Crypton (&Studio)

PV01

Smart Contract Audit

Prepared by: CryptonStudio

Date of Engagement: 28.11.2024

Version history

Version	Name	Date
0.1	First draft	4.12.2024
0.2	Internal review	9.12.2024
1.0	Recommendation plan	12.12.2024
1.1	Recommendation execution review	16.12.2024

Executive overview

The security assessment was scoped for the smart contracts of PV01. At the time of the audit, all source files were located at the link.

The team at CryptonStudio was provided 3 days for the engagement and assigned one full time security engineer to audit the security of the smart contracts. The security engineers are blockchain and smart contract security experts, with experience in advanced penetration testing, smart contract hacking, and have a deep knowledge in multiple blockchain protocols.

The purpose of this audit to achieve the following:

- Ensure the smart contracts' functions are intended.

- Identify potential security issues with the smart contracts.

In summary, CryptonStudio identified few security risks, and recommends performing further testing to validate extended safety and correctness in context to the whole set of contracts.

Vulnerabilities or issues observed by CryptonStudio are ranked based on the risk assessment methodology by measuring the likelihood of a security incident, and the impact should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. For every vulnerability, a risk level will be calculated on a scale of 1 to 5 with 5 being the highest likelihood or impact.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

Scope

CONTRACTS

- OF - PV01OracleFactory.sol
- POV - PV01PriceOracleV1.sol
- POV V2 - PV01PriceOracleV2.sol

Codebase
https://github.com/pv01-org/blockchain/

Commit
D54D4304755BC819E56DDFA04D91408B825FF36F

Description	Solidity Source	Address Deployed Ethereum Mainnet
Proxy Implementation	PV01PriceOracleV1.sol	0x73dd0dc97175985e372d1335c20102d3ad45b65b
Proxy Implementation V2	PV01PriceOracleV2.sol	0x94686A7b5014aa865e8FA24d2A9Ab2A291daA7b1
Proxy Deployed	PV01BondPerpetualVault.sol	0x1C5fF3ab750778ECB7Dfe0A8D388023270Dded9b
Oracle Factory	PV01OracleFactory.sol	0x6e3e8cc6e4602d37843663722f512666f8d7e88d

SYSTEM OVERVIEW

Privileged Functions

In the contracts

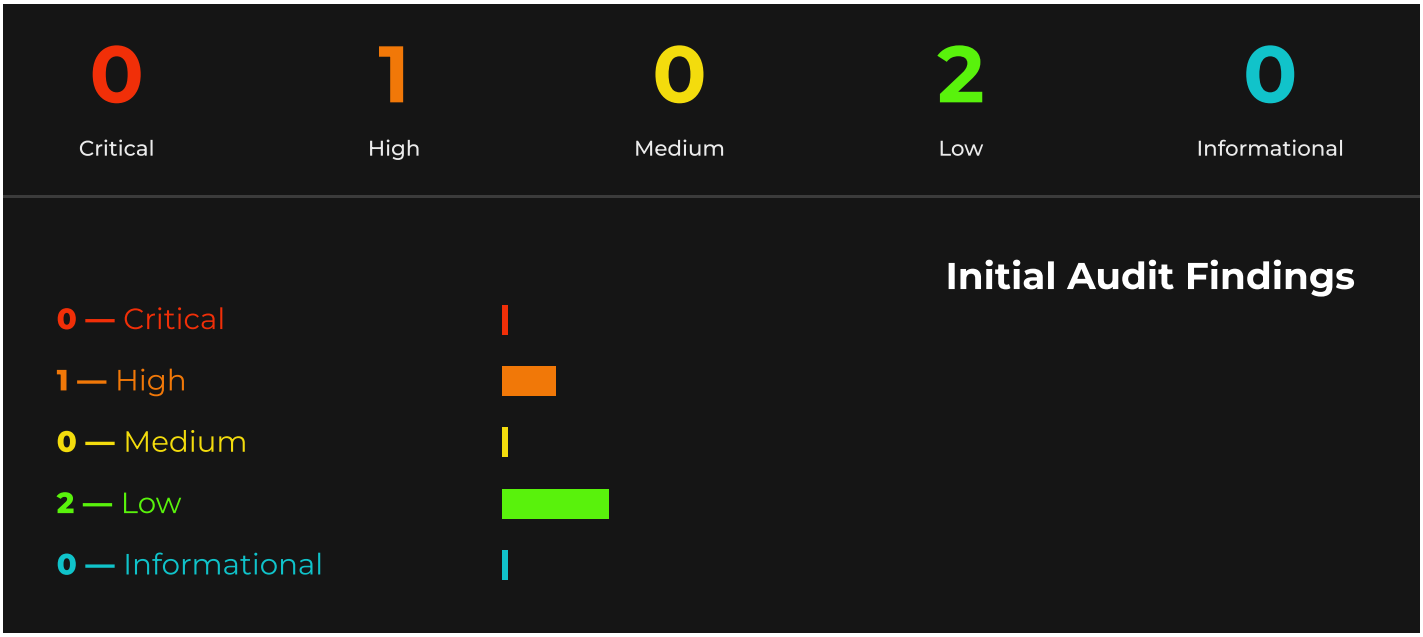
- PV01PriceOracleV1.sol
- PV01PriceOracleV2.sol
- PV01OracleFactory.sol

The “owner” role has access to privileged functions. According to the documentation, multisignature smart contracts are used to mitigate risks.

In **PV01OracleFactory,** the owner is able to create Oracles and control their versions using createOracle(), changeOracleImpl() and setImplAddress() functions

In **PV01PriceOracleV1**, the owner is able to set yield for bond, which directly affects the calculations of the oracle.

Assessment summary & findings overview



All issues were addressed either by being fixed or because they were by design. **There are no open issues.**

Issue	Severity	Status	Comment from developer
POV-01	High	Fixed	Fixed
POV-02	Low	Acknowledged	This recommendation could cause issues and won't be implemented. This function is intended to protect against accidental typos when entering a bond yield value. It is not intended to provide another layer of security protection, because Fireblocks already takes care of multi-signatures and approvals.
OF-01	Low	Acknowledged	The upgradeability pattern used is by design. Our existing Fireblocks multi-signature and approvals process handles the security side. Adding another layer of security with a time delay would prevent us from responding quickly to faults. Since we're comfortable with the Fireblocks security measures, we choose not to implement this recommendation.

FINDINGS & TECH DETAILS

POV-01

Severity — High

Status — Fixed

Description

The calculation:

bondPrice = 1e18 - bondYieldDetail.yield.mulDiv(secondsToMaturity, 365 days);

does not check for overflow or underflow conditions.

If bondYieldDetail.yield is too large or *secondsToMaturity* is significant, the product of *yield * secondsToMaturity / 365* can exceed 1e18.

It might happen if **secondsToMaturity** is much higher than 365 days or yield was set way too high by a mistake.

This leads to an overflow during the subtraction, causing the function to revert. In cases of high yields or long maturities, this issue could prevent price computation for bonds, limiting functionality.

Recommendation

Validate input data for yield. Check that the calculation result is less than 1e18.

Alleviation

Fixed

POV-02

Severity — Low

Status — Acknowledged

Description

The *setBondYieldMaxDelta()* function allows the owner to change the yield delta of the smart contract at any time without proper validation. Since the Oracle smart contract is used to calculate prices for users assets, the update process must be clear and safe.

Recommendation

While multisignature smart contracts are already in use, it is recommended to add a delay for this action so that users can familiarize themselves with the update and validate min/max for numeric values.

Alleviation

This recommendation could cause issues and won't be implemented.

This function is intended to protect against accidental typos when entering a bond yield value. It is not intended to provide another layer of security protection, because Fireblocks already takes care of multi-signatures and approvals.

In fact, being able to change this value needs to be a fast operation, and introducing a time delay could cause issues. In rare situations, we might need to react quickly to market events, and change a bond's yield by a value larger than this "max delta" default of 50bps (which is intentionally kept low). In these situations we may need to change the max delta value quickly, and actively don't want a time delay.

OF-01

Severity — Low

Status — Acknowledged

Description

The *changeOracleImpl()* function allows the owner to change the logic of the smart contract at any time. If the Oracle smart contract is updated to return incorrect or manipulated price data, dependent contracts could execute transactions based on incorrect information, leading to systemic issues.

Recommendation

While multisignature smart contracts are already in use, it is recommended to add a delay for this action so that users can familiarize themselves with the update.

Alleviation

The upgradeability pattern used is by design.

Our existing Fireblocks multi-signature and approvals process handles the security side. Adding another layer of security with a time delay would prevent us from responding quickly to faults. Since we're comfortable with the Fireblocks security measures, we choose not to implement this recommendation.

Contacts

crypton.studio

info@crypton.studio

+371 261 19 169

OUR MEDIA

Scan the QR code to open our sources