# 웹 취약점 분석

2020. 10. 23

지도교수 : 양환석 교수님

2 조 곽준영
정종영
장현수
유영선
한지예

# 목차

# 조원 편성

| 이름 | 역할 |
| --- | --- |
| 정종영(조장) | DB 구축 및 관리(총괄), 백엔드 구축 및 개발 |
| 곽준영 | 프론트엔드 구축 및 개발, ppt 작성 |
| 장현수 | 프론트엔드 구축 및 개발, 보고서 작성 |
| 한지예 | 프론트엔드 구축 및 개발, 보고서 작성 |
| 유영선 | 백엔드 구축 및 개발, ppt 작성 |

# 주제 선정(1/3)

## [기고] 여전히 위협받는 웹 애플리케이션, 해답은 없는가

[컴퓨터월드] 지난 12월, 2020학년도 대학수학능력시험 성적 발표를 이틀 앞두고, 한국교육과정평가원 홈페이지
의 허점을 이용해 수험생들
졸업생들을 대상으로 과거의

수험생들은 이 페이지에 존
리 조회했다. 이번 일은 특히
때문에, 이번 사고의 원인이
다.

## 심각한 보안홀 '애플리케이션'...보호 방안 불충분

김선애 기자 | 승인 2020.03.06 16:14 | 댓글 0

> 라드웨어, 애플리케이션 보호 위한 4가지
> 봇 관리·API 보안·DoS 방어·지속적인 보안
> 웹방화벽만으로 모든 웹 공격 방어 못해

[데이터넷] 애플리케이션이 심각한 보안홀
로 보인다. 라드웨어의 '글로벌 애플리케이
현재 조직에서 가장 빠르게 증가하는 사이
현황'에서는 33%의 조직만이 웹방화벽으로
으며, 또한 진화하는 애플리케이션 취약점
강조했다.

## 보안에 좋다던 HTTPS와 SSL, 오히려 사이버 위협이 되고 있다

안전한 트래픽의 상징인 HTTPS와 자물쇠 아이콘, 공격자들도 쉽게 따라해

암호화 기술 좋다고 홍보하고 권장하려면 실제 사용 고려한 환경 조성에도 힘써야

[보안뉴스 문가용 기자] SSL 인증서를 공격에 이용하는 사이버 범죄자들이 늘어나고 있다. 브라우저의 주소창에 HTTPS라는 글자와 자물쇠 아이콘이 나오면 안심을 하는 심리를 악용하는 수법이다.

**자주 사용하는 HTTPS와 SSL 또한 사이버 범죄에 이용당하는 추세**

# 주제 선정(2/3)

뉴스홈 > 보안/해킹

웹 호스팅 업체 '인터넷나야나' 랜섬웨어 감염

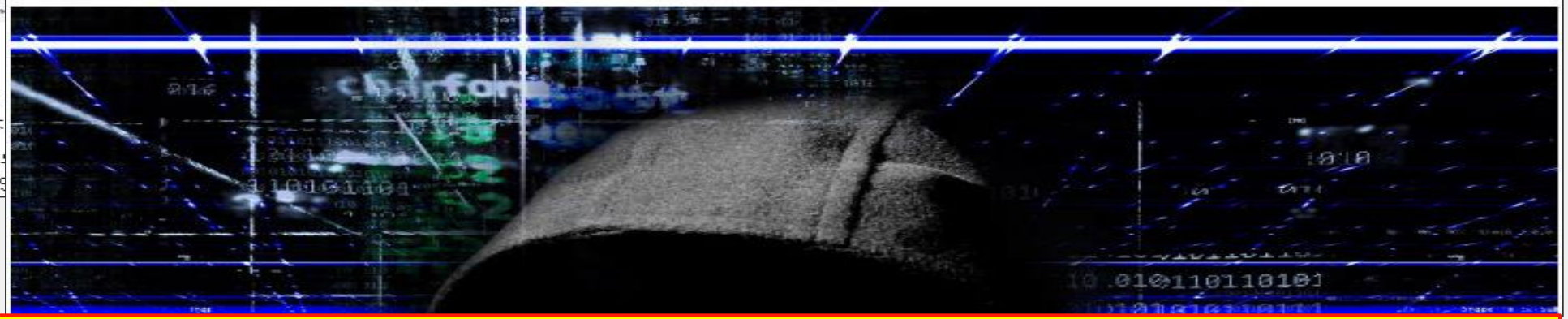
5천여 웹사이트 피해 추정…미래부·KISA, 복구 지원 나서



안랩, '플래시 취약점' 악용한 악성코드 주의보

입력 2019-08-28 14:34 수정 2019-08-28 14:34





[아이티데일리] 웹
피해 규모 역시 점치

12일 미래창조과학
구체적으로는 한국

**각종 악성코드 감염 등 사이버 침해에 대한 대응태세 확립 긴요**

# 주제 선정(3/3)

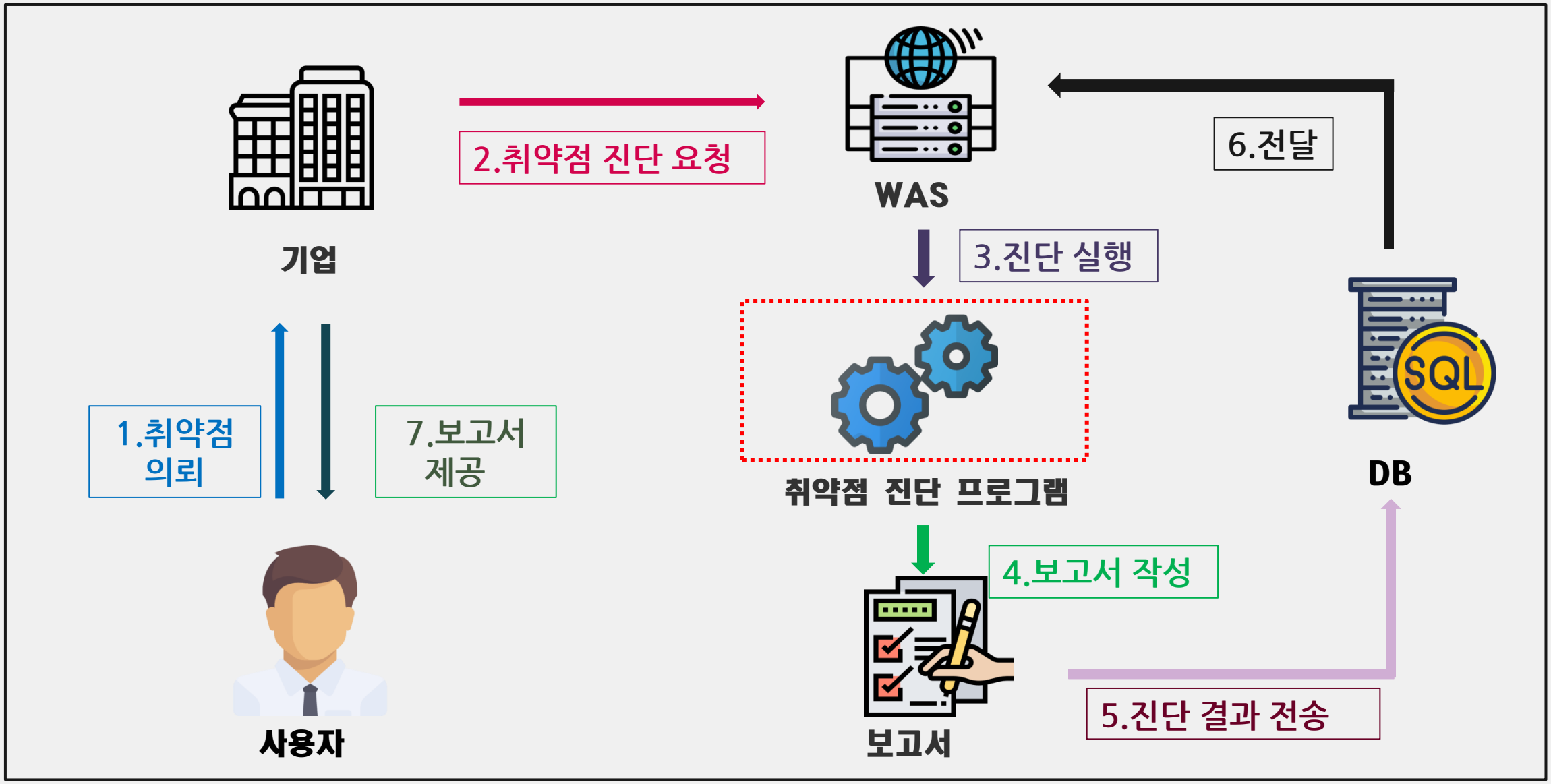
※ OWASP : The Open Web Application Security Project


owasp 10

| | |
|---|---|
| A1:2017-<br>인젝션 | SQL, OS, XXE, LDAP 인젝션 취약점은 신뢰할 수 없는 데이터가 명<br>인터프리터로 보내질 때 발생합니다. 공격자의 악의적인 데이터는<br>실행하거나 올바른 권한 없이 데이터에 접근하도록 인터프리터를 |
| A2:2017-<br>취약한 인증 | 인증 및 세션 관리와 관련된 애플리케이션 기능이 종종 잘못 구현되어 공격자들이 암호, 키, 세션<br>토큰을 위험에 노출시킬 수 있거나 일시적 또는 영구적으로 다른 사용자의 권한 획득을 위해<br>구현 상 결함을 악용하도록 허용합니다. |
| A3:2017-<br>민감한 데이터<br>노출 | 다수의 웹 애플리케이션과 API는 금융 정보, 건강 정보, 개인 식별 정보와 같은 중요한 정보를<br>제대로 보호하지 않습니다. 공격자는 신용카드 사기, 신분 도용 또는 다른 범죄를 수행하기 위해<br>보호가 취약한 데이터를 훔치거나 수정할 수 있습니다. 중요한 데이터는 저장 또는 전송할 때<br>암호화 같은 추가 보호 조치가 없으면 탈취 당할 수 있으며, 브라우저에서 주고 받을 때 각별한<br>주의가 필요합니다. |
| A4:2017-<br>XML 외부 개체<br>(XXE) | 오래되고 설정이 엉망인 많은 XML 프로세서들은 XML 문서 내에서 외부 개체 참조를 평가합니다.<br>외부 개체는 파일 URI 처리기, 내부 파일 공유, 내부 포트 스캔, 원격 코드 실행과 서비스 거부<br>공격을 사용하여 내부 파일을 공개하는데 사용할 수 있습니다. |
| A5:2017-<br>취약한 접근 통제 | 인증된 사용자가 수행할 수 있는 작업에 대한 제한이 제대로 적용되어 있지 않습니다. 공격자는<br>이러한 결함을 악용하여 다른 사용자의 계정에 접근하거나, 중요한 파일을 보거나, 다른 사용자의<br>데이터를 수정하거나, 접근 권한을 변경하는 등 권한 없는 기능과 데이터에 접근할 수 있습니다. |
| A6:2017-<br>잘못된 보안 구성 | 잘못된 보안 구성은 가장 흔하게 보이는 이슈입니다. 취약한 기본 설정, 미완성 (또는 임시 설정),<br>개방된 클라우드 스토리지, 잘못 구성된 HTTP 헤더 및 민감한 정보가 포함된 장황한 에러<br>메시지로 인한 결과입니다. 모든 운영체제, 프레임워크, 라이브러리와 애플리케이션을 안전하게<br>설정해야 할 뿐만 아니라 시기 적절하게 패치/ 업그레이드를 진행해야 합니다. |
| A7:2017-<br>크로스 사이트<br>스크립팅 (XSS) | XSS 취약점은 애플리케이션이 올바른 유효성 검사 또는 필터링 처리 없이 새 웹 페이지에 신뢰할<br>수 없는 데이터를 포함하거나, 자바스크립트와 HTML을 생성하는 브라우저 API를 활용한 사용자<br>제공 데이터로 기존 웹 페이지를 업데이트할 때 발생합니다. XSS는 피해자의 브라우저에서<br>공격자에 의해 스크립트를 실행시켜 사용자 세션을 탈취할 수 있게 만들고, 웹 사이트를<br>변조시키고, 악성 사이트로 리다이렉션할 수 있도록 허용합니다. |
| A8:2017-<br>안전하지 않은<br>역직렬화 | 안전하지 않은 역직렬화는 종종 원격 코드 실행으로 이어집니다. 역직렬화 취약점이 원격 코드<br>실행 결과를 가져오지 않더라도 이는 권한 상승 공격, 주입 공격과 재생 공격을 포함한 다양한<br>공격 수행에 사용될 수 있습니다. |
| A9:2017-<br>알려진 취약점이<br>있는 구성요소<br>사용 | 라이브러리, 프레임워크 및 다른 소프트웨어 모듈 같은 컴포넌트는 애플리케이션과 같은<br>권한으로 실행됩니다. 만약에 취약한 컴포넌트가 악용된 경우, 이는 심각한 데이터 손실을<br>일으키거나 서버가 장악됩니다. 알려진 취약점이 있는 컴포넌트를 사용한 애플리케이션과 API는<br>애플리케이션 방어를 약화시키거나 다양한 공격과 영향을 주게 합니다. |

**Owasp10에 근거하여 웹 취약점을 종합 점검하는 시스템 구축**

# 구상도


기업
2.취약점 진단 요청
WAS
6.전달
3.진단 실행
1.취약점
의뢰
7.보고서
제공
취약점 진단 프로그램
DB
SQL
4.보고서 작성
5.진단 결과 전송
사용자
보고서

# 추진 계획 경과

| 수행 업무 \ 추진 기간(2020년) | 3월 | 4월 | 5월 | 6월 | 7월 | 8월 | 9월 | 10월 |
|---|---|---|---|---|---|---|---|---|
| 자료조사 및 분석<br>-웹 취약점 종합 분석<br>-개별 시스템 설계 | | | | | | | | |
| 진단항목 결정 | | | | | | | | |
| Apm 환경 구축 | | | | | | | | |
| 진단시스템 개발 | | | | | | | | |
| 시스템 검증 및 보완 | | | | | | | | |

# 개발 환경 및 개발내용(1/12)

운영체제

Ubuntu -> 서버

웹 서버

Apache -> 취약 사이트 서버

데이터베이스

MySQL -> 정보 및 결과 저장

개발언어

Php, Html -> 웹 사이트 제작

Python -> 취약점 진단 제작

# 개발 환경 및 개발내용(2/12)

## 개발 내용(1/11)

◆DataBase(mysql) 구축

php server

```php
<?php

$conn = mysqli_connect("localhost", "root", "root","blind_db");
$no = $_GET['no'];

include 'index.php';
echo "<br>";

$userinfo = "SELECT * FROM sql_injection WHERE no =".$no."";
$useridoverlap=mysqli_query($conn, $userinfo);
$user_db_check= mysqli_fetch_array($useridoverlap);


if($user_db_check['no']=="5"){
echo "no: ".$user_db_check[0]."<br>";
echo "id: ".$user_db_check[1]."<br>";
echo "email: ".$user_db_check[3]."<br>";
}
else if($user_db_check['no']){

echo "no: ".$user_db_check[0]."<br>";
echo "id: ".$user_db_check[1]."<br>";
```

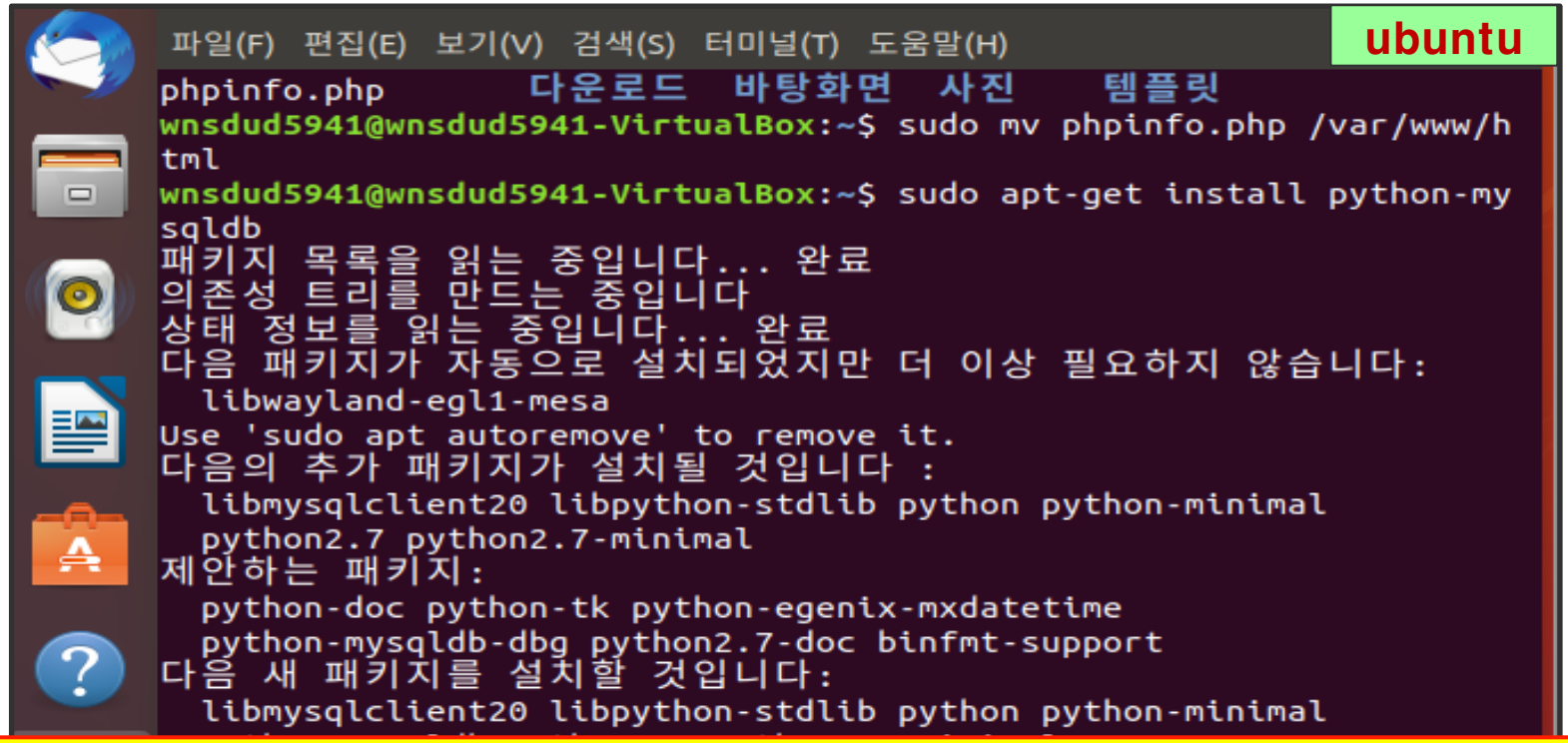
myadmin


DB 서버 및 myadmin 연동

# 개발 환경 및 개발내용(3/12)

## 개발 내용(2/11)

◆ os 연동




**파이썬과 연동하는 os 환경 구축**

# 개발 환경 및 개발내용(4/12)

## 개발 내용(3/11)

◆ 웹 사이트 개발(프로그램)

사용자 등록

```
<!-- <?php
session_start();
$id = $_SESSION['id'];
$name = $_SESSION['name'];
?> -->

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-sca
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Serendipity</title>

  <link rel="stylesheet" href="vendors/bootstrap/bootstrap.min.c
  <link rel="stylesheet" href="vendors/fontawesome/css/all.min.c
  <link rel="stylesheet" href="vendors/themify-icons/themify-ico
  <link rel="stylesheet" href="vendors/linericon/style.css">
  <link rel="stylesheet" href="vendors/owl-carousel/owl.theme.de
  <link rel="stylesheet" href="vendors/owl-carousel/owl.carousel
  <link rel="stylesheet" href="css/magnific-popup.css">
```

초기 화면

```
<?php include('db.php'); include('check.php');
 if(is_login()){
 if(isset($_SESSION['user_id'])) { header("Location: welcom
?>
<!DOCTYPE HTML>
<html>
<head>
<title>WEST</title>
<meta http-equiv="content-type" content="text/html; charset
<meta name="description" content="" />
<meta name="keywords" content="" />
<script src="js/jquery.min.js"></script>
<script src="js/skel.min.js"></script>
<script src="js/skel-layers.min.js"></script>
<script src="js/init.js"></script>
</head>
<script >
function check() {
alert(' 로그인 먼저하세요.')
}
</script>
```

진단 결과

```
<?php
 include('db.php'); include('check.php');
 if (is_login()){ ; }else
 header("Location: west.php");
 ?>
<?php
 $user_id = $_SESSION['user_id'];
 try {
 $stmt = $con->prepare('select * from yo where username=:username'); $stmt->bindParam
 } $row = $stmt->fetch();
?>
<!doctype html>
<html lang="kr">
 <head>
 <meta charset="UTF-8">
 <title>WeST</title>
</head>
<body>
 <p>진단 결과</p>
 <table class="table" border="1">
 <tr> <th>번호</th><th>제목</th><th>날짜</th><th>Download</th> </tr>
 <tbody>
 <tr> <td>1</td>
 <td><?php echo $row['username']?>의 진단 결과</td>
```

**계정 생성, 진단결과 및 초기화면 구현 등 웹 사이트 개발**

# 개발 환경 및 개발내용(5/12)

## 개발 내용(4/11)

◆ 진단 프로그램 개발(SQL Injection)

module

```
#A1 : SQL Injection
import urllib
import requests
from bs4 import BeautifulSoup
import re
from .required.common import
from .required.decoration imp
from .required.separator impo
```

함수 구현

```
def SQL_checking(extern_file):
    url = "http://demo.testfire.net"
    SQL_checking.payload = "data/payload/SQL_payloads.txt"
    max_lines = line_counting(SQL_checking.payload)
    SQL_checking.process_count = max_process(max_lines)
    list = [None]*(max_lines)
    upper_decoration(ModuleNumber.SQLINJECTION, extern_file)

    response = urllib.request.urlopen(url)
    url_read = response.read()
    soup = BeautifulSoup(url_read,"html.parser")

    for link in soup.findAll("a", href = re.compile("^(/)((?!:).)*$")):
        if 'href' in link.attrs:
            print(link.attrs['href'])
            site = url + link.attrs['href']
```

payloads

SQLinjection_payloads - Windows

파일(F) 편집(E) 서식(O) 보기(V)

```
' OR '1
' OR 1 -- -
" OR "" = "
" OR 1 = 1 -- -
' OR "" = '
ORDER BY SLEEP(5)
ORDER BY 1,SLEEP(
' AND id IS NULL; --
'''''''''''''UNION SELEC
%00
/*...*/
1' ORDER BY 1--+
1' ORDER BY 2--+
1' GROUP BY 1,2--+
```

결과값

SQL_payloads.txt | CheckedList.txt | main.p

```
= = = = = SQL Injection = = = = = =
[SQL Injection] http://demo.testfire.net/index.jsp?content=insi
[SQL Injection] http://demo.testfire.net/index.jsp?content=insi
[SQL Injection] http://demo.testfire.net/index.jsp?content=pers
[SQL Injection] http://demo.testfire.net/index.jsp?content=pers
[SQL Injection] http://demo.testfire.net/index.jsp?content=busi
[SQL Injection] http://demo.testfire.net/index.jsp?content=busi
[SQL Injection] http://demo.testfire.net/index.jsp?content=insi
[SQL Injection] http://demo.testfire.net/index.jsp?content=insi
[SQL Injection] http://demo.testfire.net/index.jsp?content=priv
[SQL Injection] http://demo.testfire.net/index.jsp?content=priv
[SQL Injection] http://demo.testfire.net/index.jsp?content=secu
```

웹 취약점 SQL Injection 구현을 위한 진단 프로그램 개발

# 개발 환경 및 개발내용(6/12)

## 개발 내용(5/11)

◆ 진단 프로그램 개발(XSS Injection)


웹 취약점 XSS Injection 구현을 위한 진단 프로그램 개발

# 개발 환경 및 개발내용(7/12)

## 개발 내용(6/11)

◆ 진단 프로그램 개발(Certification Bypass)

module

```
#A3 : Certification Bypass

import requests

from requests.auth import

from .required.decoration

from .required.separator i
```

함수 구현

```
def bypass_checking(extern_file):
    url = "http://demo.testfire.net/login.jsp"
    bypass_checking.tag = "[" + module_name(ModuleNumber.BYPASSCHECK) + "] "

    http_user = " or 1=1--"
    http_pass1 = " sss"
    http_pass2 = " or 1=1--"

    upper_decoration(ModuleNumber.BYPASSCHECK, extern_file)
    req1 = requests.get(url, auth=HTTPBasicAuth(http_user, http_pass1), verify=
    req2 = requests.get(url, auth=HTTPBasicAuth(http_user, http_pass2), verify=


    if req1.status_code == 401 and req2.status_code == 200:
        print("Page '"+url+"' is probably VULNERABLE")
        found_data = open(extern_file, "a")
        found_data.write(bypass_checking.tag + url + '\n')
        found_data.close()
        print("\033[0;32m" + bypass_checking.tag + url + " Found!\033[0m")
```

결과값

```
: : : : : : Certification Bypass : : : : : :

[Certification Bypass]http://demo.testfire.net/login
```

웹 취약점 Certification Bypass 구현을 위한 진단 프로그램 개발

# 개발 환경 및 개발내용(8/12)

## 개발 내용(7/11)

◆ 진단 프로그램 개발(Cookie)

module

```
#A4 : Cookie Modulation
from urllib.parse import urlencode
from urllib.request import Request
from .required.decoration import *
from .required.separator import *
```

함수 구현

```
def cookie_checking(extern_file):
    url = "http://demo.testfire.net/login.jsp"
    cookie_checking.tag = "[" + module_name(ModuleNumber.COOKIEMOD) + "] "
    login_form = {"id":"' or 1=1--", "pw":"' or 1=1--"}
    login_req = urlencode(login_form)
    login_req = login_req.encode('utf-8')
    request = Request(url, login_req)
    response = urlopen(request)
    cookie = response.headers.get('Set-Cookie')

    print("=========================================================================")
    print("접속한 사이트 쿠키 : "+cookie+"")
    print("=========================================================================")


    url1 = "http://demo.testfire.net/bank/main.jsp/"
    request = Request(url1)
    response1 = urlopen(request)

    cookie1 = response1.headers.get('Set-Cookie')
    print("=========================================================================")
```

결과값

```
Cookie_modulation
C:\Users\1\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/U
접속한 사이트 쿠키 : JSESSIONID=B7B25ADF2F7373DDA452837115BFE4DD; Path=/

변조된 쿠키: JSESSIONID=E2FEE9E49805B34CD8F34B7E9284FF09; Path=/; Http

Process finished with exit code 0
```

웹 취약점 Cookie modulation 구현을 위한 진단 프로그램 개발

# 개발 환경 및 개발내용(9/12)

## 개발 내용(8/11)

◆ 진단 프로그램 개발(Broken Access Control)

module

```
#A5 : Sensitive Data Exposure(Admin
import requests
from .required.common import *
from .required.decoration import *
from .required.separator import Mo
from concurrent.futures import Pro
```

함수 구현

```
def SenDataEX_Checking(extern_file):
    site = "http://demo.testfire.net"
    SenDataEX_Checking.dict_file = "data/dict/admin.txt"
    max_lines = line_counting(SenDataEX_Checking.dict_file)
    SenDataEX_Checking.process_count = max_process(max_lines)
    list = [None]*(max_lines)
    upper_decoration(ModuleNumber.DATAEXPOSURE, extern_file)
    file_to_list(ModuleNumber.DATAEXPOSURE, list, site, SenDataEX_Checking.dict_file)
    max_lines = len(list) // SenDataEX_Checking.process_count
    chunked_list = [list[i * max_lines:(i + 1) * max_lines] \
                    for i in range(SenDataEX_Checking.process_count)]
    with ProcessPoolExecutor(max_workers=SenDataEX_Checking.process_count) as pool:
        for i in range(SenDataEX_Checking.process_count):
            [pool.submit(page_request, ModuleNumber.DATAEXPOSURE, url, extern_file) for url i
```

결과값

```
- - - - - - Data Exposure - - - - - -

[Data Exposure] http://demo.testfire.net/admin

- - - - - - - - - - - - - - - - - - - -
```

웹 취약점 Broken Access Control 구현을 위한 진단 프로그램 개발

# 개발 환경 및 개발내용(10/12)

## 개발 내용(9/11)

◆ 진단 프로그램 개발(Directory_listing)

module

```
#A6 : Directory Listing
from .required.common import
from .required.decoration im
from .required.separator imp
from concurrent.futures impo
```

함수 구현

```
def dir_listing(url, extern_file):
    dir_listing.dict_file = "data/dict/directorys.txt"
    max_lines = line_counting(dir_listing.dict_file)
    dir_listing.comment_count = 13
    dir_listing.process_count = max_process(max_lines)
    list = [None]*(max_lines - dir_listing.comment_count)
    upper_decoration(ModuleNumber.DIRLISTING, extern_file)
    file_to_list(ModuleNumber.DIRLISTING, list, url, dir_listing.di
    max_lines = len(list) // dir_listing.process_count
    chunked_list = [list[i * max_lines:(i + 1) * max_lines] \
                    for i in range(dir_listing.process_count)]

    with ProcessPoolExecutor(max_workers=dir_listing.process_count)
        for i in range(dir_listing.process_count):
            [pool.submit(page_request, ModuleNumber.DIRLISTING, url, extern_file) for url in chunked_list[

    lower_decoration(ModuleNumber.DIRLISTING, extern_file)
```

payloads

```
image002
primary
arrow_red
productsOfTheYear
reset
pivot
memorial
ur
values
xc
locked
strategic
uploaded
capital
atom10
poc
mci
blink
dhs
propecia
```

결과값

```
= = = = = = Directory Listing = = = = = =
[Directory Listing] http://localhost/index
[Directory Listing] http://localhost/test
[Directory Listing] http://localhost/javascript
[Directory Listing] http://localhost/
[Directory Listing] http://localhost/21655
```

웹 취약점 Directory_listing 구현을 위한 진단 프로그램 개발

# 개발 환경 및 개발내용(11/12)

## 개발 내용(10/11)

◆ 진단 프로그램 개발(Backup File exposure)

module

```
#A7 : Backup File Exposure
from .required.common impo
from .required.decoration
from .required.separator i
from concurrent.futures im
```

함수 구현

```
def bf_checking(url, extern_file):
    bf_checking.dict_file = ["data/dict/cfile.txt", "data/dict/cdir.txt",\
                             "data/dict/bfile.txt", "data/dict/bdir.txt"]
    max_lines = line_counting(bf_checking.dict_file[0]) * line_counting(
              + line_counting(bf_checking.dict_file[1]) * line_counting
    bf_checking.process_count = max_process(max_lines)
    list = [None]*(max_lines)
    upper_decoration(ModuleNumber.BFCHECKING, extern_file)
    file_to_list(ModuleNumber.BFCHECKING, list, url, bf_checking.dict_fi
    file_to_list(ModuleNumber.BFCHECKING, list, url, bf_checking.dict_fi
    max_lines = len(list) // bf_checking.process_count
    chunked_list = [list[i * max_lines:(i + 1) * max_lines]\
                    for i in range(bf_checking.process_count)]

    with ProcessPoolExecutor(max_workers=bf_checking.process_count) as p
        for i in range(bf_checking.process_count):
            [pool.submit(page_request, ModuleNumber.BFCHECKING, url, ext
```

결과값

```
= = = = = = Backup File Checking = = = = =
[Backup File Checking] http://localhost/.
[Backup File Checking] http://localhost/.htaccess.backup
[Backup File Checking] http://localhost/.htaccess.org
[Backup File Checking] http://localhost/.htaccess.old
[Backup File Checking] http://localhost/.htaccess.zip
[Backup File Checking] http://localhost/.htaccess.tar
[Backup File Checking] http://localhost/.htaccess.rar
[Backup File Checking] http://localhost/.htaccess.7z
[Backup File Checking] http://localhost/.htaccess.log
[Backup File Checking] http://localhost/.htaccess.!
[Backup File Checking] http://localhost/.htaccess.sql
[Backup File Checking] http://localhost/.htaccess.new
[Backup File Checking] http://localhost/.htaccess.txt
[Backup File Checking] http://localhost/.htaccess.tmp
[Backup File Checking] http://localhost/.htaccess.temp
```

웹 취약점 Backup File 구현을 위한 진단 프로그램 개발

# 개발 환경 및 개발내용(12/12)

## 개발 내용(11/11)

◆ 진단 프로그램 개발(Plain Text Checking)

module

```
#A8 : Plain Text Checking
from urllib.request import urlopen
from dns.resolver import Resolver, NoAnswer, NXD
from scapy.all import *
from re import compile
from .required.common import write_found_data, S
from .required.decoration import *
from concurrent.futures import ProcessPoolExecut
```

함수 구현

```
def parse_domain(site):
    domain = site.replace('http://
    domain = domain.replace('https://', '')
    index = domain.find('/')
    if index > 0:
        domain = domain[:index]
    return domain


def check_dns(dict, domain):
    resolver = Resolver()
    try:
        answer = resolver.query(domai
        for rdata in answer:
            try:
                check_dns.index += 1
            except AttributeError:
                check_dns.index = 0
            dict.append(str(rdata.addr
    except NoAnswer:
        #print("No Answer 1")
        pass
```

결과값

```
::::::Plain Text Checking::::::
[Plain Text Checking] http://127.0.0.1 can't convert to IP Address
::::::::::::::::::::::::
```

웹 취약점 Plain Text 구현을 위한 진단 프로그램 개발

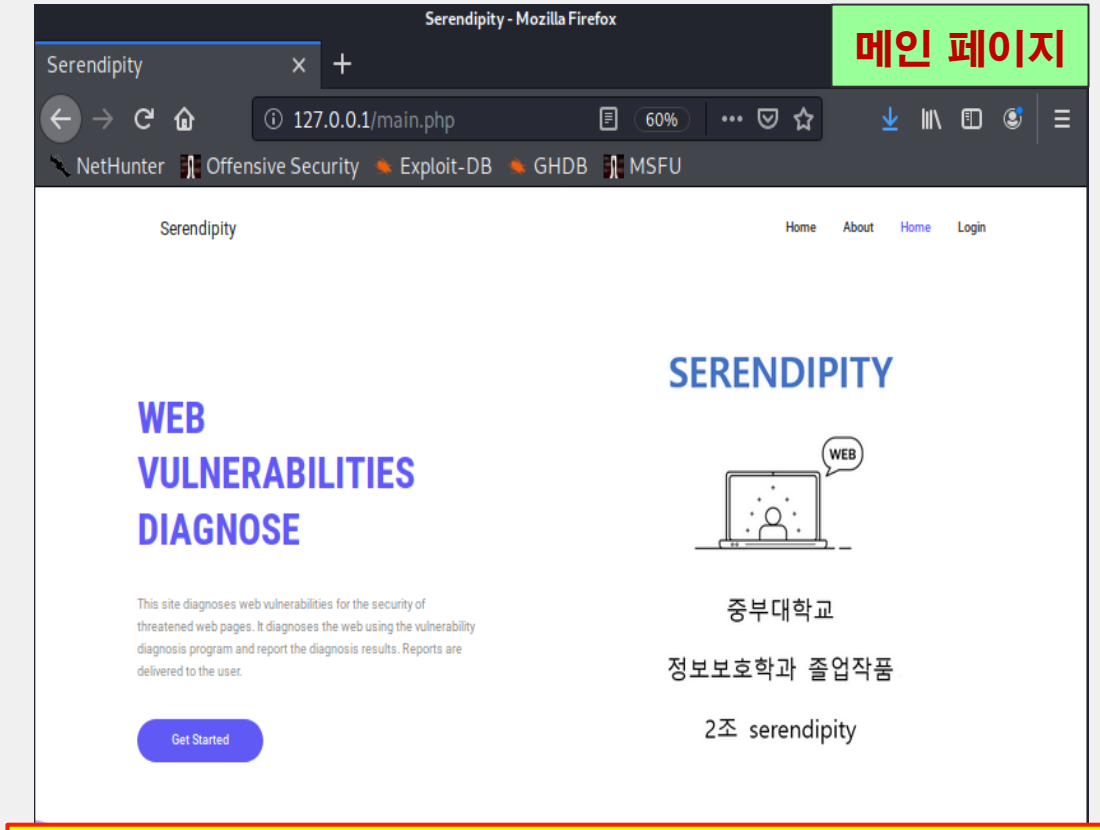
# 개발 시스템 운영(1/5)

## ◆ 웹 페이지 로그인&회원가입

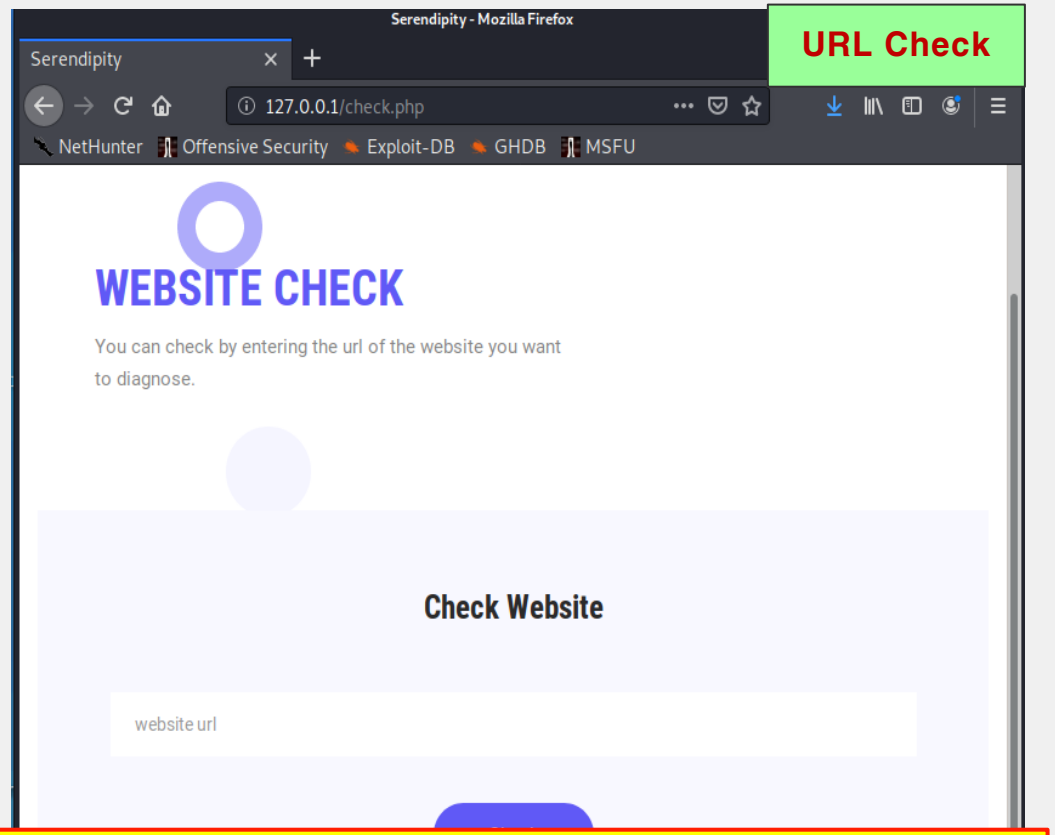

**취약점 진단 서비스를 제공하기 위한 웹 페이지 회원 가입 및 로그인 화면**

# 개발 시스템 운영(2/5)

## ◆ 웹 사이트 메인 페이지 & URL Check 페이지



메인 페이지



URL Check

취약점 진단 서비스를 제공하기 위한 메인화면 및 URL 입력 페이지 제공

# 개발 시스템 운영(3/5)

## ◆ 웹 사이트 실행 결과 확인

셸 번호.1

파일(F) 동장(A) 편집(E) 보기(V) 도움말(H)

취약점 진단 결과

```
root@kali:/var/www/html/python# ./support.sh
* * * This script runs for the Web Vulnerabilities Scanner of Serendipity * * *

1. Kill python processes for the scanner
2. Print the scanning txt lists
3. Print the incomplete quick scanning txt
4. Print last line of the scanning txt
5. Print the scanning full data
6. Print the scanning result
7. Remove the scanning txt files

Select the number : 6

= = = Sorted-Result = = =
SQL Injection     Safe
XSS Injection     Safe
Certification Bypass     Safe
Cookie Modifying     None
Data Exposure     Safe
Backup File Checking     Risk
Plain Text Checking     None
root@kali:/var/www/html/python#
```

셸 번호.1

파일(F) 동장(A) 편집(E) 보기(V) 도움말(H)

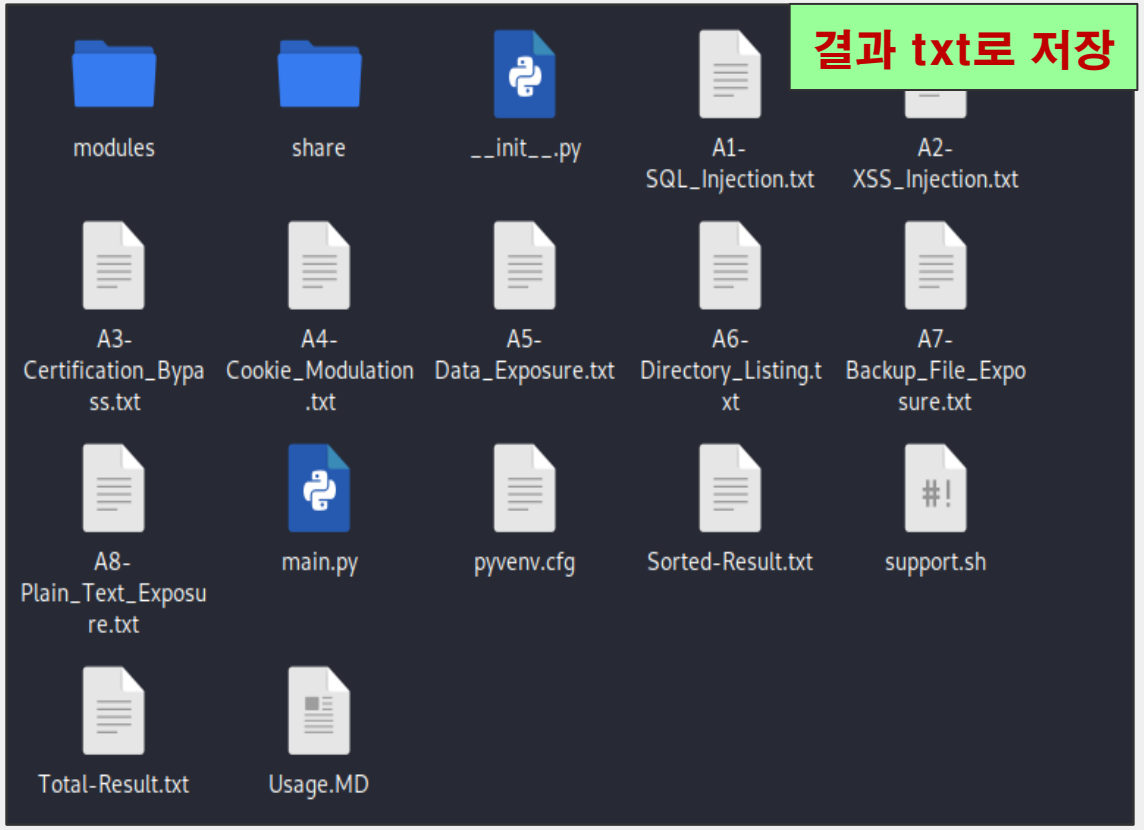
취약점 진단 세부 항목

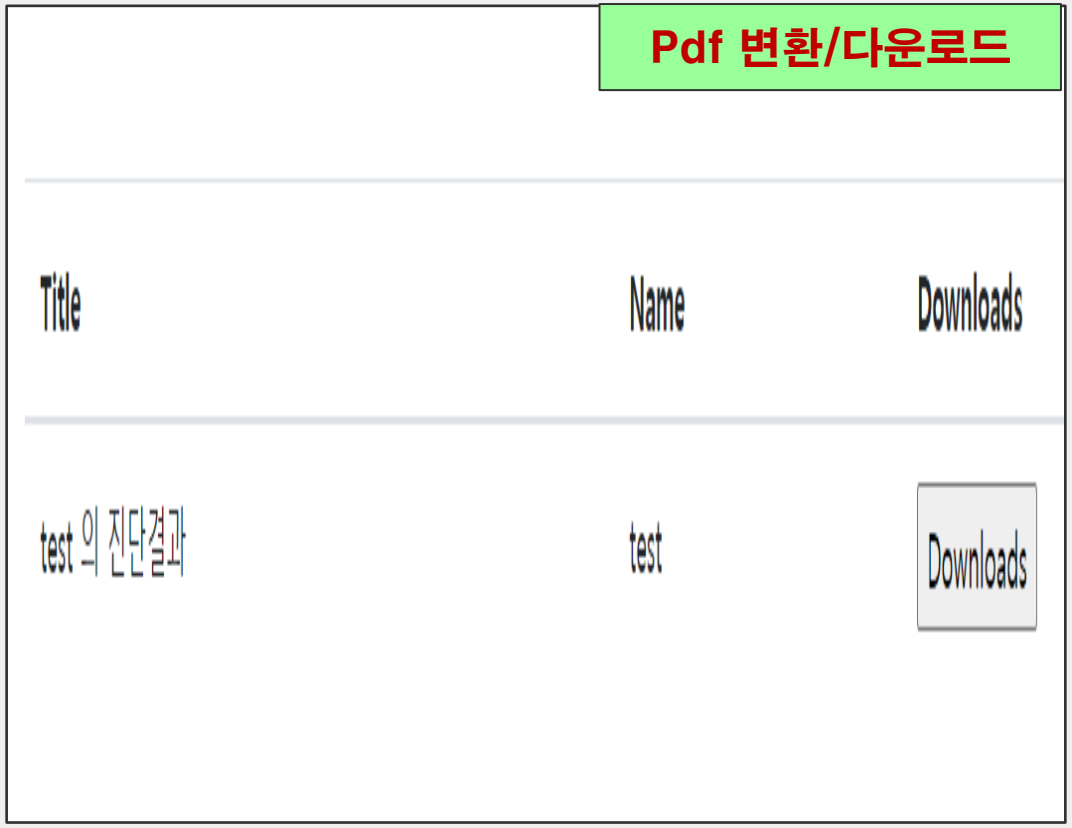
```
1. Kill python processes for the scanner
2. Print the scanning txt lists
3. Print the incomplete quick scanning txt
4. Print last line of the scanning txt
5. Print the scanning full data
6. Print the scanning result
7. Remove the scanning txt files

Select the number : 5

= = = = = = SQL Injection = = = = = =
= = = = = = = = = = = = = = = = = = = =
= = = = = = XSS Injection = = = = = =
= = = = = = = = = = = = = = = = = = = =
= = = = = = Certification Bypass = = = = = =
= = = = = = = = = = = = = = = = = = = =
= = = = = = Cookie Modifying = = = = = =
[Cookie Modifying] http://127.0.0.1 can't find cookie
= = = = = = = = = = = = = = = = = = = =
= = = = = = Data Exposure = = = = = =
= = = = = = = = = = = = = = = = = = = =
= = = = = = Directory Listing = = = = = =
[Directory Listing] http://127.0.0.1/index
= = = = = = = = = = = = = = = = = = = =
= = = = = = Backup File Checking = = = = = =
[Backup File Checking] http://127.0.0.1/.htaccess.backup
= = = = = = = = = = = = = = = = = = = =
= = = = = = Plain Text Checking = = = = = =
[Plain Text Checking] http://127.0.0.1 can't convert to IP Address
= = = = = = = = = = = = = = = = = = = =
root@kali:/var/www/html/python#
```

**웹 사이트 취약점 진단 관리자가 결과 확인**

# 개발 시스템 운영(4/5)

## ◆ 웹 사이트 실행 결과




취약점 진단 서비스를 제공하기 위한 진단 결과 다운로드 화면

# 개발 시스템 운영(5/5)

## ◆ 결과 확인 & 세부정보 확인

진단 리스트

```
SQL Injection      Risk
XSS Injection      Risk
Certification Bypass      Safe
Cookie Modifying      Safe
Data Exposure      Risk
Directory Listing      Risk
BackupFile Checking      Risk
Plain Text Checking      None
```

진단 결과

WebSite Vulnerability Scanner Report

| | Scan Information |
|---|---|
| URL | http://demo.testfire.net/index.jsp |
| TIME | 2020-10-18 16:45 |

List of Vulnerability(8/8)

| TYPE | Description | Safe or Risk |
|---|---|---|
| SQL | Injecting SQL statements to cause databases to behave abnormally | Risk |
| XSS | Attack technique for inserting scripts into websites | Risk |
| Certification | Attack targeted at login page entering ID and password | Safe |
| Cookie | Vulnerabilities in stealing information by falsifying cookie values | Safe |
| Data | Accessing the administrator page by exposing it to the end user. | Risk |
| Directory | The ability to show the directory as a list and file when you connect. | Risk |
| BackupFile | Vulnerabilities to which a directory structure is exposed | Risk |
| Plain Text | Users can obtain data if the data is sent in plain language. | Risk |

웹 사이트 취약점 진단 및 진단 결과 조회 화면

# 결 론 및 기대 효과

## ◆ 결　　론

- OWASP 10에 근거하여 웹 사이트의 취약점을 분석하고 진단하는 웹 취약점 시스템을 개발하는데 성공
- 관리자가 진단한 취약점과 대응방안을 제공하여 안전하고 효율적인 웹 사이트 운영이 가능하도록 지원하는 한편 일반 사용자 또한 취약점에 대응할 수 있게 지원

## ◆ 기대 효과

- 모든 조원들이 시스템 개발 목표를 달성하기 위해 혼연일체가 되어 노력한 결과 팀워크의 중요성을 체험하고 기술 개발역량을 배양하는 계기를 마련
- 진단 프로그램으로 웹 사이트의 취약점을 진단 & 분석, 일반 사용자 또한 침해 대응에 대한 경각심을 일깨우고 대응방법을 통해 보안에 중요성을 인식.  끝.


감사합니다