SAFETIN AUDIT

AAPTITUDE

February 27th, 2022

TABLE OF CONTENTS

I. SUMMARY
II. OVERVIEW
III. FINDINGS
IV. DISCLAIMER

SUMMARY

This report was written for AAptitude (AAPT) in order to find flaws and vulnerabilities in the AAptitude project's source code, as well as any contract dependencies that weren't part of an officially recognized library.

A comprehensive examination has been performed, utilizing Static Analysis, Manual Review, and Bogged Finance Deployment 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
- Through line-by-line manual review of the entire codebase by industry expert

OVERVIEW

PROJECT SUMMARY

Project name	AAptitude
Description	AAptitude is a Hyper-Deflationary Reflection token. AAPT will be revolutionizing the way Cryptocurrency is used in e-commerce, with a sophisticated Escrow system being built into an online marketplace.
Platform	Binance Smart Chain
Language	Solidity
Codebase	https://github.com/AAptitude/aaptitude/blob/main/AAptitude.sol

FINDINGS SUMMARY

Vulnerability	Total
● Critical	0
● Major	0
● Medium	1
● Minor	1
● Informational	2

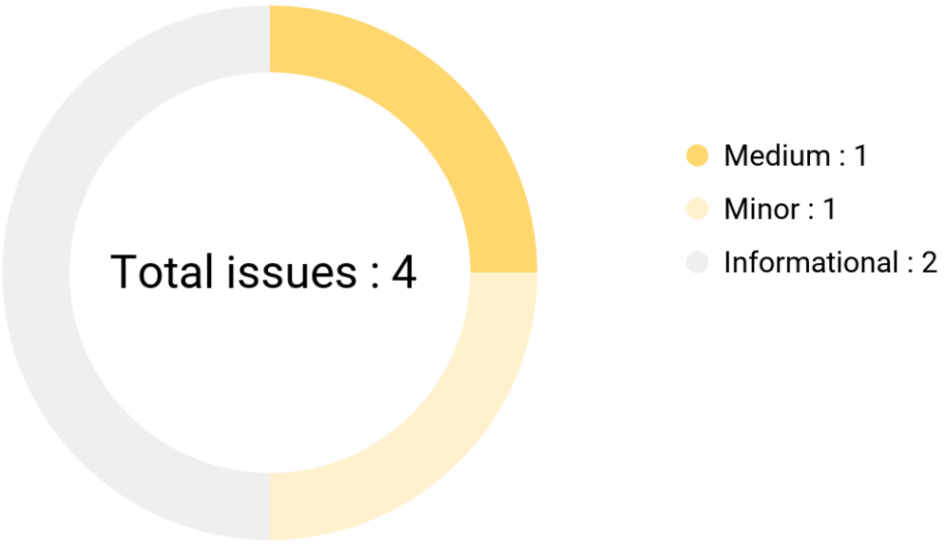
EXECUTIVE SUMMARY

AAptitude is a Binance Smart Chain project of which AAPT is the main crypto-token. The purpose of this audit is to ensure that the smart-contract of this crypto-token is secure and optimized. AAptitude's token has particular tokenomics: it is a fork of Safemoon with the addition of a transaction fee sent directly to the developer wallet after being sold automatically against BNB.

Since it is a fork of Safemoon, the smart-contract has the following two types of fees, cumulating a 10% transaction cut : the first fees (5%) are redistributed to all existing holders using a form of rebasing mechanism whilst the other 5% are accumulated internally until a sufficient amount of capital has been amassed to perform an LP acquisition. When this number is reached, the total tokens accumulated are split with half being converted to BNB and the total being supplied to the PANCAKESWAP contract as liquidity.

There have been no major or critical issues related to the codebase and all findings listed here are minor and informational. The most prominent among our findings were the centralization of the LP tokens due to the addLiquidity method.

FINDINGS



Code	Title	Severity
MSG - 1	Incorrect error message	Informational
DPD - 1	Third-party dependencies	Medium
USL - 1	Useless function deliver	Informational
CENT - 1	Centralized risk in addLiquidity	Minor

MSG-1 | Incorrect error message

Description

The error message in :

```
require(_isExcluded[account]"Account is already excluded")
```

… does not describe the error correctly.

Recommendation

The message "Account is already excluded" can be changed to "Account is not excluded".

DPD-1 | Third Party Dependencies

Description

The contract is serving as the underlying entity to interact with third party PANCAKESWAP protocols. The scope of the audit would treat those third party entities as black boxes and assume it's functional correctness. However in the real world, third parties may be compromised that led to assets lost or stolen. We understand that the business logic of the AAptitude protocol requires the interaction PancakeSwap protocol for adding liquidity to AAPT/BNB pool and swap tokens. We encourage the team to constantly monitor the statuses of those third parties to mitigate the side effects when unexpected activities are observed.

COMP-1 | Useless function deliver

Description

The function deliver, described below :

```
function deliver(uint256 tAmount) public {
        address sender = _msgSender();
        require(!_isExcluded[sender], "Excluded addresses
cannot call this function");
        (uint256 rAmount,,,,,) = _getValues(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _rTotal = _rTotal.sub(rAmount);
        _tFeeTotal = _tFeeTotal.add(tAmount);
    }
```

… is useless in AAptitude's protocol.

Recommendation

The function called deliver should be removed from the code base.

CENT-1 | Centralization risk in addLiquidity

Description

The addLiquidity function calls the UniswapV2Router.addLiquidityETH function with the address specified as owner() for acquiring the generated LP tokens from the AAPT/BNB pool. As a result, over time the _owner address will accumulate a significant portion of LP tokens. If the _owner is an EOA (Externally Owned Account) , mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

We advise the address of the UniswapV2Router.addLiquidityETH function call to be replaced by the contract itself, i.e. address (this) and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the _owner account is compromised.

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices f.e. multi signature wallets.

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 Safetin'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 Safetin 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

Safetin security assessment 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. Safetin's position is that each company and individual are responsible for their own due diligence and continuous security. Safetin'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 fun.