SQL INJECTION ATTACKS AND VULNERABILITIES

Ionel IACOB[1]
Mironela PIRNAU[2]

Abstract: *SQL Injection represents a technique of code injection which exploits a series of problems regarding the vulnerability over the data base security from the computing structure of a certain application, with the main cause being the filtering or the incorrect usage of the processed data conducted by one user. These attacks include: queries to the operating system using the system queries; the usage of the external programs under the Shell orders and queries to the back-end databases by using a SQL code. By incorporating the malicious SQL commands in the content of the parameter, the attacker may trick the application to send a malicious interrogation to the database. The SQL Injection is considered to be an attacking technique over the security and the vulnerability with major impact risk (negative) and consequences of serious levels, both professional and personal. The severity of SQL Injection attacks is limited by the ability and the imagination of the attacker, and to a lesser extent, by the counter measures of defense in depth, such as the connections with reduced privileges to the database server, etc. From the point of view of the security against attacks of SQL Injection OWASP - Open Web Application Security Project type, the validation of all input and output data is recommended, the debugging of all errors generated by the application and the usage of roles and permissions in the database.*

SQL Injection embodies the vulnerability when the attacker tries to introduce pieces of SQL code sequences in the input fields of the application, being sent afterwards towards the database server. A successful attack would allow the attacker the access both the database server and the files of the system. The SQL Injection attacks may be classified according to a series of criteria such as: the channel to obtain data from, the obtained responses from the server, the manner of response of the server, the impact point, etc.

Keywords: *database, SQL Injection, vulnerability, attacks, scanners of vulnerability*

[1] Lecturer, PhD. Romanian American University, Faculty of Computer Science for Business Management iacob.ionel@profesor.rau.ro
[2] Associate Professor, PhD. Titu Maiorescu University, Faculty of Informatics mironela.pirnau@prof.utm.ro; pirnau.mironela@profesor.rau.ro

1. Introduction

Certain weaknesses of some attacks of one hardware or software system may allow the unauthorized users to obtain access. This implies the writing of a SQL query which may allow the display, the alteration and the deletion process from the database via Web forms or directly, by using URLs.

The attacks of SQL injection type allow the hackers to implement through a certain application of a malware code to another system. These kinds of attacks include SQL queries towards the operating system through system queries, through the usage of external programs, through shell commands as well as through back-end databases calls by using SQL queries.

When a Web application sends information form a HTTP query as part of an external interrogation, this must be prudently cleaned so that the possible breaches from the searching algorithm and from the retrieved unsecured information to be avoided.

Otherwise, a hacker or an attacker of a server or of a platform may use a series of incorrect modalities of (re)query a database, by injecting special (meta) characters or malicious commands or the implementation of command modifiers, and later the Web application would consider the interrogation and would return the queries towards the external system to be executed.

The attacks performed with the help of SQL interrogations represent a very outspread and dangerous form of “code injection” because it allows the unauthorized users to take over data and to process data from the tables of a database. In order to exploit a security breach by injecting a SQL code, the attacker must identify one parameter through which the Web application allows the display and the subsequent processing of certain confidential data from the database. By the process of SQL order integration which is malicious in its parameter contents settled as “target”, the attacker may define and implement a series of interrogations that may offer various advantages such as the unrestrained access to certain information, or the alteration of data of private pleasure which affects the control and the security processes of the database. These types of attacks are not difficult to perform because there are various and quite simple possibilities of “testing the vulnerabilities” of data, but also some instruments that may identify these defects. The consequences of this process are extremely damaging because one attacker may manipulate the existent data and may cause disclaimer problems, such as: the cancellation of transactions, to modify balances, to allow the complete disclosure of the entire system data, to destroy the data or to make the data unavailable and thus, to become the administrator of the database.

The SQL code injection is very frequent within the program’s structures of the applications developed in PHP and ASP, especially due to the prevalence of older functional interfaces.

The severity of SQL injection attacks is not only limited by the ability and the imagination of the attacker, but also, to a lesser degree, by the implemented security countermeasures, as well as by the connections holding low privileges of the database server, etc. Generally, SQL Injection may be considered one type of attack with great impact and very serious consequences.

2. SQL Language

✓ The SQL history starts in the IBM laboratories in San Jose where the language was developed in the last years of the 8th decade. The initials stand for: Structured Query Language.

SQL represents a standard language for accessing the MS SQL Server, Oracle, MySQL, Sybase and Informix database servers. Most web applications must interact with a database and most programming languages for web applications such as ASP, C#, .NET, PHP or Java offer means to connect to a base and to interact with it. Each programming language implements its own means of definition, implementation and execution of SQL instructions, and the application developers often use combinations of the latter to achieve their aimed objectives. Without a deep understanding of the foundations of the database that are used within the working process, and without a clear awareness of the potential security problems resulted after the generated code, the programmer can develop uncertain, insecure and vulnerable applications towards the SQL Injection attacks.

✓ Definition and manipulation of data

For the data manipulation the following commands are used: SELECT, UPDATE, INSERT, DELETE:

- ✓ SELECT – extracts data from a table;
- ✓ UPDATE – updates data of certain entries;
- ✓ INSERT – inserts new entries in a table;
- ✓ DELETE – clears entries from a table.

For defining data, the following commands are used, such as: CREATE, ALTER, DROP:

- ✓ CREATE – creates a new table;
- ✓ ALTER – modifies the existent table;
- ✓ DROP – deletes a table, an index or views.

3. SQL Injection Attacks

SQL Injection represents one of the most devastating vulnerabilities whose major impact is reflected in the loss of confidentiality of the information stored in the database of the attacked application, such as: name and password of the user, addresses, phone numbers or security codes of the credit cards, etc.

SQL "Code Injection" represents a vulnerability which may manifest itself when there is the possibility of an attacker to influence the SQL queries that an application sends to a database. SQL Injection is not a vulnerability that exclusively affects the web applications, since any program that accepts entries to form SQL dynamic declarations may become vulnerable.

A SQL code structure will represent a possible attack to the security of the information stored in the structure of the database because there is a diversity of execution and implementation methods of the SQL standard and, implicitly, a variety of available methods regarding the specific coding options. The main manner of SQL Injection attack consists in the direct code insertion within the input and output parameters, which are subsequently chain-linked with SQL commands and then sent on a server to be executed. An attacker can modify a SQL declaration and that precise declaration will be executed with the same rights as the rights of the user by rights of the application.
When the SQL Server is used to execute commands that interact with the operating system, the process will run with the same permissions as the component element that executes the command.

SQL Injection can be used for:

- ✓ avoiding the authentication and access controls;
- ✓ determining the structure of the database;
- ✓ the enumeration of the database;
- ✓ the unauthorized access to the database (passwords, credit cards);
- ✓ unauthorized changes of the data including the deletion of the registries, of the tables, or the insertion of entries;
- ✓ performing commands of the operating system.

Examples of successful SQL attacks:

- ✓ In February, Jeremiah Jacks identified one vulnerability of the Guess.com website to the SQL Injection attacks, through which a hacker, by creating his own URL address was able to download the unauthorized information of over 200.000 clients of the application managed by that certain database, such as: names, passwords or the data on the personal credit cards (used for the online commerce);
- ✓ Subsequently, in June 2003, Jeremiah Jacks discovered that the same specific security problems SQL "code injections" were identified in the website of the e-commerce type PetCo.com. The consequences allowed the attackers to obtain the confidential information regarding over 500.000 credit cards;
- ✓ In turn, the TJK retailer in USA represented the target of the SQL Injection attacks in December 2006, being stealing from the database certain data of over 2.000.000 credit cards;

- ✓ In February 2009, a group of Romanian hackers managed to affect, from separate incidents, the security of the sites of some famous companies from the information security domain, such as: Kaspersky, F-Secure and Bitdefender;
- ✓ At the same time, the Department of Justice on USA accused Albert Gonzales in August 2009 for the theft of 130.000.000 of credit cards for the SQL Injection attacks. Among the affected companies were: Heartland Payment Systems the credit card processing companies, the 7-Eleven chain stores and the Hannaford Brothers supermarket chains;
- ✓ Also by the same "injection" method of unauthorized codes, in April 2011, the website of the Barracuda Networks company, one of the leaders in the "cloud" security of data applications domain, was successfully attacked and the database of the website (including the authentication credentials, "username" - "password" and the password hash) was entirely posted on the Internet;
- ✓ The well-known producer Sony was the target of some SQL Injection attacks in May 2011. Then, the LulzSec group succeeded to compromise some of the company's websites and further displayed in the online environment the information taken over from the stolen database.

Starting with the year 2008 a significant increase of the "code injection" actions of some categories within different online applications and associated databases could be observed, thus worldwide had been recorded hundreds of thousands of attacks, some of the being successful, over the unsecure websites.

By generally using the same action method, the hackers used a type of programs that allowed them the identification of the vulnerable aspects from the implemented applications of codes and the successful exploitation obtained illegally (the display, the modification and even the deletion of certain confidential data or data of the highest importance).

Thus, a specific sequence of "exploit" type executes SQL instructions that locate each tablet from the database structure and implements, for each column of text type within the tablet, a malicious client-side script. Due to the fact that the majority of web applications uses data from the database to create a dynamic content. Consequently the created malicious script will be run in the browser of a certain user of the application or of the compromised website.

4. Causes for an unsuccessful SQL injection attack

4.1. The incorrect manipulation of the escape characters

The SQL databases interpret the character ' (single quotation mark) as a delimitation between the programmable code and the processed data. It is assumed that anything that follows the symbol ' represents a part of a code that must be run, and anything that is framed by the symbol ‘ represents data. Therefore, a vulnerable website can be quickly identified by the simple push of the button of the single character ' in the URL address or in one field from the web page or web application. Below there is an example of code for an application that directly sends the data introduced by one user to a SQL declaration created dynamically:

$SQL = “SELECT * FROM Table WHERE field = '$_GET[“input”] ';”;

If one character ' is introduced as an entry into an application, an error message might occur. The result depends of a series of factors, such as the programming language, the used database and the used protection measures. The cause is represented by the symbol ' which was interpreted as a string delimiter. Syntactically, the executed SQL query is incorrect, and, therefore the database returns an error. The ' mark is used in SQL Injection attacks for the manipulation of the user’s query, thus the attacker can build personal queries that would be subsequently executed by the database server.

4.2. The incorrect manipulation of different types of data

The removal of the character ' through escape or the validation of input for the cancelation of the character ' is not enough. When numerical data is used, it is not necessary to be encapsulated data between characters ', because the numerical data would be considered as strings. In the below example is considered that the parameter will be a whole, and it would be written using single quotation mark.

$SQL = “SELECT * FROM Table WHERE field = $_GET[“user_id”]”;

MySQL implements the function LOAD_FILE which can read a file and returns the content of the file as a string. The following declaration may allow one attacker to read the content of the file of “password” (passwd) type, one that contains the name of the users and the attributes for the users of the system.

SELECT * FROM Table WHERE user_id=1 UNION ALL SELECT LOAD_FILE (’/etc/passwd’)

The introduced data is interpreted as SQL instructions, and the usage of the character ‘ is not mandatory.

4.3. The incorrect assembly of the queries

For certain complex applications, the web developer must allow some declarations to be created dynamically, based on the data for which the query was executed. The following code sequence allows the transmission of introduced data by the user towards a SQL declaration which is created dynamically.

$SQL="SELECT ".$_GET["col1"],",".$_GET["col2"]." FROM ". $_GET["Tabel"];

In case one attacker can manipulate the HTTP query and the attacker can replace the values introduced by the user for the name of the table and its fields, the attacker can display the names of the users and their passwords from the database. A possible example of URL built by the application is the following:

http://www.Test.ro/user.php?Tabel=utilizatori&col1=user&col2=password

4.4. Error exploitation

The unsuitable exploitation of the errors may lead to a variety of security problems for a website. The most frequent problem occurs when detailed error messages that contain information about the database and error codes are displayed to the user and to the attacker. The messages that reveal implementation details may offer to one attacker important clues regarding the possible weaknesses from the application. The error detailed messages may be used to extract information from the database regarding the way SQL Injection declarations can be modified or built.

5. The success of a SQL Injection attack

The databases of the application are implemented for several predefined users. Microsoft SQL Server uses the account of the administrator "sa", whereas MySQL uses the accounts "root" and "anonymous", but Oracle creates the accounts "SYS" and "SYSTEM". All these codes are created implicitly whenever one database is created. For these codes, the default passwords are well known.

Some system administrators and some database administrators install database servers to run, from accounts of system administrators with high privileges, such as root, SYSTEM or Administrator. But the servers of services for the databases should run as an unprivileged user, if this is possible. All these may reduce the risk of deterioration of the operating system and of other processes, in case of a successful attack executed over the database. Each type of database server requires a certain model of control over the access through which different privileges are assigned to the users' accounts regarding the prohibition to the data access, regarding the execution of stored procedures or of other functions specific to the database. In a great measure, the developers of applications program codes for the connection to a database by using predefined accounts having default privileges, instead of creating users accounts specific to the tasks of the applications, thus ignoring the fact that, when an attacker exploits one SQL Injection vulnerability within an application that connects to a database with a privileged account code can be executed in the database with the privileges of that certain account. For the increase of the security level of the database with separate privileges according with functional requirements of the obtained application.

For a hacker to succeed to implement a SQL Injection attack, this will be prior to the knowledge of the available resources, the typology of the data already installed, defined tables, as well as the list and the structure of the attributes or of the fields that may become vulnerable (or that represent the subject of the attack). When an attacker uses a SQL "code injection", the attacker will try to access the metadata of the database for several times.

The metadata represents information about the data contained by a database, such as: the name of the database or of the tables, the fields or the attributes from the structure of each table, as well as the already existed relations among them. For the MySQL server (from ver.5.0), this data is stored in the virtual database INFORMATION_SCHEMA and can be accessed through the commands SHOW DATABASES and SHOW TABLES. Each MySQL user has the right to access the tables from the databases but only the entries are visible, the ones that correspond to the objects for which the user has the corresponding access privileges.

The MySQL declaration for the enumeration of all the tables accessible to the current tables is:

```
SELECT table_schema, table_name FROM
INFORMATION_SCHEMA.tables;
```

It is not possible for the access to the database to be hidden or canceled INFORMATION_SCHEMA from a MySQL database.

In order to prevent the attacks of SQL Injection type, one of the most efficient method is implementing a series of complex techniques of validation of entry data, thus the existence of the possibility of inserting other data or inconsistent characters by the attacker does not exist. The methods through which the level of vulnerability of information can be limited, or found implemented at the level of databases and of the applications can be described as follows:

- ✓ **The usage of the well-defined variables and the definitions of the columns from the database**

The manipulation and the storage of numbers (the session IDs, codes, etc.) as whole numbers or types of proper numbers. The strings must have only alphanumeric characters and punctuation marks and the characters of SQL syntax must be rejected.

✓ **The result of the query is assigned to the well-defined variable**
If the application is searching for a numerical value, then the assignation represents the result of a whole number, so the attackers cannot extract information from the database.
For example, if a variable which was going to be displayed within a browser accepts only whole numbers, then the possibility of obtaining and of displaying the name of a column should not exist. Certain attacks are strictly restricted by this technique.

✓ **The length of data to be limited**
Many strings of characters are not limited to an adequate length for their purpose. For example, the name of one user should not be stored or manipulated by a variable of 256 characters long. The number of limited characters introduced in one field may restrain the success of SQL injections, thus reducing the length of strings of characters that one attacker can introduce.

✓ **Creating queries by the concatenation of the avoidable strings of characters**
For example, a view function is created, or another procedure that operated the variables from the application. From the concatenation of the strings of characters – the resulted query from the data taken over from the users is: "Select fields from table Where + conditions". This can be very vulnerable considering the SQL injections, but a procedure may generate an error if the introduced data is incorrect and this will not allow the manipulation of the query.

✓ **The separation of data and accessing databases on level roles**
Every application must use accounts with accessing privileges for certain tables which are needed to a function. These internal tables of the database, along with the management tables of the accounts and of the variables of the system must be accessible only to the accounts with access rights.

6. Vulnerability scanners for the databases

The best way to evaluate the possible vulnerabilities of a certain application is a scanner of vulnerabilities; a tool that is used to test the security of a system or of a network. The best way to evaluate the possible vulnerabilities of one application is a scanner for vulnerabilities, a tool that is used for testing the security of a system or of a network. This returns information about the vulnerabilities of the system, the ones that can be later used by other applications or policies to provide protection.

The applications of evaluating vulnerabilities are divided into the following types:

- ✓ **Host –** An application for the evaluation of the vulnerabilities of the host can scan and report the vulnerabilities that are found only on the computer on which it installs and does not interact with other systems. The advantage of using this application is that the scanner has complete access to all the resources of the system they can run on. The main disadvantage is that the scanner may use many of the resources of the device they run on.

- ✓ **Service –** There are applications that verify the vulnerability from a series of computers and services within the network. This type of applications varies, from scanners of ports, to scanners that detect the computers that are functioning and that are extracting data.
- ✓ **Application –** Most of the scanners of this type are dedicated to web applications. They may help to locate the pages that should not be normally accessible, but that may also execute complex operations such as the manipulation of the application to obtain information.
- ✓ **Active/passive –** The active scanners try to evaluate one type of network, a system or a service, by using attack strategies that might be used for a real attack. These scanners generally identify certain vulnerabilities caused by defective configurations or by system patches of missing systems. The advantage of these scanners is that the administrator detains the control over the scanning time. Among the disadvantages there can be mentioned that these scanners may lead to the discontinuance of the functionality of the systems, without being recommended over critical systems. The passive scanners do not affect particularly the resources of the system because they only monitor the data from one system and executes processing operations of the data on a separate device. These scanners behave similarly with the detection systems of the intrusion, meaning that they intercept the data about the system and evaluates them by using a set of rules. The main advantage of this type of scanner is that it can function continuously because it does not use the resources of the system. The disadvantage is simply represented by the fact that the passive scanners will only report the derived information which is easily available, and which will not be able to be executed thorough checks. Among the types of tests used by one passive scanner, the following can be mentioned: the determination of the version of one program or the checking of the presence of a certain program.

- ✓ ***Examples of vulnerability scanners for databases***
- ✓ **Trustwave AppDetectivePro**[3] represents a database scanners and data stores that can immediately discover configuration errors, problems regarding the access and identification control and missing updates or any other unsuitable combination of settings that may lead to the escalation of privileges, to denial-of-service attacks, to information leakage or to unauthorized modification of the data.

[3] https://www.trustwave.com/Products/Database-Security/AppDetectivePRO/

- ✓ **Integrigy AppSentry**[4] represents a new generation of security scanners and evaluation instruments of vulnerability. Unlike other security scanners, AppSentry uses audits and written security controls especially for the application that would be tested. The attackers and dissatisfied employees often exploit the security problems at different levels. Thus, only a complete and comprehensive validation would cover all the risks within one environment on various levels. The advantage of AppSentry is that it is no need for it to hold separate instruments for the operating system, for the web server, database or for the application. AppSentry represents one single instrument that may validate and audit the security of the entire technological stack of the application, from the operating system to the layer of the application. The module of validating the application searches within the entire configuration of the application and within the processing of the transactions to identify the security risks and possible frauds. By underlying the weaknesses from the audit procedure and by initiating alerts in case of suspicious actions, AppSentry can identify the missed risks resulting from the traditional testing of vulnerabilities.

The manner of testing vulnerability uses advanced penetration techniques to discover the security risks within the operating system, web servers, application servers, databases and applications. The tests for the already known exploits and for the configuration errors are externally executed towards the application with the attempt of penetrating it. The common and known ports, the web directories and the accounts of the database are tested for the identification process within the configurations.

The manner of cracking passwords (default passwords, dictionary or list attacks) by using brute force can be applied: to the operating system, the web server, the database and to the authentication process within the application.

- ✓ **Imperva Scuba** is a free instrument that scans the complex databases to discover security vulnerabilities and configuration errors. The reports provide information for measures that can be immediately considered to reduce the risk level and to update vulnerabilities. Scuba offers a number of almost 1200 of tests that can be run for different databases: Oracle, MS SQL Server, SAP Sybase, IBM DB2, Informix and MySQL. These tests can be run in real time, without the degradation of the performance of the database servers because Scuba does not exploit the vulnerabilities it identifies. Scuba offers rapidly an analysis of the security level and of the infrastructure of the database by displaying configuration defects such as weak passwords, known security risks or the lack of critical updating. All the identified risks represent a priority and are presented in reports very easy to be understood, along with the instructions regarding the way of

[4] http://www.integrigy.com/products/appsentry

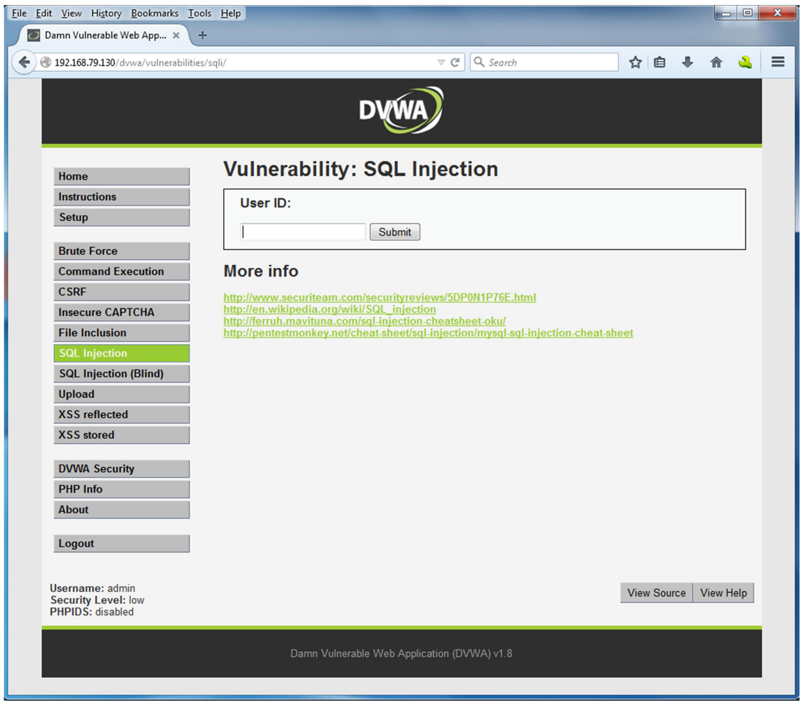
solving the problems regarding vulnerabilities. This helps to reduce the risks and to fulfill the requests conformance requirements without costs, labor force or the right type of expertise claimed by other applications. Regarding MySQL, the following journal files are useful for the debugging of the application or for improving performance.

- **Error Log**. It contains information about the errors that occur during the server's run (includes the server's switching on and off);
- **General Query Log**. Represents a general recording of the activities of the mysqld server. The server writes information in this file whenever clients connect or disconnect, and it also records each SQL instruction received from the clients. This general journal of queries can be useful when an error or malware activity is to be suspected.

7. Practical example of SQL injection

The used application is named: Damn Vulnerable Web Application. DVWA (*http://www.dvwa.co.uk/*) and it is a web application written in PHP/MySQL with major vulnerabilities. It is developed with the purpose of becoming a tool for testing the abilities and the instruments of specialists from the security domain within a legal environment.

It helps the web developers to understand better the security processes of web applications and to help the teachers/students to learn about the security of web applications within a supervised environment.



Vulnerability: SQL Injection

User ID:

More info

http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.8

Step 1. To verify the functionality of the application
Step 2. To verify the vulnerability to SQL Injection
Step 3. To exploit the vulnerability level
Step 4. To identify the tables from the databases
Step 5. To verify the fields from the tables of the database
Step 6. To extract sensitive information from the database

✓ Conclusions

In the most serious cases, the SQL injection may allow for an anonymous attacker to read or to modify all the stored data within the database and even to detain the total control over the server on which databases run. The SQL types of attacks occur when malicious data is sent to one code interpreter as part of one command of one query. The malicious data may trick the interpreter to execute involuntary commands or to access data without an appropriate authorization. As the degree of awareness over the security of web applications has evolved, the SQL Injection vulnerabilities have gradually become even more difficult to be detected and exploited. Many modern applications avoid the SQL injection by using APIs which, if properly used, are inevitably secure against the SQL types of attacks. The SQL injection usually occurs in accidental cases whenever these defensing mechanisms cannot be applied. The process of finding the SQL injection sometimes represents a difficult task, demanding perseverance to locate one or two instances from an application where ordinary controls were not applied.

References:

[1] Binbin Qu. - Design of automatic vulnerability detection system for Web application program, 4th IEEE International Conference on Software Engineering and Service Science, Beijing, 2013.

[2] Ciobanu (Defta) C.L., Ciobanu (Iacob) N. M., ”E-learning Security Vulnerabilities”, Procedia-Social and Behavioral Science Journal, Vol. 46, 2012, pg. 2297-2301.

[3] Clarke J. - SQL Injection Attacks and Defense, 2nd Edition, Syngress-Elsevier, Waltham, MA02451, USA, 2012.

[4] Damn Vulnerable Web Application (DVWA), http://www.dvwa.co.uk/ (Accessed 25 Jul.2019).

[5] Dubois P. - MySQL Cookbook: Solutions for Database Developers and Administrators, O'Reilly Media, USA, 2014.

[6] Iacob I. ”Programare Oracle 10g. Soluții informatice pentru dezvoltarea aplicaţiilor economice utilizând PL/SQL şi ORACLE DEVELOPER”, Editura Universitară, ISBN:978-606-591-313-4, București 2012;

[7] Pfleeger C.P., Pfleeger S.L., ”Security in Computing, 4th Edition, Prentice Hall”, Boston – MA, USA, 2006.

[8] PHP Top 5 https://www.owasp.org/index.php/PHP_Top_5(Accessed 20 Aug.2019)
[9] Pirnau M., ”General aspects os some causes of web application vulnerabilities”, Memoirs of the Scientific Sections of the Romanian Academy Tome XXXVIII – Computer Science, București 2015.
[10] Preventing SQL Injection, https://docs.oracle.com/cd/E41633_01/pt853pbh1 /eng/pt/tpcd/ task_PreventingSQLInjection-0749b7.html /(Accessed 25 Sep.2019).
[11] SQL Injection www.owasp.org/index.php/SQL_Injection/(Accessed 15 Aug.2019).
[12] Tăbușca Al., Tăbușcă S.M., Garais G., ”Iot and EU Law – E-Human security”,Valahia Journal of Economics Studies – Vol. 9 (23): Issue 2, 2018.
[13] Teodorescu H.N., Iftene E.F., ”Efficiency of a Combined Protection Method against Correlation Power Analysis Side-Attacks on Microsystems”, Int J Comput Commun, February 2014, 9 (1), 79–84.
[14] Wassermanng Su.Z., ”Static detection of cross-site scripting vulnerabilities”, Proceedings of the 30th International Conference on Software Engineering, New York, NY, USA: ACM, 2008, 171–180.
[15] Welcome to the Tutorial on Defending Against SQL Injection Attacks! https://download.oracle.com/oll/tutorials/SQLInjection/index.htm /(Accessed 25 Aug.2019).
[16] When SQL Injections Go Awry, ”Incident Case Study”, /https://blogs.technet.microsoft.com/antimalware/2008/05/30/when-sql-injections-go-awry-incident-case-study/ Accessed 15 Aug.2019.
[17] https://tools.cisco.com/security/center/resources/sql_injection Accessed 23 Aug.2019.