Data-Mining and Hashing to Prevent Application-Layer DDoS and SQL Injection Attacks

Ahmed Abadulla Ashlam
Department of Computer Science
University of Reading
Reading, UK
a.ashlam@pgr.reading.ac.uk

Atta Badii
Department of Computer Science
University of Reading
Reading, UK
atta.badii@reading.ac.uk

Frederic Stahl
German Research Center for Artificial Intelligence GmbH (DFKI),
Lower Saxony, Oldenburg, 26129, Germany
frederic_theodor.stahl@dfki.de

2023 IEEE International Conference on Advanced Systems and Emergent Technologies (IC_ASET) | 979-8-3503-2102-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IC_ASET58101.2023.10150694

***Abstract*—Applications built specifically for the web are rapidly growing in significance. Internet access is crucial to the smooth operation of many critical services, including medical care, banking, retail, information sharing, and transportation. Since most applications are hosted in the cloud, it makes sense for data owners to be very concerned about data integrity. Malicious actors attempting to access the cloud environment must be stopped using strong security measures. Several types of attackers target the network at the same time, using different methods. The purpose of this project is to protect the database against attacks that originate from the client side. Examples of such attacks include application-layer distributed denial-of-service attacks and SQL injection attacks. Distributed denial-of-service attacks, often known as DDoS attacks, occur at the application layer when an attacker sends a flood of requests to a target service. SQL injection attacks, on the other hand, are a kind of attack that bypasses normal safeguards by launching malicious scripts directly into the database. In order to prevent application-layer DDoS attacks and SQL injection attacks, a new method has been proposed. This strategy involves ensuring that the login data (a legitimate username and password) matches both the usernames and passwords stored in the database on the client side. Additionally, it involves being able to handle this data in the form of hashing, making use of datamining, and employing the Python programming language for the implementation of cryptographic algorithms using the SHA-256 hash function. Both of these types of attacks can be prevented by implementing this strategy. Since only a few changes to the source code of the programming language are needed, this strategy can be quickly added to any web application that has already been built. This is true no matter what programming language or database was used to build the application.**

Keywords— Data-mining, Hashing, SQL-Injection, DDoS

I. INTRODUCTION

Cloud computing is now one of the most significant security risks, as it enables attackers to exploit vulnerabilities and use system resources in order to carry out their malicious activities. As a result of the fact that millions of users share the same computational resources in a cloud-based attack of this kind are very successful. We are providing the answer to several attacks, at various phase attacks as, all inside a single system. The stakeholders will gain from the work on this project since it will be able to provide automated intrusion detection and protection of two important and frequent cloud attacks, DDOS attacks and SQL injections attacks. An attacker will use a technique called SQL injection to break into a database by executing SQL queries that contain malicious code. By using this approach, an adversary may get access to credential data such as passwords that have been saved in a database. During a distributed denial of service (DDoS) attack, the attacker will use several computers and various Internet connections in an effort to overwhelm the resource that is the target of the attack. The findings of this research provide a solution to problems such as SQL injection attacks and DDoS attacks.

Cybercriminals are employing a variety of approaches in order to get these results from their work. Both Distributed Denial-of-Service attacks (DDoS) and Structured Query Language Injection attacks (SQLi) are among the most prevalent and damaging types of cyberattacks[2-4].

One of the most common forms of attack against a database system is the SQLi. An attacker will enter a statement written in Structured Query Language into a text field in order to get access to resources or manipulate data that is saved in a database [5].

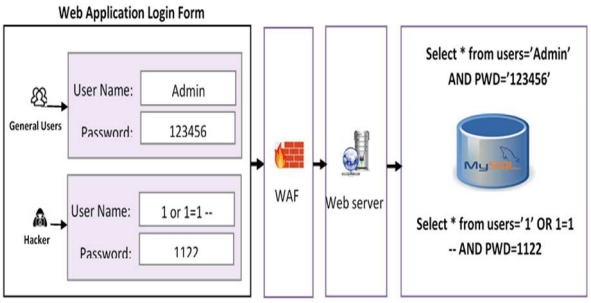

Fig. 1. SQL Injection Attack Process [1].

The most significant dangers to a web-based application are DDoS and SQLi attacks [6]. According to the threat data gathered by NetScout in 2019, there were a total of 8.4 million registered DDoS attacks. This equates to 23,000 attacks

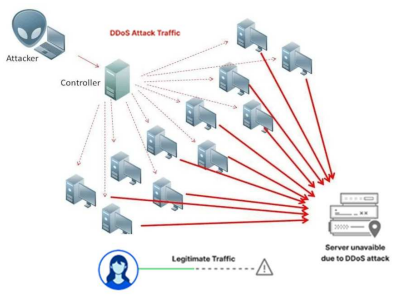

Fig. 2 DDoS Attack Process

occurring each day, 670,000 occurring each month, and 16 occurring per minute [4].

In this paper, a novel defence mechanism is developed with the goal of simultaneously detecting application-layer DDoS attacks and SQLi threats. A solution to the vast majority of cyber-attacks directed at a website might be provided by a system with two functions: detection of DDoS attacks and SQLi attacks. Most cyberattacks on a website fall into one of two categories; either they are DDoS or SQLi injection attacks. In order to avoid such attacks, a novel approach has been put third, and it entails checking that the login information (consisting of a genuine username and password) is consistent with the users and passwords that have been saved in the database on the client side. In addition to this, it requires the capability to handle the data in the form of hashing, the use of data mining, and the utilisation of the Python programming language for the creation of cryptographic methods using the SHA-256 hash function. A system with both of these capabilities has the ability to solve both of these problems. Therefore, databases, websites, and other online infrastructures can be protected from potential threats.

The remainder of the paper can be found in Section II, which is devoted to a discussion of the relevant prior work. Section III of the paper discusses the experiment and research on preventing and detecting application-layer DDoS and SQLi attacks. The findings of the experiment are presented in Section III. In conclusion, the study is summed up in Section IV.

II. Related work

The study of DDOS and SQL injection-cloud-based attacks has received contributions from a number of scholars. Since there are more hackers than ever before, a hybrid design that incorporates as many cloud attacks as possible into a single programme may be an effective and safe response. Furthermore, numerous scholarly articles provide a theoretical explanation of these vulnerability attacks and make use of simulators to get experimental findings. Thus, the outcomes from the simulation would not be representative of the actual time [7].

A. Distributed Denial of Service (DDoS)

One of the most damaging kinds of attack is known as a DDOS attack. It is mostly directed against a web server, network, or website. DDoS attacks may quickly take down an entire website by flooding it with many more requests than the server can process. They may be broken down into three distinct categories [6, 8].

1) Volume-based attacks: In volume-based DDoS attacks, the anomaly is caused by consuming the target's useable bandwidth between the target and the internet. It is performed mostly by directing a very high volume of traffic at a certain target. The UPD Flood, the ICMP Flood, and the Spoofed-Packet Flood are just a few examples [6].

2) Attacks based on protocols: These may bring a service to a standstill if they use up all of the server capacity. It is also known as a state-exhaustion attack. A case in point is that There are several instances, some of which are SYN Flood, Ping-Of-Death, and Smurf DDoS [6].

3) Application layer attacks: In this scenario, the attacker creates a connection with the target and then monopolises processes on the server, depleting its resources in the process. This attack is shown by the Slow Loris and Apache examples [6]. These attacks are more difficult to detect since they often utilise less bandwidth than other forms of attacks. Requests per second are the standard for measuring attacks at the application layer.

B. SQL Injection Attack

SQLi attacks are a kind of online hacking method in which an attacker inserts code to alter a SQL query in order to gain unauthorised access to a database. This may be accomplished through hacking a website. Since these applications and their associated databases are accessible via the Internet, they are susceptible to a wide variety of threats to information security. This is because the practise of exchanging information over the Internet by utilising a variety of channels and web applications has become a phenomenon that is fairly widespread. An example of a website that is vulnerable to SQL injection attacks is one that enables visitors to log in by inputting their username and password. Using this kind of website may help better illustrate the nature of these attacks. The following is the query that will be built in the event of a successful permitted login attempt using the username 'user' and the password '123'. SELECT * FROM users WHERE name = 'user' and password = '123', it is also feasible for a user who has malicious intentions to write the following data into the website username field by using 'user' and the password field by using '' or '1'='1'[9-11].

Web applications could expose many vulnerabilities that make them susceptible to SQLi attacks; so that there exist many ways to attack web applications via SQLi. SQLi attacks can be classified to seven categories [12]:

*1) Boolean-based SQLi attack (*tautology attack).

*2) Booleans (*true or false) are used to perform this type of SQLi.

3) Union-based SQL injection: This is the most common of all SQL injections. A UNION query is used to combine two or more selected queries into an SQL statement, thus illegally getting data from the database.

4) Error-based SQLi: This is a type of SQLi vulnerability that affects web applications that use Microsoft SQL Server. This vulnerability is caused when an attacker supplies an invalid SQL statement that causes a programming error.

5) SQLi attacks for batch query: This form of SQLi is the most dangerous as it aims to take full control of the database.

6) SQLi on a symmetric basis. This type of injection is used by hackers to impersonate a specific user using the keyword of SQL LIKE with a wildcard operator (%).

*7) Positional number systems (*Hexadecimal, decimal, binary) variation attack: Instead of using standard strings and characters for injection code, the hacker takes advantage of the diversity of SQL by using hexadecimal or decimal representations of keywords.

C. Hashing algorithms

Hashing algorithms are mathematical functions that scramble data, making it unintelligible. Since hashing techniques are one-way processes, the text cannot be deciphered by anybody else.

Modern authentication systems value hashed message strength. Since passwords are kept as hashes, the hashing process is closely related to the password system. Rainbow tables modelled the attack using SHA-256. A Java technique for SHA-256 rainbow table construction is provided. Rainbow tables work under certain circumstances. This article demonstrates how to generate a password and use rainbow tables to attack SHA-256. Rainbow tables expose three-character passwords in three seconds, according to studies. The decryption time increases directly with the password bit [13].

Blockchain distributes digitally created data without copying. Blockchain technology drives the new internet (i.e., virtual currency). Emerging smart settlement structures over decentralised cryptocurrencies allow jointly suspicious events to transact satisfactorily without third intermediaries (i.e., the reason to provide wide security to blockchain technology). Cryptography uses various algorithms for security, including MD5, AES, RSA, SHA family, etc. Hash functions are very valuable and present in practically all information security applications, therefore hashing approaches are safer. A local blockchain application using SHA-512 is being developed by the authors. 64-bit words and 1024-bit blocks make SHA-512 a safe algorithm. With a few mathematical models, they are establishing that SHA-512 is more collision-resistant than its predecessor [13].

SQL Injection is a common online attack used by hackers to acquire company data. Hackers may insert dangerous SQL statements into input fields, which the SQL database executes. Web application URLs may also contain malicious SQL queries. SQL injection attacks persist despite much study on SQLA detection and prevention. This article covers SQL Injection vulnerability, attack types, and dangers. Hashing and Aho-Corasick pattern matching algorithm safeguard online applications [14].

III. Methodology

The fundamental aim of this research paper is to propose a new method for preventing application-layer DDoS attacks and SQL injection attacks. This new method involves ensuring that the login data (a legitimate username and password) matches both the usernames and passwords stored in the database on the client side and it also involves being able to handle this data in the form of hashing, making use of data mining and hashing methods.

This technique is readily adaptable to any database or programming language, and in some instances, an attacker may be able to read, remove, or edit the data that is stored in the database. Additionally, the attacker may have the ability to make a website or application inaccessible to users, which may have far-reaching and irreversible effects on the content or behaviour of the applications that are impacted. The new system would follow the procedure that is shown in Fig 3, which offers an overview of the process.

The process of the suggested improvement to the login system is shown in Fig 3. When a user signs into a website by entering their login information, the input will first be validated for authenticity before an HTTP request is sent to the server. This happens whenever the user logs onto a website[11]. The SQL query is built by making use of the dependent components of the login data, such as the username and password, among other elements. This is done in order to generate the query. After that, methods from the field of data

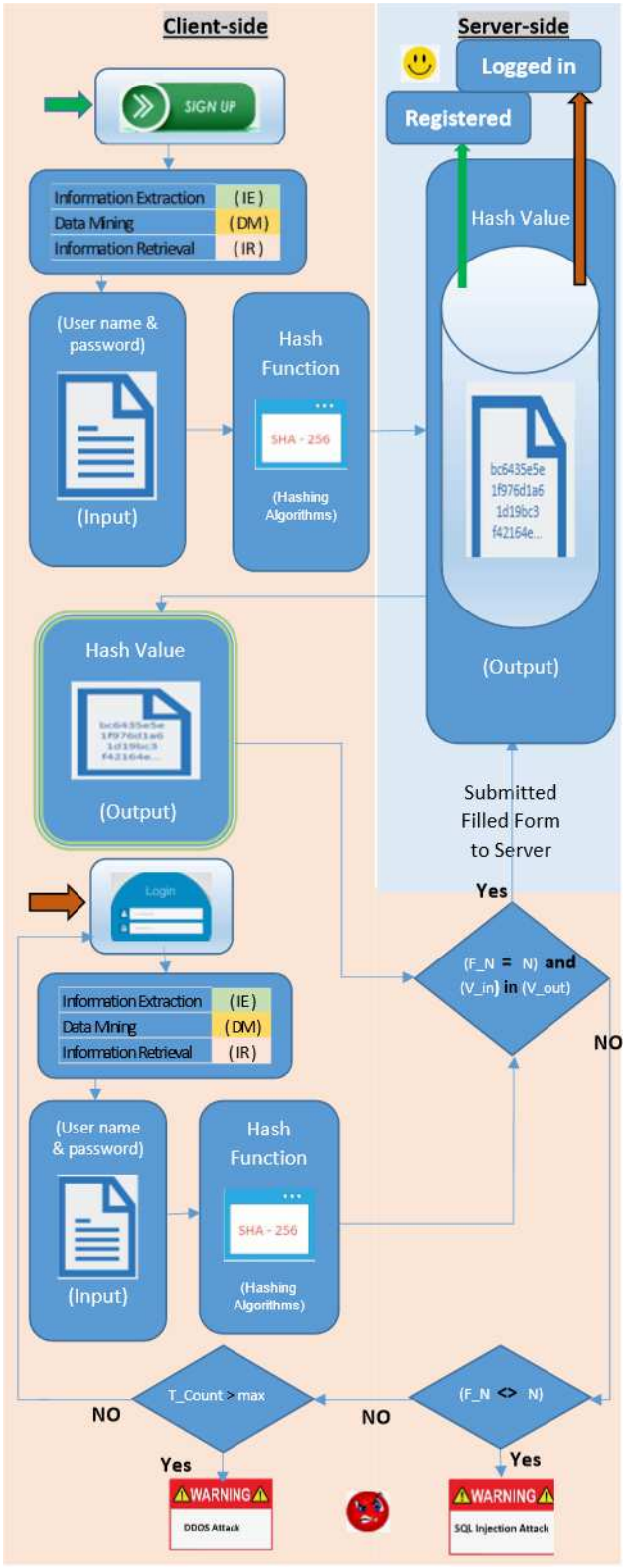

Fig. 3. Workflow for the proposed system.

mining are used in order to extract the credential components, including a password and username, from the query that was developed. Then proceed with the details provided.

*a)) The credential element values (*entered username-password) are encrypted with the SHA-256 hash function and then compared to the legitimate credential element values (saved password-username). If they match, the request is not

an attack and will be sent to the server; if not, it will move on to the next step.

b) In the event that the number of credential elements (which have been entered to log in): that were constructed in the generated query is not equal to the number of legitimate credential elements (which have been saved), this indicates that there has been a SQL injection attack. In this case, an alert will be sent, and the request will be terminated. If this is not the case, proceed to the next step.

c) If the number of unsuccessful attempts by the particular user exceeds the specified threshold, an alert will be sent; this indicates that it is an application-layer DDoS

d) attack, and the request is terminated. If the number of unsuccessful attempts does not exceed the threshold, it will disregard the input and process a new request by following the steps outlined above.

A. Hash-comparison code

The process of hashing may be split down into four separate parts, the first of which is the extraction of item values from queries via the application of data mining and analytical techniques. In the second step of the process, the values of the elements are hashed with the assistance of a function that generates hashes concurrently with the registration procedure.

This is carried out before the values of the elements are sent to the server. Third, while the user is in the process of logging in, the values of the elements are hashed using the technique that creates hashes. Next the value of these items is compared to the value of the elements that were recorded in the hash form that was obtained from the server in array format, which are shown in Fig 4.

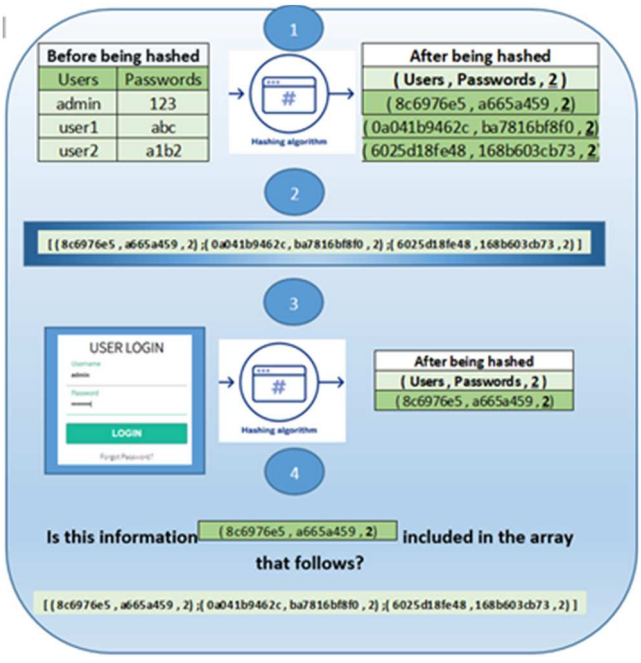

Fig. 4. The process of hashing.

B. Test Scenario

1) Consider a page for entering into a website: such as a login page, in which a user is required to input their username and password before being permitted access to the website.

2) The standard procedure for a user: to log in to the system and enter their credentials is described above. Once these steps have been completed, the user will be logged into the application.

*3) After the user enters: "*admin" as the username and "pass" as the password, the following SQL query will be generated and executed in response to the user's input:

```
SELECT * FROM tb_users WHERE
username = 'admin' AND password = ‘pass’
```

Fig. 5. Query of the user entrs

4) A data mining and analysis: approach has been used to get the values of the attribute from the created query (or any other specified value), and then the SHA-256 hash function

```
~~SELECT * FROM tb_users WHERE username =~~
'admin' ~~AND password =~~ ‘pass’
```

has been used to hash the values that have been retrieved.

Fig. 6. Query to obtain the value

5) The attribute values: that were received from the newly constructed query are hashed after the preceding step, the resultant hash is then compared to the hash that was previously produced by making use of the right attribute values, and if they match, the next step in the process is carried out.

```
SELECT * FROM tb_users WHERE
username = 'f6bf870a2a5bb6d26dd'
AND password = ‘515fe42639347efc92’
```

Fig. 7. Query of hashing value

6) The request is going to be rejected: if the hash does not match the payload. If the user performs a SQL injection query with malicious payloads or if they try to deliberately overwhelm the service, the following might happen:

```
~~SELECT * FROM tb_users WHERE username =~~
'admin' ~~AND password =~~ ‘pass’ ~~or 1=~~’1’
```

Fig. 8. Query to compare the hash value with the hash of payloads

C. Test Cases:

Applying this evaluation to the test used in these test cases, it is clear that it proves that the method, test results, and experiment can be used without any problems.

1) The query will be considered successful: if the hash of the values of the query's attributes are equal to the hash of the values of the base query's attributes and the number of items in the created query is equal to the number of stored attributes, as shown in fig 9.

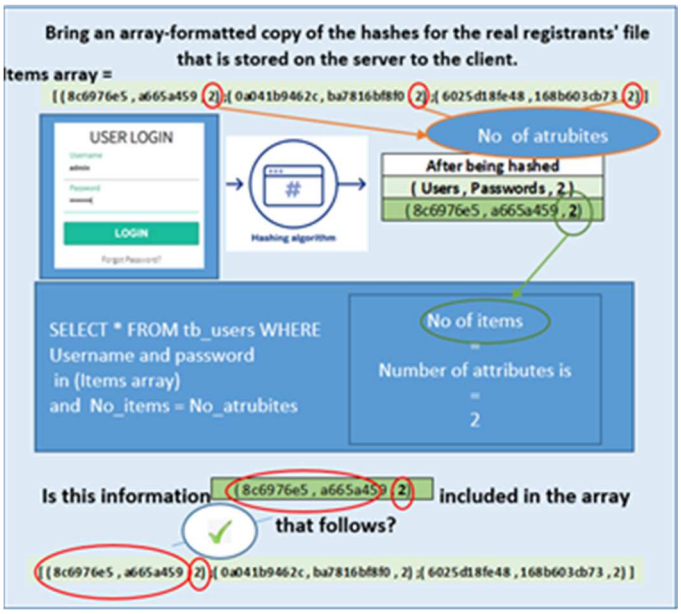

Fig. 9. Positive response to the query.

2) As can be seen in Fig 10: in the event that any SQL injection query that contains malicious payloads is applied to the login data, the hash of the credential attribute value of the login data will not match the hash of the values of the base credential attribute. Additionally, the number of credential attributes in the login data will not be the same as the number of base credential attributes. Be on guard for an attack that uses SQL injection as its method of attack.

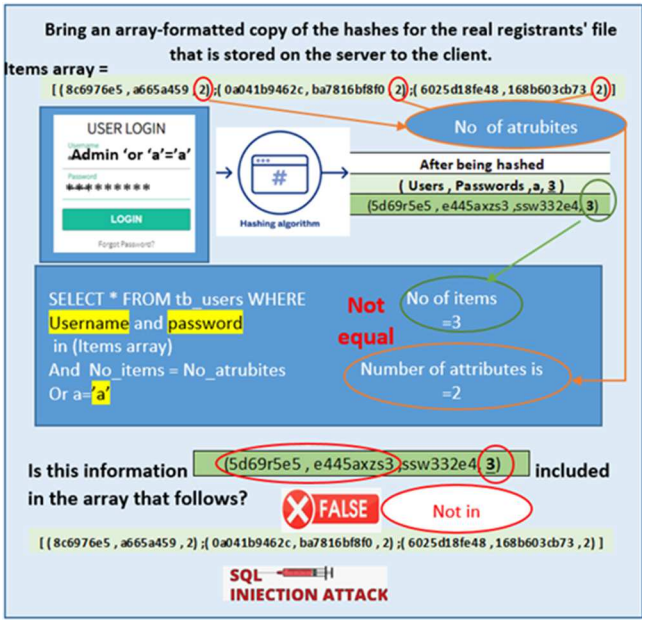

Fig. 10. Attack that uses SQL injection as its method of attack.

3) If the hash of the values: of the credential attributes does not match the hash of the values of the base credential attributes, and the number of credential attributes is the same as the number of base credential attributes, and the number of unsuccessful attempts made by the particular user is less than the threshold that was specified. Warning that an attack using DDoS is underway.

To summarise, it is important to keep in mind that the

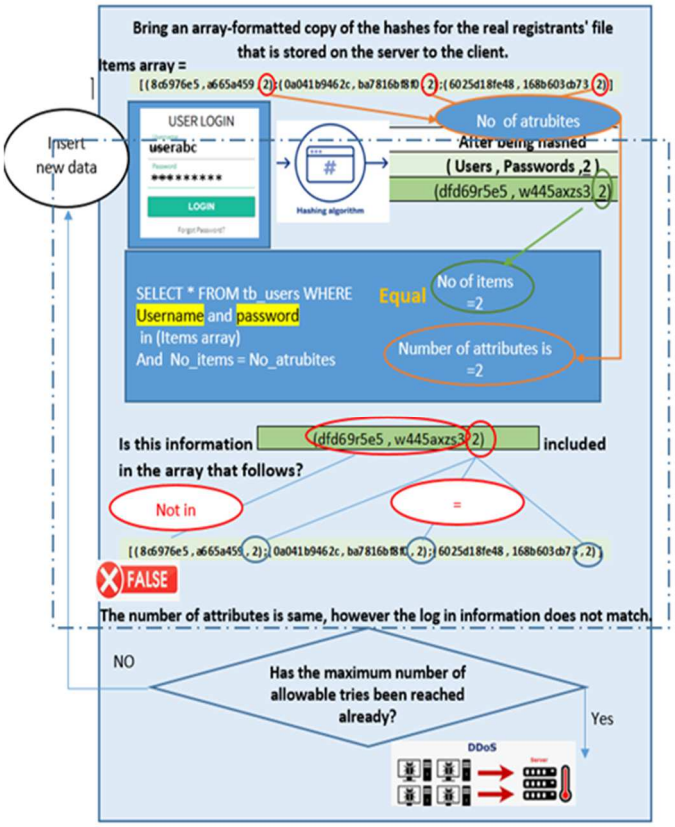

Fig. 11. Attacks using application layer DDoS attack.

request has not been sent to the server unless the valid registration data has been provided. This will secure the database from being penetrated and prevent it from overloading the service in real time. Additionally, it will increase the speed at which actual users can browse the web and retrieve data.

The notation that is used and presented in the technique can be easily added to any pre-built website built using any language or database type because it requires only a few changes to be made in the code of the language, and even under higher load conditions, it has a very trivial effect on performance due to low processing overhead. This makes it possible for the notation to be added to any pre-built website build.

IV. Conclusion and Further Work

All In this work, a unique defensive mechanism is created with the purpose of simultaneously detecting application-layer DDoS and SQLi threats. It is possible that a system that serves two purposes might offer some protection against the majority of cyberattacks that are aimed at a website.

For the detection of distributed denial-of-service (DDoS) attacks as well as the detection of SQL injection attacks, it is required to check that the login information (consisting of a genuine username and password) is consistent with the users and passwords that have been saved in the database on the client side. " In addition to this, it needs the capacity to handle the data in the form of hashing, the usage of data mining, and the utilisation of the Python programming language for the

building of cryptographic techniques using the SHA-256 hash function. A solution to each of these issues may be found in a system that has both of these qualities simultaneously. Databases, websites, and other forms of internet infrastructure may, thus, be shielded from any possible dangers that may arise. Because this method requires just a few modifications to the source code of the programming language, it may be swiftly incorporated into any web application that has already been constructed. This is the case regardless of the programming language or database used in the process of developing the application.

In the future, this complicated scenario may be researched, and in order to stop future attacks, the solution can include more sophisticated principles.

REFERENCES

[1] A. A. Ashlam, A. Badii, and F. Stahl, "A Novel Approach Exploiting Machine Learning to Detect SQLi Attacks," in 2022 5th International Conference on Advanced Systems and Emergent Technologies (IC_ASET), 2022: IEEE, pp. 513-517.

[2] P. V. S. Lakshmi, "Sql Injection Detection Analysis Using Deep Learning," Journal Of Engineering Sciences, Vol. 13, No. 08, 2022.

[3] A. BaniMustafa, M. Baklizi, and K. Khatatneh, "Machine learning for securing traffic in computer networks," International Journal of Advanced Computer Science and Applications, vol. 13, no. 12, 2022.

[4] O. K. Kambre, K. K. Shah, and P. D. Rathod, "SQL Injection Attacks and Defense Mechanisms," International Research Journal of Innovations in Engineering and Technology, vol. 7, no. 2, p. 101, 2023.

[5] J. M. Alkhathami And S. M. Alzahrani, "Detection Of Sql Injection Attacks Using Machine Learning In Cloud Computing Platform," Journal Of Theoretical And Applied Information Technology, Vol. 100, No. 15, 2022.

[6] N. N. Tuan, P. H. Hung, N. D. Nghia, N. V. Tho, T. V. Phan, and N. H. Thanh, "A DDoS attack mitigation scheme in ISP networks using machine learning based on SDN," Electronics, vol. 9, no. 3, p. 413, 2020.

[7] S. de Kinderen, M. Kaczmarek-Heß, and S. Hacks, "Towards Cybersecurity by Design: A multi-level reference model for requirements-driven smart grid cybersecurity," 2022.

[8] Z. R. Alashhab, M. Anbar, M. M. Singh, I. H. Hasbullah, P. Jain, and T. A. Al-Amiedy, "Distributed Denial of Service Attacks against Cloud Computing Environment: Survey, Issues, Challenges and Coherent Taxonomy," Applied Sciences, vol. 12, no. 23, p. 12441, 2022.

[9] S. Mishra, "SQL injection detection using machine learning," 2019.

[10] A. A. Ashlam, A. Badii, and F. Stahl, "Multi-phase algorithmic framework to prevent SQL injection attacks using improved machine learning and deep learning to enhance database security in real-time," in 2022 15th International Conference on Security of Information and Networks (SIN), 2022: IEEE, pp. 01-04.

[11] A. A. Ashlam, A. Badii, and F. Stahl, "WebAppShield: an approach exploiting machine learning to detect SQLi attacks in an application layer in run-time," World Academy of Science, Engineering and Technology:[Computer and Information Engineering], vol. 16, no. 8, 2022.

[12] Y.-S. Jang and J.-Y. Choi, "Detecting SQL injection attacks using query result size," Computers & Security, vol. 44, pp. 104-118, 2014.

[13] O. Manankova, M. Yakubova, and A. Baikenov, "Cryptanalysis the SHA-256 Hash Function using Rainbow Tables," Indonesian Journal of Electrical Engineering and Informatics (IJEEI), vol. 10, no. 4, pp. 930-944, 2022.

[14] A. Tajpour, M. Z. Heydari, M. Masrom, and S. Ibrahim, "SQL injection detection and prevention tools assessment," in 2010 3rd International Conference on Computer Science and Information Technology, 2010, vol. 9: IEEE, pp. 518-522.