d3ploy

d3ploy

SECURITY ASSESSMENT

EthosX

January 21st 2023

TABLE OF

Contents

LEGAL

Disclaimer

D3ploy audits are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts d3ploy to perform a security review. D3ploy does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

D3ploy audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort. The report is provided only for the contract(s) mentioned in the report and does not include any other potential additions and/or contracts deployed by Owner. The report does not provide a review for contract(s), applications and/or operations, that are out of this report scope.

D3ploy’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

D3ploy represents an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. D3ploy’s position is that each company and individual are responsible for their own due diligence and continuous security. The security audit is not meant to replace functional testing done before a software release. As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent manual audits and a public bug bounty program to ensure the security of the smart contracts.

D3PLOY

Introduction

D3ploy is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

Secure your project with d3ploy

We offer field-proven audits with in-depth reporting and a range of suggestions to improve and avoid contract vulnerabilities. Industry-leading comprehensive and transparent smart contract auditing on all public and private blockchains.

- Vunerability checking

  A crucial manual inspection carried out to eliminate any code flaws and security loopholes. This is vital to avoid vulnerabilities and exposures incurring costly errors at a later stage.

- Contract verification

  A thorough and comprehensive review in order to verify the safety of a smart contract and ensure it is ready for launch and built to protect the end-user

- Risk assessment

  Analyse the architecture of the blockchain system to evaluate, assess and eliminate probable security breaches. This includes a full assessment of risk and a list of expert suggestions.

- In-depth reporting

  A truly custom exhaustive report that is transparent and depicts details of any identified threats and vulnerabilities and classifies those by severity.

- Fast turnaround

  We know that your time is valuable and therefore provide you with the fastest turnaround times in the industry to ensure that both your project and community are at ease.

- Best-of-class blockchain engineers

  Our engineers combine both experience and knowledge stemming from a large pool of developers at our disposal. We work with some of the brightest minds that have audited countless smart contracts over the last 4 years.

PROJECT

Introduction

EthosX is building infrastructure for trading financial derivatives on blockchains for crypto companies and financial institutions.

EthosX is a decentralized finance platform creating end-to-end financial derivatives on blockchains. No centralized exchanges, clearinghouses, depositories, clearing banks, CSD participants etc. required. Starting with cryptocurrency options first, with the intention to move towards other crypto derivatives and traditional finance derivatives eventually.

Project Name *EthosX*

Contract Name *-*

Contract Address *-*

Contract Chain *Not Yet Deployed on Mainnet*

Contract Type *Smart Contract*

Platform *EVM*

Language *Solidity*

Network *Ethereum (ERC20)*

Codebase *Private GitHub Repository*

Total Token Supply *-*

INFO

Social

https://ethosx.xyz/

https://twitter.com/ethosx_finance

-

https://discord.gg/a2QNzdC4VB

-

https://github.com/ethosx-yc/

support@ethosx.xyz

95

PASS

AUDIT

Score

✦ Issues	5
✦ Critical	0
✦ Major	0
✦ Medium	1
✦ Low	4
✦ Informational	0
✦ Discussion	0

All issues are described in further detail on the following pages.

AUDIT Scope

CODEBASE FILES	LOCATION
ethosx-yc/otc-options/packages/hardhat/contracts /Derivatives/	✦ Private GitHub Repository
ethosx-yc/otc-options/packages/hardhat/contracts /Factories/	✦ Private GitHub Repository
ethosx-yc/otc-options/packages/hardhat/contracts /FactoryManager/	✦ Private GitHub Repository
ethosx-yc/otc-options/packages/hardhat/contracts /PriceFeed/	✦ Private GitHub Repository
ethosx-yc/otc-options/packages/hardhat/contracts /SettlementManager/	✦ Private GitHub Repository
ethosx-yc/otc-options/packages/hardhat/contracts /ThirdParty/	✦ Private GitHub Repository
ethosx-yc/otc-options/packages/hardhat/contracts /Tokens/	✦ Private GitHub Repository

REVIEW Methodology

TECHNIQUES

This report has been prepared for EthosX to discover issues and vulnerabilities in the source code of the EthosX project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic, Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from major to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective in the comments below.

TIMESTAMP

Version	*v1.0*
Date	*2022/12/23*
Descrption	*Layout project*
	Architecture / Manual review / Static & dynamic security testing
	Summary

Version	*v1.1*
Date	*2023/01/21*
Descrption	*Reaudit addressed issues*
	Final Summary

KEY Finding

TITLE	SEVERITY	STATUS
Floating and Outdated Pragma	✦ Low	Fixed
Missing Multiple Zero Address Validations	✦ Low	Fixed
Approve Fruntrunning Attack	✦ Medium	Fixed
Missing Events in Important Functions	✦ Low	Fixed
Missing Input Validation in Payoffs and Premiums	✦ Low	Fixed

IN - DEPTH Vulnerabilities

DESCRIPTION

Locking the pragma helps ensure that the contracts do not accidentally get deployed using an older version of the Solidity compiler affected by vulnerabilities.
The contracts found in the repository allowed floating or unlocked pragma to be used, i.e., ^0.8.0.
This allows the contracts to be compiled with all the solidity compiler versions above 0.8.0.

AFFECTED CODE

- ChainlinkKeeperFactory/ChainlinkKeeperFactory.sol L02
- Derivatives/Misc/DerivativeEnums.sol L03
- Derivatives/Primitives/BaseContracts/BarrierOptions.sol L03
- Derivatives/Primitives/BaseContracts/DigitalOptions.sol L03
- Derivatives/Primitives/BaseContracts/KeeperCallable.sol L03
- Derivatives/Primitives/BaseContracts/RampedOptions.sol L03
- Derivatives/Primitives/BaseContracts/Trustless.sol L03
- Derivatives/Primitives/BaseContracts/TrustlessOptions.sol L03
- Derivatives/Primitives/DeployableContracts/TrustlessAmericanDoubleBarrierOptions.sol L03
- Derivatives/Primitives/DeployableContracts/TrustlessAmericanKnockInOptions.sol L03
- Derivatives/Primitives/DeployableContracts/TrustlessAmericanKnockOutOptions.sol L03
- Derivatives/Primitives/DeployableContracts/TrustlessAmericanOptions.sol L03
- Derivatives/Primitives/DeployableContracts/TrustlessEuropeanKnockInOptions.sol L03
- Derivatives/Primitives/DeployableContracts/TrustlessEuropeanKnockOutOptions.sol L03

Issue : Floating and Outdated Pragma

Type : Floating Pragma (SWC-103)
https://swcregistry.io/docs/SWC-103

Level : Low

Remediation : Keep the compiler versions consistent in all the smart contract files.
Do not allow floating pragmas anywhere.

Alleviation / Retest : Strict Pragma is in use with version 0.8.4

IMPACTS

If the smart contract gets compiled and deployed with an older or too recent version of the solidity compiler, there's a chance that it may get compromised due to the bugs present in the older versions or unidentified exploits in the new versions.
Incompatibility issues may also arise if the contract code does not support features in other compiler versions, therefore, breaking the logic. The likelihood of exploitation is really low.

IN - DEPTH Vulnerabilities

1-2

- Derivatives/Primitives/DeployableContracts/TrustlessEuropeanOptions.sol L03
- Derivatives/Structures/BaseContracts/TrustlessStructures.sol L02
- Derivatives/Structures/DeployableContracts/Collars.sol L03
- Factories/DigitalOptionFactory.sol L03
- Factories/FactoryBase.sol L03
- Factories/RampedOptionFactory.sol L03
- FactoryManager/DigitalOptionFactoryManager.sol L03
- FactoryManager/RampedOptionFactoryManager.sol L03
- PriceFeed/ITellor.sol L02
- PriceFeed/InstrumentRegistry.sol L03
- PriceFeed/PriceFeed.sol L03
- PriceFeed/PriceFeedTester.sol L03
- PriceFeed/TellorCaller.sol L03
- SettlementManager/SettlementManager.sol L02
- ThirdParty/Actions/ActionTokenApprove.sol L03
- ThirdParty/Actions/ActionTokenTransfer.sol L03
- ThirdParty/Actions/Liquity/ActionLiquityClose.sol L02
- ThirdParty/Actions/Liquity/ActionLiquityOpen.sol L02
- ThirdParty/Actions/Liquity/ActionLiquityPayback.sol L03
- ThirdParty/Actions/Liquity/ActionLiquityPaybackClose.sol L02
- ThirdParty/Addresses/Addresses.sol L03
- ThirdParty/Addresses/GoerliAddresses.sol L03
- ThirdParty/Addresses/LocalhostAddresses.sol L03
- ThirdParty/DSProxy/DS/DSAuth.sol L03
- ThirdParty/DSProxy/DS/DSGuard.sol L03
- ThirdParty/DSProxy/DS/DSNote.sol L03
- ThirdParty/DSProxy/DS/DSProxy.sol L03
- ThirdParty/DSProxy/ProxyAuth.sol L02
- ThirdParty/DSProxy/ProxyPermission.sol L03
- ThirdParty/DSProxy/TestAction.sol L03
- ThirdParty/DSProxy/TestTarget.sol L03
- ThirdParty/Interfaces/IDSProxy.sol L03
- ThirdParty/Interfaces/IERC20.sol L03
- ThirdParty/Interfaces/IWETH.sol L03
- ThirdParty/Interfaces/Liquity/IBorrowerOperations.sol L03
- ThirdParty/Interfaces/Liquity/ITroveManager.sol L03
- ThirdParty/Mock/Liquity/MockBorrowerOperations.sol L04
- ThirdParty/Mock/Liquity/MockLUSD.sol L02
- ThirdParty/Mock/Liquity/MockTroveManager.sol L03
- ThirdParty/SettlementManagers/Liquity/SettlementManagerLiquityDigital.sol L02
- ThirdParty/SettlementManagers/Liquity/SettlementManagerLiquityRamped.sol L02
- ThirdParty/Utils/Address.sol L03
- ThirdParty/Utils/SafeERC20.sol L03
- ThirdParty/Utils/SortedCache.sol L03
- ThirdParty/Utils/SortedCache1000.sol L03
- ThirdParty/Utils/TokenUtils.sol L03
- Tokens/ProxyUSDC.sol L02

IN - DEPTH Vulnerabilities

DESCRIPTION

Multiple Solidity contracts were found to be setting new addresses without proper validations for zero addresses.

Address type parameters should include a zero-address check otherwise contract functionality may become inaccessible or tokens burned forever.

Depending on the logic of the contract, this could prove fatal and the users or the contracts could lose their funds, or the ownership of the contract could be lost forever.

AFFECTED VARIABLES

- ChainlinkKeeperFactory/ChainlinkKeeperFactory.sol - _registrar L147
- Derivatives/Primitives/BaseContracts/TrustlessOptions.sol - buyer_ L132
- Derivatives/Primitives/BaseContracts/TrustlessOptions.sol - seller_ L136
- Derivatives/Primitives/BaseContracts/TrustlessOptions.sol - settlement_manager_ L147
- Derivatives/Primitives/BaseContracts/TrustlessOptions.sol - fees_collector_ L166
- Derivatives/Primitives/BaseContracts/Trustless.sol - admin_ L21
- Derivatives/Structures/DeployableContracts/Collars.sol - buyer_ L126
- Derivatives/Structures/DeployableContracts/Collars.sol - seller_ L133
- Derivatives/Structures/DeployableContracts/Collars.sol - settlement_manager_ L144
- ThirdParty/DSProxy/DS/DSAuth.sol - owner_ L28

Issue : Missing Multiple Zero Address Validations

Type : Missing Input Validation

Level : Low

Remediation : Add a zero address validation to all the functions where addresses are being set.

Alleviation / Retest : Zero address validations have been added.

IMPACTS

If address type parameters do not include a zero-address check, contract functionality may become unavailable or tokens may be burned permanently.

IN - DEPTH Vulnerabilities

DESCRIPTION

The contract ActionTokenApprove is defining an approve function inside _lusdApprove(), which is vulnerable to a front-running attack.
Approve is well known to be vulnerable to front-running attacks. This may be exploited in cases where in case the user decides to modify the spending amount in quick succession, and the attacker sees the change and transfers more tokens than required.
The exploitation involves two parties - a victim and an attacker and the attacker must be monitoring the transactions to front-run approval calls.

VULNERABLE CODE

- ThirdParty/Actions/ActionsTokenApprove.sol L43

IMPACTS

Front-running attacks might allow attackers to front-run a user transaction and withdraw/transfer more tokens than the victim initially intended to allow.

Issue : Approve Fruntrunning Attack

Type : Frontrunning

Level : Medium

Remediation : Use safeApprove() instead of approve(). safeApprove() should only be called when setting an initial allowance or when resetting it to 0. it is recommended to use safeIncreaseAllowance() and safeDecreaseAllowance() to increase and decrease it.

https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit

Alleviation / Retest : Now a tokenapprove function is used which uses safeApprove()

IN - DEPTH Vulnerabilities

DESCRIPTION

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log — a special data structure in the blockchain. These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract was found to be missing these events on certain critical functions which would make it difficult or impossible to track these transactions off-chain.

AFFECTED CODE

The following functions were affected

- ChainlinkKeeperFactory/ChainlinkKeeperFactory.sol - allowAccess() L206
- ChainlinkKeeperFactory/ChainlinkKeeperFactory.sol - revokeAccess() L210
- Derivatives/Primitives/BaseContracts/TrustlessOptions.sol - setSettlementManager() L147
- Derivatives/Primitives/BaseContracts/TrustlessOptions.sol - setFees() L156
- Derivatives/Primitives/BaseContracts/TrustlessOptions.sol - setFeesCollector() L166

IMPACTS

Events are used to track the transactions off-chain and missing these events on critical functions makes it difficult to audit these logs if they're needed at a later stage.

Issue : Missing Events in Important Functions

Type : Missing Best Practices

Level : Low

Remediation : Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Alleviation / Retest : The functions mentioned above are now emitting events.

IN - DEPTH Vulnerabilities

DESCRIPTION

The functions to set premiums and payoffs lacked a 0 value input validation allowing buyers and sellers to set the amount to 0.

AFFECTED CODE

The following functions were affected

- Derivatives/Primitives/DeployableContracts/TrustlessAmericanOptions.sol - setPayoff() L122
- Derivatives/Primitives/DeployableContracts/TrustlessEuropeanOptions.sol - setPayoff() L103
- Derivatives/Primitives/BaseContracts/TrustlessOptions.sol - setPremium() L175

IMPACTS

Payoffs and premium amounts should always have a boundary value validation so that buyers or sellers can’t set it to 0.

Issue : Missing Input Validation in Payoffs and Premiums

Type : Missing Input Validation

Level : Low

Remediation : Implement boundary value validations on the functions to set payoffs and premiums.

Alleviation / Retest : The function has been removed from the code.

SOURCE Code

Private GitHub Repository

REPORT Appendix

FINDING CATEGORIES

The assessment process will utilize a mixture of static analysis, dynamic analysis, in-depth manual review and/or other security techniques.

This report has been prepared for EthosX project using the above techniques to examine and discover vulnerabilities and safe coding practices in EthosX's smart contract including the libraries used by the contract that are not officially recognized.

A comprehensive static and dynamic analysis has been performed on the solidity code in order to find vulnerabilities ranging from minor gas optimizations to major vulnerabilities leading to the loss of funds.

Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The testing methods find and flag issues related to gas optimizations that help in reducing the overall gas cost It scans and evaluates the codebase against industry best practices and standards to ensure compliance It makes sure that the officially recognized libraries used in the code are secure and up to date.

AUDIT SCORES

D3ploy Audit Score is not a live dynamic score. It is a fixed value determined at the time of the report issuance date.

D3ploy Audit Score is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports and scores are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts d3ploy to perform a security review.

d3ploy

WEBSITE d3ploy.co

@d3ploy_ TWITTER