Qshing 탐지 어플 개발

QR Code 속 숨겨진 피싱
피싱 사이트 탐지 어플리케이션, 안심 QR

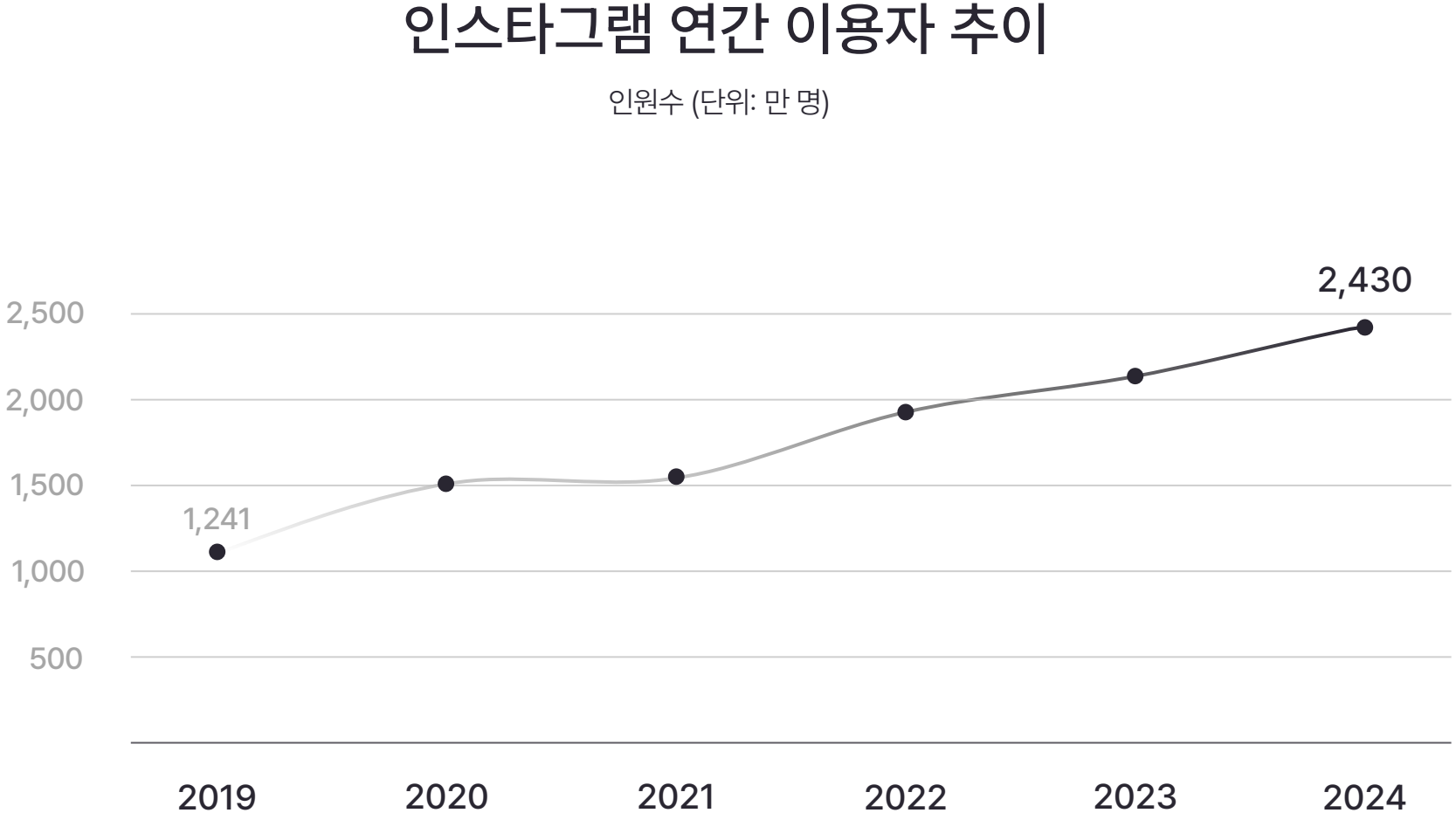
중부대학교 정보보안S/W융합전공 | 정진호 | 박우경 | 이하영 | 최수민 | 홍준희

JOONGBU UNIVERSITY

1

프로젝트 선정 이유 및 목표

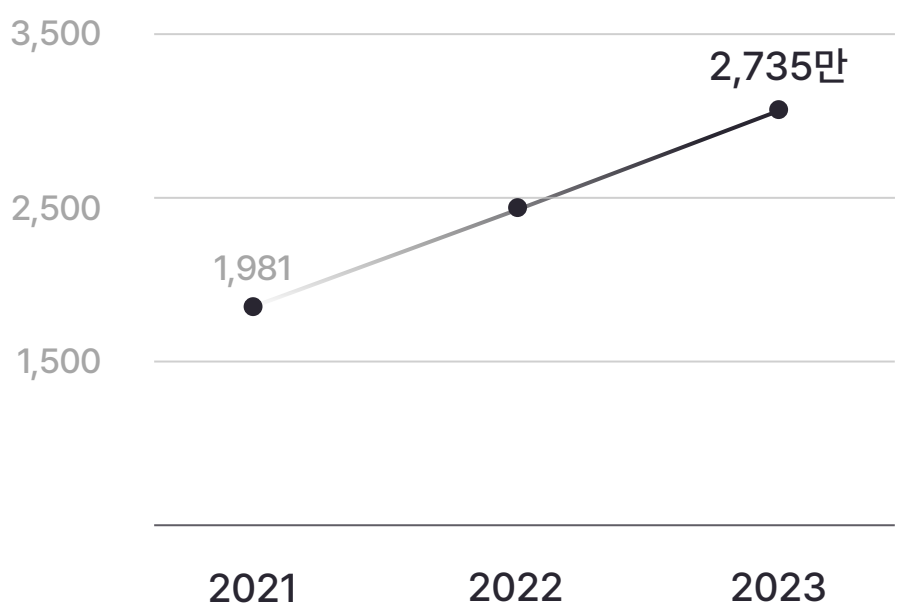
인스타그램 연간 이용자 추이

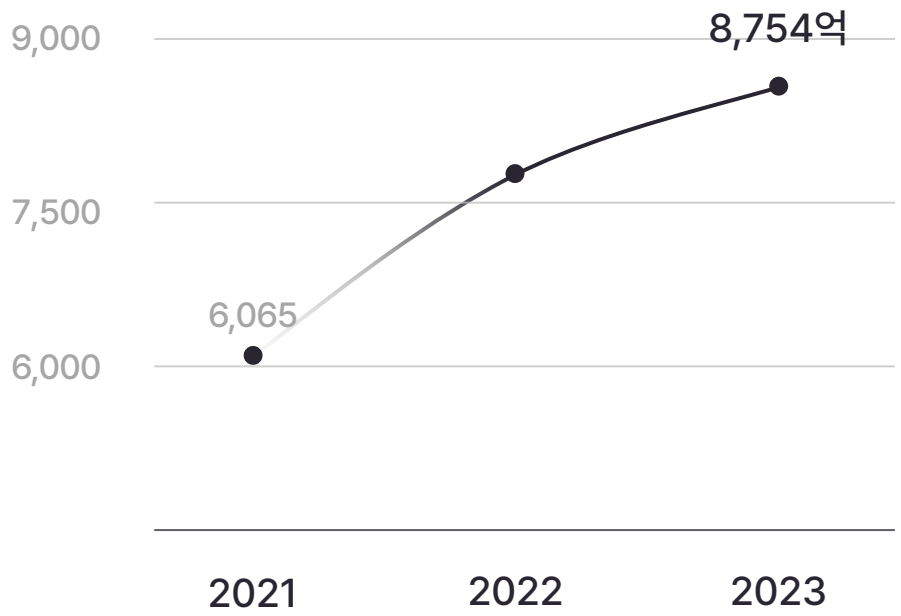


출처: 와디즈

간편결제 서비스 이용 추이

이용건수 (단위: 만 건) | 이용금액 (단위: 억원)

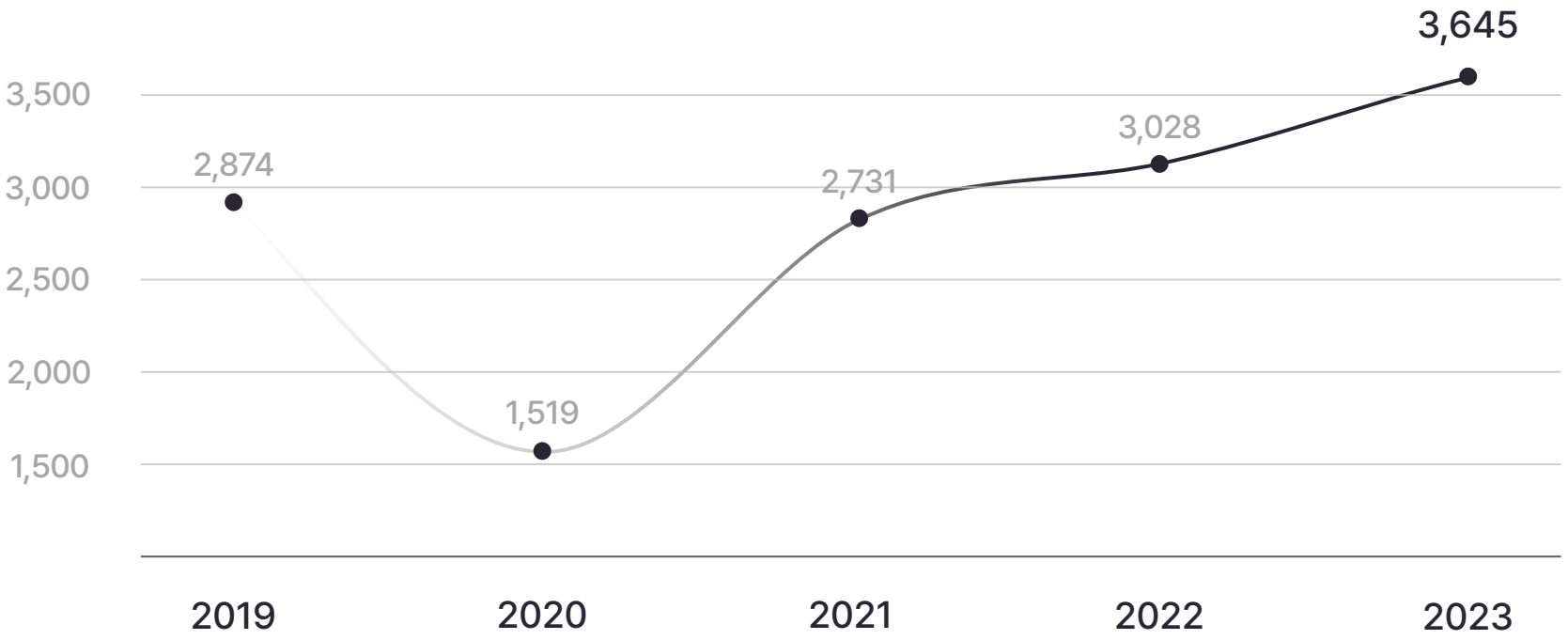




출처: 한국은행

사이버 범죄(피싱) 금융 피해 통계

인원수 (단위: 건)



출처: 경찰청

파주시, '큐싱(Qshing)' 피해 증가...
2024. 08. 26.

"검찰입니다"... 보이스피싱, 60대 여성 노린다
2024. 10. 24.

설문조사 위장 개인정보 탈취 주의해야...
2024. 09. 13.

일본 호텔 예약 사이트, 피싱 사이트로 연결
2024. 05. 12.

공공 자전거 타려고 QR 인식... 알고 보니 사기?
2023. 10. 15.

다시 돌아온 큐싱? QR 피싱, MZ 세대 노린다
2024. 01. 10.

피싱 피해금, 도박 사이트 이용한 자금 세탁
2024. 02. 21.

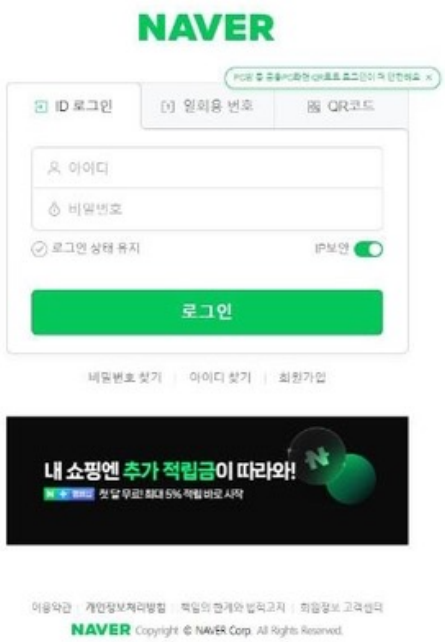
지속적인 개인정보 유출 및
금전적인 피해 사례 증가

최근 뉴스 기사만 수백 건
그러나, 현실적인 문제 해결 X

정상 사이트와 피싱 사이트 간
차이점 구분 불가

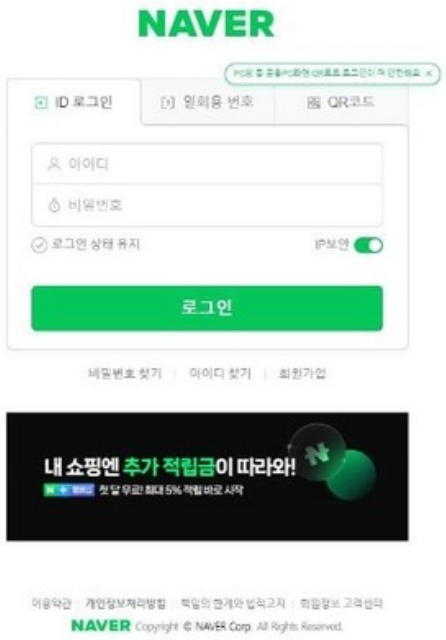
정상 사이트와 피싱 사이트 화면 비교

N사의 로그인 화면



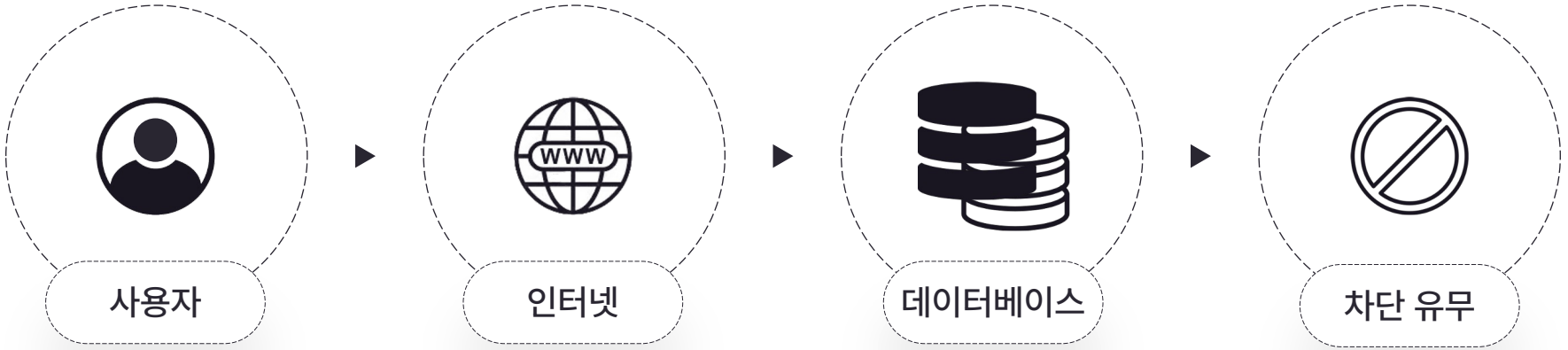
정상 사이트와 피싱 사이트 화면 비교

N사의 로그인 화면



시그니처 기반의 피싱 URL 탐지 과정

수동적인 탐지 과정 I



주기적인

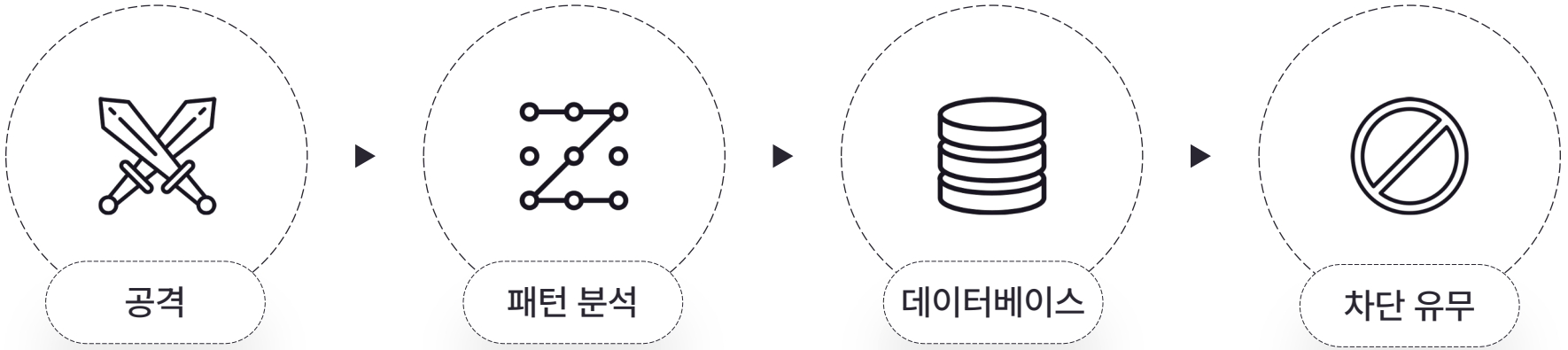
데이터베이스 업데이트 필수

피싱 사이트

도메인 주소 변경 시 차단 회피

규칙 기반의 피싱 URL 탐지 과정

수동적인 탐지 과정 II



새로운 공격 유형 시

신속하게 대응하기 어려움

Input 전처리 결과 차이점

분석 어려움

프로젝트 목표

과정

수집	크롤링	머신러닝	최종
칼럼 별 데이터 수집	Requests 방식 Selenium 방식	능동적인 차단 신속한 대처	차단율 97% 이상

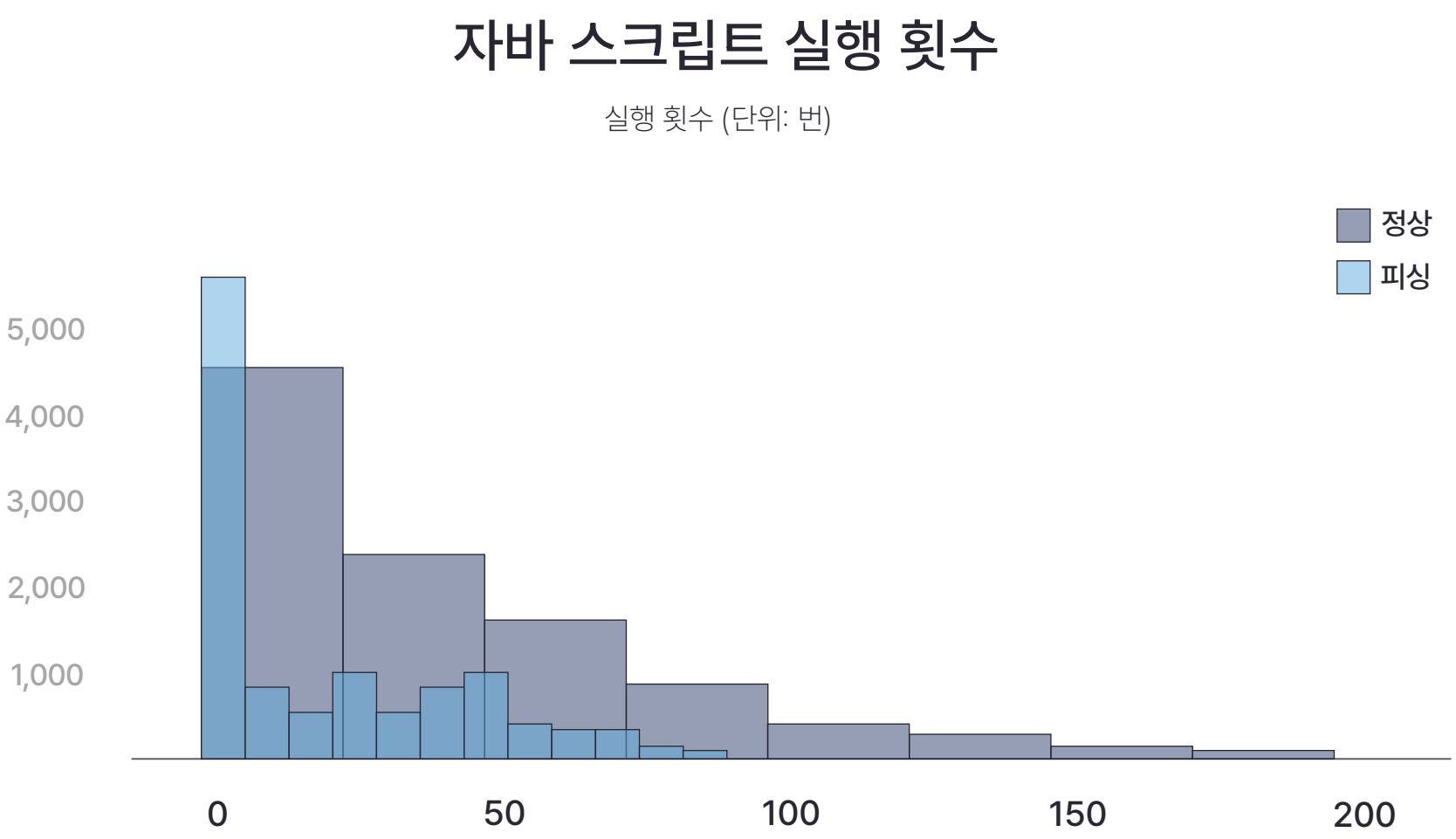
2

프로젝트 소개 및 진행 과정

칼럼 선정

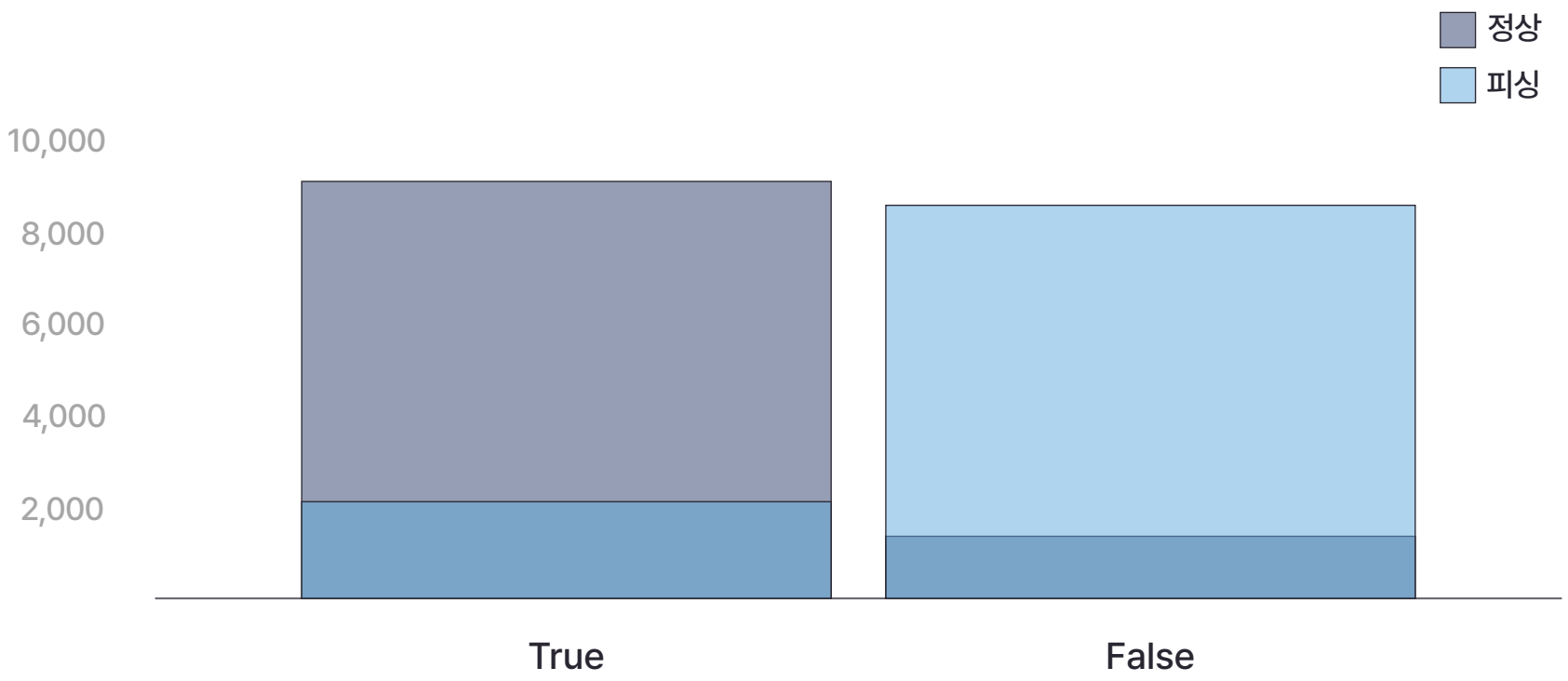
37가지

URL	IP	Country	User Country	Countries Match	Domain Age
Creation Date	Expiration Date	Registrant Name	Subdomain Count	Hidden iframe Count	Total Script Length
Obfuscated Script Length	Obfuscate Ratio	Is Obfuscated	Script Count	Meta Redirect	Window Location Redirect
AJAX Call Count	SSL Used	Cookie Access	Loading Time	Contents Size	Redirect Count
Lang	Redirection Count	External Domain Requests	Malicious File Download	Script Execution Count	Iframe Present
AJAX Calls Dynamic	Cookie Settings	Favicon	X Frame Option	SPF	TXT


자바 스크립트 실행 횟수
실행 횟수 (단위: 번)
정상
피싱
5,000
4,000
3,000
2,000
1,000
0
50
100
150
200

SPF 기능 여부

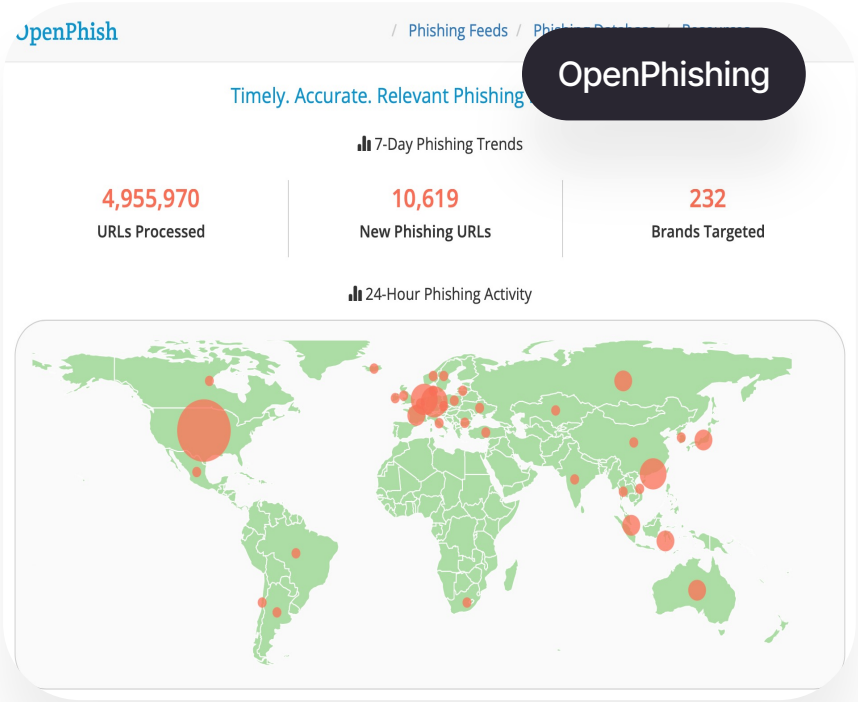
Bool형

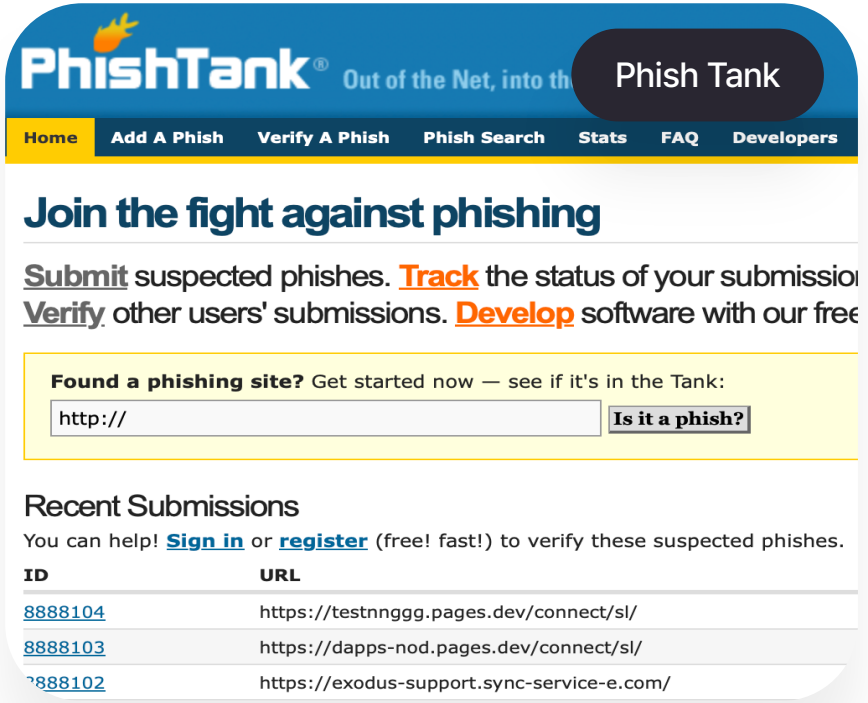



머신러닝
NumPy
XGBoost
TensorFlow
Random Forest
matplotlib
PyTorch
learn
Keras
LightGBM
pandas

1

피싱 사이트 URL 데이터셋 수집





2

각 URL 별 37가지의 칼럼 데이터셋 수집

3

하이퍼 파라미터 튜닝 설정

1

피싱 사이트 URL 데이터셋 수집

2

각 URL 별 37가지의 칼럼 데이터셋 수집

URL	IP	Country	
Creation Date	Expiration Date	Registrant Name	
Obfuscated Script Length	User Country	Countries Match	Domain Age
AJAX Call Count	Subdomain Count	Hidden iframe Count	Total Script Length
Lang	Script Count	Meta Redirect	Window Location Redirect
AJAX Calls Dynamic	Loading Time	Contents Size	Redirect Count
	Malicious File Download	Script Execution Count	Iframe Present
	X Frame Option	SPF	TXT

>

외부 링크 수	도메인 생성일	도메인 마감일	SPF
0	2022-02-13 15:58:13	2025-02-13 15:58:13	True
83	2023-08-30 10:09:23	2024-08-30 10:09:23	False
5	2021-07-12 16:05:39	2024-07-12 16:05:39	True
57	2023-08-21 00:10:21	2024-08-21 00:10:21	True
18	2005-02-23 17:07:11	2025-02-23 05:00:00	False
Favicorn	**Country**	**Server Country**	**Countries Match**
False	South Korea	South Korea	True
True	Japan	United States	False
IP	**Js_Len**	**Tld**	**Registrar**
213.166.69.26	7391	com	0
104.21.17.249	2074	world	NAMECHEAP INC

3

하이퍼 파라미터 튜닝 설정

1 피싱 사이트 URL 데이터셋 수집

2 각 URL 별 37가지의 칼럼 데이터셋 수집

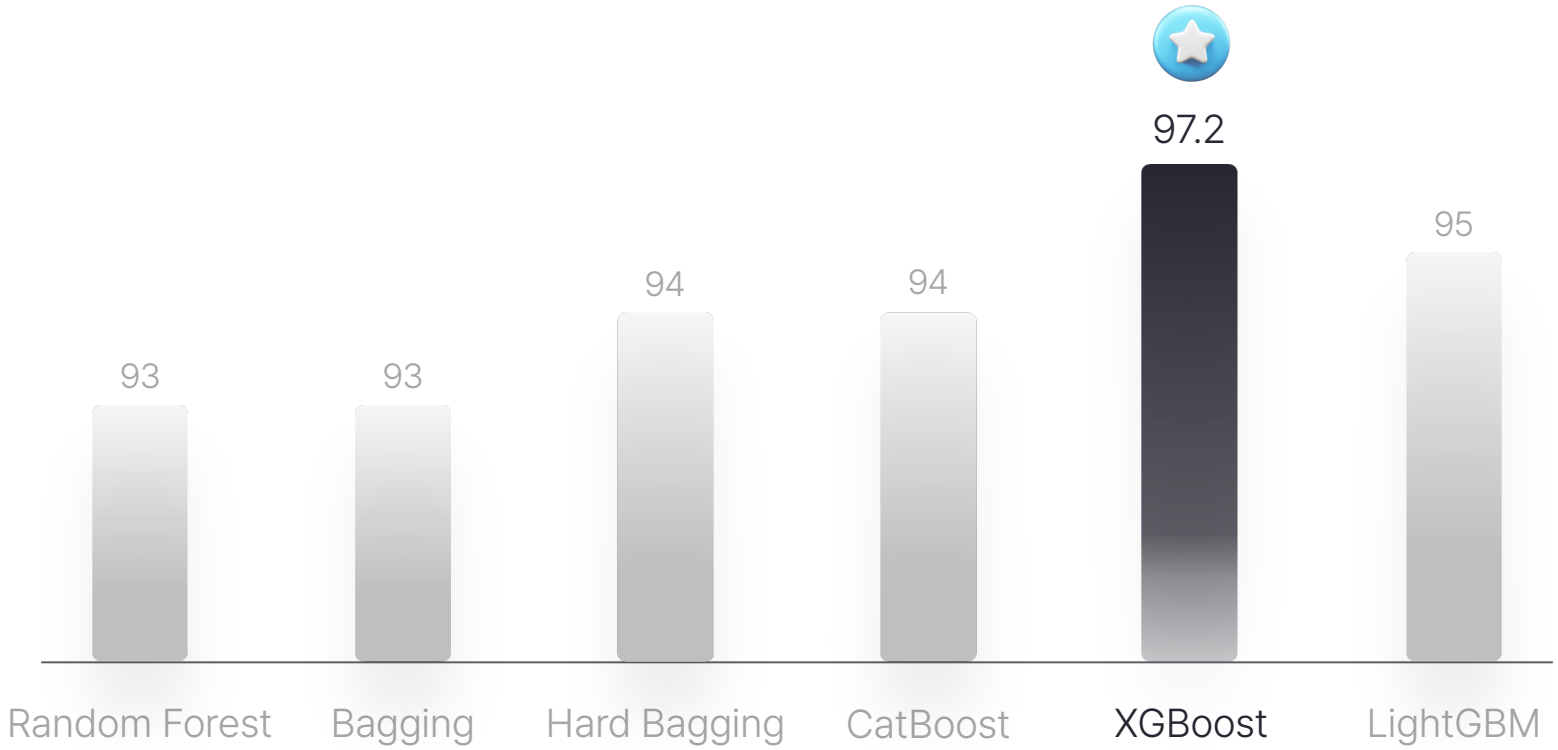
3 하이퍼 파라미터 튜닝 설정

```
# 가중치 비율에 따라 특정 열 조정
weighting_dict = {
    'script_execution_count': 1
    'Total Script Length': 10,
    'spf': 8.5,
    'txt': 8.5,
    'Script Count': 8.5,
    'AJAX Call Count': 7,
    'Redirect Count': 6,
    'Hidden Iframe Count': 5.5,
    'Domain Age (days)': 5,
    'img and texts': 4.5,
    'Content Size (bytes)': 3.5,
    'Subdomain Count': 3.5,
    'Loading Time (s)': 3.5,
    'iframe_present': 3.5,
```

가중치 비율

머신러닝 학습 정확도

정확도 (단위: %)



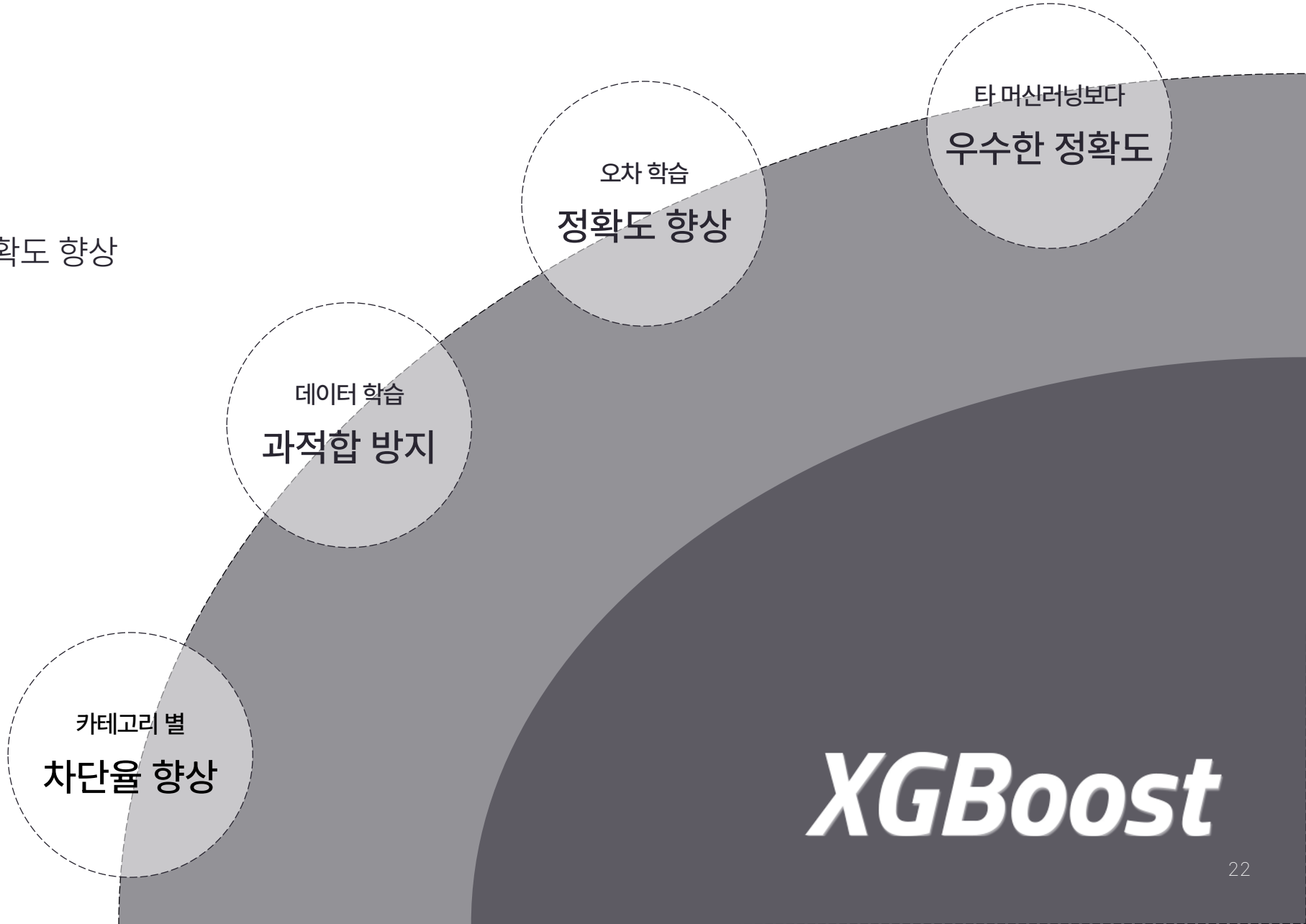
데이터 셋 크기: 19,479개

머신러닝, XGBoost

데이터 학습 과정에서 과적합 방지

오차 데이터만 반복 학습을 통해 정확도 향상

우수한 정확도 제공

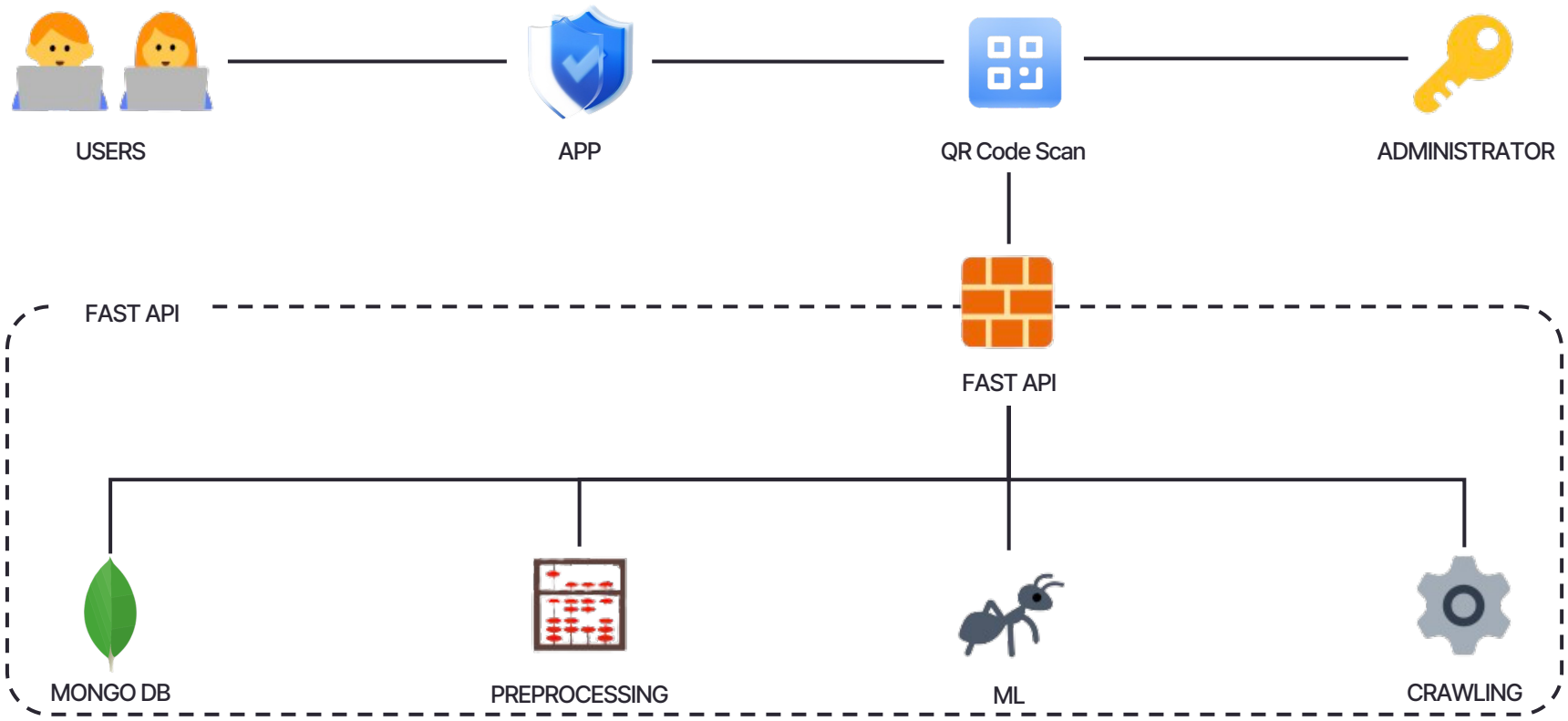


3

프로그램 소개

피싱 사이트 탐지 프로그램, 안심 QR

Flow Chart

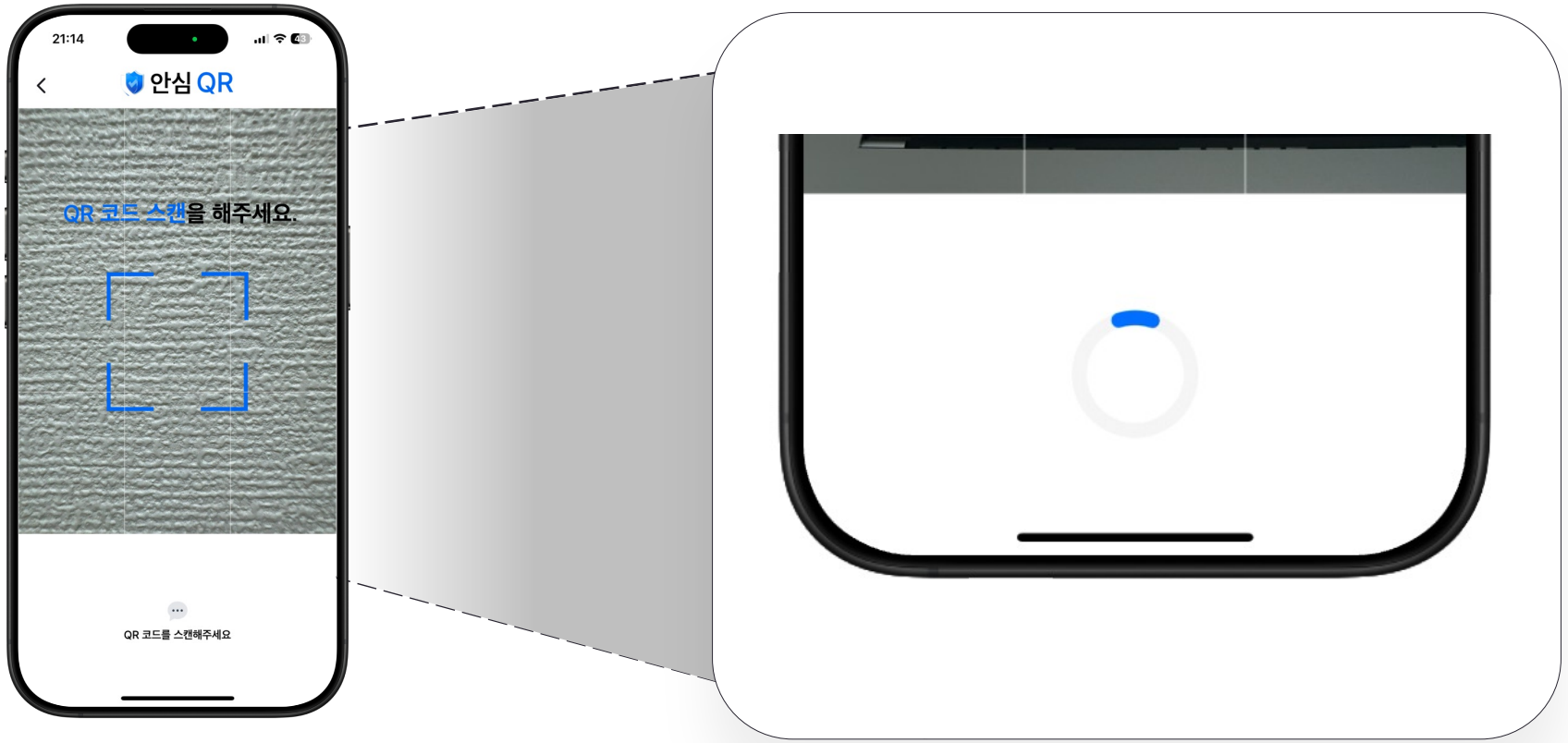


안심 QR UX/UI

메인화면

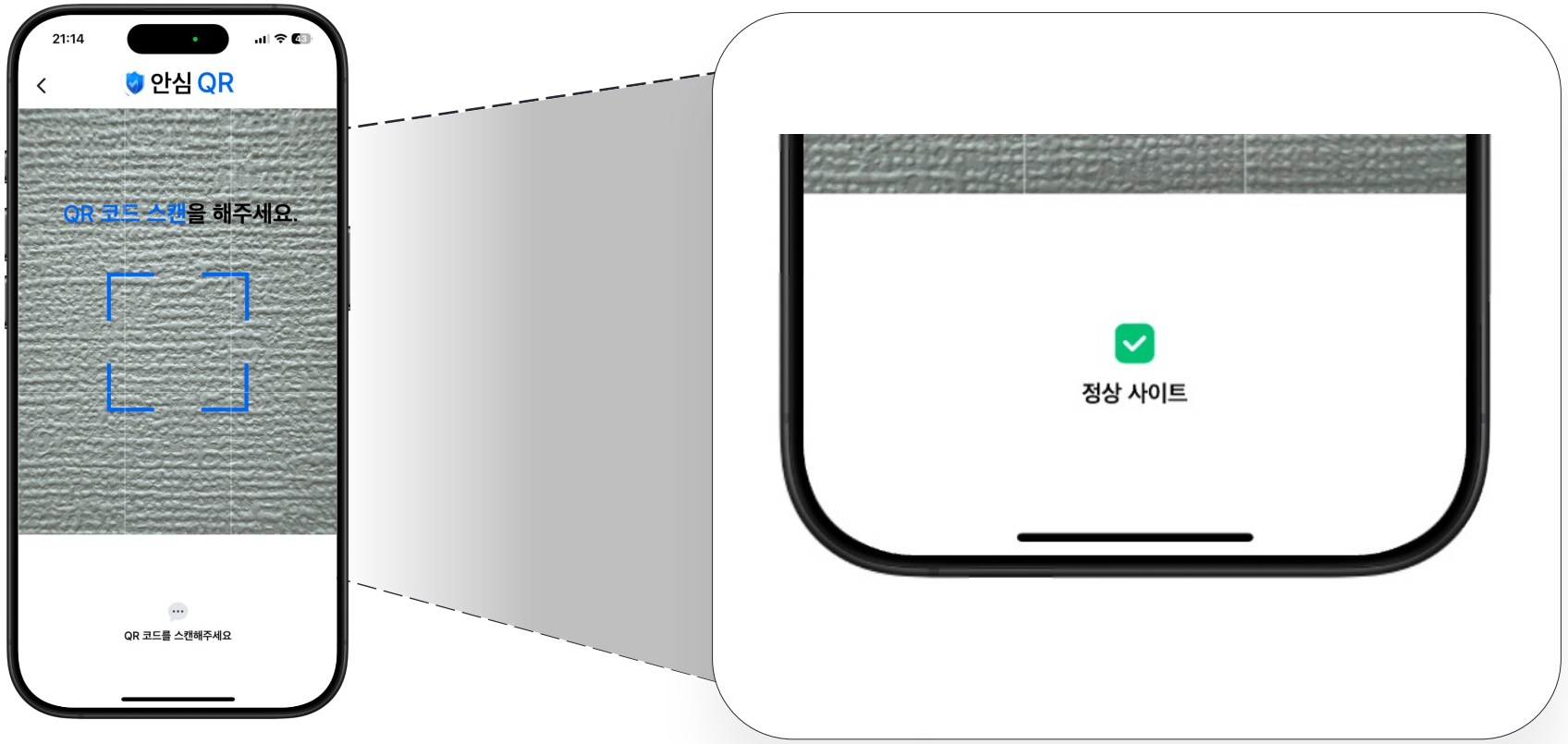
안심 QR UX/UI

분석 단계



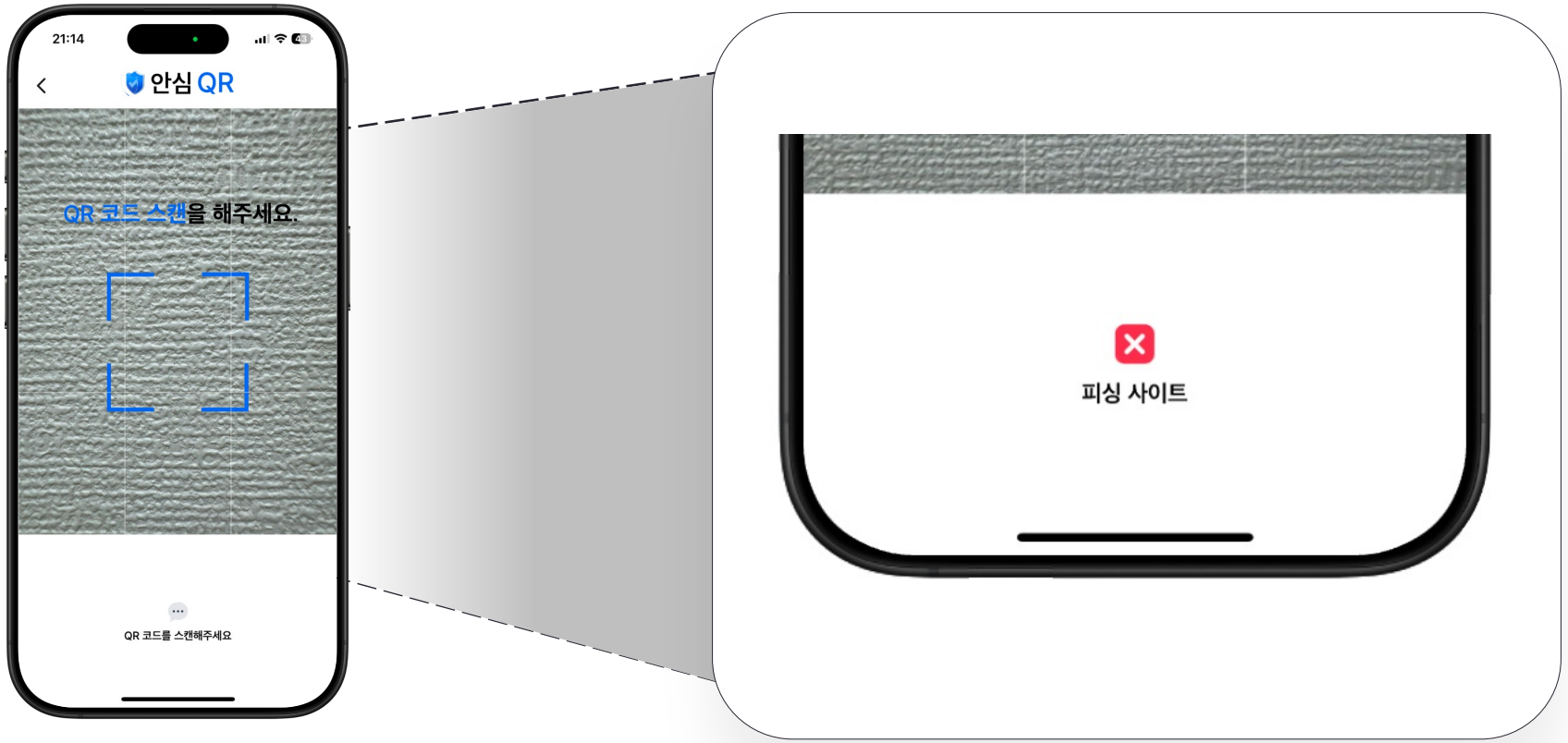
안심 QR UX/UI

분석 결과 (정상 사이트)



안심 QR UX/UI

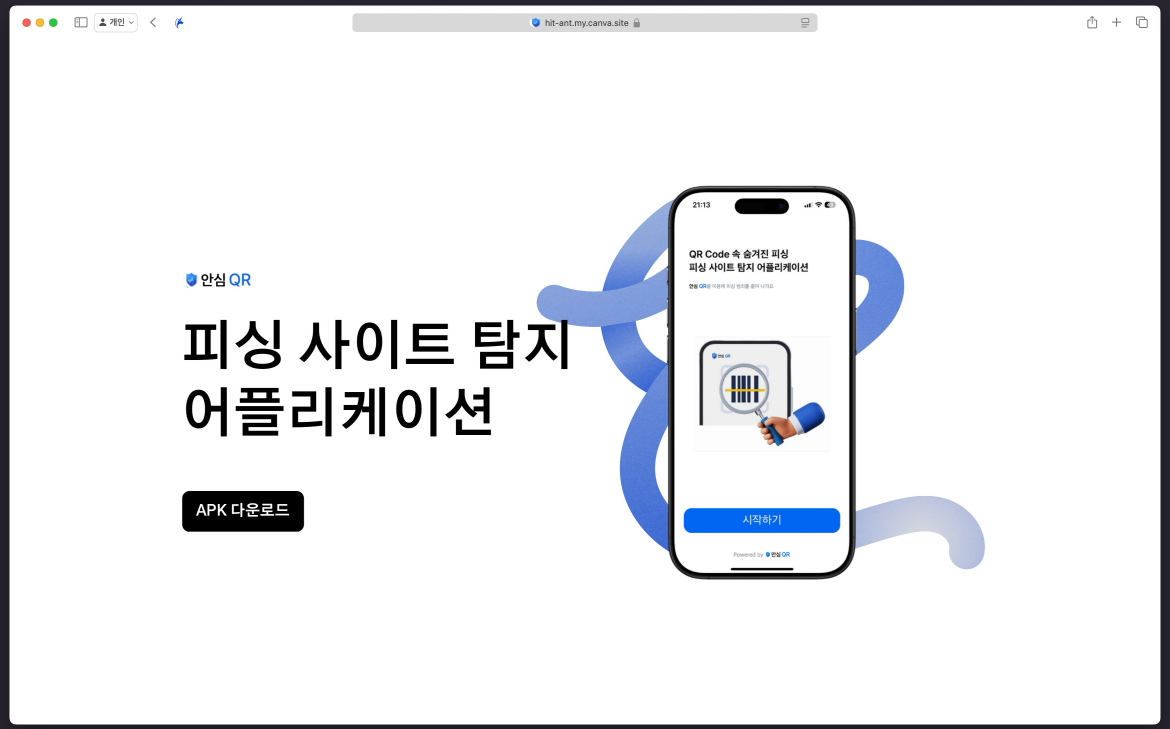
분석 결과 (피싱 사이트)



JOONGBU UNIVERSITY

안심 QR APK 설치 과정

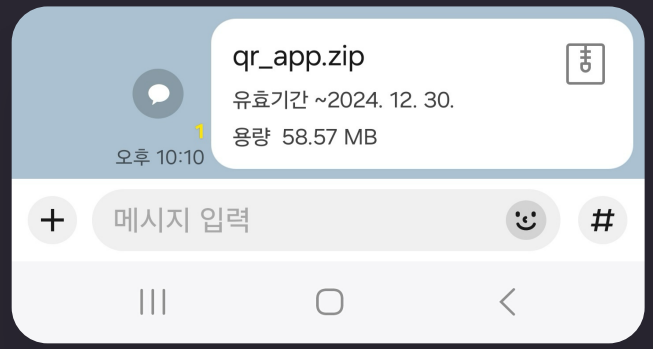
단계 00



안심 QR

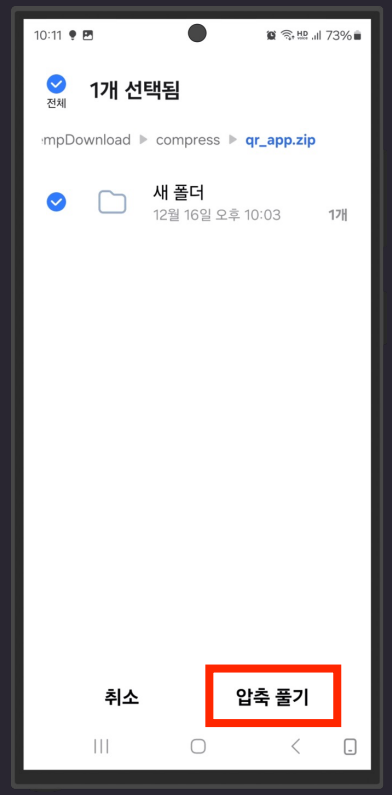
안심 QR APK 설치 과정

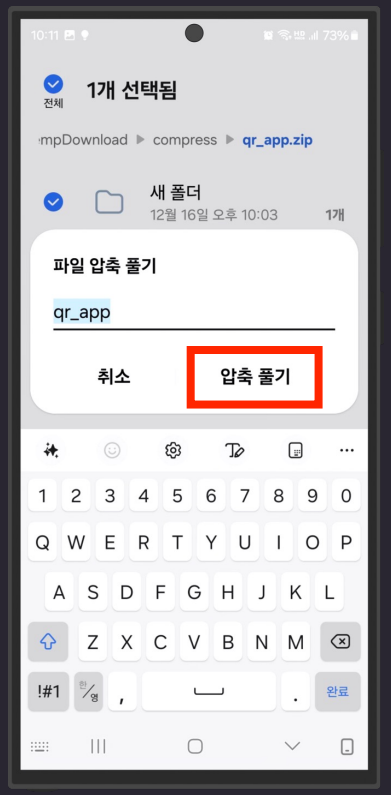
단계 01



안심 QR APK 설치 과정

단계 02

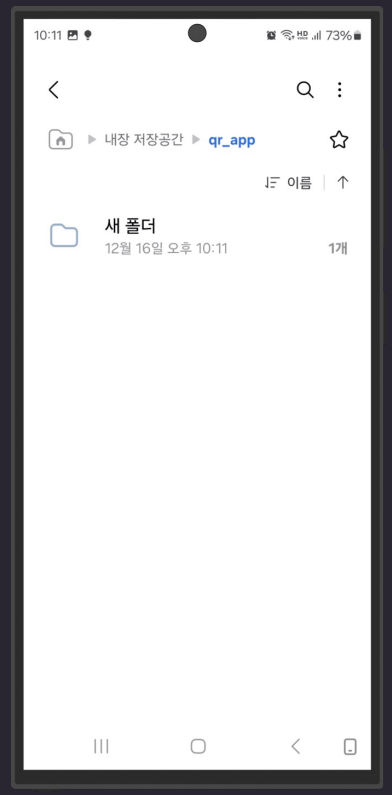


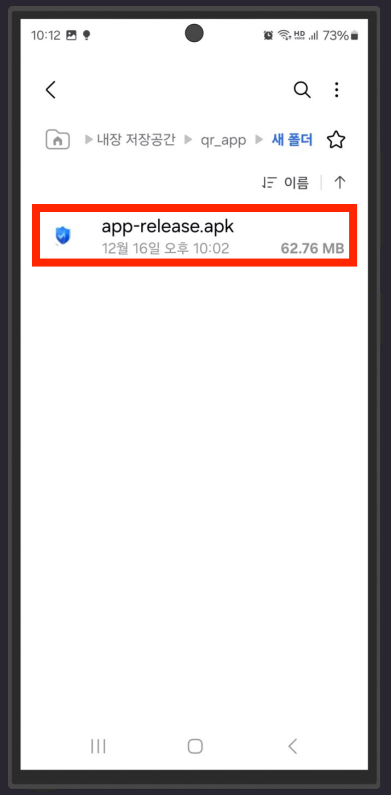


JOONGBU UNIVERSITY

안심 QR APK 설치 과정

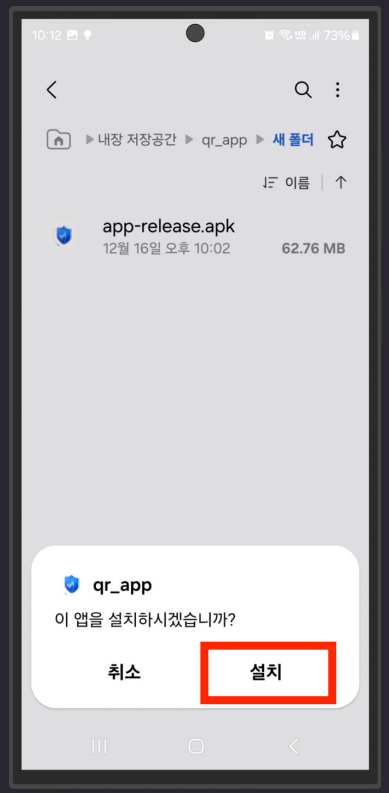
단계 03

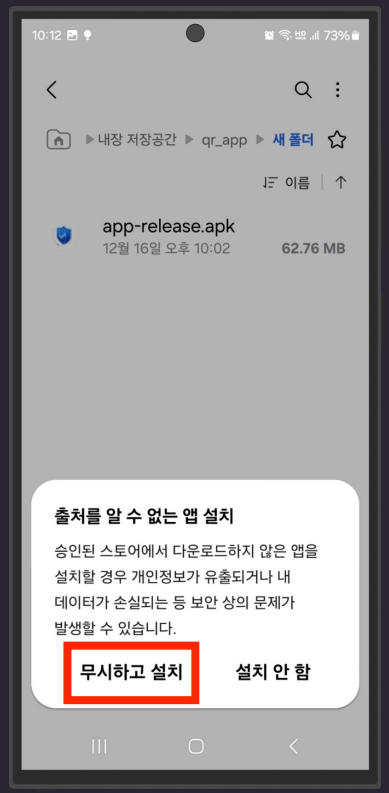


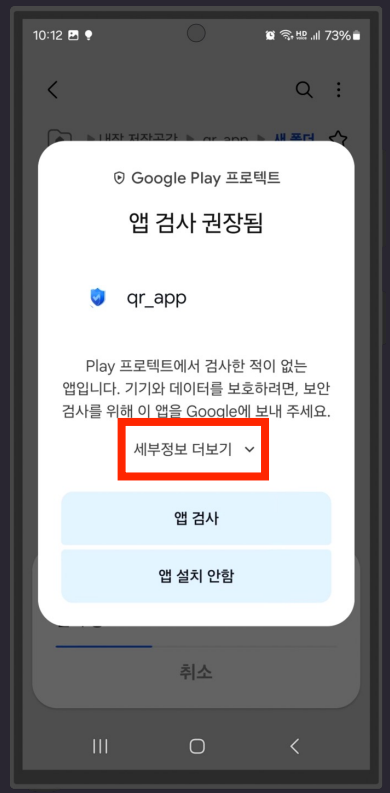


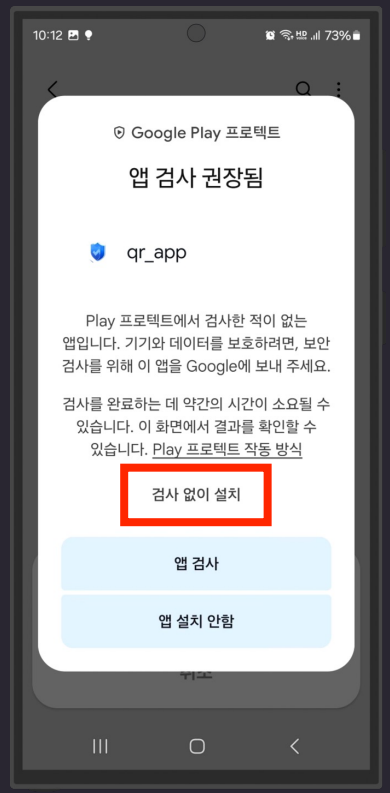
안심 QR APK 설치 과정

단계 04



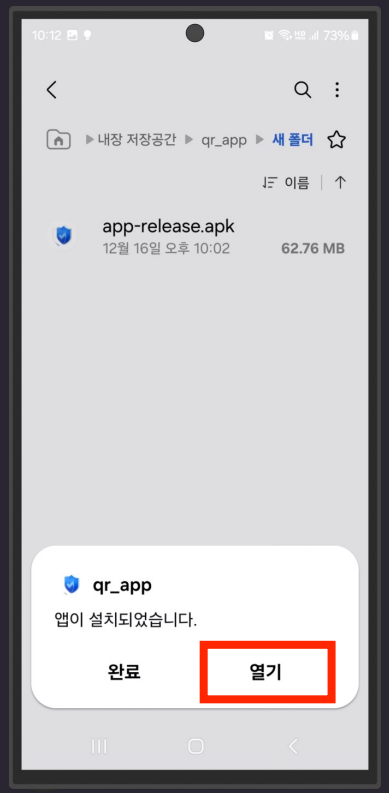






안심 QR APK 설치 과정

단계 05



JOONGBU UNIVERSITY

안심 QR APK 설치 과정

단계 06

JOONGBU UNIVERSITY

안심 QR APK 설치 과정

단계 07

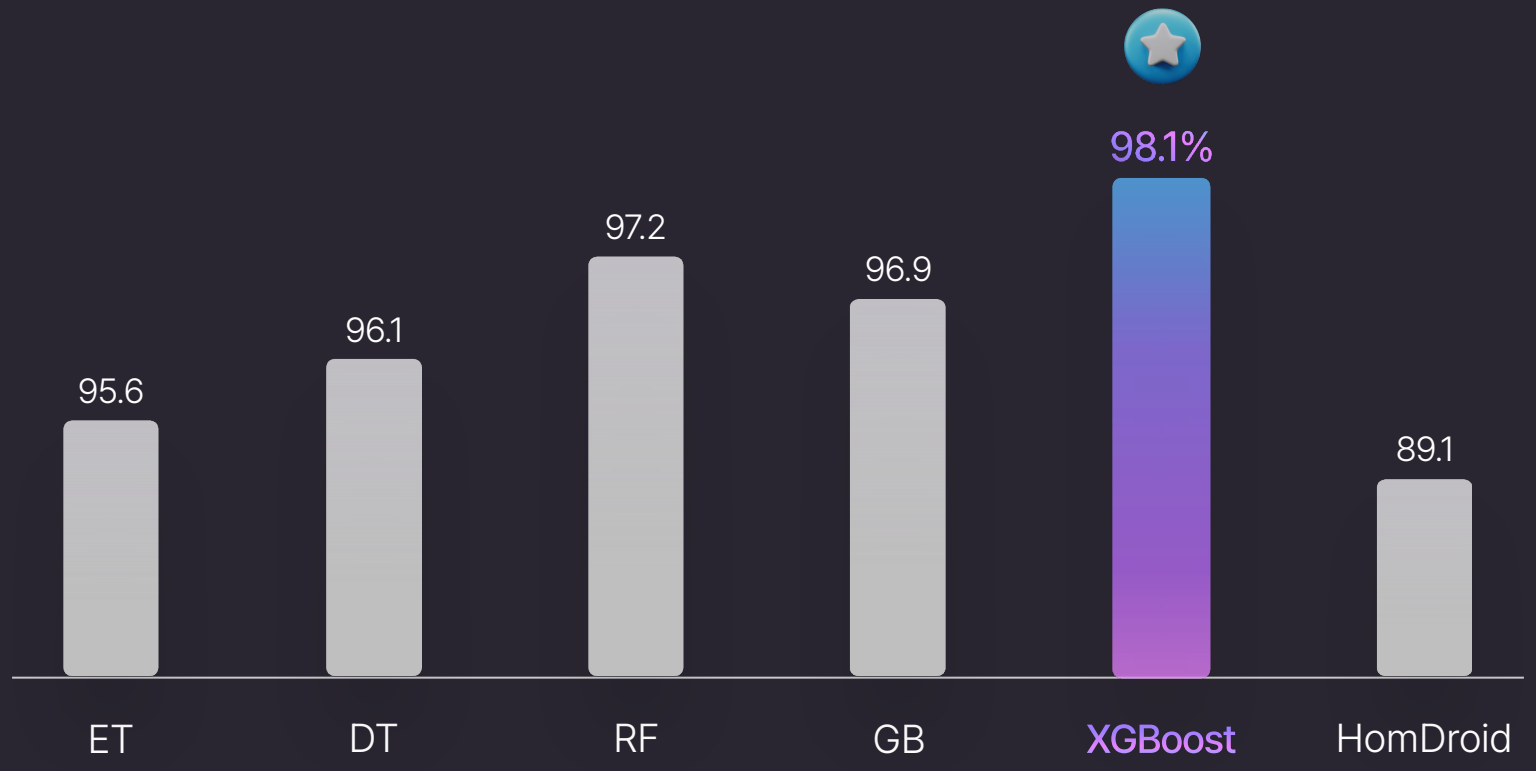
3. 제작한 피싱 사이트 탐지 어플리케이션, 안심 QR – 시연 영상

JOONGBU UNIVERSITY

JOONGBU UNIVERSITY

머신러닝 별 피싱 차단률

차단률 (단위: %)



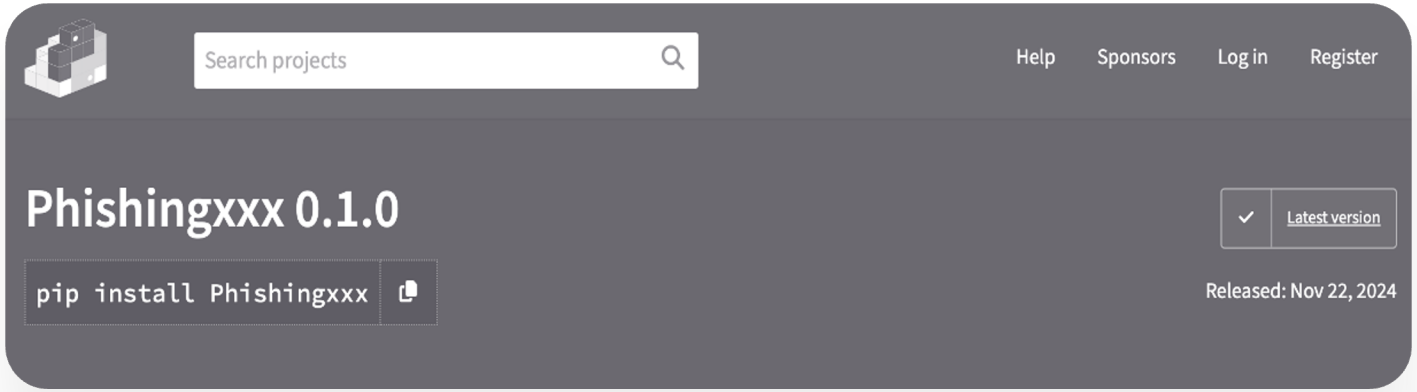
데이터 셋 크기: 20,417개

4

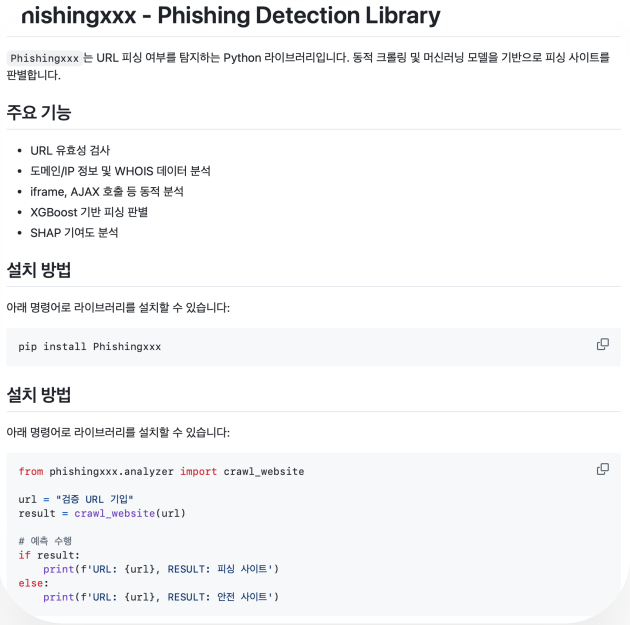
마무리

파이썬 라이브러리 배포

phishingxxx




```
bisanggu/
├── bisanggu/
│   ├── __init__.py
│   ├── analyzer.py
│   ├── models.py
│   └── utils.py
├── Phishing_model_02.pkl
├── setup.py
├── README.md
└── requirements.txt
```



nishingxxx - Phishing Detection Library

`Phishingxxx` 는 URL 피싱 여부를 탐지하는 Python 라이브러리입니다. 동적 크롤링 및 머신러닝 모델을 기반으로 피싱 사이트를 판별합니다.

주요 기능

- URL 유효성 검사
- 도메인/IP 정보 및 WHOIS 데이터 분석
- iframe, AJAX 호출 등 동적 분석
- XGBoost 기반 피싱 판별
- SHAP 기여도 분석

설치 방법

아래 명령어로 라이브러리를 설치할 수 있습니다:

```
pip install Phishingxxx
```

설치 방법

아래 명령어로 라이브러리를 설치할 수 있습니다:

```
from phishingxxx.analyzer import crawl_website

url = "검증 URL 기입"
result = crawl_website(url)

# 예측 수행
if result:
    print(f'URL: {url}, RESULT: 피싱 사이트')
else:
    print(f'URL: {url}, RESULT: 안전 사이트')
```

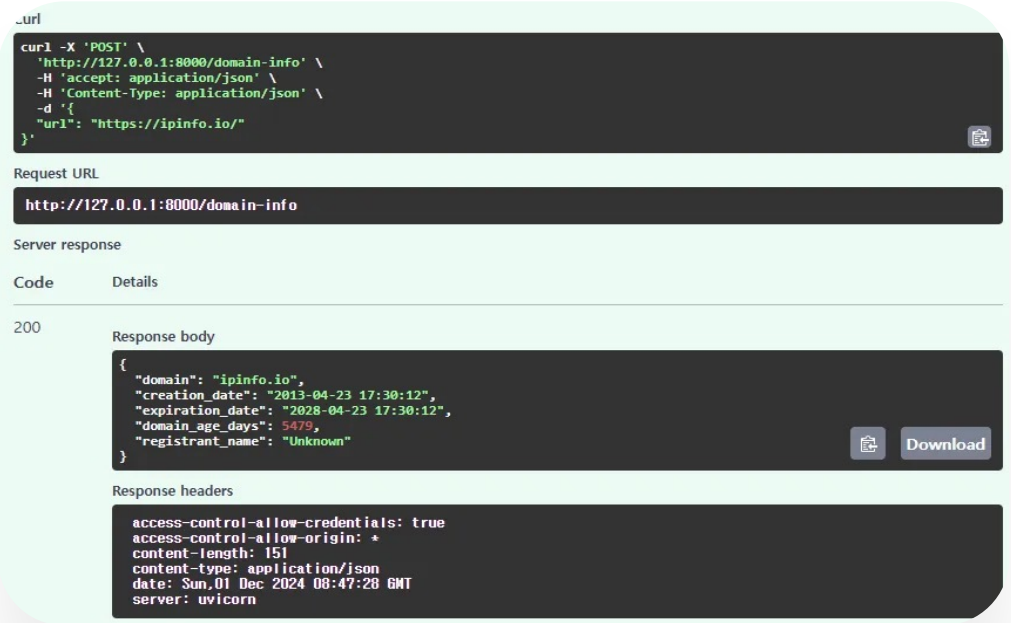
GitHub 링크

API 형식 배포 준비

API - /domain-info

```python
@app.post("/domain-info")
async def domain_info(request: URLRequest):
    domain = extract_domain_without_tld(request.url)
    try:
        # ml_models.py에 정의된 get_domain_age 호출
        creation_date, expiration_date, domain_age, registrant_name = get_domain_age(domain)

        return {
            "domain": domain,
            "creation_date": str(creation_date) if creation_date else "Not available",
            "expiration_date": str(expiration_date) if expiration_date else "Not available",
            "domain_age_days": domain_age if domain_age else "Not available",
            "registrant_name": registrant_name
        }
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Error retrieving domain info: {e}")
```



도메인 정보 조회 [도메인 생성일, 만료일, 나이, 등록자 정보]

API 형식 배포 준비

API - /check-obfuscation

```python
@app.post('/check-url')
async def check_url(request: URLRequest):
    # URL에서 도메인 추출 및 화이트/블랙리스트 비교
    is_whitelisted = is_url_in_collection(request.url, whitelist_collection)
    is_blacklisted = is_url_in_collection(request.url, blacklist_collection)

    # 데이터베이스에서 URL이 없을 경우 ML 검증을 호출
    if not is_whitelisted and not is_blacklisted:
        try:
            prediction = predict_url(request.url)
            if prediction == -1:
                raise HTTPException(status_code=400, detail="Prediction failed")
            print(prediction)
            return {
                "isWhitelisted": is_whitelisted,
                "isBlacklisted": is_blacklisted,
                "prediction": prediction,
                "message": "URL was not in database; ML verification performed."
            }
        except Exception as e:
            raise HTTPException(status_code=500, detail=f"Error during ML prediction: {str(e)}")

    # 데이터베이스에서 URL이 확인된 경우 결과 반환
    return {
        "isWhitelisted": is_whitelisted,
        "isBlacklisted": is_blacklisted,
        "prediction": None,
        "message": "URL found in database; no ML verification needed."
    }
```

1. 난독화 된 JavaScript 코드

```
{
    "script_content": "var _0x1234=function(_0x5678){return _0x5678+1;};"
}
```

2. 일반 JavaScript 코드

```
{
    "script_content": "function add(a, b) { return a + b; }"
}
```

3. 빈 JavaScript 코드

```
{
    "script_content": ""
}
```

4. 의도적으로 잘못 된 형식

```
{
    "script_contents": "var x = 10;"  // 'script_content' 키 오타
}
```

```
curl -X 'POST' \
  'http://127.0.0.1:8000/check-obfuscation' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "script_content": "var _0x1234=function(x){return x+1;};"
}'
```

Request URL

```
http://127.0.0.1:8000/check-obfuscation
```

Server response

Code	Details
200	Response body

```
{
  "script_content": "var _0x1234=function(x){return x+1;};",
  "is_obfuscated": true,
  "message": "The script is obfuscated"
}
```

Download

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: *
content-length: 116
content-type: application/json
date: Sun,01 Dec 2024 08:49:55 GMT
server: uvicorn
```

자바 스크립트 난독화 여부 판독

API 형식 배포 준비

API - /crawl-website

```python
app.post("/crawl-website")
async def crawl_and_predict(request: URLRequest):
    url = request.url

    # URL 유효성 검사
    if not url.startswith(("http://", "https://")):
        url = "http://" + url

    try:
        # 크롤링 데이터 가져오기
        crawling_data = crawl_website(url)

        if not crawling_data:
            raise HTTPException(status_code=400, detail="Failed to crawl the website")

        # 결과 반환
        return {
            "crawling_data": crawling_data
        }
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"An error occurred: {e}")
```


```
"crawling_data": {
  "URL": "https://pal.icepeng.com/",
  "IP Address": "172.66.47.60",
  "Country": "Canada",
  "User Country": "South Korea",
  "Countries Match": "No",
  "Domain Age (days)": 2922,
  "Creation Date": "2017-12-04T15:54:26",
  "Expiration Date": "2025-12-04T15:54:26",
  "Registrant Name": "On behalf of icepeng.com owner",
  "Subdomain Count": 1,
  "Hidden Iframe Count": 0,
  "Total Script Length": 320,
  "Obfuscated Script Length": 0,
  "Obfuscation Ratio": 0,
  "Is Obfuscated": false,
  "Script Count": 5,
  "Meta Redirect": 0,
  "Window Location Redirect": false,
  "AJAX Call Count": 0,
  "SSL Used": true,
  "Cookie Access": false,
  "Loading Time (s)": 0.48672938346862793,
  "Content Size (bytes)": 27754,
  "Redirect Count": 0,
  "Final URL": "https://pal.icepeng.com/",
  "redirection_count": 0,
  "external_domain_requests": 0,
  "malicious_file_downloads": 0,
  "script_execution_count": 8,
  "iframe_present": true,
  "ajax_calls_dynamic": 0,
  "cookie_settings": 0,
  "favicon": true,
  "x frame option": true,
  "spf": false,
  "txt": true,
  "lang": true,
  "img and texts": 94
  },
"prediction": "Safe site"
```


해당 URL에 대한 37가지의 칼럼 정보 크롤링

API 형식 배포 준비

API - /check-url

```
@app.post('/check-url')
async def check_url(request: URLRequest):
    # URL에서 도메인 추출 및 화이트/블랙리스트 비교
    is_whitelisted = is_url_in_collection(request.url, whitelist_collection)
    is_blacklisted = is_url_in_collection(request.url, blacklist_collection)

    # 데이터베이스에서 URL이 없을 경우 ML 검증을 호출
    if not is_whitelisted and not is_blacklisted:
        try:
            prediction = predict_url(request.url)
            if prediction == -1:
                raise HTTPException(status_code=400, detail="Prediction failed")
            print(prediction)
            return {
                "isWhitelisted": is_whitelisted,
                "isBlacklisted": is_blacklisted,
                "prediction": prediction,
                "message": "URL was not in database; ML verification performed."
            }
        except Exception as e:
            raise HTTPException(status_code=500, detail=f"Error during ML prediction: {str(e)}")

    # 데이터베이스에서 URL이 확인된 경우 결과 반환
    return {
        "isWhitelisted": is_whitelisted,
        "isBlacklisted": is_blacklisted,
        "prediction": None,
        "message": "URL found in database; no ML verification needed."
    }
```

curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/check-url' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "url": "https://pal.icepeng.com/"
}'
```

Request URL

```
http://127.0.0.1:8000/check-url
```

Server response

Code	Details
200	Response body

```
{
  "isWhitelisted": false,
  "isBlacklisted": false,
  "prediction": 0,
  "message": "URL was not in database; ML verification performed."
}
```

Download

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: *
content-length: 124
content-type: application/json
date: Sun,01 Dec 2024 11:07:55 GMT
server: uvicorn
```

해당 URL의 정상 / 피싱 여부 결과 반환

감사합니다.

JOONGBU UNIVERSITY