Journal of KIIT. Vol. 19, No. 10, pp. 125-136, Oct. 31, 2021. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2021.19.10.125

머신 러닝(Machine Learning) 기법을 활용한 암호화된 TLS 트래픽내 악성코드 탐지 기법

전덕조*, 박동규**

Malware Detection in Encrypted TLS Traffic using Machine Learning Techniques

Deok-jo Jeon*, Dong-Gue Park**

이 논문은 순천향대학교 연구비에 의하여 연구하였음.

요 약

최근 네트워크 트래픽 분석에 의한 탐지를 피하기 위해 암호화된 HTTPS 프로토콜을 사용하는 악성코드가 증가하고 있다. 따라서 암호화된 트래픽에 대한 HTTPS 검사와 고급 행위 기반 위협 탐지 및 대응 없이는 유입되는 많은 위협을 놓치게 된다. 이러한 문제를 해결하기 위해서 많은 조직에서 복호화 프록시를 사용하여 암호화된 트래픽을 모니터링하고 있다. 그러나 이 방식은 성능 영향 및 지연 시간을 포함한 많은 부정적인 영향을 유발하며, 종단 간의 보안을 침해하고 잠재적으로 개인 정보를 침해하며 조직의 신뢰를 약화시킬 수 있다. 이런 문제를 해결하기 위하여 본 논문에서는 TLS 트래픽을 복호화하지 않고도 효과적으로 악성코드를 탐지할 수 있는 방법을 제안하고자 한다. 제안된 방식은 HTTPS 트래픽으로부터 추출한 TLS 세션 메타데이터 특징 집합을 기반으로 네트워크 트래픽내의 악성 TLS 트래픽을 탐지할 수 있는 매우 정확한 분류기를 나타낸다. 실험을 통하여 제안한 방안의 효율성을 증명한다.

Abstract

In order to evade detection by network-traffic analysis, a growing proportion of malware uses the encrypted HTTPS protocol. Therefore without HTTPS inspection of encrypted traffic and advanced behavior-based threat detection and response, organizations are missing many incoming threats. In order to solve this problem, many organization utilizes decryption proxies to monitoring encrypted traffic. But, this approach offers many challenges and limitations including negative performance impacts and latency. And it potentially compromises privacy, and can erode trust in an organization. In this paper, to address these problems, we propose efficient machine learning techniques for detecting malware in TLS traffic without decryption. This work presents a highly accurate supervised classifier that can detect malicious TLS flows in network traffic based on a set of features related to TLS session metadata extracted from HTTPS traffic flows. And we verify efficiency of our method by implementing it.

Keywords:
malware detection, HTTPS, TLS, network analysis, machine learning.

* ㈜ 시큐비스타 대표이사
- ORCID: http://orcid.org/0000-0002-0343-3384
** 순천향대학교 정보통신공학과 교수(교신저자)
- ORCID: http://orcid.org/0000-0002-5864-8825

· Received: Jul. 28, 2021, Revised: Sep. 25, 2021, Accepted: Sep. 28, 2021
· Corresponding Author: Dong-Gue Park
Dept. of Information and Communication Engineering Soonchunhyang Univ.
Tel.: +82-41-530-1347, Email: dgpark@sch.ac.kr.

Ⅰ. 서 론

HTTP 보안 또는 SSL(Secure Sockets Layer)을 통한 하이퍼텍스트 전송 프로토콜이라고도 하는 HTTPS 프로토콜은 인터넷의 보안 통신을 위한 표준이며 모든 컴퓨터 네트워크에서 사용되고 있다. HTTS는 기본적으로 Hypertext Transfer Protocol을 TLS(Transport Layer Security) 또는 SSL을 사용하여 암호화하는 것을 의미한다. HTTPS에서 TLS/SSL의 역할은 HTTP 콘텐츠를 암호화하는 것이다. SSL/TLS의 사용은 인터넷에서 빠르게 증가하고 있다. Google 투명성 보고서[1] 및 Let's Encrypt 통계[2]에 따르면 2021년 5월 기준, Chrome 브라우저에 적재되는 모든 웹 페이지의 95%와 Firefox에 적재된 미국 웹 페이지의 92%가 HTTPS사용하고 있다. 인터넷에서 암호화된 네트워크 트래픽이 증가함에 따라 악성코드도 자체 통신을 보호하기 위해 HTTPS를 사용하고 있으며 이러한 경향은 점진적으로 증가하고 있다.

글로벌 보안회사 소포스에 따르면[3], 2020년 2월 현재, 악성코드의 약 25%가 TLS를 사용하여 통신을 하고 있다고 밝히고 있다. HTTPS 트래픽에서의 악성코드 탐지는 암호화 특성으로 인하여 시그니처 등을 사용하는 기존의 다양한 위협 탐지 기술로는 불가능하다. 기업에서 HTTPS 트래픽을 처리하기 위해 널리 사용하고 있는 솔루션은 HTTPS 복호화 프록시를 설치하는 방법이다. HTTPS 복호화 프록시는 클라이언트와 서버 사이에 배치되어 암호화된 트래픽을 복호화하여 악성 소프트웨어가 있는지 검사한 후, 재 암호화하여 대상 IP로 전송하는 방식을 사용한다. 이러한 접근 방식은 암호화되어 있는 악성코드 트래픽을 탐지할 때, 기존 위협 탐지 방법을 그대로 사용할 수 있다는 장점이 있지만, 복호화를 위한 높은 비용과 네트워크 성능 저하, 지연시간 증가 및 프라이버시를 보장하고자 하는 HTTPS의 본래 목적을 훼손한다는 단점이 있다.

HTTPS트래픽을 복호화하지 않고도 악성코드 트래픽을 정확하게 탐지 할 수 있는 경우에는 HTTPS의 본래 목적인 프라이버시 보장과 더불어 악성코드 탐지 비용을 획기적으로 줄일 수 있게 된다. 이를 위해 본 논문에서는 HTTPS 트래픽으로부터 다양한 특징을 추출하고 이를 기반으로 머신 러닝 기법을 적용하여 HTTPS 트래픽을 복호화하지 않고도 높은 탐지율과 낮은 오탐율을 가지는 악성코드를 탐지 방법을 제안하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서 배경 및 관련 연구를 설명하고, 3장에서는 실험 데이터의 수집 및 특징 추출 방법을 제시한다. 4장에서는 추출된 특징을 이용하여 다양한 머신 러닝 분류 알고리즘에 적용하여 실험하고, 5장에서 결론 및 개선 방향을 제시한다.

Ⅱ. 관련 연구

2.1 악성코드의 TLS 사용

악성코드 작성자가 탐지를 회피하고 자신의 이익을 보호하는 방법을 고려할 때 TLS와 같은 잘 확립되고 검증된 다목적 프로토콜이 유효한 옵션이 될 수 있다[4]-[6]. 최신 악성코드를 살펴보면 여러 작업에서 TLS를 사용하고 있다는 사실을 확인할 수 있다. Check Point Software Technologies의 보고서[7]에는 2020년 11월에 가장 많이 출현한 악성코드가 나열되어 있는데, 대부분이 TLS를 사용하고 있다. 악성코드가 TLS를 이용하는 방법은 기존 악성코드를 분석하면 다음과 같이 여러가지로 구분할 수 있다.

- 페이로드 은닉

TLS는 컴퓨터 감염을 숨기는 데 사용될 수 있다. 예를 들어, 사용자가 TLS를 사용하여 악성 웹 사이트를 방문 할 수 있으며, 이때 드라이브-바이-다운로드(Drive-by-download) 악성 코드는 사용자가 알지 못하는 사이에 스스로 설치된다. 악성 웹 서버와의 모든 통신은 암호화되기 때문에 악성 코드는 기업에서 설정한 패이로드 검사를 회피할 수 있다.

- 데이터 유출

암호화를 통해서 패스워드, 쿠키, 기업 데이터등의 민감한 데이터 유출이 은닉될 수 있다. 공격자는 서버 포트 443(일반적인 SSL대상 포트)을 사용하여 TLS로 캡슐화된 간단한 POST 요청을 통해서 의심을 유발하지 않고 중요한 정보를 유출할 수 있다.

- 명령 및 제어 (C & C)

악성코드에 감염된 컴퓨터가 TLS를 사용하여 C & C 명령을 난독화하는 경우가 증가하고 있다.

위와 같이 공격자는 암호화를 통해 패이로드 검사를 우회 할 수 있으며, 443 포트에 대한 TLS 트래픽을 사용함으로써 의심 없이 간과되도록 하고 있으며, 명령 및 제어 채널의 은닉에도 암호화를 이용함으로써 기존 보안 수단들을 우회하고 있다.

2.2 암호화된 TLS 트래픽 분석 관련 연구

TLS는 정상적인 데이터와 악성코드에 의한 데이터를 동일하게 은닉하기 때문에 이러한 통신을 구분하는 것은 매우 중요하다. 현재까지 암호화된 TLS 트래픽 분석과 관련된 많은 연구가 진행되어 왔다. 암호화된 TLS 분석과 관련하여 가장 널리 사용되는 문서 중 하나는 Cisco 팀이 작성한 "Deciphering Malware's use of TLS (without Decryption)"이다[8].

Cisco 팀은 해당 연구에서 18개의 서로 다른 악성코드의 26000개의 악성 TLS 흐름 샘플과 기업 환경에서 수집된 정상 TLS 통신을 기반으로 다음과 같은 사실을 연구하였다:

● 감염된 클라이언트의 ClientHello 패킷에서:
- 악성코드는 정상 클라이언트와 완전히 다른 암호화 모듈 모음을 사용하며, 이러한 암호 모듈모음은 종종 약하거나 구형인 경우가 많다. 이에 비해 거의 모든 정상 TLS 프로그램은 동일한 암호화 모듈 모음을 사용한다.
- 악성코드는 거의 하나 이상의 확장을 제공하지 않는 반면, 정상 클라이언트는 최대 9개의 확장을 제공한다.
- 악성코드와 정상 클라이언트의 공개 키 길이는 서로 다르다.

● 악성 서버의 ServerHello 및 Certificate 패킷에서:
- 악성코드에 의해 쿼리된 서버는 흔하지 않은 암호 제품군을 선택한다.
- 서버에서 선택한 확장에도 동일한 현상이 나타난다.
- 인증서의 유효 기간도 차이가 난다.
- 인증서가 여러 도메인 이름을 포함하도록 허용하는 인증서의 SAN(SubjectAltName) 항목 수는 상당한 차이가 있다.
- 자체 서명된 인증서를 보내는 악성 서버의 비율은 일반 서버보다 훨씬 높다.

해당 연구에서 Cisco팀은 NetFlow 및 TLS 핸드셰이크 메타 데이터의 조합을 이용하여 정상 및 악성 트래픽 간의 차이점에 중점을 두었는데, 데이터를 해독하지 않고 악성 코드 탐지와 악성 코드 계열 속성을 모두 평가하였다. 그들은 흐름 메타 데이터, 패킷 길이 및 시간 시퀀스, 바이트 분포, TLS 헤더 정보 네 가지 기능 세트를 사용하였다. 연구자들은 L1 로지스틱 회귀 분류기 모델을 사용하여 해당 데이터의 다양한 조합을 수집하여 전체적인 98.5 % - 99.6 % 악성코드 탐지율을 제시하였다. 로지스틱 회귀 분류기는 정상 트래픽으로만 훈련된 모델로 악성 트래픽을 테스트하였기 때문에 전체 분류 비율에서 오탐율을 반영하지 않았다[8][9].

Blake와 McGrew는 기존에 수행했던 연구[8][9]를 확장하여 강력한 암호화된 악성코드 탐지 분류기를 구축하고자 하였다. 해당 팀은 MalDetect라는 도구를 만들었는데, 새로운 샘플이 발견 될 때 재교육 또는 재배포를 방지하기 위해 온라인 모드에서 훈련이 가능한 온라인 랜덤 포레스트 분류기를 활용하였다[10].

해당 연구에서 정상트래픽과 애드웨어 및 악성코드 등 여러 트래픽 유형을 분류 할 수 있는 플랫폼을 개발하였으나, 미탐율은 애드웨어 및 동적 라우팅 프로토콜에서 각각 약 20% 및 50%를 나타냈다[10]. 해당 연구에서 동적으로 훈련 가능한 온라인 랜덤 포레스트 모델이라는 새로운 분석 수단을 소개했던 이유는 장기적인 사용성과 지속 가능성에 중점을 두었기 때문이다[10].

유명 보안 회사인 Lastline이 후원했던 프로젝트에서는 TLS 흐름을 분류하기 위해 TLS 메타 데이터에 중점을 두었다[11]. 해당 연구에서는 로지스틱 회귀, 랜덤 포레스트, K 최근접 이웃, 선형 판별 분석 및 선형 지원 벡터 분류기의 5가지 모델을 생성하였는데, 탐지 결과는 각각 다르게 나타났다. 전반적으로는 악성코드 분류 정확도가 97.6 % 인 모델

을 제시하였다. 해당 연구에서는 악성 및 정상 TLS 핸드셰이크 메타데이터를 비교한 바 있는데, 특히, 악성코드와 정상 트래픽의 통신 메타데이터 특징 간의 차이에 대한 통찰력을 제공하였다.

III. 암호화된 TLS 트래픽내 악성코드 탐지방법

II장에서 설명한 이전의 관련 연구[8]-[11]들은 TLS 핸드셰이크 메타 데이터를 사용할 경우, 유용한 결과를 도출할 수 있다는 사실을 입증하였다.

이전의 관련 연구에서는 흐름 메타 데이터, 패킷 길이 및 시간 시퀀스, 바이트 분포, TLS 헤더 정보 네 가지 기능 세트를 사용하였다. 그러나 바이트 분포 정보나 패킷 길이 및 시간 시퀀스는 머신러닝을 위한 유용한 특징들이지만 이를 계산하기 위해서는 원시 트래픽의 패킷의 실시간 처리를 통한 특징 추출이 필요하므로 추가적인 노력과 오버헤드가 수반된다. 따라서 본 논문에서는 이전 연구들에서 사용했던 패킷 길이 및 시간 시퀀스, 바이트 분포, 등의 특징을 사용하지 않고 오직 TLS 핸드셰이크 과정에서의 텍스트 부분에서 추출 가능한 정보만을 특징으로 하여 이전 연구에서 가장 정확한 결과를 제시했던 랜덤 포레스트와 추가로 서포트 벡터 머신 러닝 모델을 적용하여 실험함으로써 원하는 결과를 제시하고자 한다.

본 논문의 연구를 위하여 그림 1의 실험 환경과 그림 2 실험 방안을 설계하여 TLS 핸드셰이크 메타데이터 기반 악성 세션에 대한 탐지 능력을 평가하고자 한다.

그림 1의 실험 환경에서 먼저 정상 및 악성 TLS 세션을 수집하기 위하여 검증된 패킷 캡처 데이터를 수집한 후, 해당 PCAP 파일로부터 TLS 세션 데이터를 추출한다. 추출된 각 세션 데이터로부터 머신 러닝 모델에 적용할 특징을 정의한다. 다양한 특징 세트를 이용하여 다양한 머신 러닝 분류 알고리즘에 적용한 후, 결과를 검증하여 최종 결과를 도출한다.

그림 2의 실험 방안에서는 TLS 세션 메타데이터 저장소로부터 정의된 특징을 선택하여 다양한 머신 러닝 기법을 적용하고 도출된 결과를 10 중 교차 검증 기법으로 검증한 후, 결과를 분석한다.

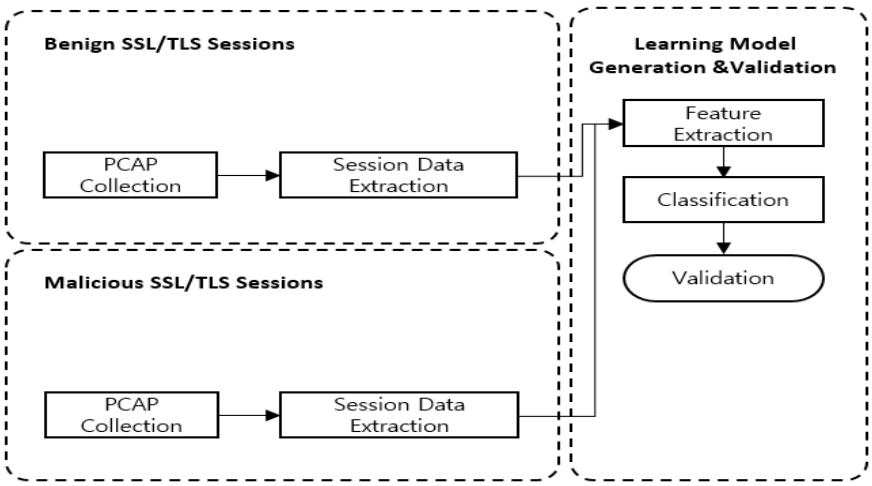

그림 1. 실험 환경
Fig. 1. Experimental environment

그림 2. 실험 방안
Fig. 2. Experimental method

3.1 정상 TLS 세션 데이터 수집

본 논문에서는 캐나다 인터넷 등록 기관(Canadian internet registration authority)이 후원하는 캐나다 사이버 보안 연구소(CIC) 그룹에서 제공한 데이터 세트를 정상 데이터 세트로 사용하였다[12]. CIRA- CIC-DoHBrw-2020 데이터는 암호화된 트래픽의 시계열 분류를 통한 DoH 터널 탐지 연구[13]를 위해 수집된 데이터로서 2020년 초에 여러 주에 걸쳐 수집되었다. CIRA-CIC-DoHBrw-2020 데이터는 악성 및 정상 네트워크 캡처 파일을 모두 포함하는 레이블이 지정된 데이터 세트를 제공하는데 본 논문에서는 정상 네트워크 캡처 파일만을 이용하였다.

정상 TLS 네트워크 캡처 파일의 구성은 표 1과 같다. 표 1의 데이터에서 그림 3과 같이 TLS 클라이언트 및 서버 핸드셰이크를 추출하여 TLS 세션 메타 데이터를 생성하였고, 머신 러닝 모델을 실험하기 위하여 110,490개의 정상 TLS 세션의 데이터 세션 샘플을 포함하는 CSV 파일을 생성하였다.

3.2 악성 TLS 세션 데이터 수집

악성 네트워크 캡처 파일은 모두 웹 사이트 Malware-traffic-analysis.net[14]에서 수집하였다. 해당 사이트는 다양한 최신 악성코드 샘플과 감염, 페이로드 전달 및 C2 트래픽 프로필이 포함된 수많은 네트워크 캡처 파일을 호스팅한다. 수집한 악성 TLS 트래픽 데이터 세트에서는 Dridex, TrickBot, Quakbot 등의 다양한 악성 코드 군에 속한 1,000 개의 악성코드 관련 트래픽 캡처 파일을 수집하였다. 수집한 TLS 기반 악성코드 종류 및 샘플 구성은 표 2와 같다.

표 2의 악성코드 샘플 트래픽 캡처 파일에서 그림 4와 같이 TLS 클라이언트 및 서버 핸드 셰이크를 추출하여 TLS 세션 메타 데이터를 생성하였고, 머신 러닝 모델을 실험하기 위하여 25,688개의 악성 TLS 세션 샘플을 포함하는 CSV 파일을 생성하였다.

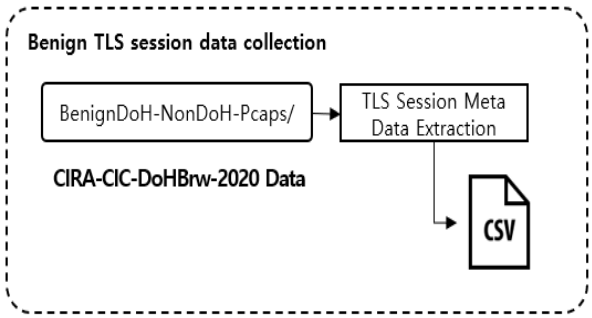

그림 3. 정상 TLS 세션 데이터 수집
Fig. 3. Benign TLS session data collection

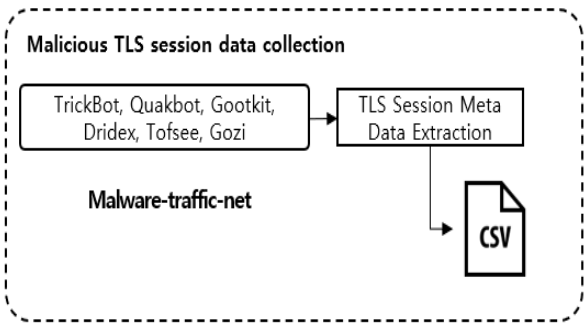

그림 4. 악성 TLS 세션 데이터 수집
Fig. 4. Malicious TLS session data collection

표 1. CIRA-CIC-DoHBrw-2020 정상 트래픽 캡처 데이터
Table 1. CIRA-CIC-DoHBrw-2020's benign traffic capture data

File name	Last modified	Size
BenignDoH_NonDoH-Chrome-AdGuard.zip	2020-07-27 12:43	4.3G
BenignDoH_NonDoH-Chrome-Cloudflare.zip	2020-07-28 19:09	4.3G
BenignDoH_NonDoH-Chrome-Google.zip	2020-07-29 18:16	4.0G
BenignDoH_NonDoH-Chrome-Quad9.zip	2020-07-31 12:26	7.4G
BenignDoH_NonDoH-Firefox-AdGuard.zip	2020-08-03 16:04	8.9G
BenignDoH_NonDoH-Firefox-CloudFlare.zip	2020-08-07 9:56	8.7G
BenignDoH_NonDoH-Firefox-Google.zip	2020-08-11 19:04	5.4G
BenignDoH_NonDoH-Firefox-Quad9.zip	2020-08-16 23:48	9.0G

표 2. Malware-traffic-analysis.net에서 수집한 TLS기반 악성코드 샘플
Table 2. TLS-based malware samples from Malware-traffic-analysis.net

Malware name	# of Malware samples
TrickBot	186
Quakbot	272
Gootkit	42
Dridex	409
Tofsee	49
Gozi	42

3.3 특징 추출

이전 관련 연구에서는 악성 TLS 핸드셰이크와 정상 TLS 핸드셰이크 간에 측정 가능한 차이를 발견하였다[11]. 따라서 본 논문에서는 NetFlow와 같은 패킷 메타데이터나 머신러닝 결과에 영향을 미치는 패킷 길이 및 시간 시퀀스와 같은 패킷 메타데이터를 사용[9]하지 않고 TLS 핸드셰이크 메타데이터에만 초점을 맞추고자 한다. TLS 핸드셰이크 과정에서 추출 가능한 통신 메타데이터는 총 511개로서 표 3과 같다.

본 논문에서는 효과적인 머신 러닝을 위한 데이터 특징 세트를 선별하기 위해서 먼저 정상 TLS 세션 메타데이터와 악성 TLS 세션 메타데이터의 비교를 통하여 차이가 발생하거나 발생 가능성이 높은 데이터 특징을 분석하여 선택하였다. 먼저 두 가지 데이터 세트에서 차이가 가장 많이 발생하는 특징을 각 데이터 세트에서 10가지를 선택하였다.

추가적인 데이터 분석으로 클라이언트가 제공하는 고유한 암호 제품군 및 서명 알고리즘의 수를 조사하여 추가하였는데, 정상 TLS 세션 데이터보다는 악성 TLS 세션 데이터에서 더 많은 암호 제품군 및 더 다양한 서명 알고리즘을 사용하는 특징이 발견되어 이를 반영하였다.

표 3. TLS 핸드셰이크 메타데이터 특징
Table 3. Features of TLS handshake metadata

Features	Size	Data Type
Source port	1	Integer
Destination port	1	Integer
TLS record type	1	Integer
Client TLS version	1	Integer
Message length	1	Integer
Cipher suite length	1	Integer
Cipher suites	351	Float
Extension length	1	Integer
Handshake type	1	Integer
Handshake length	1	Integer
Handshake version	1	Integer
Signature algorithms	36	Float
Supported groups	49	Float
Supported points	3	Integer
Server OCSP stapling	1	Integer
Server TLS version	1	Integer
Server supported TLS version	1	Integer
Server extensions	59	Float
Total	**511**	

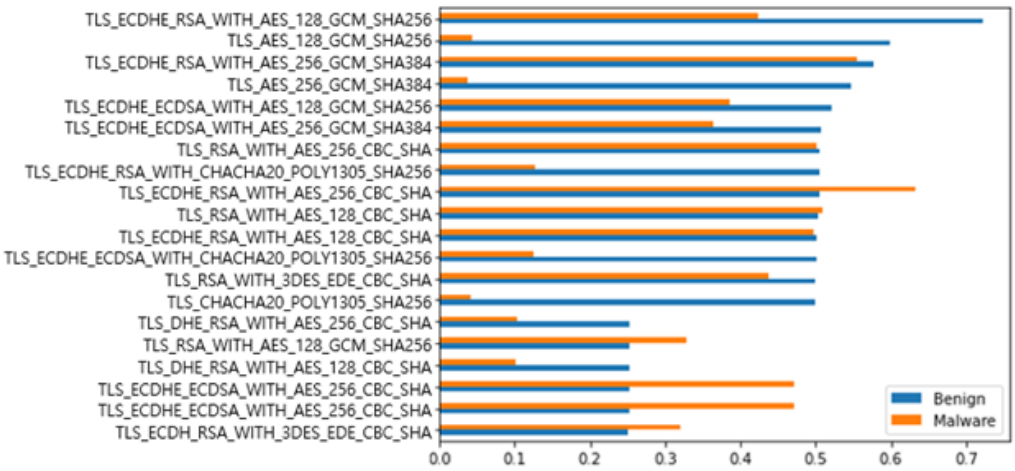

그림 5. 정상데이터의 암호 제품군 중요도
Fig. 5. Importance of cipher suite by benign data

그림 5와 6은 정상 데이터의 암호 제품군 중요도와 악성코드 데이터의 암호 제품군 중요도를 나타낸다. 여기에 정상 TLS 통신 세션과 악성코드에 의한 TLS 통신 세션이 사용하는 TCP 목적지 및 소스 포트 번호 사용의 차이에 대한 데이터 특징을 추가하였다. 정상 및 악성코드 데이터의 목적지 포트 및 출발지 포트 차이는 그림 7, 8과 같다.

TLS 세션 핸드셰이크 길이 및 메시지 길이와 같은 메타데이터 필드는 공격자가 임의로 변경할 수 없는 부분이므로 이러한 특징을 추가로 선택하여 반영하였다.

이러한 과정의 데이터 분석 및 특징 추출 과정을 통하여 표 4의 특징 세트를 추출하여 실험하였다.

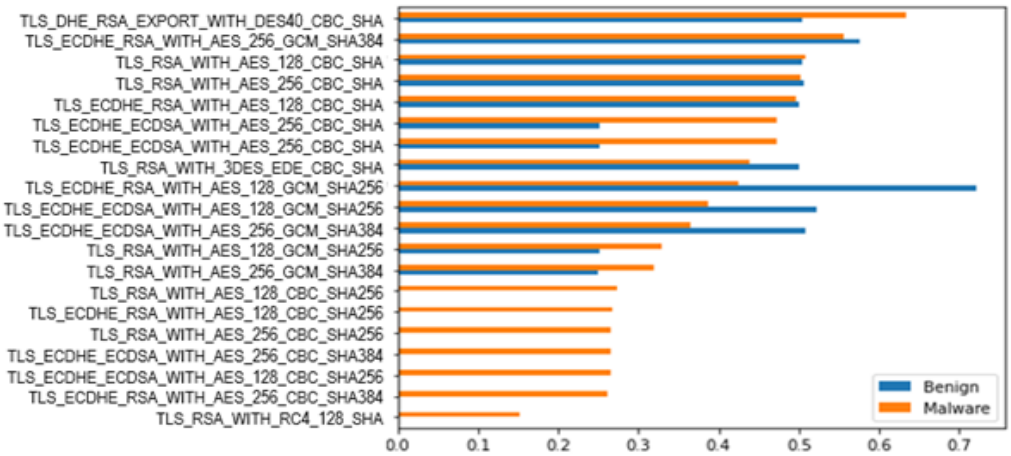

그림 6. 악성코드 데이터의 암호 제품군 중요도
Fig. 6. Importance of cipher suite by malware data

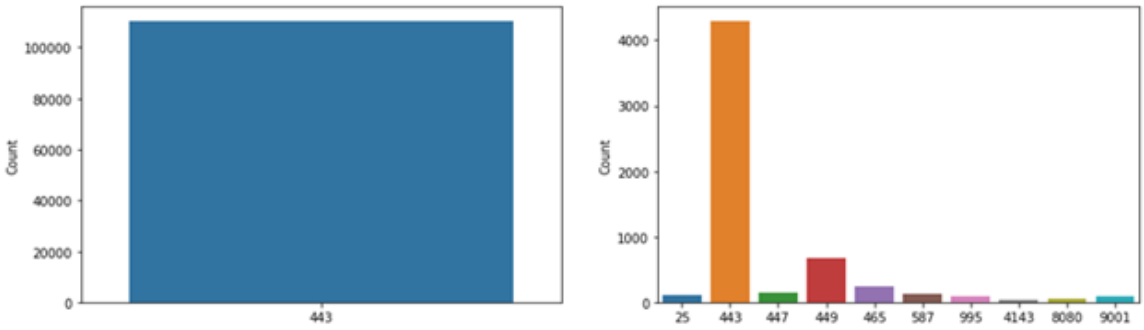

그림 7. 정상 및 악성코드 데이터의 목적지 포트 차이
Fig. 7. Difference in destination ports usage of benign data and malware data

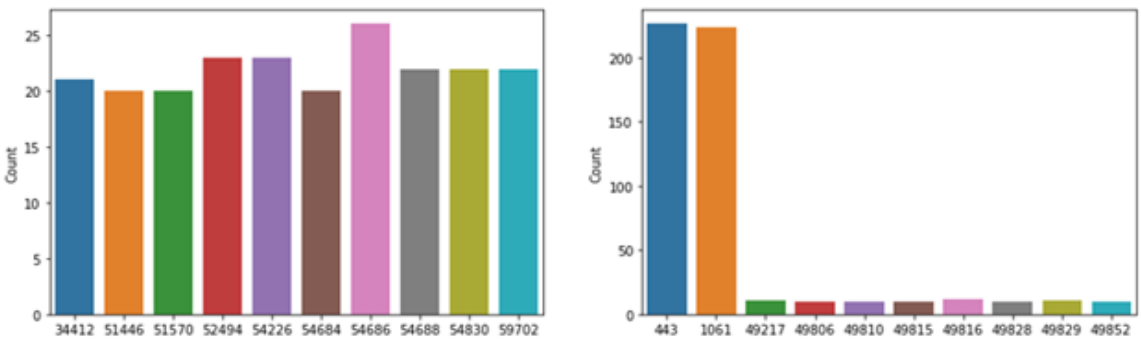

그림 8. 정상 및 악성코드 데이터의 출발지 포트 차이
Fig. 8. Difference in source ports usage of benign data and malware data

표 4. TLS 핸드셰이크 메타데이터 실험 특징 세트
Table 4. Feature set of selected TLS handshake metadata

Features	Total Size	Data Type	Selected Features	Size
Source Port	1	integer	Source Port	1
Destination Port	1	integer	Destination Port	1
TLS Record Type	1	integer	TLS Record Type	1
Client TLS Version	1	integer	Client TLS Version	1
Message Length	1	integer	Message Length	1
Cipher Suite Length	1	integer	Cipher Suite Length	1
Cipher Suites	351	Float	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA TLS_ECDH_ECDSA_WITH_NULL_SHA TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_3DES_EDE_CBC_SHA TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_256_GCM_SHA384 TLS_DHE_DSS_WITH_AES_256_CBC_SHA TLS_DHE_DSS_WITH_AES_128_CBC_SHA TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA TLS_RSA_WITH_RC4_128_SHA	20
Extension Length	1	integer	Extension Length	1
Handshake Type	1	integer	Handshake Type	1
Handshake Length	1	integer	Handshake Length	1
Handshake Version	1	integer	Handshake Version	1
Signature Algorithms	36	Float	rsa_pss_rsae_sha256 rsa_pss_rsae_sha384 rsa_pss_rsae_sha512 rsa_pkcs1_sha384 rsa_pkcs1_sha256 ecdsa_secp256r1_sha256 ecdsa_secp384r1_sha384 rsa_pkcs1_sha512	8
Server OCSP Stapling	1	integer	Server OCSP Stapling	1
Server TLS Version	1	integer	Server TLS Version	1
Server Supported TLS Version	1	integer	Server Supported TLS Version	1
Server Extensions	59	Float	renegotiation_info server_name status_request supported_groups(old: "elliptic_curves") ec_point_formats Heartbeat application_layer_protocol_negotiation signed_certificate_timestamp extended_master_secret token_binding session_ticket (old :"SessionTicket TLS") pre_shared_key supported_versions key_share Used for "random" values provided by client	15
Total	**511**			**56**

IV. 실험 및 결과 분석

4.1 분류기 구성

본 논문에서는 두 가지 분류기를 사용하였는데, 이전 연구에서 가장 좋은 결과를 보여 주었던 의사 결정 트리(Decision tree)의 앙상블 기법 중 하나인 랜덤 포레스트(Random forest)와 또 다른 분류기로는 데이터 분류에 강점이 있는 서포트 벡터 머신(SVM, Support Vector Machine)을 분류기로 선택하였다.

Blake Anderson 및 David McGrew의 연구[8][9]에서 랜덤 포레스트방법이 가장 좋은 결과를 도출하였다. 랜덤 포레스트는 의사 결정 트리를 이용하는 앙상블 기법 중 하나인데, 앙상블 기법은 여러 분류 모델(Classification)을 이용하여 데이터를 학습하고 모든 모델의 예측 결과를 평균하여 예측하는 방법이다[15]. 랜덤 포레스트는 여러 의사 결정 트리의 평균 출력을 반환한다.

랜덤 포레스트의 가장 큰 특징은 무작위성(Randomness)에 의해 트리들이 서로 조금씩 다른 특성을 갖는다는 점이다. 이 특성은 각 트리들의 예측 값들 간에 상관관계를 줄임으로써 예측 값을 일반화(Generalization)하는 의사 결정 트리의 단점을 보완한 앙상블 기법이다.

또 다른 분류기로는 데이터 분류에 강점이 있는 서포트 벡터 머신을 분류기로 선택하였다. SVM은 패턴 인식, 자료 분석을 위한 지도 학습 모델이며, 주로 분류와 회귀 분석을 위해 사용한다[16]. 두 가지 카테고리 중 어느 하나에 속한 데이터의 집합이 주어졌을 때, SVM 알고리즘은 주어진 데이터 집합을 바탕으로 새로운 데이터가 어느 카테고리에 속할지 판단하는 비확률적 이진 선형 분류 모델을 생성하는 방식으로 데이터 분류에 강점이 있다. 생성된 분류 모델은 데이터가 사상된 공간에서 경계로 표현되는데 SVM 알고리즘은 가장 합리적인 경계를 찾는 알고리즘이다. SVM은 선형 분류와 더불어 비선형 분류에서도 사용될 수 있다. SVM은 다양한 문제에 대해 강력하고 최적의 결과를 생성하는 방식이다[17].

4.2 결과 분석 방법

머신 러닝 분류 모델을 평가하기 위한 다양한 평가 기준들이 존재한다. 분류 모델에 대한 모델 평가는 데이터에 라벨이 있는 경우에 대한 모델 평가 방법을 사용한다. 모델의 정확도가 100%이면, 양성과 음성 데이터를 100% 잘 구분 할 것이다. 양성으로 예측된 영역을 Positive prediction, 음성으로 분리된 영역을 Negative prediction 이라고 한다. 실제에서는 정확도 100% 모델은 매우 드물고, 다음의 네 가지 경우가 발생한다.

양성인데 양성으로 제대로 검출한 것은 TP(True Positive), 음성인데 음성으로 제대로 검출한 것은 TN(True Negative), 양성인데 음성으로 잘못 검출한 것은 FN(False Negative), 음성인데 양성으로 잘못 검출한 것은 FP(False Positive)라고 하는데 이를 정리하여 표현하면 표 5와 같다.

표 5. 혼동메트릭스
Table 5. Confusion matrix

		Predicted	
		Positive	Negative
Observed	Positive	TP (judge True as True)	FN (judge False as False)
	Negative	FP (judge False as True)	TN (judge True as False)

표 5를 이용하여 분류 모델의 성능을 나타내는 척도 중 하나는 정확성(Accuracy)이다. 정확성은 가장 대표적으로 사용되는 머신 러닝 모델 평가지표로서 전체 데이터 중에서 제대로 분류된 데이터의 비율을 표현한다. 정확성은 (TP + TN) / (전체 데이터 수 = P + N)로 표현되며, 해당 모델이 얼마나 정확하게 분류를 하는지를 나타낸다. 에러율(Error rate)은 (FN+FP) / (전체 데이터 수 = P+N) 로 표현되며 정확성과 반대로, 전체 데이터 중에서 잘못 분류한 비율을 나타낸다.

민감도(Sensitivity) 또는 재현율(Recall)은 (TP) / P로 표현되며, 원래 양성 데이터 수에서 양성으로 분류된 비율을 나타내는데, 모델이 얼마나 정확하게 양성 값을 찾는 지를 나타낸다. 정밀도(Precision)는 TP / (TP+FP) 로 표현되며, 양성으로 예측한 데이터 중에서 실제 양성 데이터의 비율을 나타낸다.

4.3 결과 분석

본 논문에서 사용한 첫 번째 SVM 분류 모델은 무작위로 선택된 25,000개 샘플로 훈련한 후 전체 데이터 세트를 사용하여 검증한 결과, 표 6과 같이 약 96.6%의 평균 재현율로 결과를 예측하였다. SVM 모델은 전체 데이터 세트에 일부 악성코드를 제외한 대부분의 악성코드 인스턴스 즉, 99.7% 이상의 악성코드 인스턴스를 효과적으로 탐지할 수 있음을 보여주었는데, 이 결과는 Roques의 연구 결과[11] 보다 개선된 탐지 성능을 보여주며, 본 논문의 목표였던 머신 러닝 분류기를 활용한 악성 TLS 세션에 대한 정확한 식별 가능성을 나타내는 중요한 성과이다.

본 논문에서 두 번째로 사용했던 랜덤 포레스트 분류기는 악성코드 인스턴스에 의한 TLS 세션 탐지에 대해 99.5% 정확도를 나타냈으며, 이는 Roques의 연구 결과[11]와 유사한 정도의 정확도를 나타냈다.

표 6. 악성 TLS 세션 탐지 성능 비교
Table 6. Comparison of malicious TLS session detection performance.

	Algorithm	Feature selection	Accuracy	Precision	Recall
[11]	Random forest	Full Set (417 Features)	99.6	99.6	99.6
	Random forest	Reduced Set (208 features)	99.5	99.5	99.5
Proposed method	SVM	56 Features	99.7	98.9	96.6
	Random forest		99.5	99.6	99.6

Roques[11]의 연구에서 전체적인 탐지 결과는 Cisco[9]와 유사한 정확성을 제시하였다. 본 연구에서는 이전의 연구 [9][11]와는 달리 패킷 길이 및 시간 시퀀스를 사용하지 않고 TLS 세션의 텍스트 부분에서 추출 가능한 56개 특징 세트만을 이용하여 Roques[11]의 연구와 유사한 정확성을 제시하였다. Roques11]의 지적대로 약간의 정확성 개선의 차이는 특징 선택, 훈련 데이터 세트의 크기, 내용 등에 의해서 달라질 수 있다.

본 연구에서는 TLS 세션의 텍스트 부분에서 추출 가능한 특징만을 사용한 머신 러닝 분류기를 활용하여 악성 TLS 세션에 대한 정확한 식별 가능성을 제시하였다.

모든 머신 러닝 기반 분류기에는 특징 선택, 훈련 데이터 세트의 크기 및 내용, 분류기 최적화 등 많은 요인으로 부터 기인하는 정확성에 차이가 있을 수밖에 없으므로 견고성을 개선하고 오탐율을 개선해야 한다는 한계가 있다. 그럼에도 불구하고 TLS 세션의 악성 여부 분석을 하기 위해서 본 논문에서 제안한 방식이 기존의 개인정보 보호를 훼손하는 동시에 성능 저하를 유발하는 TLS패킷 암호 해독 과정이 필요한 탐지 기술에 비해 상당한 장점이 있음을 확인하였다.

V. 결 론

HTTPS 트래픽에서의 악성코드 탐지는 암호화 특성으로 인하여 시그니처 등을 사용하는 기존의 다양한 위협 탐지 기술로는 불가능하다. 기업에서 HTTPS 트래픽을 처리하기 위해 널리 사용하고 있는 솔루션은 HTTPS 복호화 프록시를 설치하는 방법으로, 이 방식은 암호화되어 있는 악성코드 트래픽을 탐지할 때, 기존 위협 탐지 방법을 그대로 사용할 수 있다는 장점이 있지만, 복호화를 위한 높은 비용과 네트워크 성능 저하, 지연시간 증가 및 프라이버시를 보장하고자 하는 HTTPS의 본래 목적을 훼손한다는 단점이 있다.

따라서 HTTPS트래픽을 복호화하지 않고도 악성코드 트래픽을 정확하게 탐지 할 수 있는 경우에는 HTTPS의 본래 목적인 프라이버시 보장과 더불어 악성코드 탐지 비용을 획기적으로 줄일 수 있게 된다. 이를 위해 본 논문에서는 HTTPS 트래픽으로부터 다양한 특징을 추출하고 이를 기반으로 머신 러닝 기법을 적용하여 HTTPS 트래픽을 복호화하지 않고도 높은 탐지율과 낮은 오탐율을 가지는 악성코드 탐지 방법을 제안하였다.

본 논문에서 제안한 머신 러닝 모델을 정상 및 악성 TLS 통신 데이터 세트에 적용하여 이전 연구

들과 비교 실험함으로써 본 논문에서 제안한 머신러닝 모델의 효율성을 검증하였다.

향후에는 악성 TLS 세션 탐지 시에 오탐율을 낮추기 위하여 여러 가지 방법을 혼용함으로써 분류기의 탐지 성능을 개선하는 연구를 수행할 예정이다.

References

[1] Google, "HTTPS encryption on the web", https://transparencyreport.google.com/https/overview. [accessed: May 02, 2021]

[2] Let 's Encrypt Statistics, https://letsencrypt.org/stats/. [accessed: May 02, 2021]

[3] Sophos News, "Nearly a quarter of malware now communicates using TLS", https://news.sophos.com/en-us/2020/02/18/nearly-a-quarter-of-malware-now-communicates-using-tls. [accessed: May 10, 2021]

[4] The TLS Protocol Version 1.0, https://tools.ietf.org/html/rfc2246. [accessed: May 12, 2021]

[5] Seb's IT blog, "Traffic analysis of a TLS session", https://megamorf.gitlab.io/2020/03/03/traffic-analysis-of-a-tls-session. [accessed: May 12, 2021]

[6] SSL Labs, "SSL Pulse", https://www.ssllabs.com/ssl-pulse. [accessed: May 11, 2021]

[7] Checkpoint Software, "November 2020's Most Wanted Malware", https://www.globenewswire.com/en/news-release/2020/12/09/2142018/0/en/November-2020-s-Most-Wanted-Malware-Notorious-Phorpiex-Botnet-Returns-As-Most-Impactful-Infection.html. [accessed: May 07, 2021]

[8] David McGrew, Blake Anderson, and Subharthi Paul, "Deciphering Malware's use of TLS", Journal of Computer Virology and Hacking Techniques, Vol. 14, No. 1, pp. 195-211, Jul. 2016. https://doi.org/10.1007/s11416-017-0306-6.

[9] Blake Anderson. Detecting Encrypted Malware Traffic (Without Decryption). June 2017. URL: https://blogs.cisco.com/security/detecting-encryptedmalware-traffic-without-decryption. [accessed: May 12, 2021]

[10] J. Liu, Y. Zeng, J. Shi, Y. Yang, R. Wang, and L. He, "MalDetect: A structure of encrypted malware traffic detection. Computers", Materials & Continua, Vol. 60, No. 2, pp. 721–739, Jan. 2019. http://dx.doi.org/10.32604/cmc.2019.05610.

[11] O. Roques, "Detecting Malware in TLS Traffic", The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05), Sep. 2019. http://dx.doi.org/10.1109/lcn.2005.35.

[12] CIRA-CIC-DoHBrw-2020, http://205.174.165.80/CICDataset/DoHBrw-2020/Dataset. [accessed: May 20, 2021]

[13] M. MontazeriShatoori, L. Davidson, G. Kaur, and A. Habibi Lashkari, "Detection of DoH Tunnels using Time-series Classification of Encrypted Traffic", 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress, Calgary, AB, Canada, pp. 63-70, Aug. 2020. https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00026.

[14] Malware-Traffic-Analysis.net, https://www.malware-traffic-analysis.net. [accessed: Jan. 20, 2021]

[15] Leo Breiman, "Random Forests", Machine Learning 45, Vol. 1, pp. 5–32, Oct. 2001.

[16] Corinna Cortes and Vladimir Vapnik, "Support-Vector Networks", Machine Learning 20, Vol. 3, pp. 273–297, Sep. 1995.

[17] Christopher Bishop, "Pattern Recognition and Machine Learning", Springer, Vol. 128, pp. 1–58, 2006.

저자소개

전 덕 조 (Deok-Jo Jeon)

1989년 2월 : 아주대학교 전자계산학과(공학사)
1995년 5월 : 뉴맥시코대학교 전자계산학과(이학석사)
2005년 10월 ~ 현재 : (주)씨큐비스타 대표이사
관심분야 : 지능형 위협 대응, 지능형 보안관제, 네트워크 보안

박 동 규 (Dong-Gue Park)

1992년 2월 : 한양대학교 전자공학과(공학박사)
1992년 3월 ~ 현재 : 순천향대학교 정보통신공학과 교수
관심분야 : 제어 시스템 보안, 네트워크보안, 시스템 보안, 모바일 보안