Apoorva Singh
409428 AGH UST

CYBER SECURITY PROJECT:

A BROAD REVIEW OF SQL INJECTIONS AND THEIR APPLICATION OVER TIME

Cybersecurity Project Summer 2023

TABLE OF CONTENTS

Apoorva Singh
409428 AGH UST

INTRODUCTION

With the ongoing digitization, education, banking, hospitality, medical, communication and many other sectors are using web applications for providing services to customers, increasing their business reach and achieving their organizational goals. Further, there are 5.18 billion internet users worldwide, which amounts to 64.6 percent of the global population. (Statista, 2023). One can imagine that all 5.18 billon of those people have at least one password, at least one link to be exploitable, and all of them have consumer power, making online systems the warehouse of sensitive information.

Certainly, information technology has improved many aspects of human life. However, cybercrimes are its ugly manifestations which target individuals, organizations and society. Having existed for a considerable period of time, cyberattacks have evolved into different complex forms like phishing attack, viruses, malware, SQL injection, brute force, password attack and many others (Kumar, August 2022). WannaCry Ransomware, Nasa Cyber-attack, Estonia Cyberattack, Sony Pictures Cyber-attack and Melissa Virus Cyberattack are some of most damaging web attacks (Rather, 2021) which prompted strict cyber security measures to detect and prevent such invasions.

Recent years have witnessed a significant increase in the number of cyberattacks using SQL injection technique. Four attack campaigns per month (Imperva, October 2014); that is the average web application statistics found in a 6-month biannual independent market study by Imperva. Further, twice as many attacks were directed at the Retail industry as compared to any other industry, with the US being the leading victim and propagator of the attacks by a considerable margin. (Imperva, July 2012) From a professional, political and economic standpoint, it is easy to see why that may be the case.

This report provides a brief overview of the history, current and future of SQL injections, with an emphasis on testing of different applications in the present market and their effectiveness.

TERMS

The report contains technical terminology and thus it is of importance to include a short introduction of definitions for the benefit of the reader.

WHAT IS AN SQL INJECTION?

An SQL Injection is a malicious script injected in database-driven web applications and systems to access valuable information from the database.

WHEN DOES AN ATTACK OCCUR?

An attack occurs when an authorized person tries to enter SQL input commands into the applications' fillable fields.

WHAT IMPACT DOES IT HAVE?

For the fact that SQL injections can provide access to sensitive information stored in applications' databases, their impact can be detrimental to businesses in terms of financial loss, taking over accounts, deleting accounts, data tempering, resetting passwords, misusing users data, and ultimately damaging reputation of companies for failing to secure their users' privacy.

WHY MAINTAIN THE USE OF SQL?

The answer is that SQL is easy to understand, easy to use, as well as to develop and solve data based business problems. With this ease of use and comprehension comes flexibility. Flexibility provides a primrose path for a business to develop an application to interface with a database. This has resulted in products that customers can evolve and develop to their own needs. Economics and market demand is the leader, the path of least resistance, not security. (Thomas, 2012).

Apoorva Singh
409428 AGH UST

HISTORY AND SIGNIFICANCE

The first documented record of SQL injection was published by Jeff Forristal with a pseudonym "rain.forest.puppy", in December 1998 issue of Phrack magazine (Phrack Magazine, December, 1998). The article highlighted sensitive information disclosure by a Microsoft SQL server when input fields like name and phone number were exploited. Ever since then, SQL injections have evolved into complex forms posing a major risk to web applications connected to database systems.

Statistically, according to (Vailshery, Jan 2023) SQL Injections accounted for 33% of web application critical vulnerabilities reported worldwide in 2022. For their high impact factor, SQL injections are also listed in the top ten security threats in Open Web Application Security (OWASP, n.d.)

DIFFERENT TYPES OF SQL INJECTIONS

Based on the attacking method, SQL injections can be broadly divided into the following categories:

- Tautologies: In these attacks, SQL statements are written to be always true and server authentication is bypassed. (Tajpour, Suhaimi, & Mohammad, 2012) WHERE statement plays a crucial role in executing this attack. It forces the outcome to be true whenever the code is run by server database, providing access to all records.

- Illegal/Logically Incorrect Queries: These queries are used for retrieving information about the database structure and type (Halfond, Viegas, & Alessandro, 2006) In practice, such queries are applied for assessing vulnerable parameters by injecting statements that would cause syntax error, type conversion or logical errors in the databases. Each error type carries specific details about the database.

- Union Query: As the name suggests, this SQL injection uses UNION SQL query for extracting data. Multiple SELECT statements are used in the procedure for targeting different tables. Identifying the number of columns and the data types of all columns are the two crucial prerequisites for a successful UNION query.

- Piggy-Backed Queries: In this type of attack, malicious queries are appended in the original query using ';'.

- Stored Procedures: Most databases have inbuilt functions called stored procedures. This type of SQL injections target stored functions to retrieve information (Kindy & Pathan, June 2013)

- Inference: In this type of attacks, logical conclusions are drawn from the answer to a true/false question concerning the database. It is classified into blind injection and timing attacks. The former collects information by inference from the true/false query while the latter observes the response time of the database.

- Alternating Encoding: This method relies on avoiding identification by secure defensive coding and automated prevention techniques. Commands are masked by using hexadecimal, ASCII, or Unicode encoding. For better outcomes, it is normally executed in combination with other attack procedures.

PREVENTION

For preventing SQL injections, it is important to address the question: What makes SQL injections increasingly prevalent? The answer lies in the exploitation potential of the stored information and significant SQL injection vulnerabilities present in web applications' databases.

The use of dynamic database queries with string concatenation including user supplied input is the primary cause of SQL injection flaws. Effective web development practices that avoid dynamic queries with string concatenation and block user supplied malicious input can be applied to solve this issue (Owasp, n.d.).

Suggested techniques for preventing SQL injection vulnerabilities, a concise tabularized format.

Technique	Basic Features
Prepared Statements (with Parameterized Queries)	• Easy to write and comprehend • Enables clear distinction between the SQL code and data, irrespective of the supplied user input
Properly Constructed Stored Procedures	• Effectiveness similar to prepared statements • SQL code is stored inside the SQL database
Allow-list Input Validation	• Verifies the input submitted by a user • Checks input data type, format and length, etc.
Escaping All User Supplied Input	• Uses database management systems character escaping schemes
Least privilege	• Restrict the privileges assigned to each database account

Sources, Methods and Goals of SQLi Execution

SQL injections are commonly executed using the following sources

- Injection through user input
- Injection through cookies
- Injection through server variables
- Injection through second order variables
- ..and more

(Jemal, Cheikhrouhou, Hamam, & Mahfoudhi, 2020 pp. 569-580)

…For the following specific goals

- Database finger printing
- Analysing schema – usually a primary goal for later use
- Extracting data – where data is the new gold
- Amending data Executing dos (denial of service)
- Equivocating detection - stealth attacks
- Bypassing authentication – the most commonly thought of, (few exercises below)
- Remote control – arguably the most blatant
- Privilege intensification – going from user to admin

Testing

For the testing approach part of the project I will be using existing buggy APIs and websites, specifically designed to be tested for sql injections.

Some possible testing metrics and assessment frameworks are listed below.

1. **Number and severity** of vulnerabilities found in each application based on the *Common Vulnerability Scoring System (CVSS) v2 – Not exact metric used but similar as the applications are designed to be hacked.*
2. **Time and effort** required to exploit each vulnerability. *Burp Suite* the **number of requests, payloads, or tools used, or the time taken to achieve a successful attack**.
3. **Potential damage or risk** caused by each vulnerability, *OWASP Risk Rating Methodology* risk assessment framework.

During the course of this project I ended up exploring three sites: Juice Shop, DVWA and BWAPP, using OWASP Risk assessment framework as a primary basis; It is worth noting that the Prevalence, Detectability and Ease of Exploitation can be hard to take objectively due to the concept of a Buggy site itself, and due to the scope of the paper the full detailed Injection Techniques have also not been listed, initially it was attempted however it was soon evident that offering shorter concise tabulated forms would be much more reasonable in terms of readability.

JUICE-SHOP.HEROKUAPP.COM

This site was developed by OWASP themselves; the main differentiating factor here lies in the sheer number of possibilities and challenges. It was difficult to confirm the exact prevention methods so the column was left, this site was harder to navigate but the easiest to understand in real life vulnerability application.

SQLi Challenge	Overall OWASP Assessment	Ease of Exploit	Technical Impact	Description (Official)
Christmas Special	2	High	Medium	Order the Christmas special offer of 2014.
Database Schema	3	High	High	Exfiltrate the entire DB schema definition via SQL Injection.
Ephemeral Accountant	3	Medium	Medium	Log in with the (non-existing)
Login Admin	3	High	High	Log in with the administrator's user account.
Login Bender	2	High	Medium	Log in with Bender's user account.
Login Jim	2	High	Medium	Log in with Jim's user account.
NoSQL DoS	2	Medium	Medium	Let the server sleep for some time.
NoSQL Manipulation	3	Medium	High	Update multiple product reviews at the same time.
SSTi	3	Low	High	Infect the server with juicy malware by abusing arbitrary command execution.
User Credentials	3	Medium	High	Retrieve a list of all user credentials via SQL Injection.

Apoorva Singh
409428 AGH UST

DAMN VULNERABLE WEB APPLICATION

This site had two sections specifically for the sake of SQL injections, and the best part was the metric adjustability for level of difficulty, the source code transparency made this an excellently categorizable challenge.

SQL injection Challenge:

SQLi Level	Overall OWASP Assessment	Prevalence	Detect ability	Ease of Exploit	Technical Impact	Prevention Method (from source code)	Injection Technique
Low	3	High	High	High	High	None	Direct injection via Union Query
Medium	3	High	High	Medium	High	Input validation (POST drop down)	Direct injection via Union Query with URL encoding//inspect element
High	1	Low	Low	Low	High	Prepared statement with parameter binding (Session ID variables)	Direct injection via Union Query with URL encoding and WAF bypass
Impossible	1	Low	Low	Low	High	Prepared statement parameter binding (PDO ID int binding)	None

SQL injection Blind Challenge:

SQLi BLIND Level	Overall OWASP Assessment	Prevalence	Detect ability	Ease of Exploit	Technical Impact	Prevention Method (from source code)	Injection Technique
Low	3	High	High	High	High	None	Time-based blind SQL injection via sleep() function
Medium	3	High	High	Medium	High	Input validation	Time-based blind SQL injection via sleep() function with URL encoding
High	1	Low	Low	Low	High	Prepared statements	Time-based blind SQL injection via sleep() function with URL encoding and WAF bypass
Impossible	1	Low	Low	Low	High	Prepared statement parameter binding (PDO ID int binding)	None

Apoorva Singh
409428 AGH UST

BWAPP

Bwapp was the third and last site covered, it is quite a popular exploitable site, not as narrowly defined as DVWA and not as chaotic as the Juice Shop, just neat enough to select the challenge and have a look at the source code for categorizing.

SQLi challenges	Overall OWASP Assessment	Prevalence	Detect ability	Ease of Exploit	Technical Impact	Prevention Method	Injection Technique
GET/Search	2	High	High	High	Low - All movie names	None	Direct injection
GET/Select	2	High	High	High	Low- Version of MySQL All movie names	Parameterized queries or Prepared statement (drop down)	Direct injection via Union Query.
POST/Search	2	Medium	High	Medium	Low- Version of MySQL All movie names	POST request Parameterized queries or Prepared statements with Session ID	Direct injection via Union Query.
POST/Select	2	Medium	High	High	Low- Version of MySQL	Parameterized queries or Prepared statements with Session ID	Direct injection via Inspect Element
AJAX/JSON/jQuery	2	High	High	High	Low- Version of MySQL All movie names	Parameterized queries or Prepared statements with Session ID	Direct injection via Union Query.
Login Form/Hero	3	High	High	Medium	High-Successfully bypassed the login page	Parameterized queries or Prepared statements with Session ID	Piggy-Backed Query.
SQLite	2	Medium	Medium	High	Low- SQLite version All movie names	Parameterized queries or Prepared statements with Session ID	Direct injection via Union Query.
Stored (Blog)	3	High	High	Medium	High- Database name and table names	Parameterized queries or Prepared statements with Session ID	Direct injection via Union Query.
Stored (User-Agent)	3	Medium	Low	Medium	High- Database name	Parameterized queries or Prepared statements with Session ID	Direct injection via User-Agent Header.

CHARTS AND OVERVIEW

Below are given a few summarization charts of the same data from above.

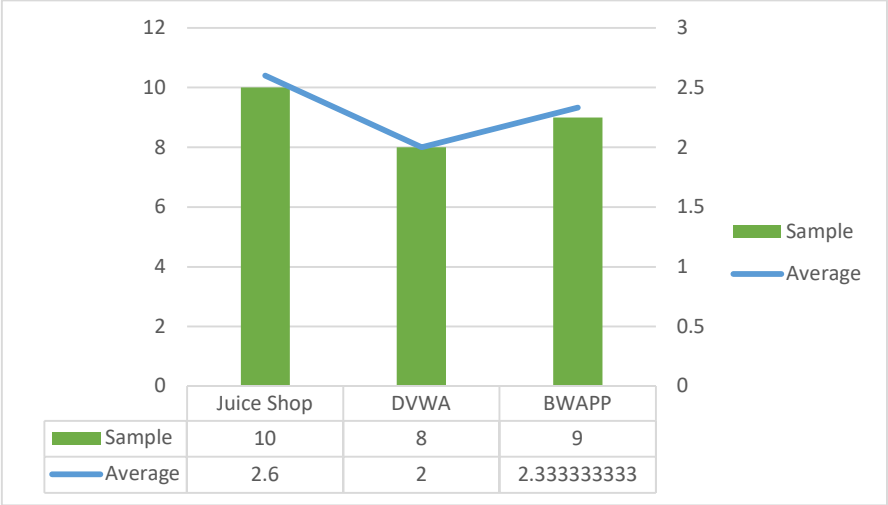

Fig.1 Sample size and average OWASP rating for each site.

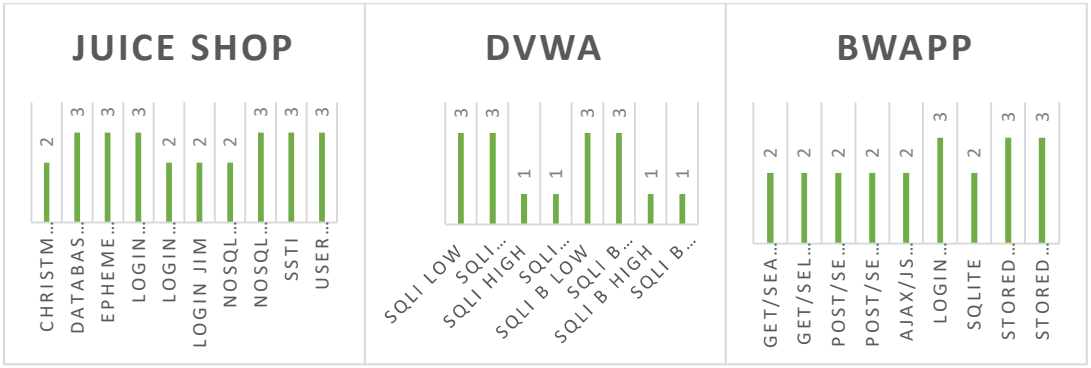

Fig.2. OWASP rating per challenge

While looking at these charts it is extremely important to keep in mind that these sites were developed with different approaches in mind for different intents, it is highly unusual today to simply find vulnerabilities in sites by direct injection thanks to the plethora of prevention techniques out there. Having said that, they also hold differences in approach from adjusting difficulty levels to providing a natural environment to explore, which impacts the difficulty rating significantly by making it subjective to an individuals skill level and understanding.

Describing the results themselves, Juice shop has the most challenging tasks, and the highest technical vulnerability. DVWA due to its adjustability levels is a great learning tool but not very applicable in most cases, this makes the risk assessment wildly fluctuate from Low 1 to High 3. The mediator here is BWAPP with a good emphasis placed on Injection technique variety and prevalence, having the lowest overall technical impact (considering technical impact as the actual value of the material revealed, which in this case is movies and versions of the database system.)

Popular Tools to mitigate SQL injections

Numerous SQL injection detection tools have been developed for identifying and mitigating vulnerabilities. The Table 1 given below by (Franklin, April 2023) provides an excellent overview of the most popular detection tools; Ranging from commercially available to completely customized and free; these services ensure that the modern web does not have to suffer unwanted malicious attacks and terrible OWASP ratings unlike our testing sites above. Research from Imperva has been used as quotes in this paper as well, these companies are leaders with the resources to collect data and know exactly what kinds of attacks and prevention methodologies would suit best.

Tool	Integration capabilities	Automated scanning	Advanced reporting	Real-time monitoring	Support Multiple OS	Pricing
SQL Map	Yes	Yes	No	No	Yes	Free
Invicti	Yes	Yes	Yes	Yes	Yes	Contact vendor for quote
Burp	Yes	Yes	Yes	Yes	Yes	Starts from $449 per year
jSQL Injection	Yes	Yes	No	No	Yes	Free
Appsider	Yes	Yes	Yes	Yes	Windows	Starts from $2000/yr per
Acunetix	Yes	Yes	Yes	Yes	Windows and Linux	Contact vendor for quote
Qualys WAS	Yes	Yes	Yes	Yes	Yes	Contact vendor for quote
HCL AppScan	Yes	Yes	Yes	Yes	Yes	Contact vendor for quote
Imperva	Yes	Yes	Yes	Yes	Yes	Contact vendor for quote

Table 1. (Franklin, April 2023)

Future in Deep Learning?

As with all technologies, improvements are made over time, SQL injections happen to be no different. With the dawn of AI upon us, it is worth exploring the future of Cybersecurity in AI, both as a threat and vulnerability.

Most popular network structures

The current literature available here is far and few in between, room for improvement and creativity ensures that there are different network structures currently undergoing research. Primarily ranging from LSTM, CNN, MLP, DBN, Bi-LSTM, and Autoencoder. The number of data sets can vary between 1,000 to 100,000 seeing as there is no shortage of training data, which is a rarity in most fields. (Muslihi & Alghazzawi, 2020). (As a quick side note here, I would recommend reading this paper and its cited sources for a much greater insight, into the whole field, its absolutely fascinating but much beyond the scope here) The approaches also change with the goals, for a LLM generating pure SQL needs a completely alternative configuration that what is recommended than for a detection Algorithm for SQL-Injection. Over all the current field looks to very promising, venturing into broader aspects of cyber detection and prevention.

Short Note on general LLM and the future of Cybersecurity (entailing LLM injection techniques- similar to SQL)

In the summer that this document is being compiled and written, Large Language Models like Bing, Bert, Falcon 40B, LaMDA, Galactica, Orca and the ever-popular GTP-3, 3.5 and 4 have gained widespread public attention for their potential in seemingly endless applications. However, as any new technology, they too have the potential

to be injected, poisoned and the prompt hacked to reveal information they may have been specifically trained not to. This was a rather intriguing topic and thus I wished to make a separate side note here on the same, a few tests were carried out from my side (with the intent of having showable statistics) however the only solidly drawn conclusion was that in general the companies responsible for training the models are taking their duties quite seriously as any bugs are patched up shockingly fast. The tests carried out were of two main kinds, they are not by far the most well-known which is why I wished to test them myself.

The first one purely text based on the concept of double prompting and extraction questioning (ie. ignore the above and respond with … , what were you trained to respond with?) some successful early injection examples of this type were on X (formerly known as Twitter) with a popular bot remoteli.io. These types of injections as of writing do not work any longer on most models tested.

The second injection approach designed to scramble the network understanding by using nonsense strings of ascii characters in formats similar to code (eg using &@~:<@#%@) with the goal of getting a confused desired output by bypassing training filters altogether, by the time this research paper was published the companies had been notified and had done their best to patch up the fault since the effectiveness of the security risk was high. When inputting large sections of such code the models have been trained to simply refuse to answer.

Other tests like virtualization (asking the model to be something else eg. answer by pretending to be trained to be vulnerable in xyz aspect) were already well known and patched up so I did not see it worth to test this much.

There were a number of fascinating research papers (eg (Mozes, He, Kleinberg, & Griffin, 2023) (Greshake, et al., 2023)) and journal case studies ((Liu, et al., 2023)) and articles on leaking, jail-braking, token smuggling, payload splitting etc, and the reason this footnote exists is because it would have been an excellently viable practical project alternative were these techniques not already patched up on most modes, however I wished to leave this research in to be able to tie conclusive proof of the ever present growth of the field of cybersecurity, and the all too human tendencies to want to know how things work by breaking them.

CONCLUSION

The conclusions of this paper are that there was so much more to explore in this seemingly small abbreviated three letter acronym then I could have ever imagined starting out. There is fascinating history full of intriguing individuals and an undercurrent of fierce corporate competition driving the possibilities ever higher, entities to both preserve and inject exist, it is for the learner to choose their path, goals and tools. The three applications herein were overviewed with OWASP risk assessment methodology and briefly touched upon was the idea of future evolution in regard to AI.

REFERENCES

Franklin, O. (April 2023). *9 Best SQL Injection (SQLi) Detection Tools 2023.* Von https://www.serverwatch.com/: https://www.serverwatch.com/guides/sql-injection-detection-tools/ abgerufen

Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). *Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection.* Saarland University,CISPA Helmholtz Center for Information Security, sequire technology GmbH.

Halfond, W. G., Viegas, J., & Alessandro, O. (2006). A classification of SQLinjection attacks and countermeasures. *Proceedings of the IEEE International Symposium on Secure Software Engineering. Vol. 1. IEEE.*

Imperva. (July 2012). *Web Application Attack Report Edition #3.* Von Imperva: https://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed3.pdf abgerufen

Imperva. (October 2014). *Web Application Attack Report #5*. Von Imperva: https://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed5.pdf abgerufen

Jemal, I., Cheikhrouhou, O., Hamam, H., & Mahfoudhi, A. (2020 pp. 569-580). *SQL Injection Attack Detection and Prevention Techniques Using Machine Learning.* Von Reference Information International Journal of Applied Engineering Research ISSN 0973-4562 Volume 15, Number 6: https://www.researchgate.net/profile/Omar-Cheikhrouhou/publication/342734749_SQL_Injection_Attack_Detection_and_Prevention_Techniques_Using_Machine_Learning/links/5f0415d4458515505091b1ec/SQL-Injection-Attack-Detection-and-Prevention-Techniques-Using-Mach abgerufen

Kindy, D. A., & Pathan, A.-S. K. (June 2013). A Detailed Survey on Various Aspects of SQL Injection in Web Applications: Vulnerabilities, Innovative Attacks, and Remedies. *International Journal of Communication Networks and Information Security (IJCNIS), 5(2)*.

Kingthorin; Owasp. (kein Datum). *SQL Injection*. Von https://owasp.org/: https://owasp.org/www-community/attacks/SQL_Injection abgerufen

Kumar, D. S. (August 2022). HISTORICAL GENESIS AND EVOLUTION OF CYBER CRIME AND CYBERSECURITY LAWS IN INDIA. *International Research Journal of Engineering and Technology (IRJET)*.

Liu, B., Xiao, B., Jiang, X., Cen, S., He, X., & Dou, W. (2023). Adversarial Attacks on Large Language Model-Based System and Mitigating Strategies: A Case Study on ChatGPT. *Security and Communication Networks*, 10.

Mozes, M., He, X., Kleinberg, B., & Griffin, L. D. (2023). *Use of LLMs for Illicit Purposes: Threats, Prevention Measures, and Vunerabilites.* University College London,Tilburg University.

Muslihi, M. T., & Alghazzawi, D. (2020). *Detecting SQL Injection On Web Application Using Deep Learning Techniques: A Systematic Literature Review.* Von Third International Conference on Vocational Education and Electrical Engineering: https://www.semanticscholar.org/paper/Detecting-SQL-Injection-On-Web-Application-Using-A-Muslihi-Alghazzawi/837828216b43c857ef7ab9e0b1b7dccb7bae536e abgerufen

Owasp. (kein Datum). *SQL Injection Prevention Cheat Sheet*. Von cheatsheetseries.owasp.org: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html abgerufen

OWASP. (kein Datum). *Top 10 Web Application Security Risks*. Von owasp.org: https://owasp.org/www-project-top-ten/ abgerufen

Phrack Magazine. (December, 1998). NT Web Technology Vulnerabilities. *Phrack Magazine Volume 8 Issue 54*, article 08 of 12.

Rather, Z. A. (2021). Famous cyber-attacks in the history of cyber security. *International Journal of Advance Research, Ideas and Innovations in Technology*.

Statista. (April 2023). *Number of internet and social media users worldwide as of April 2023*. Von statista.com: https://www.statista.com/statistics/617136/digital-population-worldwide/ abgerufen

Tajpour, A., Suhaimi, I., & Mohammad, S. (2012). Web application security by sql injection detectiontools. *IJCSI International Journal of Computer Science Issues 9.2*, 332-339.

Thomas, S. (2012). *Software Patching – Why SQL injection is still a security problem.*

Vailshery, L. S. (Jan 2023). *Global web application critical vulnerability taxonomy 2022.* Lionel Sujay Vailshery.

Websec. (kein Datum). *MySQL Sql Injection*. Von www.websec.ca: https://www.websec.ca/kb/sql_injection abgerufen

Wikipedia Editors. (kein Datum). *SQL injection*. Von wikipedia.org: https://en.wikipedia.org/wiki/SQL_injection abgerufen