Bypassing Two Factor Authentication Based On Classification Using Aho-Corasick Matching Algorithm For Nosql Databases

R. Shobana[1],MCA.,M.Phil., Dr. M. Suriakala[2],M.Sc.,M.Phil.,Ph.D.,

[1] Part time Research Scholar, University of Madras,
Assistant Professor, Department of Computer Science and Applications,
D.K.M. College for Women, Vellore- 1
Email: shobanavasu.mca@gmail.com
[2] Assistant Professor, Department of Computer science,
Government Arts College for Men, Nandanam, Chennai-35
Email: suryasubash@gmail.com

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

ABSTRACT
This paper presents a new method to protect the applications against NoSQL injection. NoSQL-Injection Attacks are a class of attacks that many of these systems are highly vulnerable to, and there is no known fool-proof defense against various attacks. Despite all vulnerabilities, two factor authentications remain the safest way to protect user accounts in traditional web apps. Two bypasses of factor authentication can occur from time to time, with various techniques involved in those attacks; the only way to do this is to maintain an iteration of one's approach to two authentication factor tests.

Keywords: Authentication, classification, matching, NoSQL database.

INTRODUCTION

Today, the Internet is becoming a widespread infrastructure for intelligence. The use of the Internet technology and rapid development have inspired the the number of lately stored data in a database. An increase in user count and a strong reliance on digital data is leading to value of spatial data secure, whether such data relates to commercial, corporate, institutional, institutional, environmental, health, personal and other internet related resources. Use with all web browsers running on any operating system can be used via Internet to all web applications. Web apps are now a commonly used interface for retrieving or inserting results [1]. The vulnerability is simply a vulnerability that enables an attacker to reduce the assurance of device knowledge. The reliability of the database was and will remain one of the most important facets of the safety of the applications. Database access gives the most sensitive information to an intruder to a risky extent. NoSQL databases have recently gained popularity because NoSQL databases are looser than standard SQL databases. NoSQL databases also have efficiency and scaling advantages because they require less relational restrictions and consistence controls [2]. Examples include MongoDB[3], CouchDB and FirebaseDB.

As it is an open source database, MongoDB can be tailored to all sizes of companies and individuals. For the program growth, the data pattern can be flexibly modified with a secondary index and full query scheme. But does this mean the NoSQL programs are injection-free? NoSQL is a vast community of computer database management systems that are an alternative to traditional relational database management systems. Primary query language is not SQL, nor do it normally include fixed table schemas. NoSQL storage framework helps users to modify data attributes and to add data anywhere. A paper design or even a compilation up-to-date is not needed.

NoSQL databases focus more on real-time data processing capabilities and are good for direct data access activity which greatly support interactive system growth. The freedom to adjust characteristics is one of the greatest benefits due to structural weakening, so the adjustment process is very easy. This great benefit, though, still affects its protection where injection attacks are the most prevalent. In the world of database relations, it is also the number one public enemy. The fact that NoSQL does not use SQL in query does not mean that it is resistant to injection threats. Many argue SQL is not working at NoSQL, but the idea is just the same: the attacker has to modify the injection grammar type. In other words, JavaScript or JSON[4] injection can also endanger the safety of SQL injection, but this is not going to be done. Our analysis reveals that the NoSQL injection attacks are still vulnerable, even though they do not use standard SQL syntax, Since they should be done in a procedural instead of a declarative SQL language like PHP injection attack and random injection of JavaScript.

In this paper aim is to design an effective NoSQL injection attacks detecting mechanism i.e.to counter injection attacks in NoSQL databases.To assure security all the websites need protections to their database. The organization of the paper is background of the study and section 2 shows the related works of existing methods and section 3 presents the research methodology. Performance analyses are discussed with existing method comparison in section 4 and finally conclude with section 5.

LITERATURE SURVEY

Many database protection vulnerabilities are triggered in this section. There are a lot of methods to read for injecting. One objective is to collect the database type and configuration to plan for other attack types which can be defined as a preparatory stage attack. This paper seeks to identify and inject vulnerability.

A NoSQL database structure, also known as a "Non-Relational" or "Not only SQL," is a data storage and database architecture technique for very broad collections of distributed data and real-time web applications. A NoSQL database system is also a common knowledge recovery data storage since it provides increased scaling, availability and quicker access to data as compared to conventional link RDBMS. It is predictable what RDBMS data requires, since its data is stored in structured tables by specifying the relation between columns. Data does not need to be saved in a hierarchical or fixed fashion in the NoSQL databases. Where it comes to efficiency and real time control over accuracy, including indexing and recovering vast quantities of information, NoSQL databases are more appropriate than relational databases. NoSQL databases have been embraced recently by several small companies, as they are shifting their growing company data to clouds, thanks to their apparent advantages in improved performance, scalable and versatility. However, there are very few studies on the security of such NoSQL or NoSQL database systems. Though NoSQL databases have numerous storage privileges, the security problem of NoSQL databases has severely influenced the need for fast and convenient data access.

Boyu Hou et al. (2016), examines maturity of security mechanisms with code-level attack and protection issues for MongoDB, a standard NoSQL database scheme. JavaScript and PHP are used as preliminary tests for NoSQL injections. After demonstration of how a server-side JavaScript injection attack on a NoSQL database expose privacy of customers, two ways are addressed to avoid the occurrence of such security issues. Our analysis is expected to make database developers not only realize that NoSQL frameworks are not prioritized for stability, but also learn how to construct a security framework for the NoSQL applications of their organizations to prevent NoSQL injections.

The OWASP TOP 10, which has always been the focus for research on network security, has always been SQL injecting with the characteristics of great damage and rapid variance. SQL injection In order to detect unknown attacks using the current rule matches process, a machine-learning-based method of SQL injection detection is suggested. and the process for SQL extraction, **Zhuang Chen et al. (2018)** analyses f finally, word2vec is chosen to process the HTTP request text data and can effectively reflect the attack-load-enclosed SQL injection functionality. This research reveals that this approach successfully addresses the issue of SQL injection into the mutation and the high rate of leakage of the rule match. Training and classification with SVM algorithm for processed samples Compared with classifying effects of statistical functions, this SQL injection classification model has a higher recognition rate.

In this article, **Hong Ma et al. (2017)** present an approach to detection using a parse tree based on the semantic structure analysis. Based on this strategy, we rely on MongoDB to suggest a complex DND web environment for NoSQL attack detection. No access or modification of source code, rewrite of source codes with additional libraries, or complex assisted devices is needed. Finally, laboratory studies have shown that the concentrations of DND are high, false positive and poor.

Md Rafid Ul Islam et al (2019) build a NoSQL injection detection tool by monitored education. Our created NoSQL injection training data set is the first in our knowledge. We create essential features manually and use different supervised learning algorithms. As determined by 10-fold cross-validation, our tool has reached 0.93 F2. Also, our NoSQL tool, NoSQLMap, is used and finds that our tool exceeds the detection rate by 36,25% for Sqreen, the only NoSQL injector detection tool accessible. It is also shown that the proposed methodology is database-agnostic with injections on MongoDB and CouchDB.

Ahmed M. Eassa et al. (2017), A web application named "NoSQL Racket" test tool is available to detect NoSQL injection attacks. Basic concept of this tool is to validate the expected NoSQL database structure by contrasting the NoSQL statement structure in the code query statement and the runtime query statement statement (dynamic analysis).

However, we faced great problems, as is the situation with connection databases using SQL as a structured query language, no popular query languages are found to drive NoSQL databases. The proposed tool is tested on four web applications and its efficacy is matched with three separate proven testers, none of which detects any attacks by NoSQL Injection. The tested method is nevertheless capable of detecting attacks by NoSQL.

In recent years, the OBDA has gained interest by using ontology as a computational layer to provide access to vast quantities of data and to explore their capacity to describ domains and cope with data incompleteness. This is achieved by mapping data in the database with the ontology vocabulary. The initial OBDA research focused on data contained in relation databases. The use of OBDA to NoSQL databases is being expanded in recent studies. In this article, the new approach to OBDA with document-oriented NoSQL databases is presented by **Thiago H. D. Araujos et al (2017)** In addition to the associated activities, our solution uses a portal with an extendable and scalable intermediate concept layer that provides links to various forms of database management systems. To test the method, we have used a real world application domain as a case study to incorporate a prototype for MongoDB.

RESEARCH METHODOLOGY

Proposed system architecture is shown in figure 1. The user request is forward to the query generator to generate the query command. The command is moves to authentication phase, Bypassing two factor authentication is the technique used to authenticate the user query commands by validating the Time based token called time based one time password and the user is valid then it moves to application authentication phase at the same time the user is not valid then it wont allow to use the application. By using data bridge the application data and the authentication data get validates with the help of classification technique. The query commands get classified by using Aho–Corasick matching methods. Malicious query are stored in the FirebaseDB, the input query can be matched with data base query. If the query get matched then the result should be malicious or it is not malicious query. The proposed system is efficient one because of using bypass authentication mechanism, mostly it prevent or block the malicious user query. Validate malicious values inputs. Validate input styles to predicted types in NoSQL databases as well.

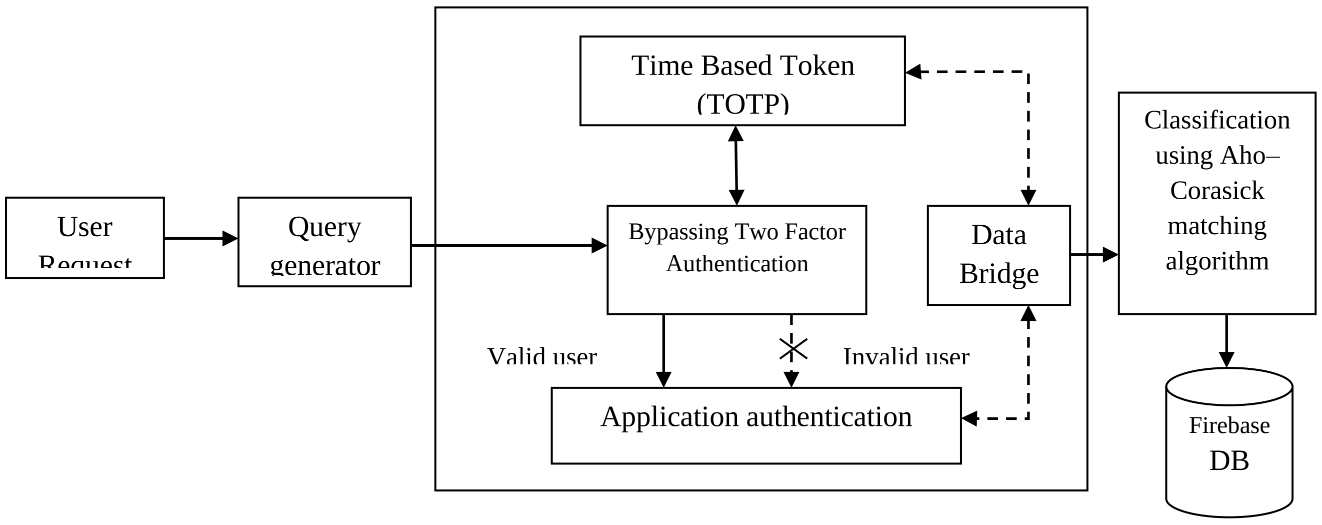

Time Based Token (TOTP)
User Request
Query generator
Bypassing Two Factor Authentication
Data Bridge
Classification using Aho–Corasick matching algorithm
Valid user
Invalid user
Application authentication
Firebase DB

Figure 1: Proposed architecture

NoSQL database storage not only removes the entire SQL language and increases stability, but also makes development easier. It relates to simplified query mechanisms and architectures in front of JavaScript. The implementation phase consists of an examination of the existing system, careful planning and the application of its constraints, methods for change and assessment of methods of transition. FirebaseDB can use the following form to check the login details:

db.users.find ({username: username, password: password});

NoSQL Injection is a weakness to authentication, which allows an assailant to insert code into the query running in the database. There are five kinds of NoSQL attacks, such as Tautology, Union requests, injections of JavaScript, piggybacked queries and violations of Cross origins. Five types of NoSQL attacks are discussed below:

1) **Tautology:** The attacker attempts to use a conditional question argument in the tautology attack that is always true. These attacks allow authentication and access systems to be bypassed by inserting code in conditional statements and creating always true expressions.

   db.accounts.find({username: username, password: password});

2) **Union Queries:** By expanding the results returned by the initial query, an attacker will extract information from the database using union dependent injection. Union queries are most often used to circumvent authentication pages and retrieve data.

   normal SQL statement + "semi-colon" + UNION SELECT

3) **JavaScript injections:** NoSQL databases have added a new class of vulnerabilities that enable JavaScript to be executed in the database sense. On the database engine, JavaScript allows for complex transactions and queries. Passing unprocessed user input to these queries may cause arbitrary JavaScript code to be injected, potentially resulting in unauthorized data extraction or modification.

   Javascript: alert("Welcome to yahoo.com");

4) **Piggybacked Queries:** The attacker injects more requests into an initial request for adding, modifying or deleting student profiles on the Grade Central website. Attackers use the conclusions to incorporate additional queries for executing the database, which may lead to arbitrary code execution by attackers, while interpreting escape sequences.

   normal SQL statement + ";" + INSERT (or UPDATE, DELETE, DROP)

5) **Stored Procedures:** When a normal SQL expression, i.e., SELECT, is created as a stored procedure, an attacker can insert another stored procedure to perform privilege escalation, to establish a service denial, and to execute remote commands as a replacement for a normal stored procedure. This is a typical type that uses a question limit (;) and the "SHUTDOWN" method of the attack shop:

   normal SQL statement + "; SHUTDOWN; "

Inject Code:

```
app.post('/user', function (req, res){
var query = {
username: req.body.username,
password: req.body.password
}
db.collection('users').find one(query, function (err, user){
console.log (user);
});
}) ;
```

Prevent NoSQL Injection

It is required to validate or escape the user's input correctly to avoid NoSQL injections. One of the first and fundamental steps is to validate user data, and validation of the intended form obtained in the application with regard to the following principles:

1) Validate the data duration and sort.

2) validate and sanitize the information for a specified sort (i.e. type casting).

Classification using Aho–Corasick multiple keyword matching algorithm

One of these classic algorithms is Aho–Corasick algorithm[2]. It is thought that during a pre-computing step of the algorithm, a finite automaton is constructed using a series of keywords and matches auto-scan the NoSQL query declaration to read every noSQL query character exactly once and to take constant time for each character read. AC methods uses a sophistication of a pattern which attempts to store set of anomaly keywords. The Aho–Corasick pseudo code is given below for a number of keywords corresponding to the algorithm,

```
Procedure AC(y,n,$I_0$)
INPUT: y←array of m bytes representing text input//SQL
Query Statement
1:UserVerification(validate)
Input password ≠null
Check the password (result = true)
Then
Generate OTP password
Else
Result = false
Ignore the user
Request Input query
n← integer shows text length //SQL Query Length
$I_0$←initial state (first character in pattern)
2: State ← $I_0$
3: For i = 1 to n do
4: While g (State, y[i] = = fail) do
5: State ← f (State)
6: End While
7: State ← g(State,.y[i])
8: If o(State)≠Φ then
9: Output i
10: Else
11: Output Φ
12: End If
13: End for
14: End Procedure
```

Bypass two factor Authentication

Two factor authentications are a way to use a mobile computer as an internet portal authenticator. While most companies regard it as a safe way to authenticate their users on their websites, it can be circumvented by two ways. The methods used to circumvent two authentication factors are built on abuse of the architecture and execution, which web application managers frequently fail to examine to allow attackers to compromise user data. The two-factor authentication mechanisms are defective from the architecture aspect to their execution. This scheme was initially developed to enhance security of online portal consumers and users. However, from time to time, two factor bypasses can develop and various methods can be used in these attacks. The only way is to retain the identification of one's own method to verify two factor authentications such that the mechanism of the assailant is easily predicted.

Time-Based Token (TOTP): An OTP method automatically produces time-based tokens with a static random key value and a dynamic time value. Time-based token is only available for 30 or 60 seconds for a given period of time.

NoSQL Database used:

The best examples of the NoSQL databases are FirebaseDB, MongoDB, CouchDB and Amazon DocumentDB. Open access and document-oriented databases are NoSQL databases. In scalable databases all databases are considered to provide high performance and availability. The main benefit of NoSQL is that, unlike RDBMS databases, all of the documents have their own schema, in which columns must be used compulsorily in each row of a chart. The short description of the firebase database is discussed below in this article.

FirebaseDB(NoSQL Database)

A cloud-hosted service is the Firebase Realtime database. Data will be saved to all linked clients as JSON and synchronized in real-time. All your customers share one Realtime database instance while you create inter-platform Apps using our iOS, Android and JavaScript SDKs and receive updates of the latest data automatically.

Realtime's main capabilities, The Realtime Database uses data syncing rather queues from traditional HTTP applications—any linked computer receives the update in milliseconds, every time data changes. Every time data changes. Provide cooperation and interactive interactions without reflecting on network code.

A portable, expression-based vocabulary known as the Firebase Realtime Database Security Rules is used in this real-time database to describe how the data is organized and how it is readable or writte to. When combined with Firebase Authentication, developers will decide who can access and how to access what data.

In contrast to the relationship data base, the Realtime Database is a NoSQL database with different optimizations and functionalities. The Realtime Database API is intended for fast operations only. That enables you to build an outstanding experience in real time that will provide uncompromising help for millions of customers.

PERFORMANCE ANALYSIS

The performance analysis of proposed system is evaluated in JavaScript and Firebase database is used for banking dataset.The parameters to be considered for implementations are:

1. Accuracy - The number of documents properly identified by the classificator is referred to.
$$\mathrm{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$$
2. True Positive Rate (TP): It is number of positive examples which the classification model has precisely forecast.
3. False Positive Rate (FP): It refers to number of negative examples which the classification model has inaccurately predicted.
4. Precision **-** is fraction of retrieved instances that are correlated.
$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$
5. Recall**-** is fraction of correlated instances that are retrieved.
$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$
6. ROC curve: are bidimensional graphs commonly used to evaluate and compare the performance of classifiers. It is a very popular method to measure the accuracy of a classification model.

The figure 2 shows the login page of firebase database app. Once the user and password is matched then only the user get authenticate to access the data.

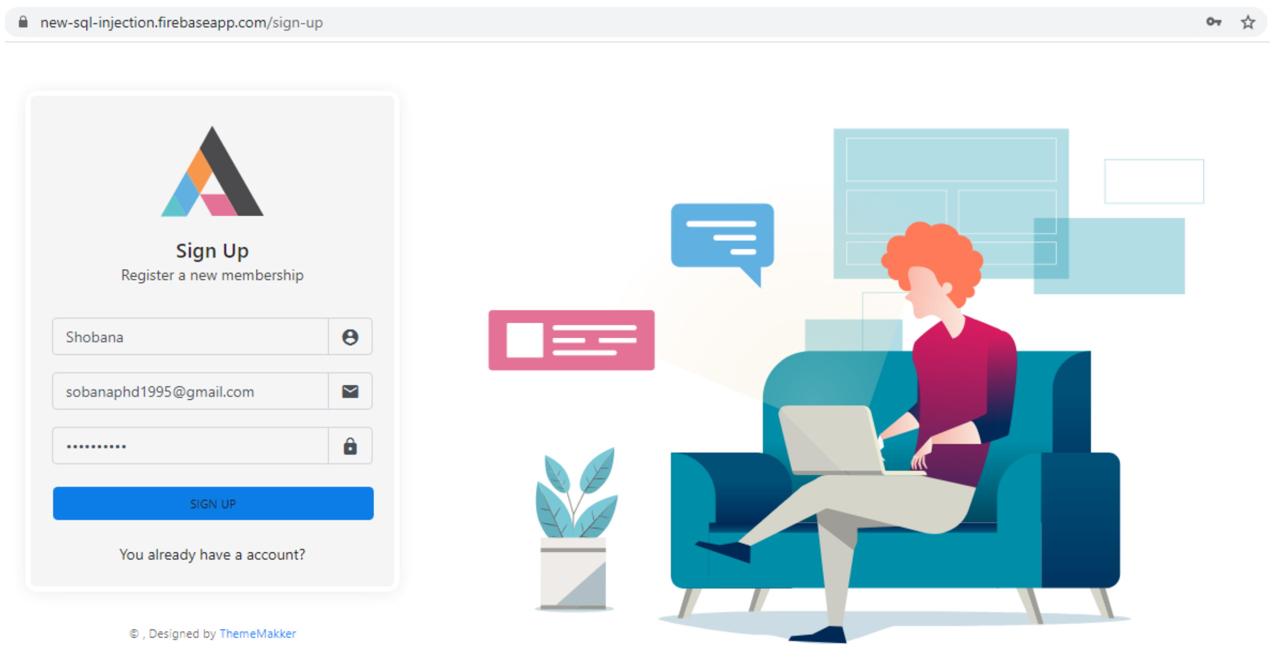

new-sql-injection.firebaseapp.com/sign-up
Sign Up
Register a new membership
Shobana
sobanaphd1995@gmail.com
••••••••••
SIGN UP
You already have a account?
© , Designed by ThemeMakker

Figure 2: Login the firebase app

Once the user is login is based on the verification of OTP token generation, the mail is forwarded to the user personal mail to get confirmation that the user is real one or not. The below figure 3 shows the verification mail is forwarded to the user.

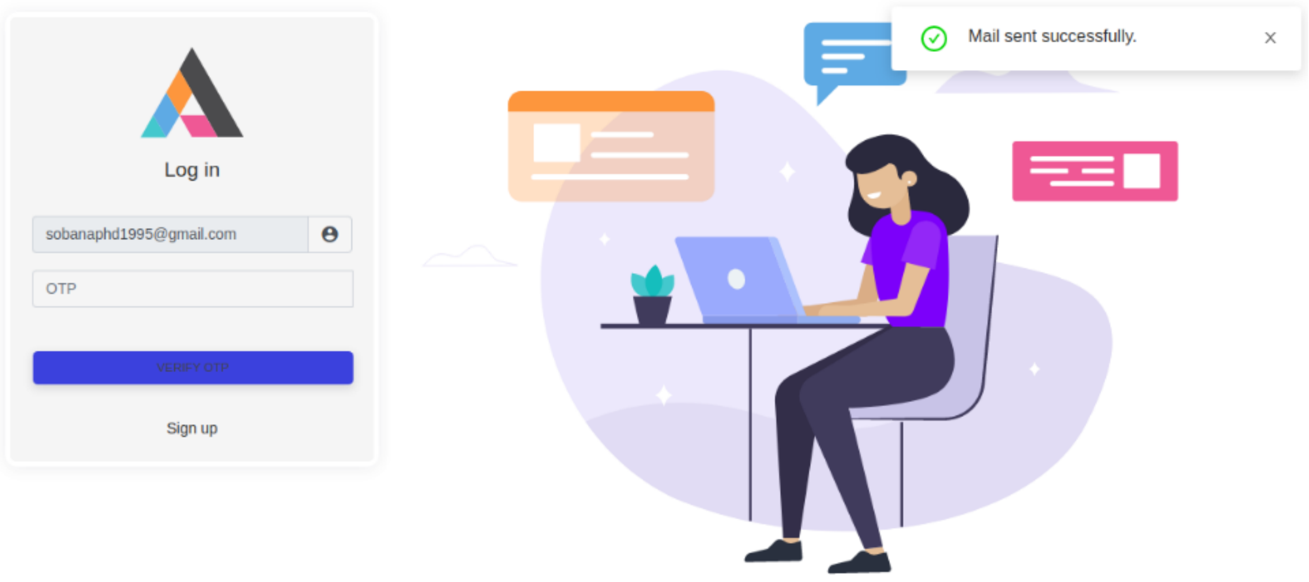
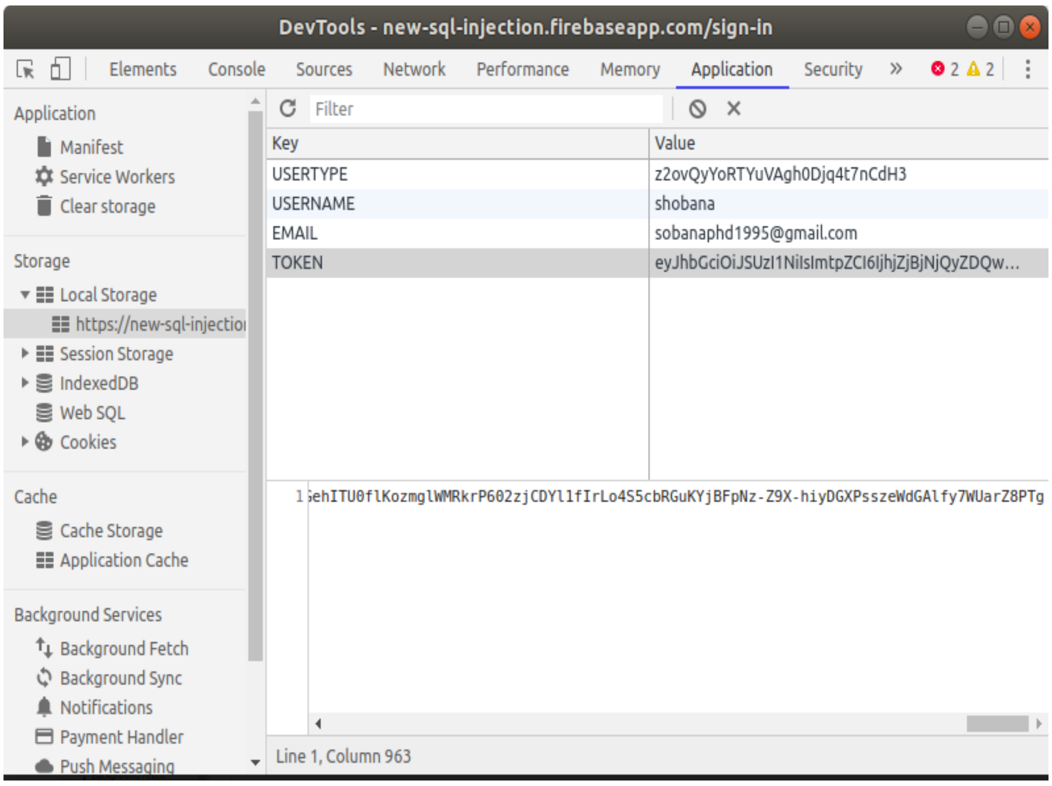


Figure 3: OTP mail verification to the user

Once the verification mail is forwarded to the user then the user validation phase begins the database user name, mail id and the token which is generated for verification is saved in the firebase Database which is shown in figure 4.



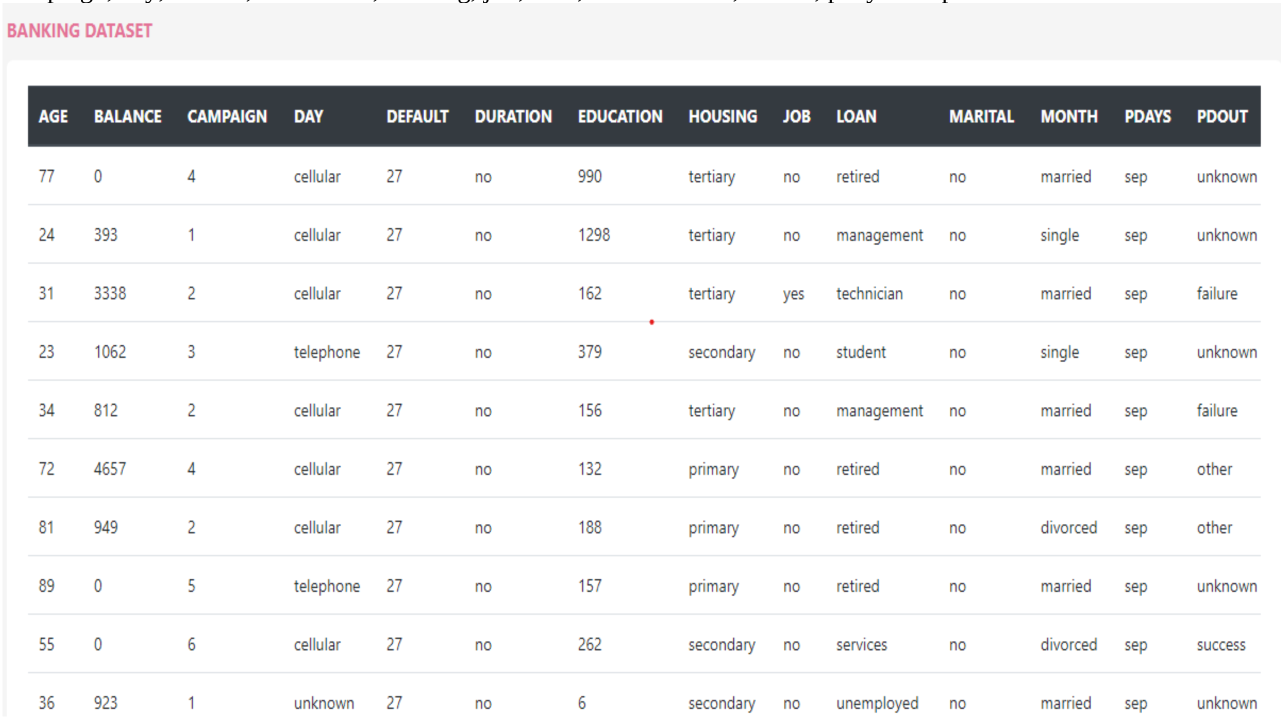
Key	Value
USERTYPE	z2ovQyYoRTYuVAgh0Djq4t7nCdH3
USERNAME	shobana
EMAIL	sobanaphd1995@gmail.com
TOKEN	eyJhbGciOiJSUzI1NiIsImtpZCI6IjhjZjBjNjQyZDQw...

```
1 ehITU0flKozmglWMRkrP602zjCDYl1fIrLo4S5cbRGuKYjBFpNz-Z9X-hiyDGXPsszeWdGAlfy7WUarZ8PTg
```

Line 1, Column 963

Figure 4: OTP Token generated for user validation

The below figure 5 shows the banking dataset which is used for proposed methodology. In this dataset, evaluations are presented in proposed work with more real life dataset would allow for comparing the performance. The attributes used for banking dataset are user personal details like name, age, account balance, campaign, day, default, educations, housing, job, loan, martial status, month, pdays and pout etc.


BANKING DATASET

AGE	BALANCE	CAMPAIGN	DAY	DEFAULT	DURATION	EDUCATION	HOUSING	JOB	LOAN	MARITAL	MONTH	PDAYS	PDOUT
77	0	4	cellular	27	no	990	tertiary	no	retired	no	married	sep	unknown
24	393	1	cellular	27	no	1298	tertiary	no	management	no	single	sep	unknown
31	3338	2	cellular	27	no	162	tertiary	yes	technician	no	married	sep	failure
23	1062	3	telephone	27	no	379	secondary	no	student	no	single	sep	unknown
34	812	2	cellular	27	no	156	tertiary	no	management	no	married	sep	failure
72	4657	4	cellular	27	no	132	primary	no	retired	no	married	sep	other
81	949	2	cellular	27	no	188	primary	no	retired	no	divorced	sep	other
89	0	5	telephone	27	no	157	primary	no	retired	no	married	sep	unknown
55	0	6	cellular	27	no	262	secondary	no	services	no	divorced	sep	success
36	923	1	unknown	27	no	6	secondary	no	unemployed	no	married	sep	unknown

Figure 5: Banking dataset used for FirebaseDB

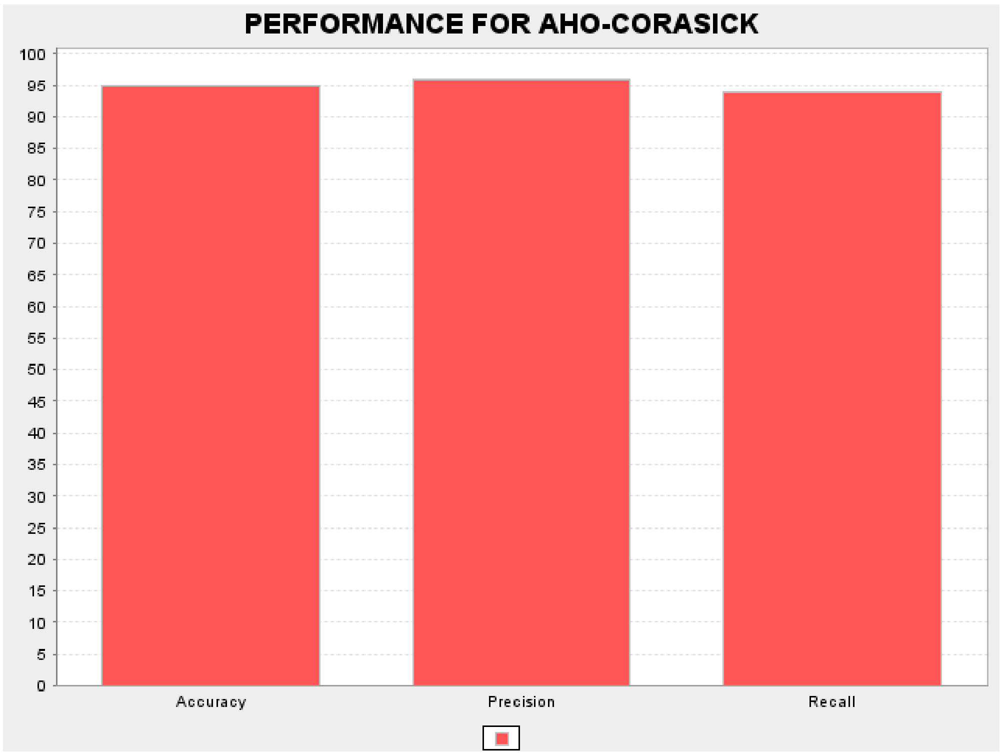


Figure 6: Performance analysis of proposed method

The above figure 6 shows the performance of proposed methodology, the parameter to be consider for evaluate the proposed method is accuracy, precision and recall.

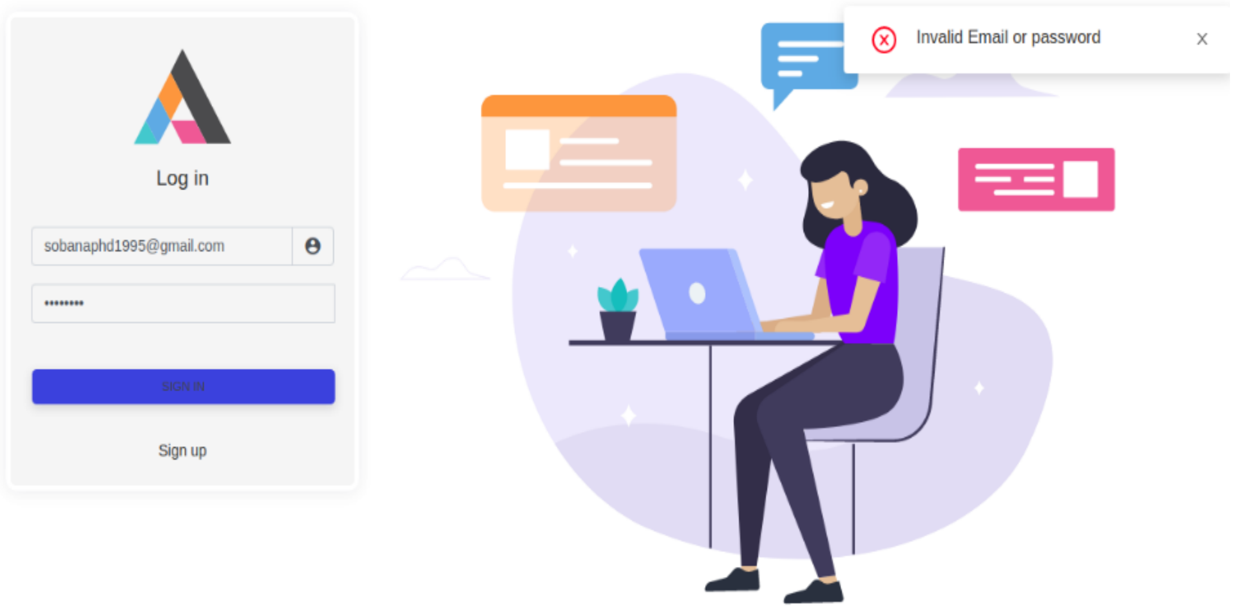


Figure 7: Malicious User verification

If the malicious user is present, the user validation phase rejects the malicious user which is shown in above figure 7. The user name and password is mismatch then it will show the email verification is invalid.

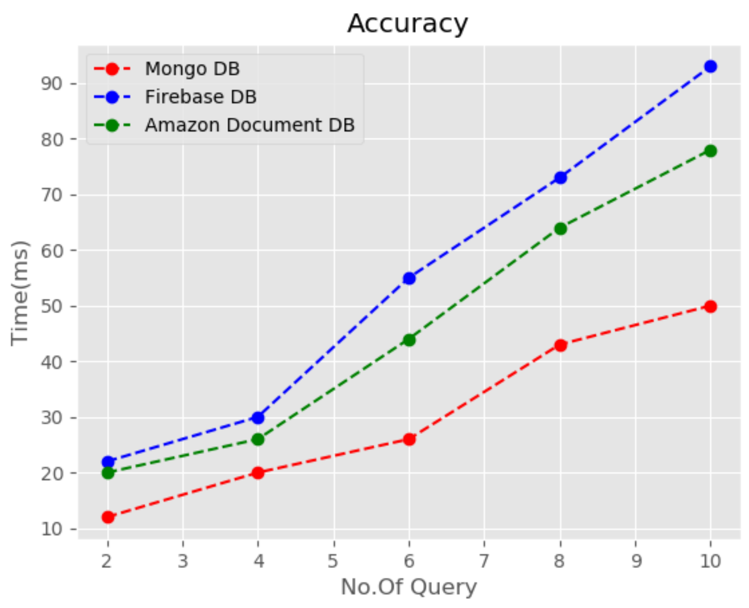


Figure 8: Accuracy of FirebaseDB and MongoDB and Amazon DocumentDB

The below figure 8 shows the accuracy calculation with FirebaseDB and realtime databases called MongoDB and Amazon DocumentDB. With classification accuracy of FirebaseDB achieves better results when compared to realtime databases of MongoDB and Amazon DocumentDB.

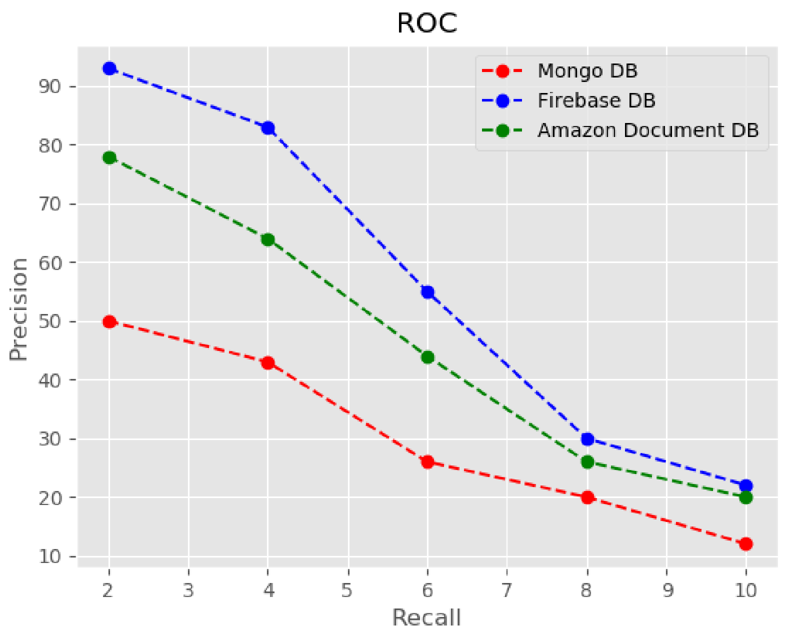


Figure 8: ROC curve of FirebaseDB and MongoDB and Amazon DocumentDB

The figure 9 shows the ROC curve of precision and recall which shows When observations are balanced within each group, ROC curves are sufficient, while accurate recalling curves are suitable for imbalanced data sets.

CONCLUSION

SQL Injection is one of the most frequent attacks on web applications. In this case, an attacker tries to use malicious manufactured input chains such that the SQL queries generated by the Web app vary from the developer's structure. This paper proposed technique is to protect the applications against NoSQL injection. In this article, the NoSQL Injection Attacks were attempt to characterize on the basis of web vulnerabilities. Also presented the classification results which is compared with realtime databases of MongoDB and Amazon DocumentDB

.

REFERENCE

1. N.S. Ali, A. Shibghatullah,(2016) “Protection Web Applications using Real-Time Technique to Detect Structured Query Language Injection Attacks”, International Journal of Computer Applications (IJCA), Volume 149, paperNo:6, September.
2. Hou, B., Qian, K., Li, L., Shi, Y., Tao, L., & Liu, J. (2016). MongoDB NoSQL Injection Analysis and Detection. 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud). doi:10.1109/cscloud.2016.57.
3. https://www.mongodb.com/
4. https://securityintelligence.com/does-nosql-equal-no-injection/
5. Zhuang Chen, Min Guo, Lin zhou (2018). Research on SQL injection detection technology based on SVM. MATEC Web of Conferences 173, pp.1-5.
6. Hong Ma, Tsu-Yang Wu, Min Chen, Rong-Hua Yang, and Jeng-Shyang Pan, (2017) “A Parse Tree-Based NoSQL Injection Attacks Detection Mechanism”, Journal of Information Hiding and Multimedia Signal Processing, Volume 8, Number 4, July.
7. Md Rafid Ul Islam ; Md. Saiful Islam ; Zakaria Ahmed ; Anindya Iqbal ; Rifat Shahriy, (2019) “Automatic Detection of NoSQL Injection Using Supervised Learning”, IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)
8. Ahmed M. Eassa, Hazem M. El-Bakry, Omar H. Al-Tarawneh, Ahmed S. Salama,(2017) “NoSQL Racket: A Testing Tool for Detecting NoSQL Injection Attacks in Web Applications”, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 8, No. 11.
9. Araujo, T. H. D, Agena, B. T., Braghetto, K. R., & Wassermann, R. (2017). OntoMongo-Ontology-Based Data Access for NoSQL. ONTOBRAS, 1908, 55-66.
10. https://docs.couchdb.org/en/latest/config/auth.html#couch_httpd_auth/require_valid_user
11. Shahmeer Amir,Four Methods to Bypass two factor Authentication, Jul 15, 2017 .
12. B. Indrani& E. Ramaraj, “X – LOG Authentication Technique To Prevent Sql Injection Attacks”, International Journal of Information Technology and Knowledge Management January-June 2011, Volume 4, No. 1, pp. 323-328.
13. Ziyahan Albeniz, Using Session Puzzling to Bypass Two-Factor Authentication, Web Security Readings ,2019.