

Notes for Presentation “Are Your Web Applications Safe? SQL Injection Attacks”

Presented 10/24/2007

Slide 3 – What SQL Injection Is

- From http://en.wikipedia.org/wiki/SQL_injection : “SQL injection is a technique that exploits a security vulnerability occurring in the database layer of an application. The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed.”
- SANS Top 20: <http://www.sans.org/top20/>

Slide 4 – What SQL Injection Is Not

- The exploited vulnerability is not in the Web Server (e.g. PHP, ASP.NET, ColdFusion), or Database Server (e.g. MySQL, Oracle, MSSQL) software, it is in the application run on top of that platform. Therefore even if the machine, Database Server software, and Web Server software are completely patched and there are no zero-day exploits, SQL Injection is still possible if any of your Web Apps are vulnerable.
- However, if an SQL Injection vulnerability exists in the application, an attacker may be able to use that to exploit other vulnerabilities (e.g. buffer overflow bugs) in the Database Server software or any other process running on the Database machine. Therefore, keeping your patches up to date can mitigate damage from SQL Injection even though it cannot prevent SQL injection.
- Of course, you could set your firewall/IPS to block all traffic not from a whitelisted IP, which would greatly reduce the chance of SQL Injection. However, that might not be feasible for public-facing web applications (or even for intranet applications).

Slide 5 - Demo #1: Unauthorized Access

- I developed this application for this presentation, with one intentional security flaw (processing a url parameter without any validation) to allow me to demonstrate SQL Injection. However, that security flaw is very common on the web. In fact, in my research for this presentation I found prominent and public websites considerably more vulnerable than this application.
- By the way, the demo application is secured by an IP lockout so you will not be able to access it. Go ahead and try if you want to, and let me know if you can. Just don't trash the DB before the end of the presentation.

Slide 9 – Let's try SQL Injection

- In ColdFusion, #variableName# outputs the value of the variable.

Slide 10 – Attack failed

- For reference, ColdFusion automatically replaces ' characters with " (two single-quotes, not a double quote), which escapes to a literal single quote in SQL. Older versions of PHP did a similar

thing with “magic quotes”, though newer versions leave that off by default. It is **not** a recommended means of dealing with SQL injection, but it can be effective in many cases because any input that is used within a string literal cannot cause significant SQL injection.

Slide 13 - Success! ... kind of

- ID = asdf would not be a syntax error if RECORD_TABLE had a compatible column named “asdf”, but that is rather unlikely.

Slide 14 - Boom!

- The problem could have been averted if the processing script checked the “ArticleID” parameter to make sure it was numeric, and printing an appropriate error message to the user.
- Parsing the actual query would probably require more sophistication than I’ve yet seen in an automatic SQL Injection attack script. Therefore, we’ll stick with the standard error messages that are much more regular. We’re acting like an automatic attack script to show that SQL Injection can happen to your application without a human targeting you specifically.

Slide 15 - Looking for the User table

- In MSSQL the sysobjects table is present in every database and stores metadata on the tables, primary keys, foreign keys, and such in the database. The sysobjects table has many columns, but the interesting ones for today are id, name, and xtype.

Slide 23 – Bingo

- Speaking of encryption, it’s a good idea to use encrypted storage for any sensitive data. Ideally an attacker will not get much of value even if they download the entire database.

Slide 24 – We’re in...

- I didn’t implement an administrative interface. Hey, it’s secure.

Slide 26 - Demo #2: Modifying data

- An attack script designed to extract sensitive data might not make this intuition, but an attack bot designed to plaster spam all over the web might. Also, the script could look through the syscolumns table to find all the columns and try each one in turn.

Slide 27 - Multiple Queries

- By default, this platform (and others) allows multiple SQL statements per transmission.

Slide 32 – Convinced of the threat yet?

- Have mercy on the application programmers. Validation is not a cut-and-dried question. Generally adding better validation is time-consuming, tedious, does not improve functionality for legitimate users, and often breaks features due to overzealous input rejection.