Online ISSN 3007-3197 Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 5 (2025)

Empowering Robust Security Measures in Node.js–Based REST APIs by JWT Tokens and Password Hashing: Safeguarding Cyber World

[1*]Khalid Hamid, [2]Muhammad Danish, [3]Ayyan Asif, [4]Younus Khan, [5]Muhammad Danish, [6]Muhammad Waseem Iqbal, [7]Umair Ali. [8]Muhammad Ibrar

Article Details

Khalid Hamid*
Department of Computer Science and Information Technology, Superior University Lahore, Lahore, 54000, Pakistan. Corresponding Author Email: khalid6140@gmail.com
Muhammad Danish
Department of Computer Science, University of New Mexico, NM; mdanish@unm.edu
Ayyan Asif
Master of Science in Data Analytics (Stem) Department of Computer Science New Mexico State University, Las Cruces, NM; ayyanasif07@gmail.com
Younus Khan
Department of Computer and Mathematical Sciences New Mexico Highlands University, Las Vegas, NM; Ynyuskhan464@gmail.com
Muhammad Danish
Department of Computer Science and Information Technology, Superior University Lahore, Lahore, 54000, Pakistan; bcsm-f21-269@superior.edu.pk
Muhammad Waseem Iqbal
Department of Software Engineering, Superior University Lahore, 54000, Pakistan; waseem.iqbal@superior.edu.pk
Umair Ali
Department of Computer Science and Information Technology, Superior University Lahore, 54000, Pakistan; bcsm-f21-259@superior.edu.pk
Muhammad Ibrar
Department of Computer and Mathematical Sciences New Mexico Highlands University, Las Vegas, NM; Mibrar@live.nmhu.edu

ABSTRACT

This research analyzes the practical implementation of the security measures on the node. js-based REST APIs. This research paper studies authorization and authentication with the use of role-based access control, JWT tokens and password hashing. It also addresses the common vulnerabilities to the APIs like SQL Injection, Cross-site scripting attacks with the help of the parameterized queries and some of the input validations. The testing was conducted by the postman which demonstrates the good results that reduce the vulnerabilities of the SQL injection attacks and cross-site scripting attacks. This research paper also looks at the trade-offs between performance and security. The findings result in the best security measure to make REST APIs secure.

INTRODUCTION

APIs are vulnerable to SQL injection attacks, stealing the JWT tokens and manipulating them, and cross-site scripting attacks. Having the security framework available and some best practices, implementing the secure APIs remains a challenge, especially in dynamic runtime environments like Node.js. This paper gives a practical approach to secure REST APIs made in node.js [1][2]. It practically implements the JWT, Helmets, authorization and authentication in such a way that they provides a way to implement them securely and safely [3][4]. The API was tested using the postman which clears almost every mentioned threat. Many studies are presented out there but this research provides a bridge between the theory and the practical implementation [5][6]. It also presents a trade-off between security and performance **[3]**[7]. The goal of this research is to analyze the effective measures in practical that need to be taken to make sure that API is safe and secure [][8].

LITERATURE REVIEW

This paper by Sattam J Alharbi summarizes the findings and provides suggestions to make the rest of the APIs more secure by analyzing the previous research papers. This paper researches the common vulnerabilities of the rest of the APIs by looking at the previous research papers and identifies how to mitigate them. It explains all parts of the APIs that need to be made secure in aspects of the testing and authorization. It provides concise details on making the APIs more secure **[9].**

This paper by Arvinda A Kumar explains the key security measures and looks at how token-based authentication can be made secure and looks at pass lib, a password or text hashing library**[2]**. This paper researches the implementation of effective security measures by looking at how JWT authentication can be done securely, how Cors policy should be set, and how HTTP error handling can be done. It looks at the previous research studies and provides a structured way to implement security on the different and vulnerable aspects of the rest APIs. **[10].** This paper by Ankit Hansraj Yadav, Neeraj Jokhoo Yadav, and Om Bhupendra Singh simply highlights the importance of knowledge for the cyber-security and software engineering students about API security **[1]**. The paper researches the common vulnerabilities and potential risks and pressurizes the importance of knowing the security of the rest APIs. It looks at the API background, common API security threats and challenges to APIs security, and solutions to these challenges to security. It efficiently gives a map that why API security is the most important **[11].**

This paper by Cao Rongqiang, and Yangang Wang implements the techniques for

authorization and authentication in website development. It analyzes the microservices architecture of simple authentication and authorization. It implements the authorization and the authentication for the rest APIs in the SCE. It looks at the previous works to find a better and more efficient way to implement authorization and authentication in website development [12]. This paper by Rajesh Kotha provides instructions and best practices for API inquiries introduced in the REST standard for the software architecture [26]. It tells about the HAMC technique which allows the authentication process and maintains the secrecy, and integrity of the software. It provides pretty concise details about the HAMC technique used to implement successful authorization and authentication and how it can be used to maintain the security of the software in which they are used [13].

This paper by Silvia Llorente Viejo analyzes the security recommendations by different organizations for rest APIs and creates an app that explains all security measures for the rest API server. This paper researches the best encryption algorithms and password-hashing functions. It applies additional measures like CSRF tokens just by analyzing the recommendations of different organizations. It provides a detailed approach to secure rest API. This paper provides very good security measures for rest APIs [14].

This paper by Priyanka Gowda and Narayna Gowda analyzes the scalability and security best practices in the rest of API design. This paper approves or states the efficient practices in making the of rest the APIs to make it secure and scalable by using several publication analyses, standards and better practices. This paper gives a structure to design a rest API in such a way that it comes out secure and scalable. This paper serves as a map to make a secure API [15]. This paper by Dinh Toan Nguyen analyzes the latest security measures provided by different API services and also analyzes the Twitter and stripe services for security importance and analyzes the author's website and C# library for API call security[24]. This paper identifies the main problem of Stripe which is encrypting the private key before sending it to the server and also Stripe must change the code quality of JavaScript to secure the public key. It provides a theoretical and practical explanation of web security and API service [25] [16].

This paper by A Sanjana, M. Anusha expounds on the methods that an attacker can use to expose the flaws in the rest APIs. This paper gives some foundational security rules on how to implement the security in the rest APIs. This paper also looks at how cloud resources can be used by using the API requests. This paper provides a handful of information on how rest APIs

can test and detect security violations using active property checkers **[17].** This paper by Pawan Kumar Bhat focuses on making a model or prototype of making an API using the C#, managing the authentication and authorization in such a way that users can insert, update, and delete the data in the API. This paper uses the O. Auth 2.0 for implementing the authorization and authentication. The database is made using a Microsoft SQL server. It puts the demand based on how the web APIs can be made perfectly without any kind of security breach and flaw **[18].**

METHODOLOGY

The methods of making rest API secure contain multiple layers of security. It consists of validating input, encrypting the sensitive data authentication and authorization processes [19]. Everything mentioned above is explained below.

API ARCHITECTURE

The goal of this stage is to identify the blueprint of the API means learning about the API endpoints and identifying which endpoints need secure access for the user so they handle more secure data [20].

IDENTIFY

So, I am going to be making an API for an e-commerce site this API will contain product information, user information, reviews on the products, cart information checkout information. So, the app will contain the endpoint written below [21].

- For Products(/api/v1/products)
- For Users(/api/v1/users)
- For Reviews(/api/v1/reviews)
- For Cart(/api/v1/carts)
- For Checkout(/api/v1/checkouts)

Now it is noticeable that some of these endpoints handle sensitive data therefore these endpoints require strict security access. Let's identify which of the endpoints require the stricter security rules by giving the public and private keywords and by public means it requires any kind of security measures among the above-mentioned security measures [22].

- For Products(/api/v1/products)**:** Some of the endpoints in this endpoint will be **private** like deleting the product, adding the products. All others will be **public** like getting the products.

- For Users(/api/v1/users)**:** All of its endpoints will be private because the data of the users will be **private.**
- For Reviews(/api/v1/reviews): All of these review endpoints will be **public** except adding the review and deleting the review.
- For Cart(/api/v1/carts)**:** All of these cart endpoints will be **private**. It will contain some kind of security.
- For Checkouts(/api/v1/checkouts)**:** All of these cart endpoints will be **private**. It will contain some kind of security.

AUTHENTICATION AND AUTHORIZATION PLANS

First of all, let's talk about **authentications**: authentication means to verify the user that he is the one who he claims to be. So, the rest APIs are stateless and Jason Web Token is also stateless so it will be efficient to use the JWT token for the authorization. This authorization will be performed on the /user/login endpoint. When a user signs up or login it is given JWT in response that contains some information about the user that helps in knowing that the user has logged in. Now this JWT token will be sent by the user in every request in the authorization header.

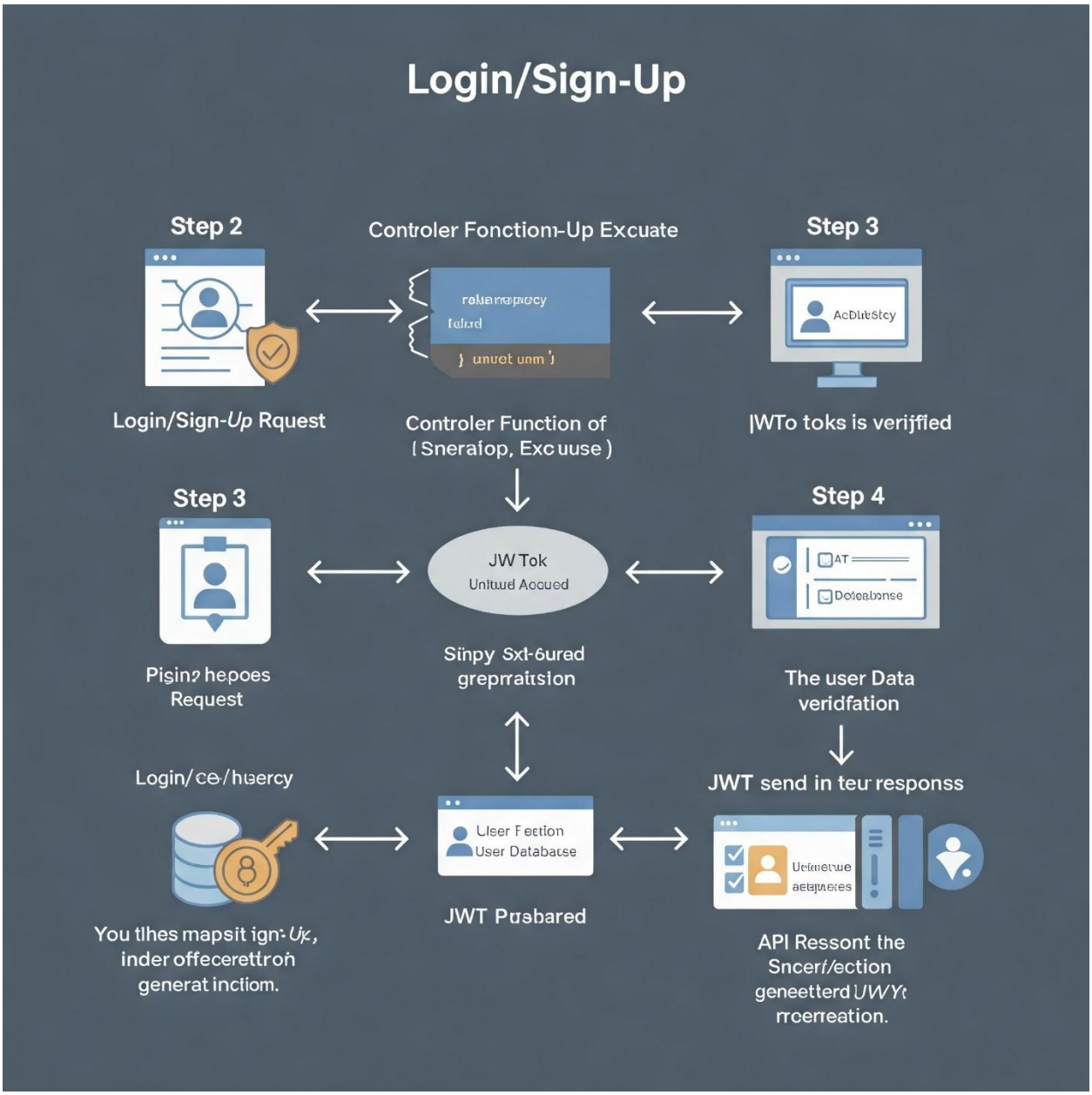

FIGURE 1: AUTHENTICATION THROUGH JWT

Secondly let's talk about **authorization**: authorization determines what processes or actions an authenticated user can perform. There are many ways to perform the authorization that help in performing the role-based access controls. Some people perform role-based access control by storing them in the JWT and extracting them when the user makes any request the JWT token is received and then from its payload the role (like admin, user, etc.) then one can make middleware function to respond in term of the role. [14]

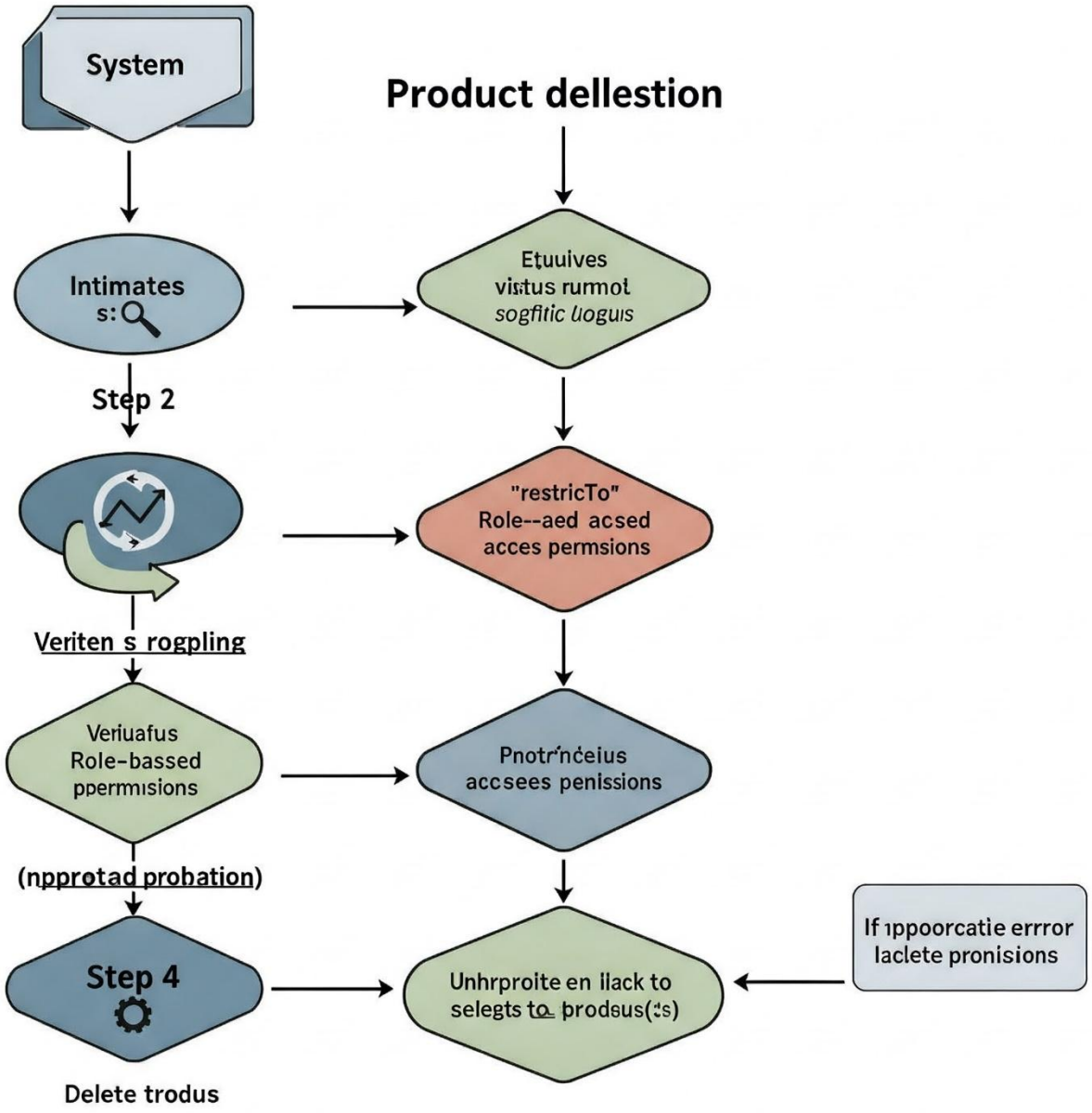

FIGURE 2: AUTHORIZATION

Then there is another method to implement the role-based access that the user enters its roles when entering its information, then we make a role-based access middleware function that can then be used to give the role-based access control like I have done in my e-commerce API and written as follows.

```
exports.restrictTo = (...roles) => {
  return (req, res, next) => {
```

```
    console.log(roles);
    if (roles.includes(req.user.role)) {
      const error = new Error(
        "You do not have the permission to perform this action"
      );
      console.log(error);
      return next(error);
    }
    next();
  };
};
```

These roles will be passed into the middleware while defining the routes:

```
authController.restrictTo("user")
```

If it is being wondered where this req.user comes from it comes from the middleware function protect:

```
exports.protect = async (req, res, next) => {
  let token = "";
  try {
    if (
      req.headers.authorization &&
      req.headers.authorization.startsWith("Bearer")
    ) {
      token = req.headers.authorization.split(" ")[1];
    } else if (req.cookies.jwt) {
      token = req.cookies.jwt;
    }
    if (!token) {
      throw new Error("You are not logged in!");
    }
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    // console.log(decoded);
    const user = await User.findById(decoded.id);
```

```
  if (!user) {
    throw new Error("The token belonging to the user not longer exist");
  }
  if (user.changedPasswordAfter(decoded.iat)) {
    throw new Error("User recently changed the password");
  }
  req.user = user;

  return next();
 } catch (error) {
  next(error);
 }
};
```

SQL INJECTION ATTACKS PROTECTIONS

These are the kinds of attacks in which the attacker inputs some kind of query into the request body and when that query is executed it produces some kind of unwanted results. One of the ways that ca prevent is by sanitizing the data. Now these attacks can be prevented in many ways but one of the ways is to use a third-party library that helps in doing this and it is specially built for node.js [13].

```
const mongoSanitize = require('express-mongo-sanitize');
const xss = require('xss-clean');
// Data sanitization against NoSQL query injection
app.use(mongoSanitize());

// Data sanitization against XSS
app.use(xss())
```

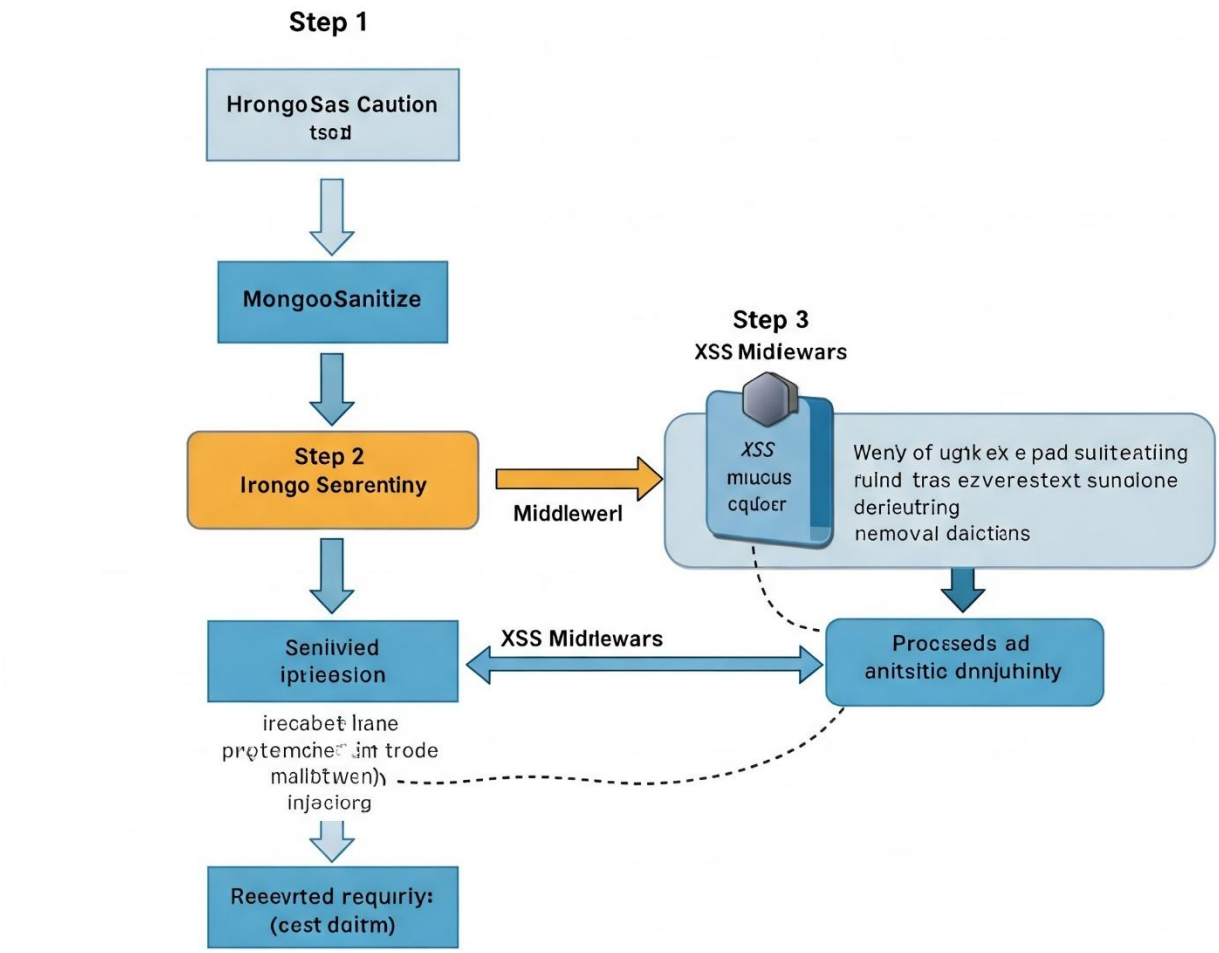

FIGURE 3: SQL INJECTION PREVENTION

HEADERS PROTECTION

Usually, the attacker attacks the API using the headers: for this, a third-party module can be used in the node. Js-based API is called **Helmet**. It also helps in securing the application from cross-site scripting attacks and as well as helps to prevent attackers not knowing about the known vulnerabilities. These help a person secure the node JS API by setting several HTTP headers **[12].** You can just implement it like below:

First, you have to import it in your main app.js file and before that install the helmet library using the NPM.

```
const helmet = require('helmet');
// Set security HTTP headers
app.use(helmet());
```

RATE LIMITING

It is a technique to control the no of requests coming from the network or going from it. This is important for better workflow and reduces the risks of attacks and the server never overloads [11]. This can be done by installing the **express-rate-limit** third-party library.

```
const rateLimit = require('express-rate-limit');
```

One can use the rate limiter like this:

```
const limiter = rateLimit({
  max: 100,
  windowMs: 60 * 60 * 1000,
  message: 'Too many requests from this IP, please try again in an hour!'
});
app.use('/api', limiter);
```

RESULT AND DISCUSSION

Hence, after applying all the security measures practically and theoretically the study successfully made our API secure from different threats. Like our express-rate-limit blocked almost all the unnecessary requests from the clients, the XSS third-party library also helped prevent the SQL injection in our requests along with mongoSanitize which helped in sanitizing the SQL injections also, then we learned about the use of the helmet library that we used as the middleware in App.js file to make the API secures from the cross-site scripting attacks and also set the HTTP header to prevent unknown vulnerabilities to expose to the attackers and we also learned about the how we can implement the role-based access control in our node.js rest API. All of this knowledge gave us the practical as well as the theoretical aspects of the security measures and implementation in the rest of API made using the node.js

CONCLUSION

In summary, this research has presented a detailed overview of the numerous security solutions that can be used to secure Node.js-based REST APIs. By employing authentication and authorization practices like JWT tokens and role-based access control, the study has developed a sound framework for safe access to sensitive resources. Using the JWT token for stateless authentication and introducing role-based access control middleware, the study can define the roles of the users and ensure proper access restrictions. In addition, the study discussed the protection against common weaknesses like SQL Injection and Cross-Site Scripting (XSS) attacks. With strategies like data sanitization through the use of express-mongo-sanitize and

XSS-clean and valid input checks, we made it possible to minimize the threats of such an attack. Securing headers in place with the helmet library additionally strengthened the defenses as a whole, making sure that our API is less exposed to typical web application security concerns. The use of rate-limiting mechanisms was also an efficient mechanism for regulating the number of requests, safeguarding against denial-of-service attacks, and promoting the stability and performance of the API. The results from the testing and implementation of the security measures reaffirm their success in protecting against vulnerabilities. The practical usage of these methods, complemented by theoretical reasoning, goes into a complete security plan that diminishes the attack risk substantially. The trade-offs between efficiency and security were also taken into account, and the steps taken maintain a balanced approach, ensuring security as well as efficiency. Therefore, this study emphasizes the need to integrate sound security practices into the design of REST APIs to make them resistant to typical attacks while ensuring optimal performance and user experience.

REFERENCES

1. Azhar, Z. (2024). Blockchain as a Catalyst for Green and Digital HR Transformation: Strategies for Sustainable Workforce Management. Open Access Library Journal, 11(9), 1-22
2. Azhar, Z. (2024). The Role of Chatbots in Enhancing Job Seekers' and Employee Experience: A Case Study on CV Warehouse. The Journal of Social Sciences Research, 10(4), 23-35.
3. Azhar, Z., Nawaz, H., Malik, A. S., & Zaidi, M. H. (2022). Strategic Impact of Cloud Computing on HR Transformation. International Journal of Social Science & Entrepreneurship, 2(2), 546–576. https://doi.org/10.58661/ijsse.v4i4.336
4. Azhar, Z., & Imran, M. (2024). Ethical Considerations in the Adoption of Artificial Intelligence in Human Resource Management: A Comprehensive Review. Journal of Emerging Technologies and Innovative Research (JETIR), 11(8).
5. Danish, M.; Shahid, S.; Ghafar, A.; Hamid, K.; Ali, N.; Ghani, A.; Ibrar, M.; Mandan, S. Security of Next-Generation Networks: A Hybrid Approach Using ML-Algorithm and Game Theory with SDWSN. 2025, 3, 18–36, doi:10.63075/wdpwrr31
6. Hamid, K.; Iqbal, M.W.; Aqeel, M.; Liu, X.; Arif, M. Analysis of Techniques for Detection and Removal of Zero-Day Attacks (ZDA). In Proceedings of the Ubiquitous Security; Wang, G., Choo, K.-K.R., Wu, J., Damiani, E., Eds.; Springer Nature:

Singapore, 2023; pp. 248–262

7. Hamid, K.; Iqbal, M.W.; Aqeel, M.; Rana, T.A.; Arif, M. Cyber Security: Analysis for Detection and Removal of Zero-Day Attacks (ZDA). In Artificial Intelligence & Blockchain in Cyber Physical Systems; CRC Press, 2023 ISBN 978-1-00-319030-1.
8. Khaliq, K.; Rahim, N.; Hamid, K.; Ibrar, M.; Ahmad, U.; Ullah, M. Ransomware Attacks: Tools and Techniques for Detection; 2024; p. 5;.
9. Ibrar, M.; Riaz, S.; Khan, Y.; Asif, A.; Hamid, K.; Iqbal, M.W.; Asim, M. Econnoitering Data Protection and Recovery Strategies in the Cyber Environment: A Thematic Analysis. International Journal for Electronic Crime Investigation 2024, 8, doi:10.54692/ijeci.2024.0804216.
10. Malik, N.; Delshadi, A.; Ibrar, M.; Hamid, K.; Aamir, M.; Ahmed, F.; Ahmad, G. Behavior and Characteristics of Ransomware - A Survey; 2024; p. 05
11. zafar, zishan; Hamid, K.; Kafayat, M.; Iqbal, M. waseem; Nazir, Z.; Ghani, A. AI-Based Cryptographical Framework Empowered Network Security. Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition) 2023, 42, 497–510, doi:10.17605/OSF.IO/W69VT.
12. Riaz, S. Khan, Y.; et al. Software Development Empowered and Secured by Integrating A DevSecOps Design. Journal of Computing & Biomedical Informatics 02 (2025) doi:10.56979/802/2025.
13. Alharbi, Sattam J., Tarek Moulah. "API Security Testing: The Challenges of Security Testing for Restful APIs." *ResearchGate*, Qassim University, 2023, www.researchgate.net/publication/371174422_API_Security_Testing_The_Challenges_of_Security_Testing_for_Restful_APIs. Accessed 30 Nov. 2024.
14. Kumar, Aravinda A., and Divya TL. "Security Measures Implemented in RESTful API Development." *ResearchGate*, 2023, www.researchgate.net/publication/384461158_Security_measures_implemented_in_RESTful_API_Development. Accessed 30 Nov. 2024.
15. Yadav, Ankit Hansraj, Neeraj Jokhoo Yadav, and Om Bhupendra Singh. "Development of RESTful WebAPI Using Token-Based OAuth 2.0 Authorization." *International Journal of Research Publication and Reviews*, vol. 5, no. 6, June 2022, pp. 257–260, www.ijrpr.com/uploads/V5ISSUE6/IJRPR30027.pdf. Accessed 30 Nov. 2024.
16. Hoang, Tan, et al. "Authentication and Authorization for RESTful WEB API in

Scientific Computing Environment." *ResearchGate*, Nov. 2019, www.researchgate.net/publication/337427360_Authentication_and_Authorization_for_RESTful_WEB_API_in_Scientific_Computing_Environment. Accessed 30 Nov. 2024.

17. Kotha, Rajesh. "Architecting Secure REST APIs with Authentication and Authorization Approaches for Web Services." *Online Scientific Research*, 2020, https://onlinescientificresearch.com/articles/architecting-secure-rest-apis-with-authentication-and-authorization-approaches-for-web-services.pdf. Accessed 30 Nov. 2024.
18. Llorente Viejo, Silvia. *Securing a REST API Server*. Master's thesis, Universitat Politècnica de Catalunya, 2021, https://upcommons.upc.edu/bitstream/handle/2117/377463/TFM___Memoria.pdf;jsessionid=DCC35F0261A6B7F76B9938336EC44B65?sequence=2. Accessed 30 Nov. 2024.
19. Gowda, Priyanka, and Ashwanth Narayana Gowda. "Best Practices in REST API Design for Enhanced Scalability and Security." *Universal Research Forum Journals*, vol. 3, no. 2, 2023, https://urfjournals.org/open-access/best-practices-in-rest-api-design-for-enhanced-scalability-and-security.pdf. Accessed 30 Nov. 2024
20. Nguyen, Dinh Toan. *Web Security: Security Methodology for Integrated Website using RESTful Web Services*. Thesis, Tampere University, 2016, https://trepo.tuni.fi/bitstream/handle/10024/99425/GRADU-1466664236.pdf?sequence=1&isAllowed=y. Accessed 30 Nov. 2024.
21. Sanjana, A., M. Anusha, G. Pravallika, and Mrs. S. Radhika. "REST APIs Cloud Service Security Checks." *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*, vol. 8, no. 6, 2020, www.ijraset.com/research-paper/rest-apis-cloud-service-security-checks. Accessed 30 Nov. 2024.
22. Bhat, Pawan Kumar, and Rajnish Kansal. "Development of RESTful WebAPI Using Token-Based OAuth 2.0 Authorization." *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 10, Oct. 2020, www.ijert.org/research/development-of-restful-webapi-using-token-based-oauth-20-authorization-IJERTV9IS100048.pdf. Accessed 30 Nov. 2024.
23. Barahona, Jorge. "Rate Limiting in Node.js." *LogRocket Blog*, 16 Feb. 2022, https://blog.logrocket.com/rate-limiting-node-js/#what-rate-limiting. Accessed 30

Nov. 2024.

24. Brewster, Owen. "Using Helmet in Node.js to Secure Your Application." *LogRocket Blog*, 21 Apr. 2022, https://blog.logrocket.com/using-helmet-node-js-secure-application/#securing-express-with-helmet.

25. "express-mongo-sanitize." *npm*, https://www.npmjs.com/package/express-mongo-sanitize. Accessed 30 Nov. 2024.

26. Lorique. "How & Why: REST API Authorization with JWT in Node.js." *Medium*, 26 May 2022, https://medium.com/@lorique/how-why-rest-api-authorization-with-jwt-in-nodejs-71d99af923c8. Accessed 30 Nov. 2024.