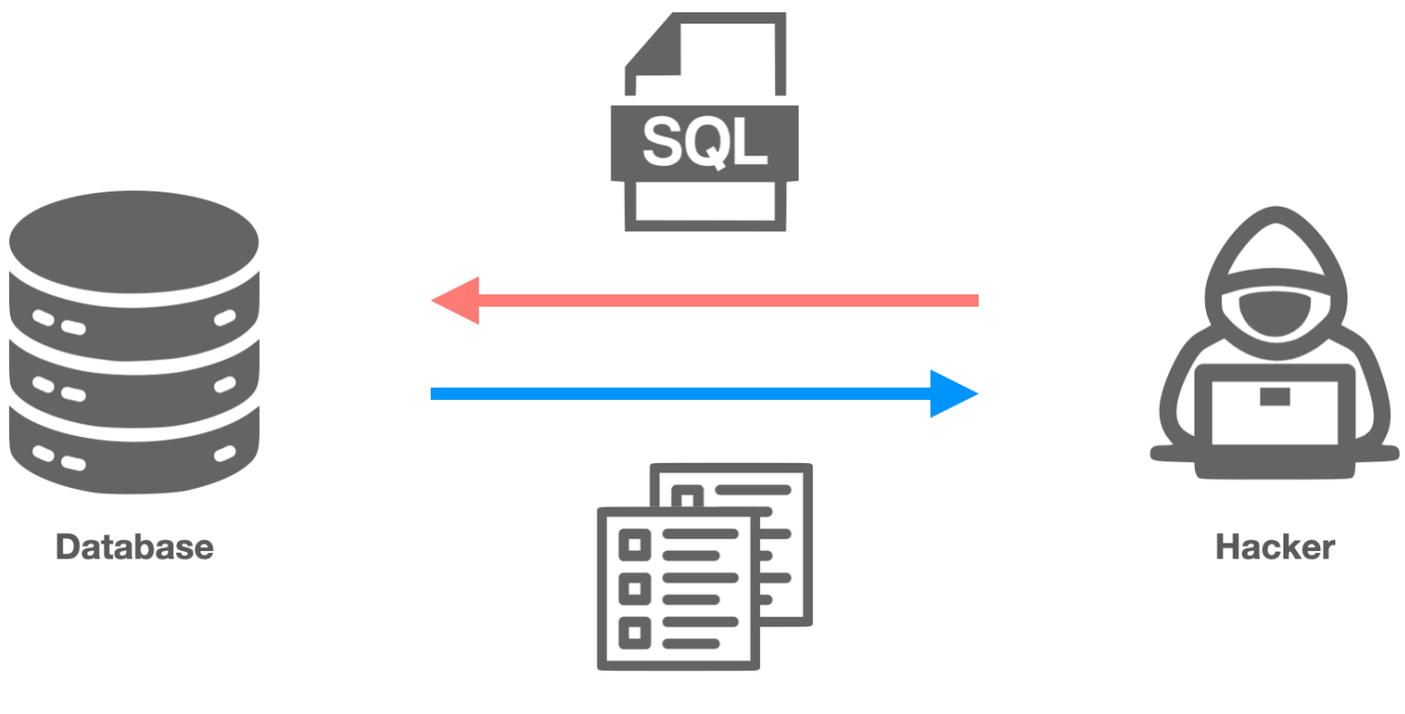
CS Study 16주차

SQL Injection

김신아

SQL Injection



악의적인 사용자가 보안상의 취약점을 이용하여, 임의의 SQL 문을 주입하고 실행되게 하여 데이터베이스가 비정상적인 동작을 하도록 조작하는 행위이다.
인젝션 공격은 OWASP Top 10에 항상 속해 있으며, 공격이 비교적 쉬운 편이고 공격에 성공할 경우 큰 피해를 입힐 수 있는 공격이다.

OWASP Top 10 - 2021

A01 : Broken Access Control (접근 권한 취약점)
엑세스 제어는 사용자가 권한을 벗어나 행동할 수 없도록 정책을 시행합니다. 만약 엑세스 제어가 취약하면 사용자는 주어진 권한을 벗어나 모든 데이터를 무단으로 열람, 수정 혹은 삭제 등의 행위로 이어질 수 있습니다.

A02 : Cryptographic Failures (암호화 오류)
Sensitive Data Exposure(민감 데이터 노출)의 명칭이 2021년 Cryptographic Failures(암호화 오류)로 변경되었습니다. 적절한 암호화가 이루어지지 않으면 민감 데이터가 노출될 수 있습니다.

A03: Injection (인젝션)
SQL, NoSQL, OS 명령, ORM(Object Relational Mapping), LDAP, EL(Expression Language) 또는 OGNL(Object Graph Navigation Library) 인젝션 취약점은 신뢰할 수 없는 데이터가 명령어나 쿼리문의 일부분으로써, 인터프리터로 보내질 때 취약점이 발생합니다.

A04: Insecure Design (안전하지 않은 설계)
Insecure Design(안전하지 않은 설계)는 누락되거나 비효율적인 제어 설계로 표현되는 다양한 취약점을 나타내는 카테고리 입니다. 안전하지 않은 설계와 안전하지 않은 구현에는 차이가 있지만, 안전하지 않은 설계에서 취약점으로 이어지는 구현 결함이 있을 수 있습니다.

A05: Security Misconfiguration (보안설정오류)
애플리케이션 스택의 적절한 보안 강화가 누락되었거나 클라우드 서비스에 대한 권한이 적절하지 않게 구성되었을 때, 불필요한 기능이 활성화 되거나 설치되었을 때, 기본계정 및 암호화가 변경되지 않았을 때, 지나치게 상세한 오류 메세지를 노출할 때, 최신 보안기능이 비활성화 되거나 안전하지 않게 구성되었을 때 발생합니다.

A06: Vulnerable and Outdated Components (취약하고 오래된 요소)
취약하고 오래된 요소는 지원이 종료되었거나 오래된 버전을 사용할 때 발생합니다. 이는 애플리케이션 뿐만 아니라, DBMS, API 및 모든 구성요소 들이 포함됩니다.

A07: Identification and Authentication Failures (식별 및 인증 오류)
Broken Authentication(취약한 인증)으로 알려졌던 해당 취약점은 identification failures(식별 실패)까지 포함하여 더 넓은 범위를 포함할 수 있도록 변경되었습니다. 사용자의 신원확인, 인증 및 세션관리가 적절히 되지 않을 때 취약점이 발생할 수 있습니다.

A08: Software and Data Integrity Failures(소프트웨어 및 데이터 무결성 오류)
2021년 새로 등장한 카테고리로 무결성을 확인하지 않고 소프트웨어 업데이트, 중요 데이터 및 CI/CD 파이프라인과 관련된 가정을 하는데 중점을 둡니다.

A09: Security Logging and Monitoring Failures (보안 로깅 및 모니터링 실패)
Insufficient Logging & Monitoring(불충분한 로깅 및 모니터링) 명칭이었던 카테고리가 Security Logging and Monitoring Failures (보안 로깅 및 모니터링 실패)로 변경되었습니다. 로깅 및 모니터링 없이는 공격활동을 인지할 수 없습니다. 이 카테고리는 진행중인 공격을 감지 및 대응하는데 도움이 됩니다.

A10: Server-Side Request Forgery (서버 측 요청 위조)
2021년 새롭게 등장하였습니다. SSRF 결함은 웹 애플리케이션이 사용자가 제공한 URL의 유효성을 검사하지 않고 원격 리소스를 가져올 때마다 발생합니다. 이를 통해 공격자는 방화벽, VPN 또는 다른 유형의 네트워크 ACL(액세스 제어 목록)에 의해 보호되는 경우에도 응용 프로그램이 조작된 요청을 예기치 않은 대상으로 보내도록 강제할 수 있습니다.

Open Web Application Security Project에 따라 악용가능성, 탐지가능성 및 영향에 대해 빈도수가 높고 보안상 영향을 크게 줄 수 있는 10가지 웹 애플리케이션 보안 취약점 목록이다.

SQL Injection 피해 사례

‘여기어때’ 개인정보 99만건 유출…‘SQL인젝션’ 공격이 원인

발행일 2017-04-26 10:40:16

위드이노베이션이 운영하는 숙박 온·오프라인 연계(O2O) 서비스 ‘여기어때’에서 유출된 고객 개인정보가 99만여 건에 이르는 것으로 조사됐다. 지난달 회사측이 발표한 침해 건수에서 더 늘어났다.

사고 원인은 조사 초창기부터 예상됐던대로 웹사이트 취약점 공격기법인 ‘SQL 인젝션’ 공격에 의해 데이터베이스(DB)가 뚫린 것으로 확인됐다.

미래창조과학부와 방송통신위원회는 여기어때 개인정보 유출 침해사고 관련 ‘민·관합동조사단’ 조사 결과를 4월 26일 밝혔다.

민·관합동조사단은 여기어때 서비스 이용고객을 대상으로 총 4817건의 협박성 음란 문자(SMS)가 발송됨에 따라 확인된 개인정보 유출 침해사고 원인 분석과 대응, 피해 방지 등을 위해 미래부, 방통위, 한국인터넷진흥원과 민간 전문가로 구성, 지난 3월7일부터 3월17일까지 조사를 벌였다.

조사단은 확보한 웹서버 로그 1560만건, 공격서버·PC 5대를 바탕으로 사고 관련자료 분석, 재연해 해킹의 구체적인 방법 및 절차, 개인정보 유출 규모 등을 확인했다.

해커는 여기어때 마케팅센터 웹페이지에 SQL 인젝션 공격을 통해 DB에 저장된 관리자 세션값(세션 아이디)을 탈취한 것으로 드러났다. SQL 인젝션 공격은 가장 흔한 웹사이트 취약점 공격으로 알려져 있다.

공격 종류 및 방법

- Error based SQL Injection
- Union based SQL Injection
- Blind based SQL Injection
  - Boolean based SQL
  - Time based SQL
- Stored Procedure SQL Injection
- Mass SQL Injection

공격 종류 및 방법 - Error based SQL Injection

논리적 에러를 이용한 SQL Injection

가장 많이 쓰이고, 대중적인 공격기법이다.

1. SELECT * FROM Users WHERE id = 'INPUT1' AND password = 'INPUT2'

Hacker

3. SELECT * FROM Users WHERE id = '' OR 1=1 - - 'AND password = 'INPUT2'

=> SELECT * FROM Users

공격 종류 및 방법 - Union based SQL Injection

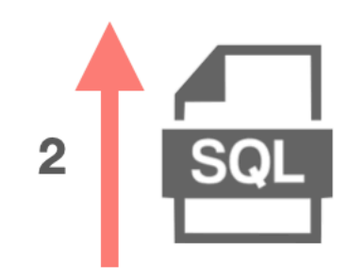
Union 명령어를 이용한 SQL Injection

두 개의 쿼리문에 대한 결과를 통합해서 하나의 테이블로 보여주게 하는 키워드이다.
정상적인 쿼리문에 Union 키워드를 사용하여 인젝션에 성공하면, 원하는 쿼리문을 실행할 수 있게 된다. Union Injection을 성공하기 위해서는 두가지의 조건이 있다.
Union하는 두 테이블의 컬럼 수가 같아야 하고, 데이터 형이 같아야 한다.

1. SELECT * FROM Board WHERE title LIKE '%INPUT%' OR contents '%INPUT%'

Board table

id	title	contents
1	hi	Hello
2	bye	Bye bye



' UNION SELECT null, id, password FROM Users- -

Hacker

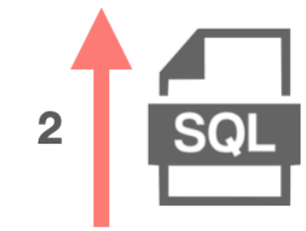
3. SELECT * FROM Board WHERE title LIKE '% ' UNION SELECT null, id, password FROM Users- -
'%' AND contents '% UNION SELECT null, id, password FROM Users- -%'

공격 종류 및 방법 - Blind SQL Injection

Boolean based SQL

데이터베이스로부터 특정한 값이나 데이터를 전달받지 않고, 단순히 참과 거짓의 정보만 알 수 있을 때 사용한다. 로그인 폼에 SQL Injection이 가능하다고 했을 때, 서버가 응답하는 로그인 성공과 로그인 실패 메시지를 이용하여, DB의 테이블 정보 등을 추출해 낼 수 있다.

1. SELECT * FROM Users WHERE id = ‘INPUT1’ AND password = ‘INPUT2’



abc123’ and ASCII(SUBSTR(SELECT name
FROM information_schema.tables
WHERE table_type=‘base table’ limit 0, 1)1,1))
> 100- -
(MYSQL일 경우)

3. SELECT * FROM Users WHERE id = ‘abc123’ and ASCII(SUBSTR(SELECT name
FROM information_schema.tables
WHERE table_type=‘base table’ limit 0, 1)1,1)) > 100- -
‘AND password = ‘INPUT2’

(로그인이 될 때까지 시도)

공격 종류 및 방법 - Blind SQL Injection

Boolean based SQL

데이터베이스의 테이블 명을 알아내는 방법이다.
인젝션이 가능한 로그인 폼을 통하여 악의적인 사용자는 임의로 가입한 abc123 이라는 아이디와 함께 abc123’ and ASCII(SUBSTR(SELECT name From infromation_schema.tables WHERE table_type=‘base table’ limi 0,1)1,1)) > 100- - 이라는 구문을 주입한다.
해당 구문은 MySQL에서 테이블 명을 조회하는 구문으로 limit 키워드를 통해 하나의 테이블만 조회하고, SUBSTR 함수로 첫글자만, 그리고 마지막 ASCII를 통해서 ascii 값으로 변환해준다.
만약 조회되는 테이블 명이 Users라면 ‘U’자가 ascii 값으로 조회가 될 것이고,
뒤의 100이라는 숫자값과 비교를 하게 된다.
거짓이면 로그인 실패가 될 것이고, 참이 될때까지 100이라는 숫자를 변경해 비교를 하면된다.
공격자는 이 프로세스를 자동화 스크립트롤 통하여 단기간 내에 테이블 명을 알아낼 수 있다.

* SUBSTR(“문자열”, “시작위치”, “길이”)

공격 종류 및 방법 - Blind SQL Injection

Time based SQL

서버로부터 특정한 응답 대신 참 혹은 거짓의 응답을 통해서 데이터베이스의 정보를 유추하는 기법이다. 사용되는 함수는 MySQL 기준으로 SLEEP 과 BENCHMARK 이다.

1. SELECT * FROM Users WHERE id = ‘INPUT1’ AND password = ‘INPUT2’

2

abc123’ OR (LENGTH(DATABASE())=1
AND SLEEP(2))- -
(혹은 BENCHIMARK() 함수 사용)
(MYSQL일 경우)

Hacker

(SLEEP 할 때까지 시도)

3. SELECT * FROM Users WHERE id = ‘abc123’ OR (LENGTH(DATABASE())=1
AND SLEEP(2))- -
‘AND password = ‘INPUT2’

공격 종류 및 방법 - Blind SQL Injection

Time based SQL

현재 사용하고 있는 데이터베이스의 길이를 알아내는 방법이다.
로그인 폼에 주입하며 임의로 abc123 이라는 계정을 생성해둔다.
악의적인 사용자가 abc123’ OR (LENGTH(DATABASE())=1 AND SLEEP(2))- -라는 구문을 주입한다. 여기서 LENGTH 함수는 문자열의 길을 반환하고, DATABASE 함수는 데이터베이스의 이름을 반환한다.
LENGTH(DATABASE())=10이 참이면 SLEEP(2)가 동작하고, 거짓이면 동작하지 않는다.
이를 통해 숫자 1 부분을 조작하여 데이터베이스의 길을 알아낼 수 있다.
만약 SLEEP이라는 단어가 치환처리 되어있다면, 또 다른 방법을 BENCHMARK나 WAIT함수를 사용할 수 있다.
BENCHMARK는 BENCHMARK(1000000,AES_ENCRYPT(‘hello’,’goodbye’)); 이와 같은 방법으로 사용 가능하다. 이 구문을 실행하면 약 4.74가 걸린다.

공격 종류 및 방법 - Stored Procedure SQL Injection

저장된 프로시저에서의 SQL Injection

저장 프로시저(Stored Procedure)은 일련의 쿼리들을 모아 하나의 함수처럼 사용하기 위한 것이다. 공격에 사용되는 대표적인 저장 프로시저는 MS-SQL에 있는 xp_cmdshell로 윈도우 명령어를 사용할 수 있게 된다.

단, 공격자가 시스템 권한을 획득해야 하므로 공격난이도가 높으나 공격에 성공한다면, 서버에 직접적인 피해를 입힐 수 있는 공격이다.

공격 종류 및 방법 - Mass SQL Injection

다량의 SQL Injection 공격

2008년 처음 발견된 공격기법으로 기존 SQL Injection과 달리 한번의 공격으로
다량의 데이터베이스가 조작되어 큰 피해를 입히는 것을 의미한다.
보통 MS-SQL을 사용하는 ASP 기반 웹 애플리케이션에서 많이 사용되며,
쿼리문은 HEX 인코딩 방식으로 인코딩하여 공격한다.
보통 데이터베이스 값을 변조하여 데이터베이스 악성 스크립트를 삽입하고,
사용자들이 변조된 사이트에 접속 시 좀비 PC로 감염되게 한다.
이렇게 감염된 좀비 PC들은 DDoS 공격에 사용된다.

대응방안

- 입력 값에 대한 검증
- Prepared Statement 구문 사용
- Error Message 노출 금지
- 웹 방화벽 사용

대응방안

입력 값에 대한 검증

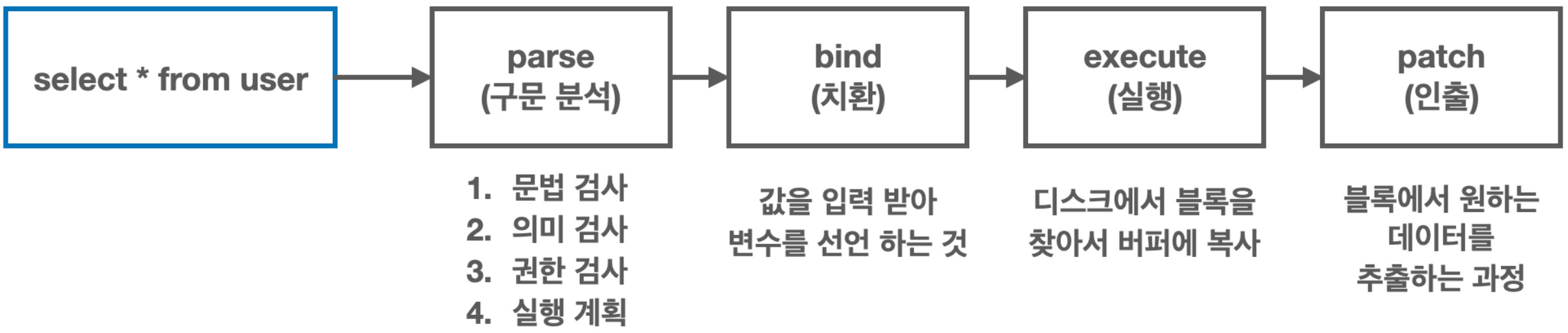
SQL Inejction에서 사용되는 기법과 키워드는 엄청나게 많다.
사용자의 입력 값에 대한 검증이 필요한데, 서버 단에서 화이트리스트 기반으로 검증해야한다.
블랙리스트 기반으로 검증하게 되면 수많은 차단리스트를 등록해야하고,
하나라도 빠지면 공격에 성공하게 되기 때문이다.
공백으로 치환하는 방법도 많이 쓰이는데, 이 방법도 취약한 방법이다.
예를 들어 공격자가 SESELECTLECT라고 입력 시 중간의 SELECT가 공백으로 치환 되면
SELECT라는 키워드가 완성되게 된다.
공백 대신 공격 키워드와는 의미 없는 단어로 치환되어야 한다.

MYSQL에서 쿼리 실행 과정

1. 구문 오류 체크(parse)
2. 공유 영역에서 해당 구문 검색(parse)
3. 권한 체크(parse)
4. 실행 계획 수립(parse)
5. 실행 계획 공유 영역에 저장(parse)
6. 쿼리 실행(execute)
7. 데이터 인출(patch)

1~7 모두 실행 : Hard Parsing

1,2,6,7,만 실행 : Soft Parsing



대응방안

Prepared Statement 구문 사용

- Prepared Statement

```
String sql = "SELECT NAME, AGE FROM TABLE WHERE userID = ?"
PreparedStatement pstmt = conn.preparedStatement(sql);
pstmt.setInt(1, userID);
ResultSet rst = pstmt.executeQuery();
```

parse 부분을 컴파일한채로 캐시에 저장하고 데이터가 bind될때까지 기다린다. 이후 patch 부분까지 실행이 완료되고 sql을 재사용할때 parse를 다시 실행하지 않고 캐시에 저장되어 있는 부분을 가져다 나머지 3단계만 실행한다. sql이 반복적일때 효율적이다.
바인딩 된 데이터는 SQL문법이 아닌 내부의 인터프리터나 컴파일 언어로 처리하므로, 문법적인 의미를 가질 수 없다. 이것이 의미하는 것은 parse 부분에서 이미 쿼리의 문법적인 처리부분이 선수행되었으므로 바인딩 관계에서 입력된 부분은 쿼리로 인식하지 않는다는 의미이다.
따라서 sql injection에 좋다.

대응방안

Prepared Statement 구문 사용

- Statement

```
String sql = "SELECT NAME, AGE FROM TABLE WHERE userID = " + userID
Statement stmt = conn.createdStatement();
ResultSet result = sstmt.executeQuery();
```

4가지 단계를 매번 실행하여 결과 값을 반환한다. 만약 반복적으로 쿼리를 수행할 경우 PreparedStatement가 DB에 적은 부하를 주며, 성능이 더 좋다.
하지만 Statement는 sql 전체를 한눈에 볼 수 있다는 장점이 있다.

대응방안

Prepared Statement 구문 사용

Prepared Statement 구문을 사용하게 되면, 사용자의 입력 값이 데이터베이스의 파라미터로 들어가기 전에 DBMS가 미리 컴파일하여 실행하지 않고 대기한다.
그 후 사용자의 입력 값을 문자열로 인식하게 하여 공격 쿼리가 들어간다고 하더라도,
사용자의 입력은 이미 의미 없는 단순 문자열이기 때문에 전체 쿼리문도 공격자의 의도대로 작동하지 않는다.

대응방안

Error Message 노출 금지

공격자가 SQL Injection을 수행하기 위해서는 데이터베이스의 정보(데이블명, 컬럼명 등)가 필요하다. 데이터베이스 에러 발생 시 따로 처리를 해지 않았다면, 에러가 발생한 쿼리문과 함께 에러에 관한 내용을 반환해준다.
여기서 테이블명 및 컬럼명 그리고 쿼리문이 노출될 수 있기 때문에,
데이터베이스에 대한 오류발생 시 사용자에게 보여줄 수 있는 페이지를 제작 혹은 메시지박스를 띄우도록 하여야 한다.

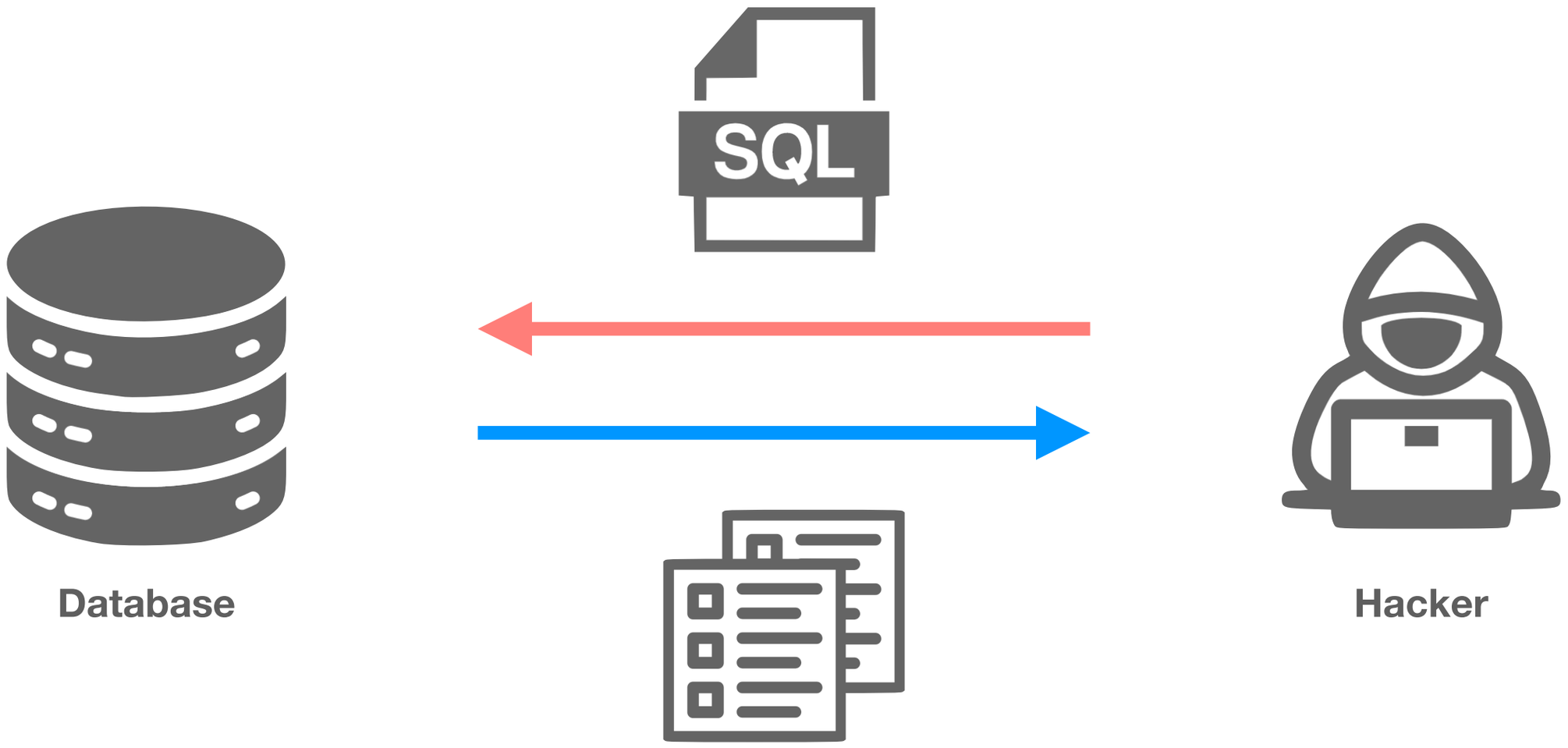
대응방안

웹 방화벽 사용

웹 공격 방어에 특화되어 있는 웹 방화벽을 사용하는 것도 하나의 방법이다.
웹 방화벽은 소프트웨어형, 하드웨어형, 프록시형 이렇게 세가지 종류로 나눌 수 있는데
소프트웨어형은 서버 내에 직접 설치하는 방법이다.
하드웨어형은 네트워크 상에서 서버 앞 단에 직접 하드웨어 장비로 구성하는 것이다.
프록시형은 DNS 서버 주소를 웹 방화벽으로 바꾸고 서버로 가는 트래픽이 웹 방화벽을 먼저 거치도록 하는 방법이다.

Thank You

별첨



별첨

1. SELECT * FROM Users WHERE id = ‘INPUT1’ AND password = ‘INPUT2’

Hacker

3. SELECT * FROM Users WHERE id = ‘’ OR 1=1 - - ‘AND password = ‘INPUT2’

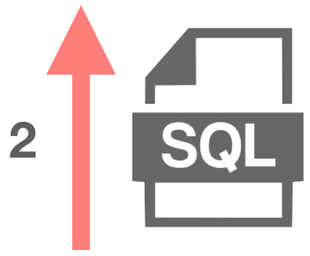
=> SELECT * FROM Users

별첨

1. SELECT * FROM Board WHERE title LIKE ‘%INPUT%’ OR contents ‘%INPUT%’

Board table

id	title	contents
1	hi	Hello
2	bye	Bye bye



‘ UNION SELECT null, id, password FROM Users- -

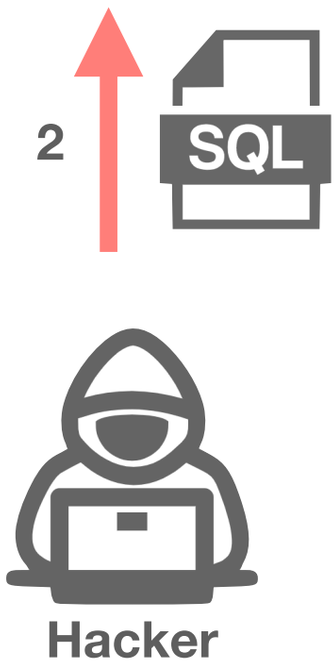
Hacker

3. SELECT * FROM Board WHERE title LIKE ‘% ’ UNION SELECT null, id, password FROM Users- -

‘%’ AND contents ’% UNION SELECT null, id, password FROM Users- -%’

별첨

1. SELECT * FROM Users WHERE id = ‘INPUT1’ AND password = ‘INPUT2’



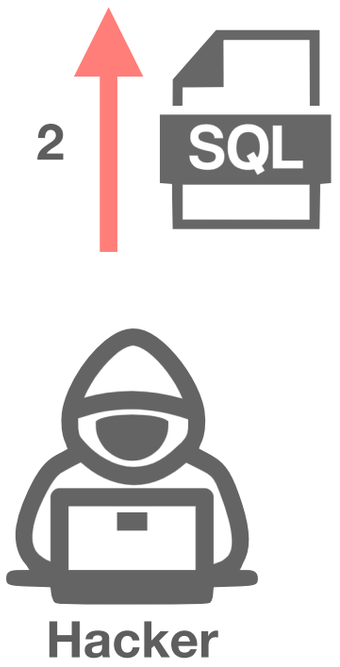
abc123’ and ASCII(SUBSTR(SELECT name
FROM information_schema.tables
WHERE table_type=‘base table’ limit 0, 1)1,1))
> 100- -
(MYSQL일 경우)

3. SELECT * FROM Users WHERE id = ‘abc123’ and ASCII(SUBSTR(SELECT name
FROM information_schema.tables
WHERE table_type=‘base table’ limit 0, 1)1,1)) > 100- -
‘AND password = ‘INPUT2’

(로그인이 될 때까지 시도)

별첨

1. SELECT * FROM Users WHERE id = ‘INPUT1’ AND password = ‘INPUT2’



abc123’ OR (LENGTH(DATABASE())=1

AND SLEEP(2))- -

(혹은 BENCHIMARK() 함수 사용)

(MYSQL일 경우)

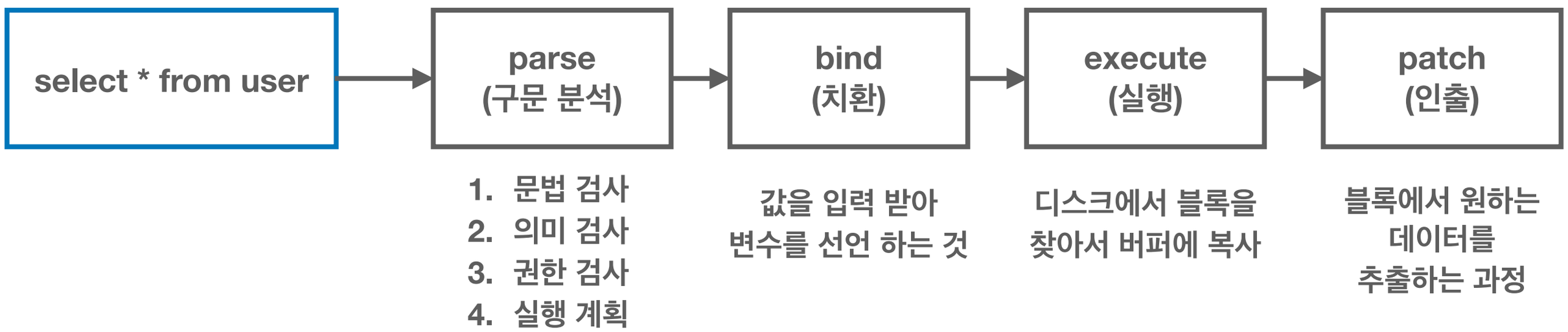
(SLEEP 할 때까지 시도)

3. SELECT * FROM Users WHERE id = ‘abc123’ OR (LENGTH(DATABASE())=1

AND SLEEP(2))- -

‘AND password = ‘INPUT2’

별첨


select * from user
parse
(구문 분석)
1. 문법 검사
2. 의미 검사
3. 권한 검사
4. 실행 계획
bind
(치환)
값을 입력 받아
변수를 선언 하는 것
execute
(실행)
디스크에서 블록을
찾아서 버퍼에 복사
patch
(인출)
블록에서 원하는
데이터를
추출하는 과정

별첨

https://noirstar.tistory.com/264

https://blog.alyac.co.kr/4135

https://www.bloter.net/newsView/blt201704260005