ISSN 1846-6168 (Print), ISSN 1848-5588 (Online)
https://doi.org/10.31803/tg-20211203090618

Preliminary communication
Received: 2021-12-03, Accepted: 2022-02-04

The Importance of Developing Preventive Techniques for SQL Injection Attacks

Nenad Bedeković, Ladislav Havaš, Tomislav Horvat*, Dražen Crčić

Abstract: Many intentionally vulnerable web applications are circulating on the Internet that serve as a legal test ground for practicing SQL injection attacks. For demonstration purposes the attacks will target an Acunetix test web application created using PHP programming language and MySQL relational database. In the practical part, the execution of the attack itself largely depends on the database management system, so the displayed syntax is intended only for the MySQL database management system. Example of an automated attack will be executed on SQLmap in a Linux Kali virtualized environment. Security guidelines with a purpose of protecting databases are also discussed.

Keywords: database; MySQL; relational database; SQL; SQL injection attack; SQLmap

1 INTRODUCTION

Nowadays, databases are used for almost all systems and websites because data must be stored somewhere. Collecting massive amounts of structured and unstructured data, known as Big data, has revolutionized today's way of life. Large data amounts are stored in a database so there is an increasing risk involved in the operation of security systems. One of possible security attacks are SQL injection attacks. An attacker sends a malicious SQL query to a vulnerable field that exploits a vulnerability in the SQL implementation of the application itself. Although this vector can be used to attack any database and the most common targets are still web applications. Attack vector is a method used by a malicious person to penetrate or unauthorized access into the vulnerable system.

Providing a unique database entry, most web applications allow visitors to enter and retrieve data. Improperly programmed applications will allow attackers to use vulnerabilities and access the database, after which an attacker can manipulate the content of a database at will. Discovering which database management system (DBMS) is being used is a key to further exploiting the vulnerability. Without discovering DBMS it is not possible to determine which tables to attack, which built-in system functions are involved and which detection methods to avoid. SQL injection is nowadays one of the most popular attack vectors because of its simplicity and does not require a lot of resources and expertise. The attack vector defines the path that is used to take advantage of the security breach.

Since 1998 until today SQL injection attacks still hold the top place according to the Open Web Application Security Project (OWASP) organization, aimed at improving software security. Interestingly, through its lifetime SQL injection attacks have not changed much. A common vulnerability, which is relatively easy to fix, continues to create problems and if not detected before implementation provides a window of opportunity for potential attackers. This paper will demonstrate a direct and automated approach for finding vulnerabilities and performing attacks using SQL injection attacks. A deliberately vulnerable free and legal PHP application on "Acunetix.com" will be used with a purpose of better understand programming errors.

Acunetix was founded to combat the alarming rise in web attacks. It delivers a comprehensive set of options for automated and manual penetration testing tools, allowing security professionals to perform vulnerability assessments and repair detected threats with just one product. The web application uses MySQL as a DBMS, so we will be using attacks that are specific to that system.

Section 2 gives an insight into SQL injection attacks related scientific papers and newspaper articles. Section 3 defines the basic SQL injection attacks concept, while Section 4 depicts about automated SQL injection attacks. Section 5 provides an analysis of the obtained results and possible prevention techniques. Finally, Section 6 concludes the paper.

2 SQL INJECTION ATTACKS

Despite the fact that more than 20 years have passed since the first SQL injection attacks and the exponential development of technology, we are still witnessing many attacks on vulnerable web-based applications. As the number of available web applications increases, so does the number of potential omissions in the systems that malicious hackers will try to use for their attacks. Numerous techniques for SQL injection attacks have been developed, as well as techniques for preventing such attacks. More about that will be said later in this paper.

2.1 A Brief History of SQL Injection Attacks

In 1998, security researcher Jeff Forristal under the pseudonym "rain.forest.puppy", published an article in Phrack Magazine, describing problems with the Microsoft SQL Server and described how a person with basic programming skills can obtain sensitive information from a database with unauthorized SQL commands on an unsecured website [1]. Author informed Microsoft about the problem but they did not perceive it as a problem. Four years later, more precisely in 2002, SQL injection attacks attracted some attention in computer security community. Daily increase in a number of computer viruses and worms alarmed security experts to start developing software tools for defend against various cyberattack methods. One of the informers was Bill Gates who sent a letter to his employees and warned them about the security their software [2]. This initiative has triggered the development of secure tools and the gaps created by poor programming are beginning to be taken more seriously. Since then, countless articles have appeared about SQL injection attacks. The first security tool, Analysis and Monitoring for Neutralizing SQL Injection Attacks (AMNESIA), for preventing and detecting SQL injection attacks appeared in 2005. AMNESIA makes a table model of possible malicious SQL code that would be used for the injection and then analyzes each entry point where the code could be inserted [3].

OWASP, a non-profit community on the Internet, aims to improve software security by regularly publishing documentation, tools, and technologies to protect against potential security risks. Every few years, OWASP publishes Top 10 SQL injections attacks and tips for protecting against ten most critical security web application risks [4].

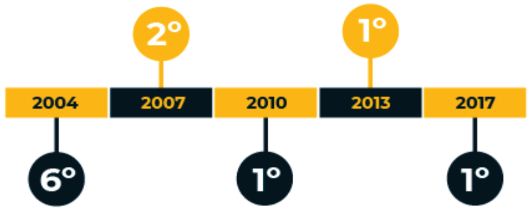


Figure 1 OWASP Top 10 SQL injection attacks by year [4]

It can be concluded that the growing popularity lies in the ease of use. On a vulnerable web application with only a few lines of code on the application layer, it is possible to send queries to the database. Unlike other attack vectors, the resources required to perform the attack are extremely minimal and reduced to a single computer. Other attacks require huge resources and time, involving thousands of computers like a botnet in carrying out attacks. Some of the SQL injection attacks are shown in Tab. 1.

Table 1 SQL injection attacks by year and description

Year	#	Description
2009	[5]	130 million stolen credit card numbers, the biggest identity theft in American history
2009	[6]	RockYou's social media web application development site suffers a leak of 32 million unencrypted user accounts and passwords stored in a text database
2011	[7]	The MySQL official DBMS page had a leak with user accounts and passwords of leading people in the company
2011	[8]	Sony Pictures, the official website of the company Sony, had a database leak with millions of unencrypted user accounts and related information stored in text format
2012	[9]	Yahoo had a database leak with 450,000 user accounts and associated passwords stored in text format
2015	[10]	Vtech, a global supplier of electrical products, suffered a database leak of five million user accounts
2016	[11]	The U.S. National Election Assistance Commission in charge of assisting in the conduct of local and state elections had a database leak of administrator accounts
2019	[12]	Georgia Institute of Technology one of America's leading technical colleges had a database leak of 1.3 million users

2.2 Web Application Architecture

Web application architecture idea is that there is a front-end interface that communicates with the back-end on the server side. Applications are built in layers, so there are several types of architectures and the choice of architecture depends on the purpose of the application. An example is a web application created using three-tier architecture, consisting of a presentation, logical and data layer. It is often used in applications as a specific type of client-server system, providing numerous benefits for the development environment by modifying the user interface, logic, and storage layers. It is always possible to flexibly develop, upgrade or relocate any layer independently of another. The basic rule of this model is that the presentation layer communicates with the data layer through the logical layer.

The presentation layer belongs to the highest layer of application architecture. It is considered as a communication layer where the client communicates with the application via the user interface and is usually developed through HTML, CSS and JavaScript.

The purpose of the logical layer is to control the application functionalities. It is the main part of the application where the obtained data from the presentation layer is computed and performed. The development of this layer is usually done through PHP, Java, Python, C ++, Ruby, .NET, etc.

The data layer consists of a database and a DBMS such as MySQL, Oracle, Microsoft SQL Server, PostgreSQL, etc. The information is here kept separate from the application and the logical layer, so it can be stored and returned on the server.

2.3 Related Work

Great interest in analyzing this security issue was aroused by Jeff Forristal, who studied the security of a growing number of web-enabled applications in the late 1990s. He published his previous research in [1], in which he studied the problem related to Web Bot programs, developed in 1997, and the security problems of MS SQL Server version 6.5.

The development of preventive methods for ASP or PHP SQL injection attacks, which are being worked on intensively, leads to rules that every application developer must incorporate into their applications. At the same time, the development of new versions of database servers and scripting languages, opens up new possibilities for attacking such systems. This is discussed by XuePing-Chen in [13]. He notes that SQL injection is often used in combination with some other hacking methods, which greatly increases the vulnerability of the system.

In addition to the analysis of standard SQL injection attack methods analysis, some authors provide an overview of advanced methods and appropriate defensive mechanisms [14].

When developing web applications, developers try to prevent SQL injection using their personal experience in coding application programs. Despite the extensive experience of these professionals, SQL injection is still the most vulnerable of typical web applications according to a 2014 report from the National Security Agency (NSA) [15].

Trying to prevent as many possible coding flaws as possible, some authors also introduce machine learning methods in prevention. Thus Kamuto and Soomlek [16] train 4 machine learning models (Support Vector Machine, Boosted Decision Tree, Artificial Neural Networks and Decision Tree) on 1100 samples of vulnerable SQL commands.

In addition to the aforementioned scientific papers on SQL code insertion attacks, there are many review papers that address the issue of comprehensive ways to use the weaknesses of web-based applications and techniques for preventing SQL injection attacks.

Recently, the authors in paper [17] provided an extensive overview of the most commonly used methods and techniques for unauthorized intrusion. Authors also proposed a method for avoiding SQL Injection based on Blockchain technology concept for overcoming this vulnerability via nodes verification using IP.

3 THE BASIC SQL INJECTION ATTACK CONCEPT

SQL injection attacks are usually performed to get inaccessible information by sending queries to the database and manipulating the applications programming logic. Such attacks allow attackers to bypass authentication and gain access to the database, thereby copying, modifying and deleting the data. Reconnaissance and information gathering are always the first step of any successful attack plan and identifying and testing the system configuration greatly helps in choosing how the attack will be performed.

SQL injection attack is a technique where a specially defined SQL query is inserted into the data entry field. This can be any input field in the application or a URL (Uniform Resource Locator) address. A security vulnerability is exploited where the data entered is not checked by the server because the success of the attack depends on the syntax of a correctly written SQL query. If application through sent query responds with an error in the database, it suggests there is a possibility that the application is vulnerable. Using information from the generated error, the attacker is able to understand how the database works.

It is also important to visualize the application programming logic in order to modify the query that caused the original error and test further database behaviour. On the data received by the user, the application server tries to connect to the database and forwards the query. The same communication channel is used for attacking and collecting information, so it is possible to get an insight into the entire database through only one input field. After a successful attack, the attacker can read and manipulate existing data by deleting or creating new ones, thus even becoming a database administrator. If the received entry has passed the check and the data matches those in the database, the query is true, and the user successfully logs in (authentication phase) but if it does not match then it is false and access will be denied. An attack occurs when the application processes a specially written user entry in a way that allows entering the command context instead of data context. This changes the structure of the SQL statement being executed. The ultimate goal of each SQL query is to be true, made up of the resulting entry (string) and command (statement) that control how the query will be executed. For example, putting any Boolean expression with an always true OR statement that returns true if either of the two conditions are true means that there will always be a true result. In this way, if the entry point is not sanitized and does not check the entered data, the attacker successfully logs on to the page, regardless of the password. Same logic can be used via URL by using web browser.

3.1 Error-based SQL Injection

When the average users encounters a confusing application error, they often ignore it and don't think much about it. In the right hands the information displayed can reveal a lot about how the system works. Errors and the information displayed in them, greatly help the attacker in choosing the vector of the attack and understanding the query sent to the database.

In order to find a way to execute a SQL injection attack, one starts by inserting special characters intended only for the database into the user application input field or directly through the URL. By monitoring the communication between the application and the database attackers follow how it will behave. In the event that an error occurs in the database, the application can generate the error as feedback and provide useful information about the database.

3.2 UNION-based SQL Injection

UNION operator is used to combine two or more SELECT statements results. Each SELECT statement must have the same number of columns, columns in each SELECT statement must have the same order and columns should contain similar data types.

When an application is vulnerable and the query results are displayed within the application response, the UNION operator can be used to retrieve data from other tables that are not visible in the database. The UNION operator is often used in attacks where databases are discovered and leaked.

3.3 Blind SQL Injection

Blind SQL injection derives name from the interaction with a database where an attacker manipulates the database without directly seeing the results of malicious actions. Unlike Error-based attacks, Blind SQL attack does not require any visual cues that something happened to the database.

In previous attack types, queries were made with predictable logical statements as the condition was true or false. Here a condition is used that could be true and thus reveal some information without showing feedback.

Using a Boolean logic, it is necessary to determine whether a query can be considered true or false. By sending a logical query to the vulnerable application database, a possibility of SQL injection attack can be determined. The feedback allows an attacker to assess how logical queries are treated, whether an application returns a true or false response to a malicious query even though data from the database is not visually displayed.

3.4 Time-based SQL Injection

Another way of performing blind SQL injection attacks are time-based attacks. There are certain variations, but most DBMS support functions for time manipulation. It is based on sending SQL queries where the database will wait a certain amount of time or be late before responding with feedback. The response time will reveal whether the result of the malicious query is true or false, because depending on condition confirmation, the time delay will be executed and the response will be longer than it usually takes.

Sometimes the results of a malicious logical query do not differ much, so the attacker creates an additional difference in the response he receives from the database. For example, MySQL server supports functions like SLEEP() and BENCHMARK() that allow the time delay.

4 AUTOMATED SQL INJECTION ATTACKS

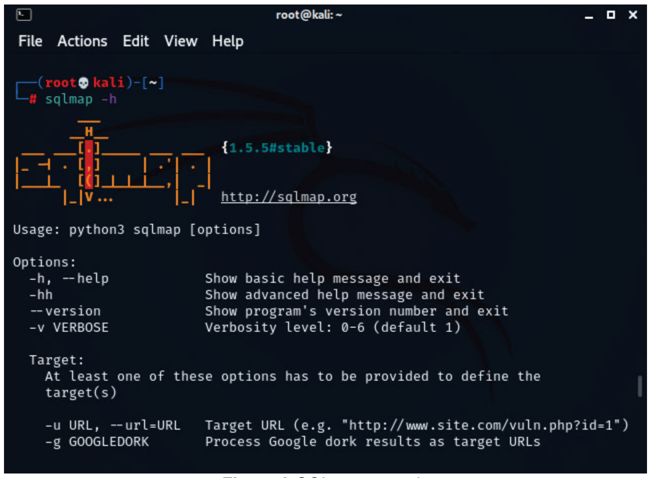
SQL injection attacks can be also performed through specialized software tools and scripts that can perform some of the key tasks required to successfully execute an attack much faster. Some systems or web applications can also be quite complicated, so manual testing requires a higher level of expertise and the ability to track large amounts of programming code. These automated tools are used not only by attackers but also by security experts because constant optimization simplifies and save time in finding vulnerabilities. It should be noted that even if automated tools save time, they are not always considered reliable because there is a possibility they will skip something or due to technical capabilities they cannot perform something that could have been done with the direct approach.

Nowadays there is a wide range of SQL injection attack tools. SQLmap searches and indexes pages on a web application and looks for input windows and forms to send malicious SQL queries with a purpose of causing a system syntax error. It also contains various customization options that allow the implementation of various attack techniques and a functionality for decrypting hashed passwords. The Acunetix web applications will be used for testing.

4.1 SQLMap

SQLmap is one of the most popular and powerful automated SQL injection attack open source tools developed in Python programming language [18]. It automates the detection process and exploits possible security vulnerabilities through SQL injection attack. If an attack is performed through an HTTP vulnerable link, SQLmap can scan, retrieve data from the database and under certain conditions input data into the database.

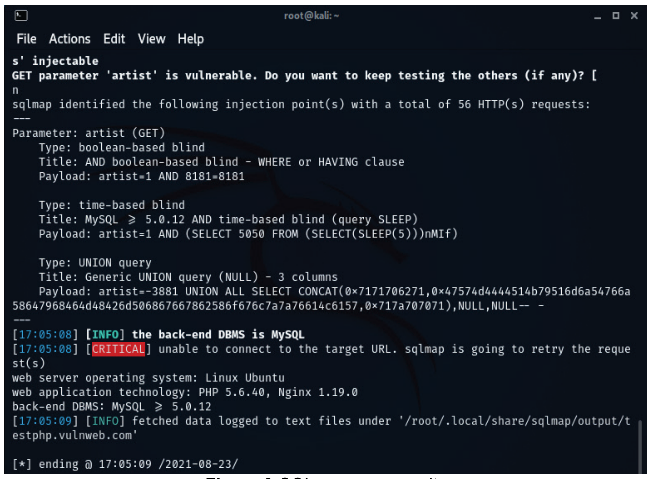
SQLmap does not have a graphical interface, so everything is done by typing console commands. For the purposes of this paper SQLmap 1.5.5 version will be used on a virtualized Linux Kali environment. Fig. 2 shows SQLmap console and how the distribution called Kali was created, which contains a large number of tools for penetration testing of computer networks and systems.



```
┌──(root💀kali)-[~]
└─# sqlmap -h

                                   {1.5.5#stable}

                                   http://sqlmap.org

Usage: python3 sqlmap [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
    target(s)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -g GOOGLEDORK       Process Google dork results as target URLs
```

Figure 2 SQLmap console

Through previous examples, it can be concluded the value in the artist parameter is vulnerable. Artist parameter is the identification value on the page, more precisely the string unique to the content. Changing the specified string would display other content, and the prevention of presented vulnerability can be done using specially prepared queries. It is dynamic and possible to change it with a SQL statement. In the console, it is sufficient to specify the link described as the target parameter. Through a series of questions SQLmap will scan and test different attack techniques. Because aspects of the web application are searched and depending on the results, SQLmap will ask what exactly we want to do and what attacks are possible if vulnerabilities are found.



```
s' injectable
GET parameter 'artist' is vulnerable. Do you want to keep testing the others (if any)? [
n
sqlmap identified the following injection point(s) with a total of 56 HTTP(s) requests:
---
Parameter: artist (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=1 AND 8181=8181

    Type: time-based blind
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: artist=1 AND (SELECT 5050 FROM (SELECT(SLEEP(5)))nMIf)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=-3881 UNION ALL SELECT CONCAT(0x7171706271,0x47574d4444514b79516d6a54766a
58647968464d48426d506867667862586f676c7a7a76614c6157,0x717a707071),NULL,NULL-- -
---
[17:05:08] [INFO] the back-end DBMS is MySQL
[17:05:08] [CRITICAL] unable to connect to the target URL. sqlmap is going to retry the reque
st(s)
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ⩾ 5.0.12
[17:05:09] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/t
estphp.vulnweb.com'

[*] ending @ 17:05:09 /2021-08-23/
```

Figure 3 SQLmap scan results

SQLmap supports attacks using the HTTP GET request. GET requests are used when a client, like a web browser, requires certain resources from a server by using the HTTP protocol. Looking at the URL links in the web browser, URL links usually start with a question mark (?), followed by the name of the variable, value and separated by an equal sign

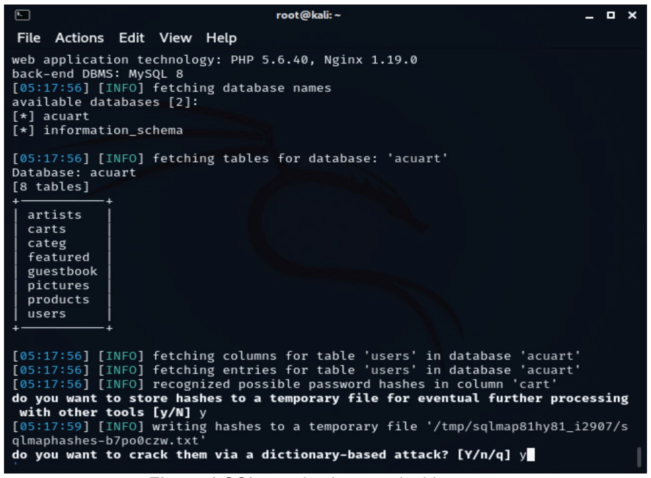
(=). If the link contains more than one parameter then they are separated by the ampersand (&) symbol. Using a specifically designed command SQLmap can confirm that the application is vulnerable by testing different variations of malicious code. SQLmap also details which SQL query should be used to perform an attack. The results shown in Fig. 3 are automatically saved in the system log file for possible future analysis.

SQLmap can also reveal that the operating system, PHP web server and MySQL database system, along with the associated versions. Accordingly, it is possible to go deeper and to find out the names of the database and extract the information. SQLmap documentation contains many configurable parameters but for the purposes of this paper only a few of them, shown in Tab. 2, will be demonstrated.

Table 2 Parameters and their description

Parameter	Description
--dbs	enumerate DBMS databases
--tables	enumerate DBMS database tables
--dump	dump DBMS database table entries
-D	DBMS database to enumerate
-T	DBMS database table(s) to enumerate

Using a few lines of code, SQLmap was able to gather useful information from the database and provide almost direct access (Fig. 4).



```
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL 8
[05:17:56] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

[05:17:56] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----------+
| artists   |
| carts     |
| categ     |
| featured  |
| guestbook |
| pictures  |
| products  |
| users     |
+-----------+

[05:17:56] [INFO] fetching columns for table 'users' in database 'acuart'
[05:17:56] [INFO] fetching entries for table 'users' in database 'acuart'
[05:17:56] [INFO] recognized possible password hashes in column 'cart'
do you want to store hashes to a temporary file for eventual further processing
 with other tools [y/N] y
[05:17:59] [INFO] writing hashes to a temporary file '/tmp/sqlmap81hy81_i2907/s
qlmaphashes-b7po0czw.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] y
```

Figure 4 SQLmap database and table names

In a real scenario the attacker would try to gain a higher level of access. To do this, attacker should try to decrypt the hashed passwords and log in as a system administrator which can allow commands directly on the server. SQLmap also supports a dictionary based attack where it immediately tries to recognize and decrypt the found hashed passwords.

5 ANALYSIS

CDNetworks, a company that operates globally by providing content network infrastructure (CDN) services and security services through cloud, published a report in November 2020 on the state of web security for the first half of 2020 [19]. The report states that attacks on web applications have increased by more than 800% than in the same period in 2019. The main reason for the sudden jump is considered to be the current COVID pandemic, which had a serious impact on the rise of cyber-attacks and it seems logical on the one hand. This global phenomenon has shut down the world population and forced us to use the internet for most of life's socio-economic activities [20]. Comparison of blocked attacks on web application for the first half of 2019 and 2020 can be seen on Fig. 5.

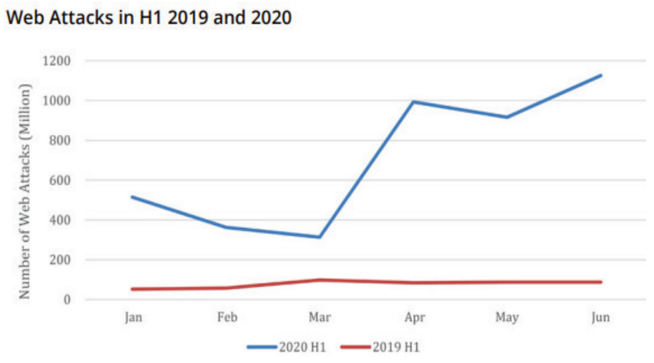


Figure 5 Comparison of blocked attacks on web applications for the first half of 2019 and 2020 [19]

In the first half of 2020, SQL injection attacks made up 16% of all web applications attacks. It indicates that even after twenty years, this vulnerability is still one of the most used vectors of attack. Fig. 6 shows types of blocked web application attacks [19].

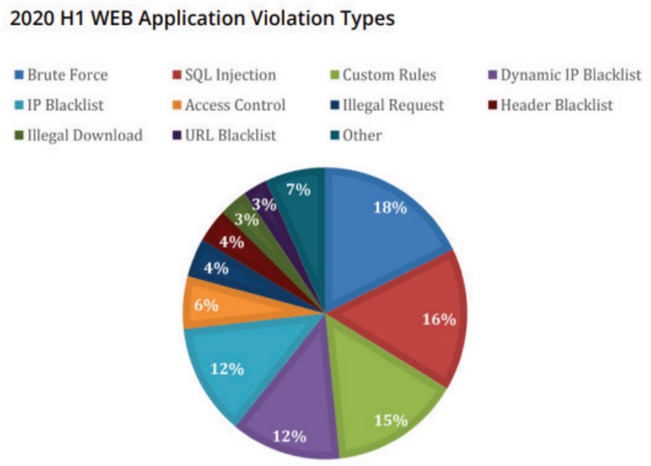


Figure 6 Types of blocked web applications attacks [19]

Akamai Technologies, a company that deals with CDN content, released similar research results in 2020. During the 24-month observation period, looking at all vertical periods SQL injection attacks made up more than 72% of all web applications attacks. The attack rate is halved to 36% if one looks at attacks targeting only the financial sector [21]. Fig. 7 shows overview of web applications attacks during the 24 months period (December 2017 - November 2019).

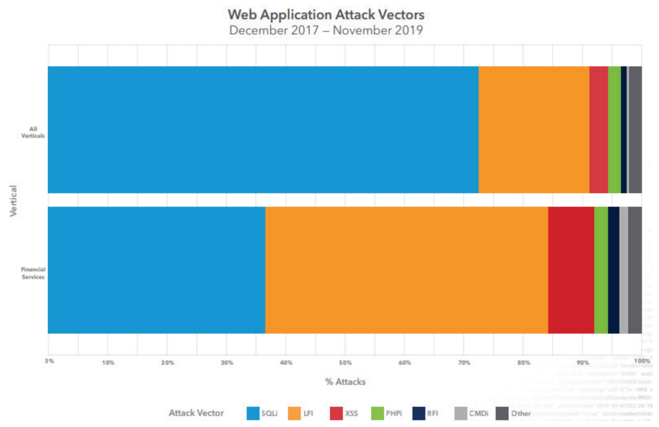


Figure 7 Overview of web applications attacks during the 24 months period, from December 2017 to November 2019 [21]

5.1 Prevention

SQL injection is an old vulnerability and usually taken into account when the application is still in a state of development. In most cases the application dragged the vulnerability out of the development phase. Developers often do not consider security aspects of the applications they work on because they place more attention on the functionalities itself. Such practices not only lead to vulnerabilities but also create additional problems that are mostly solved in the future by software patching.

Various authors suggest using more prevention techniques. Most frequent prevention techniques include using various detecting methods to validate user input as well as using type-safe SQL parameters [22]. Authors in paper [23] proposed an expectation-based prevention technique which address SQL injection attack while calculating the probability of the special characters in the attack dataset, while authors in paper [24] proposed randomization technique that modifies the user input data before the SQL query is executed at the database server.

There are also a few recommended prevention techniques. First of them is through prepared statements. When it comes to prepared statements the SQL code included in the query is predefined, which means that the database will distinguish between code and user input. If an attacker tries to enter a malicious SQL statement, the database will treat the received query as data and not as code. This approach will prevent the resulting query from turning into a malicious one. When preparing a statement, a few rules should be applied:

- prepares SQL statement with empty values are reserved places where the question mark represents a value
- associating a variable with reserved places, specifying each variable along with the data type. Allowed data types are integer, double, string and blob

Second important prevention technique are stored procedures where the principle is somewhat similar to that of prepared statements. The difference is in locating the SQL code, while the queries are prepared in the application logical layer, the stored procedures are located in the data layer called by the application. These are a set of instructions that perform a predetermined task and their advantage is that it can centralize the data access logic in one place and change access permissions. It boils down to the least privilege principle, which means that if an attacker compromises a stored procedure that contains SQL statements, damage would be limited due to stricter access to the data. Depending on the application in cases where SQL code becomes a necessary part of user input, it is important to make a list of allowed SQL statements. Creating a list of only the most necessary expressions is used to filter invalid statements so that they do not end up in the sent query.

Other researchers results also showed that a significant amount of attacks were blocked using prevention technique which only checks that there are no special characters in an input string. As described before, special characters could potentially change the complete query logic. PHP and MySQL have built-in functions, such as *mysqli_real_escape_string*, which escapes special characters in a string for use in an SQL statement, taking into account the current charset of the connection.

When writing the code that accesses a database, a good practice is to predict possible anomalies at an early stage in order to prevent vulnerabilities instead of resolving through software patches. Nowadays there are various tools for analyzing programming code by providing guidance for best programming practices, adhering to all coding standards with a purpose of identifying potential security issues. The development process should include tests in various forms against attacks because it is important to reduce the risk of future consequences.

6 CONCLUSION

In this paper, using a vulnerable web application, several SQL injection attack techniques are explored. Despite being an older and well-known vulnerability, it is still present, emphasizing the importance of implementing preventive measures. Because of the Internet, we live in a golden age of collecting and storing vast amounts of data. Today's databases allow companies to analyze them for business improvements like profits or to better understand the preferences and needs of their customers. Their security has taken many forms because the consequences of a vulnerability can be catastrophic for any business.

Manual approach demonstrates the attack potential and how to test for vulnerabilities. It has been demonstrated that SQL injection attack is not designed to interact directly with applications, instead it is the application that, given the input received, prepares the SQL code to be sent to the database. The paper also presents the SQLmap software tool, which helps to check the vulnerability of the applications in an automated way that saves time during testing. Automated tools are not a substitute for manual testing, but they are usually sufficient to determine whether an application is vulnerable or not.

In terms of protection against SQL injection attacks, it is always about checking and applying the way how to control the input and output of information in the application. The attack is based on sending a malicious SQL query and as shown in the paper the principle is almost always the same. The input coming from the user should always be considered potentially malicious and only then effective security measures can be implemented.

7 REFERENCES

[1] Forristal, J. (1998). NT Web Technology Vulnerabilities. *Phrack Magazine.* Retrieved from http://phrack.org/issues/54/8.html

[2] Gates, B. (2002). Global Information Assurance Certification, Microsoft toward Trustworthy Computing. Retrieved from https://www.giac.org/paper/gsec/4243/pillars-trustworthy-computing-displayed-patch-management/106837

[3] Halfond, W. G. J. & Orso, A. (2006). Preventing SQL injection attacks using AMNESIA. *Proceeding of the 28th International Conference on Software Engineering – ICSE '06*. https://doi.org/10.1145/1134285.1134416

[4] OWASP, (2017). OWASP Top 10. Retrieved from https://owasp.org/www-project-top-ten/

[5] Trotta, D. (2009). Three indicted in largest U.S. identity theft scheme. Retrieved from https://www.reuters.com/article/us-crime-identity-idUSTRE57G4GC20090817

[6] O'Dell, J. (2009). RockYou Hacker: 30% of Sites Store Plain Text Passwords, RockYou breach. Retrieved from https://archive.nytimes.com/www.nytimes.com/external/readwriteweb/2009/12/16/16readwriteweb-rockyou-hacker-30-of-sites-store-plain-text-13200.htm

[7] TinKode & Ne0h, (2011). MySQL breach. Retrieved from https://pastebin.com/raw/BayvYdcP

[8] BBC, (2011). Sony investigating another hack. Retrieved from https://www.bbc.com/news/business-13636704

[9] Gross, D. (2012). Yahoo hacked 450,000 passwords posted online. Retrieved from https://edition.cnn.com/2012/07/12/tech/web/yahoo-users-hacked/index.html

[10] Vtech, (2015). Data Breach on VTech Learning Lodge (update). Retrieved from https://www.vtech.com/en/press_release/2015/data-breach-on-vtech-learning-lodge-update/

[11] Menn, J., (2016). U.S. election agency breached by hackers after November vote. Retrieved from https://www.reuters.com/article/us-election-hack-commission-idUSKBN1442VC

[12] Gorgia Tech, (2019). Unauthorized Access on Georgia Tech Network Exposes Information for 1.3 Million Individuals. Retrieved from https://www.news.gatech.edu/news/2019/04/02/unauthorized-access-georgia-tech-network-exposes-information-13-million-individuals

[13] Ping-Chen, X. (2011). SQL injection attack and guard technical research. *Procedia Engineering*, 15, 4131-4135. https://doi.org/10.1016/j.proeng.2011.08.775

[14] Gudipati, V. K., Venna, T., Subburaj, S., & Abuzaghleh, O. (2016). Advanced automated SQL injection attacks and defensive mechanisms. *2016 Annual Connecticut Conference on Industrial Electronics, Technology & Automation (CT-IETA)*. https://doi.org/10.1109/ct-ieta.2016.7868248

[15] National Security Agency, 2014. Defending against the Exploitation of SQL Vulnerabilities to compromise a Network. Retrieved from https:www.iad.gov/iad/library/ia-guidance/tech-briefs/defending-against-theexploitation-of-sqlvulnerabilities-to.cfm

[16] Kamtuo, K. & Soomlek, C. (2016). Machine Learning for SQL injection prevention on server-side scripting. *2016 International Computer Science and Engineering Conference (ICSEC)*. https://doi.org/10.1109/ICSEC.2016.7859950

[17] Yunus, M. A. M., Brohan, M. Z., Nawi, N. M., & Surin, E. S. M. (2018). Review of SQL Injection: Problems and Prevention. *International Journal on Informatics Visualization*, 2 (3-2), pp. 215-219. https://doi.org/10.30630/joiv.2.3-2.144

[18] GitHub, Miroslav Stampar, (2021). SQLmap History. Retrieved from https://github-wiki-see.page/m/sqlmapproject/sqlmap/wiki/History

[19] CDNetworks, (2020). State of Web Security H1 2020. Retrieved from https://www.cdnetworks.com/wp-content/uploads/2020/11/CDNetworks-State-Of-web-Security-H1-2020.pdf

[20] Communications of the Acm, (2021). A Year in Lockdown: How the Waves of COVID-19 Impact Internet Traffic. Retrieved from https://cacm.acm.org/magazines/2021/7/253468-a-year-in-lockdown/fulltext

[21] Akamai, (2020). State of the Internet, Financial Services Hostile Takeover Attempts. Retrieved from https://www.akamai.com/content/dam/site/en/documents/state-of-the-internet/soti-security-financial-services-hostile-takeover-attempts-report-2020.pdf

[22] Li Qian, Zhenyuan Zhu, Jun Hu, & Shuying Liu. (2015). Research of SQL injection attack and prevention technology. *2015 International Conference on Estimation, Detection and Information Fusion (ICEDIF)*. https://doi.org/10.1109/icedif.2015.7280212

[23] Xiao, L., Matsumoto, S., Ishikawa, T., & Sakurai, K. (2016). SQL Injection Attack Detection Method Using Expectation Criterion. *2016 Fourth International Symposium on Computing and Networking (CANDAR)*. https://doi.org/10.1109/candar.2016.0116

[24] Mehta, P., Sharda, J., & Das, M. L. (2015). SQLshield: Preventing SQL Injection Attacks by Modifying User Input Data. *Lecture Notes in Computer Science*, 192-206. https://doi.org/10.1007/978-3-319-26961-0_12

Authors' contacts:

Nenad Bedeković, MSc, Student
University North, University Center Varaždin,
Jurja Križanića 31b, 42 000 Varaždin, Croatia
nenad.bedekovic@unin.hr

Ladislav Havaš, Assistant Professor
University North, University Center Varaždin,
Jurja Križanića 31b, 42 000 Varaždin, Croatia
ladislav.havas@unin.hr

Tomislav Horvat, PhD
(Corresponding author)
University North, University Center Varaždin,
Jurja Križanića 31b, 42 000 Varaždin, Croatia
tomislav.horvat@unin.hr

Dražen Crčić, mag. ing. el. eng.
University North, University Center Varaždin,
Jurja Križanića 31b, 42 000 Varaždin, Croatia
drazen.crcic@unin.hr