Special Report

웹 취약점과 해킹 매커니즘 #2 SQL Injection 개요

■ 개요

이번 '웹 취약점과 해킹 매커니즘'에서 다룰 첫 번째 취약점은 설계된 쿼리문에 의도하지 않은 미상의 쿼리를 임의로 삽입하여 악의적인 SQL 구문을 실행하는 공격인 'SQL Injection'이다. 해당 취약점을 통해 공격자는 데이터베이스에 직접적으로 접근해 중요 정보를 조회, 탈취할 수 있다. 그렇기 때문에 오픈소스 웹 애플리케이션 보안 프로젝트인 OWASP(The Open Web Application Security Project)의 Top 10 취약점 목록과 주요 정보통신 기반 시설 취약점 분석·평가 항목, 전자금융 기반 시설 취약점 분석/평가 항목 등의 다양한 취약점 진단 기준에 빠지지 않고 등장하고 있다.

이번 4월 호는 위 SQL Injection에 관한 내용을 다루기 전에 데이터베이스와 데이터베이스 관리용 언어인 SQL에 대한 내용을 먼저 설명할 예정이며, SQL Injection의 개념과 3가지 공격 유형 등 기본 개념도 설명할 예정이다.

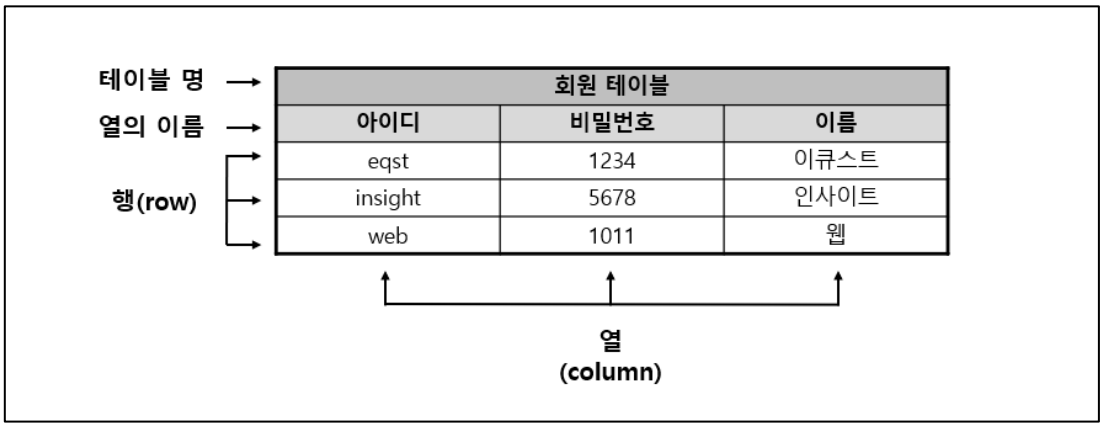
※ 실제 운영 중인 서버에 테스트 또는 공격을 하는 행위는 법적인 책임이 따르므로 개인용 테스트 서버 구축 또는 bWAPP, DVWA, WebGoat 등과 같은 웹 취약점 테스트 환경 구축을 통해 테스트하는 것을 권장한다.
※ 본 Special Report는 JSP와 Oracle Database 11gR2의 환경에서 실습을 진행한다.

■ 기본 개념

SQL Injection에 대한 내용을 진행하기에 앞서 데이터베이스와 데이터베이스 관리용 언어인 SQL(Structured Query Language)에 대해 알아보고자 한다.

Step1. 데이터베이스와 DBMS

데이터베이스란 데이터를 효율적으로 관리하기 위해 구조화한 데이터 집합을 뜻하며 DBMS(Data Base Management System)를 통해 운영 및 관리된다. 계층형, 망형, 관계형 등 다양한 종류의 DBMS가 존재하며 대부분의 DBMS는 관계형 DBMS(RDBMS)의 형태로 사용되고 있다. 관계형 DBMS의 데이터베이스는 하나 이상의 행(row)과 열(column)로 이루어진 테이블 형식으로 데이터를 제공한다.



[데이터베이스 테이블 구조]

대표적인 관계형 데이터베이스에는 ORACLE, MySQL, MS-SQL, PostgreSQL 등이 있다.

Step2. SQL (Structured Query Language)

SQL은 데이터베이스에서 질의, 수정, 삭제 등의 작업을 하는 데이터베이스 관리용 언어이며 대부분의 관계형 데이터베이스에서 사용한다. SQL이라는 언어를 통해 사용자가 원하는 데이터를 데이터베이스에 요청하는 행위인 쿼리(질의)를 작성한다.

쿼리 작성 시 주의해야 할 SQL의 언어적 특성은 다음과 같다.

1. 대소문자를 가리지 않는다.
2. SQL쿼리문은 반드시 세미콜론(;)으로 끝나야 한다.
3. 고유값을 가지는 문자열의 경우 홑따옴표(')로 감싸준다.
4. 주석[1]은 한 줄 주석의 경우 --로 나타내고 여러 줄 주석은 /* */로 감싸서 표현한다.

[1] 주석 처리를 하는 특수문자 뒤의 문장은 의미가 없어진다.

각 데이터베이스 별 주석 처리 방법은 다음과 같다.

infosec

데이터베이스	주석처리	
ORACLE MS-SQL	한 줄 주석	--
	여러 줄 주석	/* */
MySQL	한 줄 주석	-- 또는 #
	여러 줄 주석	/* */

SQL 문법은 사용 용도에 따라 데이터 정의어(DDL, Date Definition Language), 데이터 제어어(DCL, Data Control Language), 데이터 조작어(DML, Data Manipulation Language)로 구분된다.

infosec

종류	명령어	설명
데이터 정의어	CRATE ALTER DROP TRUNCATE	스키마, 테이블, 도메인, 뷰, 인덱스 생성/변경/삭제할 때 사용
데이터 제어어	COMMIT ROLLBACK GRANT REVOKE	데이터에 대한 접근 권한 부여 등 관리 목적으로 사용
데이터 조작어	SELECT INSERT UPDATE DELETE	데이터베이스에 저장된 데이터를 실질적으로 처리하는데 사용 (데이터 조회/추가/수정/삭제 등)

데이터베이스를 조회하고 관리하는 데이터 조작어가 가장 많이 쓰이며, 공격자의 주요 공격 포인트가 된다. INSERT, UPDATE, DELETE는 운영 중인 서버에 데이터를 추가, 수정, 삭제하는 영향을 줄 수 있기 때문에 사용에 주의해야 한다.

다음은 데이터 조작어의 사용법이다.

SELECT : 데이터베이스의 데이터를 조회하거나 검색하기 위한 명령어

```
SELECT [컬럼명] FROM [테이블명] WHERE [조건식];
```

INSERT : 데이터베이스에 데이터를 추가하기 위한 명령어

```
INSERT INTO [테이블명] VALUES [(데이터 값1, 데이터 값2, ...)];
```

UPDATE : 데이터베이스의 데이터 수정을 위한 명령어

```
UPDATE [테이블명] SET [필드이름1=데이터값1, 필드이름2=데이터값2...]
```

DELETE : 데이터베이스의 데이터 삭제를 위한 명령어

```
DELETE FROM [테이블명] WHERE [필드이름=데이터값];
```

■ SQL Injection

Step 1. 개념

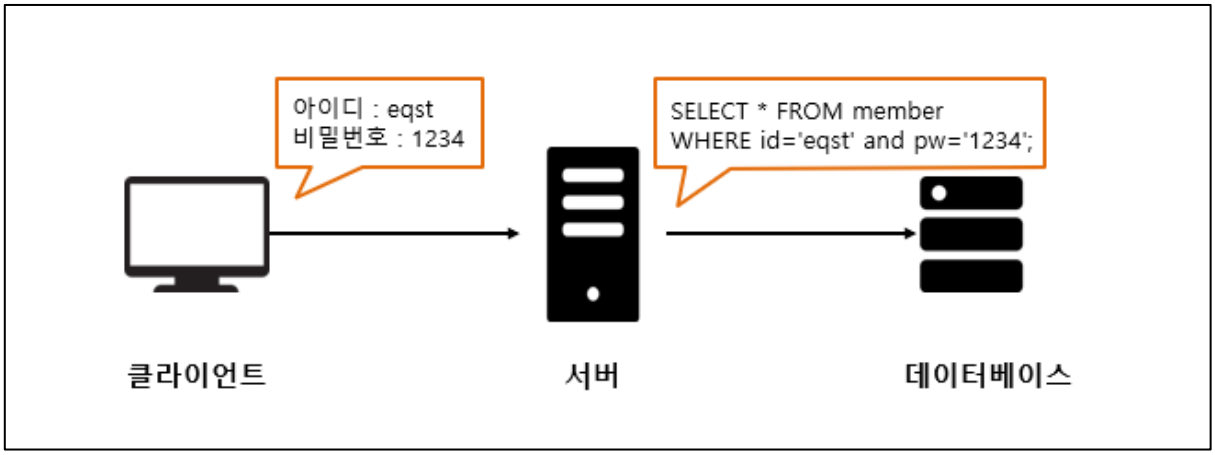
SQL Injection은 개발자가 설계한 쿼리문에 정상적인 SQL 구문이 아닌 악의적인 구문을 삽입(Injection)했을 때, 유효성 검증을 제대로 하지 않아 공격자의 의도대로 악의적인 SQL구문이 실행되는 공격이다. 웹 애플리케이션이 데이터베이스와 연동되어 있고 사용자가 입력한 값이 SQL 구문의 일부로 사용되는 환경에서 발생할 수 있다.

취약점은 사용자가 입력하는 로그인 또는 검색 기 등의 입력 폼에 SQL의 문법적 의미를 갖는 홑따옴표(') 입력 시 아래 그림과 같이 SQL 에러 메시지를 반환한다면 SQL Injection 공격이 가능한 것으로 판단할 수 있다.

[로그인 입력값으로 홑따옴표(') 입력]

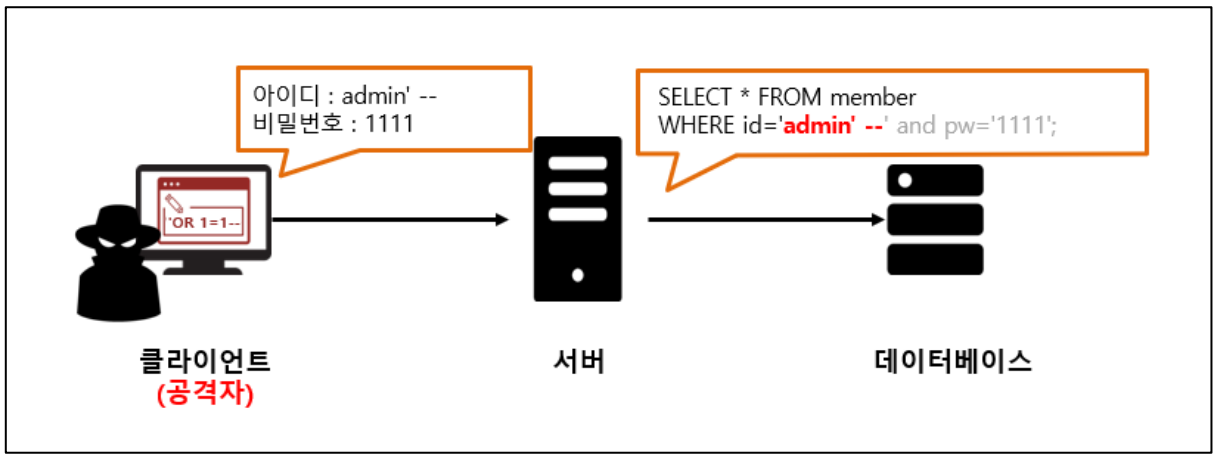
[SQL 문법 오류로 인한 에러 메시지]

아래는 일반 사용자가 웹 사이트에 등록된 자신의 아이디와 비밀번호로 로그인하는 과정에서 사용자 입력 값인 아이디와 패스워드가 서버 측의 SQL 구문에서 어떻게 동작하는지를 나타낸 것이다. 데이터베이스의 MEMBER 테이블의 값과 사용자가 입력한 아이디와 비밀번호가 일치하는 경우 정상적으로 로그인이 완료된다.



[정상적인 로그인 요청]

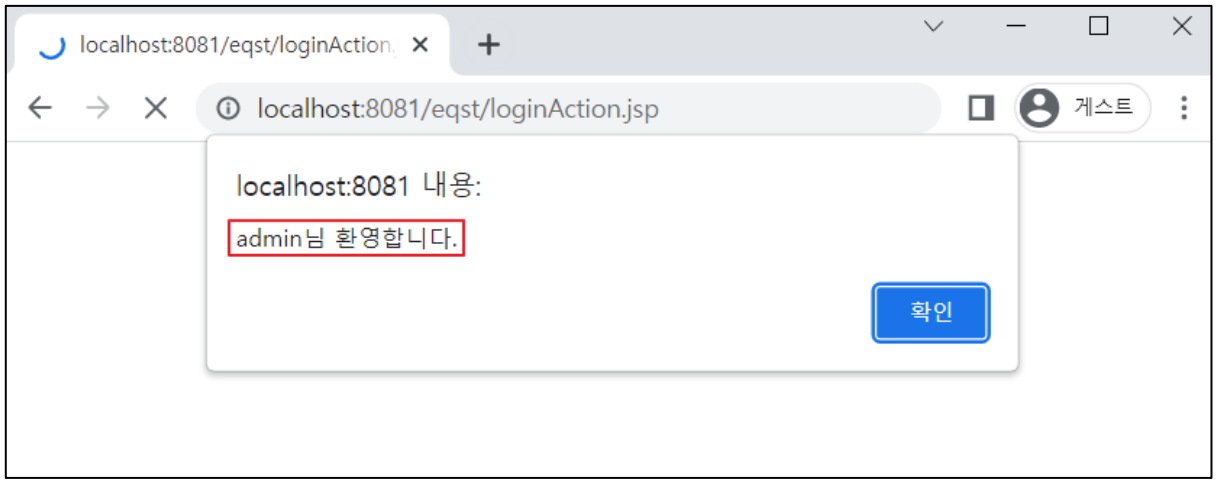
아래는 공격자가 관리자의 아이디인 admin의 존재를 알고 비밀번호는 모를 때 SQL Injection을 통해 로그인하는 과정이다. 아이디 입력 값으로 [admin' --]을 입력하고 비밀번호는 임의의 값인 [1111]을 입력하는데, 이때 아이디 입력 값의 특수문자가 SQL 문법으로 작용하여 홑따옴표(') 이후의 주석인 --으로 인해 비밀번호를 검증하는 부분이 주석 처리된다. 따라서 관리자 계정을 알고 있는 공격자는 임의의 비밀번호를 입력해도 admin 계정으로 로그인이 가능해진다.



[SQL Injection 을 이용한 공격자의 로그인 요청]

실습을 위해 구축한 웹 서버에서 위의 과정을 실행해 보면 임의의 비밀번호를 입력해도 admin 계정으로 정상 로그인 되는 것을 확인할 수 있다.

[로그인 인증 우회]



[관리자 계정으로 로그인 성공]

이렇듯 공격자는 SQL Injection을 통해 로그인을 우회할 수 있으며, 데이터베이스 조작 및 유출, 시스템 명령어 실행 등 데이터베이스에 직접적인 영향을 주는 공격을 할 수 있다.

Step 2. 공격 유형에 따른 분류

SQL Injection은 공격 유형에 따라 크게 3가지로 나뉜다. 이번 달 스페셜 리포트에서는 3가지 유형인 Union SQL Injection, Error Based SQL Injection, Blind SQL Injection에 대한 간략한 소개가 진행되며 유형별 상세 설명과 공격 과정은 다음 달부터 차례대로 연재된다.

case 1. Union SQL Injection

UNION 연산자는 두 개 이상의 SELECT문에 대한 결과를 하나로 묶어 데이터베이스에서 추출하는 특징이 있다. 공격자는 이러한 점을 이용하여 기존의 SELECT문에 원하는 데이터를 조회하기 위한 UNION SELECT문을 추가하여 데이터베이스를 조회한다.

UNION절을 사용하기 위해서는 두 가지 조건에 만족해야 한다.

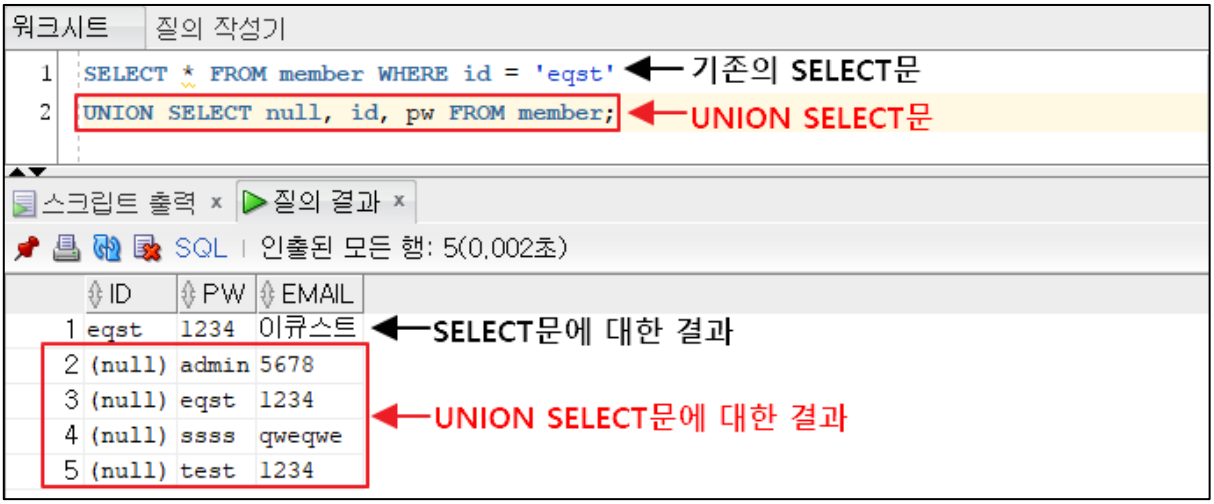
1) SELECT문과 UNION SELECT문의 컬럼 수가 동일해야 한다.

2) 컬럼은 각 순서별로 동일한 데이터형이어야 한다.

WHERE 조건절에 ORDER BY절[2]을 이용하여 컬럼 수를 추출한 후, 데이터형을 알기 위해 컬럼의 개수만큼 null 문자를 입력해서 해당 컬럼의 데이터형이 숫자인지 문자인지 판단하는 과정을 거쳐야 한다. 그 후 전체 테이블 목록에서 원하는 테이블을 선택하고 컬럼의 데이터를 추출하는 과정을 거친다.

[2] ORDER BY절은 데이터 정렬 시 사용하는 구문으로 SELECT문의 컬럼 개수보다 많은 숫자를 조회할 경우 에러를 유발하기 때문에 이를 통해 컬럼의 개수를 확인할 수 있다.

아래는 기존의 SELECT문에 UNION SELECT절을 추가하여 MEMBER 테이블의 ID와 PW 컬럼의 데이터를 조회한 결과이다. 기존의 SELECT문에 대한 질의 결과인 ID가 'eqst'에 해당하는 결과만이 조회되어야 하는데, UNION SELECT로 인해 MEMBER 테이블에 존재하는 모든 ID와 PW가 추가적으로 추출된 것을 확인할 수 있다.

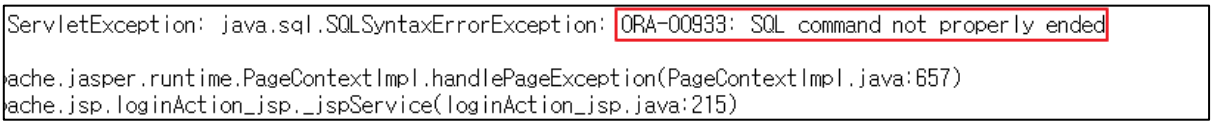


[UNION SELECT 로 조회한 결과]

case 2. Error Based SQL Injection

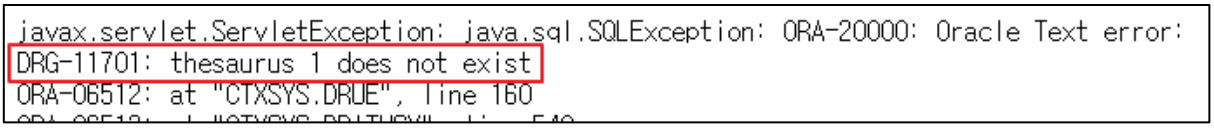
데이터베이스의 문법에 맞지 않은 쿼리문 입력 시 반환되는 에러 정보를 기반으로 공격하는 방법이다. 본 리포트의 실습 환경에 구성된 ORACLE 데이터베이스는 2가지 종류의 에러 정보를 반환한다.

1) 단순 에러 : 단순히 문법이 잘못되었다는 에러 정보



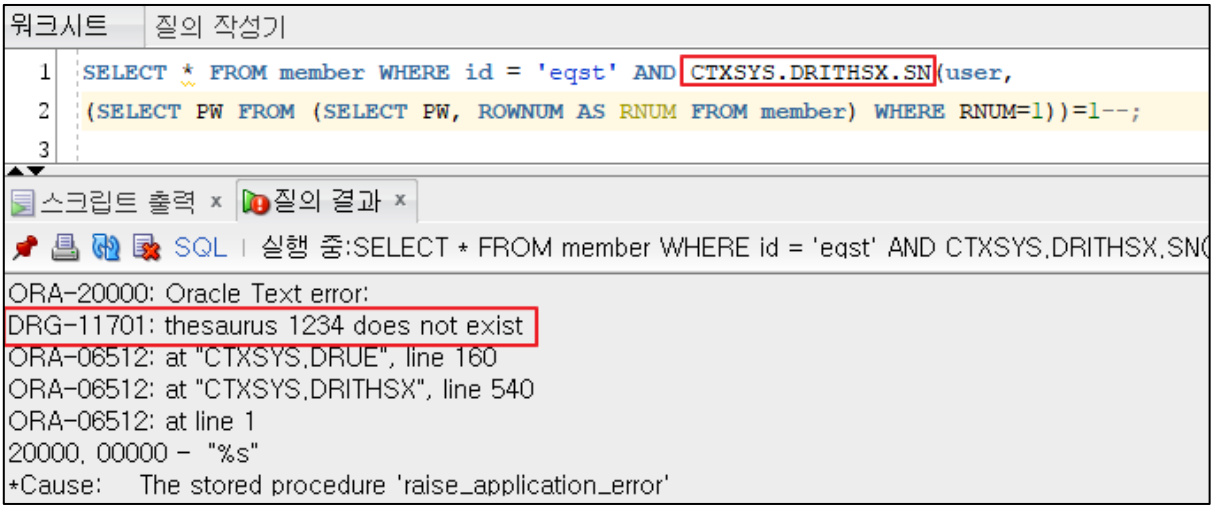
[단순 에러]

2) 정보 획득이 가능한 에러 : 무엇이 잘못되었는지에 대한 에러 정보



[정보 획득이 가능한 에러]

위와 같이 정보 획득이 가능한 에러 메시지를 공격에 사용하는 것이 Error Based SQL Injection 이다. 아래는 에러 유발함수 중 하나인 CTXSYS.DRITHSX.SN을 이용해 MEMBER 테이블의 첫 번째 컬럼의 패스워드를 조회한 결과 발생한 에러 메시지이다. MEMBER 테이블에 저장된 비밀번호 값인 '1234'가 에러 메시지를 통해 나타난 것을 확인할 수 있다.



[에러 메시지를 통해 획득한 비밀번호]

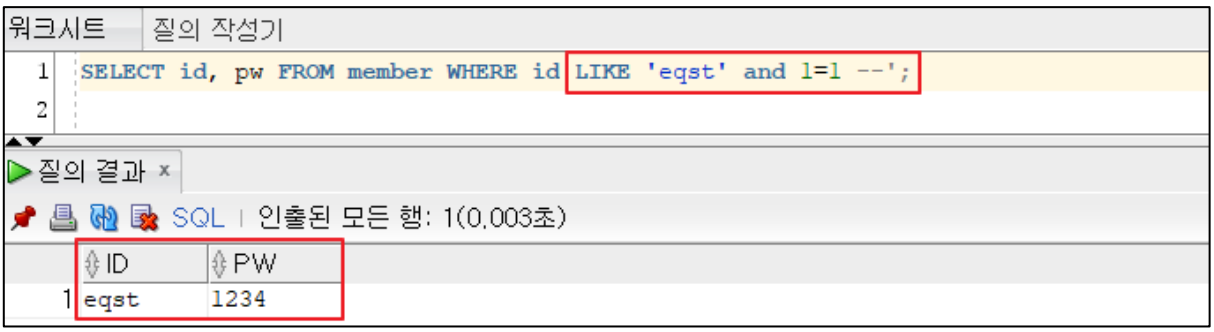
정보 획득이 가능한 에러 메시지를 유발하는 함수는 다음과 같다.

infosec

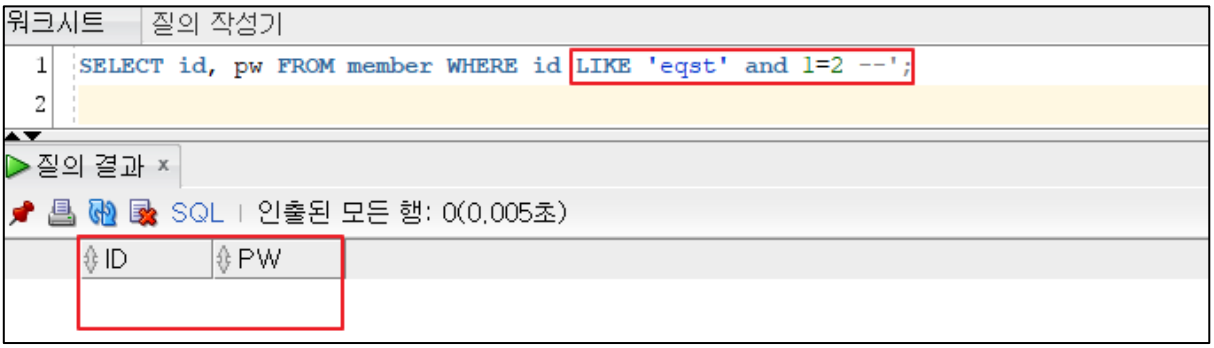
정보 획득이 가능한 에러를 유발하는 함수
UTL_INADDR.GET_HOST_NAME((원하는 서브쿼리 내용))
UTL_INADDR.GET_HOST_ADDRESS((원하는 서브쿼리 내용))
ORDSYS.ORD_DICOM.GETMAPPINGXPATH((원하는 서브쿼리 내용),user,user)
CTXSYS.DRITHSX.SN(user,(원하는 서브쿼리 내용))

case 3. Blind SQL Injection

참(True)인 쿼리문과 거짓(False)인 쿼리문 삽입 시 반환되는 데이터를 비교하여 데이터를 추출하는 공격이다. Blind SQL Injection 취약점이 있는지 확인하기 위해 쿼리가 참인 구문과 거짓인 구문의 반환 결과를 확인한다. 아래는 쿼리 결과가 참인 경우 값이 반환되는 것을 보여주며, 쿼리 결과가 거짓이므로 값이 반환되지 않는 것을 확인할 수 있다.



[쿼리 결과가 참인 경우]



[쿼리 결과가 거짓인 경우]

위와 같이 Blind SQL Injection은 조건문의 실행 결과에 원하는 값을 넣어 반환되는 쿼리의 결과에 따라 정보를 취득해 나간다. 또한 Blind SQL Injection의 가장 큰 특징은 문자열의 문자를 하나하나 추출하는 과정이 필요하다는 것이다. 1개의 문자만 확인할 수 있기 때문에 문자열을 자르는 함수를 이용해 원하는 데이터의 문자를 1개씩 확인하며 데이터를 추출한다. 문자열을 자르기 위해 사용되는 함수는 다음과 같다.

infosec

데이터베이스	함수
Oracle	SUBSTR()
MS-SQL	SUBSTRING()
MySQL	

위의 함수들을 이용해 반복적인 비교를 거쳐 데이터베이스의 전체 테이블의 개수와 이름, 컬럼의 개수와 이름, 실제 데이터를 하나씩 추출해가는 공격 기법이다. 과정이 반복되는 만큼 자동화된 스크립트를 사용하는 것이 일반적이다.

■ 관련 도구

SQL Injection 과 관련된 도구들이 몇 가지 있다. 그 중에서도 웹에서 간단한 SQL 쿼리를 테스트하는 사이트인 SQL Fiddle 과 다양한 공격 페이로드를 제공해 주는 Cheat Sheet 인 Pentest Monkey, 마지막으로 SQL Injection 취약점을 탐지하고 진단하는데 쓰이는 자동화 공격 도구인 sqlmap 에 대해 소개하겠다.

1. SQL Fiddle

· URL : http://sqlfiddle.com/

웹 브라우저에서 SQL 쿼리를 테스트할 수 있는 사이트이며 Oracle, MS-SQL, MySQL 등 다양한 데이터베이스와 버전을 지원하고 있다. 사용자가 직접 테이블을 생성하여 값을 추가할 수 있으며, 자체적으로 샘플 데이터를 제공하기 때문에 손쉽게 테스트해 볼 수 있다.

2. Cheat Sheet – Pentest Monkey

· URL : https://pentestmonkey.net/category/cheat-sheet/sql-injection

대표적인 Cheat Sheet 사이트인 Pentest Monkey 에서는 데이터베이스별 Cheat Sheet 를 제공하고 있다. 다양한 공격 페이로드를 제공해 주기 때문에 참고하기에 좋으나 의도하지 않은 결과를 낳을 수 있으므로 공격 페이로드가 어떠한 행위와 영향을 미치는지 이해하고 사용하는 것이 중요하다.

3. sqlmap

· URL : https://sqlmap.org/

SQL Injection 취약점을 탐지/진단하고 자동화하여 공격할 수 있는 오픈소스 침투 테스트 도구이다. 데이터베이스 구조 파악과 데이터 추출 등을 자동화해주기 때문에 시간을 절약할 수 있다는 장점이 있지만 과도한 네트워크 트래픽 유발로 인해 서버에 영향을 미칠 수 있으므로 실무에서는 잘 쓰이지 않으며, 개인용 테스트 서버에서의 사용을 권장한다.

■ 맺음말

SQL Injection의 개요를 살펴보았다. SQL Injection 취약점이 있을 경우, 공격자가 직접적으로 데이터베이스를 공격하여 중요 정보를 조회하고 탈취할 수 있으므로 개발자는 취약점이 발생하지 않도록 개발해야 하고, 진단자는 취약점 진단 시 누락 없이 찾는 것이 중요하다.

이어지는 5월 호에서는 SQL Injection의 3가지 공격 유형 중 SELECT문의 결합으로 정보를 추출하는 'Union SQL Injection'의 개념과 취약한 소스코드 사용 시 발생할 수 있는 공격에 대해 자세히 알아보도록 하겠다.