expelee

Building the Futuristic Blockchain Ecosystem

Security Audit Report

FOR

Crypter

OVERVIEW

The Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result	Passed
KYC Verification	Done
Audit Date	13 Feb 2023

Audit Passed With Low Risk

-Team Expelee

PROJECT DESCRIPTION

Crypter

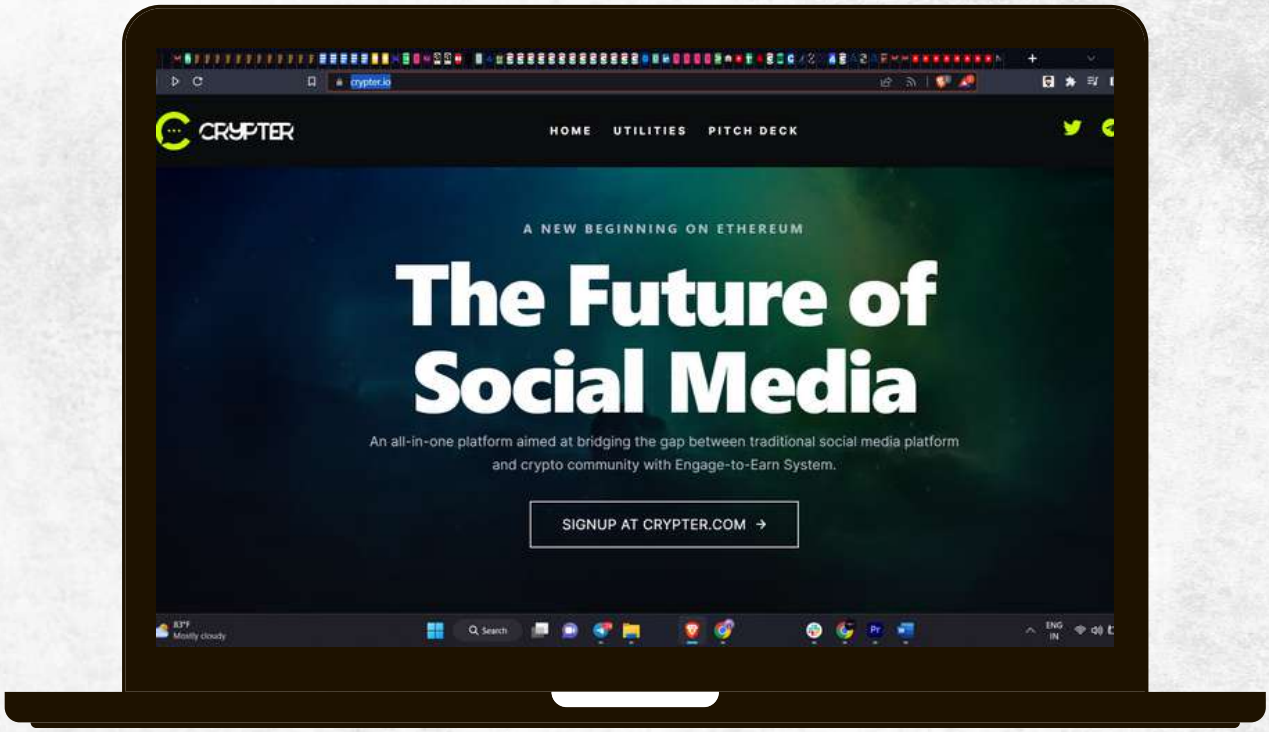
An all-in-one platform aimed at bridging the gap between traditional social media platform and crypto community with Engage-to-Earn System.

expelee

expelee.com

Social Media Profiles

Crypter



https://crypter.io/

https://t.me/cryptermain

https://twitter.com/CrypterOfficial

It's always good to check the social profiles of the project, before making your investment.

-Team Expelee

CONTRACT DETAILS

Token Name
CrypterToken

Symbol
CRYPT

Network
Ethereum

Language
Solidity

Contract Address (Verified)
0x33d40A6876ef79eB93ab78957a47227738A4f5a1

Token Type
ERC20

Total Supply
100,000,000,000

Compiler
v0.8.17+commit.8df45f5f

Optimization Enabled
Yes with 200 runs

Contract SHA-256 Checksum:
fdee430d40bd57deeac186cd9790033d0f06f909a8806e7ce6e717ab7c7d5029

Owner's Wallet
0xCb26C19f0D203D2d364DE741d19580590E96e729

Deployer's Wallet
0xCb26C19f0D203D2d364DE741d19580590E96e729

AUDIT METHODOLOGY

Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch, that lead to scams and rugpulls.

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

Tools

- DE
- Open Zeppelin
- Code Analyzer
- Solidity Code
- Complier
- Hardhat

FUNCTION OVERVIEW

Can Take Back Ownership	Not Detected
Owner Change Balance	Not Detected
Blacklist	Not Detected
Modify Fees	Not Detected
Proxy	Not Detected
Whitelisted	Not Detected
Anti Whale	Not Detected
Trading Cooldown	Not Detected
Transfer Pausable	Not Detected
Cannot Sell All	Not Detected
Hidden Owner	Not Detected
Mint	Not Detected

VULNERABILITY CHECKLIST

Design Logic	Passed
Compiler warnings.	Passed
Private user data leaks	Passed
Timestamp dependence	Passed
Integer overflow and underflow	Passed
Race conditions & reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious Event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zeppelin module	Passed
Fallback function security	Passed

RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and access control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Low Risk

Issues on this level are minor details and warning that can remain unfixed.

Informational

Information level is to offer suggestions for improvement of efficacy or security for features with a risk free factor.

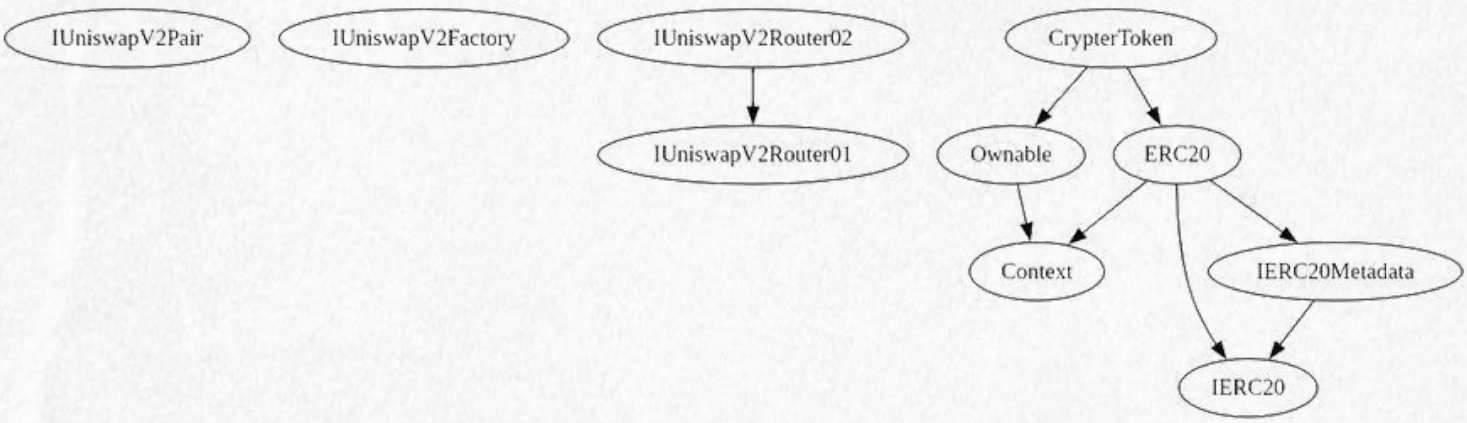
AUDIT SUMMARY

Used Tools:

Manual Review - a line by line code review has been performed by expelee team.

Slither : static analysis

Inheritance Trees:



Summary:

- Owner is not able to set buy/sell/transfer taxes over 10% (buy = sell = transfer tax always)
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to set max buy/sell/transfer amounts
- Owner is not able to disable trades
- Owner is not able to mint new tokens

Features

Static Analysis

A static analysis of contract's source code has been performed using slither. No major issues were found in the output

```
Parameter CrypterToken.swapToStable(uint256,address)._receiver (contracts/TestToken.sol#942) is not in mixedCase
Parameter CrypterToken.swapToETH(uint256,address)._amount (contracts/TestToken.sol#959) is not in mixedCase
Parameter CrypterToken.swapToETH(uint256,address)._receiver (contracts/TestToken.sol#959) is not in mixedCase
Parameter CrypterToken.setExcludeFromFee(address,bool)._account (contracts/TestToken.sol#991) is not in mixedCase
Parameter CrypterToken.setExcludeFromFee(address,bool)._bool (contracts/TestToken.sol#991) is not in mixedCase
Parameter CrypterToken.updateMinSwapAmount(uint256)._minSwapAmount (contracts/TestToken.sol#996) is not in mixedCase
Parameter CrypterToken.updateSTABLE(address)._stable (contracts/TestToken.sol#1001) is not in mixedCase
Parameter CrypterToken.enableSwapToken(bool,bool)._enabled (contracts/TestToken.sol#1007) is not in mixedCase
Parameter CrypterToken.enableSwapToken(bool,bool)._enabledManual (contracts/TestToken.sol#1007) is not in mixedCase
Parameter CrypterToken.setTaxAddresses(address,address,address)._staking (contracts/TestToken.sol#1013) is not in mixedCase
Parameter CrypterToken.setTaxAddresses(address,address,address)._rewardWallet (contracts/TestToken.sol#1013) is not in mixedCase
Parameter CrypterToken.setTaxAddresses(address,address,address)._marketingWallet (contracts/TestToken.sol#1013) is not in mixedCase
Parameter CrypterToken.setTaxFee(uint8,uint8,uint8)._stakingFee (contracts/TestToken.sol#1025) is not in mixedCase
Parameter CrypterToken.setTaxFee(uint8,uint8,uint8)._teamFee (contracts/TestToken.sol#1025) is not in mixedCase
Parameter CrypterToken.setTaxFee(uint8,uint8,uint8)._marketingFee (contracts/TestToken.sol#1025) is not in mixedCase
Parameter CrypterToken.withdrawETH(uint256)._amount (contracts/TestToken.sol#1035) is not in mixedCase
Parameter CrypterToken.recoverERC20(address,uint256)._token (contracts/TestToken.sol#1045) is not in mixedCase
Parameter CrypterToken.recoverERC20(address,uint256)._amount (contracts/TestToken.sol#1045) is not in mixedCase
Variable CrypterToken.STABLE (contracts/TestToken.sol#812) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in CrypterToken._transfer(address,address,uint256) (contracts/TestToken.sol#860-903):
        External calls:
        - _sender.transfer(gasUsed) (contracts/TestToken.sol#880)
        - _sender.transfer(totalEth) (contracts/TestToken.sol#882)
        State variables written after the call(s):
        - super._transfer(sender,recipient,amount) (contracts/TestToken.sol#887)
                - _balances[from] = fromBalance - amount (contracts/TestToken.sol#628)
                - _balances[to] += amount (contracts/TestToken.sol#631)
        - super._transfer(sender,address(this),feeAmount) (contracts/TestToken.sol#899)
                - _balances[from] = fromBalance - amount (contracts/TestToken.sol#628)
                - _balances[to] += amount (contracts/TestToken.sol#631)
        - super._transfer(sender,recipient,sendAmount) (contracts/TestToken.sol#901)
                - _balances[from] = fromBalance - amount (contracts/TestToken.sol#628)
                - _balances[to] += amount (contracts/TestToken.sol#631)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (contracts/TestToken.sol#634)
                - super._transfer(sender,recipient,sendAmount) (contracts/TestToken.sol#901)
        - Transfer(from,to,amount) (contracts/TestToken.sol#634)
                - super._transfer(sender,address(this),feeAmount) (contracts/TestToken.sol#899)
        - Transfer(from,to,amount) (contracts/TestToken.sol#634)
                - super._transfer(sender,recipient,amount) (contracts/TestToken.sol#887)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/TestToken.sol#87) is too similar to IUnisw
apV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/TestToken.sol#88)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

CrypterToken.slitherConstructorVariables() (contracts/TestToken.sol#780-1055) uses literals with too many digits:
        - initialSupply = 100000000000000000000000000000 (contracts/TestToken.sol#803)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

CrypterToken.initialSupply (contracts/TestToken.sol#803) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

CrypterToken.uniswapV2Pair (contracts/TestToken.sol#783) should be immutable
CrypterToken.uniswapV2Router (contracts/TestToken.sol#782) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

MANUAL AUDIT

Severity Criteria

Expelee assesses the severity of disclosed vulnerabilities according to a methodology based on OWASP standards.

Vulnerabilities are divided into three primary risk categories: high, medium, and low.

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious Input Handling
- Escalation of privileges
- Arithmetic
- Gas use

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

FINDINGS

- High Risk Findings:0
- Medium Risk Findings:0
- Low Risk Findings:0
- Suggestions & discussion: 0
- Gas Optimizations : 0

ABOUT EXPELEE

Expelee is a product-based aspirational Web3 Start-up. Coping up with numerous solutions for blockchain Security and constructing a Web3 Ecosystem from Deal making platform to developer hosting open platform, while also developing our own commercial and sustainable blockchain.

www.expelee.com

expeleeofficial

expelee

Expelee

expelee

expelee_official

expelee-co

expelee

Building the Futuristic **Blockchain Ecosystem**

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always Do your own research and protect yourselves from being scammed. The Expelee team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools.

Under no circumstances did Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams.

This document should not be presented as a reason to buy or not buy any particular token. The Expelee team disclaims any liability for the resulting losses.