Ecrypt Pay

A Secure Software application for Payment Gateway.

Mohammad Khaja Moinuddin,
Tejo Verma Sangaraju,
Rohith Byanagari,
Sowmya Reddy Adupala
Group- 26, 5565
Department of Computer Science
University of North Texas
Denton, Texas, United States of America

***Abstract*— This final paper outlines the development and security analysis of our project, Ecrypt Pay, which is a secure software application designed for a payment gateway. Our main focus is on ensuring the security of transactions safeguarding user information and maintaining the integrity of the system. This report showcases features such as blockchain record-keeping real-time encryption and user education. These features are aimed at creating a platform that's both safe and efficient for conducting transactions.**

I. INTRODUCTION

Making sure that the security of this payment system is absolutely crucial. In a time where digital dangers are prevalent, it is vital to establish a platform that not only improves the ease of online transactions but also strengthens them against possible risks. This assessment of security explores the security components integrated into the system to ensure the safety of both businesses and customers.

A. Motivation for Security Analysis:

The changing world of transactions requires a strong and secure payment gateway. With the increasing prevalence of cyber threats, it is crucial to prioritize the protection of transactions. Ecrypt Pay addresses these concerns, by incorporating security measures enabling both businesses and consumers to engage in transactions

The motivation behind conducting security analysis is driven by our commitment to delivering a payment gateway that not only enhances the efficiency of transactions but also establishes a robust defense against potential risks. By subjecting Ecrypt Pay to security analysis our goal is to identify, evaluate, and resolve any vulnerabilities ensuring that the software operates at the level of security. This proactive approach demonstrates our dedication to creating an environment for users to build trust and reliability in the realm of transactions.

B. Obtaining a Copy of the Software:

Ecrypt Pay has been designed with openness and accessibility, in mind. To acquire a copy of the software users can follow a process that guarantees transparency and ease of access. Please visit our GitHub repository: https://github.com/Junaidsheik/Ecrypt-Pay.git

And check out this detailed video on YouTube: https://youtu.be/I3VUqwBRhBg

II. DESCRIPTION OF THE SOFTWARE

A. User-Friendly E-commerce Website Design:

The user interface design of Ecrypt Pay is an example of prioritizing the needs and preferences of users with an emphasis, on making online shopping easy and seamless. By incorporating technologies, the design specifically focuses on the following factors;

- **Koa Web Framework:** The decision to use the Koa Web framework showcases Ecrypt Pays dedication to efficient code execution and a simplified middleware approach. With Koa support, for async/await and generators handling code becomes easier resulting in improved responsiveness and overall performance of the platform.
- **Node JS:** Ecrypt Pay's decision to embrace Node.js, an open-source runtime environment, for server-side JavaScript, showcases their dedication to scalability and efficiency. Node.js, powered by the V8 JavaScript engine enables the development of network applications that can handle loads without blocking processes. This makes it an excellent choice for building an e-commerce platform.
- **React JS:** By incorporating React JS, a popular JavaScript library developed and maintained by Facebook Ecrypt Pay demonstrates its commitment to creating dynamic user interfaces. Reacts component-based architecture simplifies the development process. Allows for the creation of captivating web applications that align perfectly with Ecrypt Pay focus on providing user experiences.

B. Blockchain Integration with Smart Contracts:

Ecrypt Pay stands out by integrating technology into its architecture ensuring transparency and security in all transactions. The architectural design is meticulous. Follows industry practices:

- **ERC 20 Standards:** The integration of ERC 20 standards underscores Ecrypt Pays dedication to interoperability within the Ethereum blockchain ecosystem. These standards establish guidelines for creating tokens while also serving as a blueprint for contracts managing digital assets like cryptocurrencies and utility tokens.

- **Truffle Development Framework**: By adopting the Truffle development framework Ecrypt Pay solidifies its commitment, to Ethereum-based application development. Truffle provides a set of tools that streamline the development process while maintaining standards of quality.

- Truffle offers a range of tools and a development environment that make it easier to create, test, and deploy contracts and decentralized applications (DApps).

C. Cryptocurrency Payment Gateway:

Ecrypt Pay takes payment gateways to the level by creating a custom gateway, for cryptocurrency transactions. This gateway is meticulously designed, incorporating cutting-edge technologies:

- **Solidity**: The utilization of the Solidity programming language, by Ecrypt Pay demonstrates their commitment to developing contracts on the Ethereum blockchain. This level and statically typed language allows for the creation of decentralized applications (DApps) and custom tokens within the Ethereum ecosystem.
- **DAI Stable Coin:** Ecrypt Pay's decision to incorporate DAI, a stable coin built on the Ethereum blockchain showcases their dedication to minimizing cryptocurrency volatility. By design DAI maintains a value making it an excellent choice for users who wish to avoid the price fluctuations associated with cryptocurrencies.
- **MongoDB Integration:** The integration of MongoDB into Ecrypt Pays payment gateway is crucial and strategic. MongoDB is a used open-source NoSQL database that ensures storage of user authentication details account information, item specifics and real time payment data. This integration guarantees recording and retrieval of transaction information.

D. Database Management (MongoDB):

MongoDB plays a role in Ecrypt Pay's data management strategy. In terms of ensuring storage, it serves as a repository for essential user data such as authentication credentials account information and details about available items, for purchase.

The MongoDB NoSQL architecture offers flexibility in handling a range of data types making it well-suited for managing transactional data that is constantly changing.

- **Real-time Payment Information:** MongoDB plays a role, in recording and retrieving up, to date payment information. Its document-oriented structure enables storage and retrieval of data ensuring smooth and transparent operation of the payment gateway.

III. THREAT MODEL

A. E-commerce website design:

- ***Threat***: Cross-site scripting (XSS) attacks on user interfaces.
- Insecure data storage leading to data breaches.
- ***Countermeasures***: Implementing user authentication
- Using input validation and output encoding.
- Encrypting sensitive data at rest and in transit.

B. Blockchain Integration with Smart Contracts:

- ***Threat***: Smart contract vulnerabilities leading to exploitation.
- Unauthorized access to blockchain data
- ***Countermeasures***: Conducting deep code reviews and testing of smart contracts
- Access controls and encryption for blockchain data

C. Cryptocurrency payment gateway:

- ***Threat***: Crypto wallet theft or compromise
- Transaction malleability attacks
- ***Countermeasures***: Using hardware wallets and multi-signature wallets.
- Transaction validation and monitoring.

D. Database (MongoDB):

- ***Threat***: Unauthorized database access
- Data leakage due to misconfigured databases
- ***Countermeasures***: Using strong authentication and authorization controls.
- Regular audit and securing the database configuration.

E. Risk assessment:

- Evaluating the impact of each threat to determine the overall risk.
- Prioritizing the threats based on their potential impact on security, confidentiality, integrity, and availability.

F. Mitigation:

- Implementing a security policy and incident response plan.
- Continuously monitoring the security of the application, payment system, and blockchain components.
- Conducting regular security training for team members and maintain strong access controls.
- Engaging with third-party security experts to perform independent security assessments.

G. Monitoring:

- Continuously monitoring the system for emerging threats and vulnerabilities.
- Staying up to date with the latest security best practices and blockchain technology updates.

IV. SECURITY ARCHITECTURE AND DESIGN

The foundation of Ecrypt Pays security system is built on a dedication to user authentication, encryption, and access controls. These measures are careful. Strategically integrated to strengthen the system:

A. User Authentication, Encryption, and Access Controls:

- **User Authentication**: Ecrypt Pay utilizes advanced protocols to verify the identity of authorized users and prevent unauthorized access. Multiple layers of authentication including verification and multi-factor authentication (MFA) are employed to ensure that only legitimate users can access the platform.
- **Encryption Protocols**: All information such, as user credentials, transaction details, and personal data undergoes robust encryption using industry algorithms. Data in transit is safeguarded through Transport Layer Security (TLS) while data at rest is protected by implementing the Advanced Encryption Standard (AES). This two-fold encryption approach provides protection for user information.
- **Access Controls:** To minimize the risk of data exposure or manipulation precise access controls are implemented based on user roles and responsibilities. This ensures that users only have access to the functionalities required for their tasks.

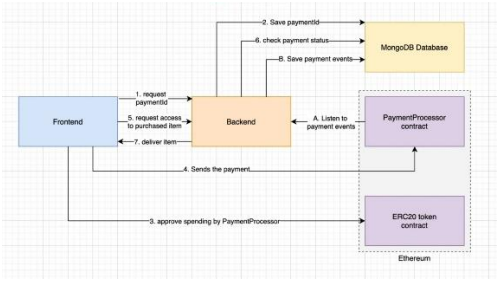
By prioritizing user authentication implementing encryption protocols and enforcing access controls Ecrypt Pay ensures a secure environment, for its user's data.

B. Secure Development Practices in Blockchain Integration:

Secure development procedures, which emphasize the use of ERC-20 standards and the Truffle development framework, serve as the foundation for Ecrypt Pay's integration of blockchain technology:

- **ERC-20 Standards**: The implementation of ERC-20 standards guarantees that Ecrypt Pays smart contracts adhere to accepted norms, within the industry. This conformity enhances interoperability enabling interaction, with applications and services based on Ethereum. Ecrypt Pays dedication to standardization simplifies the auditing process. Reduces vulnerabilities.
- **Truffle Development Framework:** Incorporating coding practices is an aspect of utilizing the Truffle development framework. Truffle provides a set of tools that facilitate the development, testing, and deployment of contracts while prioritizing security. Automated testing protocols are employed to detect and address vulnerabilities thereby bolstering the resilience of blockchain integration.
- **Transaction Monitoring:** Real-time transaction monitoring is a pivotal security feature embedded in the cryptocurrency payment gateway. Advanced anomaly detection algorithms and pattern recognition mechanisms are employed to identify potentially fraudulent or suspicious transactions promptly. Automated alerts and intervention protocols ensure swift responses to any security incidents.

- Please find the detailed image **here** **(use only the UNT account to view it)**



C. Blockchain Integration with Smart Contracts:

Our team has developed an architecture that integrates blockchain technology into our system. This includes implementing contracts to transactions.

Technologies used:

- ***ERC-2***: ERC-20 is a widely adopted Ethereum token standard, defining the basic rules for creating

fungible tokens on the Ethereum blockchain. It provides a blueprint for the smart contracts that can represent and manage digital assets like cryptocurrencies, utility tokens, and more.

- ***Truffle***: Truffle is a popular development framework for Ethereum-based blockchain applications. It provides a suite of tools and a development environment that simplifies the process of creating, testing, and deploying smart contracts and decentralized applications (DApps) on the Ethereum blockchain.
- ***MetaMask***: MetaMask is a cryptocurrency wallet software that interacts with the Ethereum blockchain. Developed by ConsenSys Software Inc., it permits users to access their Ethereum wallets via a browser extension or mobile application, enabling interaction with decentralized applications.
- MetaMask facilitates storage and management of account keys, transmission of transactions, transfer of Ethereum-based cryptocurrencies and tokens as well as secure linking to decentralized apps through compatible browsers or its built-in app's browser. It also enables websites and other decentralized apps integration using JavaScript code which lets these platforms send action prompts, signature requests, or transaction queries utilizing MetaMask as an interface.
- Here, we are using MetaMask wallet to process all the transactions. We have embedded the wallet with the Ethereum blockchain where it executes all the payments in real-time.

V. CODE ANALYSIS AND PENETRATION TESTING

This section provides an overview of the simple methods used in the development and testing stages of Ecrypt Pay. The focus is on code analysis and penetration testing.

A. Continuous Testing Approaches:

Ecrypt Pay employs user-friendly testing methods to ensure the reliability of the software. These include:

- **Manual Code Reviews:** The team collaboratively examines the codebase to identify any errors or areas for improvement.
- Reviews are conducted using version control systems like **Git**, and platforms such as **GitHub** or **GitLab** offer an intuitive interface for team collaboration.
- **Basic Automated Testing:** Simple automated tests are conducted to check if specific functions and features work as intended. This approach helps identify and address common issues early on.
- **JUnit** is a popular testing framework for Java applications. It is used to verify the accuracy and correctness of code snippets.
- **Mocha,** on the other hand, is a straightforward JavaScript testing framework that is often used in Node.js applications. It offers an easy approach to automated testing.

B. Future Milestones:

Future milestones in the development process are designed to be accessible and effective. These include:

- **User Testing Sessions:** Ecrypt Pay plans to involve users in the testing process through easy-to-participate user testing sessions. Real users will navigate the software and provide feedback on usability and areas that need attention.
- **Security Checks with Simple Tools:** Basic security checks are conducted using readily available and easy-to-use security tools. This includes scanning for common vulnerabilities to ensure the software's security in a straightforward manner.

C. User-Focused Refinement:

The refinement process of Ecrypt Pay is centered around the user and is straightforward. This includes:

- **User Feedback Surveys:** Simple user feedback surveys are conducted to gather insights. Users can express their thoughts on the software's usability, helping to identify areas for improvement.
- **Feature Prioritization Based on User Input:** User input takes priority in determining which features to prioritize. This approach ensures that the development process is guided by user needs, making the software more aligned with user expectations.

VI. SUMMARY OF THE ASSESSMENT AND LESSONS LEARNED

In this section, we will discuss the key takeaways and insights gained from our assessment of Ecrypt Pay's development, security measures, and user-focused approach.

A. Key Achievements:

- **Robust Security Measures:** Ecrypt Pay has implemented strong security measures, including user authentication, encryption protocols, and access controls. These measures work together to create a strong security perimeter, ensuring the protection of user data and fostering trust in the payment gateway.
- **Blockchain Integration**: The adoption of ERC-20 standards and the utilization of the Truffle development framework showcase Ecrypt Pay's commitment to secure and standardized blockchain integration. This approach not only enhances the reliability of the platform but also positions it favorably within the Ethereum ecosystem.

- **Cryptocurrency Payment Gateway:** Ecrypt Pay has designed a dedicated payment gateway for cryptocurrencies, utilizing hardware wallets and transaction monitoring. This proactive approach ensures the security of digital transactions and enhances user confidence in using cryptocurrencies.
- **Continuous Testing and Refinement:** Ecrypt Pay is dedicated to ongoing testing, including user testing sessions and security checks. This commitment ensures the reliability and security of the software. The user-focused refinement process, guided by regular user feedback, contributes to an iterative and user-friendly development approach.

B. Lessons Learned

- **Importance of User Engagement**: The emphasis on user testing sessions and feedback surveys has highlighted the significance of continuous user engagement. Understanding user preferences and addressing their concerns directly contributes to a more user-centric and intuitive payment gateway.

- **Iterative Development:** The iterative development approach, with continuous monitoring and refinement, has been crucial in adapting to evolving user requirements. This flexibility allows Ecrypt Pay to remain responsive to changing industry standards and emerging security challenges.
- **Strategic Use of Tools:** The selection of user-friendly tools for code analysis, penetration testing, and project management has streamlined the development process. This emphasizes the importance of choosing tools that align with the team's capabilities and the software's requirements.

CONCLUSION

In evaluating the development of Ecrypt Pay, we have gained valuable insights into their strong security measures, adaptable approach to development, and the significance of user engagement. By applying these lessons, Ecrypt Pay is poised to become a secure, user-friendly, and innovative payment gateway in the dynamic world of digital transactions.

REFERENCES

[1] Gav Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, 2014.

[2] Christian Reitwiessner, Fabian Hinzen, and Bernhard Scholz, "Smart contracts: A systematic classification and survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, pp. 1-37, 2017.

[3] Nick Johnson, "Solidity documentation," accessed September 23, 2023. [Online]. Available: https://docs.soliditylang.org/en/v0.8.13/

[4] Fabian Vogelsteller, Vitalik Buterin, et al., "EIP-20: ERC-20 Token Standard," Ethereum Improvement Proposal, 2015.

[5] "ERC-20 token standard," Ethereum Documentation, accessed September 23, 2023. Available: https://ethereum.org/en/developers/docs/standards/tokens/erc-20/

[6] Tom Boutell, "Truffle documentation," accessed September 23, 2023. [Online]. Available: https://trufflesuite.com/docs/truffle/

[7] Ryan Dahl, "Node.js documentation," accessed September 23, 2023. [Online]. Available: https://nodejs.org/en/docs/

[8] Dan Finlay, "MetaMask Documentation," accessed October 10, 2023. [Online]. Available: https://docs.metamask.io/

[9] ConsenSys. (2023). MetaMask - Bringing Web3 to Your Browser. [Online]. Available: https://metamask.io/

[10] Ethereum Project Yellow Paper. (2014). "Ethereum: A Secure Decentralised Generalised Transaction Ledger" by Gav Wood. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[11] JUnit. (2023). "JUnit 5 User Guide.". Available: https://junit.org/junit5/docs/current/user-guide/

[12] Mocha. (2023). "Mocha - the fun, simple, flexible JavaScript test framework. Available: https://mochajs.org/