URL 검사용 웹사이트 구축

2020. 10. 23

- **작품 완성:** 2020.10.18
- **보고서 작성 일정**
  2020.10.18~2020.10.23
- **조원 역할**
  송유진 - 보고서 총괄
  변하영 - 자료조사
  이지영 - 자료조사
  윤솔비 - 자료조사
  윤준호 - 자료조사

지도교수: 양환석 교수님

TEAM: 완성만

송유진, 변하영, 이지영, 윤솔비, 윤준호

목 차

조원 편성

이 름	역 할
송유진	프론트엔드 개발 [총괄]
변하영	URL 분석 및 개발
이지영	URL 분석 및 개발, 보고서 작성
윤솔비	프론트엔드 개발, PPT 작성
윤준호	DB 구축 및 연동, 보고서 작성

주제 선정

- **악성 URL 증가로 보안 위협 증대**

코로나19 관심 이용한 악성URL, 3만 4천 개 넘어

최형주 기자 | 승인 2020.04.14 14:00

SK인포섹이 지난 3월 초부터 4월 첫
(Indicator of Compromise, IOC)를 공개
터넷 프로토콜은 5232개, 피싱 URL은 5

침해지표는 해킹공격에 나타나는 침해
할 수 있다. 지표에는 악성코드를 유포
통보안취약점공개항목(CVE, Common V

SK인포섹 보안관제센터인 '시큐디움 선
두 112개이며, 악성메일 제목에는 'ATT
알리는 영어 어휘가 사용된 것으로 나타

파이어아이 "1분기, HTTPS 이용 악성 URL 26% 증가"

2019.07.15 10:06:51 / 홍하나 hhn0626@ddaily.co.kr

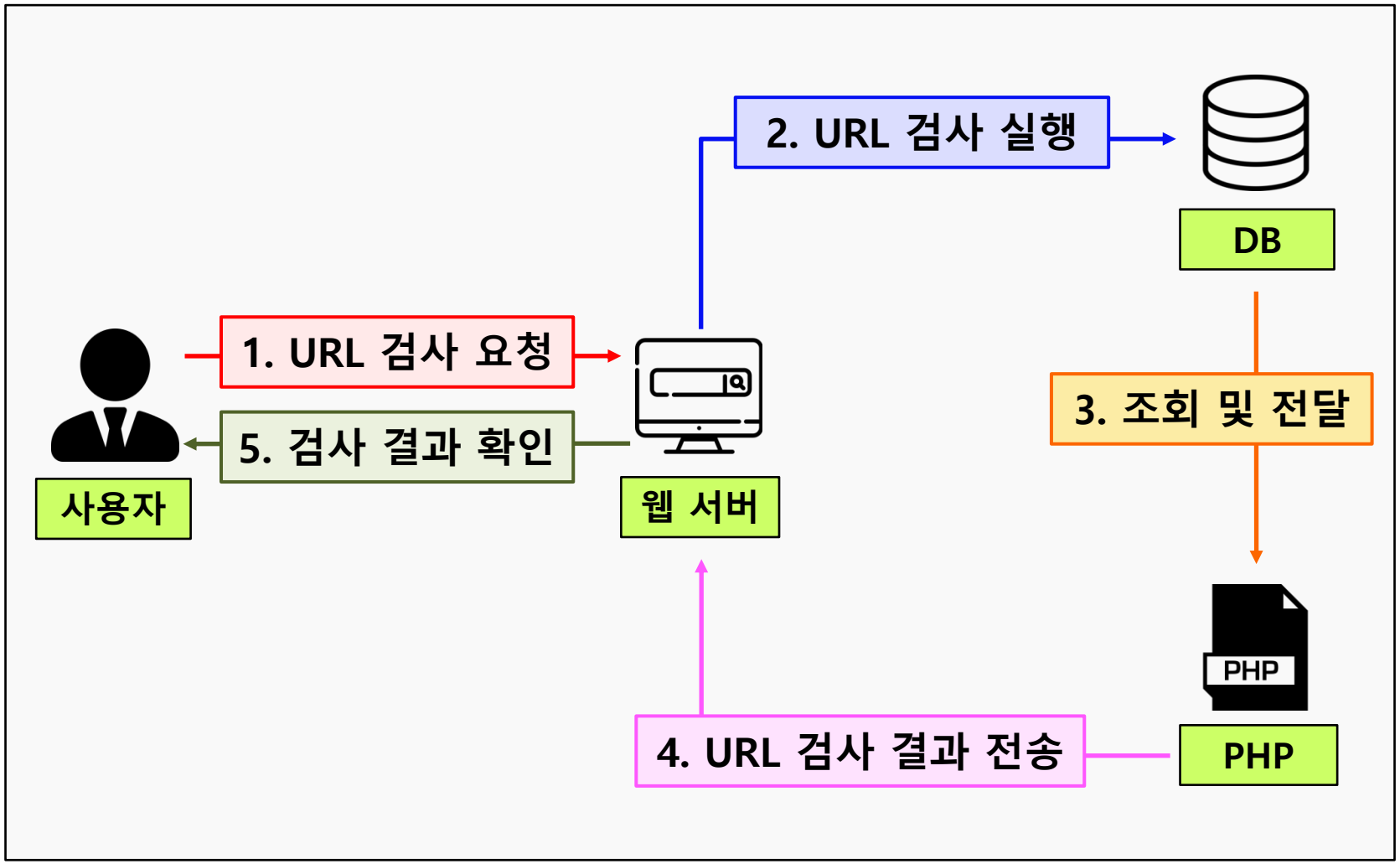
[디지털데일리 홍하나기자] 파이어아이(지사장 전수홍)가 2019년 1분기 이메일 위협 보고서를 발표했다고 15일 밝혔다.

파이어아이는 이번 보고서를 위해 13억 건의 이메일 샘플을 분석했다. 이를 통해 회사는 ▲스푸핑을 통한 피싱 시도 ▲HTTPS 암호화를 적용한 URL 기반 공격 ▲대중적인 파일 공유 서비스를 이용한 클라우드 기반 공격 등 세 개 주요 분야에서 위협 증가 추세가 발견됐다고 밝혔다.

먼저 파이어아이는 2019년 1분기 피싱 이메일 공격이 전 분기 대비 17% 증가했다고 밝혔다. 공격 활동에서 가장 많이 스푸핑된 기업 중 하나는 마이크로소프트(MS)로, 탐지된 건의 약 30%를 차지했다. 또한 원드라이브, 애플, 페이팔, 아마존이 그 뒤를

일반이용자들도 쉽게 이용할 수 있는 URL 검사 웹사이트 개발

구상도


2. URL 검사 실행
DB
1. URL 검사 요청
3. 조회 및 전달
5. 검사 결과 확인
사용자
웹 서버
PHP
4. URL 검사 결과 전송
PHP

추진 경과

추진기간 (2020년) / 수행업무	3월	4월	5월	6월	7월	8월	9월	10월
자료조사 및 연구								
프론트엔드 개발								
URL 분석 및 개발								
DB 구축 및 연동								
테스트 및 보완								

개발 환경 및 개발 내용(1/10)

개발 환경

OS

Windows 10

Development Language

HTML
JavaScript
PHP
Python

Web Server

Apache

DB

MySQL

개발 환경 및 개발 내용(2/10)

개발내용(1/9)

- **URL 검사 화면**

URL 검사화면 prog

```
<center><img src="./cdlogo.jpg" alt="My Image" width="40
  <div class="header2">
  <h3><b>의심스러운 URL을 입력해주세요</b></h3>
</div>
</center>

<div class="searchbox">
  <div class="header">
      <form name="url_searching" method="post" action="url_searching.php">
          <input type="text" name="u1" id="value" placeholder="URL을 입력해주세요"

          <div id="btn_group">
              <button type="submit"
                      id="test_btn1">
                      Search
              </button>
          </div>
```

URL 검사 서비스를 제공하기 위한 URL 검사 화면 설계 개발

개발 환경 및 개발 내용(3/10)

개발내용(2/9)

- DB 구축

url
keeperstop.com
scoophub.in
zeonic-republic.net
toa.edu.my
inetres.com
audioriver.pl
alivefoot.us
caughtinsouthie.com
recursosculturales.com
tribogames.com.br
melovin.site
barebackshemalemovie
sarina724.ir
siyred.com
we-conect.com
clubtickets.com
profidom.com.ua
diverte-me.com
hitmedia.in

DB 생성

DB 연결

```php
<?php

    //mysql 연결
    $db_host = "localhost";
    $db_user = "comonly3";
    $db_passwd = "infosec8!";
    $db_name = "comonly3";
    $conn = mysqli_connect($db_host,$db_user,$db_passwd,$db_name);

    // 문자셋 설정, utf8.
    mysqli_set_charset($conn,"utf8");
```

URL 검사 데이터 관리를 위한 DB 구축

개발 환경 및 개발 내용(4/10)

개발내용(3/9)

- **DB 동작 설계**

문자열 비교 및 리다이렉트

```
$u1_d = $_POST["u1"];
$re1 = mysqli_query($conn, "SELECT * FROM url_data WHERE url
$a_re1 = mysqli_fetch_assoc($re1);
$s_re1 = implode('',(array)$a_re1);

$re2 = mysqli_query($conn, "SELECT * FROM url_data WH
$a_re2 = mysqli_fetch_assoc($re2);
$s_re2 = implode('',(array)$a_re2);

$re3 = mysqli_query($conn, "SELECT * FROM url_data WH
$a_re3 = mysqli_fetch_assoc($re3);
$s_re3 = implode('',(array)$a_re3);

 $re_str = $s_re1===$u1_d and $s_re2===$u1_d and $s_

if($re_str){
echo ("<script>
document.location.href='http://comonly3.dothome.co.kr
</script>");
}else{
   echo ("<script>
```

블랙리스트 출력

```
<?php
$db_host = "localhost";
$db_user = "comonly3";
$db_passwd = "infosec8!";
$db_name = "comonly3";
$conn = mysqli_connect($db_host,$db_user,$db_passwd,$db_name);

mysqli_set_charset($conn,"utf8");


$result = mysqli_query($conn,"SELECT * FROM url_data");

while($row = mysqli_fetch_array($result))
{
echo "<tr>";
echo "<td>" . $row['url'] . "</td>";
echo "</tr>";
}
echo "</table>";
```

URL 검사 결과와 블랙리스트 출력을 위한 DB 동작 설계

개발 환경 및 개발 내용(5/10)

개발내용(4/9)

- URL 검사 결과

정상 URL

```
<h1><b>정상 URL</b></h1>
<form name="url_searching" method="post" action="url_searching.php">
  <input type="t
         name="u
         id="val
         onkeyup
         placeho

  <div id="btn_g
    <button id="
  </div>
</form>
</div>
<div class="head
 <h3>이 URL은 <s
```

피싱 URL

```
<h1><b>피싱 URL</b></h1>
<form name="url_searching" method="post" action="url_searching.p
    <input type="text"
           name="u1"
           id="value"
           onkeyup="filter()"
           placeholder="URL을 입력해주세요"/>

    <div id="btn_group">
        <button id="test_btn1" type="submit"><b>Search</b></button>
    </div>
</form>
</div>

<div class="header2">
```

URL 검사에 대한 결과를 화면에 출력

개발 환경 및 개발 내용(6/10)

개발내용(5/9)

- 블랙리스트

블랙리스트 prog

```
<section id="showcase">
  <h1><img src="URL.png" height="60" /> 블랙리스트 조회</h1>

  <div class="datatable">
    <table id="foo-table" class="table table-bordered">
      <thead>
        <tr>
          <td>URL</td>
        </tr>
      </thead>
      <?php include("black_db.php"); ?>
    </table>
  </div>
</section>

<!--datatables script-->
<script>
  jQuery(function ($) {
    $("#foo-table").DataTable({
      "fnInitComplete" : function() {
                $("#foo-table").css("width", "42.7cm");
```

DB의 피싱 URL 데이터를 이용하여 블랙리스트 출력

개발 환경 및 개발 내용(7/10)

개발내용(6/9)

- **Phishing Crawling 개발(1/2)**

```
df_normal_url = pd.read_csv('./dataset/정상사
df_normal_url
```

사이트 목록 파일

```
df_phishing_dataset = pd.concat([
                            df_phishing_dataset,
                            pd.DataFrame( {'url':df_abnormal_url['url'].tolist(),
                                           'is_phishing': [1 for _ in range(len(df_abnormal_url['url'].tolist()
                            ])

df_phishing_dataset
```

df_phishig_dataset.csv

	url	is_phishing
0	keeperstop.com	0
1	scoophub.in	0
2	zeonic-republic.net	0
3	toa.edu.my	0
4	inetres.com	0

정상/피싱 사이트목록.csv 파일 데이터를 처리하여
df_phishig_dataset.csv 파일 DataFrame 생성

개발 환경 및 개발 내용(8/10)

개발내용(7/9)

- **Phishing Crawling 개발(2/2)**

URL 기반 특징

```
def ip_address(url):
    is_ip = re.match(".+#d{1,3}[.]#d{1,3}[.]#d{1,3}[.
    if bool(is_ip):
        return 1
```

비정상적인 특징

```
check_port_list = [80, 443, 21, 22, 445, 1433, 1521, 3306, 3389]
port_list = []

def portscan(url, port, port_list):

    s = socket.socket(so
    s.settimeout(0.5)

    try:
        con = s.connect(
        port_list.append(
        con.close()
    except:
        pass

def find_open_port(url,

    for x in check_port_

        t = threading.Th
        t.start()
```

HTML 특징

```
url = url.replace('https://','').replace('http://','')
domain_name = re.sub('(.+)[.]([a-z]{2,3}.+)', '#g<1>',
domain_name = domain_name.split('.')[0]
if domain_name in soup_text:
    dom_in_source = 1
else:
    dom_in_source = 0

len_of_source = len(soup_text)

num_of_head = soup_text.count('<head>')
num_of_body = soup_text.count('<body>')
if num_of_head > 1 or num_of_body > 1:
    dup_HEAD = 1
else:
```

피싱 사이트 특징을 통해 URL 피싱 여부 검사

개발 환경 및 개발 내용(9/10)

개발내용(8/9)

- 피싱 사이트 탐지(1/2)

```
import requests
from bs4 import BeautifulSoup

df_phishing_dataset = pd.read_csv(os.getcwd()+'/df_p
pTesttest_dataset = pd.read_csv(os.getcwd()+'/test_dataset.csv')

set(df_phishing_dataset['open_ports'])

import re
re.sub('[^0-9 ]','','[80, 443, 1521, 3306, 3389]').split()

for idx in tqdm(df_phishing_dataset.index):
    for port in [80, 443, 21, 22, 445, 1433, 1521, 3306, 3389]:
        if port in re.sub('[^0-9 ]','',df_phishing_dataset.loc[idx, 'open_ports']).split():
            df_phishing_dataset.loc[idx, f'open_port_{port}'] = 1
            pTesttest_dataset.loc[idx, f'open_port_{port}'] = 1

new_columns = df_phishing_dataset.columns.tolist()
new_columns.remove('open_ports')

df_phishing_dataset2 = df_phishing_dataset[new_columns].copy()
pTesttest_dataset2 = pTesttest_dataset[new_columns].copy()


df_phishing_dataset2 = df_phishing_dataset2[df_phishing_dataset2['redirect_num']!='ERROR URL']
pTesttest_dataset2 = pTesttest_dataset2[pTesttest_dataset2['redirect_num']!='ERROR URL']

df_phishing_dataset2 = df_phishing_dataset2.fillna(0)
```

df_phishing_dataset.csv, test_dataset.csv 호출 및 재구성

df_phishing_dataset.csv를 불러와 모델 생성 때 쓰일 파일로 재구성

개발 환경 및 개발 내용(10/10)

개발내용(9/9)

- 피싱 사이트 탐지(2/2)

train/test set 분리

```
from sklearn.model_selection import train_test_split
df_train, df_test = train_test_split(df_phishing_dataset2, test_size=0.3, random_state=777, stratify=df_phishing_
```

로지스틱 리그레션을 위한 정규화

```
from sklearn.preprocessing i

scaler = StandardScaler()

df_train_sd = s
df_test_sd = sc
sample = scaler
```

모델 생성 및 fitting

```
from sklearn.linear_model import

model = LogisticRegression()
model.fit(df_train_sd, d
```

url 예측 및 불필요 파일 삭제

```
import num

one = np.array(sample)

print(model.predict(one))

import os

os.remove(r'Ftest.csv')
os.remove(r'test_dataset.csv')
```

로지스틱 회귀 알고리즘을 이용한 모델 생성 및 URL 피싱 여부 예측

개발 시스템 운영 (1/6)

메인 화면

개발 시스템 운영 (2/6)

프로젝트 소개 화면

PHISHING CHECKER

프로젝트 소개

완성만 홈 팀원소개 프로젝

1

url 기반 특징

- ▶url내 ip 보유
- ▶url 길이
- ▶url 단축서비스
- ▶url에 @포함
- ▶url에 //포함

팀원 소개

팀원 소개

팀장

91707815 송유진

프론트엔드 개발 [총괄]

91716880 윤솔비

프론트엔드 개발 및 PPT 작성

91514919 이지영

URL 분석 및 개발 및 보고서 작성

91613753 변하영

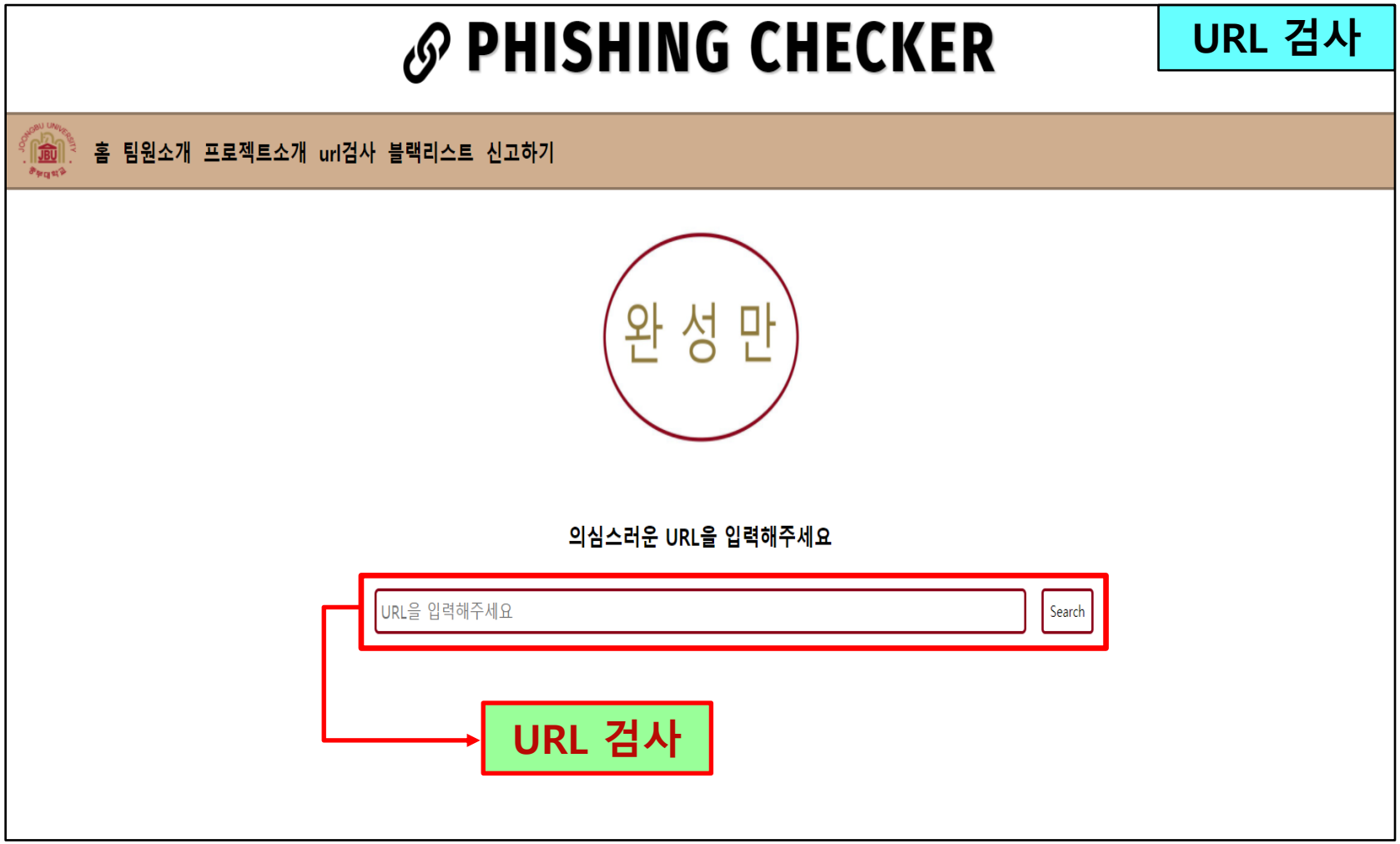
URL 분석 및 개발

91950735 윤준호

DB 구축 및 연동, 보고서 작성

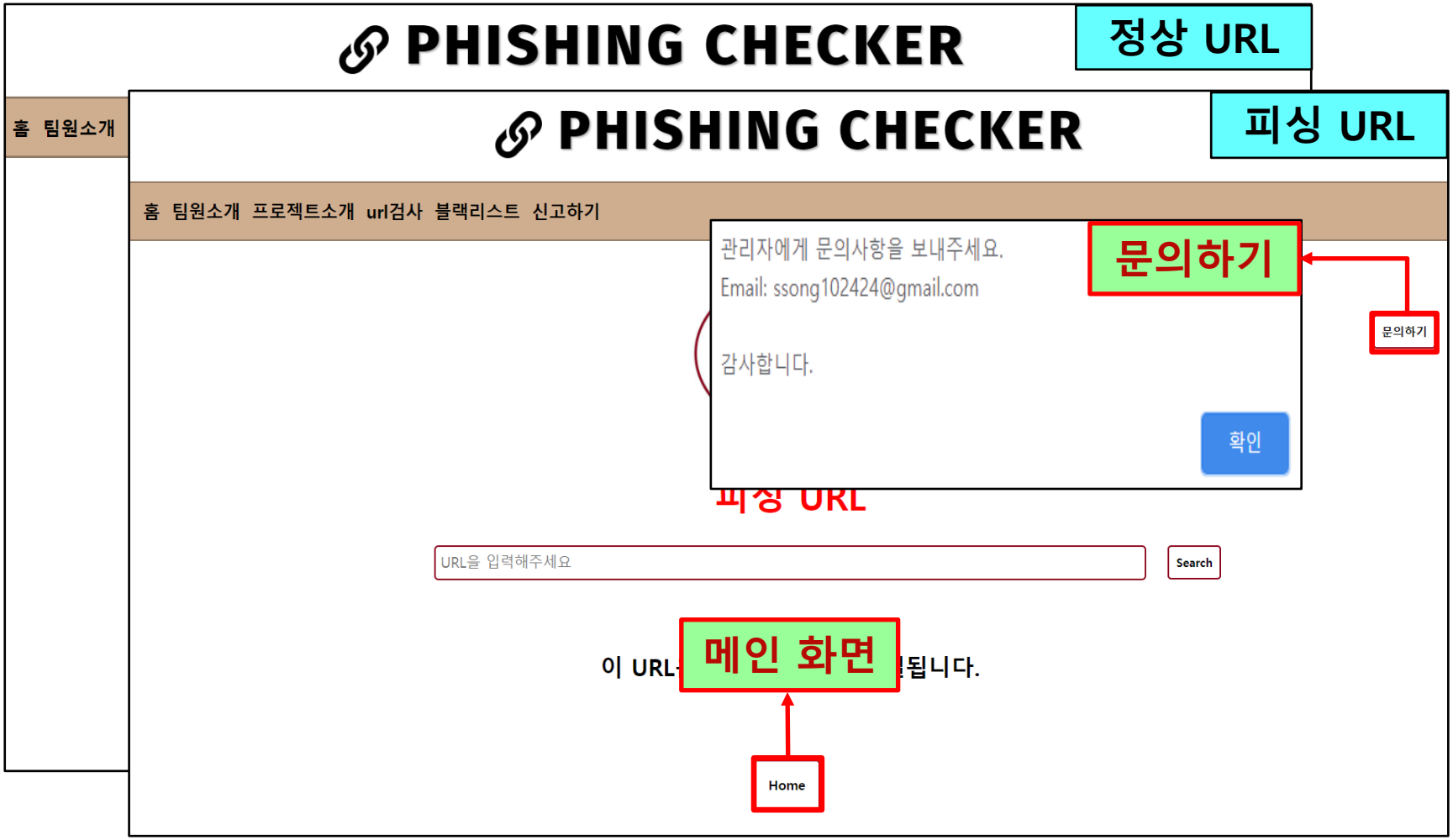
개발 시스템 운영 (3/6)

URL 검사 화면



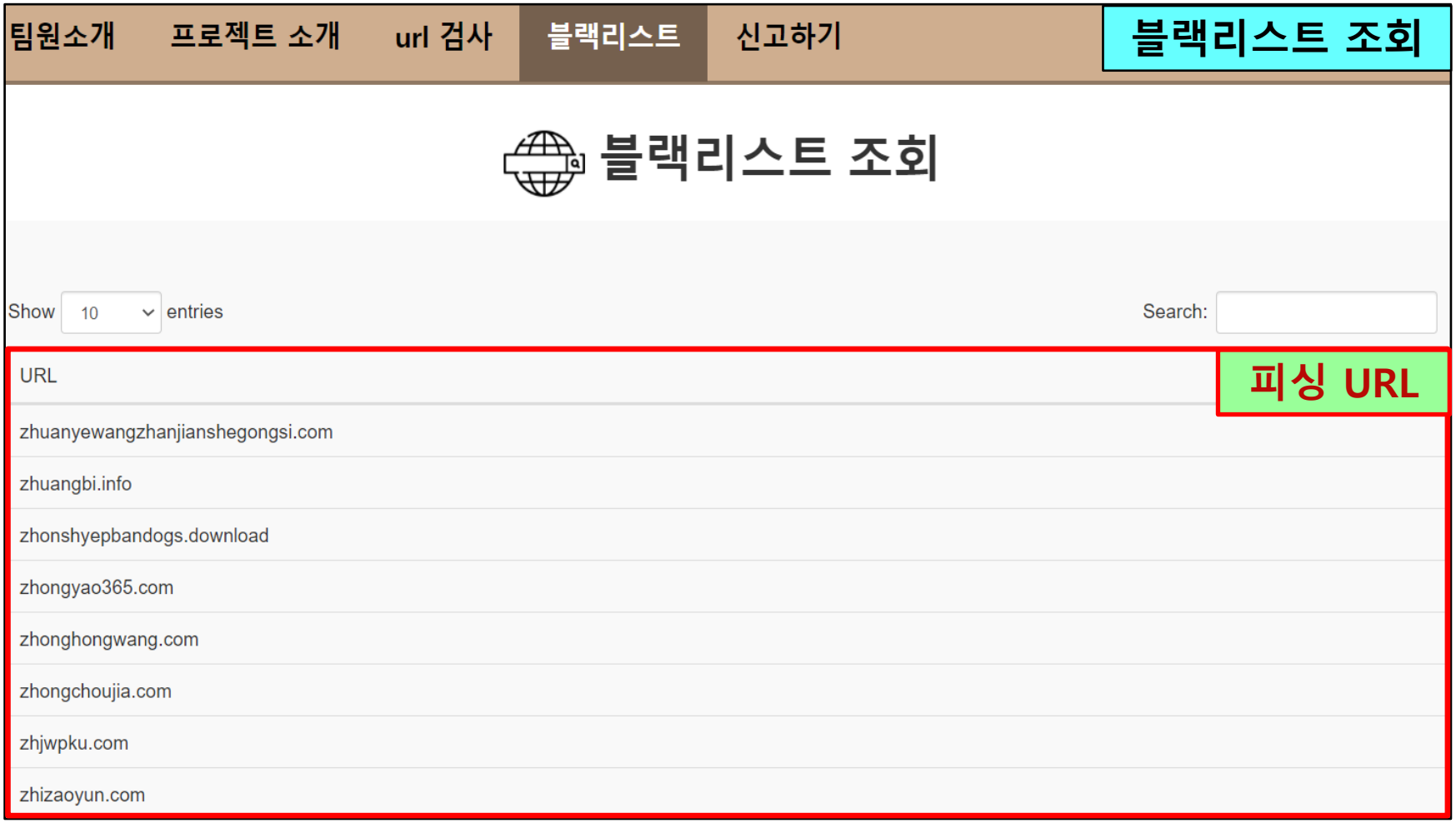
개발 시스템 운영 (4/6)

URL 검사 결과



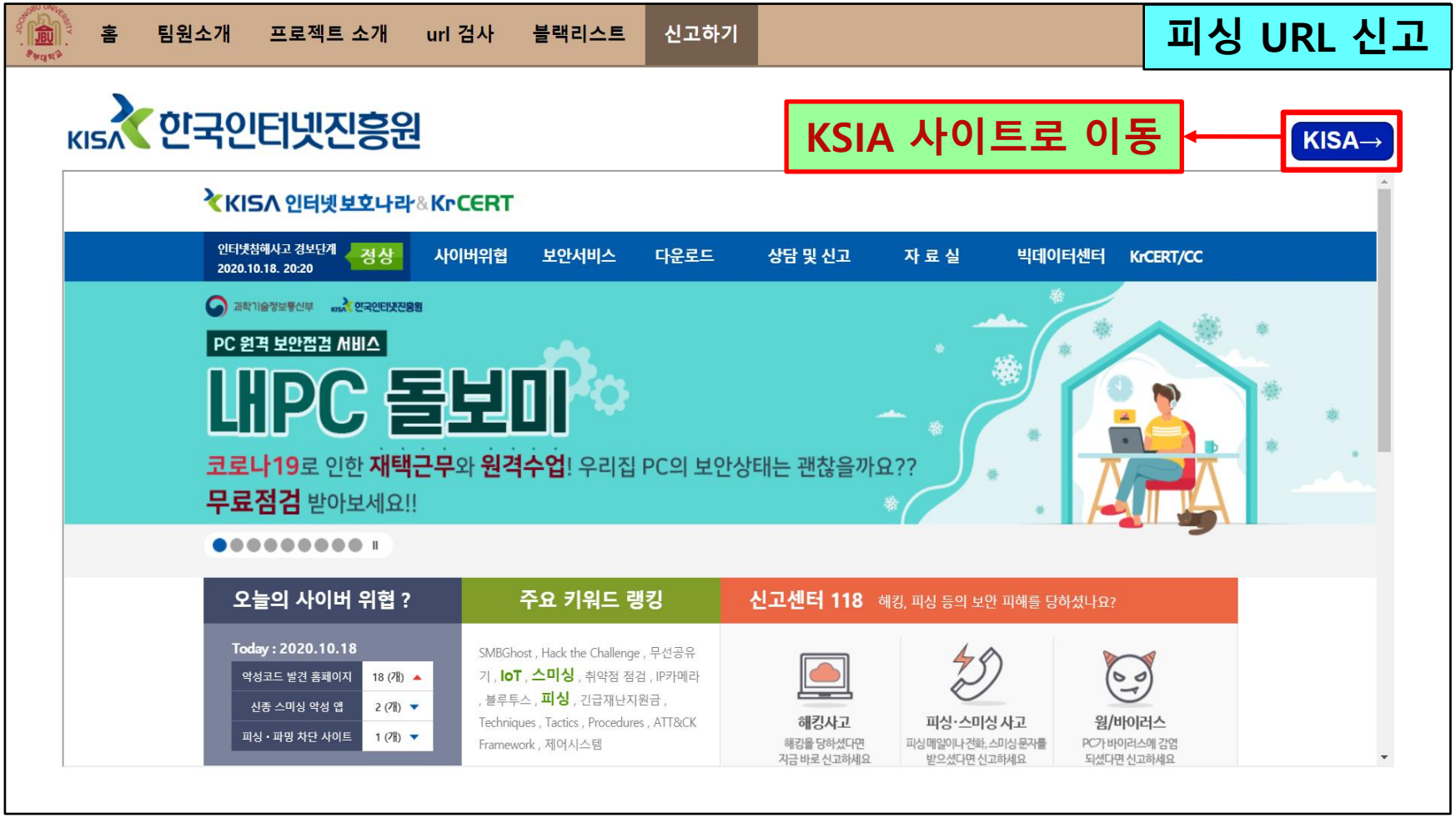
개발 시스템 운영 (5/6)

블랙리스트



개발 시스템 운영 (6/6)

피싱 URL 신고



결론 및 기대효과

- **결　　론**

  - 자체 기술력으로 URL 검사 시스템을 구현하고, 이를 기반으로 URL 검사 웹사이트를 개발하는데 성공

  - 자료 수집 및 연구 등 철저한 사전 준비와 팀원간 역할분담으로 차질없는 시스템 개발

- **기대 효과**

  - 일반 이용자들도 쉽게 URL 검사를 실시하여 악성 URL에 대한 해킹 위험 노출 방지

  - URL 분석 기술에 대한 심도있는 연구와 기술 개발을 통해 전문기술 역량을 배양. 끝.

Q & A

감사합니다