Asian Journal of Research in Computer Science

14(4): 198-211, 2022; Article no.AJRCOS.94574
ISSN: 2581-8260

A Study of Online Database Servers: The Case of SQL - Injection, How Evil that could be?

George S. Oreku [a,b*]

[a] Department of Information and Communication Technologies, Open University of Tanzania (OUT), P.O. Box 23409, Dar es Salaam, Tanzania.
[b] Faculty of Science and Forestry, School of Computing, University of Eastern Finland, P.O. Box 111, FIN-80101 Joensuu, Finland.

Author's contribution

The sole author designed, analyzed, interpreted and prepared the manuscript.

Article Information

DOI: 10.9734/AJRCOS/2022/v14i4304

Open Peer Review History:
This journal follows the Advanced Open Peer Review policy. Identity of the Reviewers, Editor(s) and additional Reviewers, peer review comments, different versions of the manuscript, comments of the editors, etc are available here: https://www.sdiarticle5.com/review-history/94574

Short Research Article

Received 10 October 2022
Accepted 13 December 2022
Published 17 December 2022

ABSTRACT

SQL injection attack is one of the most serious security vulnerabilities in many Databases Managements systems. Most of these vulnerabilities are caused by lack of input validation and SQL parameters used particularity at this time of technology revolution. The results of a SQL injection attack (SQLIA) are unpleasant because the attacker could wipe the entire contents of the victim's database or shut it down. As such, SQLIA can be used as important weapons in cyber warfare. As an attempt of breaching of number of application data bases systems two SQL injection techniques were used to successful locating vulnerable points during this research which are Blind Text Injection Differential and Error based Exploitation. The motivations behind were to find out where the databases systems are most likely to face an attack and proactively shore up those weaknesses before exploitation by hackers. The success of both techniques is a result of poor web server (online database server) design especially in the selection of error messages (or answers) they display to website users if something goes wrong. The approach through examination of error messages (error codes) did enable to precisely know the backend Database Management System (DBMS) type and version and what exactly are parameters (variables) which can allow "illegally" injecting codes (a SQL query). Additionally, the paper presents SQLIA cases and their impact in Tanzania cyber space as well as it suggests the possible mitigation ways while reflecting the collected data with what currently existing in cyberworld as far as SQL injection attack is concern to present the reality.

**Corresponding author: E-mail: george.oreku@gmail.com;*

Keywords: First SQLIA; second code injection; third cyber warfare; fourth SQLMap; fifth security; sixth database.

1. INTRODUCTION

SQL injection is a code injection technique used to attack data-driven applications in which nefarious SQL statements are inserted into an entry field for execution [1]. The nefarious SQL statements (SQL queries) can be instructed by an attacker to dump the database contents, steal credentials, or modify the entire database. According to Imperva [2] which analyzed 297,954 attacks and 22,850,023 alerts in year 2015, SQLIA rose by 150% and it was leading among all kind of web application Attacks. This is in favour of data presented in the report “state of the Internet 2020” from the artticle in [3] justifying the validy of the study.

SQL Injection vulnerabilities are a result of poor or bad programming and occur when developers combine hard-coded strings with user-input to create dynamic queries [4]. Precautions should be taken in order to prevent externally supplied user input to modify the query string such that it performs unintended actions including gaining unauthorized read or write access to the data stored in the database.

With today on going concerns online applications security is unarguably the most serious concern for Web applications, to which SQL injection (SQLi) attack is one of the most devastating attacks. Automatically testing SQLi vulnerabilities is of ultimate importance, yet is unfortunately far from trivial to implement. This is because of the existence of a huge or potentially infinite number of variants and semantic possibilities of SQL leading to AQLi attacks on various web applications [5].

According to National Security Agency (NSA), SQL injection is the most typically ways used by hackers, even the famous database organization MYSQL was hacked by these techniques on electronic records [6,7]. Presently a day, the more the quantity of vulnerabilities will be diminished, the more the quantity of threats become to increment [8]. Structured Query Language Injection Attack (SQLIA) is one of the incredible dangers of web applications threats where by lack of input validation vulnerabilities where cause to SQL injection attack on web.

Sqlmap is an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers [9]. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections. In respect to that this study did follow the basics procedures of the tool while applying different techniquics and approach.

In the other hand in addressing the problem many researchers have effectively analyzed black box scanner in vulnerability detection. Furthermore, they find out its constraints by repeatedly testing numerous black-box scanners against a wide range of vulnerable applications. A lot of work in this direction is focused on fuzzing. It deals with testing (semi)-random values [10]. Another important method to prevent web vulnerabilities is data mining and machine learning. These learning methods with a variety of web applications are considered a unique approach. However, it can also be used in source code to identify vulnerabilities [11].

With the same efforts Valeur and colleagues [12] propose the use of an Intrusion Detection System (IDS) to detect SQLIAs. Their IDS system is based on a machine learning technique that is trained using a set of typical application queries. The technique builds models of the typical queries and then monitors the application at runtime to identify queries that do not match the model. However, two recent approaches, SQL DOM [13] and Safe Query Objects [14], use encapsulation of database queries to provide a safe and reliable way to access databases. These techniques offer an effective way to avoid the SQLIA problem by changing the query-building process from an unregulated one that uses string concatenation to a systematic one that uses a type- checked API.

In [15] & [1], existing detection and prevention techniques against SQL injection attacks is presented and analyzed. The authors presented one Technique for Detection and Prevention of SQL Injection Attack using ASCII Based String Matching. Our approach has taken different turn in addressing SQL injection problem by using three different scenarios for testing security systems. The approach applied penetration

testing technique trying to find out which is the best solution for protecting sensitive data within the government network to a number of agencies were the exercise was undertaken. Specifically, penetration testing has been conducted using SQL Injection attacks [16]. As a target for the conducting such attacks, a test web server was set up in the government network.

In our approach the study first focused in exploiting SQLI vulnerabilities to identify vulnerable parameters where by identifications of parameters using Google dorks and open-source penetration testing tools was applied. However, Vega was used to identify the vulnerable parameters, SQLMap for a proof of concept that is to verify if the databases under study is actually exploitable for the given parameters.

Having the best efficiency and having the most secure applications are always in contradiction with each other. We cannot have high security without any cost [17]. Injecting a web application is the synonym of having access to the data stored in the database. The data sometimes could be confidential and of high value like the financial secret of a bank or list of financial transactions or secret information of some kind of information system, etc. An unauthorized access to this data by a crafted user can threat their confidentiality, integrity, and authority. As a result, the system could bear heavy loss in giving proper services to its users or it may face complete destruction and this is the foci or conducting this research.

2. EXPLOITATION METHODS AND TECHNIQUES USED

Although this vulnerability has been known for more than 20 years, injections still rank third in the OWASP Top 10 for web vulnerabilities. In 2022, 1162 SQL injection vulnerabilities were accepted as CVE. [18,19] In an SQL injection, the attacker inserts an SQL statement into an exchange between a client and a database server [2]. SQL (Structured Query Language) is used to represent queries to database management systems (DBMS). The maliciously injected SQL statement is designed to extract or modify data from the database server. A successful injection can lead to authentication and evasion as well as database modifications by inserting, modifying and/or deleting data, resulting in data loss and/or destruction of the entire database.

SQL Injection (SQLi) is the most common attack vector responsible for over 50% of all web application attacks today. The impact of SQL injection (SQLIA) attacks is negative because they can allow an attacker to delete or close the entire contents of a victim's database [20]. Financial services are one of the safest industries in the world, yet the number of attacks against this industry continues to rise. But how far and how? In 2022, Akamai Security Research observed a phenomenal 3.5x growth in web application and API attacks on FinServ and the alarming rate at which attackers are exploiting zero-day vulnerabilities that pose significant risks to organizations EndServ. According to Akamai's 2020 State of the Internet Report, SQLi accounts for nearly 80% of all retail, travel and hospitality web application attacks between 2018 and 2020 [21].

In this paper, we will test security systems which are implemented in government network with Firewalls and IPS. A study of 228 web servers was examined in the year 2016 in Tanzania which indicated that 102 (≈45%) servers can be SQL injection exploited. The vulnerability observed could allow data exfiltration such that it was possible to list tables and dump user accounts, emails and passwords. Some of the tables were found to contain administrative user accounts and easily crack able passwords enabling an attacker to take total control of the victim server. According to Akamai “State of the Internet” report 2020, SQLi accounts for almost 80% of all attacks against retail, travel, and hospitality web apps between 2018 and 2020 [3].

2.1 Sample Selection

A total of 220 websites (web servers for that matter) that provides web services to Tanzanians were randomly chosen and cross examined to check if are SQLI exploitable. Although the choice was random, certain criteria were considered;

- Websites that provide (have links that provide) email services
- Websites owned by government institutions, Universities, Regulators and Tanzania Revenue Authority.
- Websites owned by banks/financial institutions
- Websites ran by universities
- Website of the Communications Regulator
- Website of the Revenue Authority

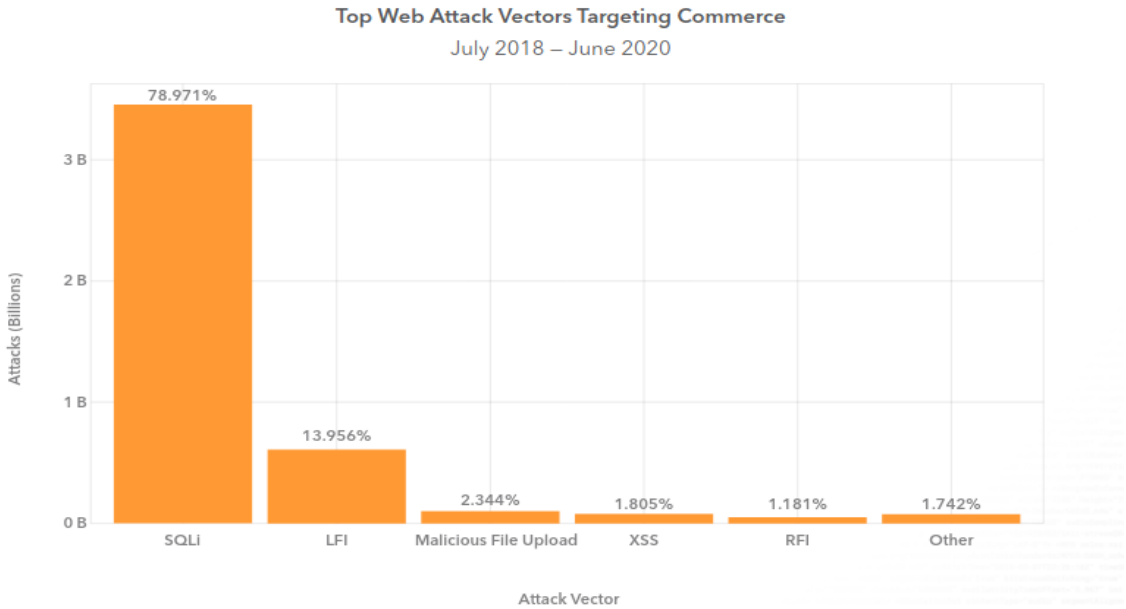


Fig. 1. Top Web attack vectors targeting commerce (Source 24)

2.2 SQL Injection Techniques

SQL Injection exploitation is all about examining error messages which web browsers (clients) get from database servers. Through examination of error messages (error codes) we can precisely be able to know the backend DBMS type and version. Also, we can also be able to know what parameters (variables) can allow us to "illegally" inject codes (a SQL query).

There are some cases where programmers are a bit careful not to reveal backend systems through "traditional" error codes, in such situations normally blind injection techniques is used to test SQLIA existence. In blind injections a number of questions are probed to the database (web server) application with TRUE or FALSE responses and by examining the answers, detailed information about the DBMS and vulnerability parameters can be identified. Several methods and tools can be used to detect presence of SQL Injection points (parameters) and perform exploitation; one can use manual techniques [22], Google dorks [23] or automated tools like SQLMap.

Google dorks and SQLMap were used during this study. Depending on how a database is designed, each database can be exploited with different technique(s). During this study at least eight exploitation techniques were observed;

- Time Delay Exploitation
- Blind Arithmetic Evaluation Differential
- Out of Band Exploitation
- Union Exploitation
- Boolean Exploitation
- Blind Text Injection Differential
- Stored Procedure
- Error based Exploitation

2.3 Injection Exploitation Method

There are two methods the browser use to send information to the web server; these are GET and POST. The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the? Character. The POST method transfers information via HTTP headers. Encoding in POST is done like in the GET method, what is different is that, with POST, the encoded information is put into a header called QUERY_STRING.

Because data in the GET method is made up by part of the URL, this method is less secure. Parameters usually remain in browser history. In the POST method, data are transferred through HTTP header; therefore security of this method depends on the HTTP protocol. By using Secure HTTP, that is HTTPS, web servers can at least guarantee the information is secure.

SQLIA are code injections attacks. Knowing which method(s) is used when a targeted server exchanges data with its clients is important as it helps an attacker to decide a method to inject her malicious codes. Experience shows that

exploiting SQL Injection parameters via GET Method is easier than in POST Method. Part of this study was to examine which method (GET or POST) is mostly employed in Tanzania websites.

3. A PERFORMANCE EVALUTION: DUMPING VULNERABLE DATABASES WITH SQLMAP

A proof of concept in illustrating SQLIA is to be able to dump database contents of a vulnerable server. To do that, SQLMap was used in this study. If you are a hacker, SQLMap is one of powerful technique used in attacking a particular web server. If you are an ICT Admin, SQLMap is one of powerful and efficient tool to examine all possible weaknesses in database applications in order to mitigate SQL Injection attacks. SQLMap not only can identify vulnerabilities, it can perform the actual hack – exploitation. This paper will show simple steps which were applied on SQLMap to exploit a DBMS.

3.1 Vulnerable Parameter Identification

The first step in examining online databases for weaknesses is to identify vulnerable parameters, that is, injection points. Google dorks and website auditing tools that are included in Kali Linux were used to identify vulnerable parameters in the selected web servers. Screenshot 1 displays typical output of a vulnerable parameter.

Things to note down from that output are Resource and Parameter; The Resource is the URL (webpage) that contains an injection point. It is through this resource the database will (might) be exploited. Second is the Parameter; this is actually the vulnerable point – it is our place to inject the "killer poison". These two (Resource and Parameter) will be inserted to the SQLMap engine for exploitation.

The detection method helps the choice of a syntax to be used in SQLMap, we know there are two methods; GET and POST. These have different syntaxes. The syntax for exploitation using GET method is summarized in Screenshot 2.

The above test was an exploitation tested in one of university based in Dar es Salaam. It is a university that offers online training. Unfortunately, its database is still vulnerable until when this paper was being compiled. As it can be seen in coming sections, database contents were successfully dumped. Hitting Enter key in the above command shown in Screenshot 2, SQLMap should start execution and if successful, the results presented in Screenshot 3 would appear.

Screenshot 1. Output for the identification of an SQL vulnerable parameter in one of the tested web server. The URL (webpage) shown in the Resource above, is what contains an SQL injection point.

Screenshot 2. Syntax for using SQLMap to exploit a web server using the GET method. The vulnerable parameter "key" in this test was given a value 1. In GET method, vulnerable parameters can be given any random value

After SQLMap finishes the launching of the attack, it displays a nice summery telling even the payload it used for exploitation. Screenshot 4 illustrates more.

3.2 Database Manipulation

After successfully performance what is instructed in the previous step, we are sure that the server is SQL Injectable. We have indeed identified injection points. The study didn't opt to inject malware, rather opted to identify existing databases in that server. One server can have more than one database.

The command used by SQLMap to identify (list) all the databases is --dbs this command was typed at the end of the command shown in Screenshot 2. For the test done, the output should take the look shown in Screenshot 5.

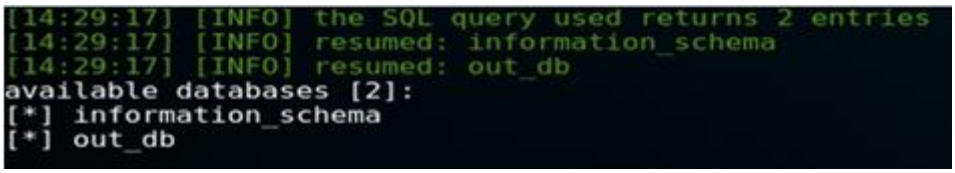
3.3 Listing Tables

The step that follows is to see what tables and columns are contained in the databases. SQLMap has secrets for that see Screenshot 6.

Screenshot 8 shows a command used to access a table in database. In this example we try to access a table called users.

Screenshot 3. A display of how SQL injection with GET method is successful. SQLMap has managed also to tell present us the DBMS (for this case is MySQL)

Screenshot 4. We learn from this output that the server can be exploited using two techniques. Error Based and UNION query. Also, SQLMap tells us the total number of injections (where we can insert our malicious code) points. For this resource (refer Screenshot 1), there were 47 different injection point.



Screenshot 5. List of databases found in the web server under observation. The server had two databases; information schema and out db

Screenshot 6. A command to list down tables in all databases. If you specify a particular database, usually using -- D option followed by database name, tables for that database will be displayed. If you don't specify a database, then all tables in all databases found in the server will be shown. This takes a considerable amount of time

Screenshot 7. Application of the --tables switch to a vulnerable server. The command extracts and displays all tables. Tables which attract hackers are those having names like users, usernames, or something similar. These tables contain usernames and password (or password hashes) for users with various privileges

Screenshot 8. How SQLMap is used to access specific tables within a database

If we can dump database contents, what else do we want? We have successfully hacked a database, using just three tools; the first which helped identify vulnerable parameter, The second (SQLMap) which helped perform injections and Third are tools which help cracking password hashes.

3.4 SQLMap Injection with POST Method

All the above examples were for the GET method. Let's see how it was done with POST method. Using POST, is a little bit tricky; we use the data option to indicate that the vulnerable parameter is a POST request. The command takes the look shown in Screenshot 10.

Screenshot 9. Dumped outputs from the users table in the database. Names are intentionally striped off for security reasons. The dumped table had password hashes of which were later cracked using other password crackers with the help of 2GB rainbow table

Screenshot 10. SQL injection using POST method. The above server which belongs to one government institution was (is, until the time of this publication) found vulnerable to SQL injections

What is important when doing POST SQL Injection with SQLMap is to make sure that you supply the vulnerable parameter with necessary data as identified during section 2.4.1 – vulnerable parameter identification. Depending on the choice of tool you use, certain tools give a very nice presentation of POST data vulnerable parameter.

Steps we followed during the GET method for database manipulations are applicable in POST as well. The end point is usually the same, see Screenshot 11.

4. RESULTS: SQL INJECTIONS THE CASE TANZANIA

4.1 Presence of SQLIA

From the study which was conducted most online database servers in Tanzanian cyber space seemed to be vulnerable to SQL Injection. The Screenshot in cooperated in this paper from number 1-11 displays results from selected servers during experiment. Fig. 5 also summarizes the presence of meta tags that reveal sensitive information like platform details and configuration characteristics in a histogram diagram which was the results of misconstruction of web server leading to code Injection. This study shows that, of the 228 web servers examined in year 2016 a total of 102 servers can be SQL injection exploited, see Fig. 1.

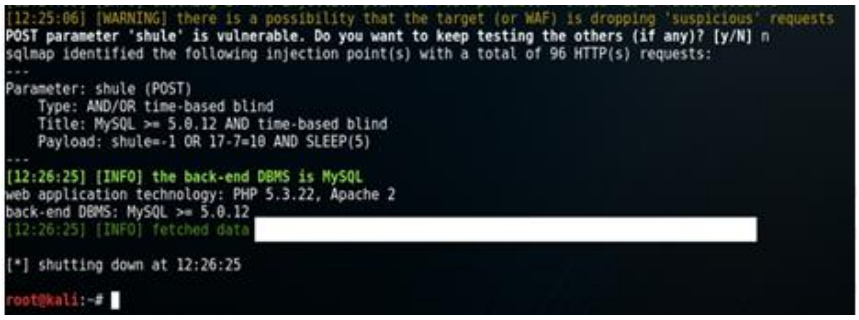
4.2 GET vs POST SQLI Exploitation Methods

From research overview it is observed that most security Admins understand the fact that SQL exploiting web servers using POST method is difficult than the GET method, because during this study it was observed that 56% of all SQL Injection cases were found exploitable by POST Method and the rest by the GET Method, see Fig. 2.

4.3 SQL Exploitation Techniques

As said earlier, depending on how a database is designed, each database system can be exploited with different technique(s). There were a total of 390 cases of exploitation techniques found in 102 servers. These techniques are as summarized in section 2.2 of this report. Fig. 3 displays those cases and their number of occurrences.

Experience is using SQLMap during this study, shows that it is quite easy to exploit SQL injection using Error Based, Boolean and Union techniques. These three techniques accounted for 41.5% of all SQL exploitation.



Screenshot 11. Output from one server which was SQL injectable using POST method. This government owned server had a total of 96 injection points which could be used for injecting any malicious code, including malware

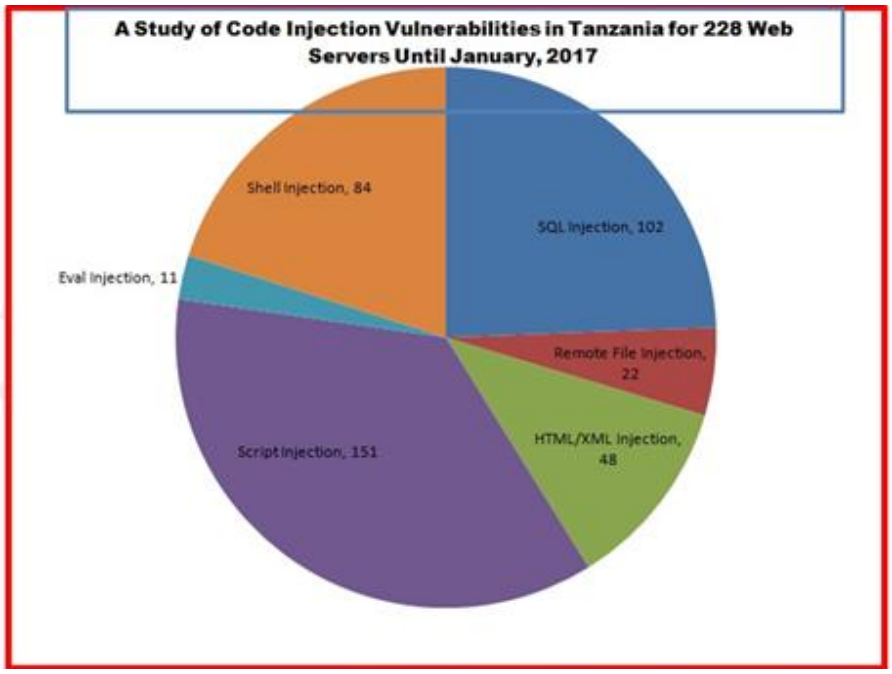


Fig. 2. Code Injections vulnerabilities as were observed in Tanzania cyber space between Januarys to December in 2016. Script Injections accounted for most injections followed by SQL. The results show that most web servers are susceptible to cyber espionage related attacks; they can be injected with malware

Note: Some of database servers were found to be exploitable using more than one SQL Injection techniques. One University which offers distant and online training had six of the above eight techniques. It was indeed possible to damp most of database contents of the server.

4.4 Web Server Misconfigurations

As explained in part of of the introduction section of this paper, SQL injection vulnerabilities are a result of poor web server design. Website auditing tools were used to identify web server that were designed poorly and existence of injection vulnerabilities in such servers. The screenshot number 12 below potrayes that vividly.

Servers were observed to have the following misconfigurations which might lead to code injection vulnerabilities:

- Meta Tags Misconfiguration
- Unspecified Character Set
- HTTP Error Disclosure
- Unsecure Cookies
- Unsafe AJAX Code
- Weaker Encryption Algorithm

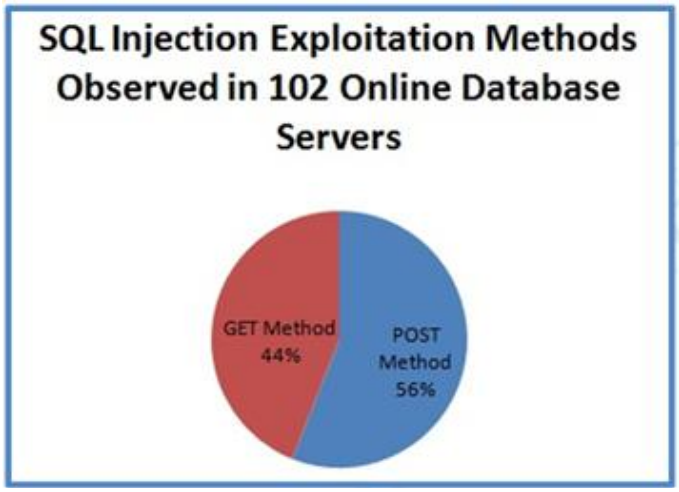


Fig. 3. Out of the 102 servers, 44% were found exploitable using GET Method. Most of these servers were those ran and operated by universities. The rest 56% which are exploitable using POST, were from banks and government institutions

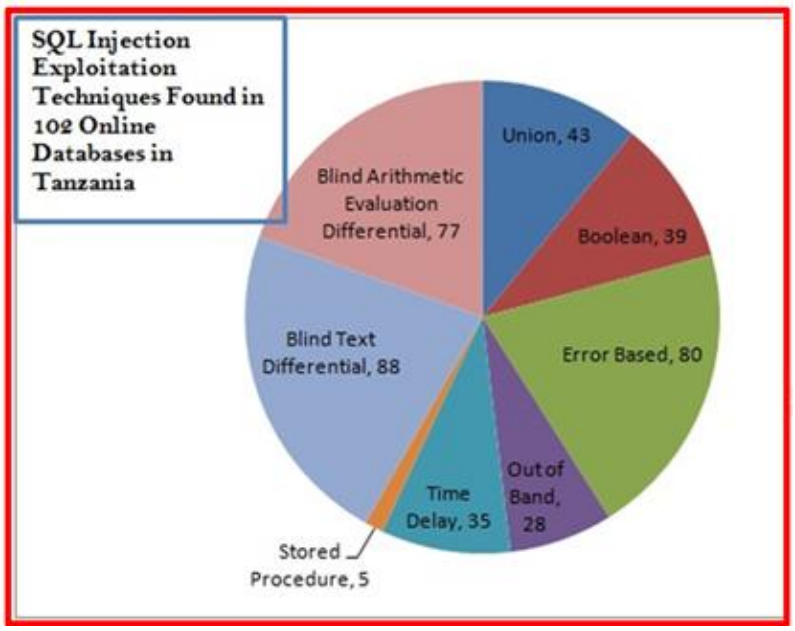


Fig. 4. Blind Text and Blind Arithmetic Differential accounts for 42.3% of all cases of SQL Injection exploitation techniques in Tanzania cyberspace. Out of Band Technique was the least observed to only 0.07% of all cases

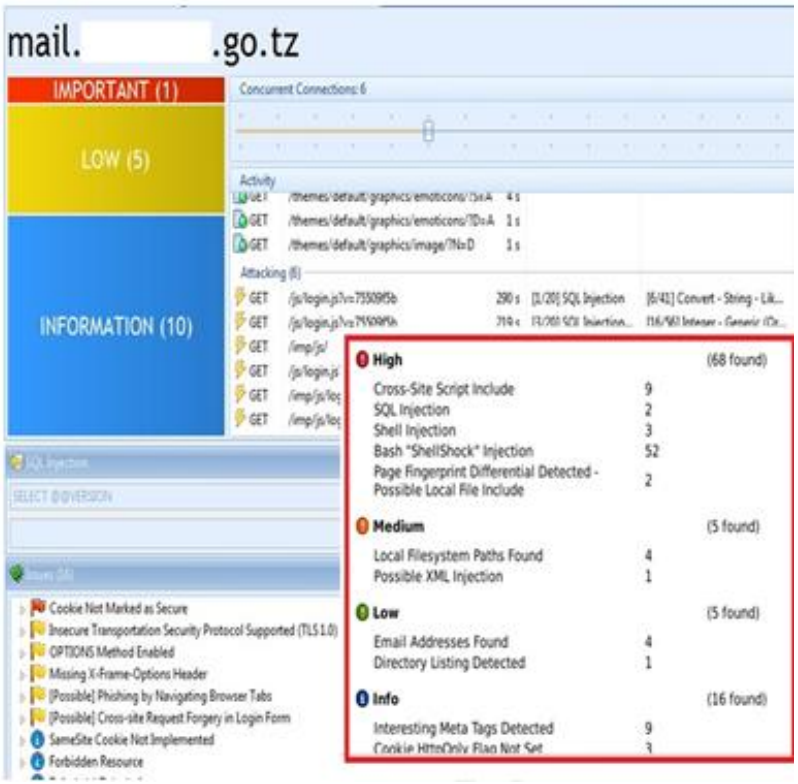
Unspecified Character Set and Meta Tag misconfiguration account to most problems (58.58%) that almost every server was observed to have. Fig. 4 summarizes poor configuration findings observed in 220 web servers surveyed in year 2016.

5. DISCUSSION

The research underteken was to discover relationships among what was done earlier and to allow the prediction of future events from present knowledge in reagard to SQL injection attacks. The tests performed on government network, have made it clear that Penetration testing is of paramount importance on todays efforts in fighting SQL injection attacks. The advantage of using this technique is that it uses the same tools as hackers do, but in a legal way. Another advantage of this technique is the final report that is produced, which provides

recommendations on avoiding security holes which can be identified on a system.

Based on the findings as shown in Fig. 1, SQL Injection accounted to 24.4% of all code injections in year 2016. This means, nearly a quarter of all web servers in Tanzania can be injected with malicious codes (including malware) using SQL injection techniques. Also, sensitive information like credentials, emails and classified documents can also be stolen by hackers who can make use of this vulnerability. This study process lays the foundation for more conclusive data collection and analysis to set the foundation of futher research on the topic.



Screenshot 12. Observations aimed at seeing server misconfigurations which lead to exploitable vulnerabilities. Observations like this one were done using open source tools. The above display is a vulnerability display as observed from one government mail server in December, 2016. Owners of the server were notified in to fix all issues

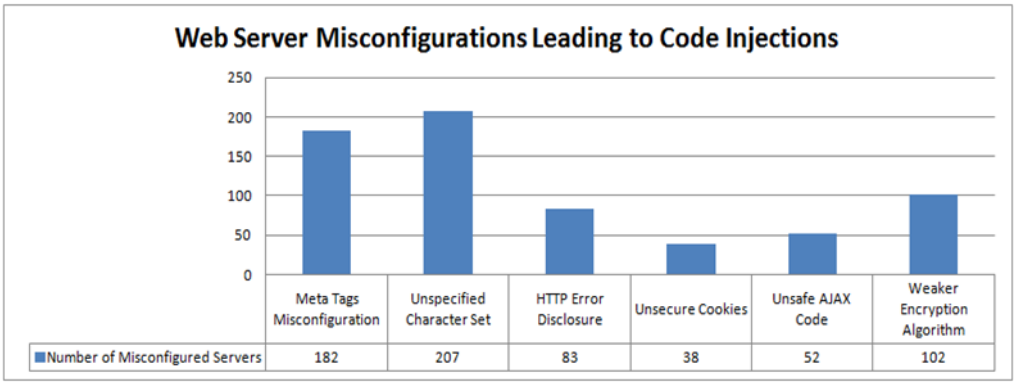


Fig. 5. Cases of website Misconfigurations observed in webservers in a study conducted in year 2016 for 220 we servers in Tanzania. One of "chilling" findings is the presence of meta tags that reveal sensitive information like platform details and configuration characteristics

However, to be able to fully exploit a server using SQL injection techniques and steal sensitive data, an attacker must be capable of dumping and cracking the dumped data. Success in cracking the data depends on how strong is the encryption algorithm used to cipher the data. Unfortunately, there were observed to be 102 cases of weaker encryption algorithms used in the 220 web servers examined. It was possible to crack the dumped data using a 1GB of rainbow tables in less than 15 minutes, which illustrates how easy it was.

The results shows that, from SQLIA an attacker can:

- Steal sensitive information by dumping database contents.
- Grab web server access credentials by cracking usernames and passwords.

Getting these credentials, an attack is practically having full control of the web server and can therefore modify data, inject malware (spyware), launch denial of service attacks and any other cybercrime related action she wishes.

A single SQL injection can impose serious vulnerabilities in online database servers. The vulnerability can be exploited by remote attackers to gain unauthorized read or write access to the victim database. Exploitation of SQL injection vulnerabilities can also allow for attacks against the logic of the installed application. Attackers may be able to obtain unauthorized access to the server hosting the database. Indeed the injection can be very evil.

6. SUGGESTIONS FOR MITIGATIONS

While there are many approaches towards mitigating SQL injection attacks [7,24,23], the findings suggests the following to be done to prevent an SQLIA;

- Use parameterized statements. This is probably the best defense against SQL injection vulnerabilities.
- Wherever possible, do not create dynamic SQL queries or SQL queries with string concatenation.
- Web developers must make sure that they do input Sanitization. Variables of string types should be filtered for escape characters, and numeric types should be checked to ensure that they are valid.
- Use of stored procedures can simplify complex queries and allow for tighter access control settings.
- Configuring database access controls can limit the impact of exploited vulnerabilities. This is a mitigating strategy that can be employed in environments where the code is not modifiable.
- Use of Object-relational mapping eliminates the need for SQL.

7. CONCLUSION

Web servers in Tanzania seem to be vulnerable to SQLIA. The vulnerabilities found from this research are exploitable and could allow adversaries steal sensitive information and take full control of the victim servers. SQLIA detected were a result of poor website design in terms of security considerations. Web servers were found to include sanitized inputs, display traditional error messages that reveal sensitive server information, creates dynamic queries and use weaker encryption algorithms.

It is strongly advised that web server security administrators should undergo proper security training on ethical hacking courses and cyber security. The course should be organized in a manner that suites Tanzanian environment; cyber space weaknesses in Tanzania might slightly differ from what are observed in other nations. Trainers must have full backgrounds knowledge of the way the internet is accessed and used.

However appropriate new grant funding and research still needed to the growing representation of underrepresented communities in the cybersecurity field particularyly on web and database attacks. SQL injections are still one of the most exploited security vulnerabilities and, as such, exist. SAST and DAST are two security testing approaches for preventing SQL injections. Both methods have a number of drawbacks. There is, however, hope: recent research has shown that fuzzing can be an especially effective method for detecting SQL injections. The Code Intelligence platform detects injections with high accuracy and almost no false positives.

COMPETING INTERESTS

Author has declared that no competing interests exist.

REFERENCES

1. Wikipedia. SQL Injection; 2017.
   Available:https://en.wikipedia.org/wiki/SQL_injection May, 2017
2. Imperva. Web Application Attack Report; 2015.
   Available:https://www.imperva.com/download.asp?id=509 April, 2016.
3. Enemy at the Gates: Analyzing Attacks on Financial Services
   Available:https://www.akamai.com/lp/soti/enemy-at-the-gates-analyzing-attacks-on-financial-services
   Access on 12, 08, 2022
4. Indrani Balasundram, Ramaraj E. Prevention of SQL injection attacks by using service oriented authentication technique. International Journal of Modeling and Optimization. 2013;3(3):286-S385.
5. Tang P, Qiu W, Huang Z, et al. SQL injection behavior mining based deep learning. In: International Conference on Advanced Data Mining and Applications. Springer, Cham, Nanjing, 2018:445–454. Google Scholar
6. Zhang L, Tan C, Yu F. "An improved rainbow table attack for long passwords." Procedia Computer Science. 2017;107:47–52.
7. Deniz Gurkan, Fatima Merchant. "Interoperable medical instrument networking and access system with security considerations for critical care". Journal of Healthcare Engineering, 2010;1(4):637- 654.
8. Zar Chi Su Su Hlaing, Myo Khaing. A detection and prevention technique on SQL injection attacks". 2020 IEEE Conference on Computer Applications (ICCA), IEEE Xplore; 2020.
9. SQLmap-Automatic SQL injection and database etakeover tool.
   Available:http://SQLmap.org
10. Duchene F, Rawat S, Richier JL, Groz R. KameleonFuzz: Evolutionary fuzzing for black-box XSS detection. In Proceedings of the 4th ACM conference on Data and application security and privacy. 2014:37-48. ACM
11. Medeiros I, Neves NF, Correia M. Automatic detection and correction of web application vulnerabilities using data mining to predict false positives. In Proceedings of the 23rd International Conference on World Wide Web. 2014:63-74. ACM.
12. Tyrone Grandison, Evimaria Terzi. Intrusion Detection Technology; 2007. DOI**:** https://doi.org/10.1007/978-0-387-39940-9_209
13. McClure RA, Kruger IH. SQL DOM: Compile time checking of dynamic SQL statements. 2005m:88-96.
14. William R. Cook, Siddhartha Rai. Safe query objects: Statically-typed objects as remotely-executable queries, Conference: 27th International Conference on Software Engineering (ICSE 2005). 2005:15-21. St. Louis, Missouri, USA
15. Balasundarama I, Ramaraj E. "An efficient technique for detection and prevention of SQL injection attack using ASCII based string matching". International Conference on Communication Technology and System Design. 2011;30(2012):183–190.
16. Manish Kumar, Indu L. Detection and prevention of SQL injection attack. International Journal of Computer Science and Information Technologies. 2014;5(1):374-377.
17. Raniah Alsahafi. SQL injection attacks: Detection and prevention techniques. International Journal of Scientific & Technology Research 2019;8(01).
18. SingCERT's Security Bulletin.
    Available:https://www.csa.gov.sg/singcert/Alerts/sb-2022-018
    Access on 12, 08, 2022
19. Khanna S, Verma AK. Classification of SQL injection attacks using fuzzy tainting. In: Sa P, Sahoo M, Murugappan M, Wu Y, Majhi B. (eds). Progress in Intelligent Computing Techniques: Theory, Practice, and Applications. Advances in Intelligent Systems and Computing, Springer, Singapore. 2018;518.
    Available:https://doi.org/10.1007/978-981-10-3373-5_46
20. Asha NM, Varun Kumar, Vaidhyanathan G. Preventing SQL injection attacks. International Journal of Computer Applications. 2012;52(13):0975 – 8887.
21. Charles MJ, Pfleeger P, Pfleeger SL. Security in computing. 5th ed.; Springer: Berlin/Heidelberg, Germany; 2004.
22. Exploit Database. Full SQL injection tutorial (MySQL); 2016.

Available:https://www.exploit-db.com/papers/13045/. 2017.

23. Johnny Long. Google hacking for penetration testers: Explore the dark side of googling. Syngress Publishing; 2005.

24. Singh Kalsi T, Kaur N, et al. “Methods for preventing SQL injection attacks: A review”. International Journal of Adanced Engineering Technology.
E-ISSN: 0976-394.

Peer-review history:
The peer review history for this paper can be accessed here:
https://www.sdiarticle5.com/review-history/94574