웹 해킹

32기 문서현

서론

해킹공부를 하고 싶었는데 어디에서부터 시작해야될 지 몰라서 해킹 분야 중 가장 배워보고 싶었던 웹 해킹 분야를 먼저 공부해보기로 하였다. 우선 이번 SSR에서는 드림핵 커리큘럼을 따라 웹 해킹의 기본 개념에 대해 학습한 뒤 드림핵에서 나온 것을 따라해보고 문제를 풀어보려고 한다.

웹 해킹은 웹 어플리케이션이나 웹 서버 등 웹 서비스 상에서 발생할 수 있는 모든 보안 취약점을 이용하여 악위적인 행위를 하는 것을 말한다. 특히 HTTP 프로토콜의 기본 포트번호인 80번을 통해 대부분의 보안 장비가 웹 서버 접근을 허용하기 때문에 쉽게 공격할 수 있다.

본론

1-1. 웹

웹은 인터넷이라는 통신망으로 구현된 전 지구적 정보공간이고, http를 이용하여 정보를 공유하는 서비스를 말한다.
이때, 정보를 제공하는 주체는 웹 서버, 정보를 받는 이용자는 웹 클라이언트라고 한다.

1-2. 웹 리소스

웹 리소스는 웹에 갖춰진 정보자산을 말한다. 모든 웹 리소스는 고유의 URI(Uniform Resource Identifier)을 갖고 이것을 이용하여 식별할 수 있다.
웹 리소스에는 HTML, CSS, JS(JavaScript) 등이 있다.
HTML은 태크와 속성으로 문서를 구조화시킬 수 있다. 그리고 CSS는 웹 문서의 생김새를 지정하여 시각화시킬 수 있는 것이고 JS는 웹 문서의 동작을 구현할 수 있다.

1-3. 웹 서비스

웹 서비스는 요청을 처리하는 부분인 백엔드, 요청을 받는 부분인 프론트엔드로 나뉘는 데 여기서 프론트엔드는 직접 보여지는 부분으로 웹 리소스로 구성된다.

1-4. 웹 서비스의 통신과정

클라이언트) 클라이언트가 브라우저를 이용하여 웹 서버에 접속.

↓

클라이언트) 이때 브라우저가 클라이언트의 요청을 해석하여 http형식으로 웹 서버에 리소스를 요청.

↓

서버) http로 전달된 클라이언트의 요청을 해석.

↓

서버) 해석된 요청에 따라 동작.(리소스를 요청하면 탐색, 입금/송금 등을 할 때는 내부적으로 연산 진행)

↓

서버) 클라이언트에게 http형식으로 리소스 전달

↓

클라이언트) 브라우저는 서버에게 받은 리소스를 시각화하여 클라이언트한테 보여줌

2-1. 인코딩 <-> 디코딩

인코딩은 사람이 인지할 수 있는 언어를 약속된 규칙에 따라 컴퓨터가 이해할 수 있는 0,1로 바꾸는 것을 말한다.

인코딩에는 아스키코드와 유니코드 등이 있다. 아스키코드는 7비트로 알파벳, 특수 문자 등을 표현할 수 있다. 하지만 아스키코드는 영어권에서 사용하기 위해 설계되었기 때문에 다른 언어를 지원하지 못했다. 그래서 유니코드가 새로 만들어졌다. 유니코드는 모든 언어의 문자를 담겠다는 목표로 만들어졌다. 유니코드는 현재 32비트로 영어, 한글, 한자, 히라가나 등 전 세계의 문자를 표현하고도 남을 정도이다.

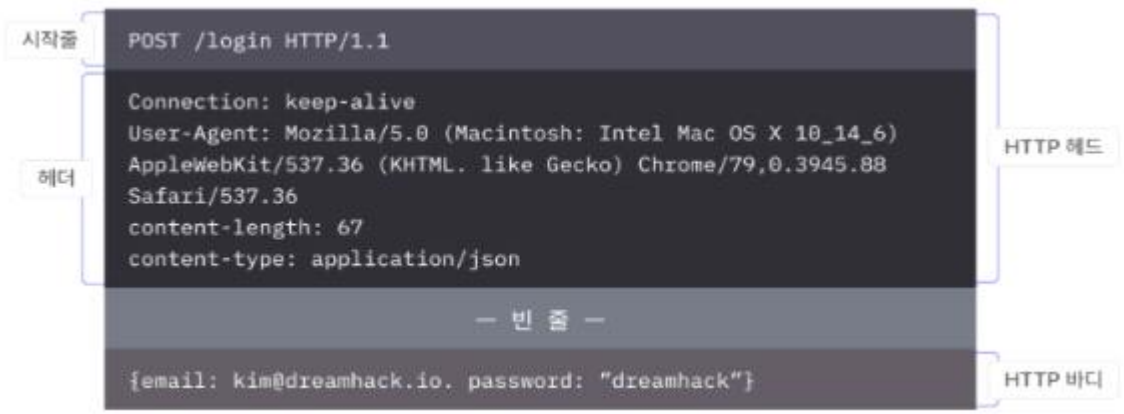
디코딩은 인코딩과 반대되는 개념으로 0,1을 사람이 이해할 수 있는 언어로 바꾸는 것을 말한다.

2-2. 통신 프로토콜

프로토콜은 규격화된 상호작용에 적용되는 약속을 말한다. 각 통신 주체가 교환하는 데이터가 명확히 해석될 수 있도록 문법이 포함되어있다.

2-3. http

http는 프로토콜 중에 하나로 서버와 클라이언트의 데이터 교환을 요청과 응답 형식으로 정의한 것을 말한다. http의 기본적으로 클라이언트가 서버에게 요청하면 서버가 응답하는 형식을 따른다.



http는 위 사진과 같이 **http 헤더**와 **http바디**로 구성되어있다. **http 헤더(Headers)**의 각 줄은 CRLF(줄바꿈)으로 구분된다. 이때, 첫 번째 줄은 시작 줄(Start line)이라고 하고, 나머지 줄은 헤더(Header)라고 한다. 그리고 **http 바디**는 헤드의 끝을 나타내는 CRLF 뒤의 모든 줄을 말한다. 바디에는 클라이언트나 서버에게 전송하려는 데이터가 담겨있다.

2-4. http 요청

http 요청은 서버에게 동작을 요구하는 메세지이다. 이때, 서버는 해당 동작이 실현 가능한지, 클라이언트가 동작을 요청할 권한이 있는 지에 대해 검토하고 적절할 때 이를 처리한다.

http 요청에서 시작줄은 메소드와 요청대상, http의 버전으로 구성되고, 각각을 띄어쓰기로 구분한다. 메소드는 요청 대상에 대하 서버가 수행하길 바라는 동작을 말하고 http표준으로

정의된 메소드는 get, head, post, put, delete, connect, options, trace, patch 총 9개가 존재한다.

get	리소스 조회	head	리소스 조회(상태줄, 헤더만)
post	요청데이터 처리	options	통신가능 옵션 설명
put	리소스 덮여쓰기or 리소스 생성	connect	서버에 대한 터널 설정
patch	리소스 부분 변경	trace	루프백 테스트 수행
delete	리소스 삭제		

2-5. http 응답

http 응답은 http 요청에 대한 결과를 반환한다. 요청을 수행했는지/안 했는지/왜 안했는지와 같은 상태 정보와 클라이언트에게 전송할 리소스가 http 응답에 포함된다.

http 응답의 시작줄은 http 버전, 상태 코드, 처리 사유로 구성되고, 각각을 띄어쓰기로 구분한다. 여기에서 http 버전은 서버에서 사용하는 http 프로토콜 버전을 말한다. 그리고 상태코드는 요청에 관한 처리 결과를 세 자릿수로 나타내고, 처리 사유는 상태 코드가 발생한 이유를 짧게 기술한 것을 말한다.

상태 코드	설명 (처리사유)	대표 예시
1xx	요청을 제대로 받았고, 처리가 진행 중임	
2xx	요청이 제대로 처리됨	• 200(OK): 성공
3xx	요청을 처리하려면, 클라이언트가 추가 동작을 취해야 함.	• 302(Found): 다른 URL로 갈 것
4xx	클라이언트가 잘못된 요청을 보내어 처리에 실패했습니다.	• 400(Bad Request): 요청이 문법에 맞지 않음 • 401(Unauthorized): 클라이언트가 요청한 리소스에 대한 인증이 실패함 • 403(Forbidden): 클라이언트가 리소스에 요청할 권한이 없음 • 404(Not Found): 리소스가 없음
5xx	클라이언트의 요청은 유효하지만, 서버에 에러가 발생하여 처리에 실패했습니다.	• 500(Internal Server Error): 서버가 요청을 처리하다가 에러가 발생함 • 503(Service Unavailable): 서버가 과부하로 인해 요청을 처리할 수 없음

2-6. https

http는 응답과 요청을 평문으로 전달되어 정보가 유출될 수 있다. 그래서 https는 TLS(Transport Layer Security)프로토콜을 도입하여서 응답과 요청과정의 모든 메세지를 암호화할 수 있다. 그렇기 때문에 요즘에는 https를 많이 사용하는 추세이고, 특히 금융이나 정부 서비스와 같은 민감한 데이터들을 취급하는 곳에서는 꼭 https를 사용한다.

1번 사진은 http 프로토콜이고 2번 사진은 https 프로토콜이다. http 프로토콜은 평문으로 되어있어 사람이 해석할 수 있지만, https 프로토콜은 암호문이어서 복호화를 하지 않는 이상 사람이 해석할 수 없다. 추가로 기본 http의 포트 번호는 80이고, https의 포트 번호는 443이다.

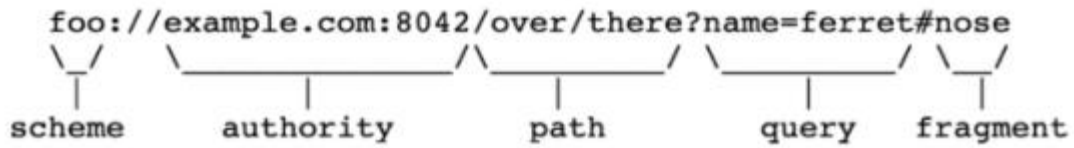
3-1. 웹 브라우저

웹 브라우저는 UX(User eXperience, 뛰어난 이용자 경험)를 제공하는 소프트웨어 중 하나이다. 이용자가 주소창에 무언가를 입력했을 때 웹 브라우저는 다음과 같은 과정을 거친다.

웹 브라우저의 주소창에 입력된 주소를 해석(URL 분석)
↓
검색한 주소를 탐색(DNS 요청)
↓
http를 통하여 해당 주소에 요청
↓
해당 주소의 http 응답 수신
↓
리소스 다운 및 웹 렌더링

3-2 URL

URL은 Uniform Resource Locator의 약자로 웹에 있는 리소스의 위치를 표현하는 문자열이다. URL은 Scheme, Authority (Userinfo, Host, Port), Path, Query, Fragment 등으로 구성된다.



위 URL을 분석해보면 scheme는 foo이고, authority는 example.com:8042이고, path는 over/there이고, query는 name=ferret, fragment는 nose임을 알 수 있다. 좀 더 자세히 살펴보면 **Scheme**는 웹 서버가 어떤 프로토콜로 통신할 지를 나타내는 것으로 현재 URL에서는 foo 프로토콜로 통신한다는 것을 알 수 있다. 그리고 **Authority**는 웹 주소를 나타내는 것으로 주소에 대한 정보를 가지고 있는 host와 포트에 대한 정보를 가지고 있는 port로 나뉜다. 현재 URL에서는 example.com:8042에서 host는 example.com이고, port는 8042임을 알 수 있다. **Path**는 접근할 웹 서버의 리소스 경로로 '/'로 구분하는 것을 말한다. 즉, 이 URL에서는 over에 있는 there로 갈 것이라는 것을 알 수 있다. **Query**는 웹 서버에 전달하는 파라미터로 URL에서 '?'뒤에 위치하는 것이다. 이 URL에서는 name=ferretr가 전달하고 싶은 파라미터라는 것을 알 수 있다. 마지막으로 **Fragment**는 메인 리소스에 존재하는 서브 리소스를 접근할 때 이를 식별하기 위한 정보를 담고 있는 것으로 '#' 문자 뒤에 위치한다. 쉽게 말해 리소스 자체의 다른 부분을 가리키는 북마크라고 할 수 있다. 이 URL에서는 nose가 Fragment임을 알 수 있다.

3-3. Domain name

URL의 구성요소인 Host는 Domain name / IP address의 값을 갖는다. IP address는 네트워크에서 통신이 이루어질 때 장치를 식별하기 위해 사용되는 주소이다. 이때, IP address는 불규칙한 숫자로 이루어져있어서 사람이 외우기 어렵기 때문에 도메인 특성을 담은 이름인 Domain name을 사용한다. 브라우저는 Domain name을 이용할 때 Domain Name Server(DNS)가 응답한 IP address를 이용한다.

예를 들어 example.com에 접속하면 아래와 같이 8.8.8.8#53으로 DNS로 이동한 후 Domain name인 example.com의 IP address는 93.184.216.34임을 알 수 있다.

```
$ nslookup example.com
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   example.com
Address: 93.184.216.34
```

→ Domain name에 대한 정보를 보고 싶으면 nslookup 명령어를 이용하면 된다.

3-4. 웹 렌더링

웹 렌더링은 서버로부터 받은 리소스를 이용자에게 시각화하는 행위를 말한다. 즉, 실시간으로 웹 사이트가 그려지는 과정을 말한다. 이때 활용되는 렌더링 엔진은 웹 사이트의 소스코드를 읽고, 각각의 요소들이 어떤 크기와 너비로 배치되는 지, 텍스트의 크기는 어떤 지, 색상은 어떤 지, 간격은 어떤 지에 대해 실시간으로 계산해 그려준다. 그 후 이용자가 볼 수 있는 실제 웹 사이트를 만들어주는데 이 과정이 1초 이내에 일어나기 때문에 렌더링 과정을 눈으로 보기는 쉽지 않다.

4-1. 개발자 도구

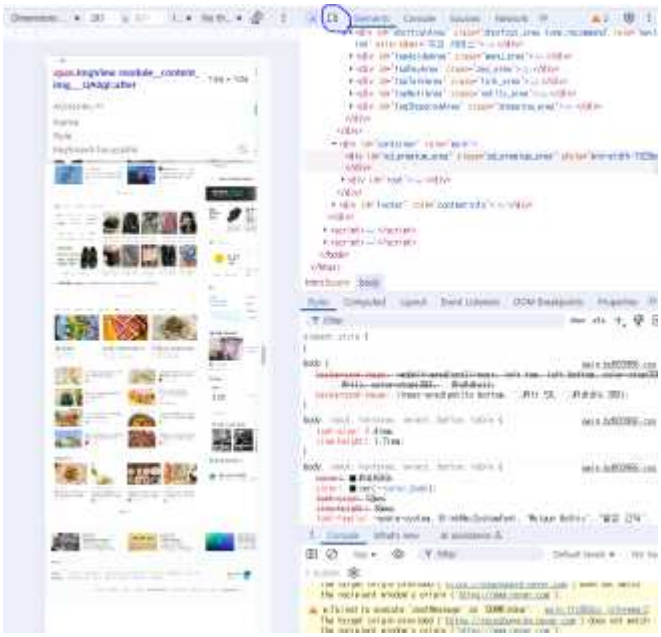
개발자 도구는 HTML과 CSS 코드를 브라우저에서 수정하고 결과를 바로 확인할 수 있는 것으로 웹 서비스를 진단하는데도 도움이 되고 웹 취약점을 찾는데도 유용하다.

4-2. 개발자 도구 실습

크롬에서는 F12를 누르면 개발자 도구를 확인 할 수 있다. 이 화면에서 **Ctrl + U**를 누르면 페이지 소스 코드를 볼 수 있다. 나는 네이버창에서 실습을 해보기로 했다.

→ 이 버튼은 **요소 검사**를 하는 버튼이다, 이 버튼을 누른 후 원하는 요소에 마우스를 올리면 그 대상의 정보가 출력된다. 그 상태에서 클릭하면 그 대상과 관련된 코드가 하이라이팅된다.

이 버튼은 디바이스 툴바이다. 이를 활용하면 화면의 비율을 원하는 값으로 바꿀 수 있다. 그래서 다양한 화면 비율을 설정하여 다른 장치에서도 잘 작동되는 지 점검할 수 있다.

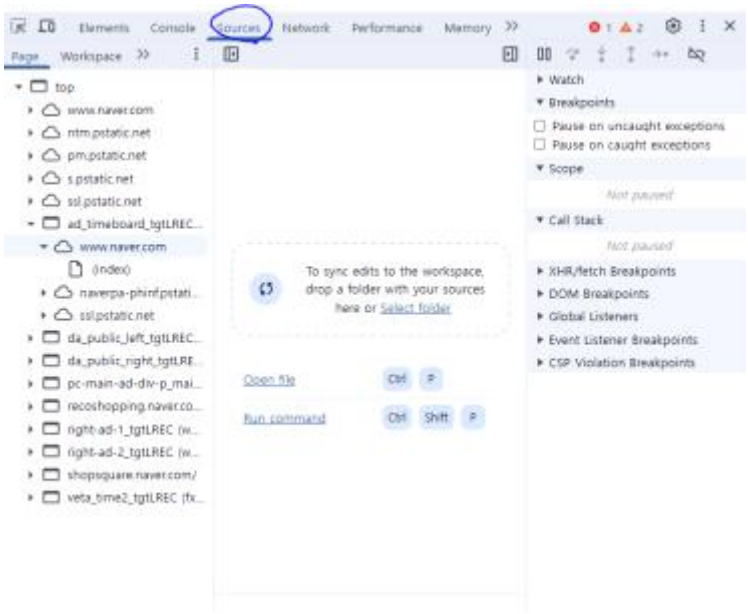


코드를 선택한 후 **F2**를 누르거나 **더블 클릭**하면 코드를 수정할 수 있다.

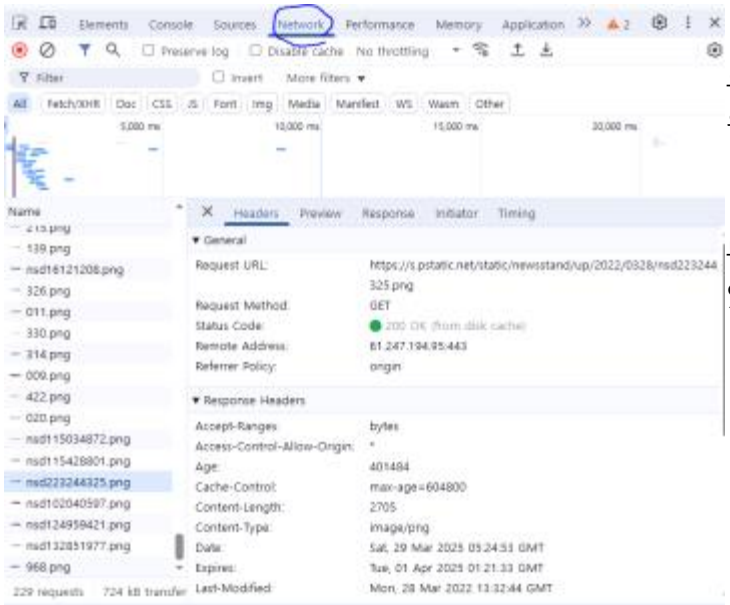
이 창은 콘솔 창이다. 콘솔 창에서는 JS에서 발생한 각종 메세지를 출력하고, 자바스크립트 코드도 실행할 수 있도록 해준다.

→콘솔 창에 다음과 같이 문자열을 출력하는 alert를 이용해 alert("hello"); 를 입력하면 hello가 출력되는 것을 볼 수 있다.

이 버튼을 누르면 아래와 같이 왼쪽은 현재 페이지의 파일시스템, 중간은 선택한 리소스 상세보기, 오른쪽은 디버깅 정보를 볼 수 있다.

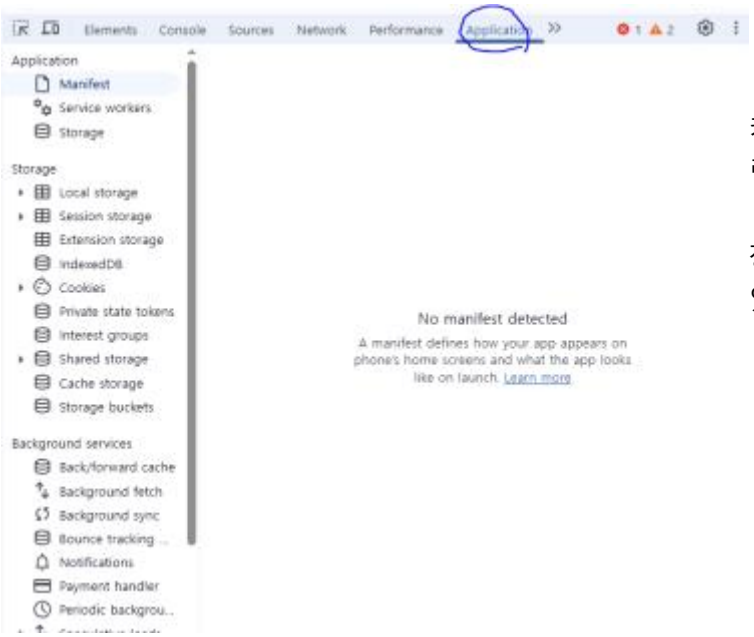


그리고 네트워크 차엥서는 서버와 데이터를 확인할 수 있다.



→그리고 좌측에 있는 로그를 우클릭하면 URL, PowerShell 등 원하는 형태로 복사할 수 있다.

→또한 네트워크창에서 ESC를 누르면 아래에 콘솔창을 띄울 수 있어서 동시 사용이 가능하다.



→ Application을 이용하면 쿠키, 캐시, 이미지, 폰트 등과 관련된 리소스를 조회할 수 있다. 또한, 쿠키에서 브라우저에 저장된 쿠기 정보를 확인할 수도 있고, 수정할 수도 있다.

4-3. 혼자 실습(문제 풀어보기)

드림핵에 있는 문제를 풀어보았다. 우선 문제파일을 다운받은 후 project.html로 들어가보았다. 그 후 F12를 눌러 개발자 도구에 들어가보았다. 그 후 문제 설명대로 Sources 탭에 들어가서 DH를 찾아보았다. 평소에 검색하는 것과 마찬가지로 ctrl + F를 눌러 DH를 검색해보았더니 플러그 찾을 수 있었다. 추가로 찾아보니 폴더를 우클릭하면 search in folder라는 항목으로도 검색할 수 있다는 것을 알았다. 내가 처음에 한 방식은 그 파일 속에서만 검색할 수 있는 기능이지만 폴더를 우클릭해서 search in folder을 이용해 검색하는 방식은 그 폴더 전체를 검색하는 기능이어서 더 유용할 것 같다.

5-1. 쿠키

http 프로토콜의 특성을 갖는 상태를 유지하기 위해 쿠키를 만들게되었다. http 프로토콜의 특성은 Connectionless, Stateless이다.

Connectionless는 하나의 요청에는 하나의 응답을 한 후 연결을 종료하고, 새 요청이 있을 때마다 새로운 연결을 맺는 특성이고, Stateless는 통신이 끝난 후 상태 정보를 저장하지 않는 특징이다.

이러한 Connectionless와 Stateless라는 특성을 가진 쿠키는 Key와 Value로 이루어져있고, 클라이언트의 정보 기록과 상태 정보를 표현하기 위해 사용한다. 그래서 쿠키가 존재하지 않으면 서버는 클라이언트가 누구인지 알지 못한다. 쿠기가 존재하면 서버는 클라이언트가 누구인지 식별해서 그에 맞는 정보를 준다. 하지만 이때 쿠키정보를 변조해 서버에게 요청을 하게 될 때 추가로 다른 검증 없다면 공격자가 이용자를 사칭하여 정보를 탈취할 수도 있다는 단점이 있다.

5-2. 세션

세션은 쿠키에 인증 상태를 저장하지만 클라이언트가 인증 정보를 변조할 수 없게 해준다. 바로 랜덤키를 이용하여 변조할 수 없게 만든다. 인증 정보는 서버에 저장하고, 클라이언트에게는 해당 정보에 접근할 수 있는 키를 주면 공격자가 정보를 변조할 수 없게 만들 수 있다. 이러한 키를 Session ID라고 한다. 브라우저는 이 키를 쿠키에 저장하고 http요청을 보낼 때 사용한다. 이때, 서버는 요청에 포함된 키에 해당 데이터를 가져와 인증 상태를 확인한다.

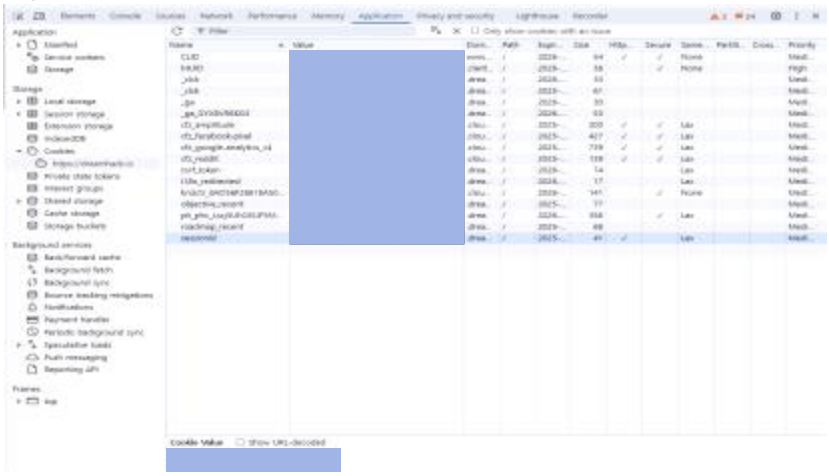
5-3. 쿠키 실습

F12로 개발자 도구를 켠 뒤 콘솔 창으로 들어가 document.cookie를 입력하면 쿠키를 확인할 수 있다. 혹은 Application 탭에서 쿠키를 펼친 후 쿠키를 확인할 수도 있다.

5-4. 세션 실습

1) 로그인 응답 확인

2) 설정된 쿠키 확인



개발자 도구에서 Application창에서 cookies를 누른 뒤 sessionid를 클릭하면 나의 쿠키를 볼 수 있다.

3) 저장된 세션 값 삭제

sessionid를 우클릭한 뒤 delete를 하면 쿠키에 저장된 세션 값이 삭제된다. 이를 삭제한 뒤 새로고침을 하면 로그아웃이 된다.

4) 세션 값 재설정

아까 2번 과정에 나와있었던 sessionid를 복사하여서 표 맨 밑을 우클릭하면 Add new라는 게 뜨는데 name에는 sessionid value에는 아까 나와있었던 sessionid를 붙여넣기한 뒤 새로고침하면 로그인할 수 있다.

왜냐하면 쿠키에는 나의 세션 정보가 저장되어있어서 서버가 이 쿠키를 이용하여 이용자를 식별하고 인증할 수 있는 것이다. 이때 공격자가 나의 쿠키를 훔쳐서 나의 인증 상태 획득하는 것을 세션 하이재킹이라고 한다.

+새로 로그인을 하면 쿠키값이 바뀌어서 직전에 사용되었던 쿠키를 이용해야만 로그인이 되는 것 같다.

5-5. 엔드포인트

엔드포인트는 서비스에 접근할 수 있는 특정 기능을 제공하는 URL이다. 엔드포인트를 요청할 때, 특정 작업을 수행하거나 데이터를 검색하기 위하여 서버와 통신한다.

5-6. 쿠키 문제

나는 문제파일을 다운로드 받은 후, 파이썬으로 그 파일을 열어보았다. 그러고 난 후 users 중에 guest와 admin이 있어서 guest로 먼저 로그인을 시도해보았다. 나는 login 함수에서 try에 pw=users[username]이라는 코드를 보고 아이디와 비밀번호가 같다는 것을 알게되었다. 그래서 아이디와 비밀번호를 guest로 입력해보니 로그인이 되었다.

Welcome !

Hello guest, you are not admin

로그인을 완료한 후 나는 개발자 도구로 이것저것 살펴보았다. 이곳에서 전에 공부한 것처럼 쿠키 부분을 잘 살펴보았다.

나는 Application에 있는 쿠키 부분을 보았는데 Name은 username으로 되어있고 Value는 guest로 되어있다는 것을 보았다. 그래서 guest를 admin으로 바꾸면 되지 않을까? 라는 생각이 들어서 admin으로 바꾸고 새로고침을 하니 아래와 같이 admin으로 로그인이 되었다.

Welcome !

Hello admin, flag is DH{7952074b69ee388ab45432737f9b0c56}

5-7. 쿠키/세션 문제

이번 문제에서는 우선 5-6에서 푼 쿠키 문제와 유사한 방식으로 접근해보았다. 그런데 파이썬을 살펴보니 이전 문제와는 달리 guest와 user의 비밀번호가 공개되어있었다. 나는 우선 각각 로그인을 해보았다. 그리고 쿠키정보를 살펴보았는데 username은 이름으로 되어있고, sessionid는 계속해서 바뀌어서 admin에 대한 정보를 찾지 못하였다. 그래서 이 문제 직전에 있던 강의를 다시 한 번 보고 왔다. 그러니 문제 URL에 추가로 /admin을 넣어보면 문제를 풀 수 있을 것 같다는 생각이 들어서 /admin을 넣어보았다.

{"536997d6b0e742ac3da1084ee8ec9332d47cad17432dbeb6c1e93589280a5bce":"admin"}

그러니 다음과 같은 화면이 생겼다. 나는 : 앞에 있는 부분이 sessionid인 것 같다는 생각이 들었다. 그래서 /admin을 삭제하고 다시 처음 화면으로 돌아와서 개발자 도구를 켠 뒤 username은 admin으로 sessionid는 아까 찾은 것을 넣어본 뒤 새로고침을 해보았더니 다음과 같이 플래그를 찾을 수 있었다!

Hello admin, flag is DH{8f3d86d1134c26fedf7c4c3ecd563aae3da98d5c}

+ 이 문제를 통해 sessionid는 계속 바뀐다는 것을 알게되었다. 그리고 /admin을 사용하는 방식으로 직전 문제를 풀려고 해보니 해당 URL이 없다고 나왔다. 그래서 파이썬 코드에 @app.route('/admin')이 존재하지 않는다면 이러한 방식으로 문제를 해결할 수 없다는 것도 알게 되었다.

5-8. SOP(Same Origin Policy)

SOP은 동일 출처 정책으로 가져온 데이터를 악의적인 페이지에서 읽을 수 없도록 하는 것을 말한다. Origin을 구분하기 위해서는 프로토콜, 포트, 호스트가 모두 같아야한다.

https://same-origin.com/ 다음과 같은 URL이 있을 때,

https://same-origin.com/frame.html

→ '/' 뒷 부분인 Path만 다르기 때문에 Same Origin

http://same-origin.com/frame.html

→ https가 http로 바뀌면 프로토콜이 다르기 때문에 Cross Origin

https://cross.same-origin.com/frame.html

→ same 앞 부분에 cross가 추가되면 호스트가 다르기 때문에 Cross Origin

https://same-origin.com:1234/

→ com 뒤에 :1234 인 포트가 생겼기 때문에 Cross Origin

5-9. CORS(Cross-Origin Resource Sharing)

CORS는 교차 출처 리소스 공유로 오리진이 다르더라도 SOP의 제한을 받지 않고 리소스를 공유하고 Cross Origin의 데이터를 처리할 수 있도록 해준다.

http 요청을 하면 아래와 같이 http 응답이 온다.

Header	설명
Access-Control-Allow-Origin	헤더 값에 해당하는 Origin에서 들어오는 요청만 처리합니다.
Access-Control-Allow-Methods	헤더 값에 해당하는 메소드의 요청만 처리합니다.
Access-Control-Allow-Credentials	쿠키 사용 여부를 판단합니다. 예시의 경우 쿠키의 사용을 허용합니다.
Access-Control-Allow-Headers	헤더 값에 해당하는 헤더의 사용 가능 여부를 나타냅니다.

결론

<배운 점>
웹 해킹을 처음 공부해보는 것이어서 쉽지 않았는데 드림핵 커리큘럼을 따라가보니 생각보다 이해할 만했고, 개념을 배우고 직접 플래그를 찾는 과정에서 어떻게 웹 해킹을 할 수 있는 지, 웹 해킹에서 취약점이 어디인지에 대한 기본을 잘 배울 수 있었던 것 같다.

<궁금한 점>
드림핵 강의에서는 http 표준 메소드가 8개라고 하였는데 검색해보니 9개가 있다고 한다. 어떤 게 맞는 건지 잘 모르겠다.
로그인 응답 확인 실습을 하고 싶었는데 개발자 도구에서 network로 들어간 뒤 preserve log를 눌렀는데도 login을 찾아야 로그인 응답을 확인할 수 있는데 network에는 login/이 없어서 직접 응답을 확인 할 수 없었다. 왜 login/이 안뜨는 지 잘 모르겠다.(5-4 세션 실습)

<어려웠던 점>
엔드포인트 부분을 학습한 뒤 문제를 푸는 것은 할 만 했는데 아직 엔드포인트가 무엇인지 정확하게 모르겠다.

<다음 제출 계획>
우선 이번 SSR을 작성하면서 새로 배운 것을 다시 한 번 복습해보고, 특히 엔드포인트 부분은 추가로 공부한 뒤 다음 SSR에도 공부한 내용을 작성할 것이다. 그리고 다음번에도 드림핵 강의를 통해서 30%정도 더 끝내고 이번 SSR처럼 이론 공부를 한 것과 실습한 것을 작성해보려고 생각 중이다.

<참고자료>

웹 해킹이란 무엇인가?
https://peemangit.tistory.com/312

드림핵 강의
https://dreamhack.io/lecture/roadmaps/1

인코딩 vs 디코딩 정확하게 이해하기
https://codingpractices.tistory.com/entry/%EC%9D%B8%EC%BD%94%EB%94%A9-vs-%EB%94%94%EC%BD%94%EB%94%A9-%EC%A0%95%ED%99%95%ED%95%98%EA%B2%8C-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0

http 메서드 종류 & 요청 흐름 100 총정리
https://inpa.tistory.com/entry/WEB-%F0%9F%8C%90-HTTP-%EB%A9%94%EC%84%9C%EB%93%9C-%EC%A2%85%EB%A5%98-%ED%86%B5%EC%8B%A0-%EA%B3%BC%EC%A0%95-%F0%9F%92%AF-%EC%B4%9D%EC%A0%95%EB%A6%AC

URL 프래그먼트 / HASH
https://velog.io/@roro/URL-%ED%94%84%EB%9E%98%EA%B7%B8%EB%A8%BC%ED%8A%B8-HASH

웹 브라우저의 렌더링이란?
https://yozm.wishket.com/magazine/detail/646/

API와 엔드포인트 이해하기: 종합 가이드
https://apidog.com/kr/blog/apis-vs-endpoints-3/