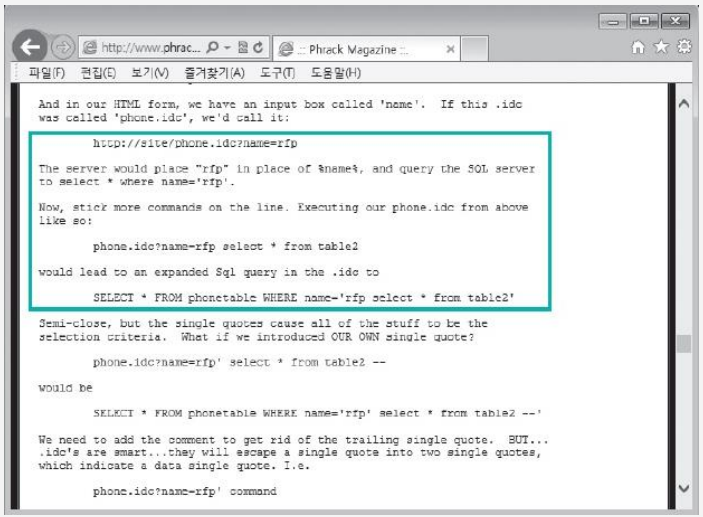
1 SQL 공격의 역사

1 SQL 인젝션 공격의 역사

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

1 SQL 인젝션 공격의 역사

[<Phrack> 54호에 실린 SQL 인젝션 공격의 개념]
(http://www.phrack.org/archives/issues/54/8.txt)



(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

1 SQL 인젝션 공격의 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 대학교의 온라인 시스템이 SQL 인젝션 취약점에 노출
- 2006년 12월 악의적인 해커가 UCLA의 학생 기록 80만 건 열람(SQL Injection)

2 SQL의 개념

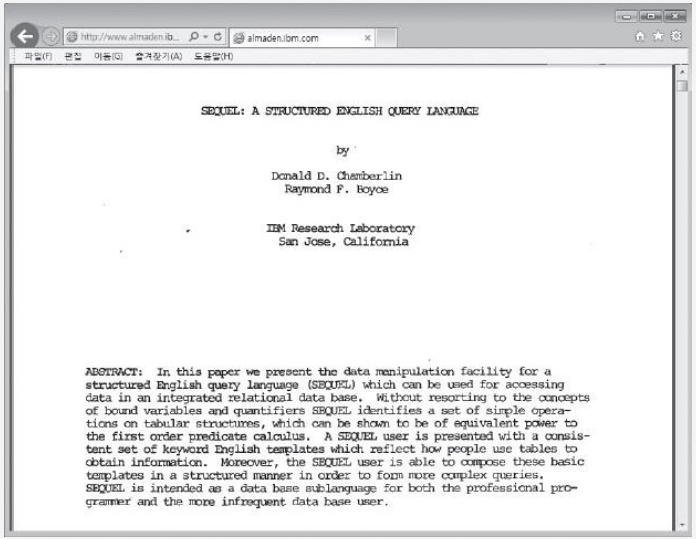
- 관계형 데이터베이스 관리 시스템(RDBMS)의 데이터를 관리하기 위해 설계된 특수 목적의 프로그래밍 언어
- 대다수 상용 또는 공개 데이터베이스 관련 프로그램은 SQL을 표준으로 함
- 1974년에 IBM의 도널드 챔벌린과 레이먼드 보이스가 개발(비절차적 언어)

2 SQL의 개념

- 초기의 명칭은 SEQUEL(구조화 영어 질의어)이었는데, SEQUEL이 영국 호커시들리 항공사의 상표라 SQL로 이름을 바꿈
- SQL:2011에 대한 SQL Framework 표준 문서 확인
  http://www.jtc1sc32.org/doc/N2151-2200/32N2153T-text_for_ballot-FDIS_9075-1.pdf

2 SQL의 개념

[SEQUEL 초기 문서]



SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California

ABSTRACT: In this paper we present the data manipulation facility for a structured English query language (SEQUEL) which can be used for accessing data in an integrated relational data base. Without resorting to the concepts of bound variables and quantifiers SEQUEL identifies a set of simple operations on tabular structures, which can be shown to be of equivalent power to the first order predicate calculus. A SEQUEL user is presented with a consistent set of keyword English templates which reflect how people use tables to obtain information. Moreover, the SEQUEL user is able to compose these basic templates in a structured manner in order to form more complex queries. SEQUEL is intended as a data base sublanguage for both the professional programmer and the more infrequent data base user.

(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

3 SQL의 구분

① 데이터 조작 언어(Data Manipulation Language, DML)

SQL DML	설명
SELECT	데이터베이스에서 데이터를 추출한다.
UPDATE	데이터베이스에 있는 데이터를 수정한다.
DELETE	데이터베이스에서 데이터를 삭제한다.
INSERT	데이터베이스에 새로운 데이터를 추가한다.

(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

3 SQL의 구분

② 데이터 정의 언어(Data Definition Language, DDL)

SQL DDL	설명
CREATE DATABASE	새로운 데이터베이스를 생성한다.
ALTER DATABASE	데이터베이스를 수정한다.
CREATE TABLE	새로운 테이블을 만든다.
ALTER TABLE	테이블을 수정한다.
DROP TABLE	테이블을 삭제한다.
CREATE INDEX	인덱스(검색 키)를 만든다.
DROP INDEX	인덱스를 삭제한다.

(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

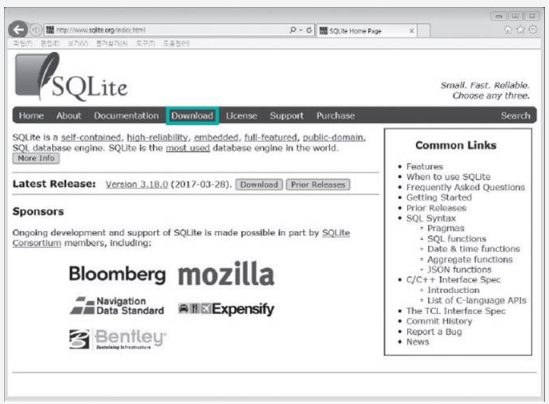
4 SQLite 설치하기

실습환경
- 윈도우 기반의 운영체제
- 필요 프로그램 : SQLite

4 SQLite 설치하기

① SQLite 사이트 접속

- http://www.sqlite.org에 접속 후 [Download] 메뉴 선택

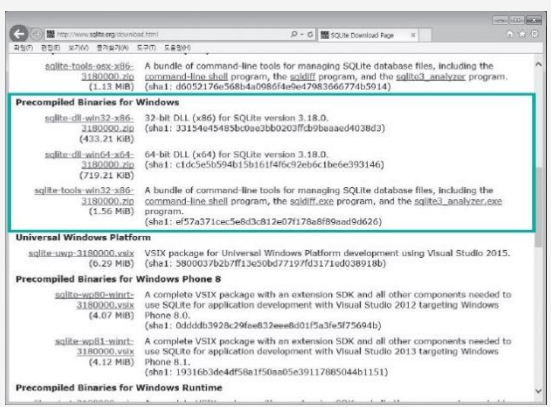


(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

4 SQLite 설치하기

② SQLite 파일 다운로드

- 'Precompiled Binaries for Windows'에 해당하는 파일을 모두 내려 받음



(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

4 SQLite 설치하기

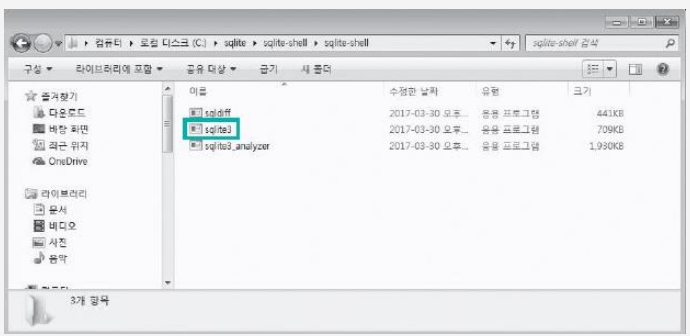
③ SQLite 파일 압축 풀기

- 내려 받은 SQLite 파일을 적당한 폴더에 옮기고 압축 풀기

4 SQLite 설치하기

④ SQLite 실행

- 편의상 ‘sqlite-tools-win32-x86-3180000’ 폴더명을 ‘sqlite-shell’로 변경
- SQLite 실행 파일인 sqlite3.exe 확인

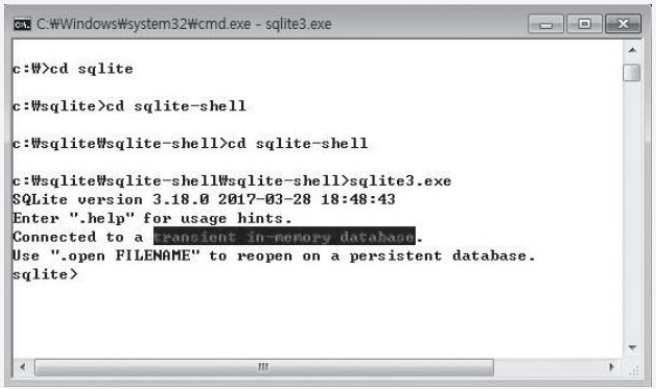


(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

4 SQLite 설치하기

④ SQLite 실행

- [시작]-[실행]에서 'cmd.exe'를 입력하여 도스창을 띄운 후 sqlite3.exe를 실행



(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

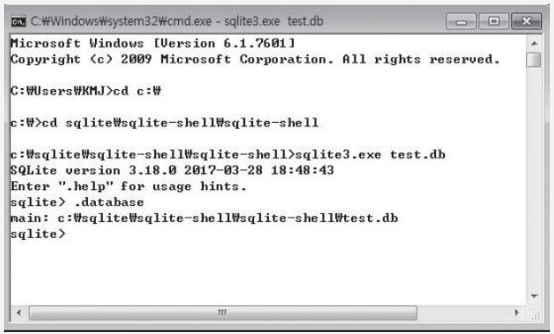
5 SQL 쿼리문 실습하기

실습환경
- 윈도우 기반의 사용자 운영체제
- 필요 프로그램 : SQLite

5 SQL 쿼리문 실습하기

① SQLite 실행 후 테스트 데이터베이스 생성

- [시작]-[실행]에서 'cmd.exe'를 입력 후 'sqlite3.exe test.db' 명령어를 실행
- '.database' 명령어를 실행하여 생성된 데이터베이스 확인



(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

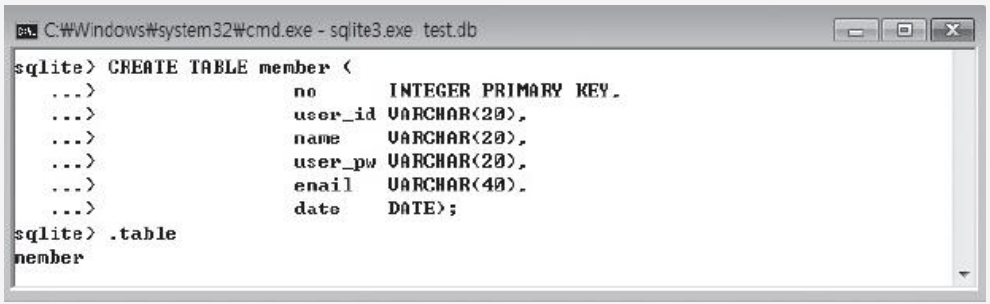
5 SQL 쿼리문 실습하기

② Create 명령어로 테이블 생성

- Create table 명령어로 member 테이블 생성
- '.table' 명령어로 member 테이블이 정상적으로 생성되었는지 확인

```
sqlite> CREATE TABLE member (
          no INTEGER PRIMARY KEY,
          user_id VARCHAR(20),
          name VARCHAR(20),
          user_pw VARCHAR(20),
          email VARCHAR(40),
          date DATE
    );
```

[member 테이블 생성]



(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

5 SQL 쿼리문 실습하기

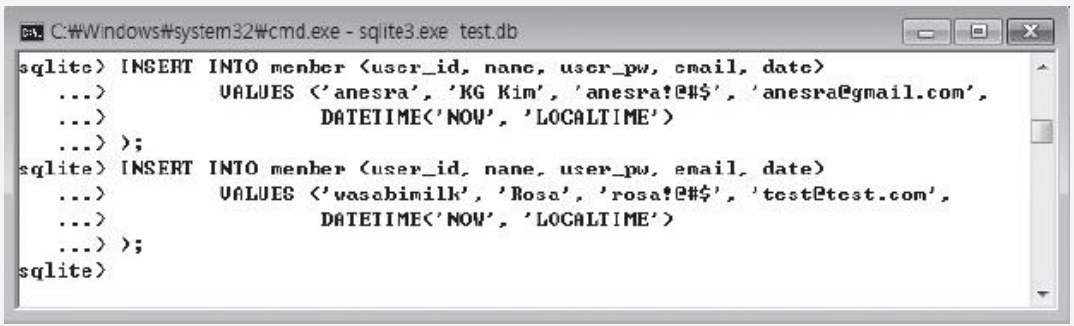
③ Insert 명령어로 데이터 입력

```
sqlite> INSERT INTO member (user_id, name, user_pw, email, date) VALUES
        ('anesra', 'KG Kim', 'anesra!@#$', 'anesra@gmail.com',
DATETIME('NOW',
        'LOCALTIME')
        );
```

```
sqlite> INSERT INTO member (user_id, name, user_pw, email, date) VALUES
        ('wasabimilk', 'Rosa', 'rosa!@#$', 'test@test.com', DATETIME('NOW',
        'LOCALTIME')
        );
```

5 SQL 쿼리문 실습하기

③ Insert 명령어로 데이터 입력



(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

5 SQL 쿼리문 실습하기

④ Select 명령어로 데이터 조회

- member 테이블의 모든 데이터 조회

```
sqlite> SELECT * FROM member;
```

- member 테이블에서 user_id가 'anesra'인 데이터 조회

```
sqlite> SELECT * FROM member WHERE user_id = 'anesra';
```

5 SQL 쿼리문 실습하기

④ Select 명령어로 데이터 조회

(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

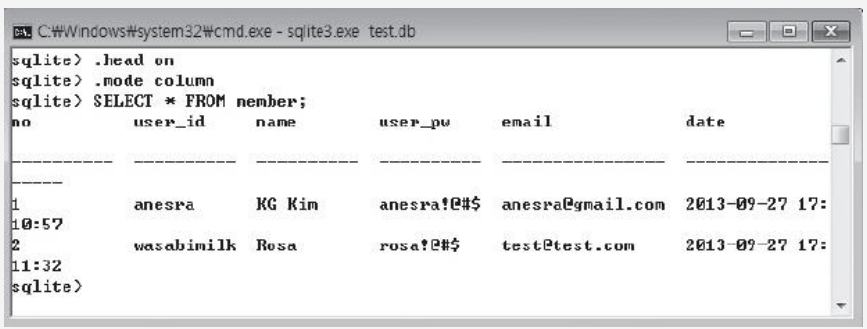
5 SQL 쿼리문 실습하기

④ Select 명령어로 데이터 조회

- 칼럼 데이터가 |(파이프)로 구분되어 있어 보기가 쉽지 않은 부분 수정

```
sqlite> .head on
sqlite> .mode column
```

[다양한 명령어로 **데이터 편집**]



(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

5 SQL 쿼리문 실습하기

⑤ Update 명령어로 데이터 수정

- Update : 데이터 내용을 수정할 때 사용

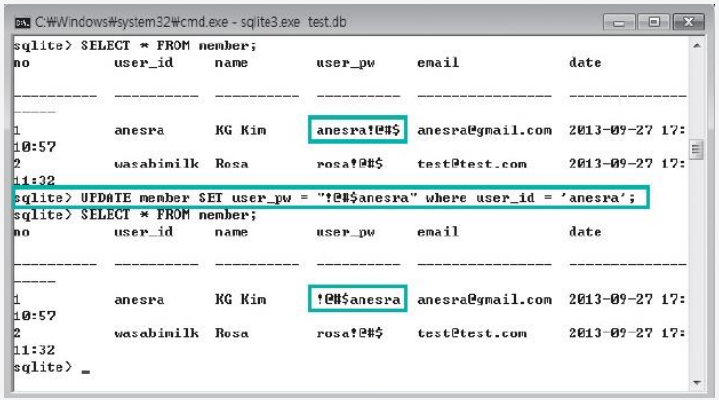
```
sqlite> UPDATE member SET user_pw = "!@#$anesra" WHERE user_id =
'anesra';
```

5 SQL 쿼리문 실습하기

⑤ Update 명령어로 데이터 수정

- Update : 데이터 내용을 수정할 때 사용

[Update 명령어로 패스워드 수정]



(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

5 SQL 쿼리문 실습하기

⑥ Delete 명령어로 데이터 삭제

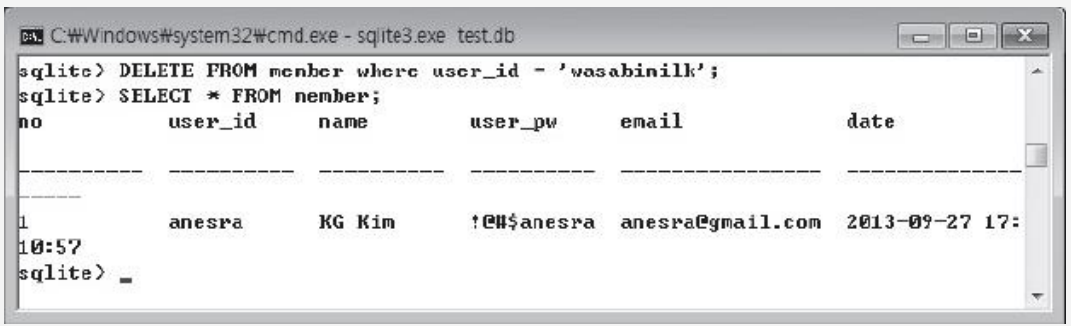
- Delete : 사용자의 데이터를 테이블에서 삭제할 때 사용

```
sqlite> DELETE FROM member WHERE user_id = 'wasabimilk';
```

5 SQL 쿼리문 실습하기

⑥ Delete 명령어로 데이터 삭제

- Delete : 사용자의 데이터를 테이블에서 삭제할 때 사용



(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

5 SQL 쿼리문 실습하기

⑦ Union 명령어로 여러 테이블의 내용 조회

- Union : 한 번에 여러 테이블의 데이터를 조회할 때 사용
- 관리자 아이디와 패스워드 정보를 포함하고 있는 admin 테이블 생성

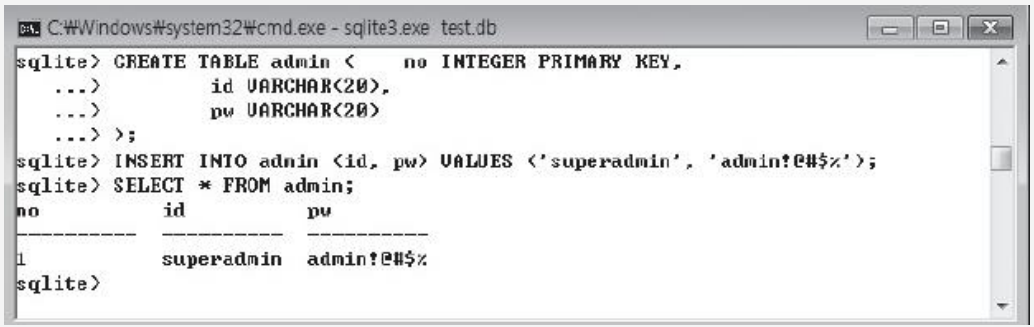
```
sqlite> CREATE TABLE admin (
          no INTEGER PRIMARY KEY,
          id VARCHAR(20),
          pw VARCHAR(20)
      );

sqlite> INSERT INTO admin (id, pw) VALUES ('superadmin', 'admin!@#$%');
```

5 SQL 쿼리문 실습하기

⑦ Union 명령어로 여러 테이블의 내용 조회

[admin 테이블 생성]



(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

5 SQL 쿼리문 실습하기

⑦ Union 명령어로 여러 테이블의 내용 조회

- '.table' 명령어로 admin과 member 두 개의 테이블이 있는 것을 확인 후 실행(칼럼 수 문제)

```
sqlite> SELECT * FROM member UNION SELECT * FROM admin;
```

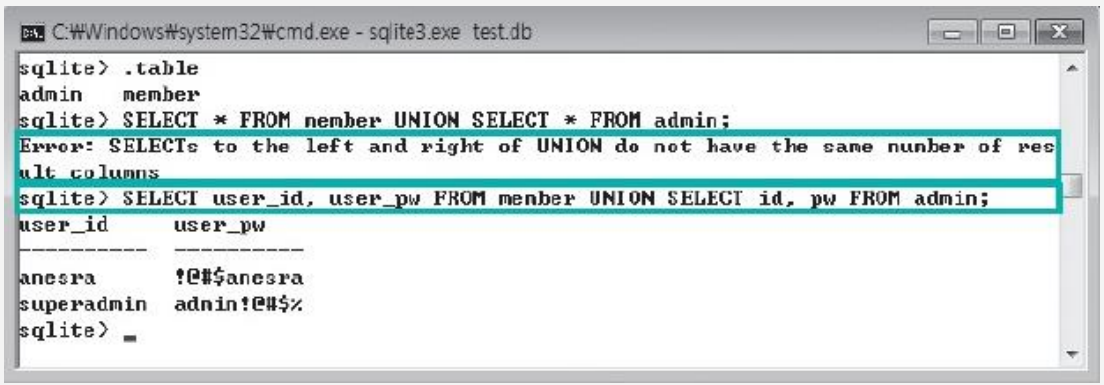
- Union 명령어를 이용하여 member와 admin의 동일한 칼럼 수를 조회

```
sqlite> SELECT user_id, user_pw FROM member UNION SELECT id, pw FROM admin;
```

5 SQL 쿼리문 실습하기

⑦ Union 명령어로 여러 테이블의 내용 조회

[Union 구문으로 두 개 테이블의 내용을 조회한 화면]



(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

2 SQL의 개념과 문법

1 SQL 인젝션 공격의 원리

- SQL 인젝션 공격에 취약점이 발생하는 곳은 웹 애플리케이션과 데이터베이스가 연동되는 부분에 공격자가 임의의 SQL 명령어를 삽입하여 공격 (참이면 된다)
- 보통 사용자 로그인 부분, 게시물 검색 부분, 우편번호 검색 부분, 자료실 등이 대표적(실제 예?)

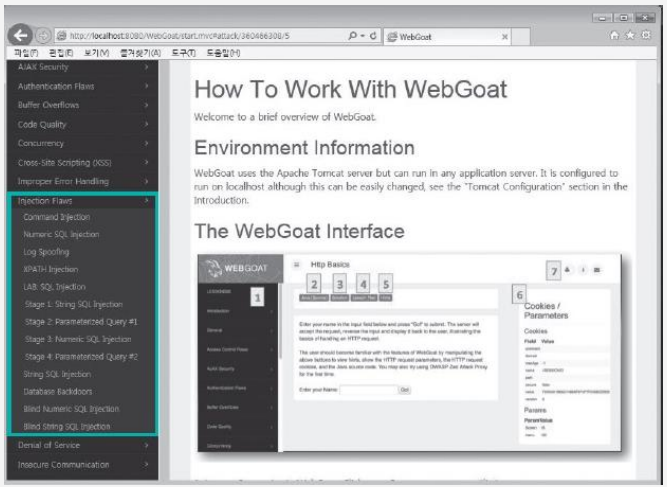
2 기본 SQL 인젝션 공격 연습하기

실습환경
- 윈도우 기반의 사용자 운영체제
- 필요 프로그램 : WebGoat

2 기본 SQL 인젝션 공격 연습하기

① WebGoat 실행

- WebGoat를 실행한 뒤
  WebGoat 사이트에 접속하여 로그인
- 화면 왼쪽의 [Injection Flaws] 메뉴 선택



[인젝션 테스트 화면]

(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

2 기본 SQL 인젝션 공격 연습하기

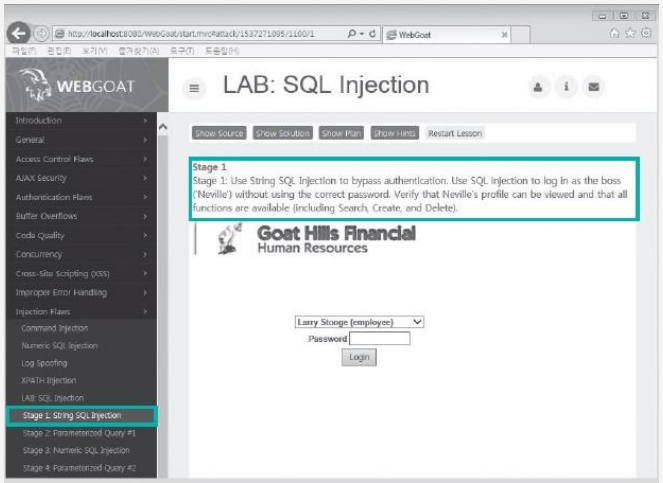
② [LAB: SQL Injection] - [Stage 1: String SQL Injection] 클릭

- 1단계

  인증을 우회하기 위해 SQL 문자열 삽입을 이용(SQL Injection)

  SQL 인젝션 공격을 통해 정확한 패스워드 없이 관리자('Neville')로 로그인

  Neville의 프로파일이 보이고 검색, 생성, 삭제 등의 모든 기능을 실행할 수 있는지 확인



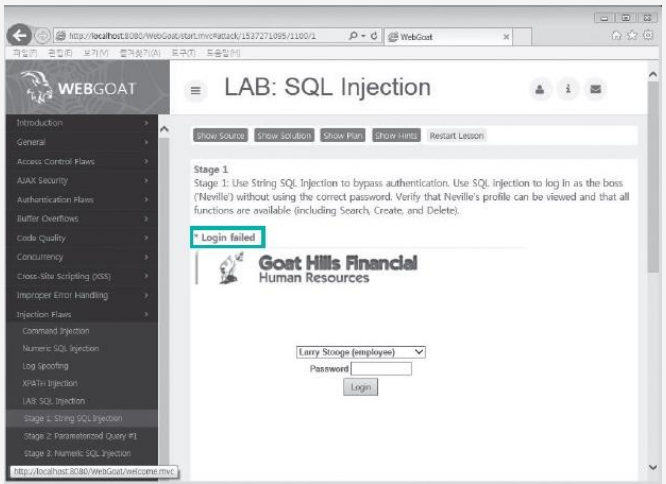
[SQL 문자열 삽입 테스트 화면]

(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

2 기본 SQL 인젝션 공격 연습하기

③ 간단한 인증 우회 SQL 문자열 삽입

- 'Larry Stooge(employee)'를 'Neville Bartholomew(admin)'을 선택하고 Password에 아무 값이나 입력



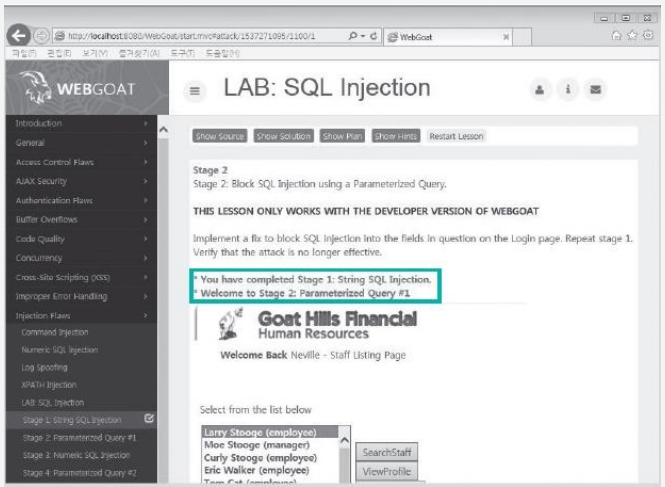
[LAB Stage 1의 문자열 삽입 로그인에 실패한 화면]

(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

2 기본 SQL 인젝션 공격 연습하기

③ 간단한 인증 우회 SQL 문자열 삽입

- 사용자로 다시 'Neville Bartholomew(admin)'을 선택하고, 패스워드 부분에 'or'='을 입력한 후 <Login>을 클릭(참만 만들면 된다)



[LAB Stage 1의 SQL 인젝션 공격에 성공한 화면]

(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

2 기본 SQL 인젝션 공격 연습하기

③ 간단한 인증 우회 SQL 문자열 삽입

- C:₩WebGoat7.1₩extract₩webapps₩WebGoat₩plugin_extracted₩org₩owasp₩webgoat₩plugin₩sqlinjection₩LoginSqlInjection.java 열기

```
114   public boolean requiresAuthentication()
115   {
116       return false;
117   }
118
119   public boolean login(WebSession s, String userId, String password)
120   {
121       // System.out.println("Logging in to lesson");
122       boolean authenticated = false;
123
124       try
125       {
126           String query = "SELECT * FROM employee WHERE userid = " + userId + " and password = '" + password + "'";
127           // System.out.println("Query: " + query);
128           try
129           {
130               Statement answer_statement = WebSession.getConnection(s)
131                       .createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
132               ResultSet answer_results = answer_statement.executeQuery(query);
133               if (answer_results.first())
134               {
135                   setSessionAttribute(s, getLessonName() + ".isAuthenticated", Boolean.TRUE);
136                   setSessionAttribute(s, getLessonName() + "." + SQLInjection.USER_ID, userId);
137                   authenticated = true;
138               }
139           } catch (SQLException sqle)
140           {
141               s.setMessage("Error logging in");
142               sqle.printStackTrace();
143           }
144       } catch (Exception e)
145       {
146           s.setMessage("Error logging in");
147           e.printStackTrace();
148       }
149
150       // System.out.println("Lesson login result: " + authenticated);
151       return authenticated;
152   }
153
```

[LAB Stage 1의 SQL 인젝션 java 파일의 소스코드]

(※출처: 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017)

2 기본 SQL 인젝션 공격 연습하기

③ 간단한 인증 우회 SQL 문자열 삽입

- 패스워드 입력값을 처리하는 소스코드(직접 입력)

```
SELECT * FROM employee WHERE userid = " + userid + " and password = '" + password+ "'
```

- 아이디와 패스워드에 각각
  Neville Bartholomew(admin), 'or'=' 입력

```
SELECT * FROM employee WHERE userid = 'Neville Bartholomew(admin)' and password ='or'='
```

  - 앞의 패스워드가 비어 있는 것은 틀린 값이지만, or 뒤에 나오는 문장이 참이 되기 때문에 해당 쿼리문의 전체 결과값은 참(참만 만들면 된다)