CERTIK

Preliminary Comments

ADX

Aug 15th, 2021

Table of Contents

Summary

This report has been prepared for ADX to discover issues and vulnerabilities in the source code of the ADX 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	ADX
Platform	Ethereum, BSC
Language	Solidity
Codebase	https://bscscan.com/token/0x6bff4fb161347ad7de4a625ae5aa3a1ca7077819 https://etherscan.io/token/0xade00c28244d5ce17d72e40330b1c318cd12b7c3
Commit	

Audit Summary

Delivery Date	Aug 15, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	ⓘ Pending	⊗ Declined	ⓘ Acknowledged	Partially Resolved	✓ Resolved
● Critical	0	0	0	0	0	0
● Major	2	2	0	0	0	0
● Medium	0	0	0	0	0	0
● Minor	2	2	0	0	0	0
● Informational	2	2	0	0	0	0
● Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
ADX	bsc/ADXToken.sol	9513071424736dd7ccf8c5ebf1e7340cf656c6221f5880c924144dc7d7f5dc70
ADT	ether/ADXToken.sol	c1f4edbf2f8362db5fd1acda9aaa7ca2257e9af41d75f182c0074acbbd864098

System Overview

The `ADXToken` token deployed on the Binance Smart Chain.

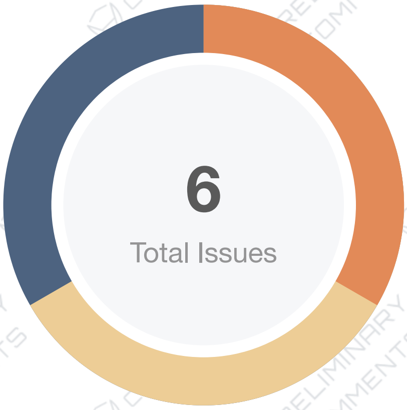
Here is some information on the `ADX` token that we found on the Binance Smart Chain:

- Total Supply: 9300000e18
- Max Cap: Unlimit
- SupplyController: 0x515144d4708a43927cf2edcbd429fb08524766fb
- SupplyController privileges:
  - SupplyController has the privilege to transfer or renounce the ownership.
  - SupplyController has the privilege to mint uncapped `ADX` to itself.

As for the `ADX` token deployed on Ethereum :

- Total Supply: 160064429672926627981986396
- Max Cap: Unlimit
- Minter: 0x9d47f1c6ba4d66d8aa5e19226191a8968bc9094e
- SupplyController privileges:
  - SupplyController has the privilege to mint `ADX` tokens to any account by the `mint()` function.
  - SupplyController has the privilege to change minter by the `changeSupplyController()` function.

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**2** (33.33%)
🟨 **Medium**	**0** (0.00%)
🟨 **Minor**	**2** (33.33%)
🟦 **Informational**	**2** (33.33%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
ADT-01	Unlocked Compiler Version Declaration	Language Specific	● Informational	(!) Pending
ADT-02	Lack of Input Validation	Volatile Code	● Minor	(!) Pending
ADT-03	Centralization Risk	**Centralization / Privilege**	● **Major**	(!) Pending
ADX-01	Lack of Input Validation	Volatile Code	● Minor	(!) Pending
ADX-02	Centralization Risk	**Centralization / Privilege**	● **Major**	(!) Pending
ADX-03	Unlocked Compiler Version Declaration	Language Specific	● Informational	(!) Pending

ADT-01 | Unlocked Compiler Version Declaration

Category	Severity	Location	Status
Language Specific	● Informational	ether/ADXToken.sol: 5	ⓘ Pending

Description

The compiler version utilized throughout the project uses the `^` prefix specifier, denoting that a compiler version that is greater than the version will be used to compile the contracts.

Recommendation

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

ADT-02 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Minor	ether/ADXToken.sol: 129, 138, 145, 153, 171, 176	ⓘ Pending

Description

The input value `supplyControllerAddr`, `to`, `from`, `spender`, `owner` and `newSupplyController` should be verified as non-zero address to prevent being mistakenly assigned as address(0) in the functions. Violation of this may cause losing the minting ability

Recommendation

We advise the client to check that the address is not zero by adding the following checks in the functions:

```
129 constructor(address supplyControllerAddr, address prevTokenAddr) public {
130    require(supplyControllerAddr != address(0), "supplyController is zero address");
131    supplyController = supplyControllerAddr;
132    PREV_TOKEN = prevTokenAddr;
133 }
```

```
138 function transfer(address to, uint amount) external returns (bool success) {
139    require(to != address(0), "transfer to the zero address");
140    balances[msg.sender] = balances[msg.sender].sub(amount);
141    balances[to] = balances[to].add(amount);
142    emit Transfer(msg.sender, to, amount);
143    return true;
144 }
```

```
145 function transferFrom(address from, address to, uint amount) external returns (bool
success) {
146    require(from != address(0), "transfer from the zero address");
147    balances[from] = balances[from].sub(amount);
148    allowed[from][msg.sender] = allowed[from][msg.sender].sub(amount);
149    balances[to] = balances[to].add(amount);
150    emit Transfer(from, to, amount);
151    return true;
152 }
```

```
153 function approve(address spender, uint amount) external returns (bool success) {
154    require(spender != address(0), "spender is zero address");
```

```
    allowed[msg.sender][spender] = amount;
    emit Approval(msg.sender, spender, amount);
    return true;
}
```

```
function mint(address owner, uint amount) external {
    require(owner != address(0), "owner is zero address");
    require(msg.sender == supplyController, 'NOT_SUPPLYCONTROLLER');
    innerMint(owner, amount);
}
```

```
function changeSupplyController(address newSupplyController) external {
    require(newSupplyController != address(0), "newSupplyController is zero address");
    require(msg.sender == supplyController, 'NOT_SUPPLYCONTROLLER');
    supplyController = newSupplyController;
}
```

ADT-03 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	ether/ADXToken.sol: 171~174, 176~179, 130	ⓘ Pending

Description

The supplyController of the contract `ADXToken` has permission to:

- mint unlimited amount of tokens to any account by calling the function `mint()`,
- change supplyController by calling the function `changeSupplyController()`,

without obtaining the consensus of the community.

Recommendation

We recommend the client carefully managing the supplyController account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

`[Certik]`: By the time of 2021-08-12 07:25 UTC, the supplyController of this contract `ADXToken.sol` is 0x9d47f1c6ba4d66d8aa5e19226191a8968bc9094e, at block height 13009490 on the Ethereum Chain.

Contract deployment is at https://etherscan.io/token/0xade00c28244d5ce17d72e40330b1c318cd12b7c3

ADX-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Minor	bsc/ADXToken.sol: 133, 142, 149, 157, 175, 180	ⓘ Pending

Description

The input value `supplyControllerAddr`, `to`, `from`, `spender`, `owner` and `newSupplyController` should be verified as non-zero address to prevent being mistakenly assigned as address(0) in the functions. Violation of this may cause losing the minting ability

Recommendation

We advise the client to check that the address is not zero by adding the following checks in the functions:

```
129 constructor(address supplyControllerAddr, address prevTokenAddr) public {
130    require(supplyControllerAddr != address(0), "supplyController is zero address");
131    supplyController = supplyControllerAddr;
132    PREV_TOKEN = prevTokenAddr;
133 }
```

```
138 function transfer(address to, uint amount) external returns (bool success) {
139    require(to != address(0), "transfer to the zero address");
140    balances[msg.sender] = balances[msg.sender].sub(amount);
141    balances[to] = balances[to].add(amount);
142    emit Transfer(msg.sender, to, amount);
143    return true;
144 }
```

```
145 function transferFrom(address from, address to, uint amount) external returns (bool
success) {
146    require(from != address(0), "transfer from the zero address");
147    balances[from] = balances[from].sub(amount);
148    allowed[from][msg.sender] = allowed[from][msg.sender].sub(amount);
149    balances[to] = balances[to].add(amount);
150    emit Transfer(from, to, amount);
151    return true;
152 }
```

```
153 function approve(address spender, uint amount) external returns (bool success) {
154    require(spender != address(0), "spender is zero address");
```

```
  allowed[msg.sender][spender] = amount;
  emit Approval(msg.sender, spender, amount);
  return true;
}
```

```
function mint(address owner, uint amount) external {
  require(owner != address(0), "owner is zero address");
  require(msg.sender == supplyController, 'NOT_SUPPLYCONTROLLER');
  innerMint(owner, amount);
}
```

```
function changeSupplyController(address newSupplyController) external {
  require(newSupplyController != address(0), "newSupplyController is zero address");
  require(msg.sender == supplyController, 'NOT_SUPPLYCONTROLLER');
  supplyController = newSupplyController;
}
```

ADX-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	bsc/ADXToken.sol: 175~178, 180~183, 134	ⓘ Pending

Description

The supplyController of the contract `ADXToken` has permission to:

- mint unlimited amount of tokens to any account by calling the function `mint()`,
- change supplyController by calling the function `changeSupplyController()`,

without obtaining the consensus of the community.

Recommendation

We recommend the client carefully managing the supplyController account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

`[Certik]`: By the time of 2021-08-12 07:02 UTC, the supplyController of this contract `ADXToken.sol` is 0x515144d4708a43927cf2edcbd429fb08524766fb, at block height 9960960 on the Binance Smart Chain.

Contract deployment is at https://bscscan.com/address/0x6bff4fb161347ad7de4a625ae5aa3a1ca7077819

ADX-03 | Unlocked Compiler Version Declaration

Category	Severity	Location	Status
Language Specific	● Informational	bsc/ADXToken.sol: 9	ⓘ Pending

Description

The compiler version utilized throughout the project uses the `^` prefix specifier, denoting that a compiler version that is greater than the version will be used to compile the contracts.

Recommendation

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

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.

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.

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.

CERTIK