2024년 02월

석사학위 논문

보안 취약점 데이터 기반 보안 지식 그래프 구축 및 확장

조선대학교 산업기술융합대학원

소프트웨어융합공학과

안 현 진

보안 취약점 데이터 기반 보안 지식 그래프 구축 및 확장

Construction and Expansion of a Security Knowledge Graph Based on Vulnerability Data

2024년 02월 23일

조선대학교 산업기술융합대학원

소프트웨어융합공학과

안 현 진

보안 취약점 데이터 기반 보안 지식 그래프 구축 및 확장

지도교수 최 준 호

이 논문을 공학석사학위신청 논문으로 제출함.

2023년 10월

조선대학교 산업기술융합대학원

소프트웨어융합공학과

안 현 진

안현진의 석사학위논문을 인준함

위원장 조선대학교 교수 김 판 구 (인)

위 원 조선대학교 교수 신 주 현 (인)

위 원 조선대학교 교수 최 준 호 (인)

2023년 11월

조선대학교 산업기술융합대학원

목 차

ABSTRACT

Ⅰ. 서론 ····· 1
A. 연구의 배경 및 목적 ····· 1
B. 연구 내용 및 구성 ····· 2

Ⅱ. 관련 연구 ····· 3
A. 지식 그래프 ····· 3
B. 딥러닝 기반 지식 그래프 생성 ····· 4

Ⅲ. 보안 취약점 데이터 기반 보안 지식 그래프 구축 및 확장 ····· 6
A. 보안 지식 그래프 생성 및 확장 프레임워크 ····· 6
B. 보안 취약점 데이터 수집 및 전처리 ····· 7
1. CVE 데이터 수집 및 전처리 ····· 7
2. 크롤링을 이용한 Exploit 데이터 수집 ····· 9
C. 텍스트 유사도 기반 CVE 데이터 분류 ····· 10
1. 텍스트 유사도 기반 분류를 위한 전처리 ····· 11
a. CVE Description 열 토큰화 ····· 11
b. 텍스트 유사도 기반 CVE 데이터 분류화를 위한 워드 임베딩 ·· 11
2. KMeans 알고리즘을 이용한 CVE 데이터 군집화 ····· 13
D. 딥러닝 기반 보안 지식 그래프 구축 ····· 15
1. REBEL 모델 ····· 15
2. REBEL을 이용한 보안 지식 그래프 생성 ····· 17
3. 보안 지식 그래프 시각화 ····· 18
E. 구글뉴스를 이용한 보안 지식 그래프 확장 ····· 19
1. 구글 뉴스 데이터 크롤링 ····· 19
2. News 데이터로부터의 triplet 추출 및 지식 그래프 확장 ····· 20
F. 보안 지식 그래프 임베딩 ····· 21

1. TransE 임베딩 ····· 21
2. DistMult, ComplEx 임베딩 ····· 22
3. HolE 임베딩 ····· 22

Ⅳ. 실험 및 평가 ····· 24

Ⅴ. 결론 ····· 27

참고문헌 ····· 29

표 목 차

[표 1] CVE Data Base에서 제공한 CVE-2022-0004 정보 ····· 7

[표 2] Exploit ID 크롤링 결과 예 ····· 9

[표 3] Exploit Type, Description 수집 결과 예 ····· 10

[표 4] CVE Description 토큰화 결과 예 ····· 11

[표 5] CVE 데이터 군집화 결과 ····· 13

[표 6] CVE 데이터로부터 triplet 추출한 결과 예 ····· 17

[표 7] “ARP Spoofing”으로 검색한 뉴스 크롤링 결과 예 ····· 19

[표 8] 임베딩 모델 평가 결과 ····· 25

[표 9] 지식 그래프 확장 후 평가 결과 ····· 26

그 림 목 차

[그림1] 지식 그래프 예 ····· 3

[그림2] 보안 지식 그래프 구축 및 확장 프레임워크 ····· 6

[그림3] 전처리 후 데이터 프레임 예 ····· 8

[그림 4] Kmeans 알고리즘을 이용한 군집화 - 2차원 시각화 ····· 14

[그림 5] REBEL 모델을 이용한 triplet 추출 예 ····· 16

[그림 6] 지식 그래프 시각화 ····· 18

[그림 7] 지식 그래프 확장 결과 시각화 ····· 20

[그림 8] TransE 임베딩 예 ····· 21

ABSTRACT

Construction and Expansion of a Security Knowledge Graph Based on Vulnerability Data

HyunJin An
Advisor : Prof. JunHo Choi, Ph.D
Industrial Technology and
Entrepreneurship Chosun University

Recently, various studies are being conducted to build and apply knowledge graphs from text data. In the security field, the need for knowledge graphs is increasing to respond to new security vulnerabilities. There is a lot of research going on to create a security knowledge graph. However, because security vulnerabilities continue to arise, updating the security knowledge graph is essential.

Therefore, in this paper, it will be suggested a method to 1) cluster of Vulnerability data based on text similarity and delete less relevant data, 2) extract Relation Triples from Vulnerability data using the REBEL model, 3) expand the knowledge graph using Google News data related to Entities. The expanded knowledge graph is expected to increase accuracy in tasks such as link prediction.

In the experiment of this paper, Four embedding models, "ComplEx", "TransE", "DistMult", and "HolE", were trained using the generated knowledge graph. Afterwards, it was evaluated using the “MMR”, “MR”, and “Hits@N”. As a result, the “HolE” model showed the highest performance. Also it was confirmed that the expansion of the knowledge graph increases the performance of the embedding model.

Ⅰ. 서론

A. 연구 배경 및 목적

지식 그래프는 현실의 데이터를 개체 간의 관계 형태로 저장하는, 그래프 구조의 지식베이스를 말한다. 지식 그래프는 각 개체 사이가 어떻게 연결되어 있는지 파악하기 위한 도움을 주기 때문에, 검색 엔진을 강화하거나 추천 시스템 등 다양한 시스템에 적용되며 활약하고 있다. 대표적으로 Google, Microsoft, Amazon과 같은 회사들이 독자적인 지식 그래프를 구축하여 검색 기능을 강화하고 있다. 또한, 지식 그래프는 자연어를 처리하는 작업에 유용하게 사용 가능하여, 딥러닝 분야에서 인공지능 모델을 학습시키기 위한 데이터로써 중요한 역할을 한다. 이러한 활용성 때문에 실시간으로 생겨나는 방대한 텍스트 데이터로부터의 지식 그래프를 구축하는 연구가 활발히 진행되고 있다.

보안 분야에서도 지식 그래프는 아주 중요한 역할을 한다. 과거로부터의 보안 취약점에 대한 데이터를 기반으로 보안 지식 그래프를 구축함으로써, 방대한 보안 취약점 데이터를 효과적으로 관리할 수 있어, 취약점에 대한 정보를 빠르게 찾을 수 있게 해주며, 보안 침해사고에 빠르게 대처할 수 있도록 도움을 준다. 또한, 보안 지식 그래프로부터 새로운 관계를 예측하는 것으로 앞으로 발생할 수 있는 보안 취약점을 예측할 수 있게 해준다. 이러한 보안 지식 그래프는 보안 전문가들이 효율적으로 보안 취약점을 관리하고 대응하는 데 큰 역할을 한다. 그리고 인공지능을 이용한 보안 프레임워크에 대한 연구가 활발해지며 보안 지식 그래프의 중요성은 더욱 높아졌다. 보안에 필요한 방대한 지식을 효율적으로 저장할 수 있는 보안 지식 그래프는 인공지능 모델을 학습시키는 중요한 자원이 된다. 이에 따라 보안 지식 그래프를 생성하고 적용하고자 하는 연구들이 진행되고 있으며, CVE 데이터를 기반으로 하여 보안 지식 그래프를 생성하고 적용하는 방법론들이 다양하게 제시되었다. 본 논문에서는 지식 그래프 자동 확장을 위한 링크 예측이나 검색 엔진 등에서의 정확도를 높이고자, 보안 취약점 데이터를 취약점에 대한 설명의 유사도를 기준으로 분류하여 관련도가 떨어지는 데이터를 삭제하고 지식 그래프를 생성하여 지식 그래프의 질을 높이고자 한다. 또한, 추출한 개체와 관련된 구글 뉴스 기사데

이터를 이용해, 지식 그래프를 확장하는 방법을 제안하고자 한다.

B. 연구 내용 및 구성

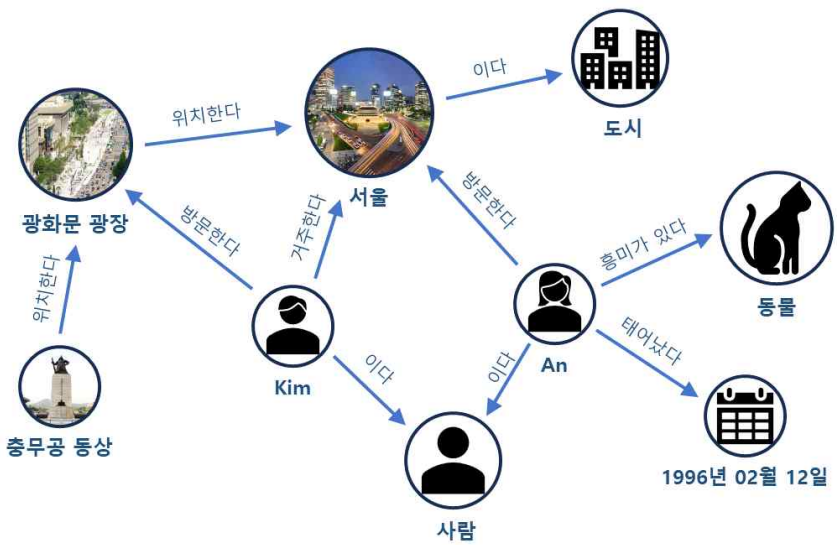
본 연구의 주요 내용은 보안 지식 그래프를 생성하기 위해, CVE Data Base에서 보안 취약점 데이터를 수집하고, 대응에 대한 정보를 Exploit Data Base에서 추가로 수집한다. 개체 간의 새로운 관계를 예측하는 링크 예측(Link Prediction)과 같은 임베딩 기반의 예측에서 정확도를 높일 수 있도록, 수집한 데이터를 보안 취약점에 대한 설명의 유사도를 기준으로 군집화한 뒤 데이터 수가 너무 적고 그래프에서 다른 그룹으로부터 멀리 떨어진 것으로 나타나는 데이터를 제거한 뒤, 지식 그래프를 생성한다. 그 후 지식 그래프 내의 개체와 관련된 구글 뉴스를 크롤링해서 얻는 데이터를 이용하여, 지식 그래프를 확장하는 지식 그래프 구축 및 확장 프레임워크를 제시하고자 한다. 이를 제안하기 위해 다음과 같은 구성으로 작성되었다.

서론에 이어 2장 관련 연구에서는 지식 그래프에 대해 설명하고, 지식 그래프를 생성하고, 지식 그래프를 확장하는 과정을 설명한다. 3장에서는 본 논문에서 제안하고자 하는 보안 취약점에 대한 설명이 유사한 데이터끼리 분류하고, 다른 데이터와 관련이 적은 데이터를 삭제하는 과정을 거치고, 지식 그래프를 확장 단계로 나누어, 보안 지식 그래프를 생성하는 전체 프레임워크를 설명한다. 4장에서는 생성한 지식 그래프의 성능을 순위 기반의 지표인 MMR, MR, Hits@N 지표를 이용해 평가하는 실험을 진행한 뒤, 마지막으로 5장에서는 결론과 향후 연구에 관해 서술하며 마무리한다.

Ⅱ. 관련 연구

A. 지식 그래프

지식 그래프는 현실의 정보를 구조적으로 표현하여, 지식을 보다 효과적으로 저장하며, 복잡한 정보를 쉽게 이해하게 도와준다. 이러한 지식 그래프는 개체(Entity)와 관계(Relation)로 이루어져 있다. 개체(Entity)는 지식 그래프의 핵심 단위로, 장소, 인물, 사건과 같이 실제 세계에 존재하는 사물 또는 개념을 말한다. 관계(Relation)은 두 개체 사이의 상호작용을 나타낸다. 예를 들어, “사람”이라는 개체와 “사과”라는 개체 사이에서는 “먹는다.”, “판매한다.”와 같은 관계가 존재할 수 있다. 지식 그래프는 이러한 개체와 관계를 그래프 구조로 표현한다. 개체는 그래프 상의 지점인 node로 표현되며, 두 개체 사이의 관계는 두 node 사이의 edge로써 표현된다. [그림 1]은 지식 그래프의 개체-관계를 node와 edge를 가지는 network 그래프로 표현한 예이다.



[그림 1] 지식 그래프 예

대규모 지식 그래프는 뉴스, 블로그, SNS, Wikipedia 등 웹사이트에서 수집할 수 있는 텍스트 데이터나 사전에 구축된 지식베이스를 활용하여 지식 그래프를 구축하고 있다. 데이터는 계속해서 생성되고, 변화하기 때문에 지식 그래프를 지속적으로 업데이트할 수 있도록 지식 그래프를 자동 생성하는 기능이 필수적이다. 지식 그래프를 자동으로 생성하기 위한 일반적인 단계는 다음과 같다. 먼저 뉴스, 기사, 논문 등 분석하고자 하는 도메인에 해당하는 텍스트 데이터를 수집해야 한다. 이렇게 수집된 텍스트 데이터를 불용어 제거, 특수문자 제거, 토큰화 등의 전처리한다. 그 후 개체명 인식(Named Entity Recognition) 기술을 이용하여, 원시 텍스트로부터 Subject, Object 두 개체를 추출하는 과정을 거친다. 개체명 인식은 주어진 텍스트로부터, 사람, 장소, 시간과 같은 특정 유형의 개체를 식별하고 추출하는 기술이다. 다음으로 형태소 분석을 통해, 품사의 배치 패턴을 찾는 등의 방법을 통해, 개체명 인식을 통해 추출한 두 개체 사이의 관계를 추출하는 과정을 거친다. 이를 통해 (Subject, Relation, Object) 형태의 관계 트리플(Relation Triple)을 생성한다. 관계 트리플에서. 개체(Entity)는 그래프의 node로, 관계(Relation)은 edge로 표현하여 그래프를 생성한다. 구축된 지식 그래프의 부족한 지식을 보완하기 위해, 추가적인 텍스트 데이터를 수집하거나, 두 개체 사이의 관계를 예측하는 링크 예측(Link Prediction) 작업을 통해, 지식 그래프를 확장하고 업데이트한다.

B. 딥러닝 기반 지식 그래프 생성

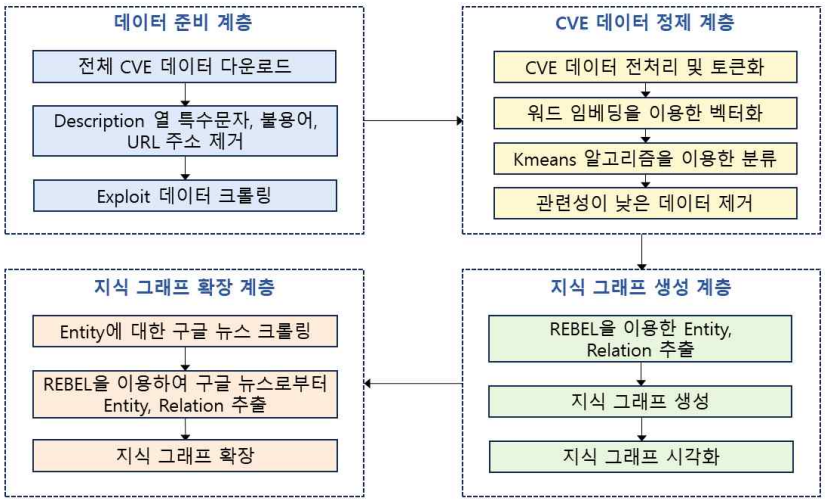
지식 그래프를 생성하기 위한 과정 중, 개체명 인식(Named Entity Recognition)과 두 개체 사이의 관계 추출(Relation Extaction) 두 과정은 대표적으로 사용되는 기술이다. 개체명 인식은 주어진 텍스트 내에서 사람, 장소와 같이 특정한 유형의 개체를 인식하는 기술이다. 그리고 관계 추출은 개체명 인식으로 인식한 두 개체 사이의 관계를 인식하는 과정이다. 개체명 인식 과정은 미리 구축해둔 개체명 사전을 이용하여 일치하는 문자열을 찾는 방법을 사용하거나, 자연어처리 라이브러리를 이용해 특정 패턴으로 이루어진 단어 조합을 찾아 인식하는 방법 등을 사용하였다. 마찬가지로 관계 추출은 문장구조를 통해 관계를 추출하거나, 단어의 패턴을 정의하여 비교하는 작업을 통해 추출하였다. 이러한 기존의 추출 방법들은 대규모의 사전을 미리 생성해야 하거나, 단어들의 패턴들을 정의해 놓아야 하는 불편함이 있었

다. 이로 인해, 많은 시간과 인력이 투입되어야 했으며, 텍스트 데이터를 전처리하는 과정에 따라, 추출되는 결과가 크게 달라졌다. 딥러닝 기술이 발전함에 따라, 개체명 인식과 관계 추출과정에서도, 딥러닝 모델이 적용될 수 있게 되었다. 사전학습 되어있거나, 입력 데이터에 따라 학습을 진행하는 딥러닝 모델이 적용되며, 사람이 해야 하는 일이 줄어들어, 지식 그래프를 생성하기 위한 과정이 단축되었다. 다양한 연구를 거치며, 개체와 관계를 추출하는 딥러닝 모델이 발전하여, 기존의 직접 자연어처리 기술을 이용하여 추출하는 방법보다 정확도가 높게 추출할 수 있게 되었다. REBEL(Relation Extraction By End-to-end Language generation) 모델의 경우, 사전 학습된 BART-large(Lewis et al. 2020) 모델을 사용하여, 입력된 텍스트 데이터로부터 개체를 추출하고, 두 개체 사이의 관계를 예측한다. 관계 예측 성능을 높이기 위해, BART 모델을 미세 조정하며 학습되었다. 이러한 REBEL 모델은 “<triplet>”, “<head>”, “<tail>”과 같은 스페셜 토큰을 이용하여 관계 트리플 예측 결과를 디코딩하고, 출력한다. REBEL 모델은 개체 인식과 관계 예측에 높은 성능을 보여, 본 논문에서는 REBEL 모델을 이용한 지식 그래프 생성 방법을 사용한다.

Ⅲ. 보안 취약점 데이터 기반 보안 지식 그래프 구축 및 확장

A. 보안 지식 그래프 생성 및 확장 프레임워크

본 절에서는 보안 취약점 데이터를 기반으로 지식 그래프를 구축하고, 확장하는 프레임워크를 설명한다. 보안 취약점 목록 데이터와 취약점 대응 데이터를 수집하고, 데이터를 전처리 과정을 수행하는 데이터 준비 계층(Data Preparation Layer), 수집된 CVE 데이터를 워드 임베딩 과정을 통해 수치화하고, K-Means 알고리즘을 이용하여 데이터를 군집화하는 CVE 데이터 정제 계층(CVE Refinement Layer), 두 개체 사이의 관계를 생성, 추출하는 REBEL 모델 기반으로 각 군집별 지식 그래프를 생성하는 지식 그래프 생성 계층(Knowledge Graph Construction Layer), 추출한 개체와 관련된 구글 뉴스를 수집하고 관계를 추출하는 과정을 통해, 해당 개체에 대한 지식 그래프를 확장하는 지식 그래프 확장 계층(Knowledge Graph Expansion Layer)로 구성하였다. [그림2]은 보안 지식 그래프를 구축하고 확장하는 방법에 대한 전체 프레임워크를 도식화한 것이다.



[그림 2] 보안 지식 그래프 구축 및 확장 프레임워크

B. 보안 취약점 데이터 수집 및 전처리

1. CVE 데이터 수집 및 전처리

보안 지식 그래프를 생성하기 위해, CVE 데이터를 수집하였다. CVE는 “정보 보안 취약점 표준 코드(Common Vulnerabilities and Exposures)”의 약자로, 공개적으로 알려진 보안 취약점 목록을 의미한다. CVE는 하드웨어, 소프트웨어에서 악용될 수 있는 보안 취약점 및 결함 정보를 말한다. 각 CVE에는 고유한 시리얼 ID가 부여되고, 설명, 참조 등의 정보가 포함되며, CVE의 정보들은 CVE Data Base (https://www.cve.org/) 사이트에서 확인할 수 있다. 보안 취약점에 대한 정보를 기반으로 지식 그래프를 생성하기 위해, CVE Data Base에서 1999년도부터 2022년도 사이에 등록된 257,660건의 CVE 데이터를 CSV 파일 형식으로 다운로드하여, 보안 취약점 데이터를 확보하였다. 확보한 데이터의 형식은 [표 1]과 같다.

Name	CVE-2022-0004
Status	Candidate
Description	Hardware debug modes and processor INIT setting that allow override of locks for some Intel(R) Processors in Intel(R) Boot Guard and Intel(R) TXT may allow an unauthenticated user to potentially enable escalation of privilege via physical access.
References	MISC:https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00613.html | URL:https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00613.html
Phase	Assigned (20211015)
Votes	None (candidate not yet proposed)
Comments	-

[표 1] CVE Data Base에서 제공한 CVE-2022-0004 정보

“Name”열은 CVE에 부여된 시리얼 ID를 의미한다. “Status“열은 CVE DataBase에 등록된 상태를 의미한다. entry 상태는 CVE 목록에 등록되는 것이 승인된 상태임을 의미하고, Candidate 상태는 목록에 포함될 수 있도록 검토가 진행되고 있음을 의미한다. 그리고 ”Description“열의 경우 CVE에 대한 간단한 설명이며,

“References”열은 관련 취약점에 대한 문서나 참조문서들이다. “Phase“열은 CVE ID를 할당받은 날짜이며, ”Votes“는 CVE 위원회에서 검토한 뒤 투표한 결과를 기록한 열이다. ”Comments“열은 CVE에 대해 남긴 댓글이다. 이렇게 수집된 데이터를 전처리하는 과정을 거쳤다. 다운로드를 통해 수집한 257,660건의 데이터 중에서 CVE-ID가 예약만 되어있는 상태이거나, 거부된 CVE-ID가 존재했다. 이 경우는 똑같은 문장으로만 이루어져 있어, 지식 그래프를 생성하는데 필요한 데이터가 아니라고 판단하여, 삭제하는 과정을 거쳤다. 삭제 과정을 거친 후, 총 188,616건의 데이터를 확보하였다. 그 후, 최신 하드웨어, 소프트웨어에 대한 데이터만을 확보하기 위하여, 전체 CVE 데이터 중에서 2020년 이후의 CVE 데이터 46,567건을 추출하여, 실험을 진행하였다. 관련 취약점에 대한 문서나 참조들이 들어있는 “References”은 URL로 구성되어 있어, 지식 그래프를 생성하는 데이터 세트로 부적절하다고 판단하여 삭제하였다. 또한, 단어로만 이루어져 트리플을 추출할 수 없는 “Status”, “Phase”, “Votes” 열 또한 삭제를 진행하였으며, “Comments”열은 비어있는 경우가 많아 삭제한 뒤, pandas 라이브러리를 이용하여, “Name”과 “Description” 두 개의 열을 가지는 데이터 프레임으로 저장하였다. 위의 전처리를 통해, [그림 3]과 같은 형태의 데이터를 확보하였다.

	CVE-ID	Summary
141863	CVE-2020-0001	In getProcessRecordLocked of ActivityManagerSe...
141864	CVE-2020-0002	In ih264d_init_decoder of ih264d_api.c, there ...
141865	CVE-2020-0003	In onCreate of InstallStart.java, there is a p...
141866	CVE-2020-0004	In generateCrop of WallpaperManagerService.jav...
141867	CVE-2020-0005	In btm_read_remote_ext_features_complete of bt...
141868	CVE-2020-0006	In rw_i93_send_cmd_write_single_block of rw_i9...
141869	CVE-2020-0007	In flattenString8 of Sensor.cpp, there is a po...
141870	CVE-2020-0008	In LowEnergyClient::MtuChangedCallback of low_...
141871	CVE-2020-0009	In calc_vm_may_flags of ashmem.c, there is a p...
141872	CVE-2020-0010	In fpc_ta_get_build_info of fpc_ta_kpi.c, ther...

[그림 3] 전처리 후 데이터 프레임 예

2. 크롤링을 이용한 Exploit 데이터 수집

더욱 상세한 지식 그래프를 생성하기 위해, 추가로 보안 침해사고 대응에 관한 정보를, Exploit Data Base 페이지(https://www.exploit-db.com)에서 추가로 수집하였다. Exploit Data Base에서 CVE-ID를 검색하여, 해당 보안 취약점의 분석 코드를 확인할 수 있고, 대응에 대한 간단한 설명 등의 정보를 획득할 수 있다. 각 CVE 데이터에 대하여, 정보를 획득하기 위해, selenium 라이브러리를 이용한 크롤링을 사용하였다. Exploit Data Base에서 CVE-ID를 기준으로 검색했을 때 URL은 “https://www.exploit-db.com/search?cve=1999-0002” 형태이다. 이 URL에서 쿼리 스트링 value에 미리 수집해 둔 CVE-ID에서 “연도-번호” 부분을 추출하여 붙이는 방법으로 요청 URL을 생성하였고, selenium 라이브러리를 이용해 데이터를 수집하였다. 결과 고유 식별번호인 Exploit ID와 Exploit Data Base에 저장된 제목 두 가지의 정보를 수집하였다. 결과 [표 2]와 같은 데이터를 얻을 수 있었다.

Name	Exploit ID	Exploit Title
CVE-2000-0002	19688	ZBServer Pro 1.5 - Remote Buffer Overflow (1) - Windows remote Exploit
CVE-2000-0009	19704	Nortel Networks Optivity NETarchitect 2.0 - PATH - Multiple local Exploit
CVE-2000-0010	19691	Tony Greenwood WebWho+ 1.1 - Remote Command Execution - Multiple remote Exploit
CVE-2000-0011	19703	AnalogX SimpleServer:WWW 1.0.1 - GET Buffer Overflow - Windows dos Exploit
CVE-2000-0012	19696	Hughes Technologies Mini SQL (mSQL) 2.0.11 - 'w3-msql' Remote Buffer Overflow - Solaris remote Exploit

[표 2] Exploit ID 크롤링 결과 예

“https://www.exploit-db.com/exploits/{Exploit_ID}” URL을 통해, 작성자나, 간단한 설명, code 등 Exploit 상세 정보를 확인할 수 있다. 해당 URL의 마지막 부분에 Path Variable의 형태로 Exploit ID가 포함되어 있다는 점을 이용하여, Exploit 상세 정보를 크롤링할 수 있었다. HTML 요소로부터 “Type” 정보를 추출하였고 “<code>” 태그 안에 간단한 설명과 대응 코드가 포함되어 있어, “<code>” 태그

내의 텍스트를 가져온 후, 코드를 제외하고 설명 부분만을 추출하고, 개행문자 등을 제거하여 Expolit에 대한 설명 데이터를 얻을 수 있었다. 결과 [표 3]과 같은 데이터를 얻을 수 있었다. 위에서 만들어 둔, CVE, Exploit 데이터 프레임을 Exploit ID를 기준으로 병합하여 데이터 프레임을 생성하며, 데이터 수집 과정을 마무리하였다.

Exploit ID	Type	Exploit Description
19688	remote	ZBServer Pro 1.5 - Remote source: https://www.securityfocus.com/bid/889/infoZBSoft ZBServer Pro is an Internet and Intranet server that supports HTTP, Gopher, FTP and Chat Services. ZBServer is available for Microsoft Windows operating systems.ZBServer Pro 1.5 has an unchecked buffer in the code that handles GET requests. This weakness allows for the execution of arbitrary code. USSR exploit:Binary exploit - 19688.exeSource code - 19688.ziphttps://gitlab.com/exploit-database/exploitdb-bin-sploits/-/raw/main/bin-sploits/19688.exehttps://gitlab.com/exploit-database/exploitdb-bin-sploits/-/raw/main/bin-sploits/19688.zipOverflow (1) - Windows remote Exploit

[표 3] Exploit Type, Description 수집 결과 예

C. 텍스트 유사도 기반 CVE 데이터 분류

현실의 정보는 계속해서 변화하고 생성되기 때문에, 지식 그래프를 지속적으로 업데이트하는 작업이 필수적으로 요구된다. 링크 예측(Link Prediction)은 두 엔티티 사이의 연결을 예측하는 것으로, 이 링크 예측을 통해 지식 그래프로부터 새로운 관계를 예측할 수 있고, 이를 통해 지식 그래프를 확장할 수 있다. 이러한 링크 예측을 수행하는 과정에서, 정확도를 높일 수 있도록 양질의 지식 그래프를 생성하고자, 본 연구에서는 CVE 데이터를 텍스트 유사도 기반으로 분류하여 관련성이 낮다고 판단되는 CVE 데이터를 제외하고, 관련성이 높은 CVE 데이터들만을 이용하여 지식 그래프를 생성하고자 한다.

1. 텍스트 유사도 기반 분류를 위한 전처리

a. CVE Description 열 토큰화

취약점에 대한 설명인 Description열의 유사도를 이용하여, CVE 데이터를 분류하고자 한다. 유사도를 비교하기 위해, 텍스트 데이터를 컴퓨터가 이해할 수 있는 수치 데이터로 변환하는 워드 임베딩 과정을 거쳐야 한다. 워드 임베딩을 위해, CVE 데이터의 Description열을 전처리하고 토큰화 하는 과정을 거쳤다. 먼저 nltk 라이브러리에서 제공하는 불용어 사전을 이용하여, 각 Description 열에 포함된 불용어와 특수문자를 제거하는 과정을 거쳤다. 그리고 텍스트 내에 포함된 URL을 제거하는 과정을 거쳐 문장만을 추출하였다. 이렇게 추출한 문장을 토큰화하는 과정을 거쳤다. 토큰화 결과는 [표 4]와 같이 얻을 수 있었다.

CVE ID	CVE Description	CVE Description Tokens
CVE-2000-0002	Buffer overflow in ZBServer Pro 1.50 allows remote attackers to execute commands via a long GET request.	buffer overflow zbserver pro allows remote attacker execute command via long get request
CVE-2000-0009	The bna_pass program in Optivity NETarchitect uses the PATH environmental variable for finding the "rm" program, which allows local users to execute arbitrary commands.	bna pas program optivity netarchitect us path environmental variable finding rm program allows local user execute arbitrary command

[표 4] CVE Description 토큰화 결과 예

b. 텍스트 유사도 기반 CVE 데이터 분류를 위한 워드 임베딩

CVE 데이터를 유사한 설명을 가지는 데이터끼리 그룹으로 묶어 분류하기 위해, CVE 데이터의 Description 열을 임베딩하는 과정을 거쳤다.

첫 번째로 TF-IDF(Term Frequency-Inverse Document Frequency: 단어 빈도-역 문서 빈도)를 이용하여 임베딩하였다. TF-IDF는 단어마다 중요한 정도를 가중치로 부여하여 임베딩하는 방법이다. 특정 문서에서만 발견되는 단어는 가중치를 높게 부여하고, 여러 문서에서 공통적으로 발견되는 단어의 경우는 가중치를 낮게 부여하는 방법으로, 문서의 유사도를 구하는 작업에서 주로 사용된다. 다음 식에서

N은 총 문서의 수를 의미한다. tf(d,t)는 문서 d에서, 단어 t의 등장 횟수를 의미하고, df(t)는 특정 단어 t가 등장한 문서들의 수를 의미한다. idf(t)는 df(t)에 반비례하는 수를 의미한다.

$$idf(t) = \log(\frac{N}{1+df(t)}) \quad \text{(수식 1)}$$

$$tf(d,t) = \frac{1}{idf(t)} \quad \text{(수식 2)}$$

CVE의 Description 데이터로 sklearn 라이브러리의 TfidfVectorizer모델을 학습시켜, 모든 CVE의 Description에 대한 벡터를 구할 수 있었다. 이렇게 생성한 벡터의 코사인 유사도를 계산하여, 문서 사이의 유사도를 측정할 수 있다.

단어의 빈도수로 가중치를 부여하고, 문서를 임베딩하는 TF-IDF 방식은 단어의 빈도만을 이용하기 때문에, TF-IDF를 이용하여 문서의 유사도를 파악할 때, 문서 내의 문맥, 의미의 유사도를 반영하지 못한다. CVE의 Description 내의 문맥의 특징을 반영한 분류 실험도 진행하기 위해, 임베딩 방법에 따른 차이를 확인하기 위해, 두 번째로 Word2Vec을 이용하여 임베딩을 진행하였다. Word2Vec은 비슷한 위치에서 등장하는 단어들은 비슷한 의미를 가질 것이라는 분포가설을 따르는 임베딩 방법이다. CVE Description열을 전처리하여 생성된 토큰들을 데이터 세트로 이용하여 Word2Vec 모델을 학습시켜, 모든 토큰에 대한 100차원 벡터를 생성하였다. Word2Vec을 이용하여, 각 토큰의 문맥을 보존하며 워드 임베딩을 수행했기 때문에, CVE 데이터의 특징을 나타내는 기준벡터는 Description에 포함된 토큰 벡터의 평균으로 나타낼 수 있을 것이다. 예를 들어, “CVE-2000-0002”의 경우 Description 열 내에, “['buffer', 'overflow', 'zbserver', 'pro', 'allows', 'remote', 'attacker', 'execute', 'command', 'via', 'long', 'get', 'request']” 이렇게 총 13개의 토큰을 가진다. 따라서 “CVE-2000-0002”의 특징을 나타내는 벡터는 13개의 토큰이 가지는 벡터의 평균값으로 나타날 것이다. numpy 라이브러리를 이용하여, 토큰 벡터의 평균을 계산하였다. 이 과정을 모든 CVE 데이터 적용하여, CVE 데이터가 가지는 기준벡터들을 구했다.

2. KMeans 알고리즘을 이용한 CVE 데이터 분류

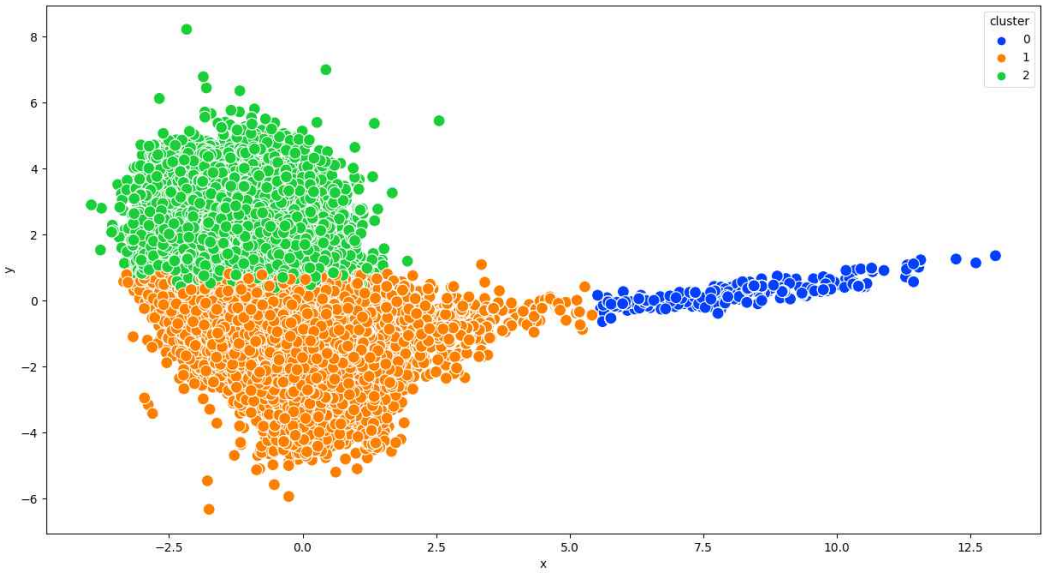
위에서 각 CVE 데이터를 분류할 수 있도록 워드 임베딩을 두가지 방법으로 수행하였고, 이를 통해 임베딩 벡터를 생성하였다. 이 임베딩 벡터를 이용하여, 비슷한 특성을 가지는 CVE 데이터끼리 군집화하는 과정을 거쳤다. 텍스트의 유사도를 기반으로 분류하기 위해, 정답이 제공되지 않은 데이터 세트의 특징을 찾아내는, 비지도 학습 방법인 KMeans 알고리즘을 이용하여 군집화하였다. KMeans 알고리즘은 데이터를 k개의 클러스터로 분류하는 알고리즘으로, 클러스터의 중심점과 데이터 사이의 거리의 합이 최소가 되도록 중심점을 옮겨가며, 군집을 형성하는 알고리즘이다. 이러한 KMeans 알고리즘이 벡터 사이의 거리가 가까울수록 유사한 특성을 가지는 CVE 데이터의 임베딩 벡터를 군집화하기에 효과적일 것이다. 군집화하는 과정에서, 최적의 클러스터 개수를 찾아내기 위해, Elbow Method를 적용하였다. Elbow Method는 각 클러스터의 중심과 데이터 사이의 거리를 합한 inertia를 기준으로 최적의 클러스터 개수를 찾아내는 알고리즘이다. CVE 기준벡터로 학습한 Kmeans 모델의 클러스터 수를 1부터 10까지 늘려가며, inertia 값을 구해보았다. Elbow Method를 사용해 확인한 결과, 최적의 클러스터 수가 3개임을 확인할 수 있다. Elbow Method를 이용해 찾은 최적의 클러스터 수 3을 하이퍼 파라미터로 설정하고, 위에서 TF-IDF 방법으로 임베딩한 벡터와, Word2Vec을 이용해 임베딩한 벡터로 각각 KMeans 모델을 학습시켰다. [표 5]는 두 경우에 대해, KMeans 모델이 그룹화를 진행한 결과이다. cluster0의 경우, TF-IDF 와 Word2Vec을 사용한 두 경우 모두 비슷한 결과를 보였으며, 다른 두 cluster에 비해 데이터 수가 매우 적게 나타나는 것을 확인할 수 있었다.

임베딩 방법	cluster0 데이터 수	cluster1 데이터 수	cluster2 데이터 수	총 데이터 수
TF-IDF	1,482	43,016	2,069	46,567
Word2Vec	1,618	36,002	8,947	46,567

[표 5] CVE 데이터 군집화 결과

그 후 차원 축소 알고리즘인 PCA(Principal Component Analysis) 알고리즘을 이용하여 3차원으로 차원 축소하였다. [그림 4]는 Word2Vec으로 임베딩한 뒤, 군집

화한 결과를 2차원으로 시각화한 결과이다. 해당 그래프와 같이, CVE 데이터의 임베딩 벡터를 기반으로, 3개의 클러스터로 분류할 수 있었다. 시각화 결과를 보았을 때, cluster0에 속하는 데이터가 다른 데이터와 멀리 떨어져 있음을 확인했다. 이를 통해, cluster0에 속하는 데이터는 다른 CVE 데이터와 Description의 유사도가 낮아, 지식 그래프에 질을 낮출 우려가 있음을 확인하였다.



[그림 4] Kmeans 알고리즘을 이용한 분류 시각화

분류가 잘 되었는지 확인하기 위해, 코사인 유사도를 이용한 유사도 비교 방법을 사용하였다. 두 벡터의 유사한 정도는 두 벡터 사이의 코사인 각도를 계산하여, 방향을 비교하는 코사인 유사도를 이용하여 계산할 수 있으며, 코사인 유사도가 1에 가까울수록 유사한 벡터이다. 두 벡터 A, B에 대하여 코사인 유사도는 다음과 같이 계산할 수 있다.

$$Cosine\ \ Similarity = \frac{A \cdot B}{\|A\|\ \|B\|} \quad \text{(수식 3)}$$

코사인 유사도를 이용하여, 클러스터 내부의 문서 사이의 유사도를 계산하여 그룹화가 잘 진행됐는지 확인하였다. TF-IDF를 이용한 경우 cluster0 내부 문서 사이의 코사인 유사도 평균값이 0.7824으로 유사한 문서끼리 잘 분류됐음을 확인할

수 있었고, 마찬가지로 Word2Vec을 이용한 경우의 cluster0 내부 문서의 코사인 유사도 평균을 계산했을 때, 0.6951로 CVE 데이터의 Description열의 유사도가 높은 문서끼리 잘 분류되었음을 확인할 수 있었다. 확인 뒤 지식 그래프의 성능을 향상시키기 위해, cluster0으로 분류된 데이터를 제외하는 과정을 거쳤다. 이 과정을 통해 cluster0에 속해있던 1,618개의 데이터를 삭제하고, 나머지 44,949개의 데이터를 이용하여 지식 그래프를 구축하였다.

D. 딥러닝 기반 보안 지식 그래프 구축

효율적인 보안 침해사고 대응을 위해, 사이버 보안 분야에서도 딥러닝 기술을 적용하여 보안 침해사고에 대응하는 연구들이 활발하게 진행되고 있다. 보안 지식 그래프는 딥러닝 모델을 학습시키고 발전시키는 중요한 데이터 소스로 사용된다. 이러한, 보안 지식 그래프를 구축하고 확장하는 것으로, 계속해서 생겨나는 보안 관련 지식을 효과적으로 담을 수 있고, 지식 검색 등 여러 방면에서 활용할 수 있다. 보안 지식 그래프를 구축하기 위해, 보안 데이터를 수집하고, 데이터로부터 지식 그래프를 생성하기 위한 트리플을 추출해야 한다. 트리플을 추출하기 위한, 방법으로 자연어처리 라이브러리를 이용하여 형태소 분석을 진행하고, 사전에 생성한 개체명 사전을 이용하거나, 품사의 패턴을 정의하고 비교하여 추출하는 방식이 사용되었으나, 딥러닝을 이용한 트리플 추출 방법이 발전하며, 처리 과정이 간단해지고, 효과적으로 추출할 수 있게 되었다. 본 논문에서는 딥러닝 기반의 트리플 추출 모델 중 REBEL 모델을 소개하고, 사용하여 지식 그래프를 구축하고자 한다.

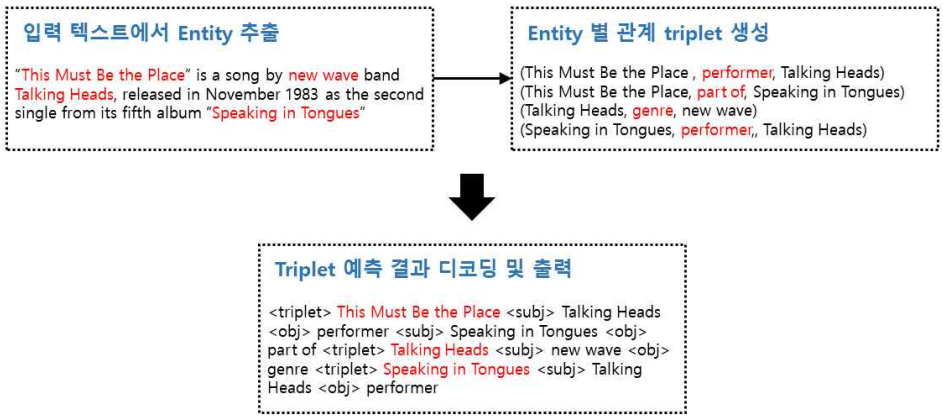
1. REBEL 모델

REBEL(Relation Extraction By End-to-end Language generation) 모델은 입력받은 원시 텍스트로부터 관계를 예측하여 트리플을 추출하는 과정에서 오류를 전파하거나, 소수의 관계 유형으로 제한되는 문제를 해결하고자. 자기 회귀 seq2seq 모델로 설계되었다. 그리고 BART-large(Lewis et al. 2020)를 기반으로 생성되었다. REBEL은 텍스트 데이터를 입력받았을 때, entity를 추출하고, 두 entity 사이의 관계를 예측하여 트리플을 생성하고 출력한다. x가 주어진 텍스트, y가 입력 텍스

트에서 트리플을 추출한 결과라고 하면, REBEL은 x가 주어지면 y를 자동회귀적으로 생성한다. 텍스트 요약, 번역 모델과 같이 Cross-Entropy loss를 사용하며, 사전 학습된 BART 모델을 미세 조정한다. 이러한 조정을 통해, 입력 텍스트로부터 관계를 예측할 확률을 높였다.

$$pBART(y \mid x) = \prod_{i=1}^{\leq n(y)} pBART(y_i \mid y_{<i}, x) \quad \text{(수식 4)}$$

REBEL 모델은 결과를 효율적으로 디코딩할 수 있도록 스페셜 토큰을 사용한다. <triple>은 head 개체가 포함된 트리플의 시작을 표시하며, 바로 뒤에 input 텍스트로부터 추출한 head 개체가 출력된다. <subj> 토큰은 head 개체의 끝과 tail 개체의 시작을 표시한다. 그리고 <obj> 토큰은 head와 tail 개체 사이 관계의 시작을 표시한다. 입력된 텍스트로부터 개체를 추출, 정렬하고 <triple>, <subj>, <obj> 순서대로 트리플 추출 결과를 출력한다. [그림 5]은 REBEL 모델이 입력된 텍스트로부터 트리플을 추출하고 결과를 출력하는 예이다. 입력 텍스트로부터 entity를 추출하여, “This Must Be the Place”, “new wave”, “Talking Heads”, “Speaking in Tongues” 네 개의 entity를 추출하였다. 추출한 entity를 head, tail로 정의하고 entity 사이의 관계를 예측하고 생성한다. [그림 5]에서 “This Must Be the Place”가 head, “Talking Heads”가 tail로 나열된 경우에서는 두 entity 사이의 관계를 “performer”라고 예측하였고, 이 결과를 (head, relation, tail) 형식으로 나열한 뒤, 스페셜 토큰을 이용해 트리플 추출 결과를 디코딩하여 출력한다.



[그림 5] REBEL 모델을 이용한 triplet 추출 예

2. REBEL 모델을 이용한 보안 지식 그래프 생성

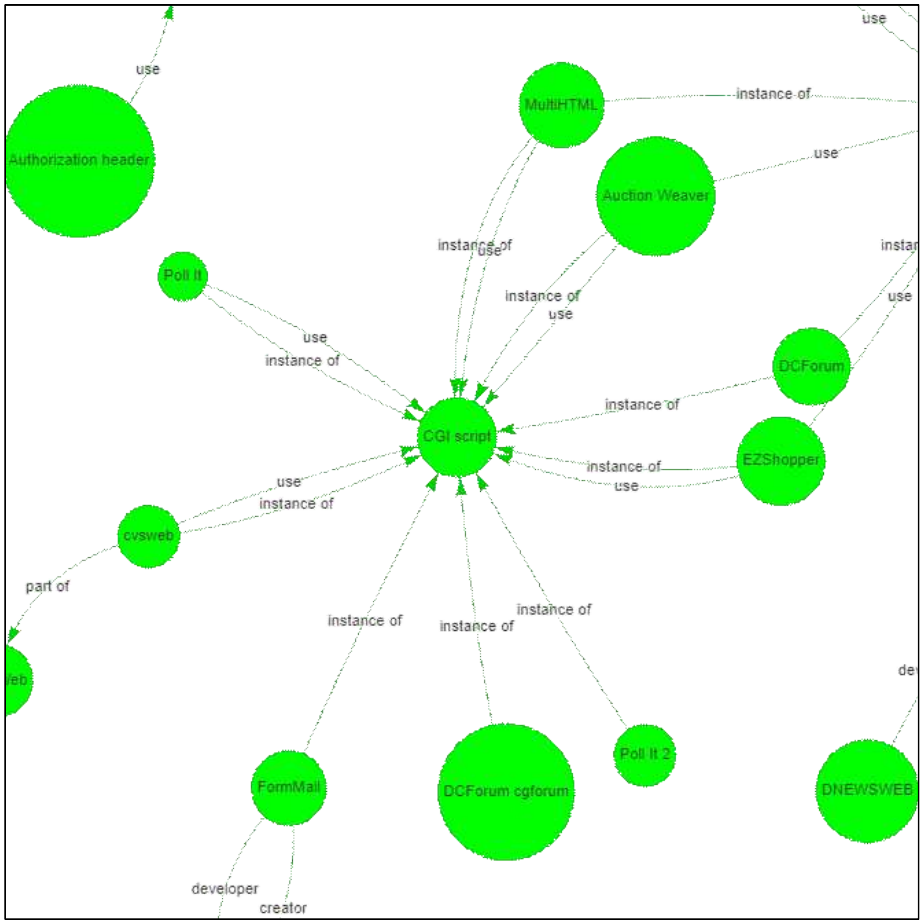
이러한 REBEL 모델을 이용하여 triplet을 추출하기 위해서, transformers 라이브러리의 AutoTokenizer와 AutoModelForSeq2SeqLM의 from_pretrined 함수를 통해 사전학습 된, “Babelscape/rebel-large” 모델을 불러왔다. 그 후, 입력받은 원시 텍스트를 사전 학습된 AutoTokenizer를 이용하여, 토큰화하였다. AutoTokenizer로 생성한 토큰의 id값들을 사전 학습된 AutoModelForSeq2SeqLM 모델에 입력하여, entity 사이의 관계를 예측하여 트리플을 생성한다. 예측 결과를 “<triplet>”, “<subj>”, “<obj>”와 같은 스페셜 토큰이 포함된 문자열로 디코딩하여 반환한다. 반환된 문자열에서 각 스페셜 토큰 사이의 값을 추출하는 과정을 통해, 최종적으로 (head, type, tail) 형식의 triplet으로 변환한다. REBEL 모델을 이용하여, CVE 데이터로부터 지식 그래프 생성을 위한 triplet을 추출하였다. CVE Description을 사전 학습 된 AutoTokenizer를 이용하여 토큰화한 뒤, REBEL 모델을 불러와서 생성한 AutoModelForSeq2SeqLM를 이용해 triplet을 추출하였다. [표 6]는 REBEL 모델을 이용하여, CVE-2020-0001의 Description에서 트리플을 추출한 결과이다.

CVE-2020-0001부터 CVE-2020-0010까지의 triplet 추출 결과 예
Entities: Android Android-8 0 Android kernel local escalation of privilege exploitation time-of-check time-of-check time-of-use vulnerability calc_vm_may_flags ashmem Relations: {'head': 'Android', 'type': 'has part', 'tail': 'Android-8 0'} {'head': 'Android-8 1', 'type': 'subclass of', 'tail': 'Android'} {'head': 'Android kernel', 'type': 'part of', 'tail': 'Android'} {'head': 'Android', 'type': 'has part', 'tail': 'Android kernel'} {'head': 'local escalation of privilege', 'type': 'subclass of', 'tail': 'exploitation'} {'head': 'time-of-check', 'type': 'subclass of', 'tail': 'exploitation} {'head': 'time-of-check time-of-use vulnerability', 'type': 'platform', 'tail': 'Android'} {'head': 'calc_vm_may_flags', 'type': 'part of', 'tail': 'ashmem', 'meta': {'spans': [[0, 128]]}}

[표 6] CVE 데이터로부터 triplet 추출한 결과 예

3. 보안 지식 그래프 시각화

위에서 생성한 지식 그래프의 관계와 형태를 확인하기 위해, 지식 그래프를 시각화하였다. 상세한 network 그래프로 시각화하기 위해, vis.js 기반의 pyvis 라이브러리의 network를 사용하였다. 추출한 triplet 중에서 “head”와 “tail”을 node로써 그래프에 표시하였고, 각 node를 잇는 edge는 triplet의 “type”으로 설정하였다. 이후 그래프를 html 파일로 저장하였다. [그림 6]은 지식 그래프를 시각화한 결과이다. 시각화 결과를 통해 보안 지식 그래프 내의 Entity들 사이의 관계를 확인할 수 있었다.



[그림 6] 지식 그래프 시각화

E. 구글 뉴스를 이용한 보안 지식 그래프 확장

생성한 지식 그래프를 더 상세하게 만들기 위해, 지식 그래프 확장 과정을 거쳤다. 지식 그래프를 확장하기 위한 추가 지식을 수집하기 위해, 구글 뉴스 데이터를 사용하였다. 구글 뉴스에서 보안 취약점과 위에서 추출한 개체 관련 뉴스 기사를 크롤링하여 데이터를 확보하고, 지식 그래프를 구축하기 위한 triplet을 추출하였다.

1. 구글 뉴스 데이터 크롤링

Google News에서 기사 본문을 크롤링하여, 정보를 추가로 수집하였다. GoogleNews 라이브러리의 키워드를 입력하면 뉴스를 검색하고, 검색된 기사의 URL을 반환하는 기능을 이용하여, 크롤링하기 위한 URL을 수집할 수 있다. 이러한 GoogleNews 라이브러리를 이용하여, 지식 그래프의 개체를 검색하여, 개체와 관련 있는 기사들의 URL을 수집하였다. 그 후 URL을 이용하여, 기사를 크롤링하고자 하였으나, selenium이나 beatifulsoup 라이브러리만을 사용하기에는 각 기사를 제공하는 플랫폼이 달라, 태그나 수집 방법이 모두 달라 어려움이 발생하였다. 이러한 문제를 해결하기 위해, 크롤링하고자 하는 URL을 입력하면, 해당 기사로부터, 제목, 본문, 날짜 등 기사 내용을 자동으로 크롤링해주는 newspaper3k 라이브러리를 사용하였다. 수집한 URL을 newspaper3k 라이브러리를 이용해 크롤링하였다. [표 7]은 “ARP Spoofing”을 Google News에서 검색하고 크롤링한 결과이다. [표 7]과 같이 기사의 제목과 본문을 가져올 수 있었고, 본문 데이터를 문장 단위로 토큰화하여, triplet을 추출할 준비를 하였다.

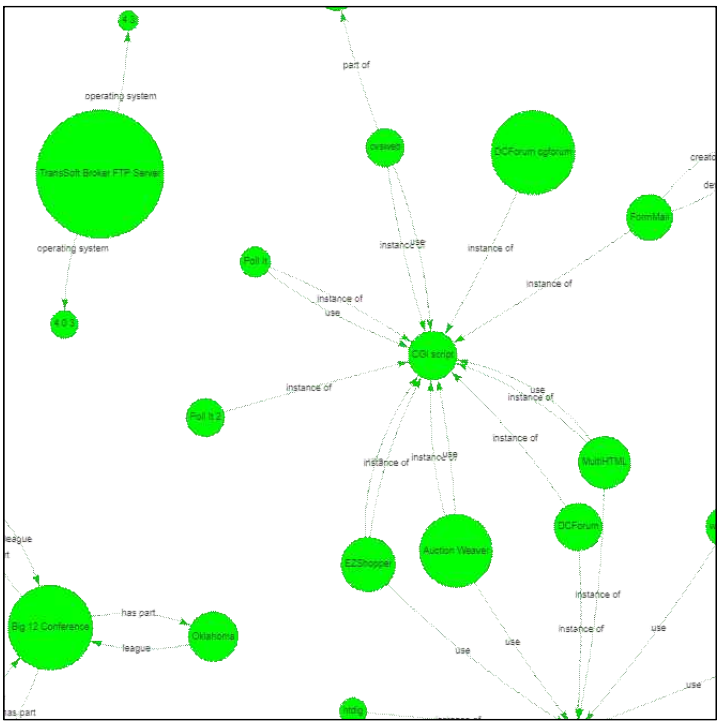
[표 7] “ARP Spoofing”으로 검색한 뉴스 크롤링 결과 예

“ARP Spoofing”으로 검색한 뉴스 크롤링 결과 예
Title: Address Resolution Protocol Working, Types, and Benefits Text: Address resolution protocol (ARP) is defined as a mechanism that helps discover the link-layer

address associated with a pre-specified internet layer address to enable the mapping of different devices on a local area network.\n\nDeveloped in the 1980s, ARP remains a crucial feature of local networks today. Advancements such as secure ARP (S-ARP) help combine network security with the convenience and efficiency of the address resolution protocol.

2. News 데이터로부터의 triplet 추출 및 지식 그래프 확장

지식 그래프를 확장하기 위해, 각 entity를 검색하여 기사 URL을 수집하였고, 기사 내용을 텍스트 유사도 분석을 이용하여, 보안과 관련된 기사를 추출하는 과정을 거쳤다. 그 이후 지식 그래프에 지식을 추가하기 위해, 본문에서 triplet을 추출하는 과정을 거쳤다. 효율적인 지식 추출을 위하여, REBEL 라이브러리를 이용하여 triplet을 추출하였다. 새롭게 추출한 triplet을 기존 triplet 데이터에 병합하여, 지식 그래프를 확장시켰다. [그림 7]은 지식 그래프를 확장시킨 결과를 시각화한 것이다. 시각화 결과를 통해, 이전 장에서 생성한 지식 그래프에서 Entity와 관계가 늘어난 것을 확인할 수 있다.



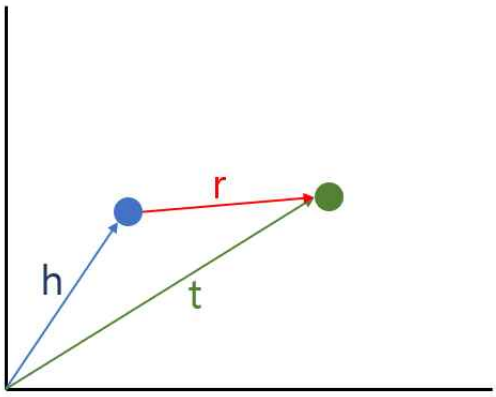
[그림 7] 지식 그래프 확장 결과 시각화

F. 보안 지식 그래프 임베딩

생성한 보안 지식 그래프를 평가하고, 여러 서비스에서 활용하기 위해, 지식 그래프를 컴퓨터가 이해할 수 있도록 수치 데이터로 변환하는 임베딩 과정이 필요하다. 지식 그래프를 임베딩 하는 경우, 지식 그래프 내의 Entity, Relation 연결과 특성을 포함하여 임베딩하는 과정이 필요하다. 이러한 지식 그래프 임베딩을 통해, 지식 그래프를 임베딩하여 생성된 벡터는 딥러닝 모델 학습에 사용할 수 있고, 추천 시스템이나 자동완성 등 다양한 서비스에 활용할 수 있다. 지식 그래프를 임베딩하기 위해 제안된 임베딩 방법 중, 일반적으로 많이 쓰이는 TransE, DistMult, ComplEx, HolE 네 가지 방법을 소개한다.

1. TransE 임베딩

TransE 임베딩 방법은 지식 그래프를 임베딩하는 방법의 하나로, 지식 그래프의 head, tail, relation을 임베딩하였을 때 head + relation이 tail의 벡터가 되게끔 학습하는 방법이다. head의 벡터를 h, relation은 r, tail을 t라고 부르기로 했을 때, [그림 8]은 TransE 임베딩을 그림으로 표현한 예이다.



[그림 8] TransE 임베딩 예

이러한 TransE 임베딩 방법은 relation의 벡터가 0이 아닌 경우, head + relation의 결과는 tail이 되지만, tail + relation의 결과가 head와 같지 않게 나타나는 비대칭 관계를 모델링 할 수 있다. 하지만 TransE 모델링에서는 대칭 관계의 데이터를

모델링 할 수 없고, 하나의 head에서 여러 tail과 관계를 갖는 1-to-N 관계를 임베딩할 수 없다는 단점을 가진다.

2. DistMult, ComplEx 임베딩

DistMult 방법은 단순히 벡터 간의 거리를 이용하여 임베딩 하였던 TransE 방법과 달리, head, relation, tail 세 벡터의 내적과 합을 이용하여 학습하는 방법이다. head와 relation의 내적과 tail의 벡터의 코사인 유사도가 높아지게끔 학습한다. 이러한 DistMult 임베딩의 경우 (수식 5)와 같은 score function을 갖는다. 이 score function에서 head, relation, tail의 순서가 바뀌어도 결과가 동일하게 나오기 때문에, DistMult 방법을 이용한 경우 비대칭 관계를 표현할 수 있으며, 1-to-N 관계도 모델링 할 수 있다. 하지만 이러한 특징으로 인해 DistMult는 대칭 관계를 모델링 할 수 없다.

$$f_r(h,t) = \sum_i h_i \cdot r_i \cdot t_i \quad \text{(수식 5)}$$

ComplEx 임베딩 방법은 DisMult에 기반하는 임베딩 방법이다. DisMult 임베딩 방법을 복소수 차원으로 확장하여, 대칭 관계, 비대칭 관계, 1-to-N 관계 모두를 표현할 수 있도록 학습하는 임베딩 방법이다. ComplEx 임베딩의 경우 score function이 (수식 6)과 같이 나타난다.

$$f_r(h,t) = \sum_i h_i \cdot r_i \cdot \overline{t_i} \quad \text{(수식 6)}$$

3. HolE 임베딩

HolE(holographic embeddings)는 지식 그래프와 같은 관계형 데이터를 임베딩하기 위해 제안된 임베딩 방법이다.[17] HolE는 벡터의 순환 상관관계에 기반하여, 지식 그래프를 임베딩 하는 방법이다. 순환 상관관계를 이용하여, 1-to-N 관계나 대칭, 비대칭 관계 등 다양한 관계를 임베딩할 수 있다. 이러한 HolE 모델은 메모리 복잡성이 낮아 학습에 사용되는 매개변수 수가 적어 효율이 높고 훈련이 쉽다.

그리고 확장성이 뛰어나 다양한 관계를 포착해낼 수 있다. HolE 모델을 기존 임베딩 모델과 비교하여 평가하기 위해 실험을 진행하였다. 40,943개의 entity와 18개의 관계 유형, 151,442개의 트리플을 포함하는 WN18 WordNet 데이터 세트로 벤치마크한 결과 TransE, TransR 방법으로 임베딩한 경우는 링크 예측 성능이 11.5%, 33.5%로 나타났으나, HolE 모델의 경우 93.0%로 높은 테스트 정확도를 보였다. 이를 통해 HolE 모델이 지식 그래프에서 트리플을 예측하는 링크 예측에 높은 성능을 보임을 입증하였다.

Ⅳ. 실험 및 결과

다운로드한 CVE 데이터 257,660건을 전처리하여 총 188,616건의 데이터를 수집할 수 있었다. 전체 CVE 데이터 중에서 2020년도의 CVE 데이터 46,567건을 사용하여 실험을 진행하였다. 이 CVE 데이터를 TF-IDF, Word2Vec 두 가지의 방법으로 워드 임베딩을 수행하고 Kmeans를 이용한 군집화를 진행하여, TF-IDF를 이용한 경우는 1,482건, 43,016건, 2,069건의 데이터를 가지는 세 클러스터로 분류되었고, Word2Vec의 경우는 1,618건, 36,002건, 8,947건의 데이터를 가지는 클러스터로 분류되었다. 분류 결과를 확인했을 때, cluster0의 경우는 두 임베딩 방법에서 비슷한 분류 결과를 보였으며, 나머지 두 클러스터에 비해 매우 적은 양의 데이터만 포함하고 있었으며, 시각화하여 확인했을 때, cluster0의 경우만 다른 데이터들로부터 멀리 떨어져있는 것으로 시각화되었다. 이를 통해 cluster0에 속한 데이터는 다른 데이터와 관련성이 적을 것이라는 것을 확인 할 수 있었다. 양질의 지식 그래프를 생성하기 위해 해당 데이터를 삭제하는 과정을 거쳤다.

그 이후 REBEL 모델을 이용하여, 지식 그래프 생성을 위한 triplet을 추출하였다. 그 후 GoogleNews를 이용해 개체와 관련된 기사를 크롤링하고 기사 본문으로부터 triplet을 추출하여, 지식 그래프를 확장하였다.

본 실험의 평가를 위해 지식 그래프를 임베딩하여 생성한 지식 그래프가 링크 예측(Link Prediction)을 수행할 때, 성능을 측정하였다. 지식 그래프를 임베딩하기 위해, 임베딩 모델 중 ComplEx, TransE, DisMult, HolE를 이용해 지식 그래프 임베딩을 학습시켰다. ComplEx 모델은 Score Function을 거리만을 가지고 계산하는 방법과 달리, head, relation, tail 세 벡터의 내적과 합으로 계산하는 임베딩 방법이며, Score Function을 복소수로 확장한다. TransE는 단순 거리를 이용하여 Score Function을 계산하는 방법으로 head와 relation의 벡터의 합이 tail의 벡터와 같게끔 하는 임베딩 방식이다. DistMult의 경우는, ComplEx처럼 세 벡터의 내적의 합을 Score Function으로 갖는 임베딩 방법이다. HolE는 순환 상관 관계를 사용하여 지식 그래프를 임베딩한다. 비대칭 관계를 표현할 수 있고, 계산의 효율성과 확장성을 증가시킨 임베딩 모델이다.

네 임베딩 모델을 학습시켜 각각의 경우를 평가하기 위해, 생성한 triplet의 80%

를 학습용 데이터로 사용했고, 20%를 테스트 데이터로 사용하여, 평가를 진행했다. 벡터의 크기는 모두 100으로 동일하게 지정하였으며, epoch 수도 동일하게 100회로 진행하였다. optimizer는 adam을 사용하였다.

모델을 평가하기 위한 지표로는 일반적으로 많이 사용되는 MRR(Mean Reciprocal Rank)과 MR, HITS@N 사용하였다. Q를 테스트 트리플 세트라고 할 때, 각 지표는 다음가 같이 정의된다. MRR은 예측 결과 순위 값의 조화 평균이다. MR은 모든 예측 결과 순위 값의 평균이다. HITS@N은 모든 예측 결과에 대해, 상위 N 순위의 비율을 말한다.

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \quad \text{(수식 7)}$$

$$MR = \frac{1}{|Q|}\sum_{i=1}^{|Q|} rank_i \quad \text{(수식 8)}$$

$$Hits@N = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\begin{cases}1 & \text{if } (rank_i \le N)\\ 0 & otherwise\end{cases} \quad \text{(수식 9)}$$

생성한 지식 그래프를 학습시킨 임베딩 모델을 평가한 결과는 [표 8]로 나타났다. Embedding 열은 그룹화 과정에서 사용한 워드 임베딩 방법이다.

model	MRR	MR	hits@10	hits@3	hits@1
ComplEx	0.22	1115.38	0.50	0.20	0.13
TransE	0.24	974.21	0.48	0.38	0.08
DistMult	0.25	1108.16	0.46	0.28	0.14
HolE	0.44	732.63	0.60	0.49	0.34

[표 8] 임베딩 모델 평가 결과

지식 그래프를 평가한 결과, 임베딩 모델 중 HolE 모델의 평가지표가 가장 높게 나타났다. 보안 지식 그래프의 경우, 1-to-N과 같은 관계가 많이 나타나, 1-to-1 관계에 효과적인 TransE 모델의 hits@1 점수가 낮게 예측되었다. 반면에 1-to-N 관계를 표현할 수 있도록 Holographic Embedding을 사용한 HolE 모델은 모든 지표에 대하여, 가장 높은 성능을 보였다. 이를 통해, 보안 지식 그래프를 임베딩하는 방법으로 HolE 모델이 가장 적합할 것이며, 예측 결과가 1순위인 경우가 34%, 3순위 안에 있을 학률이 49%, 10순위 안은 60%로 준수한 성능을 보였다. 본 논문에서

제안한 보안 지식 그래프 생성 프레임워크를 기반으로 생성한 보안 지식 그래프가 링크 예측에서 준수한 성능을 보일 수 있음을 확인하였다.

지식 그래프 내의 개체와 관련된 구글 뉴스 데이터를 이용해 각 클러스터의 지식 그래프를 확장한 뒤, 평가를 진행했다. 평가 결과는 [표 9]과 같이 도출되었다. 확장 과정을 통해 지식 그래프 내의 entity와 relation 수가 늘어남에 따라 MR, MMR 점수는 조금 낮아졌으나, Hits@N 지표에서 Hits@3, Hits@1 점수가 조금 높아졌다. 이를 통해 링크 예측 성능을 높이기 위해, 지식 그래프를 확장하고 업데이트하는 과정이 필요한 것을 다시 확인할 수 있었다. 지식 그래프를 확장한 후, 평가를 진행하는 것으로 본 논문에서 제안한 구글 뉴스를 이용한 지식 그래프 확장 프레임워크를 통해 지식 그래프가 링크 예측 실험에서 성능이 향상될 수 있음을 보였고, HolE(Holographic Embeddings) 방법을 통해 임베딩한 경우 가장 높은 성능을 보이는 것을 확인할 수 있었다.

model	MRR	MR	hits@10	hits@3	hits@1
ComplEx	0.26	1165.60	0.43	0.28	0.18
TransE	0.25	756.50	0.51	0.40	0.08
DistMult	0.28	1151.54	0.51	0.30	0.19
HolE	0.46	856.75	0.60	0.51	0.37

[표 9] 지식 그래프 확장 후 평가 결과

Ⅴ. 결론

본 논문에서는 지식 그래프 자동 확장을 위한 링크 예측이나 검색 엔진 등에서의 정확도를 높이고자, 보안 취약점 데이터를 취약점에 대한 설명의 유사도를 기준으로 분류하여 관련도가 떨어지는 데이터를 삭제하고 지식 그래프를 생성하여 지식 그래프의 질을 높이고자 한다. 또한 추출한 개체와 관련된 구글 뉴스 기사데이터를 이용해, 지식 그래프를 확장하는 방법을 제안하였다. 이를 위해 CVE Data Base에서 보안 취약점 데이터를 수집하고, 대응에 대한 정보를 Exploit Data Base에서 추가로 수집한다. 개체 간의 새로운 관계를 예측하는 링크 예측(Link Prediction)과 같은 임베딩 기반의 예측에서 정확도를 높일 수 있도록, 수집한 데이터를 보안 취약점에 대한 설명의 유사도를 기준으로 군집화한 뒤 데이터 수가 너무 적고 그래프에서 다른 그룹으로부터 멀리 떨어진 것으로 나타나는 데이터를 제거하였다. 효율적인 지식 그래프 생성을 위해, 딥러닝 기반의 REBEL 모델을 이용하여 지식 그래프를 생성한다. 그 후 지식 그래프 내의 개체와 관련된 구글 뉴스를 크롤링해서 얻는 데이터를 이용하여, 지식 그래프를 확장하는 지식 그래프 구축 및 확장 프레임워크를 제시하였다. 지식 그래프 추출 성능을 평가하기 위해, MRR(Mean Reciproccal Rank), MR(Mean Rank). HIT@N 세 가지의 순위 기반 평가지표를 사용하여 점수를 측정하였다. 지식 그래프를 ComplEx, TransE, DistMult, HolE 네 가지의 임베딩 모델에 학습시킨 뒤 각 지표를 평가하였다. 결과, MMR 점수 0.44, MR 점수 732.63, hits@10, hits@3, hits@1 점수가 각각 0.60, 0.49, 0.34로 HolE 방식의 임베딩 모델이 가장 성능이 높게 평가되었다. 이를 통해, 보안 취약점 데이터 기반으로 생성된 보안 지식 그래프의 성능이 준수함을 확인할 수 있었고, 링크 예측을 위한 지식 그래프 임베딩 과정에서 가장 적합한 임베딩 방법을 찾을 수 있었다. 지식 그래프를 확장 시킨 후 평가를 해본 결과, HolE 모델로 임베딩한 경우, 지식 그래프 내의 entity, relation의 수가 늘어나면서 MMR, MR 점수가 확장하기 전보다 조금 낮은 0.46, 856.75로 평가되었다. 하지만 hits@N 점수가 0.60, 0.49, 0.34에서 0.60. 0.51. 0.37로 hits@3, hits@1 점수가 조금 높아졌다. 이를 통해 본 논문에서 제안한 GoogleNews 데이터를 통해 지식 그래프를 확장하는 프레임워크가 보안 지식 그래프를 임베딩하여 수행한 링크 예측 성능을 높

여줄 수 있음을 확인하였다.

향후 연구에서 지식 그래프를 확장하는 과정에서 지식 그래프의 성능을 더 크게 높일 수 있도록, 링크 예측(link prediction) 등 지식 그래프를 완성하기 위한 다양한 확장 방안을 이용하여 지식 그래프 확장을 시도할 것이다. 그리고 지식 그래프를 임베딩 방법을 연구하여, 보안 지식 그래프에 효과적인 지식 그래프 임베딩 방법을 찾을 것이다. 나아가 완성한 보안 지식 그래프와 지식 그래프 임베딩을 이용하여 새롭게 발생하는 보안 취약점 대응 방안을 예측하고 생성하는 것을 목표로 한다.

참고문헌

[1] 김상필, 강정희, “Amazon Neptune- 신규 그래프 데이터베이스 서비스 활용”, 『AWS Summit Seoul 2018』, 2018

[2] Leslie F. Sikos, “Cybersecurity knowledge graphs“, 2023

[3] Bonggeun Choi, Youngjoong Ko, “Knowledge graph extension with a pre-trained language model via unified learning method”『Knowledge-Based Systems』, 2023

[4] Mehwish Alam, Frank van Harmelen, Maribel Acosta, ”Towards Semantically Enriched Embeddings for Knowledge Graph Completion“, 2023.

[5] Yongfu Wang, Ying Zhou, Xiaohai Zou, Quanqiang Miao, Wei Wang, ”The analysis method of security vulnerability based on the knowledge graph“ 『10th International Conference on Communication and Network Security』 pp. 135-145 2020

[6] Anders Mølmen Høst, Pierre Lison, Leon Moonen, “Constructing a Knowledge Graph from Textual Descriptions of Software Vulnerabilities in the National Vulnerability Database” 2023

[7] Shengzhi Qin, K. P. Chow, “Automatic Analysis and Reasoning Based on Vulnerability Knowledge Graph”『Cyberspace Data and Intelligence, and Cyber-Living, Syndrome, and Health』, pp. 3 - 19, 2019

[8] Yan Jia, Yulu Qi, Huaijun Shang, Rong Jiang, Aiping Li. “A Practical Approach to Constructing a Knowledge Graph for Cybersecurity” 『Engineering』 pp. 53-60, 2018

[9] Pere-Lluís Huguet Cabot, Roberto Navigli, “REBEL: Relation Extraction By End-to-end Language generation“, 『Findings of the Association for Computational Linguistics: EMNLP 2021』, pp. 2370-2381, 2021

[10] Xiaosheng Chen, Wendi Shen, Genke Yang, “Automatic Generation of Attack Strategy for Multiple Vulnerabilities Based on Domain Knowledge Graph”, 『IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society』, 2021

[11] Hooi EKJ, Zainal A, Maarof MA, “TAGraph: knowledge graph of threat actor.” 『2019 International Conference on Cybersecurity (ICoCSec)』, 2019

[12] Yan Jia, Yulu Qi, Huaijun Shang, Rong Jiang, Aiping Li “A Practical Approach to Constructing a Knowledge Graph for Cybersecurity”, 『 Engineering』, pp. 53-60, 2018

[13] Kaloroumakis PE, Smith MJ, “Toward a knowledge graph of cybersecurity countermeasures”, 2021

[14] M. Iannacone, S. Bohn, G. Nakamura, J. Gerth, K. Huffer, R. Bridges, et al. “Developing an ontology for cybersecurity knowledge graphs” 『Proceedings of the 10th annual cyber and information security research conference』, 2015

[15] Sikos L, “Cybersecurity knowledge graphs. Knowledge and Information Systems.”, 2023

[16] Sun Y, Lin D, Song H, Yan M and Cao L, “A Method to Construct Vulnerability Knowledge Graph based on Heterogeneous Data”, 『16th International Conference on Mobility, Sensing and Networking』, 2020

[17] Maximilian Nickel, Lorenzo Rosasco, Tomaso Poggio, “Holographic Embeddings of Knowledge Graphs”, 2015

[18] Enrico Palumbo. “Knowledge Graph Embeddings for Recommender System”, 2020.

[19] Muhan Zhang, Yixin. Chen “Link Prediction Based on Graph Neural Networks“ 『NIPS 2018』, 2018.