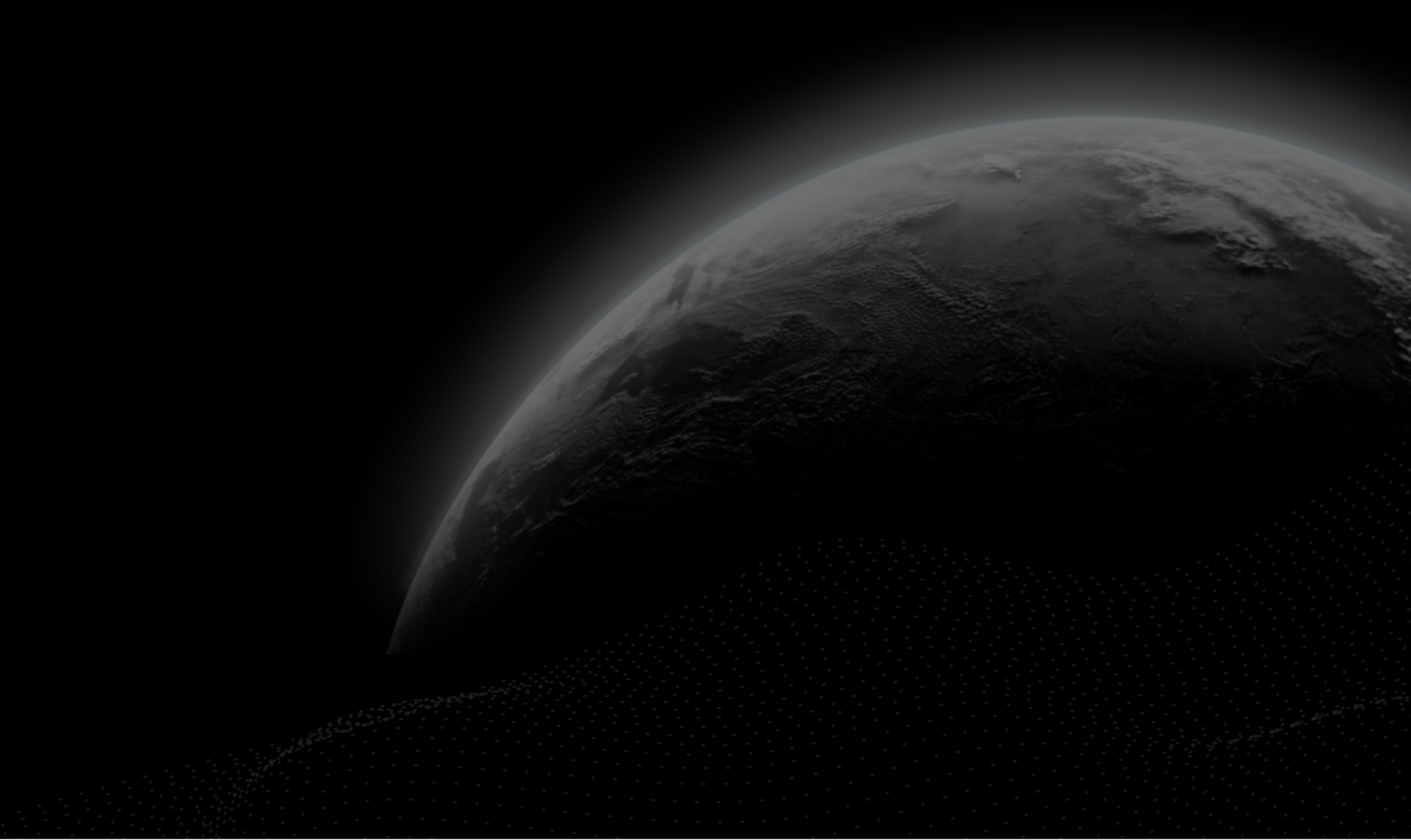
Security Assessment

Solv Protocol stUSD - Audit

CertiK Assessed on Dec 28th, 2023



CertiK Assessed on Dec 28th, 2023

Solv Protocol stUSD - Audit

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum (ETH)	Manual Review, Static Analysis

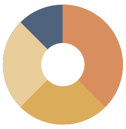
LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 12/28/2023	N/A

CODEBASE	COMMITS
https://github.com/solv-finance/stUSD View All in Codebase Page	7f71bbb7cde91e3d42fe13a13f07b6c15a9253bd View All in Codebase Page

Highlighted Centralization Risks

ⓘ Contract upgradeability

Vulnerability Summary



8	4	0	0	4	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

🟥	0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧	3	Major	1 Resolved, 2 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨	2	Medium	1 Resolved, 1 Acknowledged	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	2	Minor	2 Resolved	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦	1	Informational	1 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | SOLV PROTOCOL STUSD - AUDIT

CODEBASE | SOLV PROTOCOL STUSD - AUDIT

Repository

https://github.com/solv-finance/stUSD

Commit

7f71bbb7cde91e3d42fe13a13f07b6c15a9253bd

AUDIT SCOPE | SOLV PROTOCOL STUSD - AUDIT

6 files audited ● 6 files without findings

ID	Repo	File	SHA256 Checksum
● ACD	solv-finance/stUSD	contracts/access/AdminControl.sol	9022d187b8460f12fdfded1b7abe3f52e8e54041f89152a955e05ffe0e7f2a6b
● GCS	solv-finance/stUSD	contracts/access/GovernorControl.sol	98e1db1d20573ba9b59ef08340800dacd22a6d725394cfdbdbeac5b4e173f1ba
● ERC	solv-finance/stUSD	contracts/utils/ERC3525TransferHelper.sol	69b844ec68087448a531bccc885c4148ba2c33c216ea25779e215c4dcf0f55bd
● ISW	solv-finance/stUSD	contracts/ISftWrappedToken.sol	bc42c71ab6e9ce8150cdbd6fec8b1a576face47dfa76818ad9cdc53fa6caa024
● SWT	solv-finance/stUSD	contracts/SftWrappedToken.sol	2aae3da83575a8af01ecd210518c34d2514835c31925b88cbbce674094cb6300
● SWF	solv-finance/stUSD	contracts/SftWrappedTokenFactory.sol	1acb5c80ae653ba1cdca5b699b7f19319ab6ac87fe5ce407f6848143b0dbe5b3

APPROACH & METHODS | SOLV PROTOCOL STUSD - AUDIT

This report has been prepared for SOLV to discover issues and vulnerabilities in the source code of the Solv Protocol stUSD - Audit project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

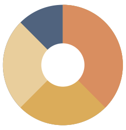
The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

FINDINGS | SOLV PROTOCOL STUSD - AUDIT



8	0	3	2	2	1
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Solv Protocol stUSD - Audit. Through this audit, we have uncovered 8 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

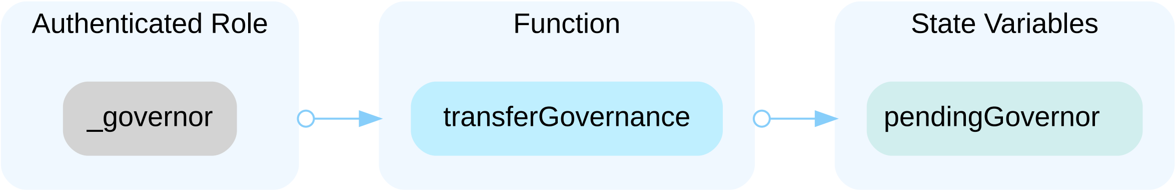
ID	Title	Category	Severity	Status
COT-01	**Centralization Related Risks**	**Centralization**	**Major**	● **Acknowledged**
SWT-01	**Centralized Control Of Contract Upgrade**	**Centralization**	**Major**	● **Acknowledged**
SWT-02	Unhandled Scenario In Burn() Function Causing Loss Of ERC-3525 Tokens	Logical Issue	Major	● Resolved
SWT-03	Third-Party Dependency Usage	Design Issue	Medium	● Acknowledged
SWT-04	The Contract Does Not Include The `onERC3525Received()` Function	Design Issue	Medium	● Resolved
COT-02	Missing Zero Address Validation	Volatile Code	Minor	● Resolved
SWT-05	Potential Divide By Zero	Logical Issue	Minor	● Resolved
SWT-06	The Design Of The Contract `SftWrappedToken`	Design Issue	Informational	● Acknowledged

COT-01 | CENTRALIZATION RELATED RISKS

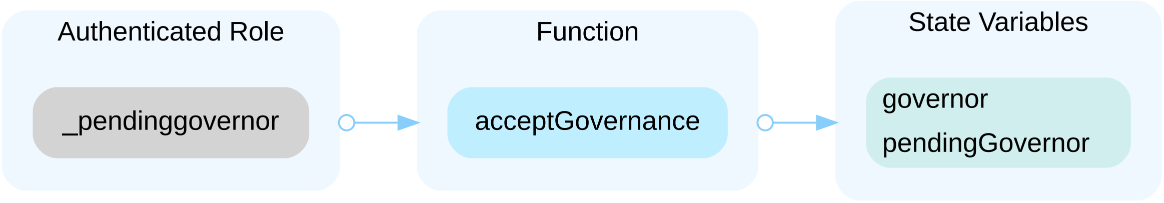
Category	Severity	Location	Status
Centralization	● Major	contracts/SftWrappedTokenFactory.sol (v1): 50, 55, 65, 75, 81, 115; contracts/access/AdminControl.sol (v1): 28, 33; contracts/access/GovernorControl.sol (v1): 28, 33	● Acknowledged

Description

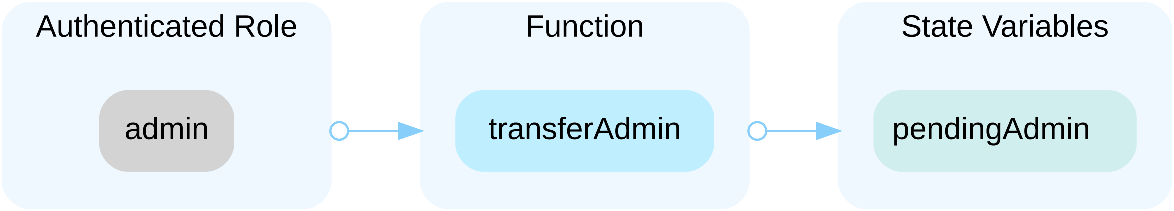
In the contract `GovernorControl` the role `governor` has authority over the functions shown in the diagram below. Any compromise to the `governor` account may allow the hacker to take advantage of this authority and set the pending governor's address.



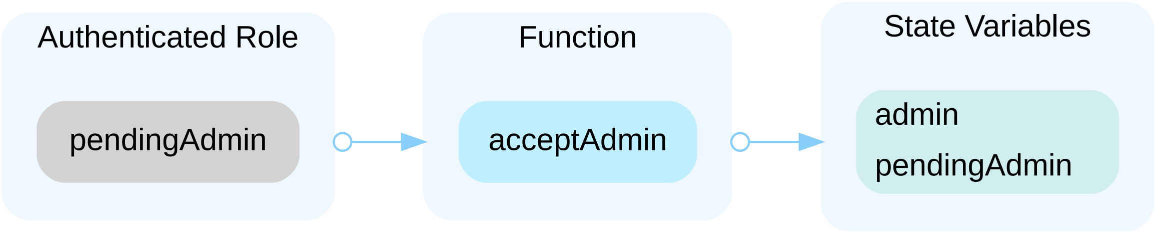
In the contract `GovernorControl` the role `pendinggovernor` has authority over the functions shown in the diagram below. Any compromise to the `pendinggovernor` account may allow the hacker to take advantage of this authority and set the governor's address.



In the contract `AdminControl` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` account may allow the hacker to take advantage of this authority and set the pending admin's address.

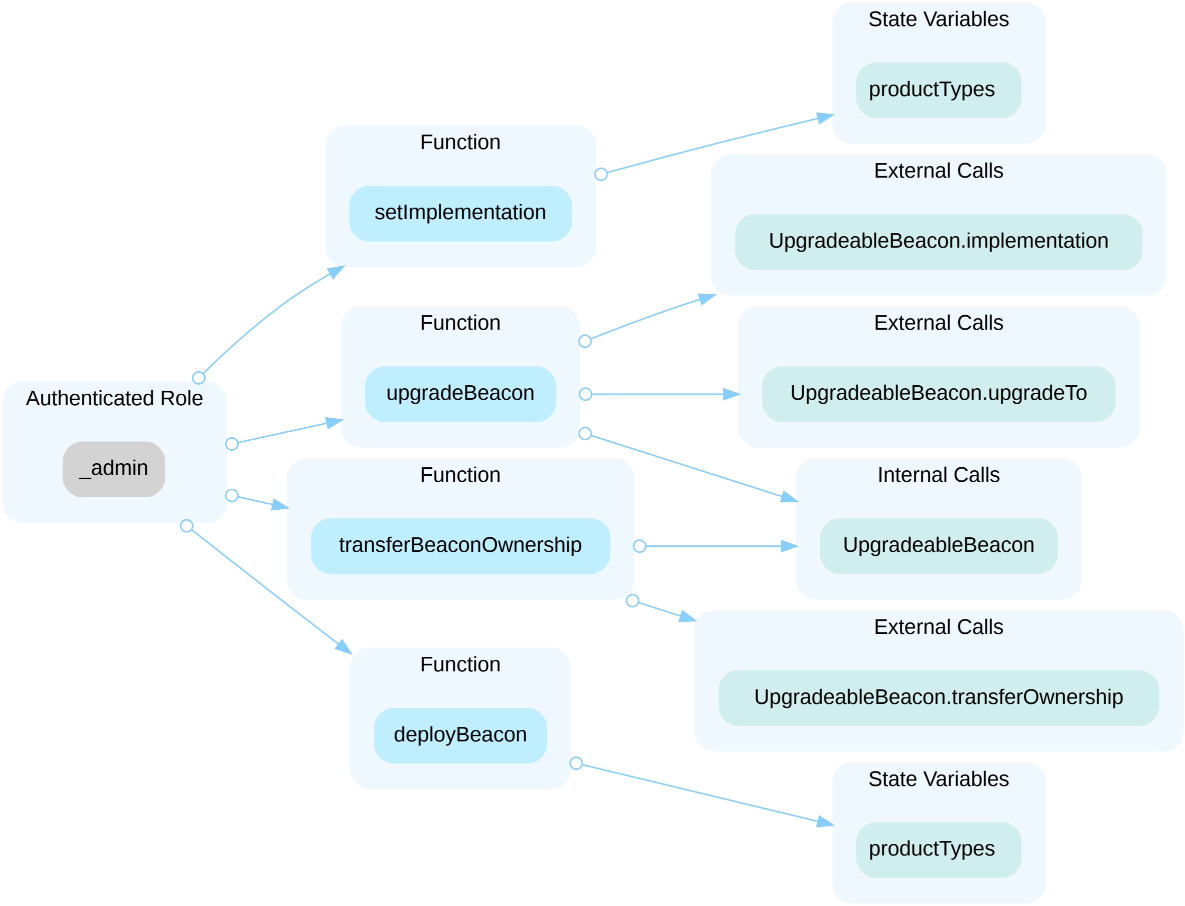


In the contract `AdminControl` the role `pendingAdmin` has authority over the functions shown in the diagram below. Any compromise to the `pendingAdmin` account may allow the hacker to take advantage of this authority and set the admin's address.

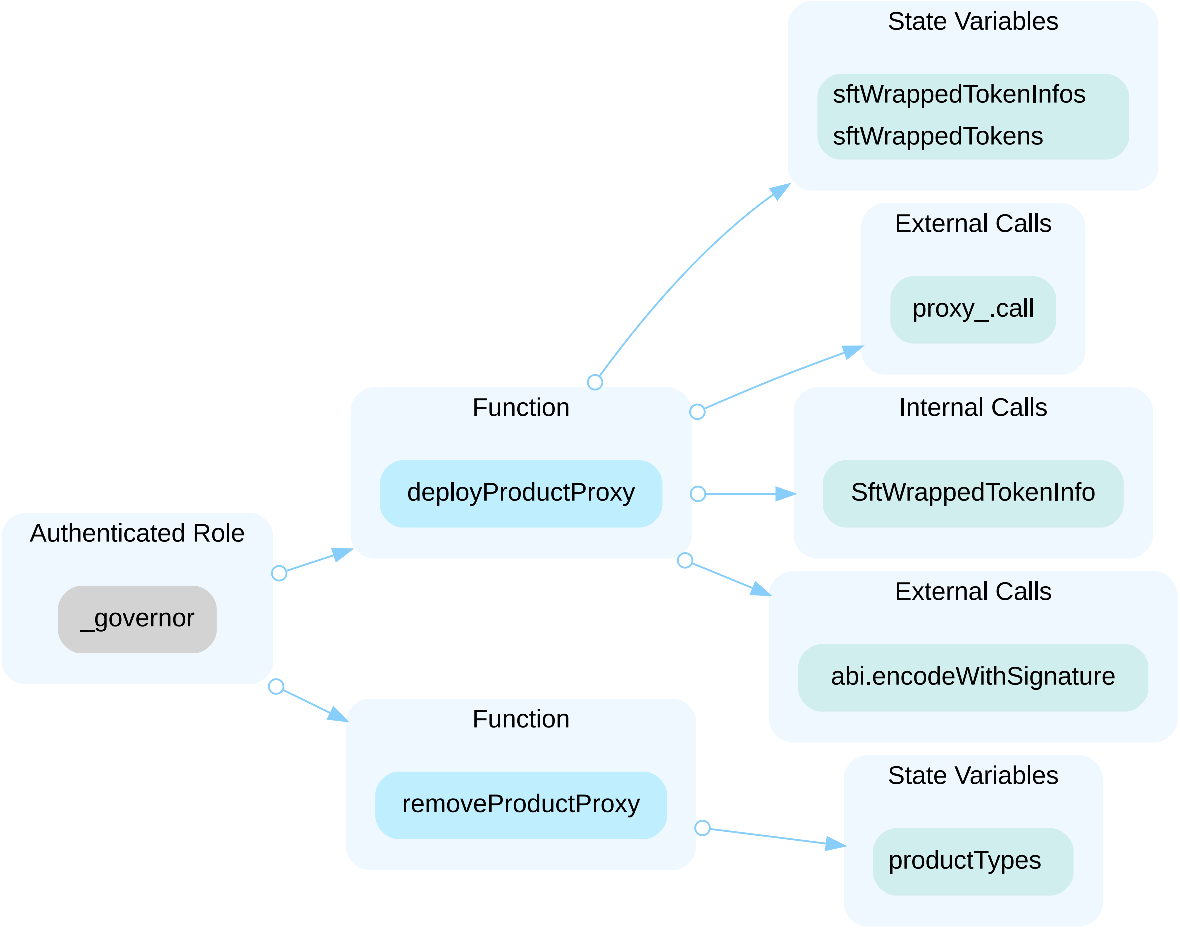


In the contract `SftWrappedTokenFactory` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` account may allow the hacker to take advantage of this authority and

- set the implementation address
- deploy the Beacon proxy
- upgrade the implementation of the Beacon proxy
- transfer the ownership of the Beacon proxy to the new owner



In the contract `SftWrappedTokenFactory` the role `governor` has authority over the functions shown in the diagram below. Any compromise to the `governor` account may allow the hacker to take advantage of this authority and deploy/remove product proxy.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[SOLV Team, 12/25/2023]:

Administrative privileges are typically transitioned to a Timelock, multi-signature addresses, or voting mechanism once the live operation has stabilized.

Admin privileges are primarily employed for deploying and upgrading Beacon-related contracts. They will be transferred to Timelock and voting mode once the operational functions of the SftWrappedToken contract are stable.

Governor privileges are mainly utilized for deploying new SftWrappedToken products. These privileges are then transferred to multi-signatory addresses upon the completion of SftWrappedTokenFactory contract deployment, and are converted to a voting mode when appropriate.

SWT-01 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	● Major	contracts/SftWrappedToken.sol (v1): 41	● Acknowledged

Description

In the contract `SftWrappedToken` , the role `admin` of the proxy has the authority to update the implementation contract behind the proxy contract.

Any compromise to the `admin` account may allow a hacker to take advantage of this authority and change the implementation contract which is pointed by proxy and therefore execute potential malicious functionality in the implementation contract.

Recommendation

We recommend that the team make efforts to restrict access to the admin of the proxy contract. A strategy of combining a time-lock and a multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. In addition, the team should be transparent and notify the community in advance whenever they plan to migrate to a new implementation contract.

Here are some feasible short-term and long-term suggestions that would mitigate the potential risk to a different level and suggestions that would permanently fully resolve the risk.

Short Term:

A combination of a time-lock and a multi signature (⅔, ⅗) wallet mitigate the risk by delaying the sensitive operation and avoiding a single point of key management failure.

- A time-lock with reasonable latency, such as 48 hours, for awareness of privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to a private key compromised;
  AND
- A medium/blog link for sharing the time-lock contract and multi-signers addresses information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.

- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.

- Provide a link to the **medium/blog** with all of the above information included.

Long Term:

A combination of a time-lock on the contract upgrade operation and a DAO for controlling the upgrade operation mitigate the contract upgrade risk by applying transparency and decentralization.

- A time-lock with reasonable latency, such as 48 hours, for community awareness of privileged operations;
  AND
- Introduction of a DAO, governance, or voting module to increase decentralization, transparency, and user involvement;
  AND
- A medium/blog link for sharing the time-lock contract, multi-signers addresses, and DAO information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.
- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.
- Provide a link to the **medium/blog** with all of the above information included.

Permanent:

Renouncing ownership of the `admin` account or removing the upgrade functionality can *fully* resolve the risk.

- Renounce the ownership and never claim back the privileged role;
  OR
- Remove the risky functionality.

Note: we recommend the project team consider the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[SOLV Team, 12/25/2023]:

Administrative privileges are typically transitioned to a Timelock, multi-signature addresses, or voting mechanism once the live operation has stabilized.

Admin privileges are primarily employed for deploying and upgrading Beacon-related contracts. They will be transferred to Timelock and voting mode once the operational functions of the SftWrappedToken contract are stable.

SWT-02 | UNHANDLED SCENARIO IN BURN() FUNCTION CAUSING LOSS OF ERC-3525 TOKENS

Category	Severity	Location	Status
Logical Issue	● Major	contracts/SftWrappedToken.sol (v1): 97	● Resolved

Description

Upon detailed review of the `burn()` function, it is discovered that it lacks proper validation checks for scenarios where the input `sftId_` corresponds to the `holdingValueSftId` managed by the contract itself. In such cases, users engage the `burn()` function, expecting to exchange their ERC-20 tokens for an equivalent amount of ERC-3525 tokens. However, due to the absence of a check verifying that `sftId_` is distinct from `holdingSftId` and ensuring the caller is indeed the owner of `sftId_`, the ERC-20 tokens are burned without the corresponding release of ERC-3525 tokens to the user. This constitutes a critical flaw as it leads to a unilateral loss for the user without the anticipated token exchange.

Proof of Concept

```
function testBurn() public{
        sftWrappedToken.mint(sftid, mintCount);
        assert(sftWrappedToken.balanceOf(address(this)) == mintCount);
        assert(erc3525.balanceOf(sftWrappedToken.holdingValueSftId()) == mintCount);

        sftWrappedToken.burn(mintCount, sftWrappedToken.holdingValueSftId());
        assert(sftWrappedToken.balanceOf(address(this)) == 0);
        assert(erc3525.balanceOf(sftWrappedToken.holdingValueSftId()) == 0);

    }
```

Recommendation

It is imperative for the contract to implement an additional mechanism that will scrutinize the ownership of `sftId_` when it is not null. In events when `sftId_` is determined to be equal to `holdingValueSftId` , the contract should default to a behavior that mirrors the zero `sftId_` condition. Specifically, this could entail generating a new ERC-3525 token with a slot compatible with `holdingValueSftId` , ensuring that users are not deprived of their tokens due to this oversight.

Alleviation

The client revised the code and resolved this issue in commit : 15f6f8b2363056b0c0dd824df09bccc40b448531

SWT-03 | THIRD-PARTY DEPENDENCY USAGE

Category	Severity	Location	Status
Design Issue	● Medium	contracts/SftWrappedToken.sol (v1): 43, 45	● Acknowledged

Description

The contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

```
43      address public wrappedSftAddress;
```

- The contract `SftWrappedToken` interacts with third party contract with `IERC3525` interface via `wrappedSftAddress` .

```
45      address public navOracle;
```

- The contract `SftWrappedToken` interacts with third party contract with `INavOracle` interface via `navOracle` .

Recommendation

The auditors understood that the business logic requires interaction with third parties. It is recommended for the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[SOLV Team, 12/25/2023]:

The contracts corresponding to `wrappedSftAddress` and `navOracle` are developed by SOLV and have no uncontrollable third-party dependencies.

SWT-04 | THE CONTRACT DOES NOT INCLUDE THE `onERC3525Received()` FUNCTION

Category	Severity	Location	Status
Design Issue	● Medium	contracts/SftWrappedToken.sol (v1): 41	● Resolved

Description

This EIP defines a simple 'Check, Notify and Response' model for better flexibility as well as simplicity:

1. No extra `safeTransferFrom` methods are needed, all callers only need to call one kind of transfer;
2. All ERC-3525 contracts MUST check for the existence of `onERC3525Received` on the recipient contract and call the function when it exists;
3. Any smart contract can implement `onERC3525Received` function for the purpose of being notified after receiving values; this function MUST return 0x009ce20b (i.e. `bytes4(keccak256('onERC3525Received(address,uint256,uint256,uint256,bytes)'))`) if the transfer is accepted, or any other value if the transfer is rejected.

There is a special case for this notification/acceptance mechanism: since ERC-3525 allows value transfer from an address to itself, when a smart contract which implements `onERC3525Received` transfers value to itself, `onERC3525Received` will also be called. This allows for the contract to implement different rules of acceptance between self-value-transfer and receiving value from other addresses.

However, this contract does not follow the above rule and does not implement the `onERC3525Received()`.

Recommendation

We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

The client revised the code and resolved this issue in commit : 7f71bbb7cde91e3d42fe13a13f07b6c15a9253bd

COT-02 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/SftWrappedToken.sol (v1): 61, 63; contracts/access/AdminControl.sol (v1): 30; contracts/access/GovernorControl.sol (v1): 30	● Resolved

Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior or vulnerabilities. For example, transferring tokens to a zero address can result in a permanent loss of those tokens.

```
61          wrappedSftAddress = wrappedSftAddress_;
```

- `wrappedSftAddress_` is not zero-checked before being used.

```
63          navOracle = navOracle_;
```

- `navOracle_` is not zero-checked before being used.

```
30          pendingAdmin = newPendingAdmin_;
```

- `newPendingAdmin_` is not zero-checked before being used.

```
30          pendingGovernor = newPendingGovernor_;
```

- `newPendingGovernor_` is not zero-checked before being used.

Recommendation

It is recommended to add a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

The client revised the code and resolved this issue in commit : 15f6f8b2363056b0c0dd824df09bccc40b448531

SWT-05 | POTENTIAL DIVIDE BY ZERO

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/SftWrappedToken.sol (v1): 132	● Resolved

Description

Performing division by zero would raise an error and revert the transaction.

```
132         return value * (10 ** decimals()) / latestNav;
```

The expression `value * (10 ** decimals()) / latestNav` may divide by zero. Its divisor has has estimated interval [0, 115792089237316195423570985008687907853269984665640564039457584007913129639935].

Recommendation

It is recommended to either reformulate the divisor expression, or to use conditionals or require statements to rule out the possibility of a divide-by-zero.

Alleviation

The client revised the code and resolved this issue in commit : 15f6f8b2363056b0c0dd824df09bccc40b448531

SWT-06 | THE DESIGN OF THE CONTRACT `SftWrappedToken`

Category	Severity	Location	Status
Design Issue	● Informational	contracts/SftWrappedToken.sol (v1): 41	● Acknowledged

Description

When a user calls `mint()` and the passed amount equals the balance of `sftId_`, the ownership of `sftId_` transfers to the contract. The initial assignment is to `holdingValueSftId`, and the remainder is inserted into `_holdingEmptySftIds`.

Subsequently, when a user calls `burn()`, and if `_holdingEmptySftIds` is empty, a new ID regenerates, and ownership remains with the current user. No ownership of `holdingValueSftId` is reclaimed. However, if `_holdingEmptySftIds` is not empty, the last ID is taken and removed. This results in the ownership of the contract transferring to the user. Nevertheless, the number of contracts may not match those initially transferred. Consequently, there will be a discrepancy in the token IDs between the `burn()` and `mint()` operations. Different ERC-3225 token IDs may exhibit varying value differences, potentially leading to losses.

Recommendation

We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

[SOLV Team, 12/25/2023]:

The purpose of `_holdingEmptySftIds` in `SftWrappedToken` contract is to reuse existing `tokenId`, thereby reducing Gas consumption. In terms of business rules, maintaining consistency between wrapped and unwrapped `tokenId` is unnecessary, as there is no meaningful distinction between different `tokenId`. For users, a `tokenId` serves merely as an identifier and does not carry any business significance. Consequently, any inconsistency between the `tokenId` of wrapped and unwrapped tokens has no impact on users.

APPENDIX | SOLV PROTOCOL STUSD - AUDIT

Finding Categories

Categories	Description
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED “AS IS” AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.