CERTIK

Security Assessment

XPLUS

Jun 1st, 2022

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Understanding

- External Dependencies
- Privileged Roles

Findings

- GLOBAL-01 : Unlocked Compiler Version
- CKP-01 : Centralization Related Risks
- CKP-02 : Missing Emit Events
- GBD-01 : In Consistent Lower Bound for Voting Period Range
- GBD-02 : Discussion on `MIN_VOTING_DELAY`
- GBI-01 : Unused Interface
- LCK-01 : Lack of Checks On Range
- NFC-01 : Potential Bypass of "Listing Fee"
- NFC-02 : Invalid "if" Condition
- NFC-03 : Lack of Input Validation
- NFT-01 : Variables That Could Be Declared as Immutable
- TCK-01 : Unchecked Low-level Call
- TCK-02 : Potential Front-Running Risk
- XPL-01 : Initial Token Distribution
- XPL-02 : Delegation Not Moved Along With `_mint`

Appendix

Disclaimer

About

Summary

This report has been prepared for XPLUS to discover issues and vulnerabilities in the source code of the XPLUS project 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;
- 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	XPLUS
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/XPLUSIO/xplus-contracts
Commit	• e87c1e5bb16dd530475230e47e52abb4e32b769c • f4c83c400205517375099f50ae7e3d1f502b9c7e • 8b000556e1360ae82fb465204c9a1ccba0f22bf8

Audit Summary

Delivery Date	Jun 01, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	2	0	0	2	0	0	0
Medium	2	0	0	0	0	0	2
Minor	4	0	0	1	0	0	3
Informational	7	0	0	2	0	0	5
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
GCC	governance/GovernanceControl.sol	3bf993958c4ea558b933a58f994246fb49e33516ed02f87a38660f8edaf88e32
TCK	governance/Treasury.sol	4fb0ef8a8329c3ea918572023ab7a8a47ee669cee05f7aa544e6daa5e7acb6a3
XPL	XPLUSToken.sol	6e2b79df26d3dab8f7436fa15559faf2cf07b87f439c808542bd2ad530072f6f
GBC	governance/GovernorBravoDelegator.sol	30914f51bb60f4279ed408a88e34d5d68948e376da3d6a43a624c93b01338225
NFC	NFTmarketplace.sol	54b6f33634e4ac663667ce40a709851507e97b5e2981199505e000d37608ec24
GBI	governance/GovernorBravoInterfaces.sol	fdd90b9eacc6cf5e6d28cc97b47952621b7535619b369de45ecf9799145e567e
TLC	governance/TimeLock.sol	7a961ba94df324f4fc0d997b34d9bf9ffa80eef83174ba9d0f833a89e5deda5a
LCK	Leveler.sol	3ce6ad9ee767d8ea225655001b006b4ca070a3d0615e68219199ad1e5741aed1
GBD	governance/GovernorBravoDelegate.sol	bd28545350b6ea3c4923e9d21e146fb969cd0ddf4f613b62d6cd636ac58595fb
CKP	governance	
NFT	NFT.sol	695465dda039d7ce6f245309282b4adf9a9bf49703d3f079d1c89fe58a3c2664

Understanding

External Dependencies

The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets.

There are a few depending injection contracts or addresses in the current project:

- `marketAddress` for contract `NFT`;
- `listFeeCollector`, `saleFeeCollector`, `purchaseToken` and `cashier` for `Market`;
- `implementation` for `GovernorBravoDelegator`;
- `timelock` and `xplus` for `GovernorBravoDelegate`;

In addition, the contract is serving as the underlying entity to interact with third-party contracts and interfaces:

- `ERC721Enumerable`, `ERC721URIStorage`, `ERC721`, `Ownable` for contract `NFT`;
- `SafeERC20`, `Counters`, `Ownable`, `Pausable` for contract `Market`;
- `ERC20`, `SafeMath`, `Arrays`, `Counters` for contract `XPLUS`;
- `Ownable` for contract `Leveler`;
- `SafeERC20Upgradeable`, `Initializable` for contract `Treasury`.
- `ContextUpgradeable`, `Initializable` for contract `GovernanceControl`;

We assume these vulnerable actors and implement proper logic to collaborate with the current project.

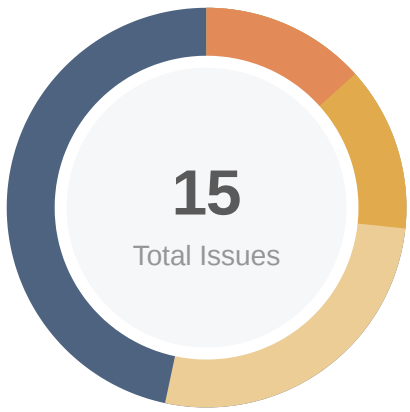
Privileged Roles

The following roles are adopted to enforce the access control:

- Role `_owner` is adopted to update configurations of the contract `Leveler`.
- Role `_owner` is adopted to update configurations of the contract `NFT` and mint reserved NFTs.
- Role `_owner` is adopted to update configurations of the contract `Market`.
- Role `admin` is adopted to update configurations of the contract `XPLUS`.
- Role `admin` is adopted to initialize and update configurations of the contract `GovernorBravoDelegate`.
- Role `admin` is adopted to execute the transaction process of the contract `Timelock`.
- Role `_governance` is adopted to update configurations of the contract `GovernanceControl`.
- Role `_governance` is adopted to distribute tokens in the contract `Treasury`.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of `Timelock` contract.

Findings



Critical	**0** (0.00%)
Major	**2** (13.33%)
Medium	**2** (13.33%)
Minor	**4** (26.67%)
Informational	**7** (46.67%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Unlocked Compiler Version	Language Specific	Informational	Acknowledged
CKP-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
CKP-02	Missing Emit Events	Coding Style	Informational	Resolved
GBD-01	In Consistent Lower Bound For Voting Period Range	Logical Issue	Minor	Resolved
GBD-02	Discussion On `MIN_VOTING_DELAY`	Logical Issue	Informational	Resolved
GBI-01	Unused Interface	Gas Optimization	Informational	Acknowledged
LCK-01	Lack Of Checks On Range	Logical Issue	Informational	Resolved
NFC-01	Potential Bypass Of "Listing Fee"	Logical Issue	Medium	Resolved
NFC-02	Invalid "if" Condition	Logical Issue	Medium	Resolved
NFC-03	Lack Of Input Validation	Volatile Code	Minor	Resolved
NFT-01	Variables That Could Be Declared As Immutable	Gas Optimization	Informational	Resolved
TCK-01	Unchecked Low-level Call	Language Specific	Minor	Resolved
TCK-02	Potential Front-Running Risk	Volatile Code	Minor	Acknowledged

ID	Title	Category	Severity	Status
XPL-01	Initial Token Distribution	**Centralization / Privilege**	**Major**	Acknowledged
XPL-02	Delegation Not Moved Along With `_mint`	Logical Issue	Informational	Resolved

GLOBAL-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational		Acknowledged

Description

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

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.8.4` the contract should contain the following line:

```
pragma solidity 0.8.4;
```

Alleviation

[XPLUS]: The complier version is locked in HardHat. The team acknowledged this issue and decided not to change the current codebase at this stage.

CKP-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**	Leveler.sol: 22, 36; NFT.sol: 40, 75, 84; NFTmarketplace.sol: 103, 111, 118, 125, 132, 138, 144; XPLUSToken.sol: 89, 117; governance/GovernanceControl.sol: 32, 37; governance/GovernorBravoDelegate.sol: 61, 265, 282, 300, 318, 334, 346, 356; governance/GovernorBravoDelegator.sol: 62, 62; governance/Treasury.sol: 32, 41, 50; governance/TimeLock.sol: 82, 102, 117	ⓘ Acknowledged

Description

In the contract `Leveler`, the role `_owner` has authority over the following functions:

- `setPriceByLevel()` will change the level price by level id.
- `setupOnce()` will init the information for each level, including `price`, `lower`, and `upper`.
- `renounceOwnership()` will renounce the ownership of the contract.
- `transferOwnership()` will transfer ownership to a new address.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority, for example, manipulate the token price.

In the contract `NFT`, the role `_owner` has authority over the following functions:

- `setMarketAddress` will change the market address.
- `safeMint()` will mint the reserved NFT.
- `safeMintById` will mint the non-reserved NFT.
- `renounceOwnership()` will renounce the ownership of the contract.
- `transferOwnership()` will transfer ownership to a new address.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority, for example, mint tokens freely.

In the contract `NFT`, the role `marketAddress` has authority over the following functions:

- `safeMintById` will mint the non-reserved NFT.

Any compromise to the `marketAddress` account may allow a hacker to take advantage of this authority, for example, mint non-reserved tokens freely.

In the contract `Market`, the role `_owner` has authority over the following functions:

- `setTokenLevel` will set leveler address for each token type.
- `setCashier` will update the `cashier` address.
- `setListFeeCollector` will update the `listFeeCollector` address.
- `setSaleFeeCollector` will update the `saleFeeCollector` address.
- `setListFeeRatio` will update the `listFeeRatio`.
- `setSaleFeeRatio` will update the `saleFeeRatio`.
- `whitelist` will change the token's whitelist status.
- `renounceOwnership()` will renounce the ownership of the contract.
- `transferOwnership()` will transfer ownership to a new address.

Since commit 8b000556e1360ae82fb465204c9a1ccba0f22bf8 the project adding following privileged funtion for the role `_owner` for `Market` contract:

- `pause()` will pause the market.
- `unpause()` will restart the market.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority, for example, manipulate the contract setting.

In the contract `XPLUS`, the role `admin` has authority over the following functions:

- `setPendingAdmin` will update the `pendingAdmin` address.
- `snapshot` will increase the value of the `_currentSnapshotId`

Any compromise to the `admin` account may allow a hacker to take advantage of this authority.

In the contract `GovernanceControl`, the role `_executor` has authority over the following functions:

- `setGovernance` will change the `_governance` address.
- `setExecutor` will change the `_executor` address.

Any compromise to the `_executor` account may allow a hacker to take advantage of this authority.

In the contract `GovernorBravoDelegate`, the role `admin` has authority over the following functions:

- `initialize` will initialize the contract.
- `setVotingDelay` update the `votingDelay`.
- `setVotingPeriod` update the `votingPeriod`.
- `setProposalThreshold` update the `proposalThreshold`.

- `setWhitelistAccountExpiration` update the `whitelistAccountExpirations`.
- `setWhitelistGuardian` update the `whitelistGuardian`.
- `initiate` set timelock as an admin.
- `setPendingAdmin` update the `pendingAdmin`.

Any compromise to the `admin` account may allow a hacker to take advantage of this authority, for example, manipulate the contract setting.

In the contract `GovernorBravoDelegator`, the role `admin` has authority over the following functions:

- `setImplementation` will update the `implementation`

Any compromise to the `admin` account may allow a hacker to take advantage of this authority.

In the contract `Treasury`, the role `_executor` has authority over the following functions:

- `increaseAllowance` will increase the allowance of a spender.
- `decreaseAllowance` will decrease the allowance of a spender.
- `transfer` will transfer tokens to specified users.
- `setGovernance` will change the `_governance` address.
- `setExecutor` will change the `_executor` address.

Any compromise to the `_executor` account may allow a hacker to take advantage of this authority, for example, manipulate the contract setting.

In the contract Timelock`, the role` admin` has authority over the following functions:

- function `queueTransaction()` will add a transaction to the execution queue.
- function `cancelTransaction()` will cancel a transaction.
- function `executeTransaction()` will execute a transaction.

Any compromise to the `admin` account may allow a hacker to take advantage of this authority, for example, manipulate the transaction execution.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged accounts' private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be

improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[XPLUS]: There would be TimeLock and 3/5 multisig at deployment and will renounce the ownership to DAO.

CKP-02 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	NFT.sol: 40~43; governance/GovernorBravoDelegate.sol: 61~89; governance/GovernanceControl.sol: 32~40	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications. For example,

- `NFT.setMarketAddress()`
- `GovernorBravoDelegate.initialize()`
- `GovernanceControl.setGovernance()`
- `GovernanceControl.setExecutor()`

Recommendation

Consider adding events for sensitive actions, and emit them in the functions.

Alleviation

[XPLUS]: The team resolved this issue in commit d168ea5102d2185c80c615c22b73fe50d4a87537 by adding aforementioned events.

GBD-01 | In Consistent Lower Bound For Voting Period Range

Category	Severity	Location	Status
Logical Issue	Minor	governance/GovernorBravoDelegate.sol: 19~20	Resolved

Description

The range of the voting period is limited by `MIN_VOTING_PERIOD` and `MAX_VOTING_PERIOD`.

```
19      /// The minimum setable voting period
20      uint256 public constant MIN_VOTING_PERIOD = 1; // About 24 hours
21
22      /// The max setable voting period
23      uint256 public constant MAX_VOTING_PERIOD = 80640; // About 2 weeks
```

But the given value is inconsistent for `MIN_VOTING_PERIOD` and `MAX_VOTING_PERIOD`. If we assume 80640 is about 2 weeks, then 24 hours should be 80640/(2*7) = 5760.

Also, the basic unit for this value is not likely to be the "block". If 5760 refers to the block number generated in one day, it is about 4 seconds per block which is much less than the actual Ethereum block time, about 12 seconds.

Reference: Block time

Recommendation

Recommend reconsidering the voting period range to avoid inaccurate voting period.

Alleviation

[XPLUS]: The team resolved this issue by changing the value of `MIN_VOTING_PERIOD` to 5760 in commit d168ea5102d2185c80c615c22b73fe50d4a87537.

GBD-02 | Discussion On `MIN_VOTING_DELAY`

Category	Severity	Location	Status
Logical Issue	Informational	governance/GovernorBravoDelegate.sol: 26	Resolved

Description

In contract `GovernorBravoDelegate`, the `MIN_VOTING_DELAY` is set to 0, which is used as a lower bound for "voting delay". Based on the "voting delay" validation, the admin can select the "voting delay" to 0. This will lead the proposal become active immediately.

```
        require(
            votingDelay_ >= MIN_VOTING_DELAY &&
                votingDelay_ <= MAX_VOTING_DELAY,
            "GovernorBravo: invalid delay"
        );
```

Recommendation

We would like to check with the team if it is the intended design.

Alleviation

[XPLUS]: This is by design. There can be a case to active the proposal with no delay.

GBI-01 | Unused Interface

Category	Severity	Location	Status
Gas Optimization	Informational	governance/GovernorBravoInterfaces.sol: 46~49	Acknowledged

Description

The interface `GovernorAlphaInterface` has not been implemented or used in the audited project.

Recommendation

Recommend removing the unused interface to save gas.

Alleviation

[XPLUS]: The team acknowledged this issue and decided not to change the current codebase at this stage.

LCK-01 | Lack Of Checks On Range

Category	Severity	Location	Status
Logical Issue	Informational	Leveler.sol: 53	Resolved

Description

In LevelRangeStruct, the `lower` and `upper` field is used to indicate the token id range. It is necessary `lower` less or equal to `upper` to avoid unexpected result in `getLevelPriceById` function.

Recommendation

Recommend validation on each pair of `lower` and `upper` value, for example:

```
require(_lowers[I] <= _uppers[i],"invalid range!");
```

Alleviation

[XPLUS]: The team resolved this issue by adding lower and upper validation for each pair in commit d168ea5102d2185c80c615c22b73fe50d4a87537.

NFC-01 | Potential Bypass Of "Listing Fee"

Category	Severity	Location	Status
Logical Issue	Medium	NFTmarketplace.sol: 161, 184	Resolved

Description

In function `listToken`, the "Listing Fee" will be collected at L157-L161. The amount of the "Listing Fee" is based on the `price` and `listFeeRatio`. The `listFeeRatio` is set by the contract owner and the `price` can be updated by the token seller through the function `changePrice()`. But inside the `changePrice()` the "Listing Fee" will not be updated with the price change.

```
156         if (listFeeRatio > 0) {
157             SafeERC20.safeTransferFrom(
158                 IERC20(purchaseToken),
159                 msg.sender,
160                 listFeeCollector,
161                 (price * listFeeRatio) / 1000
162             );
163         }
```

```
184     function changePrice(uint256 listingId, uint256 price) public {
185         Listing storage listing = _listings[listingId];
186         require(
187             listing.status == ListingStatus.Active,
188             "This listing is not active"
189         );
190         require(msg.sender == listing.seller, "Market: only seller can change");
191         require(price > 0, "Market: zero price");
192         uint256 oldPrice = listing.price;
193         listing.price = price;
194 
195         emit PriceChange(listingId, oldPrice, listing.price);
196     }
```

This will create a potential way to avoid "Listing Fee":

1. Bob lists his NFT Token and set the price as 0 in the function `listToken`.
2. Bob changes his NFT Token price by calling the function `changePrice`.

In this case, Bob will not pay the "Listing Fee" because the amount calculation (`(price * listFeeRatio) / 1000`) in the function `listToken` will return 0. After that, he can change the price he wants by invoking the

function `changePrice`.

Recommendation

Recommend update "Listing Fee" in the function `changePrice` or use fixed fees to avoid unexpected loss.

Alleviation

[XPLUS]: The team resolved this issue by moving the logic of the fees to the `buyToken` function in commit d168ea5102d2185c80c615c22b73fe50d4a87537.

NFC-02 | Invalid "if" Condition

Category	Severity	Location	Status
Logical Issue	Medium	NFTmarketplace.sol: 236	Resolved

Description

In function `buyToken()`, the fee will be calculated with `saleFeeRatio` at L237. But the `if` branch will validate the fee ratio larger than zero with `listFeeRatio` which is invalid.

```
236  if (listFeeRatio > 0) {
237        feeAmount = (listing.price * saleFeeRatio) / 1000;
238        SafeERC20.safeTransferFrom(
239            IERC20(purchaseToken),
240            msg.sender,
241            saleFeeCollector,
242            feeAmount
243        );
244  }
```

Recommendation

Recommend updating the code by checking `saleFeeRatio` instead, for example:

```
if(saleFeeRatio>0){
      feeAmount = (listing.price * saleFeeRatio) / 1000;
      SafeERC20.safeTransferFrom(
          IERC20(purchaseToken),
          msg.sender,
          saleFeeCollector,
          feeAmount
      );
}
```

Alleviation

[XPLUS]: The team resolved this issue by checking `saleFeeRatio` instead in commit d168ea5102d2185c80c615c22b73fe50d4a87537.

NFC-03 | Lack Of Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	NFTmarketplace.sol: 132~142	Resolved

Description

`listFeeRatio` and `saleFeeRatio` are the fee ratio used in functions `listToken` and `buyToken` to calculate the amount of fee. For example,

```
237    feeAmount = (listing.price * saleFeeRatio) / 1000;
```

It will be divided by the denominator "1000" to complete the calculation, which means the `listFeeRatio` and `saleFeeRatio` should not exceed 1000 to let the fee ratio valid and not exceed 100%.

Recommendation

Recommend adding `require(_feeRatio <= 1000,"invalid input");` in function `setListFeeRatio` and `setSaleFeeRatio` to avoid invalid fee ratio.

Alleviation

[XPLUS]: The team resolved this issue by adding aforementioned input validation in commit d168ea5102d2185c80c615c22b73fe50d4a87537.

NFT-01 | Variables That Could Be Declared As Immutable

Category	Severity	Location	Status
Gas Optimization	Informational	NFT.sol: 16	Resolved

Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

Recommend declaring these variables as immutable. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

Alleviation

[XPLUS]: The team resolved this issue by declaring the linked variable as immutable in commit d168ea5102d2185c80c615c22b73fe50d4a87537.

TCK-01 | Unchecked Low-level Call

Category	Severity	Location	Status
Language Specific	Minor	governance/Treasury.sol: 56	Resolved

Description

```
56  payable(to).call{value: amount};
```

The return value of a low-level call is not checked.

The low-level `call` function returns the status of the call as the first variable in the returned tuple. The status of the `call` is not asserted to be `true` which will treat the low-level call as a success even in the event it reverts.

Recommendation

Recommend checking the return value of a low-level call or log it.

Alleviation

[XPLUS]: The team resolved this issue by checking the return value of the low-level call in commit d168ea5102d2185c80c615c22b73fe50d4a87537.

TCK-02 | Potential Front-Running Risk

Category	Severity	Location	Status
Volatile Code	Minor	governance/Treasury.sol: 28~30	Acknowledged

Description

In the contract `Treasury`, malicious hackers may observe the pending transaction which will execute the `initialize` function and launch a similar transaction but with the hacker's address of `governance_` and gain the ownership of the contract.

Recommendation

We advise the client to design functionality to only allow a specific user to execute the `initialize` function.

Alleviation

[XPLUS]: The team acknowledged this issue and decided not to change the current codebase at this stage.

XPL-01 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	**Major**	XPLUSToken.sol: 86	Acknowledged

Description

All of the `XPLUS` tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute `XPLUS` tokens without obtaining the consensus of the community. Additionally, all the voting power will be transferred to the deployer, thus introducing centralization risk.

Recommendation

Recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Alleviation

[XPLUS]: Tokens will be allocated to different treasury and TimeLock after deployed and renounce the ownership to DAO.

XPL-02 | Delegation Not Moved Along With `_mint`

Category	Severity	Location	Status
Logical Issue	Informational	XPLUSToken.sol: 84~87	Resolved

Description

In the `constructor`, the `_mint()` function is called to mint tokens to the `_admin` but the `delegate` of the `_admin` is not updated.

Recommendation

Consider adding call of `_moveDelegates()` in the constructor:

```
84 constructor(address _admin) ERC20("XPLUS Token", "XPLUS") {
85         admin = _admin;
86         _mint(_admin, MAX_SUPPLY);
87         _moveDelegates(address(0), _admin, MAX_SUPPLY);
88 }
```

Alleviation

[XPLUS]: The team acknowledges the finding and this is the intended design.

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.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

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.

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 you (“Customer” or 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, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND

“AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK’S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER’S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED “AS IS” AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK’S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

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.