

**KEY POINTS**

**The lectures last week looked at database vulnerabilities. This tutorial asks you to use SQL injection against a poorly designed database. Later we will look at ways of defining the database against SQL injection. SQL Injection is the introduction of malicious SQL code into an application. For this tutorial, you are in the role of malicious user.**

FOR THE ASSIGNMENT

This tutorial demonstrates what we mean by a proof of concept artefact. It has just enough functionality to demonstrate the ease with which SQL Injection can be carried out. There is no real interface, very limited test data, no features that are not relevant to the point being made and because the assumption is that the intruder has already bypassed password security, there is no password protection.

A ‘proof of concept’ artefact has the functionality required to prove the concept that is being demonstrated – and does not have additional functionality.

**Getting Started**

Scenario:

Beaconside Traders uses a (very primitive) web front end to manage its warehouse operations. For each of the following tasks:

* identify how you could use the webform to maliciously damage the database
* Identify the vulnerabilities which let you achieve this
* identify what could be done to stop this (this will be covered in future tutorials)

**Go to:** <http://asp.soc.staffs.ac.uk/injection/> and you will see 2 files:

delete MDS.asp and MDS.asp.

**The Task**

* You will all be using the same underlying Oracle table – log any concurrency issues you come across
* The warehouse supervisor is entitled to insert and delete bins from the database and to insert new bins. You are not the supervisor but have managed to get to the forms.
  + See what information you can get about the structure of the database (Hint: provoke error messages)
  + Are you able to delete bins from the system? And how will you know if you have been successful?
  + The InsertMDS.asp file allows you (obviously) to insert. Can you do any damage to the database with this?
  + What other vulnerabilities are there and can they be prevented?

**Some Tips**

* **Where a Black Hat Hacker would start**

Assuming you do not know anything about the structure of the database, the first thing a malicious user would do would be to find out the structure of the database and test the validation. Error messages are a useful source of information.

* How can you provoke an error message?
* What can you deduce about the data in the database and the validation from the error messages you provoke?
* What information would you be able to obtain if you could access the catalog?
* **SQL INJECTION**
* The major source of vulnerability is any area where you can use SQL variables or write SQL (dynamic SQL). The deleteMDS file is the most vulnerable.
* Some sites let you write your own SQL. These sites are few and far between and not likely to be used commercially
* A web form should in theory restrict access to data. You can also use the layers of abstraction concept - for example data adaptors in SQL Server. The site you are dealing with here has some validation (assume you have already dealt with a password barrier) because you are using a form, not keying in direct.
* There are a number of ways in which SQL injection can take place but two of the most commonly used are:
* (a) The so-called ‘single quote’ trick - typically used with password fields; look at <http://www.securiteam.com/securityreviews/5DP0N1P76E.html> for more details.
* (b) including SQL code within the SQL variable, for example by writing a subquery.

**Private Study**

* Work out how you could mount a denial of service attack against the database - but don’t actually try it : the technicians will be unhappy.

**Additional Sources**

<http://unixwiz.net/techtips/sql-injection.html>

<http://msdn.microsoft.com/en-us/library/ms161953.aspx>

http://www.cmswire.com/cms/web-cms/how-they-hack-your-website-overview-of-common-techniques-002339.php