Plenty - vote escrow smart contracts

Tezos Foundation

Independent security assessment report

inference

Report version: 1.0 / date: 12.01.2023

inference

Table of contents

Version / Date	Description
1.0 / 12.01.2023	Final version

Summary

Inference AG was engaged by the Tezos Foundation to perform an independent security assessment of Plenty's vote escrow smart contracts.

We performed the security assessment based on the agreed scope, following our approach and activities as outlined in the "Project overview" chapter between 25th May 2022 and 9th January 2023. Feedback from Plenty was received and Inference performed a follow-up assessment.

Based on our scope and our performed activities, our security assessment revealed several security issues. Additionally, different observations were also made which, if resolved with appropriate actions, may improve the quality of Plenty's vote escrow smart contracts.

This report shows remaining open or partly resolved issues and observations.

Overview on issues and observations

Details for each reported issue or observation can be obtained from the "Security issues" and "Observations" sections.

	Severity / Status
Security issues	
S-PVE-001: "next_epoch" not timely executed	medium / partially closed
S-PVE-002: Small bribes	low / partially closed
S-PVE-003: Distribution of AMM fees	low / partially closed
Observations	
O-PVE-001: Race condition in entrypoints	- / open
O-PVE-002: ve_swap exchange temporarily not possible	- / open
O-PVE-003: No two-step procedure to replace admin address	- / open

Project overview

Scope

The scope of the security assessment are the vote escrow smart contracts by Plenty as follows:

- ve-core in https://github.com/Plenty-DeFi/ve-core. Our scope includes only the smart contracts with the SmartPy code and Michelson code in the following files:
  - bribe.py and michelson/bribe.tz
  - core_factory.py and michelson/core_factory.tz
  - fee_distributor.py and michelson/fee_distributor.tz
  - gauge.py and michelson/gauge.tz
  - ply_fa12.y and michelson/ply_fa12.tz
  - ve_swap.py and michelson/ve_swap.tz
  - vote_escrow.py and michelson/vote_escrow.tz
  - voter.py and michelson/voter.tz
- flat-curve https://github.com/Plenty-DeFi/flat-curve: Our scope includes only
  - “forwardFee” entrypoint in TezToCtez.py
  - “forwardFee” entrypoint in TokenToToken.py

Our initial assessment for the considered the commits:

- e89b33609be222217008f0ec9eff628e1f9ea749 for the ve-core repository
- c54ba4360ad9e364caf2de4606a93bbf2943d76f for the flat-cure repository

The following additional information was also made available for our security assessment:

- https://whitepaper.plenty.network/amm/vote-escrow-architecture
- https://whitepaper.plenty.network/amm/vote-escrow-smart-contracts
- https://github.com/Plenty-DeFi/ve-core/tree/master/docs

Our reassessment for the ve-core repository considered the commit:
e98ed6cb5877c23576ef3a2f3c8666095a523923

No reassessment of the code in scope for the flat-curve was necessary.

The Michelson code for the following Tezos mainnet contracts were also considered and assessed:

- KT18kkvmUoefkdok5mrjU6fxsm7xmumy1NEw for vote escrow
- KT1TjNjfXf96ViEBNb14MnQdpMJba5syK6VC for core_factory (including bribe and gauge)
- KT1Mr7RmJJPYKov7goTtUmdbQkGEjNqpWS4T for fee_distributor
- KT1JVjgXPMMSaa6FkzeJcgb8q9cUaLmwaJUX for ply_fa12
- KT1RX4fN5FDPcfidbDm6mFwh26fSAMgbE98M for ve_swap
- KT1Xa92Nf6evFcEbxMXencfGPmS4urNyn5wd for voter

Scope limitation

Our security assessment is based on the following key assumptions and scope limitations:

- Any potential adversarial activities conducted by the administrator of the contract or operational errors by administrators were out of scope.
- Smart contracts for tokens not in scope were considered trusted.
- Content of metadata and token metadata was out of scope. For instance: any off-chain views have not been assessed.
- Economic model including the appropriate settings of incentives, thresholds, factors, etc. was out of scope.

Methodology

Inference's methodology for smart contract security assessments on Tezos is a combination of a source code review of the smart contract source code in the high-level language (e.g. Ligo or SmartPy), and the resulting compiled code in Michelson. This approach provides additional assurance that the compiler producing the Michelson code is correct, and does not introduce any security issues. Furthermore, this approach fosters a better understanding for smart contract users of what has been reviewed and what has been effectively deployed on-chain.

In order to ensure a high quality in our security assessments, Inference is using subject matter experts having a high adversarial scenario mindset to spot potential issues in smart contracts under review. Additionally, we apply checklists derived from good practices and commonly known issues based on the Tezos smart contract assessment checklist to document our work and to ensure good issue coverage.

Furthermore, Inference maintains regular communications with the smart contract development team to ensure a correct understanding of the smart contract solution and environment, but also to make teams aware of any observations as soon as possible.

Inference's internal quality assurance procedures ensure that results of security assessments are challenged for completeness and appropriateness by a second independent expert.

Objectives

The objectives are the identification of security issues with regards to the assessed smart contracts and its conceptual design and specification. The security assessment also focuses on adversarial scenarios on specific use cases. As such the potential adversarial scenarios as listed in appendix Adversarial scenarios have been identified together with the Plenty team and checked during our security assessment.

Activities

Our security assessment activities for the defined scope were:

- Source code review of smart contract code in SmartPy
- Source code review of the deploy smart contract versions in Michelson

We applied the checklist for smart contract security assessments on Tezos, version 1.1 obtained from https://github.com/InferenceAG/TezosSecurityAssessmentChecklist. We applied the following security checklist tables:

- System / Platform
- Storage
- Gas issues and efficiency
- Code issues
- Transactions
- Entrypoint
- On-chain views
- Admin / Operator functions
- Other topics & test cases

Our activities for the reassessment were:

- Source code review of changes in the smart contract code in SmartPy
- Source code review of the smart contracts in Michelson
- Source code review of the smart contracts deployed on Mainnet
- Reassessing security issues and observation from initial assessment in case they are claimed to be resolved

Security issues

S-PVE-001: “next_epoch” not executed on time

The design of the solution expects that the “next_epoch” entrypoint in the voter smart contract is executed on time for each epoch (1 epoch = 1 WEEK, which is defined as 7 days).

The entrypoint “next_epoch” can be called by anybody, since there is no access restriction in place. “Next_epoch” moves the epoch by one WEEK by adding to the current epoch’s end one WEEK.

Voting by users is only possible for the current set epoch and only if the defined epoch’s end has not yet been reached.

Thus, based on this design there are potentially different scenarios which could materialise. For instance:

- “next_epoch” is not called for an entire epoch and thus one epoch is literally skipped Thus,
  - inflation adjustments are distributed, but cannot be claimed by users, since nobody was able to vote in the skipped epoch. Thus, corresponding PLY tokens are locked/lost in the smart contract. Furthermore, bribes can not be claimed. However, bribes are returned to the briber if they didn’t receive any votes. Thus, bribes are not locked/lost in the smart contract.
  - AMM fees, if pulled for the skipped epoch, cannot be claimed by users, since nobody was able to vote in the skipped epoch. Thus, corresponding AMM fees are locked/lost in the smart contract.
- “next_epoch” is called very late. Thus users do not have much time left to vote. Thus, inflation adjustments, bribes, and AMM fee rewards are only distributed amongst the ones who were able to vote.
- The initial drop, one month after genesis, as well as the yearly drop of the PLY emission rewards are not done in the specific case where next_epoch has not been called in the particular time frame.

Potential reasons for “next_epoch” not being executed on time within a WEEK could be (possibly a combination of):

- blockchain downtime

- operation is deliberately not included by a baker
- front-end downtime (Note: This would only allow experienced Tezos blockchain users familiar with the CLI interface to execute “next_epoch” and vote. Thus, distribution of rewards, etc. to them only).
- wrong expectations. For instance, the expectation that “somebody” will call “next_epoch” on time.

Probability - Low. We rate this as low, since the execution of “next_epoch” has to be prevented for quite some time.
Impact - Medium. We rate this as medium, since this only affects the distribution of inflation adjustments, AMM fees, and bribes for an epoch, but not the locked assets in the platform itself.
Severity - Medium.

Recommendation:
We recommend analysing the situation and potentially considering mitigation measures such as:

- Automatic execution of “next_epoch” by a bot runned by Plenty. Measures should include the monitoring of the bot and appropriate steps in case the bot has not executed “next_epoch”.
- Detailed information on how users can call relevant entrypoints in case the official front-end is not available.
- Easy execution of “next_epoch” in the official front-end.

Comment from the Plenty team:
We have a bot running on our end that duly executes the entrypoint every Thursday at 12 AM (UTC). The core team carefully monitors the execution. The scripts for the bot can be found here: https://github.com/Plenty-DeFi/next_epoch_cron

In case, the execution has not occurred at the right time due to a potential bot failure. It can be called permissionless through https://better-call.dev/mainnet/KT1Xa92Nf6evFcEbxMXencfGPmS4urNyn5wd/interact/next_epoch

Results from follow-up assessment:
We rate Plenty’s measures as appropriate. However, since Plenty’s users are part of the solution, we updated the status from “open” to “partially closed” in order to raise awareness of this situation.

S-PVE-002: Small bribes

Anybody is allowed to provide bribes for an AMM and the bribe smart contract does not ensure that the bribes fulfil certain characteristics.

Thus, since small token amounts (or NFT) can not be correctly divided between voters, this leads to the situation that nobody gets any bribe and the small token amounts would be lost/locked in the smart contract.

Probability - Low.
Impact - Low.
Severity - Low.

Recommendation:
We recommend analysing the situation and potentially considering measures such as providing clear documentation and guidelines on how to bribe and on how a bribe should look like.

Comment from the Plenty team:
Ideally, this should not be an issue since bribes would mostly be added by protocol owners or DAOs that have a sufficient understanding of the system. However, to be on the safer side, we have to give a clear warning under the `Who can bribe?` section.

Results from follow-up assessment:
We updated the status only from “open” to “partially closed” in order to raise awareness of this situation to potential users of Plenty’s bribe feature.

S-PVE-003: Distribution of AMM fees

The distribution of AMM fees has to be triggered manually by calling “pull_amm_fee” in the voter smart contract.

The entrypoint “pull_amm_fee” can be called by anybody, since there is no access restriction in place. Calling “pull_amm_fee” will forward all accumulated fees in the AMM to the fee distributor smart contract. All forwarded fees are assigned to the epoch provided as a parameter in the entrypoint call.

The provided parameter in the entrypoint call has to be smaller than the current set epoch and fees for the epoch in the parameter have not yet been pulled/added.

There is a risk that AMM fees are not correctly assigned to the epoch in case “pull_amm_fee” is not done are not always done at the same time within an epoch.

Furthermore, there is a risk in the case of a skipped epoch that future accumulated AMM fees are pulled for this skipped epoch, resulting in fees which actually have to be assigned for the corresponding “current” epoch, but are claimed by the “old” epoch voters.

Please see also security issue S-PVE-001: “next_epoch” not executed on time regarding considerations of when, and potential reasons why, this could materialise.

Probability - Low. We rate this as low, since the execution of “pull_amm_fee” has to be prevented for quite some time.
Impact - Low. We rate this as medium, since this leads only to a wrong distribution of AMM fees amongst voters for one affected epoch.
Severity - Low.

Recommendation:
We recommend analysing the situation and potentially considering mitigation measures such as:

- Automatic execution of “pull_amm_fees” by a bot runned by Plenty. Measures should include the monitoring of the bot and appropriate steps in case the bot has not executed “pull_amm_fees”.
- Easy execution of “pull_amm_fees” in the official front-end and detailed information on how users can call relevant entrypoints in case the official front-end is not available.

Comment from the Plenty team:
The scripts in the repository https://github.com/Plenty-DeFi/next_epoch_cron also handle the calling of appropriate entrypoint for fee distribution.

In the event that the bot running the scripts ends up failing, the entrypoint can be called permissionless by passing the AMM address and epoch number through https://better-call.dev/mainnet/KT1Xa92Nf6evFcEbxMXencfGPmS4urNyn5wd/interact/pull_amm_fee

Results from follow-up assessment:
We rate Plenty's measures as appropriate. However, since Plenty's users are part of the solution, we updated the status from "open" to "partially closed" in order to raise awareness of this situation.

Observations

O-PVE-001: Race condition in entrypoints

The entrypoints “core_factory%set_fee_distributor”, “voter%set_factory_and_fee_dist”, and “vote_escrow&set_voter” can be called by everyone as long as the corresponding parameters have not yet been set. After setting the parameters, these entrypoints no longer can be used to set other parameters.

Thus, depending on how smart contracts are originated and initialised, this may open a race condition where unauthorised third parties call these entrypoint first.

Comment from the Plenty team:
The race-condition issue is duly considered, and it will be handled accordingly during deployment through a batch deployment.

O-PVE-002: ve_swap exchange temporarily not possible

The exchange of PLENTY or WRAP to PLY is potentially not possible during a short time after the defined exchange end date has been reached. This is caused in the function “upate_ledger” by the applied Euclidian division and multiplication, which may temporarily lead to a failing transaction due a resulting negative number in subtraction of two timestamps as long as the ledger_balance in the checked condition is not yet zero (“0”).

Comment from the Plenty team:
This will happen for a very small duration (possibly just one block) when nearing the end of the migration period i.e 2 years from now. A warning label will be added to the website in the approaching days.

O-PVE-003: No two-step procedure to replace admin address

The entrypoint “setAdministrator” allows setting a new address for the admin role in ply_fa12 smart contract. After verification that the current admin initiated “setAdministrator”, the “setAdministrator” entrypoint directly updates the admin role with the newly provided address.

This poses a risk that the address for the admin role is set to a wrong or non-working address.

This observation affects the ply_fa12 smart contract in scope only.

Recommendation:
We recommend implementing a two-step procedure to change any critical privileged addresses. In the first step the current privileged address proposes a new address, followed by a second step in which the newly proposed address accepts this proposal. The change of the critical privileged address is only done after the acceptance at the second step.

At the very least, we recommend analysing the situation, ensuring with appropriate documentation and procedures that the admin dealing with this contract is not mistakenly replacing the administrator with a wrong address.

Comment from the Plenty team:
The setAdministrator will be called once to replace the initial tz1 address-based admin with a multi-sig address. We shall take proper care to ensure that the admin transition is done correctly.

Disclaimer

This security assessment report ("Report") by Inference AG ("Inference") is solely intended for Tezos Foundation ("Client") with respect to the Report's purpose as agreed by the Client. The Report may not be relied upon by any other party than the Client and may only be distributed to a third party or published with the Client's consent. If the Report is published or distributed by the Client or Inference (with the Client's approval) then it is for information purposes only and Inference does not accept or assume any responsibility or liability for any other purpose or to any other party.

Security assessments of a software or technology cannot uncover all existing vulnerabilities. Even an assessment in which no weaknesses are found is not a guarantee of a secure system. Generally, code assessments enable the discovery of vulnerabilities that were overlooked during development and show areas where additional security measures are necessary. Within the Client's defined time frame and engagement, Inference has performed an assessment in order to discover as many vulnerabilities of the technology or software analysed as possible. The focus of the Report's security assessment was limited to the general items and code parts defined by the Client. The assessment shall reduce risks for the Client but in no way claims any guarantee of security or functionality of the technology or software that Inference agreed to assess. As a result, the Report does not provide any warranty or guarantee regarding the defect-free or vulnerability-free nature of the technology or software analysed.

In addition, the Report only addresses the issues of the system and software at the time the Report was produced. The Client should be aware that blockchain technology and cryptographic assets present a high level of ongoing risk. Given the fact that inherent limitations, errors or failures in any software development process and software product exist, it is possible that even major failures or malfunctions remain undetected by the Report. Inference did not assess the underlying third party infrastructure, which adds further risks. Inference relied on the correct performance and execution of the included third party technology itself.

Appendix

Adversarial scenarios

General (voter, bribe, core_factory, and fee_distributor)

Scenario	Impact rating & descriptive	Assessment result
Execution of “next_epoch” is prevented or delayed.	Medium: Inappropriate distribution of inflation adjustments, AMM fees, and bribes.	**Note** This is possible. However, since an epoch is one WEEK, we regard the risk as low that the “next_epoch” is delayed / prevented for such a long time.
Assign inflation adjustments not according to the vote shares to AMMs.	Medium: Inappropriate distribution of inflation adjustments.	**Ok** Nothing identified.
Vote with more votes than actually available, resp. vote multiple times for the same epoch.	Medium: Inappropriate distribution of inflation adjustments.	**Ok** Nothing identified.
Pull AMM fees multiple times for the same AMM and epoch.	Medium: Inappropriate distribution of AMM fees.	**Note** Nothing identified. However, due to the design of how AMM fees are pulled, one epoch may get more AMM fees assigned than another. This depends on whether and at which time an AMM fee has been pulled.
Assign fees from a specific AMM to another AMM, where a particular user has voted.	Medium: Inappropriate distribution of AMM fees.	**Ok** Nothing identified.
A higher amount of AMM fees can be claimed than the user’s actual voting share.	Medium: Inappropriate distribution of AMM fees.	**Ok** Nothing identified.
Recharge gauge multiple times for the same AMM and epoch.	Medium: Inappropriate distribution of emission rewards.	**Ok** Nothing identified.
Assign emission rewards from a specific AMM to another AMM, where a particular user has voted.	Medium: Inappropriate distribution of emission rewards.	**Ok** Nothing identified.
Bribes can be claimed by unauthorised people.	Medium: Loss of assets (medium, since this is	**Ok** Nothing identified.

Bribes are locked in bribe contracts due to wrong parameters / states.	restricted to bribes only. This is only a minor amount with regards to the entire ve-core solution).	**Ok** Nothing identified.
A higher amount of bribes can be claimed than the user's actual voting share.	Medium: Inappropriate distribution of bribes.	**Ok** Nothing identified.

vote escrow

Scenario	Impact rating & descriptive	Assessment result
Obtain more voting rights than the locked PLY would grant.	Medium: Incorrect distribution of emission rewards and inflation adjustments.	**Ok** Nothing identified.
Withdraw locked PLY before end.	Medium: Incorrect distribution of emission rewards and inflation adjustments.	**Ok** Nothing identified.
Withdraw more locked PLY than actually allowed.	High: Drainage of PLY stored in the vote-escrow contract. Loss of assets.	**Ok** Nothing identified.
Transfer of vePLY, which are attached.	Medium: Incorrect distribution of emission rewards.	**Ok** Nothing identified.

gauge

Scenario	Impact rating & descriptive	Assessment result
Get a higher stake than the provided PLY would grant.	Medium: Incorrect distribution of emission rewards.	**Ok** Nothing identified.
Withdraw more PLY than the initial provided ones.	High: Drainage of PLY stored in AMM gauge. Loss of assets.	**Ok** Nothing identified.
Get more rewards than the actual stake would grant.	Medium: Incorrect distribution of emission rewards.	**Ok** Nothing identified.

Attaching a vePLY to more than one AMM gauge.	Medium: Incorrect distribution of emission rewards.	**Ok** Nothing identified.
Get a boost without providing vePLY or get a higher boost than the provided vePLY would grant.	Medium: Incorrect distribution of emission rewards.	**Ok** Nothing identified.
Removing a vePLY without losing the boost.	Medium: Incorrect distribution of emission rewards.	**Ok** Nothing identified.

ve_swap

Scenario	Impact rating & descriptive	Assessment result
More PLY can be claimed or exchanged than actually regularly could be claimed.	Medium: Inappropriate initial distribution of PLY. Minting of too much PLY.	**Ok** Nothing identified.
The provided PLENTY or WRAP can be retrieved again.	High: Minting of PLY	**Ok** Nothing identified.

ply fa12

Scenario	Impact rating & descriptive	Assessment result
No specific adversarial scenario tests were identified and performed. Entrypoints are either regular FA12 entrypoints or access restricted entrypoints (mint). Thus, please see adversarial test cases in the voter contract.		

Risk rating definition for smart contracts

Severities are quantified with two dimensions, roughly defined as follows, whereas the examples have to be regarded as indication only:

Probability of occurrence / materialisation of an issue
(bullets for a category are linked with each other with “and/or” condition.)

- Low:
  - A trusted / privileged role is required.
  - Contract may end up in the issue if other conditions, which are also unlikely to happen, are required.
- Medium:
  - A specific role or contract state is required to trigger the issue.
  - Contract may end up in the issue if another condition is fulfilled as well.
- High:
  - Anybody can trigger the issue.
  - Contract’s state will over the short or long term end up in the issue.

Impact:
(bullets for a category are linked with each other with “and/or” condition.)

- Low:
  - Non-compliance with TZIP standards
  - Unclear error messages
  - Confusing structures
- Medium:
  - A minor amount of assets can be withdrawn or destroyed.
- High:
  - Not inline with the specification
  - A non-minor amount of assets can be withdrawn or destroyed.
  - Entire or part of the contract becomes unusable.

Severity:

	Low impact	Medium impact	High impact
High probability	**High**	**Critical**	**Critical**
Medium probability	**Medium**	**High**	**Critical**
Low probability	**Low**	**Medium**	**High**

Glossary

Term	Description
Origination	Deployment of a smart contract
TZIP	Tezos Improvement Proposal