

2025-2 산학프로젝트 중간 보고서 (Team. Genesis)

전영우(2021041072)
고재현(2023078075)
심수민(2021041067)

목차

1. 프로젝트 계획
2. 구현 결과
3. 추후 계획

1. 프로젝트 계획(Sprint Backlog 기반)

↓ Genesis_SB-0N_V1 .xlsx

스프린트 ID	에픽 ID	스토리 ID	유저 스토리	태스크 ID	태스크명	소요기술	담당자
SP001	E-1	S1-1	사용자는 이메일 본문 내 URL이 피싱 의심일 경우 자동으로 탐지 및 표기되길 원한다	T1-1	메일 DB에 저장된 본문에서 URL 분류	Java / DB	고재현
				T1-2	Mail API 연동 (Gmail, Naver)	Java / Google Open API / Naver IMAP, POP3	
				T1-3	UI 디자인	React / html	
				T1-4	탐지 결과 인터페이스 표기 기능구현	React / html	
				T1-5	관리/피드백 루프 구축		
		S1-2	시스템은 의심 URL 클릭시 접속을 차단하고 경고 메시지를 제공해야한다.	T2-1	URL 차단 로직 구현	Java Spring Boot	전영우
				T2-2	경고 메세지 팝업 알림	React / html	
				T2-3	URL 의심 여부 판별	Java Spring Boot	
				T2-4	URL 접속 허용	Java Spring Boot	
				T2-5	URL 접속 차단	Java Spring Boot	
		S1-3	시스템은 외부 피싱 DB/LLM API와 연동하여 탐지 정확도를 높인다.	T3-1	메일 본문/제목 분석용 LLM 모델 설정	조사 후 선정	고재현 + 전영우
				T3-2	메일 한국어 데이터셋 확보	AI Hub/Kaggle/Hugging Space	
				T3-3	LLM 모델 학습	Python	
				T3-4	어플리케이션과 상호작용 위한 API 구현	Java Spring Boot	
	E-2	S2-1	사용자는 읽고 싶은 메일을 선택한다.	T4-1	수신된 메일 목록 화면 구현(UI)	React / html	심수민
				T4-2	메일 목록에서 클릭 이벤트 구현	React / html	
				T4-3	선택한 메일의 정보(첨부파일 유무, 본문 글)분석 로직 구현	Java	
		S2-2	시스템은 사용자가 선택한 메일의 첨부파일의 Hash값을 계산해야한다.	T5-1	첨부파일 정보 판별(압축파일/실행파일/문서파일 등 구분)	Java	
				T5-2	첨부파일 종류에 따른 처리 로직 구현(압축파일과 그외 파일일 경우로 분기)	Java	
				T5-3	파일 Hash 연산 수행 및 저장	Java Spring Boot	
				T5-4	압축파일 내부 정보 파싱(파일명 등)	Java Spring Boot	
		S2-3	시스템은 Hash값(MD5/SHA256 등)을 Virus Total API에 전송해야한다.	T6-1	파일 악성 판단 기준 설정	검색/Virus Total docs	Java Spring Boot
				T6-2	Virus Total API 연동	Java Spring Boot	
				T6-3	악성 파일 판단에 필요한 데이터 파싱	Java Spring Boot	
		S2-4	Virus Total API에서 첨부파일 해시값에 대응되는 JSON DATA 받아온다.	T7-1	메일 정보 db연동	Java Spring Boot	Java Spring Boot
				T7-2	판단 기준에 따른 결과 제공 로직 구현	Java Spring Boot	
		S2-5	시스템은 받아온 DATA를 사용자에게 GUI를 통해 제공해야한다.	T8-1	첨부파일 검사중 UI 구현	React / html	
				T8-2	첨부파일 검사 완료/결과별 UI 구현	React / html	
				T8-3	첨부파일 다운로드 경고 구현	React / html	
				T8-4	검사 결과와 UI 연동	React / html	

구현 목표

E-1. 피싱/스미싱 탐지

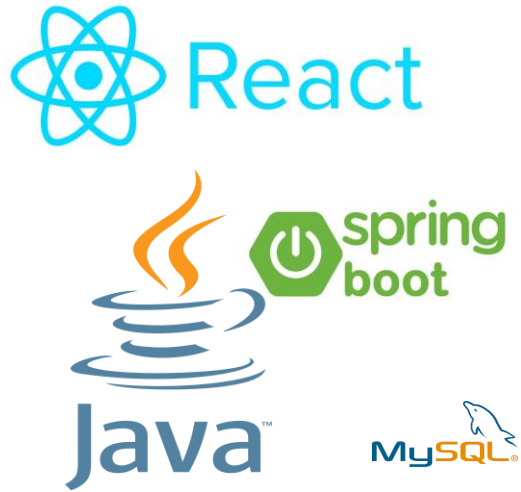
- S1-1 본문 분석
- S1-2 실시간 차단
- S1-3 외부 피싱
- DB/LLM API와 연동

E-2. 피싱/스미싱 탐지

- S2-1 메일 선택
- S2-2 Hash 연산 기능
- S2-3 외부 API 연동
- S2-4 첨부파일 정보 수집
- S2-5 GUI 제공

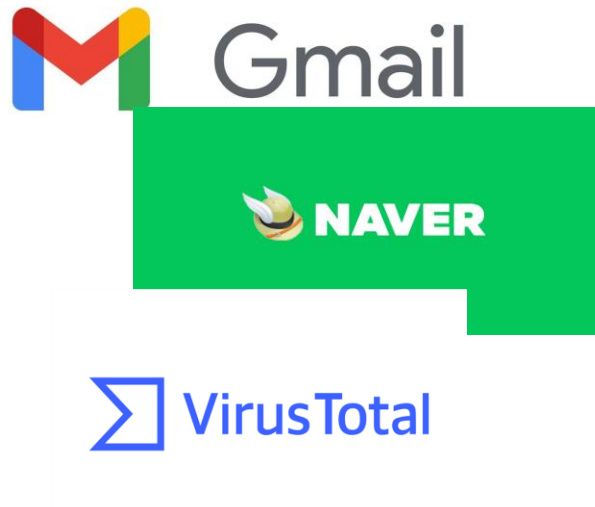
1. 프로젝트 계획(사용 도구 & 협업 도구)

사용 도구



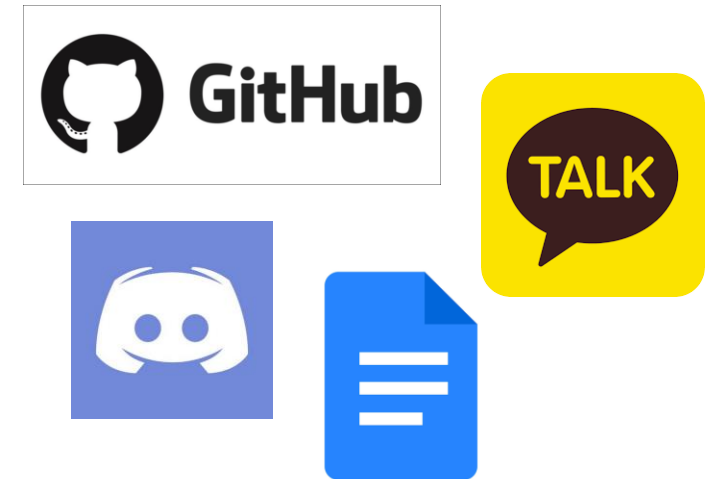
- 프론트 : React
- 백엔드 : SpringBoot
- 로직 구현 : Java
- DB : MySQL

사용 API



- 메일 로딩
: Naver IMAP / Gmail API
- 악성 메일 판단 : VirusTotal

협업 도구



- 의사소통
: KakaoTalk / Discord
- 프로젝트 관리 : Github
- 문서 작성 : Google Docs

2. 구현 결과(Gmail Apl , Naver Imap 연동)

- Google Cloud Console에서 OAuth 2.0 클라이언트 ID 발급 완료
- Gmail API를 통한 메일 자동 수신 기능 구현
- Naver IMAP 프로토콜을 이용한 메일 수신 기능 구현
- 최근 메일 목록 조회 및 상세 정보 가져오기 완료

← → ↻ 🌐 localhost:8888/Callback?code=4/0Ab32j90KXIt5QjE_zkPu3le-fuy0urfQaP1Cd29U4f5TQemJ8q8FXtUloCnELJNvd3WCw&scope=https://www.googleapis.com/auth/gmail.readonly

Received verification code. You may now close this window.

```
{
  "emails": [
    {
      "messageId": null,
      "from": "네이버 <account_noreply@navercorp.com>",
      "to": "albert0827@naver.com",
      "subject": "2단계 인증을 위한 애플리케이션 비밀번호 생성",
      "receivedDate": "2025-10-25T08:28:30.000+00:00",
      "content": "
<!-- 아웃룩용 max-width 백 -->
<!--[if (gte mso 9)](IE)]>
<table border=
<tr>
<td width=
<![endif]-->
<!-- Header -->
2단계 인증을 위한 애플리케이션 비밀번호가 생성되었습니다.
--
```

2. 구현 결과(url 추출 및 분류)

- 의심스러운 도메인 분류(단축 URL, .xyz, .top)
- 추출된 URL 정보를 EmailDto에 저장피싱 탐지 엔진
- 한글/영문 피싱 의심 키워드 탐지 ("긴급", "계좌", "클릭", "urgent" 등)
- 0-100 점수 기반 위험도 평가 (SAFE/SUSPICIOUS/DANGEROUS)

LiveWiki <noreply_livewiki@snowcorp.com>

Livewiki 개인정보 이용 · 제공 내역 및 수집 출처 안내

LiveWiki 개인정보 이용내역 안내 안녕하세요. LiveWiki입니다. LiveWiki는 개인정보보호법 제20조, 20조의2에 따라 연 1회 이상 모든 회원님께 개인정보 이용 · ...

2025. 10. 21. 오후 9:00:39

SAFE (20점)

네이버 <account_noreply@navercorp.com>

2단계 인증 로그인이 설정되었습니다.

<!-- 아웃룩용 max-width 핵 --> <!--[if (gte mso 9)](IE)> <table border="0" cellpadding="0" cellspacing=...

2025. 10. 25. 오후 5:27:46

SUSPICIOUS (25점)

Temu <email_at_kr_temuemail_com_46zyjyc5f8_22e4dc1f@privaterelay.appleid.com>

(광고)추가 할인 혜택을 받았습니다.

----- Temu -----

2025. 10. 24. 오후 9:34:21

SUSPICIOUS (25점)

Google <no-reply@accounts.google.com>

보안 알림

[image: Google] Mac에서 새로 로그인함 jhyeonkoh@gmail.com Mac 기기에서 내 Google 계정에 새로 로그인했습니다. 직접 로그인한 것...

2025. 10. 21. 오전 12:09:51

DANGEROUS (50점)

Upbit info <info@upbit.com>

[업비트] 출금완료 안내

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtm...

2025. 10. 25. 오후 10:06:05

DANGEROUS (100점)

2. 구현 결과(악성 메일 판별 LLM)

1차 시도 - 딥러닝

1. 파일 업로드 형식 인풋 입력
2. 모델 트레이닝

성과

Colab 기반 학습, 예측 LLM 구현 - 2개 버전

1. 제목, 본문 동시에 고려
2. 텍스트 기반으로 구분

업로드 위젯은 현재 셀이 실행된 브라우저 세션에서만 사용할 수 있습니다. 이 셀을 다시 실행하여 업로드 위젯을 활성화하십시오.

Starting spark_email_email_1200.csv to spark_email_email_1200.csv

파일 목록: [1: 100, 0: 100]

| index | title | body | label | |
|-------|-------|--------------|---|---|
| 0 | 1 | 광고 포함 수상 서명 | 회원님의 계정에서 비정상적인 로그인 시도가 감지되었습니다. 아래 링크에서 즉시 확인... | 1 |
| 1 | 2 | 광고 포함 반갑 이벤트 | 두자 전문가의 추천으로 단기간에 높은 수익을 올릴 수 있는 기회를 잡으세요. | 1 |
| 2 | 3 | 무료 배송 마지막 기회 | 귀하께서는 이번 달 행동의 주인공으로 선정되었습니다. 링크를 클릭해 경품을 수령하세요. | 1 |
| 3 | 4 | 광고 포함 수상 서명 | 하루 10분 투자로 200만원 수익을 올릴 수 있습니다. | 1 |
| 4 | 5 | 건강 보안 경고 | 이메일 확인 후 간단한 절차로 경종을 수령하세요. | 1 |

```
3) 학습/검증/테스트 분할

train_df, temp_df = train_test_split(df, test_size=0.3, random_state=SEED, stratify=df['label'])
valid_df, test_df = train_test_split(temp_df, test_size=0.5, random_state=SEED, stratify=temp_df['label'])
print(len(train_df), len(valid_df), len(test_df))

140 30 30

4) 토큰라이저 및 데이터셋 변환

from transformers import AutoTokenizer
from datasets import Dataset, DatasetDict

tokenizer = AutoTokenizer.from_pretrained(model_name)

def tokenize(batch):
    # Combine 'title' and 'body' into a single 'text' column for tokenization
    batch['text'] = [f'{title} {body}' for title, body in zip(batch['title'], batch['body'])]
    return tokenizer(batch['text'], truncation=True, max_length=512)

ds = DatasetDict({
    'train': Dataset.from_pandas(train_df[['title', 'body', 'label']]),
    'validation': Dataset.from_pandas(valid_df[['title', 'body', 'label']]),
    'test': Dataset.from_pandas(test_df[['title', 'body', 'label']])
})

tokenized = ds.map(tokenize, batched=True, remove_columns=['title', 'body'])
tokenized = tokenized.remove_columns(['label', 'body'])
tokenized.set_format('torch')

Map: 100% 140/140 [00:00<00:00, 2564.63 examples/s]
Map: 100% 30/30 [00:00<00:00, 985.04 examples/s]
Map: 100% 30/30 [00:00<00:00, 775.56 examples/s]
```

```
7) 추론 예시

def predict_spm(samples):
    # title + body를 하나의 텍스트로 결합
    texts = [f'{s.get("title", "")} {s.get("body", "")}'.strip() for s in samples]

    # 토큰라이저 및 모델 인력
    enc = tokenizer(
        texts,
        truncation=True,
        padding=True,
        max_length=512,
        return_tensors='pt'
    ).to(model.device)

    # 예측 수행
    with torch.no_grad():
        probs = torch.softmax(model(enc).logits, dim=-1).cpu().numpy()

    # 결과 구성
    results = []
    for s, text, p in zip(samples, texts, probs):
        results.append({
            'title': s.get('title', ''),
            'body': s.get('body', ''),
            'spam_prob': float(p),
            'pred': 'spam' if p > 0.5 else 'ham'
        })
    return results

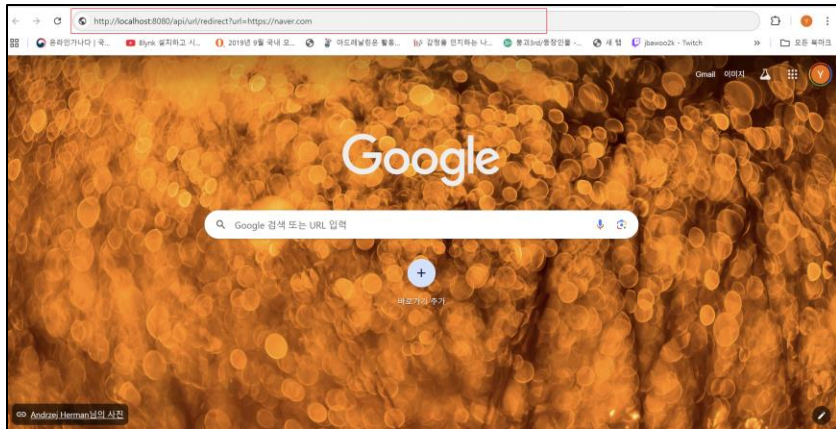
# 예시
samples = [
    {'title': 'Zbano0khaail.com 관련 보안 경고', 'body': '계정 보안 설정을 확인하세요.'},
    {'title': '[알림] 한라산(대)이행원] 개인정보 이용내역 안내', 'body': '최근 개인정보 이용 내역을 알려드립니다.'},
    {'title': '구독 이메일 관련 경고 안내 (2025년 10월 30일 시행)', 'body': '변경된 약관을 꼭 확인해 주세요.'},
    {'title': '[도서관] ★중요★ 도서관 홈페이지 비밀번호 변경 필수 안내', 'body': '비밀번호 변경을 완료하지 않으면 로그인에 제한됩니다.'},
    {'title': '[리브라] 회원 개인정보 이용 제정 내역 안내', 'body': '리브라 회원님의 개인정보 이용내역을 안내드립니다.'},
    {'title': '[광고]클라우드★ 주어진 수익 창출 안내(후회하는 30자)', 'body': '지금 바로 참여 신청 시 혜택 제공!'},
    {'title': '[TNA17ME] 전통장 건강 발효 디스커드 제품 이용 및 이 자료 영의이름 안내', 'body': '이 자료와 제품 정보를 확인해 주세요.'}
]

predict_spm(samples)
```

2. 구현 결과(URL Blocker)

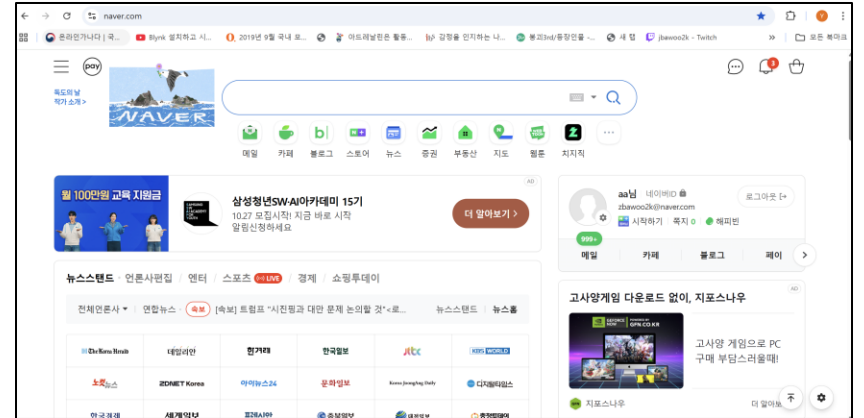
Black-list 기반 도메인 차단 기능 구현

1. 파이썬 이용한 악성 도메인 csv파일 생성
2. csv 파일 DB에 업로드
3. 스프링 부트 실행 후 리다이렉션 차단 확인

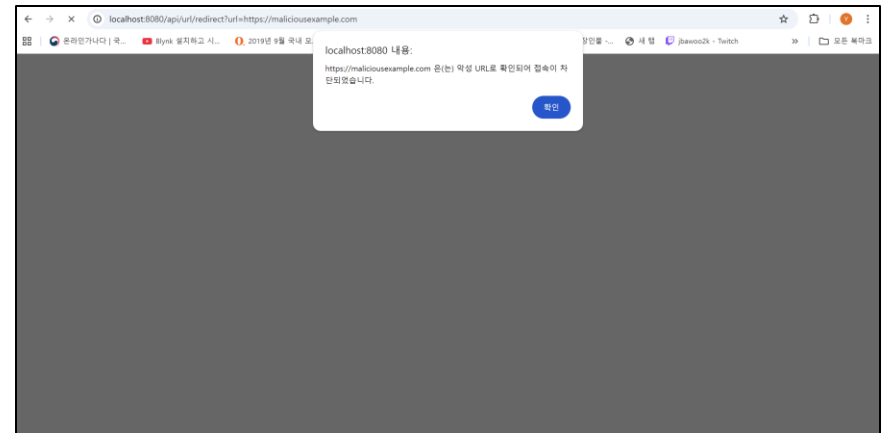


차단 로직

정상



차단



2. 구현 결과(API & Mail 테스트)

1. VT(Virus Total)에서 결과(JSON) 응답->출력

1-1. Virus Total API Key 발급 후 연동

1-2. URL에 HASH값 입력해 파일 확인 테스트

localhost:8080/api/virustotal/files/d5e974a3386fc99d2932756ca165a451

HASH값 입력

pretty print 적용 ☒

```
{
  "md5": "d5e974a3386fc99d2932756ca165a451",
  "meaningfulName": "43.docx",
  "malicious": 40,
  "suspicious": 0,
  "harmless": 0,
  "undetected": 23,
  "lastAnalysisDate": "2025-10-05 15:17:39",
  "sha256": "0193bd8bcbce9765dbecb288d46286bdc134261e4bff1f3c1f772d34fe4ec695"
}
```

2. Gmail API, Naver IMAP 사용 테스트 진행

2-1. Gmail 데이터 확인 위해 OAuth 연동 후 테스트

2-2. Naver Mail 데이터 확인 위해 IMAP 테스트

Gmail API, Google OAuth login Test

1. Google 계정으로 로그인

2. 최신 메일 가져오기

W


제목: test

From: "심수민" <suminy02@gmail.com>

첨부파일:

• [test.zip](#)

최신 메일 목록 (5개)

| 보낸 사람 | 제목 | 받은 날짜 |
|---|---|------------------------------|
| =?utf-8?B?64Sk7J2067KE?=<account_noreply@navercorp.com> | 2단계 인증을 위한 애플리케이션 비밀번호 생성 | Thu Oct 16 23:58:39 KST 2025 |
| =?utf-8?B?64Sk7J2067KE?=<account_noreply@navercorp.com> | 2단계 인증을 위한 애플리케이션 비밀번호 생성 | Thu Oct 16 22:09:12 KST 2025 |
| GitHub <noreply@github.com> | [GitHub] A personal access token (classic) has been regenerated for your account | Wed Oct 15 22:27:23 KST 2025 |
| "\"=?utf-8?B?7Lap67aB64yA7ZWZ6rWQIOuPhOyEnOq0gA==?=\"<library@chungbuk.ac.kr> | 도소樂 『도서관 속 미술관』 개최 안내 | Wed Oct 15 16:31:50 KST 2025 |
| Google <no-reply@accounts.google.com> |  @gmail.com 관련 보안 경고 | Wed Oct 15 12:13:42 KST 2025 |

2. 구현 결과 (Gmail 첨부파일 위험성 확인 & DB 연동)

1. VT에 보내 검사할 파일의 HASH값 연산(SHA256)

1-1. 만약 첨부파일이 압축파일 아니면 그대로

연산

1-2. 만약 첨부파일이 .Zip 확장자(압축파일)라면 압축해제

연산

1-3. 압축파일에 비밀번호 걸려있으면 입력 받는다.

2. SHA256값 이용해 첨부파일 악성여부 확인

2-1. SHA256값으로 VT에서 결과값 받아온다.

2-2. 받아온 결과 파싱 -> DB에 저장(추후 리포트 출력 시 사용)

2-3. DB에 존재하는 SHA256값이면 저장 없이 다음 단계 넘어

2-4. 사용자 화면에 악성파일 이름, malicious, suspicious 정보 출력

Gmail API, Google OAuth login Test

[1. Google 계정으로 로그인](#)

2. 최신 메일 가져오기

제목: test

From: "심수민" <suminy02@gmail.com>

첨부파일:

test.zip -Download VirusTotal 검사 암호 (zip인 경우)

<로그인&메일 가져오는 페이지>

EICAR test file
테스트

Gmail API, Google OAuth login Test

[1. Google 계정으로 로그인](#)

2. 최신 메일 가져오기

제목: test <압축파일 비밀번호 입력&검사>

From: "심수민" <suminy02@gmail.com>

첨부파일:

test.zip -Download VirusTotal 검사 test.txt (as: eicar.com.txt): [Malicious: 66, Suspicious: 0]

| id | created_at | harmless_count | last_analysis_date | malicious_count | md5 | meaningful_name | sha256 | suspicious_count | undetected_count |
|----|----------------------------|----------------|---------------------|-----------------|----------------------------------|-------------------------------|---|------------------|------------------|
| 1 | 2025-10-19 02:56:42.842937 | 0 | 2025-10-04 00:29:31 | 1 | 8d8b9225b063521eae09ed221049cd1c | Unconfirmed 840210.crdownload | 88f5b0ad4edb5265615d2cde0dcd6bfa4b4df1804da592bdc... | 0 | 71 |
| 2 | 2025-10-19 03:06:50.305526 | 0 | 2025-10-05 15:17:39 | 40 | d5e974a3386fc99d2932756ca165a451 | 43.docx | 0193bd8bcbce9765dbecb288d46286dc134261e4bfff13c1f... | 0 | 23 |
| 10 | 2025-10-24 16:40:34.038490 | 0 | 2025-10-24 23:57:29 | 27 | 46f5a2281c4738f6c30459b4b765577 | cmd3.exe | h6604865381a19e802488001817e0e6b73b6504aa556b784... | 0 | 44 |
| 12 | 2025-10-25 03:50:30.381090 | 0 | 2025-10-25 12:40:29 | 66 | 44d88612fea8a8f36de82e1278abb02f | eicar.com.txt | 275a021bbbf6489e54d471899f7db9d1663fc695ec2fe2a2c4... | 0 | 3 |

<검사 결과 DB에 저장>

2. 구현 결과 (발생한 문제점)

1. 한국어 데이터셋 확보

악의적 용도 활용 가능성으로 인해
한글버전 악의적 메일 데이터셋 확보가
어려운 문제점 확인

고려 중인 대안 사항

1. OpenAI API 이용한

프롬프팅 - 답변 기반 작동 형식

- 참고 <https://annajin.tistory.com/226>

2. OpenAI API Text Embedding

OpenAI API 에 입력을 줄 때 embedding
값을 활용하여 분류를 진행하는 형식

- 참고 <https://wikidocs.net/200466>

3. 직접 메일 데이터셋 생성

메일에 있는 스팸 메일/ 정상 메일을
활용하여 직접 데이터셋을 생성

4. 영어 데이터셋 임시적 활용

우선적으로 영어 데이터셋을 활용하여
정상적으로 분류가 이루어지는지 확인

2. API 사용량 제한

API 요청 횟수 제한이 있으나 초기 버전 테스트
및 개발에는 문제 없다. 단, 메일을 대용량으로
처리하려면 문제가 발생 할 수 있다.

실시간 메일 검사는 delay 있지만 수행은
가능하다고 판단됨.
사용자가 선택해 메일을 검사하도록 하는 건
선택 개수 제한 필요.

VT API : Limit 제한

- 분당 4번 요청 가능/하루 500번 요청 가능

고려 중인 대안 사항

1. 데이터베이스 이용

데이터 베이스에 검사 완료한 Hash값에 대한
정보는 저장해두고 불러와 사용하도록 구현

2. 여러 개의 API KEY이용

단, 이 경우는 API KEY 이용 차단이 된 경우가 존재
해 차선택으로 둔다.(가능하면 다른 방법 찾기)

3. 추후 계획

협의 중

1. DB 설계

통합 과정에서 문제가 발생하지 않도록 구체적인 설계가 아니더라도 상의를 통해 결정

2. 메일 분류 LLM 설계

한국어 데이터셋을 확보할 수 있도록 하되, 확보가 어려운 것으로 판단되면 프롬프팅을 이용한 질의 응답 기반 작동 하는 방식도 고려해 보기.

단, 정확도 및 안정성 측면을 고려해서 어떻게 제어할 수 있을지 대한 의견 확보하기.

진행 예정 사항

1. 개발 환경 공유

개발 중 버전 충돌로 인한 통합 실패를 방지하기 위해 각자 개발 버전 및 환경 세팅을 사전에 공유하기

2. 피싱/스미싱 탐지

데이터베이스 연동 준비

H2 Database 또는 MySQL 연동 검토 중 Entity 설계 및 Repository 구조 계획 중 메일 정보 및 URL 분류 결과 저장 기능 구현 예정

application.properties 설정 파일 작성

- 민감 정보 분리 (credentials.json, API 키 등)
- application.properties.example 파일로 예시 제공 계획

3. Sprint Backlog에 맞춰 추가 기능 구현