verichains

SECURITY AUDIT OF

PROPEASY WEBSITE

Propeasy

Public Report

Jan 12, 2024

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Solana	A decentralized blockchain built to enable scalable, user-friendly apps for the world.
SOL	A cryptocurrency whose blockchain is generated by the Solana platform.
Penetration Testing	Penetration testing (or pentesting) a security exercise where a cyber-security expert attempts to find and exploit vulnerabilities in a computer system
DApp	A Decentralised Application is an application that can operate autonomously, typically through the use of smart contracts, that run on a decentralized computing, blockchain or other distributed ledger system. Like traditional applications, DApps provide some function or utility to its users

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Jan 12, 2024. We would like to thank the Propeasy Labs for trusting Verichains Lab in testing website. Delivering high-quality audits is always our top priority.

This penetration testing (pentest) focused on identifying security flaws in code and the design of the Propeasy website. The scope of the testing is limited to the websites provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified some minor issues in the application.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Propeasy website

The Propeasy website is a DApp for `Propeasy` program. Users can connect their wallet to participate in all the sales of `Propeasy`.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the Propeasy website. It was conducted on the website *https://test.propeasy.org* and commit `9c91b0c9d4eaeeb1d4e0afaf5e8661fd4d9c69b5` from git repository *https://github.com/renec-chain/propeasy*.

Furthermore, the security check is only an immediate evaluation of the situation at the time the check was performed. An evaluation of future security levels or possible future risks or vulnerabilities may not be derived from it.

1.3. Audit methodology

Our penetration testing (pentest) process for website includes two steps:

- Website are scanned/tested for commonly known and more specific vulnerabilities using security pentest tool.
- Manual pentest for security issues. The website is manually tested to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during testing:

For Client side:

- Broken Client-side Access Control
- DOM-based XSS
- Sensitive Data Leakage
- Lack of Third-party Origin Control
- JavaScript Drift
- Sensitive Data Stored Client-Side
- Client-side Security Logging and Monitoring Failures
- Not Using Standard Browser Security Controls
- Including Proprietary Information on the Client-Side

For Server side:

- Broken Access Control

- Cryptographic Failures
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and Authentication Failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery

For API:

- Broken Object Level Authorization
- Broken Authentication
- Broken Object Property Level Authorization
- Unrestricted Resource Consumption
- Broken Function Level Authorization
- Unrestricted Access to Sensitive Business Flows
- Server Side Request Forgery
- Security Misconfiguration
- Improper Inventory Management
- Unsafe Consumption of APIs

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Propeasy Labs acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. Propeasy Labs understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Propeasy Labs agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the Propeasy Labs will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the Propeasy Labs, the final report will be considered fully accepted by the Propeasy Labs without the signature.

2. AUDIT RESULT

2.1. Overview

The Propeasy website was written in `Typescript` programming language and `ReactJS` framework.

During the information gathering, Verichains discover Propeasy website's technologies used as below:

CATEGORY	TECHNOLOGY
JavaScript frameworks	Next.js, React
UI frameworks	MUI
CDN	CloudFront
Miscellaneous	Webpack

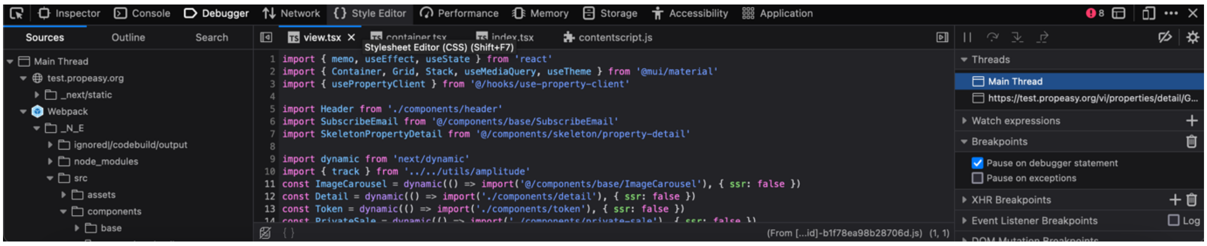
Table 2. The technology list

2.2. Findings

During the audit process, the audit team found some minor issues in the given version of Propeasy website.

2.2.1. Source code leakage due to exposed sourcemap INFORMATIVE

A source map is a file that maps from the transformed source to the original source, enabling the browser to reconstruct the original source and present the reconstructed original in the debugger. This makes debugging easier because it maps the code back to the original source. It was detected that the application is exposing the source map file. This can be used by an attacker to understand the code structure and logic of the application. This can be used to find vulnerabilities in the application.



```
import { memo, useEffect, useState } from 'react'
import { Container, Grid, Stack, useMediaQuery, useTheme } from '@mui/material'
import { usePropertyClient } from '@/hooks/use-property-client'

import Header from './components/header'
import SubscribeEmail from '@/components/base/SubscribeEmail'
import SkeletonPropertyDetail from '@/components/skeleton/property-detail'

import dynamic from 'next/dynamic'
import { track } from '../../utils/amplitude'
const ImageCarousel = dynamic(() => import('@/components/base/ImageCarousel'), { ssr: false })
const Detail = dynamic(() => import('./components/detail'), { ssr: false })
const Token = dynamic(() => import('./components/token'), { ssr: false })
const PrivateSale = dynamic(() => import('./components/private-sale'), { ssr: false })
```

Image 1. Leaked source code

RECOMMENDATION

Disable source map in production environment.

2.2.2. HTTP Strict Transport Security (HSTS) Policy Not Enabled INFORMATIVE

HTTP Strict Transport Security (HSTS) tells a browser that a website is only accessible using HTTPS. It was detected that the application doesn't implement HTTP Strict Transport Security (HSTS) as the Strict Transport Security header is missing from the response. HSTS can be used to prevent and/or mitigate some types of man-in-the-middle (MitM) attacks

RECOMMENDATION

It's recommended to implement HTTP Strict Transport Security (HSTS) into the web application.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Jan 12, 2024*	Public Report	Verichains Lab

Table 3. Report versions history