Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Zexus Finance
Domain:	Zexus.finance
Platform:	Ethereum
Language:	Solidity
Date:	October 27th, 2023

Table of contents

Introduction 4
Project Background 4
Audit Scope 5
Claimed Smart Contract Features 6
Audit Summary 9
Technical Quick Stats 10
Code Quality 11
Documentation 11
Use of Dependencies 11
AS-IS overview 12
Severity Definitions 16
Audit Findings 17
Conclusion 21
Our Methodology 22
Disclaimers 24
Appendix

- Code Flow Diagram 25
- Slither Results Log 32
- Solidity static analysis 37
- Solhint Linter 44

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the Zexus Finance Team to perform the Security audit of the Zexus Finance smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on October 27th, 2023.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

- Zexus is a decentralized Non-fungible Token(NFT) lending protocol.
- Zexus Finance is a contract that can be divided into multiples, each with unique functionalities:
    - **Utils:** Utils contract for managing Zexus.
    - **Vault:** Vault contract for storing NFTs.
    - **ZexusBorrower:** This contract is designed to manage the actions of the borrower.
    - **ZexusCollateral:** This contract is for managing collateral.
    - **ZexusLender:** This contract is designed to manage lender actions.
    - **ZexusSecurity:** This contract is responsible for managing security-related aspects of protocol.
    - **ZexusStorage:** This contract is for the management of stored data.

- There are 7 smart contracts, 3 libraries, 6 interface files which were included in the audit scope. And there were some standard library code such as OpenZepelin, which were excluded. Because those standard library code is considered as time tested and community audited, so we can safely ignore them.

Email: audit@EtherAuthority.io

Audit scope

Name	Code Review and Security Analysis Report for Zexus Finance Smart Contracts
Platform	**Ethereum / Solidity**
File 1	Utils.sol
File 1 MD5 Hash	E1D6D03C3D7E27CF79B62F76AAFB3A51
Updated File 1 MD5 Hash	03636A3F550F88D6D800DFDC4CBCA08E
File 2	Vault.sol
File 2 MD5 Hash	7F8E1EAA684F464E4C8DAC10DBCDF4E3
Updated File 2 MD5 Hash	3197A3367DD259F7C097108B726D2D48
File 3	ZexusBorrower.sol
File 3 MD5 Hash	23C2BDAB7101587E1A2CD89550A58CFA
Updated File 3 MD5 Hash	596B229FBA496806604A2C236968D4E4
File 4	ZexusCollateral.sol
File 4 MD5 Hash	F8715BF1902B9AC7236FFA19AB60D878
Updated File 4 MD5 Hash	EBF59750AD6F998D8957913C17332D8C
File 5	ZexusLender.sol
File 5 MD5 Hash	CF9FA11C84E0D8FBD1E987A2734BC750
File 6	ZexusSecurity.sol
File 6 MD5 Hash	F9F39EF7FA3A9EAC1EEAAC310CC1FBD4
Updated File 6 MD5 Hash	4DC2C2BF95FC63152847C4EEEDF1CC45
File 7	ZexusStorage.sol
File 7 MD5 Hash	E6B3112173460C6CB71F76C53F440291
Updated File 7 MD5 Hash	2DED4A39BCD2CF1EEAC9676E6B2AA661
Audit Date	October 27th, 2023
Revised Audit Date	November 2nd, 2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 Utils.sol • Interest Fee: 2.5% • Principal Fee: 1% **Admin Specifications:** • Adds NFTs to the whitelist. • Removes NFTs from the whitelist. • Whitelist currencies. • Blacklist currencies. • Change fees on the platform. **Other Specifications:** • The contract is utilized for managing Zexus. • Allows whitelisting / blacklisting NFTs and supported currencies for Zexus protocol.	**YES, This is valid. The smart contract owner uses a multisignature wallet to ensure security.**
File 2 Vault.sol **Zexus Role Specifications:** • Withdraw NFTs. **Operator Role Owner Specifications:** • Emergent withdrawal of NFT. **Other Specifications:** • The Vault contract is a secure method for storing non-fungible tokens (NFTs).	**YES, This is valid.**
File 3 ZexusBorrower.sol **Admin Specifications:** • Setup vault address.	**YES, This is valid.**

Other Specifications: • The contract is a document that outlines the process of managing borrower actions.	
File 4 ZexusCollateral.sol **Admin Specifications:** • Setup vault address. **Other Specifications:** • The contract outlines the process for managing collateral.	**YES, This is valid.**
File 5 ZexusLender.sol **Admin Specifications:** • Setup vault address. **Other Specifications:** • The contract is a document that outlines the process for managing lender actions.	**YES, This is valid.**
File 6 ZexusSecurity.sol **Admin Specifications:** • Triggers stopped by the admin role owner. • Returns to normal state by the admin role owner. **Other Specifications:** • The contract outlines the process for managing security-related aspects on a protocol.	**YES, This is valid.**
File 7 ZexusStorage.sol **Admin Specifications:** • Allow updating the platform wallet. **Zexus Role Specifications:** • Allow updating collateral metadata.	**YES, This is valid.**

• Allow updating collateral nfts. • Allow updating collateral terms. • Delete Borrowers Offers. • Update Active Collaterals. • Update currency. • Update Nft Whitelisted. **Other Specifications:** • The contract is for the management of stored data.	

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secured”**. Also, these contracts contain owner control, which does not make them fully decentralized.

We used various tools like Slither, Solhint 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 Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium, 3 low and 0 very low level issues.

We confirm that all issues have been fixed in the revised smart contract code.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Email: audit@EtherAuthority.io

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 7 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in Zexus Finance are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Zexus Finance Protocol.

The Zexus Finance team has not provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are well commented on smart contracts.

Documentation

We were given a Zexus Finance smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are well commented on. And the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

Utils.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	addNFT	external	access by only admin role	Fixed
3	removeNFT	external	access by only admin role	Fixed
4	whitelistCurrencies	external	access by only admin role	Fixed
5	delistCurrencies	external	access by only admin role	Fixed
6	encodeNFTs	external	Passed	No Issue
7	decodeNFTs	external	Passed	No Issue
8	hashLoan	external	Passed	No Issue
9	hashCollateral	external	Passed	No Issue
10	recoverLoan	external	Passed	No Issue
11	recoverCollateral	external	Passed	No Issue
12	changeFee	external	access by only admin role	Fixed
13	calculateFee	external	Passed	No Issue
14	calculateRepayment	external	Passed	No Issue
15	_domainSeparatorV4	internal	Passed	No Issue
16	_buildDomainSeparator	read	Passed	No Issue
17	_hashTypedDataV4	internal	Passed	No Issue
18	eip712Domain	read	Passed	No Issue
19	_EIP712Name	internal	Passed	No Issue
20	_EIP712Version	internal	Passed	No Issue
21	pause	write	access by only admin role	No Issue
22	unpause	write	access by only admin role	No Issue

Vault.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	withdrawNFTs	external	access by only zexus role	Fixed
3	withdrawNFT	external	access by only zexus role	Fixed
4	onERC721Received	external	Passed	No Issue

5	emergencyWithdrawal	external	access by only operator role	Fixed
6	pause	write	access by only admin role	No Issue
7	unpause	write	access by only admin role	No Issue

ZexusBorrower.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	acceptLoan	external	Passed	No Issue
3	repayLoan	external	Passed	No Issue
4	acceptLoanExtensionAsBorrower	external	Passed	No Issue
5	_acceptLoanExtensionSameLender	write	Passed	No Issue
6	_acceptLoanExtensionDifferentLender	write	Passed	No Issue
7	_withdrawNFT	write	Passed	No Issue
8	setUpVault	write	access by only admin role	No Issue
9	pause	write	access by only admin role	No Issue
10	unpause	write	access by only admin role	No Issue

ZexusCollateral.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	pause	write	access by only admin role	No Issue
3	unpause	write	access by only admin role	No Issue
4	addNFTCollateral	external	Passed	Fixed
5	updateCollateral	external	Passed	Fixed
6	inArrays	write	Passed	No Issue
7	cancelCollateral	external	Passed	No Issue
8	_withdrawNFT	write	Passed	No Issue
9	setUpVault	write	access by only admin role	No Issue

ZexusLender.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	pause	write	access by only admin role	No Issue
3	unpause	write	access by only admin role	No Issue
4	acceptBorrowerOffer	external	Passed	No Issue
5	loanDefaulted	external	Passed	No Issue
6	acceptLoanExtensionAsLender	external	Passed	No Issue
7	_withdrawNFT	write	Passed	No Issue
8	setUpVault	external	access by only admin role	No Issue

ZexusSecurity.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	pause	write	access by only admin role	No Issue
3	unpause	write	access by only admin role	No Issue
4	onlyRole	read	Passed	No Issue
5	supportsInterface	read	Passed	No Issue
6	hasRole	read	Passed	No Issue
7	_checkRole	internal	Passed	No Issue
8	_checkRole	internal	Passed	No Issue
9	getRoleAdmin	read	Passed	No Issue
10	grantRole	write	access by only admin role	No Issue
11	revokeRole	write	access by only admin role	No Issue
12	renounceRole	write	Passed	No Issue
13	_setRoleAdmin	internal	Passed	No Issue
14	_grantRole	internal	Passed	No Issue
15	_revokeRole	internal	Passed	No Issue
16	paused	read	Passed	No Issue
17	whenNotPaused	modifier	Passed	No Issue
18	whenPaused	modifier	Passed	No Issue
19	_pause	internal	Passed	No Issue
20	_unpause	internal	Passed	No Issue
21	nonReentrant	modifier	Passed	No Issue

ZexusStorage.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	pause	write	access by only admin role	No Issue
3	unpause	write	access by only admin role	No Issue
4	updateZexusWallet	external	access by only admin role	No Issue
5	updateBorrowersOffers	external	access by only zexus role	No Issue
6	updateBorrowersOffersNfts	external	access by only zexus role	No Issue
7	updateBorrowersOffersCollateral	external	access by only zexus role	No Issue
8	deleteBorrowersOffers	external	access by only zexus role	No Issue
9	getBorrowersOffers	external	Passed	No Issue
10	updateActiveCollaterals	external	access by only zexus role	No Issue
11	getActiveCollaterals	external	Passed	No Issue
12	deleteActiveCollaterals	external	access by only zexus role	No Issue
13	updateCurrency	external	access by only zexus role	No Issue
14	getSupportedCurrency	external	Passed	No Issue
15	updateNftWhitelisted	external	access by only zexus role	No Issue
16	getNftWhitelisted	external	Passed	No Issue
17	nonces	read	Passed	No Issue
18	useNonce	write	access by only zexus role	No Issue
19	collateralId	external	access by only zexus role	No Issue
20	loanId	external	access by only zexus role	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No critical severity vulnerabilities were found in the contract code.

High Severity

No high severity vulnerabilities were found in the contract code.

Medium

No medium severity vulnerabilities were found in the contract code.

Low

(1) Infinite Loop:

Vault.sol

In emergencyWithdrawal, withdrawNFTs functions, for loop _nfts and _ids array length must have some limit set to save the gas.

Utils.sol

In addNFT, removeNFT, whitelistCurrencies, delistCurrencies, functions, for loop _nfts and _currencies array length must have some limit set to save the gas.

ZexusCollateral.sol

In addNFTCollateral, updateCollateral functions for loop array length should have some limit to save the gas.

Resolution: The upper bound should have a certain limit for loops.

Status: **Fixed**

(2) Function input parameters lack of check:

Variable validation is not performed in the below functions:

Vault.sol

- withdrawNFT = _receiver

- withdrawNFTs = _receiver
- emergencyWithdrawal = _to

Resolution: We advise to put validation: int type variables should not be empty and greater than 0, and address type variables should not be address(0).
Status: **Fixed**

(3) The fee limit is not set: **Utils.sol**

```
/// @notice change fees on platform
/// @param _type enum from FeeType
/// @param _fee value of new fee
function changeFee(    ⛽ undefined gas
    FeeType _type,
    uint256 _fee
) external onlyRole(DEFAULT_ADMIN_ROLE) whenNotPaused {
    if (_type == FeeType.interestFee) {
        interestFee = _fee;
        emit FeeChanged("interestFee", _fee);
    } else {
        principalFee = _fee;
        emit FeeChanged("principalFee", _fee);
    }
}
```

In the changeFee function, the fee limit is not set. Admin can set it to any number.

Resolution: We suggest adding explicit limits while setting the value of the fee variable.
Status: **Fixed**

Very Low / Informational / Best practices:

No very low severity vulnerabilities were found in the contract code.

Centralization

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. Following are Admin functions:

Utils.sol

- addNFT: Adds NFTs to the whitelist by the admin role owner.
- removeNFT: Removes NFTs from the whitelist by the admin role owner.
- whitelistCurrencies: Whitelist currencies by the admin role owner.
- delistCurrencies:Blacklist currencies by the admin role owner.
- changeFee: Change fees on the platform by the admin role owner.

Vault.sol

- withdrawNFTs: Withdraw NFTs by the zexus role owner.
- withdrawNFT: Withdraw NFT by the zexus role owner.
- emergencyWithdrawal: Emergent withdrawal of NFT by the operator role owner.

ZexusBorrower.sol

- acceptLoan: Accept the loan by the borrower to accept the backend loan.
- repayLoan: Repay the loan by the borrower to repayLoan.
- acceptLoanExtensionAs Borrower: The borrower can accept a proposed loan extension.
- setUpVault: Setup vault address by the admin role owner.

ZexusCollateral.sol

- updateCollateral: Updating collateral by the owner of collateral.
- cancelCollateral: Allows the cancellation of the collateral by the owner of the collateral.
- setUpVault: Setup vault address by the admin role owner.

ZexusLender.sol

- acceptLoanExtensionAsLender: Accept borrower offer as lender by the original loaner.

- setUpVault: Setup vault address by the admin role owner.

ZexusSecurity.sol

- pause: Triggers stopped state by the admin role owner.
- unpause: Returns to normal state by the admin role owner.

ZexusStorage.sol

- updateZexusWallet: Allow updating the platform wallet by the admin role owner.
- updateBorrowersOffers: Allow updating collateral metadata by the zexus role owner.
- updateBorrowersOffersNfts: Allow updating collateral nfts by the zexus role owner.
- updateBorrowersOffersCollateral: Allow updating collateral terms by the zexus role owner.
- deleteBorrowersOffers: Delete Borrowers Offers by the zexus role owner.
- updateActiveCollaterals: Update Active Collaterals by the zexus role owner.
- deleteActiveCollaterals: Delete Active Collaterals by the zexus role owner.
- updateCurrency: Update currency by the zexus role owner.
- updateNftWhitelisted: Update Nft Whitelisted by the zexus role owner.
- useNonce: Use Nonce by the zexus role owner.
- collateralId: Collateral ID by the zexus role owner.
- loanId: Loan ID by the zexus role owner.

AccessControl.sol

- grantRole: Grants `role` to `account` can be set by the owner.
- revokeRole: Revokes `role` from `account` by the owner.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We had observed 3 low severity issues in the smart contracts. We confirm that all issues have been fixed in the revised smart contract code. So, **it's good to go for the production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

The security state of the reviewed contract, based on standard audit procedure scope, is **"Secured".**

Email: audit@EtherAuthority.io

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

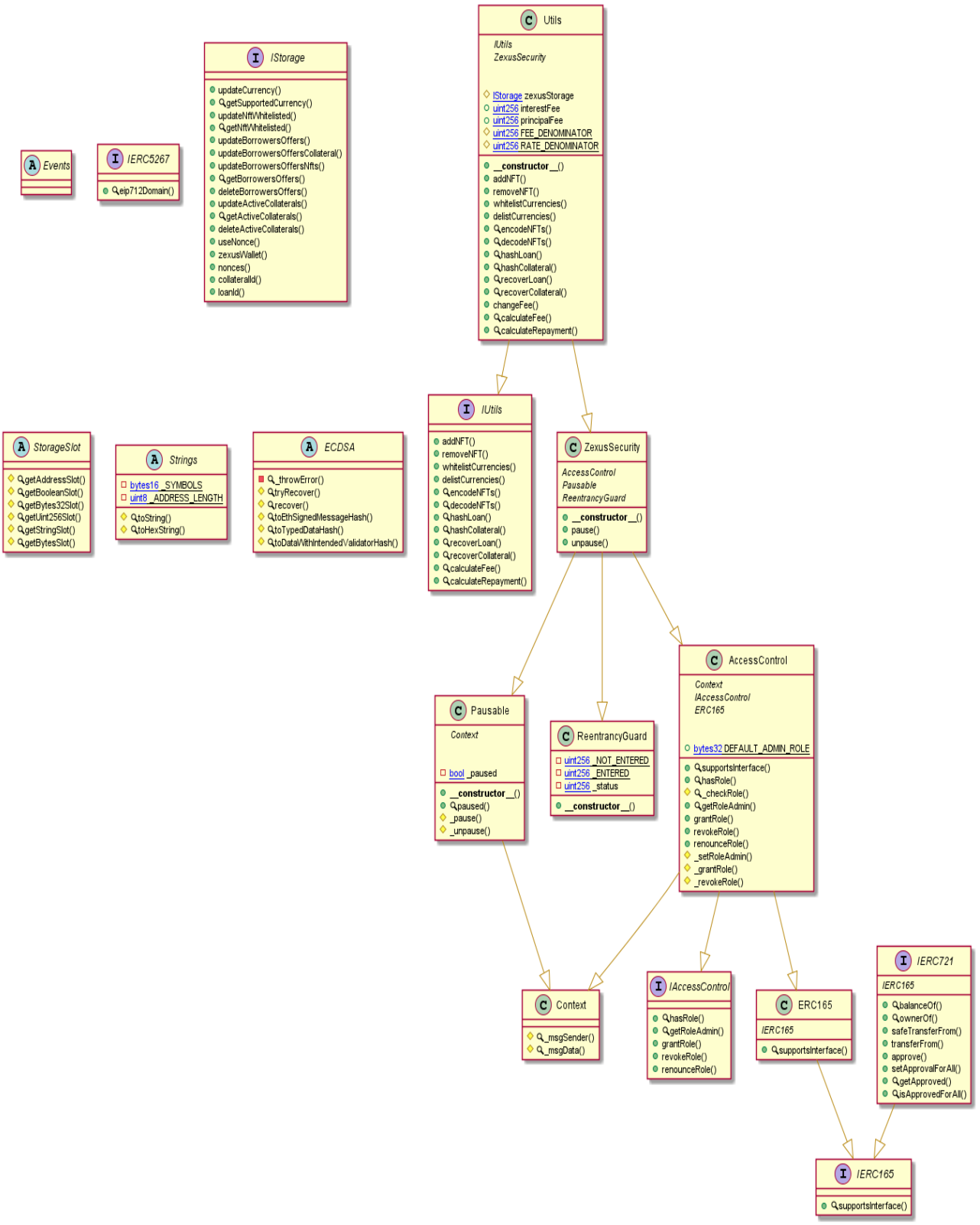
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

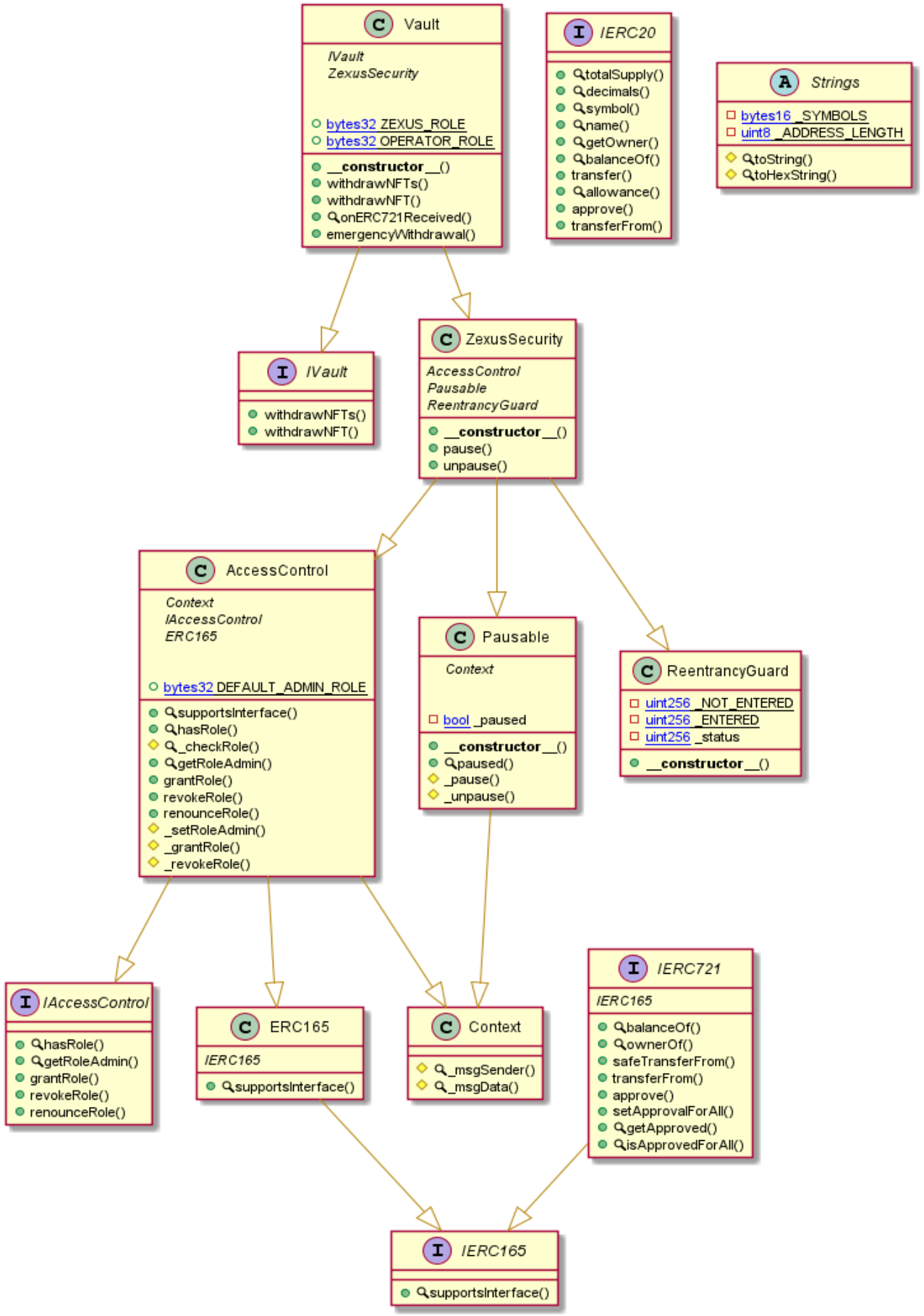
Appendix

Code Flow Diagram - Zexus Finance

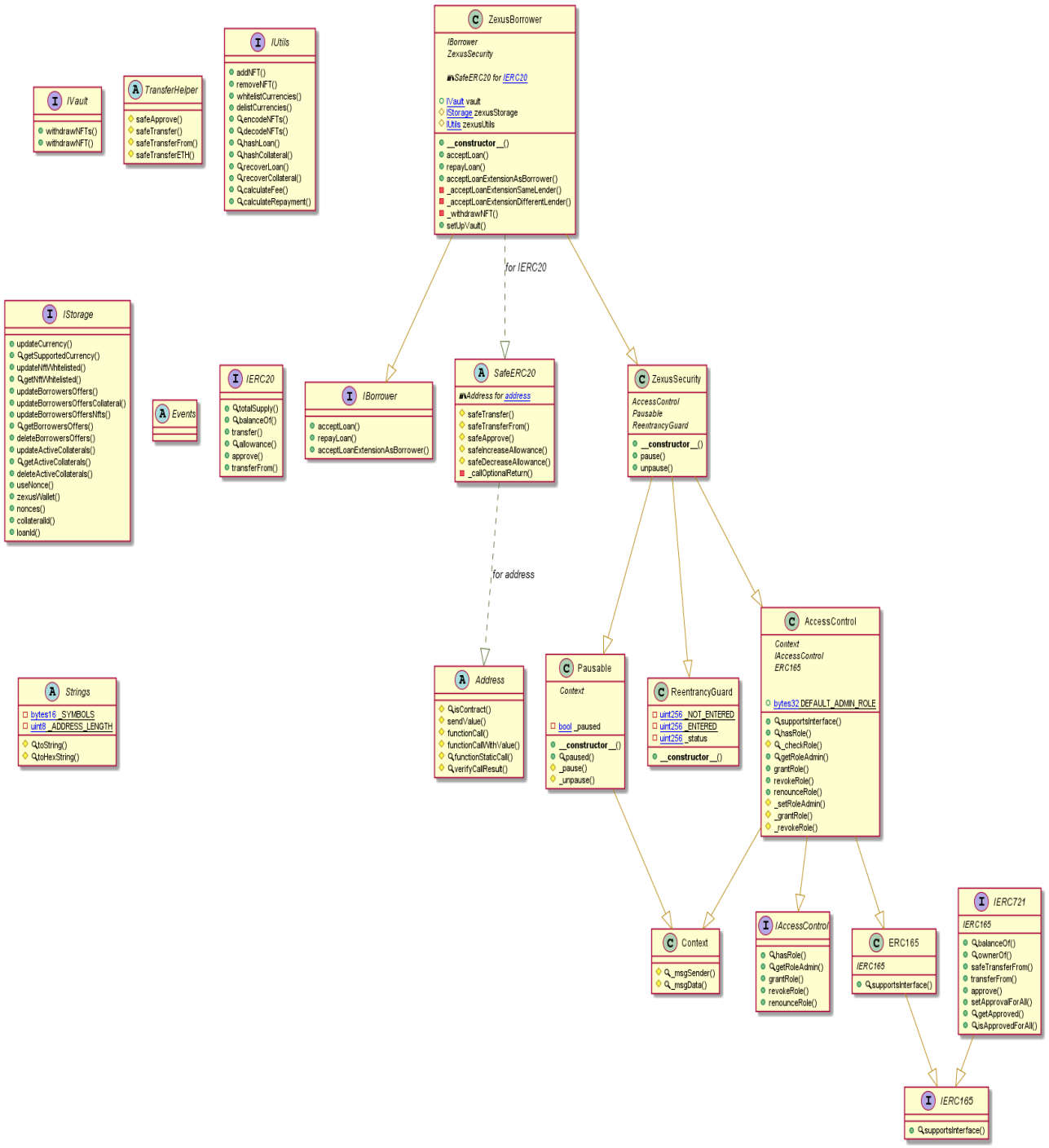
Utils Diagram



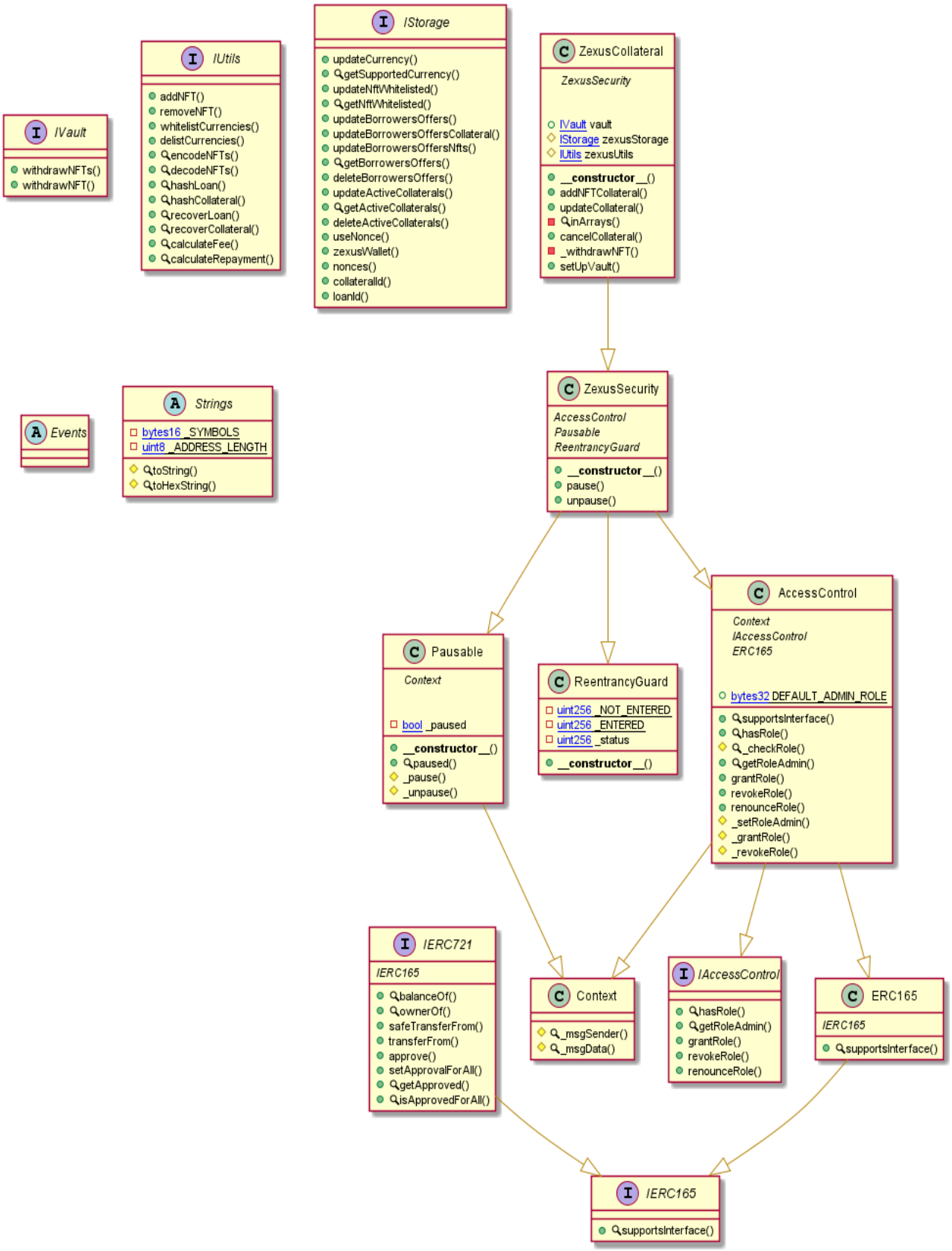
Vault Diagram



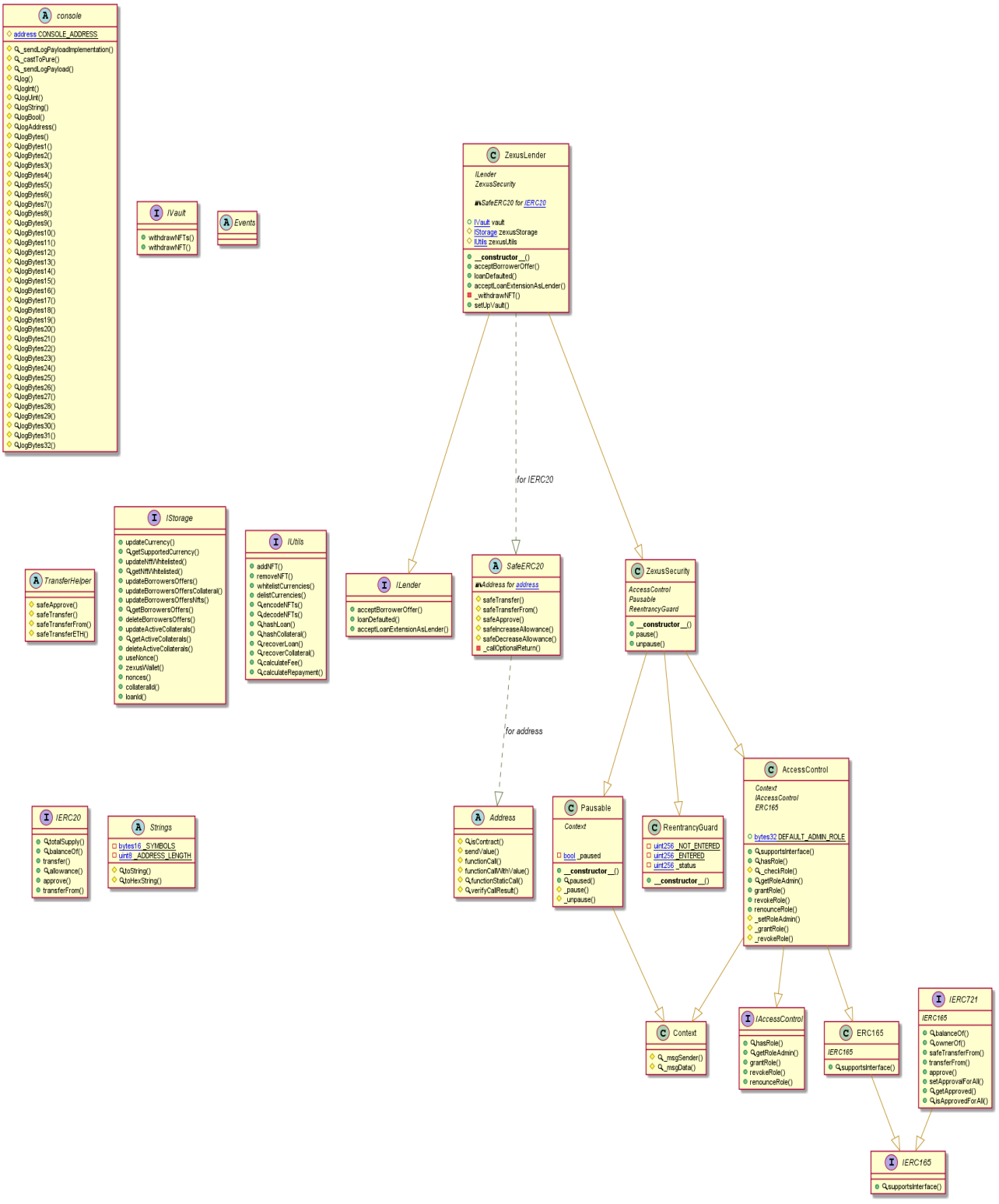
ZexusBorrower Diagram



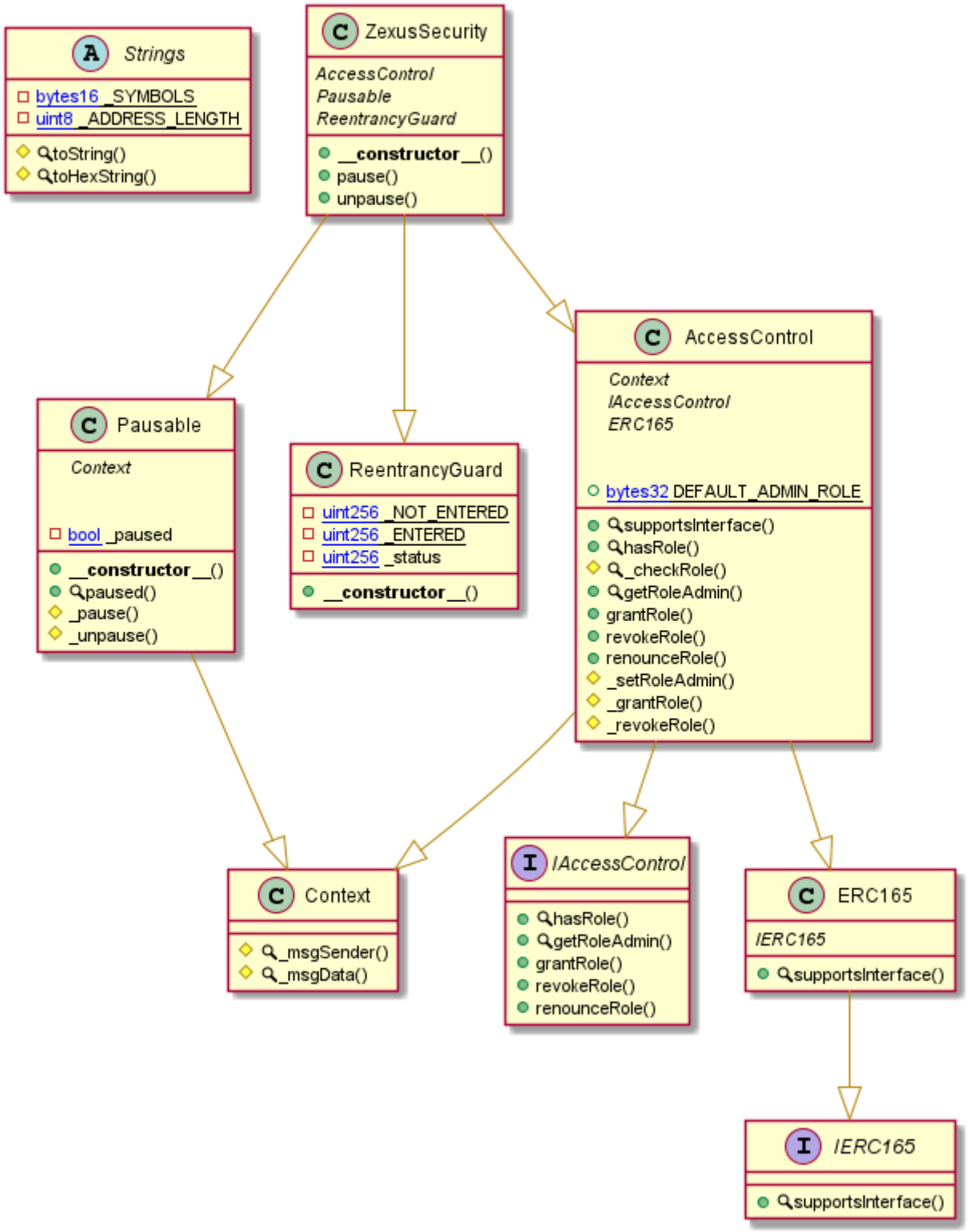
ZexusCollateral Diagram



ZexusLender Diagram

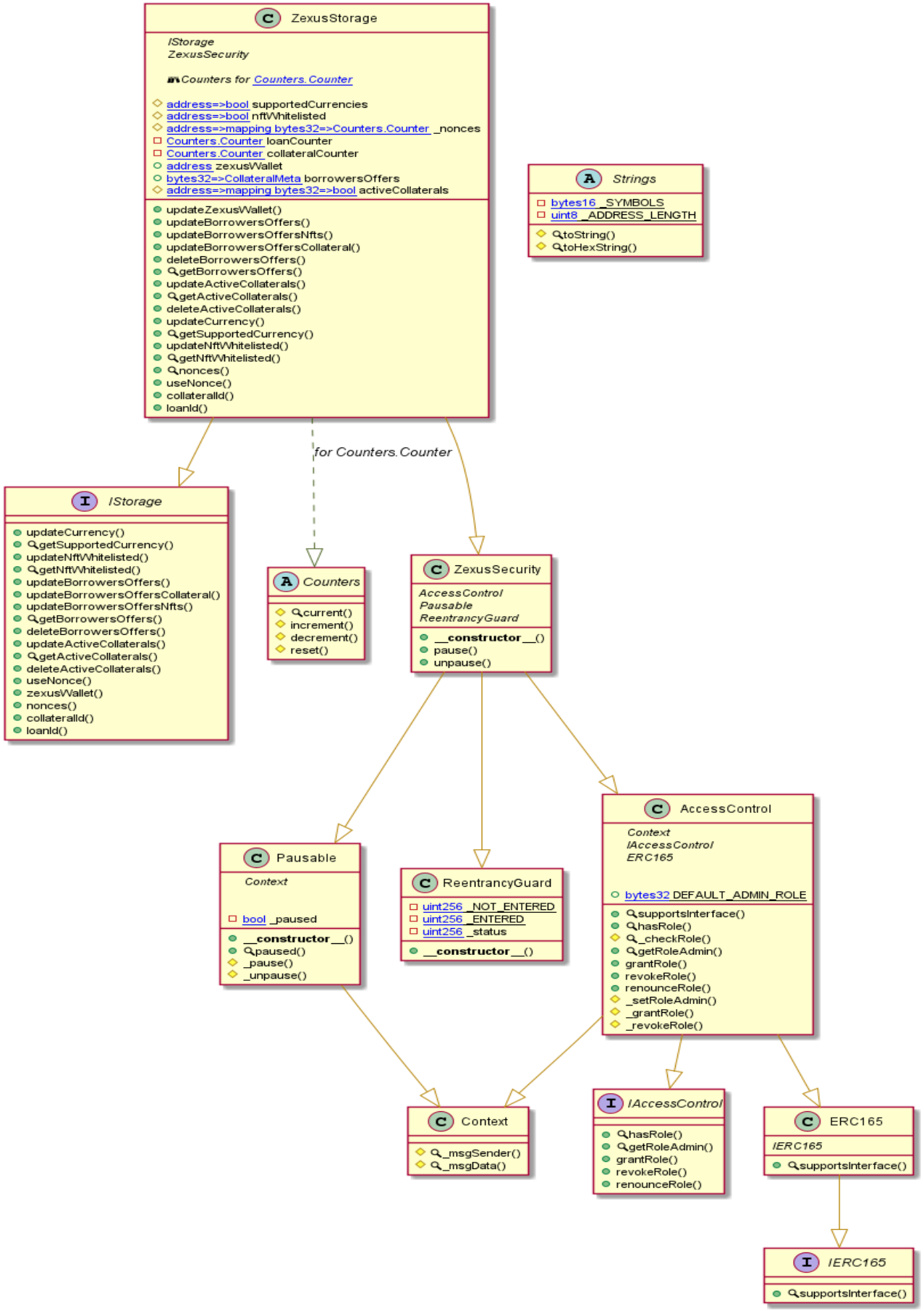


ZexusSecurity Diagram



Email: audit@EtherAuthority.io

ZexusStorage Diagram



Slither Results Log

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

Slither log >> Utils.sol

```
Utils.addNFT(address[]) (Utils.sol#973-980) has external calls inside a loop: zexusStorage.updateNftWhitelisted(_nfts[i],true) (Utils.sol#977)
Utils.removeNFT(address[]) (Utils.sol#984-991) has external calls inside a loop: zexusStorage.updateNftWhitelisted(_nfts[i],false) (Utils.sol#988)
Utils.whitelistCurrencies(address[]) (Utils.sol#995-1002) has external calls inside a loop: zexusStorage.updateCurrency(_currencies[i],true) (Utils.sol#999)
Utils.delistCurrencies(address[]) (Utils.sol#1006-1013) has external calls inside a loop: zexusStorage.updateCurrency(_currencies[i],false) (Utils.sol#1010)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in Utils.addNFT(address[]) (Utils.sol#973-980):
        External calls:
        - zexusStorage.updateNftWhitelisted(_nfts[i],true) (Utils.sol#977)
        Event emitted after the call(s):
        - Events.WhitelistNFT(_nfts) (Utils.sol#979)
Reentrancy in Utils.delistCurrencies(address[]) (Utils.sol#1006-1013):
        External calls:
        - zexusStorage.updateCurrency(_currencies[i],false) (Utils.sol#1010)
        Event emitted after the call(s):
        - Events.BlacklistCurrency(_currencies) (Utils.sol#1012)
Reentrancy in Utils.removeNFT(address[]) (Utils.sol#984-991):
        External calls:
        - zexusStorage.updateNftWhitelisted(_nfts[i],false) (Utils.sol#988)
        Event emitted after the call(s):
        - Events.BlacklistNFT(_nfts) (Utils.sol#990)
Reentrancy in Utils.whitelistCurrencies(address[]) (Utils.sol#995-1002):
        External calls:
        - zexusStorage.updateCurrency(_currencies[i],true) (Utils.sol#999)
        Event emitted after the call(s):
        - Events.WhitelistCurrency(_currencies) (Utils.sol#1001)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
StorageSlot.getAddressSlot(bytes32) (Utils.sol#225-230) uses assembly
        - INLINE ASM (Utils.sol#227-229)
StorageSlot.getBooleanSlot(bytes32) (Utils.sol#235-240) uses assembly
        - INLINE ASM (Utils.sol#237-239)
StorageSlot.getBytes32Slot(bytes32) (Utils.sol#245-250) uses assembly
        - INLINE ASM (Utils.sol#247-249)
StorageSlot.getUint256Slot(bytes32) (Utils.sol#255-260) uses assembly
        - INLINE ASM (Utils.sol#257-259)
StorageSlot.getStringSlot(bytes32) (Utils.sol#265-270) uses assembly
        - INLINE ASM (Utils.sol#267-269)
StorageSlot.getStringSlot(string) (Utils.sol#275-280) uses assembly
        - INLINE ASM (Utils.sol#277-279)
StorageSlot.getBytesSlot(bytes32) (Utils.sol#285-290) uses assembly
        - INLINE ASM (Utils.sol#287-289)
StorageSlot.getBytesSlot(bytes) (Utils.sol#295-300) uses assembly
        - INLINE ASM (Utils.sol#297-299)
ShortStrings.toString(ShortString) (Utils.sol#324-334) uses assembly
        - INLINE ASM (Utils.sol#329-332)
Strings.toString(uint256) (Utils.sol#389-405) uses assembly
        - INLINE ASM (Utils.sol#397-399)
ECDSA.tryRecover(bytes32,bytes) (Utils.sol#448-462) uses assembly
        - INLINE ASM (Utils.sol#453-457)
ECDSA.toEthSignedMessageHash(bytes32) (Utils.sol#501-507) uses assembly
        - INLINE ASM (Utils.sol#502-506)
ECDSA.toTypedDataHash(bytes32,bytes32) (Utils.sol#513-521) uses assembly
        - INLINE ASM (Utils.sol#514-520)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version^0.8.17 (Utils.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Function EIP712._EIP712Name() (Utils.sol#641-643) is not in mixedCase
Function EIP712._EIP712Version() (Utils.sol#652-654) is not in mixedCase
Parameter Utils.addNFT(address[])._nfts (Utils.sol#974) is not in mixedCase
Parameter Utils.removeNFT(address[])._nfts (Utils.sol#985) is not in mixedCase
Parameter Utils.whitelistCurrencies(address[])._currencies (Utils.sol#996) is not in mixedCase
Parameter Utils.delistCurrencies(address[])._currencies (Utils.sol#1007) is not in mixedCase
Parameter Utils.encodeNFTs(address[],uint256[])._assets (Utils.sol#1019) is not in mixedCase
Parameter Utils.encodeNFTs(address[],uint256[])._id (Utils.sol#1020) is not in mixedCase
Parameter Utils.decodeNFTs(bytes)._nfts (Utils.sol#1028) is not in mixedCase
Parameter Utils.hashLoan(Loan)._loan (Utils.sol#1035) is not in mixedCase
Parameter Utils.hashCollateral(CollateralOffer)._collateral (Utils.sol#1059) is not in mixedCase
Parameter Utils.recoverLoan(Loan,bytes)._loan (Utils.sol#1085) is not in mixedCase
Parameter Utils.recoverCollateral(CollateralOffer,bytes)._collateral (Utils.sol#1095) is not in mixedCase
Parameter Utils.changeFee(FeeType,uint256)._type (Utils.sol#1105) is not in mixedCase
Parameter Utils.changeFee(FeeType,uint256)._fee (Utils.sol#1106) is not in mixedCase
Parameter Utils.calculateFee(FeeType,uint256,uint256,uint256)._type (Utils.sol#1123) is not in mixedCase
Parameter Utils.calculateFee(FeeType,uint256,uint256,uint256)._interestRate (Utils.sol#1124) is not in mixedCase
Parameter Utils.calculateFee(FeeType,uint256,uint256,uint256)._duration (Utils.sol#1125) is not in mixedCase
Parameter Utils.calculateFee(FeeType,uint256,uint256,uint256)._principal (Utils.sol#1126) is not in mixedCase
Parameter Utils.calculateRepayment(uint256,uint256,uint256)._interestRate (Utils.sol#1144) is not in mixedCase
Parameter Utils.calculateRepayment(uint256,uint256,uint256)._duration (Utils.sol#1145) is not in mixedCase
Parameter Utils.calculateRepayment(uint256,uint256,uint256)._principal (Utils.sol#1146) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

ShortStrings.slitherConstructorConstantVariables() (Utils.sol#304-383) uses literals with too many digits:
        - _FALLBACK_SENTINEL = 0x00000000000000000000000000000000000000000000000000000000000000FF (Utils.sol#306)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Utils.zexusStorage (Utils.sol#954) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Utils.sol analyzed (19 contracts with 84 detectors), 69 result(s) found
```

Slither log >> Vault.sol

```
Vault.withdrawNFTs(address[],uint256[],address) (Vault.sol#409-417) has external calls inside a loop: IERC721(_nfts[i]).transfe
rFrom(address(this),_receiver,_ids[i]) (Vault.sol#415)
Vault.emergencyWithdrawal(address,address,uint256[]) (Vault.sol#439-447) has external calls inside a loop: IERC721(_token).tran
sferFrom(address(this),_to,_ids[i]) (Vault.sol#445)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Strings.toString(uint256) (Vault.sol#97-115) uses assembly
        - INLINE ASM (Vault.sol#102-104)
        - INLINE ASM (Vault.sol#107-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

AccessControl._setRoleAdmin(bytes32,bytes32) (Vault.sol#226-230) is never used and should be removed
Context._msgData() (Vault.sol#163-165) is never used and should be removed
Strings.toHexString(address) (Vault.sol#135-137) is never used and should be removed
Strings.toHexString(uint256) (Vault.sol#117-121) is never used and should be removed
Strings.toHexString(uint256,uint256) (Vault.sol#123-133) is never used and should be removed
Strings.toString(uint256) (Vault.sol#97-115) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (Vault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.1
6
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Parameter Vault.withdrawNFTs(address[],uint256[],address)._nfts (Vault.sol#410) is not in mixedCase
Parameter Vault.withdrawNFTs(address[],uint256[],address)._ids (Vault.sol#411) is not in mixedCase
Parameter Vault.withdrawNFTs(address[],uint256[],address)._receiver (Vault.sol#412) is not in mixedCase
Parameter Vault.withdrawNFT(address,uint256,address)._nft (Vault.sol#420) is not in mixedCase
Parameter Vault.withdrawNFT(address,uint256,address)._id (Vault.sol#421) is not in mixedCase
Parameter Vault.withdrawNFT(address,uint256,address)._receiver (Vault.sol#422) is not in mixedCase
Parameter Vault.emergencyWithdrawal(address,address,uint256[])._to (Vault.sol#440) is not in mixedCase
Parameter Vault.emergencyWithdrawal(address,address,uint256[])._token (Vault.sol#441) is not in mixedCase
Parameter Vault.emergencyWithdrawal(address,address,uint256[])._ids (Vault.sol#442) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
Vault.sol analyzed (13 contracts with 84 detectors), 20 result(s) found
```

Slither log >> ZexusBorrower.sol

```
Reentrancy in ZexusBorrower.acceptLoan(Loan,bytes) (ZexusBorrower.sol#804-895):
        External calls:
        - require(bool,string)(_loan.nonce == zexusStorage.nonces(_loan.creator,_loan.collateralId),Wrong nonce) (ZexusBorrower
.sol#819-823)
        - zexusStorage.useNonce(_loan.creator,_loan.collateralId) (ZexusBorrower.sol#825)
        - TransferHelper.safeTransferFrom(_loan.currency,_loan.creator,zexusStorage.zexusWallet(),principalFee) (ZexusBorrower.
sol#845-850)
        - TransferHelper.safeTransferFrom(_loan.currency,_loan.creator,msg.sender,_loan.value - principalFee) (ZexusBorrower.so
l#852-857)
        - _idLoan = zexusStorage.loanId() (ZexusBorrower.sol#859)
        - zexusStorage.updateBorrowersOffers(_loan.collateralId,CollateralMeta(nfts,msg.sender,collateral,block.timestamp)) (Ze
xusBorrower.sol#874-882)
        - zexusStorage.updateBorrowersOffersCollateral(_loan.collateralId,collateral) (ZexusBorrower.sol#883-886)
        - zexusStorage.updateActiveCollaterals(msg.sender,_loan.collateralId,true) (ZexusBorrower.sol#888-892)
        Event emitted after the call(s):
        - Events.CollateralAccepted(_loan.collateralId,_idLoan) (ZexusBorrower.sol#894)
Reentrancy in ZexusBorrower.acceptLoanExtensionAsBorrower(Loan,bytes,bytes32) (ZexusBorrower.sol#965-1032):
        External calls:
        - require(bool,string)(_loan.nonce == zexusStorage.nonces(_loan.creator,_loan.collateralId),Wrong nonce) (ZexusBorrower
.sol#991-995)
        - zexusStorage.useNonce(_loan.creator,_loan.collateralId) (ZexusBorrower.sol#996)
        - _acceptLoanExtensionSameLender(_loan,collateral,borrower) (ZexusBorrower.sol#1011)
                - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (ZexusBorrower.sol#56-58)
                - TransferHelper.safeTransferFrom(loan.currency,borrower,loan.creator,repayment - interestFee - loan.value) (Ze
xusBorrower.sol#1039-1044)
```

```
                - TransferHelper.safeTransferFrom(loan.currency,loan.creator,collateral.loaner,repayment - interestFee) (ZexusB
orrower.sol#1097-1102)
                - TransferHelper.safeTransferFrom(loan.currency,loan.creator,borrower,loan.value - repayment - principalFee) (Z
exusBorrower.sol#1103-1108)
        - _idLoan = zexusStorage.loanId() (ZexusBorrower.sol#1017)
        - zexusStorage.updateBorrowersOffersCollateral(_idCollateral,collateral) (ZexusBorrower.sol#1028)
        Event emitted after the call(s):
        - Events.LoanExtended(_idCollateral,_idLoan) (ZexusBorrower.sol#1031)
        - Events.LoanRepaid(_idCollateral,oldLoan) (ZexusBorrower.sol#1030)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ZexusBorrower.acceptLoan(Loan,bytes) (ZexusBorrower.sol#804-895) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_loan.deadline > block.timestamp,Deadline) (ZexusBorrower.sol#818)
ZexusBorrower.repayLoan(bytes32) (ZexusBorrower.sol#899-958) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= collateralMeta.timestamp + collateralMeta.collateral.duration,Repayment overd
ue) (ZexusBorrower.sol#911-915)
ZexusBorrower.acceptLoanExtensionAsBorrower(Loan,bytes,bytes32) (ZexusBorrower.sol#965-1032) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_loan.deadline > block.timestamp,Deadline) (ZexusBorrower.sol#990)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Pragma version^0.8.17 (ZexusBorrower.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in TransferHelper.safeApprove(address,address,uint256) (ZexusBorrower.sol#24-34):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (ZexusBorrower.sol#27-29)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (ZexusBorrower.sol#36-46):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (ZexusBorrower.sol#39-41)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (ZexusBorrower.sol#48-63):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (ZexusBorrower.sol#56-58)
Low level call in TransferHelper.safeTransferETH(address,uint256) (ZexusBorrower.sol#65-68):
        - (success) = to.call{value: value}(new bytes(0)) (ZexusBorrower.sol#66)
Low level call in Address.sendValue(address,uint256) (ZexusBorrower.sol#283-288):
        - (success) = recipient.call{value: amount}() (ZexusBorrower.sol#286)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (ZexusBorrower.sol#310-321):
        - (success,returndata) = target.call{value: value}(data) (ZexusBorrower.sol#319)
Low level call in Address.functionStaticCall(address,bytes,string) (ZexusBorrower.sol#327-336):
        - (success,returndata) = target.staticcall(data) (ZexusBorrower.sol#334)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ZexusBorrower.acceptLoan(Loan,bytes)._loan (ZexusBorrower.sol#805) is not in mixedCase
Parameter ZexusBorrower.acceptLoan(Loan,bytes)._signature (ZexusBorrower.sol#806) is not in mixedCase
Parameter ZexusBorrower.repayLoan(bytes32)._collateralId (ZexusBorrower.sol#900) is not in mixedCase
Parameter ZexusBorrower.acceptLoanExtensionAsBorrower(Loan,bytes,bytes32)._loan (ZexusBorrower.sol#966) is not in mixedCase
Parameter ZexusBorrower.acceptLoanExtensionAsBorrower(Loan,bytes,bytes32)._signature (ZexusBorrower.sol#967) is not in mixedCas
e
Parameter ZexusBorrower.acceptLoanExtensionAsBorrower(Loan,bytes,bytes32)._idCollateral (ZexusBorrower.sol#968) is not in mixed
Case
Parameter ZexusBorrower.setUpVault(address)._vault (ZexusBorrower.sol#1123) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
ZexusBorrower.zexusStorage (ZexusBorrower.sol#792) should be immutable
ZexusBorrower.zexusUtils (ZexusBorrower.sol#793) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
ZexusBorrower.sol analyzed (20 contracts with 84 detectors), 63 result(s) found
```

Slither log >> ZexusCollateral.sol

```
ZexusCollateral.addNFTCollateral(address[],uint256[],Collateral) (ZexusCollateral.sol#549-594) has external calls inside a loop
: ! zexusStorage.getNftWhitelisted(_assets[i]) (ZexusCollateral.sol#568)
ZexusCollateral.addNFTCollateral(address[],uint256[],Collateral) (ZexusCollateral.sol#549-594) has external calls inside a loop
: IERC721(_assets[i]).safeTransferFrom(msg.sender,address(vault),_ids[i]) (ZexusCollateral.sol#572-576)
ZexusCollateral.updateCollateral(Collateral,bytes32,address[],uint256[],uint256[]) (ZexusCollateral.sol#603-695) has external c
alls inside a loop: require(bool,string)(zexusStorage.getNftWhitelisted(_assets[i]),NFT not whitelisted) (ZexusCollateral.sol#6
43-646)
ZexusCollateral.updateCollateral(Collateral,bytes32,address[],uint256[],uint256[]) (ZexusCollateral.sol#603-695) has external c
alls inside a loop: vault.withdrawNFT(_assets[i],_ids[i],msg.sender) (ZexusCollateral.sol#659)
ZexusCollateral.updateCollateral(Collateral,bytes32,address[],uint256[],uint256[]) (ZexusCollateral.sol#603-695) has external c
alls inside a loop: IERC721(_assets[i]).safeTransferFrom(msg.sender,address(vault),_ids[i]) (ZexusCollateral.sol#663-667)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in ZexusCollateral.addNFTCollateral(address[],uint256[],Collateral) (ZexusCollateral.sol#549-594):
        External calls:
        - IERC721(_assets[i]).safeTransferFrom(msg.sender,address(vault),_ids[i]) (ZexusCollateral.sol#572-576)
        - _idCollateral = zexusStorage.collateralId() (ZexusCollateral.sol#581)
        - zexusStorage.updateBorrowersOffers(_idCollateral,CollateralMeta(encodedNFTs,msg.sender,_collateral,0)) (ZexusCollater
al.sol#583-591)
        Event emitted after the call(s):
        - Events.CollateralAdded(msg.sender,_idCollateral) (ZexusCollateral.sol#593)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Strings.toString(uint256) (ZexusCollateral.sol#238-256) uses assembly
        - INLINE ASM (ZexusCollateral.sol#243-245)
        - INLINE ASM (ZexusCollateral.sol#248-250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
AccessControl._setRoleAdmin(bytes32,bytes32) (ZexusCollateral.sol#487-491) is never used and should be removed
Context._msgData() (ZexusCollateral.sol#305-307) is never used and should be removed
Strings.toHexString(address) (ZexusCollateral.sol#276-278) is never used and should be removed
Strings.toHexString(uint256) (ZexusCollateral.sol#258-262) is never used and should be removed
Strings.toHexString(uint256,uint256) (ZexusCollateral.sol#264-274) is never used and should be removed
Strings.toString(uint256) (ZexusCollateral.sol#238-256) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (ZexusCollateral.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0
.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Parameter ZexusCollateral.addNFTCollateral(address[],uint256[],Collateral)._assets (ZexusCollateral.sol#550) is not in mixedCas
e
Parameter ZexusCollateral.addNFTCollateral(address[],uint256[],Collateral)._ids (ZexusCollateral.sol#551) is not in mixedCase
Parameter ZexusCollateral.addNFTCollateral(address[],uint256[],Collateral)._collateral (ZexusCollateral.sol#552) is not in mixe
dCase
Parameter ZexusCollateral.updateCollateral(Collateral,bytes32,address[],uint256[],uint256[])._collateral (ZexusCollateral.sol#6
04) is not in mixedCase
Parameter ZexusCollateral.updateCollateral(Collateral,bytes32,address[],uint256[],uint256[])._idCollateral (ZexusCollateral.sol
#605) is not in mixedCase
Parameter ZexusCollateral.updateCollateral(Collateral,bytes32,address[],uint256[],uint256[])._assets (ZexusCollateral.sol#606)
is not in mixedCase
Parameter ZexusCollateral.updateCollateral(Collateral,bytes32,address[],uint256[],uint256[])._ids (ZexusCollateral.sol#607) is
not in mixedCase
Parameter ZexusCollateral.updateCollateral(Collateral,bytes32,address[],uint256[],uint256[])._actions (ZexusCollateral.sol#608)
 is not in mixedCase
Parameter ZexusCollateral.inArrays(address,uint256,address[],uint256[])._asset (ZexusCollateral.sol#699) is not in mixedCase
Parameter ZexusCollateral.inArrays(address,uint256,address[],uint256[])._id (ZexusCollateral.sol#700) is not in mixedCase
Parameter ZexusCollateral.inArrays(address,uint256,address[],uint256[])._assets (ZexusCollateral.sol#701) is not in mixedCase
Parameter ZexusCollateral.inArrays(address,uint256,address[],uint256[])._ids (ZexusCollateral.sol#702) is not in mixedCase
Parameter ZexusCollateral.cancelCollateral(bytes32)._collateralId (ZexusCollateral.sol#717) is not in mixedCase
Parameter ZexusCollateral.setUpVault(address)._vault (ZexusCollateral.sol#747) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

ZexusCollateral.zexusStorage (ZexusCollateral.sol#534) should be immutable
ZexusCollateral.zexusUtils (ZexusCollateral.sol#535) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
ZexusCollateral.sol analyzed (15 contracts with 84 detectors), 31 result(s) found
```

Slither log >> ZexusLender.sol

```
Reentrancy in ZexusLender.acceptBorrowerOffer(bytes32) (ZexusLender.sol#2350-2407):
        External calls:
        - TransferHelper.safeTransferFrom(collateral.currency,msg.sender,zexusStorage.zexusWallet(),principalFee) (ZexusLender.
sol#2373-2378)
        - TransferHelper.safeTransferFrom(collateral.currency,msg.sender,creator,collateral.value - principalFee) (ZexusLender.
sol#2380-2385)
        - _idLoan = zexusStorage.loanId() (ZexusLender.sol#2387)
        - zexusStorage.updateBorrowersOffers(_idCollateral,CollateralMeta(nfts,creator,collateral,block.timestamp)) (ZexusLende
r.sol#2394-2402)
        - zexusStorage.updateActiveCollaterals(creator,_idCollateral,true) (ZexusLender.sol#2404)
        Event emitted after the call(s):
        - Events.CollateralAccepted(_idCollateral,_idLoan) (ZexusLender.sol#2406)
Reentrancy in ZexusLender.acceptLoanExtensionAsLender(CollateralOffer,bytes) (ZexusLender.sol#2449-2547):
        External calls:
        - require(bool,string)(collateral.nonce == zexusStorage.nonces(collateralCreator,_idCollateral),Wrong nonce) (ZexusLend
er.sol#2460-2464)
        - zexusStorage.useNonce(collateralCreator,_idCollateral) (ZexusLender.sol#2465)
        - _idLoan = zexusStorage.loanId() (ZexusLender.sol#2501)
        - TransferHelper.safeTransferFrom(collateral.currency,collateralMeta.creator,zexusStorage.zexusWallet(),interestFee + p
rincipalFee) (ZexusLender.sol#2516-2521)
        - TransferHelper.safeTransferFrom(collateral.currency,collateralMeta.creator,collateral.loaner,repayment - interestFee
- collateral.value) (ZexusLender.sol#2523-2528)
        - TransferHelper.safeTransferFrom(collateral.currency,collateral.loaner,collateralMeta.creator,collateral.value - repay
ment - principalFee) (ZexusLender.sol#2530-2535)
        - zexusStorage.updateBorrowersOffersCollateral(_idCollateral,_collateral) (ZexusLender.sol#2541-2544)
        Event emitted after the call(s):
        - Events.LoanExtended(_idCollateral,_idLoan) (ZexusLender.sol#2546)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
console._sendLogPayloadImplementation(bytes) (ZexusLender.sol#9-24) uses assembly
        - INLINE ASM (ZexusLender.sol#12-23)
console._castToPure(function(bytes)) (ZexusLender.sol#26-32) uses assembly
        - INLINE ASM (ZexusLender.sol#29-31)
Address.verifyCallResult(bool,bytes,string) (ZexusLender.sol#1885-1903) uses assembly
        - INLINE ASM (ZexusLender.sol#1895-1898)
Strings.toString(uint256) (ZexusLender.sol#2038-2056) uses assembly
        - INLINE ASM (ZexusLender.sol#2043-2045)
        - INLINE ASM (ZexusLender.sol#2048-2050)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version^0.8.17 (ZexusLender.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in TransferHelper.safeApprove(address,address,uint256) (ZexusLender.sol#1587-1597):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (ZexusLender.sol#1590-1592)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (ZexusLender.sol#1599-1609):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (ZexusLender.sol#1602-1604)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (ZexusLender.sol#1611-1626):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (ZexusLender.sol#1619-1621)
Low level call in TransferHelper.safeTransferETH(address,uint256) (ZexusLender.sol#1628-1631):
        - (success) = to.call{value: value}(new bytes(0)) (ZexusLender.sol#1629)
Low level call in Address.sendValue(address,uint256) (ZexusLender.sol#1827-1832):
        - (success) = recipient.call{value: amount}() (ZexusLender.sol#1830)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (ZexusLender.sol#1854-1865):
        - (success,returndata) = target.call{value: value}(data) (ZexusLender.sol#1863)
Low level call in Address.functionStaticCall(address,bytes,string) (ZexusLender.sol#1871-1880):
        - (success,returndata) = target.staticcall(data) (ZexusLender.sol#1878)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Contract console (ZexusLender.sol#5-1553) is not in CapWords
Parameter ZexusLender.acceptBorrowerOffer(bytes32)._idCollateral (ZexusLender.sol#2351) is not in mixedCase
Parameter ZexusLender.loanDefaulted(bytes32)._collateralId (ZexusLender.sol#2413) is not in mixedCase
Parameter ZexusLender.acceptLoanExtensionAsLender(CollateralOffer,bytes)._signature (ZexusLender.sol#2451) is not in mixedCase
Parameter ZexusLender.setUpVault(address)._vault (ZexusLender.sol#2558) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
ZexusLender.zexusStorage (ZexusLender.sol#2336) should be immutable
ZexusLender.zexusUtils (ZexusLender.sol#2337) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
ZexusLender.sol analyzed (21 contracts with 84 detectors), 434 result(s) found
```

Slither log >> ZexusSecurity.sol

```
Strings.toString(uint256) (ZexusSecurity.sol#8-26) uses assembly
        - INLINE ASM (ZexusSecurity.sol#13-15)
        - INLINE ASM (ZexusSecurity.sol#18-20)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

AccessControl._setRoleAdmin(bytes32,bytes32) (ZexusSecurity.sol#259-263) is never used and should be removed
Context._msgData() (ZexusSecurity.sol#77-79) is never used and should be removed
Strings.toHexString(address) (ZexusSecurity.sol#46-48) is never used and should be removed
Strings.toHexString(uint256) (ZexusSecurity.sol#28-32) is never used and should be removed
Strings.toHexString(uint256,uint256) (ZexusSecurity.sol#34-44) is never used and should be removed
Strings.toString(uint256) (ZexusSecurity.sol#8-26) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (ZexusSecurity.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
ZexusSecurity.sol analyzed (9 contracts with 84 detectors), 9 result(s) found
```

Slither log >> ZexusStorage.sol

```
Strings.toString(uint256) (ZexusStorage.sol#145-163) uses assembly
        - INLINE ASM (ZexusStorage.sol#150-152)
        - INLINE ASM (ZexusStorage.sol#155-157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

AccessControl._setRoleAdmin(bytes32,bytes32) (ZexusStorage.sol#396-400) is never used and should be removed
Context._msgData() (ZexusStorage.sol#214-216) is never used and should be removed
Counters.decrement(Counters.Counter) (ZexusStorage.sol#127-133) is never used and should be removed
Counters.reset(Counters.Counter) (ZexusStorage.sol#135-137) is never used and should be removed
Strings.toHexString(address) (ZexusStorage.sol#183-185) is never used and should be removed
Strings.toHexString(uint256) (ZexusStorage.sol#165-169) is never used and should be removed
Strings.toHexString(uint256,uint256) (ZexusStorage.sol#171-181) is never used and should be removed
Strings.toString(uint256) (ZexusStorage.sol#145-163) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (ZexusStorage.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.
6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Parameter ZexusStorage.updateZexusWallet(address)._wallet (ZexusStorage.sol#461) is not in mixedCase
Parameter ZexusStorage.updateBorrowersOffers(bytes32,CollateralMeta)._idCollateral (ZexusStorage.sol#471) is not in mixedCase
Parameter ZexusStorage.updateBorrowersOffers(bytes32,CollateralMeta)._collateralMeta (ZexusStorage.sol#472) is not in mixedCase
Parameter ZexusStorage.updateBorrowersOffersNfts(bytes32,bytes)._idCollateral (ZexusStorage.sol#481) is not in mixedCase
Parameter ZexusStorage.updateBorrowersOffersCollateral(bytes32,Collateral)._idCollateral (ZexusStorage.sol#490) is not in mixed
Case
Parameter ZexusStorage.updateBorrowersOffersCollateral(bytes32,Collateral)._collateral (ZexusStorage.sol#491) is not in mixedCa
se
Parameter ZexusStorage.deleteBorrowersOffers(bytes32)._idCollateral (ZexusStorage.sol#497) is not in mixedCase
Parameter ZexusStorage.getBorrowersOffers(bytes32)._idCollateral (ZexusStorage.sol#503) is not in mixedCase
Parameter ZexusStorage.updateActiveCollaterals(address,bytes32,bool)._user (ZexusStorage.sol#509) is not in mixedCase
Parameter ZexusStorage.updateActiveCollaterals(address,bytes32,bool)._idCollateral (ZexusStorage.sol#510) is not in mixedCase
Parameter ZexusStorage.updateActiveCollaterals(address,bytes32,bool)._active (ZexusStorage.sol#511) is not in mixedCase
Parameter ZexusStorage.getActiveCollaterals(address,bytes32)._user (ZexusStorage.sol#517) is not in mixedCase
Parameter ZexusStorage.getActiveCollaterals(address,bytes32)._idCollateral (ZexusStorage.sol#518) is not in mixedCase
Parameter ZexusStorage.deleteActiveCollaterals(address,bytes32)._user (ZexusStorage.sol#524) is not in mixedCase
Parameter ZexusStorage.deleteActiveCollaterals(address,bytes32)._idCollateral (ZexusStorage.sol#525) is not in mixedCase
Parameter ZexusStorage.updateCurrency(address,bool)._token (ZexusStorage.sol#531) is not in mixedCase
Parameter ZexusStorage.updateCurrency(address,bool)._supported (ZexusStorage.sol#532) is not in mixedCase
Parameter ZexusStorage.getSupportedCurrency(address)._token (ZexusStorage.sol#538) is not in mixedCase
Parameter ZexusStorage.updateNftWhitelisted(address,bool)._token (ZexusStorage.sol#543) is not in mixedCase
Parameter ZexusStorage.updateNftWhitelisted(address,bool)._whitelisted (ZexusStorage.sol#544) is not in mixedCase
Parameter ZexusStorage.getNftWhitelisted(address)._token (ZexusStorage.sol#550) is not in mixedCase
Parameter ZexusStorage.nonces(address,bytes32)._collateralId (ZexusStorage.sol#554) is not in mixedCase
Parameter ZexusStorage.useNonce(address,bytes32)._collateralId (ZexusStorage.sol#560) is not in mixedCase
Variable ZexusStorage._nonces (ZexusStorage.sol#447) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
ZexusStorage.sol analyzed (12 contracts with 84 detectors), 35 result(s) found
```

Solidity Static Analysis

Utils.sol

Gas costs:

Gas requirement of function Utils.addNFT is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 37:4:

Gas costs:

Gas requirement of function Utils.removeNFT is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 48:4:

This on local calls:

Use of "this" for local functions: Never use "this" to call functions in the same contract, it only consumes more gas than normal local calls.
more
Pos: 152:29:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 73:8:

Email: audit@EtherAuthority.io

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')
Pos: not available

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 30:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 199:19:

Vault.sol

Gas costs:

Gas requirement of function Vault.withdrawNFTs is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 27:4:

Gas costs:

Gas requirement of function Vault.emergencyWithdrawal is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 57:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 62:8:

Constant/View/Pure functions:

IVault.withdrawNFT(address,uint256,address) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 11:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 52:8:

ZexusBorrower.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ZexusBorrower.acceptLoanExtensionAsBorrower(struct Loan,bytes,bytes32): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 199:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 52:33:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 50:27:

Gas costs:

Gas requirement of function ZexusBorrower.unpause is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 19:4:

Similar variable names:

ZexusBorrower.acceptLoanExtensionAsBorrower(struct Loan,bytes,bytes32) : Variables have very similar names "oldLoan" and "_idLoan". Note: Modifiers are currently not considered by this static analysis.
Pos: 250:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 358:8:

ZexusCollateral.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ZexusCollateral.cancelCollateral(bytes32): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 199:4:

Gas costs:

Gas requirement of function ZexusCollateral.addNFTCollateral is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 32:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 125:8:

Similar variable names:

ZexusCollateral.inArrays(address,uint256,address[],uint256[]) : Variables have very similar names "_asset" and "_assets". Note: Modifiers are currently not considered by this static analysis.
Pos: 188:29:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 231:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 144:16:

ZexusLender.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ZexusLender.acceptLoanExtensionAsLender(struct CollateralOffer,bytes): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 142:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 152:38:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

more

Pos: 50:27:

Gas costs:

Gas requirement of function ZexusLender.setUpVault is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 251:4:

Similar variable names:

ZexusLender._withdrawNFT(bytes,address) : Variables have very similar names "_nft" and "_nfts". Note: Modifiers are currently not considered by this static analysis.

Pos: 247:27:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 51:8:

ZexusStorage.sol

Gas costs:

Gas requirement of function ZexusStorage.updateZexusWallet is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 30:4:

Constant/View/Pure functions:

ZexusStorage.loanId() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 145:4:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 97:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 116:8:

Solhint Linter

Utils.sol

```
Compiler version ^0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path
@openzeppelin/contracts/token/ERC721/IERC721.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:3
global import of path
@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
Pos: 1:4
global import of path Events.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:6
global import of path IStorage.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:7
global import of path IUtils.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:8
global import of path Model.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:10
global import of path ./ZexusSecurity.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:11
Explicitly mark visibility of state
Pos: 5:17
Explicitly mark visibility of state
Pos: 5:25
Explicitly mark visibility of state
Pos: 5:26
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:28
```

Vault.sol

```
Compiler version ^0.8.17 does not satisfy the ^0.5.8 semver
requirement
```

```
Pos: 1:1
global import of path
@openzeppelin/contracts/token/ERC721/IERC721.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:3
global import of path @openzeppelin/contracts/token/ERC20/IERC20.sol
is not allowed. Specify names to import individually or bind all
exports of the module into a name (import "path" as Name)
Pos: 1:4
global import of path
@openzeppelin/contracts/access/AccessControl.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:5
global import of path IVault.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:6
global import of path ./ZexusSecurity.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:7
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:15
```

ZexusBorrower.sol

```
Compiler version ^0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path
@openzeppelin/contracts/token/ERC721/IERC721.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:3
global import of path
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
Pos: 1:4
global import of path @openzeppelin/contracts/token/ERC20/IERC20.sol
is not allowed. Specify names to import individually or bind all
exports of the module into a name (import "path" as Name)
Pos: 1:5
global import of path IVault.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:7
global import of path IUtils.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:8
global import of path IStorage.sol is not allowed. Specify names to
```

```
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:9
global import of path Model.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:11
global import of path Events.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:12
global import of path TransferHelper.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:13
global import of path IBorrower.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:16
Explicitly mark visibility of state
Pos: 5:24
Explicitly mark visibility of state
Pos: 5:25
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:29
Avoid making time-based decisions in your business logic
Pos: 34:51
Avoid making time-based decisions in your business logic
Pos: 21:113
Avoid making time-based decisions in your business logic
Pos: 13:145
Avoid making time-based decisions in your business logic
Pos: 34:223
Error message for require is too long
Pos: 9:231
```

ZexusCollateral.sol

```
Compiler version ^0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path Model.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:3
global import of path Events.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:4
global import of path
@openzeppelin/contracts/token/ERC721/IERC721.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:8
```

```
global import of path ./ZexusSecurity.sol is not allowed. Specify
names to import indivdually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:9
Explicitly mark visibility of state
Pos: 5:15
Explicitly mark visibility of state
Pos: 5:16
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:20
Error message for require is too long
Pos: 9:42
Error message for require is too long
Pos: 9:95
Error message for require is too long
Pos: 9:103
Error message for require is too long
Pos: 9:204
```

ZexusLender.sol

```
Compiler version ^0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path
@openzeppelin/contracts/token/ERC721/IERC721.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:3
global import of path
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol is not
allowed. Specify names to import individually or bind all exports of
the module into a name (import "path" as Name)
Pos: 1:4
global import of path IUtils.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:8
global import of path IStorage.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:9
Unexpected import of console file
Pos: 1:11
global import of path hardhat/console.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:11
global import of path Model.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:13
global import of path Events.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
```

```
(import "path" as Name)
Pos: 1:14
global import of path TransferHelper.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:15
global import of path ./ZexusSecurity.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:17
global import of path ILender.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:18
Explicitly mark visibility of state
Pos: 5:28
Explicitly mark visibility of state
Pos: 5:29
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:32
Error message for require is too long
Pos: 9:49
Avoid making time-based decisions in your business logic
Pos: 28:92
Avoid making time-based decisions in your business logic
Pos: 17:121
Avoid making time-based decisions in your business logic
Pos: 39:151
```

ZexusSecurity.sol

```
Compiler version ^0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path
@openzeppelin/contracts/access/AccessControl.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:3
global import of path @openzeppelin/contracts/security/Pausable.sol
is not allowed. Specify names to import individually or bind all
exports of the module into a name (import "path" as Name)
Pos: 1:4
global import of path
@openzeppelin/contracts/security/ReentrancyGuard.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:5
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:10
```

ZexusStorage.sol

```
Compiler version ^0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path @openzeppelin/contracts/utils/Counters.sol is
not allowed. Specify names to import individually or bind all exports
of the module into a name (import "path" as Name)
Pos: 1:3
global import of path Model.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:5
global import of path IStorage.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:6
global import of path ./ZexusSecurity.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:7
Explicitly mark visibility of state
Pos: 5:14
Explicitly mark visibility of state
Pos: 5:15
Explicitly mark visibility of state
Pos: 5:16
Explicitly mark visibility of state
Pos: 5:25
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io