소켓 기반 프로세스 수준 네트워크 트래픽 자동 라벨링 프레임워크

김지민, 장윤성, 유경민, 남승우, 김주성, 김명섭*
고려대학교, 고려대학교*

{illiard1209, brave1094, rudals2710, nam131119, jsung0514, tmskim*}@korea.ac.kr,

Socket-based Automatic Labeling Framework For Process-level Network Traffic Classification

Ji-Min Kim, Yoon-Seong Jang, Gyeong-Min Yu, Seung-Woo Nam, Ju-Sung Kim, Myung-Sup Kim*

Korea Univ., Korea Univ.*

요 약

네트워크 트래픽 분류(Network Traffic Classification, NTC)는 네트워크 보안 및 관리의 핵심 기술이며, 고품질 라벨링 데이터셋은 분류 모델의 성능을 결정하는 핵심 요소이다. 그러나 기존 데이터셋 구축 방식은 수동 라벨링에 의존하여 정확성이 떨어지고, 필터링을 하지 않아 무결성이 떨어진다. 또한 단일 호스트 수집에 특화되어 있으며, 호스트의 에이전트 장애 시 트래픽 수집의 연속성이 보장되지 않는다. 본 논문에서는 Windows 호스트의 소켓 상태를 주기적으로 폴링하고 ETW(Event Tracing for Windows)를 통해 보조적으로 네트워크 이벤트를 수집하여, 네트워크 미러링으로 캡처한 PCAP 파일의 각 세션을 해당 프로세스에 자동으로 매칭하는 프레임워크를 제안한다. 다단계 매칭 파이프라인과 후처리를 통해 실제 운영 환경에서 99.8%의 프로세스 매칭률을 달성하였으며, 복수 호스트에 대해 24 시간 연속 수집이 가능함을 확인하였다.

Ⅰ. 서 론

네트워크 트래픽 분류는 네트워크에서 발생하는 트래픽을 응용 프로그램 또는 서비스 단위로 식별하는 기술로, 침입 탐지, QoS(Quality of Service) 관리, 이상 행위 탐지 등 다양한 네트워크 보안 및 관리 영역에서 핵심적인 역할을 수행한다[1]. 최근에는 딥러닝 기반 분류 모델[2][3]이 활발히 연구되고 있으며, 이러한 모델의 성능은 학습에 사용되는 데이터셋의 품질과 다양성에 크게 의존한다.

이러한 네트워크 트래픽 분류 모델을 활용함에 있어서 데이터셋의 구축은 분류 연구의 출발점이자 가장 어려운 과제다. 이상적인 데이터셋은 각 네트워크 세션이 어떤 응용 프로그램에 의해 생성되었는지 정확하게 라벨링되어 있어야 한다. 그러나 모든 패킷에 해당 정보가 포함된 것이 아니므로, 별도의 라벨링 메커니즘이 필요하다. 기존에는 수집 시나리오에 의한 시간으로 자르는 형식이나 IP 대역대로 라벨링하는 방식을 사용하였지만, 해당 방식은 노이즈나 백그라운드 트래픽 발생에 취약하다는 문제점을 가지고 있다.

본 논문은 이러한 문제를 해결하기 위해 자동 라벨링 프레임워크를 제안한다. 해당 프레임 워크는 Windows 호스트에서 소켓 상태를 주기적으로 폴링하는 동시에, ETW 를 통해 네트워크 이벤트 발생 시점의 프로세스 정보를 보조적으로 수집한다. 이렇게 수집된 프로세스-포트 매핑 정보와 네트워크 미러링을 통해 캡처된 PCAP 파일의 세션을 다단계 매칭 전략을 사용함으로써 라벨을 자동으로 부여한다. 이를 통해 수동 개입 없이 실제 운영 환경의 트래픽에 대한 대규모 라벨링 데이터셋을 지속적으로 생성 가능하다.

Ⅱ. 관련연구

NTC 연구에서 데이터셋은 모델의 학습과 평가를 좌우하는 핵심 요소이다. 본 절에서는 기존 주요 NTC 데이터셋의 특성과 라벨링 방식을 분석하고, 이들이 공통적으로 가지는 한계를 논의한다.

ISCX-VPN-NonVPN[4] 데이터셋은 VPN 과 일반 트래픽을 구분하기 위해 구축된 데이터셋이다. 라벨링 시 목표 응용 하나만 실행하고 나머지 모든 서비스와 응용을 종료한 상태에서 캡처하는 방식을 취하여, 특정 시간대의 트래픽 전체를 해당 응용으로 간주한다. 이 방식은 단순하고 직관적이나, 복수의 응용이 동시에 실행되는 실제 환경을 반영하지 못하며, OS 백그라운드 서비스 트래픽의 혼입 가능성이 있고, 매 세션마다 수동 실행이 필요

이 논문은 과기정통부.정보통신기획평가원의 정보통신방송표준개발지원(R&D,정보화) 사업(No. RS-2025-02219319, 양자컴퓨터 공격에도 안전한 양자암호 기반 제로트러스트 보안 네트워크/서비스 및 제어/관리 기술 표준개발) 및 2023 년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원(P0024177, 2023 년 지역혁신클러스터육성)의 지원을 받아 수행된 연구임.

하여 확장성이 제한된다.

CIC-IDS2017[5] 데이터셋은 침입 탐지 시스템 평가를 위한 데이터셋으로, 정상 트래픽과 공격 트래픽을 사전에 정의된 시간대에 따라 구분하여 라벨을 부여한다. 그러나 시간대 기반 라벨링은 공격 시간대에 혼재하는 정상 트래픽의 정밀한 분류가 어렵다.

호스트 기반 패킷 캡처 도구인 Microsoft Network Monitor[6]는 대상 호스트에서 직접 네트워크 트래픽을 캡처하면서 갓 패킷에 프로세스 정보를 태깅할 수 있어 라벨링에 유리하다. 그러나 이 방식은 단일 호스트에서만 동작하며, 고트래픽 환경에서는 캡처 도구 자체의 CPU 및 디스크 I/O 부하로 인해 패킷 손실이 발생할 수 있다. 또한 호스트에 장애가 발생하면 캡처 프로세스도 함께 중단되어 트래픽 수집의 연속성이 보장되지 않는다는 한계가 있다.

정리하면, 기존의 네트워크 트래픽 라벨링 방식은 크게 세 가지로 분류된다. 첫째, ISCX-VPN-NonVPN 과 같이 단일 응용만 실행한 상태에서 시간대 기반으로 라벨을 부여하는 방식, 둘째, CIC-IDS2017 과 같이 사전 정의된 시나리오에 따라 시간 구간별로 라벨을 부여하는 방식, 셋째, Network Monitor 와 같이 호스트에서 직접 패킷을 캡처하며 프로세스 정보를 태깅하는 방식이다. 그러나 이들은 각각 한계를 가진다. 시간대 기반 라벨링은 백그라운드 트래픽의 혼입으로 인해 데이터셋의 무결성이 보장되지 않으며, 단일 응용 실행 방식은 복수 호스트로의 확장이 어렵고 수동 개입이 필요하다. 호스트 기반 캡처는 고트래픽 환경에서의 패킷 손실과 호스트 장애 시 수집 중단이라는 근본적 문제를 가진다. 본 연구는 이러한 문제를 해결하기 위해 네트워크 미러링을 통한 호스트 독립적 트래픽 수집과 소켓 폴링 기반의 자동 프로세스 매칭을 결합한 프레임워크를 제안한다. 이를 통해 호스트 장애와 무관한 안정적 수집, 백그라운드 트래픽까지 정확히 분류하는 무결성, 그리고 복수 호스트로의 용이한 확장성을 동시에 달성하고자 한다.

Ⅲ. 본론

3.1 시스템 개요

본 프레임워크는 크게 네트워크 트래픽 수집 모듈, 소켓 로그 수집 모듈, 매칭 엔진의 세가지 구성 요소로 이루어진다. 네트워크 트래픽은 스위치의 포트 미러링 기능을 통해 대상 호스트의 모든 패킷을 미러링 서버에서 PCAP 형태로 캡처한다. 소켓 로그는 각 Windows 호스트에서 SocketLogger.exe 가 주기적으로 소켓 테이블을 폴링하여 프로세스-포트 매핑 정보를 기록하고 중앙 서버로 전송한다. 매칭 엔진은 수집된 PCAP 세션의 로컬 포트와 소켓 로그를 대조하여 각 세션의 소유 프로세스를 결정한다.

본 연구에서는 라벨링의 단위를 프로세스로 정의하였는데, 그 이유는 다음과 같다. 운영체제는 네트워크 소켓을 프로세스 단위로 관리하며, 각 소켓은 해당 소켓을 생성한 프로세스의 PID(Process ID)에 귀속된다. Windows 의 GetExtendedTcp(Udp)Table API 는 활성 소켓별로 로컬 주소, 원격 주소와 함께 소유 PID 를 반환하므로, 네트워크 세션과 프로세스 간의 대응 관계를 시스템 수준에서 직접 확인할 수 있다. 즉, 프로세스는 네트워크 활동의 최소 식별 가능 단위이자 운영체제가 제공하는 가장 신뢰할 수 있는 라벨링 기준이다. 이는 응용 프로그램 이름이나 서비스명과 같은 상위 수준의 라벨이 PID 로부터 파생 가능하다는 점에서도 실용적이다.

3.2 네트워크 트래픽 미러링

호스트가 연결된 스위치에서 포트 미러링을 설정하여 해당 호스트의 모든 인바운드/아웃바운드 트래픽을 미러링 포트로 복제한다. 미러링 서버는 이를 수신하여 호스트 IP 별로 분류된 PCAP 파일을 생성한다. PCAP 파일은 1 분 단위로 분할 저장되며, 파일명에 타임스탬프를 포함하여 시간순 처리를 가능하게 한다.

미러링 기반 수집은 호스트에 추가적인 부하를 주지 않으면서 전체 패킷을 캡처할 수 있다는 장점이 있다. 또한 호스트 에이전트의 장애와 무관하게 트래픽 수집이

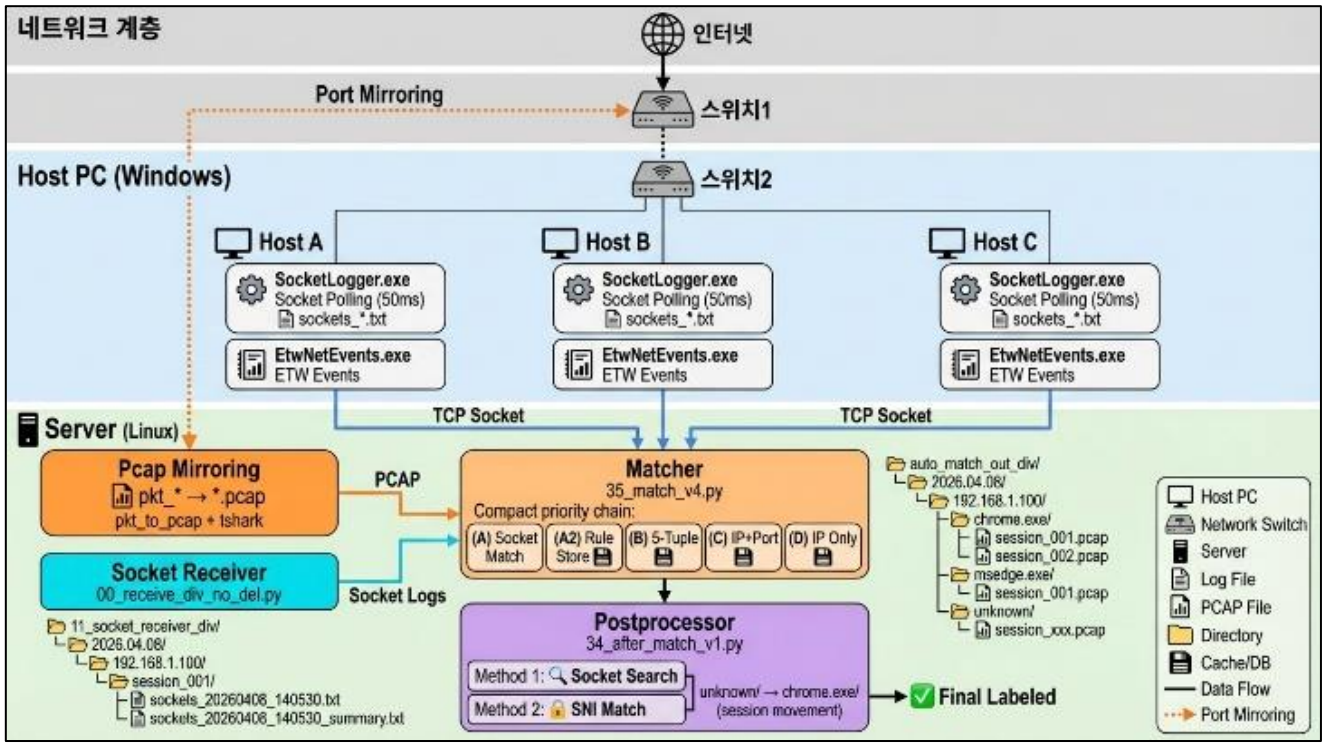

그림 1. 프레임워크 전체 시스템 구조

독립적으로 이루어지므로 시스템 안정성이 높다.

3.3 소켓 로그 수집

각 Windows 호스트에 배포되는 SocketLogger.exe 는 활성화된 모든 TCP/UDP 소켓의 로컬 주소, 원격 주소, PID 및 프로세스명을 수집한다. 수집 주기는 50ms 이며, 한 회의 폴링에 소요되는 시간은 평균 약 1.7ms 로 호스트 성능에 미치는 영향이 최소화된다.

소켓 폴링과 병행하여 ETW[7] 기반의 보조 수집 모듈이 동일 호스트에서 실행된다. ETW 는 커널 레벨에서 네트워크 연결의 생성·종료 이벤트를 실시간으로 캡처하여, 폴링 간격 사이에 생성되고 소멸되는 극히 짧은 수명의 연결에 대한 프로세스 정보를 보완적으로 제공한다. ETW 에서 수집된 PID-프로세스명 매핑은 메모리 내 캐시로 유지되며, 소켓 폴링 로그와 함께 매칭 엔진에 전달된다.

3.4 매칭 엔진

매칭 엔진은 PCAP 파일의 각 패킷에서 5-tuple 을 추출하고, 호스트 측 로컬 포트를 결정한 뒤 다단계 우선순위에 따라 프로세스를 결정한다.

단계	방법	설명
A	소켓 직접 매칭	로컬 포트를 소켓 로그에서 직접 조회하여 프로세스 식별
A2	규칙 저장소	소켓 매칭으로 검증된 원격 IP+포트 기반 규칙 (다중 히트 검증 필요)
B	5-tuple 캐시	동일 5-tuple 의 이전 매칭 결과 재사용
C	원격 IP + 포트 캐시	동일 원격 IP+포트에 대한 이전 매칭 결과 활용 (아웃바운드 전용)
D	원격 IP 캐시	동일 원격 IP 에 대한 통계적 추론 (최소 30 회, 85% 이상 비율 조건)

표 1. 매칭 엔진의 다단계 파이프라인

단계 A 는 가장 정확한 매칭으로, PCAP 타임스탬프에 가장 가까운 소켓 스냅샷을 기준으로 최대 10 개의 이전 스냅샷까지 역추적하여 매칭을 시도한다. 규칙 저장소(A2)는 캐시와 달리 다중 히트 검증 후에만 등록되며, 7 일의 TTL 을 적용하여 오래된 규칙이 자동 만료되도록 한다.

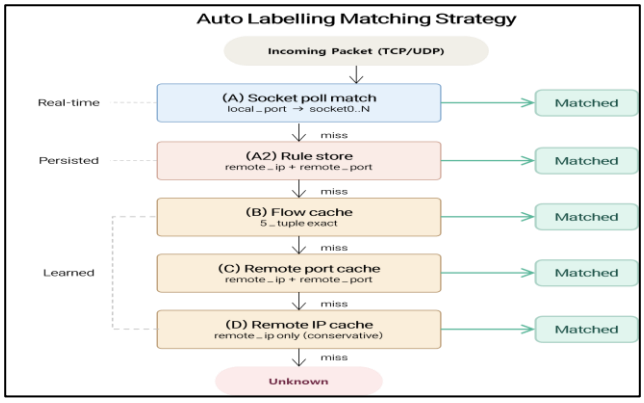

그림 2. 매칭 엔진의 다단계 매칭 흐름도

3.5 후처리

매칭 엔진에서 Unknown 으로 분류된 세션에 대해 두 가지 후처리를 수행한다. 첫째, 소켓 로그 전체 탐색은 해당 호스트의 모든 날짜에 걸친 소켓 로그를 인덱싱하여 로컬 포트와 프로세스의 대응 관계를 통계적으로 구축하고, unknown 세션의 로컬 포트를 조회한다. 이는 매칭 엔진이 참조하는 시간 범위 밖의 소켓 로그에서 정보를 발견할 수 있는 경우에 효과적이다.

둘째, SNI(Server Name Indication) 매칭[8]은 TLS Client Hello 패킷에서 추출한 서버 이름을 기준으로, 이미 라벨링된 세션의 SNI-프로세스 대응 테이블을 학습하여 unknown 세션에 적용한다. 후처리는 30 초 간격으로 지속적으로 실행되며, 매칭 엔진의 실시간 처리와 병행하여 미식별 세션을 점진적으로 해소한다.

Ⅳ. 결론

4.1 실험 환경

실험은 실제 운영 중인 네트워크 환경에서 수행하였다. 복수의 Windows 호스트에 SocketLogger.exe 를 배포하고, 각 호스트의 트래픽을 스위치 미러링을 통해 수집하였다. 수집 서버에서 매칭 엔진과 후처리 모듈을 24 시간 연속으로 운영하며 결과를 측정하였다.

항목	값	비고
수집 호스트 수	7 대	Windows 10/11
수집 기간	24 시간	
전체 세션 수	127,245 개	
전체 응용 수	115 개	Unknown 제외

표 2. 실험 환경 구성

4.2 응용 별 트래픽 분포

수집된 127,245 개 세션에서 총 115 개의 고유 응용 프로그램이 식별되었다. 표 3 은 세션 수 상위 10 개 응용의 분포를 나타낸 것으로, Windows 서비스 호스트(svchost.exe)가 전체의 56.01%로 가장 높은 비율을 차지하였으며, 웹 브라우저(chrome.exe)가 15.77%로 뒤를 이었다. 이어서 원격 접속(sshd.exe), 개발 도구(Code.exe), AI 어시스턴트(Perplexity.exe) 등 다양한 종류의 응용이 관찰되었다. 이는 실제 업무 환경에서 복수의 응용이 동시에 활동하는 트래픽 패턴을 보여주며, 단일 응용만 실행하여 수집하는 기존 방식과의 차별점이다.

순위	응용 프로그램	세션 수	비율(%)
1	svchost.exe	71,273	56.01
2	chrome.exe	20,067	15.77
3	sshd.exe	9,150	7.19
4	Code.exe	4,170	3.28
5	Perplexity.exe	2,996	2.35
6	lghub_agent.exe	1,980	1.56
7	parsecd.exe	1,959	1.54
8	msedgewebview2.exe	1,867	1.47
9	Notion.exe	1,780	1.40
10	System	1,515	1.19
-	기타 105 개 응용	10,228	8.04

표 3. 매칭 세션 내 응용 프로그램 분포

4.3 매칭 결과

실험 결과, 전체 127,245 개 세션 중 실시간 매칭 엔진만으로 114,731 개 세션이 매칭되어 90.17%의 매칭률을 달성하였다. 이후 소켓 전체 탐색 및 SNI 기반 후처리를 적용한 결과, 126,985 개 세션이 매칭되어 최종 매칭률 99.80%를 기록하였다. 표 3 은 처리 단계별 매칭 결과를 요약한 것이다.

처리 단계	매칭 세션	미식별 세션	매칭률
매칭 엔진	114,731 개	12,514 개	90.17%
후처리 적용 후	126,985 개	260 개	99.80%

표 4. 매칭 엔진 및 후처리 매칭률

후처리 전 미식별 세션 12,514 개 중 12,254 개가 후처리를 통해 해소되어, 최종 미식별 세션은 260 개(0.20%)로 감소하였다. 후처리에서 해소된 세션의 대부분은 소켓 로그 전체 탐색에 의한 것으로, 매칭 엔진이 참조하는 시간 범위 밖의 소켓 로그에서 로컬 포트와 프로세스의 대응 관계를 발견한 경우이다. SNI 기반 매칭은 TLS 트래픽에 한정되어 보조적 역할을 수행하였다.

4.4 Unknown 세션 원인 분석

ETW 는 호스트 부하를 최소화하기 위해 모든 이벤트를 기록하지 않으므로, 폴링 간격 사이에 매우 짧게 생성되었다가 종료되는 이벤트들은 수집되지 않을 수 있다. 이처럼 수집되지 않은 이벤트에 해당하는 트래픽은 프로세스와의 매칭이 어려워, 결과적으로 Unknown 으로 분류되는 것으로 확인되었다.

4.5 결론 및 향후 연구

본 논문에서는 Windows 호스트의 소켓 상태 로그를 활용하여 네트워크 세션에 응용 프로그램 레이블을 자동으로 부여하는 프레임워크를 제안하고 구현하였다. 다단계 매칭 파이프라인(소켓 직접 매칭 → 규칙 기반 → 캐시 기반 매칭 → SNI 후처리)을 통해, 후처리를 포함하여 99.8% 이상의 세션 매칭률을 달성하였으며, 수동적 수집 방법에 의존하지 않고도 네트워크 세션을 프로세스 수준에서 라벨링할 수 있음을 보였다.

기존 데이터셋 구축 방법과 비교하여 본 프레임워크는 다음과 같은 차별성을 가진다. 첫째, 통제된 실험 환경이 아닌 실제 운영 환경에서 자동으로 데이터를 수집하므로 트래픽의 다양성과 현실성이 확보된다. 둘째, 소켓 폴링은 호스트에 미치는 영향이 극히 낮아 사용자 체감 성능에 영향을 주지 않는다. 셋째, 미러링 기반 수집과 분리되어 있어 호스트 에이전트 장애 시에도 트래픽 수집이 중단되지 않는 안정적 구조이다. 넷째, PCAP 의 IP 필터링과 SocketLogger.exe 의 배포를 통해 수집 대상을 쉽게 확장할 수 있다.

향후 연구 방향은 다음과 같다. 첫째, 현재 50ms 주기의 소켓 폴링에서 발생하는 약 0.20%의 미식별 세션을 줄이기 위해 폴링 주기 최적화 및 ETW 이벤트 기반 보완 전략을 고도화할 계획이다. 둘째, 본 프레임워크로 생성된 대규모 라벨링 데이터셋을 활용하여 딥러닝 기반 트래픽 분류 모델의 학습 및 성능 평가를 수행함으로써, 자동 라벨링과 분류 모델 간의 선순화 구조를 구축하고자 한다. 셋째, svchost.exe 나 chrome.exe 는 그 안에서 세부적인 서비스로 분화가 가능하기 때문에, 더 세분화된 라벨링 방법론을 연구하여 데이터셋의 정밀도를 향상시킬 계획이다.

참 고 문 헌

[1] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," IEEE Communications Surveys & Tutorials, vol. 21, no. 2, pp. 1988-2014, 2019.

[2] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in Proc. IEEE Int. Conf. Intelligence and Security Informatics (ISI), pp. 43-48, 2017.

[3] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, "MIMETIC: Mobile encrypted traffic classification using multimodal deep learning," Computer Networks, vol. 165, 106944, 2019.

[4] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Encrypted and VPN Traffic using Time-related Features," in Proc. 2nd Int. Conf. Information Systems Security and Privacy (ICISSP), pp. 407-414, 2016.

[5] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP), pp. 108-116, 2018.

[6] Microsoft, "Collect data using Network Monitor," Microsoft Support, 2024. [Online]. Available: https://learn.microsoft.com/en-us/troubleshoot/windows-client/networking/collect-data-using-network-monitor

[7] Microsoft, "Event Tracing for Windows (ETW)," Microsoft Learn, 2024. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/etw/event-tracing-portal

[8] B. Anderson and D. McGrew, "TLS Beyond the Browser: Combining End Host and Network Data to Understand Application Behavior," in Proc. ACM Internet Measurement Conference (IMC), pp. 379-392, 2019.