제로트러스트 모델을 위한 딥러닝 기반의 비정상 행위 탐지

김서영, 정경화, 황유나, 양대헌
이화여자대학교 사이버보안전공
larabbit@ewhain.net, 7942jgh@ewhain.net, ovo6v6@ewhain.net, nyang@g.ewha.ac.kr

Abnormal Behavior Detection for Zero Trust Security Model Using Deep Learning

Seo-Young Kim, Kyung-Hwa Jeong, Yuna Hwang, Dae-Hun Nyang
Dept. of Cyber Security, Ewha Womans University

요 약

최근 네트워크의 확장으로 인한 공격 벡터의 증가로 외부자뿐 아니라 내부자를 경계해야 할 필요성이 증가함에 따라, 이를 다룬 보안 모델인 제로트러스트 모델이 주목받고 있다. 이 논문에서는 reverse proxy 와 사용자 패턴 인식 AI 를 이용한 제로트러스트 아키텍처를 제시하며 제로트러스트의 구현 가능성을 보이고, 새롭고 효율적인 전처리 과정을 통해 효과적으로 사용자를 인증할 수 있음을 제시한다. 이를 위해 사용자별로 마우스 사용 패턴, 리소스 사용 패턴을 인식하는 딥러닝 모델을 설계하였다. 끝으로 제로트러스트 모델에서 사용자 패턴 인식의 활용 가능성과 확장성을 보인다.

1. 서론

최근 IoT, 클라우드 기술 등의 등장으로 서버에 새로운 공격벡터가 생겨나고 있으며 내부자 권한 탈취를 통한 측면 이동 공격 등의 보안 위협이 증가하고 있다. 특히 COVID-19 사태 이후에는 재택근무의 확산 등 비대면 원격 접속을 이용하는 경우가 증가하였고, 이 때문에 내/외부를 막론하고 모든 사용자를 경계할 필요성이 생기며 이를 다룬 보안 모델인 '제로트러스트(Zero Trust)'가 새롭게 주목받고 있다. 제로트러스트 모델이란, 모든 사용자를 신뢰하지 않고 철저한 확인을 통해 필요한 만큼의 접근 권한을 제공하는 보안 모델이다.

제로트러스트를 적용한 기존의 대표적 사례로서 Google 의 BeyondCorp 를 살펴보면 가장 핵심이 되는 부분은 접근제어 엔진이다. BeyondCorp 의 접근제어 엔진은 기기 정보, 사용자 정보, 취약점 정보를 보고 사용자의 차단 여부를 결정한다.[1] 하지만 이는 정적인 정보로 사용자의 디바이스를 탈취하여 사용자인 척 접근하는 위장 공격에 취약한 한계를 가지므로, 동적이면서 사용자 본인을 인증할 방법이 필요하다.

따라서 이 논문에서는 사용자 패턴 인식 AI 를 이용한 제로트러스트 아키텍처를 제시하여 reverse proxy 및 AI 를 통한 사용자의 마우스 사용 패턴 분석과 리소스 사용 패턴 분석을 통해, 로그인한 내부자에 대해서도 실시간 사용자 인증을 수행하여 제로트러스트를 구현할 수 있음을 보인다. 또한, 사용자 마우스 및 리소스 사용 패턴 분석에서 새롭고 효율적인 전처리 과정을 통해 높은 성능을 달성함을 보이고, reverse proxy 와 시연 서버의 구현을 통해 이 논문에서 제시하는 모델이 실제로 일반 회사를 비롯한 모든 산업체의 서버에 적용되어 제로트러스트를 구현할 수 있음을 보인다.

이미 시중에 제로트러스트 구현물이 존재하기에 모든 것을 구현하기보다는 핵심 부분인 접근제어 엔진 개선에 비중을 두었으며, 외부 네트워크를 이용한 공격을 탐지하는 IPS, IDS, 방화벽 등은 이 논문에서 다루는 주제에서 벗어나므로, 내부네트워크, 내부자들로 인한 공격에 집중하여 아키텍처를 설계, 구현했다.

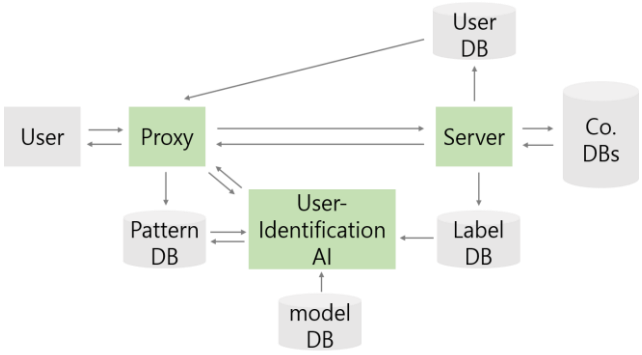
2. 사용자 인증 수단 선정 이유

사용자들은 보편적으로 패스워드를 입력하여 회사 서버에 접속한다. 하지만 패스워드는 사용자 자체보다는 사용자의 아이디를 인증하는 방식이다. 그러므로 공격자가 패스워드만 알면 사용자인 척 회사 서버에 접속해 사용할 수 있다. 또한 패스워드 인증은 세션 동안 한 번만 이루어지기에 사용자가 로그인한 이후 잠시 자리를 비운 사이 공격자가 접근하여 공격할 수 있다(런치박스 공격). 이러한 위험이 있기에 지속

해서 사용자 본인이 맞는지 인증할 수 있는 방법이 필요하며, 이 논문에서는 정해진 시간마다 사용자의 리소스 사용량과 마우스 움직임을 통해 사용자 본인을 인증하는 방식을 선택하였다. 마우스와 리소스 데이터는 사용자가 인증을 위한 별도의 과정을 수행하지 않아도 되고 추가적인 보조 하드웨어 장치 없이 수집 가능하기 때문에 사용성이 높다는 장점이 있다.

3. 비정상행위 탐지 시스템 구조

관리자 기능은 제로 트러스트 보안 모델을 사용하는 모든 다양한 서버에서 사용 가능하다. 이 절에서는 딥러닝 기반의 비정상행위 탐지 시스템의 구조에 관해 설명한다. 이 서버 운영은 회사의 보안 관제 등에서 관리하며, 이하 관리자로 통칭한다.



(그림 1) 제로트러스트 프록시를 적용한 서버 아키텍처.

(그림 1)은 전체 서비스 구조도를 나타낸 것이다. 사용자가 프록시에게 보내는 요청은 사용자 인증과 서비스 이용, 두 가지로 분류할 수 있다.

3.1 사용자 인증 과정

먼저 사용자 패턴을 프록시에 전송하여 인증 받는 과정을 살펴보면 다음과 같다. 이 과정은 정해진 일정 시간(T)마다 일어난다.

1. 사용자는 사용자 에이전트[1]를 이용해 프록시에게 계정 정보(ID), 자신의 패턴 정보를 전송한다.

2. 프록시는 회사의 사용자 DB 에서 요청한 사용자의 계정이 활성화되어있는지(접속이 가능한지) 확인한다. 만약 차단된 계정이라면 프록시는 사용자의 접속을 금지한다. 이때 사용자의 패턴은 저장하지 않는다.

3. 접속 가능한 계정임이 확인되면 프록시는 사용자의 패턴을 패턴 DB 에 저장한다.

4. 프록시는 AI 에게 사용자의 계정 정보를 알려주며 사용자 패턴 인증을 요청한다.

5. AI 는 프록시에서 저장한 사용자의 패턴을 통해 검사를 수행하여 그 결과에 따라 다음 경우로 나뉜다.

5-1. AI 분석 결과와 계정 정보가 일치할 경우, 사용자 패턴의 label 을 해당 계정 정보와 일치시킨다. 이때 사용자는 문제없이 계정을 사용할 수 있다.

5-2. AI 분석 결과가 계정 정보와 다를 경우, AI 는 사용자의 계정 정보, 판단 정확도, AI 가 예측한 계정 사용자(label)를 프록시를 거쳐 서버에게 보낸다.

6. 관리자는 관리자페이지 등을 통해 프록시가 보낸 AI 의 분석 결과를 확인할 수 있고, 사용자 패턴 label 을 올바르게 수정할 수 있다. 관리자는 오프라인(통화 등의 방법)으로 실제 사용자를 확인 후 다음 경우로 나누어 행동한다.

6-1. AI 의 분석과는 다르게 정상 사용자임이 확인된다면 패턴의 label 을 정상으로 수정한다. 이때 사용자 계정에는 아무 일도 일어나지 않는다.

6-2. AI 가 '벌점 부여'를 권고하고 관리자도 이에 동의할 경우, 해당 사용자의 패턴 label 에 다른 사용자나 공격자에 해당하는 unknown 으로 설정되며, 해당 계정에 '벌점'이 부여된다. 벌점이 임계값을 넘으면 계정은 자동 차단되어 이후 계정 사용이 불가하다.

6-3. AI 의 분석 결과에서 '계정 차단'을 권고하고 관리자가 이에 동의했을 경우, 관리자는 해당 사용자의 패턴 label 을 다른 사용자나 unknown 으로 설정하여, 관리자페이지에서 즉각 계정을 차단할 수 있다.

관리자가 수정한 label 은 근무 시간 동안 쌓은 사용자의 데이터와 함께 AI 의 재학습 시에 사용된다.[2] 재학습은 근무시간에 발생하는 오탐을 줄이기 위함이며 매일 자정 이후에 진행된다.

3.2 서비스 이용 과정

사용자가 서비스를 이용하는 과정은 다음과 같다. 이 과정은 사용자가 서비스를 이용하고자 요청을 보낼 때마다 일어난다.

1. 사용자는 프록시에게 ID 와 이용하고자 하는 서비스 종류를 전송한다.

2. 프록시는 회사의 사용자 DB 에서 요청한 사용자의 계정의 활성화 여부를 확인한다. 차단된 계정이라면 사용자의 접속을 금지한다.

3. 사용 가능한 계정이면 서버는 사용자에게 서비스를 제공한다.

3.3 Bypass 권한 부여 과정

정상 사용자임에도 계정이 차단되어 서비스 이용이 불가능할 경우, 사용자는 관리자에게 연락하여 계정 활성화를 요청할 수 있다. 하지만 빈번한 AI 의 오탐으로 인해 업무에 지장을 받는다면 관리자에게 연

[1] 사용자가 이용하는 디바이스에서 에이전트를 이용해 사용자의 패턴을 수집한다.

락하여 하루 동안 AI 검증 없이 서비스를 이용할 수 있는 Bypass 권한을 얻을 수 있다.

관리자는 사용자의 연락을 받고 사용자의 계정 앞으로 Bypass 권한을 얻을 수 있는 key 를 발급해준다. key 는 사용자가 기억하기 쉬우면서도 유추하기 어려운 임의의 단어로 발급한다. 그 후 오프라인으로 사용자에게 key 를 전달한다.

사용자는 해당 key 를 이용하여 token 형식으로 Bypass 권한을 발급받게 된다. token 은 인증 시스템에서 주로 사용하는 Json Web Token 을 사용해 구현했고, 발급받는 즉시 자동으로 사용자 클라이언트 쿠키에 저장된다. 쿠키는 TLS 인증서를 사용하기 때문에 안정성이 보장된다.

token 은 사용자가 서버를 사용하는 동안에는 자동으로 refresh 되지만 idle time 이 일정 시간을 넘을 경우에는 만료되어 사용이 불가하다. 때문에 사용자는 다시 key 를 입력함으로써 token 을 재발급받아야 Bypass 권한을 유지할 수 있다.

Bypass 권한이 있는 사용자의 서비스 이용 과정은 다음과 같다.

1. 프록시에게 사용자 인증을 요청할 때, 사용자 쿠키에 저장되어 있는 token 이 같이 전송된다.

2. 프록시는 해당 사용자가 token 이 유효한지 확인하기 위해 서버에게 verify 요청을 보낸다.

3. 서버는 받은 token 을 검증하여 Bypass 여부를 상태 코드를 이용해 프록시에게 알려준다.

4. 프록시의 행동은 권한 여부에 따라 다음과 같이 나뉜다.

4-1. Bypass 권한이 있는 사용자의 경우, token 유효기간 동안 AI 검증이 일어나지 않아 사용자는 원활하게 서비스를 이용할 수 있다. 프록시는 Bypass 권한이 유지되는 동안 받은 사용자 패턴의 label 을 해당 사용자로 저장한다.

4-2. Bypass 권한이 없는 사용자의 경우, 프록시는 AI 에게 사용자의 계정 정보를 알려주며 사용자 인증을 요청한다. 이후는 앞서 설명한 3.1 사용자 인증 과정과 동일하게 진행된다.

key 는 발급된 날의 자정까지만 사용이 가능하며 그 이후에는 자동 파기된다. 따라서 자정 이후로는 key 를 이용하여 Bypass 권한을 더 이상 얻을 수 없다.

4. 사용행위 패턴을 이용한 인증

AI 는 프록시를 통해 정해진 시간마다 사용자 ID, 리소스, 마우스 사용 행위 파일을 입력 받는다. 전달받은 각 파일을 1 차 파싱하고 불필요한 열은 삭제한 후 저장한다. 1 차 처리가 완료된 행위 파일에서 해당 파일을 대표할 수 있는 특징을 추출해 하나의 행으로 만든다. 이것을 저장된 MinMaxscaler 를 이용해 정규화 해준다. 그리고 사용자별로 저장된 모델을 적용하여 사용자 본인이 맞는지 판단한다. 그 결과는 0 과 1 사이의 값으로 표현되고, 사용자의 패턴과 유사할수록 1 에 가깝게 나타난다.

4.1 데이터셋

리소스 데이터셋은 Windows OS 에 한정하여 수집했으며 Windows 내장 앱인 성능 모니터를 사용했다. 10 명의 사용자가 각각 3 일동안 리소스를 사용하는 데이터를 얻었다. 얻은 attribute 로는 서버에서 실행되고 있는 프로세스들이 사용하고 있는 메모리양, 프로세스가 사용하는 가상메모리 크기와 처리용량의 비율, 프로세스가 데이터를 저장하기 위해 사용하는 RAM 의 양, 프로세스가 활성상태를 유지한 시간의 비율 등이 있다. 이는 하드웨어 고유의 특성보다는 프로세스와 관련된 데이터이다.

마우스 데이터셋은 Mouse-Data-Collector[4]를 사용하여 3 명의 사용자가 각각 3 일동안 마우스를 사용하는 데이터를 얻었고, Mouse Challenge Data Set[5]로부터 10 명의 사용자의 마우스 사용 데이터를 얻었다. 얻은 attribute 로는 event 발생 시간, button(NoButton/Left/Right), state(move/drag/pressed/released), 마우스의 x 좌표, y 좌표가 있다.

4.2 사용 행위 패턴 분류 모델

사용자의 리소스와 마우스 사용 데이터를 통해 사용자를 인증할 수 있는 최적의 모델을 찾고자 다양한 알고리즘과 특징 추출 방법을 적용해봤다.

가장 보편적인 DNN, 정해진 시간 동안의 마우스 움직임 패턴을 하나의 이미지로 나타내기 때문에 이미지 분류에 우수한 능력을 보이는 CNN, 마우스와 리소스 패턴에서 앞선 행동이 이어지는 행동에 영향을 미칠 수 있다고 판단하여 LSTM 을 적용해 보았다.

(1) 리소스

<표 1> 리소스 모델별 성능

모델	데이터	데이터 수	Input	Output	성능
DNN	Real	89154	(1,7)	(1,10)	Acc = 0.5551
LSTM	Real	16701	(1,7)	(1,3)	Acc = 0.9601 F1-score = 0.9601
DNN	Real	4275	(1,140)	(1,1)	Acc =0.9993 F1-score = 0.9967

성능 모니터로 수집한 데이터를 Real로 기재했다.

정해진 시간 동안의 리소스 사용량을 이어 붙여 한 개의 행로 표현하는 특징 추출 방법과 DNN 을 사용하는 것이 가장 좋은 성능을 보였다.

(2) 마우스

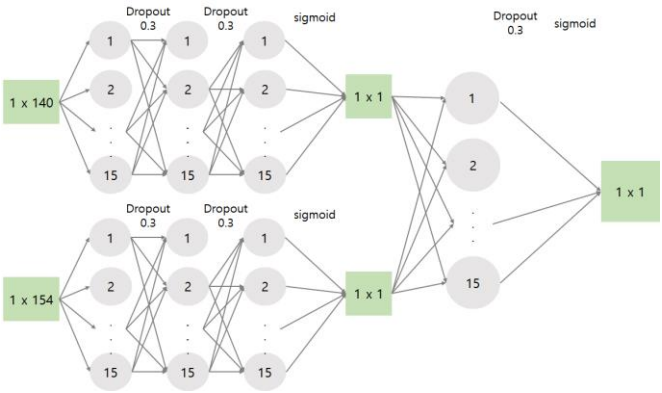
<표 2> 마우스 모델별 성능

모델	데이터	데이터 수	Input	Output	성능
2DCNN	Balabit	1702	(1,219,13,32)	(1,4)	Acc = 0.5953
1DCNN -LSTM	Balabit	6557	(1, 332, 13)	(1,10)	Acc = 0.5005 F1-score = 0.5061
LibSVM	Real	1078	(1,154)		Acc = 0.395
DNN	Balabit	1721	(1,154)	(1,1)	Acc = 0.9917 F1-score = 0.7108
	Real	969			Acc = 0.9473 F1-score = 0.9166

Real는 [4]를 통해 수집한 데이터, Balabit는 [5]에서 수집한 데이터를 의미한다.

마우스는 각 event 마다 정해진 시간동안 차지하는 비율(7)과 소요시간의 통계값(7x5), 8 개의 방향에 대한 소요시간과 거리의 중간값(7x8x2) 등 총 154 개의 열을 추출했다.[3] 이것을 DNN 을 이용하여 분류하는 것이 가장 좋은 성능을 보였다.

(3) 병합 모델



(그림 2) 리소스와 마우스 병합 모델 구조.

앞서 구한 각 최적의 모델들의 아웃풋을 다중입력 처리하여 하나의 결과로 만들어내는 최종 사용 행위 패턴 분류 모델을 만들었다. 해당 모델은 kernel_initialize 로 'glorot_uniform'을 사용하고 히든 레이어 사이에는 30% dropout 을 적용했으며 각 아웃풋 레이어에는 'sigmoid'를 적용했다. 그 외의 레이어는 'relu'를 활성함수로 적용했다.

5. 실험 결과

<표 3> 동시 수집 데이터에 대한 모델 성능

데이터	accuracy	precision	recall	f1-score
리소스(DNN)	1.00	1.00	1.00	1.00
마우스(DNN)	0.8986	0.8620	0.8576	0.8598
병합모델	1.00	1.00	1.00	1.00

3 명의 데이터 중 1140 개는 리소스와 마우스를 동시에 수집해 병합 모델을 평가하기 위해 사용되었다.

5.1 성능평가

리소스 모델은 Acc=1.0, F1-score=1.0, 마우스 모델은 Acc=0.8986, F1-score=0.8598 의 성능을 보였다. 앞서 테스트한 것보다 마우스의 성능이 떨어지는 이유는, 정해진 시간 동안 수집되는 데이터의 양이 리소스와 마우스가 다른데, 병합을 위하여 이를 맞추기 위해 마우스에 움직임이 없는 상태인 idle 상태도 포함하였기 때문이다. 실제로 테스트를 해본 결과, 병합 모델의 성능은 높았지만 리소스 사용 패턴 분석에 의존하는 정도가 높았다. 이는 리소스 모델의 정확도가 거의 1 에 가깝기 때문이다. 그러므로 병합 모델이 아니라 패턴에 따라 개별 모델을 사용하고 정책을 정의하여 사용자를 인증하는 방식을 고려해볼 수 있다.

6. 결론

이 논문에서는 사용자의 마우스 및 리소스 사용 패턴 분석을 통한 제로트러스트 인증 아키텍처를 제시하였고, 이 시스템이 높은 보안성과 편리함을 제공할 수 있음을 보였다.

또한 제안한 아키텍쳐는 일반 컴퓨터 기기뿐만 아니라, 모바일 디바이스나 다양한 IoT 기기에 적용할 수 있다. 예를 들어 모바일 디바이스의 경우 사용자 리소스 사용 패턴이나 키보드 사용 패턴 등을 통해서 사용자의 고유한 패턴을 알 수 있다. 따라서 적절한 사용자 에이전트가 마련된다면 제안한 아키텍처는 확장성을 가질 수 있고, 동적 바이오 인증 기법을 통한 제로트러스트 모델 구현의 시작이라는 의의를 가진다.

참고문헌

[1] Rory Ward, Betsy Beyer, "BeyondCorp: A New Approach to Enterprise Security", *;login:*, Vol. 39, No.6, pp.6-11, 2014.

[2] Anima B. Akter, Jasim Mahmood, Rahman K. Abir, Rulapaugh Adam, Hasanuzzaman Md. "User Authentication from Mouse Movement Data Using SVM Classifier", Lecture Notes in Computer Science, Vol.2016, No.10052, pp.692-700, 2016.

[3] AeChan Kim, MoHyun Park, DongHoon Lee "AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection", *IEEE Access*, Vol.8, pp.70245-70261, 2020

[4] "Mouse-Data-Collector", Github, last modified Dec 14. 2018, accessed Mar 26. 2021, https://github.com/dflehel/Mouse-Data-Collector.

[5] "Mouse-Dynamics-Challenge", Github, last modified Sep 21. 2018, accessed Mar 26. 2021, https://github.com/balabit/Mouse-Dynamics-Challenge.