

모의해킹 보고서

2017/01/23 ~ 2017/01/24

이헌진

목 차

1. 개요

- 1.1 전체 개요
- 1.2 시나리오
- 1.3 서버 구동

2. 점검항목

2.1 인젝션

- 가. 로그인 인증 우회
- 나. SQL Injection

2.2 XSS(Cross Site Scripting)

- 가. 자바스크립트
- 나. iframe
- 다. 대응 방안

2.3 악성 파일 실행

- 가. 웹쉘 업로드
- 나. 대응 방안

2.4 URL 접근 통제 실패

- 가. 디렉터리 인덱싱
- 나. 대응 방안

2.5 불안전한 직접 개체 참조

- 가. 파일 및 디렉터리 추측 가능
- 나. 대응 방안

2.6 취약한 인증 및 세션 관리

- 가. 취약한 계정 / 패스워드
- 나. 대응 방안

2.7 CSRF(Cross Site Request Forgery)

- 가. 쿠키/세션값 변조를 통한 인증/권한 우회
- 나. 대응방안

1. 개요

1.1 전체 개요

전체 개요	
시스템 사양	우분투 14.04
IP	192.168.222.132
DB	mysql 5.5.53
Language	php 5.5.9
Server	apache2

표 1 서버 개요

1.2 시나리오

- 가. 인젝션을 통한 계정탈취 가능성
- 나. XSS 공격을 통한 악성코드 삽입 및 악성 홈페이지 이동 가능성
- 다. 사용자 인증 우회 가능성
- 라. 불안정한 URL 통제
- 마. 악성 파일 업로드 가능성

1.3 서버 구동

← → ↻ 127.0.0.1 ☆ 🍪 ⋮

환영합니다

ID :

PW :

로그인

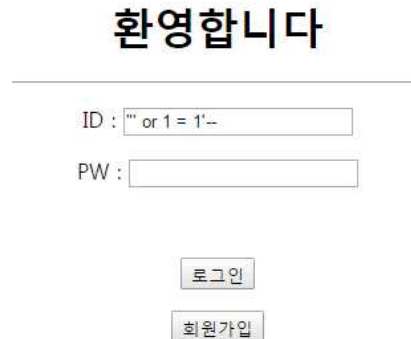
회원가입

그림 1 서버 구동

2. 점검항목

2.1 인젝션 (Injection)

가. 로그인 인증우회



환영합니다

ID :

PW :

그림 2 로그인 인증 우회

- 로그인 인증우회를 시도한 모습

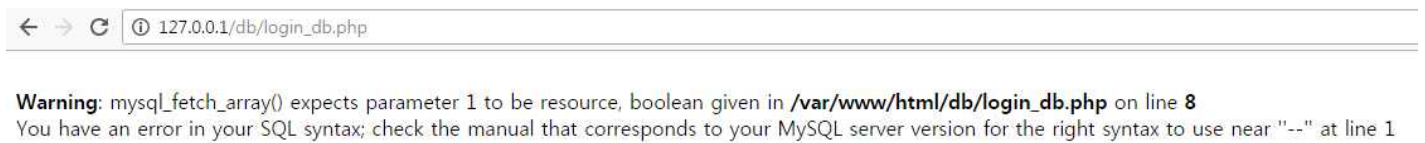


그림 3 로그인 에러 메시지

- 다음과 같은 에러 메시지를 확인 후 조금씩 맞춰볼 수 있음
- /var/www/html/db/login_db.php를 보고 어디를 사용하는지 경로를 알 수 있음



환영합니다

ID :

PW :

그림 4 로그인 인증우회 시도



그림 5 로그인 우회 ID 통과

- 로그인 우회 구문을 에러 메시지를 보고 많은 시도를 했음
- ID는 우회성공

나. SQL 명령어 인젝션 실행

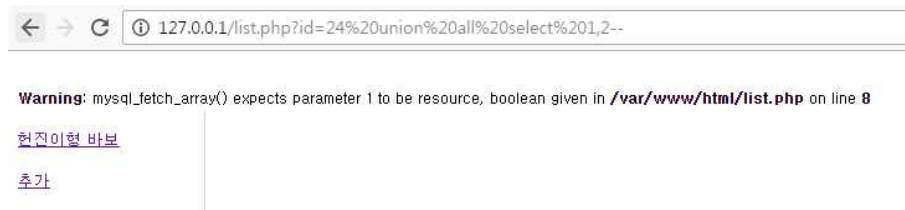


그림 6 union sql injection

- union sql injection (“union all select 1,1 --”, “union all select 1,2 --”)
- 기본 1 = 1 인젝션 (“id=1 and 1#”)
- Blind Injection(SQLmap툴) 공격 시도
- 공격 실패

다. 대응방안

```
1 <?php
2 mysql_connect('localhost','root','asd123');
3 mysql_select_db('sid31337');
4 $id = $_POST['login_id'];
5 $pass = $_POST['login_pw'];
6 $getId = "select id from people where id='$id' ";
7 $getId = mysql_query($getId) or die("No");
8 $getId = mysql_fetch_array($getId) or die(mysql_error());
```

그림 7 die명령



그림 8 die 명령 실행

- die 명령으로 오류메시지를 차단 가능
- “/var/www/html/db/login_db.php”가 안 보이는 것을 확인 가능

```
<?php
mysql_connect('localhost','root','asd123');
mysql_select_db('sid31337');
$id = mysql_real_escape_string($_POST['login_id']);
$pass = mysql_real_escape_string($_POST['login_pw']);
$getId = "select id from people where id='$id' ";
$getId = mysql_query($getId) or die("No");
$getId = mysql_fetch_array($getId) or die(mysql_error());
```

그림 9 mysql_real_escape_string

- mysql_real_escape_string 함수로 SQL인젝션 방지

2.2 XSS(크로스 사이트 스크립팅)

가. 자바스크립트



그림 10 자바스크립트 삽입

- 스크립트 실행여부를 확인하기 위해 자바스크립트 삽입



그림 11 스크립트 실행

- 스크립트가 실행되는 걸 확인



그림 12 document.cookie

- document.cookie 명령으로 현재 자신의 쿠키 값 확인

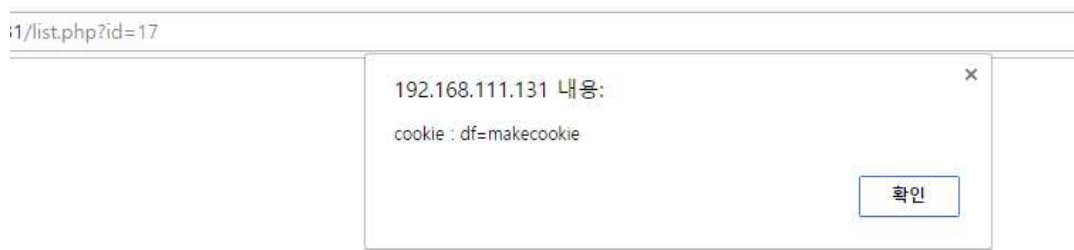


그림 13 쿠키값 확인

- 쿠키값 탈취성공
- 이 원리를 이용해서 이 게시글을 읽은 사용자들 쿠키 값을 수집 가능
- `<script>alert("XSS")</script></code>`
- `"><script>alert("XSS")</script></code>`
- `<script> alert(String.fromCharCode(116,101,115,116))</script></code>`
- 클라이언트 측에서 특정문자열을 걸러낼 수 있으므로 프록시로도 공격 시도
- 다양한 공격을 시도 가능



그림 14 XSS 공격1



그림 15 XSS 공격2

- 해커 페이지에 쿠키 값을 날림
- 관리자가 확인을 하였을 경우

```
root@ubuntu:/var/www/html# cat data.txt
17-01-23 18:14:52 cookie=makemake good
```

그림 16 cookie값

- 다음과 같이 쿠키 값 전송 확인

나. iframe

- html 태그들 중에서 한 가지
- 사용자가 원하는 프레임 안에 다른 사이트를 보여줄 수 있음

제목 : 네이버~

본문 :

```
<iframe src="http://www.dyu.ac.kr"
height="300" width="500"
frameborder="1">No!</iframe>
```

file upload : 선택된 파일 없음

그림 17 iframe



그림 18 iframe 명령 확인

- iframe 명령으로 홈페이지 구동 확인
- 악용 가능성 있음

제목 : 네이버~

본문 :

```
<iframe src="http://www.dyu.ac.kr"
height="0" width="0"
frameborder="0">No!</iframe>
```

file upload : 선택된 파일 없음

- 화면에는 아무것도 보이지 않지만 공격자가 원하는 페이지로 접근 가능

다. 대응방안

```
<?php
mysql_connect('localhost', 'root', 'asd123');
mysql_select_db('sid31337');
switch($_GET['mode']){
    case 'insert':
        $result = mysql_query("INSERT INTO board (title, des, created) VALUES ('
        ".htmlspecialchars($_POST['title'])."', '".htmlspecialchars($_POST['des'])."', now())");
```

그림 20 htmlspecialchars

- htmlspecialchars 명령으로 XSS공격 방어
- PCRE(preg) 정규표현식이나 POSIX(ereg) 정규 표현식으로 방어
- 특수 문자 필터링

2.3 악성 파일 실행

가. 웹 셸 업로드

파일 업로드가 가능한 게시판에 exe나 웹 스크립트 파일이 업로드가 되는지 확인



그림 21 파일 업로드

- php로 된 파일이 업로드가 가능한지 확인
- php 웹 셸을 업로드 시도

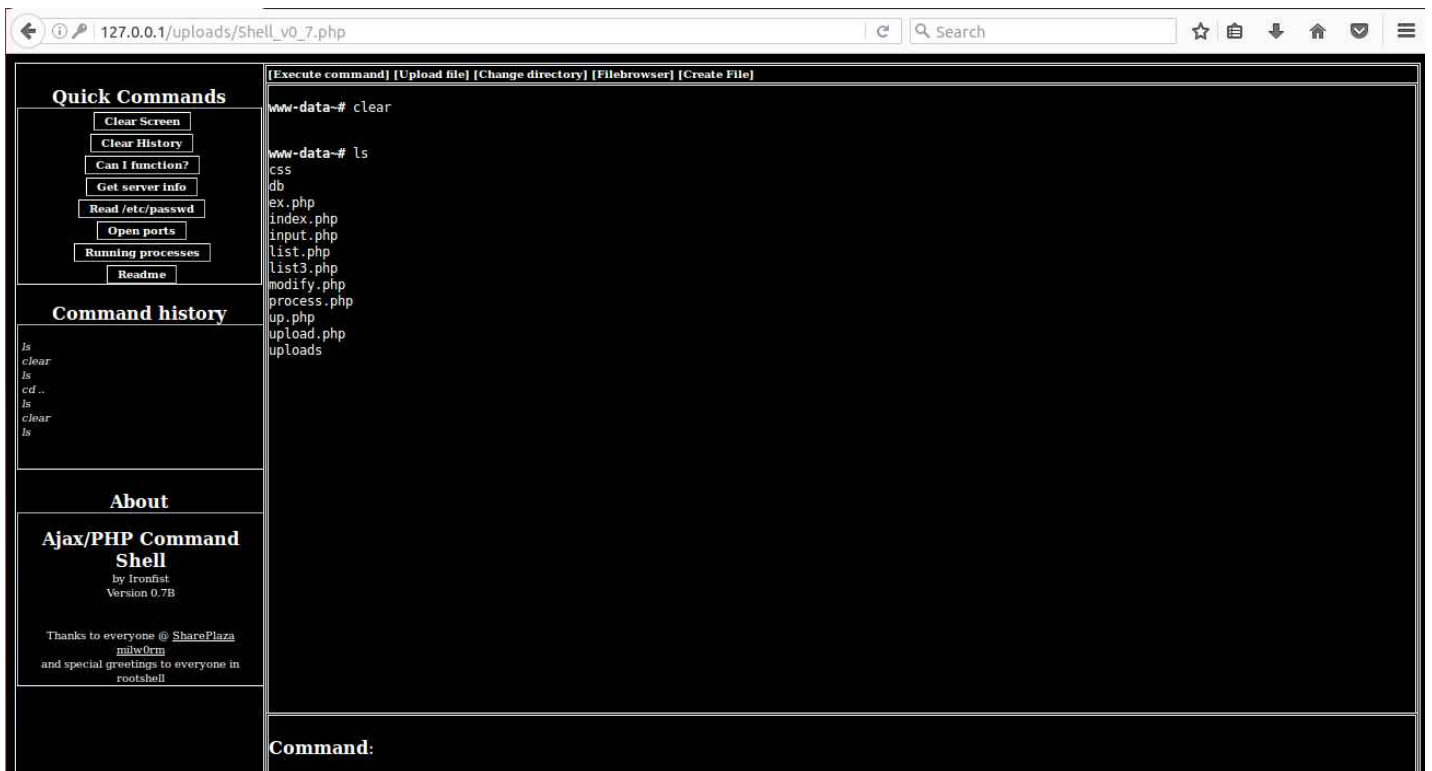


그림 22 웹 셸 업로드 성공

- php로 만들어진 웹 셸이 업로드 되는 것을 확인 가능



그림 23 웹 셸 명령 실행

- 웹 셸에서 명령어도 잘 동작하는 것을 확인 가능

나. 대응방안

1. 필요한 확장자만 사용할 수 있도록 체크

```
$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
" && $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
```

그림 24 확장자 체크

2. 특수문자 필터링
3. 업로드 폴더의 실행 권한 제거
4. 파일이름을 난수화 하여 저장

```
if (move_uploaded_file($_FILES["fileUpload"]["tmp_name"], $target_dir.md5(ba
sename($_FILES["fileUpload"]["name"])))) {
    echo "The file ". basename( $_FILES["fileUpload"]["name"]). " has been u
ploaded.";
```

그림 25 파일명 난수

```
root@ubuntu:/var/www/html/uploads# ls
21d70cd61b9d4c6eb244a48288b78ff4  shell_v0_7.php
```

그림 26 파일명 md5 해시화

2.4 URL 접근 통제 실패

가. 디렉터리 인덱싱



그림 27 디렉터리 인덱싱

- 경로 내에서 디렉토리에 어떤 파일이 있는지 확인가능

나. 대응 방안

```
<Directory /var/www/>
#       Options Indexes FollowSymLinks
       AllowOverride None
       Require all granted
</Directory>
```

그림 28 디렉토리 리스팅

- /etc/apache2/apache2.conf 설정을 변경
- Options를 찾아서 Indexes 제거



그림 29 디렉토리 리스팅

2.5 불안정한 직접개체 참조

가. 파일 및 디렉터리 추측가능



그림 30 list.php

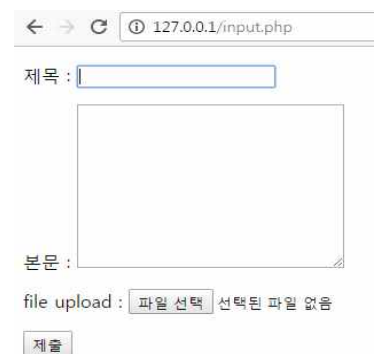


그림 31 input.php

- 현재 수정 및 삭제 항목이 보이지 않음
- list.php, input.php 리스트랑 삽입하는 php인걸 확인 가능



그림 32 modify.php

- 추측으로 list.php를 modify.php로 바꾸니 수정가능
- 권한이 없는데 권한확인을 하지 않으므로 접근 가능

나. 대응 방안

- 권한이 없을시 접근 차단
- 추측하기 어려운 이름으로 사이트 생성

2.6 취약한 인증 및 세션관리

가. 취약한 계정 / 패스워드

```
POST /db/login_db.php HTTP/1.1
Host: 127.0.0.1
Content-Length: 32
Cache-Control: max-age=0
Origin: http://127.0.0.1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML,
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0
Referer: http://127.0.0.1/
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: cookie=make make good; PHPSESSID=90fsb8hjdi1t2n3isb36j83f4
Connection: close

login_id=gjswls1&login_pw=asd123
```

그림 33 Burp Suite

- 로그인시 아이디랑 패스워드가 평문으로 전송되는 것을 확인 가능

나. 대응방안

- 아이디와 패스워드를 해시화후 전송(SHA, md5)

2.7 CSRF(Cross Site Request Forgery)

가. 쿠키/세션값 변조를 통한 인증/권한 우회



그림 34 CSRF

- 관리자가 게시물을 작성했는데 별로 마음에 안 듦

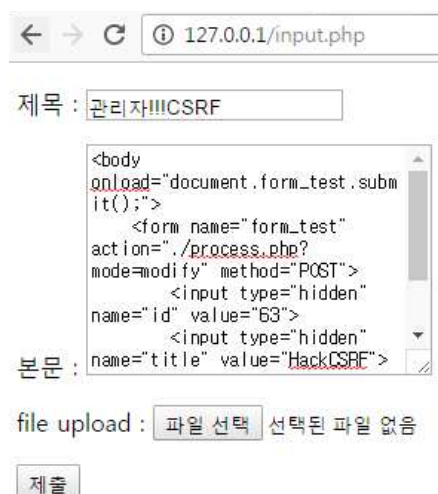


그림 35 CSRF

- 관리자가 볼 수 있게 게시글 작성
- body onload 게시 글을 읽는 순간 form_test가 submit 됨



그림 36 CSRF

- 관리자가 '관리자!!!CSRF' 글을 읽는 순간 관리자 권한으로 코드가 실행 됨
- 관리자 공지 글이 해커가 원하는 코드로 변조

나. 대응 방안

- CSRF공격은 XSS공격과 방법이 유사함
- XSS 대응 방안을 실천하면 CSRF 공격도 방어

3. OWASP ZAP

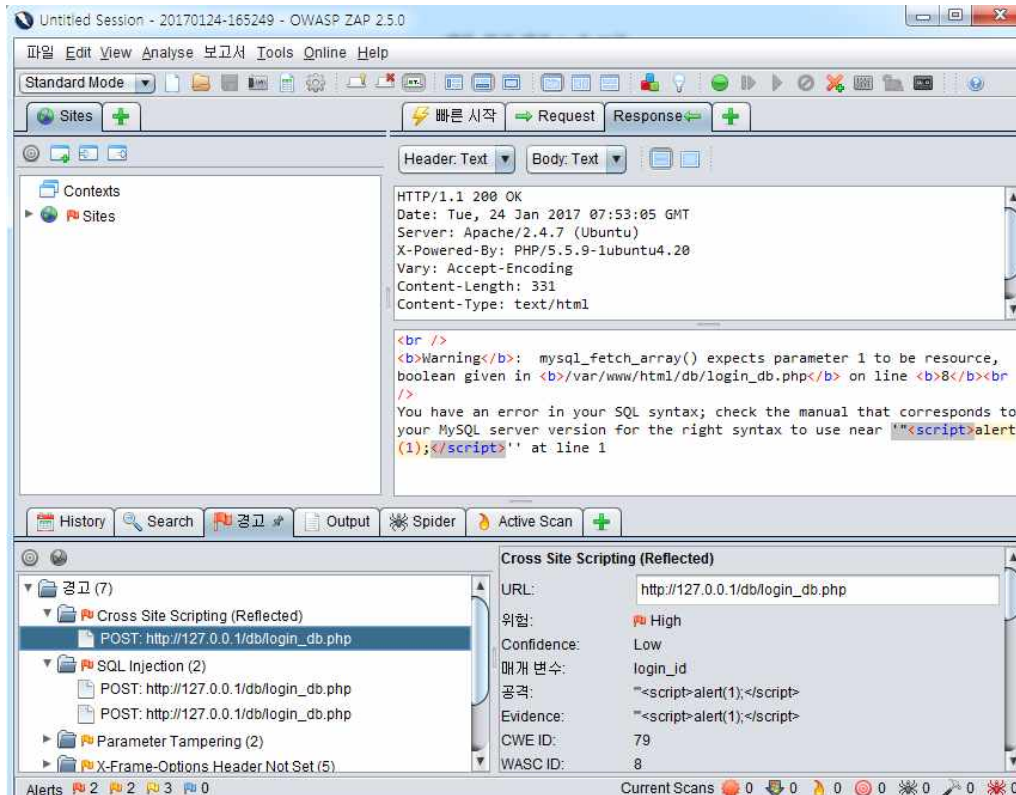


그림 37 OWASP ZAP

- OWASP ZAP은 OWASP에서 만든 웹 보안 취약점 점검 툴
- 게시판을 스캔을 해보니 XSS와 SQL 인젝션 취약점 존재 가능성이 존재한다고 알려줌
- Fuzz공격을 시도했으나 공격실패(인젝션, XSS)