Bugs Cost Money!
An analysis of the SQL injection bug

By Akash Desai

The software development life cycle today sees bugs being introduced at every stage. Bugs mean money today in the business sense, and a seemingly insignificant bug can be worth millions of dollars to a business.

A bug's life

The software development life cycle today sees bugs being introduced at every stage. Bugs mean money today in the business sense, and a seemingly insignificant bug can be worth millions of dollars to a business. The dollar worth of bugs can be seen as almost proportional to the stage of the development life cycle in which they are caught. A study by NIST indicates that a meager 45 percent of all bugs are actually caught in the same stage where they crept in[1]. The study also suggests that the testing and debugging stages amount to a whopping 80 percent of the total life cycle cost. No matter which stage a bug creeps in from, this much is clear: The cost of fixing a bug caught early is far less than the cost of fixing a bug overlooked for some time.

The painful injection

The consequence of bugs is that applications, Web servers and a host of user interfaces become vulnerable to a variety of security loopholes, such as input validation, access validation, exception handling errors, configuration errors and race conditions. Among these, input validation can be especially intriguing – exploiting it often takes little effort and can lead to complete compromise of the target system. In an input validation attack, an attacker provides an application with an input which the application does not check properly. The application then responds to the specially crafted input, providing the attacker with access to the resource the application is protecting, often a database.

We focus our attention on what has remained, even today, one of the simplest and yet most widespread input validation bugs. These are well known on the Web as "SQL Injection bugs," often in the context of SQL Injection attacks, as hackers are taking advantage of this loophole more and more. SQL Injection bugs are non-validated inputs which leave open a feast of vulnerabilities ready to be devoured by malicious attackers on the Web. The end result of these bugs is that attackers can execute arbitrary SQL queries and commands on the backend database server. Putting things into perspective, if this database belonged to a bank, and an attacker could execute queries on it, all customer information would therefore be compromised.

No matter which stage a bug creeps in from, this much is clear: The cost of fixing a bug caught early is far less than the cost of fixing a bug overlooked for some time.

Consider an electronic form that accepts a username and a password. Ideally, this form ends up querying a database in an attempt to retrieve information. For example, let us assume that the User table shown below is one of the tables in the database:

User

Username	Password	ID_Number

The ID Number of the user is to be utilized in each transaction the user performs. A user enters a username and password, and then a query is executed which retrieves this information about the user for authenticated entry. The query would go something like this:

```
Select ID_Number from User where Username = 'erm' and Password = 'enterprise_risk_management'
```

In the event that this form has an SQL Injection bug left behind from the development stages of the information system, an attacker sitting on the electronic form console can take advantage of this bug in the following manner:

```
Username: ronaldinho
```

1 *The Economic Impacts of Inadequate Infrastructure for Software Testing.* Prepared by RTI for NIST. http://www.nist.gov/director/prog-ofc/report02-3.pdf