creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학전문석사 학위 연구보고서

딥러닝을 활용한 금융권 웹 패킷 이상탐지 기법

2023년 2월

서울대학교 공학전문대학원

응용공학과 응용공학전공

정 재 식

딥러닝을 활용한 금융권 웹 패킷 이상탐지 기법

지도교수 권 태 경

이 프로젝트 연구보고서를 공학전문석사 학위
연구보고서로 제출함
2023년 2월

서울대학교 공학전문대학원
응용공학과 응용공학전공
정 재 식

정재식의 공학전문석사 학위 연구보고서를 인준함
2023년 2월

위 원 장 곽 우 영 (인)

위 원 한 훈 (인)

위 원 권 태 경 (인)

국문초록

다른 산업에 비해 금융 산업의 정보 보안은 금융 소비자의 자산과 직결되어 있어 매우 중요하다. 금융권에 대한 공격은 패킷 기반의 공격이 흐름 기반의 공격에 비해 압도적으로 큰 빈도를 차지하고 있다. 이러한 패킷 기반의 공격에 대응하기 위해 일반적으로 패턴 탐지를 위한 룰 세트를 탑재한 침입탐지장비를 활용하여 공격 패킷을 효과적으로 탐지 및 대응하고 있다. 그러나 핀테크 산업의 발전과 다양한 금융서비스가 개발됨에 따라 트래픽의 양은 절대적으로 증가하고 있다. 또한 지금도 지속적으로 새로운 양상의 공격 방법이 출현하고 있다. 이에 대응하기 위해 본 연구보고서에서는 다양한 형태의 아스키 코드 기반의 입력 데이터를 통해 구성한 딥러닝 모델 4가지를 제안한다. 제안한 모델을 활용하여 기존의 침입탐지 체계를 보완하여 패턴탐지 기반의 탐지 방법으로 발견하지 못하던 공격을 찾아내고, 탐지 절차를 효율적으로 개선할 수 있는 방법을 제안하고자 한다.

주요어 : 딥러닝, 아스키 코드, HTTP, 금융권, 패킷 탐지
학 번 : 2021-20708

목 차

제 1 장 서 론 ····· 1
제 1 절 연구 배경 및 목적 ····· 1
제 2 절 연구 범위 및 연구 보고서 구성 ····· 4

제 2 장 배경 지식 및 관련 연구 ····· 6
제 1 절 금융권 네트워크 보안관제 ····· 6
1. 침입 탐지 시스템 ····· 6
2. 금융 부문 통합보안관제 ····· 7
3. 스노트 룰 ····· 8
제 2 절 웹 패킷 공격 탐지 관련 연구 ····· 9

제 3 장 데이터 수집 및 전처리 ····· 14
제 1 절 데이터 수집 ····· 14
제 2 절 HTTP 패킷 추출 ····· 16
제 3 절 아스키 코드 빈도 입력 데이터 ····· 18
제 4 절 아스키 코드 인코딩 입력 데이터 ····· 21

제 4 장 딥러닝 모델 구성 ····· 23
제 1 절 아스키 코드 빈도 모델 ····· 23
제 2 절 아스키 코드 가중치 빈도 모델 ····· 24
제 3 절 아스키 코드 인코딩 모델 ····· 25
제 4 절 아스키 코드 인코딩 합성곱 신경망 모델 ····· 27

제 5 장 실험 및 결과 ····· 29
제 1 절 실험 개요 및 환경 ····· 29
1. 실험 개요 ····· 29
2. 실험 환경 ····· 30
제 2 절 모델별 실험 결과 ····· 30
1. 아스키 코드 빈도 모델 결과 ····· 31
2. 아스키 코드 가중치 빈도 모델 결과 ····· 31
3. 아스키 코드 인코딩 모델 결과 ····· 32
4. 아스키 코드 인코딩 합성곱 신경망 모델 결과 ····· 33
5. 결과 비교 분석 ····· 34
제 3 절 앙상블 모델 적용 ····· 35
1. soft voting 앙상블 모델 ····· 36
2. weighted soft voting 앙상블 모델 ····· 37

제 6 장 결 론 ····· 39
제 1 절 연구 성과 및 활용 방안 ····· 39
1. 의심 패킷 분석 절차 개선 ····· 39
2. 딥러닝에 의한 자동 공격 분석 ····· 40
3. IDS 미탐지 패킷 탐지 ····· 41
제 2 절 연구의 한계와 후속 연구 방안 ····· 44
1. 연구의 한계 ····· 44
2. 후속 연구 방안 ····· 45

참고문헌 ····· 48
Abstract ····· 50

표 목 차

[표 2-1] 스노트 룰 구조 ····· 9
[표 2-2] 스노트 룰 예시 - web parser logic 취약점 ····· 9
[표 2-3] 오 등이 제안한 공격 패킷 분류 체크 리스트 예시 ··· 12
[표 2-4] 주요 관련 연구 ····· 13
[표 3-1] HTTP request 패킷의 구조 ····· 17
[표 4-1] HTTP request 패킷의 통계적 특성 ····· 27
[표 5-1] 실험 환경 내역 ····· 30
[표 5-2] 모델별 학습 결과 값 ····· 33
[표 5-3] soft voting 앙상블 모델 결과 값 ····· 36
[표 5-4] weighted soft voting 앙상블 모델 결과 값 ····· 38
[표 6-1] 딥러닝 모델로 탐지한 IDS 미탐지 패킷 예시 ····· 43

그 림 목 차

[그림 1-1] 금융보안원의 보안관제 프로세스 ····· 2
[그림 1-2] 네트워크 패킷의 구조 ····· 4
[그림 2-1] 금융보안원 통합보안관제 네트워크 개요 ····· 7
[그림 2-2] 우리나라 3선 보안관제 개요 ····· 8
[그림 2-3] 오토인코더 관련 연구에서 제안한 데이터 전처리 방식 10
[그림 2-4] Jiaxin Liu 등이 제안한 블록 시퀀스 개념 ····· 11
[그림 3-1] 아스키 코드를 활용한 입력 데이터 전처리 예시 ·· 19
[그림 4-1] 아스키 코드 빈도 모델 ····· 23
[그림 4-2] 가중치 적용 아스키 코드 빈도 모델 ····· 25

[그림 4-3] 아스키 코드 인코딩 모델 ····· 26
[그림 4-4] 아스키 코드 인코딩 합성곱 신경망 모델 ····· 28
[그림 5-1] 아스키 코드 빈도 모델 학습회차별 정확도 ····· 31
[그림 5-2] 아스키 코드 가중치 빈도 모델 학습회차별 정확도 32
[그림 5-3] 아스키 코드 인코딩 모델 학습회차별 정확도 ····· 33
[그림 5-4] 아스키 코드 인코딩 합성곱 신경망 모델 학습회차별 정확도 34
[그림 5-5] soft Voting 앙상블 모델 예시 ····· 36
[그림 5-6] weighted soft voting 앙상블 모델 예시 ····· 37
[그림 6-1] 관제 분석 화면 개선 예시 ····· 40
[그림 6-2] 딥러닝을 통한 사전 분석 실시 ····· 41
[그림 6-3] 딥러닝을 통한 IDS 미탐지 패킷 탐지 ····· 42
[그림 6-4] 딥러닝 분석을 통한 선순환 운영 구조 ····· 44
[그림 6-5] 학습 기간별 모델간의 비교 ····· 46

제 1 장 서 론

제 1 절 연구 배경 및 목적

사단법인 금융보안원에서는 국내 300여개의 금융기관(은행, 증권사, 보험사 등)으로 유입되는 트래픽을 수집하여 악의를 가진 공격이나 이상 행위 등에 대한 탐지를 수행하고 있다[3]. 여타의 산업분야와 비교할 때 금융권을 향한 악의적인 공격은 금융 소비자에게 직접적인 금전적 손실을 야기할 수 있다. 따라서 금융 분야에서의 네트워크를 통한 악의적인 공격을 탐지하는 일은 매우 중요하다.

이러한 탐지활동은 크게 흐름 기반 탐지 방식(flow based methods)과 패킷 기반(packet based methods) 탐지 방식으로 구분되고 있다[11]. 두 방식은 각각 다른 영역의 정보를 사용하여 탐지하는데, 흐름 기반 탐지의 경우 트래픽의 메타데이터(크기, 시간, 빈도 등)를 사용하며 패킷 기반 탐지는 패킷 자체의 내용(주로 payload)을 기반으로 이상 패킷을 탐지하게 된다. 금융보안원에서 탐지하고 있는 공격은 대부분 패킷 기반 탐지 방식으로 찾아내고 있으며, 흐름 기반 탐지의 경우 디도스[1] 공격과 같이 네트워크의 흐름에 의한 공격의 탐지에 주로 사용된다.

패킷 기반의 공격은 그 종류와 공격 양상이 매우 다양하여 수만 가지 이상의 방법으로 공격이 이루어지며, 대부분 룰 기반(rule based) 탐지로 이루어진다. 룰 기반 탐지는 패킷에 특정 문자열 패턴의 일치 여부를 통해 공격을 탐지하는 방식인데 이러한 룰 세트(rule set) 리스트는 침입탐지 시스템 제조사가 제공하거나 공격별로 분석을 통해 기관별로 만들어서 사용하기도 한다.

공격은 끊임없이 변형되며 계속 진화하는데 이를 대응하기 위해 새로

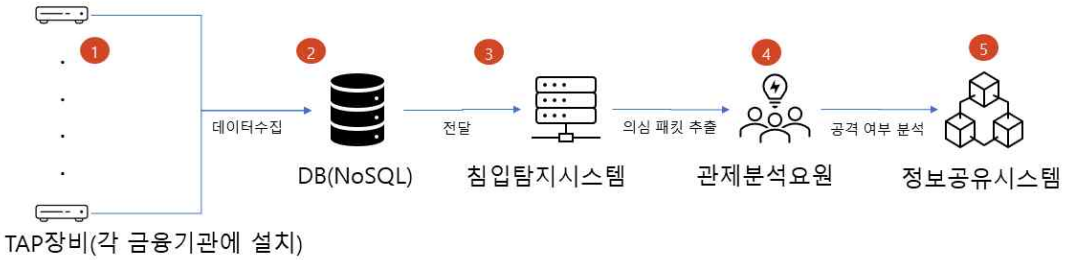
---

1) distributed denial of service(DDoS): 분산된 공격자를 배치하여 공격 대상 시스템의 리소스를 고갈시키는 서비스 거부 공격을 동시에 수행하는 방법

운 공격이 나타날 때 마다 새로운 룰을 만들어서 침입탐지시스템에 적용해야한다. 따라서 이와 같은 룰 기반 탐지의 경우 기존의 룰 세트에 없던 새로운 공격[2]이 나타나면, 그 공격이 성공하여 일반에 공개되기 전까지는 룰을 만들 수 없어서 예방 활동이 어렵다.

본 연구에서는 기존에 공격으로 분석된 데이터와 정상 데이터를 활용하여 지도학습 기반의 딥러닝 기반의 모델을 생성하여 별도의 룰 세트가 없어도 공격 여부를 탐지할 수 있는 방법을 제안하고자 한다. 네트워크 트래픽의 페이로드[3]를 분석하여 공격 여부를 판단할 수 있는 여러 가지 딥러닝 모델의 구현 방법과 해당 모델로 탐지를 수행한 결과를 소개할 것이다.

본 연구의 목적은 대량으로 유입되는 금융권의 네트워크 패킷에 대응하기 위해 금융보안원의 보안 관제 프로세스에 딥러닝 모델을 적용하여 관제의 효율성을 전반적으로 향상시키는 것이다. 본 연구의 목적에 대해 구체적으로 설명하기 이전에 현재 금융보안원의 보안 관제 수행 절차를 파악하여 그 개선 방안을 검토하고자 한다.



[그림 1-1] 금융보안원의 보안관제 프로세스

금융보안원의 보안관제 프로세스는 [그림 1-1]과 같이 요약할 수 있다. 우선 각 금융회사에 설치된 test access point(TAP)[4] 장비를 통해

2) 이러한 공격을 제로 데이 공격(zero-day attack)이라 한다.
3) 페이로드(payload): 전송의 근본이자 목적인 데이터를 말한다. 예를 들어 편지봉투에 적히는 주소와 우체국 등의 정보가 패킷의 헤더와 메타데이터라면 실제 편지봉투에 들어있는 편지 자체가 페이로드이다.
4) test access point : 네트워크의 모니터링을 위해 설치된 구간의 모든 트래픽

금융회사의 인터넷 유입구간(방화벽 외부)의 트래픽을 모두 복사하여 금융보안원으로 전송한다. 전송된 패킷들은 구조화된 빅데이터 솔루션[5]에 저장되고 이를 침입탐지 시스템[6]으로 전송한다. 침입탐지 시스템에서는 패킷 내에 존재하는 패턴을 룰 세트 리스트 기반으로 탐지하여 공격으로 의심되는 패킷을 정리하여 분류한다. 이렇게 정리된 리스트는 금융보안원 보안관제 요원들이 분석하여 최종적으로 공격 패킷인지 아니면 정상 패킷인지 여부를 기록한다. 마지막으로 기록된 공격 시도에 대한 내용은 실시간으로 금융보안원의 정보공유 시스템을 통해 각 금융기관 보안 담당자에게 전파된다.

보안관제 프로세스를 살펴보면 보안관제 요원이 최종적으로 공격 여부를 판단하므로 보안관제 요원의 기술적 숙련도에 따라 효율성 및 정확도에 차이가 생기게 된다. 이를 보완하기 위해 딥러닝을 기반으로 분석된 정보를 제공하여 전체적인 효율성과 정확도를 높이고자하는 것이 본 연구의 첫 번째 목적이다.

또한 침입탐지 시스템의 경우 사전에 정의된 패턴에 일치하는 패킷들만 의심패킷으로 분류하고 그 외의 패킷은 모두 정상패킷으로 처리한다. 따라서 조금만 변형된 공격이 들어와도 탐지하지 못하게 된다. 정상패킷으로 처리되는 건수는 매우 많아(일간 60억여건) 사람이 다시 확인하기에는 물리적으로 불가능한 상황이다. 이렇게 침입탐지 시스템이 정상으로 분류한 패킷들에 대해 본 연구에서 제안한 모델을 적용하여 딥러닝을 통해 다시 한 번 검토할 수 있도록 만드는 것이 본 연구의 또 하나의 목적이다.

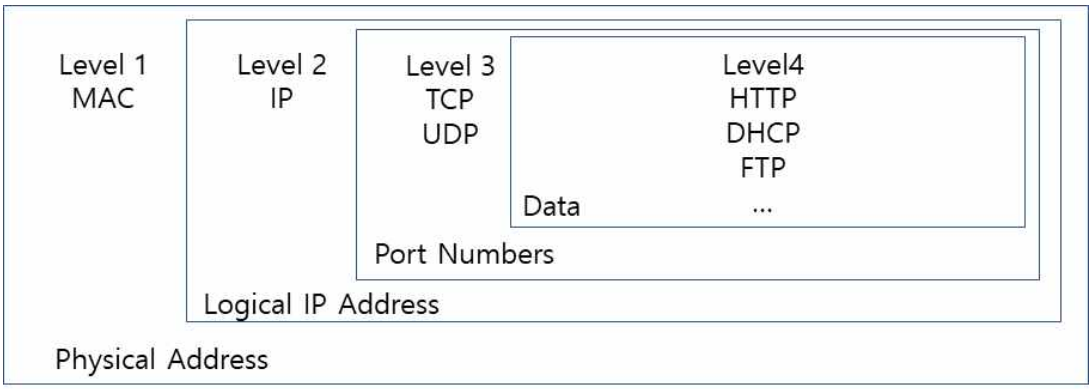
을 복제하여 모니터링 장비로 전송해 준다.

5) 빅데이터 솔루션은 기존의 SQL 기반의 데이터베이스보다 입력시간을 빠르게 수행해야 하므로 주로 No-SQL DB를 사용하며 금융보안원은 Splunk 솔루션을 사용 중이다.

6) 침입 탐지 시스템(intrusion detection system) : 줄여서 IDS 혹은 IPS(intrusion prevention system)라고 부른다. 정규표현식 기반의 패턴 탐지를 위한 룰을 기반으로 네트워크 패킷의 이상행위를 탐지한다.

제 2 절 연구 범위 및 연구 보고서 구성

앞서 1절에서 설명한 바와 같이 네트워크의 공격에 대한 탐지는 흐름 기반 탐지 방식과 패킷 기반 탐지 방식으로 구분되는데 본 연구에서는 패킷 기반탐지 방식을 중심으로 딥러닝 모델을 연구할 것이다. 그 이유는 일반적으로 흐름 기반 탐지 방식은 네트워크 패킷의 메타데이터만 포함하고 있어 금융보안원이 주로 수행하고 있는 침입탐지 시스템의 프로세스와는 다른 영역이며, 이러한 흐름 기반 탐지 방식은 임계값 기반으로 자동화가 많이 이루어져 있기 때문이다.



[그림 1-2] 네트워크 패킷의 구조[1]

패킷 기반 탐지 방식을 사용하려면 네트워크 패킷의 내용을 담고 있는 페이로드 데이터가 필요하다. 페이로드 데이터는 네트워크 표현 계층에 기록되며 거의 대부분 hypertext transfer protocol(HTTP) 형식으로 나타난다. 따라서 본 연구에서는 HTTP 패킷의 내용을 분석 대상으로 정하였다. 실제로 금융보안원 보안 관제 프로세스로 탐지되는 공격 패킷은 대부분 HTTP 패킷 기반의 공격이기도 하다.

네트워크 패킷은 시스템의 내부로 유입되는 패킷과 외부로 유출되는 패킷이 있다. 전자를 대상으로 하는 탐지는 외부로부터 들어오는 공격에 대비를 위함이 주된 목적이며 후자의 경우 내부 사용자의 이상 행위 혹은 기 설치된 악성코드의 행위에 대한 탐지를 주요 목적으로 한다. 본

연구는 금융권 대상으로 유입되는 공격을 탐지하는 것이 주요 목적이므로 내부로 유입되는 패킷을 대상으로 공격행위를 탐지하는 것으로 범위를 정하였다.

본 연구의 범위를 다시 한 번 정리하자면 패킷을 탐지하는 방식은 패킷의 페이로드 데이터를 중심으로 하는 패킷 기반의 탐지를 중심으로 수행할 것이다. 따라서 탐지 대상 패킷은 주로 암호화 되지 않은[7] HTTP 패킷이 된다. 또한 네트워크의 방향은 외부에서 금융기관으로 유입되는 패킷만을 대상으로 한다.

본 장에서는 연구의 배경과 목적 그리고 범위에 대해 기술하였다. 2장에서는 본 연구에 필요한 배경지식과 관련된 선행 연구를 소개하고 3장에서는 연구에 사용할 데이터의 수집 내역과 모델에 적용하기 위한 입력 데이터 형태로 만드는 과정을 소개한다. 4장에서는 3장에서 생성한 데이터 형태를 입력받는 딥러닝 모델에 대해 구성하는 방법을 제안하고 5장에서 해당 모델에 대한 학습 결과를 분석한다. 끝으로 6장에서는 본 연구의 내용을 다시 한 번 정리하고 실제 회사에서 운용 방안 및 효과에 대해 검토하고 마지막으로 연구의 한계점과 후속연구에 대한 아이디어에 대해 설명하고자 한다.

7) 금융보안원에 유입되는 패킷은 사전에 협약된 금융기관과는 복호화 된 패킷을 받고 있지만 그렇지 않은 경우 암호화된 상태로 받게 된다.

제 2 장 배경 지식 및 관련 연구

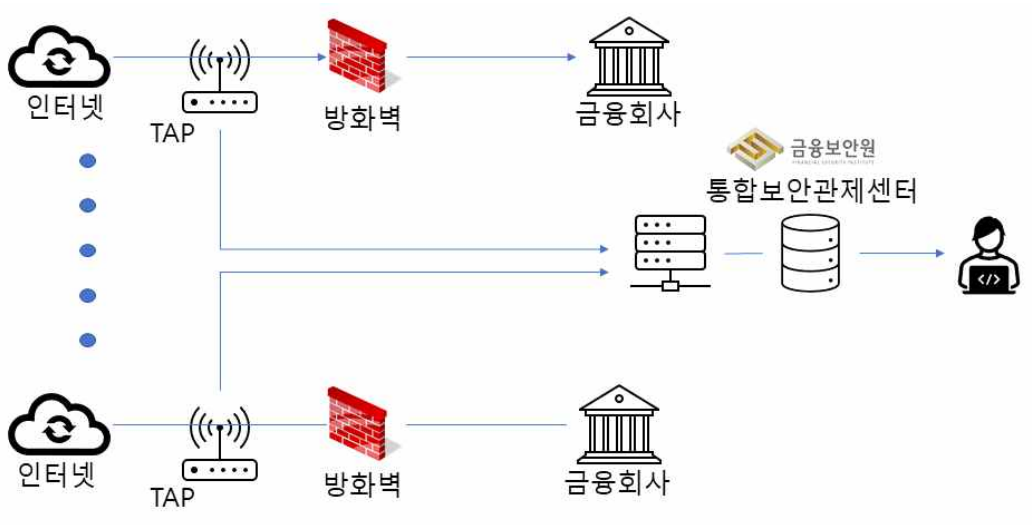
앞서 1장에서는 본 연구의 배경 및 목적 그리고 연구 범위에 대해 개괄적으로 기술하였다. 본 장에서는 금융권 네트워크 보안관제와 관련된 주요 배경지식에 대해 설명한다. 또한 침해위협 공격 패킷에 대응하기 위해 딥러닝을 사용한 연구를 포함한 다양한 선행 연구도 이어서 소개하고자 한다.

제 1 절 금융권 네트워크 보안관제

1. 침입 탐지 시스템

침입 탐지 시스템(intrusion detection system)은 약자로 IDS라고 부르며 유사한 개념으로는 침입 방지 시스템(intrusion prevention system)이라는 용어로도 사용한다. 침입 방지 시스템은 탐지한 침입 공격에 대해 차단이나 패킷을 버리는 등의 방어적인 행위를 포함한 개념이고 대부분의 침입 탐지 시스템도 같은 행동을 취할 수 있기에 두 가지 개념은 크게 차이가 없다.

침입 탐지 시스템은 네트워크의 다양한 구간에 설치할 수 있으며 사용하려는 용도에 따라 설치하는 구간이 정해진다. 금융보안원의 경우 각 금융회사로 유입되는 패킷 중 악의적인 공격의 패킷을 사전에 탐지하는 것을 목적으로 침입탐지 시스템을 사용한다. 따라서 각 금융회사 서버의 방화벽 외부의 인터넷 인입 구간의 패킷을 주 탐지 대상으로 설정하여 운영되고 있다. 방화벽에서 악성 IP에 대해 차단을 하고 있으나 차단된 IP에 대해서도 공격 시도를 탐지하기 위하여 방화벽 외부에서 설치하여 사용하고 있다. 금융권 통합보안관제의 네트워크 구성에 대한 개요는 [그림 2-1]과 같다.



[그림 2-1] 금융보안원 통합보안관제 네트워크 개요

침입탐지 시스템은 다양한 방법으로 네트워크의 이상 신호를 탐지한다. 크게 오용 탐지(misuse detection)와 이상행위 탐지(anomaly based detection)로 나눌 수 있는데[2], 전자의 경우 패킷 기반 탐지 방식으로도 부르며 패킷의 실제 내용에 특정 패턴이 포함되어있는지를 탐지하는 방식이다. 후자의 경우 일상적인 동작이나 흐름과 다른 네트워크의 움직임을 포착하는 방식으로 주로 패킷의 실제 데이터 보다는 메타 데이터를 사용하여 탐지하는 방식이다. 패턴 탐지가 아니라는 점에서 딥러닝을 활용한 탐지 또한 이상행위 탐지의 일종이라 할 수 있다. 금융보안원의 보안관제센터는 두 가지 탐지 방법을 모두 적용하고 있다. 단, 이상행위 탐지의 경우 대부분 자동화 되어있고 임계값 기반으로 처리할 수 있어 별도의 인력이 항상 투입될 필요는 없다. 그에 비해 오용 탐지의 경우 패턴에 의해 탐지된 내용에 대해 다시 한 번 확인이 필요하여 관제 요원이 추가로 분석을 수행하고 공격 여부를 판단한다.

2. 금융 부문 통합보안관제

우리나라의 보안관제는 3선 관제로 구성되어 있다[3]. 가장 상위에 국

가보안기관으로 국가사이버안보센터가 있으며 2선으로는 금융, 의료, 행정, 지방자치단체 등 부문 별로 구성된 부문 관제 기관이 있다. 마지막으로 3선에 개별 금융기관, 행정기관, 민간 회사 등으로 구성되어 있다. 3선 보안관제의 구성은 계층 형으로 구성되어 있으며 [그림 2-2]에서 개괄적인 형태를 확인할 수 있다.

[그림 2-2] 우리나라 3선 보안관제 개요

금융보안원의 경우 금융 부문 2선 보안 관제를 담당하고 있으며 300여개의 금융권의 회사들(은행, 증권회사, 보험사 등)의 부문 관제를 담당하고 있다. 이를 수행하기 위하여 [그림2-1]과 같이 금융회사로 유입되는 네트워크 패킷을 수집하여 365일 24시간 사이버 공격에 대해 감시 및 대응을 수행하고 있다.

3. 스노트 룰

스노트(Snort)는 침입탐지 시스템에 가장 널리 적용하여 사용하고 있는 오픈소스 기반의 프레임워크이다. 1998년 소스파이어(Sourcefire)사의 CTO Martin Roesch에 의해 발표되었으며 2013년에 소스파이어사가 시스코(Cisco)사에 인수되어 현재는 시스코사가 소유권을 가지고 있다. 오용 탐지에 가장 많이 사용되고 있으며 금융보안원 또한 현재 이 솔루션을 사용하고 있다. 스노트에서 사용하는 탐지를 위한 규칙을 스노트 룰이라고 부르며 시그니처라고 부르기도 한다.

스노트 룰은 크게 룰 헤더와 룰 옵션으로 구성되며 그 기본 구조는 [표 2-1]과 같다.

[표 2-1] 스노트 룰 구조[4]

rule header	action	패턴 일치 시 수행하는 작업 종류 ex: alert, pass, drop, reject, log 등
	protocol	통신 프로토콜 종류 ex: tcp, udp, icmp, ip, any 등
	IP port	출발지 IP – port 도착지 IP - port
rule option	msg	로그에 기록되는 내용
	content	패킷에서 탐지하는 패턴

룰 헤더 부분은 주로 패킷의 메타데이터를 기반으로 필터링하는 내용이 들어가 있으며 실제 중요한 탐지를 위한 패턴은 룰 옵션의 content항목에 기록된다. 위의 규칙에 맞추어 구성된 스노트 룰에 대한 예시는 [표 2-2]와 같은 형태로 갖추고 있다.

[표 2-2] 스노트 룰 예시 - web parser logic 취약점

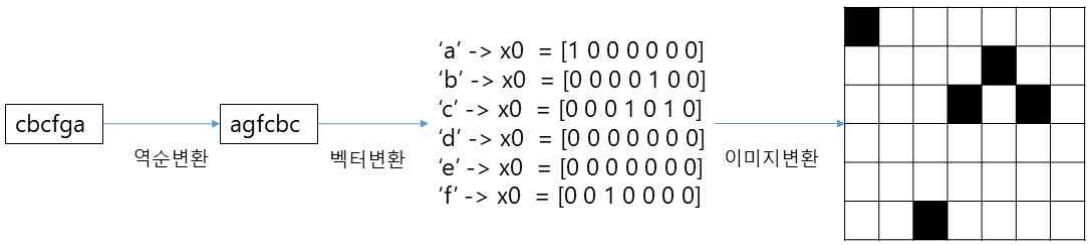
```
alert tcp any any -> any any ( msg:
"F-INV-APP-180809-Web_Parser_Logic_Vulnerability";
flow:to_server,established; uricontent:"/..|3b|/"; sid:1101011; rev:1;)
```

제 2 절 웹 패킷 공격 탐지 관련 연구

네트워크 패킷을 분석하여 공격을 탐지하는 기존 연구는 다양하게 수행되어 왔다. 일반적으로 딥러닝을 활용하여 모델을 만들 때 가장 중요한 일은 풍부하고 다양한 데이터를 확보하는 일이다. 일반적으로 네트워크 트래픽에서 정상 트래픽은 많이 획득할 수 있으나, 공격 트래픽의 경우 특별한 장비와 기술을 가진 인력이 선별해 내지 않으면 확보하기 어렵다. 이런 한계를 극복하기 위해 비지도 학습을 통해 공격행위를 식별하는 연구가 다수 존재한다.

비지도 학습의 연구는 대부분 오토인코더를 이용하여 수행한다. 대표적인 사례로 박 등은 패킷 탐지를 위해 문자열을 이진 이미지로 변환하여 2차원 합성곱 신경망 기반의 오토인코더를 사용하여 공격을 탐지하는 방법을 제안하였다[5]. 위양성률(false positive rate)을 주요 측정지표로 정하였는데 약 0.03 수준의 결과를 도출했다.

해당 연구에서 사용한 기법은 일반적인 형태의 이차원의 합성곱 신경망 오토인코더를 사용하였다. 일반적인 오토인코더를 사용하였으나 입력 문자를 변환하는 방식에 초점을 맞추었다. 우선 문자열을 역순으로 변환하고 원 핫 인코딩을 응용하여 2차원의 데이터로 변환하고 이를 오토인코더의 입력값으로 이용한다. [그림 2-3]에서 이 과정을 간략히 도식화하였다.

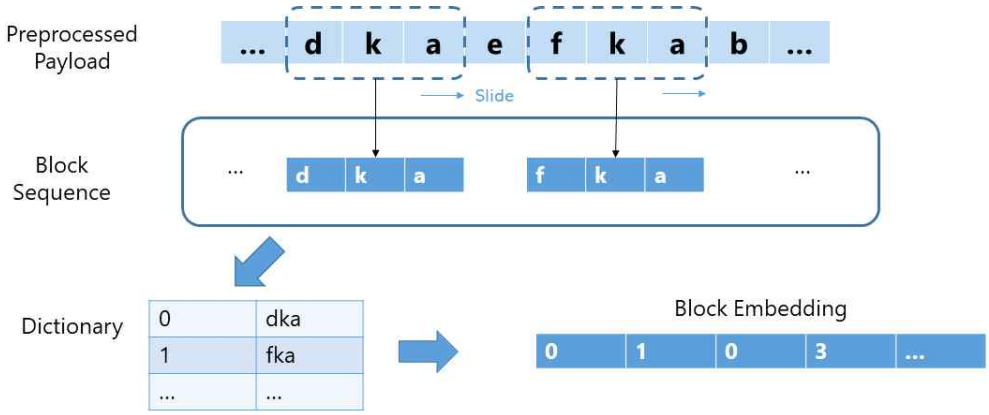


[그림 2-3] 오토인코더 관련 연구에서 제안한 데이터 전처리 방식[5]

이외에도 Vartouni 등도 오토인코더를 사용하여 페이로드를 분석하는 연구를 수행하였으며[20] 대부분의 비지도 학습 기반의 연구는 오토인코더를 사용한다. 본 연구에서는 비교적 많은 양의 실제 금융권 트래픽 데이터를 확보한 바, 비지도 학습 기반의 방식은 사용하지 않는다. 또한 사전에 효용성을 확인하기 위해 오토인코더를 사용한 여러 가지 방법을 적용해보았으나 지도학습 기반 모델에 비해 성능이 좋지 않았다.

지도학습을 사용하여 패킷 기반 탐지를 수행한 연구도 다수 존재한다. Jiaxin Liu 등은 본 연구와 동일하게 패킷 기반 탐지를 주 연구 대상으로 패킷 페이로드 분석에 중점을 둔 연구를 수행하였으며, 일반적인 네트워크 패킷이 자연어와는 달리 단어 단위로 명확하게 분리되지 않는 점을 극복

하기 위해 블록 시퀀스라는 개념을 도입하여 몇 개의 문자를 블록으로 구성하여 패킷을 전처리하고 학습을 시도하였다[6]. 해당 연구에서는 문자열을 일정길이의 블록으로 끊어내고 일정 간격으로 탐색하며 블록을 추출해 낸다. 추출한 블록을 사전(dictionary)화시키고 이를 다시 임베딩하여 최종적인 입력값을 사전을 참조하여 생성한다. [그림 2-4]를 통해 이 과정을 간략하게 도식화 하였다.



[그림 2-4] Jiaxin Liu 등이 제안한 블록 시퀀스 개념[6]

입력 문자열을 하나씩 보지 않고 좌우로 연결된 몇 개의 문자열을 같이 검토하여 학습한다는 개념으로 볼 때 일차원 합성곱 신경망과 유사한 접근 방법을 사용하였다. 주요 측정 지표는 재현율을 사용하였으며 공개된 페이로드 데이터를 담고 있는 CSIC 2010 데이터[8]를 대상으로 0.99의 재현율을 달성했다. 이외에도 합성곱 신경망을 사용한 연구는 다수 존재하며[14] 이와 같은 연구에 착안하여 본 연구에서는 일차원 합성곱 신경망을 사용한 모델 또한 시도해 보았다.

HTTP 패킷에 중점을 두고 공격에 대한 분석을 수행한 연구도 다수 존재한다. Kakavand는 HTTP의 페이로드에 등장하는 문자에 대해 주성분(principal component anlysis, PCA) 분석을 통해 그 특성을 파악하고

8) CSIC(Spanish Research Natiounal Council)에서 테스트를 위해 자동으로 생성한 네트워크 트래픽 데이터, 테스트를 위해 임의로 생성한 데이터이다.

공격을 분류하는 연구를 통해 ISCX 2012 데이터 셋에서 97%의 재현율을 달성하였다[15]. Zolotukhin은 n-gram을 사용하여 HTTP 로그 분석을 통해 공격을 탐지하는 방법을 제안하였다[18]. 임의로 생성한 데이터를 사용하였으며 약 99%의 정확도를 도출하였다. Li 등은 HTTP 트래픽에 대해 자연어 처리기반 기술을 적용하여 공격을 분석하였다[19]. 이 연구에서 CSIC 2010 데이터에 대해 99%의 정확도를 달성하였다. Wang 등은 문자의 통계적 분포의 특성을 활용하여 공격을 분류하는 연구를 수행하였다[16]. 본 연구에서 사용한 아스키 코드 빈도 데이터 또한 일종의 입력 데이터의 통계적 특성이지만 본 연구에서는 그 데이터를 가공하여 딥러닝을 적용한 점이 차별화 된 점이다.

그 외 Ingham 등 다양한 연구자들이 HTTP의 분석에 중점을 둔 연구를 수행하였다[17][21]. 그 중 한 가지인 오 등이 수행한 연구는 HTTP 패킷의 헤더정보와 페이로드 일부 키워드의 정보를 바탕으로 보안관제 수행에 있어 공격 유형의 분류를 자동화 하는 방법을 제안하였다[7]. 이 연구는 경험적이며 통계적인 지식을 기반으로 수행한 연구이다.

해당 연구의 내용은 패킷에 대해 수십 가지의 조건을 체크리스트로 만들어 특정 조건에 만족하는 경우에 대해 공격을 분류하는 방식이다. 체크리스트를 확인하며 어떤 공격 분류에 해당하는지 판별하는 데에 의의를 두고 있으며 제안한 조건들의 경우 일반적인 침입 탐지 시스템에서 모두 탐지할 수 있는 내용이다.

[표 2-3] 오 등이 제안한 공격 패킷 분류 체크 리스트 예시

indicator	example
HTTP DDoS	check if the contents-length is more than 500
web shell	check if the words of body are in BL like '@eval' or 'base64-decoding' string
SQL injection	check if uri has the DB Query
XSS	check if the payload has abnormal script
crypto currency	check if the payload has the strings 'submit' ∩ 'Xmrig'∩ 'Xmr-stack' or 'Coinhive'

대부분의 침입 탐지 시스템은 이러한 내용은 모두 자동으로 분류할 수 있기에 침입 탐지 시스템을 보유한 경우 참고할 만한 내용은 없다.

다만, 일반적인 침입 탐지 시스템의 탐지에 대한 개념을 이 연구를 통해 쉽게 이해할 수 있다.

[표 2-4] 주요 관련 연구

연도	저자	기법	데이터 전처리	데이터셋	탐지결과
2014	Zolotukhin et al.	support vector, data mining	n-gram	generated data	Acc 99.2%
2016	Kakavand et al.	PCA	Mahalanobis Distance Map	ISCX 2012 DARPA 2010	recall 97% recall 100%
2018	Park et al.	CNN, autoencoder	one hot encoding	private data	FPR 3%
2018	Vartouni et al.	stacked autoencoder	n-gram	CSIC 2010	Acc 88.32%
2019	Liu et al.	LSTM, CNN	block sequence	CICIDS 2017 CISX 2012	Acc 99.92% Acc 99.64%
2020	Li et al.	Boosting classification	weighted word2vec	CSIC 2010	Acc 99.49%
2022	Jeong	DNN, CNN	ASCii code frequency, encoding	CICIDS 2017 CSIC 2010 private data	Acc 99.83% Acc 93.50% Acc 99.22%

[표 2-4]에서 앞서 소개된 다양한 관련 연구에 대한 특징을 정리하였으며 마지막 줄에 본 연구의 특징을 기술하였다. 패킷 기반 탐지 방식에 대한 연구는 다양하게 수행되어 왔으며 기존에 수행된 연구들도 서로 유사한 부분이 다수 존재한다.

본 연구가 기존의 연구와 차별이 되는 부분은 문자열의 빈도를 누적 계산하거나 아스키 코드의 값을 사용하는 비교적 간단한 방식으로 입력 데이터를 전 처리하여 모델을 생성한 점이다. 전처리를 상대적으로 단순화하여 실시간으로 유입되는 대용량의 패킷의 분석에 장점을 가진다. 또한 여기에 룰 세트 기반의 가중치를 적용하는 등의 새로운 기법을 적용한 점도 차별점이다. 마지막으로 기존의 연구들이 주로 공개된 테스트용 데이터를 사용하였으나 본 연구에서는 우리나라 금융권의 실제 데이터를 바탕으로 연구를 수행한 점도 차별점이라 할 수 있다.

제 3 장 데이터 수집 및 전처리

앞서 1장과 2장에서 본 연구의 배경을 포함하여 공격 웹 패킷을 탐지하기 위한 선행 연구 사례들을 살펴보았다. 앞서 소개한 연구들과 마찬가지로 본 연구의 목적 또한 금융권의 웹 패킷 중 공격 행위를 포함한 이상 패킷을 탐지하는 것이다. 이를 위해서 가장 먼저 필요한 것은 연구에 사용할 데이터인 웹 패킷이다. 본 장에서는 연구에 사용할 데이터를 수집하고 수집한 데이터를 딥러닝 모델에 적용하기 위하여 가공하는 방법을 논의하고자 한다.

제 1 절 데이터 수집

금융권의 공격을 실시간으로 탐지하는 것이 본 연구의 가장 큰 목적이므로 실제 금융권에서 송수신 되는 데이터를 사용하는 것이 가장 효율적이다. 본 연구에서는 금융보안원 보안관제센터의 양해를 구하여 실제 금융권의 데이터를 대상으로 연구를 수행하였다.

금융보안원으로 유입되는 금융회사들의 패킷은 스플렁크(Splunk)[9]와 같은 No-SQL 기반의 빅데이터 솔루션에 의해 구조화된 데이터로 저장된다. 이 데이터를 활용하여 침입 탐지 시스템이 사전 정의된 룰 세트에 의해 공격 의심 패킷을 탐지한다. 탐지된 의심 패킷은 금융보안원의 보안관제센터에 근무하는 관제 요원들에 의해 공격 여부에 대해 최종 분석되어 레이블링 작업이 완료된다. 정상 패킷은 공격 패킷에 비해 많은 양이 수집된다. 앞서 소개된 [그림 1-1]의 세 번째 프로세스에서 침입탐지 시스템이 의심 패킷으로 탐지하지 않은 모든 패킷들이 정상 패킷에 해당되며 자동으로 레이블링이 완료된다. 본 연구는 패킷의 공격 여부를 탐

9) 웹 기반 인터페이스를 바탕으로 빅데이터를 검색 및 모니터링할 수 있는 비정형 데이터 분석 솔루션

지하는 이진 분류가 주목적이므로 두 가지 유형의 레이블링이 완료된 데이터 세트로 연구를 수행한다.

금융보안원은 금융권 네트워크 패킷에 대해 많은 양의 데이터를 가지고 있으며, 특히 기존 업무 수행 과정에서 이미 레이블링이 완료된 데이터를 다수 보유하고 있어 딥러닝을 통해 지도학습을 수행함에 있어 상당한 장점을 가지고 있다. 다만, 업무의 특성 및 데이터의 소유권 문제로 데이터 자체는 공개할 수 없으며 본 연구에서는 이 데이터에 대한 실험 결과를 통해 제안한 방법의 유효성에 대해 검증을 실시할 것이다.

본 연구를 위해 2021년 2월에 금융권 전체에서 유입된 네트워크 트래픽 패킷을 대상 데이터를 선정하였다. 일반적으로 공격 패킷은 정상 패킷에 비해 그 비율이 극도로 낮다. 이번 연구에 사용한 데이터를 들여다보면 2월 한 달간 공격으로 분석된 패킷의 전체 수는 하루 동안 일반적으로 들어오는 패킷의 양의 1%정도 수준밖에 되지 않는다. 데이터 비율을 맞추어 주기 위해 2월 전체 수집된 패킷 중 공격으로 분석된 패킷 전체와 2월 21일 하루에 수집된 정상 패킷을 대상 데이터로 정의하였으며 총 271,370건으로 집계되었다.

공격데이터와 정상데이터의 비율은 비슷하게 조정하였다. 실제 유입되는 패킷의 공격 대 정상 비율과 구성된 데이터 세트의 공격 대 정상 비율은 다르지만 실제 유입되는 패킷의 비율로 학습을 할 경우 딥러닝 모델이 모두 정상이라고 판정만 해도 정확도가 99%이상이 되므로, 공격 패킷을 무시할 소지가 높아진다. 따라서 상대적으로 극소량의 공격 데이터를 고려하여 균형을 맞춘 데이터 세트 구성이 필요하다. 데이터 세트의 비율은 다양한 시도 끝에 경험적으로 획득하였고 공격과 정상의 비율이 비슷할 때 가장 효과적인 학습이 가능하였다.

훈련데이터와 검증데이터의 분리도 중요하다. 일반적으로 모든 데이터가 다르다고 가정하면 수집한 데이터를 무작위로 일정 비율로 나누어 사용하는 경우가 많다. 그러나 네트워크 패킷 특히 공격 패킷의 경우 공격자가 한 번에 같은 공격을 여러 금융회사로 연속하여 시도하는 경우가 존재한다. 만약 금융보안원과 같은 부문 관제 기관이 아니라면 단일 시

도처럼 보일 수 있지만 금융보안원은 금융 부문의 패킷을 모두 수집하므로 유사한 패킷이 동일 시간대에 집중해서 유입되게 된다. 따라서 정확한 학습을 위해 훈련 데이터와 검증 데이터는 서로 다른 시간에 유입된 데이터를 사용하여 검증 데이터에 훈련 데이터가 포함되지 않도록 분리하여 구성하였다.

위에서 언급한 레이블링이 완료된 자료 중 데이터의 비율 및 훈련데이터와 검증데이터의 분리를 고려하여 실험 데이터를 구성하였다. 훈련데이터와 검증 데이터 모두 공격 데이터와 정상데이터를 분리하여 구성하였다. 훈련 데이터 중 공격 데이터는 2021년 2월 1일부터 2월 23일까지 수집된 104,874건의 전체 공격 패킷을 사용하였으며 정상 데이터는 2021년 2월 21일 하루 동안 수집된 정상패킷 중 임의로 100,000건을 추출하여 사용하였다. 검증 데이터의 경우 2021년 2월 24일부터 2월 28일까지 수집된 29,086건의 공격 패킷을 공격 데이터로 사용하였고 정상 데이터는 2021년 2월 21일 하루 동안 수집된 정상패킷 중 훈련데이터와 중복되지 않는 37,410건의 패킷을 임의로 추출하여 사용하였다.

제 2 절 HTTP 패킷 추출

앞서 수집된 데이터는 모두 raw 패킷이다. 즉 가공이 전혀 되지 않아 연구에 불필요한 데이터들도 많이 포함되어 있다. 따라서 1절에서 수집된 데이터를 학습이 용이하도록 가공하여 사용하여야 한다.

금융보안원으로 유입되는 raw 패킷은 방대한 메타데이터를 포함하고 있다. IP, port, MAC주소 등 실제 공격 내용과는 무관한 데이터가 모두 포함된다. 딥러닝을 통해 네트워크 패킷을 학습시키기 위해서는 raw 패킷에서 중요한 내용만을 추출하여 학습을 수행하는 것이 의미 없는 정보의 학습을 피할 수 있다는 점에서 보다 효율적이다. 따라서 실제 패킷의 내용이 담긴 표현 계층, 즉 HTTP 패킷에 대한 분석이 필요하다. 분석을 통해 반복되거나 다양성이 떨어지는 데이터는 제외시키고 핵심적인 부분을 추출하여 입력 데이터로 활용해야 한다.

금융보안원은 외부로부터 금융회사로 유입되는 단방향의 트래픽을 대상으로 공격 여부를 분석한다. 따라서 HTTP 프로토콜의 request와 response 구문 중 request 패킷의 구조를 분석하는 것이 중요하다.

HTTP request는 크게 start-line, header, body 3개의 부분으로 구성되어 있으며 각 필드별 상세 내용은 [표 3-1]과 같다[8]. start-line 부분은 HTTP 패킷에서 가장 먼저 등장하는 부분이다. HTTP method와 관련된 명령어가 처음에 등장하고, 다음으로 request target 그리고 마지막에 HTTP 버전이 기술된다. 세 가지 요소 중 가변적인 텍스트로 이루어진 부분은 request target 부분이며 흔히 알고 있는 인터넷 주소와 몇 가지 인자가 붙어있는 형태로 구성된다. 반면에 HTTP method는 get, post, put, delete 등 여러 가지 명령어가 있으나 대부분 get 혹은 post이다. HTTP 버전도 마찬가지로 1.0 혹은 1.1이 대부분이며 추후에 버전 증가에 따라 달라질 수 있으나 공격 여부에 크게 영향을 미치지 않는 정보이다.

[표 3-1] HTTP request 패킷의 구조[8]

구성	내용
start line	HTTP method / request target / HTTP version Ex) GET http://naver.com HTTP/1.1
headers	Common : Date, connection, Cache-Control, Content-Length Request : Host, User_Agent, Accept, Authorization, Origin, Referer Response : Access-Control-Allow-Origin, Allow, Content-Disposition, Location
body	some message or nothing

headers 부분은 여러 가지 메타데이터를 담고 있으며 전송일자, 패킷의 길이, 캐시컨트롤 정보 등이 들어가 있다. 다만, 패킷마다 필요한 부분만 채워져서 보내므로 실제로는 패킷마다 내용이 없는 필드도 다수 존재하며 대부분 분류형 데이터라서 학습의 효용성이 낮을 것으로 판단하였다. 실제로 몇 가지 headers의 데이터 필드를 채택해 보았지만 성능에 영향이 없는 것으로 관측되어 본 연구의 입력데이터에서는 제외시켰다.

body 부분은 실제 HTTP의 본문에 해당되며 모든 요청에 본문이 포함되어 있지는 않다. 실제로 1절에서 연구에 사용하기 위해 선정한 데이터 204,874건 중에서도 28,526건 정도만 body 값을 가지고 있었다.

본 연구에서는 HTTP 패킷 중 start line 내부의 request target 부분의 전체 텍스트와 body 부분의 전체 텍스트를 추출하여 입력데이터로 사용하였다. 그 외의 HTTP 패킷의 내용들은 패킷별로 크게 내용의 차이가 없는 분류형 데이터(categorical data)이다. 또한 초기 실험 단계에서 다양한 방법으로 딥러닝 모델의 입력 값에 이들을 포함하여 수행하여 보았지만 모델의 성능개선에는 크게 영향이 없었다.

제 3 절 아스키 코드 빈도 입력 데이터

raw 패킷에서 학습에 필요한 부분을 결정하였고 이제 대상 텍스트를 딥러닝을 위해 실수 형 배열 형태로 가공 및 생성해주는 작업이 필요하다. 본 절에서는 그 중 한 가지 방법으로 아스키 코드의 출현 빈도를 활용하는 방법에 대해 기술한다.

1. 아스키 코드 빈도 입력 데이터 생성 방법

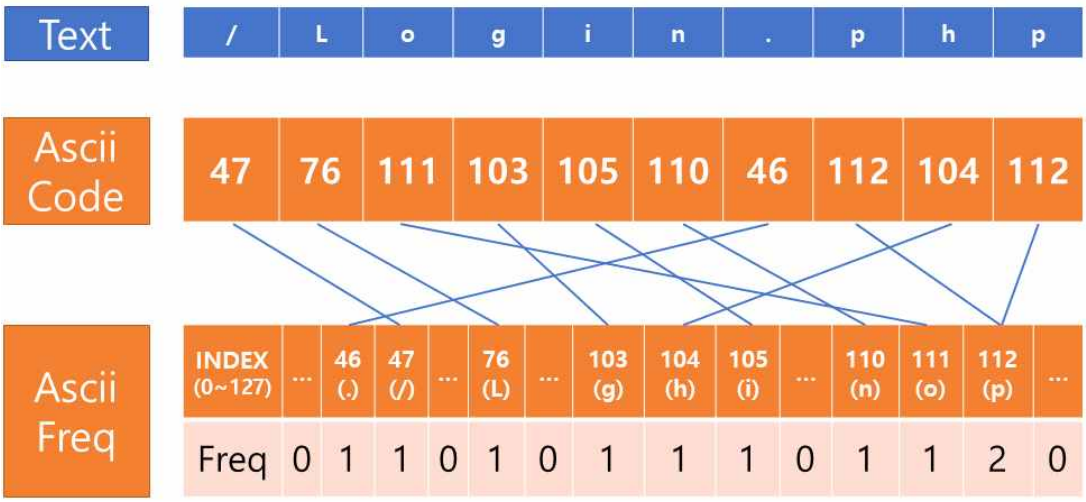
앞서 기술한 바와 같이 본 연구에서 사용할 딥러닝 모델에 대한 입력 값은 모두 비정형적인 문자열로 이루어져 있다. HTTP packet의 payload는 일반적인 자연어와는 다르게 HTTP 프로토콜에 맞게 설계되어 있어 단어 단위로 구분이 명확하게 되지 않는다. 따라서 문자 단위로 분해하여 입력 값의 특징을 추출하고자 하였으며 그 방법 중 하나로 아스키 코드 빈도(ASCII code frequency)를 계산하는 방법을 사용하였다. 일반적으로 텍스트 분석에 사용하는 n-gram에서 unigrams를 사용하는 방식과 유사하지만 아스키 코드라는 사전에 정의된 범위 내에서 한정시켜 사용한다는 점이 다르다.

대부분의 통신에 사용되는 문자는 아스키 코드로 이루어져 있으며 간혹 유니코드가 포함되는 경우도 있지만, 공격 명령어는 모두 아스키 코드 기반으로 작성된 코드에 의해 이루어진다. 따라서 아스키 코드 문자 내에서 대부분의 패킷의 내용을 분석할 수 있다. 또한 128개의 비교적 적은 수의 문자로 구성되어 있는 점 또한 모델의 학습 및 예측 속도에서 장점을 가진다.

텍스트로부터 아스키 코드 빈도를 사용한 입력 특징을 추출하는 방법은 다음과 같다:

1) HTTP 패킷에서 텍스트 부분을 추출한다.
2) 아스키 코드의 크기에 맞게 128바이트 크기의 배열을 생성한다.
3) 이 배열에 추출된 텍스트를 읽으며 빈도를 기록한다.
4) 0에 가까운 값을 만들기 위해 배열 내의 개별 수치를 빈도의 최대값으로 나누어 준다.

[그림 3-1]은 문자열을 아스키 코드 빈도 형태로 생성한 예시이다. 예시에 나온 문자열에서 소문자 ‘p’는 2번 등장하며 ‘p’에 해당하는 아스키 코드 112번이므로 입력 배열 인덱스 중 112번 항목에 2가 채워지게 된다. 한 번도 출현하지 않은 인덱스에는 0이 채워지고 한번 씩 출현한 ‘L’, ‘o’, ‘g’ 등에 해당하는 인덱스에는 1이 채워진다.



[그림 3-1] 아스키 코드를 활용한 입력 데이터 전처리 예시

이러한 방식으로 입력데이터를 생성할 경우 문자열의 길이에 무관하게 항상 동일한 크기의 입력데이터를 생성할 수 있다는 장점이 있다. 문자간의 위치에 대한 정보는 소실되지만 문자 빈도에 대해서는 손실이 없이 보존하는 데이터 형태를 지니게 된다.

2. 개선된 아스키 코드 빈도 입력 데이터 생성 방법

아스키 코드 빈도 모델의 개선을 위해 데이터 전처리과정을 발전시킨 모델로써 크게 두 가지의 전처리 과정을 추가하였다.

먼저 기존의 아스키 코드 128바이트를 사용하던 입력값을 출력가능한 문자(printable character)만 추출하여 96바이트로 축소시킨다. 아스키 코드 0번에서 31번까지는 출력이 불가능한 제어문자로 이루어져있어 실제 유입되는 패킷의 내용에도 출현하는 경우가 거의 없으므로 입력값의 크기는 128바이트에서 96바이트로 줄어들어도 정보의 양은 변화가 없다.

이렇게 줄어든 96바이트에 사전에 만들어진 가중치를 곱하여 입력 값을 일괄적으로 수정을 한다. 가중치는 입력값과 동일한 크기의 배열로 구성되어 있으며 아래와 같은 과정을 거쳐 생성한다:

1) 침입정보시스템에 정의되어있는 스노트룰과 같은 탐지용 룰 세트의 리스트를 읽어온다.
2) 리스트는 [표 2-1]의 구조를 갖추고 있는데 content 부분에 존재하는 탐지용 패턴 문자만 추출하여 시그니쳐 별로 한행씩 기록한다.
3) 추출된 행을 앞서 소개한 대하여 아스키 코드 빈도 배열로 바꾸어 준다.
4) 생성된 128×n 개의 배열에 대해 각 행별 전체 합을 구한다. 그 결과 128×1의 배열이 생성된다.
5) 128×1의 배열의 원소 중 최대값으로 전체를 나누어주고 1을 더해준다. 그 결과 배열은 모두 1과 2사이의 값을 가지게 된다.
6) 5)에서 구한 가중치 배열 값을 기존의 3절 1항에서 생성된 입력배

열에 곱한다. 만약 출력가능한 아스키 코드만 사용하는 경우에는 뒤의 96바이트에만 적용하면 된다.

가중치를 곱해주는 것은 두 가지 측면에서 의미가 있다. 첫째, 각 기관에서 운영하는 오용탐지 기반의 보안 관제 체계를 입력 값에 반영하여 기관별로 맞춤식 운영이 가능하다는 점이다. 본 실험에서는 스노트룰 커뮤니티 룰 세트 3.0(Snort rule community rule set 3.0)[9]의 3488건의 룰 세트에 대해 적용하였으며 실제 운영하는 침입탐지 시스템에 사용하는 룰 세트로 가중치를 만들면 더 좋은 결과를 만들 수 있을 것으로 예상된다. 둘째는 공격 탐지 패턴에 자주 등장하는 문자에 대해 가중치를 주어서 공격을 구성하는 스크립트에 자주 등장하는 문자를 더욱 강조되게 만들어 입력 값에 증폭효과를 주어 탐지 특성을 명백하게 구성할 수 있는 효과를 유도한다.

제 4 절 아스키 코드 인코딩 입력 데이터

앞서 3절에서는 아스키 코드의 출현 빈도를 활용하여 입력 데이터를 생성하는 방법에 대해 기술하였다. 출현 빈도를 활용한 방법이 텍스트의 종(縱)적인 특성을 활용하는 방법이라면 이번 절에서는 아스키 코드를 활용하여 텍스트를 횡(橫)적으로 가공하는 방법을 소개하고자 한다.

아스키 코드 빈도 데이터는 텍스트의 히스토그램을 통해 빈도를 파악할 수 있지만, 문자 간의 순서에 대한 특성은 보존되지 않고 출현 빈도만 누적된 정보를 가지게 된다. 합성곱 신경망이나 순환 신경망의 장점을 활용하기 위해서는 특정 시퀀스의 주변 정보를 참조하기 용이한 입력 특성을 가져야 한다. 이를 위해 텍스트의 순서 정보를 보존하며 적절한 실수 형태의 배열로 인코딩을 할 필요가 있는데, 일반적으로 텍스트 기반 학습에서는 워드 벡터형태로 변환 후 임베딩 층을 사용하는 경우가 많다. 그러나 본 연구에서는 입력 텍스트가 자연어와 달리 단어로 명확히 구별되지 않는다. 이러한 특성을 고려하여 단어단위가 아닌 문자단위

로 입력데이터를 구성하는 것이 유리하다고 판단하였다. 또한 대부분의 의미 있는 문자는 아스키 코드로 이루어진 점을 고려하여, 아스키 코드 값으로 인코딩하여 생성한 입력데이터를 사용하였다.

아스키 코드 인코딩은 비교적 간단하게 수행할 수 있다. 단 입력 패킷의 길이가 모두 상이하므로 딥러닝 모델을 활용하기 위해서는 패킷의 텍스트를 변환한 입력 값은 모두 고정된 값으로 동일하게 만들어야 한다. 텍스트의 각 문자에 해당하는 아스키 코드 번호를 입력하여 실수형의 배열을 생성하고 모든 패킷이 동일한 크기를 가지도록 만들기 위해 패딩(padding)[10]을 수행한다. 마지막으로 아스키 코드의 최대값(127)으로 모든 배열의 원소를 나누어 학습이 용이한 형태의 데이터를 생성한다. 본 연구에서는 제로 패딩을 통해 입력값의 길이를 맞추어 주었다. 앞서 소개한 [그림 3-1]의 과정에서 두 번째 줄에 표시된 부분이 아스키 코드 인코딩을 수행한 예시이다.

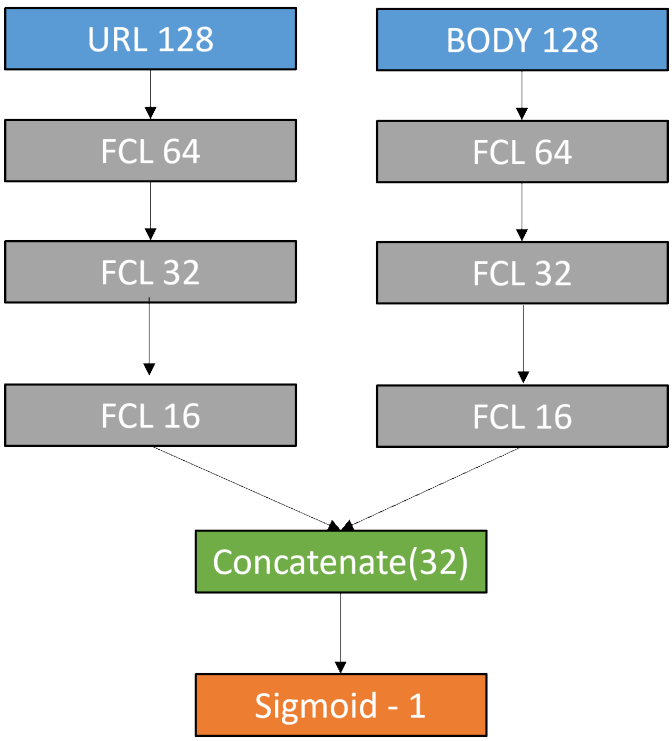
10) 고정된 입력값의 길이를 통일하기 위해 비어있는 값을 채우는 작업. 0으로 채우는 제로패딩, 앞의 내용을 반복해서 채우는 패딩 방법 등이 있다.

제 4 장 딥러닝 모델 구성

지금까지 딥러닝의 학습을 위한 준비 작업을 수행하였다. 이번 장에서는 가공된 입력 데이터를 학습하기 위한 모델의 생성 방법에 대해 논의하고자 한다. 3절에 소개된 4개의 입력 데이터 형태에 각각 대응하는 모델을 하나씩 기술하겠다.

제 1 절 아스키 코드 빈도 모델

아스키 코드 빈도 모델은 앞서 소개된 3장 3절의 아스키 코드 빈도 입력 데이터를 사용한 모델이다. 아스키 코드 빈도 입력 데이터는 패킷 텍스트의 길이에 무관하게 항상 128바이트의 고정된 크기의 입력으로 생성되므로 일반적인 텍스트 시퀀스 데이터를 사용할 때 고려하는 패딩은 수행할 필요가 없다. 해당 모델에 대한 구조는 [그림 4-1]과 같다.



[그림 4-1] 아스키 코드 빈도 모델

입력층은 앞서 3장 2절에서 기술한 바와 같이 HTTP 패킷의 request target 부분과 body 부분을 분리하여 다중입력 방식으로 구성하였다. 입력 값은 3개의 완전 연결 층(fully connected layer, FCL)을 거쳐 출력층 직전의 은닉층까지 각각의 텍스트를 학습 한다. 마지막의 은닉층의 연결을 통해 앞서 분리하였던 다중입력을 하나로 연결하고 시그모이드(sigmoid) 층으로 출력해낸다. 이 때 출력되는 값은 입력 데이터가 공격 패킷일 확률을 나타내며 0.5이상의 값이 나오면 공격으로 판별하게 된다. 그 중에서도 1에 가까울수록 공격일 확률은 더욱 높다고 볼 수 있다.

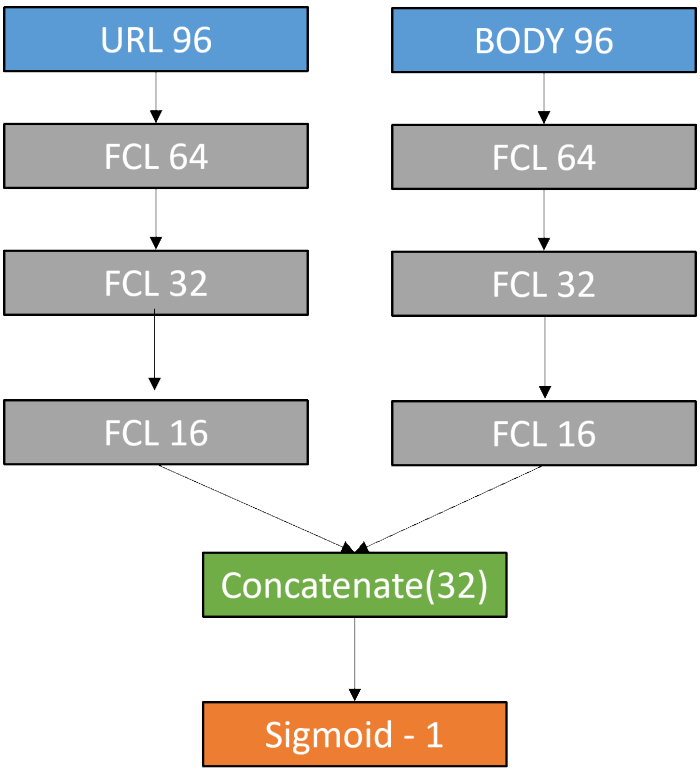
각 은닉층은 30%의 드롭아웃(drop out)[11]을 일괄로 적용하였으며 활성 함수는 Rectified Linear Unit(ReLU)를 사용하였다. 옵티마이저는 Adam을 사용하고 손실(loss)함수는 이진 분류에서 일반적으로 사용하는 binary cross entropy를 이용해 학습을 수행하였다. 이 모델을 포함하여 이후에 사용한 초매개변수(hyper parameter)[12]값들은 별도의 언급이 없는 한 동일한 값을 사용하였다. 초매개변수 값들은 경험적으로 획득한 값이며 상세한 연구를 통해 개선할 여지는 남아 있다.

제 2 절 가중치 적용 아스키 코드 빈도 모델

아스키 코드 빈도 모델에 가중치를 적용한 형태이므로 입력 데이터나 모델의 구성은 1절의 모델과 크게 다른 점은 없다. 다만 출력가능한 아스키 코드만을 사용하였기 때문에 입력층의 크기가 1절의 모델에 비해 약간 작다. 가중치를 적용한 입력데이터를 사용한 모델은 [그림 4-2]와 같이 구성된다.

---

11) 딥러닝 기반 학습 수행 시, 과대적합을 방지하기 위해 고의로 학습된 은닉층의 일정 부분을 초기화시키는 방법.
12) 학습절차를 제어하는 다양한 인자, 학습 시 사용자가 임의로 결정한다.

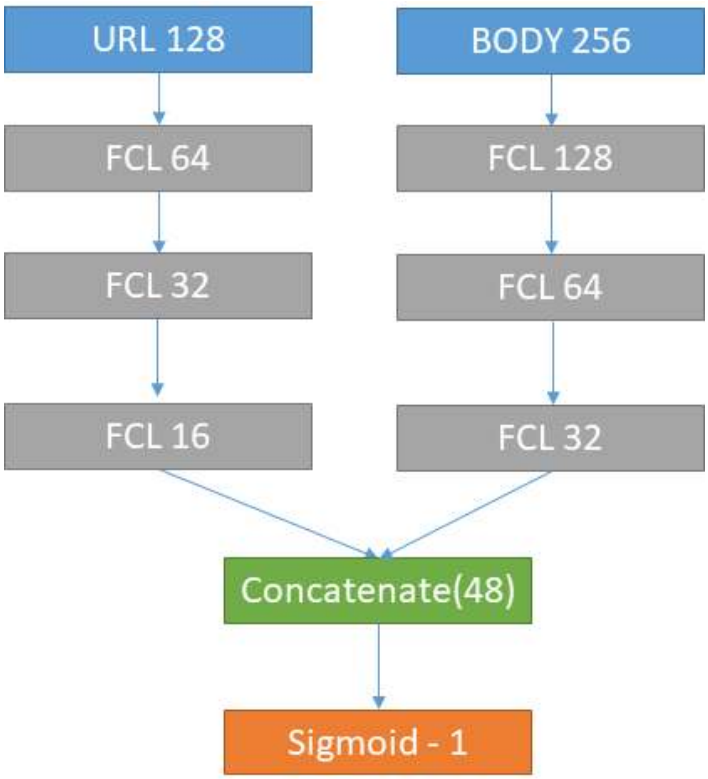


[그림 4-2] 가중치 적용 아스키 코드 빈도 모델

입력층은 1절과 마찬가지로 다중 입력을 사용하였으며 각 입력층의 크기가 1절의 모델과 달리 96바이트로 구성된 점이 가장 큰 차이점이다.

제 3 절 아스키 코드 인코딩 모델

아스키 코드 인코딩 모델은 앞서 추출한 3장 4절의 아스키 코드 인코딩 입력 데이터를 사용한 모델이다. 이 입력 데이터를 사용하여 두 가지 모델을 구현하였는데 그중 첫 번째 모델이 일반적인 완전 연결 층을 사용한 모델이다. 대략적인 구조는 1절의 모델과 유사하며 해당 모델에 대한 구조는 [그림 4-3]과 같다.



[그림 4-3] 아스키 코드 인코딩 모델

입력층의 모양이 1절의 모델과 약간의 차이가 있다. 이는 금융권에 유입되는 패킷의 통계적인 특성을 파악하여 평균 길이를 기반으로 패딩을 수행하였기에 body 부분이 더 긴 입력 값을 가지게 되었다. 그 외는 앞서 소개한 모델과 유사하며 초매개변수 또한 1절의 모델과 동일하게 적용하였다.

학습 데이터에 대한 통계적 특성을 분석하여 HTTP 패킷의 URL 부분과 body 부분의 고정된 크기를 정하였다. 3장에서 정의한 데이터에 대해 수행하였으며 기본적인 통계적 특성을 구하였다. 통계 데이터의 정확성을 위하여 내용이 비어있는 패킷들은 계산 시 제외를 시켰으며 도출된 결과는 [표 4-1]과 같다.

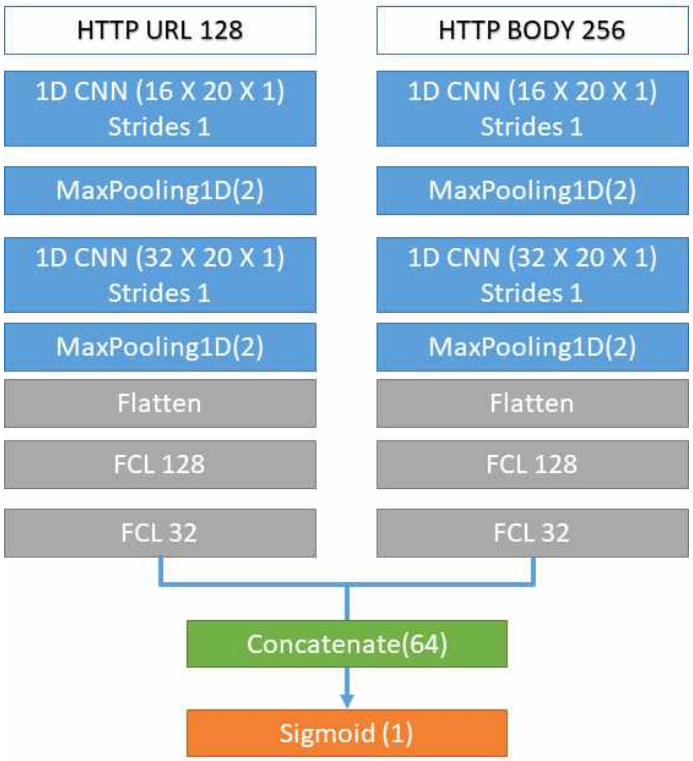
[표 4-1] HTTP Request 패킷의 통계적 특성

구분	URL 문자열의 길이	Body 문자열의 길이
최소값	1	1
최대값	2,916	6,451
평균	69.23	180.41
표준편차	136.38	321.93

대상 데이터의 통계적 특성을 보면 body 부분의 문자열의 길이의 평균값이 URL 문자열에 비해 상당히 길다는 것을 알 수 있다. 가공되기 이전의 텍스트의 길이를 고려하여 다중입력 모델의 입력층의 크기를 결정하면 입력데이터에 내포된 정보의 손실을 줄일 수 있다. 그 결과로 본 연구에서는 URL 부분과 body 부분의 입력값 길이에 2배의 차이를 두고 다중 입력 층을 구성하였다.

제 4 절 아스키 코드 인코딩 합성곱 신경망 모델

아스키 코드 인코딩 합성곱 신경망은 3절의 모델과 동일한 입력값을 사용하지만 합성곱 신경망 층을 배치하여 단일 문자단위가 아닌 문자 주변의 패턴까지 고려할 수 있도록 설계한 모델이다. 1차원 합성곱 신경망(convolutional neural network, CNN)을 입력 층 직후에 배치하여 모델을 구성하였으며 풀링과 flatten 층을 통해 다시 완전 연결층에 연결하였다. 마지막에는 URL과 body부분을 연결하여 sigmoid 출력층에 연결하여 구성을 마쳤다. 본 모델에 대한 상세 구조는 [그림 4-4]와 같다.



[그림 4-4] 아스키 코드 인코딩 합성곱 신경망 모델

1차원 합성곱 신경망과 맥스풀링 2개층을 구성하고 마지막에는 완전 연결층을 사용한 모델이다. 커널의 크기는 일반적으로 문자열 기반의 패턴 탐지 시에 사용하는 패턴의 크기의 평균값을 고려하여 실험을 반복하여 일반적으로 가장 좋은 결과가 나오는 수치로 결정하였다. 본 모델에서 커널의 크기는 20으로 설정하였으며 그 외의 초매개변수는 앞서 소개한 모델과 동일한 수치를 사용하여 학습을 수행하였다

제 5 장 실험 및 결과

3절에서는 딥러닝에 활용할 수 있도록 데이터를 가공하였으며 4절에서는 가공한 데이터를 학습시키기 위한 모델을 구성하였다. 본 절에서는 앞서 소개된 4개의 모델에 대해 실제로 학습을 수행한 결과를 소개하고 더 나아가 각 모델이 가진 장점을 활용한 앙상블 모델을 구성해 보았다.

제 1 절 실험 개요 및 환경

1. 실험 개요

실험은 3장 1절에서 정의한 데이터를 바탕으로 수행하였다. 정확한 결과를 위해 훈련데이터와 검증데이터는 중복되지 않도록 분리하여 취급하였으며 128개의 미니 배치사이즈로 나누어 30회 반복하여 학습하였다. 학습 시 콜백함수를 사용하여 직전 학습회차보다 검증 정확도가 낮아질 때 마다 학습률(learning rate)을 10%씩 낮추었다. 학습 과정에서 정확도가 가장 높은 학습회차의 모델을 체크포인트로 저장하였다. 최종 결과 도출에 사용한 모델은 체크포인트로 저장된 최적의 모델이다.

실험 시 사용한 주요 하이퍼파라미터는 다양한 시도 끝에 경험적으로 결정하였다. 은닉층의 활성화 함수는 rectified linear unit(ReLU)을 사용하였으며 출력층의 경우 0과 1사이의 값을 출력하는 sigmoid 함수를 사용하였다. 드롭아웃은 모든 완전 연결층에 대해 30%씩 적용하였으며 optimizer는 adaptive moment estimation(ADAM)을 사용하였다. 마지막으로 손실 함수는 이진 분류에서 일반적으로 사용하는 binary cross entropy를 사용하였다.

2. 실험 환경

본 연구는 데이터 유출을 막기 위해 금융보안원에 직접 방문 혹은 근무하며 실시하였으며, 사용한 데이터는 금융보안원 퇴거 시 복사본은 모두 삭제하고 결과에 대한 내용만 기록하는 과정을 거쳐 수행하였다. 실제 사용했던 프로그래밍 언어와 주요 라이브러리 및 단말기 정보는 [표 5-1]과 같다.

[표 5-1] 실험 환경 내역

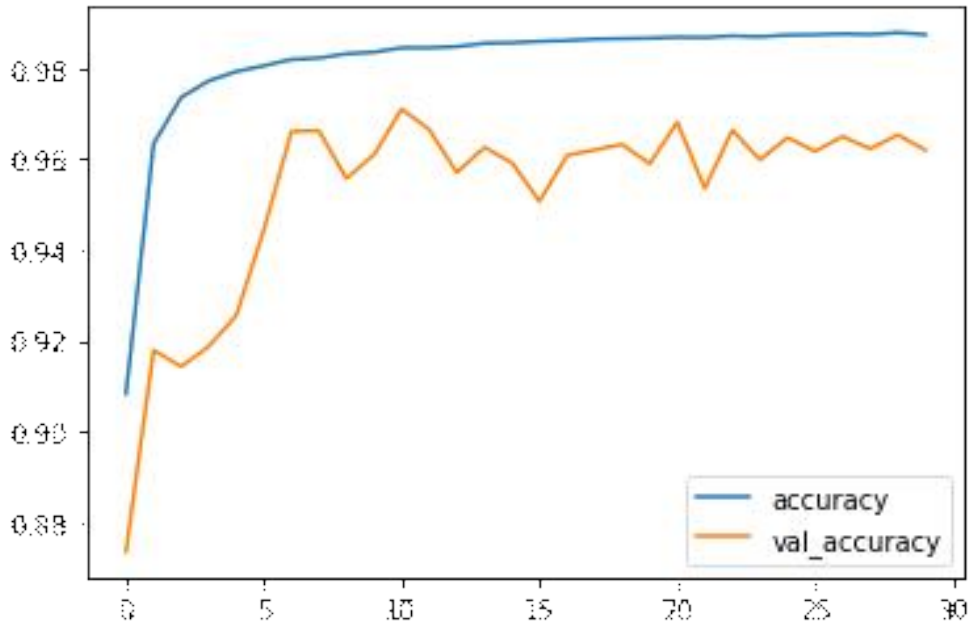
구분	사용 환경	내용
language & library	python	3.8.5
	keras	2.4.3
	tensorflow-gpu	2.5.0
	numpy	1.19.2
	pandas	1.1.3
	conda	4.10.1
단말기	CPU	Intel core i7-8565U 1.80GHz(8 CPU)
	RAM	16G
	GPU	Geforce GTX 1050 Ti(4G)

제 2 절 모델별 실험 결과

앞서 소개한 환경 하에서 실제로 학습 및 검증한 내용을 모델별로 하나씩 기술하고자 한다. 각 모델별로 학습회차별 훈련 정확도와 검증정확도를 기록하여 그래프를 그려보았다. 해당 그래프를 통해 어느 정도 시점에 최적화가 이루어지게 되었는지, 과적합(overfitting)이 일어나지는 않았는지, 학습은 충분히 이루어지었는지 등을 판단할 수 있다.

1. 아스키 코드 빈도 모델 결과

30회차의 학습 중 약 5회차 시점부터 학습 및 검증 정확도가 가파르게 증가하여 최적치에 도달하였다. 이후 회차의 학습에서 검증데이터에 대해서는 크게 성능 향상이 없음을 확인할 수 있다. 학습회차별 정확도는 [그림 5-1]에서 확인할 수 있다. 11회차의 체크포인트를 최적의 모델로 채택하였으며 이 때 검증 정확도는 0.9711이며 F1-score는 0.9670을 기록하였다. 본 모델을 포함하여 이후의 모델 학습 결과에 대한 상세한 수치는 본 장의 말미에 정리하여 기술하겠다.

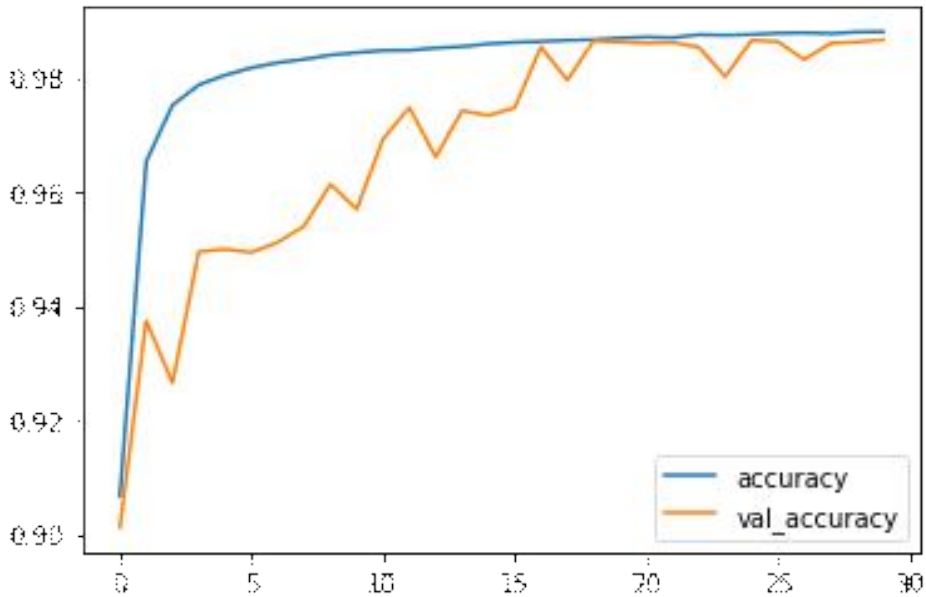


[그림 5-1] 아스키 코드 빈도 모델 학습회차별 정확도

2. 아스키 코드 가중치 빈도 모델 결과

[그림 5-2]에서 나타난 바와 같이 전체 학습 회차 동안 점진적으로 검증 정확도가 높아졌다. 30회차에 도달했을 때는 훈련 정확도에 근접할 정도로 성능이 개선되었다. 일반 아스키 코드 빈도 모델과 비교했을 때 근소한 차이지만 우수한 성능을 내고 있음을 알 수 있다. 본 실험에서는

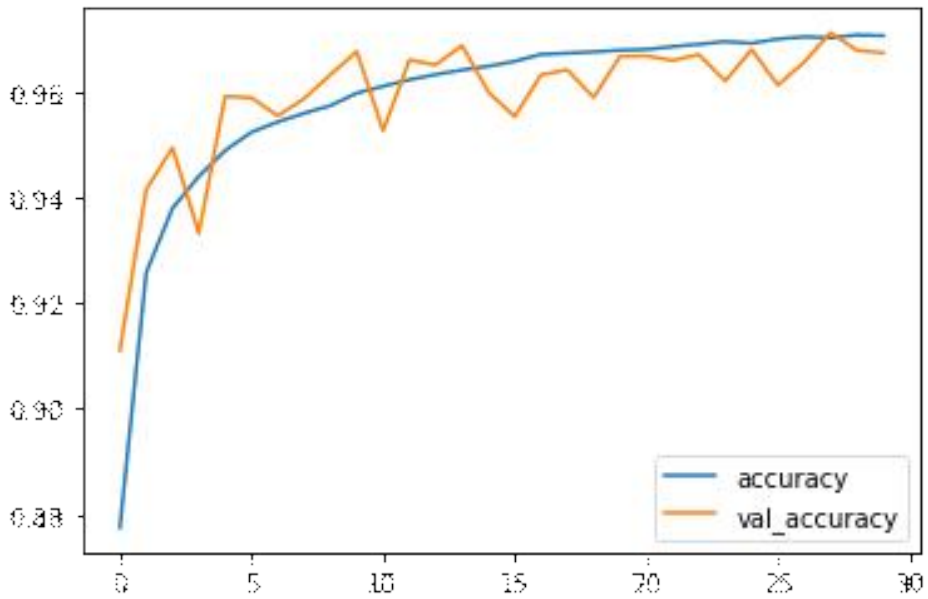
공개된 커뮤니티 룰 기반으로 생성한 가중치를 사용하였으나 만약 각 기관에 구축된 침입탐지시스템의 룰 세트를 기반으로 가중치를 생성하여 적용할 경우 더 우수한 성능을 기대할 수 있을 것이다.



[그림 5-2] 아스키 코드 가중치 빈도 모델 학습회차별 정확도

3. 아스키 코드 인코딩 모델 결과

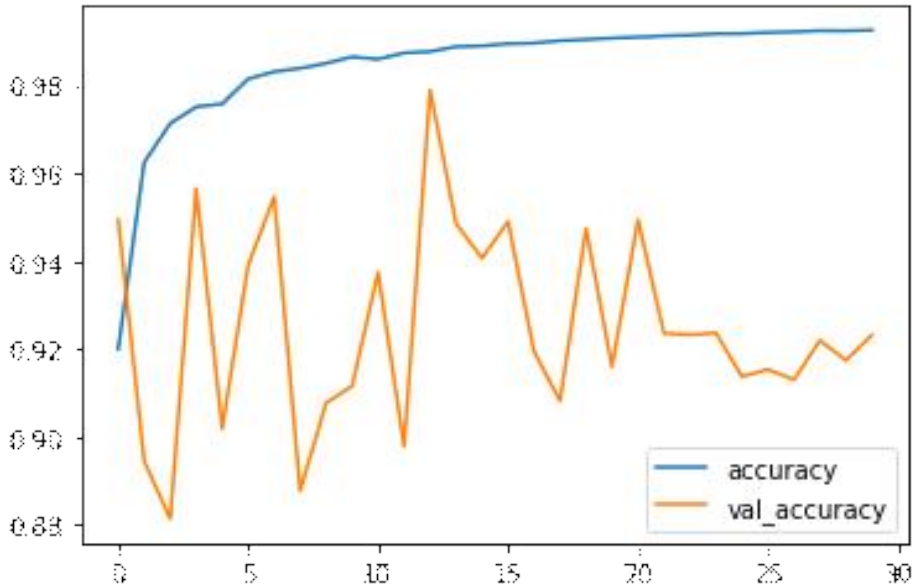
[그림 5-3]은 아스키 코드 인코딩 모델의 학습회차별 정확도를 표시한 그림이다. 초기에 검증 정확도가 학습 정확도 보다 높게 나오는 회차가 다수 발생하였다. 이후에는 점진적으로 증가하며 훈련 정확도에 근접하는 결과를 보이고 있다. 검증 정확도가 가장 높게 나온 28회차의 체크포인트를 최종 결과로 선정하였고 0.9712의 검증 정확도와 0.9677의 F1-score를 기록하였다.



[그림 5-3] 아스키 코드 인코딩 모델 학습회차별 정확도

4. 아스키 코드 인코딩 합성곱 신경망 모델 결과

합성곱 신경망 모델은 [그림 5-4]에 나타난 바와 같이 학습 회차의 증가에 따른 검증 지표들의 향상은 눈에 띄지 않았다. 다만 다른 모델들에 비해 회차별로 검증정확도의 진폭이 상대적으로 크게 나타났다. 합성곱 신경망 모델의 경우 은닉층의 학습대상 변수가 일반적인 완전 연결층에 비해 많아서 학습 시간이 상대적으로 많이 소요된다. 본 실험에서는 일반적인 완전 연결층 기반의 모델에 비해 10배정도로 더 많은 학습 시간이 소요되는 데에 비해 정확도는 크게 개선되지는 않았다. 대량의 패킷에 대해 실시간으로 탐지를 할 경우 성능 문제를 고려하여 본 모델을 사용할 필요가 있음을 확인할 수 있었다. 13회차의 체크포인트를 선정하였고 0.9790의 정확도와 0.9764의 F1-score를 기록하였다.



[그림 5-4] 아스키 코드 인코딩 합성곱 신경망 모델 학습회차별 정확도

5. 결과 비교 분석

앞서 학습한 4가지의 모델들에 대해 최적의 체크포인트 모델을 바탕으로 대표 측정치를 비교한 내용을 [표 5-2]에 정리하였다.

[표 5-2] 모델별 학습 결과 값

구분	metrics	아스키 코드 빈도 모델	아스키 코드 가중치 빈도 모델	아스키 코드 인코딩 모델	아스키 코드 인코딩 합성곱 신경망 모델
훈련	accuracy	0.9847	**0.9878**	0.9705	0.9872
	precision	0.9882	**0.9919**	0.9745	0.9900
	recall	0.9818	0.9842	0.9680	**0.9848**
	F1-Score	0.9850	**0.9880**	0.9712	0.9874
검증	accuracy	0.9711	**0.9867**	0.9712	0.9790
	precision	0.9865	**0.9934**	0.9673	0.9795
	recall	0.9482	**0.9766**	0.9682	0.9733
	F1-Score	0.9670	**0.9849**	0.9677	0.9764

전반적으로 4개의 모델 모두 유사한 성능을 보이고 있다. 아스키 코드 가중치 빈도 모델의 경우 훈련 재현율을 제외하면 전반적으로 다른 모델에 비해 상대적으로 근소하게 우수한 수치를 나타내고 있다. 입력데이터나 다양한 환경에 따라 결과치는 다르게 나타날 수 있어 한 가지 모델이 특별히 우수하다고 할 수는 없다. 다만 4가지 모델 모두의 검증 정확도가 0.97 이상으로 나타난 걸로 보아 제안한 모든 모델에 대해 유효한 성능을 기대할 수 있다.

제 3 절 앙상블 모델[13] 적용

앞서 학습한 4가지의 모델들에 대해 최적의 체크포인트 모델을 바탕으로 대표 측정치를 비교해보았다. 현재 정확도도 높은 편이라고 할 수 있지만, 앞서 1장 2절에서 언급한 활용방안을 고려하면 매일 1,000만 건 이상 유입되는 패킷의 페이로드 중 의심 패킷을 높은 밀도로 찾아야 하므로 위음성(false negative)의 수를 가능한 최소화 시켜야 하는 점에서 더욱 정밀도(recall)을 높일 필요가 있다.

일반적으로 앙상블 모델은 서로 성질이 다른 모델을 조합하였을 때 효과가 크다[10]. 본 연구에서 제안한 모델은 입력 데이터를 크게 2가지 형태로 사용하였으며, 같은 형태의 입력데이터를 사용한 경우도 은닉층을 서로 다른 형태로 구성하였기에 앙상블 모델을 구성할 경우 더욱 향상된 결과를 기대할 수 있다.

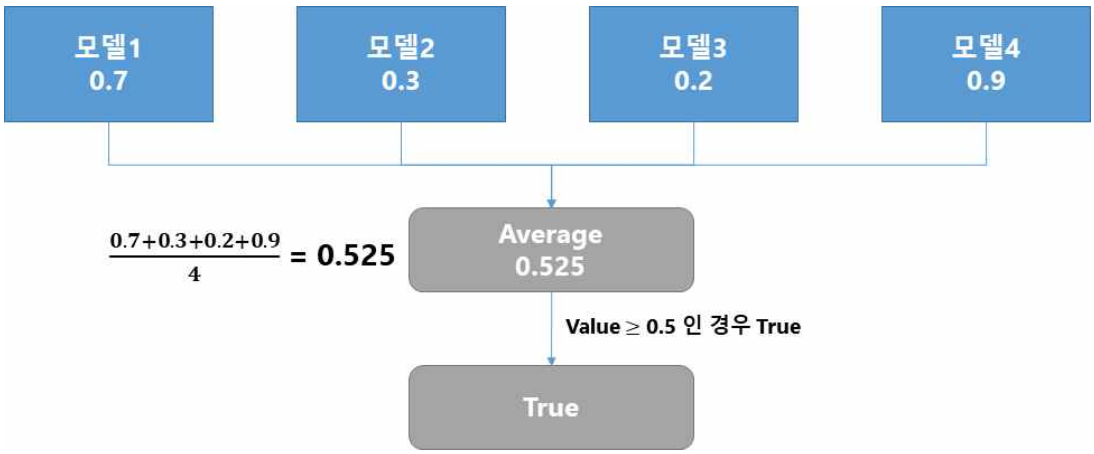
앙상블 모델은 4장에서 소개한 4개 모델의 조합으로 구성할 수 있다. 본 연구에서는 다양한 조합을 시도한 결과 가장 좋은 성능을 나타낸 두 가지 모델을 소개하고자 한다.

---

13) 여러 개의 개별 모델의 결과를 조합하여 최적의 모델로 일반화하는 방법. 일반적으로 입력 값의 형태, 은닉층의 구조 등이 상이한 모델들을 조합할 경우 좋은 결과를 얻을 수 있다.

1. soft voting 앙상블 모델

soft voting 앙상블 모델은 가장 많이 사용되는 방법 중의 하나이다. 개별 모델의 결과를 더하여 평균값을 구하여 최종 판정을 정하는 방식이며 본 연구에서는 4개의 모델을 모두 적용하여 결과 값을 도출하였다. [그림 5-5]는 soft voting 앙상블 모델의 예시를 나타내고 있다.



[그림 5-5] soft voting 앙상블 모델 예시

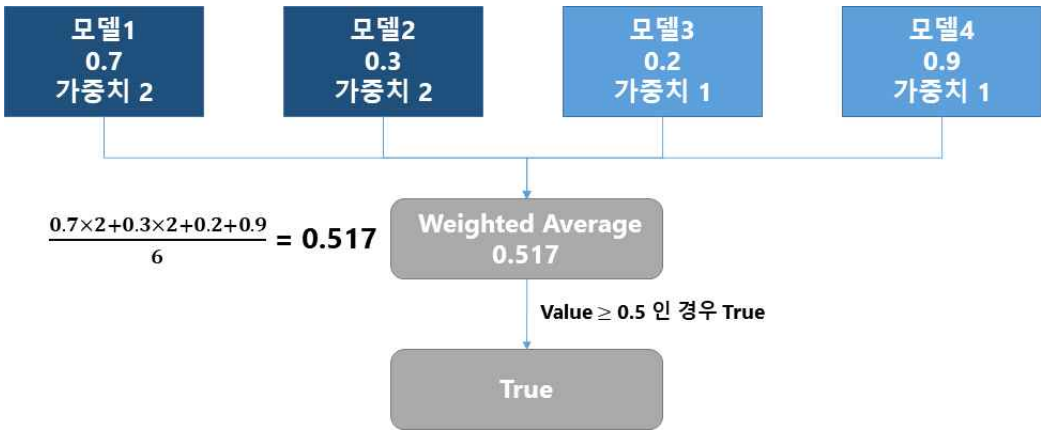
soft voting 앙상블 모델의 결과 값은 [표 5-3]과 같다. 단일 개별 모델과의 성능 비교를 위해 5장 5절에서 확인한 개별 모델 중 가장 성능이 높았던 아스키 코드 가중치 빈도 모델의 결과 값을 같이 표기하여 상대적으로 앙상블 모델의 효과를 확인할 수 있도록 하였다.

[표 5-3] soft voting 앙상블 모델 결과 값

metrics	아스키 코드 가중치 빈도 모델	soft voting 앙상블 모델
accuracy	0.9867	0.9922
precision	0.9934	0.9923
recall	0.9766	0.9904
F1-score	0.9849	0.9914

2. weighted soft voting 앙상블 모델

weighted soft voting 앙상블 모델은 앞서 설명한 soft voting 모델을 응용한 방법으로 soft voting 앙상블 모델이 결과치에 대해 단순 평균값을 사용하는데 비해 weighted soft voting 앙상블 모델은 특정모델의 결과치에 가중치를 부여하여 가중평균값을 구하여 최종적인 결과치로 사용한다. [그림 5-6]에서 weighted soft voting 앙상블 모델에 대해 예시를 통해 동작 방식을 이해할 수 있다.



[그림 5-6] weighted soft voting 앙상블 모델 예시

2절에서 비교한 모델별 결과를 바탕으로 상위 두 개(아스키 코드 가중치 빈도 모델, 아스키 코드 인코딩 합성곱 신경망 모델)의 모델에 대해 2의 가중치를 부여하고 나머지 두 개의 모델(아스키 코드 빈도 모델, 아스키 코드 인코딩 모델)은 그대로 가중치 1을 적용하여 실험을 수행하였다. weighted soft voting 앙상블 모델의 결과 값은 [표 5-4]와 같다. 이전과 마찬가지로 성능 비교를 위해 개별 모델 하나와 soft voting 앙상블 모델의 결과 값을 같이 표기하였다.

[표 5-4] weighted soft voting 앙상블 모델 결과 값

metrics	아스키 코드 가중치 빈도 모델	soft voting 앙상블 모델	weighted soft voting 앙상블 모델
accuracy	0.9867	0.9922	0.9922
precision	0.9934	0.9923	0.9922
recall	0.9766	0.9904	0.9905
F1-score	0.9849	0.9914	0.9914

단독 모델을 사용한 경우와 앙상블 모델을 사용한 경우 소수점 두 번째 자리 이하의 차이가 날 정도로 근소한 차이를 보이고 있다. 그러나 정밀도를 제외한 모든 측정치에서 전반적으로 앙상블 모델이 우수한 결과를 보이고 있어 충분히 채택할 가치가 있을 것으로 판단했다. 구성한 두 개의 앙상블 모델 사이의 차이는 거의 없는 것으로 확인되었다.

제 6 장 결 론

보고서의 1장에서는 본 연구를 수행하는 목적과 배경에 대해 설명하였고, 2장에서는 관련된 배경지식과 선행연구에 대해 소개하였다. 3장에서는 연구를 위한 데이터 수집과 딥러닝 모델에 입력하기 위한 전처리과정에 대해 기술하였으며 4장에서는 구성한 4가지 딥러닝 모델에 대해 설명하였다. 구성된 모델에 대한 실험결과 및 앙상블 모델에 관한 내용은 5장에 소개하였다. 그리고 마지막으로 이번 장에서는 연구 성과와 한계점, 후속 연구 방안 등에 대해 논의하고자 한다.

제 1 절 연구 성과 및 활용 방안

본 연구를 통해 제안된 4가지 모델 및 이를 활용한 앙상블 모델을 활용하면 유입되는 모든 네트워크 패킷에 대해 사전에 공격 여부에 대한 탐지를 수행할 수 있다. 수행 결과 sigmoid 함수에 의해 출력되는 실수값은 해당 패킷이 공격 패킷일 확률을 나타내게 된다. 이 점을 통해 현재 금융보안원에서 수행 중인 보안관제 활동에 적용할 수 있는 몇 가지 개선점에 대해 아래에서 검토한 결과를 기술하고자 한다.

1. 의심 패킷 분석 절차 개선

기존의 탐지 프로세스의 경우, IDS가 선별한 의심 패킷에 대해 해당 패킷의 메타데이터와 페이로드 데이터가 실제 관제 분석 요원에게 전달되고 이를 관제 요원이 하나 씩 살피면서 실제 공격인지 여부를 확인하는 방식으로 수행되고 있다.

본 연구 결과를 활용하면 기존 프로세스에서 전달되는 정보에 더하여 딥러닝 모델이 분석한 공격에 대한 확률 기반 예측 수치를 관제 분석 요

원에게 사전정보로 전달할 수 있다.

AS IS

Date	Time	IP(src)	IP(dst)	...	Detected
2022.5.19	10:33:11	123.456.2.32	211.32.4.123	...	SQL Injection
2022.5.19	10:34:12	123.456.2.32	211.32.4.123	...	Brute force
2022.5.19	10:34:21	123.456.2.32	211.32.4.123	...	App Vulnerable

TO BE

Date	Time	IP(src)	IP(dst)	...	Detected	DL Analysis
2022.5.19	10:33:11	123.456.2.32	211.32.4.123	...	SQL Injection	98.83%
2022.5.19	10:34:12	123.456.2.32	211.32.4.123	...	Brute force	37.21%
2022.5.19	10:34:21	123.456.2.32	211.32.4.123	...	App Vulnerable	88.83%

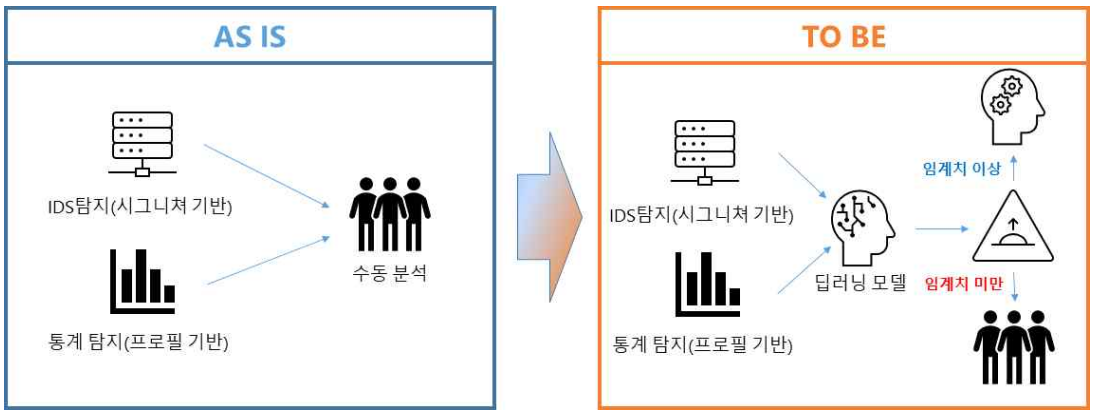
[그림 6-1] 관제 분석 화면 개선 예시

[그림 6-1]에 제시된 바와 같이 기존에 제공 되는 정보에 'DL Analysis' 라는 필드를 추가로 제공하여 관제 분석 요원이 실제 분석을 수행할 때 해당 패킷이 공격 패킷일 확률에 대해 미리 예측한 내용을 참고할 수 있게 된다. 이를 통해 실제로 수행한 분석 내용을 딥러닝 예측 값과 비교하여 분석을 정확하게 수행하였는지 검토해 볼 수 있는 여지를 제공할 수 있다. 자신이 분석한 내용 즉 정탐 혹은 오탐 여부에 대해 딥러닝 예측 값과 비교를 통해 실수를 줄이고 작업속도를 향상 시킬 수 있을 것으로 기대한다.

2. 딥러닝에 의한 자동 공격 분석

앞서 설명한 개념이 기존 보안 관제 분석 프로세스의 틀을 해치지 않는 범위에서의 개선 사항이라면 이번에 설명할 내용은 조금 더 보안 관제 분석 프로세스에 깊게 관여하는 개선 방안이 될 수 있다. IDS에서 탐지된 공격 의심 패킷을 관제 분석 요원에게 전달하기 전에 딥러닝 모델에 우선 적용하여 공격에 대해 사전 분석을 실시한다. 사전분석결과 공격 확률이 높은 패킷의 경우 자동 분석 프로세스를 가동하여 사람의 개입 없이 스스로 처리할 수 있도록 하고, 확률이 낮은 패킷에 대해서만 관제 분석 요원에게 전달한다.

딥러닝에 의한 탐지 확률이 낮을수록 공격 패킷인지 여부가 확실하지 않은 패킷일 것을 예상하여 이렇게 구성하였다. 이 경우 관제 분석 요원이 실제로 근무할 때 분석하여야하는 패킷의 절대적인 양이 감소할 수 있는 효과를 기대할 수 있다. 현재 프로세스와 차이점은 [그림 6-2]를 통해 개략적으로 파악할 수 있다.



[그림 6-2] 딥러닝을 통한 사전 분석 실시

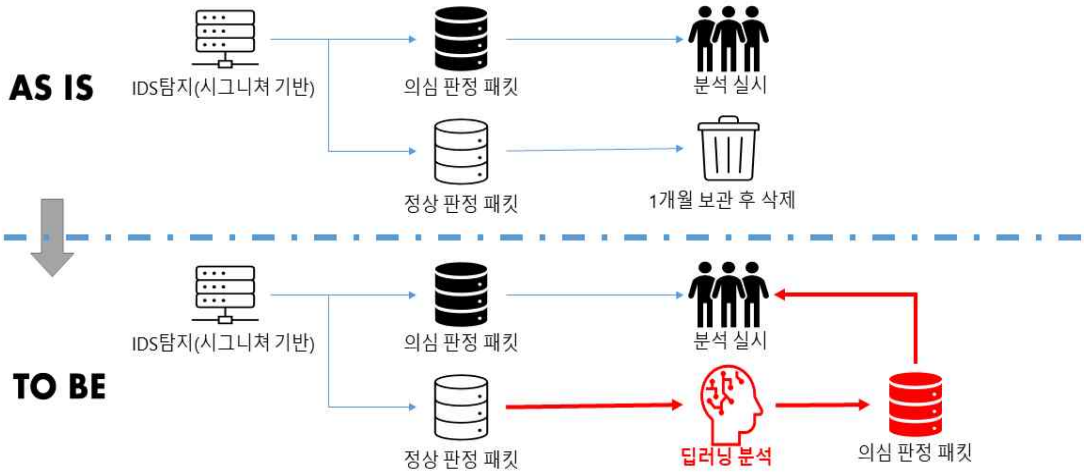
3. IDS 미탐지 패킷 탐지

예를 들어 일반적인 상황에서 어떤 금융 기관에 1000개의 네트워크 패킷이 유입된다면 그중 IDS에 의해 공격 의심 패킷으로 선별되는 패킷은 1개정도이다. 나머지 999개의 패킷은 정상적인 패킷이라고 판단하여 별도의 조치를 취하지 않고 내부 시스템으로 전달된다.

IDS는 기존에 알려지거나 분석이 완료된 공격만 탐지할 수 있으므로 새롭게 등장한 알려지지 않은 공격행위 혹은 기존의 공격에 변형이 이루어진 경우에 대한 탐지는 어렵다. 따라서 999개의 패킷 중에는 여전히 신규 혹은 변형 공격의 존재에 대한 위험성이 남아 있는 상태이다. 이 999개의 패킷에 대해서도 다시 한 번 검증해 볼 수 있다면 더욱 빈틈없는 보안관제 체계를 갖출 수 있을 것이다.

가벼운 예시로 1000개의 패킷이 유입되는 상황을 가정하였지만 실제로 금융권을 대상으로 유입되는 패킷은 하루에 1억 건을 상회한다. 따라

서 IDS가 정상으로 판단한 나머지 패킷은 1억 건에서 10만 건을 제외한 나머지 패킷에 해당한다. 이 패킷들을 모두 사람이 분석하기에는 분명한 한계가 존재한다. 딥러닝 모델은 이 부분에 대해 보완적인 역할을 수행할 수 있다.



[그림 6-3] 딥러닝을 통한 IDS 미탐지 패킷 탐지

[그림 6-3]에 기술된 바와 같이 정상 판정 패킷에 대해 다시 딥러닝에 의한 탐지를 실시하면 극소수의 비율로 의심 판정 패킷이 등장하게 된다. 실제로 본 연구를 통해 개발한 모델로 이 같은 과정을 수행하여 [표 6-1]과 같은 공격을 탐지하였다. 미탐지 패킷 대상 딥러닝 모델 예측은 앞서 선정한 데이터인 2021년 2월 21일의 정상데이터 127,870,579건 중 우선적으로 약 10만여 건에 대해 수행한 결과이다.

[표 6-1] 딥러닝 모델로 탐지한 IDS 미탐지 패킷 예시[12][13]

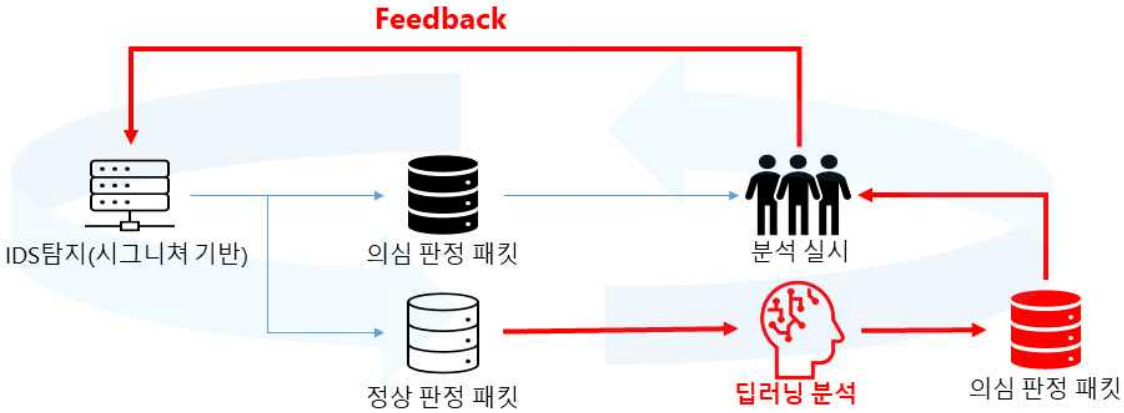
공격종류/공격명	공격 탐지 payload 예시	설명
Webscan	GET/elrekt.php?s=/index/thinkapp/invokefunction&function=call_user_func_array&vars[0]=md5&vars[1][]=HelloThinkPHP	php 특정 프로시져 탐지를 위한 스캔 공격
Zeroshell 관련 취약점 공격	GET/cgi-bin/kerbynet?Action=x509view&Section=NoAuthREQ&User=&x509type='a/etc/sudo.tar.-cf /dev/null/dev/null	Zeroshell의 취약점을 이용한 Root 권한 명령어 삽입 공격
Thinkphp 취약점 공격	GET/elrekt.php?s=/index/think/app/invokefunction&function=call_user_func_array&vars[0]=md5&vars[1][]=HelloThinkPHP	중국에서 사용하는 PHP 프레임워크의 원격 명령 실행관련 취약점을 이용한 공격
응용프로그램 취약점 공격	GET /shell?cd /tmp;wget http://xxx/jaw;sh jaw;	공격대상 서버에 botnet 설치를 시도

[표6-1]에 표기된 바와 같이 실험용 데이터 중 정상패킷에서 침입 탐지 시스템에서 탐지하지 못한 공격들이 몇 가지 탐지되었다. 기존의 침입 탐지 시스템이 이러한 공격들을 탐지하지 못하는 이유는 몇 가지 있을 수 있다.

우선 침입 탐지 시스템의 경우 많은 탐지 규칙을 탐지할수록 성능이 떨어진다. 따라서 주기적으로 탐지 규칙의 수를 조절하게 된다. 이 때 웹 스캔과 같은 단순히 시스템의 약점을 탐색하는 성격의 공격의 경우 위험성이 낮다고 판단하여 운용하는 주체의 판단에 따라 탐지 규칙 리스트에서 제외시키고 운영하는 경우가 많다. 또한 특정 국가에서만 사용하는 응용프로그램을 대상으로 하는 공격 또한 그 국가의 상황에 맞추어 제외시키기도 한다. 아직 밝혀지지 않은 공격이 탐지된다면 가장 유익하게 활용할 수 있을 것이고 그렇지 않더라도 침입 탐지 시스템이 왜 탐지하지 못했는가에 대해 분석을 하고 침입탐지 시스템의 룰 세트를 조정할 수 있다.

만약 탐지된 패킷에서 지금까지 알려지지 않은 공격이 발견 되거나 변형된 공격이 발견된다면 분석을 통해 새로운 시그니쳐를 생성하여 기존에 구성되어 있는 침입 탐지 시스템의 탐지 규칙을 개선 및 보완할 수

있다. 이를 통해 [그림 6-4]와 같이 인간과 인공지능이 서로 보완할 수 있는 선순환구조를 구축할 수 있다.



[그림 6-4] 딥러닝 분석을 통한 선순환 운영 구조

제 2 절 연구의 한계와 후속 연구 방안

앞서 본 연구의 결과를 바탕으로 실제 활용할 수 있는 방안을 소개하였다. 이번 절에서는 본 연구를 수행하면서 느꼈던 한계점과 본 연구의 결과를 토대로 모델의 성능을 지속적으로 유지 및 발전할 수 있도록 할 수 있는 후속 연구 방안에 대해 논의하고자 한다.

1. 연구의 한계

IDS에 의해 탐지된 결과를 관제 분석요원이 수동으로 레이블링한 데이터를 기반으로 학습을 시킨 결과 기존의 IDS에 탑재된 룰과 유사한 유형의 공격의 탐지에 강한 경향을 나타내었다. 따라서 완전히 새로운 패턴의 공격을 찾는 데는 한계가 있으며 기존공격과 유사하거나 변형된 수준의 공격의 탐지에 강하다. 이점을 보완하기 위해서는 탐지 방법을 다각화할 필요가 있다. 기존의 지도학습 방법에 더하여 오토인코더를 활

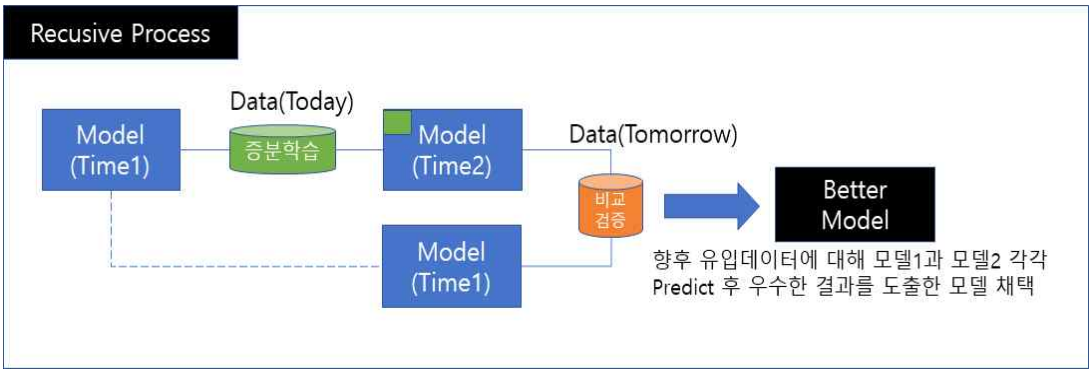
용하는 등의 비지도 학습 기반의 탐지방법을 접목해 보는 것도 좋은 시도가 될 수 있을 것이다. 궁극적으로는 완전히 새로운 공격을 탐지하는 것은 불가능할지도 모른다. 그러나 지금까지 관찰된 공격들을 보면 대부분은 진화 및 변형의 과정을 거쳐 탄생하고 있다. 이런 부분에서 유사성에 중점을 둔 탐지 방법일지라도 충분히 연구할 가치가 있다.

본 연구의 경우 과거에 실제로 네트워크를 통해 송수신된 실제 데이터를 기반으로 연구가 수행되었다. 따라서 데이터의 수집, 취급, 공개 등에 대해 한계가 분명히 존재하였다. 현재는 개인정보와 기업의 데이터가 중요한 자산이 되고 있는 이른바 데이터의 시대임을 고려할 때, 연구에 데이터를 사용하는 것은 많은 제약과 유의점을 수반한다. 더 많은 데이터를 적용하고 접근법을 확대하여 연구를 발전시킬 수 있는 가능성을 기대하고 있다.

2. 후속 연구 방안

본 연구에서는 실제 공격 행위를 탐지하는 모델을 생성하는 것에 집중하였다. 그러나 실제 환경에서는 끊임없이 다량의 데이터가 유입되고 공격의 경향도 시시각각 변화한다. 어제까지 유입된 데이터를 학습한 모델에 대해 오늘의 데이터와 내일의 데이터를 학습시켜야 지속적인 모델의 유효성을 담보할 수 있다. 그러나 최신 데이터를 학습하는 것만이 능사는 아니다. 만약 어떤 날의 데이터에 디도스 공격과 같은 노이즈 성격의 패킷이 많이 포함되어 있다면 그 날의 데이터는 오히려 학습을 하지 않는 편이 낫다.

따라서 오늘의 데이터를 학습한 모델과 그렇지 않은 모델 중 어떤 것이 더 우수할 지를 판단할 모델이 필요하다. 이 과정을 단순하게 표현하여 오늘 하루치의 데이터 학습 여부에 대해서만 고려하면 [그림 6-5]와 같이 모델의 검토 과정을 나타낼 수 있다.



[그림 6-5] 학습 기간별 모델간의 비교

그러나 시간의 흐름에 따라 모델을 나누는 방법은 다양해진다. 증분학습 모델간의 비교에서 학습기간의 단위, 한 타임라인 내에서 몇 개의 모델을 보존하여 서로 비교할 것인가의 문제, 모델의 비교우위 결정 방법에 따라 복잡한 체계를 기반으로 구축될 수 있다.

본 연구에서는 모델 내부의 구현에 중점을 두었다면 증분학습 기간별 최적 모델의 설정은 모델 자체를 최적으로 운용하는 방법을 강구하는 문제가 된다. 이 문제는 실제 환경에 적용하여 활용할 때 꼭 필요할 것으로 예상이 된다.

모델자체의 개선을 논하자면 앞서 연구의 한계라고 느꼈던 탐지 방법의 다각화에 대해 고려해볼 수 있다. 본 연구에 의해 생성된 모델과 오토인코더 모델, machine learning 모델 등을 활용하여 다양한 앙상블 모델을 구축 및 활용해 보는 것도 후속 연구로써 의미가 있을 것이다.

또한 앞서 연구 범위에서 제외한 흐름기반의 탐지를 본 연구와 접목시키는 방법도 생각해 볼 수 있다. 하나의 공격이 이루어지기 전에는 사전 작업이 필요하다. 즉 공격은 단일 패킷으로 이루어져 있기 보다는 실제 공격이 일어나는 패킷의 전후로 공격과 연관된 패킷들이 송수신 되는 경우가 많다. 이러한 점을 고려하여 특정 공격자가 보내는 패킷의 흐름과 절차를 고려하여 분석을 시도한다면 더욱 정밀한 탐지가 가능할 것으로 예상이 된다.

본 연구에서는 금융소비자의 재산을 보호하기 위해 선봉적인 역할을 수행하는 금융보안원이 수행하는 금융권 보안관제를 강화할 목적으로 딥러닝 기반 탐지 모델을 제시하였다. 4가지의 모델을 제안하였으며 제안한 모델을 기반으로 구현한 앙상블 모델을 통해 99.22% 수준의 정확도를 달성하였다. 모델의 적용방안과 효과에 대해 면밀한 검토와 추가적인 후속 연구를 통해 더욱 치밀한 금융권 보안관제 체계의 구축에 기여할 수 있을 것으로 기대한다.

참 고 문 헌

[1] Network Encyclopedia. Network Packet [Online]. Available: https://networkencyclopedia.com/network-packet/ (downloaded 2021, Jan. 1)

[2] Wikipedia contributors. (2022, January 18). Intrusion detection system. In Wikipedia, The Free Encyclopedia. Retrieved 07:59, January 20, 2022, from https://en.wikipedia.org/w/index.php?title=Intrusion_detection_system&oldid=1066455133

[3] Financial Security Institute. 2016 Financial security institute Annual report. Financial Security Institute, 2017.

[4] Cisco and/or its affiliates.. Snort [Online]. Available: https://snort.org (accessed 2022, Jan. 21)

[5] S. Park, M. Kim, and S. Lee, "Anomaly detection for http using convolutional autoencoders", IEEE Access, vol. 6, pp. 70 884 - 70 901, 2018.

[6] J. Liu, X. Song, Y. Zhou, X. Peng, Y. Zhang, P. Liu, and D. Wu, "Deep anomaly detection in packet payload" , arXiv preprint arXiv:1912.02549, 2019.

[7] C. Oh, J. Lee, H. Moon, H. Roh, K. Kim and T. Kwon, "A Heuristic Approach to Classify Malicious HTTP Web Traffic for Supporting Efficient Cybersecurity Monitoring" , Proc. of the KIISE Korea Computer Congress 2020, pp. 992-994, 2020.

[8] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). Hypertext transfer protocol-HTTP/1.1.

[9] https://snort.org/downloads/#rule-downloads

[10] Chollet, Francois. "Deep learning with Python", Simon and Schuster, 2021.

[11] Alaidaros, Hashem, Massudi Mahmuddin, and Ali Al Mazari. "An overview of flow-based and packet-based intrusion detection performance in high speed networks." Proc. of the International Arab Conference on Information Technology. 2011.

[12] https://www.abuseipdb.com
[13] https://www.exploit-db.com
[14] Chen, Yahui. "Convolutional neural network for sentence classification." MS thesis. University of Waterloo, 2015.
[15] Kakavand, Mohsen, et al. "Effective dimensionality reduction of payload-based anomaly detection in TMAD model for HTTP payload." KSII Transactions on Internet and Information Systems (TIIS) 10.8 (2016): 3884-3910.
[16] Wang, Ke, and Salvatore J. Stolfo. "Anomalous payload-based network intrusion detection." International workshop on recent advances in intrusion detection. Springer, Berlin, Heidelberg, 2004.
[17] Ingham, Kenneth L., and Hajime Inoue. "Comparing anomaly detection techniques for http." International workshop on recent advances in intrusion detection. Springer, Berlin, Heidelberg, 2007.
[18] Zolotukhin, Mikhail, et al. "Analysis of HTTP requests for anomaly detection of web attacks." 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing. IEEE, 2014.
[19] Li, Jieling, Hao Zhang, and Zhiqiang Wei. "The weighted word2vec paragraph vectors for anomaly detection over HTTP traffic." IEEE Access 8 (2020): 141787-141798.
[20] Vartouni, Ali Moradi, Saeed Sedighian Kashi, and Mohammad Teshnehlab. "An anomaly detection method to detect web attacks using stacked auto-encoder." 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS). IEEE, 2018.
[21] Luo, Yuxuan, et al. "PU learning in payload-based Web anomaly detection." 2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC). IEEE, 2018.

Abstract

A Financial Web Packet Anomaly Detection Scheme Using Deep Learning

정 재 식(Jeong Jaesik)
Department of Engineering Practice
Graduate School of Engineering Practice
Seoul National University

Compared to other industries, information security in the financial industry is very important considering its business characteristics. In attacks on the financial sector, packet-based attacks have overwhelmingly higher frequency than flow-based attacks, and in order to respond to these packet-based attacks, Intrusion Detection System equipped with a rule set for pattern detection is used to effectively detect and respond to anomal packets. However, due to the development of the fintech industry and the development of various financial services, absolutely increasing traffic and zero-day attacks are continuously appearing. To respond to this, in this paper, I proposed four deep learning models using various ASCII code-based input data. By utilizing the proposed models, it is expected that the level of security control in the financial sector will be further improved by supplementing the deficiencies of the existing intrusion detection system.

keywords : deep learning, ASCII code, HTTP, financial, packet based, anomaly detection
Student Number : 2021-20708