

# WEB 취약점 스터디

---

권문정

---

# Contents

---

- 01** WEB 취약점 공격 개요
- 02** WEB 취약점 공격 특징 및 목적
- 03** WEB 취약점 공격 유형
- 04** WEB 취약점 공격 원리/방법
- 05** WEB 취약점 탐지 및 대응 방법

# 01. WEB 취약점 공격 개요

---

## ■ 웹 취약점이란?

"하나 이상의 위협에 의해 익스플로잇(보안 취약점을 이용한 공격)될 수 있는 자산 또는 자산들의 그룹의 약점"

- 국제 표준화 기구(ISO)

"시스템의 디자인, 구현 또는 작업 그리고 관리에서의 결함이나 약점으로서, 시스템의 보안 정책을 침해하기 위해 익스플로잇(보안 취약점을 이용한 공격)될 수 있는 것."

- 국제 인터넷 표준화 기구(IETF)

## ■ 웹 취약점 공격이란?

웹 사이트의 취약점을 공격하는 기술적 위협

Ex) 웹 페이지를 통하여 권한이 없는 시스템에 접근하거나 데이터 유출 및 파괴와 같은 행위

## 02. WEB 취약점 공격 특징 및 목적

---

### ■ 웹 애플리케이션 취약점 공격의 발생 원인

사용자가 웹 브라우저/다양한 도구로 서버 쪽 애플리케이션에 개발자들이 예상치 못했던 임의의 입력 값을 보낼 수 있음  
→ Attacker가 입력값을 조작하여 애플리케이션의 로직/기능/데이터 훼손 가능

### ■ 웹 취약점 공격의 목적 예시

- 인터넷 쇼핑 시 물건을 싼 가격에 사기 위해 **HTML** 폼 필드에 보이지 않는 값으로 주어진 해당 제품의 가격 값을 변조
- 다른 인증된 사용자의 세션을 도용하기 위해 **HTTP** 쿠키 안에 포함돼 전달되는 세션 토큰 조작
- 애플리케이션 처리상의 로직 결함을 악용하기 위해 정상적으로 제공될 특정 값 제거
- 악의적인 데이터베이스 쿼리를 강제 삽입하여 계정 비밀번호 등 민감한 데이터에 접근이 가능하도록 내부의 데이터베이스에 의해 처리될 입력 값 조작

## 03. WEB 취약점 공격 유형

### ■ 웹 취약점 공격의 유형

- 웹 애플리케이션의 취약점 공격 (제일 빈번)
- 웹 엔진 취약점 공격
- 각종 웹 서버 및 미들웨어 기본 제공 샘플 파일 공격

### ■ OWASP TOP 10 : 상위 10위의 가장 빈번한 웹 어플리케이션 취약점 공격 유형

The screenshot shows the OWASP Top 10 2021 website. The browser address bar displays <https://owasp.org/Top10/>. The page title is "OWASP Top 10:2021".

**Left Sidebar (Navigation):**

- OWASP Top 10:2021
- Home
- Notice
- Introduction
- How to use the OWASP Top 10 as a standard
- How to start an AppSec program with the OWASP Top 10
- About OWASP
- Top 10:2021 List**
- A01 Broken Access Control
- A02 Cryptographic Failures
- A03 Injection
- A04 Insecure Design
- A05 Security Misconfiguration
- A06 Vulnerable and Outdated Components
- A07 Identification and Authentication Failures
- A08 Software and Data Integrity Failures
- A09 Security Logging and Monitoring Failures
- A10 Server Side Request Forgery (SSRF)

**Main Content Area:**

## Introduction

Welcome to the OWASP Top 10 - 2021

Welcome to the latest installment of the OWASP Top 10! The OWASP Top 10 2021 is all-new, with a new graphic design and an available one-page infographic you can print or obtain from our home page.

**Right Sidebar (Table of contents):**

#### Table of contents

- Welcome to the OWASP Top 10 - 2021
- What's changed in the Top 10 for 2021
- Methodology
- How the categories are structured
- How the data is used for selecting categories
- Why not just pure statistical data?
- Why incidence rate instead of frequency?
- What is your data collection and analysis process?
- Data Factors
- Thank you to our data contributors
- Thank you to our sponsors

## 03. WEB 취약점 공격 유형

### ■ OWASP TOP 10 유형별 정리

#### 1) Broken Access Control (접근 권한 취약점)

접근 제어는 사용자가 권한을 벗어나 행동할 수 없도록 정책을 시행

접근 제어가 취약하면 사용자가 주어진 권한을 벗어나 모든 데이터를 무단으로 열람·수정·삭제할 수 있음

#### 2) Cryptographic Failures (암호화 오류)

'민감 데이터 노출'의 명칭이 2021년에 'Cryptographic Failures(암호화 오류)'로 변경

적절한 암호화가 이루어지지 않으면 민감 데이터가 노출될 수 있음

#### 3) Injection (인젝션)

SQL, NoSQL, OS 명령, ORM(Object Relational Mapping), LDAP, EL(Expression Language) 또는 OGNL(Object Graph Navigation Library) 인젝션 취약점은 신뢰할 수 없는 데이터가 명령어나 쿼리문의 일부분으로써, 인터프리터로 보내질 때 취약점 발생

#### 4) Insecure Design (안전하지 않은 설계)

'안전하지 않은 설계'는 누락되거나 비효율적인 제어 설계로 표현되는 다양한 취약점.

안전하지 않은 설계와 안전하지 않은 구현에는 차이가 있지만, 안전하지 않은 설계에서 취약점으로 이어지는 구현 결함 발생 가능

#### 5) Security Misconfiguration (보안 설정 오류)

애플리케이션 스택의 적절한 보안 강화가 누락되었거나 클라우드 서비스에 대한 권한이 적절하지 않게 구성되었을 때, 불필요한 기능이 활성화 되거나 설치되었을 때, 기본계정 및 암호화가 변경되지 않았을 때, 지나치게 상세한 오류 메시지를 노출할 때, 최신 보안기능이 비활성화 되거나 안전하지 않게 구성되었을 때 발생

## 03. WEB 취약점 공격 유형

### ■ OWASP TOP 10 유형별 정리

#### 6) Vulnerable and Outdated Components (취약하고 오래된 요소)

취약하고 오래된 요소는 지원이 종료되었거나 오래된 버전을 사용할 때 발생  
애플리케이션 뿐만 아니라, DBMS, API 및 모든 구성 요소들이 포함됨

#### 7) Identification and Authentication Failures (식별 및 인증 오류)

원래 '취약한 인증'으로 알려졌지만, 식별 실패까지 포함해 더 넓은 범위를 포함할 수 있도록 변경된 항목  
사용자의 신원확인, 인증 및 세션관리가 적절히 되지 않을 때 취약점이 발생할 수 있음

#### 8) Software and Data Integrity Failures(소프트웨어 및 데이터 무결성 오류)

2021년 새롭게 등장, 무결성을 확인하지 않고 소프트웨어 업데이트 등을 할 때 발생

#### 9) Security Logging and Monitoring Failures (보안 로깅 및 모니터링 실패)

'불충분한 로깅 및 모니터'링 명칭이었던 명칭이 '보안 로깅 및 모니터링 실패'로 변경됨  
로깅 및 모니터링 없이는 공격활동을 인지할 수 없다. 이 카테고리는 진행중인 공격을 감지 및 대응하는데 도움 됨

#### 10) Server-Side Request Forgery (서버 측 요청 위조)

2021년에 등장  
SSRF 결함은 웹 애플리케이션이 사용자가 제공한 URL의 유효성을 검사하지 않고 원격 리소스를 가져올 때마다 발생  
이를 통해 공격자는 방화벽, VPN 또는 다른 유형의 네트워크 ACL(액세스 제어 목록)에 의해 보호되는 경우에도  
응용 프로그램이 조작된 요청을 예기치 않은 대상으로 보내도록 강제할 수 있음

# 03. WEB 취약점 공격 원리/방법

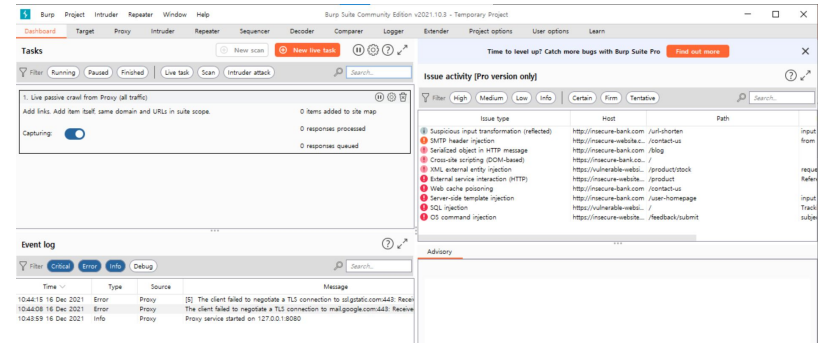
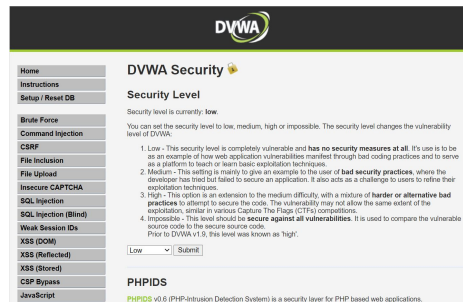
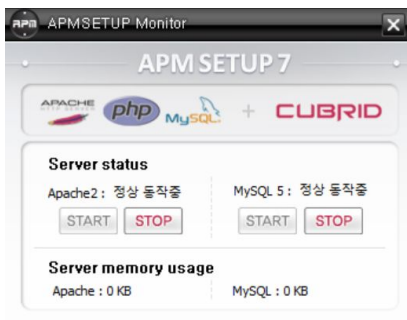
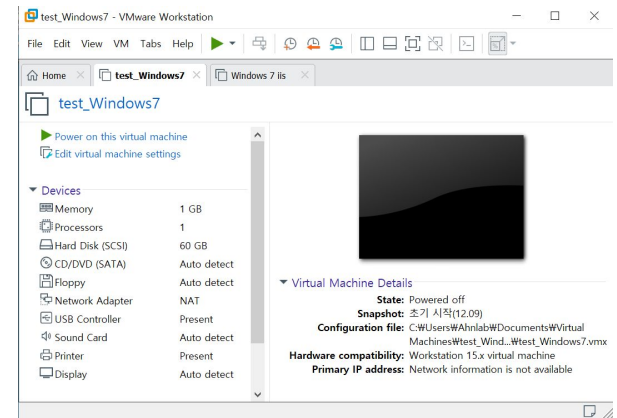
## ■ 실습 환경

### 1. 가상 환경 구축

- VMware Workstation Pro(free trial 30일)
- OS : Window10 64 bit

### 2. 실습 도구

- APMSETUP7 (window 32 bit)
  - Apache(2.2.14)
  - PHP(5.1.41-community)
  - MySQL5.1.41-community MySQL Community Server (GPL)
- Damn Vulnerable Web Application (DVWA)
  - : 취약점 진단/모의해킹을 할 수 있는 웹 어플리케이션. PHP/MySQL 환경. Security Level : Low로 설정
- Burp Suite
  - : 프록시(Proxy)를 사용하여 네트워크에서 통신하는 패킷을 가로채 분석 및 조작, 취약점을 확인하는 도구

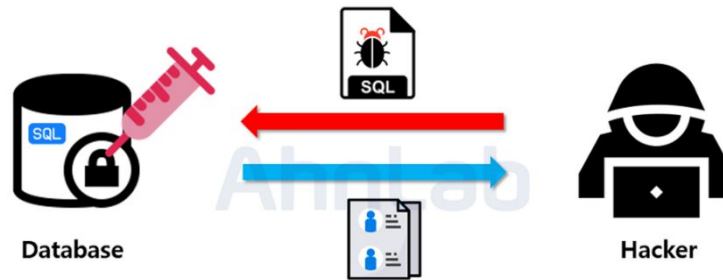




## 03. WEB 취약점 공격 원리/방법 – SQL Injection

### ■ SQL Injection이란?

악의적인 사용자가 보안상의 취약점을 이용하여 임의의 **SQL문**을 주입하고 실행되게 하여 데이터베이스가 개발자가 의도하지 않은 동작을 하도록 조작하는 행위. 로그인 우회, 계정 비밀번호 조회 등이 있음.



### ■ SQL Injection 소스 코드 예시(DVWA – Security Level : Low)

```
127.0.0.1/dvwa/vulnerabilities/view_source.php?id=sqli&security=low

SQL Injection Source
vulnerabilities/sqli/source/low.php

<?php
if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    switch ( $_DVWA[ 'SQLI_DB' ] ) {
        case MYSQL:
            // Check database
            $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
            $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( "<pre>" . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : (($___m

            // Get results
            while( $row = mysqli_fetch_assoc( $result ) ) {
                // Get values
                $first = $row["first_name"];
                $last = $row["last_name"];
```

## 03. WEB 취약점 공격 원리/방법 – SQL Injection

### ■ Error based SQL Injection

- 로그인 시에 입력한 값은 SELECT문의 WHERE절에 들어간다는 점을 활용하여 논리적 에러를 이용한 공격

Ex) ' OR 1=1 –

- WHERE 절에 있는 싱글 쿼터를 닫기 위한 싱글 쿼터와 OR 1=1 라는 구문을 이용하여 WHERE 절을 모두 참으로 만들고, -- 를 넣어줌으로 뒤의 구문을 모두 주석 처리해 줌
- 테이블에 있는 모든 정보를 조회 가능, 가장 먼저 만들어진 계정으로 로그인에 성공하게 됨  
→ 보통 관리자 계정을 맨 처음 만들기 때문에 관리자 계정에 로그인 할 수 있게 됨  
관리자 계정을 탈취한 악의적인 사용자는 관리자의 권한을 이용해 또 다른 2차 피해를 입힐 수 있음

① SELECT \* FROM Users WHERE id = 'INPUT1' AND password = 'INPUT2'



③ SELECT \* FROM Users WHERE id = ' ' OR 1=1 -- ' AND password = 'INPUT2'  
=> SELECT \* FROM Users

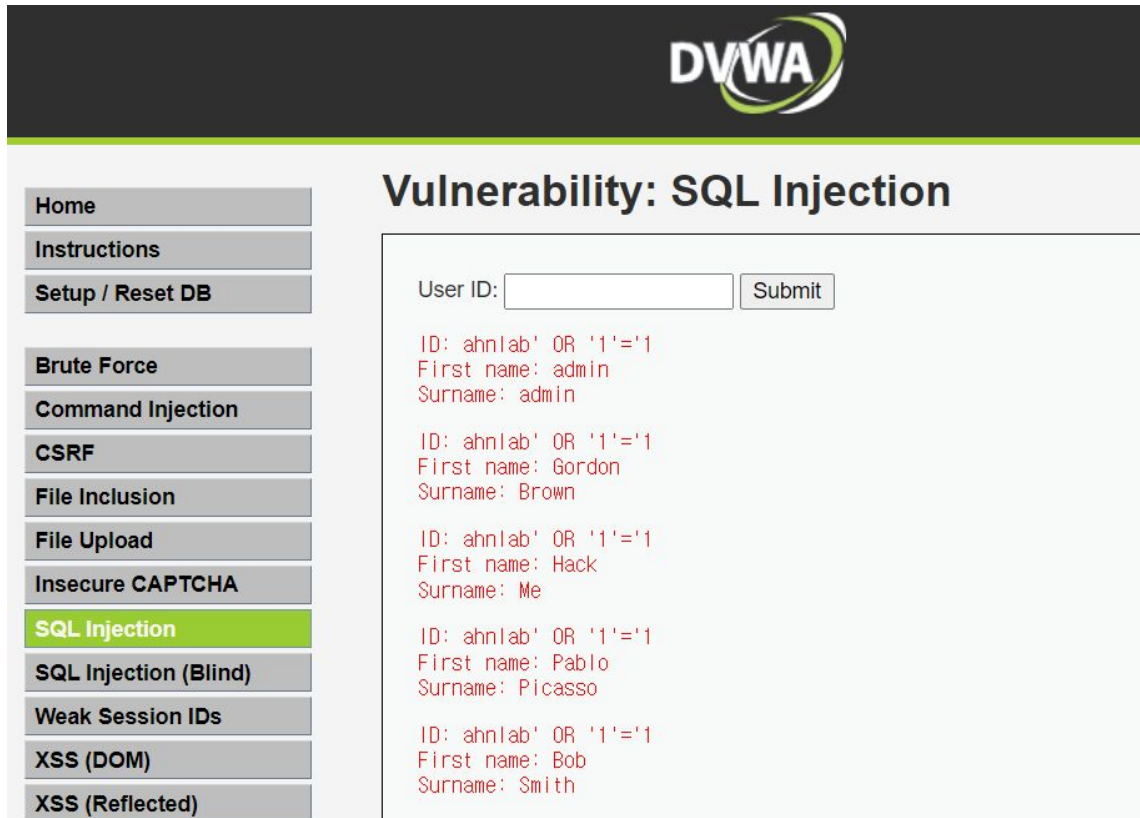
## 03. WEB 취약점 공격 원리/방법 – SQL Injection

### ■ Error based SQL Injection 실습

Ex) ID 입력란에 `ahnlab' OR '1'='1` 입력하기

→ ID를 검색하는 WHERE 조건절에 OR 비교문을 추가하는 악의적인 구문 삽입

- 데이터베이스가 테이블에 있는 모든 열을 검사한 후, ID가 `ahnlab`이거나 1의 값이 1을 만족시키는 열을 보여줌
- 1의 값은 항상 1과 동일, 따라서 데이터베이스는 테이블에 있는 모든 레코드를 반환해서 나타냄



**DVWA**

### Vulnerability: SQL Injection

Home  
Instructions  
Setup / Reset DB  
Brute Force  
Command Injection  
CSRF  
File Inclusion  
File Upload  
Insecure CAPTCHA  
**SQL Injection**  
SQL Injection (Blind)  
Weak Session IDs  
XSS (DOM)  
XSS (Reflected)

User ID:

ID: ahlalab' OR '1'='1  
First name: admin  
Surname: admin

ID: ahlalab' OR '1'='1  
First name: Gordon  
Surname: Brown

ID: ahlalab' OR '1'='1  
First name: Hack  
Surname: Me

ID: ahlalab' OR '1'='1  
First name: Pablo  
Surname: Picasso

ID: ahlalab' OR '1'='1  
First name: Bob  
Surname: Smith

## 03. WEB 취약점 공격 원리/방법 – SQL Injection

### ■ UNION based SQL Injection

- UNION 연산자는 두 개나 그 이상의 SELECT문을 하나의 결과로 합칠 때 사용됨  
→ SELECT문에 SQL Injection 취약점이 존재할 경우, UNION문으로 두 번째 SELECT문을 실행할 수 있음
- 공격 성공 조건 : Union 하는 두 테이블의 컬럼 수, 데이터 형이 같아야 함

Ex) Board 라는 테이블에서 게시글을 검색하는 쿼리문에 UNION 연산자로 id와 password 열의 내용을 요청

- 1) 입력값을 title 과 contents 열의 데이터랑 비교한 뒤 비슷한 글자가 있는 게시글을 출력
- 2) 여기서 입력값으로 UNION 연산자와 함께 열 수를 맞춰서 SELECT 구문을 넣어주면, 두 쿼리문이 합쳐져 하나의 테이블로 보임
- 3) 공격 성공 시에 사용자의 개인정보가 게시글과 함께 화면에 보임

게시글 조회

① SELECT \* FROM Board WHERE title LIKE '%INPUT%' OR contents '%INPUT%'

Board table

id	title	contents
1	hi	Hello
2	bye	Bye bye



' UNION SELECT null,id,passwd FROM Users--



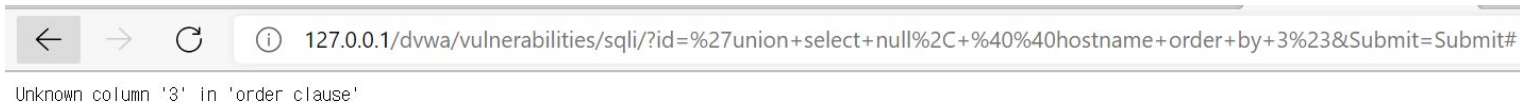
③ SELECT \* FROM Board WHERE title LIKE '% ' UNION SELECT null,id,passwd FROM Users --  
'% ' AND contents '% UNION SELECT null,id,passwd FROM Users -- '%'

## 03. WEB 취약점 공격 원리/방법 – SQL Injection

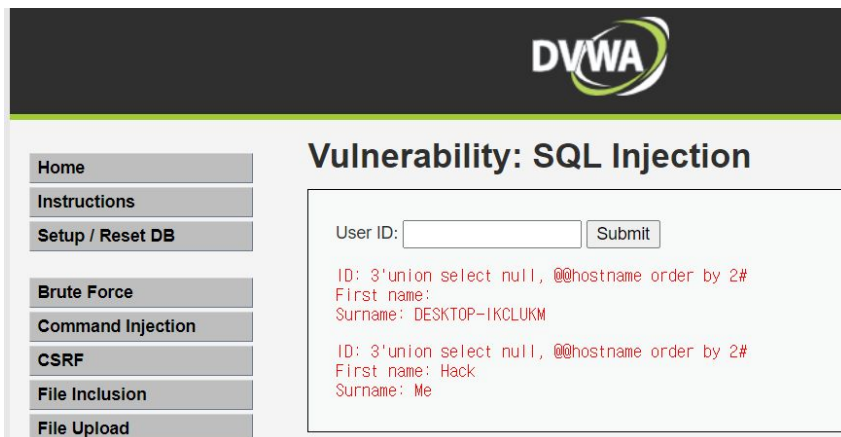
### ■ UNION based SQL Injection 실습1) 열 개수 확인하기

Ex) 'union select null, @@hostname order by [n]#  
hostname을 보여주고, [n]번째 열을 기준으로 정렬하라는 문구임

- 'union select null, @@hostname order by 3#  
→ 3번째 열이 없다는 에러 메시지가 출력됨



- 'union select null, @@hostname order by 2#  
→ 열의 개수가 2개임을 알 수 있음



## 03. WEB 취약점 공격 원리/방법 – SQL Injection

### ■ UNION based SQL Injection 실습2) 시스템 관리자, DB 이름 확인하기

- 'union all select system\_user(), user()#  
→ system\_user(), user() 함수로 시스템 관리자를 확인할 수 있음



- 'union select null, database()#  
→ database() 함수를 사용하여 현재 데이터베이스 이름을 알 수 있음



## 03. WEB 취약점 공격 원리/방법 – SQL Injection

### ■ UNION based SQL Injection 실습3) 데이터베이스 버전 확인하기

- 'union select 1, @@version#  
→ 데이터베이스의 버전을 확인할 수 있음  
데이터베이스 버전 별로 취약점이 다르기 때문에 공격 전에 참고 가능



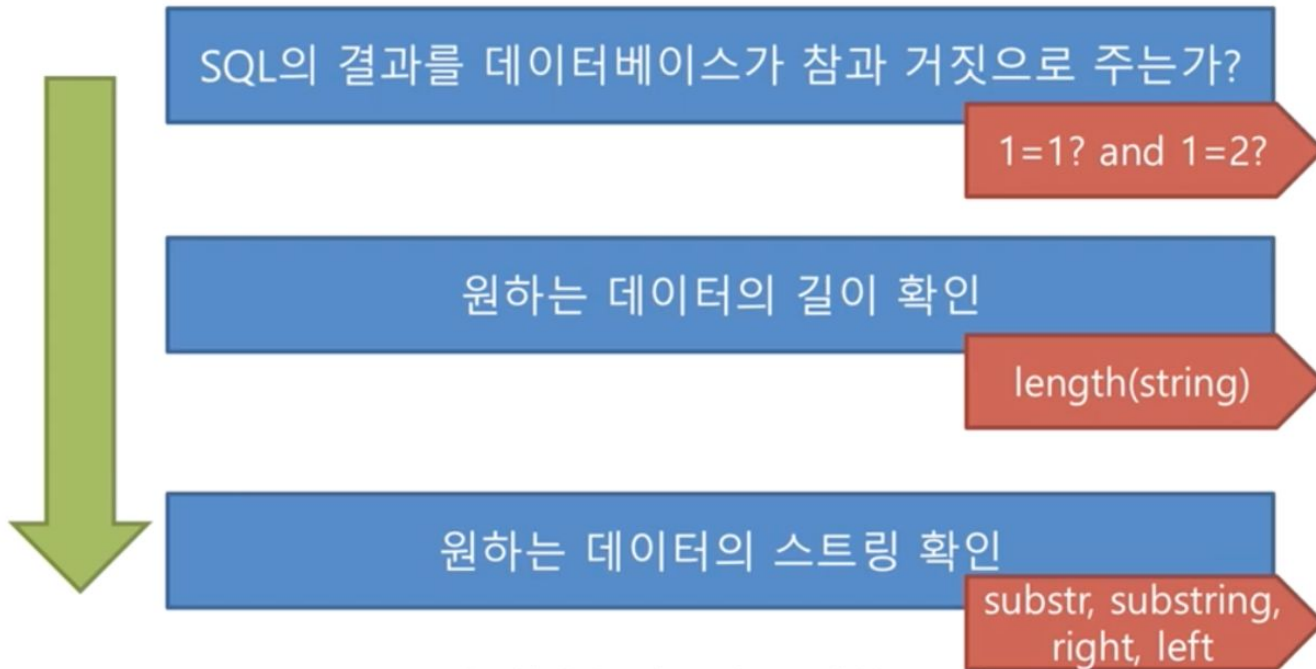
The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The top header features the DVWA logo. On the left, there is a sidebar with navigation links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection (which is highlighted in green). The main content area is titled "Vulnerability: SQL Injection". It contains a form with a "User ID:" label, an input field, and a "Submit" button. Below the form, the output of the query is displayed in red text: "ID: 'union select 1, @@version#", "First name: 1", and "Surname: 5.1.41-community". At the bottom, there is a section titled "More Information" with a list of links: [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection), <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>, [https://owasp.org/www-community/attacks/SQL\\_injection](https://owasp.org/www-community/attacks/SQL_injection), and <https://bobby-tables.com/>.

## 03. WEB 취약점 공격 원리/방법 – SQL Injection

### ■ Blind SQL Injection이란?

- SQL 쿼리 요청에 대한 응답 결과가 참, 거짓에 따라 데이터를 획득하는 공격
- 데이터 베이스 내에 데이터를 하나씩 요청하여 이에 대한 응답으로 필드 값이나 테이블 이름 같은 정보를 찾아냄
- 하나의 쿼리당 하나의 응답을 주고 받기 때문에 많은 쿼리 요청이 전제되어야 함
- 원하는 정보를 얻기까지 소요시간이 길

#### • 공격 플로우





## 03. WEB 취약점 공격 원리/방법 – SQL Injection

### ■ Blind SQL Injection 종류1) Boolean based

- 참, 거짓에 따라 SQL Injection을 반복적으로 수행하여 추측하는 방식
- 테이블 이름을 알아내는 **Boolean Based SQL Injection**

① SELECT \* FROM Users WHERE id = 'INPUT1' AND password = 'INPUT2'



②



abc123' and ASCII(SUBSTR(SELECT name FROM information\_schema.tables  
WHERE table\_type='base table' limit 0,1),1,1)) > 100 --

(MySQL 일 경우)

③ SELECT \* FROM Users WHERE id = 'abc123' and ASCII(SUBSTR(SELECT name FROM  
information\_schema.tables WHERE table\_type='base table' limit 0,1),1,1)) > 100 --  
' AND password = 'INPUT2'

로그인이 될 때까지 시도

- 로그인 폼을 통하여 임의로 ID를 만들고(abc123), MySQL 에서 테이블 명을 조회하는 구문 주입
- [임의로 가입한 ID]' and ASCII(SUBSTR(SELECT name From information\_schema.tables WHERE table\_type='base table' limit 0,1),1,1)) > 100 --
  - limit 키워드 : 하나의 테이블만 조회
  - SUBSTR 함수 : 첫 글자만 조회
  - ASCII : ascii 값으로 변환
- 만약 조회되는 테이블 명이 Users 라면 'U' 자가 ascii 값으로 조회가 될 것이고, 뒤의 100 이라는 숫자 값과 비교함 거짓이면 로그인 실패가 될 것이고, 참이 될 때까지 뒤의 100이라는 숫자를 변경해 가면서 비교하면 됨
- 공격자는 이 프로세스를 자동화 스크립트를 통하여 단기간 내에 테이블 명을 알아 낼 수 있음

## 03. WEB 취약점 공격 원리/방법 – SQL Injection

### ■ Blind SQL Injection 종류2) Time based

- 쿼리에 대한 응답이 "참"일 경우, **sleep()**함수를 사용하여 특정 시간만큼 지연시키는 공격 방식
- 거짓일 경우에 **sleep()** 함수가 실행되지 않으므로 시간 지연없이 응답이 이루어짐

- 데이터베이스의 길이를 알아내는 **Time based SQL Injection**

① SELECT \* FROM Users WHERE id = 'INPUT1' AND password = 'INPUT2'



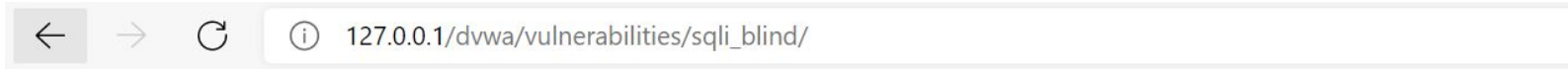
③ SELECT \* FROM Users WHERE id = 'abc123' OR (LENGTH(DATABASE())=1 AND SLEEP(2)) --  
' AND password = 'INPUT2'                      SLEEP 할 때까지 시도

- 로그인 폼을 통하여 임의로 ID를 만들고(abc123), MySQL 에서 데이터베이스의 길이를 조회하는 구문 주입
- Ex) [임의로 가입한 ID]' OR (LENGTH(DATABASE())=1 AND SLEEP(2))
  - LENGTH 함수 : 문자열의 길이를 반환
  - DATABASE 함수 : 데이터베이스의 이름을 반환
- 주입된 구문에서, LENGTH(DATABASE()) = 1 가 참이면 SLEEP(2) 가 동작하고, 거짓이면 동작하지 않음.  
→ 이를 통해서 숫자 1 부분을 조작하여 데이터베이스의 길이를 알아 낼 수 있음  
만약 SLEEP 이라는 단어가 치환 처리 되어있다면, 또 다른 방법으로 BENCHMARK 나 WAIT 함수를 사용 할 수 있음.  
Ex) BENCHMARK : BENCHMARK(1000000,AES\_ENCRYPT('hello','goodbye'));

## 03. WEB 취약점 공격 원리/방법 – SQL Injection

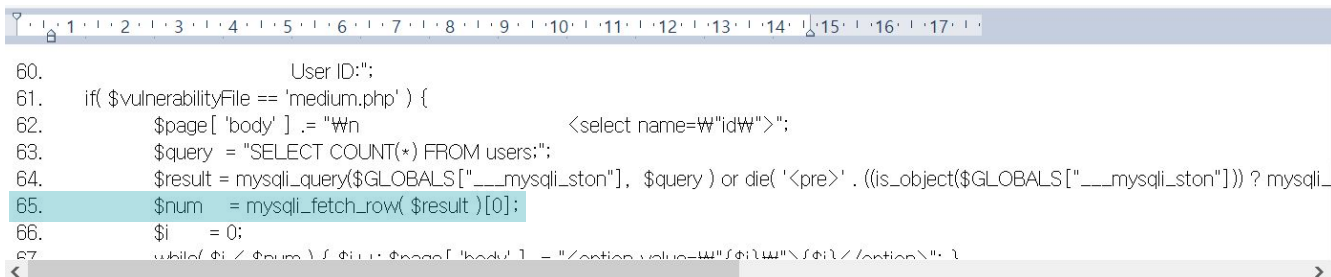
### ■ Blind SQL Injection 실습

- 실습 웹 애플리케이션 DVWA에서 다음과 같은 에러 메시지로 진행 불가



Parse error: syntax error, unexpected '[' in C:\WAPM\_Setup\htdocs\DVWA\vulnerabilities\sqli\_blind\index.php on line 65

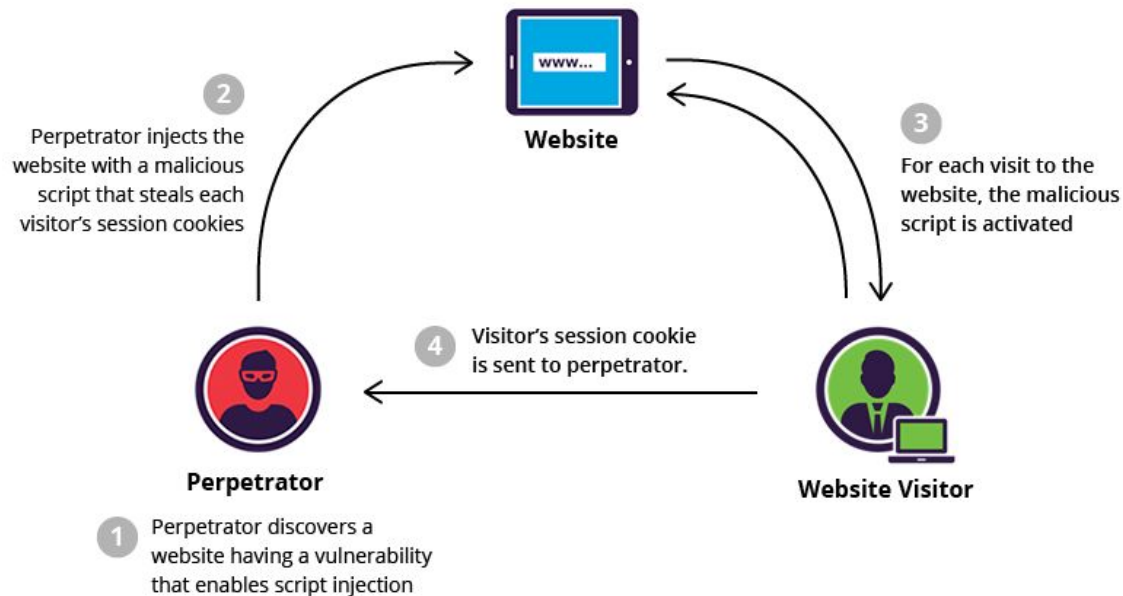
- 해당 경로에서 오류를 발생시키는 문자를 찾아보려 하였지만, 발견이 불가하였음



## 03. WEB 취약점 공격 원리/방법 - XSS

### ■ XSS 공격이란?

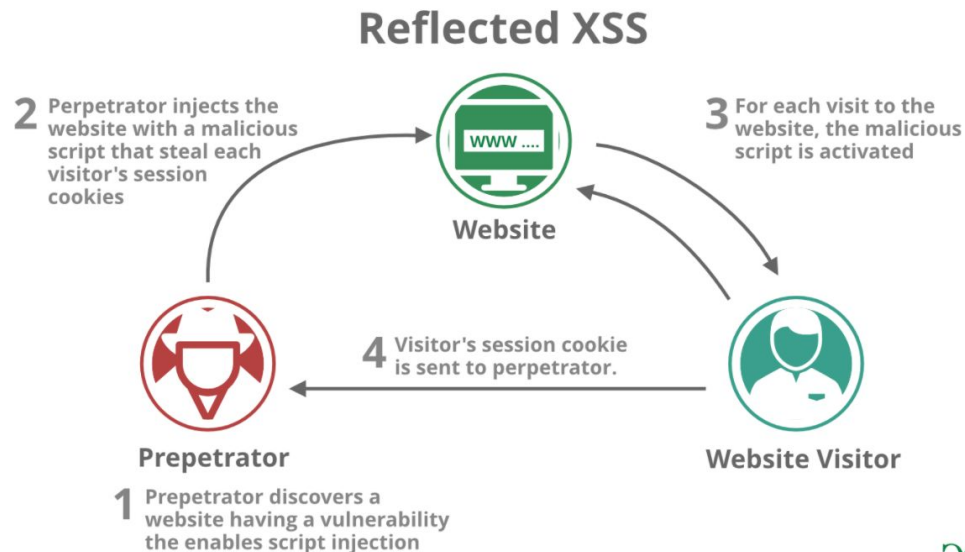
- 웹 어플리케이션에서 사용자 입력 값에 대한 필터링이 제대로 이루어지지 않을 경우, 공격자가 입력이 가능한 폼에 악의적인 스크립트를 삽입하여 해당 스크립트가 희생자 측에서 동작하도록 하여 악의적인 행위를 수행하는 취약점
- 공격자가 피해자의 브라우저에서 스크립트를 실행해 사용자 세션을 하이재킹하고, 웹 사이트를 훼손하거나 사용자를 악성 사이트로 리다이렉션할 수 있게 해줌
- 애플리케이션에서 새 웹 페이지가 적절한 검증 또는 회피 없이 신뢰할 수 없는 데이터를 포함하거나, **HTML** 또는 자바스크립트를 생성할 수 있는 브라우저 **API**를 사용해 사용자가 입력한 데이터로 기존 웹 페이지를 업데이트할 때 발생



## 03. WEB 취약점 공격 원리/방법 – XSS

### ■ Reflected XSS

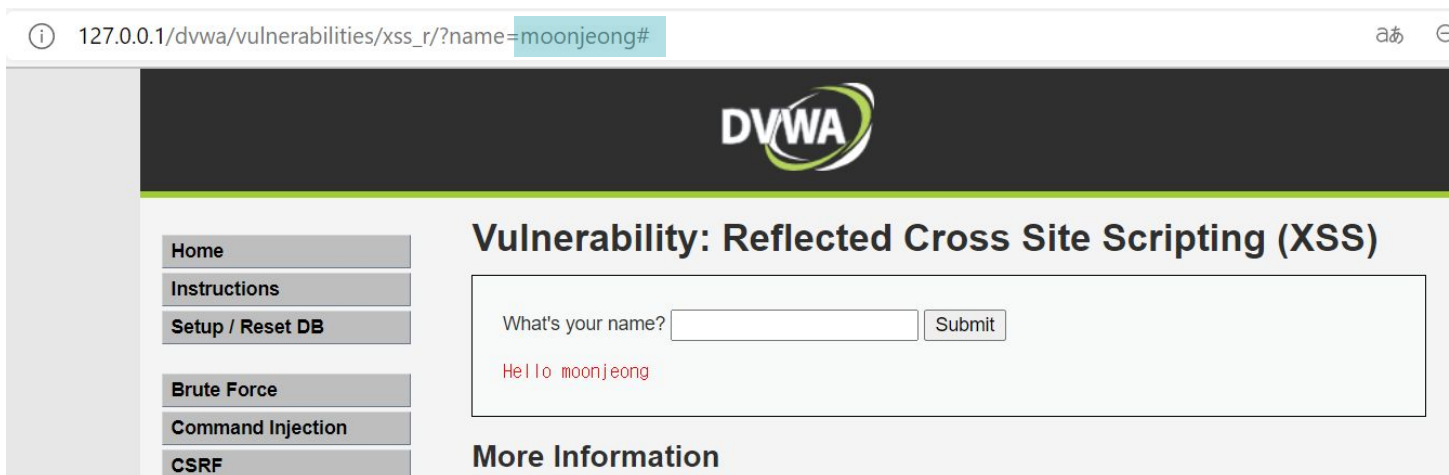
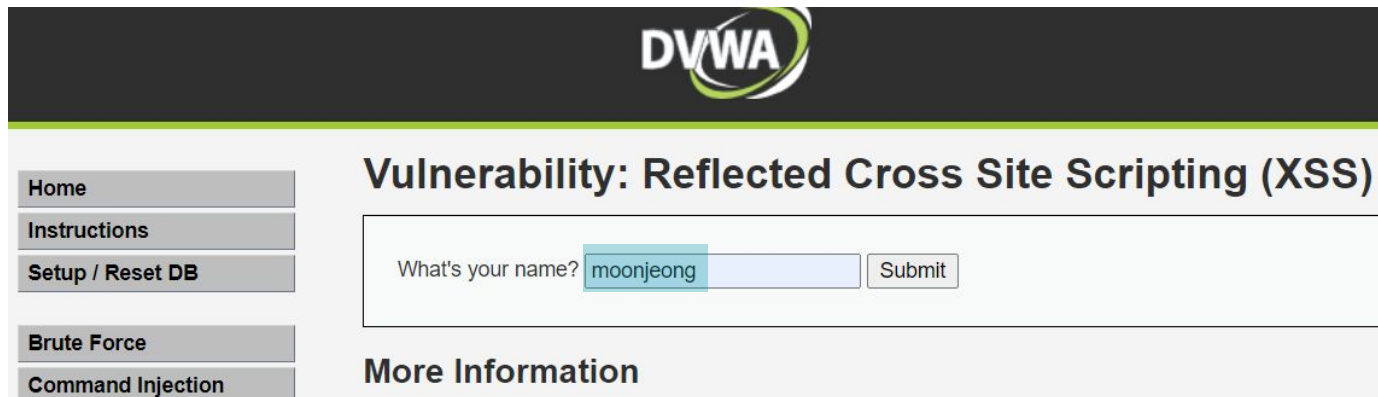
- 일회성의 HTTP Request와 Response로 이루어진 공격
  - 공격자가 악의적인 스크립트를 작성 하고 작성 후 웹 서버 측에 요청 시 웹 응용 프로그램에서 피해자의 브라우저로 반사될 때 발생
  - 이메일 또는 웹사이트 등을 통해 공격자가 미리 준비한 링크를 사용자가 클릭하게끔 유도하는 방식
  - 일회성이며, 지속적이지 않음.
- 공격 방법
    - 1) 공격자가 악성 스크립트가 포함된 URL을 사용자에게 노출시킴
    - 2) 사용자가 브라우저 내 해당 링크를 클릭했을 때 공격자가 정의한 악성 스크립트가 실행
    - 3) 해당 악성 스크립트는 브라우저 내 비밀 정보(세션, 쿠키, 토큰값 등)를 공격자에게 전달



## 03. WEB 취약점 공격 원리/방법 – XSS

### ■ Reflected XSS 실습1)

- what's your name? 칸에 넣은 문자열이 웹 브라우저 주소 창에 그대로 출력됨을 확인할 수 있음



## 03. WEB 취약점 공격 원리/방법 – XSS

### ■ Reflected XSS 실습

- what's your name? 칸에 alert 스크립트를 작성하여 alert창이 뜨게 할 수 있음
- Ex) `<script> alert('alert message');</script>`

[Home](#)  
[Instructions](#)  
[Setup / Reset DB](#)  
  
[Brute Force](#)  
[Command Injection](#)  
[CSRF](#)

## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello

### More Information

i 127.0.0.1/dvwa/vulnerabilities/xss\_r/?name= <script> +alert%28%27alert+message%27%29%3B<%2Fscript>#

127.0.0.1의 메시지

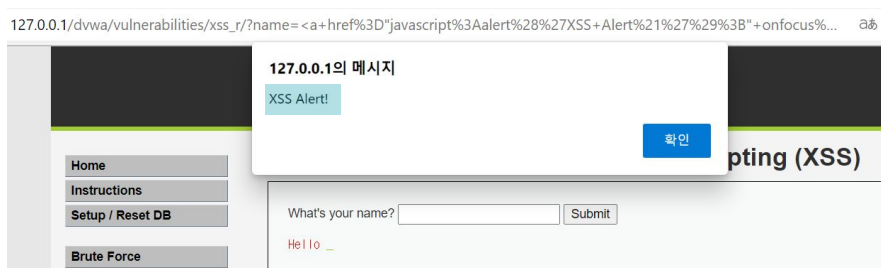
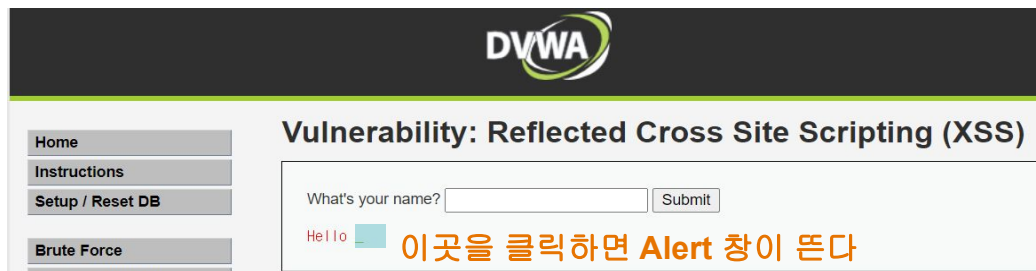
alert message

확인

## 03. WEB 취약점 공격 원리/방법 - XSS

### ■ Reflected XSS 실습

- a태그를 활용하여 페이지 내에 링크를 사용할 수 있음.
- Ex) `<a href="javascript:alert('XSS Alert!');" onfocus="this.blur()"> </a>`

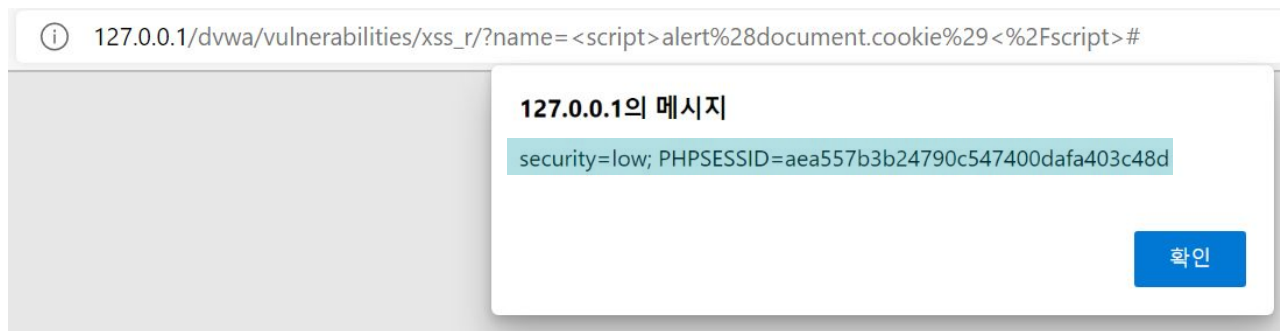




## 03. WEB 취약점 공격 원리/방법 – XSS

### ■ Reflected XSS 실습

- 본 페이지의 Cookie 값을 Alert 창으로 띄울 수 있음
- Ex) `<script>alert(document.cookie)</script>`



## 03. WEB 취약점 공격 원리/방법 - XSS

### ■ Reflected XSS 실습

- document.location을 이용하여 지정한 위치로 리다이렉트시키는 스크립트
- 공격자는 사용자가 눈치채지 못하도록 진짜같은 가짜 웹 페이지를 만들어서 리다이렉트 시킬 수도 있다.  
→ 회원 가입, 로그인 시에 개인정보 유출 발생
- Ex) `<script>document.location = 'https://www.naver.com/'</script>`

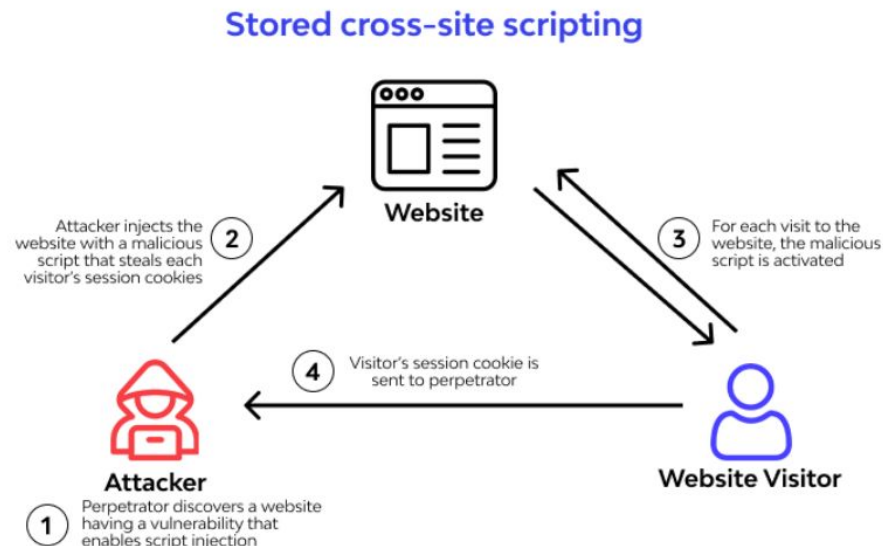


<리다이렉트 된 홈페이지>

## 03. WEB 취약점 공격 원리/방법 - XSS

### ■ Stored XSS

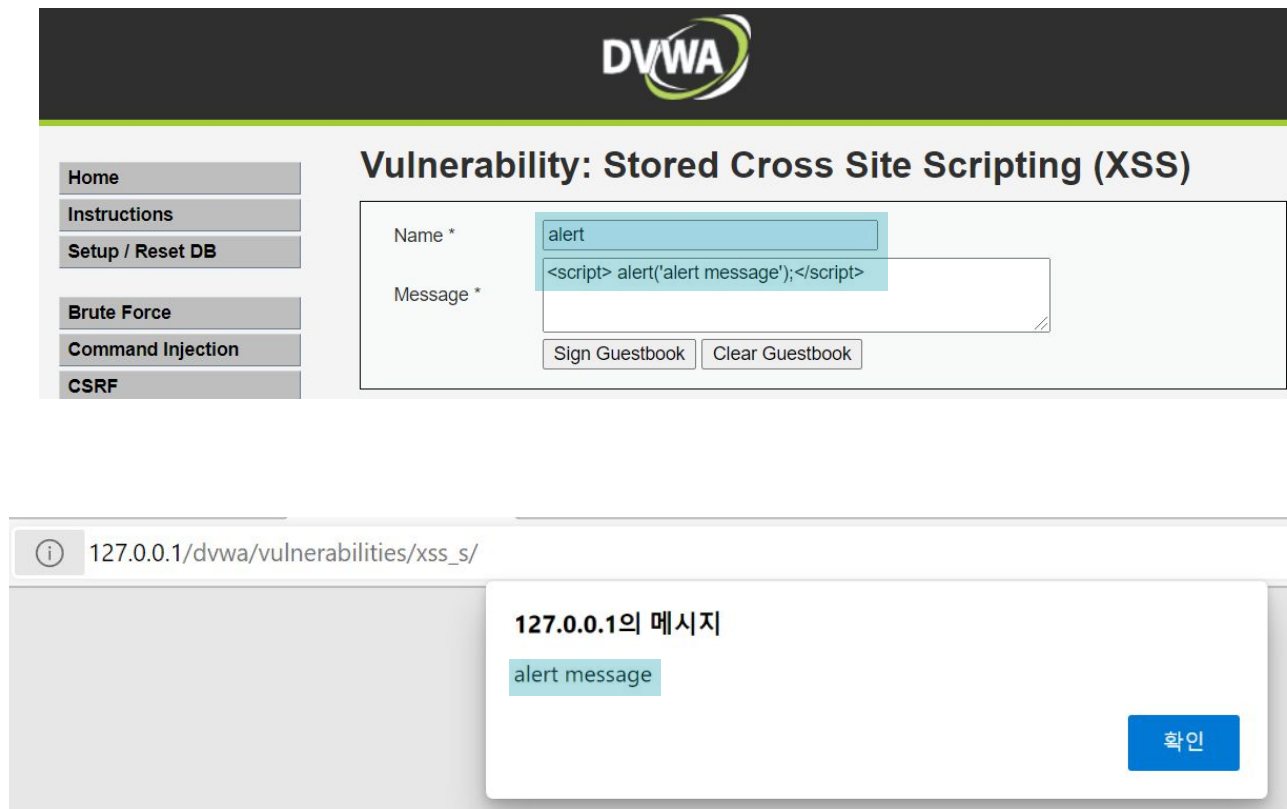
- 웹 브라우저에서 스크립트를 실행하는 것은 **Reflected XSS** 공격과 동일하나, 웹 서버에 스크립트를 저장했다가 실행된다는 차이점이 있음
- **Reflected XSS** 공격보다 공격 경로가 다양하기 때문에 더 위험하다고 판단할 수 있음
- 공격 방법
  - 1) 공격자는 **XSS** 공격 스크립트를 웹 사이트 방명록이나 게시판 등에 삽입한 후, 다른 사용자들이 방명록이나 게시판 등을 방문해 공격자가 작성한 게시물을 클릭하기를 기다림.
  - 2) 사용자가 공격자가 작성한 게시물을 실행하면 스크립트 코드가 사용자에게 전달됨. 이 때 웹 브라우저는 스크립트 코드를 실행해서 세션 쿠키가 공격자에게 전달됨.
  - 3) 공격자는 사용자로부터 받은 세션 쿠키를 사용해 사용자의 권한으로 웹 사이트 접속이 가능해짐.



## 03. WEB 취약점 공격 원리/방법 – XSS

### ■ Stored XSS 실습

- 게시판에 글을 작성하여 사용자가 클릭했을 때 Alert 창에 메시지를 띄울 수 있음
- Ex) `<script> alert('alert message');</script>`



The image shows a screenshot of the DVWA (Damn Vulnerable Web Application) interface. The top header displays the DVWA logo. On the left, there is a sidebar with navigation links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and CSRF. The main content area is titled "Vulnerability: Stored Cross Site Scripting (XSS)". It contains a form with two input fields: "Name \*" and "Message \*". The "Name \*" field contains the text "alert", and the "Message \*" field contains the payload `<script> alert('alert message');</script>`. Below the form are two buttons: "Sign Guestbook" and "Clear Guestbook".

Below the DVWA interface, a browser window is shown with the address bar displaying `127.0.0.1/dvwa/vulnerabilities/xss_s/`. A modal dialog box is open, titled "127.0.0.1의 메시지" (Message from 127.0.0.1). The dialog contains the text "alert message" and a blue button labeled "확인" (Confirm).

## 03. WEB 취약점 공격 원리/방법 – XSS

### ■ Stored XSS 실습

- 게시판에 글을 작성하여 사용자가 클릭했을 때 **alert**을 이용하여 본 페이지의 쿠키값을 출력할 수 있음
- Ex) `<script>alert(document.cookie)</script>`

**DVWA**

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

Home  
Instructions  
Setup / Reset DB  
Brute Force  
Command Injection  
CSRF  
File Inclusion

i 127.0.0.1/dvwa/vulnerabilities/xss\_s/

#### 127.0.0.1의 메시지

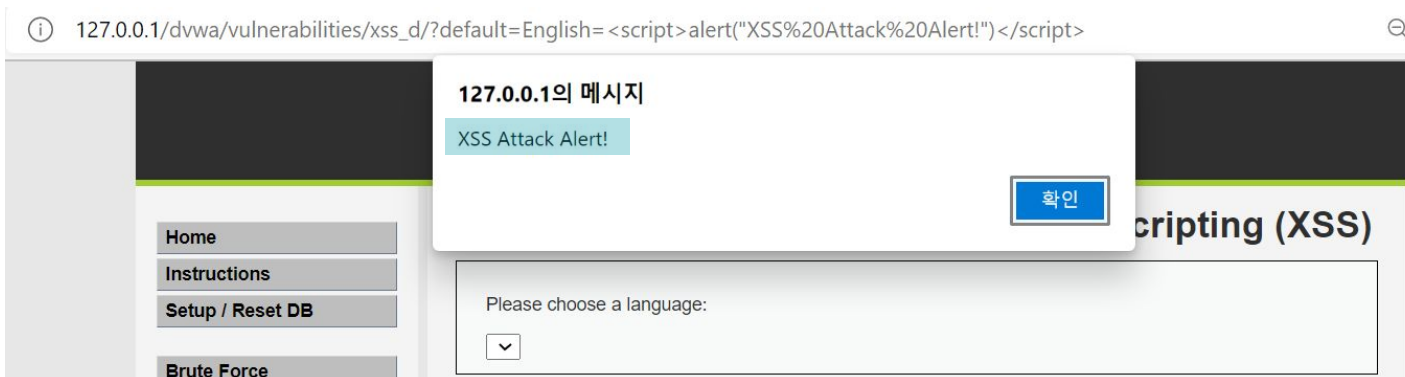
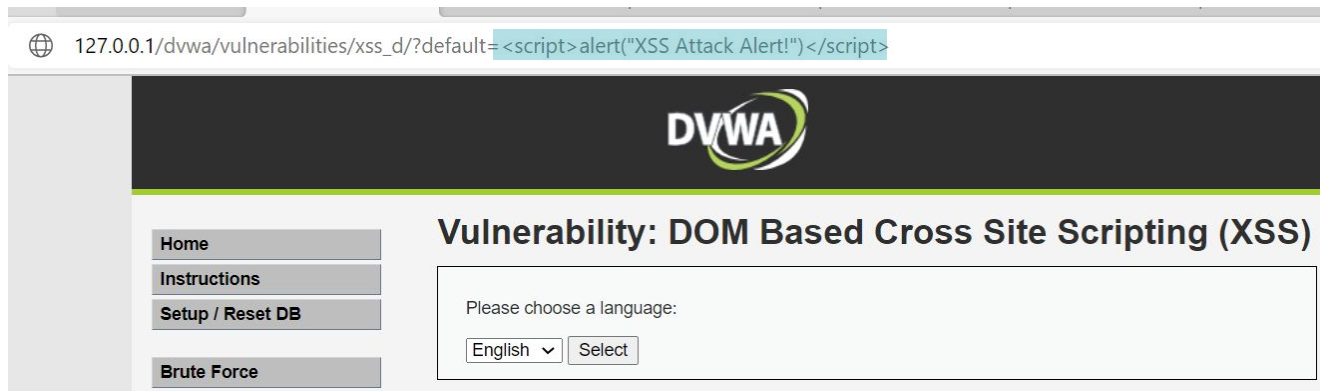
security=low; PHPSESSID=aea557b3b24790c547400dafa403c48d

확인

## 03. WEB 취약점 공격 원리/방법 – XSS

### ■ DOM XSS 실습

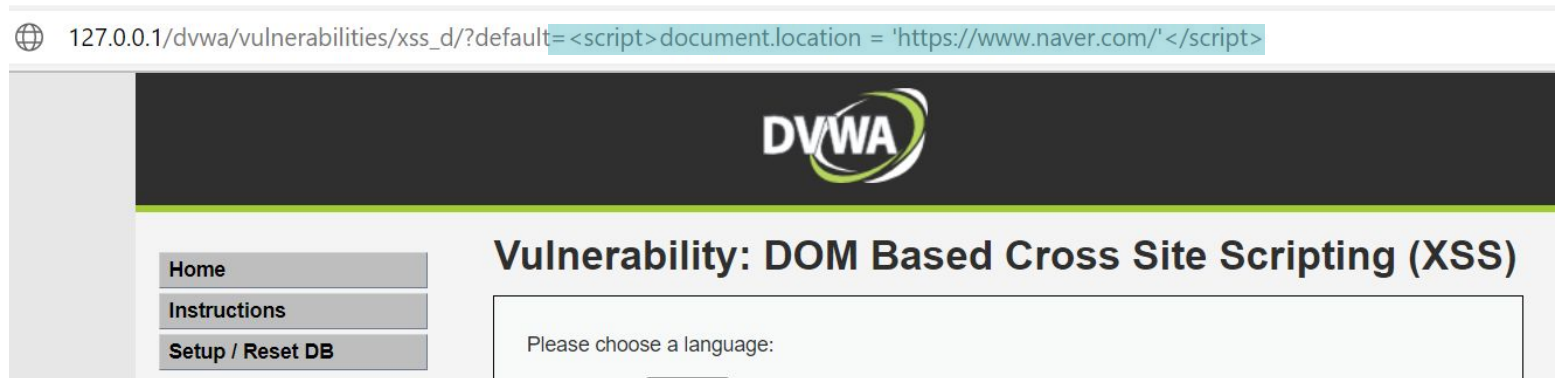
- 웹 브라우저의 주소창에 스크립트를 삽입하여 **Alert** 창을 띄울 수 있음
- Ex) `<script>alert("XSS Attack Alert!")</script>`



## 03. WEB 취약점 공격 원리/방법 - XSS

### ■ DOM XSS 실습

- 웹 브라우저의 주소창에 스크립트를 삽입하여 다른 사이트로 리다이렉트 시킬 수 있음.  
정교한 가짜 페이지를 만들어서 회원 가입, 로그인 유도로 개인 정보 탈취 가능
- Ex) `<script>document.location = 'https://www.naver.com/'</script>`

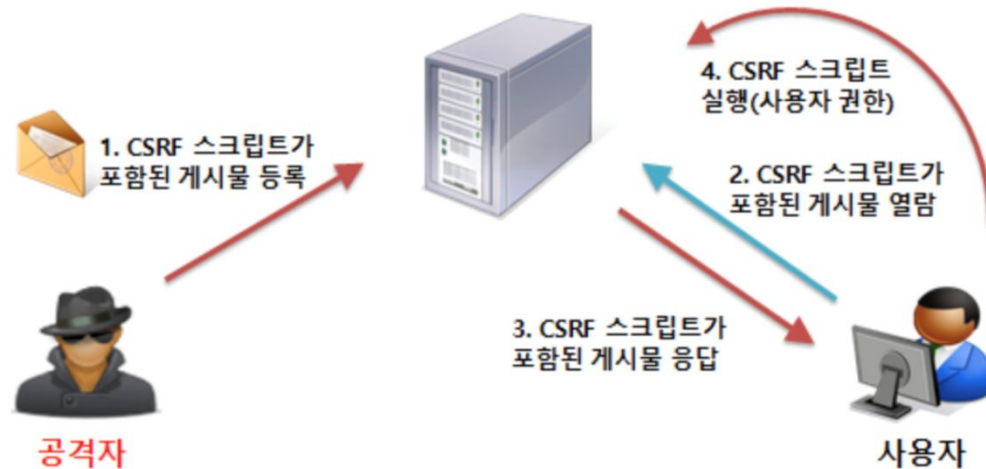


<리다이렉트 된 홈페이지>

## 03. WEB 취약점 공격 원리/방법

### ■ CSRF 공격이란?

- 사용자가 자신의 의지와 무관하게 공격자가 의도한 행동을 하여 특정 웹 페이지를 보안에 취약하게 한다거나 수정, 삭제 등의 작업을 하게 만드는 공격 방법
- 공격 방법
  - 1) 공격자는 이메일이나 게시판에 **CSRF** 스크립트가 포함된 게시물을 전송
  - 2) 관리자는 공격자가 등록한 **CSRF** 스크립트가 포함된 게시물을 확인
  - 3) 관리자가 **CSRF** 스크립트가 포함된 게시물을 열람하면, 관리자의 권한으로 공격자가 원하는 **CSRF** 스크립트 요청이 발생
  - 4) 공격자가 원하는 **CSRF** 스크립트가 실행되어, 관리자 및 사용자의 피해가 발생





## 03. WEB 취약점 공격 원리/방법

### ■ CSRF 공격 예시

- 페이스북에 글을 쓸 때 아래 코드와 같은 폼이 전송된다고 가정함.
- 피싱 사이트에 똑같이 페이스북에 글쓰기를 요청하는 폼이 숨겨져 있고, 그 내용으로 가입하면 10만원을 준다는 사기성 광고가 있음.
- 사용자가 피싱 사이트에 접속하여 가입함으로써 본인의 페이스북 계정으로 해당 글이 등록됨

```
<form action="http://facebook.com/api/content" method="post">
  <input type="hidden" name="body" value="여기 가입하면 돈 10만원 드립니다." />
  <input type="submit" value="Click Me"/>
</form>
```

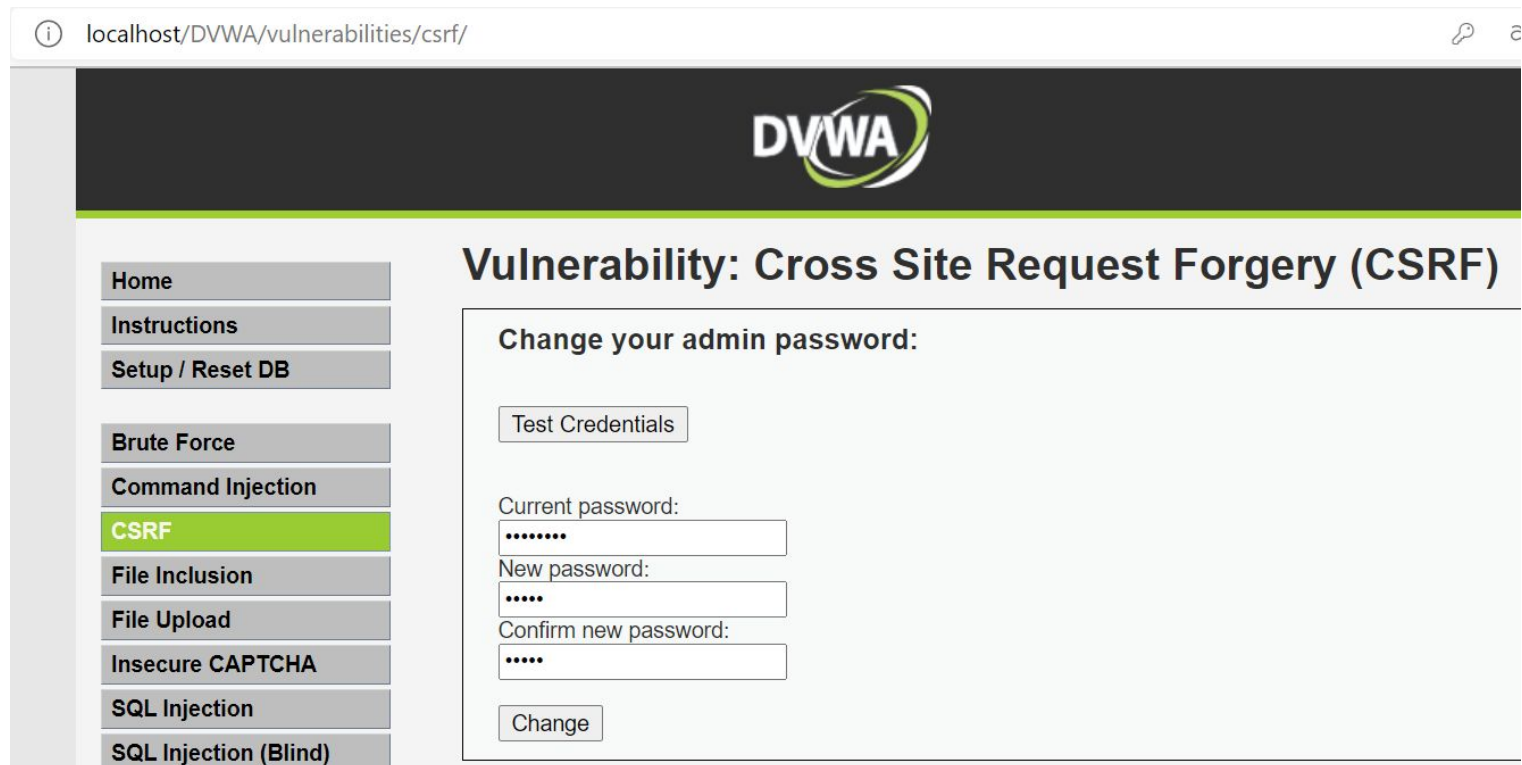
#### • XSS vs CSRF

	XSS	CSRF
공격 대상	클라이언트	서버
공격 방식	공격자가 Script를 이용하여 직접 공격	서버에서 제공하는 기능을 도용
Script 사용 여부	반드시 사용	<ul style="list-style-type: none"><li>• GET 방식의 요청에는 사용하지 않아도 됨</li><li>• POST 방식에서는 &lt;img&gt; 태그를 쓸 수 없음 자동으로 요청해야 하므로 &lt;script&gt;를 사용</li></ul>
분석	XSS Injection Vector만 발견하면 공격	도용하고자 하는 기능의 요청/응답의 처리 방식을 분석해야 함
공격 탐지	감지 가능	감지 불가능. 실제 요청과 도용된 요청이 서버에서 보기에 차이가 없음

## 03. WEB 취약점 공격 원리/방법

### ■ CSRF 실습 1) DVWA admin 계정 비밀번호 변경

- DVWA의 admin 비밀번호를 CSRF로 변경하는 실습 페이지  
현재 비밀번호와 변경할 비밀번호 입력 후 **Change** 버튼 클릭하면 CSRF 공격을 할 수 있음
- 현재 비밀번호 : password  
변경할 비밀번호 : newps



localhost/DVWA/vulnerabilities/csrf/

**DVWA**

**Vulnerability: Cross Site Request Forgery (CSRF)**

Change your admin password:

Test Credentials

Current password:  
.....

New password:  
.....

Confirm new password:  
.....

Change

Home  
Instructions  
Setup / Reset DB  
Brute Force  
Command Injection  
**CSRF**  
File Inclusion  
File Upload  
Insecure CAPTCHA  
SQL Injection  
SQL Injection (Blind)

## 03. WEB 취약점 공격 원리/방법

### ■ CSRF 실습 1) DVWA admin 계정 비밀번호 변경

- 현재 비밀번호와 변경된 비밀번호가 URL 창에 표시되는 것을 확인할 수 있음

localhost/dvwa/vulnerabilities/csrf/?password\_current=password&password\_new=newps&password\_conf=newps...

**DVWA**

**Vulnerability: Cross Site Request Forgery**

Change your admin password:

Test Credentials

Current password:

New password:

Confirm new password:

Change

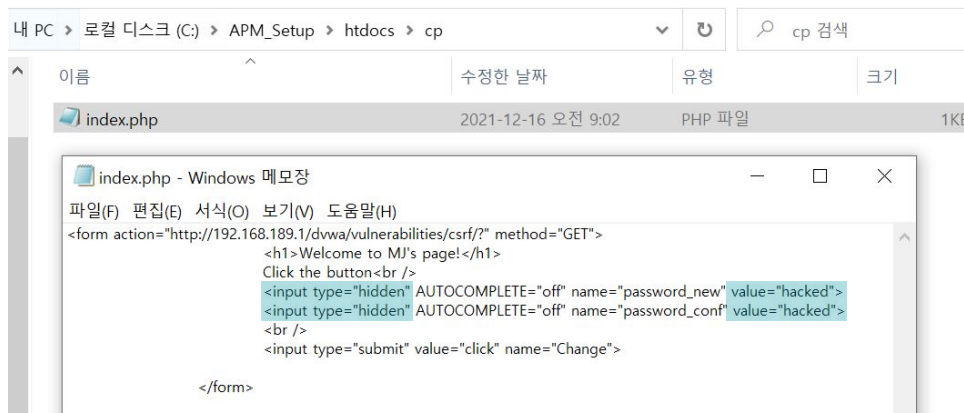
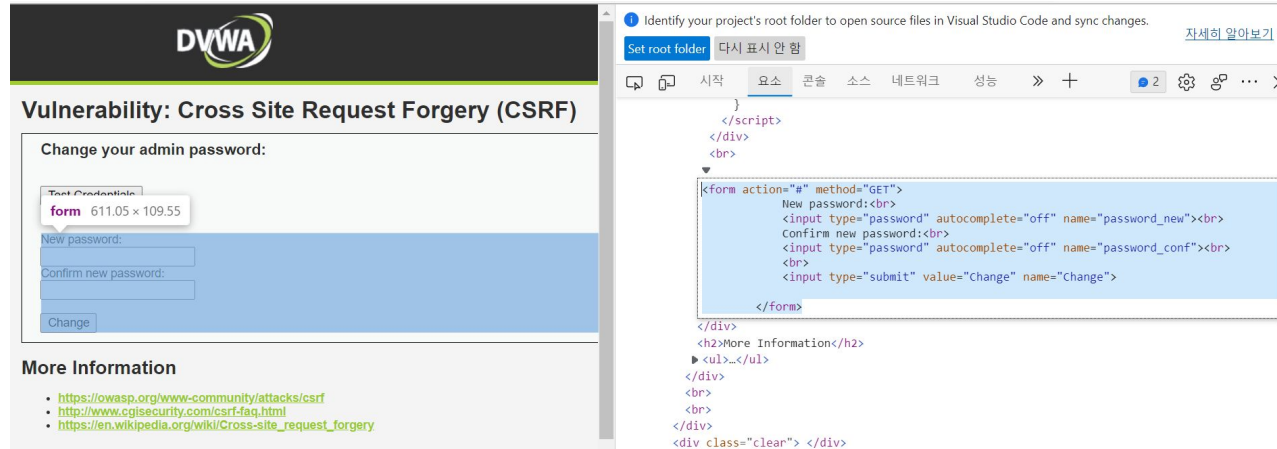
Password Changed.

Home  
Instructions  
Setup / Reset DB  
Brute Force  
Command Injection  
**CSRF**  
File Inclusion  
File Upload  
Insecure CAPTCHA  
SQL Injection  
SQL Injection (Blind)  
Weak Session IDs

### 03. WEB 취약점 공격 원리/방법

#### ■ CSRF 실습 2) 비밀번호를 변경하는 피싱 사이트

- DVWA의 CSRF 실습 페이지의 소스코드를 가져와 PHP로 간단한 피싱용 웹 페이지를 제작



- 비밀번호가 변경 칸을 숨기기 위해 input type을 "hidden"으로 설정
- 버튼을 클릭하면 비밀번호가 "hacked"로 변경됨

→ 버튼을 클릭하면 사용자는 자신도 모르게 비밀번호가 변경됨

<DVWA를 참고하여 제작한 웹 페이지>

## 03. WEB 취약점 공격 원리/방법

### ■ CSRF 실습 2) 비밀번호를 변경하는 피싱 사이트

- 제작한 웹 페이지를 본인에게 메일로 보냄

#### 링크 수정

표시할 텍스트:

링크 대상:

☒ 웹 주소

☐ 이메일 주소

어느 URL에 이 링크를 연결하시겠습니까?

[링크 테스트](#)

무엇을 입력해야 할지 잘 모르시겠습니까? 우선 링크할 웹페이지를 찾으세요. [검색 엔진](#)을 사용하면 편리합니다. 그런 다음 브라우저의 주소표시줄에 있는 주소를 복사하고 위 상자에 붙여넣습니다.

취소

확인

Welcome to my page!

Moonjeong Kwon

Welcome to my page!

Click [here](#) to enter my page

- from Moonjeong Kwon

≡  Gmail

🔍 메일 검색

 편지쓰기



받은편지함

724



별표편지함



다시 알림 항목



보낸편지함



Welcome to my page! 받은편지함 x



Moonjeong Kwon <englishsssh@gmail.com>

나에게 ▾

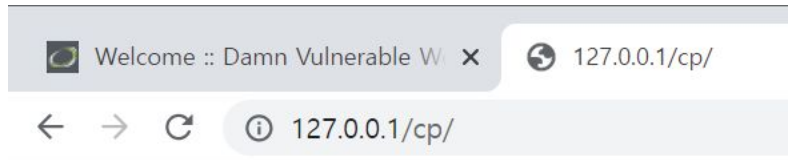
Click [here](#) to enter my page

<[here](#)를 클릭하면 피싱 페이지로 이동함>

## 03. WEB 취약점 공격 원리/방법

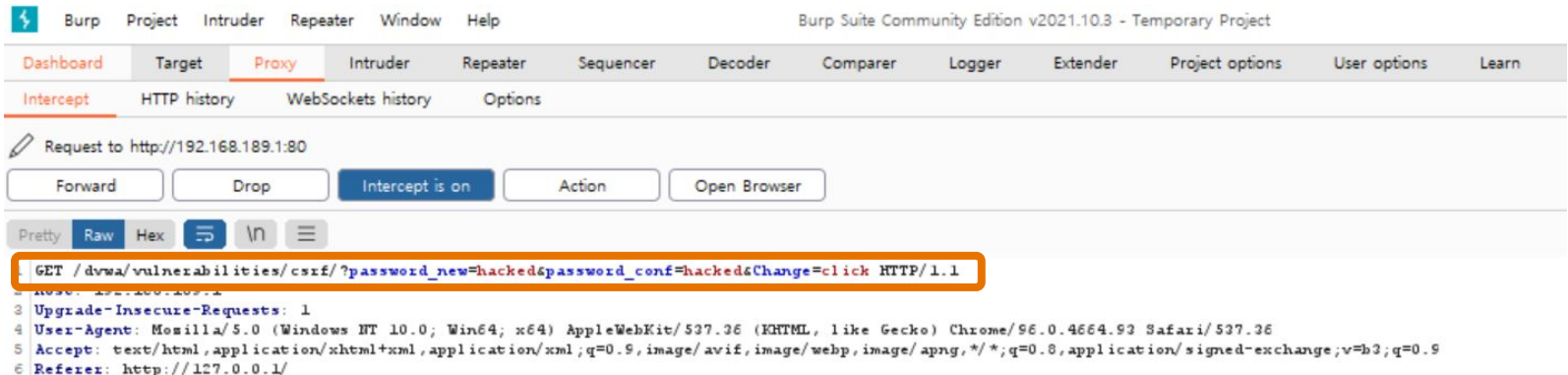
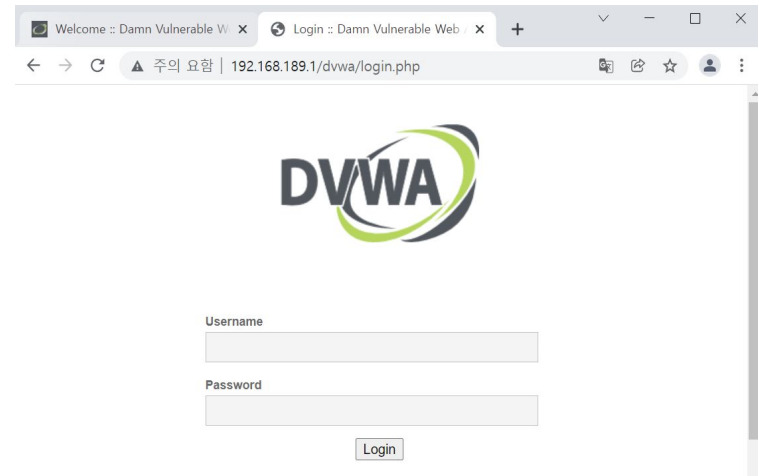
### ■ CSRF 실습 2) 비밀번호를 변경하는 피싱 사이트

- [here](#)를 클릭하면 직접 제작한 피싱용 웹 페이지로 이동
- [click] 버튼을 클릭하면 비밀번호가 바뀌며 DVWA 홈페이지로 이동함 (비밀번호 변경 여부는 Burp Suite로 확인)



Welcome to MJ's page!

Click the button



<Burp Suite로 비밀번호가 변경됨을 확인함>

## 04. 웹 취약점 탐지 및 대응 방법

### ■ SQL Injection의 탐지

- 로그 예시

```
/services/service.php?sno=113653&subsno=113653&sa=showcontent&lang=kr&sn=board&code=free&page=1'  
%20and%201=2%20union%20select%
```

```
page=1' and 1=2 union select ~~~~,CONCAT(0x27,0x7c,0x5f,0x7c)
```

- 위의 구문에서 **page** 파라미터의 값에 **Single Quote**(인용구)를 삽입하였음.  
즉, 해당 파라미터에 **String Type**의 **SQL Query**를 삽입하였다는 의미가 된다.
- 공격 구문을 삽입하여 강제 **Escape** 처리를 하였다면 사용자가 삽입한 **Single Quote**로 인하여 에러 발생.
- 문자열을 표현하기 위해서는 **Single Quote**를 쌍으로 써야 하지만 사용자가 삽입한 문자열로 인하여 **"** 처럼 **Single Quote**가 세 개가 되어서 에러가 발생한다는 의미이다.
- **"/services/service.php?sno=113653~"**에서 **GET**방식에 의한 요청임을 알 수 있음.
- 위의 공격 요청 구문의 길이를 보면 총 **1,024 characters**이다. **GET Method**를 이용해서 충분히 전송 가능한 데이터의 길이이며 또한 공격구문의 마지막을 보면 **~CONCAT(0x27,0x7c,0x5f,0x7c)**으로 구문이 끊겨 보이지도 않음.  
→ 로그를 관리하는 시스템의 로깅 길이가 **1024**로 제약되어 있거나 다른 이유로 절삭 되었을 수 있기 때문이다.  
그렇다면 **1,024** 보다 공격 구문의 길이가 좀더 길 수 있다고 예상 할 수 있음

## 04. 웹 취약점 탐지 및 대응 방법

### ■ SQL Injection의 탐지

- 구글링으로 3200건의 동일한 공격이 있었음을 알 수 있음.
- 완벽한 구문 획득  
page=1' and 1=2 union select CONCAT(0x27,0x7c,0x5f,0x7c),CONCAT(0x27,0x7c,0x5f,0x7c),CONCAT(0x27,0x7c,0x5f,0x7c) /\*



- CONCAT(Param1,Param2,Param3)로 3개의 파라미터를 사용
- 인터넷에서 구한 완성된 문장에는 파라미터에 /\* 를 사용하고 있음을 알 수 있음. 질의어에서 /\* \*/이용하여 주석 처리를 할 경우 쌍으로 사용해야 하나 단일 사용을 허용하는 DB는 MySQL에서만 가능  
→ 공격 대상 DB는 MySQL임을 알 수 있음



## 04. 웹 취약점 탐지 및 대응 방법

### ■ SQL Injection의 대응 1) 개발 관점

- 입력값 검증
  - 사용자의 입력이 DB Query 에 동적으로 영향을 주는 경우, 입력된 값이 개발자가 의도한 값인지 검증하기  
Ex. 영향을 줄 수 있는 문자들) /\*, -, ', ", ?, #, (, ), ;, @, =, \*, +, union, select, drop, update, from, where, join, substr, user\_tables, user\_table\_columns, information\_schema, sysobject, table\_schema, declare, dual,...
  - 공백 대신 키워드와 의미 없는 단어로 치환 되어야 함  
Ex) 공격자가 SESELECTLECT 라고 입력 시中间的 SELECT가 공백으로 치환이 되면 SELECT 라는 키워드가 완성됨
  - 종류 : 입력값의 타입과 패턴 검증, 영향을 줄 수 있는 특수 문자 검증
- 특정한 데이터를 받을 때 빈 문자열 여부, 길이 검증하기  
Ex) 휴대폰 번호 입력 시의 입력값 검증

```
import re

def is_valid_phone(phone_number):
    # None 혹은 빈 문자열인지 체크한다
    if phone_number is None or not len(phone_number):
        return False

    # 정확히 10자리의 유효한 핸드폰 번호 문자열만을 포함한다
    phone_format = re.compile(r'^\w{10}$')
    return phone_format.match(phone_number) is not None
```

## 04. 웹 취약점 탐지 및 대응 방법

### ■ SQL Injection의 대응 1) 개발 관점

- 영향을 줄 수 있는 특수 문자 피하기  
Ex) 슬래쉬, 단일따옴표와 쌍따옴표, 개행문자(\n 과 \r), 그리고 null(\0) 피하기 (Python)

```
def escape_input(input):
    char_replacements = [
        # 다른 특수 문자들에 있는 역슬래시를 거르지 않고 특수문자들을 거르기 위해
        # 역슬래시를 먼저 걸러줘야 한다
        ('\\', '\\\\'),
        ('\'', '\\\''),
        ('\\0', '\\\\0'),
        ('\\n', '\\\\n'),
        ('\\r', '\\\\r'),
        ('\"', '\\\"'),
    ]

    for char_to_replace, replacement in char_replacements:
        if char_to_replace not in input:
            continue
        input = input.replace(char_to_replace, replacement)

    return input
```

- Ex) 특수문자 Escape spquence를 걸러내는 함수 사용하기 (Python)

```
import _mysql

phone_param = _mysql.escape_string(phone_input)
```

## 04. 웹 취약점 탐지 및 대응 방법

### ■ SQL Injection의 대응 1) 개발 관점

- 데이터베이스 에러 메시지 사용자에게 보여주지 않기  
에러메시지는 데이터에 대해 많은 정보를 공격자에게 제공할 수 있다.  
따라서 사용자에게는 일반적인 데이터베이스 에러 메시지를 제공하고, 개발자가 해당 오류에 접근할 수 있는 상세 에러를 로그로 남긴다.
- 데이터베이스 권한 제한하기  
사용자가 하지 않아도 될 권한(ex. 데이터베이스 삭제)을 허용하지 않는다.  
사용자에게 필요한 최소한의 권한만 준다.
- 동적 쿼리 사용하지 않기
  - PreparedStatement 사용 예시(Python)

```
from mysql import connector

# 데이터베이스와 연결하고 커서를 인스턴스화한다.
cnx = connector.connect(database='bakery')
cursor = cnx.cursor(prepared=True)

statement = "SELECT * FROM customers WHERE phone = ?"
cursor.execute(statement, (phone_input, ))
```

## 04. 웹 취약점 탐지 및 대응 방법

### ■ SQL Injection의 대응 1) 개발 관점

- 동적 쿼리 사용하지 않기
  - 저장 프로시저 사용 예시(Python)
- input\_phone이라는 스트링 파라미터를 받아오는 get\_customer\_from\_phone()이라는 저장 프로시저를 생성

```
DELIMITER //
```

```
CREATE PROCEDURE get_customer_from_phone
```

```
(IN input_phone VARCHAR(15))
```

```
BEGIN
```

```
    SELECT * FROM customers
```

```
    WHERE phone = input_phone;
```

```
END //
```

```
DELIMITER;
```

- 저장 프로시저 호출(Python)

```
from mysql import connector
```

```
# 데이터베이스와 연결하고 커서를 인스턴스화한다.
```

```
cnx = connector.connect(database='bakery')
```

```
cursor = cnx.cursor()
```

```
cursor.callproc('get_customer_from_phone', args=(phone_input,))
```

## 04. 웹 취약점 탐지 및 대응 방법

### ■ SQL Injection의 대응 2) 서버 관점

- 서버 단에서 화이트리스트 기반으로 검증해야 함  
→ 블랙리스트 기반으로 검증하게 되면 수많은 차단리스트를 등록해야 하고, 하나라도 빠지면 공격에 성공하게 되기 때문
- 네트워크 기반 혹은 호스트 기반의 침입 탐지 시스템(Intrusion Detection Systems, IDS)을 튜닝해 SQL 인젝션 공격을 탐지할 수 있음.  
네트워크 기반 IDS는 데이터베이스 서버의 모든 연결을 모니터링하고 의심스러운 활동에 플래그를 지정할 수 있음. 호스트 기반 IDS는 웹 서버 로그를 모니터링하고 특이 사항이 발생하면 경고할 수 있다.

### ■ SQL Injection의 대응 3) 보안 장비 관점

- 웹 방화벽 사용  
웹 방화벽은 소프트웨어 형, 하드웨어 형, 프록시 형 이렇게 세가지 종류로 나눌 수 있음.
  - 소프트웨어 형 : 서버 내에 직접 설치
  - 하드웨어 형은 네트워크 상에서 서버 앞 단에 직접 하드웨어 장비로 구성
  - 프록시 형은 DNS 서버 주소를 웹 방화벽으로 바꾸고 서버로 가는 트래픽이 웹 방화벽을 먼저 거치도록 하는 방법

## 04. 웹 취약점 탐지 및 대응 방법

### ■ XSS의 탐지

- 일반적인 HTML 삽입 점검

- 서버의 응답에 HTML을 삽입하려면 입력에서 HTML을 받아들여야 하며, 모든 URL 매개변수가 그 대상이 될 수 있음.
- 게시판이나 댓글이라면 제목, 본문, 이메일, 첨부파일 이름, 게시자 이름, 별명, 관련URL 등 조작할 수 있는 모든 값이 점검 대상이 됨
- 부등호(<, >) 문자의 처리를 확인하여 HTML 삽입 여부를 확인
- 이 두 문자를 다른 값으로 변경하지 않는 대부분의 웹 애플리케이션은 HTML 삽입 취약점이 발생할 수 있음.

- 탐지 예시 1)

```
GET http://192.168.189.238:8080/WebGoat/attack?Screen=1330&menu=900&Username=00000&SUBMIT=Search HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: */*
Accept-Language: en-US,en;q=0.5
X-Requested-With: XMLHttpRequest
Referer: http://192.168.189.238:8080/WebGoat/start.mvc
Cookie: JSESSIONID=CE447BFD08F5694FD087123CCB888CB1
Connection: keep-alive
Content-Length: 0
Host: 192.168.189.238:8080
```

## 04. 웹 취약점 탐지 및 대응 방법

### ■ XSS의 탐지

- 검색어를 Username 변수에 GET 방식으로 전달
- Username 변수에 HTML 입력이 가능
- HTML 삽입이 가능한 것을 알았으므로 자바스크립트 입력이 가능한지 다음 문장으로 실험  
→ <script>alert(document.cookie)</script>

```
root@kali:~# curl --cookie "JSESSIONID=CE447BFD08F5694FD087123CCB888CB1" "http://192.168.189.238:8080/WebGoat/attack?Screen=1330&menu=900&Username=%0A%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E%0A&SUBMIT=Search" 2> /dev/null | grep document.cookie
<script>alert(document.cookie)</script>
<script>alert(document.cookie)</script>
root@kali:~#
```

- 위의 예에서 정상적으로 <script> 엘리먼트가 삽입된 것을 확인할 수 있음.  
즉, Username 변수에는 HTML 삽입 취약점이 있으며 XSS 공격이 가능하다.

## 04. 웹 취약점 탐지 및 대응 방법

### ■ XSS의 대응 1) 개발 관점

- 입력 값 제한

사용자의 입력값을 제한하여 스크립트를 삽입하지 못하게 제한해야 함.

Ex) `http://www.qubitsec.io?page=1` 위와 같은 URL 에서 사용자의 입력값인 **page** 가 화면에 바로 뿌려지는데, 이때 **page** 의 값을 숫자만으로 허용한다면 그 외의 입력값에는 출력되지 않기 때문에 대응 가능

- 스크립트 영역에 출력 자제

이벤트 핸들러 영역에 스크립트가 삽입되는 경우 보호기법들을 우회할 수 있기 때문에 사용자의 입력을 출력하는 것을 최대한 자제해야 하며, 필요한 경우 대응 방안과 함께 사용

### ■ XSS의 대응 2) 서버 관점

- 입력 값 치환

XSS 공격은 기본적으로 `<script>` 태그를 사용하기 때문에 XSS 공격을 차단하기 위해 태그 문자(`<`, `>`) 등 위험한 문자 입력 시 문자 참조(HTML entity)로 필터링하고, 서버에서 브라우저로 전송 시 문자를 인코딩

**\*HTML 문자 참조란?** ASCII 문자를 동일한 의미의 HTML 문자로 변경하는 과정

Ex) 문자 `"<"`는 동일한 의미의 HTML `"&lt;"` 로 변경

HTML 엔터티는 대부분의 인터프리터(특히, 브라우저)에서 특수한 의미를 가지지 않으며, 단순한 문자로 처리됨. 이렇게 인코딩하면 사용자는 `<script>`가 `<script>`로 보이지만 HTML 문서에서는 `&lt;script&gt;`로 나타나서 브라우저에서 일반 문자로 인식하고 스크립트로 해석되어 실행되지는 않음

ASCII 문자	참조 문자	ASCII 문자	참조 문자
&	&amp;	"	&quot;
<	&lt;	'	&#x27;
>	&gt;	/	&#x2F;
(	&#40;	)	&#41;

[그림 13] 위험 문자 인코딩



## 04. 웹 취약점 탐지 및 대응 방법

### ■ CSRF의 탐지

- CSRF는 XSS와 다르게 탐지가 불가능함.  
실제 요청과 도용된 요청이 서버에서 보기에 차이가 없기 때문

### ■ CSRF의 대응) 개발자 관점

- **make token & check**

JSP에서 랜덤 값을 생성해서 서버로 전송하거나, 로그인시 **auth token** 값을 생성한 후 생성된 그 토큰을 세션에 넣어둠.

화면 단에선 해당 토큰을 빼내어 **hidden**값에 넣어 놓음.

이런 식으로 가지고 있다가, **submit** 처리시 세션에 있는 값과 비교

#### 1. token값을 session 생성

로그인 시, 또는 작업화면 요청시 **CSRF** 토큰을 생성하여 세션에 저장

```
1 session.setAttribute("CSRF_TOKEN",UUID.randomUUID().toString()); // 요청 페이지에 CSRF 토큰을 셋팅하여 전송한다.
```

#### 2. 페이지의 hidden값에 넣기

```
1 <input type="hidden" name="_csrf" value="${CSRF_TOKEN}" />
```

## 04. 웹 취약점 탐지 및 대응 방법

### ■ CSRF의 대응

3. form이 전송되면, 컨트롤러나 비즈니스 로직에서 해당 토큰을 받아서 검사  
form을 통해서 전송된 놈인지 이상하게 넘어온 놈인지를 체크하기 위함

```
1 // 파라미터로 전달된 csrf 토큰 값
2 String param = request.getParameter("_csrf"); // 세션에 저장된 토큰 값과 일치 여부 검증
3 if (request.getSession().getAttribute("CSRF_TOKEN").equals(param)) {
4     return true;
5 } else {
6     response.sendRedirect("/"); return false;
7 }
```

# References

---

- 2021년 가장 위험한 10가지 웹 애플리케이션 보안취약점  
<https://www.ahnlab.com/kr/site/securityinfo/secunews/secuNewsView.do?seq=30748>
- SQL Injection의 공격 원리  
[SQL Injection이란? \(SQL 삽입 공격\) — 보안과 개발을 다 하고싶은 욕심쟁이 \(tistory.com\)](#)
- Stored XSS 공격  
[Security Analysis :: Stored XSS 공격 \(tistory.com\)](#)
- Blind SQL Injection  
[Security Analysis :: Blind SQL 인젝션 \(tistory.com\)](#)
- DVWA 웹 취약점 모의 훈련 웹 애플리케이션  
[DVWA 웹 취약점 모의 훈련 도구 리뷰 – JHYEON](#)
- CSRF의 공격과 방어 기법  
[CSRF 공격과 방어 기법 \(velog.io\)](#)
- SQL Injection 탐지 로그  
[N3015M INNOVATION\[PERSONAL LABORATORY\] :: \[관제\]SQL Injection 로그분석 사례1 \(tistory.com\)](#)
- HTML 삽입 취약점 점검 방법  
[HTML 삽입, XSS 공격 탐지방법 \(webhack.dynu.net\)](#)
- Snort를 활용한 XSS 공격 탐지  
[\[네트워크\] 스노트\(snort\) content를 사용한 XSS 공격 탐지 \(tistory.com\)](#)

# References

---

- XSS 대응 방안  
[XSS 대응방안 – PLURA'S BLOG](#)
- CSRF에 대하여  
<https://m.blog.naver.com/pinkpitapat/22141855691>