정 보 보 호 학 회 지
제 31 권 제5호, 2021. 10

사이버 위협 인텔리전스를 위한 TLS Fingerprinting 동향

노 희 준*

요 약

사이버 공격이 점점 복잡해지고 빠르게 진화하면서, 기존에 알려졌거나 향후 예견되는 위협에 대한 근거를 바탕으로 한 지식인 사이버 위협 인텔리전스(CTI)를 구축하고 공유하려는 노력이 최근 주목을 받고 있다. 특히, Transport Layer Security(TLS)와 같은 암호화 프로토콜을 활용해 암호화된 인터넷 트래픽이 증가하면서 패킷 페이로드(payload)의 분석이 어려워짐에 따라, 암호화된 네트워크 트래픽을 핑거프린팅(fingerprinting)하고 이 정보를 공유해 위협의 근거로 활용하고자 하는 시도들이 등장하고 있다. 본 논문에서는 최근에 제안된 주요 TLS 핑거프린팅 기법 및 이를 사이버 위협 인텔리전스 구축을 위해 데이터베이스화 하는 사례를 소개한다.

Ⅰ. 서 론

최근 사이버 공격이 점점 복잡해지고 빠르게 진화하면서 기존에 알려졌거나 향후 예견되는 위협에 대한 근거를 바탕으로 한 지식인 사이버 위협 인텔리전스(Cyber Threat Intelligence, CTI)를 구축하고 공유하려는 노력이 최근 주목을 받고 있다. 사이버 위협 인텔리전스라는 용어의 정의는 학계 및 산업계에서 다양하게 논의되고 있으나, 2013년에 Gartner의 보고서[1]에서 등장한 *"evidence-based knowledge, including context, mechanisms, indicators, implications and actionable advice, about an existing or emerging menace or hazard to assets that can be used to inform decisions regarding the subject's response to that menace or hazard.* (자산에 대한 기존 또는 새로운 위협이나 위협이 있을 때, 이에 대한 자산 주체의 대응에 관한 의사결정에 사용할 수 있는 맥락, 메커니즘, 지표, 시사점, 및 실행가능한 조언을 포함한 증거 기반 지식)"이 보편적으로 인용되고 있고, 미국 국립표준기술연구소(National Institute of Standards and Technology, NIST)의 SP 800-150[2]에서는 *"threat information that has been aggregated, transformed, analyzed, interpreted, or enriched to provide the necessary context for decision-making processes*(의사결정 과정에서 필수적인 맥락을 제공하고자 취합, 변환, 분석, 해석, 또는 강화된 위협 정보)"라고 정의하고 있다.

사이버 위협 정보 공유의 중요성을 인식에 기반을 두어 발간된[3] NIST SP 800-150[2]에서 논의하는 위협 정보에는 다양한 유형이 있으나, 주요 유형으로는 침해지표(Indicators of Compromise, IoC), 공격자의 전략, 전술, 및 그 과정 (Tactics, Techniques, Procedures, TTPs), 보안 경고(Security Alerts), 위협 인텔리전스 보고서(Threat Intelligence Reports), 도구 설정(Tool Configurations)을 들 수 있다.

이 중에서도 침해지표는 공격이 임박했거나 이미 침해가 발생했을 수 있음을 나타내는 기술적인 아티팩트(Artifact) 또는 네트워크/시스템의 이벤트를 일컫는다. 네트워크 프로토콜과 관련된 침해지표를 예로 들면 다음과 같다[4].:

- 네트워크 트래픽의 IPv4 및 IPv6 주소
- 네트워크 트래픽의 DNS 도메인명, DNS 리졸버(resolver)의 캐시(cache) 또는 로그(log)
- 네트워크 트래픽의 TLS 서버 네임 인디케이션(Server Name Indication, SNI) 값
- 네트워크 트래픽의 바이너리 코드 서명 인증서 또는 (SHA256 해시와 같은) TLS 인증서 정보
- 네트워크 트래픽이나 파일 시스템 아티팩트에서 계산된, 악성 바이너리나 악성 스크립트의 암호화 해시

본 연구는 2021년도 한국과학기술정보연구원(KISTI) 주요사업 과제로 수행한 것입니다.
* 고려대학교 일반대학원 사이버보안학과 (조교수, hjroh@korea.ac.kr)

사이버 위협 인텔리전스의 정의에서 추론할 수 있듯이, 위와 같은 침해지표 데이터를 확보해 리스트 형식으로 나열한 것은 사이버 위협 인텔리전스라고 할 수는 없다. 해당 침해지표가 가지는 맥락을 파악할 수 있는 증거가 필요하기 때문이다. 또한, 특정 상황에 한정된 침해지표 데이터의 분석만으로는 그 맥락을 이해하고 분석하는데 한계가 있다.

이와 같은 문제 인식으로 인해, 위협 정보 공유가 하나의 시장을 이룰 정도[5][6]로 활발히 이루어지고 있으며, 국내에서도 국내외 기업 및 기관 간의 위협 정보를 공유하기 위한 시스템인 한국인터넷진흥원(KISA) C-TAS가 알려져 있다. 또한 AlienVault OTX, ThreatCrowd, ThreatMiner 같은 침해지표 중심의 공개 출처 위협 정보(Open Source Intelligence, OSINT) 취득 및 관리 시스템으로부터 연관 관계나 문맥을 표현하는 지식 그래프를 구축하는 시스템[7][8]도 제안되고 있다.

하지만 현재의 공개 출처 위협 정보를 제공, 교환하는 시스템은 최근 TLS와 같은 암호화 프로토콜을 활용해 암호화된 인터넷 트래픽이 증가하면서 점차적으로 활용에 제약이 생기는 상황이다. 예를 들어 AlienVault OTX는 응용 페이로드에서 추출 가능한 URL, 도메인명, SHA256 해시값 등을 제공하고 있지만, HTTPS나 DNS over TLS와 같이 TLS 기반으로 암호화하는 경우, (추이적 신뢰(Transitive Trust)로 정당화되어야만 사용가능한 SSL/TLS Interception Proxy 없이는) 미들박스(middlebox)에서 침해지표를 추출하기 어렵다[9].

이러한 문맥에서, 최근 학술적인 측면에서 주로 이루어지던 네트워크 트래픽 분석[10][11]을 사이버 위협 인텔리전스에 활용하고자, 암호화된 TLS 트래픽으로부터 운영체제, 프로세스 등의 정보를 식별하고자 특징을 추출, 분석하는 TLS 핑거프린팅(fingerprinting)이 산업계 및 보안 커뮤니티를 중심으로 제안되고 있다. 본고에서는 주요 TLS 핑거프린팅 기법 및 이를 데이터베이스화하는 사례를 소개한다.

Ⅱ. 주요 TLS 핑거프린팅 기법

본 절에서는 먼저 TCP 플로우를 핑거프린팅하는 주요 공개 도구의 기법을 소개한 뒤, (TLS의 전신인) SSL 및 TLS 핑거프린팅을 위한 기법을 오픈 소스 유틸리티와 주요 활용 사례 측면에서 각각 소개한다.

2.1. TCP/IP 스택 핑거프린팅 도구와 적용 기법

인터넷 트래픽 플로우로부터 특징을 프로파일링하여 해당 플로우를 생성한 응용이나 운영체제를 식별하는 연구는 World Wide Web(WWW)이 보급되는 1990년대에서부터 발견[12]할 수 있으며, 네트워크 정보 및 보안 감사를 위한 오픈 소스 유틸리티인 nmap은 1998년에 공개한 기고[13]에서 원격 호스트에 특별한 패킷을 송신해 그 응답에서 특징을 추출하고, 450개 이상의 시그니처로 이루어진 데이터베이스와 비교해서 운영체제를 식별하는 TCP/IP 스택 핑거프린팅 기법을 공개[14]한 바 있다.

nmap의 초기 핑거프린팅 기법은 FIN probe, BOGUS flag probe, TCP Initial Sequence Number (ISN) Sampling, Don't Fragment bit, TCP Initial Window, ACK Value, ICMP Error Message Quenching, ICMP Message Quoting, ICMP Error message echoing integrity, Type of Service, Fragmentation Handling, TCP Options, Exploit Chronology, SYN Flood Resistance와 같은 다양한 핑거프린팅 기법으로 이루어져 있다. 하지만 이 기법은 패킷을 송신해서 그 응답을 관찰하는 능동적 핑거프린팅 기법이라는 점에서 한계가 존재한다.

한편, p0f[15]는 2000년에 발표된 수동적 TCP/IP 핑거프린팅 도구로, 네트워크 스니핑(sniffing)을 통해 확보한 패킷 정보를 핑거프린팅해 데이터베이스와 비교하여 운영체제 정보를 식별할 수 있다는 장점을 가지고 있다. p0f의 초기 버전은 인터넷 커뮤니티를 통해 얻어진 200여개의 시그니처를 통한 규칙 기반 핑거프린팅 기법을 사용한 한계가 있다. Beverly[14]는 p0f의 규칙 기반 핑거프린팅 기법을 Internet Exchange Point(IXP)에서 수집한 데이터셋(dataset)을 p0f에 적용하였을 때 5%의 호스트의 운영체제 식별에 실패하였으며, 이를 간단한 통계적 분류기를 도입해 개선할 수 있음을 보였다.

2.2. SSL/TLS 핑거프린팅 기법 활용 사례

Secure Sockets Layer(SSL)은 1995년 2월에 일반 공개된 종단간 암호화 프로토콜로, TCP 위에서 동작

한다. SSL의 마지막 버전인 SSL 3.0은 2015년 6월에 폐지 예정(deprecated)으로 지정[16]되었다.

TLS는 SSL과 상호호환성(interoperability)은 없으나 하위호환성(backward compatibility)을 지원하도록 설계한 SSL 3.0의 개선판으로, TLS 1.0 프로토콜 명세를 담은 RFC 2246[17]은 1999년 1월에 출판되었다. TLS 프로토콜은 2006년 TLS 1.1(RFC 4346[18]), 2008년 TLS 1.2(RFC 5246[19]), 2018년 TLS 1.3(RFC 8446[20])로 점진적으로 개선되었으며, 2021년 3월 RFC 8996[21]이 출판되면서 TLS 1.0과 TLS 1.1이 폐지 예정으로 지정되었다.

기존 TCP/IP 프로토콜 조합은 대다수의 운영체제에 프로토콜이 내장되어 있었기 때문에, 심층 패킷 분석(Deep Packet Inspection, DPI)과 같이 응용 계층의 페이로드에서 특징을 수집하고 분석하는 기법을 병행하거나 응용 계층과 연관된 TCP/IP 헤더 필드를 분석하지 않는 한, 2.1절에서 살펴본 바와 같이 주로 운영체제와 그 버전을 식별하는 핑거프린팅 기법이 주를 이루었다.

하지만 Korczynski and Duda[22]는 (PayPal과 같이 서버에서 실행 중인) 응용 별로 각 SSL/TLS 플로우의 SSL/TLS 메시지 타입 시퀀스(sequence)를 추출한 뒤 마르코프 체인(Markov Chain)으로 핑거프린팅 및 레이블링(labeling)한 뒤 최대 우도 규준(Maximum Likelihood Criterion)으로 응용을 식별할 수 있음을 보였다. Anderson et al.[23]은 샌드박스에서 수집한 TLS 플로우에서 암호화되지 않은 메타데이터(Metadata)에서 특징 정보를 추출해 핑거프린팅하여 클라이언트와 서버 모두에서 멀웨어 패밀리(Malware Family)를 분류할 수 있음을 보였다. Anderson and McGrew[24]는 TLS 플로우의 메타데이터를 해당 플로우와 시간 또는 속성 차원에서의 맥락을 공유하는 (암호화되지 않은) HTTP 플로우 및 DNS 플로우의 메타데이터도 연관시킨 뒤 통계적으로 특징 선택을 수행하는 핑거프린팅 및 분류 기법이 높은 정확도로 악성/양성 여부를 판정해냄을 보였다. 따라서 적절한 SSL/TLS 핑거프린팅 기법을 통해 핑거프린트 데이터베이스를 위협 정보로 공유할 경우, 페이로드 암호화도 인한 가시성 제약 상황에서도 효과적인 사이버 보안 인텔리전스 구축이 가능할 것으로 기대된다.

2.3. OSINT를 위한 SSL/TLS 핑거프린팅 기법

한편, SSL/TLS 핑거프린팅에 기반을 둔 다양한 연구들이 존재함에도 불구하고, 공개 출처 위협 정보(OSINT)를 일반 공개한 SSL/TLS 핑거프린팅 기법은 상대적으로 소수에 불과하다. 이는 일반성을 가진 핑거프린트 데이터베이스 구축의 어려움과 데이터베이스의 경제적 가치 등이 다양하게 작용한 것으로 여겨진다.

이러한 기법들 중 초기 기법은 Qualys SSL Labs의 Ivan Ristic이 2009년에 공개한 HTTPS 클라이언트 핑거프린팅을 지원하는 Apache 웹 서버 모듈인 mod_sslhaf[25]이다. mod_sslhaf는 HTTPS 요청 메시지의 헤더 정보 중 하나인 User-Agent에 운영체제 및 웹 브라우저에 관한 정보가 담겨있는 점과 SSL/TLS의 메타데이터 중 지원 가능한 암호화스위트(ciphersuite) 목록이 SSL/TLS 라이브러리와 운영체제에 따라 달라진다는 점에 주목하여 클라이언트에 대한 핑거프린팅 데이터베이스를 구축, 공개하였다. mod_sslhaf 및 핑거프린팅 데이터베이스는 2014년 이후 업데이트 되고 있지는 않지만, Qualys SSL Lab에서는 SSL/TLS 클라이언트(웹 브라우저)의 취약점을 분석하는 테스트 서비스를 제공하고 있으며 SSL/TLS 서버의 테스팅을 위한 API인 ssllabs-scan을 제한된 라이센스 하에 무료로 제공하고 있다.

다음으로, Marek Majkowski[26]는 수동적 TCP/IP 핑거프린팅 도구인 p0f v3.0에 SSL/TLS 핑거프린팅을 위한 패치를 2012년에 발표하면서 핑거프린트 데이터베이스를 공개하였다. 이 패치는 SSL 버전, SSL/TLS 클라이언트의 암호화스위트 리스트, SSL/TLS 확장 리스트, 기타 SSL/TLS 플래그 정보를 수집한다는 점에서 SSL/TLS의 메타데이터를 상당 부분 수집한다. 하지만 해당 패치와 데이터베이스는 초기 공개 이후 업데이트 되고 있지 않다.

2015년에 공개된 Lee Brotherston[27]의 FingerPrinTLS는 UNIX 계열 운영체제에서 실시간 패킷 스니핑과 함께 클라이언트(웹 브라우저)의 User Agent를 식별하는 TLS 핑거프린팅을 수행하는 도구이다. Brotherston은 대량의 데이터를 처리하는 과정에서 TLS의 Client Hello 메시지만을 빠르게 판별해내는 Berkeley Packet Filter를 발견하여 처리 성능을 개선하였으며, 그림 1과 같은 JSON 형식의 핑거프린

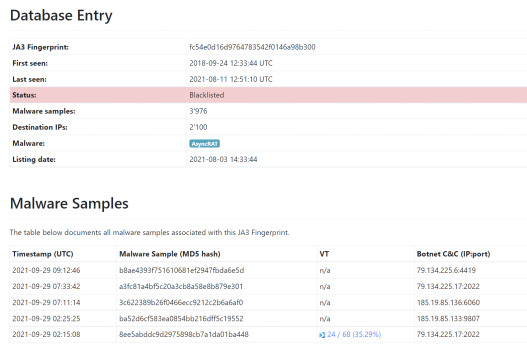
```
{
  "id": 0,
  "desc": "AppleWebKit/600.7.12 or 600.1.4",
  "record_tls_version": "0x0301",
  "tls_version": "0x0303",
  "ciphersuite_length": "0x004A",
  "ciphersuite": "0x00FF 0xC024 0xC023 0xC00A
0xC009 0xC008 0xC028 0xC027 0xC014 0xC013
0xC012 0xC026 0xC025 0xC005 0xC004 0xC003
0xC02A 0xC029 0xC00F 0xC00E 0xC00D 0x006B
0x0067 0x0039 0x0033 0x0016 0x003D 0x003C
0x0035 0x002F 0x000A 0xC007 0xC011 0xC002
0xC00C 0x0005 0x0004",
  "compression_length": "1",
  "compression": "0x00",
  "extensions": "0x0000 0x000A 0x000B 0x000D",
  "e_curves": "0x0017 0x0018 0x0019",
  "sig_alg": "0x0501 0x0401 0x0201 0x0403
0x0203",
  "ec_point_fmt": "0x00"
}
```


〔그림 1〕 FingerPrinTLS의 JSON 핑거프린트

트를 출력한다. FingerPrinTLS의 핑거프린트 데이터베이스는 2017년 3월에 마지막으로 업데이트되었다.

미국의 클라우드 기반 소프트웨어 기업인 Salesforce의 세 엔지니어 John B. Althouse, Jeff Atkinson, Josh Atkins는 기존 연구[26][27]에 영감을 받아 클라이언트의 멀웨어 여부를 탐지하기 위한 JA3[28]라는 MD5 해시 기반 TLS 핑거프린팅 기법을 2017년에 제안하였다. JA3는 Brotherston과 비슷하게 TLS의 Client Hello 메시지로부터 SSL 버전, 암호화 스위트 리스트, 확장 리스트, 타원 곡선 확장 정보를 추출해 MD5 해시로 저장하는 방식으로 핑거프린팅하는 것이 특징이다.

JA3 개발자들은 커뮤니티 친화적으로, JA3 구현을 (Bro[29]의 후속인) Zeek와 파이썬과 연동되도록 하였으며, 웹 브라우저의 User Agent 뿐만 아니라 Apple OS X와 Linux의 응용 프로세스에 대한 JA3 해시값 데이터베이스를 공개하였다. 또한 2019년에는 Client Hello 메시지만을 핑거프린팅하는 JA3의 한계를 극복하고자 서버의 TLS 핑거프린팅을 위한 기법인 JA3S[30]를 발표하였다. 이러한 이유로 JA3는 Abuse.ch를 포함한 여러 OSINT 취득 및 관리 시스템



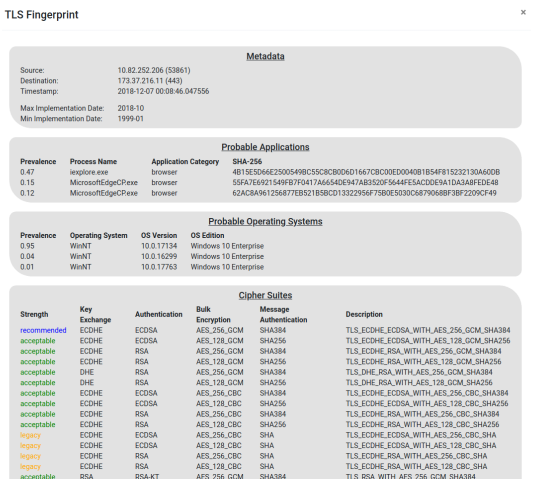
〔그림 2〕 저명한 OSINT인 Abuse.ch에서 제공하는 멀웨어에 대한 SSL 블랙리스트 예시 페이지. 상단에 JA3 핑거프린트를 관찰할 수 있다.

에서 활용되고 있다.

지금까지 소개한 TLS 핑거프린팅 기법들은 보안 커뮤니티를 위해 초기 데이터베이스와 함께 오픈 소스화를 진행하여 많은 주목을 받았다. 하지만 이 기법들은 학술 연구와는 독립적으로 실무 경험을 바탕으로 제안된 기법이라는 점에서 한계가 있으며, 특히 충분한 TLS 트래픽 데이터셋이 없는 상태에서 이론과 성능 평가 없이 공개되었다는 점으로 인해 후속 연구 또는 공정한 성능 평가가 필요한 상황이다.

반면 2.1절에서 기 소개한 Cisco의 Advanced Security Research Group(ASRG)에서 발표한 연구인 Anderson et al.[23], Anderson and McGrew[24]에서 사용된 TLS 핑거프린팅 기법을 위한 오픈 소스 소프트웨어 패키지인 Cisco Joy[31]는 구체적인 통계 자료와 성능 평가 결과가 공개되었다는 점에서 높은 잠재력을 가지고 있다. TLS 핑거프린팅 기법 측면에서, Anderson and McGrew[24]는 네트워크 프로토콜에 대한 전문적인 이해를 바탕으로 Joy에 TLS 플로우에서 수집가능한 거의 모든 메타데이터, 플로우를 구성하는 패킷의 패킷 길이 시퀀스와 도착간 시간(Sequences of Packet Length and Time) 시퀀스, 및 암호화된 페이로드 데이터의 바이트 분포(Byte Distribution)를 수집할 수 있는 기능을 구현하였다. 또한 샌드박스에서 각 멀웨어를 실행해서 정확하게 레이블된 멀웨어 데이터셋과 엔터프라이즈 네트워크에서 확보한 정상 데이터셋을 확보하였다. 이를 바탕으로 통계적 방법에 기반을 둔 특징 선택 기법을 적용하는 TLS 핑거프린팅 기법은 멀웨어 탐지라는 특화된 상황에 최적화되어 있다고 할 수 있다.

또한 2019년에 발표된 연구인 Anderson and


TLS Fingerprint

Metadata

Source: 10.82.252.206 (53861)
Destination: 173.37.216.11 (443)
Timestamp: 2018-12-07 00:08:46.047556

Max Implementation Date: 2018-10
Min Implementation Date: 1999-01

Probable Applications

Prevalence	Process Name	Application Category	SHA-256
0.47	iexplore.exe	browser	4B15ED66E2500549BC55C8C80D6D1667CBC00ED0040B1B54F815232130A60DB
0.15	MicrosoftEdgeCP.exe	browser	55FA7E6921549FB7F0417A6654DE947AB3520F5644FE5ACDDE9A1DA3A8FEDE48
0.12	MicrosoftEdgeCP.exe	browser	62AC8A961256877EB521B5BCD13322956F75B0E5030C6879068BF3BF2209CF49

Probable Operating Systems

Prevalence	Operating System	OS Version	OS Edition
0.95	WinNT	10.0.17134	Windows 10 Enterprise
0.04	WinNT	10.0.16299	Windows 10 Enterprise
0.01	WinNT	10.0.17763	Windows 10 Enterprise

Cipher Suites

Strength	Key Exchange	Authentication	Bulk Encryption	Message Authentication	Description
recommended	ECDHE	ECDSA	AES_256_GCM	SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
acceptable	ECDHE	ECDSA	AES_128_GCM	SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
acceptable	ECDHE	RSA	AES_256_GCM	SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
acceptable	ECDHE	RSA	AES_128_GCM	SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
acceptable	DHE	RSA	AES_256_GCM	SHA384	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
acceptable	DHE	RSA	AES_128_GCM	SHA256	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
acceptable	ECDHE	ECDSA	AES_256_CBC	SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
acceptable	ECDHE	ECDSA	AES_128_CBC	SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
acceptable	ECDHE	RSA	AES_256_CBC	SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
acceptable	ECDHE	RSA	AES_128_CBC	SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
legacy	ECDHE	ECDSA	AES_256_CBC	SHA	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
legacy	ECDHE	ECDSA	AES_128_CBC	SHA	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
legacy	ECDHE	RSA	AES_256_CBC	SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
legacy	ECDHE	RSA	AES_128_CBC	SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
acceptable	RSA	RSA-KT	AES_256_GCM	SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384

〔그림 3〕 Cisco Joy를 통해 확보할 수 있는 TLS 핑거프린트 정보

McGrew[32]에서는 기존 연구를 통해 확인한 주요 메타데이터 특징을 선정해 (정보의 손실이 발생하는 MD5 해시가 아니라) 복원가능한 포맷으로 요약하고, 플로우를 생성한 실행 파일의 SHA-256 해시값으로 레이블링 한 데이터 통합 기반 TLS 클라이언트 핑거프린팅 기법을 제안하였다. 그림 3은 해당 핑거프린팅 기법을 통해 확인할 수 있는 정보의 예시이다.

Cisco ASRG는 Joy의 프론트엔드가 파이썬 2로 작성되어 최신 운영체제에서의 호환성 문제가 발생하는 걸 개선하고자, 파이썬 3으로 작성된 새 소프트웨어 패키지인 Mercury[32]를 발표하였다. Mercury에는 TLS 핑거프린팅을 위한 naive Bayes 분류기 구현도 포함되어 있으며 그 구현 상세와 성능 평가 결과는 Anderson and McGrew[33]에서 확인할 수 있다. 또한 Joy에서부터 지속적으로 업데이트하던 TLS 핑거프린트 데이터베이스를 Mercury 포맷에 맞추어 업데이트를 하였다. 2021년 9월 현재 기준으로 최신 업데이트는 2020년 9월에 이루어졌다.

Ⅲ. 결 론

지금까지 사이버 위협 인텔리전스 구축 시 활용할 수 있는 네트워크 프로토콜 침해지표인 TLS 핑거프린트의 생성 기법에 대해 소개하였다. 현재 OSINT에서는 실무 경험을 바탕으로 제안된 TLS 핑거프린팅 기법이 주류를 이루고 있는 상황임을 확인할 수 있었으며, Cisco에서 공개한 연구 결과들은 높은 학술적 검증 수준으로 인해 잠재력이 높다고 할 수 있지만, 타 보안 커뮤니티에서의 활용은 아직까지 제한적인 상황이다. 따라서 현존하는 여러 TLS 핑거프린팅 기법이 사이버 보안 인텔리전스 구축에 어떠한 장단점을 가지는지에 대한 후속 연구가 진행되어야 할 것이다.

참 고 문 헌

[1] R. McMillan, *Definition: Threat Intelligence*, Gartner Research, May 2013. URL: https://www.gartner.com/doc/2487216/definition-threat-intelligence

[2] C. Johnson, L. Badger, D. Waltermire, J. Snyder, and C. Skorupka, *Guide to Cyber Threat Information Sharing*, NIST Special Publication 800-150, National Institute of Standards and Technology, October 2016.

[3] 김대건, 백승수, 유동희, "사이버위기에 대응하기 위한 국가정보기관의 사이버위협정보 공유 역할에 대한 고찰," *디지털융복합연구*, 15(6), pp. 51-59, 2017.

[4] K. Paine, O. Whitehouse, and J. Sellwood, "Indicators of Compromise (IoCs) and Their Role in Attack Defence," draft-pain e-smart-indicators-of-compromise-03, Internet-Draft, July 12, 2021.

[5] K. Thomas, R. Amira, A. Ben-Yoash, O. Folger, A. Hardon, A. Berge, E. Bursztein, and M. Bailey, "The Abuse Sharing Economy: Understanding the Limits of Threat Exchanges," in *Proc eedings of the 19th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*, September 19-21, 2016.

[6] A. Zibak and A. Simpson, "Cyber Threat Information Sharing: Perceived Benefits and Barriers," in *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES)*, August 2019.

[7] 김경한, 이슬기, 김병익, 박순태, "OSINT기반의 활용 가능한 사이버 위협 인텔리전스 생성을 위한 위협 정보 수집 시스템," *정보보호학회지*, 29(6), December 2019.

[8] P. Gao, X. Liu, E. Choi, B. Soman, C. Mishra, K. Farris, and D. Song, “A System for Automated Open-Source Threat Intelligence Gathering and Management,” in *Proceedings of the 2021 ACM SIGMOD/PODS Conference*, June 20-25, 2021.

[9] G. S. Poh, D. M. Divikaran, H. W. Lim, J. Ning, and A. Desai, “A Survey of Privacy-Preserving Techniques for Encrypted Traffic Inspection over Network Middleboxes,” *arXiv* 2101.04338v1, 2021.

[10] M. Conti, Q. Q. Li, A. Maragno, and R. Spolaor, “The Dark Side(-Channel) of Mobile Devices: A Survey on Network Traffic Analysis,” *IEEE Communications Surveys & Tutorials*, 20(4), pp. 2658-2713, Fourth Quarter 2018.

[11] P. Dimou, J. Fajfer, N. Muller, E. Papadogiannaki, E. Rekleitis, and F. Strasak, Encrypted Traffic Analysis: Use Cases & Security Challenges, European Union Agency for Cybersecurity (ENISA), November 2019.

[12] K. C. Claffy, H.-W. Braun, and G. C. Polyzos, “A Parameterizable Methodology for Internet Traffic Flow Profiling,” *IEEE Journal of Selected Areas in Communications*, 13(8), pp. 1481-1494, October 1995.

[13] Fyodor, Remote OS detection via TCP/IP Stack FingerPrinting, October 18, 1998. URL: https://nmap.org/nmap- fingerprinting-article.txt

[14] R. Beverly, “A Robust Classifier for Passive TCP/IP Fingerprinting,” in *Proceedings of the 5th International Workshop on Passive and Active Network Measurement (PAM)*, April 2004.

[15] M. Zalewski, p0f – passive os fingerprinting tool, BugTraq mailing list, nmap, June 10, 2000. URL: https://seclists.org/bugtraq/2000/Jun/141

[16] R. Barnes, M. Thomson, A. Pironti, and A. Langley, Deprecating Secure Sockets Layer Version 3.0, RFC 8996, IETF, June 2015.

[17] T. Dierks and C. Allen, The TLS Protocol Version 1.0, RFC 2246, IETF, January 1999.

[18] T. Dierks and E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.1, RFC 4346, IETF, April 2006.

[19] T. Dierks and E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, IETF, April 2008.

[20] E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446, IETF, April 2018.

[21] K. Moriarty and S. Farrell, Depreca- ting TLS 1.0 and TLS 1.1, RFC 8996, IETF, March 2021.

[22] M. Korczynski, and A. Duda, "Markov Chain Fingerprinting to Classify Encry- pted Traffic," in *Proceedings of the 33rd IEEE International Conference on Computer Communications (INFO- COM)*, April 2014.

[23] B. Anderson, S. Paul, and D. McGrew, “Deciphering malware’s use of TLS (without de-cryption),” *Journal of Computer Virology and Hacking Techniques*, 14(3), pp. 195- 211, August 2018.

[24] B. Anderson and D. McGrew, "Identifying Encrypted Malware Traffic with Contextual Flow Data," in *Proceedings of the 9th ACM Workshop on Artificial Intelligence and Security (AISec)* co-located with ACM CCS, October 2016.

[25] I. Ristic, HTTP Client Fingerprinting Using SSL Handshake Analysis, 2009. URL: https://www.ssllabs.com/projects /client-finger-printing/

[26] M. Majkowski, SSL Fingerprinting for p0f, June 2012. URL: https://idea. popcount.org/2012-06-17-ssl-fingerprinting-for-p0f/

[27] L. Brotherston, TLS Fingerprinting: Smarter Defending & Stealthier Attacking, September 2015. URL: https://blog.squarelemon.com/tls-fingerprinting/

[28] J. Althouse, Open Sourcing JA3: SSL/TLS Client Fingerprinting for Malware Detection, July 2017. URL: https://engineering.salesforce.com/open-sourcing-ja3-92c9e53c3c41

[29] V. Paxson, “Bro: System for Detecting Network Intruders in Real-Time,” in *Proceedings of the 7th USENIX Security Symposium*, Jaunary 1998.

[30] J. Althouse, TLS Fingerprinting with JA3 and JA 3S, January 2019. URL: https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967

[31] Cisco Systems, joy: A Package for Capturing and Analyzing Network Flow Data and Intrafow Data, for Network Research, Forensics, and Security Monitoring, 2016. URL: https://github .com/cisco/joy

[32] B. Anderson and D. McGrew, “TLS Beyond the Browser: Combining End Host and Network Data to Understand Application Behavior,” in *Proceedings of the 2019 ACM Internet Measurement Conference (IMC)*, October 2019.

[33] B. Anderson and D. McGrew, “Accurate TLS Fingerprinting using Destination Context and Knowledge Bases,” Arxiv 2009.01939, September 2020.

〈저자소개〉

노 희 준 (Heejun Roh)

종신회원

2009년 2월 : 고려대학교 컴퓨터학과 졸업

2011년 2월 : 고려대학교 컴퓨터·전파통신공학과 석사

2017년 2월 : 고려대학교 컴퓨터·전파통신공학과 박사

2019년 3월~현재 : 고려대학교 일반대학원 사이버보안학과 조교수

<관심분야> 유무선 네트워크 보안, IoT 보안