



Smart Contract Security Audit Report

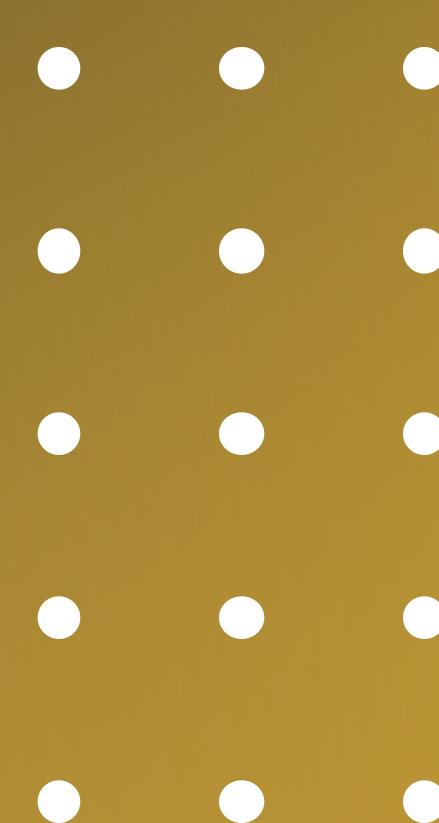
JavaScript

December 2022

Security Status



www.hacksafe.io



Audit Details



Audited project

JavaScript



Deployer address

0x8a75ff000589d114288fa6b7ca9fe51b9fb69283



Client contacts

JavaScript Team



Blockchain

Ethereum



Website

Not provided

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and TechRate and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (HackSafe) owe no duty of care towards you or any other person, nor does HackSafe make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and HackSafe hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HackSafe hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HackSafe, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Procedure

Step 1 - In-Depth Manual Review

Manual line-by-line code reviews to ensure the logic behind each function is sound and safe from various attack vectors. This is the most important and lengthy portion of the audit process (as automated tools often cannot find the nuances that lead to exploits such as flash loan attacks).

Step 2 - Automated Testing

Simulation of a variety of interactions with your Smart Contract on a test blockchain leveraging a combination of automated test tools and manual testing to determine if any security vulnerabilities exist.

Step 3 – Leadership Review

The engineers assigned to the audit will schedule meetings with our leadership team to review the contracts, any comments or findings, and ask questions to further apply adversarial thinking to discuss less common attack vectors.

Step 4 - Resolution of Issues

Consulting with the team to provide our recommendations to ensure the code's security and optimize its gas efficiency, if possible. We assist project team's in resolving any outstanding issues or implementing our recommendations.

Step 5 - Published Audit Report

Boiling down results and findings into an easy-to-read report tailored to the project. Our audit reports highlight resolved issues and any risks that exist to the project or its users, along with any remaining suggested remediation measures. Diagrams are included at the end of each report to help users understand the interactions which occur within the project.

Background

HackSafe was commissioned by to JavaScript perform an audit of smart contracts:

- <https://etherscan.io/token/0x5046e860ff274fb8c66106b0ffb8155849fb0787#code>

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

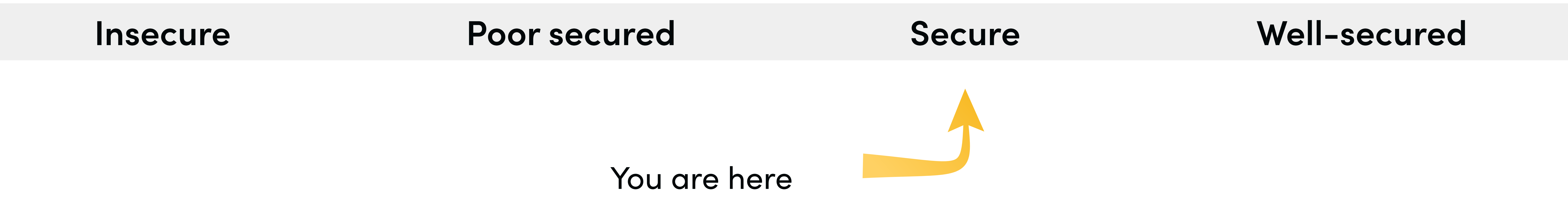
Contract Details

Token contract details for 02.12.2022

Token Type	: DEFI
Contract name	: JavaScriptToken
Contract address	: 0x5046E860ff274fb8c66106B0Ffb8155849fB0787
Total supply	: 7,991,996
Token ticker	: JS
Decimals	: 18
Token Holders	: 1,523
Transactions count	: 7,649
Compiler version	: v0.4.18+commit.9cf6e910
Contract deployer address	: 0x8a75ff000589d114288fa6b7ca9fe51b9fb69283
Owner address	: 0x8a75ff000589d114288fa6b7ca9fe51b9fb69283

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secure”**. This token contract does contain owner control, which do not make it fully decentralized as owner does have control over smart contract.



We used various tools like Slither, Mythril and Remix IDE. At the same time this finding is based on critical analysis of the manual audit. All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the issues checking status.

We found 0 critical, 0 high, 0 medium and 2 low.

Social profiles

Twitter profile	: https://twitter.com/JS_Token
Coinmarket profile	: https://coinmarketcap.com/currencies/javascript-token/
Coingecko profile	: https://www.coingecko.com/en/coins/javascript-token/

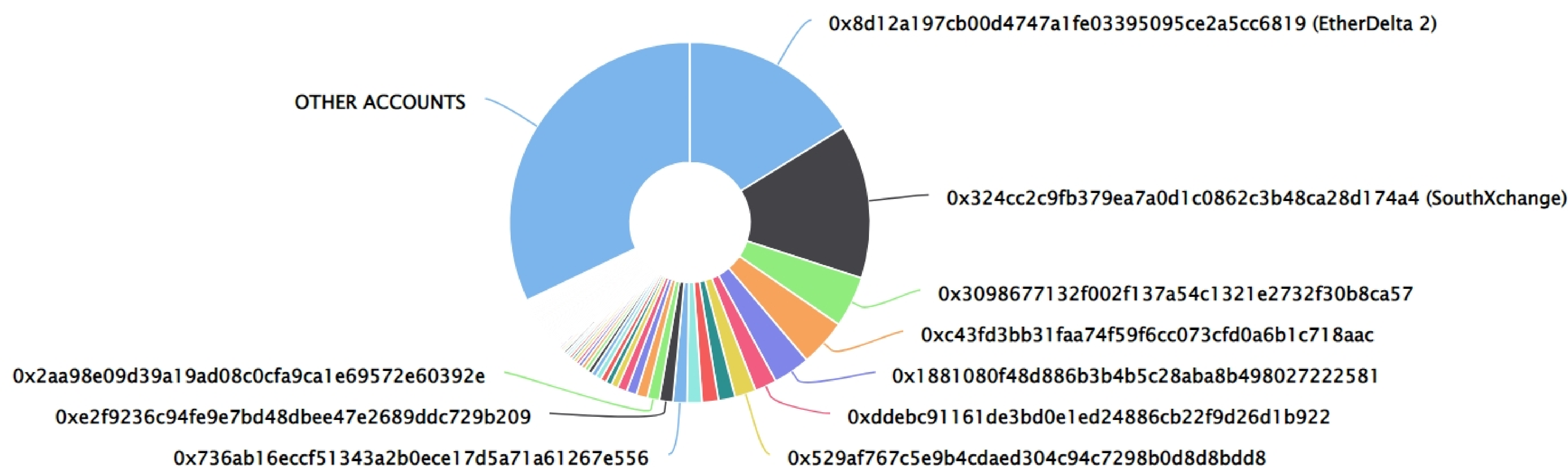
JavaScript Distribution

💡 The top 100 holders collectively own 67.91% (5,427,006.75 Tokens) of JavaScript

💡 Token Total Supply: 7,991,996.00 Token | Total Token Holders: 1,523


JavaScript Top 100 Token Holders

Source: Etherscan.io



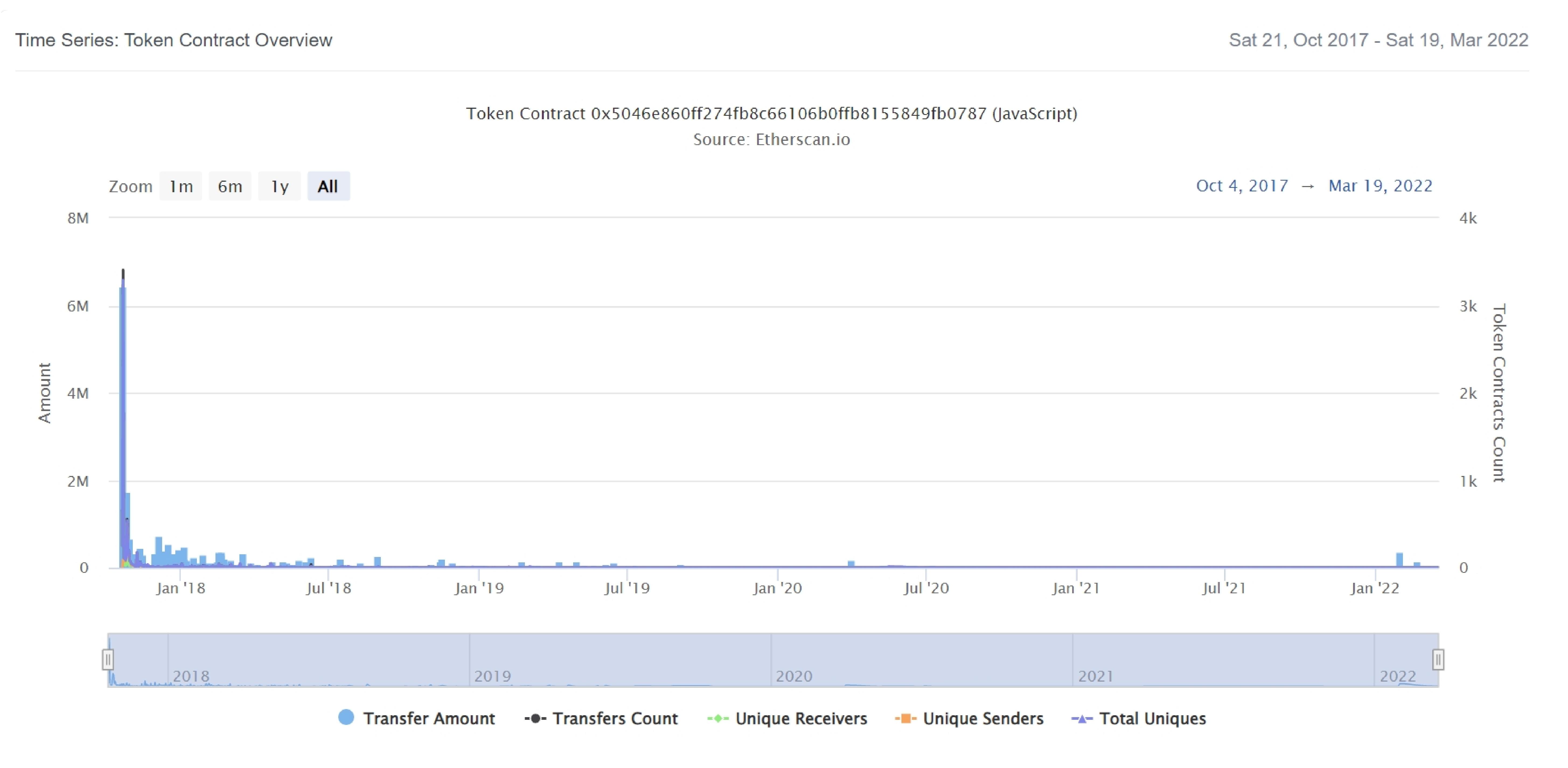
JavaScript Top 20 Token Holders

(A total of 5,427,006.75 tokens held by the top 100 accounts from the total supply of 7,991,996.00 token)

Rank	Address	Quantity (Token)	Percentage
1	 EtherDelta 2	1,294,622.90989122	16.1990%
2	SouthXchange	1,102,388.41892471	13.7937%
3	0x3098677132f002f137a54c1321e2732f30b8ca57	365,122.70659918	4.5686%
4	0xc43fd3bb31faa74f59f6cc073cfd0a6b1c718aac	338,428.201	4.2346%
5	0x1881080f48d886b3b4b5c28aba8b498027222581	265,080	3.3168%
6	0xddebc91161de3bd0e1ed24886cb22f9d26d1b922	155,623.317	1.9472%
7	0x529af767c5e9b4cdaed304c94c7298b0d8d8bdd8	151,361.00582714	1.8939%
8	0xa6b11e0638642d805892b09836ecfc6aeabea340	120,239	1.5045%
9	0x78ad799f6e4fadee929e9eb0f43bfbd210ca1d02	118,742.94670151	1.4858%
10	0x005a693b929c288419b4e1bdb3f9180f56a38ff	105,000	1.3138%
11	0x736ab16eccf51343a2b0ece17d5a71a61267e556	101,000	1.2638%
12	0xe2f9236c94fe9e7bd48dbee47e2689ddc729b209	95,691	1.1973%
13	0x2aa98e09d39a19ad08c0cfa9ca1e69572e60392e	87,214.05052959	1.0913%
14	0x7ada78f3d8db97191bfad4a37b542c2384279ed1	79,799.918	0.9985%
15	0x02a97ed35ba18d2f3c351a1bb5bba12f95eb1181	73,051.83	0.9141%
16	0x02ac05bc35938108fb443f4ae09a903fb924ade2	68,286	0.8544%
17	0x2949c422ff8dd6fe6392313524e4dd45d61188b8	49,160.2865353	0.6151%
18	0xfb5210d964171148c6c59a13ff164f49ae36080f	42,956.866	0.5375%
19	0xefa7abbd5fe2aef47b325619b0824d41835f5d6c	41,985	0.5253%
20	0xd9e26a5e2cc9f4e55ecc51329bb3197791c5ca2e	39,344.27994907	0.4923%

JavaScript Distribution

JavaScript Overview



Contract functions details

+ForeignToken

-[Pub] balanceOf

-[Pub] transfer

+ERC20Basic

-[Pub] balanceOf

-[Pub] transfer

+ERC20 (ERC20Basic)

-[Pub] allowance

-[Pub] transferFrom

-[Pub] approve

+JavaScriptToken (ERC20)

-[Pub] JavaScriptToken #

-[Pub] transferOwnership #

-modifiers: onlyOwner

-[Pub] getEthBalance

-[Pub] distributeJST #

-modifiers: onlyOwner, canDistr

-[Pub] balanceOf

-[Pub] transfer #

-[Pub] transferFrom #

-[Pub] approve #

-[Pub] allowance

-[Pub] finishDistribution #

-modifiers: onlyOwner

-[Pub] withdrawForeignTokens #

(\$) = payable function

= non-constant function

Issues Checking Status

No.	Title	Status
1.	Unlocked Compiler Version	Low issue
2.	Missing Input Validation	Passed
3.	Race conditions and Reentrancy. Cross-function race conditions.	Passed
4.	Possible delays in data delivery	Passed
5.	Oracle calls.	Passed
6.	Timestamp dependence.	Passed
7.	Integer Overflow and Underflow	Passed
8.	DoS with Revert.	Passed
9.	DoS with block gas limit.	Passed
10.	Methods execution permissions.	Passed
11.	Economy model of the contract.	Passed
12.	Private use data leaks.	Passed
13.	Malicious Event log.	Passed
14.	Scoping and Declarations.	Passed
15.	Uninitialized storage pointers.	Passed
16.	Arithmetic accuracy.	Passed
17.	Design Logic.	Passed
18.	Safe Open Zeppelin contracts implementation and usage.	Passed
19.	Incorrect Naming State Variable	Passed
20.	Too old version	Low issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution.

Security Issues

✔ Critical Severity Issues

No critical severity issue found.

✔ High Severity Issues

No high severity issue found.

✔ Medium Severity Issues

No medium severity issue found.

✔ Low Severity Issues

Two low severity issue found.

1. Unlocked Compiler Version.

• Description

The contract utilizes an unlocked compiler version. An unlocked compiler version in the contract's source code permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

• Recommendation

It is advisable that the compiler version is alternatively locked at the lowest version possible so that the contract can be compiled. For example, for version ^0.4.16 the contract should contain the following line:

```
pragma solidity 0.4.18;
```

2. Old compiler version

• Description

Contract has been deployed using too old solidity version.

• Recommendation

It is advisable to deploy contract using any of the latest version of solidity

Centralization

Owner Privileges :

- JavaScript Contract:
 - Owner can renounce and transfer ownership.
 - Owner can distribute tokens.
 - Owner can finish distribution.

This smart contract has some functions which can be executed by the admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble as smart contract ownership has not been renounced. Following are Admin functions:

- transferOwnership
- distributeJST
- finishDistribution

Conclusion

Smart contract contains low severity issues! The further transfer and operations with the fund raised are not related to this particular contract.

HackSafe note: Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.