Security Assessment

Lever

Jun 12th, 2021

Table of Contents

Summary

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

The auditing process pays special attention to the following considerations:

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

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

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Lever
Description	Lever is the first AMM-based decentralized margin trading platform on Ethereum, where users can easily earn interest through lending and perform leveraged trading.
Platform	Ethereum, BSC, Heco
Language	Solidity
Codebase	https://github.com/levernetwork/protocol-v2
Commit	1e37358f3f65d52761d26132efa55585fbbde706

Audit Summary

Delivery Date	Jun 12, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	11
Critical	0
Major	0
Medium	0
Minor	3
Informational	8
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
SLL	deployments/StringLib.sol	a724138f0d1ad9871ffae334848f0511856f73425fa0cf63ea1904c1e1981555
VTH	deployments/VariableTokensHelper.sol	0cd949e385cbaeea9cdfb4777378769af88653bb7e8405d7ef882c2a9cd5a464
XTA	deployments/XTokensAndRatesHelper.sol	5ff12c33251c234017cb3b7ac3b25e3667808358a6c447acd92e0e450a926349
ICA	interfaces/IChainlinkAggregator.sol	28974b04e70fd8eab226869f673623fa70eef24143028f5b802d3853f307418a
ICD	interfaces/ICreditDelegationToken.sol	125568736c49a159181a45539a0d8d855c5a10f0169ce734f7adf65e7d6dbcd1
IER	interfaces/IERC20.sol	2a4f43d584487883a3682ca7a8d893547e06e3e62d02fdea23240d385410247d
IEC	interfaces/IERC20Detailed.sol	ff0e3e3be350d76f4b82cf0d02ab9c20080918b899d69172d112f62fce66fb86
IEW	interfaces/IERC20WithPermit.sol	d106fd5c3448a4c0a6e756b422e8d0961f35d46c3c420100875184fc13e4786f
IMP	interfaces/IMarginPool.sol	74aa270363f819ef843e9da212e71c50b23bd3f63f46a225e177f35b59a73427
IMA	interfaces/IMarginPoolAddressesProvider.sol	c76422305bf4160a5f952ed1fd0f3dd957b07cdc96ba5c9c873280793cd292b3
IMC	interfaces/IMarginPoolCollateralManager.sol	ba47729e6cde47a568ee86b6548311a03cfa5e5a76e48f2f5d0e57099e3dc187
IPO	interfaces/IPriceOracle.sol	dfdec67cda259719670454b208a2f1c4256a2c2ae22edd8bb116b2a33a16ee7c
IPG	interfaces/IPriceOracleGetter.sol	63974b7301b3a6e47ff5ce77a82f69000dfcdf8707815bdb4fdab3cb2e1e6e68
IRI	interfaces/IReserveInterestRateStrategy.sol	634f4ff349f8af05b1ff78adf8f928f4f1dd4681b1e3e354fb9cb5c68b0d8116
ISB	interfaces/IScaledBalanceToken.sol	00433055f1a71ec98dcd47b9c69a218931ba496251d28577fa12072c727641cf
ITC	interfaces/ITokenConfiguration.sol	85539bfd27b3c5c437f8b397661d474e4f63c51166cd44324b11efaff6c1ca68

ID	file	SHA256 Checksum
IUV	interfaces/IUniswapV2Router01.sol	0df6e5429688d5a5417c19ac1bc09f03d1c3d2b06230399a4530e1b05ff0083d
IUR	interfaces/IUniswapV2Router02.sol	0eabf4f06a2c2ca666e929da3eb0e963009736978677020e4738de0e58333ae0
IVD	interfaces/IVariableDebtToken.sol	e699f6da54bbbe693cf47e969c611ade3370998b55081c91c25f284cc6d876a2
IXT	interfaces/IXToken.sol	a7e41862c04fe0b6322b7956b9fe11b28686e9c9d1c3be66119c2d897fe7c729
DRI	protocol/Marginpool/DefaultReserveInterestRateStrategy.sol	b4aa1fcd566d6b3056c652ca804f95d05412b557eabf61dbe7a1078a554fafd6
MPM	protocol/Marginpool/MarginPool.sol	5139ce39b629330f583675e9a22f45d440ea4af096ce8890ff870a44592a267c
MPC	protocol/Marginpool/MarginPoolCollateralManager.sol	8712913986677367a1161c0ee31ed9e97e35633665aa543a2e693b7f49ec155d
MPL	protocol/Marginpool/MarginPoolConfigurator.sol	c3b6560337b6963cb4f84eb60400d9e2f6e643e87a2d7d5811041f576dba7fd5
MPS	protocol/Marginpool/MarginPoolStorage.sol	3259e1c0b1e1c20a61503c491e9e09a6e0d76da91f28d76f10c6ace3b553c278
OBM	protocol/Marginpool/OrderBook.sol	0dc6cb238d8b1b589b035d9f3ec574e9d6333f3080b26896c13ac33efffbf1ae
MPA	protocol/configuration/MarginPoolAddressesProvider.sol	b1a520683ffb9262fd284ee661a1c91d15bdcf9eff01f164da9451d24369fd92
RCL	protocol/libraries/configuration/ReserveConfiguration.sol	7037ae9413c6e6c30916cf92220ba420b9730c120cc08bef2e034142d1340afe
UCL	protocol/libraries/configuration/UserConfiguration.sol	5398b860b98dbab76d4532dd85b99d658e0436a4415afbf95d461761fc59f651
ELL	protocol/libraries/helpers/Errors.sol	61c083c2ac018a11bf6d7f8655516a6c2cafd82da9becbfab840f258405120de
HLL	protocol/libraries/helpers/Helpers.sol	93c1f6d091ccea0c46e93737550fbc93e92d827395a6abffc184848f79ba2272

ID	file	SHA256 Checksum
GLL	protocol/libraries/logic/GenericLogic.sol	228d8f408b894c73918fe2a94ec3ad973c2e4afc9091c0bb3873ea446dd128ae
RLL	protocol/libraries/logic/ReserveLogic.sol	1ea3ef1a5f7ad7c49e466db6051b08539755f8a25383af29edb66f4771004438
VLL	protocol/libraries/logic/ValidationLogic.sol	2633a12ab639a68081bdf2e141fdfeb72608ea523547a1a1db7378989674b188
MUL	protocol/libraries/math/MathUtils.sol	42d555f4c7c99decd1313183cab11a98df9083d66e75c1fe46132534cc88f8a5
PML	protocol/libraries/math/PercentageMath.sol	27151d877a4b253d9ba09e587199e759bd5afe121cf1c7ca3af9f36302791b38
WRM	protocol/libraries/math/WadRayMath.sol	581472f74bac0394560a57fe96a362c2a4b56917cd9ff72f6d6a7bd800ab97d3
DTL	protocol/libraries/types/DataTypes.sol	6591148864a9553018321edb5cc52d5c51300f3ceea1897218e44d091d283d89
BIA	protocol/libraries/upgradeability/BaseImmutableAdminUpgradeabilityProxy.sol	77d0974ca2920d23c7f66af88ddfb2eb6620a703d12cc565ec6725db3df06e6e
IIU	protocol/libraries/upgradeability/InitializableImmutableAdminUpgradeabilityProxy.sol	e4301fc0f8214074ec11764972f33fb4e08857237b0f4363cf36885d29cd7e60
VIL	protocol/libraries/upgradeability/VersionedInitializable.sol	e208a1297526b7ac6bc7262b801706733dda375a889036fe62536aa4778986ae
IEL	protocol/tokenization/IncentivizedERC20.sol	86aad1064206135efd66106ac639462e0e0d9343d54c588abefe6ae90f408fd2
VDT	protocol/tokenization/VariableDebtToken.sol	00ac4ded3b0e6fdeb41e145f36894e8e08ed7df96ad3165d88fec2ff5024665b
XTL	protocol/tokenization/XToken.sol	3c8ab1f700acf08c82115cd97a2a803cfe1ce3ecc38d932080233ee665b79a6d
DTB	protocol/tokenization/base/DebtTokenBase.sol	8d503854d52a3b0bbcc18c73e80136f8b4ab20eb45502d7e1ccc8762629bbca1

Review Summary

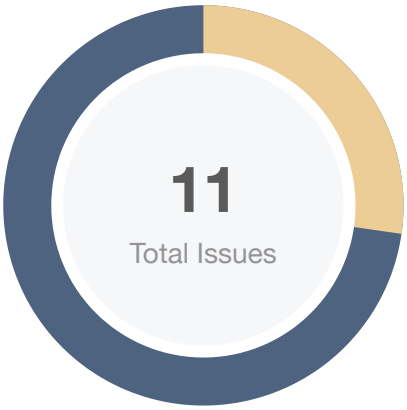
This protocol based on the Aave V2 protocol and improved. This protocol removed the flash loan and add the pending order function. This protocol enables depositors to provide liquidity and earn a passive income proportionate to their deposit borrowers can utilize the deposited capital by borrowing it in an overcollateralized fashion.

Certain mathematical formulas are defined within the code, which means implementing the formulas detailed in the whitepaper of Aave, including linear and compound interest. We validated these formulas as well as rounding adaptations made to commonly used libraries such as `WadRayMath`.

The protocol optimizes its configuration parameters via bitwise operations, enabling a single data bit to represent a boolean flag. We validated that the bitwise operand utilized by the `ReserveConfiguration` and `UserConfiguration` implementation of the said single bit representation. The protocol utilizes an adaptation of the `openzeppelin` proxy pattern whereby a versioning system was introduced. We verified that the implementation was sound and does not deviate from the standard.

Overall, no serious vulnerabilities were observed in the codebase and the code itself was well readable and developed conforming to the latest standards in Solidity.

Findings



Critical	**0** (0.00%)
Major	**0** (0.00%)
Medium	**0** (0.00%)
Minor	**3** (27.27%)
Informational	**8** (72.73%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
DTB-01	Redundant Expression	Coding Style	Informational	Acknowledged
MPM-01	Compiler Error	Compiler Error	Minor	Resolved
MPM-02	Using Undefined Error Code	Compiler Error	Minor	Resolved
MPM-03	Missing Emit Events	**Centralization / Privilege**	**Informational**	**Acknowledged**
MPM-04	Missing Zero Address Validation	Optimization	Informational	Resolved
OBM-01	Missing An Input Parameter	Coding Style	Informational	Resolved
RLL-01	Unused Return Value	Gas Optimization	Informational	Resolved
XTA-01	Unused Variables	Gas Optimization	Informational	Resolved
XTL-01	Unused Variables	Gas Optimization	Informational	Resolved
XTL-02	Uninitialized Variables	Logical Issue	Minor	Resolved
XTL-03	Comparison Before Division	Optimization	Informational	Resolved

DTB-01 | Redundant Expression

Category	Severity	Location	Status
Coding Style	Informational	protocol/tokenization/base/DebtTokenBase.sol: 79, 80, 91, 92	Acknowledged

Description

Some functions in this contract contain redundant code.

Functions like `allowance`, `transfer`, `approve`, `transferFrom`, `increaseAllowance` and `decreaseAllowance`.

Recommendation

Consider removing them.

Alleviation

No Alleviation.

MPM-01 | Compiler Error

Category	Severity	Location	Status
Compiler Error	Minor	protocol/Marginpool/MarginPool.sol: 318	Resolved

Description

This statement is grammatically wrong in solidity， maybe it lacks `(`.

Recommendation

Consider modifying it like below:

```
  ValidationLogic.validateSwap(_user, _reserves, _usersConfig[_user], _reservesList,
_reservesCount, _addressesProvider.getPriceOracle());
```

Alleviation

The team heeded our advice and changed in new version.

MPM-02 | Using Undefined Error Code

Category	Severity	Location	Status
Compiler Error	Minor	protocol/Marginpool/MarginPool.sol: 85	Resolved

Description

The error code `Errors.SDT_BURN_EXCEEDS_BALANCE` is undefined.

Recommendation

Consider adding the error code in `Errors.sol`.

Alleviation

The team heeded our advice and changed in new version.

MPM-03 | Missing Emit Events

Category	Severity	Location	Status
Centralization / Privilege	Informational	protocol/Marginpool/MarginPool.sol: 111~123	Acknowledged

Description

Several sensitive actions are defined without event declarations.

Functions like: `setCollateralManager`, `setBorrowFee`, `setWithdrawFee`.

Recommendation

Consider adding events for sensitive actions, and emit them in the function like below.

```
event _setBorrowRate(uint8 fee);

function setBorrowFee(uint8 _fee) override external onlyMarginPoolConfigurator {
   require(_fee <= 100, "fee must be less than 1%");
   borrowFee = _fee;
   emit _setBorrowRate(_fee);
}
```

Alleviation

No alleviation.

MPM-04 | Missing Zero Address Validation

Category	Severity	Location	Status
Optimization	Informational	protocol/Marginpool/MarginPool.sol: 111~113	Resolved

Description

The parameter `_collateralManager` is missing address zero checks.

Recommendation

Consider adding zero address checks, for example:

```
  function setCollateralManager(address _collateralManager) override external
onlyMarginPoolConfigurator {
    require(_collateralManager != address(0), "ERR_ZERO_ADDRESS");
    collateralManager = _collateralManager;
  }
```

Alleviation

The team heeded our advice and changed in new version.

OBM-01 | Missing An Input Parameter

Category	Severity	Location	Status
Coding Style	Informational	protocol/Marginpool/OrderBook.sol: 183~190	Resolved

Description

This calling of function `swapOrderWithUni` lacks an input parameter `isUni`, though the default value of bool type is `false` when it's missing, we recommend adding `false` as default value to improve code readability. And the default name of the warning message in Line 192 should be `"SushiSwap failed"`.

Recommendation

We recommend the following coding style:

```
bool result =
  pool.swapOrderWithUni(
    order.maker,
    order.amountInOffered,
    order.amountOutExpected,
    createPair(order.tokenIn, order.tokenOut),
    isOpenPosition,
    false
  );
require(result, "SushiSwap failed");
```

Alleviation

The team heeded our advice and changed in new version.

RLL-01 | Unused Return Value

Category	Severity	Location	Status
Gas Optimization	Informational	protocol/libraries/logic/ReserveLogic.sol: 114, 237~275	Resolved

Description

These return values `newLiquidityIndex` and `newVariableBorrowIndex` are unused, for code conciseness, the method `_updateIndexes` does not need to return values.

Alleviation

The team heeded our advice and changed in new version.

XTA-01 | Unused Variables

Category	Severity	Location	Status
Gas Optimization	Informational	deployments/XTokensAndRatesHelper.sol: 15	Resolved

Description

Variables like `EIP712_REVISION` and `EIP712_DOMAIN` in the contract `XToken.sol` and `pool` in the contract `XTokensAndRatesHelper.sol` are never used.

Recommendation

We recommend removing these unused variables.

Alleviation

The team heeded our advice and changed in new version.

XTL-01 | Unused Variables

Category	Severity	Location	Status
Gas Optimization	Informational	protocol/tokenization/XToken.sol: 108, 113	Resolved

Description

Variables like `EIP712_REVISION` and `EIP712_DOMAIN` in the contract `XToken.sol` and `pool` in the contract `XTokensAndRatesHelper.sol` are never used.

Recommendation

We recommend removing these unused variables.

Alleviation

The team heeded our advice and changed in new version.

XTL-02 | Uninitialized Variables

Category	Severity	Location	Status
Logical Issue	Minor	protocol/tokenization/XToken.sol: 127	Resolved

Description

The variable `DOMAIN_SEPARATOR` is uninitialized and it could be declared `constant` and name it as UPPER_CASE_WITH_UNDERSCORE style.

Recommendation

We recommend initializing the variable before use it. And decalre it constant, rename it as UPPER_CASE_WITH_UNDERSCORE style.

Alleviation

The team heeded our advice and changed in new version.

XTL-03 | Comparison Before Division

Category	Severity	Location	Status
Optimization	Informational	protocol/tokenization/XToken.sol: 514~530	Resolved

Description

The `require` checks in the `if` and `else` statement could be improved.

Recommendation

we recommend changing the code as the following example:

```
if (block.timestamp >= periodFinish) {
  require(reward <= balance, "Provided reward too high");
  rewardsDuration = _rewardsDuration;
  periodFinish = block.timestamp.add(rewardsDuration);
} else {
  uint256 remaining = periodFinish.sub(block.timestamp);
  uint256 leftover = remaining.mul(rewardRate);
  reward = reward.add(leftover);
  require(reward <= balance, "Provided reward too high");
  rewardRate = reward.div(remaining);
}
```

Alleviation

The team heeded our advice and changed in new version.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.