**SQL Injection**

|  |  |
| --- | --- |
| 과 목 | 시스템보안 |
| 담 당 교 수 | 송유진 |
| 조 원 | 김진우, 김태환 |
| 제출일자 | 2018년 10월 31일 |

1. 명령 삽입 취약점(Injection)?

* 전송 받는 인수에 포함된, 특정명령을 실행하는 코드를 적절히 필터링하지 못하면 삽입공격에 대한 취약점이 발생.
* 클라이언트의 요청에서 신뢰 할 수 없는 데이터가 명령어나 질의문의 일부분으로 인터프리터로 보내질 때 발생한다.

1. SQL Injection?

* SQL언어에 이루어지는 Injection 공격이다.
* 쿼리의 참과 거짓일 때 서버 반응 만으로 데이터를 얻어내는 기술을 Blind SQL Injection이라고 한다.
* 공격의 종류는 크게 인증 우회, 데이터 노출, 원격 명령 실행 3가지 유형이 있다.

1. SQL Injection 동작 원리
   1. 동적 구문 생성

* 쿼리문에 들어가는 파라미터 값이 계속 변동된다.
  1. 부적절한 특수 문자 처리
* 따옴표(‘), 더블파이프(||), 마침표(.), 주석처리(\*/), 쌍 따옴표(“) 등의 특수문자들이 쿼리문으로 인식될 수 있다.
  1. 잘못된 유형 처리
* LOAD\_FILE() 같은 함수를 Injection에 사용한다면 특수문자 없이 공격을 할 수 있다.
  1. 부적절한 에러 처리
* 입력 값에 대한 적절한 에러 처리를 하지 않아서, 에러 메시지에 서버의 정보가 노출될 수 있다.
  1. 부적절한 중복 요청 처리
* ???

개발자는 으레 사용자가 자신이 개발해 놓은 논리 과정을 따라 오길 원한다.

정해진 데이터 흐름, 예를 들어 1번 폼을 완료하고 2번 폼, 3번 폼을 완료할 거라고 예상하고

개발자는 1번, 2번에서만 입력값 검증 코드를 작성한다. 근데 1,2번에 올바른 폼을 작성하고

3번에 입력값을 조작한다면 공격이 일어날 것이다.

1. 공격의 종류
   1. 인증우회(Auth Bypass)

* SQL 쿼리문의 TRUE/FALSE의 논리적 연산 오류를 이용하여 로그인 인증 쿼리문이 무조건 TRUE가 나오게 만들어 인증을 무력화하는 방법이다.
  1. 데이터 노출(Data Disclosure)
* 웹사이트의 Database 에러정보를 파악 유무에 따라 공격방식이 결정된다

1. Error based 방식
   * + Database의 에러메세지를 기반으로 분석한다.
     + SQL 구문 중 **group by, having** 구문을 사용한다.
2. Union based 방식
   * + **order by** 구문을 사용하여 컬럼 개수를 알아낸다.
     + **Union** 연산자를 사용해 쿼리를 합쳐 정상페이지를 출력한다.
     + **Database()** 사용하여 데이터베이스 명을 알아낸다.
3. Time based 방식
   * + 에러가 발생하지 않는 사이트에 사용된다.
     + SQL 구문 중 waitfor delay 명령어를 사용하여 참,거짓을 판별한다.
     + 상당한 시간을 요한다.

※ group by : 데이터들을 원하는 그룹으로 나누는 구문

※ having : 집계 함수를 가지고 조건비교를 할 때 사용 한다.

※ Union : 집합 연산자 중 하나로 두 쿼리문의 결과를 합치는 연산자다.

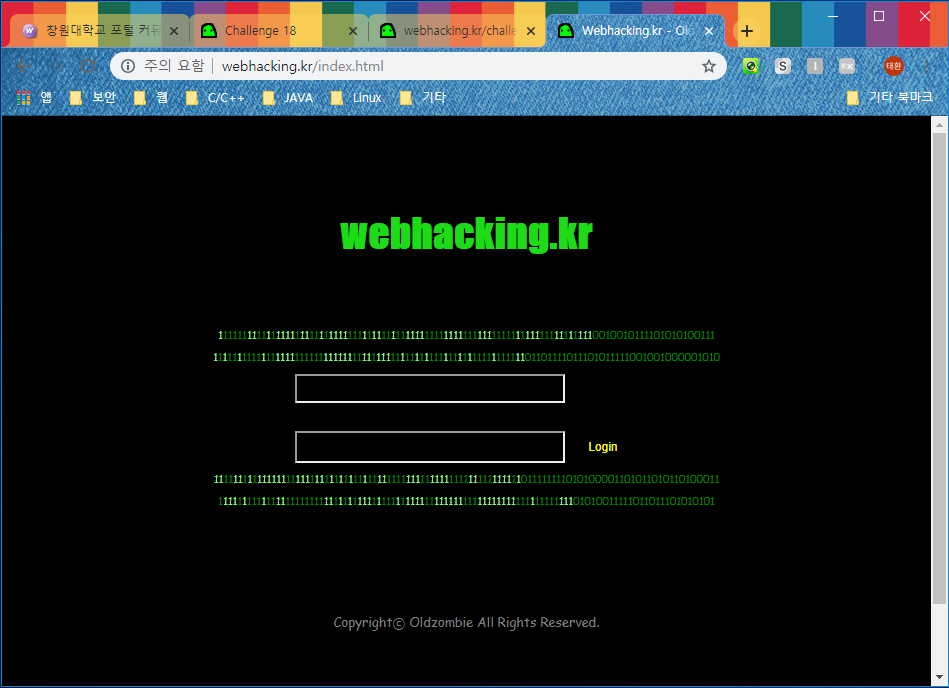
※ order by **i** : field명 i를 기준으로 정렬하는 구문으로 field 대신 숫자를 사용하면,

i 번째 필드 값을 기준으로 정렬한다.

* 1. 원격명령 실행(Remote Command Excute)
* MSSQL에서 주로 사용되며 저장 프로시저를 이용하여 공격한다.



1. SQL 실습
   1. 실습용 서버 : webhacking.kr



* 1. 실습용 Tool
* Google Chrom 관리자 모드

1. SQL Injection 방어법
   1. 값을 받을 때 특수문자 여부를 검사하여 방어한다.
   2. SQL 서버 오류 발생시 해당하는 에러 메시지를 표시하지 않는다.
   3. statement 대신 특수문자를 막아주는 preparestatement 사용한다.
   4. 패스워드 같은 입력 값을 hash function을 적용 후 저장한다.

8. 출처

* <https://namu.wiki/w/SQL%20injection>
* <https://blog.naver.com/shackerz/220423861377>
* <http://mrrootable.tistory.com/25>
* <http://asfirstalways.tistory.com/360>
* <https://sir.kr/pg_tip/14486>