https://doi.org/10.7840/kics.2024.49.4.556

다중 프로토콜 분석을 위한 프로토콜 구조 및 시퀀스 탐색 방법

조 현 우•, 박 지 환*, 채 명 호**, 이 해 영***, 임 완 수°

Protocol Structure and Sequence Detection Method for Multi-Protocol Analysis

Hyunwoo Cho•, Jihwan Park*, Myoungho Chae**, Haeyoung Lee***, Wansu Lim°

요 약

본 논문에서는 악성 코드 및 해킹 시스템 등의 사이버 공격을 방지하기 위한 프로토콜 역공학에서 다중 프로토콜 분석을 위한 프로토콜 구조 및 시퀀스 탐색 방법을 제안한다. 다중 프토토콜 데이터는 2개 이상의 정의되지 않은 프로토콜로 이루어진 데이터로 프로토콜의 분류가 필요하다. 본 연구에서는 다중 프로토콜 분류를 위해 계층적 군집화 방법을 사용하고 프로토콜 구조 및 시퀀스 탐색 알고리즘의 성능 및 계산복잡도의 향상을 위해 프로토콜 메시지의 payload를 제거하는 과정을 수행한다. 프로토콜 구조 및 시퀀스 탐색 알고리즘으로 sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘과 CSP (Contiguous Sequential Pattern) 알고리즘을 사용하여 제안하는 방법에 대한 성능평가를 수행하였다. Sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘과 CSP 알고리즘 모두에서 계층적 군집화와 payload를 제거하는 과정을 모두 수행하면 향상된 결과를 얻을 수 있었다.

키워드 : 프로토콜 역공학, 프로토콜, 시퀀스 탐색, 연속적인 시퀀스 패턴(CSP) 알고리즘, 사이버 공격

Key Words : protocol reverse engineering, protocol, sequence detection, contiguous sequence pattern (CSP) algorithm, cyberattack

ABSTRACT

This paper introduces a protocol structure and sequence detection method designed for multi-protocol analysis in the field of protocol reverse engineering, with the aim of mitigating cyber threats such as malware and system hacking. Multi-protocol data, involving two or more undefined protocols, requires effective protocol classification. To address this, our study employs a hierarchical clustering method for multi-protocol classification, enhancing the performance and reducing the computational complexity of the protocol structure and sequence detection algorithm by removing payload of messages. The proposed method is evaluated using both a frequent sequence detection algorithm with a sliding window and a Contiguous Sequential Pattern (CSP) algorithm for protocol structure and sequence detection. Results demonstrate that the inclusion of hierarchical clustering and payload removal in both the frequent sequence detection algorithm and the CSP algorithm leads to notable performance enhancements.

※ 본 연구는 방위사업청의 재원으로 국방과학연구소의 지원을 받아 수행된 연구임 (311JJ5-912967201).
• First Author : Kumoh National Institute of Technology, Department of Electronic Engineering, jm00020@kumoh.ac.kr, 학생회원
° Corresponding Author : Sungkyunkwan University, Electrical and Computer Engineering, wansu.lim@skku.edu, 정회원
* Kumoh National Institute of Technology, Department of Computer Software Engineering, ehfdkdhswlgh@kumoh.ac.kr, 학생회원
** Agency for Defense Development, mhchae4940@naver.com
*** School of Physics, Engineering and Computer Science, University of Hertfordshire, h.lee@erts.ac.uk
논문번호 : 202310-120-C-RN, Received October 30, 2023; Revised January 10, 2024; Accepted January 10, 2024

Ⅰ. 서 론

최근 4차 산업혁명과 AI 기술의 발전 등으로 인터넷 및 네트워크 시스템의 중요성 역시 증가하고 있다. 이러한 기술의 발전으로 많은 정보를 인터넷 및 네트워크 시스템에서 공유하기 때문에 네트워크 관리 및 보안은 매우 중요한 부분이다. 하지만 악성 코드 및 해킹 시스템과 알려지지 않은 비공개 프로토콜을 사용한 사이버 공격이 꾸준히 발생하고 있으며 비공개 프로토콜은 어떠한 구조로 구성되어 있는지를 알 수 없으므로 프로토콜 안의 메시지가 어떠한 내용인지를 알 수 없어 사이버 공격을 방지하기가 매우 어렵다. 따라서 비공개 프로토콜을 이용한 사이버 공격을 방지하기 위해 알지 못하는 프로토콜을 분석하는 프로토콜 역공학 연구가 활발히 이루어지고 있다[1-3].

프로토콜 역공학은 알 수 없거나 문서화 등으로 정의되지 않은 네트워크 프로토콜의 구조 및 의미를 추출하는 것을 목표로 하는 연구이다. 즉 프로토콜에 어떤 유형의 메시지가 존재하는지, 메시지는 어떤 형식으로 구성되어 있는지 등에 대한 것을 추출하는 것이다. 전통적인 프로토콜 역공학은 사람이 직접 분석하는 방법을 사용하였다. 사람이 직접 수행하면 시간이 매우 오래 걸리고 분석자의 능력에 따라 결과가 다르게 나오는 결과가 발생한다. 따라서 이러한 문제를 해결하기 위해 프로토콜 역공학의 자동화에 관한 연구가 이루어지고 있다[4,5]. 자동 프로토콜 역공학 방법은 실행 트레이스 기반 분석과 네트워크 트레이스 기반 분석으로 분류된다. 실행 트레이스 기반 분석을 사용한 자동 프로토콜 역공학 방법으로는 ICSREF (Industrial Control Systems Reverse Engineering Framework), BITY가 있다[6,7]. ICSRER는 산업용 바이너리를 위한 역공학 프레임워크로 사전 지식 데이터베이스를 구축하여 코드시스 바이너리 파일에 대하여 자동화된 분석을 수행한다[6]. BITY는 support vector machine을 이용하여 바이너리의 타입을 분류한다[7]. 그리고 네트워크 트레이스 기반 분석을 사용한 자동 프로토콜 역공학 방법은 NetPlier, CSP (Contiguous Sequential Pattern)가 있다[8,9]. NetPlier는 랜덤 변수와 관측된 메시지의 joint distribution을 이용하여 메시지의 Clustering을 수행하고 메시지 형식과 state machine을 추정한다[8]. CSP는 일반적인 시퀀스 탐색 알고리즘과 달리 연속적인 시퀀스만 고려하는 알고리즘이며 반복적인 CSP를 이용하여 추출한 field의 속성을 정의하여 프로토콜의 구조를 확인할 수 있다[9]. 본 연구에서는 프로토콜의 구조 및 시퀀스 탐색을 수행하는 알고리즘 중 하나로 CSP를 사용한다. 실행 트레이스 기반 분석 방법은 타겟 프로토콜을 사용하는 프로그램의 실행 트레이스를 사용하여 분석하는 방법이다. 실행 트레이스 기반 분석 방법은 타겟 프로토콜을 사용하는 프로그램을 사용할 수 있는 상황에서만 활용할 수 있으므로 일반적인 상황에서는 적합하지 않다. 따라서 네트워크 트래픽에서 타겟 프로토콜을 캡처하여 사용하는 네트워크 트레이스 기반 분석 방법을 자동 프로토콜 역공학에서 주로 사용한다.

이에, 본 연구는 네트워크 트레이스 기반 분석 방법을 사용하며 2개 이상의 정의되지 않은 프로토콜이 존재하는 환경을 고려하여 계층적 군집화를 통하여 정의되지 않은 프로토콜의 분류를 수행한다. 그 후 프로토콜 구조 및 시퀀스 탐색의 성능 및 수행 시간 감소를 위한 payload 제거 과정을 수행한다. 이러한 전처리 과정 후 프로토콜 구조 및 시퀀스 탐색 알고리즘을 이용한 프로토콜을 분석하게 된다.

Ⅱ. 다중 프로토콜 구조 및 시퀀스 탐색 방법

그림 1은 본 연구에서 제안하는 정의되지 않은 프로토콜의 구조 및 시퀀스 탐색 방법을 보여준다. 그림 1에서 볼 수 있듯이 제안하는 방법은 3개의 과정을 통하여 프로토콜의 구조 및 시퀀스를 탐색한다. 기존 프로토콜 역공학에서는 정의되지 않은 프로토콜이 하나만 존재하는 경우를 가정하여 이루어졌다[8,9,19]. 따라서 2개 이상의 정의되지 않은 프로토콜이 존재할 때는 프로토콜의 구조 및 시퀀스를 정확히 찾는 것에 어려움이 있다. 본 연구에서는 이러한 문제점을 해결하기 위하여 계층적 군집화를 이용하여 2개 이상의 정의되지 않은 프로토콜에 대한 사전 분류를 수행하는 방식으로 해결한다. 본 연구에서는 인터넷 계층과 전송 계층의 프로토콜만을 고려하며 그중 ARP (Address Resolution Protocol), ICMP (Internet Control Message Protocol), TCP (Transmission Control Protocol)를 사용한다. 프로토콜 메시지는 header 부분과 payload 부분으로 나누어진다. 인터넷

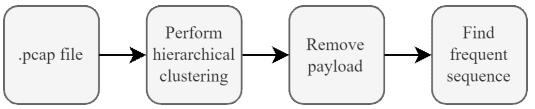

그림 1. 다중 프로토콜 구조 및 시퀀스 탐색 방법
Fig. 1. Multi-protocol structure and sequence detection method

및 전송 계층 프로토콜의 메시지에서 payload는 상위 계층 메시지나 랜덤한 데이터를 포함하여 메시지의 header 부분을 분석하는 것에 방해가 된다. 따라서 본 연구에서는 메시지 일부를 이용한 빈도율 계산을 통해 payload를 제거한다. 이를 통해 프로토콜 분석에 필요한 시간 및 성능을 향상할 수 있었다. 정의되지 않은 프로토콜의 구조 및 시퀀스를 찾기 위해 본 연구에서는 sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘과 CSP 알고리즘을 사용하여 ARP, ICMP, TCP로 이루어진 다중 프로토콜 데이터에 대한 성능을 비교한다.

2.1 계층적 군집화

군집화에는 K-means 알고리즘, mean shift 알고리즘, GMM (Gaussian Mixture Model) 알고리즘, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) 알고리즘, 계층적 군집화 알고리즘 등의 여러 가지 방법의 군집화 방법이 있다. K-means 알고리즘과 같은 군집화 알고리즘은 군집의 중심점과의 거리에 따라 군집화를 수행하고 mean shift 알고리즘과 DBSCAN 알고리즘은 밀도를 기준으로 군집화를 수행한다[10-12]. GMM 알고리즘은 데이터가 다양한 Gaussian 분포를 따른다는 가정으로 데이터가 어느 분포에 따르는지를 이용한 군집화 방법이고 계층적 군집화는 각 데이터를 계층적으로 유사한 그룹과 통합하여 군집화를 수행하는 방법으로 사전에 군집의 수를 정하지 않아도 수행할 수 있다[13,14]. 정의되지 않은 프로토콜은 랜덤한 값이 포함되어 있어 밀도를 이용한 군집화 방법을 사용했을 때 정확한 결과를 얻을 수 없고 정의되지 않은 프로토콜이 2개 이상 있는 경우에는 정확한 군집의 수를 사전에 결정할 수 없다. 따라서 본 연구에서는 계층적 군집화를 이용하여 최적의 군집 수를 결정하고 군집화를 수행하였다.

그림 2는 본 연구에서는 제안하는 정의되지 않은 프로토콜에 대한 계층적 군집화 방법을 보여준다. 프로토콜의 메시지 길이는 header와 payload를 합친 길이로 프로토콜마다 메시지 길이는 다르다. 그러나 군집화를 수행하기 위해서는 모든 메시지 길이가 같아야 하므로 모든 메시지의 길이를 통일하는 과정이 필요하다 메시지 길이를 통일하는 방법으로는 최대 길이로 통일하는 방법, 최소 길이로 통일하는 방법, 평균 길이로 통일하는 방법이 있다. 메시지 길이가 통일할 길이보다 짧다면 부족한 byte만큼 0으로 채우고 길다면 긴 부분을 잘라내어 메시지 길이를 통일한다. 이

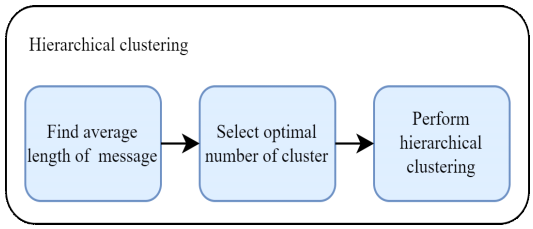

그림 2. 계층적 군집화
Fig. 2. Hierarchical clustering

는 군집화를 수행하는 과정에서만 적용되며 이후 과정에서는 길이가 통일되지 않은 데이터를 사용한다. 표 1은 각각의 방법으로 메시지 길이를 통일하였을 경우의 군집화를 위한 지표인 silhouette 지표와 군집화 정확도를 보여준다. ARP, ICMP, TCP를 각각 1000개씩 사용하여 군집화를 수행한 결과이다. 최대 길이를 사용하는 경우에서 가장 높은 Silhouette 지표를 보여주었고 길이가 짧아질수록 silhouette 지표가 낮아지는 것을 확인할 수 있었으며 군집화 정확도는 큰 차이가 없는 것도 확인할 수 있었다. 하지만 길이가 짧아질수록 silhouette 지표가 낮아지는 것으로 보아 많은 프로토콜의 분류 및 비슷한 시퀀스를 갖는 프로토콜의 군집화에서는 짧은 길이는 불리할 수 있다. 하지만 최대 길이의 경우에는 계산의 복잡도가 증가하고 매우 긴 길이의 프로토콜과 짧은 길이의 프로토콜 여러 개의 군집화를 수행할 경우, 최대 길이를 이용한 군집화는 불필요한 많은 연산을 발생시킬 수 있다. 따라서 본 연구에서는 최소 길이나 최대 길이가 아닌 평균 길이로 통일하는 방법을 사용하여 군집화를 수행하였으며 계층적 군집화에서는 silhouette 지표를 활용하여 최적의 군집 수를 결정하였고 유사도 측정은 유클리드 거리를 사용하였다. (1)은 silhouette 지표를 구하는 식으로 $s(i)$는 i번째 데이터의 silhouette 계수의 값이고 $a(i)$와 $b(i)$는 각각 i번째 데이터와 같은 군집의 다른 데이터와의 평균 거리, 가장 가까운 군집과의 평균 거리를 의미한다. $s(i)$의 값은 -1에서 1의 값을 가지며 1에 가까울수록 근처 군집과 멀리

표 1. 메시지의 길이 통일 방법에 따른 silhouette 지표와 군집화 정확도
Table 1. Silhouette score and clustering accuracy based on message length normalization method

Method	Silhouette score	Clustering accuracy
Min	0.476	1.0
Average	0.480	1.0
Max	0.537	1.0

떨어져 있음을 의미하고 0에 가까울수록 근처 군집과 가까움을 의미한다. 만약 음의 값을 가진다면 잘못된 군집에 할당되었음을 의미한다[15]. silhouette 지표에서 데이터 간의 거리는 (2)를 이용하여 계산하였다. $d(\boldsymbol{p},\boldsymbol{q})$는 유클리드 거리로 $\boldsymbol{p}=(p_1,p_2,\cdots,p_n)$와 $\boldsymbol{q}=(q_1,q_2,\cdots,q_n)$의 거리를 나타낸다.

$$s(i)=\frac{b(i)-a(i)}{\max(a(i),b(i))} \tag{1}$$

$$d(\boldsymbol{p},\boldsymbol{q})=\sqrt{\sum_{i=1}^{n}(q_n-p_n)^2} \tag{2}$$

알고리즘 1은 최적의 군집 수를 결정하는 알고리즘이다. 군집화할 데이터 포인트의 리스트를 입력으로 받아 최적의 군집 수를 반환한다. 알고리즘 1. 라인 1, 2에서 silhouette 지표를 계산할 군집 수의 범위를 정하고 silhouette 점수를 저장할 $silhouette$를 생성한다. 알고리즘 1. 라인 4-7에서는 병합 군집을 수행하여 군집의 수인 k에 따른 데이터 리스트에 군집 라벨을 지정하고 silhouette 지표를 계산하여 $silhouette$에 저장한다. 알고리즘 1. 라인 9, 10에서는 $silhouette$에서 최댓값을 갖는 인덱스를 찾고 그 인덱스에 해당하는 k 값을 최적의 군집 수로 결정한다.

Algorithm 1: Select Optimal Number of Clusters

Input : $dataList$ – a list of data points to cluster
Output : $k_{optimal}$ – the optimal number of cluster based on sihouette score

```
1  range_k ← list from 2 to 10
2  silhouette ← empty list
3  for each k in range_k do
4      clustering = Agglomerative Clustering(k)
5      clusterLabel = clustering.fit_predict(dataList)
6      score = Silhouette Score(dataList, clusterLabel)
7      silhouette.append(score)
8  end for
9  index = argmax(silhouette)
10 k_optimal = range_k[index]
11 return k_optimal
```

알고리즘 1. 최적의 군집 수 선택
Algorithm 1. Select optimal number of clusters

알고리즘 1을 통해 구한 최적의 군집 수로 프로토콜의 군집화를 수행하여 프로토콜을 분류한다.

2.2 Payload 제거

메시지는 header 부분과 payload 부분으로 나누어진다. 인터넷 및 전송 계층 프로토콜의 메시지에서 payload는 상위계층 메시지나 랜덤한 데이터를 포함하여 header 부분의 정확한 분석을 방해하고 분석에 필요한 시간을 증가시킨다. 따라서 기존 프로토콜 역공학에서는 payload가 없다고 가정하거나 payload 길이를 1로 고정하는 방법 등을 이용하였다. 하지만 실제 메시지는 payload가 없거나 payload 길이가 1일 수 없으므로 본 연구에서는 프로토콜 일부분을 사용한 빈도율 계산을 통해 payload를 제거하는 알고리즘을 제안한다.

그림 3은 payload를 제거하는 방법을 보여준다. 계층적 군집화로 프로토콜을 분류하였기 때문에 군집에서 일부분만을 사용하여 payload 영역을 식별한다. 본 연구에서는 각 군집에서 10%의 메시지를 사용하였다. Payload는 랜덤한 값을 가지고 있으므로 빈번한 시퀀스가 추출되지 않은 영역을 payload로 간주한다. 따라서 payload 영역을 식별하는 부분에 있어 빈번한 시퀀스의 정확도는 크게 중요한 결과가 아니며 모든 메시지를 사용한다면 빈번한 시퀀스 탐색 알고리즘 수행 시간이 증가하기 때문에 본 연구에서는 10%의 메시지만 사용하여 payload 영역을 식별한다. 이때 10%만을 사용한 이유는 다양한 비율로 payload 영역을 식별하였을 때 수행 시간 및 payload 영역 식별 정확도에 있어 최적이었기에 사용하였다.

알고리즘 2는 payload 영역을 식별하는 알고리즘으로 10%의 메시지을 입력으로 받아 payload의 시작 인덱스를 반환한다. 알고리즘 2. 라인 2에서 빈번한 시퀀스 탐색 알고리즘을 이용하여 빈번한 시퀀스를 추출한다. 알고리즘 2에서 사용한 빈번한 시퀀스 탐색 알고리즘은 2.3절에서 자세히 설명한다. 알고리즘 2. 라인 4-7에서는 추출한 빈번한 시퀀스의 시작 인덱스와 길이 정보를 이용하여 시퀀스가 끝나는 지점을

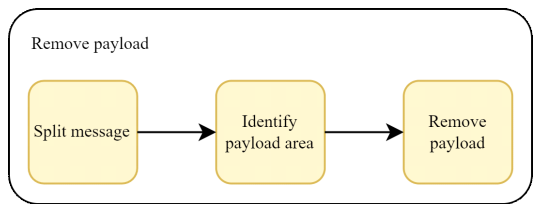

그림 3. Payload 제거 과정
Fig. 3. Process of payload removal

Algorithm 2: Identify Payload Area

Input : $hexList_{10\%}$ – a list of 10% of hexadecimal messages

Output : $index_{payload}$ – the starting index for payload

```
1  index_payload = 0
2  frequent sequence
       = find frequent sequence(hexList_10%)
3  for each s in frequent sequence do
4      index_temp = s.start + s.length
5      if index_temp ≥ index_payload then
6          index_payload = index_temp
7      end if
8  end for
9  return index_payload
```

알고리즘 2. payload 영역 식별
Algorithm 2. Identify payload area

$index_{temp}$에 저장하고 이를 $index_{payload}$와 비교하여 $index_{temp}$가 큰 경우에만 $index_{payload}$를 갱신한다. Payload는 항상 랜덤한 값을 가지기 때문에 빈번한 시퀀스 탐색 알고리즘에서 추출되지 않는다. 따라서 추출된 빈번한 시퀀스의 가장 마지막 인덱스 이후 부분을 payload 영역이라 할 수 있어 빈번한 시퀀스의 가장 마지막 인덱스를 저장하고 있는 $index_{payload}$가 payload의 시작점을 의미한다. 알고리즘 2를 통해 구한 payload 시작 인덱스를 이용하여 메시지의 payload 영역을 제거한다. 그림 4는 TCP에서 payload 영역 식별을 시각적으로 나타낸 것이다. 메시지에서 추출한 빈번한 시퀀스는 다양한 색으로 나타내었고 흰 부분

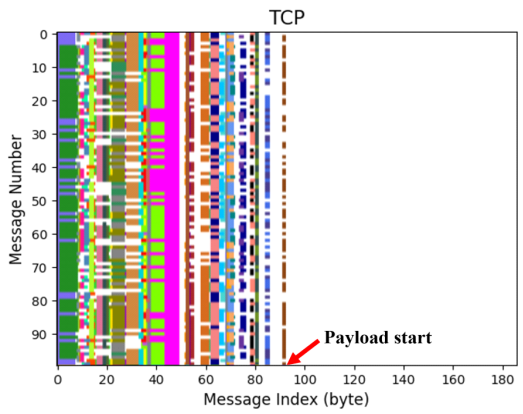

그림 4. TCP의 payload 영역 식별 결과
Fig. 4. Result of identifying payload area of TCP

표 2. 프로토콜의 payload 시작점과 예상 시작점
Table 2. Payload index and payload prediction index for protocols

Protocol	payload index (byte)	payload prediction index (byte)
ARP	X	X
ICMP	42	41
TCP	86	94

은 payload 영역이다. 따라서 빈번한 시퀀스와 흰 부분의 경계가 payload의 시작점이 된다. Y축인 Message Number은 메시지의 구분을 위해 번호로 5인 경우, 5번째 메시지를 의미하고 X축인 Message Index는 시퀀스의 위치를 나타낸다. 표 2는 본 연구에서 사용하는 프로토콜인 ARP, ICMP, TCP의 실제 payload 영역의 시작점과 제안하는 알고리즘으로 구한 payload 영역의 시작점을 보여준다. 표 2의 결과를 위해 프로토콜당 1000개의 메시지를 사용하였으며 payload의 길이는 100byte로 고정하였다. Payload가 없는 ARP는 payload가 없다고 예상하였고 ICMP는 거의 정확히 예측하였으며 TCP의 경우에는 약 8.5%의 오차가 발생하였다.

2.3 프로토콜 구조 및 시퀀스 탐색

프로토콜 구조 및 시퀀스를 탐색하는 알고리즘으로는 특정 데이터가 발생하는 빈도를 이용한 Apriori 알고리즘[16], 다중 문자열 패턴을 매칭하는 Aho-Corasick 알고리즘[17], 순차적인 패턴을 분석하는 GSP (Generalized Sequential Pattern) 알고리즘[18], 순차적이면서 연속적인 패턴을 분석하는 CSP 알고리즘[9] 등이 있다. 본 연구에서는 sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘과 CSP 알고리즘을 사용하였다. Sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘은 정해진 길이의 시퀀스만을 추출하는 기존의 빈번한 시퀀스 알고리즘과 달리 다양한 길이의 시퀀스를 추출할 수 있도록 개선한 알고리즘이다. CSP 알고리즘은 GSP 알고리즘과 달리 연속적인 시퀀스만을 고려하는 알고리즘이며 Recursive CSP 알고리즘을 이용하여 각 시퀀스의 field type을 결정하는 알고리즘으로 프로토콜의 구조를 확인할 수 있다.

2.3.1 Sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘

알고리즘 3은 sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘으로 16진수로 이루어진 프로토콜

Algorithm 3: Find Frequent Sequences Algorithm with Sliding Window

Input : $hexList$, Min_{freq}, $windowSize$

Output : $frequent\ sequence$

```
1  sequence = ∅
2  for each hexstream in hexList do
3      for each i in
         0 to (hexstream.Len − windowSize) do
4      sequence = sequence ∪
           hexstram[i : i+windowSize]
5      end for
6  end for
7  frequent sequence = ∅
8  for each seq in sequence do
9      if seq ∉ frequent sequnces then
10     Calculate frequency percentage
                   percentage_freq of seq
11         if Min_freq ≤ percentage_freq then
12     frequent sequence =
           frequent sequence ∪ seq
13         end if
14         for each sub seq in seq do
15     Calculate frequency percentage
           percentage_freq of sub seq
16             if Min_freq ≤ percentage_freq
                                   then
17     frequent sequence =
       frequent sequence ∪ sub seq
18             end if
19         end for
20     end if
21 end for
22 return frequent sequence
```

알고리즘 3. Sliding window를 사용한 빈번한 시퀀스 탐색
Algorithm 3. Find frequent sequences algorithm with sliding window

패킷과 최소 빈번도, window의 크기를 입력으로 받아 빈번한 시퀀스를 출력한다. 알고리즘 3. 라인 2-6은 프로토콜 패킷에서 window의 크기만큼씩 잘라 $sequence$를 생성한다. 알고리즘 3. 라인 9-13에서 seq가 $frequent\ sequence$에 포함되어 있지 않으면 seq의 빈번도를 계산하여 최소 빈번도보다 크다면

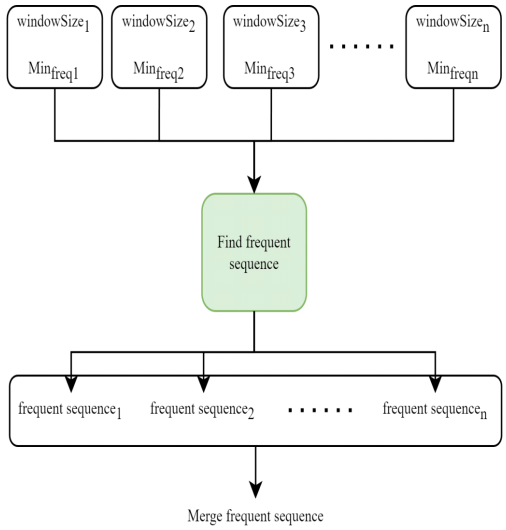

그림 5. 다양한 window 크기와 최소 빈번도를 사용한 빈번한 시퀀스 탐색 알고리즘
Fig. 5. Frequent sequence detection algorithm with different window sizes and minimum frequencies

seq를 $frequent\ sequence$에 추가한다. 알고리즘 3. 라인 14-19에서는 seq를 구성하는 $sub\ seq$에 대한 빈번도를 계산하여 $frequent\ sequence$에 추가한다.

그림 5는 본 연구에서 sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘을 사용한 방법이다. window 크기와 최소 빈번도를 다양하게 변경하여 다양한 시퀀스의 결과를 얻을 수 있다. 다양한 시퀀스의 중복을 제거하는 등의 합병 과정을 거쳐 최종적으로 빈번한 시퀀스를 추출한다. 그림 5와 같은 방법을 사용함으로써 기존의 빈번한 시퀀스 탐색 알고리즘보다 다양한 길이의 시퀀스를 추출할 수 있다.

2.3.2 CSP 알고리즘

CSP 알고리즘은 연속적인 시퀀스를 추출하는 알고리즘으로써 CSP를 반복적으로 사용하여 추출한 시퀀스의 field type을 결정하고 추출한 field 정보를 이용하여 프로토콜의 구조를 추정하는 알고리즘이다[9]. 그림 6은 CSP 알고리즘의 동작 과정을 보여준다. 첫 번째 CSP를 이용하여 field format을 추출한다. 첫 번째 CSP로 추출한 field format은 값과 길이가 변하지 않는 field type인 SF(v) (Static Field(v))만으로 구성된다. 그 후 반복적인 CSP를 이용하여 먼저 추출한 SF(v)에서 값이나 길이가 변하지만 예측 가능한 field type인 DF(v) (Dynamic Field(v))를 추출한다. 두 번째 CSP는 SF(v)와 DF(v)로 구성된 field format을 이용하여 message format을 추출한다. 마지막으로 추출

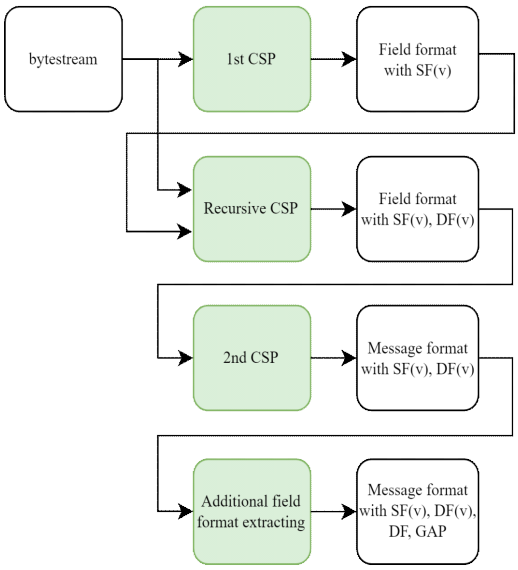

그림 6. CSP 알고리즘 동작 과정
Fig. 6. Process of CSP algorithm

된 message format의 빈 field는 길이만 예측 가능한 field type인 DF (Dynamic Field)와 값과 길이 모두 예상할 수 없는 field type인 GAP으로 채우는 과정을 수행한다. CSP 알고리즘을 수행하면 프로토콜의 field 단위로 나눌 수 있으며 각 field의 값이나 길이가 변경될 수 있는 field인지 고정된 field인지를 알 수 있다. 이러한 field type의 특징을 활용한다면 각 field의 의미를 파악하는 것에 큰 이점이 된다.

Ⅲ. 성능평가

성능평가에 사용할 데이터는 ARP, ICMP, TCP로 이루어진 다중 프로토콜 데이터이다. 다중 프로토콜 데이터는 Python 라이브러리인 Scapy를 사용하여 각각 생성한 1000개의 패킷을 pcap 파일로 저장하였다. ARP, ICMP는 이더넷 header를 사용하였고, TCP는 WLAN header를 사용하였으며 특정한 서비스를 고려하여 생성한 것이 아닌 단순히 임의의 데이터를 전송하는 서비스를 사용하였으며 TCP와 같이 가변적인 프로토콜 메시지의 길이는 고정하여 사용하였다. 또한 ARP는 payload 부분이 없으며 ICMP, TCP는 payload 부분이 100 bytes로 이루어져 있다. 다중 프로토콜 데이터에 계층적 군집화와 payload 제거 모두 하지 않은 경우 (Case 1), 계층적 군집화만 수행하고 payload 제거하지 않은 경우 (Case 2), 계층적 군집화, payload 제거 모두 수행하는 경우 (Case 3)로 나누어서 프로토콜 구조 및 시퀀스 탐색 알고리즘에 적용하였다.

3.1 Case 1

Case 1은 다중 프로토콜 데이터에 계층적 군집화와 payload 제거를 모두 하지 않은 경우이다. 그림 7은 sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘을 사용한 결과를 보여준다. 그림 7의 색칠된 부분은 추출된 시퀀스를 나타내고 흰 부분은 시퀀스를 추출하지 못한 부분이다. 그림 7에서 볼 수 있듯이 sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘은 많은 부분의 시퀀스를 추출하지 못 하였고 CSP 알고리즘의 경우는 빈번한 시퀀스를 전혀 추출하지 못하였다. 두 알고리즘 모두 시퀀스 및 프로토콜의 구조를 대부분 추출하지 못하였다. 이는 프로토콜마다 동일한 구조나 시퀀스가 거의 없기 때문이다. 이러한 결과를 통하여 다중 프로토콜 데이터에서는 프로토콜을 분류하는 계층적 군집화가 필요하다는 것을 확인할 수 있었다.

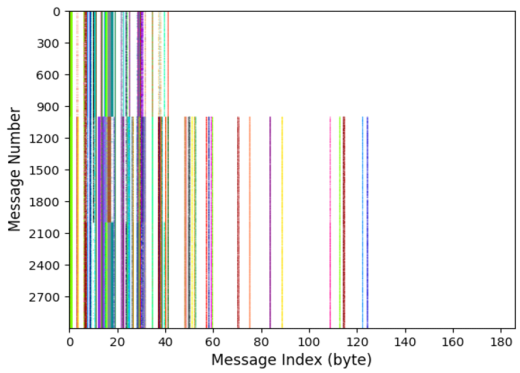

그림 7. Sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘 수행 결과 (Case 1)
Fig. 7. Result of frequent sequence detection algorithm with sliding window (Case 1)

3.2 Case 2

Case 2는 다중 프로토콜 데이터를 계층적 군집화를 통해 분류하고 payload는 제거하지 않은 경우이다. 그림 8은 sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘을 수행한 결과이다. 패킷에 따라 추출된 시퀀스는 다양한 색상으로 표현하였고 흰 영역은 추출하지 못한 부분이다. Sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘은 header뿐만 아니라 payload에서도 시퀀스를 추출한 것을 확인할 수 있다. 이는 많은 시퀀스를 찾아낸 것으로 보이지만 header

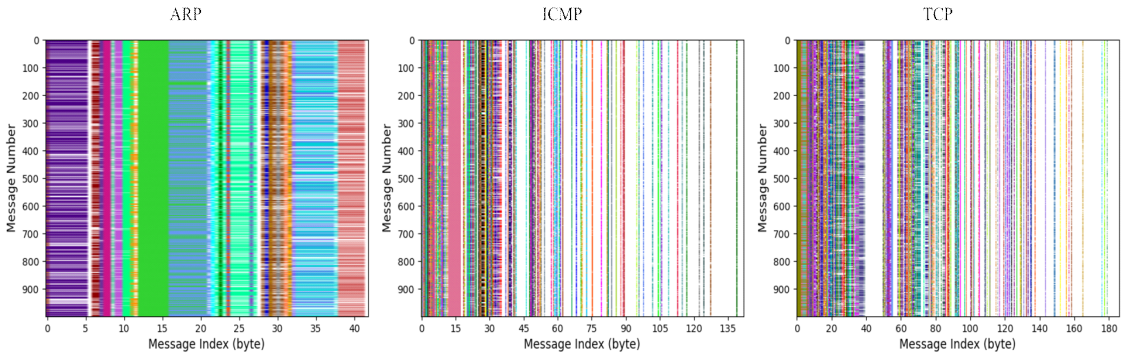

그림 8. Sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘 실행 결과 (Case 2)
Fig. 8. Result of frequent sequence detection algorithm with sliding window (Case 2)

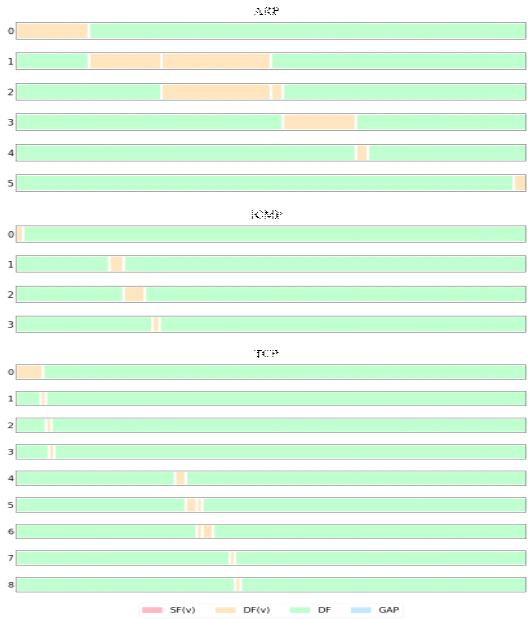

그림 9. CSP 알고리즘 수행 결과 (Case 2)
Fig. 9. Result of CSP algorithm (Case 2)

를 분석하는 것에 있어 불필요한 시퀀스를 추출한 것이다. 그림 9는 CSP 알고리즘을 사용한 결과이다. Payload가 없는 ARP는 6가지의 message format을 통하여 비교적 다양한 시퀀스를 찾을 수 있었다. 하지만 payload 부분이 존재하는 ICMP와 TCP에서는 DF가 많이 추출되고 추출된 시퀀스의 길이도 짧고 적은 모습을 확인할 수 있다.

3.3 Case 3

Case 3은 계층적 군집화와 payload 제거 모두 수행한 경우로 본 연구에서 제안하는 방법이다. 그림 10에서 볼 수 있듯이 sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘은 payload가 제거됨으로써 payload 부분에서 추출되는 불필요한 시퀀스 추출을 방지할 수 있게 되었다. 그림 11과 같이 CSP 알고리즘은 payload를 제거하기 전의 결과보다 향상된 결과를 얻을 수 있었다.

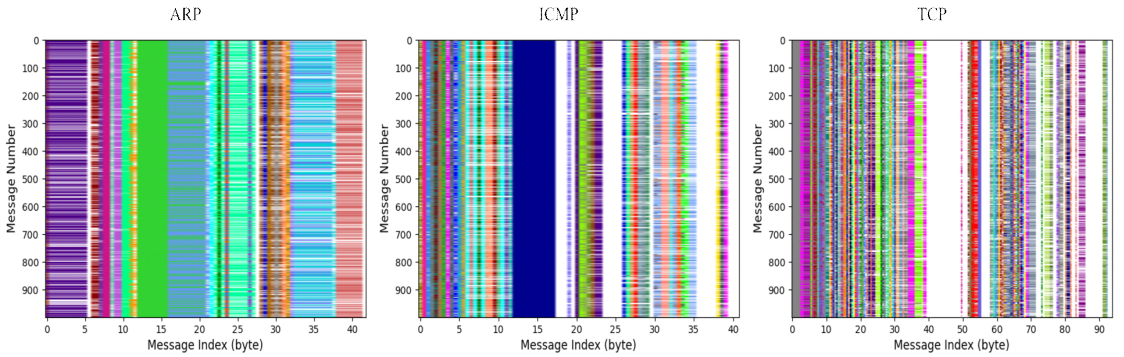

그림 10. Sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘 실행 결과 (Case 3)
Fig. 10. Result of frequent sequence detection algorithm with sliding window (Case 3)

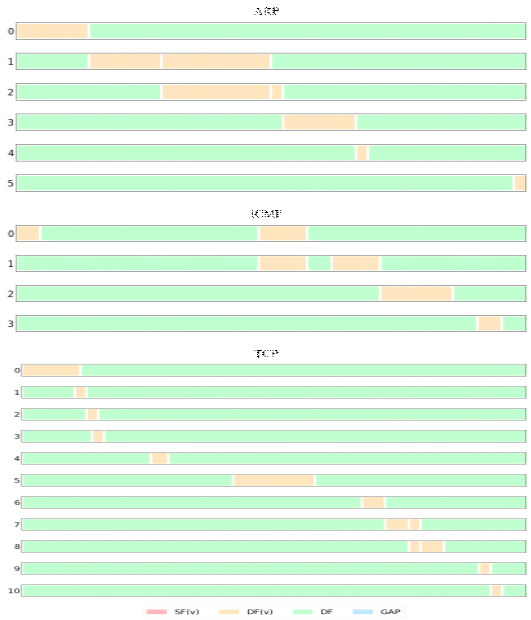

그림 11. CSP 알고리즘 수행 결과 (Case 3)
Fig. 11. Result of CSP algorithm (Case 3)

3.4 시퀀스 추출률 비교

표 3은 각 Case에서 두 알고리즘의 시퀀스 추출률을 보여준다. 시퀀스 추출률은 단순히 추출된 시퀀스의 길이를 전체 프로토콜의 길이로 나눈 값이다. Case 1에서 시퀀스 추출률을 계산할 때 시퀀스 길이가 1인 경우는 제외하였다. 연속적인 시퀀스를 발견하지 못하여 대부분 길이가 1인 시퀀스를 발견하기 때문이다. Case 2, 3은 계층적 군집화를 통하여 프로토콜을 분류하였기 때문에 시퀀스의 길이가 1보다 큰 경우가 대부분이며 field의 크기가 1인 경우도 존재함으로 Case 2, 3에서는 길이가 1인 시퀀스도 추출률 계산에 포함하였다. 표 3의 Case 2에서 괄호 안의 숫자는 payload를 제외하고 추출률을 계산한 결과이다. Sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘은 Case 1에서 거의 시퀀스를 추출하지 못하였고 Case 2에서는 CSP 알고리즘보다 많은 시퀀스를 추출하였지만 payload에서 불필요한 시퀀스를 추출하였기 때문에 높은 추출률을 보여주었다. Case 3에서는 Case 2보다 향상된 결과를 보여주었지만 CSP 알고리즘보다는 낮은 추출률을 보여주었다. CSP 알고리즘은 Case 1에서 시퀀스를 추출하지 못하였지만 payload가 없는 ARP의 경우는 Case 2와 3 모두에서 약 71%의 시퀀스를 추출하였고 ICMP와 TCP도 Case 2와 3에서 추출률의 향상이 있었다. Case 3의 추출률 향상이 매우 커 보이지만 본 연구에서는 payload의 길이를 100byte로 설정하였기에 빈번한 시퀀스가 추출되는 header가 payload보다 짧아 Case2에서 비교적 낮은 추출률을 보여준다.

표 3. 프로토콜 구조 및 시퀀스 탐색 알고리즘에 따른 시퀀스 추출률
Table 3. Sequence detection rate based on protocol structure and sequence detection algorithm

		With sliding window	CSP algorithm [9]
Case 1	ARP	0.071	0
	ICMP	0.060	0
	TCP	0.041	0
Case 2	ARP	0.417	0.714
	ICMP	0.135 (0.424)	0.098 (0.333)
	TCP	0.221 (0.396)	0.179 (0.388)
Case 3	ARP	0.417	0.714
	ICMP	0.434	0.439
	TCP	0.431	0.536

계층적 군집화와 payload 제거 중 계층적 군집화가 시퀀스 추출률 향상에 더 큰 기여를 보여준다. 다양한 프로토콜이 섞인 데이터에서는 프로토콜에 따라 메시지의 구조가 다르므로 빈번한 시퀀스를 쉽게 찾을 수 없다. 따라서 계층적 군집화를 통해 프로토콜을 분류하면 시퀀스 추출률이 크게 향상된다. 반면 payload 제거는 빈번한 시퀀스가 거의 추출되지 않는 payload를 제거하여 추출률이 향상되며 시퀀스 탐색 알고리즘의 수행 시간 단축에 도움을 준다. 또한 프로토콜에 따라 성능향상의 정도가 다른 이유는 프로토콜마다 메시지의 구조가 다르기 때문이다. 예를 들어, ARP의 경우는 같은 값을 갖는 field가 많지만, ICMP와 TCP는 상황에 따라 다른 값을 가지는 field가 많다. Case 3에서는 CSP 알고리즘이 sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘보다 높은 시퀀스 추출률을 보여주고 Case1과 2에서는 sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘이 CSP 알고리즘보다 높은 시퀀스 추출률을 보여준다. Sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘은 각 메시지마다 후보 시퀀스를 생성하여 빈번도를 계산하며 다양한 window 크기를 사용하여 시퀀스를 추출하고 CSP 알고리즘은 후보 시퀀스를 전체 메시지에 대하여 생성하여 공통된 후보 시퀀스를 사용하여 빈번도를 계산하고 반복 수행을 통하여 시퀀스 추출뿐만 아니라 메시지 구조를 추정한다. Sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘은 메시지마다

후보 시퀀스를 생성하기 때문에 CSP 알고리즘보다 많은 시퀀스를 추출할 수 있지만 payload가 포함된 경우 불필요한 시퀀스를 추출한다.

Ⅳ. 결론 및 향후 연구

본 연구에서는 정의되지 않은 프로토콜이 2개 이상 포함된 다중 프로토콜 데이터에서 프로토콜의 구조 및 시퀀스를 탐색하는 방법을 제안하였다. 제안하는 방법은 하나의 정의되지 않은 프로토콜만 고려하는 기존의 프로토콜 역공학과 달리 다양한 프로토콜을 사전 정보가 없어도 분석할 수 있다. 또한 프로토콜의 payload 영역을 식별하고 제거하는 과정이 포함되어 있어 시퀀스 탐색 알고리즘의 성능을 향상할 수 있다.

다중 프로토콜 데이터에서 프로토콜의 분류를 위하여 계층적 군집화 방법을 사용하였으며 프로토콜의 구조 및 시퀀스 탐색 알고리즘의 성능 및 계산복잡도를 향상하기 위하여 프로토콜의 payload를 제거하는 알고리즘을 제안하였다. 그리고 프로토콜의 구조 및 시퀀스 탐색 알고리즘으로 sliding window를 사용한 빈번한 시퀀스 탐색 알고리즘과 CSP 알고리즘을 사용하여 제안하는 방법에 대한 성능평가를 수행하였다. 다중 프로토콜 데이터에 계층적 군집화와 payload 제거 모두 하지 않은 경우, 시퀀스의 추출이 거의 불가능하였다. 계층적 군집화만 수행하고 payload 제거하지 않은 경우, payload가 없는 ARP는 많은 시퀀스를 추출할 수 있었고 payload가 있는 ICMP와 TCP는 시퀀스 추출이 가능했지만 많은 시퀀스를 추출하지 못하였다. 계층적 군집화, payload 제거 모두 수행하는 경우, payload를 제거하기 전보다 많은 시퀀스를 추출할 수 있었고 프로토콜의 길이가 payload만큼 줄어 알고리즘 실행 시간을 줄일 수 있었다. 이러한 결과를 통하여 제안하는 방법이 다중 프로토콜 분석에 효과적인 것을 확인할 수 있었다.

본 연구를 통하여 추출한 프로토콜의 구조 및 시퀀스의 의미를 탐색하는 연구 및 프로토콜 finite state machine에 대한 연구를 진행할 예정이다. 또한 ARP, ICMP, TCP뿐만 아니라 다른 프로토콜을 사용하여 제안하는 방법에 대한 성능을 확인할 예정이다.

References

[1] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, “Dynamic malware analysis in the modern era—A state of the art survey,” *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1-48, 2019.

[2] A. Moser, C. Krugel, and E. Kirda, “Exploring multiple execution paths for malware analysis,” *IEEE Symp. Secur. and Privacy*, pp. 231-245, 2007.

[3] K. Shin, D. Jung, M.-J. Choe, and H.-M. Cho, “A study protocol reverse engineering research trend and construction of optimal environment for malware analysis,” *J. Korea Inst. Inf. Secur. & Cryptol.*, vol. 32, no. 2, pp. 175-186, 2021.

[4] Y. Huang, H. Shu, F. Kang, and Y. Guang, “Protocol revese-engineering methods and tools: A survey,” *Comput. Commun.*, vol. 182, pp. 238-254, 2022.

[5] B. D. Sija, Y.-H. Goo, K.-S. Shim, H. Hasanova, M.-S. Kim, and Z. Liu, “A survey of automatic protocol reverse engineering approaches methods and tools on the inputs and outputs view,” *Secur. and Commun. Netw.*, vol. 2018, 2018. (https://www.hindawi.com/journals/scn/2018/8370341/)

[6] K. Anastasis and M. Michail, “ICSREF: A framework for automated reverse engineering of industrial control systems binaries,” *arXiv preprint arXiv:1812.03478*, 2018. (http://dx.doi.org/10.14722/ndss.2019.23271)

[7] Z. Xu, C. Wen, and S. Qin, “Learning types of binaries,” *Int. Conf. Formal Eng. Meth.*, pp. 430-446, 2017.

[8] Y. Ye, Z. Zhang, F. Wang, X. Zhang, and D. Xu, “NETPLIER: Probabilistic network protocol reverse engineering from message traces,” *Netw. and Distrib. Syst. Secur. Symp.*, pp. 1-18, 2021.

[9] Y.-H. Goo, K.-S. Shim, M.-S. Lee, and M.-S. Kim, “Protocol specification extraction based on contiguous sequential pattern algorithm,” *IEEE Access*, vol. 7, pp. 36057-36074, 2019. (http://dx.doi.org/10.1109/ACCESS.2019.2905353)

[10] S. Lloyd, “Least squares quantization in PCM,” *IEEE Trans. Inf. Theory*, vol. 28, no.

2, pp. 129-137, 1982.

[11] Y. Cheng, "Mean shitf, mode seeking, and clustering," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 17, no. 8, pp. 790-799, 1995.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *The Second Int. Conf. Knowledge Discovery and Data Mining*, vol. 96, no. 34, pp. 226-231, 1996.

[13] S. P. Chatzis, D. I. Kosmopoulos, and T. A. Varvarigou, "Signal modeling and classification using a robust latent space model based on *t* distributions," *IEEE Trans. Signal Process.*, vol. 53, no. 3, pp 949-963, 2008.

[14] R. Sibson, "SLINK: An optimally efficient algorithm for the single-link cluster method," *The Comput. J.*, vol. 16. no. 1, pp. 30-34, 1973.

[15] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Computational and Appl. Math.*, vol. 20, pp. 53-65, 1987.

[16] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *20th Int. Conf. Very Large Data Bases*, vol. 1215, pp. 487-499, 1994.

[17] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Commun. ACM*, vol. 18, no. 6, pp. 333-340, 1975.

[18] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," *Int. Conf. Extending Database Technol.*, pp. 1-17, 1996.

[19] X. Wang, K. Lv, and B. Li, "IPART: An automatic protocol reverse engineering tool based on global voting expert for industrial protocols," *Int. J. Parallel, Emergent and Distrib. Syst.*, vol. 35, no. 3, pp. 376-395, 2020.

조 현 우 (Hyunwoo Cho)
2021년 3월~현재 : 금오공과대학교 전자공학과 박사과정
<관심분야> 통신 프로토콜, 프로토콜 역공학
[ORCID:0000-0003-4504-0965]

박 지 환 (Jihwan Park)
2018년 3월~현재 : 금오공과대학교 컴퓨터소프트웨어공학과
<관심분야> 통신 프로토콜, 프로토콜 역공학

채 명 호 (Myoungho Chae)
2014년 2월~현재 : 국방과학연구소 선임연구원
<관심분야> 통신 프로토콜, 프로토콜 역공학
[ORCID:0000-0001-7741-1818]

이 해 영 (Haeyoung Lee)
2023년 9월~현재 : University of Hertfordshire, UK
<관심분야> 통신 프로토콜, 무선자원관리
[ORCID:0000-0002-5760-6623]

임 완 수 (Wansu Lim)
2024년 3월~현재 : 성균관대학교 전자전기공학부 교수
<관심분야> 통신 프로토콜, 프로토콜 역공학
[ORCID: 0000-0003-2533-3496]