SQL Injection prevention system

Voitovych O.P.[1], Yuvkovetskyi O.S.[2]

[1] Ph.D., associate professor of Information Security Department, Vinnytsia National Technical University
Khmelnytske shosse str., 95, Vinnytsia, Ukraine, voytovych.op@gmail.com
[2] Student, Information Security Department, Vinnytsia National Technical University
Khmelnytske shosse str., 95, Vinnytsia, Ukraine, alexyuvkovetskyi@gmail.com

Abstract: Existing vulnerabilities of Web-resources threaten the regular work of information systems. The most common Web-resource vulnerability is SQL Injection. This article describes the known approaches to protect Web-applications against SQL Injection attacks and offers SQL Injection prevention system for filtrating incoming data.

Keywords: Web-resources vulnerabilities, Web-applications, classification of Web-resources vulnerabilities, SQL Injection.

Навісний захист від SQL-ін'єкцій

Войтович О.П.[1], Ювковецький О.С.[2]

1К.т.н., доцент кафедри захисту інформації, Вінницький національний технічний університет
вул. Хмельницьке шосе 95, м. Вінниця, Україна, voytovych.op@gmail.com
2студент кафедри захисту інформації, Вінницький національний технічний університет
вул. Хмельницьке шосе 95, м. Вінниця, Україна, alexyuvkovetskyi@gmail.com

Анотація: існуючі вразливості web-ресурсів ставлять під загрозу нормальну роботу інформаційно-комунікаційних систем. Найбільш поширеними загрозами web-ресурсів є sql-ін'єкції. У даній статті описані відомі підходи для захисту від sql-ін'єкцій, а також запропоновано навісний захист для фільтрації вхідних даних.

Ключові слова: Вразливість Web-ресурсів, Web-додатки, класифікація вразливостей Web-ресурсів, SQL-ін'єкція.

INTRODUCTION

There are many web-resources, which are not satisfying modern safety requirements. Web vulnerabilities could jeopardize the finances, assets, personal data and other valuable organization resources. This may cause the bankruptcy or complete liquidation of the company. Today we have many classifications [1], which organizes different types of vulnerabilities and attacks based on various parameters. We consider such classification as OWASP [2], Markov taxonomy [3], WASC threats classification [4] Microsoft STRIDE threat model and the Kaspersky Lab classification [5]. In these and other sources were described the main Web-resources vulnerabilities and attacks [6].

The most common threats of web-resources allocated in each classification are SQL-injections. SQL injection is a type of security exploit in which the attacker adds Structured Query Language (SQL) code to a Web form input box to gain access to resources or make changes to data. An SQL query is a request for some action to be performed on a database. Typically, on a Web form for user authentication, when a user enters their name and password into the text boxes provided for them, those values are inserted into a SELECT query. If the entered values are found as expected, the user is allowed access; otherwise, access is denied.

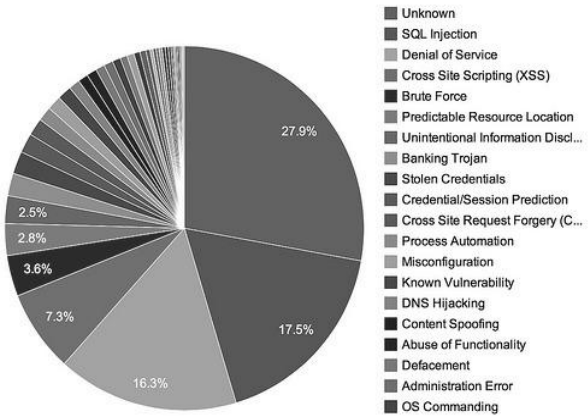

Figure 1 – Web-resource vulnerability statistic

SQL INJECTIONS STATISTIC

We are currently seeing more than 50,000 attacks per day that fall into our SQL Injection categorization [7]. Most of them is automated and try to compromise well known vulnerabilities in common CMS's and

web projects (Joomla, WordPress, vBulletin, etc).

The database is the heart of most Web applications: it stores the data needed for the Websites and applications to "survive". It stores user credentials and sensitive financial information. It stores preferences, invoices, payments, inventory data, etc. Through the combination of a database and Web scripting language we as developers can produce sites that keep clients happy, pay the bills and, most importantly, run our businesses.

SQL INJECTION PREVENTION SYSTEM

SQL-injection, depending on the type of database implementation and the conditions may allow the attacker to execute arbitrary database query (e.g., read the contents of any table, remove, change or add data), get the opportunity to read and / or write local files and execution of arbitrary commands on the server [8]. The attack by using SQL-injection becomes possible because of incorrect handling of input used in SQL-queries.

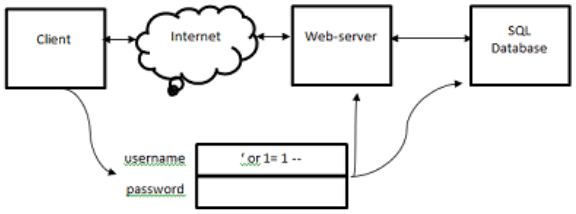

Figure 3 – Typical SQL Injection

In the example above, the username and password are easy to read, but instead of checking for a matching password, it now checks for an empty password, or the conditional equation of 1=1 in username. This means that if the password field is empty and username field has “OR 1 equals 1” (which it does), then a valid row is found in the users table. Notice how the last quote is commented out with a single-line comment delimiter (–). This stops ASP (Active Server Pages) from returning an error about any unclosed quotations.

According to security experts, the reason that SQL injection and many other exploits, such as cross-site scripting, are possible because security is not sufficiently emphasized in development. To protect the integrity of Web sites and applications, experts recommend simple precautions during development such as controlling the types and numbers of characters accepted by input boxes.

To protect against SQL-injection attack we have offered system that performs three functions:

1. Escaping single and double quotes.

Passing along with the value of any string parameter single or double quotes (depending on their use in the attachment), an attacker can modify the structure formed SQL-query. If the parameter that is passed in the query string framed in single quotes, script receives a request with the parameter value quotes' symbol and comments - (or / *), the application processes the request correctly.

```
$name = $db-
>real_escape_string($_POST["name"]);
$query = "SELECT * FROM `users` WHERE `login`
= "".$name."";
$string = "{user_input}";
$bad = array('&', '"', "'", '<', '>', '\\',
'%', '_');
$good = array('&amp;', '&quot;', '&#039;',
'&lt;', '&gt;', '&#092;', '&#037;',
'&#095;');
$sanitized = str_replace($bad, $good,
$string);
```

1. Check the type or compulsory data type assignment.

It is necessary to check all entered data by performing inspection type, length, format and range data. When implementing precautions against malicious input, we need to consider architecture and script execution. For example, if a variable, which was later inserted into the request to be kept numeric data, you must use the forced reduction derived type to a numeric.

```
$id = settype($_GET["id"],"integer");
$email = settype($_GET["id"],"varchar");
```

2. Check the structure of input data by using regular expressions.

If you know that input data have a structure, such as a customer e-mail, phone or date, it makes sense to "drive" the meanings by using regular expressions. Thus, verified the accuracy of the entered data and simultaneously cut off all unwanted characters and unacceptable modification of transmitted input. However, regular expressions can be used also in case of simple queries. For example, you can ensure that the input parameter is a four-digit number, or that it only lowercase letters, number 5 to 10 characters.

```
var ck_fname = /^[a-zA-Za-яA-
яЁёàâäèéêëîïôœùÀÂÄÈÉÊËÎÏÔŒÜŸÇ\-]{2,20}$/;
var ck_email =/^[a-zA-Za-яA-Я_\d][-a-zA-
Za-яA-Я0-9_\.\d]*\@[a-zA-Za-яA-Я\d][-a-zA-
Za-яA-Я\.\d]*\.[a-zA-Za-яA-Я]{2,4}$/;
var ck_password = /^[A-Za-z0-
9!@#$%^&*()_]{6,20}$/;
if (!ck_fname.test(fname)) {
  errors[errors.length] = "Enter valid
First Name .";
        } if (!ck_email.test(email)) {
errors[errors.length] = "You must enter a
valid email address.";
   } if (!ck_password.test(password)) {
  errors[errors.length] = "You must enter
a valid Password min 6 char."; }
```

However, we need to understand that attackers use different methods for different databases, so the same attack on the site using the MySQL database and the website using the PostgreSQL database will have different consequences.

SUMMARY

The proposed curtain protection is designed as a set of scripts that connect to the Web-page resources. This is helping to prevent the attack executions via SQL-injection. This prevention system is mainly designed to work with MySQL.

REFERENCES

[1] Статистика уязвимостей веб-приложений 2012 [Електронний ресурс] – Режим доступу : URL : http://ptsecurity.ru/download/analitika_web.pdf - Назва з екрану.

[2] 2013 Top 10 Vulnerabilities List [Electronic Resource] // 2013: Proceedings. – Mode of access: WWW.URL: https://www.owasp.org/index.php/Top_10_2013-Top_10. - Title from the screen.

[3] Марков, А. С. Систематика уязвимостей и дефектов безопасности программных ресурсов [Електронний ресурс] : Защита информации. Инсайд 3. - 2013. - С. 56-61. – Режим доступу : URL : http://www.npo-echelon.ru/doc/is_taxonomy.pdf - Назва з екрану

[4] Common Web Application Vulnerabilities [Electronic Resource]. – Mode of access: WWW.URL: https://cve.mitre.org.- Title from the screen.

[5] Защита от эксплойтов в Антивирусе Касперского [Електронний ресурс]. – Режим доступу : URL : http://www.kaspersky.ru/downloads/pdf/technology_auto_protection_from_exploit.pdf. - Назва за екрану.

[6] Mirkovich J. A. Taxonomy of DDoS Attack and DDoS Defense Mechanisms [Electronic Resource] // 2004: Proceedings. – Mode of access: WWW.URL: http://www.eecis.udel.edu/ ~sunshine/publications/ccr.pdf. - Title from the screen.

[7] Kid D. SQL Injection And The Threat They Present [Electronic Resource]// 2014: Proceedings. – Mode of access: WWW.URL: https://blog.sucuri.net/2014/10/website-attacks-sql-injection-and-the-threat-they-present.html. – Title from the screen.

[8] Clarke J. SQL Injection Attacks and Defense, Second Edition [Text] / J. Clarke. – Syngress, 2012. – 576 p.