서의이상 징후 탐지 연구. Journal of The Korea Institute of Information Security & Cryptology (JKIISC), 23(5).

[5] Schneider, Johannes, Sebastian Obermeier, and Roman Schlegel. "Cyber security maintenance for SCADA systems." Proceedings of the 3rd International Symposium for ICS & SCADA Cyber Security Research. British Computer Society, 2015.

[6] Igure, V.M. and Laughter, S.A. and Williams, R.D.: Security issues in SCADA networks. Computers & Security. vol. 25, no. 7, pp. 498 - 506, Elsevier (2006)

[7] Jung, Woo-suk, et al. "Whitelist representation for FTP service in SCADA system by using structured ACL model." Network Operations and Management Symposium (APNOMS), 2016 18th Asia-Pacific. IEEE, 2016.

[8] Barbosa, Rafael Ramos Regis, Ramin Sadre, and Aiko Pras. "Flow whitelisting in SCADA networks." International journal of critical infrastructure protection 6.3 (2013): 150-158.

[9] Kang, Dong-Ho, et al. "Whitelist Generation Technique for Industrial Firewall in SCADA Networks." Frontier and Innovation in Future Computing and Communications. Springer

정 우 석 (Woo-Suk Jung)

2015년 : 고려대학교 컴퓨터정보학과 졸업
2015년 ~ 현재 : 고려대학교 컴퓨터정보학과 석사과정
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

윤 정 한 (Jeong-Han Yun)

2001년 2월 KAIST 전산학과 졸업
2003년 2월 KAIST 전산학과 석사
2011년 2월 KAIST 전산학과 박사
2010년 12월~현재 ETRI부설국가보안기술연구소 선임연구원
<관심분야> 프로그램 분석, 제어시스템 네트워크 침입탐지

김 신 규 (Sin-Kyu Kim)

2000년 2월: 연세대학교 기계전자공학부 졸업
2002년 2월: 연세대학교 컴퓨터과학과 석사
2014년 2월: 연세대학교 컴퓨터과학과 박사
2003년 12월~현재: ETRI부설국가보안기술연구소 선임연구원/실장
<관심분야> 스마트그리드 보안, 국가기반시설 보안, 취약점 분석

구 영 훈 (Young-Hoon Goo)

2016년 : 고려대학교 컴퓨터정보학과 학사
2016년~현재 : 고려대학교 컴퓨터정보학과 석사과정
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

이 성 호 (Sung-Ho Lee)

2016년~현재 : 고려대학교 컴퓨터 정보학과 석사과정
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 트래픽 분류

김 명 섭 (Myung-Sup Kim)

1998년 : 포항공과대학교 전자계산학과 학사
2000년 : 포항공과대학교 전자계산학과 석사
2004년 : 포항공과대학교 전자계산학과 박사
2006년: Dept. of ECS, Univ of Toronto Canada
2006년~현재 : 고려대학교 컴퓨터정보학과 교수
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 멀티미디어 네트워크

KNOM Review '17-01 Vol.20 No.01
https://doi.org/10.22670/knom.2017.20.1.17

HPC 응용의 런타임 실행 패턴에 따른 소프트웨어 정의 계산 구조 분석

오 지 선, 이 효 정, °김 윤 희

An Analysis of Software-defined Compute Architecture for HPC Application Runtime Execution Patterns

Ji Sun Oh, Hyojeong Lee, °Yoonhee Kim

요 약

최근 기기들의 고성능화에 따라 데이터양의 폭증 및 트래픽 동적 변이가 심화되고 더불어 클라우드 컴퓨팅의 확산으로 데이터 서비스의 다양성을 고려한 소프트웨어 중심의 자원 관리가 중요해지고 있다. 대용량 데이터를 고속으로 처리해야 하는 데이터 집약적인 HPC 응용은 실행 요구사항 중 실시간 분석 및 예측을 요구하는 경우 적시적 데이터 처리 및 분석과 결과 검증에 시간제한 요소를 고려하여 소프트웨어 정의 중심의 계산 및 전송구조를 정의하고 실행해야 한다. 본 논문에서는 고성능 데이터 전송 환경인 ScienceDMZ에 대한 국내외 동향과 소프트웨어 정의 기술을 바탕으로 실시간 처리 및 결과 검증 문제를 해결을 요구하는 응용에 대한 실행 패턴을 분석한다. 이를 위하여 기상 예보 계산 모델의 실행 패턴 분석과 시간적인 한계를 가진 고성능 연구를 위한 적시적 데이터 저장과 계산 실험의 통합적 연동의 필요성을 제안한다. 또한 응용 프로그램의 이해를 기반으로 자원 사용 사례 패턴에 따라 실행 자원을 동적으로 할당 및 반환하는 것이 중요하다. 본 논문은 데이터 집약적인 HPC 응용 사례로 WRF 모형을 선정하여 수집된 입력 데이터의 변화에 따른 분석의 런타임 특성을 분석한다. 분석 결과를 통해 적시적으로 데이터를 할당하는 HPC 응용을 위한 소프트웨어 정의 계산 구조를 정의하고 중요성을 제고하였다.

Key Words : HPC, ScienceDMZ, SDN, SDC, WRF

ABSTRACT

In the advent of high performeding and sophisticated devices, explosion of large-scale data and dynamic variation of traffic change are intensifying. Cloud computing enables software-based resource management considering diversity of data services over on-demand computing infrastructure. As real-time analysis of large-scale data and forecasting of data-intensive HPC application are required, software definition technology should be applied to temporal data processing, analysis, and result verification. In this paper, we have investigated trends and software definition technologies of ScienceDMZ, which provides high speed data transmission environment for scientific big data. In order to solve the problem of real-time processing and result verification, we propose a temporal application and integration of data storage and calculation experiment for high performance research with time limit through weather forecast infrastructure. It is also important to identify resource use case patterns based on the understanding of application programs and assign them appropriately. This paper analyzes the runtime characteristics of input data by selecting WRF model as a data - intensive HPC application. We defined the execution structure of the HPC application that allocates the data in a timely manner through the analysis results and raised the importance.

※이 논문은 2017 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2017R1A2B4005681)
° Corresponding Author : Dept. of Computer Science, Sookmyung Women's University, yulan@sookmyung.ac.kr
논문번호 : KNOM2017-01-04, Received June 13, 2017; Revised July 20, 2017; Accepted August 10, 2017

Ⅰ. 서 론

소프트웨어 정의 기술은 IT 인프라를 동적으로 관리하여 응용 실행의 최적화를 위한 적시적 자원 관리 기술을 제공한다. 최근 서비스 특성에 따라 효율적인 네트워크 구조 개선을 위한 OpenFlow[1]와 같은 오픈소스 기반의 SDN(Software-defined Network) 기술을 적용한 동적 네트워크 관리 연구가 활발하다. 이와 함께 기기들의 고성능화에 따른 데이터의 폭증, 트래픽 동적 변이 심화와 더불어 클라우드 컴퓨팅의 확산과 서비스의 다양성을 고려한 소프트웨어 중심의 관리가 중요해지고 있다. 이러한 대규모의 데이터를 처리하는 과학 계산 응용의 요구사항을 충족시키기 위해 SDN을 시작으로 스토리지, 인프라 그리고 데이터센터 전체 영역으로 확산되었다. 고성능 환경을 활용한 SDC(Software-defined Compute), SDS(Software-defined Storage) 기술이 새롭게 정의되고, 관련 연구가 활발히 진행 중이다.

기존의 HPC연구는 필요 자원이 독점적으로 활용 가능한 시점에 분석을 진행하는 일괄 배치 작업 형태를 활용하고 있다. 그러나 기상 재난 예측과 같은 과학계산 응용은 일괄 배치 처리로 끝나기 어렵고, 연속적인 여러 단계를 거쳐 장시간 처리 되는 경우가 많다. 이러한 응용은 실시간에 가까운 분석 및 결과 활용을 요구하기 때문에 런타임 특성에 따라 필요로 하기 때문에 자원의 동적 처리를 요구한다. 그러나 과학계산 응용의 자원 및 실행 인프라의 실시간 성능 처리와 결과 검증 문제에 대한 연구가 부족한 상황이다. 따라서 전송, 보안, 인프라의 실시간 성능 처리 및 결과 검증 문제를 해결하여 시간적 제한을 가지고 결과를 도출해야 하는 고성능 연구 환경이 필요하다.

본 논문은 국내외에서 진행되고 있는 고성능 데이터 전송이 가능한 ScienceDMZ 환경 연구에 대한 조사와 관련 소프트웨어 정의 기술(SDN, SDC)기술에 대해 설명하였다. 실시간 예측 분석의 일례로 기상 재난 예측 분석 시나리오를 소개하고 실시간 침수 예보를 가능토록 하는 소프트웨어 정의 인프라 구조를 제안하였다. 시나리오를 바탕으로 대기 연구 및 운영예측 응용을 활용하여 실시간 예측 분석 성능에 영향을 미치는 요소를 파악하기 위한 실험을 진행하였다. 이를 바탕으로 과학응용 데이터 집중형 응용을 효율적으로 수행하는 인프라를 구축하기 위하여 소프트웨어 사용자 정의 기술인 SDC, SDN을 적용하여 대용량 데이터 처리가 필요한 HPC 응용 실행 구조를 제안한다.

본 논문의 나머지 부분은 다음과 같이 다섯 개의 순서로 구성되어 있다. 제 2장에서는 관련 연구에 대해 소개하고 제 3장에서는 기상 재난 예제 시나리오를 통해 실시간 예보 인프라 구조를 설명한다. 제 4장에서는 기상 응용의 실시간 예측 성능에 영향을 미치는 요소 분석 실험과 그 결과를 보인다. 제 5장에서는 실시간 과학 응용을 위한 소프트웨어 의 정의 HPC응용 실행 구조를 제안한다. 마지막 결론으로 향후 구축할 인프라에 대한 계획을 포함한다.

Ⅱ. 관련 연구

2.1. ScienceDMZ

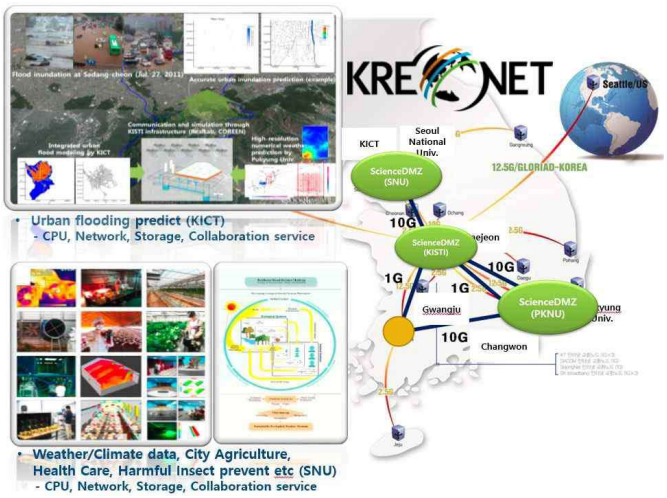


그림 1. 국내 대기과학분야 ScienceDMZ기반의 고성능 인프라[2]

Fig. 1. High-performance infrastructure based on ScienceDMZ in Korea's atmospheric science field[2]

미국 ESnet에서 제안된 ScienceDMZ는 데이터 집중형 과학을 위한 통합적 전송체계이며 개념적 용어이다. TCP환경에서의 최적화된 전송 성능의 보장을 위하여 트래픽 손실 없는 네트워크 상태를 추구한다. 망 분리는 초고속

전송 환경 구성을 위해 일반적인 데이터 트래픽과 ScienceDMZ 데이터 트래픽에 의해 이루어진다. 트래픽 손실이 없는 경우 LAN과 international 구간에 대해 Line Rate의 90% 이상의 전송 성능을 나타낸다. 또한 고성능 데이터 전송을 위해 저장, 캐싱, 보안 기능까지 감안한 전용 노드(DTN)가 필요하다. 데이터 전용 전송 노드 구축을 통해 과학데이터 전송 환경을 구축한다. 또한 전송 성능을 보장하기 위해 ScinceDMZ간 연결된 DTN의 성능을 실시간으로 검증하여 성능을 보장한다[2][3]. 이를 기반으로 고성능 데이터 전송 서비스를 확충하여 고성능 연구 환경을 제안하였다. 그림 1은 국내 ScienceDMZ기반의 고성능 인프라이다.

2.2. SDN, SDC

SDN 기술은 이종의 스위치와 라우터를 통해 손쉽게 네트워크를 제어할 수 있는 OpenFlow에 기반하여 출현하였다. 소프트웨어 중심의 네트워킹 기술 개발을 위해 네트워크를 컴퓨터 시스템으로, OpenFlow는 하드웨어와 네트워크 운영체제 사이를 연결하는 인터페이스로 하여 네트워킹 시스템 모델을 정의한다. SDN 기술은 네트워크 사용자가 하드웨어 기반 연구에서 벗어나 다양한 응용들을 손쉽게 개발할 수 있는 환경을 제공할 것이다[4]. SDC 기술은 SDN의 개념에서 확장되었다. 데이터센터의 새로운 트랜드 기술로 데이터와 자원을 작업에 올바르게 배치하는데 있어서 작업의 동적인 수행을 고려한다[5]. 응용 실행의 최적화를 위한 자원관리 기술인 SDC를 통해 데이터센터의 자원을 효율적으로 사용할 수 있다.

Ⅲ. 기상 재난 예제 시나리오

고성능 전송, 보안, 인프라의 실시간 성능 처리 및 결과 검증 문제가 해결되지 않는 기존 환경에서 시간적인 한계를 가지고 결과를 도출해야 하는 고성능 연구는 ScienceDMZ와 같은 인프라 구축이 필요하다. 응용의 예로는 국지성 태풍 예보와 같은 기상 재난 안전 관련 분석, 예측의 경우이다.

기상청은 정기적인 예보를 위해 과거 데이터를 바탕으로 다년간 개발된 기상모델을 이용하여 슈퍼컴퓨터와 같은 고성능 자원을 활용한 단기, 중기, 장기 예보를 실시하고 있다. 그러나 갑작스러운 기상 변화에 대한 분석 및 예측 서비스는 시간적 제약 하에 어려움이 있다. 그림 2는 실시간 침수 예보를 가상화 하여 제공하는 인프라 구조를 나타낸다. 과거의 재난, 지형, 기상정보와 변동하는 기상 데이터를 위성 및 레이더로 측정한다. 데이터 고속전송 및 분석을 위한 DMZ 기술을 통한 ICT인프라를 구축하여 지형 및 광역, 국지 구름 모델 데이터를 전송한다. 데이터를 통해 기상 재난 상황의 원인분석 및 모델을 구현하며 분석 결과를 통해 강우 예상 데이터를 전처리하고 기상 모델에 적용한다. 과학적 모델 기반 기상재난을 예측하여 개선방안을 제시하고 재발 방지를 위한 경보 인프라를 제공한다.

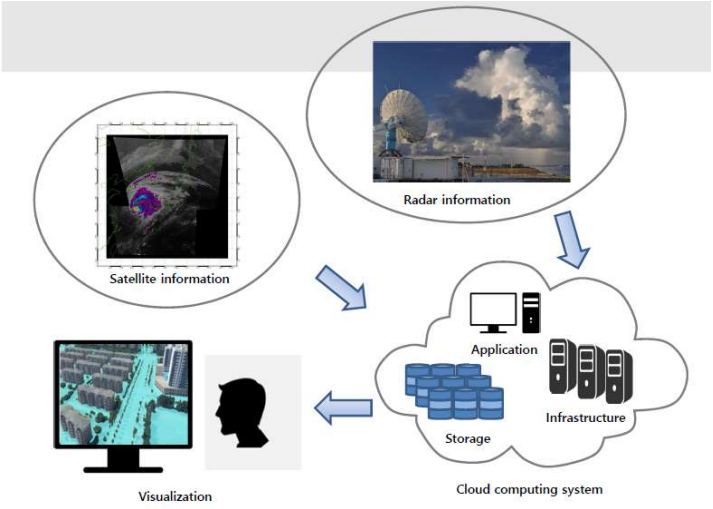


그림 2 . 실시간 침수 예보 인프라[6]
Fig. 2. Real-time flood forecasting infrastructure[6]

이처럼 국지적 태풍 예보, 도로별 침수, 홍수지역 예측 서비스 등과 같은 실시간 분석 및 적용 연구는 대용량 데이터의 이동, 충분한 고성능 자원, 새로운 분석 모델 개발 등을 필요로 한다. 대용량의 기상, 기후 데이터의 전송, 저장, 처리가 필요하며 데이터 분석 적용을 위한 데이터 흐름 연구가 필요하다. 고성능 데이터 전송이 가능한 ScienceDMZ 환경에서 데이터 저장과 계산 실험의 적시적 적용 및 통합적 연동이 가능한 상황을 제공해야 한다. 또한 응용

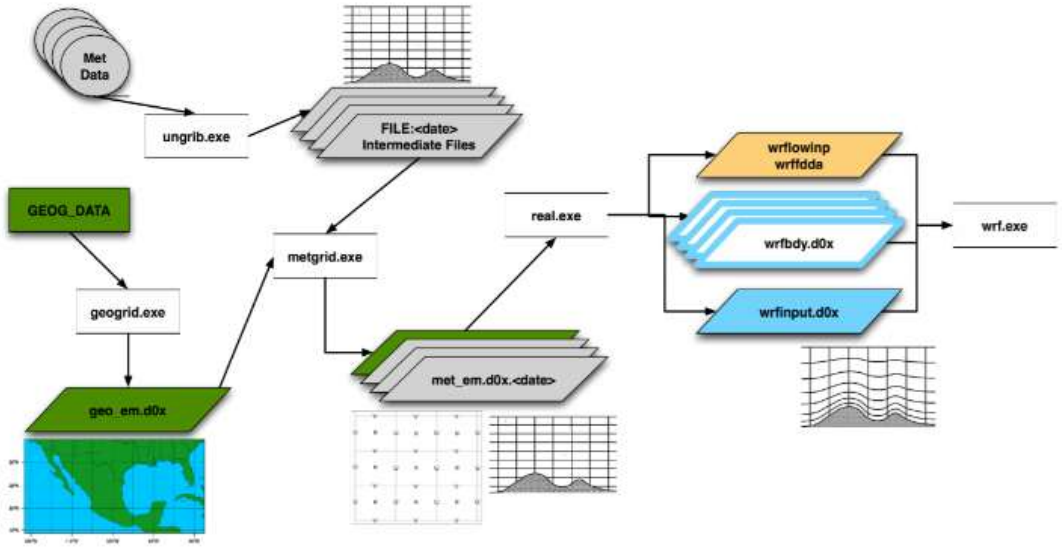


그림 3 . WRF 응용 실행 구조도[8]
Fig. 3. Execution structure of WRF application[8]

프로그램의 이해를 기반한 지능적 데이터 배치와 계산 배치가 우선적으로 필요하다.

Ⅳ. 대기 예측 모델 실행 부하 실험 및 분석

대기 연구 및 운영 예측 응용인 WRF(Weather Research and Forecasting)[7]는 전처리 단계인 WPS(WRF Pre-processing System)와 모형 본체인 WRF로 구성되어 이상실험(ideal case) 또는 실제 기상 환경에 대한 실험과 예보(real case)를 실시한다.

데이터 집약적인 응용 실험에 앞서 WRF 응용의 특성을 분석하였다. 그림 3은 WRF응용 실행 구조도이며 이에 따른 실행 과정은 다음과 같다. WPS에서는 WRF 본체에 사용될 도메인과 변수를 보간 한다. 그림 3의 geogrid.exe, ungrib.exe와 metgrid.exe 모듈이 이에 해당된다. geogrid.exe 모듈은 전구 데이터 GEOG_DATA를 입력 받아 예보를 수행할 도메인을 정의한 geo_em_d0x 파일을 출력한다. ungrib.exe 모듈은 해당 지역의 기상 데이터 Met Data에서 WRF 모델 구동에 사용되는 변수를 추출하고 이를 이진 파일로 표현한다. metgrid.exe 모듈은 geogrid.exe 모듈이 정의한 도메인에 ungrib.exe 모듈이 추출한 변수를 보간 한다. 이러한 전처리 과정이 완료되면 WRF 본체의 모듈 중 real.exe가 모델 구동에 사용될 초기장 wrfinput 파일과 경계장 wrfbdy 파일을 생성한다. 초기장과 경계장을 입력으로 wrf.exe 모듈이 적분으로 이루어진 실제 예보 연산을 수행한다.

본 실험에서는 AWIP(All Weather Insulated Panels)에서 제공하는 미국 동부 해안의 겨울 폭풍 데이터 재분석 자료를 사용하였다. 적분기간은 2000년 1월 24일 12시부터 1월 25일 12시까지 총 24시간이며, 측정 도메인 크기는 2190×1800km^2 이다. 실험환경은 Intel(R)Core(TM)i7 CPU950@3.07GHz, 8core, RAM26GB 이며, WRF version은 3.6.1을 사용하였다.

표 1. 해상도 변수
Table 1. Resolution variable

Contents	Description
E_WE	동서 격자의 수
E_SN	남북 격자의 수
DX	동서 격자의 크기(m)
DY	남북 격자의 크기(m)

위 AWIP 재분석 자료 기준으로 실제 연산이 이루어지는 WRF 모형 본체의 수행 시간이 WRF 모델 총 수행 시간의 약 94퍼센트를 차지한다. 따라서 본 연구의 데이터 집약적인 응

용 실험을 위해 WRF모델 본체에 사용되는 도메인의 해상도를 변경하였다. 표 1은 도메인의 해상도를 조절하기 위한 변수를 나타낸다.

표 2. 연구에 사용된 실험 디자인
Table 2. Experiment design used in this study

Experiment	E_WE, E_SN	DX, DY (m)
30km	74, 61	30000, 30000
20km	111, 92	20000, 20000
10km	222, 183	10000, 10000
5km	444, 366	5000, 5000
3km	740, 610	3000, 3000

도메인의 동서 길이는 (E_WE-1)*DX, 남북 길이는 (E_SN-1)*DY이다. 도메인의 기존 크기를 유지하면서 고해상도로 조절하기 위해 DX, DY의 값을 감소시키고, E_WE와 E_SN 변수 값을 증가시켜 실험하였다. 실험은 총 5가지로 진행하였으며 각 실험의 변수 값은 표 2와 같다.

WRF 모형 본체의 각 모듈의 수행 시간 및 메모리 점유율 사용량은 glances 툴을 사용하여 측정하였다. 그림 4는 real.exe와 wrf.exe의 수행 시간 및 최대 메모리 사용량을 보여준다. 격자 크기에 따른 각 모듈의 수행 시간은 격자 크기가 작아져 해상도가 높아질수록 증가한다. 격자 크기 30km, 20km, 10km의 경우 real.exe 모듈에서 최대 5.7초의 수행 시간이 소요되고 메모리 사용량이 감지되지 않을 정도로 미미하여 수행 시간에 영향을 미치지 않는다. 반면 5km의 경우에 92.3%의 메모리 사용량을 보이고 수행 시간은 142초로 급격히 증가한다. wrf.exe 모듈의 경우에 격자 크기 30km, 20km, 10km의 각각 최대 메모리 사용량은 4.6%, 8,8%, 30%이며, 5km의 경우 최대 92.7%이다.

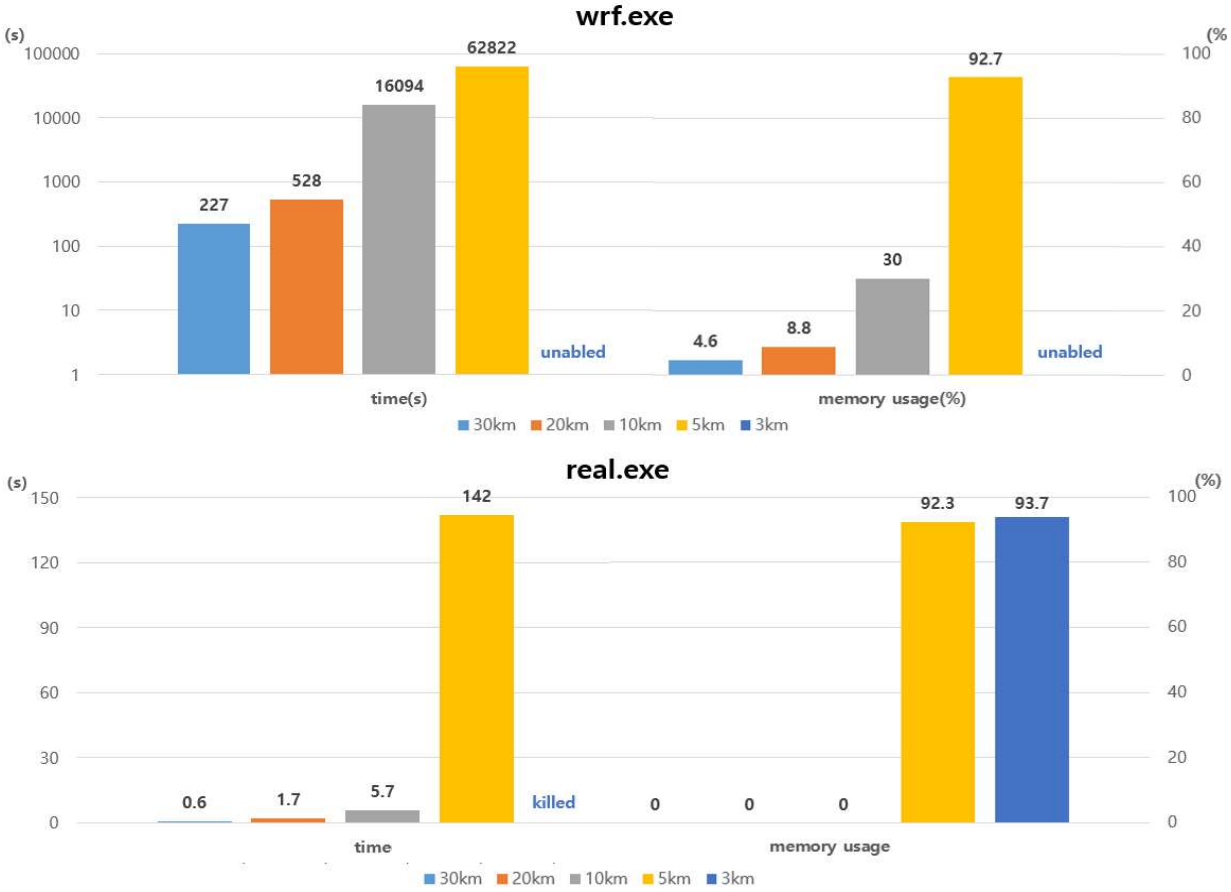


그림 4 . real.exe와 wrf.exe의 시간 및 최대 메모리 사용량
Fig. 4. Time and maximum memory usage of real.exe and wrf.exe

수행 시간 또한 1074분 2초(62822초)로 real.exe 모듈의 경우와 같이 급격히 증가함을 확인하였다. 3km의 경우에는 real.exe 모듈 단계에서 메모리를 최대 93.7%까지 사용하면서 수행 도중 비정상적으로 종료되었다.

본 실험 결과에 따르면 WRF 모형 본체에 사용되는 입력 데이터의 부하가 커질수록 수행 시간과 메모리 사용량이 급격히 증가함을 알 수 있다. 따라서 데이터 집약적 예보 수행을 위해서는 메모리 자원을 적시적으로 활용하는 인프라 구축 필요하다.

Ⅴ. HPC 응용 실행을 위한 소프트웨어 정의 컴퓨트(SDC) 구조

본 논문에서는 앞 장의 응용 시나리오에 적합한 소프트웨어 정의 기술을 활용하여 응용의 특성을 고려한 실시간 HPC응용 실행 구조를 정의하였다.

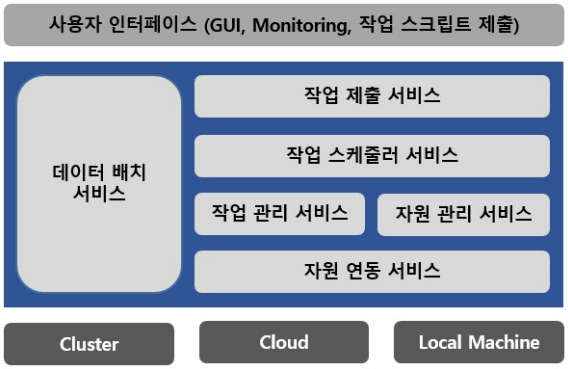


그림 5 . 소프트웨어 정의 컴퓨트(SDC) 구조
Fig. 5. Software defined compute (SDC) structure

그림 5는 본 논문에서 제안하는 기능 구조도이다. 이 구조는 차세대 고성능 인프라에서 발생하는 데이터 전송 및 계산 배치의 문제를 효과적으로 해결할 수 있도록 데이터 집약적인 응용을 위한 데이터 배치 및 작업 스케줄러를 포함한다. 소프트웨어 정의 컴퓨트(SDC) 구조는 다음 단계와 같은 작업으로 진행된다. 사용자는 사용자가 지정한 작업 제출 내용이 명세 되어 있는 파일을 사용자 인터페이스를 통해 제출한다. 작업 제출 모듈에서 제출된 파일의 구문을 분석하고 응용 프로그램의 유형, 실행 파일 경로, 매개 변수, 종속성 등과 같은 정보를 추출한다. 파일에 지정된 매개 변수 범위에 따라 작업을 생성한 다음 해당 작업의 정보와 함께 작업 관리 모듈로 전달한다. 작업 관리 모듈은 스케줄링 정책을 기반으로 작업들을 스케줄링하고 자원 관리 모듈로 제출한다. 자원 관리 모듈은 할당된 여러 유형의 자원에 따라 사전에 설정된 계획을 통해 가상머신을 실행한다.[9]

플랫폼의 작업 제출 및 작업 스케줄러 서비스는 응용의 특성 분석을 통해 찾아낸 각각의 특징 정보를 이용하여 세밀한(fine-grained) 자원 요구를 만족시키는 SDC 기술 기반의 작업 스케줄링 기법을 제공한다. 예를 들어 4장의 실험을 통해 기상 재난 예측 모델의 HPC 응용은 메모리 자원 집중 사용 패턴을 보여주었다. 이러한 응용의 메모리 자원 사용 패턴 분석을 기반으로 메모리 자원 재구성 기법을 수행할 경우, 필요 시점에 동적으로 메모리 할당과 반환이 가능하다[10]. 이를 바탕으로 응용의 "bursty" 자원 요구가 있는 실행 단계에서 메모리 유휴 자원 사용을 최대화하여 수행할 수 있도록 한다. 또한 메모리 자원의 조기 반환 시에는 대기 작업에게 유휴 상태의 메모리를 동적으로 조기에 할당할 수 있으므로 대기 작업의 수 및 시간을 줄이고 시스템 작업 자원 사용 효율을 극대화할 수 있다.

기상 재난과 같은 HPC 응용은 대규모의 입력 데이터, 중간 데이터와 출력 데이터를 생성한다. 데이터의 생성 시점 및 종류에 따라 사용되는 용도와 사용 빈도가 다르다. 따라서 데이터의 생성 시점과 종류를 분류하고 그 결과에 따라 관리, 사용한다. 이러한 데이터 배치 기법을 통해 응용에 사용되는 데이터 특성 이해를 기반으로 하여 지능적으로 데이터를 분배, 배치해야 한다. 데이터 집약적인 HPC 응용의 데이터와 작업 간의 관계 정보를 통해 데이터 이동을 최소화하는 데이터 배치 서비스를 제공하는 것이 한 예이다. 또한 데이터 배치 전략에서 성능 하락의 큰 요인인 네트워크 병목현상으로 인한 지연을 최소화 한다. SDN환경에서의 데이터 움직임을 최소화하기 위해 네트워크 전송 속도 및 대역폭을 고려한 서비스를 제공한다.

Ⅵ. 결 론

본 논문에서는 HPC 응용 실행을 위한 소프트웨어 정의 컴퓨트 구조 대해 정의하였다. 응

용의 특성과 데이터의 이동을 고려하면서 효과적인 스케줄링을 제공하여 자원의 효율성 및 활용성을 극대화 할 수 있다. 또한ScienceDMZ와 같은 국가적 차원의 고성능 데이터 전송체계에서 데이터배치 기법은 데이터 집약적인 HPC 응용 과학 연구를 위한 기술 발전으로 이어진다. 현재 그리고 향후 국가적으로 추진되고 있는 다양한 과학 응용 개발 사업에도 원천기술로써 적극적으로 활용이 가능하다.

References

[1] http://www.openflow.org/videos

[2] 문정훈, and 이민선. "데이터 집중형 (Data-Intensive Science) 과학을 위한 ScienceDMZ 기반 빅데이터 전송 연구." 한국정보과학회 학술발표논문집 : 1359-1361, 2016

[3] Dart, Eli, et al. "The science dmz: A network design pattern for data-intensive science." Scientific Programming 22.2: 173-185, 2014

[4] ETRI, "미래 네트워킹 기술 SDN," 2012 Electronics and Telecommunications Trends.

[5] Nam, Yoonsung, et al. "Workload-aware resource management for software-defined compute." Cluster Computing 19.3 : 1555-1570, 2016

[6] Seungsoo Lee, "Development and Application of 1D-2D Coupled Urban Inundation Model for Real-time forecasting based on Remote Sensing Technique", KNOM tutorial 2016

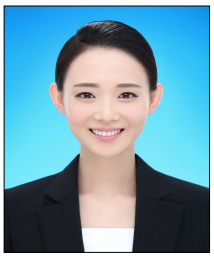
[7] WRF, http://www2.mmm.ucar.edu/wrf/users/

[8] http://www2.mmm.ucar.edu/wrf/users/

[9] 안주림, et al. "과학 응용 실험을 위한 하이브리드 자원 활용 소프트웨어정의 컴퓨팅 (SDC) 플랫폼 설계." 한국정보과학회 학술발표논문집 : 1499-1501, 2017

[10] Adufu, Theodora, Yoonhee Kim. "Dynamic Memory Allocation for Scientific Workflows in Containers." 정보과학회논문지 44.5 : 439-448, 2017

오 지 선 (Ji sun Oh)

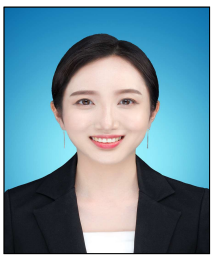


2017년 숙명여자대학교, 컴퓨터과학과 학사

2017년 ~ 현재 숙명여자대학교, 컴퓨터과학과 석사과정

<관심분야> 분산시스템, 클라우드 컴퓨팅

이 효 정 (Hyojeong Lee)



2014년 ~ 현재 숙명여자대학교, 컴퓨터과학과 학사과정

<관심분야> SDN, 빅데이터 처리 기술, 기계학습

김 윤 희 (Yoonhee Kim)

1991년 숙명여자대학교 학사

1996년 Syracuse University 전산학과 석사

2000년 Syracuse University 전산학과 박사

1991년 ~ 1994년 한국전자통신연구소(ETRI) 연구원

2000년 ~ 2001년 Rochester Institute of Technology 컴퓨터공학과 조교수

2001년 ~ 2004년 숙명여자대학교 컴퓨터과학과 조교수

2004년 ~ 2009년 숙명여자대학교 컴퓨터과학과 부교수

2009년 ~ 현재 숙명여자대학교 소프트웨어학부 교수

<관심분야> 클라우드 컴퓨팅, 그리드/클라우드 관리

KNOM Review '17-01 Vol.20 No.01
https://doi.org/10.22670/knom.2017.20.1.24

SDN기반 IoT 게이트웨이에서 DoS 공격 영향 평가

최진태◆, 뉘엔 신 응억*, 김경백°

Assessing the impact of DoS Attack on SDN based IoT Gateway

Jintae Choi◆, Sinh-Ngoc Nguyen*, Kyungbaek Kim°

요 약

최근 IoT 디바이스가 보편화 되면서 그 수가 빠르게 증가하고 있다. 또한, CCTV, 난방기기, 자동차, 냉장고 등 인간의 삶에 매우 중요한 기기 또한 IoT에 포함되고 있다. 이러한 상황에 2016년 9월 미라이 공격은 수십만대의 IoT 디바이스를 감염시키고, 이를 이용하여 대규모 DDoS 공격을 감행하였다. 이는 저사양의 IoT 기기에서 발생하는 대량의 공격 트래픽에 대한 대책 마련이 매우 중요함을 보여준다. 또한, IoT 환경에서 발생하는 트래픽은 IoT 기기들 간의 통신에 악영향을 끼쳐, IoT 시스템의 정상적인 동작을 방해할 수 있다. 최근 연구에서 DoS 공격이 IoT 디바이스들의 통신에 미치는 영향에 대해 분석하여 무선 IoT 기기가 특히 DoS 공격에 영향을 많이 받는 것이 확인되었다. 본 논문에서는 DoS공격이 유무선 IoT 통신 환경에 미치는 영향을 더욱 자세히 분석하기 위해, 다양한 구성요소를 가지는 SDN기반 IoT 게이트웨이 환경에서 DoS 공격의 영향을 측정하고 분석하였다. 또한, DoS 공격 발생시 IoT 네트워크 트래픽의 QoS를 유지하기 위해 SDN기반 IoT 게이트웨이에 Queue 관리 기능을 적용하고, 이를 통해 DoS 공격상황에서 IoT 기기 간의 통신 성능 개선이 가능함을 확인하였다.

Key Words : IoT, DoS, SDN (Software Defined Networking), Gateway, AP, Queue, Management

ABSTRACT

Recently, the number of IoT devices has been increasing rapidly. Also, devices such as CCTV, heaters, cars and refrigerators are also included in the IoT. In September 2016, the Mirai botnet infected with hundreds of thousands of IoT devices and launched a huge scale DDoS attack. This shows that preventing heavy attack traffic on IoT devices with limited resources is very important. In addition, traffic in an IoT environment may adversely affect the communication between IoT devices and interfere with normal operations of the IoT system. A recent study found that the impact of DoS attacks on the communication of the IoT devices is severe to the wireless IoT devices. In this paper, we measure and analyze the effect of DoS atack on SDN based IoT gateway with various scenarios to access the impact of DoS attacks in wired and wireless IoT Communication environment in detail. In addition, in order to maintain the QoS of the IoT network traffic during a DoS attack, Queue management technique is applied to SDN based IoT gateway and the viability of improving the performance of IoT communication under DoS attack by using adaptive Queue management.

※이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (NRF-2017R1A2B4012559). 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음 (IITP-2017-2016-0-00314).

◆ First Author : Chonnam National University Department of Information Security, jefroh1100@gmail.com
° Corresponding Author : Chonnam National University of Department of Electronics and Computer Engineering, kyungbaekkim@jnu.ac.kr

논문번호 : KNOM2017-01-05, Received May 20, 2017; Revised June 30, 2017; Accepted August 15, 2017

Ⅰ. 서 론

IoT(Internet of Things)는 주변 사물이 유선 또는 무선으로 네트워크에 연결하여 정보를 공유할 수 있는 기술이나 환경을 뜻한다. 최근 이러한 IoT 디바이스가 보편화 되면서 그 수가 빠르게 증가하고 있다. IT 분야의 리서치 기업 가트너는 2014년 64억 개의 IoT 디바이스가 2020년에는 135억 개에 달할 것으로 추측한다. 더불어 2014년을 기준으로 2015년에는 IoT 디바이스의 사용률이 30%나 증가하였다. [1,2,4] IoT 디바이스의 증가와 함께 IoT의 영역 또한 점차 늘어나 CCTV, 난방기기, 냉장고, 자동차등과 같이 인간의 삶에 밀접한 기기들도 IoT 디바이스의 영역에 포함 될 것으로 예상된다. [3,4]

이렇게 IoT 디바이스의 수는 날이 갈수록 증가하고, 인간의 삶에 더욱 중요한 기기가 되어가고 있지만 IoT 디바이스에 대한 보안은 여전히 취약한 상태이다. 이러한 상황에 2016년 9월, 수십만 대의 IoT 디바이스를 감염시켜 대규모 DDoS 공격을 일으킨 미라이 봇이 등장하였다. 또한, 미라이 봇의 소스코드가 공개되어 앞으로 더 많은 변종 IoT 봇이 등장할 것으로 예상된다. [5] 이와 같이, 다수의 IoT 디바이스를 감염시킬 수 있는 봇이 등장함으로서 이를 악용하여 IoT 네트워크 자체를 마비시킬 수 있는 공격 또한 고려해야 한다.

이에 따라, 최근 논문에서는 유, 무선으로 연결된 IoT 디바이스에 DoS 공격이 가해질 경우 IoT 디바이스들의 통신에 미치는 영향을 실험 하였다. 실험 결과, 무선으로 연결된 IoT 디바이스로 DoS 공격이 가해질 경우 모든 무선 IoT 디바이스들이 매우 큰 영향을 받는 것으로 나타났다. 이 디바이스들은 통신 속도가 느려지기도 하였으며, 통신 불능 상태가 되는 경우도 발생하였다. 이는 앞으로 실생활에 사용될 무선 IoT 디바이스에 대한 DoS 공격의 대비가 매우 중요하다는 것을 보여준다. 특히, IoT 기반의 홈서비스에서는 1개의 공유기에 모든 IoT 기기가 무선으로 연결되는 경우가 많기 때문에 앞으로 나타날 다양한 IoT 게이트웨이 환경에서의 실험은 더욱 더 중요하다. [4,6]

본 논문에서는 다양한 IoT 게이트웨이 환경에서 실험하기 위해 2개의 무선 네트워크 인터페이스를 가진 SDN 기반 IoT 게이트웨이를 구현한다. [6,12] 이와 같은 새로운 환경에서의 실험에 앞서, 먼저 1개의 무선 네트워크 인터페이스를 가지는 SDN기반 IoT 게이트웨이 환경에서 실험을 진행한다. 이를 위해, 유선 IoT 디바이스에서 무선 IoT 디바이스로 DoS 공격을 수행하는 동안 공격을 수행하거나 받지 않는 유선-유선, 무선-유선, 무선-무선 IoT 기기들의 통신 상태에 대해 측정한다. [4,6] 다음으로 2개의 무선 네트워크 인터페이스를 가진 SDN기반 IoT 게이트웨이 환경에서 실험을 진행하여 서로 다른 무선 네트워크 인터페이스가 DoS 공격에 어떠한 영향을 받는지 측정하고 분석한다. 또한, IoT 네트워크 환경에서 DoS 공격의 영향을 최소화하여 IoT 네트워크 통신의 QoS를 유지할 수 있도록 Queue 관리를 적용한 결과를 측정하고 분석한다. [12]

Ⅱ. 관련 연구

2.1 SDN

SDN(Software-Defined Networking)은 소프트웨어로 정의된 네트워크를 의미한다. 이는 기존의 라우터나 스위치와 같은 네트워크 장비의 제어 평면과 데이터 전달 평면을 분리하여 SDN 컨트롤러를 통해 데이터 전달 평면을 제어 및 관리 하는 개념이다. 또한 SDN은 네트워크 장비의 기능을 정의할 수 있는 API를 외부에 개방하여 다양한 네트워크 제어 기술과 라우팅 프로토콜을 개발하여 동작할 수 있도록 한다. [5]

SDN 구조는 크게 데이터 전달 평면 (Data Plane), 제어 평면 (Control Plane), 응용 평면 (Application Plane)으로 구성된다. 제어 평면에서는 SDN 컨트롤러를 이용하여 데이터 전달 평면의 네트워크 장비들을 제어 및 관리한다. 기존의 네트워크에서는 각 네트워크 장비마다 제어하고 관리해야 했지만, SDN은 SDN 컨트롤러에서 통합적으로 네트워크 장비를 관리하기 때문에 네트워크 관리 및 제어에 매우 효율적이며, 편리한 서비스를 제공한다. 이를 위해, 제어 평면과 데이터 전달 평면을 상호 연결 시켜주는 표준화된 프로토콜인 Open southbound API가 필요하다. 현재 가장 많이 사용되는 프로토콜은 OpenFlow이다. 이러한 표준화된 프로토콜을 통해 특정 장비에 국한되지 않고 네트워크망을 제어 및 관리할 수 있다. [5]

또한, SDN 컨트롤러의 네트워크 OS 기능을 자유롭게 개발할 수 있도록 제어 평면과 응용 평면

간에 Northbound API를 제공한다. 이를 통해 네트워크 관리자는 자신의 목표에 맞는 라우팅 프로토콜, 트래픽 엔지니어링, QoS 관리 등을 제공할 수 있다. [5]

데이터 전달 평면은 네트워크 하드웨어 장비로 구성된다. 이러한 장비들은 기존의 네트워크 장비들과 다르게 패킷제어 및 관리 기능 없이 패킷 포워딩 기능만을 가지고 있으며, SDN 컨트롤러에 의해 제어 및 관리 된다. [5]

2.2 DoS 공격

DoS (Denial of Service) 공격이란 서비스 거부 공격을 뜻한다. 이는 악의적으로 유명 사이트나 목표가 되는 시스템의 자원을 점유하여 목표 시스템의 서비스를 원하는 사람이 서비스 받지 못하도록 하는 공격 기법이다. 이러한 DoS 공격은 DDoS (Distributed Denial of Service)공격과 DRDoS (Distributed Reflection Denial of Service)공격으로 발전하였다.

DDoS 공격은 분산 서비스 공격으로, 여러 대의 PC를 좀비 PC로 감염시킨 후 좀비 PC를 사용하여 목표가 되는 시스템을 공격하는 공격 기법이다. 최근에는 좀비 PC대신 IP를 가지며 인터넷에 연결 할 수 있는 다수의 IoT 기기들을 사용하여 DoS 공격을 감행하였다. 이에 따라 IoT 디바이스 또한 DoS 공격에 매우 위험한 위치해 있으며, 이에 대한 피해 정도와 대응방법을 구상할 필요가 있다.

DR-DoS 공격이란 분산 반사 서비스 거부 공격으로, DDoS의 발전된 형태이다. 이 공격기법은 좀비 PC가 직접 목표물을 공격하지 않고 DNS 서버나 NTP서버를 반사 서버로 이용하여 목표물을 공격하는 공격기법이다. 이러한 반사 서버에 목표물의 IP로 스푸핑된 소스 IP를 사용하여 공격 요청 패킷을 전송하게 되면, 공격 요청 패킷을 받은 반사 서버는 스푸핑된 IP를 가진 목표물에 대량의 패킷을 전송함으로써 공격이 감행된다. 또한, 최근에는 SSDP공격과 같은 IoT 프로토콜을 이용한 DR-DOS 공격 또한 등장하고 있다. [8,9,10]

Ⅲ. 실험 환경 구축

최근 연구에서는 상용 게이트웨이에 존재하는 1개의 무선 네트워크 인터페이스를 사용하여 실험하였다. [4] 하지만, 본 논문에서는 다양한 IoT 게이트웨이 환경에서의 실험 및 DoS 공격의 영향을 줄이기 위한 Queue 관리 기능 적용을 위해 2개의 무선 네트워크 인터페이스를 가진 SDN 기반 유무선 IoT 게이트웨이를 구현하였다.

3.1 SDN 기반 IoT 게이트웨이

SDN은 다양한 유, 무선 네트워크 인터페이스와 프로토콜을 제공하며 사용자 정의대로 게이트웨이를 관리 할 수 있다. 이러한 이유로 SDN을 사용하여 유무선 IoT 게이트웨이를 구성한다. 이를 위해 H/W로 라즈베리파이3를 사용했으며, OS로는 라즈비안, SDN 스위치 기능을 사용하기 위해 openvswitch를 설치한다. Open vSwitch는 OVS 라고 불리며, OpenFlow 프로토콜을 지원하며 스위치기능을 가진 프로그램이다.

3.2 실험 환경 구성

SDN기반 IoT 게이트웨이 환경에서 DoS 공격이 발생했을 경우 IoT 디바이스들의 통신에 대한 영향을 측정하기 위해 그림 1과 같은 SDN 기반 스위치 네트워크 토폴로지를 구성하였다. 그림 1의 구성요소는 다음과 같다. SDN 컨트롤러인 OpenDayLight(ODL) Beryllium, 5개의 OVS-Switch, Vswitch로 구현한 1개의 SDN 기반 IoT 게이트웨이를 사용하였다. 무선 네트워크 인터페이스는 usb wifi dongle을 사용하였다. 표 1은 무선 네트워크 인터페이스의 세부 특성을 보여준다. [6,11,12]

무선 네트워크 인터페이스를 활용하는 SDN기반 IoT 네트워크의 구현을 위해 라즈베리파이3에 라즈비안 운영체제를 사용한다. 그리고 유선 네트워크와 무선 네트워크를 연결하기 위해 OVS를 설치하여 브릿지를 생성한다. 이를 위해 hostap tool을 사용하

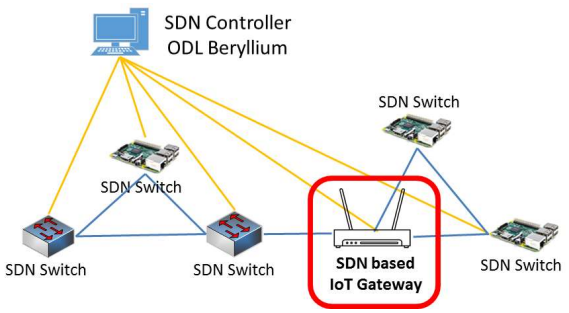


그림 1. SDN 기반 스위치 네트워크 토폴로지
Fig. 1. SDN-based switch network Topology

표 1. 무선 랜 usb wifi dongle 세부 특성
Table 1. Wireless LAN usb wifi dongle Details

카테고리	특징
CPU	리얼텍 RTL8188
인터페이스	USB 2.0
무선 주파수 대역폭	2.4 GHz
무선 지원 규격	IEEE 802.11n/b/g
무선 속도	N150
무선 수신 채널	싱글 밴드

여 IoT 게이트웨이를 설정하였다. 또한, 무선 IoT 디바이스의 연결을 위해 isc-dhcpserver를 사용하여 자동으로 ip를 제공하도록 만들었다.

추가적으로, DoS 공격 트래픽을 제한하기 위해, SDN기반 IoT 게이트웨이에 플로우별 Queue 관리 기능을 적용한다. Queue 기능을 적용하게 되면 각 개별 트래픽의 최소, 최대 전송률을 설정할 수 있다. 이후에 ovs-ofctl 명령어를 이용하여 지정된 입력 port와 출력 port로 구분되는 플로우에 Queue를 설정한다. 예를 들어 다음 그림 2와 같이 Queue1은 Host1에서 Host3로 가는 플로우에, Queue2는 Host2에서 Host3로 가는 플로우에 설정된다. Queue1은 0.1Kb/s로 트래픽을 제한하며, Queue2는 10Kb/s로 트래픽을 제한한다.

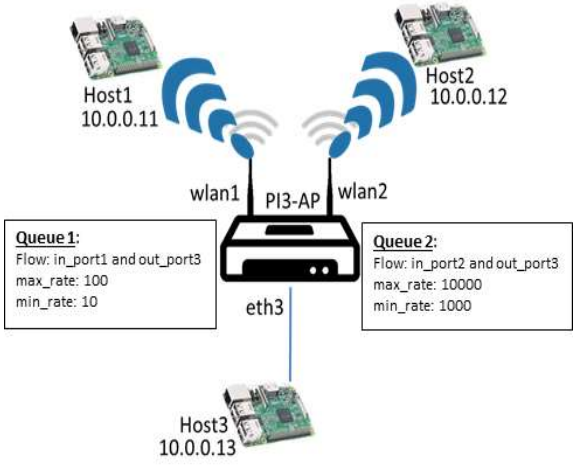


그림 2. SDN network QoS 구현
Fig. 2. SDN network QoS implementation

Ⅳ. 실험 및 분석

4.1 실험 측정 방법

그림 3, 5와 같은 실험환경에서 유선에서 유선 공격, 무선에서 무선 공격, 유선에서 무선 공격, 무선에서 유선 공격에 대한 각 IoT 디바이스들의 통신 상태를 비교한다. DoS 공격은 hping3 DoS 공격 툴을 사용한다. 공격 실행 시에는 표2와 같이 공격 속도 옵션을 각각 다르게 하여 IoT 디바이스의 통신 상태를 측정한다. IoT 디바이스의 통신 상태 측정은 1초당 1개의 Ping을 60번씩 보내어 주고받은 패킷 왕복시간인 Ping RTT(Round Trip Time)의 최솟값, 평균값, 최댓값과 손실률을 측정한다. 4.2 섹션에서는 한 개의 무선 네트워크 인터페이스를 가지는 SDN기반 IoT 게이트웨이 환경에서 유선에서 무선으로 DoS 공격을 수행하였을 경우 각 IoT 디바이스의 통신 상태를 측정한다. 4.3 섹션에서는 두 개의 무선 네트워크 인터페이스를 가지는 SDN기반 IoT 게이트웨이 환경에서 상황별 DoS 공격에 대한 각 IoT 디바이스 사이의 통신 상태를 측정한다. 4.4 섹션에서는 4.2와 4.3에서 분석한 결과를 토대로 Queue 관리 기능을 적용하여 DoS 공격에 대한 IoT 디바이스의 통신 개선이 가능함을 보인다.

4.2 IoT GW with one wireless network interface

첫 번째 실험에서는 1개의 무선 네트워크 인터페이스를 가진 SDN기반 IoT 게이트웨이 환경에서 DoS 공격에 대한 영향을 분석한다. 선행 연구에서는 IoT 게이트웨이 환경에서 DoS 공격이 이루어졌을 경우, 각 IoT 디바이스들의 통신 상태를 측정하였다. [4] 하지만 측정할 당시 DoS 공격을 받는 IoT 디바이스와의 통신 상태를 측정하였기 때문에 공격을 수행하거나 받지 않는 IoT 디바이스들의 통

표 2. hping3 공격 속도 옵션
Table 2. Attack speed option of hping3.

옵션	공격속도 (packet for second)
u1000	100
u100	1000
u50	5000
u10	10000

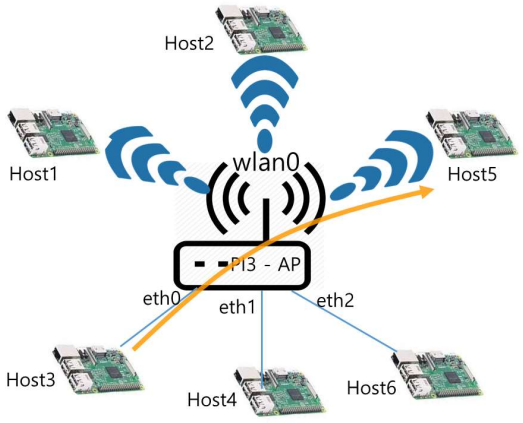


그림 3. 1개의 무선 네트워크 인터페이스 IoT 게이트웨이 구성 및 공격 경로
Fig. 3. Gateway Configuration with one wireless network interface and attack a path

신에 대해서는 정확히 알 수가 없었다. 이에 따라 본 논문의 첫 번째 실험에서는 선행 연구에서 DoS 공격의 영향을 많이 받았던 유선 IoT 디바이스에서 무선 IoT 디바이스로 DoS 공격이 발생할 경우, 공격을 수행하거나 공격을 받지 않는 다른 IoT 디바이스간의 통신 상태를 측정한다. 이를 위해 그림 3과 같은 1개의 무선 네트워크 인터페이스를 가진 SDN 기반 IoT 게이트웨이 환경을 구성한다. Host1, Host2, Host5는 무선 네트워크 인터페이스인 wlan0에 연결하였고, Host3, Host4, Host6은 유선으로 SDN 기반 IoT 게이트웨이에 연결하였다.

그림 4는 유선에서 무선으로 DoS 공격을 수행할 경우, 공격을 수행하거나 받지 않는 IoT 디바이스들 사이의 Ping RTT와 손실률을 측정한 그래프이다. 공격은 Host3에서 Host5로 수행하였다.

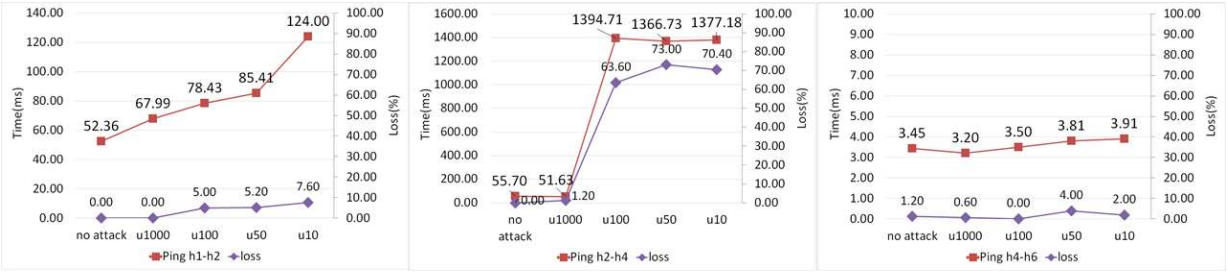
그림 4-(a)는 DoS 공격을 받지 않는 무선 IoT 디바이스들 사이의 Ping RTT를 나타낸 그래프이다. Ping h1-h2 보면 직접적으로 공격을 하거나 받지 않았음에도 불구하고 Ping RTT가 2배 이상 느려졌으며, 손실률 또한 최고 7.6%로 측정되었다.

그림 4-(b)는 DoS공격을 수행하거나 받지 않는 유선 IoT 디바이스와 무선 IoT 디바이스 사이의 통신을 나타내는 그래프이다. Ping h2-h4에서는 25배 가까이 통신 속도가 느려졌다. 하지만, 유선-유선 통신을 나타내는 그림4의 (c) Ping h4-h5 에서는 DoS 공격에 대한 영향이 전혀 없는 것으로 나타났다.

이를 통해, 유선에서 무선으로 DoS 공격이 발생하는 상황에서, 무선 IoT 디바이스와의 통신은 모두 DoS 공격에 큰 영향을 받는 것을 알 수 있다. 또한 공격을 수행하거나 받지 않는 유선 IoT 디바이스들 사이의 통신에는 전혀 영향을 받지 않음을 알 수 있다.

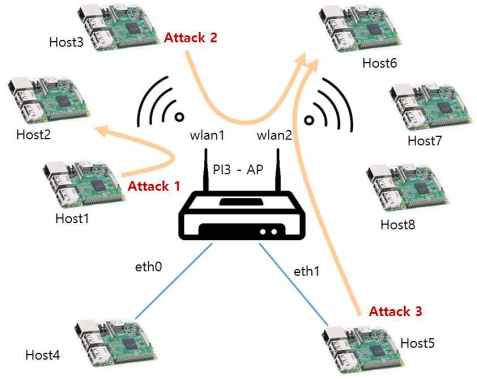
4.3 IoT GW with two wireless network interfaces

앞의 실험에서는 다수의 유선 네트워크 인터페이스와 1개의 무선 네트워크 인터페이스를 가진 IoT 게이트웨이 환경에서 실험하였다. 하지만, 앞으로 등장할 다양한 IoT 게이트웨이 환경에 대한 대비 또한 필요하다. 이를 위해, 두 번째 실험에서는 2개의 무선 네트워크 인터페이스를 가지는 SDN기반 IoT 게이트웨이 환경에서 DoS 공격이 발생했을 경우 각 IoT 디바이스의 통신 상태를 실험한다. 이 경우에도 공격을 수행하거나 수행하지 않는 IoT 디바이스 사이의 통신 상태를 측정한다. 이를 위해, 그림 5와 같이 2개의 무선 네트워크 인터페이스 wlan1, wlan2를 가진 SDN 기반 IoT 게이트웨이



(a) 무선-무선 통신 (b) 유선-무선 통신 (c) 유선-유선 통신

그림 4. 1개의 무선 네트워크 인터페이스 IoT 게이트웨이 환경에서 유선에서 무선(h3->h5) DoS 공격 상황의 실험 결과
Fig. 4. Experiment result on from wired to wireless(h3->h5) in an IoT gateway with one wireless network interface



Attack 1 : Attack within a single wireless network interface (Section 4.3-1)
Attack 2 : Attack between different wireless network interfaces (Section 4.3-2)
Attack 3 : Attack between wireless interface and wired interfaces (Section 4.3-3)

그림 5. 2개의 무선 네트워크 인터페이스 IoT 게이트웨이 구성도 및 각 공격 경로
Fig. 5. Gateway Configuration with two wireless network interfaces and attack paths

환경을 구성한다. Host1, Host2, Host3은 무선 네트워크 인터페이스인 wlan1에 연결하였고, Host6, Host7, Host8은 또 다른 무선 네트워크 인터페이스인 wlan2에 연결하였다. Host4, Host5는 유선으로 SDN 기반 IoT 게이트웨이에 연결하였다.

1) 하나의 무선 인터페이스 내의 DoS 공격

그림 6은 wlan1에 연결된 무선 IoT 디바이스에서 같은 무선 네트워크 인터페이스에 연결된 다른 무선 IoT 디바이스로 DoS 공격이 발생했을 경우 각 디바이스의 Ping RTT와 손실률을 측정한 그래프이다. DoS 공격은 Host1에서 Host2로 수행하였다.

그림 6-(a)의 Ping h3-h6은 wlan1에 연결된 공격을 수행하거나 받지 않는 무선 IoT 디바이스에서 wlan2에 연결된 무선 IoT 디바이스와의 통신 상태를 측정한 결과이다. 이 통신에서는 공격이 없는 상황에 비해 4배 가까이 통신 속도가 느려졌으며, 손실률 또한 0%에서 7.6%로 증가하는 것을 볼 수 있다.

그림 6-(b)의 Ping h7-h8은 wlan2에 연결된 무선 IoT 디바이스들 사이의 통신 상태를 나타낸다. 이 실험 결과에 따르면, wlan2에 연결된 무선 IoT 디바이스들 사이의 통신에는 DoS 공격의 영향이 없음을 확인하였다.

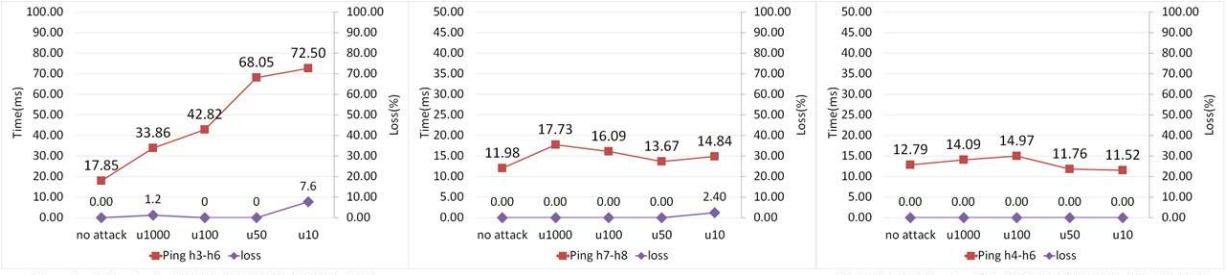
그림 6-(c)의 Ping h4-h6는 wlan2에 연결된 무선 IoT 디바이스와 유선으로 연결된 IoT 디바이스 사이의 통신 상태를 보여준다. 이 통신에서도 영향이 없음을 볼 수 있다.

이를 통해, 두개의 서로 다른 무선 네트워크 인터페이스 사이에 직접적인 DoS 공격이 발생하지 않을 경우, DoS 공격을 받지 않는 무선 인터페이스에 연결된 무선 IoT 디바이스들 사이의 통신은 DoS 공격에 영향을 받지 않는 것을 알 수 있다.

2) 서로 다른 무선 인터페이스 사이의 DoS 공격

그림 7은 wlan1에 연결된 무선 IoT 디바이스에서 wlan2에 연결된 다른 무선 IoT 디바이스로 DoS 공격이 발생했을 경우, 각 IoT 디바이스들 사이의 Ping RTT와 손실률을 측정한 그래프이다. DoS 공격은 Host3에서 Host6로 수행하였다.

그림 7-(a)의 Ping h1-h2는 같은 무선 네트워크 인터페이스 wlan1에 연결된 무선 IoT 디바이스들 사이의 통신 상태를 나타낸 그래프이다. 이 그래프에서 Ping h1-h2의 Ping RTT는 19.90ms에서 74.32ms로 4배 가까이 느려졌으며, 손실률은 0%에서 7.6%로 증가하였다. 이 두 디바이스는 공격을 직접적으로 수행하거나 받지 않는 상황이지만, DoS 공격이 이루어지는 무선 네트워크 인터페이스에 함



(a) wlan1와 wlan2 기기들 사이의 무선-무선 통신 (b) wlan2에 연결된 무선 기기들 사이의 통신 (c) 유선 기기와 wlan2에 연결된 무선 기기 사이의 통신

그림 6. 하나의 무선 네트워크 인터페이스 내의 DoS 공격 실험 결과.
Fig. 6. Results of DoS attacks Experiment within a single wireless network interface.

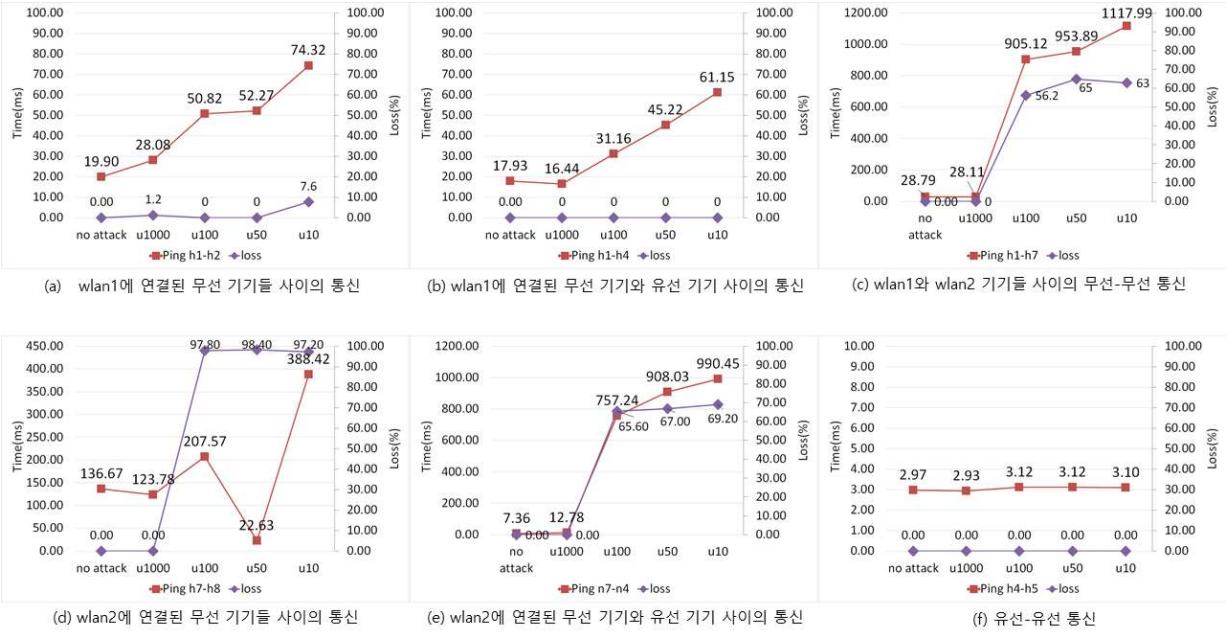


그림 7. 서로 다른 무선 네트워크 인터페이스 사이의 DoS 공격 실험 결과.
Fig. 7. Results of DoS attacks Experiment between different wireless network interfaces.

께 연결되어 있어 많은 영향을 받는 것을 알 수 있다.

그림 7-(b)의 Ping h1-h4는 무선 네트워크 인터페이스 wlan1에 연결된 무선 IoT 디바이스와 유선으로 연결된 IoT 디바이스와의 통신 상태를 나타낸다. 이 경우에는 손실률은 발생하지 않았지만, Ping RTT는 17.93ms에서 61.15ms로 4배 가까이 증가하였다.

그림 7-(c)의 Ping h1-h7은 wlan1에 연결되어 있는 무선 IoT 디바이스와 wlan2에 연결되어 있는 무선 IoT 디바이스와의 통신 상태와 손실률을 나타내는 그래프이다. 이 그래프에서 Ping RTT는 평상시 28.79ms에서 최고 1117.99ms로 40배 가까이 느려졌으며, 손실률 또한 0%에서 최고 65%까지 증가하였다.

그림 7-(d)의 Ping h7-h8은 무선 네트워크 인터페이스 wlan2에 연결되어 있으며, DoS 공격을 받지 않는 무선 IoT 디바이스들 사이의 Ping RTT와 손실률을 나타낸 그래프이다. 이 그래프에서 Ping RTT는 136.67ms에서 최고 388.42로 2배 증가하였으며, DoS 공격속도 u1000부터는 손실률이 97%이상을 기록하였다.

그림 7-(e)의 Ping h7-h4는 유선 IoT 디바이스와 wlan2에 연결되어 있는 무선 IoT 디바이스와의 통신 상태와 손실률을 나타낸 그래프이다. 이 그래프에서 Ping RTT는 평상시 7.36ms에서 최고 990.45ms로 135배 가까이 느려졌으며, 손실률은 0%에서 최고 69.2%로 증가하였다.

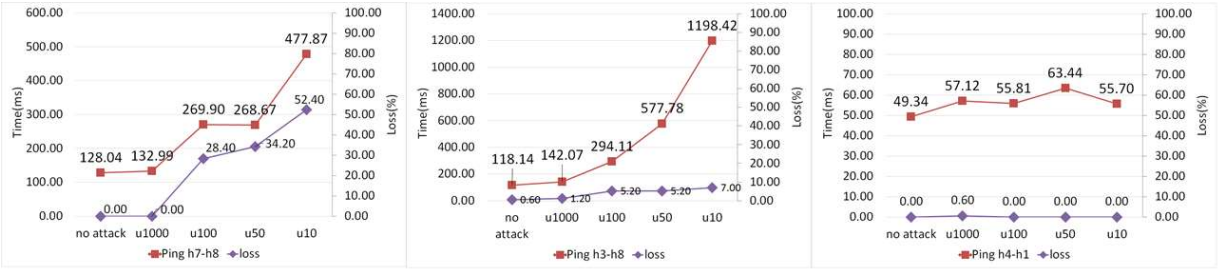
그림 7-(f)의 Ping h4-h5는 유선 IoT 디바이스 사이의 통신 상태와 손실률을 나타낸 그래프이다. 이 그래프에 따르면 무선에서 무선으로 DoS 공격을 하는 경우, 유선 IoT 디바이스사이의 통신은 영향받지 않음을 볼 수 있다.

이를 통해, 서로 다른 무선 네트워크 인터페이스 사이에서 DoS 공격이 일어날 경우, 공격이 발생하는 인터페이스의 무선 IoT 디바이스들의 통신은 DoS 공격에 많은 영향을 받으며, 공격받는 인터페이스에 연결된 무선 IoT 디바이스들은 거의 통신 불가에 가까운 결과를 볼 수 있다.

3) 유선에서 무선 인터페이스로의 DoS 공격

그림 8은 유선 IoT 디바이스에서 wlan2의 무선 IoT 디바이스로 DoS 공격이 발생 했을 경우, 각 IoT 디바이스들 사이의 Ping RTT와 손실률을 측정한 그래프이다. DoS 공격은 Host5에서 Host6로 수행하였다.

그림 8-(a)의 Ping h7-h8 과 손실률은 DoS 공격이 발생했을 때의 h7과 h8 사이의 통신 상태를 나타낸다. 이 두 무선 IoT 디바이스는 직접적으로 공격을 받고 있지는 않지만, 공격받고 있는 무선 네트



(a) wlan2에 연결된 무선 기기들 사이의 통신 (b) wlan1와 wlan2 기기들 사이의 무선-무선 통신 (c) wlan1에 연결된 무선 기기와 유선 기기 사이의 통신

그림 8. 유선 네트워크 인터페이스와 무선 네트워크 인터페이스 사이의 DoS 공격 실험 결과.
Fig. 8. Results of DoS attacks Experiment between wired interface and wireless interface

워크 인터페이스에 연결되어 있어 DoS 공격에 매우 큰 영향을 받고 있다. Ping RTT는 128.04ms에서 477.87ms로 4배 가까이 증가하였고, 손실률은 최고 52.4%에 달했다.

그림 8-(b)의 Ping h3-h8은 공격받지 않는 wlan1에 연결된 무선 IoT 디바이스와 공격받고 있는 wlan2에 연결된 무선 IoT 디바이스와의 통신 상태를 나타낸다. 이 경우의 Ping RTT는 118ms에서 1198ms로 10배 가까이 통신 속도가 느려졌으며, 손실률은 최고 7%에 달했다.

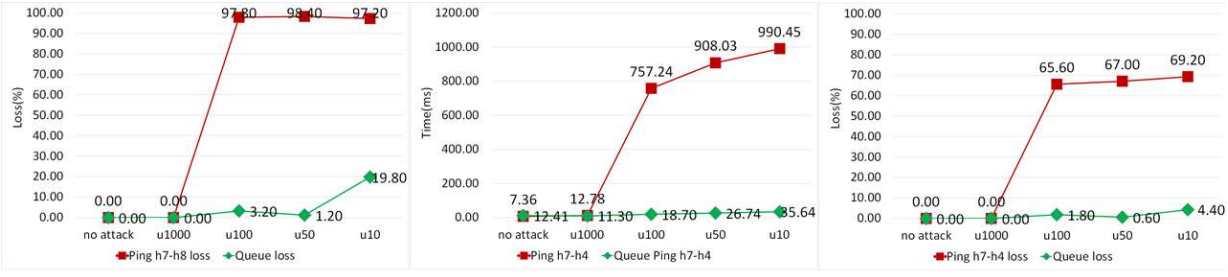
그림 8-(c)의 Ping h4-h1은 유선 IoT 디바이스와 공격받지 않는 무선 네트워크 인터페이스인 wlan1에 연결된 무선 IoT 디바이스와의 통신 상태를 나타낸다. 이 경우에는 DoS 공격에 대한 영향이 전혀 없을 알 수 있다.

이를 통해, 유선 인터페이스와 무선 인터페이스 모두 DoS 공격을 직접 받지 않는다면, DoS 공격에 대한 영향이 없음을 알 수 있다.

4.4 QoS 기능 적용

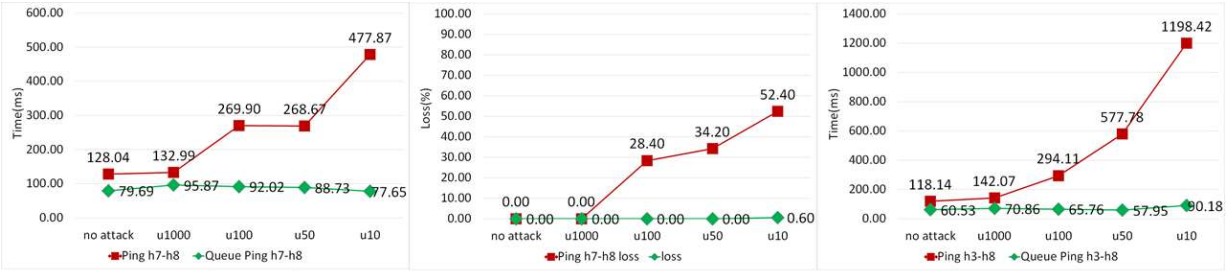
이번 절에서는 DoS 공격의 영향을 많이 받은 공격 상황을 선택하여 Queue 관리 기능을 적용하고, 그에 따른 결과를 산출한다. Queue 관리 기능은 IoT 디바이스에서 생성된 공격 플로우에 적용하였다. Queue 기능을 적용할 시에는 각 공격 플로우마다 최대 bit 전송률을 0.1Kb/s로 설정하였고, 최소 bit 전송률을 0.01Kb/s로 설정한다. 예를 들어, 그림 7처럼 Host 3에서 Host6로 DoS 공격이 이루어지는 경우는 DoS 공격 플로우가 무선 네트워크 인터페이스인 wlan1과 wlan2를 거쳐 흘러가게 된다. 이러한 플로우에 위와 같은 Queue 관리 기능을 적용한다. 이러한 방법으로 플로우의 최대 bit/s 전송률을 설정하게 되면 네트워크 인터페이스에 대량의 DoS 공격 플로우가 입력되는 즉시 제한하기 때문에 다른 네트워크 인터페이스의 QoS를 보장할 수 있다.

그림 9는 그림 7의 공격 상황에서 Queue 관리를 시행한 결과를 보여준다. 그림 9-(a)를 보면, Queue 관리를 시행하기 전에 100%에 가까운 손실률이 최소 1.2%, 최고 19.80%로 현저하게 낮아지는 것을 볼 수 있다. 또한, 그림 9-(b)를 보면 990.45ms 였던 Ping RTT가 최고 35.64ms까지 속도가 빨라지는 것을 볼 수 있다. 그림 9-(c)를 보면 69.20%에 달했던 손실률이 최고 4.40%로 현저하게 줄어든 것을 볼 수 있다.



(a) 그림7-(d)의 손실률 비교 (b) 그림7-(e)의 Ping RTT 비교 (c) 그림7-(e)의 손실률 비교

그림 9. 서로 다른 무선 네트워크 인터페이스 사이의 DoS 공격 상황에 Queue 관리 기능 적용 결과.
Fig. 9. The result of the Queue Management Function applying for DoS attacks between different wireless network interfaces.



(a) 그림8-(a)의 Ping RTT 비교 (b) 그림8-(a)의 손실률 비교 (c) 그림8-(b)의 Ping RTT 비교

그림 10. 유선 네트워크 인터페이스와 무선 네트워크 인터페이스 사이의 DoS 공격 상황에 Queue 관리 기능 적용 결과.
Fig. 10. The result of the Queue Management Function applying for DoS attacks between a wired network interface and a wireless network interface.

그림 10-(a)를 보면, 477ms에 달했던 Ping RTT가 Queue 관리 기능 적용이후 최고 95.87ms로 매우 빨라졌다. 그림 10-(b)를 보면 52.4%에 달했던 손실률이 거의 0%에 가까워졌다. 그림 10-(c)를 보면 1198.42ms에 달했던 Ping RTT가 최대 80.18ms까지 매우 빨라진 것을 확인 할 수 있다.

이를 통해, IoT 게이트웨이 환경에서 DoS 공격이 발생하였을 때 IoT 게이트웨이의 Queue 관리를 통해 DoS 공격으로 인한 IoT 디바이스의 통신 영향을 줄일 수 있음을 증명하였다.

Ⅴ. 결 론

본 논문에서는 다양한 구성요소를 가지는 SDN기반 IoT 게이트웨이 환경에서 DoS 공격에 대한 영향을 알아보기 위하여 2개의 무선 네트워크 인터페이스를 가진 SDN기반 유무선 IoT 게이트웨이를 구현하였다. 이후 IoT 게이트웨이 환경에 DoS 공격이 발생할 경우 각 IoT 디바이스들의 통신 상태를 실험하였다. 실험 결과, 한 개의 무선 네트워크 인터페이스 내부에서 DoS 공격이 발생 했을 경우에는 다른 무선 네트워크 인터페이스에 연결된 IoT 디바이스들은 영향을 받지 않는 것으로 나타났다. 그러나, 한 무선 네트워크 인터페이스에서 다른 무선 네트워크 인터페이스로 공격이 가해질 경우, 두 인터페이스에 연결된 무선 IoT 디바이스들이 영향을 받는 것으로 나타났으며, 무선에서 무선 공격 상황에서는 유선-유선 통신은 영향을 받지 않는 것으로 나타났다. 이러한 결과를 종합하여 보면, 무선 IoT 디바이스가 DoS 공격을 받거나 공격을 수행한다면 같은 무선 네트워크 인터페이스에 연결된 무선 IoT 디바이스들은 통신에 영향을 받는 것으로 나타났다.

이러한 결과에 따라, DoS 공격 발생시 IoT 디바이스의 QoS를 유지하기 위해 Queue 관리 기능을 적용하였다. 적용 후의 결과와 이전 결과의 통신 속도와 손실률을 비교하여 DoS 공격 발생시 IoT 게이트웨이의 Queue 관리가 충분한 효과가 있음을 증명하였다. 하지만, 이러한 Queue 관리는 네트워크 인터페이스 단위로 수행되기 때문에, 다수의 무선 IoT 디바이스가 1개의 무선 네트워크 인터페이스에 연결되는 특성상 하나의 무선 네트워크 인터페이스에 연결된 무선 IoT 디바이스 각각의 QoS를 보장하기 힘들다. 때문에, 중요한 무선 IoT 디바이스의 QoS를 유지하기 위해서는 무선 IoT 디바이스를 다수의 무선 네트워크 인터페이스에 분산시켜 연결시킨 후, Queue 관리를 수행한다면 더 효과적일 수 있다. 이에 따라, Queue 관리를 동적으로 수행하는 기술을 연구하여 더욱 효과적인 Queue 관리 기능을 구현할 계획이다.

References

[1] 유성경, " 미라이 봇넷, IoT의 DDoS 공격 ①" (2017), 접속일시(8. 28. 2017), http://news.grayhash.com/category/malware/%EB%AF%B8%EB%9D%BC%EC%9D%B4%20%EB%B4%87%EB%84%B7.

[2] http://www.gartner.com/newsroom/id/3165317.

[3] 현대경제연구원. "사물인터넷(IoT) 관련 유망 산업 동향 및 시사점". 2016. 07. 11.

[4] Jintae Choi, Kyungbaek Kim, "Analysis of wireless and wired DDoS Attack under IoT environment", knom, 2017.

[5] 한국인터넷진흥원, "2016년 Mirai 악성코드 동

향” 2016. 12

[6] Yungee Lee, Wangkwang Lee, Giwon Shin, Kyungbaek Kim, “Assessing the Impact of DoS Attacks on IoT Gateway.” 11th KIPS International Conference on Ubiquitous Information Technologies and Applications (CUTE 2016), December 19-21, 2016, Bangkok, Thailand

[7] 백상헌, 장인선, 서동은, 이종화 (2015). 미래 네트워크의 새로운 패러다임 SDN/NFV에 대하여. 한국통신학회지(정보와통신), 32(7), 82-92.

[8] Ki-Taek Lee, Seung-soo Baek, Seung-joo Kim, "Study on the near-real time DNS query analyzing system for DNS amplification attacks." Journal of the Korea Institute of Information Security & Cryptology 25(2), 303-311 (9pages), 2015. 4.

[9] 한국인터넷진흥원, “NTP 서버 보안 가이드”, 2015. 1

[10] Ju-Hye Oh, Keun-Ho Lee. "Attack Scenarios and Countermeasures using CoAP in IoT Environment." Journal of the Korea Convergence Society 7.4 (2016): 33-38.

[11] Seung-Hoon Baek, “Open vSwitch와 Mininet을 이용한 가상 네트워크 생성과 OpenDaylight를 사용한 네트워크 제어 실험” 15. 2015, https://www.slideshare.net/rkawkxms/2015316opendaylight-45868170

[12] Sinh Ngoc Nguyen, Yungee Lee, Wangkwang Lee, Giwon Shin, kyungbaek kim, “Design of SDN based IoT Gateway for detecting and Preventing DoS attack”, 2016년도 한국스마트미디어학회(KISM) 추계학술대회 논문집, pp. 223-225, 호남대학교, 광주, October 28-29, 2016

최 진 태 (Jin-tae Choi)

2017년 2월 : 조선대학교 정보통신공학과 학사

2017년 3월~현재 전남대학교, 정보보안협동과정 석사과정

<관심분야> 네트워크 트래픽 분석, 정보보안

뉘엔 신 응억 (Sinh Ngoc Nguyen)

2009: VietNam National University Ho Chi Minh City - University of Information Technology (B.S. Degree)

2016: Chonnam National University, South Korea (M.S. Degree).

2013~2015: Software Engineer at Integrated Circuit Design Research and Education Center

2016~now : School of Electronics and Computer Engineering

김 경 백 (Kyungbaek Kim)

1999년 : 한국과학기술원 전기 및 전자공학과 학사 졸업

2001년 : 한국과학기술원 전기 및 전자공학과 석사 졸업

2007년 : 한국과학기술원 전기 및 전자공학과 박사 졸업

2007년~2011년 : University of California Irvine, 박사 후 연구원

2012년~2015년 : 전남대학교 전자컴퓨터공학부 조교수

2016년 ~ 현재 : 전남대학교 전자컴퓨터공학부 부교수

<관심분야> 분산시스템, 미들웨어, 피어투피어/오버레이 네트워크, 소셜 네트워크, SDN>

https://doi.org/10.22670/knom.2017.20.1.34

선형 SVM기반의 안드로이드 악성코드 탐지 성능 향상 방안

김 기 현◆, 최 미 정°

Towards Linear SVM-based Android Malware Detection Performance Improvement

Ki-Hyun Kim◆, Mi-Jung Choi°

요 약

최근 모바일 기기의 성능이 향상되면서 다양한 모바일 서비스들이 등장하고 있다. 서비스의 다양화는 모바일 사용자들에게 편리함을 주었으며, 스마트폰 사용자 수는 점점 증가하고 있다. 사용자 수의 증가로 인하여 스마트폰을 공격하여 사용자들의 민감한 정보를 탈취하는 악성코드들이 많이 발생하고 있다. 스마트폰 OS 중 안드로이드의 경우 악성코드를 포함한 어플리케이션 발견의 수가 가장 많으며, 탐지율이 가장 낮다. 이에 따라 본 논문에서는 안드로이드 환경에서 악성코드를 탐지하기 위해 기계학습 분류기 선형 SVM(Support Vector Machine)을 적용한 악성코드 탐지 시스템을 개발하고, 시스템을 통해 악성코드를 분석하고자 한다. 악성코드 탐지 시스템의 분석적 성능 향상을 보이기 위해 정규화를 사용한 결과와 사용하지 않은 결과를 보인다. 또한 본 논문에서 개발한 악성코드 탐지 시스템이 새로운 악성코드에 대한 탐지 가능성에 대한 가능성을 확인하기 위해 트레이닝 셋에 있는 악성코드에 대한 악성코드 탐지 결과와 트레이닝 셋에 없는 악성코드 탐지 결과를 비교한다.

Key Words : Malware Detection, Dynamic Analysis, SVM(Support Vector Machine)

ABSTRACT

Recently, as performance of mobile machine improves, various services are emerging. Various services gave convenience for mobile users and the number of smartphone users gradually increased. Malwares attack a user's smartphone information or a user's sensitive information. Among smartphone OS, Android has the highest number of applications including malicious code, and the lowest Malware detection rate. In this paper, we develop a malicious code detection system applying Linear SVM (Support Vector Machine) to detect malicious code in Android environment and analyze the malicious code through the system. In order to show the analytical performance improvement of malicious code detection system, we show the result of using normalization and the result of not using it. In addition, we compare malicious code detection result in training set with malicious code detection result not in training set in order to check possibility of detection possibility of new malicious code developed by malicious code detection system developed in this paper.

※ 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-00158, 국가 차원의 침해사고 대응을 위한 사이버 위협 인테리전스 분석(CTI) 및 정보 공유 기술 개발)
◆ First Author : 강원대학교 컴퓨터과학과, kkh1258@kangwon.ac.kr
° Corresponding Author : 강원대학교 컴퓨터과학과, mjchoi@kangwon.ac.kr

논문번호 : KICS2017-01-06 Received June 8, 2017; Revised July 6, 2017; Accepted August 20, 2017

Ⅰ. 서 론

최근 모바일 기기의 성능이 PC와 유사해지면서 다양한 서비스들이 계속적으로 나타나고 있다. 예를 들어 은행 결제 서비스, 커뮤니케이션, 쇼핑, 카메라 등등 많은 서비스들이 줄지어 등장하고 있다. 서비스의 다양화는 사용자들에게 생활의 편리함을 주었고, 컴퓨터나 다른 장치를 사용해야 했던 일들이 이제는 장소를 불문하고 모바일 기기로 모든 일들을 처리할 수 있는 시대가 되었다. 모바일 기기는 이제 사람들 사이에서 없어서는 안 될 물건이 되었지만, 이에 반대로 모바일 기기를 악의적인 목적으로 사용자의 민감한 정보를 탈취하거나 다른 악성코드를 설치하여 다른 공격을 시도하는 악성코드들이 증가하고 있다. 안철수 연구소의 ASEC 보고서[1]에 따르면, 스마트폰 악성코드는 2011년 8,290개에서 2012년 약 26만개로 폭발적으로 증가했고, 2013년에 100만개를 돌파한 이후 지속적으로 증가하고 있다. 이후로 꾸준한 증가세를 보이고 있으며, 올해도 모바일 악성코드의 양적인 증가 추세가 이어질 것으로 보이는 가운데, 스마트폰 뱅킹 사용자 대상 공격이 심화되고, 스마트폰 결제 서비스를 타겟으로 하는 새로운 위협이 등장할 것이라고 발표했다. 그림 1은 안철수 연구소에서 발표한 스마트폰 악성코드 통계 자료[1]이다. 그림 1에 따르면 2012년에 비해 2014년에 악성코드 증가율이 444%를 나타내고 있다. 이에 따라 모바일 기기를 대상으로 하는 악성코드에 대한 보안이 점점 중요해 지고 있다.

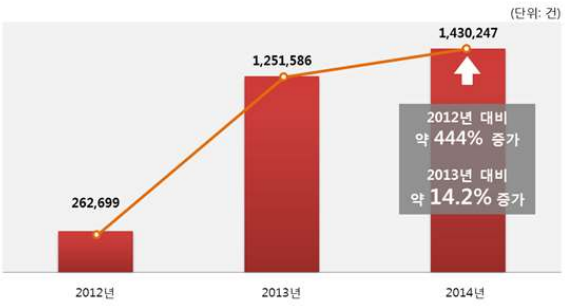


그림 1. 스마트폰 악성코드 신고 건수
Fig. 1. number of smartphone malware report

우리가 사용하는 유명한 모바일 기기 운영체제의 종류는 5가지로 iOS, 안드로이드, 윈도우, 블랙베리, 심비안이 있다. 5가지의 운영체제 중 가장 높은 점유율을 자랑하는 운영체제는 안드로이드 이다. 그림 2는 IDC에서 발표한 자료로 전 세계의 모바일 기기 운영체제의 점유율[2]을 나타내고 있다. 그림 2에 따르면 안드로이드의 점유율이 대략 80% 정도의 점유율을 보이고, 그 뒤로 iOS가 대략 15% 정도를 차지하고 있다. 안드로이드의 경우 많은 사용자들이 사용하지만, 보안적으로는 취약하다는 평가를 갖는다. 그 이유는 바로 개방성 때문이다. 개방적인 운영체제로써 사용자들이 악성코드의 배포가 굉장히 용이하며, 웹 또는 다른 경로를 통한 어플리케이션의 다운로드가 가능하다. 웹으로 들어가 나도 모르게 어플리케이션을 다운받아 보면 악성코드일 가능성이 있다. 이에 따라 본 실험 환경은 악성코드에 굉장히 쉽게 노출되어 있는 안드로이드를 실험 환경으로 사용한다.

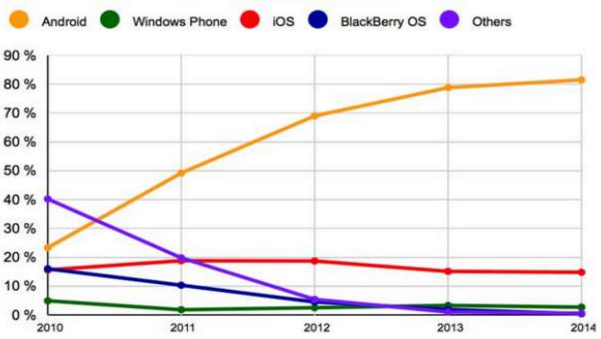


그림 2. 전 세계 스마트폰 운영체제 별 점유율
Fig. 2. The whole world smartphone OS market share

안드로이드 악성코드가 증가함에 따라 안드로이드를 환경으로 하는 악성코드 탐지 연구 기법들은 다음과 같다. 정적 분석 기법(Static analysis), 동적 분석 기법(Dynamic analysis), 시계열 기반 분석 기법(Time-series-based analysis), 하이브리드 기법(Hybrid analysis)으로 분류할 수 있다. 정적 분석 기법[3,4]은 시스템의 특정 문자열이나 소스코드를 분석하여 악성코드를 탐지하며, 최근에도 많이 사용하고 있고, 예전 PC 환경에서도 많이 사용는 기법이며, 동적 분석 기법[5,6]은 기계학습을 많이 사용하여 악성코드를 탐지하는 기법이다. 시계열 분석 기법[7]은 시간의 흐름에 따라 시그니쳐를 추출하여 시계열 모델을 사용하여 악성코드를 탐지하는 기법이며, 하이브리드 기법[8,9]은 두 가지 이상의 feature 또는 두가지 이상는 기법을 사용하여 악성코드를 탐지하는 기법이다. 본 논문에서는 동적 분석 기법을 사용하여 악성코드 탐지를 수행하며, 악성코드 탐지를 수행하기 위해 전처리 과정인 정규

화를 수행하여 실험에 사용한다. 정규화를 수행한 결과와 정규화를 수행하지 않은 결과를 비교하여 어느 정도의 성능 향상을 보였는지 보여준다. 또한 기계학습 알고리즘 SVM은 이미 악성코드 탐지에 대한 성능이 검증되었지만, 트레이닝 셋에 없는 경우의 새로운 악성코드 탐지에 대한 검증은 아직 의문이다. 따라서 테스트 셋에서 트레이닝 없는 경우를 비교하여 새로운 악성코드에 대한 대처가 가능한지 실험을 통하여 결과를 도출한다.

Ⅱ. 관련 연구

안드로이드 플랫폼 환경에서 악성코드를 탐지하는 기법들 중 동적 분석 기법을 적용한 논문에 대해 소개한다. 또한 기존 동적 분석 기법과 본 논문에서 제안하고자 하는 방법의 차이와 기존 동적 분석 기법의 한계점을 설명한다.

ASAF SHABTAI[5]의 논문은 모바일 기기에서 발생할 수 있는 자원정보 88개를 추출하여 그 자원정보 중 가장 높은 순위를 갖는 50개의 자원정보를 추출하여 기계학습 알고리즘 BN, Logistic, NB, Kmeans, DTJ48 6종류를 사용하여 악성코드를 탐지하는 실험을 수행하였다. 하지만 모바일 기기에서 88개의 feature를 추출하려면 모바일 기기의 루트권한을 얻어야 하기 때문에 루팅을 수행해야 한다는 한계점이 존재한다. 따라서 본 논문에서는 루팅을 수행하지 않고 추출할 수 있는 4가지 feature를 추출하여 악성코드 탐지를 수행한다.

Hyo-Sic Ham[6]의 논문은 악성코드 탐지를 수행하기 위해 루팅을 사용하지 않고 메모리, 네트워크, CPU, SMS, 전화, 배터리, 프로세스 분야에서 32개의 자원정보를 추출하였다. 또한 자원정보 중 메모리 분야를 Native와 Dalvik으로 나누어 feature를 추출하였으며, 이와 같이 추출한 32개의 자원정보를 RandomForest, SVM을 사용하여 SVM의 우수성을 증명하였다. 하지만 상기 논문에서 사용된 자원정보는 안드로이드 2.3.3버전에서 사용이 가능하며, 최근 안드로이드 버전은 5.0 이상으로 넘어가고 있다. 안드로이드 버전 4.0이상으로 넘어가면서 상기 논문에서 추출한 32개의 자원정보를 추출하기 위해서는 루팅이 필요하다는 한계점이 존재한다. 또한 동적 분석 기법을 사용하는 이유로는 새로운 악성코드에 대한 대처를 위하여 사용한다고 설명하고 있지만 그에 대한 증명은 하지 않았다는 한계점이 존재한다. 따라서 본 논문에서는 동적 분석 기법이 새로운 악성코드에 대한 탐지가 가능한지 가능하지 않은지 증명하기 위해 기계학습 알고리즘 SVM을 이용하여 실험을 수행하며, 트레이닝 셋에 악성코드 정보가 없는 경우 악성코드 탐지 성능이 어느 정도 정확도를 갖는지 확인한다.

Ⅲ. 악성코드 탐지 방법

본 장에서는 안드로이드 환경에서 동적 분석 기법을 활용한 악성코드 탐지 과정에 대해 자세히 설명한다.

1) SVM을 이용한 악성코드 탐지 시스템의 구조

본 논문에서는 안드로이드 환경에서 악성코드를 탐지하기 위해 기계학습 분류기 중 하나인 선형 SVM을 이용하여 악성코드 탐지 시스템을 설계하였다. 그림 3은 본 논문에서 제안한 악성코드 탐지 시스템의 구조를 나타내고 있다.

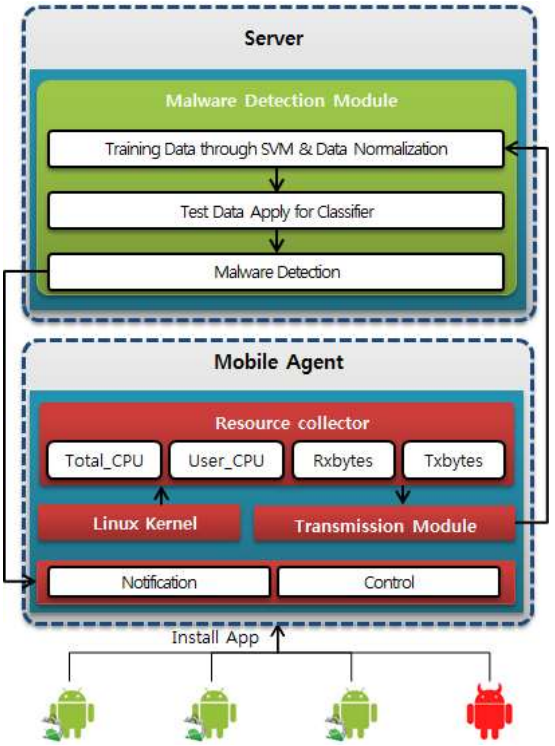


그림 3. 악성코드 탐지 시스템 구조
Fig. 3. Malware detection system architecture

악성코드 탐지 시스템은 총 4가지의 모듈 Resource Collector Module, Transmission Module, Malware Detection Module, Control Module로 나누어 설명할 수 있다.

첫 번째로 Resource Collector Module은 리눅스 커널 단에서 네트워크와 CPU의 자원정보 4가지를 추출한다. 표 1은 리눅스 커널 단에서 추출한 자원 정보 4가지를 나타낸다. 리눅스에서 자원정보를 추출하기 위해서 CPU의 경우 리눅스 명령어 'top' 명령어를 사용하여 Total_CPU, User_CPU를 추출하고, 네트워크의 경우 'proc'폴더에 있는 Rxbyte와 Txbyte의 계산 파일을 사용하여 추출한다. Resouce Collector Module은 10초 단위로 데이터를 추출하며, 10초 이하로 데이터를 추출할 경우 클라이언트에서 Alarm을 이용하여 시스템을 구성하였기 때문에 Alarm을 발생하는 리소스의 소모와 Receiver 에서 Alarm을 받아주기 위한 리소스의 소모가 크게 일어나 시스템의 오버헤드가 커 악성코드 탐지 시스템이 동작하지 않는 현상이 발생한다.

표 1. 악성코드 탐지를 위한 자원 정보
Table 1. Feture information for malware detection

카테고리	자원 정보	역할
네트워크	Rxbytes	프로세스로부터 다운로드 받은 네트워크 수
	Txbytes	프로세스로부터 업로드 한 네트워크 수
CPU	Total_CPU	프로세스로부터 사용한 CPU 사용률
	User_CPU	프로세스로부터 사용한 User CPU 사용률

두 번째로 Transmission Module은 4가지의 자원정보를 추출하여 벡터화 시킨 후 자원정보를 서버로 전송한다.

세 번째로 Malware Detection Module은 클라이언트로부터 전송 받은 데이터를 정규화를 수행한다. 정규화를 수행하는 이유는 SVM 알고리즘에 적용하였을 때 하나의 값으로 수렴할 때 계산 범위를 줄이기 위함이다. 계산의 범위를 줄여 데이터 셋의 분석 속도를 빠르게 해주고, 자실의 값이 1과 -1 사이에 놓이게 되므로 더 높은 성능을 보일 수 있다. 표 1과 같은 다른 대상의 데이터를 같은 기준의 단위로 나타낼 수 있기 때문에 데이터들을 같은 기준으로 볼 수 있게 되며, 그 결과 분석의 결과를 높일 수 있다. 정규화는 해당 값에 데이터 전체에 대한 평균을 빼주고 데이터 전체에 대한 분산으로 나누어 계산한다. 이는 수식 (1)에 자세히 나타낸다. 정규화 과정이 끝난 후 미리 만들어 놓은 트레이닝 셋에 10초 마다 전송되는 테스트 셋을 비교한다. 만약 악성코드가 발견되었다고 판단되면 클라이언트로 악성코드가 발견되었다는 신호를 보낸다.

$$Z = \frac{X - m}{\nu} \tag{1}$$

마지막으로 Control Module은 Malware Detection Module에서 전송된 신호를 바탕으로 악성코드가 탐지되었음을 사용자에게 알린다. 서버로부터 전송된 신호가 3번 연속으로 전달되면 악성코드가 탐지된 것으로 판단하고 사용자에게 Notification을 전송 후 악성코드에 대한 정보를 추출하여 해당 어플리케이션을 삭제한다.

2) 악성코드 탐지 시스템에 사용한 데이터 셋

본 장에서는 악성코드 탐지 시스템에서 사용한 정상 어플리케이션과 비정상 어플리케이션에 대해 설명하고, 선형 SVM에 사용하기 위해 사용된 데이터 셋에 대해 설명한다.

표 2. 사용한 어플리케이션의 종류
Table 2. Using application classify

Data type	Application name
Normal application	Facebook
	instargram
	youTube
	Naver
	kakaoTalk
Abnormal application	Anserver
	Angry Birds Rio Unlocker
	GoldDream
	DroidKungFu
	ZScaler
	FakeApp.AL
	Smspy

악성코드 탐지 시스템에서 사용한 정상어플리케이션은 총 5개를 사용하였고, 비정상 어플리케이션

은 총 7개를 사용하였다. 표 2는 시스템에서 사용한 어플리케이션 종류이다.

본 논문에서 사용된 어플리케이션 중 정상어플리케이션들은 평소 사람들이 가장 많이 사용하는 어플리케이션을 대표로 뽑아서 사용하였으며, 비정상 어플리케이션은 최근 국내 및 국외에서 활동하던 비정상 어플리케이션 샘플이다. 표 2에 나타난 어플리케이션들 중 정상 어플리케이션을 모두 설치한 후 비정상 어플리케이션을 하나 씩 수행한 후 데이터 셋을 추출하였다.

모바일 기기에서 추출한 데이터는 트레이닝 셋과 테스트 셋으로 나누어진다. 트레이닝 셋의 경우 악성코드와 정상을 분리하기 위해 사용되며, 정상 데이터의 수는 총 4000개이고, 비정상 데이터의 수는 비정상 어플리케이션 별로 1500개씩 총 10500개를 사용하였다. 테스트 셋의 경우 트레이닝 셋에 없는 데이터로써 시스템의 성능을 테스트하기 위해 사용되며, 정상 데이터의 수는 총 1000개이고, 비정상 데이터의 수는 비정상 어플리케이션 별로 300개씩 총 2100개를 사용하였다.

본 논문에서 선형 SVM의 악성코드 탐지 실험에서 새로운 악성코드에 대한 선형 SVM의 탐지 성능을 가늠하기 위해 데이터 셋을 트레이닝 셋7개를 만들었다. 각 트레이닝 셋에서는 정상 어플리케이션은 그대로 두고 비정상 어플리케이션의 데이터 셋 중에서 다른 종류의 비정상 데이터 셋을 하나씩 뺀 후 실험에 사용하였다. 실험에서 사용된 선형 SVM 툴은 SSVM[10]을 사용하였으며, 사용된 cost 값은 10부터 10000까지 반복 수행을 통해 Accuracy 값이 가장 높은 값인 10000으로 선정하였으며, Iteration 값은 10부터 200까지 반복 수행을 통해 Accuracy 값이 가장 높은 값인 200을 사용하였다.

Ⅳ. 실험 결과

본 장에서는 앞서 설명한 동적 분석 기법을 이용한 악성코드 탐지 방법을 수행한 성능에 대해 서술한다. 4.1절에서는 실험에 사용된 성능 지표에 대해 설명을 하고, 4.2절에서는 정규화를 사용한 실험 결과와 정규화를 사용하지 않은 실험 결과를 비교하고, 4.3절에서는 선형 SVM 알고리즘이 트레이닝 셋에 악성코드에 대한 정보가 없을 경우 다른 악성코드가 대신하여 탐지할 수 있는지 여부를 확인하고자 한다.

1) 성능 평가 지표

본 논문에서는 실험의 결과를 나타내기 위해 성능 평가 지표에 대해 설명한다. 실험에 사용된 지표는 TPR, FPR, Precision, Accuracy를 사용한다. 각 성능 평가 지표를 구하기 위해서는 시스템의 판단 결과에 따른 혼동행렬이 필요하다. 표 9는 성능 평가 지표를 계산하기 위한 혼동행렬이다.

혼동행렬에서 사용되는 TP, FP, TN, FN에 대해 설명한다. TP(True Positive)는 실제 정상 어플리케이션을 시스템이 정상 어플리케이션으로 판단한 수치이고, TN(True Negative)는 실제 비정상 어플리케이션을 시스템이 비정상 어플리케이션으로 판단한 수치이다. 본 논문에서는 실제 비정상 어플리케이션을 시스템이 다른 비정상 어플리케이션으로 판단한 수치 또한 TN으로 계산한다. FP(False Positive)는 실제 비정상 어플리케이션을 시스템은 정상 어플리케이션으로 판단한 수치이며, FN(False Negative)는 실제 정상어플리케이션을 비정상 어플리케이션으로 판단한 수치이다. 위의 통계 정보를 바탕으로 본 논문에서는 TPR(True Positive Rate), FPR(False Positive rate), Presicion, Accuracy를 구한다. 각 지표에 대한 수식은 다음과 같다.

표 3. 성능 평가 지표 계산을 위한 혼동행렬
Table 3. Confusion matrix for performance evaluation indicator

		Predicted data	
		Positive	Negative
Actual data	Positive	TP(True Positive)	FN(False Negative)
	Negative	FP(False Positive)	TN(True Negative)

$$\text{TPR} : \frac{TP}{TP+FN}\times 100 \quad (2)$$

$$\text{FPR} : \frac{FP}{FP+TN}\times 100 \quad (3)$$

$$\text{Precision} : \frac{TP}{TP+FP}\times 100 \quad (4)$$

$$\text{Accuracy} : \frac{TP+TN}{TP+TN+FP+FN}\times 100 \quad (5)$$

2) 정규화 수행 결과

선형 SVM을 이용한 악성코드 탐지 실험을 수행하기 위해 얻어진 데이터를 통해 정규화를 수행한 데이터와 정규화를 수행하지 않은 데이터의 탐지 결과를 비교한다. 표 11은 정규화를 수행하지 않은 탐지 결과를 나타내며, 표 12는 정규화를 수행한 탐지 결과를 나타낸다.

표 4. 정규화를 수행하지 않은 악성코드 별 탐지 결과
Table 4. None nomalization malware detection result

	TPR	FPR	Pre	Acc
Anserver	100%	0.33%	99.89%	99.92%
Angry Birds Rio Unlocker	100%	0.1%	99.98%	99.98%
Droid KungFu	99.1%	6%	98.21%	97.92%
Gold Dream	100%	0.1%	99.98%	99.98%
Smspy	100%	7%	97.92%	98.37%
ZScaler	99.89%	0.1%	99.98%	99.92%
FakeApp. AL	100%	0.1%	99.98%	99.98%

표 5. 정규화를 수행한 악성코드 별 탐지 결과
Table 5. nomalization malware detection result

	TPR	FPR	Pre	Acc
Anserver	100%	0.1%	99.89%	99.98%
Angry Birds Rio Unlocker	99.6%	0.1%	99.98%	99.69%
Droid KungFu	99.1%	7%	97.93%	98.37%
Gold Dream	100%	0.1%	99.98%	99.98%
Smspy	100%	4.3%	98.71%	98.99%
ZScaler	99.89%	0.1%	99.98%	99.92%
FakeApp. AL	100%	0.1%	99.98%	99.98%

정규화를 사용하지 않은 악성코드 별 탐지 결과를 보면 평균 TPR이 99% 이상의 결과가 나왔으며, FPR은 평균 1.9% 미만의 결과를 가져왔다. Precision은 평균 99.28%의 결과를 가져왔으며, Accuracy는 평균 99.43%의 결과를 가져왔다. 정규화를 사용한 악성코드 별 탐지 결과에서는 평균 TPR이 99% 이상의 결과를 가져왔고, FPR이 1.6% 미만의 결과를 가져왔다. Precision은 평균 99.49%의 결과를 가져왔으며, Accuracy는 평균 99.7%의 결과를 가져왔다. 정규분포 표준화를 사용한 결과와 사용하지 않은 결과를 비교하였을 때 Precision은 0.2%의 차이를 보이고, Accuracy는 0.3%의 차이로 정규화를 사용한 결과가 더 높게 나타났다. 악성코드 별 탐지 결과뿐만 아니라 수행 시간 측면에서도 높은 성능을 보인다. 정규화를 사용하지 않은 수행 시간은 4.2초 정도의 시간이 걸리고, 정규화를 사용한 수행 시간은 3.8초 정도로 0.4초 정도가 더 빠르다. 악성코드를 탐지하는데 있어 수행결과를 단 0.1초라도 빠르게 잡아야 악성코드가 수행되는 것을 막을 수 있다. 따라서 정규화를 사용하면 더 높은 성능을 가질 수 있다.

3) 트레이닝 셋에 없는 악성코드 별 탐지 결과

기존 연구에서 동적 분석 기법은 새로운 악성코드에 대한 대처가 가능하다는 표현이 많이 나왔었다. 따라서 동적 분석 기법이 새로운 악성코드에 대해 대처가 어느 정도 가능한지 확인한다. 표 6는 트레이닝 셋에서 해당 악성코드 데이터 셋을 제거했을 때의 악성코드 탐지 수행 결과를 나타낸 표이다.

Anserver 악성코드의 경우 TPR이 99% 이상의 성능을 보였으며, FPR은 1% 미만의 성능을 보였으며, Precision은 99%이상의 성능을 보였고, Accuracy는 99% 이상의 성능을 보였다. Angry Birds Rio Unlocker 악성코드의 경우 TPR이 99% 이상의 성능을 보였으며, FPR은 1% 미만의 성능을 보였으며, Precision은 99%이상의 성능을 보였고, Accuracy는 99% 이상의 성능을 보였다. Droid KungFu 악성코드의 경우 TPR이 99% 이상의 성능을 보였으며, FPR은 4% 미만의 성능을 보였으며, Precision은 99%이상의 성능을 보였고, Accuracy는 99% 이상의 성능을 보였다. Gold Dream 악성코드의 경우 TPR이 99% 이상의 성능을 보였으며, FPR은 1% 미만의 성능을 보였으며, Precision은 99%이

상의 성능을 보였고, Accuracy는 99% 이상의 성능을 보였다. Smspy 악성코드의 경우 TPR이 99% 이상의 성능을 보였으며, FPR은 30.07%의 성능을 보였으며, Precision은 81.26%이상의 성능을 보였고, Accuracy는 82.30% 이상의 성능을 보였다. Zcaler 악성코드의 경우 TPR이 99% 이상의 성능을 보였으며, FPR은 1% 미만의 성능을 보였으며, Precision은 99%이상의 성능을 보였고, Accuracy는 99% 이상의 성능을 보였다. FakeApp.AL 악성코드의 경우 TPR이 99% 이상의 성능을 보였으며, FPR은 0% 미만의 성능을 보였으며, Precision은 99%이상의 성능을 보였고, Accuracy는 99% 이상의 성능을 보였다. 대부분의 악성코드가 높은 성능을 보였다. 하지만 Smspy 악성코드의 경우 FPR이 30% 이상의 결과가 나왔기 때문에 굉장히 높은 비율이라고 볼 수 있다. 하지만 100개 중 70개는 탐지할 수 있다고 사료되며, 대부분의 악성코드를 탐지 가능하다고 볼 수 있다. 따라서 SVM을 이용하여 악성코드를 탐지를 수행할 경우 새로운 악성코드의 탐지가 가능하다. 하지만 추후 Smspy와 같은 악성코드의 경우 향후 어떤 방법으로 탐색할 수 있을지 향후 탐지 방안을 찾아볼 예정이다.

표 6. 트레이닝 셋에 없는 악성코드 별 탐지 결과
Table 6. Malware Detection Results without Training Sets

	TPR	FPR	Pre	Acc
Anserver	99.98%	0.99%	99.69%	99.76%
Angry Birds Rio Unlocker	99.98%	0.1%	99.98%	99.98%
Droid KungFu	99.98%	3.32%	98.99%	99.22%
Gold Dream	99.98%	0.1%	99.98%	99.98%
Smspy	99.98%	30.07%	81.26%	82.30%
ZScaler	99.98%	0.1%	99.98%	99.98%
FakeApp. AL	99.98%	0.1%	99.98%	99.98%

V. 결론 및 향후 연구

본 논문에서는 안드로이드 플랫폼에서 선형 SVM 알고리즘을 사용하여 악성코드 탐지 성능을 향상시키고 새로운 악성코드에 대한 대처 능력에 대한 실험을 수행하였다. 탐지 성능을 향상시키기 위해서 루팅을 수행하지 않고 추출할 수 있는 feature를 추출하였으며, feature의 수가 많을수록 많은 수행시간이 걸리기 때문에 최소의 feature를 추출하여 수행하였다. 또한 탐지 성능 향상을 위해 정규화를 수행한 결과 0.4초 정도 시간을 단축하였으며, Accuracy의 경우 0.3%의 성능 향상을 보였다. 정규화를 수행하였을 때 더 높은 성능을 보였다.

동적 분석 기법이 나타난 배경은 새로운 악성코드에 대한 탐지를 위해서이다. 본 논문에서는 새로운 악성코드에 대해 선형 SVM 알고리즘이 얼마나 대처할 수 있는지 알아보기 위해 실험을 수행하였다. 실험은 트레이닝 셋에서 악성코드를 하나씩 뺀 후 수행했을 경우 성능에 대해 알아보았다. 그 결과 대부분의 악성코드에 대해 FPR이 4%이하의 성능으로 높은 성능을 보였지만, Smspy 악성코드의 경우 30.07%로 탐지하지 못했다고 볼 수 있다. 탐지하지 못한 악성코드의 경우 탐지하지 못한 악성코드의 경우 정상과 비슷한 패턴을 보이기 때문에 탐지하지 못하였다. 따라서 향후 연구에서 유사한 패턴을 보이는 악성코드에 대한 탐지 방법을 모색할 예정이다.

향후 연구로는 하이브리드 기법을 이용하여 더 높은 성능을 보이는 악성코드 탐지 방법을 제안할 것이며, 정상과 유사한 패턴을 갖는 악성코드에 대한 탐지 방법을 모색할 예정이다. 또한 제한된 기기가 아닌 다른 기종의 이식성을 높일 수 있는 방안을 모색할 계획이다.

References

[1] 안철수연구소 ASEC, "2015 모바일 보안위협 예상 트렌드 Big4", 2015.

[2] IDC, "Android and iOS Squeeze the Competition, Swelling to 96.3% of the Smartphone Operating System Market for Both 4Q14 and CY14, According to IDC ", 2015.

[3] 전철, et al. 효율적인 안드로이드 애플리케이션 검수를 위한 견고한 퍼미션 기반 악성 애플리케이션 여과 기법. 보안공학연구논문지, 2013, 10.2.

[4] ENCK, William: ONGTANG,

Machigar; MCDANIEL, Patrick. On lightweight mobile phone application certification. In: Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009. p. 235-245.
〔5〕 SHABTAI, Asaf, et al. Mobile malware detection through analysis of deviations in application network behavior. Computers & Security, 2014, 43: 1-18.
〔6〕 Hyo-sik Ham, Mi-Jung Choi, "Analysis of Android Malware Detection Performance using Machine Learning Classifiers", International Conference on ICT Convergence, 2013.
〔7〕 김환희; 최미정. 자기회귀 이동평균 모델을 이용한 안드로이드 악성코드 탐지 기법. 한국통신학회 종합 학술 발표회 논문집 (동계) 2015, 2015, 145-146.
〔8〕 ZHAO, Shuai, et al. Attack Tree Based Android Malware Detection with Hybrid Analysis. In: Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on. IEEE, 2014. p. 380-387.
〔9〕 ELBASIONY, Reda M., et al. A hybrid network intrusion detection framework based on random forests and weighted k-means. Ain Shams Engineering Journal, 2013, 4.4: 753-762.
〔10〕 LEE, Changki; JANG, Myeonggil. Named entity recognition with structural SVMs and pegasos algorithm. Korean Journal of Cognitive Science, 2010, 21.4: 655-667.

김 기 현 (Ki-Hyun Kim)

2014년 2월 : 강원대학교 컴퓨터과학과 학사
2014년 3월 ~ 현재 : 강원대학교 컴퓨터과학과 석사

<관심분야> SDN/NFV, 정보보안

최 미 정 (Mi-Jung Choi)

1998년 : 이화여자대학교 컴퓨터공학과 학사
2000년 : 포항공과대학교 컴퓨터공학과 석사
2004년 : 포항공과대학교 컴퓨터공학과 박사
2004~2005년 : 프랑스 INRIA 연구소 박사후 연구원
2005~2006년 : 캐나다 워터루 대학 컴퓨터과학부 박사후 연구원
2006~2008년 : 포항공대 컴퓨터공학과 연구 조교수
2008년 ~ 현재 : 강원대학교 컴퓨터과학과 부교수
<관심분야> SDN/NFV, 정보보안