Software Supply Chain Attack Management Using Blockchain

A Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Bachelor of Technology
in
Computer Science & Engineering

By-
Anuj Kesarwani (20208022)
Harshit Chaudhari (20208054)
Garlapati Tarun (20208047)
Jitendra Kumar (20208060)

to the

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY ALLAHABAD

May 2023

UNDERTAKING

We declare that the work presented in this report titled **"Software Supply Chain Attack Management in Software Domain",** submitted to the Computer Science and Engineering Department, Motilal Nehru National Institute of Technology Allahabad, Prayagraj, for the award of the Bachelor of Technology degree in Computer Science & Engineering, is my original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I accept that my degree may be unconditionally withdrawn.

May 2023

Anuj kesarwani (20208022)
Harshit Chaudhari (20208054)
Garlapati Tarun(20208047)
Jitendra Kumar (20208060)

CERTIFICATE

Certified that the work contained in the report titled
"Software Supply Chain Attack Management using Blockchain",
has been carried out under my supervision and this work has not been submitted elsewhere for a degree.

By-
Anuj Kesarwani (20208022)
Harshit Chaudhari (20208054)
Garlapati Tarun (20208047)
Jitendra Kumar (20208060)

Prof. D.K. Yadav
Computer Science and Engineering Dept.
M.N.N.I.T., Allahabad

May 2023

Acknowledgments

We wish to thank the Institute and the Department of Computer Science and Engineering, for giving us the opportunity to work on this project. We would like to thank the Head of the CSE department **Prof. D.K. Yadav**, who mentored us and guided us through the entire process. We also wish to thank the Director, **Prof. R. S. Verma**. This project has been a learning experience through which we got to know about **‘Blockchain’**, and it allowed us to get hands-on experience with building a real product.

Contents

Chapter 1

Introduction

Software components and services have become the main building blocks for distributed software projects, allowing software engineering to become truly global. With the rise of distributed components and orchestrated software services, developers can now leverage these building blocks to create highly modular and scalable software systems. A software component is a binary building block that can be easily reused across multiple software projects. This allows developers to focus on building unique features and functionality while relying on pre-built components for common tasks.

Despite the benefits of software components, there are still concerns about trust and security[1]. When evaluating a software component, system integrators treat it as a black box, meaning they have limited visibility into the component's implementation details. This can create uncertainties around the component's trustworthiness, security, and potential vulnerabilities[5].

To address these concerns, we have created a decentralized platform that utilizes blockchain technology and InterPlanetary File System (IPFS) to enable developers to securely upload, share, and reuse software components. With our platform, developers can leverage the benefits of pre-built software components, while also ensuring that they are secure, trustworthy, and free of vulnerabilities. Our platform provides a transparent and decentralized approach to component sharing, which allows developers to easily discover, evaluate, and reuse software components without compromising their security or privacy. This documentation provides an overview of our platform's architecture, functionality, and implementation details, highlighting its potential benefits for developers and organizations looking to leverage the power of distributed software components and services.

1.1) Motivation

The use of open-source software has become increasingly prevalent in recent years, with many developers relying on pre-built components and libraries to accelerate their development process. While open-source repositories have made it easier for developers to find and use these components, there are concerns about the trustworthiness of the code they contain. Many mechanisms have been used to create open-source repositories, but they do not always provide sufficient trust in the security of the code. This lack of trust can make it difficult for developers to evaluate the quality of the code and assess the potential risks associated with using it in their projects. To address this challenge, there is a growing need for new technologies that can increase trust in open-source repositories. The advent of blockchain technology and its immutable nature make it the perfect technology for building an open-source repository that enables software developers to upload and share their component's code with complete confidence in their security.

Chapter 2

Problem Description and Related Work

In this chapter, we introduce the problem description and related works toward the trust of third-party components in the supply chain.

2.1 Problem description

Software components and services are the main building blocks for distributed software projects, enabling them to make software engineering really global. Global software often is made using distributed components or orchestrating software services. The software component is a binary building block for software systems. Because it is binary, it can be the object of commerce (selling and buying) and object of easy third-part composition. One can reuse the functionality of the component without disclosing implementation details.
However, trust for binary components or services is still an open question. System integrators can evaluate components only as a black box, thus they cannot be sure that the component is trustful, secure, and does not contain vulnerable code.

Because the components and services are provided to the system without the possibility of a detailed inspection of its internals, global system integrators are walking on edge to approve such building blocks inside the system. Existing trust-assurance methods are either **Invasive or Non- invasive**. Invasive methods are made for stress testing of components, for the prediction of its inner properties, or for emulating real-time use of them. Some invasive methods treat the component as a "gray box", or "glass box" and even use reverse engineering methods in order to disclose the internal structure and/or the algorithms implemented. This is on the edge in the legal context and gives a lot of ethical questions. Non-Invasive methods are based on certificates and signatures. etc. The component is treated as reliable and trustful of some authority (including big corporations) claims so.

The purpose of this project is to tackle the main issues of the software components and services trust and to present the blockchain-based model of software metadata registration and retrieval to increase the trust for the software composition entities.

In general, software supply chain attacks aim to inject malicious code into a software product. Frequently, attackers tamper with the end product of a given vendor such that it carries a valid digital signature, as it is signed by the respective vendor, and may be obtained by end-users through trusted distribution channels, e.g. download or update sites.
A prominent example of such supply chain attacks is **Not Petya**, a ransomware concealed in a malicious update of a popular Ukrainian accounting software [1]. In 2017, Not Petya targeted Ukrainian companies but also hit global corporations, causing damage worth billions of dollars.

2.2 Related work

Related and proposed references that discussed how the OSS supply chain can be tracked is, by proposing to create three types of networks: dependency network, code reuse network and knowledge flow network [8].
Another reference has briefly discussed the issue of the trusted software supply chain, where their proposed solution is a blockchain-enabled governance framework for a trusted software supply chain that tries to address the challenges, such as compliance and governance, provenance, security, integrity, etc [5]. This governance framework describes methodology where authors have provided realization of this framework utilizing the TestRPC Ethereum blockchain network.

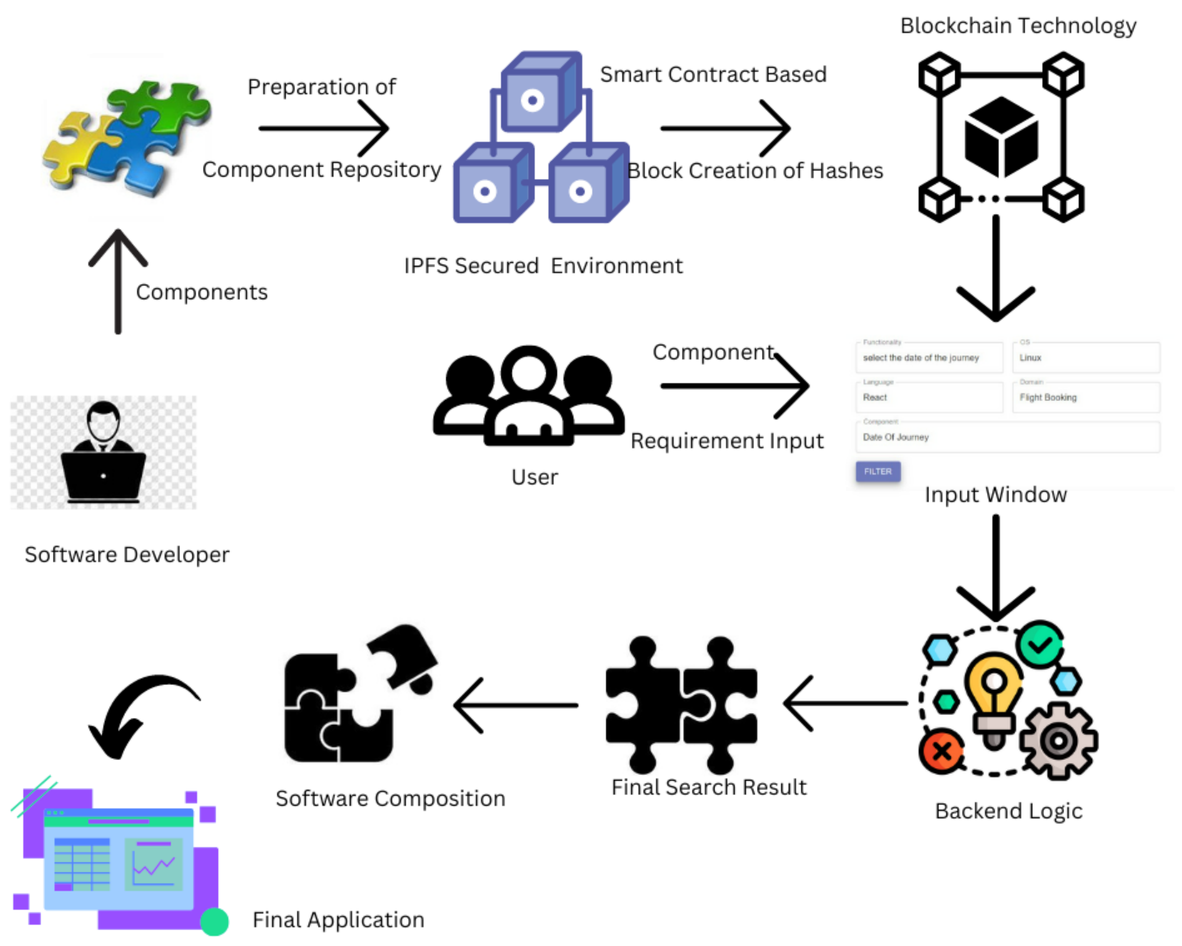
A similar architecture was proposed by the reference [3], where the solution is intended for improving supply chain quality management, by adopting 'Blockchain technology'. This blockchain-based supply chain quality management framework has four layers, based on different functions. As a solution that relies on 'smart contracts', it is a promising approach and should be followed once the design is applied for a real-world application. Another study on product traceability is referenced in the paper [5], where they have proposed a solution called OriginChain, which employs a geographically distributed private blockchain at the traceability service provider. Another framework for securing the supply chain is proposed in reference [6], where TrustChain design uses a consortium blockchain to track interactions among supply chain participants and to dynamically assign trust and reputation scores based on these interactions.

A research paper from 2018 referred [4], interviewed 18 experts on blockchain disruption in supply chains and industrial applications, concluding that blockchain is not disruptive. Instead, it is a sustaining innovation. An important distinction was made by the referred publisher [5],where they have kept some information off-chain, i.e.they proposed a combination of blockchain and SupplyChain(Off-chain) coordinated by a Server Blockchain Interface , which receives data for each Server Supplier . SupplyChain challenges were present in the
the referred paper [2], where it was emphasized over the importance of trustworthiness of systems through security, safety, privacy, resilience and reliability.

Chapter 3

Proposed work

Open Source Repository - Step-By-Step Working



1. User Authentication - Traceability, Transparency is one of the important features of the project Open Source Repository. In order to know the actual programmer for a code we need to know the authenticity of the user. To achieve this we are using the Mongoose package to make the user schema, bcrypt to hash the password, json web token is generated while logging in and stored in a cookie named jwt token to authenticate the session.This website also allows the features of sending otp and changing password for the authenticated user using nodemailer

2.Uploading Software Component - This the main part of our project where the user can make repository of the his code with the details of like domain,language,functionality,operating system and description which helps the users to know the code details and helps them while searching i used the multer to store the files at the server end and from the path where the file is stored by the user the IPFS hash is generated using the IPFS client and upon successfully generating the IPFS hash of the file the whole details are being stored in the local storage with their keys and navigated to the transaction page . In this transaction page the transaction takes place(this needs a metamask wallet with sufficient amount of the sepolia ether) which helps us to save the details in the contract address the contract address stores the data of the component,functionality,os,language,file hash,transaction hash upon successful completion of the transaction which requires the cost of the 0.00000005 + gas price and extra sepolia ether to interact with the smart contract for the database MongoDB to keep note off the details which are required to navigate to the specific block(repo) in the MongoDB the details of the block number(repo no) and the code details are added in the array

3.Displaying the software components - This page corresponds to one of the important feature of our project which is transparency and core concept of open source where a user regardless of his contributions can see this page which consists of the component names and the date when they are created in form of cards by default the matching percentage of these card are "0" but if you use the search bar to fill the components and the filter button then the whole array of the details stored at the contract address will be compared and a matching percentage number is generated then the whole array gets displayed in the sorted descending order of the matching percentage this will help the users to search for a particular component and see its details.Upon clicking the see details button of the card then an new page with the corresponding blockNo (repo no) is opened which contains all the details from the component,functionality,os,language,description,file hash, transaction hash, the creator of that particular code in this way it helps the developers to get their hands on a particular component code.

4.Contact Us & Profile - As the saying goes nothing is perfect no matter what the technologies we use at the end it's not 100% perfect so in order for the users to share their grievance regarding their transactions of the notification regarding the authenticity of the uploaded code the users can notify us as then based on the severity of the problem we will solve this is done by the use of the formspree and profile page this page contains the basic details of the users and the no of contributions the user hash made

5. Display the Transactions from the smart contract - One of the important feature that the website provides you is the security no one could change the contents stored in the smart contract and the transaction details are directly fetched by the smart contract and one could check their transactions from the official sepolia etherscan website these indices of the code added are store in the MongoDB database so your contents can be viewed by your credentials only.

Chapter 4

Experimental Setup

For experimental setup following technologies are used along with brief explanations about their integration and resulting user interface (GUI) so as to facilitate the steps on how users can interact with the system.

4.1 Technologies Used:

1. **Smart Contract -** As the name itself says the contract which is smart itself to execute itself according to the rules and regulations of the smart contract when written , a smart contract in this project of the Open Source Repository containing functions to :

(i) Generate hash from the details like previous hash,functionality,component,os,language,file hash,transaction hash (here we are using the sha256 to generate the hash)

(ii)Return the number of repositories made at that particular time for that particular contract address

(iii) Return the details of the repository this will require repository number as an input to return the details that are stored in the smart contract component,functionality,filehash,description,transaction hash,language,os

(iv) Add details to the smart contract memory this will require the condition of the present previous hash is equal to the previous repository hash upon satisfying the condition current repository details will be added to the smart contract memory.

2. **IPFS (InterPlanetary File System):** IPFS is a decentralized file storage and sharing system which works on the p2p network this makes sure that the content of the code is not changed as unlike the server the contents aren't stored at a single server or system instead its stores at all the nodes which makes the whole website safe and secure In your project, IPFS is used to store the software component files uploaded by users. When a user uploads a file, the file is converted into an IPFS hash through the IPFS client which can be used by the API keys generated in the infura and stored on the IPFS network. This ensures that the files are decentralized and can be accessed from anywhere in the world.

3. **Metamask:** In order to make the transaction that ensure the authentic owner of the code we need metamask as metamask is compatible with Ethereum and Ethereum compatible network as we are using the Ethereum blockchain from the beginning we are using metamask this wallet has about 21 million active wallets.

4. **Alchemy:** Alchemy is a blockchain infrastructure provider that provides tools and services for developers to build on the blockchain with a good UI and in giving details regarding the traffic of requests and responses, unlike Infura. In this project, Alchemy is used to deploy the smart contract on the Ethereum network. Alchemy provides a user-friendly interface for deploying and interacting with the smart contract instead you can also use the Infura if you wish.

5. **Hardhat:** Hardhat is a development environment for building/deploying and testing smart contracts. In your project,

Hardhat is used to compile, deploy, and test the smart contract. The best thing about the hardhat is hardhat is very easy to use and only the smart contract we have coded is deployed, unlike truffle which deploys migrations files and also which makes the users search for their contract address. Hardhat provides a testing framework that allows developers to test the functionality of the smart contract and if you need to make any changes then you can deploy again and again until you have enough sepolia ether.

6. **Sepolia Test Network (Ethereum):** Sepolia Test Network is a test network of the Ethereum blockchain. In your project, Sepolia Test Network is used to test the functionality of the smart contract before deploying it on the main Ethereum network. The main reason we have chose the sepolia test network because unlike Goerli and Ropsten this works on proof of stake and from the news Ethereum plans to move to proof of stake by the end of this year which makes the condition risky for someone who deployed their smart contract on the goerli and also the faucets plays an important role in order to use a particular test network unlike goerli sepolia provides one with 0.5 sepolia ether per day. This ensures that the smart contract is working as expected and that any bugs or issues are resolved before going live.

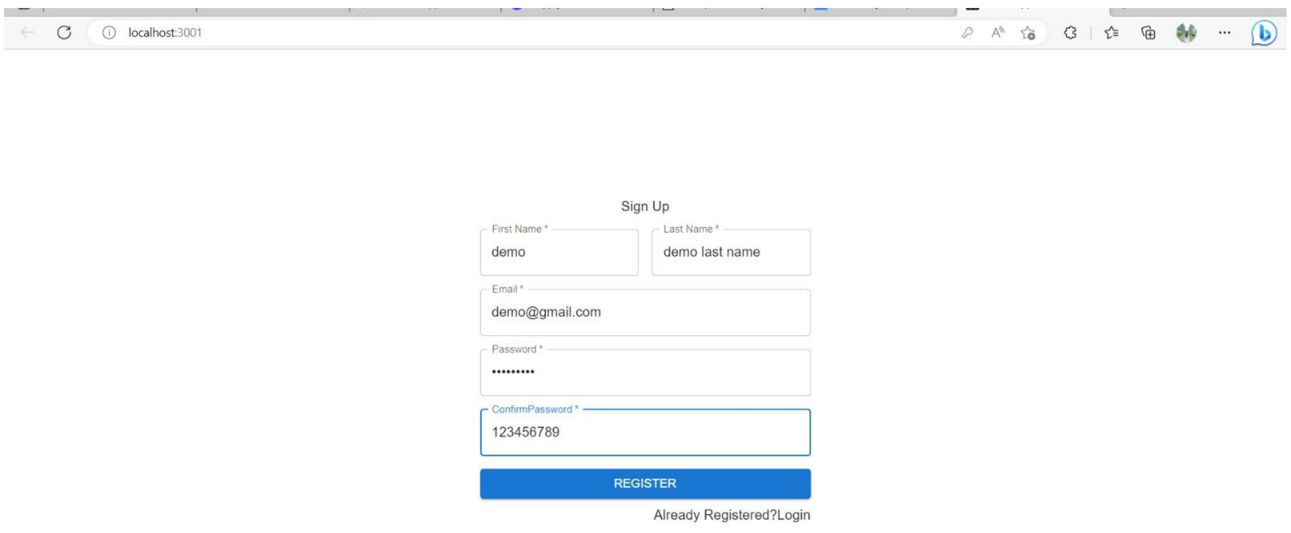
The Open Source Repository provides a secure and decentralized platform for developers to share their work without compromising the integrity of their code. To achieve this work without compromising the integrity of their code. To achieve this, the platform employs a range of technologies. Smart contracts ensure the security and immutability of data stored on the blockchain, while IPFS offers a decentralized file storage system that is protected from hacking attempts due to its peer-to-peer networking capabilities. Metamask provides user authentication and transaction facilitation, while Alchemy and Hardhat offer tools and services for building, deploying, and testing smart contracts. Furthermore, the Sepolia Test Network provides a testing environment for smart contracts before they are deployed on the main

Ethereum network. The platform is designed to be decentralized as much as possible, ensuring that developers who share malicious code are held responsible. If users encounter issues related to transactions or their status, they can contact the developer team, who will address the problem. Overall, the use of these technologies creates a reliable and robust platform for developers to share their work.

4.2 User Interface and Operation Steps:

If the user wants to **access the components** then the following steps are required–

Step 1: The user need to do Signup if they have not been a member of this Open Source repository i.e. new user.


localhost:3001

Sign Up

First Name * demo

Last Name * demo last name

Email * demo@gmail.com

Password * •••••••••

ConfirmPassword * 123456789

Fig 1.1 SignUp page

If the user is already a member of this Open Source repository then they simply need to login.

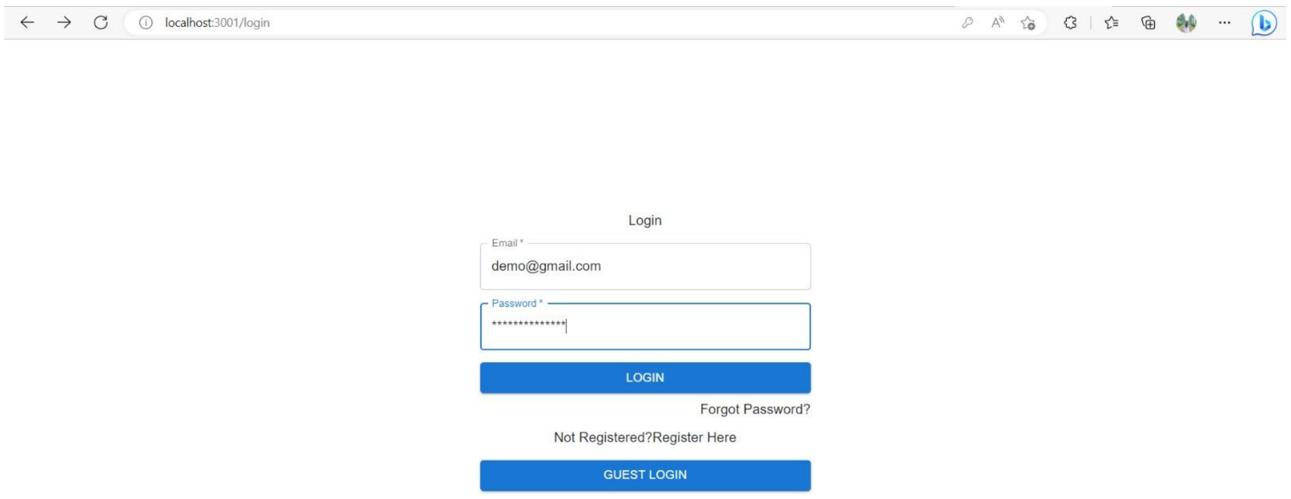
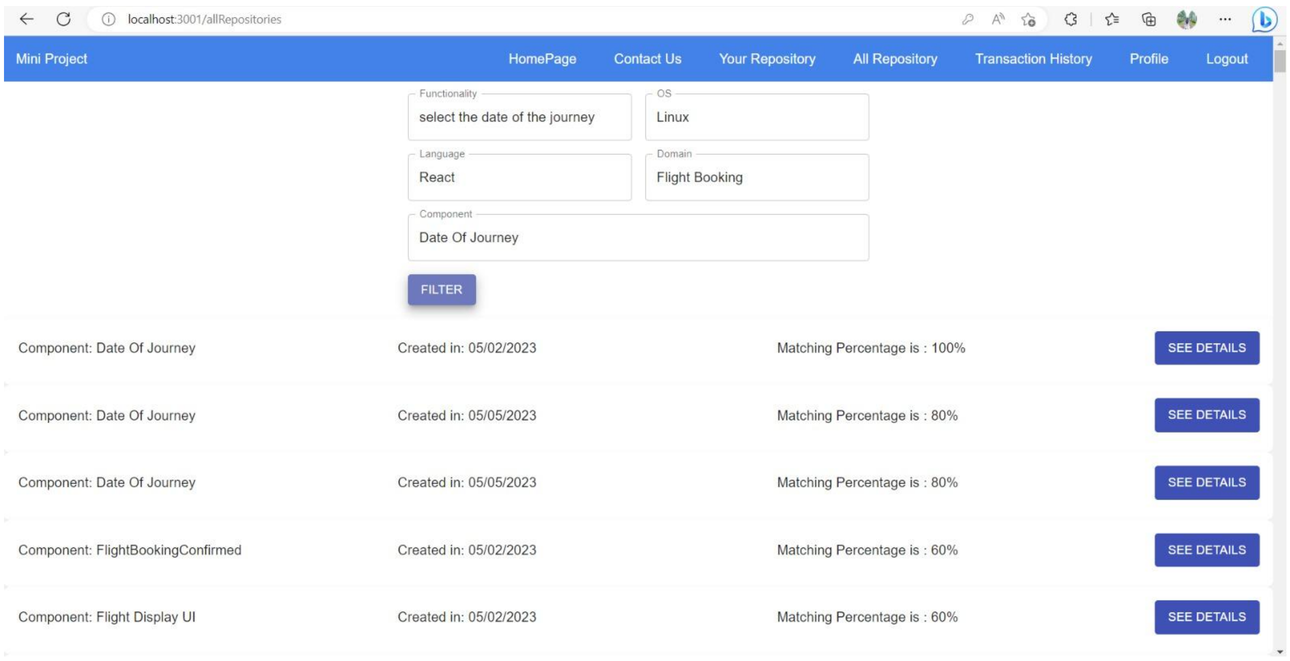


Fig 1.2 Developer login page

Step 2: If the Login is successful then users will be welcomed with 'Main GUI' where they can fill in the requirements like- Components name, OS name, and Framework.

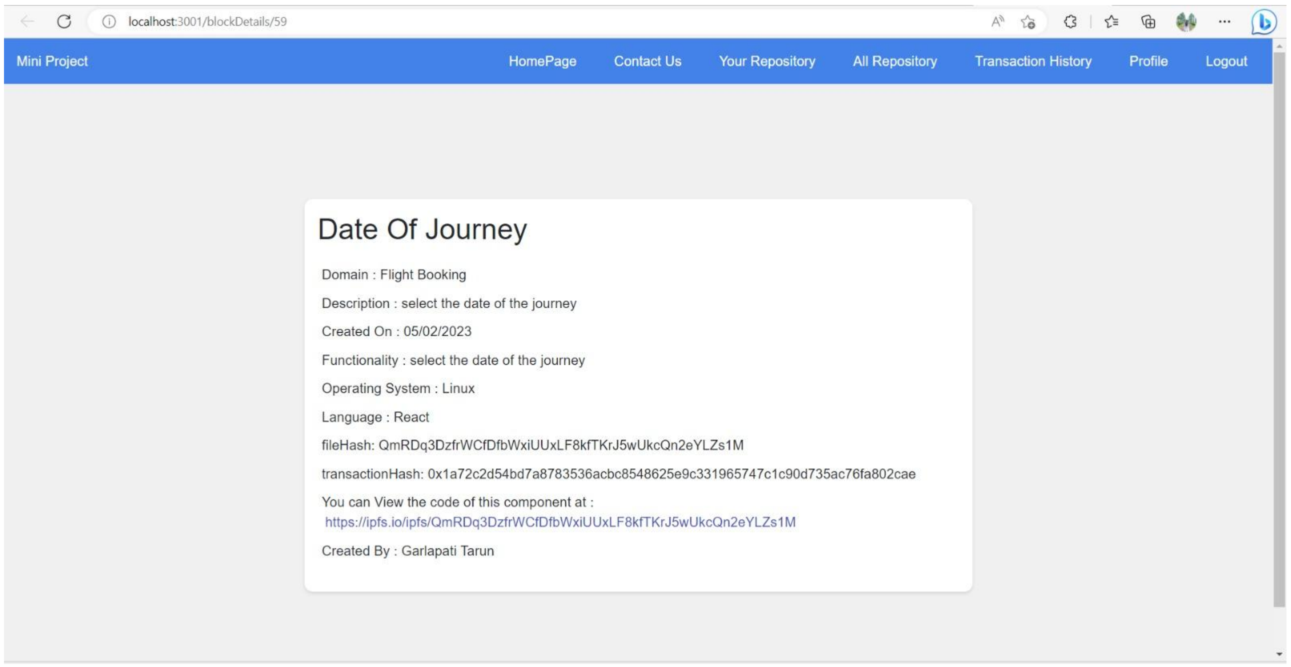


Functionality: select the date of the journey
OS: Linux
Language: React
Domain: Flight Booking
Component: Date Of Journey

Component: Date Of Journey	Created in: 05/02/2023	Matching Percentage is : 100%	SEE DETAILS
Component: Date Of Journey	Created in: 05/05/2023	Matching Percentage is : 80%	SEE DETAILS
Component: Date Of Journey	Created in: 05/05/2023	Matching Percentage is : 80%	SEE DETAILS
Component: FlightBookingConfirmed	Created in: 05/02/2023	Matching Percentage is : 60%	SEE DETAILS
Component: Flight Display UI	Created in: 05/02/2023	Matching Percentage is : 60%	SEE DETAILS

Fig 1.3 Project GUI showing the matching percentage of requests

Step 3: According to the user request, the Repository System matches the percentage of requests and gives the details of matched components.



Date Of Journey

Domain : Flight Booking

Description : select the date of the journey

Created On : 05/02/2023

Functionality : select the date of the journey

Operating System : Linux

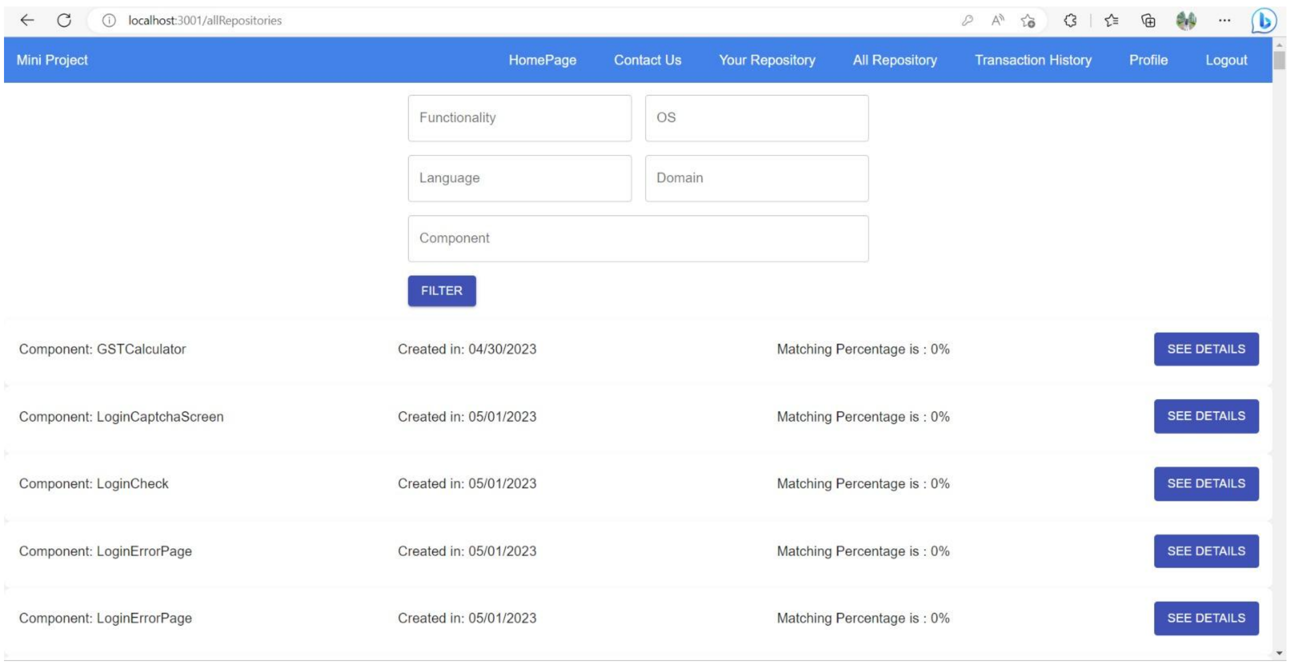
Language : React

fileHash: QmRDq3DzfrWCfDfbWxiUUxLF8kfTKrJ5wUkcQn2eYLZs1M

transactionHash: 0x1a72c2d54bd7a8783536acbc8548625e9c331965747c1c90d735ac76fa802cae

You can View the code of this component at :
https://ipfs.io/ipfs/QmRDq3DzfrWCfDfbWxiUUxLF8kfTKrJ5wUkcQn2eYLZs1M

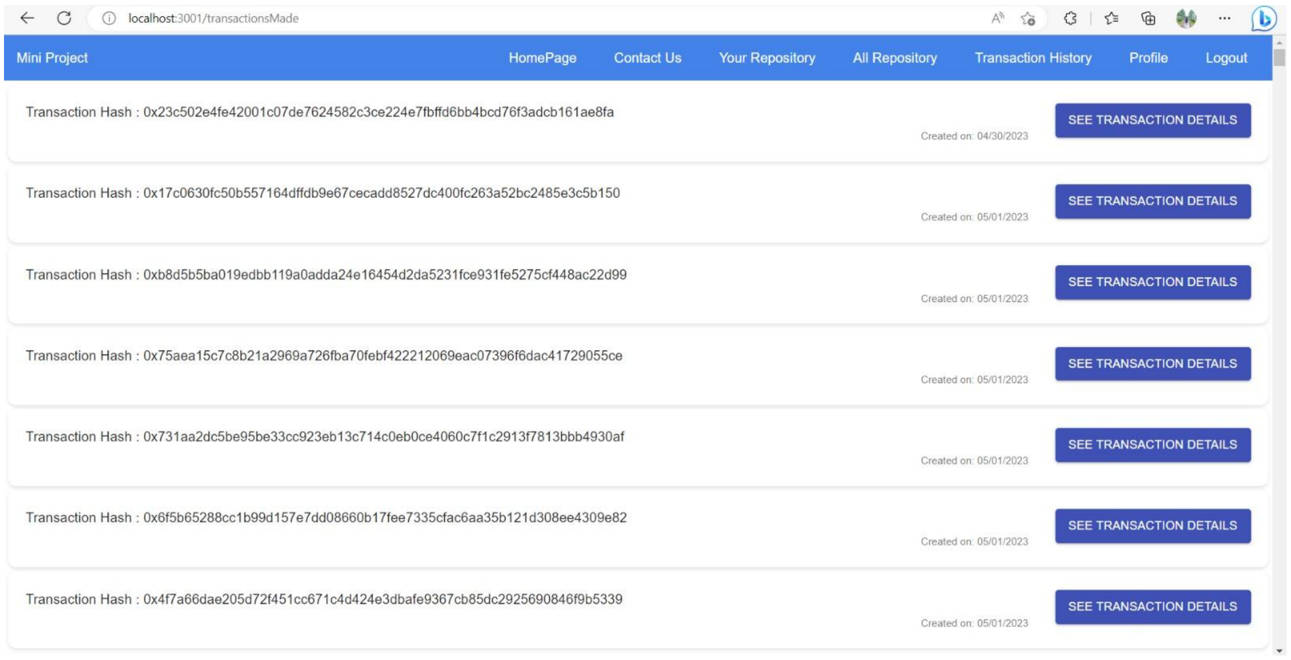
Created By : Garlapati Tarun



Component: GSTCalculator	Created in: 04/30/2023	Matching Percentage is : 0%	SEE DETAILS
Component: LoginCaptchaScreen	Created in: 05/01/2023	Matching Percentage is : 0%	SEE DETAILS
Component: LoginCheck	Created in: 05/01/2023	Matching Percentage is : 0%	SEE DETAILS
Component: LoginErrorPage	Created in: 05/01/2023	Matching Percentage is : 0%	SEE DETAILS
Component: LoginErrorPage	Created in: 05/01/2023	Matching Percentage is : 0%	SEE DETAILS

Fig 1.4 Result according to the request

Note: Users can easily monitor components stored on the IPFS network and their hash is secure via blockchain by pushing on the blockchain network.



Transaction Hash : 0x23c502e4fe42001c07de7624582c3ce224e7fbffd6bb4bcd76f3adcb161ae8fa
Created on: 04/30/2023

Transaction Hash : 0x17c0630fc50b557164dffdb9e67cecadd8527dc400fc263a52bc2485e3c5b150
Created on: 05/01/2023

Transaction Hash : 0xb8d5b5ba019edbb119a0adda24e16454d2da5231fce931fe5275cf448ac22d99
Created on: 05/01/2023

Transaction Hash : 0x75aea15c7c8b21a2969a726fba70febf422212069eac07396f6dac41729055ce
Created on: 05/01/2023

Transaction Hash : 0x731aa2dc5be95be33cc923eb13c714c0eb0ce4060c7f1c2913f7813bbb4930af
Created on: 05/01/2023

Transaction Hash : 0x6f5b65288cc1b99d157e7dd08660b17fee7335cfac6aa35b121d308ee4309e82
Created on: 05/01/2023

Transaction Hash : 0x4f7a66dae205d72f451cc671c4d424e3dbafe9367cb85dc2925690846f9b5339
Created on: 05/01/2023

Fig 1.5 Transaction hash of components

Chapter 5

Conclusion and Future Work

In this project, we have worked on the blockchain and IPFS technology to reduce the software supply chain attack on using third-party components that are present in an un-trust environment.
We have studied various software supply chain attacks that are happening in the past and discussed the working of various tools and technology. This model reduces the risk of supply chain attacks and developers can now create secure software using third-party components.

Our future plan is to work on providing more secure components code by developing a model that checks complete components code before uploading it to the web3.0 application by the third parties and if there is any malicious code in it, then do not let it be uploaded. And this model does not allow two components of the same name to be uploaded. And also another future work is to ensure that there was no attack anywhere in the middle during the download of the components code, the component code that the user is getting is the same component code that was in the IPFS network.

Chapter 6

References

[1] Greenberg, Andy. "The untold story of NotPetya, the most devastating cyberattack in history." *Wired, August* 22 (2018).

[2] Martin, Robert Alan. "Visibility & control: addressing supply chain challenges to trustworthy software-enabled things." *2020 IEEE Systems Security Symposium (SSS)*. IEEE, 2020.

[3] Porru, Simone, et al. "Blockchain-oriented software engineering: challenges and new directions." *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2017.

[4] Della Valle, Fabio, and Miquel Oliver. "Blockchain enablers for supply chains: How to boost implementation in industry." *IEEE access* 8 (2020): 209699-209716.

[5] Marjanović, Jelena, Nikola Dalčeković, and Goran Sladić. "Improving critical infrastructure protection by enhancing software acquisition process through blockchain." *7th Conference on the Engineering of Computer Based Systems*. 2021.

[6] Malik, Sidra, et al. "Trustchain: Trust management in blockchain and iot supported supply chains." *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2019

[7] Giedrimas, Vaidas. "The role of blockchain for increase trust on software components and services." *2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT)*. IEEE, 2020.

[8] Graham, James, Jeffrey Hieb, and John Naber. "Improving cybersecurity for industrial control systems." *2016 ieee 25th international symposium on industrial electronics (isie)*. IEEE, 2016.