비지도학습 기반 악성 URL 탐지 시스템

팀 명 : 피싱 클리너
지 도 교 수 : 양환석 교수님
팀 장 : 정여진
팀 원 : 서장석
양승원
정채영

2024. 11.

중부대학교 정보보호학과

목차

1. 서론

1.1 연구배경

악성 URL 탐지는 사이버 보안 분야에서 중요한 문제로, 최근 AI 기술을 활용한 연구가 활발히 진행되고 있다. 피싱은 사용자로부터 개인정보를 훔치기 위해 악의적인 웹사이트를 이용하는 대표적인 사이버 공격으로, 그 수와 정교함이 지속적으로 증가하고 있다. 기존의 탐지 방법은 새로운 피싱 사이트를 실시간으로 탐지하는 데 한계가 있다. AI를 활용한 피싱 사이트 탐지 연구는 머신러닝과 딥러닝 기술을 통해 URL 패턴, 사이트 구조, 콘텐츠 등을 분석하여 기존 방식의 한계를 극복하고 실시간으로 피싱 사이트를 탐지하는 것을 목표로 한다. 특히, AI 모델은 대규모 데이터를 학습하여 이전에 알려지지 않은 새로운 피싱 공격에도 대응할 수 있어, 피싱 방지의 효과적인 대안으로 주목받고 있다.

<최근 4년 간 신고·차단된 피싱사이트 현황>

(단위 : 건)

구분	2016	2017	2018	2019.8월	합계
피싱사이트 신고·차단	4,286	10,469	9,522	7,063	31,340

※출처 : 과학기술정보통신부

[그림 1] 피싱사이트 근황

1.2 연구 필요성

AI를 활용한 탐지 기술은 대규모 데이터를 학습하여 URL 패턴, 웹사이트 콘텐츠의 특징 등을 자동으로 분석하고, 새로운 피싱 시도에 신속하게 대응할 수 있는 능력을 제공한다. 특히, AI 모델은 데이터 기반 학습을 통해 이전에 탐지되지 않은 피싱 공격까지 탐지할 수 있어 기존 방식보다 효율적이다. 이러한 이유로 피싱 사이트 탐지에 AI 기술을 적용한 연구는 사이버 보안의 필수적 요소로 자리 잡고 있으며, 실시간으로 진화하는 피싱 공격에 대처하기 위해 필수적이다.

1.3 연구 목적 및 주제선정

본 연구는 악성 URL에 의한 피해를 방지하기 위한 방법을 고민하였으며, 과거부터 최신까지 악성 URL들이 공통적으로 갖는 유의미한 특징을 조사하여 특징값을 선정하고, 머신러닝 기반 악성 URL 예측 시스템을 구현하는 데 중점을 두어 진행하였다. 특히, 비지도 학습을 활용하여 악성 URL과 정상 URL 간의 차이를 학습된 레이블 없이 자동으로 분류하는 것을 중요하게 생각하였다. 비지도 학습 알고리즘을 통해 데이터의 숨겨진 패턴을 분석하고, 악성 URL의 특징을 클러스터링 기법으로 탐지하여 새로운 공격에 대한

대응력을 높였다. 이를 통해 실시간으로 진화하는 악성 URL을 탐지하고, 수동으로 라벨링 하지 않은 데이터에서도 피싱 사이트를 효과적으로 식별할 수 있는 시스템을 개발하여 GUI에서 구현 가능하도록 하였다.

2. 관련 연구

2.1 Python

Phython은 웹 애플리케이션, 소프트웨어 개발, 데이터 과학, 기계 학습(ML)에 널리 사용되는 프로그래밍 언어이다. Python의 주요 특징으로는 직관적이고 간결한 문법이 있어 배우기 쉽고, 다양한 라이브러리와 프레임워크가 제공되어 복잡한 작업도 간편하게 처리할 수 있다는 점이 있다. 이러한 이유로 Python은 웹 개발, 데이터 분석, 인공지능, 자동화 등 여러 분야에서 널리 사용되며, 초보자부터 전문가까지 폭넓은 사용자층을 보유하고 있다.

2.2 K-means

K-Means는 비지도 학습 알고리즘으로, 데이터를 K개의 클러스터로 그룹화한다. 사용자가 클러스터 수(K)를 지정한 후, K개의 초기 중심점을 선택하고 각 데이터 포인트를 가장 가까운 중심에 할당하여 클러스터를 형성한다. 그런 다음, 각 클러스터의 데이터 포인트 평균을 계산해 중심을 업데이트하고, 클러스터 중심이 더 이상 변경되지 않거나 반복 횟수에 도달할 때까지 이 과정을 반복한다. K-Means는 간단하고 효율적이며 대규모 데이터셋에서도 빠르게 작동하지만, K 값을 미리 설정해야 하고 초기 중심 선택에 민감하여 주로 고객 세분화와 데이터 마이닝에 활용된다.

2.3 Gmm(Gaussian Mixture Model)

Gaussian Mixture Model (GMM)은 여러 개의 가우시안 분포(정규 분포)를 혼합하여 데이터의 분포를 모델링하는 확률적 모델이다. GMM은 비지도 학습 알고리즘으로, 데이터의 잠재적인 클러스터 구조를 발견하는 데 사용된다. 이 모델은 K개의 가우시안 분포로 구성되며, 각 가우시안은 데이터의 특정 클러스터를 나타낸다. 각 클러스터는 평균과 공분산 행렬로 정의되고, GMM은 각 가우시안 분포의 평균, 공분산, 혼합 비율(각 클러스터가 전체 데이터에서 차지하는 비율) 등의 파라미터를 학습한다. GMM은 Expectation-Maximization (EM) 알고리즘을 사용하여 이러한 파라미터를 최적화하고, 데이터 포인트를 각 클러스터에 확률적으로 할당하여 클러스터링을 수행한다. GMM은 클러스터의 형태가 구형이 아닐 때에도 유용하게 사용되며, 주로 데이터 마이닝, 이미지 처리 및 음성 인식 등 다양한 분야에서 활용된다.

2.4 MeanShift

MeanShift는 비지도 학습 클러스터링 알고리즘으로, 데이터의 밀도가 높은 지역을 찾아 클러스터를 형성하는 데 사용된다. 이 알고리즘은 각 데이터 포인트를 시작점으로 하여 주변 데이터 포인트의 밀도를 기반으로 평균을 계산하고, 이를 통해 해당 데이터 포인트를 이동한다. 이 과정을 반복하여 데이터 포인트가 밀도가 높은 지역으로 이동하게 되며, 결국 같은 밀도 지역에 속하는 데이터 포인트들이 클러스터를 형성한다. MeanShift는 클러스터의 수를 미리 지정할 필요가 없고, 비구조적인 데이터에 대해 효과적으로 클러스터링을 수행할 수 있는 장점이 있어, 이미지 처리와 패턴 인식 등 다양한 분야에서 활용된다.

2.5 AgglomerativeCluster

Agglomerative Clustering은 비지도 학습 클러스터링 알고리즘으로, 계층적 클러스터링 방법 중 하나이다. 이 알고리즘은 처음에 각 데이터 포인트를 개별 클러스터로 시작하여, 가장 유사한 클러스터를 반복적으로 병합해 나가는 방식으로 작동한다. 먼저 클러스터 간의 거리 또는 유사성을 계산한 후, 가장 가까운 두 클러스터를 찾아 병합한다. 이 과정을 클러스터가 하나로 합쳐질 때까지 반복하며, 최종적으로는 계층적인 클러스터 구조를 형성하게 되며 데이터 분석과 패턴 인식 분야에서 널리 활용된다.

2.6 RandomForest

Random Forest는 지도학습 알고리즘 중 결정 트리 기반의 앙상블 학습 기법으로, 분류와 회귀 문제에 모두 사용된다. 이 알고리즘은 여러 개의 결정 트리를 생성하고, 각 트리의 예측을 종합하여 최종 결과를 도출한다. 전체 데이터셋에서 랜덤하게 샘플을 선택하고, 각 트리를 생성할 때 일부 특성만을 사용하여 과적합을 방지한다. 최종 예측은 분류의 경우 각 트리의 투표 결과를 기반으로 하고, 회귀의 경우 예측값의 평균을 사용한다. Random Forest는 높은 정확도와 견고성을 제공하여 다양한 분야에서 널리 활용된다.

2.7 LogisticRegression

Logistic Regression은 지도학습 알고리즘 중 이진 분류 문제를 해결하기 위한 통계적 모델로, 입력 변수와 이진 결과(0 또는 1) 간의 관계를 모델링 한다. 이 알고리즘은 입력 특성의 선형 조합을 사용하여 특정 사건의 발생 확률을 예측하고, 시그모이드 함수(sigmoid function)를 통해 0과 1 사이의 값으로 변환한다. 모델 학습은 최대 우도 추정(Maximum Likelihood Estimation) 방법을 통해 이루어지며, 이진 크로스 엔트로피(binary cross-entropy)를 손실 함수로 사용하여 예측값과 실제값 간의 차이를 최소화한다. Logistic Regression은 다양한 분야에서 이진 분류에 널리 활용한다.

2.8 Softmax

각 데이터 포인트가 특정 클러스터에 속할 확률을 계산하는 데 사용되는 함수이다. 클

러스터링 모델에서 URL의 특성 벡터와 클러스터 중심 간의 거리를 측정하여, 이 거리를 음수로 변환한 후 Softmax 함수를 적용한다. 이 과정에서 가까운 클러스터일수록 높은 점수를 받아 각 클러스터에 속할 확률이 높아진다. 결과적으로 각 클러스터에 대한 확률은 0과 1 사이의 값으로 나타나며, 모든 확률의 합은 1이 된다. 이를 통해 URL의 피싱 가능성을 판단하고, 각 클러스터에 대한 신뢰도를 제공할 수 있다.

3. 본론

3.1 특징값 추출

특징값 추출을 하는 이유는 머신러닝 모델이 URL을 기반으로 악성 여부를 예측할 수 있도록 필요한 정보(특징)를 제공하기 위함이다. URL의 다양한 특징값을 추출하면, 모델이 단순한 문자열이나 도메인 정보를 넘어 URL의 패턴, 구조, 보안 상태 등을 학습할 수 있다. 이를 통해 악성 URL의 일반적인 특성과 패턴을 찾아내어, 안전한 URL과 악성 URL을 더 정확하게 구분할 수 있게 된다. 즉, 특징값 추출은 모델 성능을 향상시키고 예측 정확도를 높이는 핵심 과정이다. 학습 과정에서 특징값들은 시중에 있는 논문을 참고하여 적용 가능한 15개의 특징값을 최종 선발하였으며 그 종류는 다음과 같다.

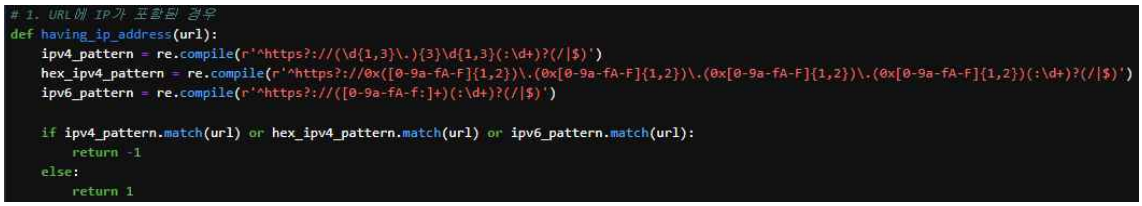
CATEGORY	FEATURES	NUMBER
URL 구조적 특징	URL 길이, URL 단축 서비스, 서브도메인 사용, 접두사 및 접미사('-'), ('//') 포함 여부, ('@') 포함 여부, URL favicon 로드 여부	8개
도메인 정보 및 보안요소	DNS 레코드 존재 여부, 도메인 등록 기간, 도메인 나이, HTTPS 사용 여부, 비표준 포트 사용 여부	5개
웹 페이지 행동 기반	마우스 오른쪽 클릭 비활성화 여부, URL 리디렉션 횟수	2개

[표 1] 특징값 추출

추출 과정은 Python에서 작업하였으며 도메인과 같은 정보는 urllib 라이브러리를 활용하여 추출했다. csv 라이브러리를 사용하여 출력된 특징값을 저장하였으며 모델학습을 위한 sklearn 라이브러리 등의 다양한 라이브러리를 사용하여 작업했다. 일반적인 머신러닝을 활용한 악성 URL탐지에서 벗어나고자 비지도학습을 사용했다. 비지도학습을 악성 URL 탐지에 활용하면 레이블이 없는 데이터를 사용할 수 있어 레이블링 작업의 부담을 줄일 수 있다. 또한, 알려지지 않은 새로운 악성 URL 패턴을 발견할 가능성이 있으며, 지속적으로 데이터를 학습하여 시스템이 확장성과 유연성을 가질 수 있다. 이로써 새로운 데이터에 즉시 적응할 수 있어 악성 URL의 진화에도 효과적으로 대응할 수 있

다. 다음은 15가지의 특징값을 추출하기 위한 함수 코드를 설명한다.

3.3.1 Having_ip_address



```python
# 1. URL에 IP가 포함된 경우
def having_ip_address(url):
    ipv4_pattern = re.compile(r'^https?://(\d{1,3}\.){3}\d{1,3}(:\d+)?(/|$)')
    hex_ipv4_pattern = re.compile(r'^https?://0x([0-9a-fA-F]{1,2})\.(0x[0-9a-fA-F]{1,2})\.(0x[0-9a-fA-F]{1,2})\.(0x[0-9a-fA-F]{1,2})(:\d+)?(/|$)')
    ipv6_pattern = re.compile(r'^https?://([0-9a-fA-f:]+)(:\d+)?(/|$)')

    if ipv4_pattern.match(url) or hex_ipv4_pattern.match(url) or ipv6_pattern.match(url):
        return -1
    else:
        return 1
```

[그림 2] having_ip_address 함수

URL에 IP 주소가 포함되어 있는지를 확인하는 having_ip_address 함수이다. 이 함수는 세 가지 정규 표현식을 사용하여 IPv4, 16진수 IPv4, 그리고 IPv6 주소 형식을 검사한다. URL이 이들 중 하나와 일치하면 -1(피싱)을 반환하고, 일치하지 않으면 1(정상)을 반환하여 IP 주소의 포함 여부를 나타낸다.

3.3.2 Url_length

```python
# 2. 긴 URL 사용
def url_length(url):
    if len(url) < 54:
        return 1
    elif len(url) >= 54 and len(url) <= 75:
        return 0
    else:
        return -1
```

[그림 3] url_length 함수

URL의 길이에 따라 값을 반환하는 url_length 함수이다. URL의 길이가 54자 미만이면 1(정상)을, 54자 이상 75자 이하이면 0(의심)을, 75자 이상이면 -1(피싱)을 반환해서 URL의 길이에 따른 평가를 나타낸다.

3.3.3 Shortening_service

```python
# 3. URL 단축 서비스 사용
shorteners = ['bit.ly', 'kl.am', 'cli.gs', 'bc.vc', 'po.st', 'v.gd', 'bkite.com', 'shorl.com', 'scrnch.me', 'to.ly', 'adf.ly', 'x.co',
              'iurl.com', 'ad.vu', 'migre.me', 'su.pr', 'smallurl.co', 'cutt.us', 'filoops.info', 'shor7.com', 'yfrog.com', 'tinyurl.com',
              'u.to', 'ow.ly', 'ff.im', 'rubyurl.com', 'r2me.com', 'post.ly', 'twitthis.com', 'buzurl.com', 'cur.lv', 'tr.im', 'bl.lnk',
              'tiny.cc', 'lnkd.in', 'q.gs', 'is.gd', 'hurl.ws', 'om.ly', 'prettylinkpro.com', 'qr.net', 'qr.ae', 'snipurl.com', 'ity.im',
              't.co', 'db.tt', 'link.zip.net', 'doiop.com', 'url4.eu', 'poprl.com', 'tweez.me', 'short.ie', 'me2.do', 'bit.do', 'shorte.st',
              'go2l.ink', 'yourls.org', 'wp.me', 'goo.gl', 'j.mp', 'twurl.nl', 'snipr.com', 'shortto.com', 'vzturl.com', 'u.bb', 'shorturl.at',
              'han.gl', 'wo.gl', 'wa.gl']

def shortening_service(url):
    domain = urlparse(url).netloc
    if domain in shorteners:
        return -1
    return 1
```

[그림 4] shortening_service 함수

URL이 단축 서비스에서 생성된 것인지 확인하는 shortening_service 함수이다. 먼저, urlparse를 사용해 URL에서 도메인을 추출한 다음, 미리 정의된 단축 서비스 목록인 shorteners와 비교하게 된다. 도메인이 단축 서비스 목록에 있다면 -1(피싱)을 반환하고, 그렇지 않으면 1(정상)을 반환하여 URL의 단축 서비스에 해당 여부를 나타낸다.

3.3.4 Having_at_symbol

```python
# 4. URL에 '@' 기호 포함
def having_at_symbol(url):
    if '@' in url or '%40' in url:
        return -1
    else:
        return 1
```

[그림 5] having_at_symbol 함수

URL에 '@' 기호가 포함되어 있는지를 확인하는 having_at_symbol 함수이다. 함수는 URL에 '@' 기호가 있거나 URL 인코딩된 형태인 '%40'가 포함되어 있다면 -1(피싱)을 반환하고, 그렇지 않으면 1(정상)을 반환하여 '@' 기호의 포함 여부를 나타낸다.

3.3.5 Double_slash_redirecting

```python
# 5. '//'를 사용한 리다이렉션
def double_slash_redirecting(url):
    if '//' in url[7:]:
        return -1
    else:
        return 1
```

[그림 6] double_slash_redirecting 함수

URL에서 '//'를 사용한 리다이렉션을 확인하는 double_slash_redirecting 함수이다. 함수는 URL의 7번째 문자 이후에 '//'가 포함되어 있는지를 검사한다. 만약 해당 부분에 '//'가 있다면 -1(피싱)을 반환하여 리다이렉션의 가능성을 나타내고, 그렇지 않으면 1(정상)을 반환하여 리다이렉션이 없음을 나타낸다.

3.3.6 Prefix_suffix

```python
# 6. URL의 접두사/접미사에 '-'가 포함된 경우
def prefix_suffix(url):
    # ip형식일 경우
    if having_ip_address(url) == -1:
        return 0
    try:
        domain = get_tld(url, as_object=True)

        if '-' in domain.domain:
            return -1
        elif '-' in domain.subdomain:
            return -1
        else:
            return 1
    except tld.exceptions.TldDomainNotFound:
        return -1
```

[그림 7] prefix_suffix 함수

URL의 접두사 또는 접미사에 '-'가 포함되어 있는지를 확인하는 prefix_suffix 함수이다. 먼저, having_ip_address 함수를 호출하여 URL이 IP 형식인지 검사한다. 만약 IP 형식이라면 0(의심)을 반환한다. 그렇지 않는 경우에는 get_tld 함수를 사용해 URL에서

최상위 도메인을 추출하고, 도메인이나 서브도메인에 '-'가 포함되어 있는지를 확인한다. 만약 '-'가 포함되어 있다면 -1(피싱)을 반환하고, 포함되어 있지 않으면 1(정상)을 반환한다. 또한, URL이 잘못된 형식인 경우 예외를 처리하여 -1(피싱)을 반환한다.

3.3.7 Having_sub_domain

```python
# 7. 서브도메인이 있는지 확인
def having_sub_domain(url):
    if having_ip_address(url) == -1:
        return 0
    # www. 제거
    if 'www.' in url[:12]:
        url = url.replace('www.', '')

    try:
        domain = get_tld(url, as_object=True)

        if domain.subdomain == '': # 서브 도메인이 없을 경우
            return 1
        dot = domain.subdomain.count('.')
        if dot == 0: # 서브 도메인이 있는 경우 .이 없으면 의심 1개 이상이면 피싱
            return 0
        else:
            return -1
    except tld.exceptions.TldDomainNotFound:
        return -1
```

[그림 8] having_sub_domain 함수

URL에 서브도메인이 있는지를 확인하는 having_sub_domain 함수이다. 먼저, URL이 IP 형식인지 확인하여 IP인 경우 0(의심)을 반환한다. 그 후, URL의 처음 12자에서 'www.'를 제거한 뒤, get_tld 함수를 사용해 최상위 도메인을 추출한다. 서브도메인이 없는 경우 1(정상)을 반환하고, 서브도메인에 '.'가 없는 경우 의심스러운 상황으로 0(의심)을 반환한다. 서브도메인에 '.'가 하나 이상 있는 경우에는 -1을 반환하여 피싱 가능성을 나타낸다. 또한, 잘못된 URL 형식인 경우 예외를 처리하여 -1(피싱) 을 반환한다.

3.3.8 Domain_registration_length

```python
# 8. 도메인 등록 기간
def get_total_date(url):
    domain = whois.whois(url)
    expiration_date = domain.expiration_date
    if isinstance(expiration_date, list):
        expiration_date = expiration_date[0]
    updated_date = domain.updated_date
    if isinstance(updated_date, list):
        updated_date = updated_date[0]
    if expiration_date is None or updated_date is None:
        return None
    total_date = (expiration_date - updated_date).days
    return total_date
```

[그림 9] domain_registration_length 함수

주어진 URL의 도메인 등록 기간을 평가하는 domain_registration_length 함수이다. 이 함수는 get_total_date를 호출해 등록 기간을 일수로 계산한 후, 만약 None이 반환되면 0(의심)을 반환한다. 등록 기간이 365일 이하이면 -1(피싱)을 반환하고, 365일을

초과하면 1(정상)을 반환하여 안정적인 도메인으로 평가한다. 또한, 도메인 정보를 가져오는 과정에서 오류가 발생하면 -1(피싱)을 반환하고, 다른 예외가 발생하면 0(의심)을 반환하여 에러 처리를 수행한다.

3.3.9 Favicon

```python
# 9. Favicon
def favicon(url):
    try:
        response = requests.get(url)
        response.raise_for_status()
        soup = BeautifulSoup(response.text, 'html.parser')
        parsed_url = urlparse(url)
        base_domain = parsed_url.netloc
        favicon_tags = soup.find_all('link', rel=lambda value: value and 'icon' in value.lower())
        if not favicon_tags:
            return 1
        for tag in favicon_tags:
            href = tag.get('href')
            if href:
                favicon_domain = urlparse(href).netloc
                if favicon_domain == '':
                    return 1
                else:
                    return -1
        return 1
    except:
        return -1
```

[그림 10] favicon 함수

URL의 파비콘(favicon)을 확인하는 favicon 함수이다. 주어진 URL에 HTTP GET 요청을 보내고, 응답이 성공적이면 HTML을 파싱(parsing)한다. 〈link〉 태그에서 rel 속성이 'icon'인 모든 태그를 찾고, 파비콘 태그가 없으면 1(정상)을 반환한다. 각 파비콘 태그의 href 속성을 확인해, 비어있거나 도메인이 없으면 1(정상)을 반환하고, 파비콘 도메인이 기본 도메인과 다르면 -1(피싱)을 반환한다. 요청 중 오류가 발생하면 -1(피싱)을 반환하여 에러를 처리한다.

3.3.10 Port

```python
# 10. 비표준 포트 사용
def remove_schemes(url):
    if 'http://' in url:
        url = url.replace('http://', '')
    elif 'https://' in url:
        url = url.replace('https://', '')
    else:
        return url
    return url

def port(url):
    domain = remove_schemes(url)
    try:
        ip = socket.gethostbyname(domain)
    except:
        return -1
    socket.setdefaulttimeout(2)
    ports = [80, 21, 22, 23, 445, 1433, 1521, 3306, 3389]
    for port in ports:
        s = socket.socket()
        if port == 80:
            try:
                s.connect((ip, port))
                s.close()
            except:
                return -1
        else:
            try:
                s.connect((ip, port))
                s.close()
                return -1
            except:
                pass
    return 1
```

[그림 11] port 함수

URL에서 비표준 포트 사용 여부를 확인하는 port 함수이다. URL에서 'http://' 또는 'https://'를 제거하고, 도메인을 IP 주소로 변환시킨다. IP 변환에 실패하면 -1(피싱)을 반환한다. 주요 포트(80, 21, 22 등)에 대해 연결을 시도하며, 포트 80에 성공하면 -1(피싱)을 반환하고, 나머지 포트에서 연결에 성공하면 -1(피싱)을 반환한다. 모든 포트에서 연결이 실패하면 1(정상)을 반환하여 비표준 포트가 사용되지 않음을 나타낸다.

3.3.11 Https_Token

```python
# 11. 도메인 부분에 HTTPS 존재 여부
def https_token(url):
    try:
        domain = url.split('/')[2]
    except IndexError:
        return 1
    if "https" in domain:
        return -1
    else:
        return 1
```

[그림 12] https_token 함수

URL의 도메인 부분에 "https"가 포함되어 있는지를 확인하는 https_token 함수이다.

URL을 '/'로 분리해 도메인을 추출하고, 도메인에 "https"가 있으면 -1(피싱)을, 없으면 1(정상)을 반환한다. 만약 도메인을 찾지 못하면 1(정상)을 반환하여 안전한 상태로 처리한다.

3.3.12 Age_of_domain

```python
# 12. 도메인 수명(6개월 미만)
def age_of_domain(url):
    try:
        domain_info = whois.whois(url)
        expiration_date = domain_info.expiration_date
        if isinstance(expiration_date, list):
            expiration_date = expiration_date[0]
        current_date = datetime.now()
        remain_date = expiration_date - current_date
        if remain_date >= timedelta(days=182):
            return 1
        else:
            return -1
    except:
        return 0
```

[그림 13] age_of_domain 함수

주어진 URL의 도메인 수명을 확인하는 age_of_domain 함수이다. whois 라이브러리를 사용해 도메인 정보를 가져오고, 만료일을 확인한 후 남은 날짜를 계산한다. 만약 남은 날짜가 182일 이상이면 1(정상)을 반환하고, 그렇지 않으면 -1(피싱)을 반환한다. 오류가 발생하면 0(의심)을 반환한다.

3.3.13 Dns_record

```python
# 13. DNS 기록(유/무)
def dns_record(url):
    try:
        whois.whois(url)
    except:
        return -1
    return 1
```

[그림 14] dns_record 함수

주어진 URL에 대한 DNS 기록의 존재 여부를 확인하는 dns_record 함수이다. whois 라이브러리를 사용해 도메인 정보를 가져오고, 만약 DNS 기록을 가져오는 데 오류가 발생하면 -1(피싱)을 반환한다. DNS 기록이 존재하면 1(정상)을 반환한다.

3.3.14 Count_redirection

```python
# 14. 리다이렉션 횟수(피싱 최대 4번)
def count_redirection(url):
    try:
        count = 0
        res = requests.head(url, allow_redirects=True, timeout=5)
        for resp in res.history:
            if resp.status_code in [301, 302]:
                count += 1
        if count <= 1:
            return 1
        elif 2 <= count < 4:
            return 0
        else:
            return -1
    except:
        return 0
```

[그림 15] count_redirection 함수

주어진 URL의 리다이렉션 횟수를 세주는 기능을 하는 count_redirection 함수이다. requests 라이브러리를 사용해 URL에 요청을 보내고, 응답의 리다이렉션 히스토리를 확인한다. 리다이렉션이 1회 이하이면 1(정상)을, 2회에서 3회 사이면 0(의심)을, 4회 이상이면 -1(피싱)을 반환한다. 오류가 발생하면 0(의심)을 반환한다.

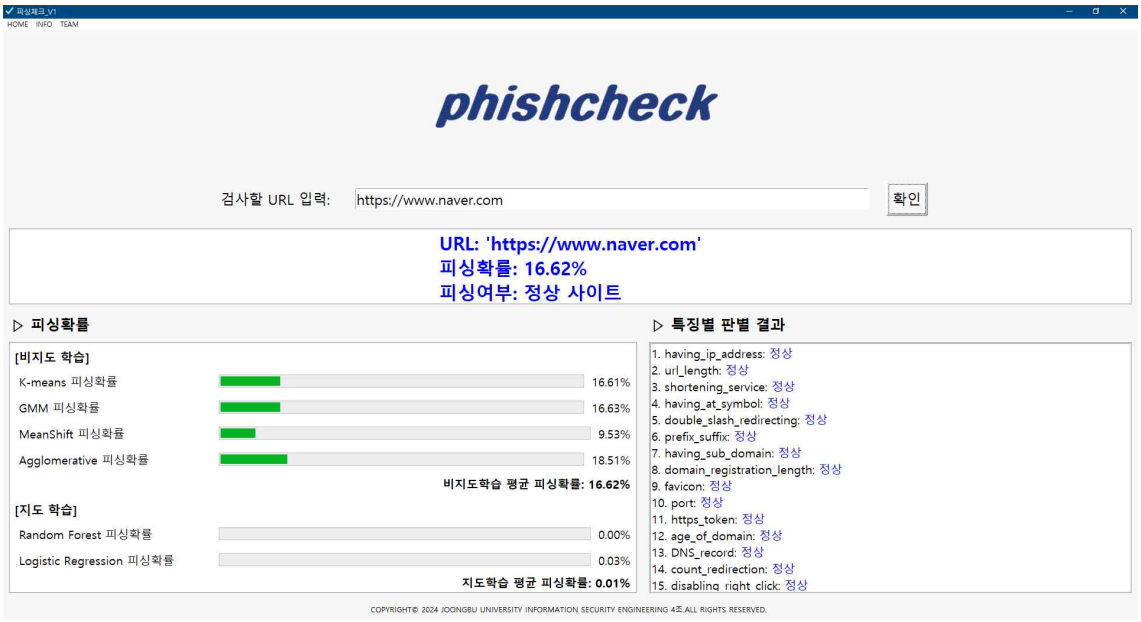
3.3.15 Disabling_right_click

```python
# 15. 우클릭 금지
def disabling_right_click(url):
    try:
        headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'
        }
        res = requests.get(url, timeout=5, headers=headers)
        if 'event.button==2' in res.text:
            return -1
        else:
            return 1
    except:
        return 0
```

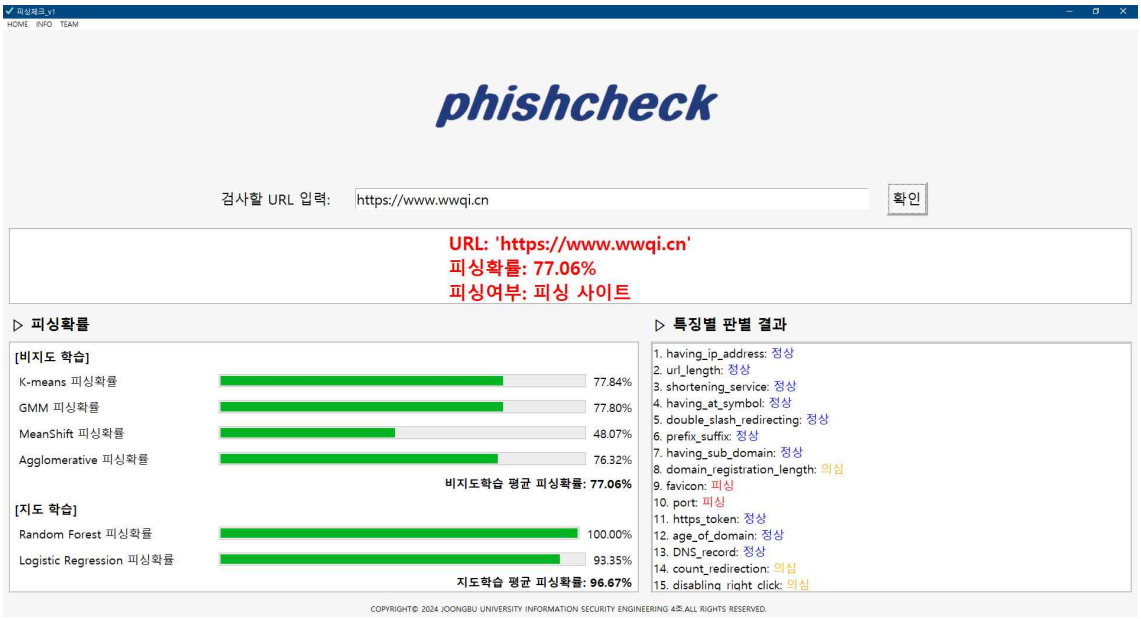
[그림 16] disabling_right_click 함수

주어진 URL에서 우클릭 방지를 확인하는 기능을 하는 disabling_right_click 함수이다. requests 라이브러리를 사용해 해당 URL에 HTTP GET 요청을 보내고, 사용자 에이전트 헤더를 설정하여 웹사이트의 응답을 받는다. 응답 본문에 event.button==2라는 문자열이 포함되어 있으면 우클릭이 금지된 것으로 간주하고 -1(피싱)을 반환한다. 그렇지 않으면 1(정상)을 반환한다. 요청 중 오류가 발생하면 0(의심)을 반환한다.

3.2 GUI



[그림 17] GUI - 정상사이트 판별



[그림 18] GUI - 피싱사이트 판별

4. 분석

4.1 활용결과 및 성능

본 프로젝트는 파이썬(Python) 환경에서 비지도학습 알고리즘으로 악성URL 판별을 위해 훈련과정에서 정상데이터 15,000개, 피싱데이터 7,000개 총 22,000개의 데이터로 각각의 알고리즘을 학습시켰다. 4가지의 비지도학습 알고리즘 K-Means, Gmm, MeanShift, AgglomerativeCluster의 학습률은 각각 91.72% / 91.67% / 88.26% / 88.37% 로 각각 나타났다. 본 프로젝트의 데이터를 학습한 결과 80% 이상의 학습률을 가졌기 때문에, 모델이 데이터에서 신뢰할 수 있는 패턴을 학습했음을 인지할 수 있었다. 또한, 모델이 훈련 과정에서 과적합 되지 않았으며, 각각의 알고리즘이 새로운 데이터에 대해 일반화할 수 있는 능력을 갖추었다고 생각하였다. 또한 프로젝트의 진행이 적절한지 확인하기 위하여 지도학습 알고리즘 중 RandomForest와 LogisticRegression을 가져와서 비지도학습과 같은 데이터로 학습 시켰을 때 학습률이 96.91% / 95.36%로 본 학습데이터의 무결성을 확인하였다.

Algorithm	DataSet	learning rate
K-Means	정상:15,000&피싱:7,000	91.72%
Gmm	정상:15,000&피싱:7,000	91.67%
MeanShift	정상:15,000&피싱:7,000	88.26%
AgglomerativeCluster	정상:15,000&피싱:7,000	88.37%

[표 2] 비지도학습 알고리즘 학습률

Algorithm	DataSet	learning rate
RandomForest	정상:15,000&피싱:7,000	96.91%
LogisticRegression	정상:15,000&피싱:7,000	95.36%

[표 3] 지도학습 알고리즘 학습률

URL의 피싱 확률은 Softmax 함수를 사용하여, URL의 특정 벡터와 클러스터 중심 간의 거리를 측정해 계산된다. 이때 지도 학습을 제외한 4가지 비지도 학습 알고리즘을 기반으로, 입력된 URL의 피싱률에 따라 피싱 확률을 산출한다. 각 알고리즘의 극단적인 편향을 줄이기 위해 최고값과 최저값을 제외한 두 개의 확률 평균을 사용했다. 피싱 확률의 임계값은 균형 잡힌 탐지율을 위해 70%로 설정되었다. 이 값은 피싱 탐지 모델에서 흔히 사용하는 오탐(false positive)과 미탐(false negative)을 고려한 것이다. 따라서 피싱 확률이 70% 미만이면 정상 사이트로, 70% 이상이면 피싱 사이트로 분류된다.

4.2 추후 보완사항

훈련데이터 추가 : 훈련데이터셋의 정상사이트와 피싱사이트의 비율을 기존 이외의 데이터를 추가하여 각 알고리즘의 학습률을 향상시킨다.

GUI UI 및 UX 개선 : 사용자의 편리성을 위하여 사용자의 인터페이스와 사용자 경험을 개선한다.

5. 결론

5.1 결론

비지도학습 기반 악성 URL 탐지 시스템은 기존의 지도학습 방식에서 벗어나, 라벨이 없는 데이터를 활용하여 잠재적인 피싱 및 악성 URL을 탐지하는 효과적인 방법이다. 이 접근 방식은 URL 데이터를 미리 정의된 범주 없이 분류하고, 새로운 유형의 위협을 자동으로 식별할 수 있는 능력을 갖추고 있어, 빠르게 진화하는 피싱 기법에 적응할 수 있다. 결론적으로, 비지도학습 모델(K-Means, GMM, MeanShift, Agglomerative Cluster 등)은 대량의 URL 데이터를 분석하고 잠재적인 악성 패턴을 찾아내는 데 탁월한 성능을 발휘한다. 특히, 이 시스템은 피싱과 정상 사이트 간의 경계가 불분명하거나 새로운 형태의 공격이 등장했을 때 효과적으로 대응할 수 있다. 또한, 비지도학습 기반 탐지 시스템은 라벨링된 데이터가 부족한 환경에서도 높은 효율성을 보여, 다양한 위협 상황에 적합한 솔루션을 제공한다.

5.2 기대효과

비지도학습 기반 악성 URL 탐지 시스템은 여러 가지 기대효과를 제공한다. 우선, 사전에 라벨링된 데이터가 필요하지 않기 때문에 새로운 유형의 피싱 및 악성 URL 탐지에 매우 효과적이다. 이는 빠르게 진화하는 사이버 위협 환경에서 알려지지 않은 패턴도 쉽게 식별할 수 있게 하며, 그 결과로 신종 악성 URL이나 피싱 사이트로부터의 보안 취약성을 줄이는 데 큰 역할을 할 수 있다. 또한, 라벨링 과정이 필요 없기 때문에 라벨링 비용 절감이라는 실질적인 효과도 기대할 수 있다. 지도학습 모델에서처럼 대규모의 데이터셋을 수동으로 라벨링하는 과정 없이, URL의 특징을 기반으로 자동으로 그룹화하고 분석할 수 있어 시간과 비용을 크게 줄일 수 있다. 마지막으로, 이 시스템은 지속적인 학습 및 개선 가능성을 가지고 있다. 주기적으로 새롭게 수집된 데이터를 기반으로 재학습을 진행하거나, 기존 모델을 개선할 수 있어 새로운 유형의 위협이 등장할 때마다 즉각적으로 대응할 수 있다. 이와 같은 기대효과를 통해 비지도학습 기반 악성 URL 탐지 시스템은 사이버 보안 분야에서 매우 중요한 기술로 자리잡을 것이며, 이를 통해 악성 URL로 인한 피해를 줄이고 보안의 효율성을 극대화할 수 있는 실질적인 해결책이 될 수 있다.

6. 별첨

6.1 팀원소개

▷ 팀 장	정여진	92015441	GMM 알고리즘 담당
▷ 팀 원	서장석	91913543	Meanshift 알고리즘 담당
	양승원	91913737	K-means 알고리즘 담당
	정채영	92015465	AgglomerativeCluster 알고리즘 담당

6.2 시연영상

▷ 졸업작품 시연영상

[그림 19] 시연영상 QR코드

6.3 발표자료(별첨)