RAG 기반 CTI 리포트 IoC 자동 추출 파이프라인 구축

스마트보안전공 202334860 한보윤

인공지능융합보안

01 프로젝트 개요

배경

위협 인텔리전스 리포트가 방대함
+
포맷 불규칙
↓
수동 분석의 한계 → 자동화 필요성

문제점

표현 변형(hxxp://, [.] 등)과
긴 문서로 단순 키워드 검색의 한계
↓
IoC 수집 지연되어
탐지 공백 및 대응 지연 발생 가능성

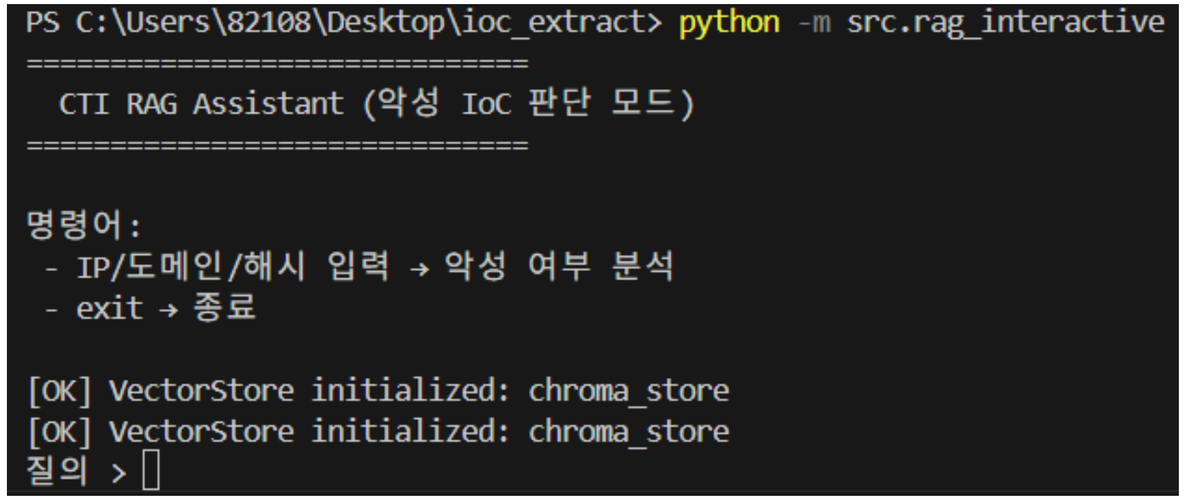
프로젝트 목표

RAG로 관련 문단 검색
+
LLM, Regex로 IoC 추출
↓
정확성있는 근거 문단 확보를 통한
사고 대응력 향상에 도움

‘RAG 및 LLM로 CTI 리포트에서 IoC 자동 추출/정규화하는 파이프라인’

02 전체 과정

1. pdf 형식의 CTI report에서 PDF 텍스트 추출
2. 언어 감지(한글 / 영어) 후 한국어 문서는 translator.py 에서 영어로 번역 후 저장
3. ioc에 대해 패턴 전처리 진행 (hxxp → http 등)
4. regex로 후보 IoC 추출 {"type": "domain", "value": "abc.com"}
5. 후보 주변 문맥과 함께 LLM 호출로 ioc 판단 및 근거 문장 생성(한글)
6. 벡터화 후 vector DB에 저장 (SentenceTransformer 임베딩 기반)

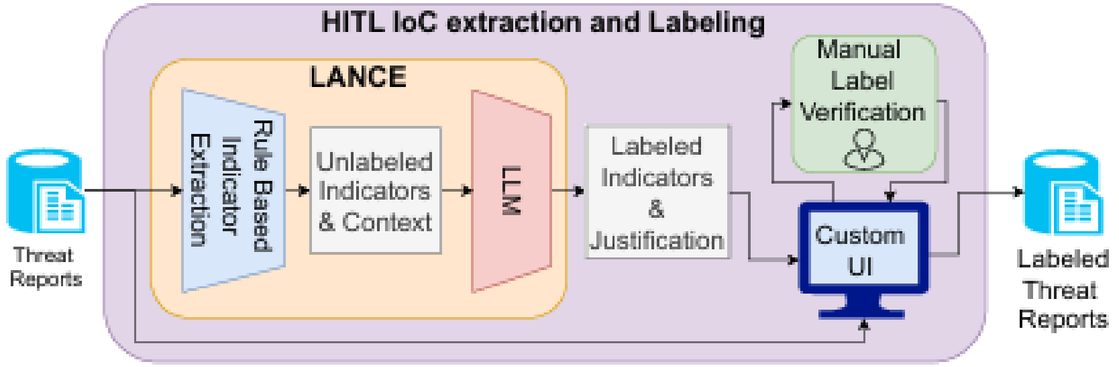


추후 LLM에 IoC를 옵션으로 질의하면 악성 여부와 근거 문장을 함께 제시

03 관련 연구

Uncovering Reliable Indicators:
Improving IoC Extraction from Threat Reports

Evangelos Froudakis*, Athanasios Avgetidis*, Sean Tyler Frankum†,
Roberto Perdisci†, Manos Antonakakis*, Angelos Keromytis*
*Georgia Institute of Technology
{efroudakis3, avgetidis, manos, angelos}@gatech.edu
†University of Georgia
{Sean.Frankum, perdisci}@uga.edu



설명 가능한 IoC 데이터셋 구축 - 50개의 pdf 형식의 Threat Report로 1,791개 IoC를 추출

→ HITL(human-in-the-loop) 중심의 연구

해당 연구와의 차별점 : 자동화된 설명 가능한 IoC 추출 및 평가 시스템 구현 + LLM 질의를 통한 IoC 검색

04 평가용 데이터셋

Mendeley Data

CTI and APT Related Dataset and Source Code for the Paper in Short: DEVIL

Files

Root > Zip-Files

apt-groups.zip	9.19 KB
aptnotes-downloader.zip	207 KB
apt-reports.zip	3.51 GB
countries.zip	1.64 KB
cti-transformation-app.zip	4.56 MB
extracted-iocs.zip	4.3 MB
graph-db-backup.zip	169 MB
ioc-searcher-app.zip	51.6 KB
malware-families.zip	13.8 KB
report-analyser.zip	4 MB
ttps.zip	4.43 KB

```
ip4 1.0.7.0
ip4 125.214.195.17
ip4 196.29.166.218
md5 1507e7a741367745425e0530e23768e6
md5 1bfbc0c9e0d9ceb5c3f4f6ced6bcfeae
md5 1f7897b041a812f96f1925138ea38c46
md5 4cc10ab3f4ee6769e520694a10f611d5
md5 6dffcfa68433f886b2e88fd984b4995a
md5 7b4a8be258ecb191c4c519d7c486ed8a
md5 85d316590edfb4212049c4490db08c4b
md5 911de8d67af652a87415f8c0a30688b2
md5 c1364bbf63b3617b25b58209e4529d8c
md5 cb52c013f7af0219d45953bae663c9a2
```

Devil dataset

데이터 내부의 apt-reports 파일과 extracted-iocs 파일 사용
→ 각각 ioc 추출 / 검증용
→ 30개 리포트 선별 (3,096개의 IoC 담겨있음)
→ 악성 라벨링된 ioc와 추출한 ioc 비교로 성능 평가

05 디렉토리 구조 및 코드 기능

```
ioc_extract/
├── extracted-iocs/
│   └── sample_30_en/
│       ├── iocs/                ← Ground truth IoC 정답 (.iocs 파일들)
│       └── reports/             ← CTI 보고서 PDF 30개 (입력)
│
├── outputs/                     ← NLP 파이프라인 결과 (.json)
│
├── evaluation_report.csv        ← Precision/Recall/F1 평가 결과 (자동 생성)
│
├── .env                         ← OpenAI API 키 및 환경변수 저장
│
└── src/                         ← 핵심 코드 디렉토리
    ├── __init__.py
    ├── config.py                ← 전역 설정 (USE_LLM_LABELER 등)
    ├── evaluate_ioc_accuracy.py ← NLP + 평가 자동 통합 실행 메인 파일
    ├── ioc_patterns.py          ← Regex 기반 IoC 후보 탐지 모듈
    ├── llm_labeler.py           ← LLM 기반 IoC 검증 / 이유(reason) 생성
    ├── nlp_pipeline.py          ← 전체 PDF 처리 및 IoC 추출 (LLM 호출 포함)
    ├── translator.py            ← 다국어 → 영어 번역 (M2M100)
    ├── utils_text.py            ← 텍스트 전처리 및 공용 함수
    ├── vector_store.py          ← ChromaDB 사용 (cosine similarity 기반)
    └── rag_interactive.py       ← RAG 질의 테스트용 콘솔 인터페이스
```

주요 코드 기능

nlp_pipeline - 메인 코드
evaluate_ioc_accuracy - 평가 코드

1.평가코드에 메인코드 실행 로직을 넣어,
devil dataset에 대해 전체 파이프라인 실행 후 평가까지 진행

2.메인코드에 외부 pdf에 대해 ioc 추출 추가 진행으로,
vectorDB의 데이터 구축

06 정규화 진행 전후

regex로 후보 ioc 추출 시
정규화 진행으로 형식 통일 → hxxp → http / [.] → .

정규화 전

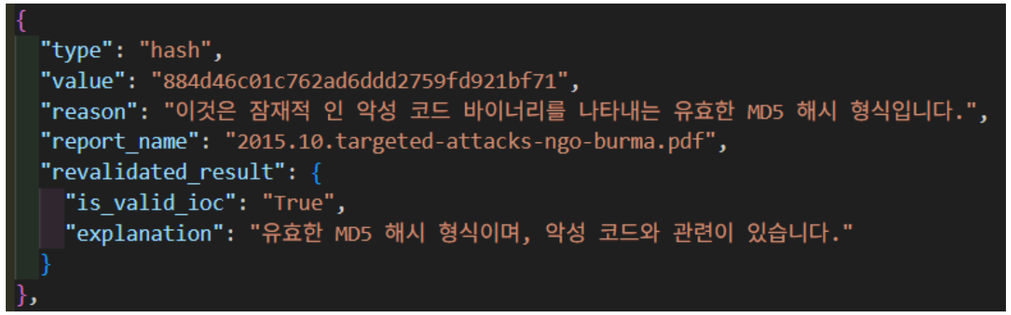
– FRP & LCX C2
hxxp://sk1.m00nlight[.]top:80
hxxps://fk.m00nlight[.]top:443

정규화 후

```
"type": "URL",
"value": "http://sk1.m00nlight.top:80",
```

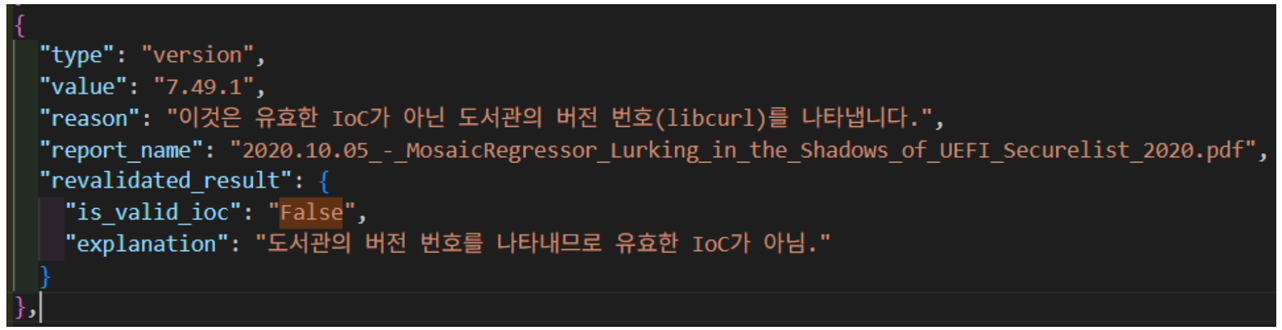
```
"type": "URL",
"value": "https://fk.m00nlight.top:443",
```

07 실행 결과 - 평가코드



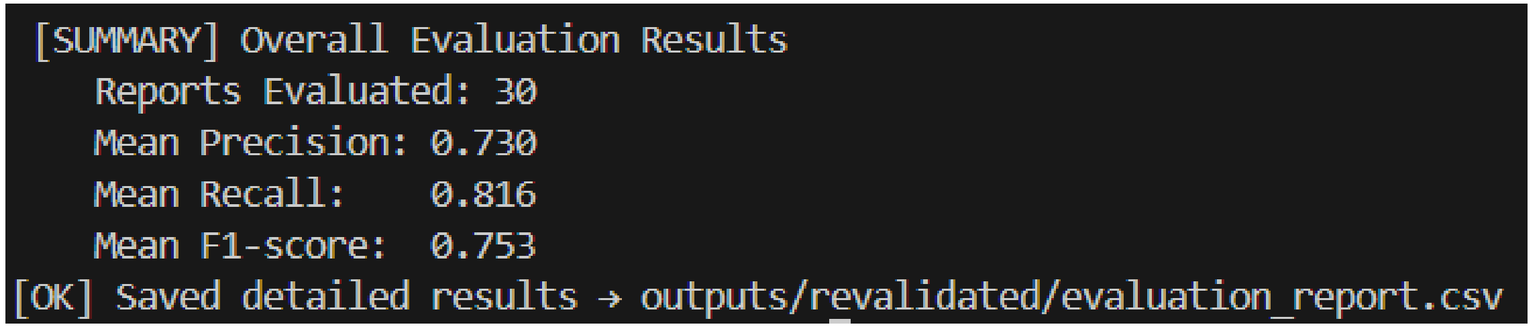
```
{
  "type": "hash",
  "value": "884d46c01c762ad6ddd2759fd921bf71",
  "reason": "이것은 잠재적 인 악성 코드 바이너리를 나타내는 유효한 MD5 해시 형식입니다.",
  "report_name": "2015.10.targeted-attacks-ngo-burma.pdf",
  "revalidated_result": {
    "is_valid_ioc": "True",
    "explanation": "유효한 MD5 해시 형식이며, 악성 코드와 관련이 있습니다."
  }
},
```

저장된 json 파일 - IoC True 식별



```
{
  "type": "version",
  "value": "7.49.1",
  "reason": "이것은 유효한 IoC가 아닌 도서관의 버전 번호(libcurl)를 나타냅니다.",
  "report_name": "2020.10.05_-_MosaicRegressor_Lurking_in_the_Shadows_of_UEFI_Securelist_2020.pdf",
  "revalidated_result": {
    "is_valid_ioc": "False",
    "explanation": "도서관의 버전 번호를 나타내므로 유효한 IoC가 아님."
  }
},
```

저장된 json파일 - IoC False 식별



```
[SUMMARY] Overall Evaluation Results
   Reports Evaluated: 30
   Mean Precision: 0.730
   Mean Recall:    0.816
   Mean F1-score:  0.753
[OK] Saved detailed results → outputs/revalidated/evaluation_report.csv
```

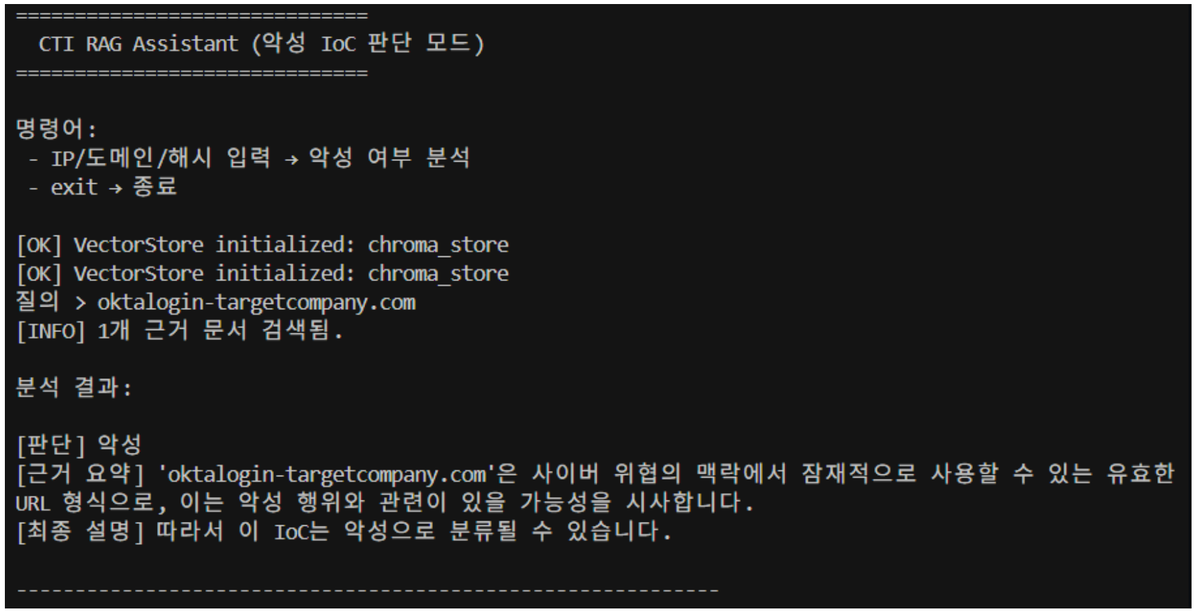
73% 이상의 탐지율

08 실행 결과 - 파이프라인

새 pdf를 pdf 모음 폴더에 경로 설정 후 nlp_pipeline.py 실행하면 DB 저장까지 자동으로 진행


```
"type": "URL",
"value": "oktalogin-targetcompany.com",
"reason": "이것은 사이버 위협의 맥락에서 잠재적으로 사용할 수있는 유효한 URL 형식입니다.",
"report_name": "250729.pdf"
```


Devil dataset(평가용 데이터셋) 아닌 외부 데이터로 실험
: nlp_pipeline.py를 돌려 json 파일 생성
+
생성된 데이터들은 vectorDB에 저장됨



질의를 통해 나온 결과
: 실제 IoC에 대한 악성 유무 판단과 근거 제시가 적절히 이루어짐