Smart Contract Code Review

4th June 2021

Audited by – VG & AHR (OF-10) The_SwarmVersion: 1.0

Table of Content

Executive Summary

The purpose of this report

Smart Contract Code Review

This document aims to record the vulnerabilities found from a code review conducted by UBIK Group. The detected vulnerabilities are plotted against Best Practice Guidelines laid downby the community.

Execution Strategy

Our execution strategy incorporates proven methodologies, extremely qualified personnel, and ahighly responsive approach to managing deliverables and the utilization of proprietary software.

Methodology

The code audit was carried out using the specification of SWC (Smart Contract Weakness Classification) and CWE (Common Weakness Enumeration). The assessment was conducted using acombination of proprietary software and manual testing by highly skilled individuals.

Vulnerability Overview

Timeline and Audit Log

The Security Code Audit of Smart Contract of Synthereum Contract started on 27th April 2021 and ended on 25th May 2021. Where in total of 8 contracts were audited.

The following contracts were audited:

- *PoolOnChainPriceFeed.sol*
- *PoolOnChainPriceFeedCreator.sol*
- *PoolOnChainPriceFeedFactory.sol*
- *PoolOnChainPriceFeedLib.sol*
- *IPoolOnChainPriceFeed.sol*
- *IPoolOnChainPriceFeedStorage.sol*

Vulnerabilities Detected

There was a total of 3 vulnerabilities identified in the contracts and below mentioned chart shows its respective distribution

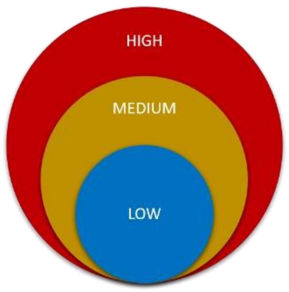


Fig 1. Vulnerability Classification

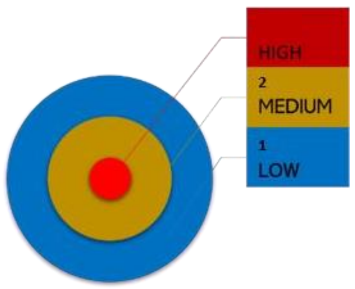


Fig 2. Vulnerability Breakdown in Numbers

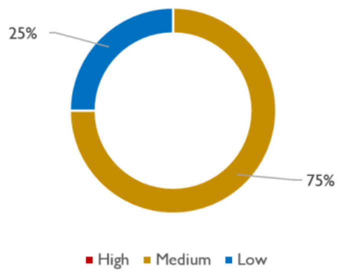


Fig 3. Vulnerability Distribution in %

HIGH RISK - A total of 0 classified as high risk vulnerabilities detectedContract Files Affected -

MEDIUM RISK - A total of 2 classified as medium risk vulnerabilities detected

Contract Files Affected - *PoolOnChainPriceFeed.sol, PoolOnChainPriceFeedCreator.sol, PoolOnChainPriceFeedFactory.sol, PoolOnChainPriceFeedLib.sol, IPoolOnChainPriceFeed.sol, IPoolOnChainPriceFeedStorage.sol*

CWE References

CWE-477 - Incorrect function "checkParams" state mutability

CWE-477 - Incorrect function "getRoleMemberCount" state mutability

LOW RISK - A total of 1 classified as low risk vulnerabilities detected

Contract Files Affected - *PoolOnChainPriceFeed.sol, PoolOnChainPriceFeedCreator.sol, PoolOnChainPriceFeedFactory.sol, PoolOnChainPriceFeedLib.sol, IPoolOnChainPriceFeed.sol, IPoolOnChainPriceFeedStorage.sol*

CWE References

CWE-447 - Use of "tx.origin" as a part of authorization control

Exploit Effort & Resource Classification

Rating	Definition of Risk Rating	Definition of Resource Requirement to Exploit	Definition of Effort to Exploit
HIGH	Deficiency creates a vulnerability that could result in loss of system control or override a desired function or give access to critical or sensitive information.	Recommendation either requires the purchase of hardware or, requires significant research and resources to exploit	To exploit the weakness requires a high level of expertise and advanced knowledge of smart contract design, and programming
MEDIUM	Deficiency creates an exposure to a larger, but limited loss of confidentiality or integrity, as the result of many user accounts being compromised, or restricted functions being accessed.	Recommendation may require the purchase of hardware or software and/or requires moderate, research and implementation activities to exploit	Requires medium level of effort. No tools are available but sample code or other similar exploits are known
LOW	Deficiency creates limited exposure to the compromise of user accounts or unauthorized access to data	Recommendation may require the purchase of minor hardware or software and/or requires minor research and implementation activities to exploit	Easy to exploit with known methods or tools with minimal modifications

Exploit Efforts & Resource Analysis

The following graphs below provide insight into the exploit efforts and resources needed in order to successfully complete or carry out exploitation mapped against the 3 vulnerabilities detected

Exploit Effort

Of the 3 Security issues identified, 3 vulnerabilities would require a low level to no resources to exploit.

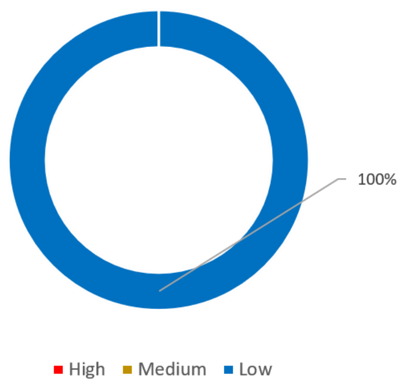


Fig 4. Exploit Effort Breakdown in %

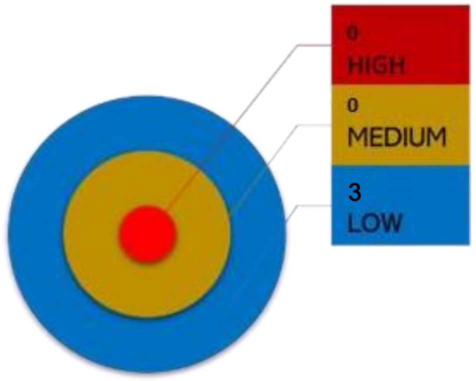


Fig 5. Exploit Effort Breakdown in numbers

Exploit Resource Requirements

Of the 3 Security issues identified, 3 vulnerabilities that can be exploited require low to no resources to exploit.

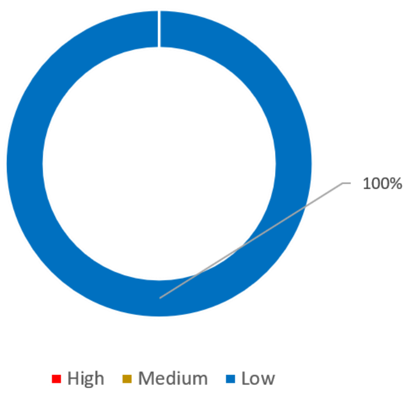


Fig 6. Resources need to exploit in %

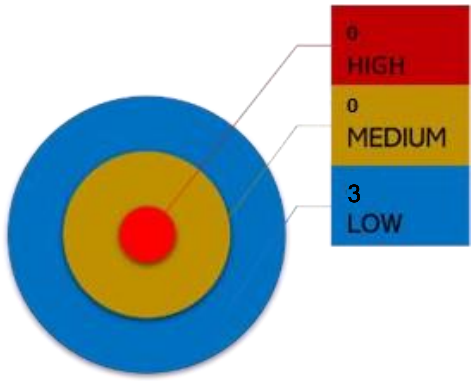


Fig 7. Resources needed to exploit in numbers

Remediation Resource Requirements

Rating	Definition of Risk Rating	Definition of Resource Requirement to Remediate	Definition of Effort to Remediate
HIGH	Deficiency creates a vulnerability that could result in loss of system control or override a desired function or give access to critical or sensitive information.	Recommendation either requires the purchase of hardware or, requires significant changes to the code base or research and resources to remediate	To remediate the vulnerabilities requires a high level of expertise and advanced knowledge of smart contract design, and programming
MEDIUM	Deficiency creates an exposure to a larger, but limited loss of confidentiality or integrity, as the result of many user accounts being compromised, or restricted functions being accessed.	Recommendation may require the purchase of hardware or software and/or requires moderate changes to the codebase and/or research and implementation activities to remediate the vulnerability	Requires medium level of effort and changes to remediate.
LOW	Deficiency creates limited exposure to the compromise of user accounts or unauthorized access to data	Recommendation may require the purchase of minor hardware or software and/or requires minor changes in the codebase to remediate against the vulnerability	Easy to remediate with minimal modification or effort

Remediation Resource Requirements

Of the 3 Security issues identified, remediation efforts and resources required in 3 circumstances are considered Low. Therefore, minimal resources and programming efforts are required to implement satisfactory remediation. However one severity, the high severity *Improper Following ofSpecification by Caller*, may require significant efforts to remediate

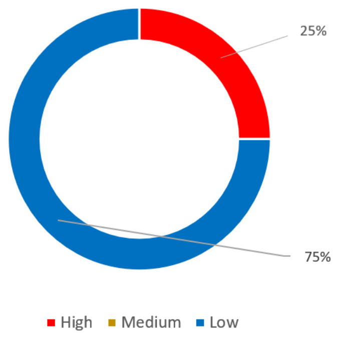


Fig 8. Resources need to remediate in %

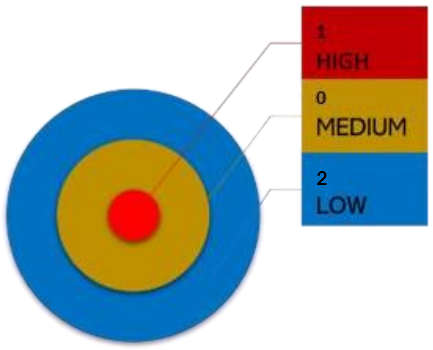
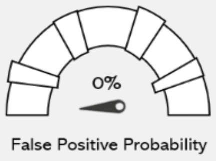


Fig 9. Resources needed to remediate in numbers

Finding Details

Severity MEDIUM

Category: Improper Coding Practice



100%
True Positive Probability

Contract Name/s

List of Contracts Affected

- *PoolOnChainPriceFeed.sol*
- *PoolOnChainPriceFeedCreator.sol*
- *PoolOnChainPriceFeedFactory.sol*
- *PoolOnChainPriceFeedLib.sol*
- *IPoolOnChainPriceFeed.sol*
- *IPoolOnChainPriceFeedStorage.sol*

SWC Reference

SWC 111 Deprecated Solidity Functions

CWE Reference

CWE-829 Use of Obsolete Function

Description

The code uses deprecated or obsolete functions, which suggests that the code has not been actively reviewed or maintained.

Code Reference/s

function getRoleMemberCount(bytes32 role) public view returns (uint256) {
Line 43

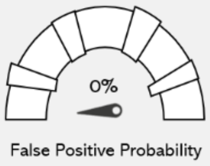
Remediation

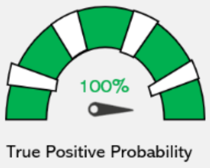
Solidity provides alternatives to the deprecated constructions. Most of them are aliases, thus replacing old constructions will not break current behavior.

Severity

MEDIUM

Category: Improper Coding Practice





Contract Name/s

List of Contracts Affected

- *PoolOnChainPriceFeed.sol*
- *PoolOnChainPriceFeedCreator.sol*
- *PoolOnChainPriceFeedFactory.sol*
- *PoolOnChainPriceFeedLib.sol*
- *IPoolOnChainPriceFeed.sol*
- *IPoolOnChainPriceFeedStorage.sol*

SWC Reference

SWC 111Deprecated Solidity Functions

CWE Reference

CWE-829 Use of Obsolete Function

Description

The code uses deprecated or obsolete functions, which suggests that the code has not been actively reviewed or maintained.

Code Reference/s

function checkParams(
Line 1126

Remediation

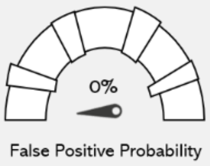
Solidity provides alternatives to the deprecated constructions. Most of them are aliases, thus replacing old constructions will not break current behavior.

Severity**LOW**

Category: Improper Coding Practice

Contract Name/s List of Contracts Affected

- *PoolOnChainPriceFeed.sol*
- *PoolOnChainPriceFeedCrea*



100%
True Positive Probability

SWC Reference

[SWC - 115](#) – Use of Tx.origin as per of authorization control

CWE Reference

[CWE-115](#) Use of Obsolete Functions

Description

tx.origin is a global variable in Solidity which returns the address of the account that sent the transaction. Using the variable for authorization could make a contract vulnerable if an authorized account calls into a malicious contract. A call could be made to the vulnerable contract that passes the authorization check since tx.origin returns the original sender of the transaction which in this case is the authorized account.

Code Reference/s

```
(tx.origin == msg.sender, 'Account must be an EOA');
Line 139
```

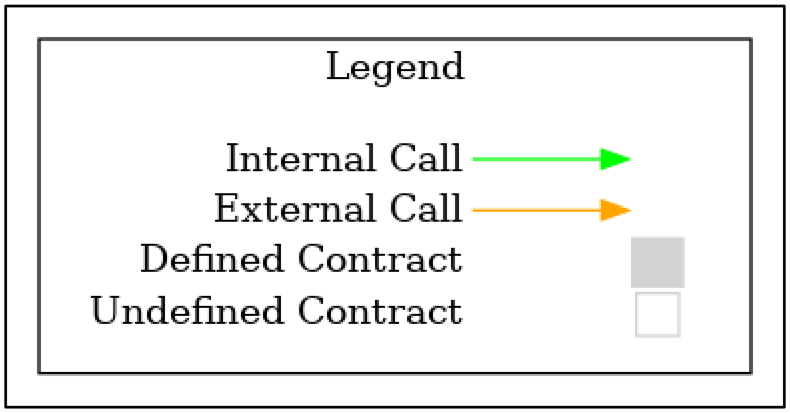
Remediation

Use `msg.sender` instead of `tx.origin`

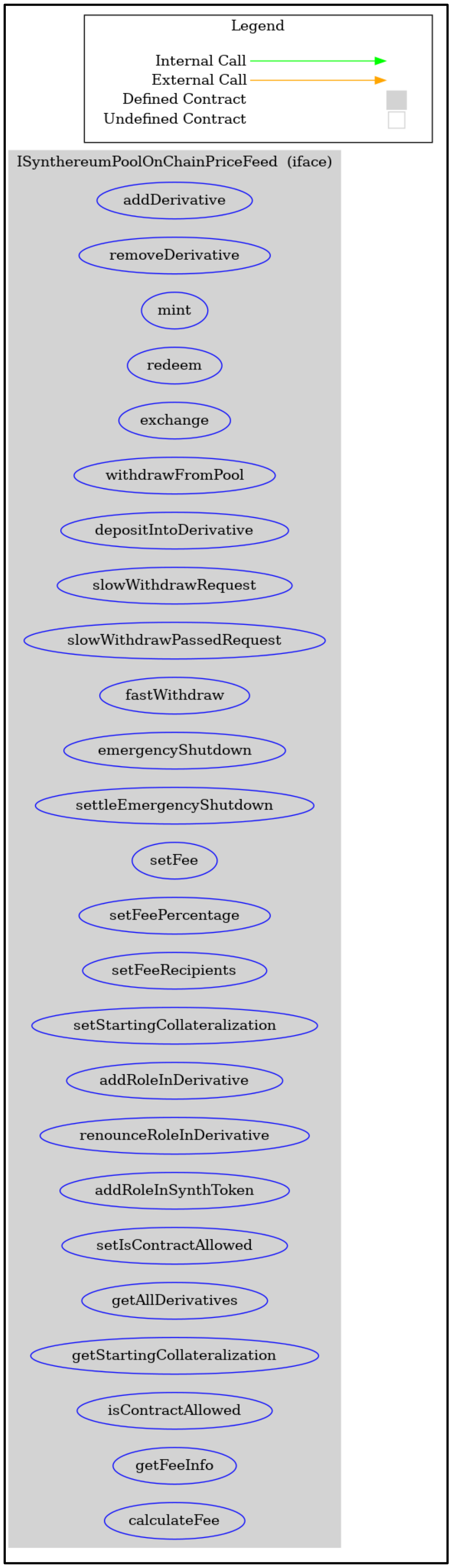
Ancillary Reports

SoloGraph

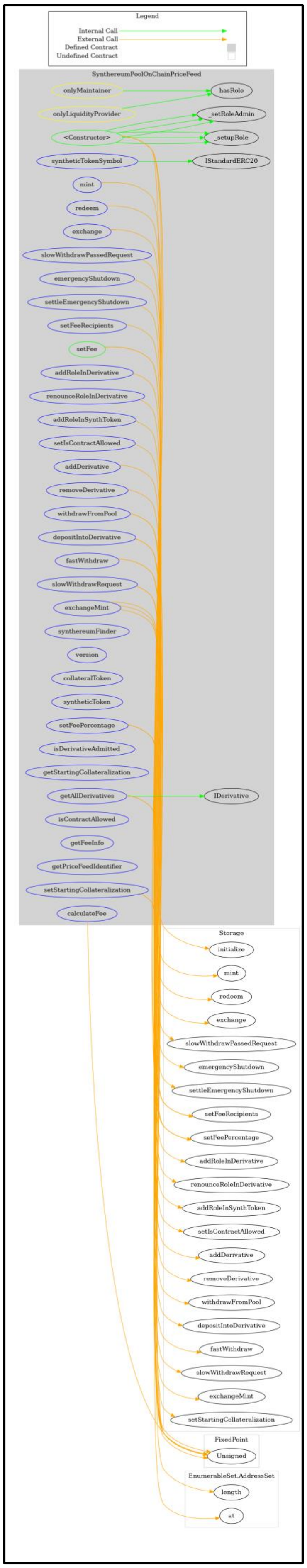
1. IPoolOnChainPriceFeedStorage.sol



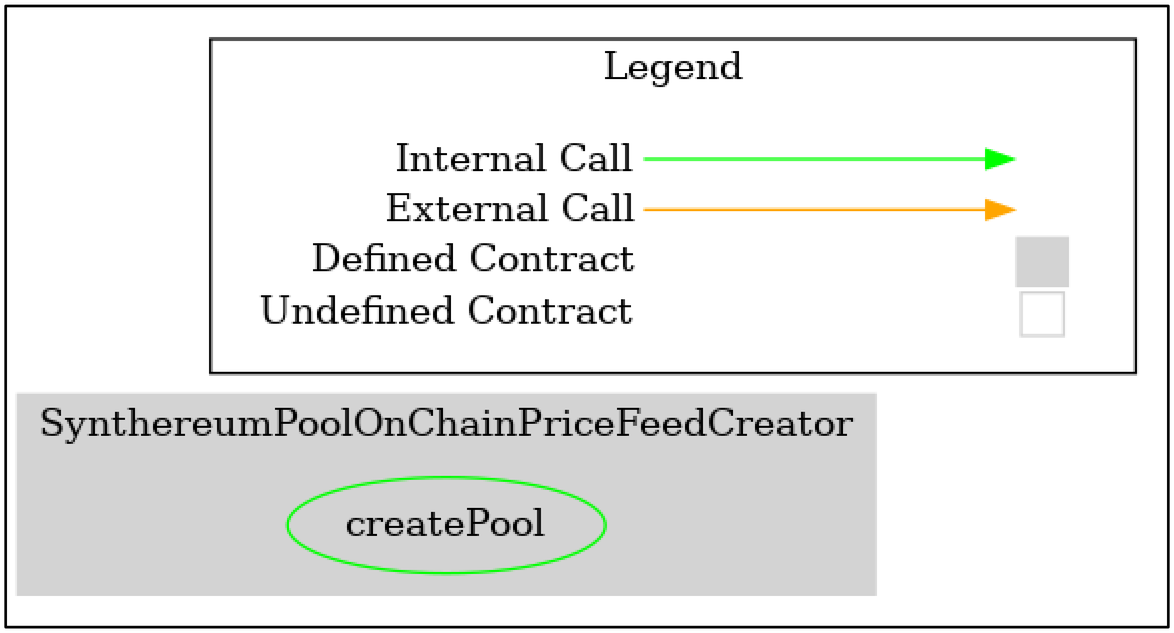
2.IPoolOnChainPriceFeed.sol



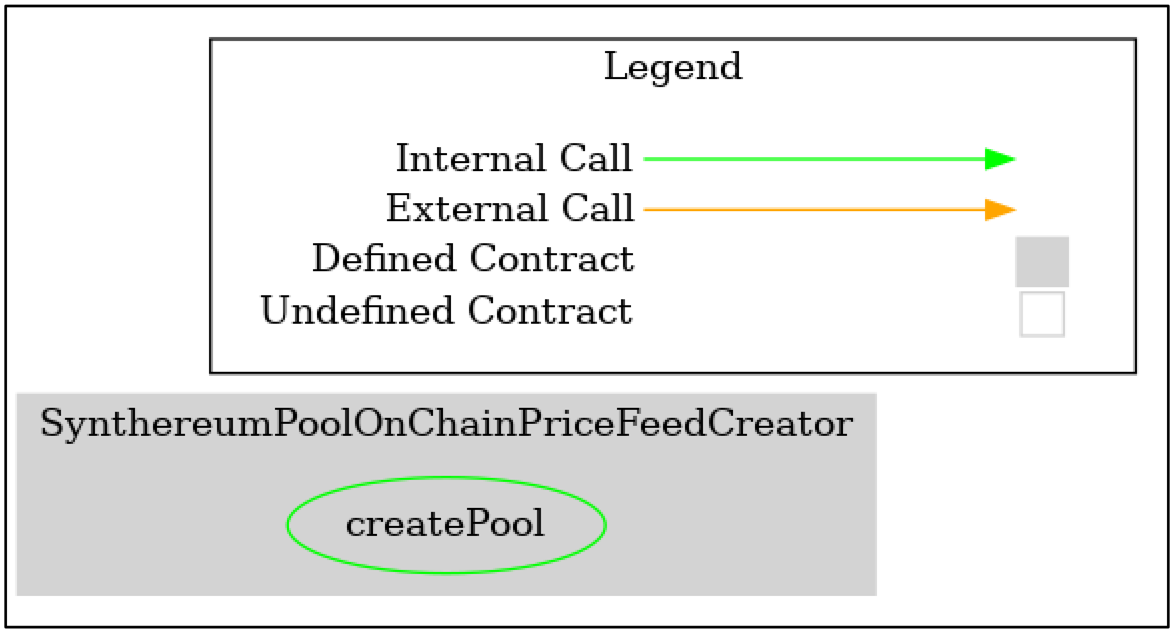
3.PoolOnChainPriceFeed.sol



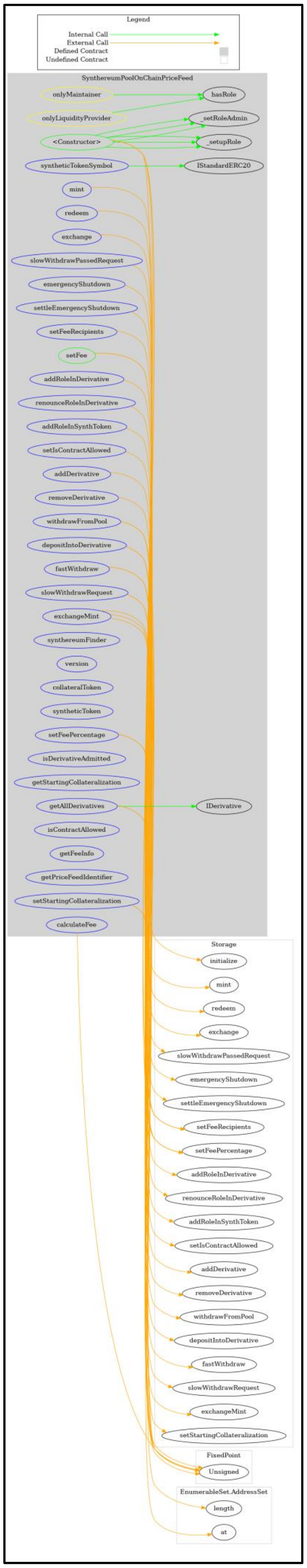
4. PoolOnChainPriceFeedCreator.sol



5.PoolOnChainPriceFeedFactory.sol



6. PoolOnChainPriceFeedLib.sol



Additional Notes

- Reports from Mythx are attached below

Conclusions

- The code is clean and of excellent standard and one of the best we have audited. The high severity noted above in this report is deemed as an architectural oversight. However, there is little to no resources needed to exploit this. It may prove to be something that can effects the security of the contracts and indeed credibility and may require heavy resources to remediate.

Open Cases

CASE	OPEN	INTITIAL REVIEW DATE	SECOND REVIEW DATE
HIGH RISK - A total of 0 classified as high risk vulnerabilities detected Contract Files Affected -	NO	04/06/2021	TBA
MEDIUM RISK - A total of 2 classified as medium risk vulnerabilities detected Contract Files Affected - *PoolOnChainPriceFeed.sol, PoolOnChainPriceFeedCreator.sol, PoolOnChainPriceFeedFactory.sol, PoolOnChainPriceFeedLib.sol, IPoolOnChainPriceFeed.sol, IPoolOnChainPriceFeedStorage.sol* CWE References CWE-477 - Incorrect function "checkParams" state mutability CWE-477 - Incorrect function "getRoleMemberCount" state mutability	Yes	04/06/2021	TBA
LOW RISK - A total of 1 classified as low risk vulnerabilities detected Contract Files Affected - *PoolOnChainPriceFeed.sol, PoolOnChainPriceFeedCreator.sol, PoolOnChainPriceFeedFactory.sol, PoolOnChainPriceFeedLib.sol, IPoolOnChainPriceFeed.sol, IPoolOnChainPriceFeedStorage.sol* CWE References CWE-447 - Use of "tx.origin" as a part of authorization control	Yes	04/06/2021	TBA

DISCLAIMER

[CLIENT]: Synthereum

V1: Original Report without remediation [ORIGINALTESTDATE]: 04/06/2021

V2: Remediation Report [REMEDIATIONTESTDATE]: TBA

CURRENT VERSION AS OF 04/05/2021: V1

V1 This review is marked as V.1, which was conducted by Ubik's certified security engineers. We identified several security vulnerabilities and provided remediation advice to Synthereum. As of yet there has been no set remediation report date.

V2 IF CORRECTED: After being notified by Synthereum that all vulnerabilities have been corrected, Ubik Group will perform a remediation test (V.2) on [REMEDIATIONTESTDATE] to confirm that all vulnerabilities and issues identified were either corrected or had been adequately addressed through other controls.

While no application or system can be 100% secure, all of our security findings were corrected or addressed and it is our opinion that the contracts tested are reasonably well written from a security perspective and the applications and supporting systems are deployed, configured and implemented in a secure manner. IF NOT FULLY CORRECTED The review was conducted by Ubik's certified security engineers. We identified several security vulnerabilities and provided remediation advice to Synthereum.

After being notified by Synthereum that these selected vulnerabilities had been corrected, Ubik Group performed a remediation test on [REMEDIATIONTESTDATE] and confirmed that these selected vulnerabilities were either corrected or had been adequately addressed through other controls. There were findings identified by Ubik Group that were not validated as corrected. Please contact Synthereum for further information regarding these findings and their resolution status. DISCLAIMER: Ubik Group conducted this testing on the smart contracts that existed as of April 27th, 2021. Information security threats are continually changing, with new vulnerabilities discovered on a daily basis, and no application can ever be 100% secure no matter how much security testing is conducted. This report is intended only to provide documentation that Synthereum has corrected all findings noted by Ubik Group as of [REMEDIATIONTESTDATE].

This report cannot and does not protect against personal or business loss as the result of use of the applications or systems described. Ubik Group offers no warranties, representations or legal certifications concerning the applications, code or systems it tests. All software includes defects: nothing in this document is intended to represent or warrant that security testing was complete and without error, nor does this document represent or warrant that the application tested is suitable to task, free of other defects than reported, fully

DISCLAIMER - Compliant with any industry standards, or fully compatible with any operating system, hardware, or other application. By using this information you agree that Ubik Group shall be held harmless in any event.

MythX

REPORT 60AE28F84BBCE500198922F0

Created	Wed May 26 2021 10:54:48 GMT+0000 (Coordinated Universal Time)
Number of analyses	4
User	6082eefae7c9265b53011cce

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
9f400dba-2112-4924-93cf-2fbd40d16bf9	contracts/synthereum-pool/v3/PoolOnChainPriceFeedCreator.sol	4
04cc3b72-4bdb-4afb-bdf0-c4a6d0497dc7	contracts/synthereum-pool/v3/PoolOnChainPriceFeedFactory.sol	4
7c4f734e-7e67-431e-944f-eb0540ce0e45	contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol	3
119f183a-b309-4986-84be-5729190b73c0	contracts/synthereum-pool/v3/PoolOnChainPriceFeed.sol	4

Analysis 9f400dba-2112-4924-93cf-2fbd40d16bf9 MythX

Started	Wed May 26 2021 10:55:21 GMT+0000 (Coordinated Universal Time)
Finished	Wed May 26 2021 10:57:06 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Synthereum-Pool/V3/PoolOnChainPriceFeedCreator.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	2	2

ISSUES

MEDIUM

SWC-000

Incorrect function "getRoleMemberCount" state mutability

Function "getRoleMemberCount" state mutability is considered "view" by compiler, but should be set to non-payable (default).

Source file

@openzeppelin/contracts/access/AccessControl.sol

Locations

```
}

function getRoleMemberCount(bytes32 role) public view returns (uint256) {
return _roles[role].members.length();
}

function getRoleMember(bytes32 role, uint256 index)
```

MEDIUM

SWC-000

Incorrect function "checkParams" state mutability

Function "checkParams" state mutability is considered "view" by compiler, but should be set to non-payable (default).

Source file

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol`

Locations

```
* @param expiration Expiration time of the transaction
*/
function checkParams(
ISynthereumPoolOnChainPriceFeedStorage.Storage storage self,
IDerivative derivative,
uint256 feePercentage,
uint256 expiration
) internal view checkDerivative(self, derivative) {
require(now <= expiration, 'Transaction expired');
require(
self.fee.feePercentage.rawValue <= feePercentage,
'User fee percentage less than actual one'
);
}

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.12"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedCreator.sol`

Locations

```
// SPDX-License-Identifier: AGPL-3.0-only
pragma solidity ^0.6.12;
pragma experimental ABIEncoderV2;
```

LOW

SWC-115

Use of "tx.origin" as a part of authorization control.

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source file

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol`

Locations

```
) {
if (!self.isContractAllowed) {
require(tx.origin == msg.sender, 'Account must be an EOA');
}
_;
```

Analysis 04cc3b72-4bdb-4afb-bdf0-c4a6d0497dc7 MythX

Started	Wed May 26 2021 10:55:21 GMT+0000 (Coordinated Universal Time)
Finished	Wed May 26 2021 10:57:28 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Synthereum-Pool/V3/PoolOnChainPriceFeedFactory.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	2	2

ISSUES

MEDIUM

SWC-000

Incorrect function "getRoleMemberCount" state mutability

Function "getRoleMemberCount" state mutability is considered "view" by compiler, but should be set to non-payable (default).

Source file

@openzeppelin/contracts/access/AccessControl.sol

Locations

```
}

function getRoleMemberCount(bytes32 role) public view returns (uint256) {
return _roles[role].members.length();
}

function getRoleMember(bytes32 role, uint256 index)
```

MEDIUM

SWC-000

Incorrect function "checkParams" state mutability

Function "checkParams" state mutability is considered "view" by compiler, but should be set to non-payable (default).

Source file

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol`

Locations

```
* @param expiration Expiration time of the transaction
*/
function checkParams(
ISynthereumPoolOnChainPriceFeedStorage.Storage storage self,
IDerivative derivative,
uint256 feePercentage,
uint256 expiration
) internal view checkDerivative(self, derivative) {
require(now <= expiration, 'Transaction expired');
require(
self.fee.feePercentage.rawValue <= feePercentage,
'User fee percentage less than actual one'
);
}

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.12"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedFactory.sol`

Locations

```
// SPDX-License-Identifier: AGPL-3.0-only
pragma solidity ^0.6.12;
pragma experimental ABIEncoderV2;
import {IDerivative} from '../../derivative/common/interfaces/IDerivative.sol';
```

LOW

SWC-115

Use of "tx.origin" as a part of authorization control.

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source file

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol`

Locations

```
) {
if (!self.isContractAllowed) {
require(tx.origin == msg.sender, 'Account must be an EOA');
}
_;
```

Analysis 7c4f734e-7e67-431e-944f-eb0540ce0e45

MythX

Started	Wed May 26 2021 10:55:31 GMT+0000 (Coordinated Universal Time)
Finished	Wed May 26 2021 10:57:27 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Synthereum-Pool/V3/PoolOnChainPriceFeedLib.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	1	2

ISSUES

MEDIUM

SWC-000

Incorrect function "checkParams" state mutability

Function "checkParams" state mutability is considered "view" by compiler, but should be set to non-payable (default).

Source file

contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol

Locations

```
* @param expiration Expiration time of the transaction
*/
function checkParams(
ISynthereumPoolOnChainPriceFeedStorage.Storage storage self,
IDerivative derivative,
uint256 feePercentage,
uint256 expiration
) internal view checkDerivative(self, derivative) {
require(now <= expiration, 'Transaction expired');
require(
self.fee.feePercentage.rawValue <= feePercentage,
'User fee percentage less than actual one'
);
}

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.12"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol`

Locations

```
// SPDX-License-Identifier: AGPL-3.0-only
pragma solidity ^0.6.12;
pragma experimental ABIEncoderV2;
```

LOW

SWC-115

Use of "tx.origin" as a part of authorization control.

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source file

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol`

Locations

```
) {
if (!self.isContractAllowed) {
require(tx.origin == msg.sender, 'Account must be an EOA');
}
_;
```

Analysis 119f183a-b309-4986-84be-5729190b73c0 MythX

Started	Wed May 26 2021 10:55:31 GMT+0000 (Coordinated Universal Time)
Finished	Wed May 26 2021 10:57:42 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Synthereum-Pool/V3/PoolOnChainPriceFeed.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	2	2

ISSUES

MEDIUM

SWC-000

Incorrect function "getRoleMemberCount" state mutability

Function "getRoleMemberCount" state mutability is considered "view" by compiler, but should be set to non-payable (default).

Source file

@openzeppelin/contracts/access/AccessControl.sol

Locations

```
}

function getRoleMemberCount(bytes32 role) public view returns (uint256) {
return _roles[role].members.length();
}

function getRoleMember(bytes32 role, uint256 index)
```

MEDIUM

SWC-000

Incorrect function "checkParams" state mutability

Function "checkParams" state mutability is considered "view" by compiler, but should be set to non-payable (default).

Source file

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol`

Locations

```
* @param expiration Expiration time of the transaction
*/
function checkParams(
ISynthereumPoolOnChainPriceFeedStorage.Storage storage self,
IDerivative derivative,
uint256 feePercentage,
uint256 expiration
) internal view checkDerivative(self, derivative) {
require(now <= expiration, 'Transaction expired');
require(
self.fee.feePercentage.rawValue <= feePercentage,
'User fee percentage less than actual one'
);
}

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.12"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`contracts/synthereum-pool/v3/PoolOnChainPriceFeed.sol`

Locations

```
// SPDX-License-Identifier: AGPL-3.0-only
pragma solidity ^0.6.12;
pragma experimental ABIEncoderV2;
```

LOW

SWC-115

Use of "tx.origin" as a part of authorization control.

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source file

`contracts/synthereum-pool/v3/PoolOnChainPriceFeedLib.sol`

Locations

```
) {
if (!self.isContractAllowed) {
require(tx.origin == msg.sender, 'Account must be an EOA');
}
_;
```