

## Chapter 7.9

# SQL Code Poisoning: The Most Prevalent Technique for Attacking Web Powered Databases

**Theodoros Tzouramanis**  
*University of the Aegean, Greece*

### ABSTRACT

*This chapter focuses on the SQL code poisoning attack. It presents various ways in which a Web database can be poisoned by malicious SQL code, which can result in the compromise of the system. Subsequently, techniques are described for the detection of SQL code poisoning and a number of lockdown issues that are related to this type of attack are discussed. This chapter also reviews security mechanisms and software tools that protect Web applications against unexpected data input by users; against alterations of the database structure; and against the corruption of data and the disclosure of private and confidential information, all of which are owed to the susceptibility of these applications to this form of attack.*

### INTRODUCTION

Web application attacks are continuously on the rise, posing new risks for any organization that

have an “online presence.” The *SQL code poisoning* or *SQL injection attack* (CERT, 2002) is one of the most serious threats faced by database security experts. Today it is the most common technique used for attacking, indirectly, Web powered databases and disassembling effectively the secrecy, integrity, and availability of Web applications. The basic idea behind this insidious and pervasive attack is that predefined logical expressions within a predefined query can be altered by simply injecting operations which always result in true or false statements. With this simple technique, the attacker can run arbitrary SQL queries and thus they can extract sensitive customer and order information from e-commerce applications, or they can bypass strong security mechanisms and compromise the backend databases and the file system of the data server. Despite these threats, a surprisingly high number of systems on the Internet are totally vulnerable to this attack.

This chapter focuses on the SQL code poisoning attack. It presents various ways in which a Web database can be poisoned by malicious SQL

## SQL Code Poisoning

code, which can result in the compromise of the system. Subsequently, techniques are described for the detection of SQL code poisoning and a number of lockdown issues that are related to this type of attack are discussed. This chapter also reviews security mechanisms and software tools that protect Web applications against unexpected data input by users; against alterations of the database structure; and against the corruption of data and the disclosure of private and confidential information, all of which are owed to the susceptibility of these applications to this form of attack.

### BACKGROUND

Online businesses and organizations are protected these days by some kind of software or hardware firewall solution (Therriault & Newman, 2001). The purpose of the firewall is to filter network traffic that passes into and out of the organization's network, limiting the use of the network to permitted, "legitimate" users. One of the conceptual problems with relying on a firewall for security is that the firewall operates at the level of IP addresses and network ports. Consequently, a firewall does not understand the details of higher level protocols such as hypertext transfer protocol, that is, the protocol that runs the Web applications.

There is a whole class of attacks that operate at the application layer and that, by definition, pass straight through firewalls. SQL code poisoning is one of these attacks. It takes advantage of non-validated input vulnerabilities to pass SQL commands through a Web application for execution by a backend database, that is, the heart of most Web applications. Attackers take advantage of the fact that programmers often chain together SQL commands with user-provided parameters, and can therefore embed SQL commands inside these parameters. Therefore, the attacker can execute malicious SQL queries on the backend database server through the Web application.

In order to be able to perform SQL code poisoning hacking, all an attacker needs is a Web browser and some guess work to find important table and field names. This is why SQL code poisoning is one of the most common application layer attacks currently being used on the Internet. The inventor of the attack is the Rain Forest Puppy, a former hacker and, today, a security advisor to international companies of software development.

## THE SQL CODE POISONING ATTACK

### SQL Code Poisoning Principles

SQL code poisoning is a particularly insidious attack since it transcends all of the good planning that goes into a secure database setup and allows malicious individuals to inject code directly into the database management system (DBMS) through a vulnerable application (Spett, 2002). The basic idea behind this attack is that the malicious user counterfeits the data that a Web application sends to the database aiming at the modification of the SQL query that will be executed by the

*Figure 1. A typical user login form in a Web application*

Login	
Enter your username and password to login	
<ul style="list-style-type: none"><li>• Forgotten your password? <a href="#">Click here.</a></li><li>• Not a member yet? <a href="#">Sign up here.</a></li></ul>	
Username	<input type="text"/>
Password	<input type="password"/>
<input type="button" value="Enter"/>	

9 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:

[www.igi-global.com/chapter/sql-code-poisoning/8025](http://www.igi-global.com/chapter/sql-code-poisoning/8025)

## Related Content

---

### Experience Matters: The Role of Vicarious Experience in Secure Actions

Leigh A. Mutchler and Merrill Warkentin (2020). *Journal of Database Management* (pp. 1-20).

[www.irma-international.org/article/experience-matters/249168](http://www.irma-international.org/article/experience-matters/249168)

### Multidimensionality in Statistical, OLAP and Scientific Databases

Arie Shoshani (2003). *Multidimensional Databases: Problems and Solutions* (pp. 46-68).

[www.irma-international.org/chapter/multidimensionality-statistical-olap-scientific-databases/26964](http://www.irma-international.org/chapter/multidimensionality-statistical-olap-scientific-databases/26964)

### Understanding Organizational Transformation from IT Implementations: A Look at Structuration Theory

T. Ariyachandra and L. Dong (2009). *Advanced Principles for Improving Database Design, Systems Modeling, and Software Development* (pp. 329-345).

[www.irma-international.org/chapter/understanding-organizational-transformation-implementations/4305](http://www.irma-international.org/chapter/understanding-organizational-transformation-implementations/4305)

### Integrity Constraint Checking for Multiple XML Databases

Praveen Madiraju, Rajshekhar Sunderraman, Shamkant B. Navathe and Haibin Wang (2009). *Advanced Principles for Improving Database Design, Systems Modeling, and Software Development* (pp. 158-177).

[www.irma-international.org/chapter/integrity-constraint-checking-multiple-xml/4298](http://www.irma-international.org/chapter/integrity-constraint-checking-multiple-xml/4298)

### Fighting Pandemics with Physical Distancing Management Technologies

Veda C. Storey, Roman Lukyanenko and Camille Grange (2022). *Journal of Database Management* (pp. 1-16).

[www.irma-international.org/article/fighting-pandemics-with-physical-distancing-management-technologies/305731](http://www.irma-international.org/article/fighting-pandemics-with-physical-distancing-management-technologies/305731)