논문 19-44-08-13 The Journal of Korean Institute of Communications and Information Sciences '19-08 Vol.44 No.08
https://doi.org/10.7840/kics.2019.44.8.1535

네트워크 프로토콜에 대한 역공학을 통한 상태추론 기법

정 택 현◆, 송 원 종*, 김 기 천°

State Inference Method through Reverse Engineering for Network Protocols

Tack-hyun Jung◆, Won-jong Song*, Kee-cheon Kim°

요 약

본 논문은 전송계층의 프로토콜을 Network Trace 기반의 역공학(Reverse Engineering)을 수행하여, 그 구조와 논리 그리고 상태를 추론하는 아키텍처를 제안한다. 공개되지 않거나 제한적으로 공개된 프로토콜을 통해 통신하는 경우 통상적인 방법으로는 그 정보를 알 수 없으므로 다양한 문제점과 보안 이슈가 발생한다. 따라서, 본 연구에서 제안하는 아키텍처는 구조나 논리를 모르는 프로토콜을 대상으로도 군집화, 이벤트 분석 등 여러 기술을 통해 프로토콜이 갖는 Syntax, Semantics, Timing 정보를 추출하는 것을 목표로 한다. 다양한 사전연구를 적극적으로 활용하여 구현된 해당 아키텍처는 FTP(TCP), UDP, QUIC 등 다양한 프로토콜을 대상으로 아무런 사전정보 없이, 그 구조와 논리를 파악하고 검증하는 실험에서 높은 성능을 보였으며, 프로토콜이 갖는 FSM(Finite State Machine)을 자동화된 오토마타로 정확히 표현하는 결과를 보였다. 또한, 이러한 결과를 토대로 UDP와 같은 비신뢰성 프로토콜에서도 프로토콜 자체의 수정 없이, 오류정정이나 혼잡제어와 같은 기술을 수행할 수 있도록 하는 응용방안에 관해서도 제시한다.

Key Words : Protocol, Reverse Engineering, Machine Learning

ABSTRACT

This paper proposes an architecture that inferences a structure, logic and the status of the transport layer protocol through the network trace-based reverse engineering. when communicating with protocols that are not open or limitedly open to the public, various problems and security issues can rise because the protocol's information can't be interpreted by an usual way. Hence, the proposed architecture in this paper uses clustering, event analysis, and various other skills to extract syntax, semantic, and timing from protocols of which structure or logic is not known. Our architecture, which is implemented with various prior technologies, shows high performance in our experiment in identifying and verifying the structure of various protocols such as FTP(TCP), UDP, QUIC and others with absence of any prior information. And the protocols' FSM(Finite Stat Machine) was exactly deduced through the automated automata. Based on these results, we also propose an application method that can perform the error correction and the congestion control without modifying a non-reliable protocol such as UDP.

※ 본 연구는 2019년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보·컴퓨팅기술개발사업의 지원(No. NRF-2017M3C4A7083678,더 나은 Web 경험을 위한 자율제어 네트워킹 애널리틱스 기술)을 받아 수행되었습니다.
◆ First Author : Konkuk University Department of IT Convergence Information Security, tackhyun12@konkuk.ac.kr, 학생회원
° Corresponding Author : Konkuk University Department of Computer Engineering, kcim@konkuk.ac.kr, 종신회원
* Konkuk University Department of IT Convergence Information Security, swj0630@konkuk.ac.kr, 웹회원
논문번호 : 201905-067-D-RN, Received May 3, 2019; Revised May 31, 2014; Accepted June 5, 2019

Ⅰ. Introduction

최근 인터넷 환경에서는 다양한 프로토콜을 통해 많은 정보교환이 이루어지고 있다. 정보를 교환하기 위해 메시지를 주고받는 호스트들은 대부분 공개된 국제표준 프로토콜이나 TCP와 같은 범용적인 프로토콜을 통해 통신을 수행한다. 하지만 사물인터넷(IoT)과 클라우드와 같은 새로운 분야가 발전하고 점차 생활에 밀접한 관계를 띄기 시작함에 따라, 기존 프로토콜의 한계점이 나타나기 시작했다.[1] 예를 들어, 기존의 TCP 프로토콜은 신뢰성이 보장되지만, IoT 기기 등에서 활용하기에는 너무 무거워서 지연이 발생한다. 이러한 문제점이 발생함에 따라 경량화 혹은 속도향상 등 다양한 연구를 통해 새로운 프로토콜이 개발되었고, 새로운 프로토콜의 개발은 서비스의 품질 등 편의성 면에서는 많은 개선이 되었지만, 반대로 보안과 같은 분야에서는 새로운 문제가 대두되기 시작하였다.

보안의 측면에서는 프로토콜의 구조나 논리가 공개되지 않았거나 제한적으로 공개된 경우 통상적인 방법으로는 해당 프로토콜의 정보를 알 수 없다. 이는 관리하거나 통제할 방안이 없다는 점을 의미하기도 한다. 이러한 프로토콜 외에도 자체적으로 개발되거나 수정이 가해진 프로토콜의 경우 암호화, 혼잡제어 등의 기능이 지원되지 않는 경우가 상당수 존재하며[2] 이로 인해 네트워크의 다양한 문제점과 보안 문제가 발생한다.

본 논문에서는 이러한 다양한 문제점을 해결하기 위해 구조나 논리를 모르는 전송계층의 프로토콜을 패킷 Trace를 기반으로 역공학을 수행하여 정보와 상태를 추론하는 아키텍처를 제안한다.

Ⅱ. Related Works

2.1 Protocol Reverse Engineering

프로토콜 역공학이란, 사전적으로는 프로토콜을 분석하여 현재 구성 요소와 그 의존성을 파악하고, 프로토콜 추상화 및 설계 정보를 추출 및 생성하는 과정을 의미한다. 프로토콜 역공학과 관련된 연구는 인터넷망이 구성되었을 때부터 존재했으며, 그 존재 의의는 기술의 발전이 빨라지고 있는 현재 더욱 주목받고 있다. 이러한 프로토콜 역공학은 주로 2단계로 구분되어 수행된다.

1) 프로토콜 메시지의 종류, 구조, 크기의 분석
2) 상태 머신(State-Machine)을 통한 상관관계의 분석

전통적인 프로토콜 역공학은 사람이 수작업으로 프로토콜의 구조를 파악하여 어떻게 메시지를 전송하며 어떤 형태로 State-Machine을 구성하는지를 분석하였다. 하지만 기술의 발전에 따라 현재는 군집화와 같은 다양한 자동화 방법을 통해 각 프로토콜의 특징으로부터 구조와 상태를 분석하는 단계로 발전해 왔다. 본 연구에서도 이러한 흐름에 따라 군집화, Fuzzing 등의 다양한 기술을 접목하여 프로토콜을 역공학을 수행하는 아키텍처를 생성 및 제안하였다. 이러한 내용은 본 논문의 3장에 자세히 기술되어 있다.

2.2 FSM(Finite State Machine)[3]

FSM이란, 시스템이나 아키텍처를 설계할 때 사용되는 모델이다. 이는 시스템이나 아키텍처가 유한한 상태를 가진다고 가정하고, 각각의 상태별 동작과 상태 간의 전이에 관한 내용을 설계하고 정의한다. 본 연구에서는 프로토콜 또한 유한한 상태를 가지고 있다는 특징에서 착안하여, FSM을 통해 프로토콜을 오토마타로 표현하고자 한다. 실제 FSM 부분을 구현하는 데는 선행연구[3]의 수식과 이론을 참조하였다.

$$Specification = \{\sum Message\ Type = \{M_1, M_2, .., M_n\}, \sum Transition = \{T_1, T_2, .., T_k\}\} \quad (1)$$

$$M = \{F_1, F_2, .., F_m\} \quad (2)$$

$$T = \{transition\ probability, M_i \rightarrow M_j\} \quad (3)$$

수식(1)은 프로토콜의 사양을 의미하며 메시지의 유형(M)과 메시지 간의 전이(T)들을 원소로 갖는다. 수식(2)은 하나의 메시지 유형을 의미하며, 구성하는 필드들을 순서대로 원소로 갖고 있다고 가정한다.[3] 수식(3)은 메시지 유형 간 전이를 의미하며, 이에 해당하는 확률을 원소로 갖는다. 이를 통해 메시지의 유형 (2)을 state로 하고, 각 메시지 유형 간 전이(3)를 기반으로 하여 프로토콜의 FSM 표현할 수 있다.[3] 이러한 원리에 따라 프로토콜을 역공학 하는 과정에서 획득하는 정보를 기반으로 오토마타를 표현한다.

Ⅲ. Proposed Methodology

3.1 Proposed Architecture

네트워크 환경에서 여러 호스트는 다양한 프로토콜을 사용하여 메시지를 교환한다. 본 연구에서 제안하는 아키텍처는 [Fig 1]과 같이, 교환과정에서 발생하

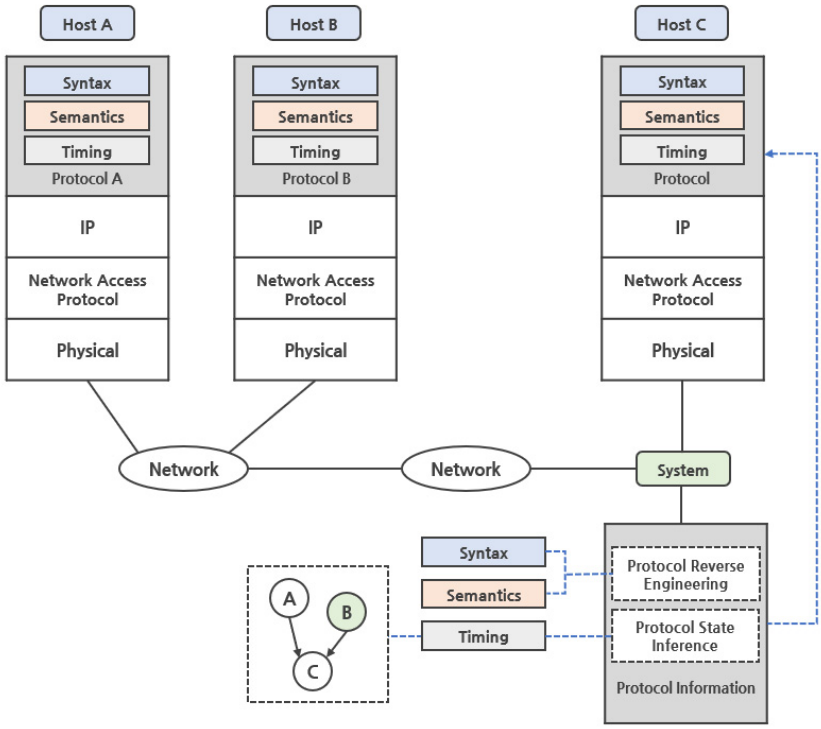

그림 1. 본 연구에서 제안하는 프로토콜 상태추론 아키텍처
Fig. 1. Architecture of Protocol State Inference

는 네트워크 Trace를 기반으로 프로토콜의 구조나 논리를 역공학을 통해 분석하고자 한다.

3.1.1 Kernel Bypass for Pre-Preparation

네트워크의 프로토콜을 분석하여 추론하기 위해서는 Kernel-Level이 아닌, User-Level에서의 작업이 필요하다. 따라서 본 연구에서는 PF_RING[5] 라이브러리를 기반으로 동기식 Kernel-Bypass 환경을 구축하였다. [Fig 2]는 이러한 Kernel-Bypass 환경의 구조를 표현한다. PF_RING을 통해 구현된 연구환경은 메시지를 Control 영역과 Data 영역으로 구분하여 네트워킹을 수행한다.[5]

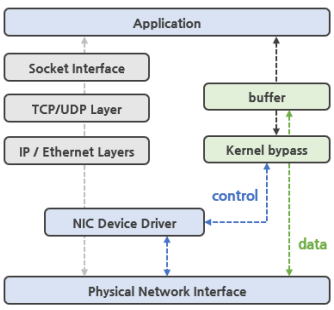

그림 2. 본 연구에서 활용된 커널 바이패스 환경
Fig. 2. Kernel bypass environment used in the study

또한, 별도의 Thread를 생성하여 패킷 캡처만 수행하도록 전담시키기 때문에 일반적인 Linux Kernel 대비 오버헤드 발생이 더 적게 발생한다는 연구결과가 있다.[4] 이러한 PF-RING을 통해 구현된 환경의 장점은 메시지가 손실(Loss)되어 역공학이 제대로 수행되지 않는 경우를 방지하는 효과가 있다.

3.1.2 Message Capture

하나의 프로토콜의 정보를 완전히 분석하기 위해서는 대량의 메시지 Trace가 필요하다. 소량의 정보만으로는 프로토콜이 갖는 Syntax, Semantics, Timing의 정확한 의미를 파악할 수 없다.

특히, Timing의 정보는 메시지의 순서와 연관되기 때문에 메시지의 개수에 대한 의존성이 클 수밖에 없다. 만일 적은 수의 메시지 Trace를 분석 표본으로 활용한다면, 특정 메시지의 통신이 누락되는 경우가 발생할 수 있다. 이러한 특징에 따라 Real-Time 기반의 분석이 아닌, Network-Trace 기반의 분석방식을 적용하였다. 이러한 분석방식은 역공학에 사용될 메시지 데이터를 반드시 User-Level에서 수행하는 과정이 필

요하다. 따라서, PF_RING의 TCPDUMP[5]를 기반으로 메시지 수집 기능을 구현하였다.

3.1.3 Process of Proposed Architecture

메시지를 수집하고 분석하는 과정에서 가장 중요한 것은 분석대상을 정확히 추출하는 것이다. 여러 종류의 프로토콜의 메시지가 혼재되어 있으면, 분석에 어려움이 생길 수 있다.

따라서, 본 연구에서는 [Fig 3]과 같은 분석절차를 적용한다. 먼저 Network-Trace에 혼재된 여러 프로토콜의 메시지를 Clustering 알고리즘을 통해 구분한 뒤, 그 결과별로 이벤트 분석을 수행하여 프로토콜의 Syntax, Semantics, Timing 정보를 획득한다. 최종적으로는 이러한 결과를 기반으로 프로토콜의 상태추론을 수행하는 개념이다.

3.2 Protocol Clustering

여러 프로토콜을 구분하기 위한 군집화 알고리즘은 DBSCAN(Density-based spatial clustering of applications with noise)[7]을 활용한다. DBSCAN이란, 비지도 학습의 대표적인 알고리즘으로, 군집의 개수를 따로 지정하지 않아도 효율적으로 데이터를 분석하여 군집화를 수행한다.[8] DBSCAN의 군집화 결과는 여러 번 반복 수행하더라도 같은 결과가 일관성 있게 나타난다는 특징이 있다.[7]

[Fig 4]는 DBSCAN을 통해 수행하고자 하는 군집화의 방향과 목적성을 의미한다. 이는, 그림에 표시된 3개의 군집이 프로토콜 A, B, C로 명확히 구분된다고 가정할 때, 새로운 점(P)가 발생하면 3개의 군집 중에서 거리와 밀도 기반으로 가까운 군집을 찾아서 속하거나, 최소 개수(minPts)와 밀도(eps)를 기반으로 새

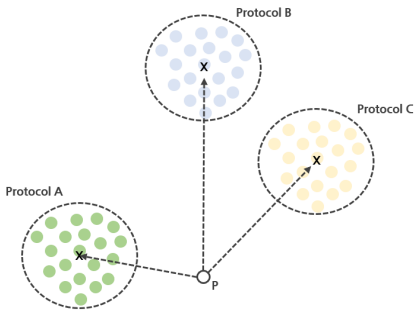

그림 4. DBSCAN 알고리즘을 통한 군집화
Fig. 4. Clustering using DBSCAN Algorithm

로운 군집을 생성하는 과정을 의미한다.

본 연구는 이러한 목적성을 갖는 군집화 모델을 구현하기 위해 다양한 선행연구를 참고하였다.[19,20] 먼저, Network-Trace에서 출발지와 목적지 주소를 기준으로 메시지의 횟수와 시간 그리고 크기와 순서 등을 Feature로 선정한다. 이후, 군집화의 성능을 개선하기 위한 과정으로 특정한 기준시간 동안 선정된 Feature 데이터를 대상으로 휴리스틱(Heuristic) 하게 일차적으로 가공하는 Feature Engineering을 수행한다.

[Fig 5]는 2개의 프로토콜이 혼재된 Trace에서 군집화를 수행한 결과의 예시이다. 그림의 초록색과 빨간색은 각기 다른 프로토콜의 데이터를 의미한다. 이러한 결과는 군집이 명확하게 구분되지 않으며, 중복되는 구간이 크게 발생함을 의미한다. 따라서, 본 연구는 시간 데이터를 더욱 정밀하게 가공하여 이러한 문제를 해결한다. 시간과 관련된 Feature를 밀리세컨드(millisecond) 단위로 duration 혹은 interval 등으로 가공하면 군집화 수행 시 성능향상에 큰 도움이 된다. 일반적으로는 Kernel이 밀리세컨드 단위의 네트워킹

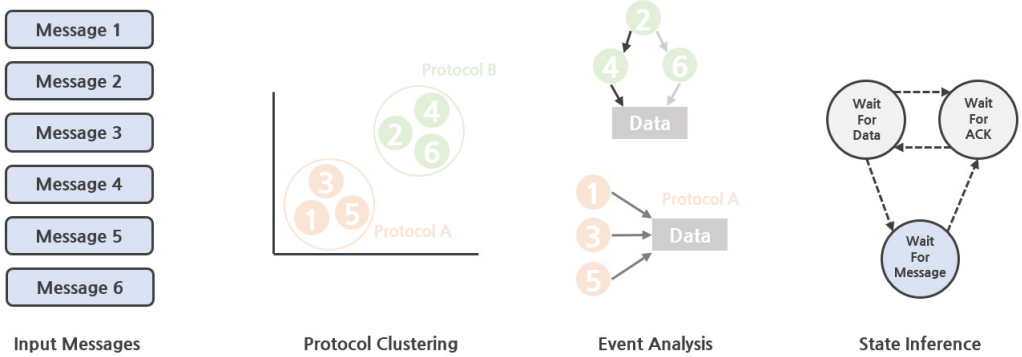

그림 3. 네트워크 프로토콜의 상태를 추론하기 위한 과정
Fig. 3. Process for inference of protocol's state

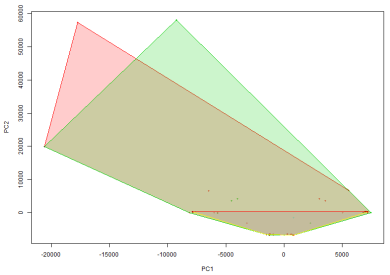

그림 5. Feature Engineering을 적용하지 않은 DBSCAN의 결과
Fig. 5. Result of Feature Engineering unapplied DBSCAN

을 지원하지 않는 경우가 있다. 이는 초 미만의 데이터가 모두 0으로 기재되는 문제점이 있지만, 앞서 Kernel-Bypass 환경을 구축하였기 때문에 이러한 처리가 가능하다.

데이터에서 의미를 제대로 표현하는 특징을 갖는 성분을 추려내는 과정을 주성분 분석(PCA)[21]이라고 한다. 본 연구는 이러한 주성분 분석을 심층 신경망 기반의 Auto-Encoder[20]을 활용하여 구현하였다. 이는 활성화 함수를 sigmoid, ReLU과 같은 비선형이 아닌, 선형 함수를 사용하는 것과 손실함수로 MSE(Mean Squared Error)를 사용하여 주성분 분석의 효과를 얻을 수 있다.

[Fig 6]은 이러한 일련의 과정 이후의 DBSCAN의 결과이다. 그림과 같이 2개의 각기 다른 프로토콜이 군집화를 통해 분류된 것을 확인할 수 있다.

3.3 Message Event Analysis

군집화를 통해 프로토콜을 분류한 이후에는 프로토콜이 갖는 여러 종류의 메시지들을 세부적으로 분류하는 과정을 수행한다. 프로토콜은 상호 정보교환을 위해 다양한 메시지를 교환한다. 이러한 메시지들은 정의된 순서에 따라 서로 다른 다양한 구조와 의미가 있다. 이는 일반적으로 프로토콜의 Syntax, Semantics, Timing으로 구분되어 정의된다.

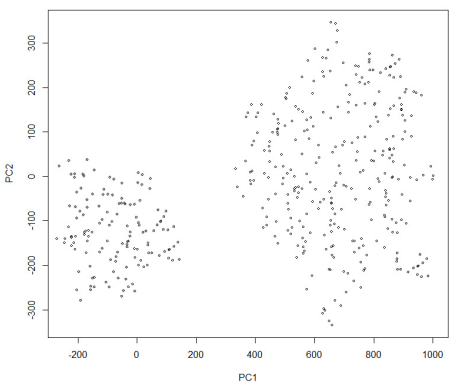

그림 6. Feature Engineering을 적용한 DBSCAN의 결과
Fig. 6. Result of Feature Engineering applied DBSCAN

프로토콜에서 Syntax란, 메시지가 갖는 Field와 Data의 각각의 형식과 크기 등을 의미한다. Semantics이란, 메시지가 갖는 의미를 뜻하며, Timing은 여러 개의 메시지 중 언제 어떠한 순서를 통해 메시지를 전송할 것인지와 같은 정보를 포함한다. 프로토콜의 상태를 추론하기 위해서는 반드시 이 모든 정보가 필요하다.[9]

기존의 다양한 연구에서 프로토콜 역공학을 통해 이러한 정보를 추출하는 방법론을 다뤄왔지만, 현재까지 표준화된 추출 방법은 존재하지 않는다. 따라서 이러한 다양한 방법 중 메시지에서 Static Field를 추출하여 이벤트를 구분하는 방법으로 구현하였다.

3.3.1 Message Field Extraction

[Fig 7]을 살펴보면, TCP에서 세션을 수립하는 과정에서 교환되는 메시지의 Flags 필드가 “SYN” 등 고정된 값을 찾을 수 있다. 이처럼 정적인 값을 가지는 필드를 Static Field라고 하며, 가변적으로 변하는 값을 가지는 필드를 Dynamic Field라고 한다. 본 연구에서는 고정된 길이를 갖는 Static Field에 주목하였고, 고정된 상수 키워드를 식별하여 필드 전체 길이에서 그 길이만큼 빼는 방식으로 Data의 길이를 계산한다. 이는 아래의 수식(4)으로 정의된다.

$$Data_{size} = Field_{size} - Keyword_{size} \tag{4}$$

[Fig 8]을 살펴보면, 고정된 값을 갖는 상수 키워드들이 반복적으로 존재한다. 이처럼 반복되는 키워드를 추출하는 과정이 선행되어야 수식(4)을 적용할 수 있다. 따라서 [Algorithm 1]은 정적 키워드를 추출하는 방법과 과정을 정의한다. 메시지에서 필드를 추출하고 필드에서 키워드를 찾는 순으로 수행된다.

메시지에서 필드를 구분하는 과정은 Apriori[23]알고리즘을 기반으로 정적필드를 추출하는 방법을 제안한 선행연구를[22]기반으로 한다. Apriori는 연관규칙분석

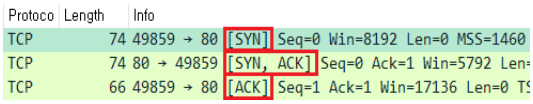

그림 7. 메시지의 필드가 가진 고정된 값을 갖는 상수 키워드
Fig. 7. Constant Keyword with fixed value in the field of message

그림 8. 메시지에서 반복되는 고정된 키워드
Fig. 8. Repeated static Keyword in message

을 통해 집합이 얼마나 빈번하게 발생하는가를 분석하고 일련의 규칙을 만드는 알고리즘이다. 선행연구는[22] 이러한 알고리즘에 3가지 지지도를 적용하는 Statistics Filter 단계와 필드의 메시지 내 위칫값들의 분포를 고려하는 Position-Value Filter의 단계를 통해 메시지의 필드를 추출하는 이론을 정립하였다.[22] 따라서 본 논문은 이러한 이론을 근간으로 구현을 수행하며, 필드의 추출이 완료되면, 그 값을 1byte씩 증감 탐색하여 리스트에 저장한 뒤, 반복적으로 다른 메시지의 필드에 포함된 값이 있는지 검증하여 키워드를 찾는다.

Algorithm 1 : Keyword Extraction

```
1.  Input : Messages msgs

    def read_bytes(field, length):
        data_byte = msg.read(length)
2.      if len(data_byte) != length:
            raise Exception("error")
        return data_byte

    temp_list, keyword_list = []
    for msg in msgs:
        for field in msg:
            length = 3
3.          for i in range(length, field.size):
                keyword = read_byte(field, length)
                if keyword in temp_list:
                    keyword_list.append(keyword)
                else:
                    temp_list.append(keyword)

4.  Output : keyword_list
```

3.3.2 Timing Inference

획득한 키워드를 통해 해당 메시지가 갖는 순서를 파악할 수 있다. 이는, 대량의 메시지 Traffic에서 각각의 키워드가 몇 회 반복되었는지 비율과 순서 등을 산정하여 구한다. 또한, 하나의 키워드를 기준으로 이후에 나타나는 다음 키워드와 연관성과 종속성을 찾는 작업을 반복적으로 수행토록 구현한다.

[Fig 9]는 키워드를 기반으로 연관성을 탐지 과정의 예시이다. TCP는 "SYN" 키워드를 가진 메시지 이후의 순서에서 "SYN+ACK" 키워드의 메시지가 발생한다. 이후에는 "ACK" 키워드의 메시지가 나타나게 되는데, 이러한 메시지 순서의 교환을 두 호스트의 IP 주소를 교차하여 검증하는 것으로 구할 수 있다. 이를 통해 프로토콜이 갖는 Timing 정보를 획득한다.

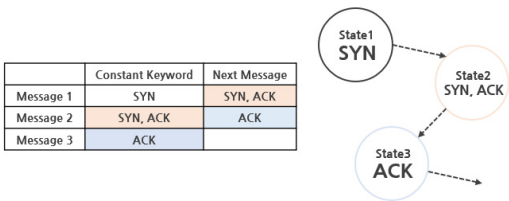

그림 9. 키워드를 통한 Timing과 Semantics 분석 구조
Fig. 9. Analytical structure of Timing and Semantics through Keyword

3.3.3. Message Semantics Inference

프로토콜의 Timing 정보와 (4)를 통해 얻은 Syntax 정보를 통해 메시지가 갖는 의미를 파악할 수 있다. UDP처럼 단순히 데이터 메시지를 보내는 프로토콜도 있지만, 세션을 확보하고 제한적으로 데이터를 보내는 프로토콜도 존재한다. 따라서, Timing 정보를 기반으로 키워드의 연관성과 의미를 검증한다.

먼저, 전체 Network-Trace에서 데이터 메시지를 식별한다. 이는, Syntax 정보 중 메시지의 크기나 키워드 정보들을 통해 알 수 있다. 이후, 데이터 메시지를 보내는 전/후에 발생하는 메시지를 Timing 정보에서 확인하는 과정을 수행한다. 이를 통해 메시지 간의 연관성을 찾을 수 있다. 또한, 전체 통신기록에서 각각의 메시지 유형이 몇 회 수행 및 반복되는지를 산술적으로 비율을 계산하여 획득한다.

이러한 결과는 메시지가 어떠한 의미가 있는지 추론할 수 있게 하며, 데이터 메시지와 다른 메시지를 명확히 구분할 수 있다. 이는 Semantics 정보의 획득을 의미한다

3.3.4 Verification through Fuzzing

Fuzzing은 [Algorithm 1]의 과정에서 획득한, 메시지가 가지고 있는 필드 크기의 검증 및 보강에 활용한다.

[Fig 10]는 획득한 필드의 키워드 뒤에 “A”와 “a”를 추가하는 행위를 반복하는 예시를 의미한다. 필드의 값에 Fuzz 데이터를 추가한 뒤, 메시지로 재조립하여 통신의 과정에서 오류 발생을 검증한다. 이를 통해, 각각의 필드의 정확한 offset 크기를 획득한다. “A”와 “a”를 교차로 반복하는 이유는 오류가 발생하였을 때, 몇 번째의 문자열에서 발생하였는지 쉽게 찾기 위함이다. Fuzz를 수행함에서, 그림의 msg4와 같이 키워드가 없거나, 탐지하지 못한 필드의 경우에는 키워드 없이 Fuzzing을 진행한다.

Fuzzing은 네트워크의 추가적인 Trace를 만들게 된다는 단점이 존재하며, 패킷을 조립하고 만드는 과정에서 지연이 발생하고 분석에 오랜 시간이 소요되는 문제가 있다. 따라서 기계학습을 접목하여 효율성 혹은 속도를 높이거나, Mininet[10]과 같은 가상 네트워크를 활용하여, Trace를 만들지 않는 등의 추가적인 개선 방안의 필요성이 제기된다.

Seq	Keyword of Field	Fuzz
msg1	SYN	SYN **AaAaAaAaAa**
msg2	SYN, ACK	SYN, ACK **AaAaAa**
msg3	ACK	ACK **AaAaAaAaAa**
msg4		**AaAaAaAaAa.....AaA**

그림 10. 메시지의 필드를 대상으로 fuzzing을 수행하는 예시
Fig. 10. Sample to perform fuzzing on fields in message

3.4 Protocol State Inference

3.4.1. State Inference through buffer

본 연구에서는 프로토콜의 상태를 추론하는 것을 목적으로 한다. 상태의 추론방식은 Stop-and-Wait[25]와 유사한 방법으로, 통신 중 메시지의 수신이 완료되었는지와 수신 중인지를 구분하여 파악하는 것으로 한다. [Fig 11]은 제안하는 네트워크 상태추론 아키텍처를 의미한다.

그림에서 Timing에 표시된 정보()는 Host A가 보낸 메시지를 Host B가 수신할 때, 해당 메시지의 의미와 순서의 확인 과정을 의미한다. 즉, Host A가 보낸 현재 메시지는 ⓒ이며, Ⓐ 메시지 이후에 나타나고 다음 메시지로는 Ⓓ 가 발생함을 의미한다. 이러

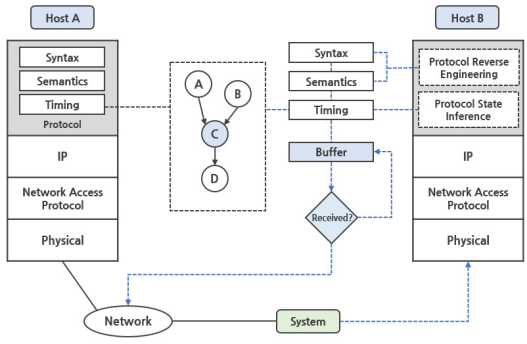

그림 11. 네트워크 프로토콜의 상태를 추론하는 아키텍처 및 원리
Fig. 11. Architecture and principles Inferring the state of network protocols

한 정보는 앞서 획득한 Timing 정보를 기반으로 한다.

본 연구에서 제안하는 상태추론 아키텍처는 데이터 메시지를 대상으로만 수행한다. 이는, 모든 메시지를 대상으로 상태추론의 적용이 가능하지만, 세션을 수립하는 과정의 메시지를 추론하는 것은 의미가 없다고 판단되기 때문이다.

데이터 메시지를 대상으로, 획득한 필드의 길이를 모두 합산하여 그 수치와 같은 길이의 버퍼를 생성하며, 수신과정에서 수신된 크기만큼 버퍼에 하나씩 누적하여 크기를 비교하는 방식으로 전송의 완료 여부를 추론한다. [Algorithm 2]는 이러한 상태추론 기술에 대한 정의를 의미한다.

Algorithm 2 : State Inference

1. **Input :** Message msg, Field_size fs, Buffer buf
2.
```
buf = [sum(fs)]
state = False
while(msg):
    if msg.arrive():
        buf = [1] * msg.arrive_size()
        if buf.is_Full():
            state = True # Received
            break;
        else:
            state = False # Receiving
```
3. **Output :** state (True/False)

3.4.2 Application method

본 절에서는 상태추론 기술을 통해 오류정정이나 혼잡제어 등을 지원하지 않는 프로토콜에서도 프로토콜 자체의 수정 없이, 이를 수행할 수 있도록 하는 응

용방안에 관해서 기술한다.

[Fig 12]는 TCP에서 흔히 사용하는 메시지 재전송과 같은 맥락의 응용방안을 의미한다. Sender는 메시지를 전송한 후, Receiver로부터 ACK 메시지를 수신할 때까지 대기한다. 이후, Receiver는 상태추론 기술을 통해 수신의 진행 상황을 확인한다. 만일, 수신이 완료되면 Receiver는 ACK 메시지를 통해 Sender에게 알리게 된다. 만약, Receiver에게 제한시간 내에 정상적으로 도달하지 못하거나, Sender의 자체적인 Time-out이 발생한 경우에는 메시지의 재전송을 수행한다.

이러한 상태추론 기술은 UDP와 같이 비-신뢰성 프로토콜에도 적용 가능하며, 이를 통해 네트워크에서 발생하는 다양한 응용 처리를 추가 적용할 수 있다. 하지만, 이는 Receiver 호스트와 Sender 호스트 양쪽 모두에 해당 아키텍처가 설치되어야 하는 단점이 있으며, 상태를 추론하는 과정에서 추가적인 지연이 발생하는 문제가 있다. 따라서, Real-Time으로 동작해야 하는 실제 네트워크에는 시기적으로 적용하기 어려우므로, 특수한 목적성을 갖는 환경이나, SDN(Software Defined Network) 등에 적용하는 것이 현재로서는 최적으로 판단된다.

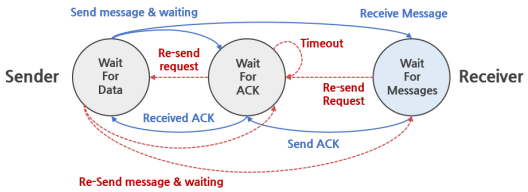

그림 12. 상태추론을 통한 응용방안
Fig. 12. Application method through state inference

Ⅳ. Evaluation Results and Analysis

4.1 Evaluation Targets

성능평가에서는 다양한 결과를 고찰하기 위해, TCP 기반의 FTP 프로토콜과 UDP 프로토콜 그리고 Google에서 개발한 QUIC[12]프로토콜을 실험대상으로 한다. 실험에 활용된 Trace는 3가지 프로토콜을 통한 통신이 동시에 수행되어 혼재된 상황이다. FTP는 다양한 명령어의 사용하면서 여러 개의 파일을 보내는 통신을 수행하였고, UDP는 단순한 문자열("text data goes here")을 불규칙하게 5,000건을 발송하는 통신을 수행하였다. 그리고 QUIC은 YouTube를 통해 비디오 스트리밍을 지속 수행하는 통신이 수행되었다.

[Table 1]은 이러한 성능평가의 대상에 대한 정보를 의미한다.

표 1. 성능평가 대상의 통신기록에 대한 요약
Table 1. Summary of communication records for performance evaluation targets

Trace	Protocol	Message Count	Contents
1	FTP(TCP)	1,682	File Transfer
	UDP	5,000	Text Transfer
	QUIC	786	Video Stream

4.2 Discussion

4.2.1 FTP Protocol Analysis Results

[Fig 13]는 FTP 프로토콜을 대상으로 수행한 결과 화면을 의미한다. 그림의 상단은 FTP 프로토콜의 Traffic Trace를 대상으로 Wireshark[24] 프로그램을 통해 표시한 화면이고, 하단은 제안하는 아키텍처의 수행결과이다.

Wireshark는, Open-Source 프로그램으로 다양한 프로토콜을 통한 통신기록을 확인하거나 교육에 활용되는 대표적인 패킷 분석 프로그램이다. 이러한 분석은 프로토콜의 Specification을 알고 있는 가정에서 구조 분석을 수행하므로, 정확한 결과가 보장된다. 하지만 본 아키텍처는 Specification을 모르는 상태에서 역공학을 통해 이러한 분석이 수행된다. 따라서 본 아키텍처의 성능평가는 공개된 프로토콜을 대상으로 아키텍처와 Wireshark의 결과를 비교하여 그 결과를 고찰

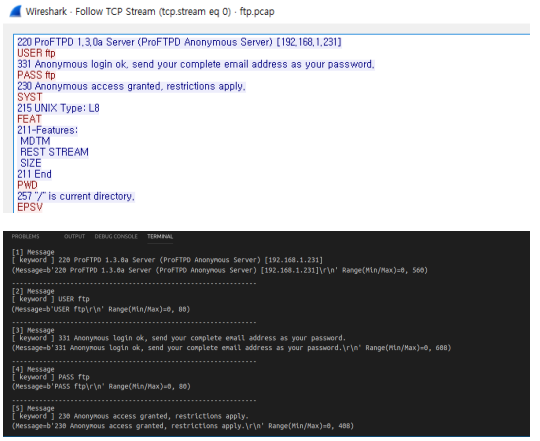

그림 13. FTP 대상 성능평가 수행결과 (위: WireShark 프로그램의 결과 / 아래: 본 아키텍처의 결과)
Fig. 13. Results of performance evaluation targeted for FTP protocol (Top: Wireshark program result, Bottom: Proposed architecture result)

하고자 한다.

[Table 2]는 FTP를 대상으로 성능평가를 수행한 결과 중 일부이다. 이는, 역공학을 통해 정확히 메시지 내 상수 키워드를 식별하였으며, 필드의 크기를 구한 것을 알 수 있다. 표의 Field Size는 메시지의 필드 크기를 의미하는데, 1번의 메시지의 필드 크기가 500보다 크게 되면 오류가 발생하는 점을 의미한다. 이는 Fuzzing을 통해 검증된 결과지만, fragmentation과 reassembly를 수행하는 장비에 영향을 받는 것으로 확인되었다.

표 2. FTP 메시지를 대상으로 역공학 수행결과
Table 2. Result of Reverse Engineering with FTP Message

Message	Keyword	Field Size
1	220 ProFTPD 1.3.0a Ser..	Range (0, 500)
2	USER ftp	Range (0, 80)

4.2.2 UDP Protocol Analysis Results

[Fig 14]는 UDP 프로토콜을 대상으로 수행한 성능평가 결과이다. 그림에 표현된 오토마타는 UDP가 하나의 State만을 갖는다는 점을 의미한다. 이러한 결과는 역공학을 통해 획득한 Timing과 Semantics 정보를 기반으로 한다. 또한, 그림을 살펴보면, 정확히 필드의 구조에서 데이터 문자열(“text data goes here”)만 추출한 것을 확인할 수 있다.

[Table 3]는 UDP를 대상으로 성능평가를 수행한 결과 중 일부이다. 정확히 메시지 내 데이터와 크기를 구한 것을 알 수 있다. 필드의 크기가 152를 초과하는 경우 Fuzzing에서 오류가 발생하는 것을 확인하였는데, 이는 마찬가지로 IP fragmentation과 reassembly을 수행하는 장비에 큰 연관성을 갖는다.

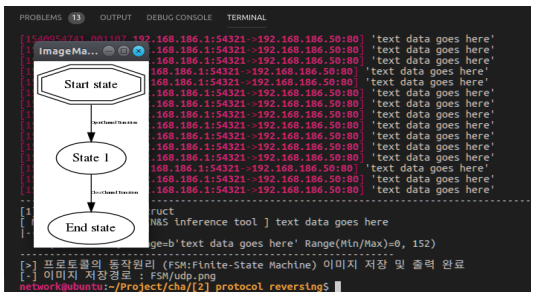

그림 14. UDP 메시지를 대상으로 역공학 수행결과
Fig. 14. Result of Reverse Engineering with UDP Message

표 3. UDP 프로토콜의 역공학 결과의 일부
Table 3. Part of reverse engineering result of UDP Protocol

Message	Data	Field Size
1	text data goes here	Range (0, 152)
2	text data goes here	Range (0, 152)

4.2.3 QUIC Protocol Analysis Results

[Fig 15]는 QUIC 프로토콜을 대상으로 수행한 성능평가 결과이다. 그림에 표현된 오토마타는 QUIC이 1회의 세션을 맺는 과정(Zero-RTT)이 있고, 이후부터는 UDP와 같이 데이터를 일방적으로 전송함을 의미한다. 이러한 결과는 마찬가지로 역공학을 통해 획득한 Timing과 Semantics 정보를 기반으로 획득한다.

[Fig 16]은 QUIC을 통해 수행된 비디오 스트리밍이 정확히 필드의 구조에서 데이터만 구분된 것을 의미한다. Syntax의 정보를 통해 정확히 암호화 통신이 수행되고 있는 데이터 부분만 추출할 수 있다. QUIC은 패킷이 전송 중 변조나 훼손되었을 경우 정정 비트를 통해 훼손된 비트를 복구하는 FEC를 제공하여[12] 이를 탐지함에서는 한계점을 보였었지만, 현재는 해당 기능이 표준에서 제외됨을 확인하였다.[13]

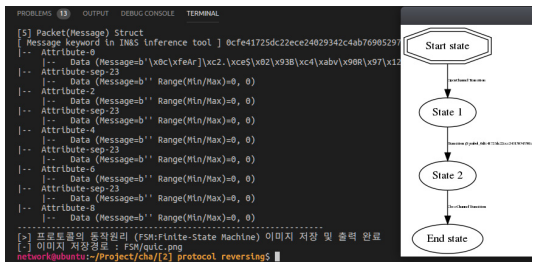

그림 15. QUIC 메시지를 대상으로 역공학 수행결과
Fig. 15. Result of Reverse Engineering with QUIC Message

그림 16. QUIC 메시지를 대상으로 역공학 수행결과(2)
Fig. 16. Result of Reverse Engineering with QUIC Message (2)

Ⅴ. Conclusion

본 논문에서는 구조나 논리가 공개되지 않은 프로토콜을 포함하여, 전송계층의 모든 프로토콜을 Trace 기반으로 역공학(Reverse Engineering)을 수행하여 정보와 상태를 추론하는 아키텍처를 제안하였다. UDP, QUIC과 같은 전송계층의 프로토콜을 통해 해당 아키텍처의 성능평가를 수행한 결과, 메시지가 갖는 Syntax, Semantics, Timing 등의 정보를 상당히 높은 정확도로 추출할 수 있었다. 다만, QUIC 프로토콜이 갖는 전방 오류정정 FEC 등에 대한 분석에서는 한계점을 보인다. 이러한 특별한 경우를 제외하고는 원하는 목표에 근접한 결과를 얻었기 때문에, 일부 개선사항을 적용하여 후속 연구를 진행할 예정이다. 먼저, Trace 기반의 역공학이 갖는 문제점인 데이터의 양에 대한 의존성을 GANs[14,15]모델을 통해 개선하고자 한다. 또한, Clustering과 Event Analysis에서 발생하는 지연을 최소화하는 것도 주된 연구 쟁점 중 하나이며, 마지막으로는 SDN 환경에서 실제로 해당 아키텍처를 적용한 결과를 추가로 연구할 계획이다.

References

[1] W. Shang, Y. Yu, R. Droms, and L. Zhang, *Challenges in IoT Networking via TCP/IP Architecture*(2016), Retrieved May 23, 2019, from https://nam ed-data.net/wp-content/uploads/2016/02/ndn-0038-1-challenges-iot.pdf

[2] S. Zamfir, T. Balan, I. Iliescu, and F. Sandu, "A security analysis on standard IoT protocols," *2016 Int. Conf. Applied and Theoretical Electricity (ICATE)*, pp. 1-6, Craiova, Romania, Oct. 2016.

[3] Y. Goo, K. Shim, J. Park, H. Hasanova, and M. Kim, "Architecture of network trace-based protocol reverse engineering system," in *Proc. KICS Conf.*, pp. 600-601, Korea, Nov. 2017.

[4] L Deri, "Improving passive packet capture: Beyond device polling," in *Proc. SANE*, vol. 2004, pp. 85-93, Oct. 2004.

[5] ntop.org, *PF_RING User Guide* (2017), Retrieved May 22, 2019, from https://www.ntop.org/guides/pf_ring/

[6] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proc. 2006 SIGCOMM workshop on Mining Network Data*, pp. 281-286, Pisa, Italy, Sep. 2006.

[7] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*, pp. 226-231, Portland, USA, Aug. 1996.

[8] G. S. Kim and I. R. Jeong, "Practical privacy-preserving DBSCAN clustering over horizontally partitioned data," *Korea Inst. Inf. Secur. and Cryptology*, vol. 20, no. 3, pp 105-111, Jun. 2010.

[9] P. M. Comparetti, G. Wondracek, C. Kruegel, and E. Kirda, "Prospex: Protocol specification extraction," *2009 30th IEEE Symp. Secur. and Privac*y, pp. 110-125, Berkeley, USA, May 2009.

[10] R. L. S. Santos, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using mininet for emulation and prototyping software-defined networks," *2014 IEEE COLCOM*, pp. 1-6, Bogota, Colmbia, Jun. 2014.

[11] D. Kruetz, F. Ramos, and P. Versissimo, "Software-defined networking: A comprehensive survey," in *Proc. IEEE 103.1*, vol. 103, no. 1, pp. 14-76, Jan. 2015.

[12] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W. Chang, and Z. Shi, "The QUIC transport protocol: Design and internet-scale deployment," in *Proc. Conf. ACM Special Interest Group on Data Commun.*, pp. 183-196, Los Angeles, USA, Aug. 2017.

[13] G. Carlucci, L. De Cicco, and S. Mascolo, "HTTP over UDP: an Experimental investigation of QUIC" in *Proc. 30th Annu. ACM Symp. Applied Computing*, pp. 609-614, Salamanca, Spain, Apr. 2015.

[14] Ian J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A.

Courville, and Y. Bengio “Generative adversarial nets,” *Advances in Neural Inf. Process. Syst.*, pp. 2672-2680, Montreal, Canada, Dec. 2014.

[15] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radfor, and X. Chen, “Improved techniques for training GANs,” *Advances in Neural Inf. Process. Syst. 29*, pp. 2234-2242, Barcelona, Spain, Dec. 2016.

[16] S. Gorbunov and A. Rosenbloom, “AutoFuzz: Automated network protocol fuzzing framework,” *Int. J. Comput. Sci. and Netw. Secur.*, vol. 10, no. 8, pp. 239-245, Aug. 2010.

[17] T. Krueger, H. Gascon, N. Kramer, and K. Rieck, “Learning stateful models for network honeypots,” in *Proc. 5th ACM Workshop on Secur. and Artificial Intell.*, pp. 37-48, Carolina, USA, Oct. 2012.

[18] A. Trifilo, S. Burschka, and E. Biersack, “Traffic to protocol reverse engineering,” *IEEE Symp. Computational Intell. Secur. and Defense Appl.*, pp. 1-8, Ottawa, Canada, Jul. 2009.

[19] J. Erman, M. Arlitt, and A. Mahanti, “Traffic classification using clustering algorithms,” *SIGCOMM Workshop on Mining Network Data*, pp. 281-286, 2006.

[20] J. Höchst, L. Baumgärtner, M. Hollick, and B. Freisleben, “Unsupervised traffic flow classification using a neural autoencoder,” *2017 IEEE 42nd Conf. Local Computer Netw. (LCN)*, Singapore, Singapore, Nov. 2017.

[21] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometrics and Intelligent Laboratory System*, vol. 2, no. 1-3, pp. 37-52, Aug. 1987.

[22] M. S. Lee, K. S. Shim, Y. H. Goo, and M. S. Kim, “A study on the analysis of the private protocol structure using the apriori algorithm,” in *Proc. Symp. KICS*, pp. 748-749, Jeju, Korea, Jun. 2018.

[23] R. Agrawal and R. Srikant. “Fast algorithms for mining association rules,” *Int. Conf. Very Large Database(VLDB)*, pp. 487-499, San Fransico, USA, Sep. 1994.

[24] Graham Bloice, *Wireshark Userguide 3.1.0*(2013), Retrieved May 31, 2019, from https://www.wireshark.org/download/docs/developer-guide.pdf.

[25] J. H. Kim and J. K. Lee, “Performance analysis of the CSMA / CA protocol using stop-and-wait ARQ method in wireless LANs,” *J. KICS,* vol. 21, no. 5, pp. 1208-1220, May 1996.

[26] H. Eun, Y. Kim, C. Jung, and C. Kim, “Adaptive sampling of initial cluster centers for simple linear iterative clustering,” *J. KICS,* vol. 43, no. 1, pp. 20-23, Jan. 2018.

[27] J. W. Cho, H. R. Cheon, W. Lee, J. C. Ahn, S. Jin, and J. H. Kim, “MAC protocol technology trends for UAV networks,” *J. KICS*, vol. 42, no. 6, pp. 1216-1224, Jun. 2017.

[28] H. I. Park, H. J. Park, S. H. Lee, and J. S. Choi, “Design and analysis of multicast based lightweight demand response protocol for energy IoT environment,” *J. KICS*, vol. 43, no. 7, pp. 1163-1175, Jul. 2018.

[29] W. Cui, J. Kannan, and H. J. Wang, “Discoverer: Automatic protocol reverse engineering from network traces,” *USENIX Secur. Symp.*, pp. 1-14, Boston, USA, Aug. 2007.

[30] N. john, S. K. Shukla, and T. C. Clancy, “A survey of automatic protocol reverse engineering tools,” *ACM Computing Surveys(CSUR)*, vol. 48, no. 3, Article 40, Feb. 2016.

[31] B. D. Sija, Y. H. Goo, K. S. Shim, H. Hasanova, and M. S. Kim, “A survey of automatic protocol reverse engineering approaches, methods, and tools on the inputs and outputs view,” *Secur. and Commun. Netw.*, vol. 2018, Article ID 8370341, Feb. 2018.

정 택 현 (Tack-hyun Jung)

2018년~현재 : 건국대학교 IT융합정보보호학과 석사과정

<관심분야> 통신공학, 사이버 보안, 기계학습, 소프트웨어 공학

[ORCID:0000-0002-9172-0817]

김 기 천 (Kee-cheon Kim)

1992년 : Northwestern Univ. 공학박사

1998년~현재 : 건국대학교 컴퓨터공학과 교수

<관심분야> 통신공학, 사이버 보안, 미래인터넷, IoT

[ORCID:0000-0003-3445-3334]

송 원 종 (Won-jong Song)

2019년~현재 : 건국대학교 IT융합정보보호학과 석사과정

<관심분야> 통신공학, 사이버 보안, 기계학습, 소프트웨어 공학

[ORCID:0000-0002-5278-3751]