크로스 사이트 요청 변조(CSRF) 공격

목차

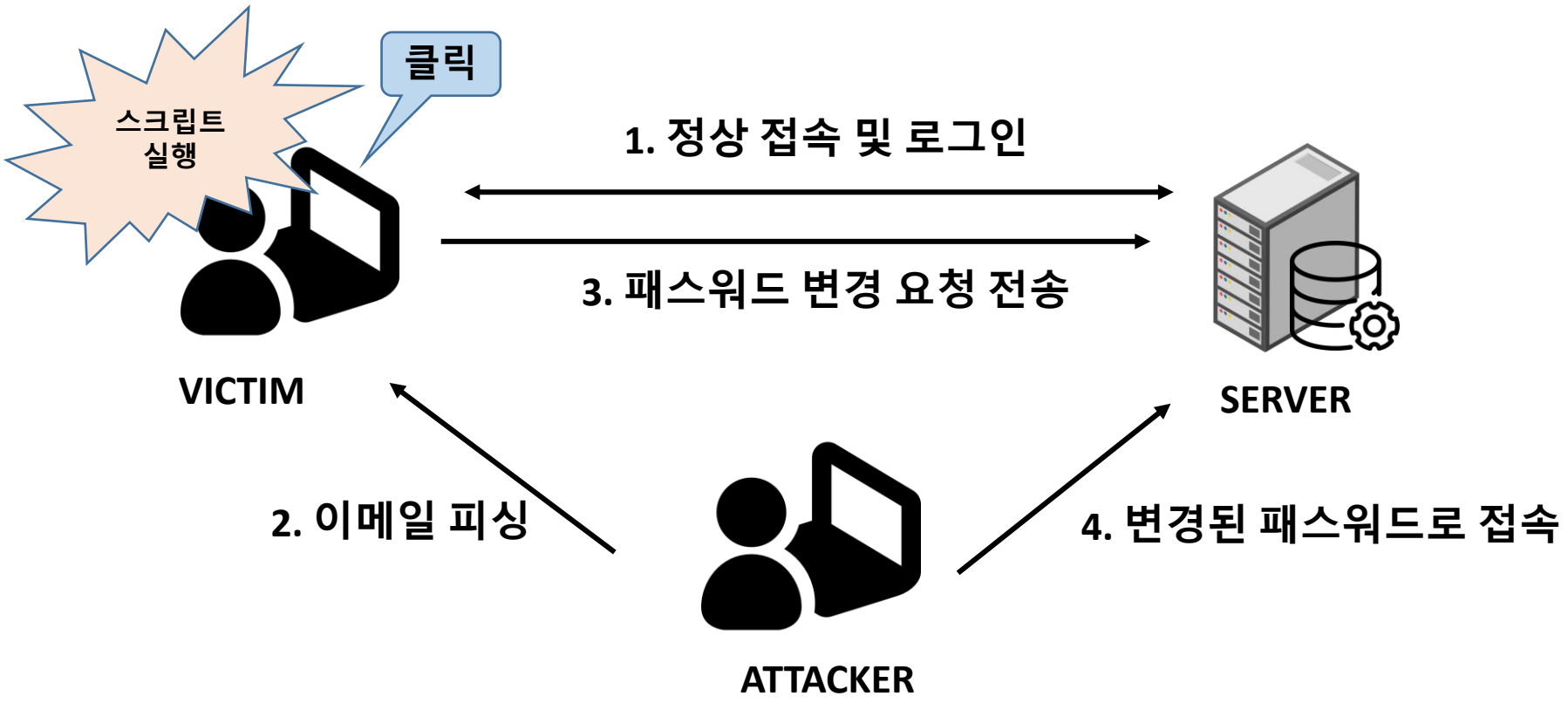
[크로스 사이트 요청 변조 공격]

What is Cross Site Request Forgery(CSRF)

- 크로스 사이트 요청 변조는 Cross Site Request Forgery(CSRF) 로 나타냄
- 많은 개발자들이 CSRF 취약점에 대해 잘 알지 못하고 애플리케이션을 개발하기 때문에 여전히 CSRF 취약점이 많이 발견
- CSRF 취약점을 이용한 공격은 2008년 1080만 명의 개인정보가 유출된 옥션 해킹 사건의 공격 방법 중 하나
- CSRF 취약점은 공격자가 피싱을 이용하여 공격 대상이 되는 사용자에 악성 링크를 누르게 하고, 링크를 클릭하면 사용자 모르게 사용자가 로그인되어 있는 웹 사이트의 어떤 기능을 실행
- 예상 시나리오 하나를 예로 들면, 공격자가 한 사용자의 패스워드를 본인이 모르는 사이에 변경한 다음, 변경된 패스워드를 이용하여 웹 사이트에 접속

What is Cross Site Request Forgery(CSRF)

- CSRF 취약점은 크로스 사이트 스크립트와 이름의 일부가 동일하고 두 공격 모두 공격 과정에서 피싱을 이용하는 특성 때문에 리플렉티드 크로스 사이트 스크립팅 취약점과 혼동할 수 있지만 피싱 이후 진행되는 과정이 다름



What is Cross Site Request Forgery(CSRF)

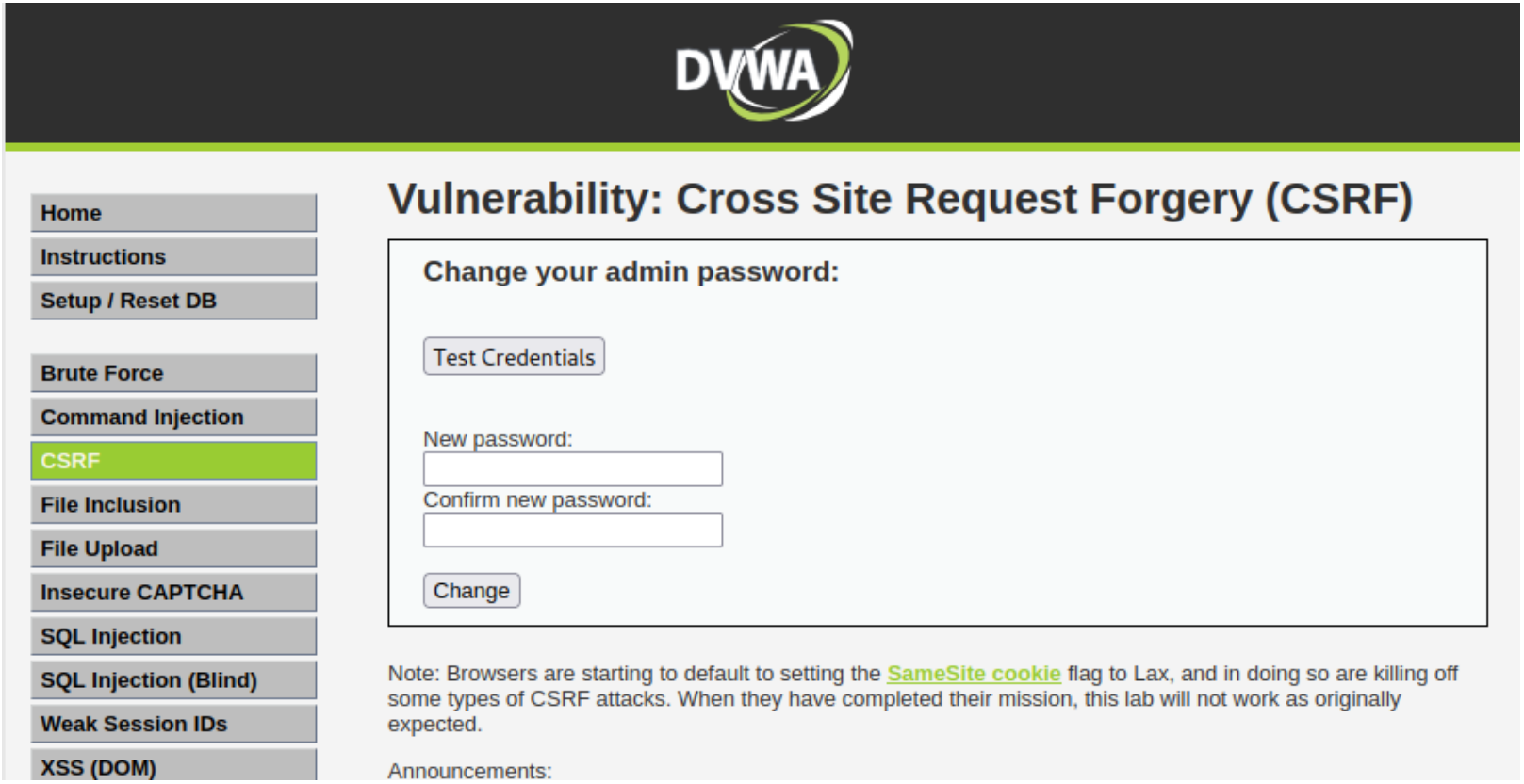
- CSRF 공격을 이용하여 사용자의 패스워드 변경 과정을 살펴보면,
- 1. 사용자가 웹 사이트에 정상적으로 접속하여 로그인
- 2. 사용자가 웹 사이트에 로그인되어 있는 동안, 공격자가 이메일을 보내 악성 코드를 포함하고 있는 페이지를 여는 링크를 클릭하도록 피싱
- 3. 만약 사용자가 링크를 클릭하면, 공격자가 지정한 패스워드로 변경하는 요청이 자신이 이전에 로그인되어 있던 웹 사이트로 자동으로 전송되고, 그 결과 사용자가 모르는 사이에 자신의 패스워드가 변경
- 4. 공격자는 변경된 패스워드를 이용하여 사용자 계정으로 로그인 실행
- CSRF 공격의 특징은 1번 과정이 필수 조건으로 선행되어야 한다는 점이며, 그래야 3번 과정에서 로그인된 세션 정보가 쿠키 등으로 전달되어 웹 사이트가 요청된 기능을 실행 (1번 조건이 쉽지 않아 보일 수 있음)

What is Cross Site Request Forgery(CSRF)

- 웹 브라우저로 인터넷을 사용할 때를 생각해보면, 대부분의 웹 브라우저가 탭 기능을 지원하고 있고 많은 인터넷 사용자가 이러한 탭 기능을 편리하게 사용하고 있음
- 하나의 탭에서 어떤 웹 사이트를 로그인하여 사용하다가 로그인한 상태로 그대로 둔 채, 새로운 탭을 열어 다른 작업을 하는 경우가 흔함
- 만일 로그인핸 채로 둔 웹 사이트에 CSRF 취약점이 존재하는 경우, 1번 조건이 성립되고 쉽게 CSRF 공격에 당할 수 있음
- 탭 기능을 사용하지 않을 수 없으니 일일이 로그인한 웹 사이트를 로그아웃 하는 것도 번거롭기 때문에 1번 과정의 위험성이 높음
- CSRF 공격을 당하지 않으려면, 피싱 공격의 위험에 항상 주의를 기울여야 하며, 모르는 사용자가 보낸 이메일을 함부로 열지 않거나 이메일 내용이나 게시판 등에서 링크를 누르기 전에 항상 URL을 확인하는 습관을 갖는게 좋음

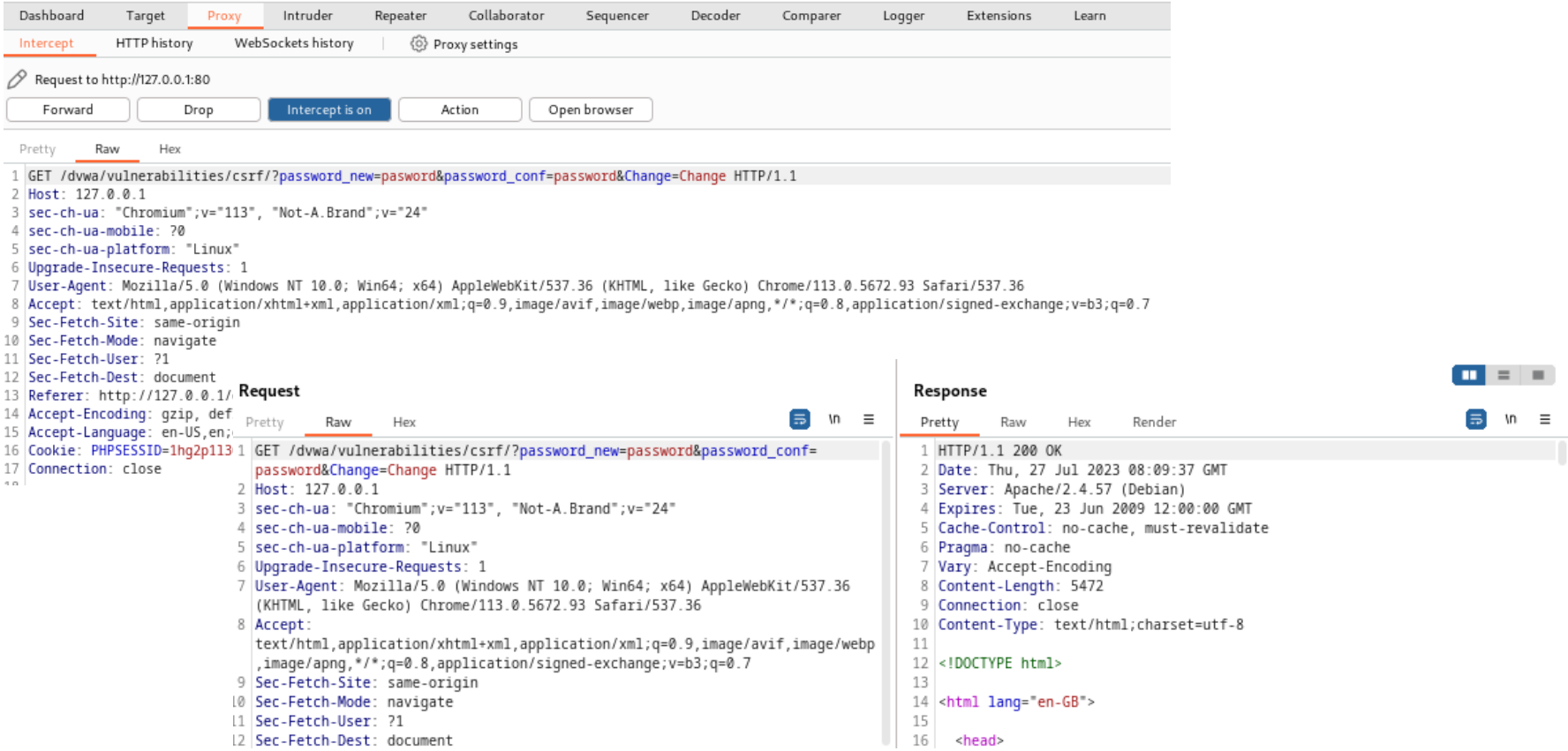
Cross Site Request Forgery(CSRF) Attack Practice

- DVWA의 CSRF 메뉴를 클릭하여 실습 페이지 실행
- [목표] 정상적인 패스워드 변경 페이지를 사용하지 않고 CSRF 공격 POC 코드를 이용하여 사용자의 패스워드를 변경



Cross Site Request Forgery(CSRF) Attack Practice

- 실습 페이지에는 패스워드를 변경하는 폼이 표시되며, password라는 패스워드로 변경 실행
- 패스워드를 변경할 때 아래와 같은 요청 메시지가 전송되는 것을 버프스위트의 프록시 히스토리 기능에서 확인



Cross Site Request Forgery(CSRF) Attack Practice

- 요청 메시지 내용을 살펴보면, GET 메소드로 요청이 전송되며 password_new, password_conf 파라미터에 입력한 패스워드가 전달되고, Change 파라미터에는 Change라는 값이 전달
- 쿠키 헤더에는 PHPSESSID 쿠키와 몇몇 다른 쿠키가 전달되고 있음
- 즉, PHPSESSID 쿠키만 랜덤한 값을 가지고 있을 뿐, 나머지 파라미터 및 헤더 정보 등은 모두 지정된 값을 가짐
- 따라서 공격자는 사용자의 랜덤한 PHPSESSID 쿠키 값만 알아내거나 전달시킬 수 있으면 위의 파라미터 값들을 포함한 요청 메시지를 생성하여 패스워드를 변경할 수 있음
- 쿠키를 전달시키 위한 방법으로 CSRF 공격을 사용

Cross Site Request Forgery(CSRF) Attack Practice

- 사용자가 로그인되어 있을 때에는 웹 페이지를 요청할 때마다 쿠키가 웹 브라우저에 의해 자동으로 전달
- 따라서 공격자가 피싱 공격이나 다른 사회 공학 기법을 이용하여 해당 사용자로 하여금 로그인된 웹 사이트의 링크를 누르게 만들면 쿠키를 전달시킬 수 있음
- 이러한 이유로 인해 사용자 로그인되어 있어야 하는 것이 CSRF 공격의 필수 조건으로 로그인이 되어 있지 않으면 쿠키가 전달되지 않아서 공격이 성공할 수 없음
- (팁) 어떤 정보를 변경하는 요청 메시지(예. POST 메소드 요청)에 쿠키 이외에는 랜덤한 값이 없다면 CSRF 공격에 취약할 수 있음

Cross Site Request Forgery(CSRF) Attack Practice

- DVWA 실습 페이지를 대상으로 CSRF 공격을 수행하는 HTML 파일을 활용하여 공격 실습 (csrf.html 파일)

```
<!--
 PoC: CSRF
 Author: stayp05 (www.secuacademy.com)
-->

<html>
<meta charset="UTF-8">
<head>
</head>

<script language="javascript">

 function poc() {

  var host='localhost';

  var req_uri = "http://" + host +
  "/dvwa/vulnerabilities/csrf/?password_new=hacker&password_
  conf=hacker&Change=Change";
  var xmlhttp = new XMLHttpRequest();
```

Cross Site Request Forgery(CSRF) Attack Practice

- POC 코드 중에서 크게 주목해야할 부분은 11~28번째 줄의 스크립트 부분과 이 스크립트를 실행시키기 위한 30~34번째 줄의 <body>태그 부분
- 스크립트 부분에서 poc()라는 함수를 정의하고 있는데, 17번째 줄에서 요청처럼 URL과 파라미터를 똑같이 구성
- 이때 패스워드 관련 파라미터는 test 대신 hacker라는 문자열로 변경했는데, CSRF 공격이 성공하면 패스워드 hacker로 변경됨
- 18~23번째 줄에서 자바스크립트의 XMLHttpRequesst()를 사용한 AJAX 기번을 이용하여 poc()가 호출될 때 새로운 요청이 전송되도록 만드는데, 그중에서 특히 21번째 줄에서 withCredentials 속성을 true로 설정하여 요청이 전송될 때 웹 브라우저가 쿠키를 자동으로 같이 전송하도록 만듦
- 31~33번째 줄의 바디 부분에는 피싱을 위한 문구를 삽입하고, 피싱을 당한 사용자가 Click! 링크를 누르면 poc() 함수가 실행되도록 만듦

Cross Site Request Forgery(CSRF) Attack Practice

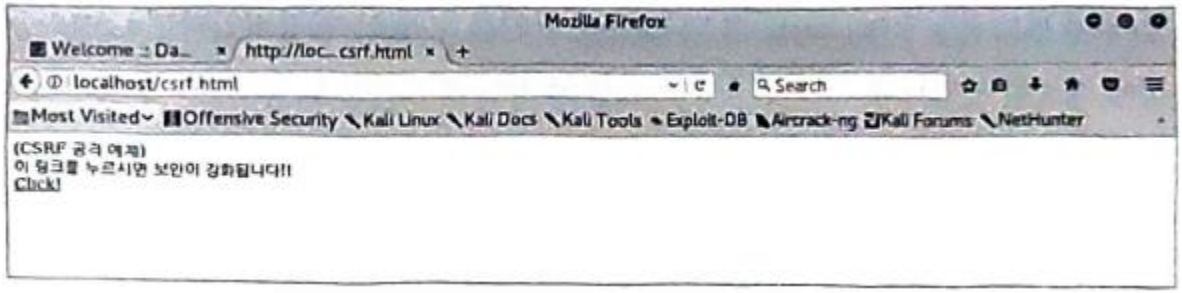
- 정리하면, 공격자가 피싱을 하여 사용자가 위 html 파일 안의 링크를 누르게 되면, 자동으로 패스워드를 변경하는 CSRF 공격 요청이 웹 애플리케이션으로 전달
- 사용자가 DVWA에 로그인되어 있는 경우 쿠키도 자동으로 포함되어 전송되기 때문에 웹 애플리케이션은 사용자가 로그인되어 있는 것으로 파단하고 요청을 처리하게 되어 패스워드가 변경
- 이 POC 코드 HTML 파일을 칼리 리눅스 웹 디렉터리에 올려서 CSRF 공격 실습을 진행하기 위해 vi 에디터 프로그램을 사용하여 csrf.html 파일을 열고 15번째 줄의 host를 DVWA의 IP 주소로 변경

Cross Site Request Forgery(CSRF) Attack Practice

- 변경한 csrf.html 파일을 웹 디렉터리인 /var/www/html/로 옮긴 다음 아파치 웹 서버를 실행하여 csrf.html을 웹으로 접근할 수 있도록 만듦

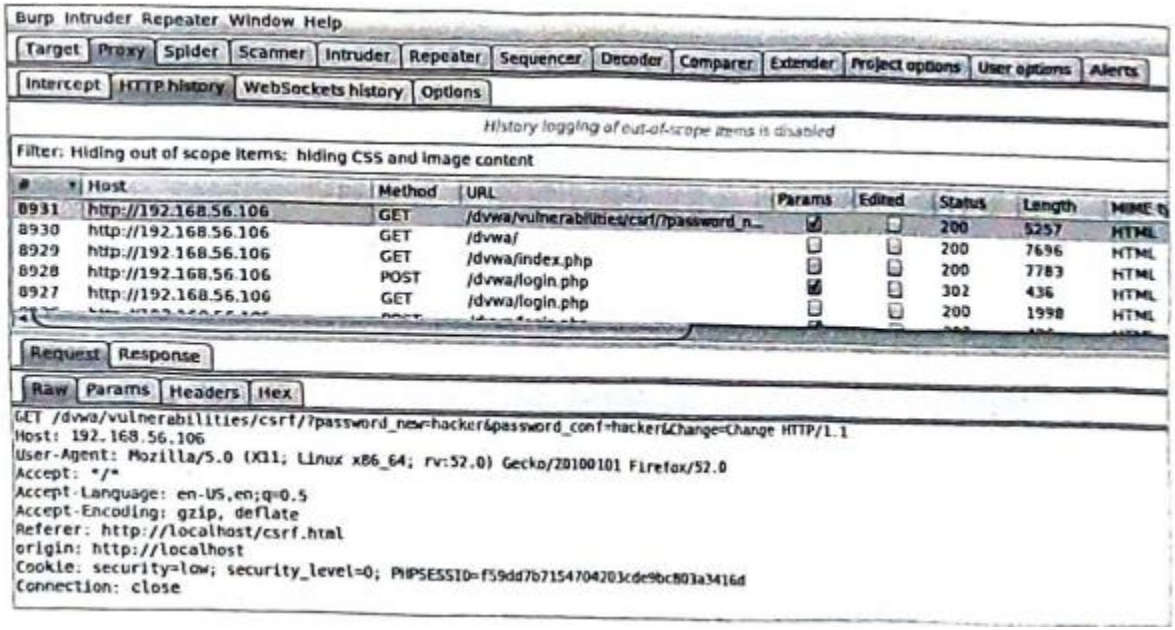
```
root@kali:~/Downloads# mv csrf.html /var/www/html/
root@kali:~/Downloads# service apache2 start
```

- 이제 공격을 당하게 될 사용자 입장에서 웹 브라우저를 실행하여 DVWA에 로그인 수행 (로그인 안될 경우 처음 접속시 비밀번호 변경한 건지 확인)
- 이후에 새로운 탭을 열고 아래 주소를 입력하여 csrf.html 파일을 열어 사용자가 피싱을 당해 csrf.html 파일을 열게 되었다고 가정함
- http://localhost/csrf.html



Cross Site Request Forgery(CSRF) Attack Practice

- Click! 링크를 누르면 공격이 실행되고, 공격이 실행될 때 화면에 표시되는 Done(완료)이라는 메시지 창이 나타남
- 버프 스위트의 프록시 히스토리를 보면 아래와 같이 패스워드를 hacker로 변경하는 요청이 전송된 것을 확인하며 PHPSESSID도 함께 전송



Cross Site Request Forgery(CSRF) Attack Practice

- 이때의 응답 코드(Status 칼럼)가 200으로 표시되었다면 패스워드가 정상적으로 변경
- 웹 브라우저에서 DVWA에 로그인했던 탭으로 돌아가서 로그아웃 한 후에 패스워드가 hacker로 변경되었는지 확인
- 변경되었다면 CSRF 공격에 성공

Cross Site Request Forgery(CSRF) Attack Defense

- CSRF 공격에 대응하기 위해서는 아래와 같은 두 가지 방법이 일반적
- 1. 요청 메시지의 레퍼러(Referer) 헤더를 검사하여 웹 메일이나 타 사이트에서 피싱을 당해 전송되는 요청을 실행하지 않는다. 레퍼러 헤더는 해당 요청을 링크하고 있던 이전 웹 페이지의 주소를 알려주는 헤더이다. 예를 들어, 레퍼러 헤더의 값이 http://localhost/csrf.html이 라고 되어 있다. 우리가 실습할 때 해당 경로의 파일로부터 링크를 클릭했기 때문이다. 만일 피싱에 의한 것이 아니라 정상적인 경로로 패스워드 변경 요청이 되었다면 레퍼러 헤더에는 DVWA 사이트 내부의 URL이 표시되었을 것이다. 따라서 레퍼러 헤더를 확인하여 요청을 전송시킨 출처 페이지를 확인한 후 정상적인 요청인지 판별

Cross Site Request Forgery(CSRF) Attack Defense

- CSRF 공격에 대응하기 위해서는 아래와 같은 두 가지 방법이 일반적
- 2. CSRF 토큰을 사용한다. 쿠키 외에 공격자가 추측할 수 없는 값이 요청 메시지의 파라미터 등에 포함되어 있다면 공격자는 CSRF 공격을 성공시키기 어려워진다. 이때 CSRF 공격 대응을 위해 포함시키는 랜덤한 값을 CSRF 토큰이라고 한다. CSRF 토큰을 이용한 대응 방식은 다음과 같이 구현할 수 있다.
  - 먼저 웹 애플리케이션이 CSRF 토큰을 매 응답마다 랜덤하게 생성하여 히든 폼 필드 등을 통해 클라이언트로 보낸다.
  - 클라이언트는 이전 응답 메시지에 포함된 토큰 값을 다음 요청 시 포함시켜 전송한다.
  - 웹 애플리케이션이 CSRF 토큰 값을 검사하면 정상적인 과정을 통해 전달된 요청인지 확인할 수 있다.

Cross Site Request Forgery(CSRF) Attack Defense

- 보안 레벨을 하이 단계로 설정하면 CSRF 토큰이 어떻게 사용되는지 직접 확인할 수 있음
- 하이 단계의 실습 페이지에 접속한 후 소스코드를 보면 user_token이라는 히든 타입의 폼 필드가 있음

Cross Site Request Forgery(CSRF) Attack Defense

- 70번째 줄을 보면 user_token이 표시됨
- Value로 지정 문자열이 웹 애플리케이션이 랜덤하게 생성한 CSRF 토큰 역할
- 이제 패스워드 변경 요청을 보내면 CSRF 토큰이 user_token 파라미터를 통해 전달되게 됨