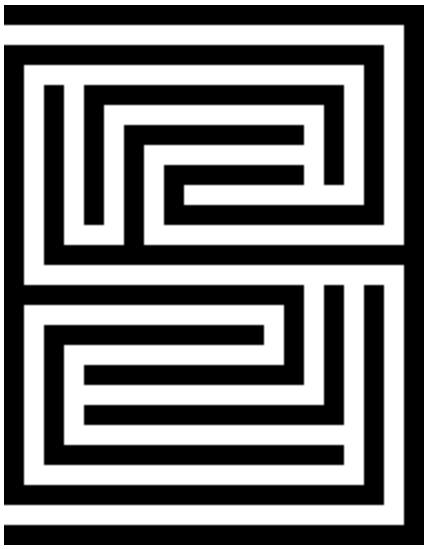

APWINE SAS Audit Public Report

PROJECT: APWINE SAS Audit

January 2020

Prepared For:

APWINE SAS Team | APWINE SAS

https://apwine.fi

Prepared By:

Jonathan Haas | Bramah Systems, LLC, a ThreatKey company

jonathan@bramah.systems

Table of Contents

APWine Protocol Review

Executive Summary

Scope of Engagement

Bramah Systems, LLC was engaged in January of 2021 to perform a comprehensive security review of the APWINE smart contracts (specific contracts denoted within the appendix). Our review was conducted over a period of three days by both members of the Bramah Systems, LLC. executive staff.

Bramah Systems completed the assessment using manual, static and dynamic analysis techniques.

Timeline

Review Commencement: January 18th, 2021

Report Delivery: January 22nd, 2021

Engagement Goals

The primary scope of the engagement was to evaluate and establish the overall security of the APWINE protocol, with a specific focus on trading actions. In specific, the engagement sought to answer the following questions:

- Is it possible for an attacker to steal or freeze tokens?
- Does the Solidity code match the specification as provided?
- Is there a way to interfere with the contract mechanisms?
- Are the arithmetic calculations trustworthy?

Contract Specification

Contract specification was provided in the form of code comments and functional unit tests, along with a verbose specification section throughout many source code elements, which provided justification for infrastructure decisions and structural fundamentals.

Overall Assessment

Bramah Systems was engaged to evaluate and identify any potential security concerns within the codebase of the APWINE SAS Protocol. During the course of our engagement, Bramah Systems identified numerous instances wherein the team deviated materially from established best practices and procedures of secure software development within DLT, as our report details.

These aside, the team otherwise used **thoroughly** reviewed and vetted components and provided details as to the token structure, economics, and intent, which helped Bramah highlight any potential concerns with their approach.

Disclaimer

As of the date of publication, the information provided in this report reflects the presently held, commercially reasonable understanding of Bramah Systems, LLC.'s knowledge of security patterns as they relate to the APWINE SAS Protocol, with the understanding that distributed ledger technologies ("DLT") remain under frequent and continual development, and resultantly carry with them unknown technical risks and flaws. The scope of the review provided herein is limited solely to items denoted within "Scope of Engagement" and contained within "Directory Structure". The report does NOT cover, review, or opine upon security considerations unique to the Solidity compiler, tools used in the development of the protocol, or distributed ledger technologies themselves, or to any other matters not specifically covered in this report.

The contents of this report must NOT be construed as investment advice or advice of any other kind. This report does NOT have any bearing upon the potential economics of the APWINE SAS protocol or any other relevant product, service or asset of APWINE SAS or otherwise. This report is not and should not be relied upon by APWINE SAS or any reader of this report as any form of financial, tax, legal, regulatory, or other advice.

To the full extent permissible by applicable law, Bramah Systems, LLC. disclaims all warranties, express or implied. The information in this report is provided "as is" without warranty, representation, or guarantee of any kind, including the accuracy of the information provided. Bramah Systems, LLC. makes no warranties, representations, or guarantees about the APWINE SAS Protocol. Use of this report and/or any of the information provided herein is at the users sole risk, and Bramah Systems, LLC. hereby disclaims, and each user of this report hereby waives, releases, and holds Bramah Systems, LLC. harmless from, any and all liability, damage, expense, or harm (actual, threatened, or claimed) from such use.

Timeliness of Content

All content within this report is presented only as of the date published or indicated, to the commercially reasonable knowledge of Bramah Systems, LLC. as of such date, and may be superseded by subsequent events or for other reasons. The content contained within this report is subject to change without notice. Bramah Systems, LLC. does not guarantee or warrant the accuracy or timeliness of any of the content contained within this report, whether accessed through digital means or otherwise.

Bramah Systems, LLC. is not responsible for setting individual browser cache settings nor can it ensure any parties beyond those individuals directly listed within this report are receiving the most recent content as reasonably understood by Bramah Systems, LLC. as of the date this report is provided to such individuals.

General Recommendations

Best Practices & Solidity Development Guidelines

Solidity version pragma not locked

Solidity source files indicate the versions of the compiler they can be compiled with via a version specifier (the pragma).

pragma solidity ^ 0.6.17; // Compiles with 0.6.17 and later

pragma solidity 0.6.3; // Compiles with 0.6.3 exclusively

As later compiler versions may handle certain language aspects in a way the developer may not have foreseen, it is recommended to use a locked version pragma.

Resolution: The team introduced a fix in commit **40bb00115f4e62ff6863875f167b580d5995863d**.

Redundant comparison to zero could result in underflow

As the uint data type can never be negative, comparison with zero (greater than or equal) is redundant and likely to result in underflow issues. It is suggested that arithmetic using this logic is rewritten.

Lines: RateFuture.sol, L56

Resolution: The team introduced a fix in **bfba30096c255c5cb63bca290d9f66b8cea6cb2b**.

Sensitive parameter changing functions should emit an event

As various parameter setting functions all allow for modification of potential rewards flow to users, it is suggested that these functions emit an event on invocation (including functions that involve contract initialization)

Lines: Throughout

Resolution: Event emittance was introduced in commit ID

4f26b59cd69d9e11bd9558ded1700ee128c53401 which removes these concerns.

TODO items remain within source code

TODO items still remain within the source code, indicating certain structural elements that should be decided before publication to the blockchain.

Lines: Future.sol, L176,195,231

Resolution: The team has introduced functionality pertaining to these TODO items and removed the outstanding TODO labels as of commit **40bb00115f4e62ff6863875f167b580d5995863d**.

Numerous typographical errors

Throughout the protocol, there are multiple lines which feature typographical errors, particularly within error messages returned (examples presented below):

require(_periodIndex < getNextPeriodIndex(), "The isnt any fyt for this period yet");

require(registrations[_user].startIndex == nextIndex, "The is not ongoing registration for the next period");

Resolution: These issues, where applicable, have been resolved over the course of numerous commits, but especially through the following: **04bf463020c20b6defab855b3cba0f1c2094307e** for the protocol repository and **098cba373e454b6de5d094c7f4004fb10ae76480**.

Usage of send and transfer considered against best-practice

Following EIP-1884, the usage of **transfer and send** is no longer suggested, due to changing gas costs in their usage. Use **.call.value(...)("")** instead. The contract presently makes use of **transfer** throughout.

Resolution: Usage of **.call.value(...)("")** was adopted as of commit ID **18b6be92c6a1e318f9cb79c3aa19bedd48e3a757**.

Visibility should be explicitly defined

A number of variables lack explicit visibility definitions, which could lead to misuse.

File: contracts/upgradability/Proxy.sol, L16

File: contracts/upgradability/Proxy.sol , L15

File: contracts/protocol/GaugeController.sol, L38

File: contracts/protocol/futures/StreamFuture.sol, L13

Resolution: The team introduced a fix in commit ID **512fbf15c9dab2a64a5792716a15b615499ce180**.

Specific Recommendations

Unique to the APWine Protocol

Highly permissive owner account and centralization of power

The deploying account possesses a number of highly actions (namely, changing various distribution and reward preferences -- notably mentioned in the scope of time-based promotions). This deploying account should (where possible) minimize usage of the associated key (e.g. performing transactions, using as a regular user account) and perform other operational security best practices. Potentially, this could involve transferring ownership to a MultiSignature governance.

Resolution: The team has introduced a multi-signature wallet deployment scheme as of commit **262a758cbf2cf34263010012d8df46d356d782f4** which alleviates these concerns.

Usage of ReentrancyGuard suggested

Throughout the protocol, multiple areas of concern exist for potential reentrancy. It is suggested that a ReentrancyGuard be added to prevent potential exploitation, especially in functions that violate the checks-effects-interactions pattern.

Resolution: ReentrancyGuard has been added as of commit ID **c3974c700bb1e1e741e0c864832fd505d4173bc2**.

Toolset Warnings

Unique to the APWine Protocol

Overview

In addition to our manual review, our process involves utilizing static analysis and formal methods in order to perform additional verification of the presence of security vulnerabilities (or lack thereof). An additional part of this review phase consists of reviewing any automated unit testing frameworks that exist.

The following sections detail warnings generated by the automated tools and confirmation of false positives where applicable.

Compilation Warnings

No warnings were present at time of compilation.

Test Coverage

The contract repository possesses extensive unit test coverage throughout. This testing provides a variety of unit tests which encompass the various operational stages of the contract.

Static Analysis Coverage

The contract repository underwent heavy scrutiny with multiple static analysis agents, including:

- Securify
- MAIAN
- Mythril
- Oyente
- Slither

In each case, the team had either mitigated relevant concerns raised by each of these tools or provided adequate justification for the risk (such as adhering to the ERC-20 standard).

Directory Structure

At time of review, the directory structure of the APWine smart contracts repository appeared as it does below. Our review, at request of APWine, covers the Solidity code (*.sol) as of commit-hash **f9f41eb24e638c77189ef3d630a62eb54e8d1551** of the APWine protocol repository (presented first), and **407788a0419a56385c91f925ba3e6d34377698e8** of the token repository (presented second)

```
├── README.md
├── contracts
│   ├── Migrations.sol
│   ├── interfaces
│   │   ├── ERC20.sol
│   │   ├── IProxyFactory.sol
│   │   ├── apwine
│   │   │   ├── IController.sol
│   │   │   ├── IFuture.sol
│   │   │   ├── IFutureFactory.sol
│   │   │   ├── IFutureVault.sol
│   │   │   ├── IFutureWallet.sol
│   │   │   ├── IGaugeController.sol
│   │   │   ├── IIBTFutureFactory.sol
│   │   │   ├── ILiquidityGauge.sol
│   │   │   ├── IRegistry.sol
│   │   │   ├── ITreasury.sol
│   │   │   ├── tokens
│   │   │   │   ├── IAPWToken.sol
│   │   │   │   ├── IAPWineIBT.sol
│   │   │   │   └── IFutureYieldToken.sol
│   │   │   └── utils
│   │   │       ├── IAPWineMath.sol
```

```
│   │   │       └── IAPWineNaming.sol
│   │   └── platforms
│   │       ├── aave
│   │       │   └── IAToken.sol
│   │       └── yearn
│   │           └── IyToken.sol
│   ├── protocol
│   │   ├── Controller.sol
│   │   ├── GaugeController.sol
│   │   ├── LiquidityGauge.sol
│   │   ├── Registry.sol
│   │   ├── Treasury.sol
│   │   ├── futureFactories
│   │   │   ├── FutureFactory.sol
│   │   │   └── IBTFutureFactory.sol
│   │   ├── futures
│   │   │   ├── Future.sol
│   │   │   ├── FutureVault.sol
│   │   │   ├── RateFuture.sol
│   │   │   ├── StreamFuture.sol
│   │   │   ├── futureWallets
│   │   │   │   ├── FutureWallet.sol
│   │   │   │   ├── RateFutureWallet.sol
│   │   │   │   └── StreamFutureWallet.sol
│   │   │   └── platforms
│   │   │       ├── aave
│   │   │       │   ├── AaveFuture.sol
│   │   │       │   └── AaveFutureWallet.sol
│   │   │       └── yearn
```

```
│   │   │           ├── yTokenFuture.sol
│   │   │           └── yTokenFutureWallet.sol
│   │   └── tokens
│   │       ├── APWineIBT.sol
│   │       ├── ClaimableERC20.sol
│   │       └── FutureYieldToken.sol
│   ├── upgradability
│   │   ├── BaseAdminUpgradeabilityProxy.sol
│   │   ├── BaseUpgradeabilityProxy.sol
│   │   ├── InitializableAdminUpgradeabilityProxy.sol
│   │   ├── InitializableUpgradeabilityProxy.sol
│   │   ├── Proxy.sol
│   │   ├── ProxyFactory.sol
│   │   └── UpgradeabilityProxy.sol
│   └── utils
│       ├── APWineMaths.sol
│       └── APWineNaming.sol
├── migrations
│   ├── 1_initial_migration.js
│   ├── 2_deploy_apwine_core.js
│   ├── 3_deploy_apwine_future.js
│   ├── 4_transfer_ownership.js
│   └── common.js
├── networks.js
├── package-lock.json
├── package.json
├── test
│   ├── APWineCore.test.js
│   ├── APWineUtils.test.js
```

```
│   ├── AaveFuture.test.js
│   ├── LiquidityMining.test.js
│   ├── YearnFuture.test.js
│   ├── common.js
│   └── initialize.js
├── test-environment.config.js
├── truffle-config.js
└── yarn.lock

20 directories, 69 files

├── README.md
├── contracts
│   ├── APWToken.sol
│   ├── APWVesting.sol
│   ├── Migrations.sol
│   └── interfaces
│       ├── IAPWToken.sol
│       └── IAPWVesting.sol
├── migrations
│   ├── 1_initial_migration.js
│   ├── 2_deploy_token_and_vesting.js
│   ├── 3_transfer_ownership.js
│   └── common.js
├── networks.js
├── package-lock.json
├── package.json
├── test
│   ├── APWine.test.js
```

```
│   └── common.js
└── truffle-config.js

4 directories, 16 files
```