Scientific Research Publishing

Open Journal of Applied Sciences, 2023, 13, 2080-2093
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

Securing Stock Transactions Using Blockchain Technology: Architecture for Identifying and Reducing Vulnerabilities Linked to the Web Applications Used (MAHV-BC)

Kpinna Tiekoura Coulibaly[1], Abdou Maïga[2], Jerome Diako[1], Moustapha Diaby[1*]

[1]LASTIC, African Higher School of Information and Communication Technologies (ESATIC), Abidjan, Ivory Coast
[2]UFRMI, Felix Houphouet Boigny University (UFHB), Abidjan, Ivory Coast
Email: tiekoura77@yahoo.fr, *moustapha.diaby@esatic.edu.ci

How to cite this paper: Coulibaly, K.T., Maïga, A., Diako, J. and Diaby, M. (2023) Securing Stock Transactions Using Blockchain Technology: Architecture for Identifying and Reducing Vulnerabilities Linked to the Web Applications Used (MAHV-BC). *Open Journal of Applied Sciences*, **13**, 2080-2093.
https://doi.org/10.4236/ojapps.2023.1311163

Received: October 31, 2023
Accepted: November 25, 2023
Published: November 28, 2023

Abstract

This paper deals with the security of stock market transactions within financial markets, particularly that of the West African Economic and Monetary Union (UEMOA). The confidentiality and integrity of sensitive data in the stock market being crucial, the implementation of robust systems which guarantee trust between the different actors is essential. We therefore proposed, after analyzing the limits of several security approaches in the literature, an architecture based on blockchain technology making it possible to both identify and reduce the vulnerabilities linked to the design, implementation work or the use of web applications used for transactions. Our proposal makes it possible, thanks to two-factor authentication via the Blockchain, to strengthen the security of investors' accounts and the automated recording of transactions in the Blockchain while guaranteeing the integrity of stock market operations. It also provides an application vulnerability report. To validate our approach, we compared our results to those of three other security tools, at the level of different metrics. Our approach achieved the best performance in each case.

Keywords

Stock Market Transactions, Action, Smart Contracts, Architecture, Security, Vulnerability, Web Applications, Blockchain and Finance, Cryptography, Authentication, Data Integrity, Transaction Confidentiality, Trust, Economy

1. Introduction

The stock market, as a financial institution, plays a central role in the economy

by facilitating the exchange of values, stimulating investments, and contributing to economic growth. It acts as a hub where investors and businesses meet to trade financial assets such as stocks and bonds. Within the West African Economic and Monetary Union (UEMOA), the stock market is of particular importance as a driver of economic development in the region. The advent of digital technology has significantly reshaped the way stock trading is conducted. Web applications have emerged as essential tools to facilitate these financial exchanges by allowing investors to access markets and conduct transactions online. This digital transformation has brought undeniable advantages in terms of speed, accessibility, and efficiency of operations. However, this increasing digitalization of stock transactions also exposes systems to new IT security risks and challenges. The web applications that power these transactions are increasingly becoming the target of malicious attacks, aimed at exploiting vulnerabilities and security flaws for illicit gains. Among these challenges, the attack on the integrity of transaction data stored in the database is particularly feared. This could involve manipulating the number of securities an investor holds or even deleting an investor's account, which represents a major risk. The consequences of such attacks go beyond financial losses, including loss of investor confidence and damage to the reputation of financial institutions.

Thus, improving the security of stock transactions is of paramount importance to protect investors, prevent financial losses, maintain market stability, protect the economy, and comply with regulations. This helps to strengthen investor confidence and promote fair and transparent functioning of the stock market. Indeed, this security improvement guarantees the protection of investors against fraud, market manipulation and identity theft, thus encouraging their participation in the stock market. Improving the security of transactions reduces the risk of financial losses due to fraudulent activity or human error, which helps preserve the value of investments and minimize negative consequences for investors. Furthermore, the stability of the financial market is maintained thanks to the reduction of the risks of market manipulation and fraudulent transactions and the absence of artificial distortions of asset prices. Since the stock market plays a crucial role in the economy by mobilizing capital for businesses and facilitating investment, enhanced security of stock transactions protects the economy from systemic risks and helps promote economic growth. It is therefore imperative to put robust security mechanisms in place to prevent such threats and protect sensitive transactions. Blockchain technology, recognized for its ability to guarantee data security and prevent fraud, presents itself as a promising solution to meet these challenges. Based on principles of decentralization, immutability, and transparency, Blockchain offers a distributed ledger that can effectively secure stock market transactions by eliminating the risks of falsification and manipulation of data. Its application to securing web applications for stock market transactions within the UEMOA can provide an additional layer of confidence by making transactions verifiable and inviolable.

Furthermore, vulnerabilities can come from negligence or weakness in the design or use of the application. In this case, it is important that the security system can detect them and provide a report to remedy them.

The main objective of this project is to design and implement a security system to guarantee the integrity of stock transactions and the confidentiality of sensitive data within the web application.

Our work will be structured around four main sections. In the first part, we will discuss a description of the state of the art. In the second part we will define the problem then the third part concerns the presentation of our contribution. The fourth part will be devoted to the discussion of the results obtained. We will end with a conclusion with research perspectives.

2. State of the Art

Several research studies have focused on the problem of securing applications with or without the use of blockchain. In this section we give the most relevant ones in the literature.

Putri and *et al.* (2020) [1], implemented a two-factor authentication system based on the Ethereum blockchain for secure connection. This system uses a DApp (Decentralized Application) for the generation of tokens and does not use any third parties. The designed token generation system can generate 3164 tokens per second to avoid collisions. Double factor authentication is carried out by the DApp without the user intervening, thus avoiding MITM (Man in The Middle) attacks. However, this needs to use a Dapp can lead to a certain familiarity with Dapps, which presents a disadvantage. Other limitation of this system is the failure to consider vulnerabilities relating to the source code of the application.

Tanrıverdi (2020) [2], developed an authentication system (SSO) based on blockchain and implemented for web applications. In this system, a public address and a private key are defined on the blockchain network for users. This is a private blockchain. The public address and private key are used for the 2FA method via the mobile application that has been developed. In the operation of the solution, the data entered by the user is sent by the authentication service connection module which accesses the data flow through the MultiChain API and verifies the username and the password received. The inadequacy of this security system lies in the fact that it requires a private blockchain, which is potentially more complex to implement. Like previous approaches, it does not consider the identification of vulnerabilities linked to errors in the application source code.

Shanmuigapriya *et al.* (2021) [3], use an architecture based on a private blockchain, in collaboration with traditional stock market participants, to facilitate stock trading. The platform does not introduce major changes to the trading logic and maintains the traditional parts of the system, thus promoting the adoption of decentralized stock exchange platforms. As a weakness, this architecture requires the collaboration of traditional actors and admits technical

complexity. In addition, it does not make it possible to identify internal vulnerabilities arising from the source code of the application.

Al-Shaibani *et al.* (2020) [4], presented a new blockchain-based architecture for a fully decentralized stock market platform. The architecture is based on the Ethereum smart contract which is implemented on a consortium and a permissioned network.

Raounak Benabidallah *et al.* (2019) [5] for their part proposed a software code vulnerability analysis meta-scanner called CVMS (Code Vulnerability Meta-Scanner) using a combination of existing vulnerability analysis tools, namely Fortify, SpotBugs and Yag Suite. They first aggregated the vulnerability analysis tools to im-prove their effectiveness and then proposed a heuristic which combines these tools according to their precision in identifying each category of vulnerabilities. If the pro-posed solution makes it possible to effectively identify vulnerabilities coming from the source code of the application, it does not, however, secure the application from vulnerabilities linked to its use.

3. Problem

The analysis of the various works presented above has shown that most of them, notably the approaches [1] [2] [3] [4] do not consider the identification of vulnerabilities at the level of the application's source code, anything which constitutes a significant limit for an application which manages sensitive stock market transaction data. Indeed, the source code itself represents a vulnerability be-because it is very sensitive to leaks. It is an asset widely distributed in our organizations, for example through version control systems such as GitHub, code clones on developers' machines, copies saved in the cloud, in messaging systems and in public forums. Consequently, if said code contains a certain number of inadequacies in its development such as containing sensitive or secret in-formation (API keys and credentials, security certificates, database access keys, etc.) or code from open-source libraries, SaaS tools, and other external components, malicious actors can access it and execute active or passive threats against the application.

The approach [5] does not integrate blockchain technology into the operation of the application. It therefore does not ensure a good level of security of access to the application, nor does it guarantee trust between actors and the integrity of transactions.

The inadequacies of the solutions presented raise the following problem: how to implement a robust security system for web applications for managing stock market transactions guaranteeing their integrity, confidentiality, and immutability of the sensitive data handled.

To resolve this problem, we propose an approach based on Ethereum blockchain technology which makes it possible to secure the application through, on the one hand, the identification of possible vulnerabilities in the source code with a view to remedying them, and on the other hand, secure user access control.

4. Contribution

To improve transaction security, our Blockchain-based solution uses advanced cryptography algorithms. Each transaction is encrypted and linked to the previous one, thus forming a blockchain. This makes transactions virtually impossible to forge or modify. Furthermore, the decentralized structure of the model ensures transaction verification by consensus between network nodes, which makes malicious attacks more difficult. The risks of fraud or data manipulation are also reduced due on the one hand to the transparency of the system which makes all transactions visible to all participants in the network and on the other hand because of Smart Contracts which facilitate, automatically verify, and execute the agreement terms.

Our solution revolves around two main objectives: 1) the identification of vulnerabilities in the source code of the application and 2) the management of the connection of investors to the application and the immutable recording of transactions carried out in the Ethereum blockchain.

- In terms of identifying vulnerabilities contained in the application source code.

We proposed a new source code vulnerability analysis tool called MAHV (Hybrid Vulnerability Analysis Method). It is based on a hybrid approach using supervised learning and a combination of static vulnerability analysis tools following the majority voting heuristic. The principle of our algorithm consists of using the results given by the tools used and integrating them into a weighted vote to decide on the level of vulnerability of the applications exploited to allow a code review to be carried out. The process describing our approach is described as follows:

Step 1: Since vulnerability analysis tools do not have the same way of identifying the same vulnerability, we first establish a correspondence between the results of the tools and the real vulnerabilities, to provide a common reference of the vulnerabilities for the tools used in the combination. The operation is based on CWE categorization and produces a correspondence matrix.

Step 2: we estimate the confidence rate of the vulnerability analysis tools used by evaluating each of them using the Juliet test suite [6]. We obtain a confidence rate matrix.

Step 3: In this step, we use static code analysis tools as well as our supervised learning model as well as the correspondence matrix to determine the existence of vulnerabilities. The learning model used is a vulnerability prediction model (VPM). We then aggregate the results of each tool collected, for each corpus of code, in the form of a vote weighted by the confidence rate of the tools. Below is our vulnerability prediction algorithm (VPM) (**Algorithm 1**).

Figure 1 presents the architecture of our source code vulnerability analysis system called MAHV.

- In terms of the establishment of two-factor authentication and recording of transactions in the blockchain:

Algorithm 1. VPM construction algorithm.

1. Select candidate projects (with long vulnerability histories)
2. Collect vulnerabilities
3. Choose the granularity of the study (part of code or individual who will be the subject of learning)
4. Extract features (meaningful variables to use as features for the learning algorithm)
5. Create the database (set of vulnerable and non-vulnerable individuals characterized and labeled after merging the collected vulnerabilities and characteristics)
6. Build the prediction model using the decision tree algorithm
7. Evaluate the prediction model used on the test set

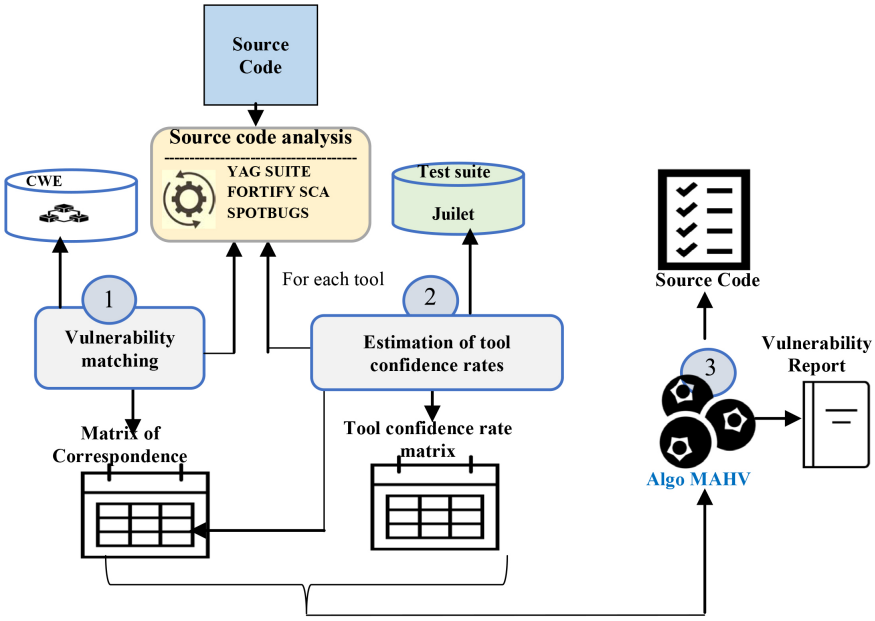

Figure 1. Process for building our hybrid vulnerability analysis tool.

The architecture of our system is based on a web application. To optimize security and reduce the risks linked to vulnerabilities, we have deployed a three-level architecture isolating the code logic from potential attacks from the Web. The process begins by setting up an API interface (Application Programming Interfaces) (**Figure 2**) which ensures communication between the front-end and the back-end using HTTP requests.

To use Ethereum blockchain technology, we have integrated a smart contract. The latter allows transaction information to be permanently recorded in the blockchain. Co-ordination between the smart contract and our Web API is ensured by the Web3.js library. This JavaScript interface facilitates the transmission of transaction data to the smart contract, while providing control and verification mechanisms.

At the same time, we implemented a two-factor authentication (2FA) system using blockchain. Conventional 2FA relies on sending tokens via messaging or link validation. Thus, our approach provides for the creation of a mobile token verification application, functioning as a decentralized application (DApp). This

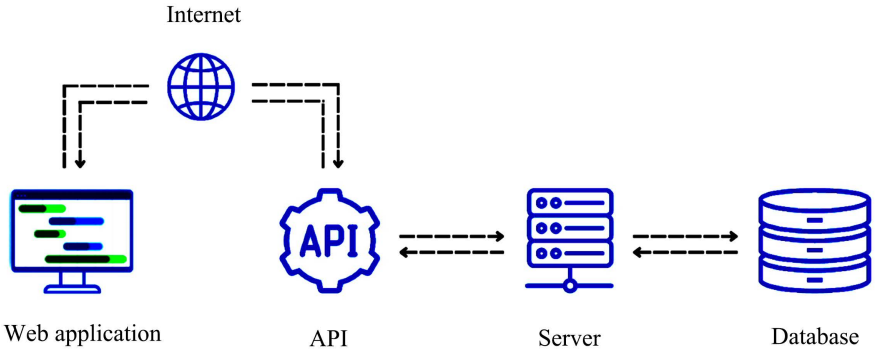

Figure 2. How the Web API works.

DApp establishes communication with a specific node on the blockchain dedicated to this purpose. Upon authentication, the token is transferred to the blockchain, saved on the specific node, and simultaneously sent to the user's mobile application. The authentication process simply requires the user to press the verification button, thus initiating a comparison between the token registered on the blockchain and that of the mobile application.

The overall process unfolds as follows: When the investor enters their username and password, our Web API initiates an HTTP request containing the connection information. This request is directed to the web server for verification within the database. When the information proves to be correct, a confirmation page is displayed to the customer. This page includes a validation button, triggering the two-factor authentication process. By pressing this button, a token with a limited validity period (5 minutes) is generated and transmitted to the blockchain via a mobile verification application previously installed by the client. The customer must open the mobile application and activate the verification button that appears. At this stage, a verification procedure is put in place between the token stored in the blockchain and that kept in the mobile application. Once this process has been successfully completed, the customer is authorized to access their account securely. Once connected, the investor can then make subscriptions to the Public Savings Calls (APE) which are in progress. When an investor makes a subscription on the platform, the details of this transaction are sent to our Web API. The API, using Web3.js, transfers this data to the smart contract in the Ethereum blockchain. The smart contract then records the transaction information in the blockchain in an immutable and transparent manner, thus guaranteeing their integrity and traceability.

This sophisticated architecture makes it possible to combine two-factor authentication and transaction recording in the blockchain, which provides increased security to our stock transaction management application. Investors can interact with the platform with confidence, knowing that their accounts are protected by a two-step verification mechanism. Additionally, the indisputable traceability of transactions guarantees data integrity, thereby strengthening trust and transparency within the ecosystem.

Overall, our contribution is summarized in the combination of the two security aspects of the stock transaction management application described previously, namely: the identification of source code vulnerabilities (MAHV) and the limitation of external vulnerabilities linked to the connection to the application via Blockchain (BC) technology. The final architecture of our system which we call MAHV-BC is as follows (Figure 3).

5. Evaluation of Our Model

5.1. Experimentation

To evaluate the performance of our MAHV-BC approach, we will compare it with other approaches from the literature. The objective is to verify whether our approach gives better results than the tools used individually. The results will be

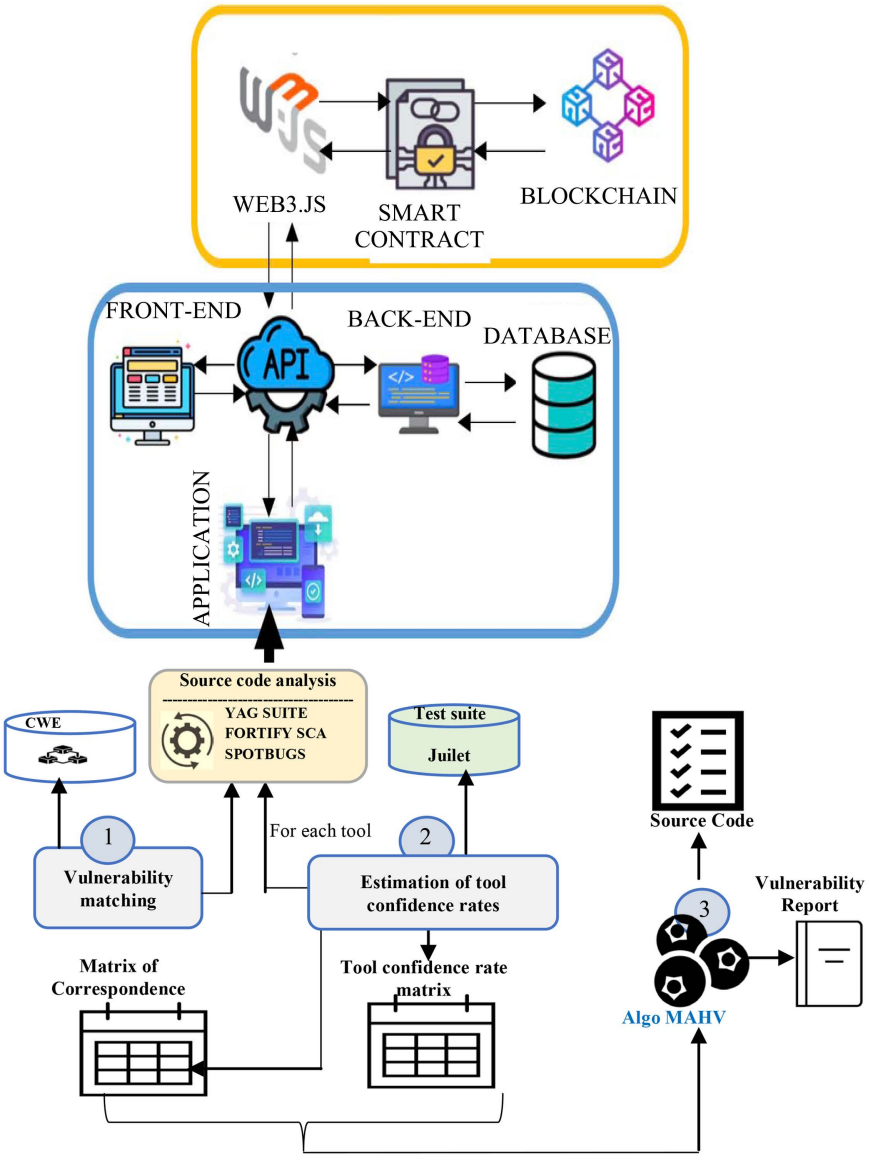

Figure 3. Architecture of our MAHV-BC system.

compared against three metrics: 1) the confidence rate of the tool, 2) the detection accuracy of the tools through the calculation of their macro-average and 3) the overall results in terms of Precision/Reminder.

To do this, we used data from the National Security Agency (NSA) Center for Assured Software (CAS) [7]. This center has developed artificial test cases called Juliet Test Cases, to test static analysis tools.

The test cases use the CWE list as a basis for naming and organization and include 113 CWE entries including 11 software errors deemed dangerous and 14 errors not detectable by static code analysis.

Our approach is compared to the following three approaches:

- Approach by Shanmuigapriya *et al.* (2021) [3].
- The CVMS meta-scanners for vulnerability analysis by Raounak *et al.* (2019) [5].
- The Yag Suite tool [8].

Thus, for each of the four approaches and for each vulnerability category detected, we calculate the metrics as follows:

Confidence rate:

$$TC_{ov} = \frac{VP_{ov} + VN_{ov}}{VP_{ov} + FP_{ov} + VN_{ov} + FN_{ov}} \tag{1}$$

Macro-averages of tools: average of the tool only over all the vulnerabilities it covers.

$$\mathcal{M}_o = \frac{\sum_{c \in V} Exactness_{ov}}{|V|} \tag{2}$$

Overall macro-averages of tools: average of the tool o over all the vulnerabilities covered in this study.

$$\mathcal{M}g_o = \frac{\sum_{c \in V} Exactitude_{ov}}{|Vc|} \tag{3}$$

Accuracy or rate of good alerts among all alerts returned

$$Pr_o = \frac{VP_o}{VP_o + FP_o} \tag{4}$$

The Recall or rate of true alerts among all alerts that should be returned

$$Rap_o = \frac{VP_o}{VP_o + FN_o} \tag{5}$$

The F-measure or average between precision and recall

$$F\_measure_o = \frac{2 * Pr_o * Rap_o}{Pr_o + Rap_o} \tag{6}$$

With:

TC_{ov} : The confidence rate of tool *o* for vulnerability *v*.

VP_{ov} : True Positive, number of individuals correctly detected for correctly detected by the tool *o* for vulnerability *v*.

VN_{ov} : True Negatives, number of individuals considered not to contain vulnerability *v*, and who are truly non-vulnerable.

FP_{ov} : False Positives, represents the number of individuals that the tool *o* considered to contain vulnerability *v* even though they do not contain it.

FN_{ov} : False negatives are the number of individuals where the tool *o* did not detect the vulnerability *v* even though they contain it.

V: all vulnerabilities detected by the tool o.

$\mathcal{M}_o$: Macro-average of tool *o.*

$\mathcal{M}g_o$: Macro-average of tool *o.*

Vc: set of vulnerabilities covered in the study.

5.2. Results and Discussions

1) Comparison of tool confidence rates

Table 1 summarizes the different results obtained after testing based on 10 vulnerabilities covered. Figure 4 presents a comparison of our tool to others according to the precision in detecting vulnerabilities.

Looking at the results in Figure 4, we notice that for all the identifiers, our MAHV-BC approach is better. Furthermore, unlike certain tools, our solution manages to detect most of the vulnerabilities analyzed.

2) Comparison of tools according to vulnerability detection accuracy.

To do this, we calculated the different averages (macro-average and overall macro-average) for each approach. The results are described in Table 2.

In Figure 5, we compare our tool to the other three (3) based on vulnerability detection accuracy.

The analysis of Figure 5 also shows that our MAHV-BC approach is better than others in vulnerability detection accuracy.

Table 1. Vulnerability identification tool confidence rating test results.

Tools / Identifiers	Shanmuigapriya *et al.*	CVMS	Yag Suite	MAHV-BC
1	0.61	0.82	0.64	0.84
2	0.45	0.63	0.61	0.70
3	0.60	0.75	0.74	0.80
4	0.57	0.77	0.75	0.78
5	0.40	0.76	0.64	0.76
6	0.55	0.83	0.1	0.86
7	0.1	0.57	0.57	0.59
8	0.63	1.0	1.0	1.12
9	0.49	0.64	0.64	0.71
10	0.1	0.82	0.1	0.85

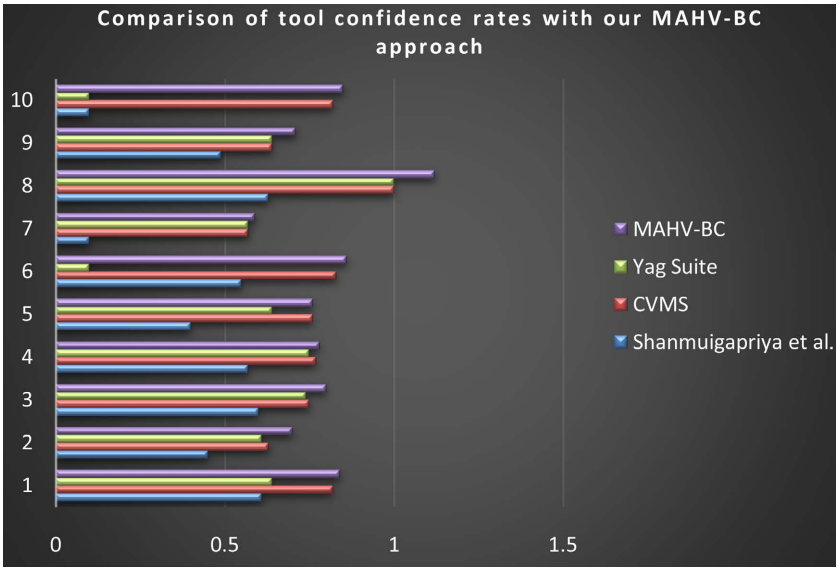

Figure 4. Comparison of tool confidence rates.

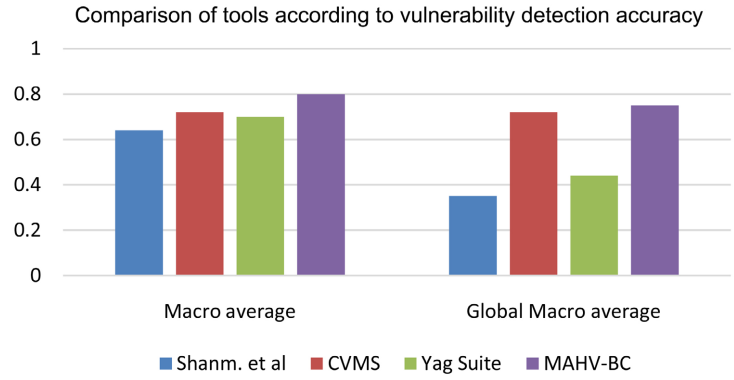

Figure 5. Comparison of tools based on vulnerability detection accuracy.

Table 2. Vulnerability identification.

Performances	Shanmuigapriya and *et al.*	CVMS	Yag Suite	MAHV-BC
Macro average	0.64	0.72	0.70	0.80
Global macro average	0.35	0.72	0.44	0.75

3) Comparison of tools from the point of view of their average performance

This result requires the calculation of three metrics namely Precision, Recall and F-measure. Table 3 shows the test results for each tool.

In Figure 6, we compared the performance level of our MAHV-BC tool to that of other vulnerability identification tools.

The results above show the effectiveness of our approach in terms of the Recall metric and the F-measure. However, for the precision metric, the Yag suite tool is better. This is explained by the fact that it has the advantage of evaluating the relevance of vulnerabilities better than others and does not only detect vulnerabilities but focuses more on reducing the number of false positives (FP).

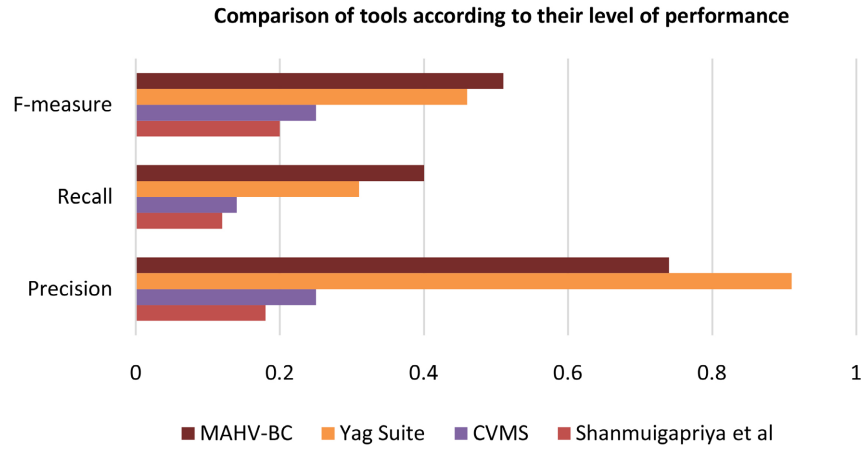

Figure 6. Comparison of tools according to their performance.

Table 3. Vulnerability scanning tool performance chart.

Tools	Precision	Recall	F-measure
Shanmuigapriya *et al.*	0.18	0.12	0.20
CVMS	0.25	0.14	0.25
Yag Suite	0.91	0.31	0.46
MAHV-BC	0.74	0.40	0.51

In short, considering the different results presented above, our approach generally gives better results than other analysis tools taken individually from the point of view of the number of vulnerabilities and performance. In other words, it makes it possible to identify vulnerabilities with better precision than existing tools and with fewer false positives.

6. Conclusions

In this paper, we proposed a security approach for a stock trading web application operating with blockchain technology. Our not only allows us to limit vulnerabilities linked to access to the application through rigorous control via the blockchain, but also and above all it allows us to improve the source code by identifying vulnerabilities. For the identification and prediction of internal vulnerabilities in the source code, we have coupled a machine learning algorithm (decision trees) with a combination of static vulnerability identification tools. All this allowed our approach to have the best results in terms of number of vulnerabilities and performance, compared to the vulnerability analysis tools presented, in particular the CVMS tools, Yag suite and that of Shanmuigapriya *et al.* These results show the advantage of combining static analysis tools with machine learning and blockchain. This approach makes it possible to cover a larger set of vulnerabilities than the analysis tools taken individually.

The research content of this article can be applied to the actual operation of the stock market to strengthen transaction security. It is important to note that

this requires frank collaboration with stakeholders including regulators, financial and stock exchange institutions. The implementation of our solution will allow, on the one hand, those responsible for these stock markets, to improve the source codes of the web applications they use thanks to the detection of vulnerabilities and on the other hand will guarantee trust between the actors, and the various investors while protecting their data and transactions.

There are many possibilities for improving our approach, by exploiting a hybrid combination of dynamic and static tools with blockchain technology.

Acknowledgements

The authors thank the editor and three anonymous referees for their helpful comments and suggestions.

Author Contributions

Conceptualization, K. T. C.; methodology, K. T. C.; software, J. D. and A. M.; validation, K. T. C. and A. M.; formal analysis, K. T. C., M. D. and A. M.; investigation, K. T. C.; resources, K. T. C. and A. M.; data curation, J. D. and M. D.; writing—original draft preparation, K. T. C.; writing review and editing, K. T. C. and J. D.; visualization, J. D.; supervision, M. D.; project administration, K. T. C..; funding acquisition, K. T. C. and A. M. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Conflicts of Interest

The authors declare that there is no conflict of interests in the publication of this paper.

References

[1] Putri, M.C.I., Sukarno, P. and Wardana, A.A. (2020) Two Factor Authentication Framework Based on Ethereum Blockchain with dApp as Token Generation System Instead of Third-Party on Web Application. *Register*: *Jurnal Ilmiah Teknologi Sistem Informasi*, **6**, 74-85.

[2] Tanrıverdi, M. (2020) Design and Implementation of Blockchain Based Single Sign-On Authentication System for Web Applications. *Sakarya University Journal of Computer and Information Sciences*, **3**, 343-354. https://doi.org/10.35377/saucis.03.03.757459

[3] Shanmuigapriya, T., Sasirekha, S., Joe Louis Paul, I. and Sowmya, R. (2021) Blockchain Enabled Smart Contract For Stock Exchange. https://www.tojqi.net/index.php/journal/article/view/8520

[4] Al-Shaibani, H., Lasla, N. and Abdallah, M. (2020) Consortium Blockchain-Based Decentralized Stock Exchange Platform. *IEEE Access*, **8**, 123711-123725. https://doi.org/10.1109/ACCESS.2020.3005663

[5] Benabidallah, R., *et al.* (2019) Designing a Code Vulnerability Meta-Scanner. *International Conference on Information Security Practice and Experience*, 194-210. https://doi.org/10.1007/978-3-030-34339-2_11

[6] Boland, T. and Black, P.E. (2012) Juliet 1.1 C/C++ and Java Test Suite. *Computer*, **45**, 88-90. https://doi.org/10.1109/MC.2012.345

[7] NSA (2012) Juliet Test Suite v1.2 for Java. https://samate.nist.gov

[8] YAGAAN Software Security. Yag Suite. https://www.yagaan.com/en/

Supplementary Materials

We used data from the National Security Agency's (NSA) Center for Assured Software (CAS) [7]: NSA. Juliet Test Suite v1.2 for Java. https://samate.nist.gov. Accessed: 2020-08-27. 2012. This center has developed artificial test cases called Juliet Test Cases, to test static analysis tools.