

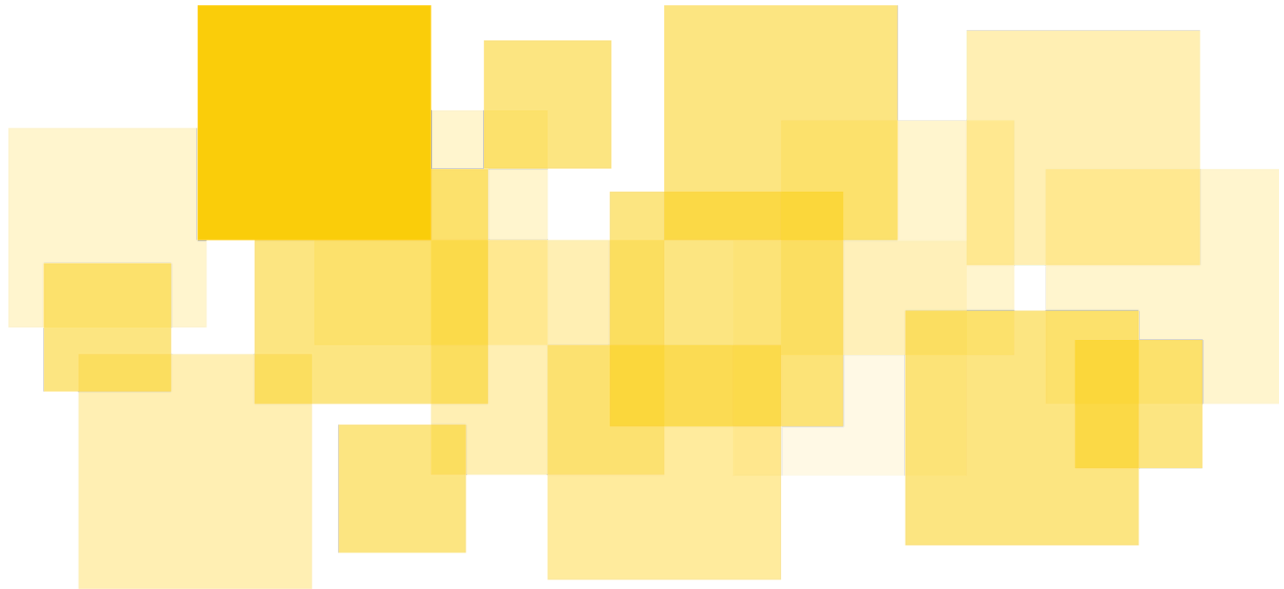


# Security Audit Report

---

## ClickPesa Oracle Aggregator Stellar

Delivered: February 20, 2025



Prepared for ClickPesa Debt Fund by





# Table of Contents

---

- [Disclaimer](#)
- [Executive Summary](#)
- [Goal](#)
- [Scope](#)
- [Methodology](#)
- [Platform Logic and Features Description](#)
  - [Oracle Aggregator](#)
- [Invariants](#)
- [Findings](#)
  - [\[A1\] The Oracle Aggregator Can Be Hijacked By Third-Parties](#)
    - [Description](#)
    - [Scenario](#)
    - [Recommendation](#)
    - [Status](#)
  - [\[A2\] The Oracle Aggregator Performs Calls to Contracts That Can Be Updated Whilst Itself Cannot](#)
    - [Description](#)
    - [Recommendation](#)
    - [Status](#)
- [Informative Findings](#)
  - [\[B1\] Best Practices and Notable Particularities](#)
  - [\[B2\] ClickPesa's Oracle Aggregator Reliance on USDC Price](#)
    - [Description](#)
    - [Recommendation](#)
    - [Status](#)



## Disclaimer

---

This report does not constitute legal or investment advice. You understand and agree that this report relates to new and emerging technologies and that there are significant risks inherent in using such technologies that cannot be completely protected against. While this report has been prepared based on data and information that has been provided by you or is otherwise publicly available, there are likely additional unknown risks which otherwise exist. This report is also not comprehensive in scope, excluding a number of components critical to the correct operation of this system. This report is for informational purposes only and is provided on an "as-is" basis and you acknowledge and agree that you are making use of this report and the information contained herein at your own risk. The preparers of this report make no representations or warranties of any kind, either express or implied, regarding the information in or the use of this report and shall not be liable to you or any third parties for any acts or omissions undertaken by you or any third parties based on the information contained herein.

Smart contracts are still a nascent software arena, and their deployment and public offering carries substantial risk.

Finally, the possibility of human error in the manual review process is very real, and we recommend seeking multiple independent opinions on any claims which impact a large quantity of funds.



# Executive Summary

---

ClickPesa Debt Fund, also referred to in this report as, simply, ClickPesa, engaged Runtime Verification Inc. to conduct a security audit of the Oracle contract code. The objective was to review the platform's business logic and implementation in Rust (Soroban) and identify any issues that could cause the system to malfunction or be exploited.

The audit was conducted over two calendar weeks (November 18, 2024, through December 2, 2024) and focused on analyzing the security of the source code of ClickPesa's Oracle. This oracle enables users and other protocols/contracts to fetch the exchange rate of ClickPesa's liquidity token. The primary user of this oracle is designed to be a Blend liquidity pool.

The audit led to identifying issues of potential severity for the protocol's health, which have been identified as follows:

- Third-party code execution: [\[A1\] The Oracle Aggregator Can Be Hijacked By Third-Parties](#), [\[A2\] The Oracle Aggregator Performs Calls to Contracts That Can Be Updated Whilst Itself Cannot](#);
- Potential threats to users' fund integrity: [\[A1\] The Oracle Aggregator Can Be Hijacked By Third-Parties](#), [\[A2\] The Oracle Aggregator Performs Calls to Contracts That Can Be Updated Whilst Itself Cannot](#), [\[B2\] ClickPesa's Oracle Aggregator Reliance on USDC Price](#).

In addition, several informative findings and general recommendations also have been made, including:

- Best practices and code optimization-related particularities: [\[B1\] Best Practices and Notable Particularities](#);
- Notes on potential economic risks: [\[B2\] ClickPesa's Oracle Aggregator Reliance on USDC Price](#);
- Blockchain-related particularities: [\[B1\] Best Practices and Notable Particularities](#).

The client has acknowledged all of the reported findings. When not intended by design according to the contract's business logic, the findings themselves will be addressed in a newer version of the audited contract that is currently in development.



## Goal

---

The goal of the audit is threefold:

- Review the high-level business logic (protocol design) of ClickPesa's system based on the provided documentation and code;
- Review the low-level implementation of the system for the individual Soroban smart contract (Oracle);
- Analyze the integration between abstractions of the modules interacting with the contract in the scope of the engagement and reason about possible exploitive corner cases.

The audit focuses on identifying issues in the system's logic and implementation that could potentially render the system vulnerable to attacks or cause it to malfunction. Furthermore, the audit highlights informative findings that could be used to improve the safety and efficiency of the implementation.



## Scope

---

The scope of the audit is limited to the code contained in a public Github repository provided by the client ([ClickPesa-Debt-Fund/oracle-aggregator](#)). Within the repository and among other files, a contract is highlighted as in the scope of the engagement. The repository and contract are described below:

1. oracle-aggregator: The files that compose this repository were [forked](#) by the client, [ClickPesa Debt Fund](#), which has been originally forked this repo from [Blend Capital's original source](#). From it, a single contract was audited:
  - Oracle (main file `src/contract.rs` , commit id `e57d0607f793a0927fc7df7d6ca7be4abcc1b83d` , branch `main` ): implements the core of the protocol, handling requests for the oracle information as well as prices for specific assets;

The comments provided in the code, a general description of the project, including samples of tests used for interacting with the platform, and online documentation provided by the client were used as reference material.

The audit is limited in scope to the artifacts listed above. Off-chain, auto-generated, or client-side portions of the codebase, as well as deployment and upgrade scripts, are not in the scope of this engagement.

Commits addressing the findings presented in this report have also been analyzed to ensure the resolution of potential issues in the protocol.



## Methodology

---

Although the manual code review cannot guarantee to find all possible security vulnerabilities as mentioned in our [Disclaimer](#), we have followed the approaches described below to make our audit as thorough as possible.

First, we rigorously reasoned about the business logic of the code, validating security-critical properties to ensure the absence of loopholes in the business logic. To this end, we carefully analyzed all the proposed features of the platform and the actors involved in the lifetime of deployed instances of the audited contracts.

Second, we thoroughly reviewed the contracts' source code to detect any unexpected (and possibly exploitable) behaviors. To facilitate our understanding of the platform's behavior, higher-level representations of the Rust codebase were created, where the most comprehensive were:

- Modeled sequences of logical operations, considering the limitations enforced by the identified invariants, checking if all desired properties hold for any possible input value;
- Manually built high-level function call maps, aiding the comprehension of the code and organization of the protocol's verification process;
- Made use of static analyzers such as [Scout](#) to identify commonly identifiable issues;
- Created abstractions of the elements outside of the scope of this audit to build a complete picture of the protocol's business logic in action.

This approach enabled us to systematically check consistency between the logic and the provided Soroban Rust implementation of the system.

Furthermore, once the creation of higher-level abstractions and the process of understanding contracts was complete, attempts to use the protocol's business logic to break the collected invariants were performed, resulting in some of the findings presented in this report.

Finally, we conducted rounds of internal discussions with security experts over the code and platform design, aiming to verify possible exploitation vectors and identify improvements for the analyzed contracts. As an outcome of these research sessions and discussions, code optimizations have also been discovered.

Additionally, given the nascent Stellar-Soroban development and auditing community, we reviewed [this list](#) of known Ethereum security vulnerabilities and attack vectors and checked



whether they apply to the smart contracts and scripts; if they apply, we checked whether the code is vulnerable to them.





## Platform Logic and Features Description

---

The ClickPesa Debt Fund empowers global investors to align their investments with positive social change. By funding local businesses, they can generate financial returns while driving sustainable growth.

ClickPesa's strategy involves collaborating with experienced Microfinance Institutions (MFIs) with a strong presence in Africa. These MFIs are well-equipped to support Small and Medium Enterprises (SMEs), especially women-owned SMEs. By partnering with these MFIs, ClickPesa aims to reach and empower businesses across the continent.

ClickPesa enables its on-chain functionalities by establishing a Blend pool, where its proprietary token, CPCT (ClickPesa Collateral Token), represents a tokenized version of Microfinance loan books. These loan books consist of small loans provided to SMEs. Through agreements with Microfinance institutions, ClickPesa secures full claims on these loan books. CPCT tokens are issued to match the exact value of the loan book, representing this claim. ClickPesa then adds CPCT tokens to the Blend pool as collateral and borrows USDC, which is channeled through Microfinances to fund SME loans.

To ensure accurate on-chain pricing for CPCT, both for users and Blend pools, ClickPesa has developed an oracle. In its current version, the oracle maintains a 1-to-1 price peg between CPCT and USDC. The contract representing this oracle is the scope of this engagement.

### Oracle Aggregator

---

Based on the Oracle Aggregator design provided by Blend ([ref](#)), ClickPesa's oracle was designed to provide the price of assets through means of other oracles, registered in this aggregator. This contract follows the Stellar Ecosystem Proposal 40 standard ([SEP-0040](#)).

Below is the function diagram of ClickPesa's oracle aggregator segregated by the contract interface available to callers and its storage manipulating functions.

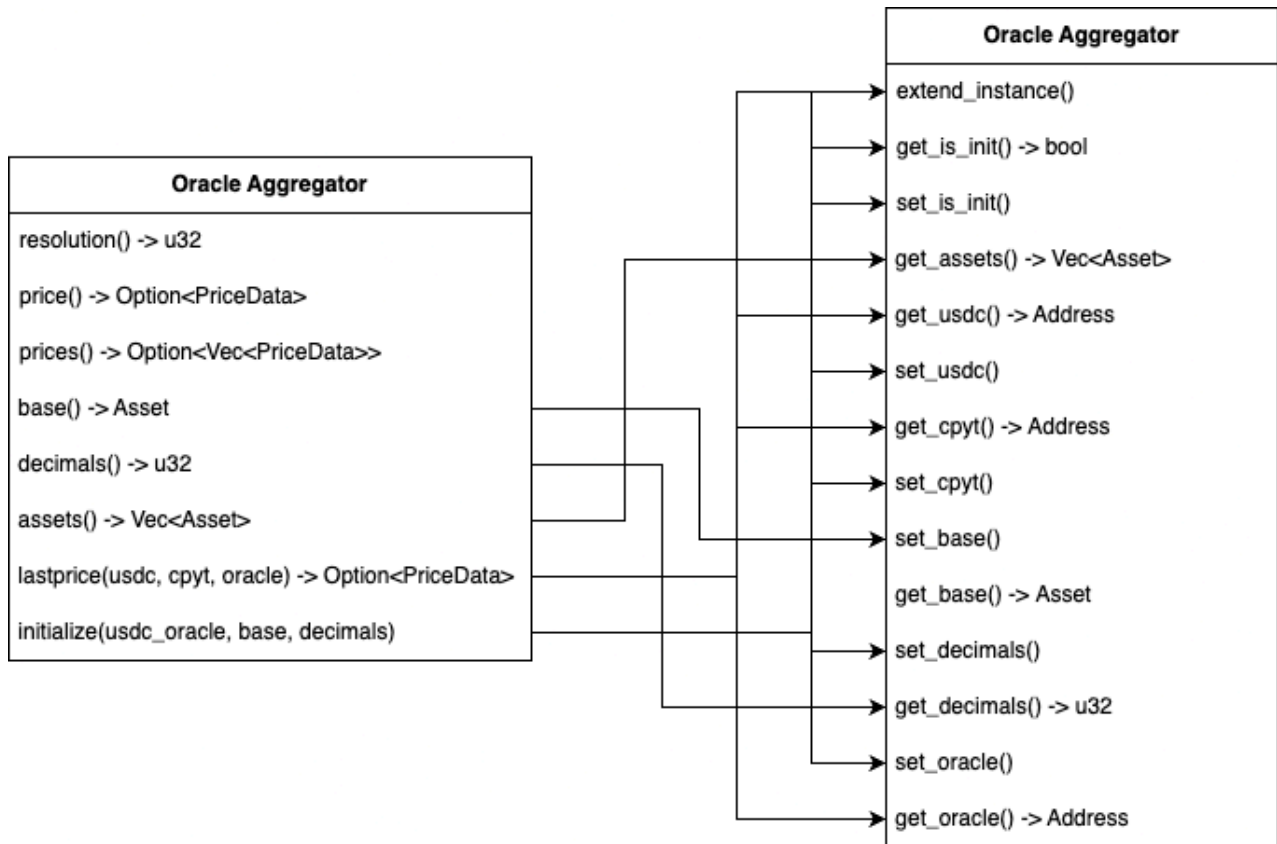


Figure 1: Oracle aggregator function operations and storage diagram.

Of the functions available in this contract, `resolution` , `price` , and `prices` panic if called.

`base` , `decimals` , and `assets` return the storage values for the respective variables these functions represent. `base` and `decimals` push the persistent storage variables' time-to-live, while `assets` returns a vector containing the addresses of USDC and CPCT.

`initialize` is supposed to be called after the contract's deployment by its deployer. From its parameters, it sets the addresses of USDC, CPCT, and the off-chain USDC oracle address and, from the latter, instantiates a `PriceFeed` contract to fetch its price and decimals, setting its own variables of the same name.

`last_price` is called when requesting the latest exchange rate of an asset supported by the oracle. In this case, the only assets supported are USDC and CPCT. Given that CPCT is pegged to the price of USDC, the returned value by `last_price` is the return of a call for the off-chain USDC oracle's `last_price` function.



Despite having a design of an oracle aggregator, it is important to highlight that ClickPesa's oracle aggregator is hardcoded to support only one oracle, which is off-chain USDC. This is done intentionally, given the purpose of this contract.

A Blend liquidity pool for ClickPesa is deployed and operational at the moment. Following the onchain data of the oracle referenced by the aggregator, [the address of the oracle](#) contains source code matching the implementation of [reflector-network's oracle contract](#).



## Invariants

---

During the audit, invariants have been defined and used to guide part of our search for possible issues with ClickPesa's oracle aggregator. With the help of the client's documentation, intended business logic, and references collected during the audit, the following invariants were gathered:

- There should not be administrative representatives to the oracle;
- Once initiated, no storage value of the contract will be overridden;
- There should only be one source of values for this oracle;
- Only two assets are to be supported by this oracle: USDC, and CPCT;

While being the guiding points of the contract's security analysis, aspects outside of the scope covered by the above invariants were investigated. Also, given the reliance of ClickPesa's oracle aggregator on the USDC oracle, data availability and reliability were not a primary focus of the investigation, although were considered as well during this engagement.



## Findings

---

Findings presented in this section are issues that can cause the system to fail, malfunction, and/or be exploited, and should be properly addressed.

All findings have a severity level and an execution difficulty level, ranging from low to high, as well as categories in which it fits. For more information about the classifications of the findings, refer to our [Smart Contract Analysis](#) page (adaptations performed where applicable).



# [A1] The Oracle Aggregator Can Be Hijacked By Third-Parties

Severity: High

Difficulty: Medium

Recommended Action: Fix Design

Not addressed by client

## Description

At the time of writing, Stellar-Soroban does not allow constructors into their contracts, meaning that the deployment and initialization function calls will be submitted to the blockchain as two different transactions.

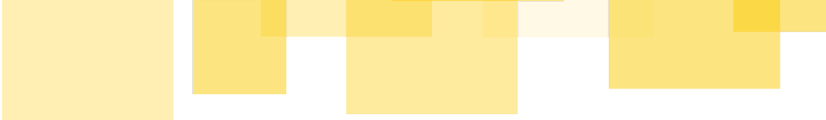
This can be particularly dangerous if a malicious user becomes aware of the activities of an account address used by the protocol admin(s), and attempts to intercept the initialization transaction after the contract deployment. By controlling the contract initialization, the malicious user is able to provide any address as the USDC oracle address which ClickPesa's contract relies upon. This will result in calls to the `last_price` function potentially executing malicious code if the user replicates the interface of the USDC oracle.

Given that ClickPesa already has a pool in the Blend protocol which makes usage of a contract with the same interface as the one being audited, which, in turn, points to the USDC oracle intended to be used by the audited contract, potential attackers already have all the necessary information to execute this attack, once ClickPesa's oracle aggregator is deployed.

## Scenario

An attack by a malicious 3rd party can be, for instance:

- 1 - ClickPesa admin deploys an oracle aggregator contract for testing and initializes it (two consecutive transactions);
- 2 - Malicious user collects information about the contract and its initialization;
- 3 - Malicious user starts to track activities of admin addresses on mainnet;
- 4 - ClickPesa admin deploys the oracle aggregator in mainnet;
- 5 - Malicious user front runs admin calling the initialization function, passing a personalized contract with the same interface of the oracle supported by the contract in scope;



6 - The ClickPesa is now tied to a deployed oracle pointing to malicious code. Given its interface, there is no way to modify the oracle to which this aggregator contract points.

Another possibility is that the admins may not link the oracle to the liquidity pools that they are designed to be used, or may not publicize the address of their newly deployed oracle if they notice that someone else initialized it in their place; this may lead to a denial of service/griefing attack by the malicious user, where any deployment attempt coming from a known admin address will be followed by an attempt of front running by the malicious user.

---

## Recommendation

We highly recommend that the deployment and initialization of all contracts that require initialization happen programmatically through a deployer smart contract (<https://soroban.stellar.org/docs/tutorials/deployer>). This way, as we wait for constructors to be introduced to Soroban, ClickPesa's team will have a way to deploy and initialize its oracle and prevent scenarios similar to the above from happening.

---

## Status

The client has acknowledged this finding, which will be addressed in a newer version of the audited contracts that are currently in development.



# [A2] The Oracle Aggregator Performs Calls to Contracts That Can Be Updated Whilst Itself Cannot

Severity: High

Difficulty: High

Recommended Action: Fix Design

Not addressed by client

## Description

Following the nature of Oracle Aggregator according to its intended design, each call requesting the price of either USDC or CPCT will be forwarded to another oracle. Based on the design analysis, provided documentation, and audit research, we believe that the implementation of ClickPesa's Oracle Aggregator is not ideal when considering its integration with an oracle using [reflector-network's oracle code](#). Of this oracle implementation, we note that it has two functions in particular that we should highlight: `config` and `update_contract`.

`config` will update the administrative and operational variables of the oracle (including `admin`, `base`, `decimals`, and `resolution`), and `update_contract` will modify the contract code stored in that address.

ClickPesa's Oracle Aggregator has no way of modifying its own storage variables after initialization, meaning that any potential modification to the oracle that it points to will lead this aggregator to display incorrect information to its users potentially. Furthermore, an additional danger comes from the possibility of the oracle referenced by ClickPesa's aggregator being hacked, leading to callers of ClickPesa's oracle aggregator to execute malicious code.

## Recommendation

Support ways of modifying the oracle aggregator administrative variables, as well as updating the contract and freezing its operations in case of any manifesting issue with the oracle referenced by the audited contract.

## Status





The client has acknowledged this finding, which will be addressed in a newer version of the audited contracts that are currently in development.



## Informative Findings

---

The findings presented in this section do not necessarily represent any flaw in the code itself. However, they indicate areas where the code may need external support or deviate from best practices. We have also included information on potential code size reductions and remarks on the operational perspective of the contract.



## [B1] Best Practices and Notable Particularities

---

Severity: Informative

Recommended Action: Fix Code

Not addressed by client

---

### Description

Here are some notes on the protocol particularities, comments, and suggestions to improve the code or the business logic of the protocol in a best-practice sense. They do not in themselves present issues to the audited protocol but are advised to either be aware of or to be followed when possible, and may explain minor unexpected behaviors on the deployed project.

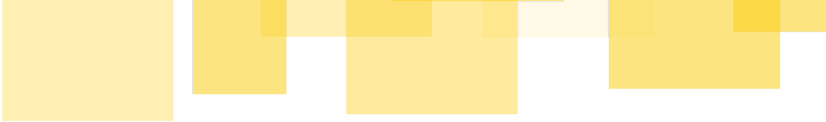
1. Soroban SDK version is outdated. The version used in the audited contract is 20.5.0;
2. In the initialization function, storage is modified, but no events announcing the storage modification are broadcasted;
3. There are `unwrap()` operations in five different points of `storage.rs`. This function, usually used for error handling, will call panic if an error occurs or if the value being unwrapped is None without a custom panic message. There is no appropriate handling of the potential panic this function can produce;
4. The documentation above initialize does not match the actual implementation of it;
5. Whenever storage is extended, it is extended by 31 days. The threshold is 30 days. This can be taxing to the user that ends up extending the time-to-live of the contract storage, and may lead to eventual contract archival due to the 1 day window in which the contract can have its storage time extended;

---

### Recommendations

For each of the topics elaborated above, we recommend implementing the following approaches into the protocol's contracts:

1. For security reasons, we should always use the latest Soroban SDK version (21.0.1);
2. Given that this is the single time the storage is modified and it is the contract initialization, this is not perceived it as an issue, but still stands as a best practice recommendation;
3. We recommend validating that the item retrieved from storage can actually be unwrapped, panicking if not, but with a proper informational error message;

- 
4. Documentation should always reflect the code it is used to describe;
  5. A lower threshold would be best to avoid the contract possibly expiring (for lack of use within a single day) or a single user being taxed with a higher-cost transaction to restore the full storage time. Perhaps explore using a 10 or 15 days threshold.
- 

### **Status**

The client has acknowledged this finding, which will be addressed in a newer version of the audited contracts that are currently in development.



## [B2] ClickPesa's Oracle Aggregator Reliance on USDC Price

---

Severity: Informative

Recommended Action: Fix Design

Not addressed by client

---

### Description

As discussed further in the [Platform Logic and Features Description](#), the oracle aggregator is designed to always provide the same price as CPCT and USDC. This is hard-coded into the contract code; in other words, the operation of fetching prices for this asset is not parametric, modifiable, or, in this specific case, different, regardless of which asset price is requested.

This implementation decision may have considerable impacts in case of a malfunction in the business logic of any of the two tokens. For instance, if USDC depegs, the price of CPCT will be modified accordingly, or if, for any reason, the pools that mint CPCT become insolvent and this causes a modification in CPCT's market price, users can acquire this token to exchange for USDC in any protocol reliant on ClickPesa's oracle aggregator, thus acquiring USDC as a discount price.

---

### Recommendation

Approaches to managing issues related to the above are not trivial, considering the design choice of pegging CPCT's price to USDC. Still, in case such issues happen, we recommend having approaches to halt the operation of the oracle to prevent the propagation of problems originating from any potential price differences between the tokens (as suggested in finding [\[A2\] The Oracle Aggregator Performs Calls to Contracts That Can Be Updated Whilst Itself Cannot](#)).

---

### Status

The client has acknowledged this finding, which will be addressed in a newer version of the audited contracts that are currently in development.