Superfluid Finance

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **February 21st, 2022 - March 15th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	03/15/2022	Omar Alshaeb
0.2	Draft Review	03/18/2022	Gabi Urrutia
0.3	Draft Update	03/29/2022	Omar Alshaeb
0.4	Draft Update Review	03/30/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Omar Alshaeb	Halborn	Omar.Alshaeb@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Superfluid engaged Halborn to conduct a security audit on their smart contracts beginning on February 21st, 2022 and ending on March 15th, 2022 . The security assessment was scoped to the smart contracts provided in the GitHub repository Superfluid repository.

1.2 AUDIT SUMMARY

The team at Halborn was provided three weeks for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn did not detect any security vulnerabilities, but did identify an informational issue that can be addressed by the Superfluid team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the bridge code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn 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. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:
The security assessment was scoped to the following smart contracts:

- BatchLiquidator.sol
- ConstantFlowAgreementV1.sol
- Definitions.sol
- InstandDistributionAgreementV1.sol
- MaticBridgedNativeSuperToken.sol
- NativeSuperToken.sol
- Superfluid.sol
- SuperfluidGovernanceBase.sol
- SuperfluidOwnableGovernance.sol
- Superupgrader.sol
- TOGA.sol
- TokenCustodian.sol
- SuperfluidToken.sol
- SuperToken.sol
- SuperTokenFactory.sol
- AgreementBase.sol
- SuperfluidGovernanceII.sol

Commit ID: 2fb0afd711479a3ca373de12d6643c0655f27b49

1.5 WORK DONE

In addition to the manual and automated testing that can be seen in this report, during the audit, the security engineer manually reviewed the source code and focused on the host contract, which is the core of the protocol.

One of the important tests was to double-check if the previous critical vulnerability that the protocol had was properly mitigated. It has been confirmed that there is no way to exploit it anymore because you always have the transaction context hashed and stored in a state variable and then matched against the agreements contracts, which properly secures the contracts.

After taking into account all the possible attack vectors that the contracts can have, it has been shown that always having the current transaction context hashed and stored in a state variable in the host contract absolutely reduces the chances that an attack is successful. Since most transactions sent to the Superfluid protocol have to go through the same process, attack vectors are also reduced.

In summary, Halborn finds the use of this context hash very useful to have complete control over the transactions going into Superfluid smart contracts and to prevent any kind of malicious data injection and modifying the transaction context to will of the attacker.

1.6 FUTURE AUDITS

Halborn suggests the need for a new security audit whenever important new functionality is developed in the core of the protocol. Focusing on the host contract, and also on the main agreements (constant flow and instant distribution).

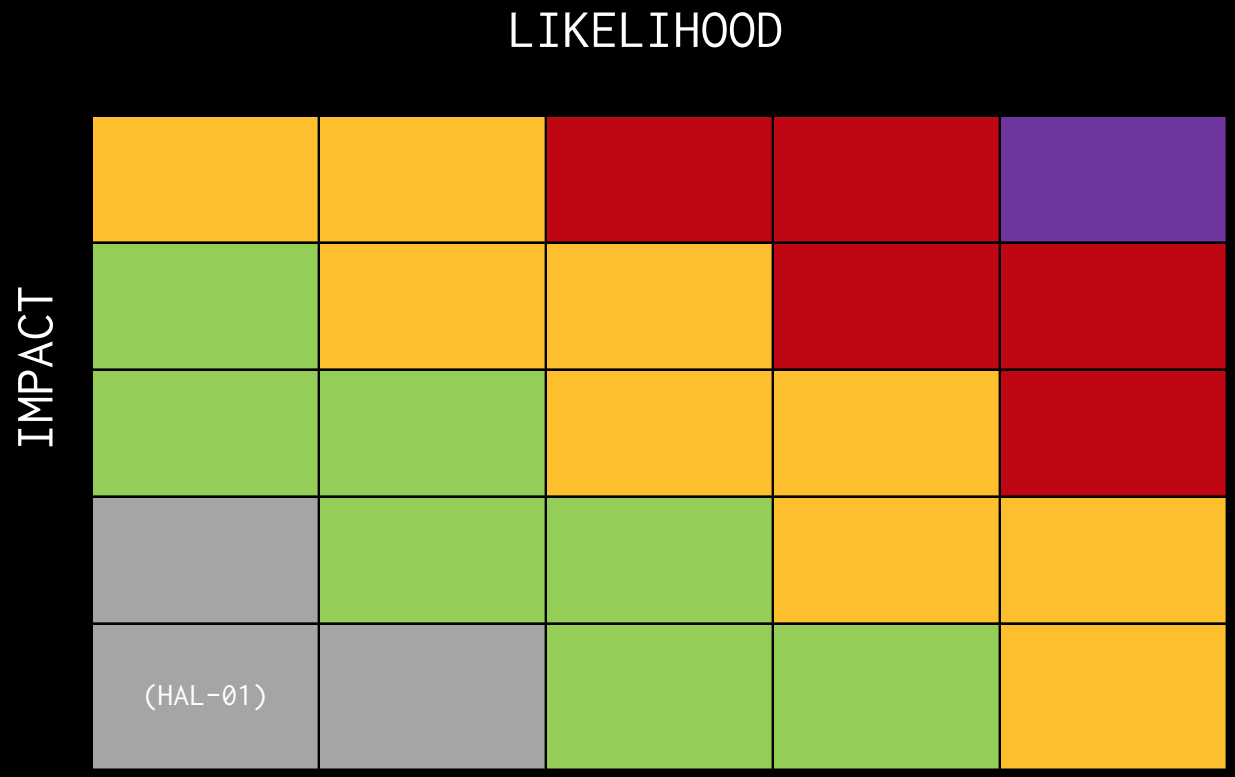
Moreover, it is essential to audit the security of new upcoming projects that use the Superfluid protocol, as the way they interface with the core protocol can create security vulnerabilities in third-party apps.

In summary, it would be very beneficial for the project to perform a security audit in the following scenarios:

- Changes have been made to the host contract
- Changes have been made to the constant flow agreement contract
- Changes have been made to the instant distribution agreement contract
- New agreement contracts have been developed
- New projects aiming to use the Superfluid protocol

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	0	1



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	-

FINDINGS & TECH DETAILS

3.1 (HAL-01) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

Some functions are marked as public, but never called directly within the same contract or in any of its descendants.

Code Location:

Listing 1

```
- Superfluid.isAppJailed(ISuperApp) (contracts/superfluid/Superfluid.sol#396-403)
- TokenCustodian.flush(IERC777,address) (contracts/utils/TokenCustodian.sol#22-30)
- SuperfluidGovernanceBase.getRewardAddress(ISuperfluid,ISuperfluidToken) (contracts/gov/SuperfluidGovernanceBase.sol#221-230)
- SuperfluidGovernanceBase.setRewardAddress(ISuperfluid,ISuperfluidToken,address) (contracts/gov/SuperfluidGovernanceBase.sol#232-244)
- SuperfluidGovernanceBase.clearRewardAddress(ISuperfluid,ISuperfluidToken) (contracts/gov/SuperfluidGovernanceBase.sol#246-256)
- SuperfluidGovernanceBase.getCFAv1LiquidationPeriod(ISuperfluid,ISuperfluidToken) (contracts/gov/SuperfluidGovernanceBase.sol#265-275)
- SuperfluidGovernanceBase.setCFAv1LiquidationPeriod(ISuperfluid,ISuperfluidToken,uint256) (contracts/gov/SuperfluidGovernanceBase.sol#277-289)
- SuperfluidGovernanceBase.clearCFAv1LiquidationPeriod(ISuperfluid,ISuperfluidToken) (contracts/gov/SuperfluidGovernanceBase.sol#291-301)
- SuperfluidGovernanceBase.getSuperTokenMinimumDeposit(ISuperfluid,ISuperfluidToken) (contracts/gov/SuperfluidGovernanceBase.sol#310-318)
- SuperfluidGovernanceBase.clearSuperTokenMinimumDeposit(ISuperfluid,ISuperToken) (contracts/gov/SuperfluidGovernanceBase.sol#329-336)
```

```
11 - SuperfluidGovernanceBase.isTrustedForwarder(ISuperfluid,
↳ ISuperfluidToken,address) (contracts/gov/SuperfluidGovernanceBase.
↳ sol#346-357)
12 - SuperfluidGovernanceBase.enableTrustedForwarder(ISuperfluid,
↳ ISuperfluidToken,address) (contracts/gov/SuperfluidGovernanceBase.
↳ sol#359-371)
13 - SuperfluidGovernanceBase.disableTrustedForwarder(ISuperfluid,
↳ ISuperfluidToken,address) (contracts/gov/SuperfluidGovernanceBase.
↳ sol#373-385)
14 - SuperfluidGovernanceBase.clearTrustedForwarder(ISuperfluid,
↳ ISuperfluidToken,address) (contracts/gov/SuperfluidGovernanceBase.
↳ sol#387-398)
15 - SuperfluidGovernanceBase.isAuthorizedAppFactory(ISuperfluid,
↳ address) (contracts/gov/SuperfluidGovernanceBase.sol#420-430)
16 - SuperfluidGovernanceBase.authorizeAppFactory(ISuperfluid,address
↳ ) (contracts/gov/SuperfluidGovernanceBase.sol#436-454)
17 - SuperfluidGovernanceBase.unauthorizeAppFactory(ISuperfluid,
↳ address) (contracts/gov/SuperfluidGovernanceBase.sol#460-469)
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

If these functions are not intended to be called internally or by descendants, it is better to mark them as external to reduce gas costs.

MANUAL TESTING

The following test cases have been executed either on a forked Ropsten network or directly on the Ropsten network.

4.1 Constant Flow Agreement

- After creating a constant flow agreement between two users. This test case attempts to delete it from a user who is neither the sender nor the recipient. Since the sender's account is not critical, this is not allowed.

Proof of Concept:

PoC-1:

Listing 2: Proof of Concept using Brownie

```
# Users to use for the test
user = a[0]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory
↳ ())
ida = InstantDistributionAgreementV1.at('0
↳ xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Check an existing CFA beetwen user1 and user2
output.redd("cfa.getFlow(fUSDCx, user1, user2) -> " + str(cfa.
↳ getFlow(fUSDCx, user1, user2)))
# Delete flow using another user
encoded_tx = cfa.signatures['deleteFlow'] + eth_abi.encode_abi(['
↳ address', 'address', 'address', 'bytes',], (fUSDCx, user1, user2,
```

```
↳ b'',)).hex()
20 bytes_encoded_tx = to_bytes(encoded_tx[:-64],'bytes')
21 tx = host.callAgreement(cfa.address, bytes_encoded_tx, 0, {'from':
↳  user})
```

Result:

```
>>> # Users to use for the test
user = a[0]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Check an existing CFA beetwen user1 and user2
output.redd("cfa.getFlow(fUSDCx, user1, user2) → " + str(cfa.getFlow(fUSDCx, user1, user2)))
# Delete flow using another user
encoded_tx = cfa.signatures['deleteFlow'] + eth_abi.encode_abi(['address', 'address', 'address', 'bytes',], (fUSDCx, user1, user2, b'',)).hex()
bytes_encoded_tx = to_bytes(encoded_tx[:-64],'bytes')
tx = host.callAgreement(cfa.address, bytes_encoded_tx, 0, {'from': user})
cfa.getFlow(fUSDCx, user1, user2) → (1645831102, 771604938270, 2777778688622592, 0)
  File "<console>", line 21, in <module>
  File "brownie/network/contract.py", line 1629, in __call__
    return self.transact(*args)
  File "brownie/network/contract.py", line 1502, in transact
    return tx["from"].transfer(
  File "brownie/network/account.py", line 644, in transfer
    receipt, exc = self._make_transaction(
  File "brownie/network/account.py", line 752, in _make_transaction
    exc = VirtualMachineError(e)
  File "brownie/exceptions.py", line 121, in __init__
    raise ValueError(str(exc)) from None
ValueError: Execution reverted during call: 'execution reverted: CFA: sender account is not critical'. This transaction will likely revert. If you wish to broadcast, include `allow_revert
```

- After creating a constant flow agreement between two users. This test case attempts to delete it from an address that is not the host contract of the CFA contract where the agreement has been created. There is a specific requirement to prevent this case from happening.

PoC-2:

Listing 3: Proof of Concept using Brownie

```
1 # Users to use for the test
2 user = a[0]
3 user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
4 user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
5 # fUSDCx address
6 fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
7 # Import all contracts needed from the Ropsten network
8 host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
9 gov = SuperfluidGovernanceII.at(host.getGovernance())
10 superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory
↳ ())
11 ida = InstantDistributionAgreementV1.at('0
```

```
↳ xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
12 cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
13 supertoken = SuperToken.at(fUSDCx)
14 # Setting gas limit
15 network.gas_limit(6700000)
16 # Check an existing CFA beetwen user1 and user2
17 output.redd("cfa.getFlow(fUSDCx, user1, user2) -> " + str(cfa.
↳ getFlow(fUSDCx, user1, user2)))
18 # Delete flow using another host
19 cfa.deleteFlow(fUSDCx, user1, user2, 0, {'from': user})
```

Result:

```
>>> # Users to use for the test
user = a[0]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Check an existing CFA beetwen user1 and user2
output.redd("cfa.getFlow(fUSDCx, user1, user2) -> " + str(cfa.getFlow(fUSDCx, user1, user2)))
# Delete flow using another host
cfa.deleteFlow(fUSDCx, user1, user2, 0, {'from': user})
cfa.getFlow(fUSDCx, user1, user2) -> (1645831102, 771604938270, 2777778688622592, 0)
  File "<console>", line 19, in <module>
  File "brownie/network/contract.py", line 1629, in __call__
    return self.transact(*args)
  File "brownie/network/contract.py", line 1502, in transact
    return tx["from"].transfer(
  File "brownie/network/account.py", line 644, in transfer
    receipt, exc = self._make_transaction(
  File "brownie/network/account.py", line 752, in _make_transaction
    exc = VirtualMachineError(e)
  File "brownie/exceptions.py", line 121, in __init__
    raise ValueError(str(exc)) from None
ValueError: Execution reverted during call: 'execution reverted: unauthorized host'. This transaction will likely revert. If you wish to broadcast, include `allow_revert:True` as a transa
```

- After creating a constant flow agreement between two users. This test case attempts to delete it by injecting malicious bytes into the context ctx. Since the agreement contract is checking and comparing the current ctx with the previously hashed context in the host contract, this is not possible.

PoC-3:

Listing 4: Proof of Concept using Brownie

```
1 # Users to use for the test
2 user = a[0]
3 user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
4 user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
```

MANUAL TESTING

```
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory
↳ ())
ida = InstantDistributionAgreementV1.at('0
↳ xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Check an existing CFA beetwen user1 and user2
output.redd("cfa.getFlow(fUSDCx, user1, user2) -> " + str(cfa.
↳ getFlow(fUSDCx, user1, user2)))
# Delete flow using another host
encoded_tx = "0
↳ xb4b333c6000000000000000000000002dc36872a445adf0bff63cc0eeee52a2b8
↳
01625
↳ f000000000000000000000000291b1056c3c104205d5cf157f173e16ab3d1965f
0000000000000000000000
↳ e8603a92664ae818bf8c31920786afb61c6cc7dc000000000000
0000000000000000000000000000000000000000000000000000
8000000000000000000000000000000000
0000000000000000000000000000000000000000000000000"
bytes_encoded_tx = to_bytes(encoded_tx,'bytes')
tx = host.callAgreement(cfa.address, bytes_encoded_tx, 0, {'from':
↳  user})
```

Result:

```
>>> # Users to use for the test
user = a[0]
user1 = '0×291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0×e8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0×2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0×F2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0×AD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0×aD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Check an existing CFA beetwen user1 and user2
output.redd("cfa.getFlow(fUSDCx, user1, user2) → " + str(cfa.getFlow(fUSDCx, user1, user2)))
# Delete flow using another host
encoded_tx = "0×b4b333c6000000000000000000000000002dc36872a445adf0bff63cc0eeee52a2b801625f000000000000000000000000291b1056c3c104205d5cf157f173e16ab3d1965f000000000000000000000000e8603a92664
000000000000000000000000000000000000000000000000000000000000000000000000800000000000000000000000000000000000000000000000000"
bytes_encoded_tx = to_bytes(encoded_tx,'bytes')
tx = host.callAgreement(cfa.address, bytes_encoded_tx, 0, {'from': user})
cfa.getFlow(fUSDCx, user1, user2) → (1645831102, 771604938270, 2777778688622592, 0)
  File "<console>", line 21, in <module>
  File "brownie/network/contract.py", line 1629, in __call__
    return self.transact(*args)
  File "brownie/network/contract.py", line 1502, in transact
    return tx["from"].transfer(
  File "brownie/network/account.py", line 644, in transfer
    receipt, exc = self._make_transaction(
  File "brownie/network/account.py", line 752, in _make_transaction
    exc = VirtualMachineError(e)
  File "brownie/exceptions.py", line 121, in __init__
    raise ValueError(str(exc)) from None
ValueError: Execution reverted during call: 'execution reverted: invalid ctx'. This transaction will likely revert. If you wish to broadcast, include `allow_revert:True` as a transaction
```

4.2 Super Tokens

- This test case attempts to trigger an XSS vulnerability in the Superfluid's app by creating a new Super Token with an HTML tag in the name and adding the token to the app. Since the application correctly encodes the characters, this is not possible.

Proof of Concept:

PoC-1:

Listing 5: Proof of Concept using Brownie

```
# Users to use for the test
user = a[0]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
```

MANUAL TESTING

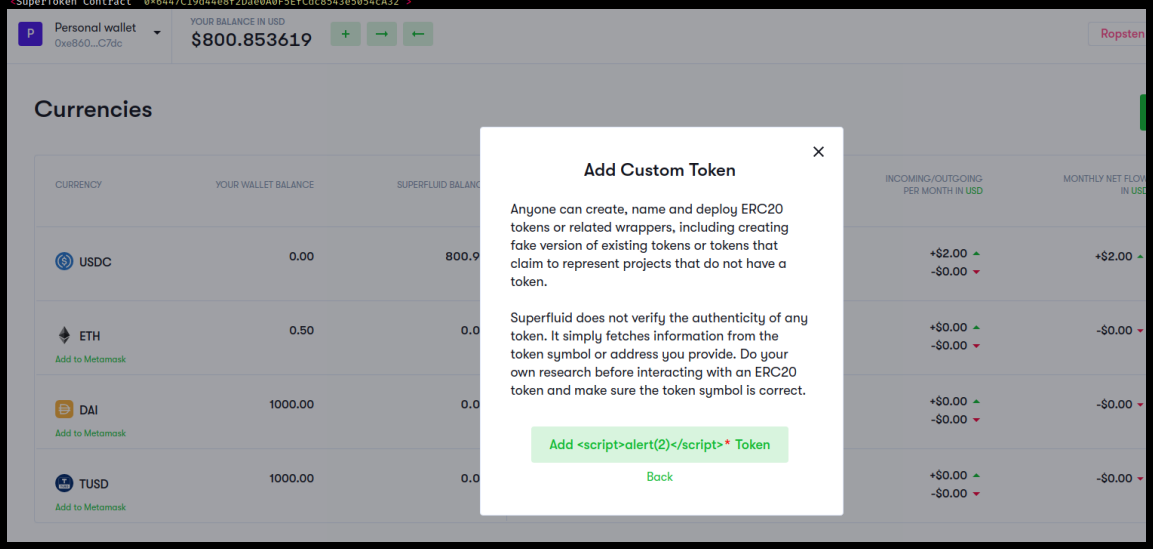
```
12 cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
13 supertoken = SuperToken.at(fUSDCx)
14 # Deploy new SuperToken contract
15 mockTOken = SuperToken.deploy(host, {'from': user})
16 # Init the new SuperToken
17 tx = mockTOken.initialize(supertoken, 18, "<script>alert(1)</
↳ script>", "<script>alert(2)</script>", {'from': user})
```

Result:

```
>>> mockTOken = SuperToken.deploy(host, {'from': user})
Transaction sent: 0x9007fb130deec09d7bba4fed794707ed73f28b1eac9af78e2b1b4018f4133ca5
  Gas price: 20.515283175 gwei   Gas limit: 5210066   Nonce: 11
  SuperToken.constructor confirmed   Block: 12098483   Gas used: 4736424 (90.91%)
  SuperToken deployed at: 0x6447C19d44e8f2Dae0A0F5EfCdc8543e5054cA32

>>> supertoken.decimals()
18
>>> tx = mockTOken.initialize(supertoken, 18, "<script>alert(1)</script>", "<script>alert(2)</script>", {'from': user})
Transaction sent: 0x7039ea8c119401736532faec0d82307ea9a1628e0aa55b6e26701f3a25ff5439
  Gas price: 17.979264504 gwei   Gas limit: 189178   Nonce: 12
  SuperToken.initialize confirmed   Block: 12098536   Gas used: 169803 (89.76%)

>>> mockTOken
<SuperToken Contract '0x6447C19d44e8f2Dae0A0F5EfCdc8543e5054cA32'>
```



- After wrapping some Super Tokens to an account, this test case attempts to send some tokens to an arbitrary recipient from a non-operator account. Since the Super Token contract thoroughly checks whether or not the sender of the transaction is an operator of the account sending the tokens, this behavior is not allowed.

PoC-2:

Listing 6: Proof of Concept using Brownie

```
# Users to use for the test
user = a[0]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory
↳ ())
ida = InstantDistributionAgreementV1.at('0
↳ xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Send tokens from a non operator account
tx = supertoken.operatorSend(user1, user2, 10, 0x0, 0x0, {'from':
↳ user})
```

Result:

```
>>> # Users to use for the test
user = a[0]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Send tokens from a non operator account
tx = supertoken.operatorSend(user1, user2, 10, 0x0, 0x0, {'from': user})
  File "<console>", line 15, in <module>
  File "brownie/network/contract.py", line 1629, in __call__
    return self.transact(*args)
  File "brownie/network/contract.py", line 1502, in transact
    return tx["from"].transfer(
  File "brownie/network/account.py", line 644, in transfer
    receipt, exc = self._make_transaction(
  File "brownie/network/account.py", line 727, in _make_transaction
    raise VirtualMachineError(e) from None
  File "brownie/exceptions.py", line 121, in __init__
    raise ValueError(str(exc)) from None
ValueError: Gas estimation failed: 'execution reverted: SuperToken: caller is not an operator for holder'. This transaction will likely revert. If you wish to broadcast, you must set the
```

- This test case attempts to upgrade more ERC20 tokens than are available in the account. Since the function that upgrades ERC20 tokens to Super Tokens securely executes the necessary transfers, this is not possible.

PoC-3:

Listing 7: Proof of Concept using Brownie

```
# Users to use for the test
user = a[0]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory
↳ ())
ida = InstantDistributionAgreementV1.at('0
↳ xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Upgrade more tokens than available
tx = supertoken.upgrade(10, {'from': user})
```

Result:

```
>>> # Users to use for the test
user = a[0]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Upgrade more tokens than available
tx = supertoken.upgrade(10, {'from': user})
  File "<console>", line 15, in <module>
  File "brownie/network/contract.py", line 1629, in __call__
    return self.transact(*args)
  File "brownie/network/contract.py", line 1502, in transact
    return tx["from"].transfer(
  File "brownie/network/account.py", line 644, in transfer
    receipt, exc = self._make_transaction(
  File "brownie/network/account.py", line 727, in _make_transaction
    raise VirtualMachineError(e) from None
  File "brownie/exceptions.py", line 121, in __init__
    raise ValueError(str(exc)) from None
ValueError: Gas estimation failed: 'execution reverted: ERC20: transfer amount exceeds balance'. This transaction will likely revert. If you wish to broadcast, you must set the gas limit
```

- This test case attempts to downgrade more Super Tokens than are available in the account. Since the function that downgrades Super Tokens to ERC20 tokens is safely executing the necessary transfers, this is not possible.

PoC-4:

Listing 8: Proof of Concept using Brownie

```
# Users to use for the test
user = a[0]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory
↳ ())
ida = InstantDistributionAgreementV1.at('0
↳ xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Downgrade more tokens than available
tx = supertoken.downgrade(100, {'from': user})
```

Result:

```
>>> # Users to use for the test
user = a[0]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Downgrade more tokens than available
tx = supertoken.downgrade(100, {'from': user})
  File "<console>", line 15, in <module>
  File "brownie/network/contract.py", line 1629, in __call__
    return self.transact(*args)
  File "brownie/network/contract.py", line 1502, in transact
    return tx["from"].transfer(
  File "brownie/network/account.py", line 644, in transfer
    receipt, exc = self._make_transaction(
  File "brownie/network/account.py", line 727, in _make_transaction
    raise VirtualMachineError(e) from None
  File "brownie/exceptions.py", line 121, in __init__
    raise ValueError(str(exc)) from None
ValueError: Gas estimation failed: 'execution reverted: SuperfluidToken: burn amount exceeds balance'. This transaction will likely revert. If you wish to broadcast, you must set the gas
```

4.3 Instant Distribution Agreement

- After a publisher has created an index, this test case attempts to update it from another user by injecting malicious bytes into the context ctx. Since the agreement contract is checking and comparing

the current ctx with the previously hashed context in the host contract, this is not possible.

Proof of Concept:

PoC-1:

Listing 9: Proof of Concept using Brownie

```
# Users to use for the test
user = a[0]
secondUser = a[1]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory
↳ ())
ida = InstantDistributionAgreementV1.at('0
↳ xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Update index
encoded_tx = ida.signatures['updateIndex'] + eth_abi.encode_abi(['
↳ address', 'uint32', 'uint128', 'bytes',], (supertoken.address, 0,
↳ 1, b'',)).hex()
bytes_encoded_tx = to_bytes(encoded_tx,'bytes')
tx = host.callAgreement(ida.address, bytes_encoded_tx, 0, {'from':
↳  secondUser})
```

Result:

```
>>> # Users to use for the test
user = a[0]
secondUser = a[1]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Update index
encoded_tx = ida.signatures['updateIndex'] + eth_abi.encode_abi(['address', 'uint32', 'uint128', 'bytes',], (supertoken.address, 0, 1, b'',)).hex()
bytes_encoded_tx = to_bytes(encoded_tx,'bytes')
tx = host.callAgreement(ida.address, bytes_encoded_tx, 0, {'from': secondUser})
  File "<console>", line 20, in <module>
  File "brownie/network/contract.py", line 1629, in __call__
    return self.transact(*args)
  File "brownie/network/contract.py", line 1502, in transact
    return tx["from"].transfer(
  File "brownie/network/account.py", line 644, in transfer
    receipt, exc = self._make_transaction(
  File "brownie/network/account.py", line 752, in _make_transaction
    exc = VirtualMachineError(e)
  File "brownie/exceptions.py", line 121, in __init__
    raise ValueError(str(exc)) from None
ValueError: Execution reverted during call: 'execution reverted: invalid ctx'. This transaction will likely revert. If you wish to broadcast, include `allow_revert:True` as a transaction
```

- After a publisher has created an index, this test case attempts to distribute it from another user by injecting malicious bytes into the ctx context. Since the agreement contract is checking and comparing the current ctx with the previously encrypted context in the host contract, this is not possible.

PoC-2:

Listing 10: Proof of Concept using Brownie

```
# Users to use for the test
user = a[0]
secondUser = a[1]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory
↳ ())
ida = InstantDistributionAgreementV1.at('0
↳ xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
```

```
17 # Distribute
18 encoded_tx = ida.signatures['distribute'] + eth_abi.encode_abi(['
↳ address', 'uint32', 'uint256', 'bytes',], (supertoken.address, 0,
↳ 1, b'',)).hex()
19 bytes_encoded_tx = to_bytes(encoded_tx,'bytes')
20 tx = host.callAgreement(ida.address, bytes_encoded_tx, 0, {'from':
↳  secondUser})
```

Result:

```
>>> # Users to use for the test
user = a[0]
secondUser = a[1]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Distribute
encoded_tx = ida.signatures['distribute'] + eth_abi.encode_abi(['address', 'uint32', 'uint256', 'bytes',], (supertoken.address, 0, 1, b'',)).hex()
bytes_encoded_tx = to_bytes(encoded_tx,'bytes')
tx = host.callAgreement(ida.address, bytes_encoded_tx, 0, {'from': secondUser})
  File "<console>", line 20, in <module>
  File "brownie/network/contract.py", line 1629, in __call__
    return self.transact(*args)
  File "brownie/network/contract.py", line 1502, in transact
    return tx["from"].transfer(
  File "brownie/network/account.py", line 644, in transfer
    receipt, exc = self._make_transaction(
  File "brownie/network/account.py", line 752, in _make_transaction
    exc = VirtualMachineError(e)
  File "brownie/exceptions.py", line 121, in __init__
    raise ValueError(str(exc)) from None
ValueError: Execution reverted during call: 'execution reverted: invalid ctx'. This transaction will likely revert. If you wish to broadcast, include `allow_revert:True` as a transaction
```

- After a publisher has created an index and a subscription to that index, this test case attempts to update another user's subscription by injecting malicious bytes into the ctx context. Since the agreement contract is checking and comparing the current ctx with the previously encrypted context in the host contract, this is not possible.

PoC-3:

Listing 11: Proof of Concept using Brownie

```
1 # Users to use for the test
2 user = a[0]
3 secondUser = a[1]
4 user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
5 user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
6 # fUSDCx address
7 fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
```

```
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory
↳ ())
ida = InstantDistributionAgreementV1.at('0
↳ xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Update Subscription
encoded_tx = ida.signatures['updateSubscription'] + eth_abi.
↳ encode_abi(['address', 'uint32', 'address', 'uint128', 'bytes',],
↳ (supertoken.address, 0, user.address, 1, b'',)).hex()
bytes_encoded_tx = to_bytes(encoded_tx,'bytes')
tx = host.callAgreement(ida.address, bytes_encoded_tx, 0, {'from':
↳  secondUser})
```

Result:

```
>>> # Users to use for the test
user = a[0]
secondUser = a[1]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Update Subscription
encoded_tx = ida.signatures['updateSubscription'] + eth_abi.encode_abi(['address', 'uint32', 'address', 'uint128', 'bytes',], (supertoken.address, 0, user.address, 1, b'',)).hex()
bytes_encoded_tx = to_bytes(encoded_tx,'bytes')
tx = host.callAgreement(ida.address, bytes_encoded_tx, 0, {'from': secondUser})
  File "<console>", line 20, in <module>
  File "brownie/network/contract.py", line 1629, in __call__
    return self.transact(*args)
  File "brownie/network/contract.py", line 1502, in transact
    return tx["from"].transfer(
  File "brownie/network/account.py", line 644, in transfer
    receipt, exc = self._make_transaction(
  File "brownie/network/account.py", line 752, in _make_transaction
    exc = VirtualMachineError(e)
  File "brownie/exceptions.py", line 121, in __init__
    raise ValueError(str(exc)) from None
ValueError: Execution reverted during call: 'execution reverted: invalid ctx'. This transaction will likely revert. If you wish to broadcast, include `allow_revert:True` as a transaction
```

- After a publisher has created an index and a subscription to that index has been created, this test case attempts to revoke another user's subscription by injecting malicious bytes into the ctx context. Since the agreement contract is checking and comparing the current ctx with the previously encrypted context in the host contract, this is not possible.

PoC-4:

Listing 12: Proof of Concept using Brownie

```
# Users to use for the test
user = a[0]
secondUser = a[1]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory
↳ ())
ida = InstantDistributionAgreementV1.at('0
↳ xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Revoke Subscription
encoded_tx = ida.signatures['revokeSubscription'] + eth_abi.
↳ encode_abi(['address', 'address', 'uint32', 'bytes',], (supertoken
↳ .address, user.address, 0, b'',)).hex()
bytes_encoded_tx = to_bytes(encoded_tx,'bytes')
tx = host.callAgreement(ida.address, bytes_encoded_tx, 0, {'from':
↳  secondUser})
```

Result:

```
>>> # Users to use for the test
user = a[0]
secondUser = a[1]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Revoke Subscription
encoded_tx = ida.signatures['revokeSubscription'] + eth_abi.encode_abi(['address', 'address', 'uint32', 'bytes',], (supertoken.address, user.address, 0, b'',)).hex()
bytes_encoded_tx = to_bytes(encoded_tx,'bytes')
tx = host.callAgreement(ida.address, bytes_encoded_tx, 0, {'from': secondUser})
  File "<console>", line 20, in <module>
  File "brownie/network/contract.py", line 1629, in __call__
    return self.transact(*args)
  File "brownie/network/contract.py", line 1502, in transact
    return tx["from"].transfer(
  File "brownie/network/account.py", line 644, in transfer
    receipt, exc = self._make_transaction(
  File "brownie/network/account.py", line 752, in _make_transaction
    exc = VirtualMachineError(e)
  File "brownie/exceptions.py", line 121, in __init__
    raise ValueError(str(exc)) from None
ValueError: Execution reverted during call: 'execution reverted: invalid ctx'. This transaction will likely revert. If you wish to broadcast, include `allow_revert:True` as a transaction
```

- After a publisher has created an index and a subscription to that index has been created, this test case attempts to unsubscribe from another user who is neither the publisher nor the subscriber. Since the function that removes a subscription correctly checks with a requirement whether the user submitting the transaction is the publisher or the subscriber, this behavior is not allowed.

PoC-5:

Listing 13: Proof of Concept using Brownie

```
# Users to use for the test
user = a[0]
secondUser = a[1]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory
↳ ())
ida = InstantDistributionAgreementV1.at('0
↳ xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
```

```
# Setting gas limit
network.gas_limit(6700000)
# Delete Subscription
encoded_tx = ida.signatures['deleteSubscription'] + eth_abi.
↳ encode_abi(['address', 'address', 'uint32', 'address', 'bytes',],
↳ (supertoken.address, user.address, 0, user.address, b'',)).hex()
bytes_encoded_tx = to_bytes(encoded_tx[:-64],'bytes')
tx = host.callAgreement(ida.address, bytes_encoded_tx, 0, {'from':
↳  secondUser})
```

Result:

```
>>> # Users to use for the test
user = a[0]
secondUser = a[1]
user1 = '0x291b1056C3C104205d5cf157F173E16a83d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# Delete Subscription
encoded_tx = ida.signatures['deleteSubscription'] + eth_abi.encode_abi(['address', 'address', 'uint32', 'address', 'bytes',], (supertoken.address, user.address, 0, user.address, b'',)).hex()
bytes_encoded_tx = to_bytes(encoded_tx[:-64],'bytes')
tx = host.callAgreement(ida.address, bytes_encoded_tx, 0, {'from': secondUser})
  File "<console>", line 20, in <module>
  File "brownie/network/contract.py", line 1629, in __call__
    return self.transact(*args)
  File "brownie/network/contract.py", line 1502, in transact
    return tx["from"].transfer(
  File "brownie/network/account.py", line 644, in transfer
    receipt, exc = self._make_transaction(
  File "brownie/network/account.py", line 752, in _make_transaction
    exc = VirtualMachineError(e)
  File "brownie/exceptions.py", line 121, in __init__
    raise ValueError(str(exc)) from None
ValueError: Execution reverted during call: 'execution reverted: IDA: E_NOT_ALLOWED'. This transaction will likely revert. If you wish to broadcast, include `allow_revert:True` as a transact
```

4.4 Transparent OnGoing Auction

- After implementing the TOGA contract and setting a user as the PIC of a Super Token, this test case attempts to change the output rate of another user who is not the current PIC of the Super Token. Since the function that makes this change correctly checks whether the sender of the transaction is the current PIC, this is not allowed.

Proof of Concept:

PoC-1:

Listing 14: Proof of Concept using Brownie

```
# Users to use for the test
user = a[0]
secondUser = a[1]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory
↳ ())
ida = InstantDistributionAgreementV1.at('0
↳ xAD1e87F0C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0
↳ xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# <TokenCustodian Contract '0
↳ x3f22c41db59D4ae306bd8F1b5b29dFB7e8629969'>
tokenCustodian = TokenCustodian.at('0
↳ x3f22c41db59D4ae306bd8F1b5b29dFB7e8629969')
# <TOGA Contract '0xc8865b0a4f7eDa5B6E8858e6ED3f780952A14D07'>
toga = TOGA.at('0xc8865b0a4f7eDa5B6E8858e6ED3f780952A14D07')
# Change exit rate
tx = toga.changeExitRate(supertoken, 10, {'from': secondUser})
```

Result:

```
>>> # Users to use for the test
user = a[0]
secondUser = a[1]
user1 = '0x291b1056C3C104205d5cf157F173E16aB3d1965f'
user2 = '0xe8603a92664aE818BF8c31920786AFB61C6CC7dc'
# fUSDCx address
fUSDCx = '0x2dC36872a445adF0bFf63cc0eeee52A2b801625f'
# Import all contracts needed from the Ropsten network
host = Superfluid.at('0xF2B4E81ba39F5215Db2e05B2F66f482BB8e87FD2')
gov = SuperfluidGovernanceII.at(host.getGovernance())
superTokenFactory = SuperTokenFactory.at(host.getSuperTokenFactory())
ida = InstantDistributionAgreementV1.at('0x4D1e87F8C74341ecAFc1d27349dD6e650f5bAdD7')
cfa = ConstantFlowAgreementV1.at('0xaD2F1f7cd663f6a15742675f975CcBD42bb23a88')
supertoken = SuperToken.at(fUSDCx)
# Setting gas limit
network.gas_limit(6700000)
# <TokenCustodian Contract '0x3f22c41db59D4ae306bd8F1b5b29dFB7e8629969'>
tokenCustodian = TokenCustodian.at('0x3f22c41db59D4ae306bd8F1b5b29dFB7e8629969')
# <TOGA Contract '0xc8865b0a4f7eDa5B6E8858e6ED3f780952A14D07'>
toga = TOGA.at('0xc8865b0a4f7eDa5B6E8858e6ED3f780952A14D07')
# Change exit rate
tx = toga.changeExitRate(supertoken, 10, {'from': secondUser})
  File "<console>", line 22, in <module>
  File "brownie/network/contract.py", line 1629, in __call__
    return self.transact(*args)
  File "brownie/network/contract.py", line 1502, in transact
    return tx["from"].transfer(
  File "brownie/network/account.py", line 644, in transfer
    receipt, exc = self._make_transaction(
  File "brownie/network/account.py", line 752, in _make_transaction
    exc = VirtualMachineError(e)
  File "brownie/exceptions.py", line 121, in __init__
    raise ValueError(str(exc)) from None
ValueError: Execution reverted during call: 'execution reverted: TOGA: only PIC allowed'. This transaction will likely revert. If you wish to broadcast, include `allow_revert:True` as a
```

AUTOMATED TESTING

5.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

ConstantFlowAgreementV1.sol AgreementBase.sol

```
ConstantFlowAgreementV1._changeFlowToApp(address,ISuperfluidToken,ConstantFlowAgreementV1.FlowParams,ConstantFlowAgreementV1.FlowData,bytes,ISuperfluid.Context,ConstantFlowAgreementV1.FlowChangeType).vars (contracts/agreements/ConstantFlowAgreementV1.sol#581) is a local variable never initialized
ConstantFlowAgreementV1.updateFlow(ISuperfluidToken,address,int96,bytes).flowParams (contracts/agreements/ConstantFlowAgreementV1.sol#168) is a local variable never initialized
ConstantFlowAgreementV1.createFlow(ISuperfluidToken,address,int96,bytes).flowParams (contracts/agreements/ConstantFlowAgreementV1.sol#129) is a local variable never initialized
ConstantFlowAgreementV1.deleteFlow(ISuperfluidToken,address,address,bytes).flowParams (contracts/agreements/ConstantFlowAgreementV1.sol#206) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Reentrancy in ConstantFlowAgreementV1._changeFlow(uint256,ISuperfluidToken,ISuperfluidToken,ConstantFlowAgreementV1.FlowParams,ConstantFlowAgreementV1.FlowData) (contracts/agreements/ConstantFlowAgreementV1.sol#645-746):
        External calls:
        - token.updateAgreementData(flowParams.flowId,_encodeFlowData(newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#716)
        - totalSenderFlowRate = _updateAccountFlowState(token,flowParams.sender,oldFlowData.flowRate.sub(flowParams.flowRate,CFA: flowrate overflow),depositDelta,0,currentTimestamp) (contracts/agreements/ConstantFlowAgreementV1.sol#720-727)
                - token.settleBalance(account,dynamicBalance) (contracts/agreements/ConstantFlowAgreementV1.sol#428)
                - token.updateAgreementStateSlot(account,0,_encodeFlowData(state)) (contracts/agreements/ConstantFlowAgreementV1.sol#435)
        - totalReceiverFlowRate = _updateAccountFlowState(token,flowParams.receiver,flowParams.flowRate.sub(oldFlowData.flowRate,CFA: flowrate overflow),0,0,currentTimestamp) (contracts/agreements/ConstantFlowAgreementV1.sol#728-735)
                - token.settleBalance(account,dynamicBalance) (contracts/agreements/ConstantFlowAgreementV1.sol#428)
                - token.updateAgreementStateSlot(account,0,_encodeFlowData(state)) (contracts/agreements/ConstantFlowAgreementV1.sol#435)
        Event emitted after the call(s):
        - FlowUpdated(token,flowParams.sender,flowParams.receiver,flowParams.flowRate,totalSenderFlowRate,totalReceiverFlowRate,flowParams.userData) (contracts/agreements/ConstantFlowAgreementV1.sol#738-745)
Reentrancy in ConstantFlowAgreementV1._changeFlowToApp(address,ISuperfluidToken,ConstantFlowAgreementV1.FlowParams,ConstantFlowAgreementV1.FlowData,bytes,ISuperfluid.Context,ConstantFlowAgreementV1.FlowChangeType) (contracts/agreements/ConstantFlowAgreementV1.sol#487-636):
        External calls:
        - vars.cbdata = AgreementLibrary.callAppBeforeCallback(cbStates,ctx) (contracts/agreements/ConstantFlowAgreementV1.sol#520)
        - (None,cbStates.appAllowanceGranted,None) = _changeFlow(currentContext.timestamp,currentContext.appAllowanceToken,token,flowParams,oldFlowData) (contracts/agreements/ConstantFlowAgreementV1.sol#522-525)
                - token.settleBalance(account,dynamicBalance) (contracts/agreements/ConstantFlowAgreementV1.sol#428)
                - token.updateAgreementStateSlot(account,0,_encodeFlowData(state)) (contracts/agreements/ConstantFlowAgreementV1.sol#435)
                - token.updateAgreementData(flowParams.flowId,_encodeFlowData(newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#716)
        Event emitted after the call(s):
        - FlowUpdated(token,flowParams.sender,flowParams.receiver,flowParams.flowRate,totalSenderFlowRate,totalReceiverFlowRate,flowParams.userData) (contracts/agreements/ConstantFlowAgreementV1.sol#738-745)
                - (None,cbStates.appAllowanceGranted,None) = _changeFlow(currentContext.timestamp,currentContext.appAllowanceToken,token,flowParams,oldFlowData) (contracts/agreements/ConstantFlowAgreementV1.sol#522-525)
Reentrancy in ConstantFlowAgreementV1.deleteFlow(ISuperfluidToken,address,address,bytes) (contracts/agreements/ConstantFlowAgreementV1.sol#196-208):
        External calls:
        - _makeLiquidationPayouts(token,availableBalance,flowParams,oldFlowData,currentContext.msgSender) (contracts/agreements/ConstantFlowAgreementV1.sol#232-237)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount.toUint256(),0) (contracts/agreements/ConstantFlowAgreementV1.sol#786-792)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount_scope_0.toUint256(),totalRewardLeft.mul(- 1).toUint256()) (contracts/agreements/ConstantFlowAgreementV1.sol#796-802)
        - newCtx = _changeFlowToApp(receiver,token,flowParams,oldFlowData,newCtx,currentContext,FlowChangeType.DELETE_FLOW) (contracts/agreements/ConstantFlowAgreementV1.sol#243-246)
                - appCtx = ISuperfluid(msg.sender).appCallbackPush(ctx,ISuperApp(inputs.account),inputs.appAllowanceGranted,inputs.appAllowanceUsed,inputs.token) (contracts/agreements/AgreementLibrary.sol#180-185)
                - ISuperfluid(msg.sender).appCallbackPop(ctx,appAllowanceUsedDelta) (contracts/agreements/AgreementLibrary.sol#196)
                - token.settleBalance(account,dynamicBalance) (contracts/agreements/ConstantFlowAgreementV1.sol#428)
                - newCtx = ISuperfluid(msg.sender).callAppAfterCallback(ISuperApp(inputs.account),callData,inputs.noopBit == SuperAppDefinitions.AFTER_AGREEMENT_TERMINATED_NOOP,newCtx) (contracts/agreements/AgreementLibrary.sol#134-138)
                - cbdata = ISuperfluid(msg.sender).callAppBeforeCallback(ISuperApp(inputs.account),callData,inputs.noopBit == SuperAppDefinitions.BEFORE_AGREEMENT_TERMINATED_NOOP,appCtx) (contracts/agreements/AgreementLibrary.sol#99-103)
                - token.updateAgreementStateSlot(account,0,_encodeFlowData(state)) (contracts/agreements/ConstantFlowAgreementV1.sol#435)
                - vars.cbdata = AgreementLibrary.callAppBeforeCallback(cbStates,ctx) (contracts/agreements/ConstantFlowAgreementV1.sol#520)
                - token.updateAgreementData(flowParams.flowId,_encodeFlowData(newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#716)
                - (vars.appContext,None) = AgreementLibrary.callAppAfterCallback(cbStates,vars.cbdata,newCtx) (contracts/agreements/ConstantFlowAgreementV1.sol#538)
                - token.updateAgreementData(flowParams.flowId,_encodeFlowData(vars.newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#573)
                - newCtx = ISuperfluid(msg.sender).ctxUseAllowance(ctx,vars.newFlowData.deposit,appAllowanceDelta) (contracts/agreements/ConstantFlowAgreementV1.sol#596-600)
                - newCtx = ISuperfluid(msg.sender).jailApp(newCtx,ISuperApp(flowParams.receiver),SuperAppDefinitions.APP_RULE_NO_CRITICAL_RECEIVER_ACCOUNT) (contracts/agreements/ConstantFlowAgreementV1.sol#612-615)
                - token.settleBalance(flowParams.sender,- userDamageAmount) (contracts/agreements/ConstantFlowAgreementV1.sol#622-625)
                - token.settleBalance(flowParams.receiver,userDamageAmount) (contracts/agreements/ConstantFlowAgreementV1.sol#626-629)
        Event emitted after the call(s):
        - FlowUpdated(token,flowParams.sender,flowParams.receiver,flowParams.flowRate,totalSenderFlowRate,totalReceiverFlowRate,flowParams.userData) (contracts/agreements/ConstantFlowAgreementV1.sol#738-745)
                - newCtx = _changeFlowToApp(receiver,token,flowParams,oldFlowData,newCtx,currentContext,FlowChangeType.DELETE_FLOW) (contracts/agreements/ConstantFlowAgreementV1.sol#243-246)
Reentrancy in ConstantFlowAgreementV1.deleteFlow(ISuperfluidToken,address,address,bytes) (contracts/agreements/ConstantFlowAgreementV1.sol#196-208):
        External calls:
        - _makeLiquidationPayouts(token,availableBalance,flowParams,oldFlowData,currentContext.msgSender) (contracts/agreements/ConstantFlowAgreementV1.sol#232-237)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount.toUint256(),0) (contracts/agreements/ConstantFlowAgreementV1.sol#786-792)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount_scope_0.toUint256(),totalRewardLeft.mul(- 1).toUint256()) (contracts/agreements/ConstantFlowAgreementV1.sol#796-802)
        - newCtx = _changeFlowToNonApp(token,flowParams,oldFlowData,newCtx,currentContext) (contracts/agreements/ConstantFlowAgreementV1.sol#248-250)
                - token.settleBalance(account,dynamicBalance) (contracts/agreements/ConstantFlowAgreementV1.sol#428)
                - newCtx = ISuperfluid(msg.sender).ctxUseAllowance(ctx,newFlowData.deposit,depositDelta) (contracts/agreements/ConstantFlowAgreementV1.sol#466-470)
                - token.updateAgreementStateSlot(account,0,_encodeFlowData(state)) (contracts/agreements/ConstantFlowAgreementV1.sol#435)
                - token.updateAgreementData(flowParams.flowId,_encodeFlowData(newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#716)
        Event emitted after the call(s):
        - FlowUpdated(token,flowParams.sender,flowParams.receiver,flowParams.flowRate,totalSenderFlowRate,totalReceiverFlowRate,flowParams.userData) (contracts/agreements/ConstantFlowAgreementV1.sol#738-745)
                - newCtx = _changeFlowToNonApp(token,flowParams,oldFlowData,newCtx,currentContext) (contracts/agreements/ConstantFlowAgreementV1.sol#248-250)
Reentrancy in ConstantFlowAgreementV1.deleteFlow(ISuperfluidToken,address,address,bytes) (contracts/agreements/ConstantFlowAgreementV1.sol#196-208):
        External calls:
```

```
        - _makeLiquidationPayouts(token,availableBalance,flowParams,oldFlowData,currentContext.msgSender) (contracts/agreements/ConstantFlowAgreementV1.sol#232-237)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount.toUint256(),0) (contracts/agreements/ConstantFlowAgreementV1.sol#786-792)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount_scope_0.toUint256(),totalRewardLeft.mul(-1).toUint256()) (contracts/agreements/ConstantFlowAgreementV1.sol#796-802)
        - newCtx = _changeFlowToApp(sender,token,flowParams,oldFlowData,newCtx,currentContext,FlowChangeType.DELETE_FLOW) (contracts/agreements/ConstantFlowAgreementV1.sol#254-257)
                - appCtx = ISuperfluid(msg.sender).appCallbackPush(ctx,ISuperApp(inputs.account),inputs.appAllowanceGranted,inputs.appAllowanceUsed,inputs.token) (contracts/agreements/AgreementLibrary.sol#180-185)
                - ISuperfluid(msg.sender).appCallbackPop(ctx,appAllowanceUsedDelta) (contracts/agreements/AgreementLibrary.sol#196)
                - token.settleBalance(account,dynamicBalance) (contracts/agreements/ConstantFlowAgreementV1.sol#428)
                - newCtx = ISuperfluid(msg.sender).callAppAfterCallback(ISuperApp(inputs.account),callData,inputs.noopBit == SuperAppDefinitions.AFTER_AGREEMENT_TERMINATED_NOOP,newCtx) (contracts/agreements/AgreementLibrary.sol#134-1
38)
                - cbdata = ISuperfluid(msg.sender).callAppBeforeCallback(ISuperApp(inputs.account),callData,inputs.noopBit == SuperAppDefinitions.BEFORE_AGREEMENT_TERMINATED_NOOP,appCtx) (contracts/agreements/AgreementLibrary.sol#99-
103)
                - token.updateAgreementStateSlot(account,0,_encodeFlowData(state)) (contracts/agreements/ConstantFlowAgreementV1.sol#435)
                - vars.cbdata = AgreementLibrary.callAppBeforeCallback(cbStates,ctx) (contracts/agreements/ConstantFlowAgreementV1.sol#528)
                - token.updateAgreementData(flowParams.flowId,_encodeFlowData(newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#716)
                - (vars.appContext,None) = AgreementLibrary.callAppAfterCallback(cbStates,vars.cbdata,newCtx) (contracts/agreements/ConstantFlowAgreementV1.sol#538)
                - token.updateAgreementData(flowParams.flowId,_encodeFlowData(vars.newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#573)
                - newCtx = ISuperfluid(msg.sender).ctxUseAllowance(ctx,vars.newFlowData.deposit,appAllowanceDelta) (contracts/agreements/ConstantFlowAgreementV1.sol#596-600)
                - newCtx = ISuperfluid(msg.sender).jailApp(newCtx,ISuperApp(flowParams.receiver),SuperAppDefinitions.APP_RULE_NO_CRITICAL_RECEIVER_ACCOUNT) (contracts/agreements/ConstantFlowAgreementV1.sol#612-615)
                - token.settleBalance(flowParams.sender,- userDamageAmount) (contracts/agreements/ConstantFlowAgreementV1.sol#622-625)
                - token.settleBalance(flowParams.receiver,userDamageAmount) (contracts/agreements/ConstantFlowAgreementV1.sol#626-629)
        Event emitted after the call(s):
        - FlowUpdated(token,flowParams.sender,flowParams.receiver,flowParams.flowRate,totalSenderFlowRate,totalReceiverFlowRate,flowParams.userData) (contracts/agreements/ConstantFlowAgreementV1.sol#738-745)
                - newCtx = _changeFlowToApp(sender,token,flowParams,oldFlowData,newCtx,currentContext,FlowChangeType.DELETE_FLOW) (contracts/agreements/ConstantFlowAgreementV1.sol#254-257)
Reentrancy in ConstantFlowAgreementV1.deleteFlow(ISuperfluidToken,address,address,bytes) (contracts/agreements/ConstantFlowAgreementV1.sol#196-288):
        External calls:
        - _makeLiquidationPayouts(token,availableBalance,flowParams,oldFlowData,currentContext.msgSender) (contracts/agreements/ConstantFlowAgreementV1.sol#232-237)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount.toUint256(),0) (contracts/agreements/ConstantFlowAgreementV1.sol#786-792)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount_scope_0.toUint256(),totalRewardLeft.mul(-1).toUint256()) (contracts/agreements/ConstantFlowAgreementV1.sol#796-802)
        - newCtx = _changeFlowToApp(address(0),token,flowParams,oldFlowData,newCtx,currentContext,FlowChangeType.DELETE_FLOW) (contracts/agreements/ConstantFlowAgreementV1.sol#254-262)
                - appCtx = ISuperfluid(msg.sender).appCallbackPush(ctx,ISuperApp(inputs.account),inputs.appAllowanceGranted,inputs.appAllowanceUsed,inputs.token) (contracts/agreements/AgreementLibrary.sol#180-185)
                - ISuperfluid(msg.sender).appCallbackPop(ctx,appAllowanceUsedDelta) (contracts/agreements/AgreementLibrary.sol#196)
                - token.settleBalance(account,dynamicBalance) (contracts/agreements/ConstantFlowAgreementV1.sol#428)
                - newCtx = ISuperfluid(msg.sender).callAppAfterCallback(ISuperApp(inputs.account),callData,inputs.noopBit == SuperAppDefinitions.AFTER_AGREEMENT_TERMINATED_NOOP,newCtx) (contracts/agreements/AgreementLibrary.sol#134-1
38)
                - cbdata = ISuperfluid(msg.sender).callAppBeforeCallback(ISuperApp(inputs.account),callData,inputs.noopBit == SuperAppDefinitions.BEFORE_AGREEMENT_TERMINATED_NOOP,appCtx) (contracts/agreements/AgreementLibrary.sol#99-
103)
                - token.updateAgreementStateSlot(account,0,_encodeFlowData(state)) (contracts/agreements/ConstantFlowAgreementV1.sol#435)
                - vars.cbdata = AgreementLibrary.callAppBeforeCallback(cbStates,ctx) (contracts/agreements/ConstantFlowAgreementV1.sol#528)
                - token.updateAgreementData(flowParams.flowId,_encodeFlowData(newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#716)
                - (vars.appContext,None) = AgreementLibrary.callAppAfterCallback(cbStates,vars.cbdata,newCtx) (contracts/agreements/ConstantFlowAgreementV1.sol#538)
                - token.updateAgreementData(flowParams.flowId,_encodeFlowData(vars.newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#573)
                - newCtx = ISuperfluid(msg.sender).ctxUseAllowance(ctx,vars.newFlowData.deposit,appAllowanceDelta) (contracts/agreements/ConstantFlowAgreementV1.sol#596-600)
                - newCtx = ISuperfluid(msg.sender).jailApp(newCtx,ISuperApp(flowParams.receiver),SuperAppDefinitions.APP_RULE_NO_CRITICAL_RECEIVER_ACCOUNT) (contracts/agreements/ConstantFlowAgreementV1.sol#612-615)
                - token.settleBalance(flowParams.sender,- userDamageAmount) (contracts/agreements/ConstantFlowAgreementV1.sol#622-625)
                - token.settleBalance(flowParams.receiver,userDamageAmount) (contracts/agreements/ConstantFlowAgreementV1.sol#626-629)
        Event emitted after the call(s):
        - FlowUpdated(token,flowParams.sender,flowParams.receiver,flowParams.flowRate,totalSenderFlowRate,totalReceiverFlowRate,flowParams.userData) (contracts/agreements/ConstantFlowAgreementV1.sol#738-745)
                - newCtx = _changeFlowToNonApp(address(0),token,flowParams,oldFlowData,newCtx,currentContext,FlowChangeType.DELETE_FLOW) (contracts/agreements/ConstantFlowAgreementV1.sol#259-262)
Reentrancy in ConstantFlowAgreementV1.deleteFlow(ISuperfluidToken,address,address,bytes) (contracts/agreements/ConstantFlowAgreementV1.sol#196-288):
        External calls:
        - _makeLiquidationPayouts(token,availableBalance,flowParams,oldFlowData,currentContext.msgSender) (contracts/agreements/ConstantFlowAgreementV1.sol#232-237)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount.toUint256(),0) (contracts/agreements/ConstantFlowAgreementV1.sol#786-792)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount_scope_0.toUint256(),totalRewardLeft.mul(-1).toUint256()) (contracts/agreements/ConstantFlowAgreementV1.sol#796-802)
        - newCtx = _changeFlowToNonApp(token,flowParams,oldFlowData,newCtx,currentContext) (contracts/agreements/ConstantFlowAgreementV1.sol#264-266)
                - token.settleBalance(account,dynamicBalance) (contracts/agreements/ConstantFlowAgreementV1.sol#428)
                - newCtx = ISuperfluid(msg.sender).ctxUseAllowance(ctx,newFlowData.deposit,depositDelta) (contracts/agreements/ConstantFlowAgreementV1.sol#466-470)
                - token.updateAgreementStateSlot(account,0,_encodeFlowData(state)) (contracts/agreements/ConstantFlowAgreementV1.sol#435)
                - token.updateAgreementData(flowParams.flowId,_encodeFlowData(newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#716)
        Event emitted after the call(s):
        - FlowUpdated(token,flowParams.sender,flowParams.receiver,flowParams.flowRate,totalSenderFlowRate,totalReceiverFlowRate,flowParams.userData) (contracts/agreements/ConstantFlowAgreementV1.sol#738-745)
                - newCtx = _changeFlowToNonApp(token,flowParams,oldFlowData,newCtx,currentContext) (contracts/agreements/ConstantFlowAgreementV1.sol#264-266)
Reentrancy in ConstantFlowAgreementV1.deleteFlow(ISuperfluidToken,address,address,bytes) (contracts/agreements/ConstantFlowAgreementV1.sol#196-288):
        External calls:
        - _makeLiquidationPayouts(token,availableBalance,flowParams,oldFlowData,currentContext.msgSender) (contracts/agreements/ConstantFlowAgreementV1.sol#232-237)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount.toUint256(),0) (contracts/agreements/ConstantFlowAgreementV1.sol#786-792)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount_scope_0.toUint256(),totalRewardLeft.mul(-1).toUint256()) (contracts/agreements/ConstantFlowAgreementV1.sol#796-802)
        - newCtx = ISuperfluid(msg.sender).jailApp(newCtx,ISuperApp(sender),SuperAppDefinitions.APP_RULE_NO_CRITICAL_SENDER_ACCOUNT) (contracts/agreements/ConstantFlowAgreementV1.sol#271-274)
        - newCtx = _changeFlowToApp(receiver,token,flowParams,oldFlowData,newCtx,currentContext,FlowChangeType.DELETE_FLOW) (contracts/agreements/ConstantFlowAgreementV1.sol#278-281)
                - appCtx = ISuperfluid(msg.sender).appCallbackPush(ctx,ISuperApp(inputs.account),inputs.appAllowanceGranted,inputs.appAllowanceUsed,inputs.token) (contracts/agreements/AgreementLibrary.sol#180-185)
                - ISuperfluid(msg.sender).appCallbackPop(ctx,appAllowanceUsedDelta) (contracts/agreements/AgreementLibrary.sol#196)

                - token.settleBalance(account,dynamicBalance) (contracts/agreements/ConstantFlowAgreementV1.sol#428)
                - newCtx = ISuperfluid(msg.sender).callAppAfterCallback(ISuperApp(inputs.account),callData,inputs.noopBit == SuperAppDefinitions.AFTER_AGREEMENT_TERMINATED_NOOP,newCtx) (contracts/agreements/AgreementLibrary.sol#134-1
38)
                - cbdata = ISuperfluid(msg.sender).callAppBeforeCallback(ISuperApp(inputs.account),callData,inputs.noopBit == SuperAppDefinitions.BEFORE_AGREEMENT_TERMINATED_NOOP,appCtx) (contracts/agreements/AgreementLibrary.sol#99-
103)
                - token.updateAgreementStateSlot(account,0,_encodeFlowData(state)) (contracts/agreements/ConstantFlowAgreementV1.sol#435)
                - vars.cbdata = AgreementLibrary.callAppBeforeCallback(cbStates,ctx) (contracts/agreements/ConstantFlowAgreementV1.sol#528)
                - token.updateAgreementData(flowParams.flowId,_encodeFlowData(newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#716)
                - (vars.appContext,None) = AgreementLibrary.callAppAfterCallback(cbStates,vars.cbdata,newCtx) (contracts/agreements/ConstantFlowAgreementV1.sol#538)
                - token.updateAgreementData(flowParams.flowId,_encodeFlowData(vars.newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#573)
                - newCtx = ISuperfluid(msg.sender).ctxUseAllowance(ctx,vars.newFlowData.deposit,appAllowanceDelta) (contracts/agreements/ConstantFlowAgreementV1.sol#596-600)
                - newCtx = ISuperfluid(msg.sender).jailApp(newCtx,ISuperApp(flowParams.receiver),SuperAppDefinitions.APP_RULE_NO_CRITICAL_RECEIVER_ACCOUNT) (contracts/agreements/ConstantFlowAgreementV1.sol#612-615)
                - token.settleBalance(flowParams.sender,- userDamageAmount) (contracts/agreements/ConstantFlowAgreementV1.sol#622-625)
                - token.settleBalance(flowParams.receiver,userDamageAmount) (contracts/agreements/ConstantFlowAgreementV1.sol#626-629)
        Event emitted after the call(s):
        - FlowUpdated(token,flowParams.sender,flowParams.receiver,flowParams.flowRate,totalSenderFlowRate,totalReceiverFlowRate,flowParams.userData) (contracts/agreements/ConstantFlowAgreementV1.sol#738-745)
                - newCtx = _changeFlowToApp(receiver,token,flowParams,oldFlowData,newCtx,currentContext,FlowChangeType.DELETE_FLOW) (contracts/agreements/ConstantFlowAgreementV1.sol#278-281)
Reentrancy in ConstantFlowAgreementV1.deleteFlow(ISuperfluidToken,address,address,bytes) (contracts/agreements/ConstantFlowAgreementV1.sol#196-288):
        External calls:
        - _makeLiquidationPayouts(token,availableBalance,flowParams,oldFlowData,currentContext.msgSender) (contracts/agreements/ConstantFlowAgreementV1.sol#232-237)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount.toUint256(),0) (contracts/agreements/ConstantFlowAgreementV1.sol#786-792)
                - token.makeLiquidationPayouts(flowParams.flowId,liquidator,flowParams.sender,rewardAmount_scope_0.toUint256(),totalRewardLeft.mul(-1).toUint256()) (contracts/agreements/ConstantFlowAgreementV1.sol#796-802)
        - newCtx = ISuperfluid(msg.sender).jailApp(newCtx,ISuperApp(sender),SuperAppDefinitions.APP_RULE_NO_CRITICAL_SENDER_ACCOUNT) (contracts/agreements/ConstantFlowAgreementV1.sol#271-274)
        - newCtx = _changeFlowToNonApp(token,flowParams,oldFlowData,newCtx,currentContext) (contracts/agreements/ConstantFlowAgreementV1.sol#283-285)
                - token.settleBalance(account,dynamicBalance) (contracts/agreements/ConstantFlowAgreementV1.sol#428)
                - newCtx = ISuperfluid(msg.sender).ctxUseAllowance(ctx,newFlowData.deposit,depositDelta) (contracts/agreements/ConstantFlowAgreementV1.sol#466-470)
                - token.updateAgreementStateSlot(account,0,_encodeFlowData(state)) (contracts/agreements/ConstantFlowAgreementV1.sol#435)
                - token.updateAgreementData(flowParams.flowId,_encodeFlowData(newFlowData)) (contracts/agreements/ConstantFlowAgreementV1.sol#716)
        Event emitted after the call(s):
        - FlowUpdated(token,flowParams.sender,flowParams.receiver,flowParams.flowRate,totalSenderFlowRate,totalReceiverFlowRate,flowParams.userData) (contracts/agreements/ConstantFlowAgreementV1.sol#738-745)
                - newCtx = _changeFlowToNonApp(token,flowParams,oldFlowData,newCtx,currentContext) (contracts/agreements/ConstantFlowAgreementV1.sol#283-285)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
UUPSUtils.implementation() (contracts/upgradability/UUPSUtils.sol#18-22) uses assembly
        - INLINE ASM (contracts/upgradability/UUPSUtils.sol#19-21)
UUPSUtils.setImplementation(address) (contracts/upgradability/UUPSUtils.sol#25-33) uses assembly
        - INLINE ASM (contracts/upgradability/UUPSUtils.sol#26-32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['^0.7.6', '>=0.4.24<0.8.0', '>=0.5.0', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '>=0.7.0', '^0.7.0']
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#3)
        - >=0.4.24<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#4)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#3)
        - 0.7.6 (contracts/agreements/AgreementBase.sol#2)
        - 0.7.6 (contracts/agreements/AgreementLibrary.sol#2)
        - ABIEncoderV2 (contracts/agreements/AgreementLibrary.sol#3)
        - 0.7.6 (contracts/agreements/ConstantFlowAgreementV1.sol#2)
        - >=0.7.0 (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#2)
        - >=0.7.0 (contracts/interfaces/superfluid/Definitions.sol#2)
        - >=0.7.0 (contracts/interfaces/superfluid/ISuperAgreement.sol#2)
        - >=0.7.0 (contracts/interfaces/superfluid/ISuperApp.sol#2)
        - >=0.7.0 (contracts/interfaces/superfluid/ISuperToken.sol#2)
        - >=0.7.0 (contracts/interfaces/superfluid/ISuperTokenFactory.sol#2)
        - >=0.7.0 (contracts/interfaces/superfluid/ISuperfluid.sol#2)
        - ABIEncoderV2 (contracts/interfaces/superfluid/ISuperfluid.sol#4)
        - >=0.7.0 (contracts/interfaces/superfluid/ISuperfluidGovernance.sol#2)
        - >=0.7.0 (contracts/interfaces/superfluid/ISuperfluidToken.sol#2)
        - >=0.5.0 (contracts/interfaces/tokens/ERC20WithTokenInfo.sol#2)
        - >=0.5.0 (contracts/interfaces/tokens/TokenInfo.sol#2)
        - ^0.7.0 (contracts/libs/Int96SafeMath.sol#3)
        - 0.7.6 (contracts/upgradability/UUPSProxiable.sol#2)
```

```
        - 0.7.6 (contracts/upgradability/UUPSUtils.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) is never used and should be removed
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#89-91) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145) is never used and should be removed
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
ContextDefinitions.decodeCallInfo(uint256) (contracts/interfaces/superfluid/Definitions.sol#83-89) is never used and should be removed
ContextDefinitions.encodeCallInfo(uint8,uint8) (contracts/interfaces/superfluid/Definitions.sol#91-96) is never used and should be removed
Initializable._isConstructor() (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#52-54) is never used and should be removed
Int96SafeMath.div(int96,int96,string) (contracts/libs/Int96SafeMath.sol#50-57) is never used and should be removed
SafeCast.toInt128(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#123-126) is never used and should be removed
SafeCast.toInt16(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#177-180) is never used and should be removed
SafeCast.toInt32(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#159-162) is never used and should be removed
SafeCast.toInt64(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#141-144) is never used and should be removed
SafeCast.toInt8(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#195-198) is never used and should be removed
SafeCast.toUint128(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#33-36) is never used and should be removed
SafeCast.toUint16(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#78-81) is never used and should be removed
SafeCast.toUint32(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#63-66) is never used and should be removed
SafeCast.toUint64(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#48-51) is never used and should be removed
SafeCast.toUint8(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#93-96) is never used and should be removed
SafeMath.add(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#85-89) is never used and should be removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#190-193) is never used and should be removed
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#152-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#210-213) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#170-173) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#24-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#60-63) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#70-73) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#45-53) is never used and should be removed
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#35-38) is never used and should be removed
SuperAppDefinitions.getAppLevel(uint256) (contracts/interfaces/superfluid/Definitions.sol#27-29) is never used and should be removed
SuperAppDefinitions.isAppJailed(uint256) (contracts/interfaces/superfluid/Definitions.sol#32-34) is never used and should be removed
SuperfluidGovernanceConfigs.getAppFactoryConfigKey(address) (contracts/interfaces/superfluid/Definitions.sol#186-190) is never used and should be removed
SuperfluidGovernanceConfigs.getAppRegistrationConfigKey(address,string) (contracts/interfaces/superfluid/Definitions.sol#179-184) is never used and should be removed
SuperfluidGovernanceConfigs.getTrustedForwarderConfigKey(address) (contracts/interfaces/superfluid/Definitions.sol#173-177) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3) is too complex
Pragma version^0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#3) is too complex
Pragma version^0.4.24<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#4) is too complex
Pragma version^0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version^0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777.sol#3) is too complex
Pragma version^0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version^0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#3) is too complex
Pragma version^0.7.0 (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/Definitions.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperAgreement.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperApp.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperToken.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperTokenFactory.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperfluid.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperfluidGovernance.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperfluidToken.sol#2) allows old versions
Pragma version^0.5.0 (contracts/interfaces/tokens/ERC20WithTokenInfo.sol#2) allows old versions
Pragma version^0.5.0 (contracts/interfaces/tokens/TokenInfo.sol#2) allows old versions
Pragma version^0.7.0 (contracts/libs/Int96SafeMath.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)

Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable AgreementBase._host (contracts/agreements/AgreementBase.sol#12) is not in mixedCase
Struct ConstantFlowAgreementV1._StackVars_changeFlowToApp (contracts/agreements/ConstantFlowAgreementV1.sol#482-486) is not in CapWords
Constant SuperfluidGovernanceConfigs.CFAv1_LIQUIDATION_PERIOD_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#167-168) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

ConstantFlowAgreementV1 (contracts/agreements/ConstantFlowAgreementV1.sol#24-892) does not implement functions:
        - UUPSProxiable.proxiableUUID() (contracts/upgradability/UUPSProxiable.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

SuperAppDefinitions.APP_LEVEL_FINAL (contracts/interfaces/superfluid/Definitions.sol#22) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_LEVEL_SECOND (contracts/interfaces/superfluid/Definitions.sol#25) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_JAIL_BIT (contracts/interfaces/superfluid/Definitions.sol#31) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AGREEMENT_CALLBACK_NOOP_BITMASKS (contracts/interfaces/superfluid/Definitions.sol#39) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_CREATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#40) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_CREATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#41) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_UPDATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#42) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_UPDATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#43) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_TERMINATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#44) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_TERMINATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#45) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_REGISTRATION_ONLY_IN_CONSTRUCTOR (contracts/interfaces/superfluid/Definitions.sol#51) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_REGISTRATION_FOR_EOA (contracts/interfaces/superfluid/Definitions.sol#52) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_REVERT_ON_TERMINATION_CALLBACK (contracts/interfaces/superfluid/Definitions.sol#53) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_CRITICAL_SENDER_ACCOUNT (contracts/interfaces/superfluid/Definitions.sol#54) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_CRITICAL_RECEIVER_ACCOUNT (contracts/interfaces/superfluid/Definitions.sol#55) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_READONLY (contracts/interfaces/superfluid/Definitions.sol#56) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_NOT_CLEAN (contracts/interfaces/superfluid/Definitions.sol#57) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_MALFORMATED (contracts/interfaces/superfluid/Definitions.sol#58) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_COMPOSITE_APP_IS_NOT_WHITELISTED (contracts/interfaces/superfluid/Definitions.sol#59) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_COMPOSITE_APP_IS_JAILED (contracts/interfaces/superfluid/Definitions.sol#60) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_MAX_APP_LEVEL_REACHED (contracts/interfaces/superfluid/Definitions.sol#61) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
ContextDefinitions.CALL_INFO_CALL_TYPE_AGREEMENT (contracts/interfaces/superfluid/Definitions.sol#79) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
ContextDefinitions.CALL_INFO_CALL_TYPE_APP_ACTION (contracts/interfaces/superfluid/Definitions.sol#80) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
ContextDefinitions.CALL_INFO_CALL_TYPE_APP_CALLBACK (contracts/interfaces/superfluid/Definitions.sol#81) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
SuperfluidGovernanceConfigs.SUPERFLUID_REWARD_ADDRESS_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#164-165) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
SuperfluidGovernanceConfigs.CFAv1_LIQUIDATION_PERIOD_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#167-168) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
SuperfluidGovernanceConfigs.SUPERTOKEN_MINIMUM_DEPOSIT_KEY (contracts/interfaces/superfluid/Definitions.sol#170-171) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

getCodeAddress() should be declared external:
        - UUPSProxiable.getCodeAddress() (contracts/upgradability/UUPSProxiable.sol#15-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

InstandDistributionAgreementV1.sol AgreementBase.sol

```
InstantDistributionAgreementV1.revokeSubscription(ISuperfluidToken,address,uint32,bytes).vars (contracts/agreements/InstantDistributionAgreementV1.sol#413) is a local variable never initialized
InstantDistributionAgreementV1.claim(ISuperfluidToken,address,uint32,address,bytes).vars (contracts/agreements/InstantDistributionAgreementV1.sol#769) is a local variable never initialized
InstantDistributionAgreementV1.updateSubscription(ISuperfluidToken,uint32,address,uint128,bytes).vars (contracts/agreements/InstantDistributionAgreementV1.sol#479) is a local variable never initialized
InstantDistributionAgreementV1.approveSubscription(ISuperfluidToken,address,uint32,bytes).vars (contracts/agreements/InstantDistributionAgreementV1.sol#328) is a local variable never initialized
InstantDistributionAgreementV1.deleteSubscription(ISuperfluidToken,address,uint32,address,bytes).vars (contracts/agreements/InstantDistributionAgreementV1.sol#692) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

InstantDistributionAgreementV1.approveSubscription(ISuperfluidToken,address,uint32,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#319-401) ignores return value by AgreementLibrary.callAppAfterCallback(cbStates,vars.c
bdata,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#374)
InstantDistributionAgreementV1.approveSubscription(ISuperfluidToken,address,uint32,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#319-401) ignores return value by AgreementLibrary.callAppAfterCallback(cbStates,vars.c
bdata,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#395)
InstantDistributionAgreementV1.revokeSubscription(ISuperfluidToken,address,uint32,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#404-466) ignores return value by AgreementLibrary.callAppAfterCallback(cbStates,vars.cb
data,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#462)
InstantDistributionAgreementV1.updateSubscription(ISuperfluidToken,uint32,address,uint128,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#469-580) ignores return value by AgreementLibrary.callAppAfterCallback(cbStates
,vars.cbdata,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#572)
InstantDistributionAgreementV1.updateSubscription(ISuperfluidToken,uint32,address,uint128,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#469-580) ignores return value by AgreementLibrary.callAppAfterCallback(cbStates
,vars.cbdata,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#575)
InstantDistributionAgreementV1.deleteSubscription(ISuperfluidToken,address,uint32,address,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#682-757) ignores return value by AgreementLibrary.callAppAfterCallback(cbStates
,vars.cbdata,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#751)
InstantDistributionAgreementV1.claim(ISuperfluidToken,address,uint32,address,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#759-815) ignores return value by AgreementLibrary.callAppAfterCallback(cbStates,vars.cbdata,
newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#810)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

InstantDistributionAgreementV1._getSubscriptionData(ISuperfluidToken,bytes32) (contracts/agreements/InstantDistributionAgreementV1.sol#993-1011) has external calls inside a loop: adata = token.getAgreementData(address(this),sId,2) (c
ontracts/agreements/InstantDistributionAgreementV1.sol#1000)
InstantDistributionAgreementV1.realtimeBalanceOf(ISuperfluidToken,address,uint256) (contracts/agreements/InstantDistributionAgreementV1.sol#84-132) has external calls inside a loop: iId = token.getAgreementStateSlot(address(this),acc
ount,_SUBSCRIBER_SUB_DATA_STATE_SLOT_ID_START + subId,1)[0] (contracts/agreements/InstantDistributionAgreementV1.sol#112-115)
InstantDistributionAgreementV1._getIndexData(ISuperfluidToken,bytes32) (contracts/agreements/InstantDistributionAgreementV1.sol#908-924) has external calls inside a loop: adata = token.getAgreementData(address(this),iId,2) (contracts
/agreements/InstantDistributionAgreementV1.sol#915)
SlotsBitmapLibrary.findEmptySlotAndFill(ISuperfluidToken,address,uint256,uint256,bytes32) (contracts/agreements/SlotsBitmapLibrary.sol#23-56) has external calls inside a loop: token.updateAgreementStateSlot(account,dataStateSlotIDSta
rt + slotId,slotData) (contracts/agreements/SlotsBitmapLibrary.sol#42-45)
SlotsBitmapLibrary.findEmptySlotAndFill(ISuperfluidToken,address,uint256,uint256,bytes32) (contracts/agreements/SlotsBitmapLibrary.sol#23-56) has external calls inside a loop: token.updateAgreementStateSlot(account,bitmapStateSlotId,
slotData) (contracts/agreements/SlotsBitmapLibrary.sol#48-51)
SlotsBitmapLibrary.listData(ISuperfluidToken,address,uint256,uint256) (contracts/agreements/SlotsBitmapLibrary.sol#79-113) has external calls inside a loop: dataList[nSlots] = token.getAgreementStateSlot(address(this),account,dataSta
teSlotIDStart + slotId,1)[0] (contracts/agreements/SlotsBitmapLibrary.sol#102-105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in InstantDistributionAgreementV1._updateIndex(ISuperfluidToken,address,uint32,bytes32,InstantDistributionAgreementV1.IndexData,uint128,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#250-285):
        External calls:
        - token.settleBalance(publisher,(- int256(newIndexValue - idata.indexValue)).mul(int256(idata.totalUnitsApproved))) (contracts/agreements/InstantDistributionAgreementV1.sol#263-264)
        - _adjustPublisherDeposit(token,publisher,int256(newIndexValue - idata.indexValue).mul(int256(idata.totalUnitsPending))) (contracts/agreements/InstantDistributionAgreementV1.sol#266-267)
                - token.updateAgreementStateSlot(publisher,_PUBLISHER_DEPOSIT_STATE_SLOT_ID,data) (contracts/agreements/InstantDistributionAgreementV1.sol#960-963)
        - token.updateAgreementData(iId,_encodeIndexData(idata)) (contracts/agreements/InstantDistributionAgreementV1.sol#271)
        Event emitted after the call(s):
        - IndexUpdated(token,publisher,indexId,oldIndexValue,newIndexValue,idata.totalUnitsPending,idata.totalUnitsApproved,userData) (contracts/agreements/InstantDistributionAgreementV1.sol#273-281)
Reentrancy in InstantDistributionAgreementV1.approveSubscription(ISuperfluidToken,address,uint32,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#319-401):
        External calls:
        - vars.cbdata = AgreementLibrary.callAppBeforeCallback(cbStates,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#360)
        - vars.sdata.subId = _findAndFillSubsBitmap(token,subscriber,vars.iId) (contracts/agreements/InstantDistributionAgreementV1.sol#370)
                - SlotsBitmapLibrary.findEmptySlotAndFill(token,subscriber,_SUBSCRIBER_SUBS_BITMAP_STATE_SLOT_ID,_SUBSCRIBER_SUB_DATA_STATE_SLOT_ID_START,iId) (contracts/agreements/InstantDistributionAgreementV1.sol#1037-1042)
                - token.updateAgreementStateSlot(account,dataStateSlotIDStart + slotId,slotData) (contracts/agreements/SlotsBitmapLibrary.sol#42-45)
                - token.updateAgreementStateSlot(account,bitmapStateSlotId,slotData) (contracts/agreements/SlotsBitmapLibrary.sol#48-51)
        - token.createAgreement(vars.sId,_encodeSubscriptionData(vars.sdata)) (contracts/agreements/InstantDistributionAgreementV1.sol#371)
        - AgreementLibrary.callAppAfterCallback(cbStates,vars.cbdata,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#374)
        - vars.cbdata = AgreementLibrary.callAppBeforeCallback(cbStates,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#377)
        - token.updateAgreementData(vars.iId,_encodeIndexData(vars.idata)) (contracts/agreements/InstantDistributionAgreementV1.sol#384)
        - _adjustPublisherDeposit(token,publisher,- balanceDelta) (contracts/agreements/InstantDistributionAgreementV1.sol#385)
                - token.updateAgreementStateSlot(publisher,_PUBLISHER_DEPOSIT_STATE_SLOT_ID,data) (contracts/agreements/InstantDistributionAgreementV1.sol#960-963)
        - token.settleBalance(publisher,- balanceDelta) (contracts/agreements/InstantDistributionAgreementV1.sol#386)
        - token.settleBalance(subscriber,balanceDelta) (contracts/agreements/InstantDistributionAgreementV1.sol#389)
        - vars.sdata.subId = _findAndFillSubsBitmap(token,subscriber,vars.iId) (contracts/agreements/InstantDistributionAgreementV1.sol#391)
                - SlotsBitmapLibrary.findEmptySlotAndFill(token,subscriber,_SUBSCRIBER_SUBS_BITMAP_STATE_SLOT_ID,_SUBSCRIBER_SUB_DATA_STATE_SLOT_ID_START,iId) (contracts/agreements/InstantDistributionAgreementV1.sol#1037-1042)
                - token.updateAgreementStateSlot(account,dataStateSlotIDStart + slotId,slotData) (contracts/agreements/SlotsBitmapLibrary.sol#42-45)
                - token.updateAgreementStateSlot(account,bitmapStateSlotId,slotData) (contracts/agreements/SlotsBitmapLibrary.sol#48-51)
        - token.updateAgreementData(vars.sId,_encodeSubscriptionData(vars.sdata)) (contracts/agreements/InstantDistributionAgreementV1.sol#392)
        - AgreementLibrary.callAppAfterCallback(cbStates,vars.cbdata,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#395)
        Event emitted after the call(s):
        - IndexSubscribed(token,publisher,indexId,subscriber,userData) (contracts/agreements/InstantDistributionAgreementV1.sol#399)
        - SubscriptionApproved(token,subscriber,publisher,indexId,userData) (contracts/agreements/InstantDistributionAgreementV1.sol#400)
Reentrancy in InstantDistributionAgreementV1.claim(ISuperfluidToken,address,uint32,address,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#759-815):

        - _adjustPublisherDeposit(token,publisher,- int256(pendingDistribution)) (contracts/agreements/InstantDistributionAgreementV1.sol#798)
                - token.updateAgreementStateSlot(publisher,_PUBLISHER_DEPOSIT_STATE_SLOT_ID,data) (contracts/agreements/InstantDistributionAgreementV1.sol#960-963)
        - token.settleBalance(publisher,- int256(pendingDistribution)) (contracts/agreements/InstantDistributionAgreementV1.sol#799)
        - token.updateAgreementData(vars.sId,_encodeSubscriptionData(vars.sdata)) (contracts/agreements/InstantDistributionAgreementV1.sol#803)
        - token.settleBalance(subscriber,int256(pendingDistribution)) (contracts/agreements/InstantDistributionAgreementV1.sol#804)
        Event emitted after the call(s):
        - IndexDistributionClaimed(token,publisher,indexId,subscriber,pendingDistribution) (contracts/agreements/InstantDistributionAgreementV1.sol#806)
        - SubscriptionDistributionClaimed(token,subscriber,publisher,indexId,pendingDistribution) (contracts/agreements/InstantDistributionAgreementV1.sol#807)
Reentrancy in InstantDistributionAgreementV1.createIndex(ISuperfluidToken,uint32,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#139-158):
        External calls:
        - token.createAgreement(iId,_encodeIndexData(IndexData(0,0,0))) (contracts/agreements/InstantDistributionAgreementV1.sol#152)
        Event emitted after the call(s):
        - IndexCreated(token,publisher,indexId,context.userData) (contracts/agreements/InstantDistributionAgreementV1.sol#154)
Reentrancy in InstantDistributionAgreementV1.deleteSubscription(ISuperfluidToken,address,uint32,address,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#682-757):
        External calls:
        - vars.cbdata = AgreementLibrary.callAppBeforeCallback(cbStates,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#721)
        - token.updateAgreementData(vars.iId,_encodeIndexData(vars.idata)) (contracts/agreements/InstantDistributionAgreementV1.sol#731)
        - _clearSubsBitmap(token,subscriber,vars.sdata.subId) (contracts/agreements/InstantDistributionAgreementV1.sol#735)
                - SlotsBitmapLibrary.clearSlot(token,subscriber,_SUBSCRIBER_SUBS_BITMAP_STATE_SLOT_ID,subId) (contracts/agreements/InstantDistributionAgreementV1.sol#1052-1056)
                - token.updateAgreementStateSlot(account,bitmapStateSlotId,slotData) (contracts/agreements/SlotsBitmapLibrary.sol#73-76)
        - _adjustPublisherDeposit(token,publisher,- balanceDelta) (contracts/agreements/InstantDistributionAgreementV1.sol#740)
                - token.updateAgreementStateSlot(publisher,_PUBLISHER_DEPOSIT_STATE_SLOT_ID,data) (contracts/agreements/InstantDistributionAgreementV1.sol#960-963)
        - token.settleBalance(publisher,- balanceDelta) (contracts/agreements/InstantDistributionAgreementV1.sol#741)
        - token.terminateAgreement(vars.sId,2) (contracts/agreements/InstantDistributionAgreementV1.sol#745)
        - token.settleBalance(subscriber,balanceDelta) (contracts/agreements/InstantDistributionAgreementV1.sol#748)
        - AgreementLibrary.callAppAfterCallback(cbStates,vars.cbdata,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#751)
        Event emitted after the call(s):
        - IndexUnitsUpdated(token,publisher,indexId,subscriber,0,userData) (contracts/agreements/InstantDistributionAgreementV1.sol#755)
        - IndexUnsubscribed(token,publisher,indexId,subscriber,userData) (contracts/agreements/InstantDistributionAgreementV1.sol#753)
        - SubscriptionRevoked(token,subscriber,publisher,indexId,userData) (contracts/agreements/InstantDistributionAgreementV1.sol#754)
        - SubscriptionUnitsUpdated(token,subscriber,publisher,indexId,0,userData) (contracts/agreements/InstantDistributionAgreementV1.sol#756)
Reentrancy in InstantDistributionAgreementV1.revokeSubscription(ISuperfluidToken,address,uint32,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#404-466):
        External calls:
        - vars.cbdata = AgreementLibrary.callAppBeforeCallback(cbStates,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#442)
        - token.updateAgreementData(vars.iId,_encodeIndexData(vars.idata)) (contracts/agreements/InstantDistributionAgreementV1.sol#448)
        - _clearSubsBitmap(token,subscriber,vars.sdata.subId) (contracts/agreements/InstantDistributionAgreementV1.sol#451)
                - SlotsBitmapLibrary.clearSlot(token,subscriber,_SUBSCRIBER_SUBS_BITMAP_STATE_SLOT_ID,subId) (contracts/agreements/InstantDistributionAgreementV1.sol#1052-1056)
                - token.updateAgreementStateSlot(account,bitmapStateSlotId,slotData) (contracts/agreements/SlotsBitmapLibrary.sol#73-76)
        - token.updateAgreementData(vars.sId,_encodeSubscriptionData(vars.sdata)) (contracts/agreements/InstantDistributionAgreementV1.sol#457)
        - token.settleBalance(subscriber,balanceDelta) (contracts/agreements/InstantDistributionAgreementV1.sol#459)
        - AgreementLibrary.callAppAfterCallback(cbStates,vars.cbdata,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#462)
        Event emitted after the call(s):
        - IndexUnsubscribed(token,publisher,indexId,subscriber,userData) (contracts/agreements/InstantDistributionAgreementV1.sol#464)
        - SubscriptionRevoked(token,subscriber,publisher,indexId,userData) (contracts/agreements/InstantDistributionAgreementV1.sol#465)
Reentrancy in InstantDistributionAgreementV1.updateSubscription(ISuperfluidToken,uint32,address,uint128,bytes) (contracts/agreements/InstantDistributionAgreementV1.sol#469-580):
        External calls:
        - vars.cbdata = AgreementLibrary.callAppBeforeCallback(cbStates,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#507)
        - vars.cbdata = AgreementLibrary.callAppBeforeCallback(cbStates,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#510)
        - token.updateAgreementData(vars.iId,_encodeIndexData(vars.idata)) (contracts/agreements/InstantDistributionAgreementV1.sol#523)
        - token.updateAgreementData(vars.iId,_encodeIndexData(vars.idata)) (contracts/agreements/InstantDistributionAgreementV1.sol#533)
        - token.createAgreement(vars.sId,_encodeSubscriptionData(vars.sdata)) (contracts/agreements/InstantDistributionAgreementV1.sol#545)
        - token.updateAgreementData(vars.iId,_encodeIndexData(vars.idata)) (contracts/agreements/InstantDistributionAgreementV1.sol#548)
        - _adjustPublisherDeposit(token,publisher,- balanceDelta) (contracts/agreements/InstantDistributionAgreementV1.sol#555)
                - token.updateAgreementStateSlot(publisher,_PUBLISHER_DEPOSIT_STATE_SLOT_ID,data) (contracts/agreements/InstantDistributionAgreementV1.sol#960-963)
        - token.settleBalance(publisher,- balanceDelta) (contracts/agreements/InstantDistributionAgreementV1.sol#556)
        - token.settleBalance(subscriber,balanceDelta) (contracts/agreements/InstantDistributionAgreementV1.sol#560)
        - token.updateAgreementData(vars.sId,_encodeSubscriptionData(vars.sdata)) (contracts/agreements/InstantDistributionAgreementV1.sol#566)
        - AgreementLibrary.callAppAfterCallback(cbStates,vars.cbdata,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#572)
        - AgreementLibrary.callAppAfterCallback(cbStates,vars.cbdata,newCtx) (contracts/agreements/InstantDistributionAgreementV1.sol#575)
        Event emitted after the call(s):
        - IndexUnitsUpdated(token,publisher,indexId,subscriber,units,userData) (contracts/agreements/InstantDistributionAgreementV1.sol#578)
        - SubscriptionUnitsUpdated(token,subscriber,publisher,indexId,units,userData) (contracts/agreements/InstantDistributionAgreementV1.sol#579)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
SlotsBitmapLibrary.listData(ISuperfluidToken,address,uint256,uint256) (contracts/agreements/SlotsBitmapLibrary.sol#79-113) uses assembly
        - INLINE ASM (contracts/agreements/SlotsBitmapLibrary.sol#109-112)
```

```
UUPSUtils.setImplementation(address) (contracts/upgradability/UUPSUtils.sol#25-33) uses assembly
        - INLINE ASM (contracts/upgradability/UUPSUtils.sol#26-32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.7.6', '>0.4.24<0.8.0', '>0.5.0', '>0.6.0<0.8.0', '>0.6.2<0.8.0', '>0.7.0']
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#3)
        - >0.4.24<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#4)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777.sol#3)
        - >0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#3)
        - 0.7.6 (contracts/agreements/AgreementBase.sol#2)
        - 0.7.6 (contracts/agreements/AgreementLibrary.sol#2)
        - ABIEncoderV2 (contracts/agreements/AgreementLibrary.sol#3)
        - 0.7.6 (contracts/agreements/InstantDistributionAgreementV1.sol#2)
        - 0.7.6 (contracts/agreements/SlotsBitmapLibrary.sol#2)
        - >0.7.0 (contracts/interfaces/agreements/IInstantDistributionAgreementV1.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/Definitions.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperAgreement.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperApp.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperToken.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperTokenFactory.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperfluid.sol#2)
        - ABIEncoderV2 (contracts/interfaces/superfluid/ISuperfluid.sol#4)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperfluidGovernance.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperfluidToken.sol#2)
        - >0.5.0 (contracts/interfaces/tokens/ERC20WithTokenInfo.sol#2)
        - >0.5.0 (contracts/interfaces/tokens/TokenInfo.sol#2)
        - 0.7.6 (contracts/libs/Uint128SafeMath.sol#2)
        - 0.7.6 (contracts/upgradability/UUPSProxiable.sol#2)
        - 0.7.6 (contracts/upgradability/UUPSUtils.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) is never used and should be removed
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#89-91) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145) is never used and should be removed
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
ContextDefinitions.decodeCallInfo(uint256) (contracts/interfaces/superfluid/Definitions.sol#83-89) is never used and should be removed
ContextDefinitions.encodeCallInfo(uint8,uint8) (contracts/interfaces/superfluid/Definitions.sol#91-96) is never used and should be removed
Initializable._isConstructor() (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#52-54) is never used and should be removed
SafeCast.toInt128(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#123-126) is never used and should be removed
SafeCast.toInt16(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#177-180) is never used and should be removed
SafeCast.toInt32(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#159-162) is never used and should be removed
SafeCast.toInt64(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#141-144) is never used and should be removed
SafeCast.toInt8(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#195-198) is never used and should be removed
SafeCast.toUint16(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#78-81) is never used and should be removed
SafeCast.toUint256(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#105-108) is never used and should be removed
SafeCast.toUint32(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#63-66) is never used and should be removed
SafeCast.toUint64(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#48-51) is never used and should be removed
SafeCast.toUint8(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#93-96) is never used and should be removed
SafeMath.add(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#85-89) is never used and should be removed
SafeMath.div(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#135-138) is never used and should be removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#198-203) is never used and should be removed
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#152-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#210-213) is never used and should be removed
SafeMath.sub(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#101-104) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#170-173) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#24-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#60-63) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#70-73) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#45-53) is never used and should be removed

SignedSafeMath.sub(int256,int256) (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#69-74) is never used and should be removed
SuperAppDefinitions.getAppLevel(uint256) (contracts/interfaces/superfluid/Definitions.sol#27-29) is never used and should be removed
SuperAppDefinitions.isAppJailed(uint256) (contracts/interfaces/superfluid/Definitions.sol#32-34) is never used and should be removed
SuperfluidGovernanceConfigs.getAppFactoryConfigKey(address) (contracts/interfaces/superfluid/Definitions.sol#186-198) is never used and should be removed
SuperfluidGovernanceConfigs.getAppRegistrationConfigKey(address,string) (contracts/interfaces/superfluid/Definitions.sol#179-184) is never used and should be removed
SuperfluidGovernanceConfigs.getTrustedForwarderConfigKey(address) (contracts/interfaces/superfluid/Definitions.sol#173-177) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>0.6.4<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3) is too complex
Pragma version>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#3) is too complex
Pragma version>0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#4) is too complex
Pragma version>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777.sol#3) is too complex
Pragma version>0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#3) is too complex
Pragma version>0.7.0 (contracts/interfaces/agreements/IInstantDistributionAgreementV1.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/Definitions.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperAgreement.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperApp.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperToken.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperTokenFactory.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperfluid.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperfluidGovernance.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperfluidToken.sol#2) allows old versions
Pragma version>0.5.0 (contracts/interfaces/tokens/ERC20WithTokenInfo.sol#2) allows old versions
Pragma version>0.5.0 (contracts/interfaces/tokens/TokenInfo.sol#2) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable AgreementBase._host (contracts/agreements/AgreementBase.sol#12) is not in mixedCase
Struct InstantDistributionAgreementV1._SubscriptionOperationVars (contracts/agreements/InstantDistributionAgreementV1.sol#309-316) is not in CapWords
Constant SuperfluidGovernanceConfigs.CFAV1_LIQUIDATION_PERIOD_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#167-168) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

InstantDistributionAgreementV1 (contracts/agreements/InstantDistributionAgreementV1.sol#25-1079) does not implement functions:
        - UUPSProxiable.proxiableUUID() (contracts/upgradability/UUPSProxiable.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

InstantDistributionAgreementV1._MAX_NUM_SUBS (contracts/agreements/InstantDistributionAgreementV1.sol#56) is never used in InstantDistributionAgreementV1 (contracts/agreements/InstantDistributionAgreementV1.sol#25-1079)
SuperAppDefinitions.APP_LEVEL_FINAL (contracts/interfaces/superfluid/Definitions.sol#22) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_LEVEL_SECOND (contracts/interfaces/superfluid/Definitions.sol#25) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_JAIL_BIT (contracts/interfaces/superfluid/Definitions.sol#31) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AGREEMENT_CALLBACK_NOOP_BITMASKS (contracts/interfaces/superfluid/Definitions.sol#39) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_CREATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#40) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_CREATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#41) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_UPDATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#42) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_UPDATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#43) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_TERMINATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#44) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_TERMINATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#45) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_REGISTRATION_ONLY_IN_CONSTRUCTOR (contracts/interfaces/superfluid/Definitions.sol#51) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_REGISTRATION_FOR_EOA (contracts/interfaces/superfluid/Definitions.sol#52) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_REVERT_ON_TERMINATION_CALLBACK (contracts/interfaces/superfluid/Definitions.sol#53) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_CRITICAL_SENDER_ACCOUNT (contracts/interfaces/superfluid/Definitions.sol#54) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_CRITICAL_RECEIVER_ACCOUNT (contracts/interfaces/superfluid/Definitions.sol#55) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_READONLY (contracts/interfaces/superfluid/Definitions.sol#56) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_NOT_CLEAN (contracts/interfaces/superfluid/Definitions.sol#57) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_MALFORMATED (contracts/interfaces/superfluid/Definitions.sol#58) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_COMPOSITE_APP_IS_NOT_WHITELISTED (contracts/interfaces/superfluid/Definitions.sol#59) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_COMPOSITE_APP_IS_JAILED (contracts/interfaces/superfluid/Definitions.sol#60) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_MAX_APP_LEVEL_REACHED (contracts/interfaces/superfluid/Definitions.sol#61) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
ContextDefinitions.CALL_INFO_CALL_TYPE_AGREEMENT (contracts/interfaces/superfluid/Definitions.sol#79) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)

ContextDefinitions.CALL_INFO_CALL_TYPE_APP_ACTION (contracts/interfaces/superfluid/Definitions.sol#80) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
ContextDefinitions.CALL_INFO_CALL_TYPE_APP_CALLBACK (contracts/interfaces/superfluid/Definitions.sol#81) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
SuperfluidGovernanceConfigs.SUPERFLUID_REWARD_ADDRESS_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#164-165) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
SuperfluidGovernanceConfigs.CFAV1_LIQUIDATION_PERIOD_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#167-168) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
SuperfluidGovernanceConfigs.SUPERTOKEN_MINIMUM_DEPOSIT_KEY (contracts/interfaces/superfluid/Definitions.sol#170-171) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

findEmptySlotAndFill(ISuperfluidToken,address,uint256,uint256,bytes32) should be declared external:
        - SlotsBitmapLibrary.findEmptySlotAndFill(ISuperfluidToken,address,uint256,uint256,bytes32) (contracts/agreements/SlotsBitmapLibrary.sol#23-56)
clearSlot(ISuperfluidToken,address,uint256,uint32) should be declared external:
        - SlotsBitmapLibrary.clearSlot(ISuperfluidToken,address,uint256,uint32) (contracts/agreements/SlotsBitmapLibrary.sol#58-77)
listData(ISuperfluidToken,address,uint256,uint256) should be declared external:
        - SlotsBitmapLibrary.listData(ISuperfluidToken,address,uint256,uint256) (contracts/agreements/SlotsBitmapLibrary.sol#79-113)
getCodeAddress() should be declared external:
        - UUPSProxiable.getCodeAddress() (contracts/upgradability/UUPSProxiable.sol#15-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Superfluid.sol

```
Superfluid.mapAgreementClasses(uint256) (contracts/superfluid/Superfluid.sol#193-209) contains an incorrect shift operation: (bitmap & (1 << i)) > 0 (contracts/superfluid/Superfluid.sol#203)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

Superfluid._isCtxValid(bytes) (contracts/superfluid/Superfluid.sol#913-915) uses a dangerous strict equality:
        - ctx.length == 0 && keccak256(bytes)(ctx) == _ctxStamp (contracts/superfluid/Superfluid.sol#914)
Superfluid.cleanCtx() (contracts/superfluid/Superfluid.sol#1018-1021) uses a dangerous strict equality:
        - require(bool,string)(_ctxStamp == 0,SF: APP_RULE_CTX_IS_NOT_CLEAN) (contracts/superfluid/Superfluid.sol#1019)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Contract locking ether found:
        Contract UUPSProxy (contracts/upgradability/UUPSProxy.sol#18-37) has payable functions:
         - Proxy.fallback() (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#63-65)
         - Proxy.receive() (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#71-73)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

Reentrancy in Superfluid._callAgreement(address,ISuperAgreement,bytes,bytes) (contracts/superfluid/Superfluid.sol#564-600):
        External calls:
        - (success,returnedData) = _callExternalWithReplacedCtx(address(agreementClass),callData,ctx) (contracts/superfluid/Superfluid.sol#594)
                - (success,returnedData) = target.call(callData) (contracts/superfluid/Superfluid.sol#933)
        State variables written after the call(s):
        - _ctxStamp = 0 (contracts/superfluid/Superfluid.sol#599)
Reentrancy in Superfluid._callAppAction(address,ISuperApp,bytes) (contracts/superfluid/Superfluid.sol#613-648):
        External calls:
        - (success,returnedData) = _callExternalWithReplacedCtx(address(app),callData,ctx) (contracts/superfluid/Superfluid.sol#639)
                - (success,returnedData) = target.call(callData) (contracts/superfluid/Superfluid.sol#933)
        State variables written after the call(s):
        - _ctxStamp = 0 (contracts/superfluid/Superfluid.sol#647)
Reentrancy in Superfluid.callAgreementWithContext(ISuperAgreement,bytes,bytes,bytes) (contracts/superfluid/Superfluid.sol#666-698):
        External calls:
        - (success,returnedData) = _callExternalWithReplacedCtx(address(agreementClass),callData,newCtx) (contracts/superfluid/Superfluid.sol#687)
                - (success,returnedData) = target.call(callData) (contracts/superfluid/Superfluid.sol#933)
        State variables written after the call(s):
        - newCtx = _updateContext(context) (contracts/superfluid/Superfluid.sol#694)
                - _ctxStamp = keccak256(bytes)(ctx) (contracts/superfluid/Superfluid.sol#870)
Reentrancy in Superfluid.callAppActionWithContext(ISuperApp,bytes,bytes) (contracts/superfluid/Superfluid.sol#700-728):
        External calls:
        - (success,returnedData) = _callExternalWithReplacedCtx(address(app),callData,newCtx) (contracts/superfluid/Superfluid.sol#717)
                - (success,returnedData) = target.call(callData) (contracts/superfluid/Superfluid.sol#933)
        State variables written after the call(s):
        - newCtx = _updateContext(context) (contracts/superfluid/Superfluid.sol#724)
                - _ctxStamp = keccak256(bytes)(ctx) (contracts/superfluid/Superfluid.sol#870)
Reentrancy in Superfluid.callAppAfterCallback(ISuperApp,bytes,bool,bytes) (contracts/superfluid/Superfluid.sol#455-489):
        External calls:
        - (success,returnedData) = _callCallback(app,false,isTermination,callData,ctx) (contracts/superfluid/Superfluid.sol#465)
                - (success,returnedData) = address(app).staticcall{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#957)
                - (success,returnedData) = address(app).call{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#960)
        State variables written after the call(s):
        - _jailApp(app,SuperAppDefinitions.APP_RULE_CTX_IS_READONLY) (contracts/superfluid/Superfluid.sol#475)
                - _appManifests[app].configWord |= SuperAppDefinitions.APP_JAIL_BIT (contracts/superfluid/Superfluid.sol#841)
        - _jailApp(app,SuperAppDefinitions.APP_RULE_CTX_IS_MALFORMATED) (contracts/superfluid/Superfluid.sol#483)
                - _appManifests[app].configWord |= SuperAppDefinitions.APP_JAIL_BIT (contracts/superfluid/Superfluid.sol#841)
Reentrancy in Superfluid.callAppBeforeCallback(ISuperApp,bytes,bool,bytes) (contracts/superfluid/Superfluid.sol#431-453):
        External calls:
        - (success,returnedData) = _callCallback(app,true,isTermination,callData,ctx) (contracts/superfluid/Superfluid.sol#441)
                - (success,returnedData) = address(app).staticcall{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#957)
                - (success,returnedData) = address(app).call{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#960)
        State variables written after the call(s):
        - _jailApp(app,SuperAppDefinitions.APP_RULE_CTX_IS_MALFORMATED) (contracts/superfluid/Superfluid.sol#449)
                - _appManifests[app].configWord |= SuperAppDefinitions.APP_JAIL_BIT (contracts/superfluid/Superfluid.sol#841)
Reentrancy in Superfluid.registerAgreementClass(ISuperAgreement) (contracts/superfluid/Superfluid.sol#135-154):
        External calls:
        - proxy.initializeProxy(address(agreementClassLogic)) (contracts/superfluid/Superfluid.sol#145)
        State variables written after the call(s):
        - _agreementClassIndices[agreementType] = _agreementClasses.length (contracts/superfluid/Superfluid.sol#152)
        - _agreementClasses.push((agreementClass)) (contracts/superfluid/Superfluid.sol#151)
Reentrancy in Superfluid.updateSuperTokenFactory(ISuperTokenFactory) (contracts/superfluid/Superfluid.sol#249-268):
        External calls:
        - proxy.initializeProxy(address(newFactory)) (contracts/superfluid/Superfluid.sol#257)
        State variables written after the call(s):

        - _superTokenFactory = ISuperTokenFactory(address(proxy)) (contracts/superfluid/Superfluid.sol#258)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Superfluid._batchCall(address,ISuperfluid.Operation[]).amount_scope_0 (contracts/superfluid/Superfluid.sol#764) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Variable 'Superfluid._batchCall(address,ISuperfluid.Operation[]).amount (contracts/superfluid/Superfluid.sol#757)' in Superfluid._batchCall(address,ISuperfluid.Operation[]) (contracts/superfluid/Superfluid.sol#748-795) potentially us
ed before declaration: (sender,receiver,amount) = abi.decode(operations[i].data,(address,address,uint256)) (contracts/superfluid/Superfluid.sol#764-765)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in Superfluid._callCallback(ISuperApp,bool,bool,bytes,bytes) (contracts/superfluid/Superfluid.sol#940-980):
        External calls:
        - (success,returnedData) = address(app).staticcall{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#957)
        - (success,returnedData) = address(app).call{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#960)
        State variables written after the call(s):
        - _jailApp(app,SuperAppDefinitions.APP_RULE_NO_REVERT_ON_TERMINATION_CALLBACK) (contracts/superfluid/Superfluid.sol#971)
                - _appManifests[app].configWord |= SuperAppDefinitions.APP_JAIL_BIT (contracts/superfluid/Superfluid.sol#841)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Superfluid._callCallback(ISuperApp,bool,bool,bytes,bytes) (contracts/superfluid/Superfluid.sol#940-980):
        External calls:
        - (success,returnedData) = address(app).staticcall{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#957)
        - (success,returnedData) = address(app).call{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#960)
        Event emitted after the call(s):
        - Jail(app,reason) (contracts/superfluid/Superfluid.sol#842)
                - _jailApp(app,SuperAppDefinitions.APP_RULE_NO_REVERT_ON_TERMINATION_CALLBACK) (contracts/superfluid/Superfluid.sol#971)
Reentrancy in Superfluid.callAppAfterCallback(ISuperApp,bytes,bool,bytes) (contracts/superfluid/Superfluid.sol#455-489):
        External calls:
        - (success,returnedData) = _callCallback(app,false,isTermination,callData,ctx) (contracts/superfluid/Superfluid.sol#465)
                - (success,returnedData) = address(app).staticcall{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#957)
                - (success,returnedData) = address(app).call{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#960)
        Event emitted after the call(s):
        - Jail(app,reason) (contracts/superfluid/Superfluid.sol#842)
                - _jailApp(app,SuperAppDefinitions.APP_RULE_CTX_IS_MALFORMATED) (contracts/superfluid/Superfluid.sol#483)
        - Jail(app,reason) (contracts/superfluid/Superfluid.sol#842)
                - _jailApp(app,SuperAppDefinitions.APP_RULE_CTX_IS_READONLY) (contracts/superfluid/Superfluid.sol#475)
Reentrancy in Superfluid.callAppBeforeCallback(ISuperApp,bytes,bool,bytes) (contracts/superfluid/Superfluid.sol#431-453):
        External calls:
        - (success,returnedData) = _callCallback(app,true,isTermination,callData,ctx) (contracts/superfluid/Superfluid.sol#441)
                - (success,returnedData) = address(app).staticcall{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#957)
                - (success,returnedData) = address(app).call{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#960)
        Event emitted after the call(s):
        - Jail(app,reason) (contracts/superfluid/Superfluid.sol#842)
                - _jailApp(app,SuperAppDefinitions.APP_RULE_CTX_IS_MALFORMATED) (contracts/superfluid/Superfluid.sol#449)
Reentrancy in Superfluid.registerAgreementClass(ISuperAgreement) (contracts/superfluid/Superfluid.sol#135-154):
        External calls:
        - proxy.initializeProxy(address(agreementClassLogic)) (contracts/superfluid/Superfluid.sol#145)
        Event emitted after the call(s):
        - AgreementClassRegistered(agreementType,address(agreementClassLogic)) (contracts/superfluid/Superfluid.sol#153)
Reentrancy in Superfluid.updateAgreementClass(ISuperAgreement) (contracts/superfluid/Superfluid.sol#156-164):
        External calls:
        - proxiable.updateCode(address(agreementClassLogic)) (contracts/superfluid/Superfluid.sol#162)
        Event emitted after the call(s):
        - AgreementClassUpdated(agreementType,address(agreementClassLogic)) (contracts/superfluid/Superfluid.sol#163)
Reentrancy in Superfluid.updateSuperTokenFactory(ISuperTokenFactory) (contracts/superfluid/Superfluid.sol#249-268):
        External calls:
        - proxy.initializeProxy(address(newFactory)) (contracts/superfluid/Superfluid.sol#257)
        - _superTokenFactory.initialize() (contracts/superfluid/Superfluid.sol#262)
        - UUPSProxiable(address(_superTokenFactory)).updateCode(address(newFactory)) (contracts/superfluid/Superfluid.sol#265)
        Event emitted after the call(s):
        - SuperTokenFactoryUpdated(_superTokenFactory) (contracts/superfluid/Superfluid.sol#267)
Reentrancy in Superfluid.updateSuperTokenLogic(ISuperToken) (contracts/superfluid/Superfluid.sol#270-278):
        External calls:
        - UUPSProxiable(address(token)).updateCode(code) (contracts/superfluid/Superfluid.sol#276)
        Event emitted after the call(s):
        - SuperTokenLogicUpdated(token,code) (contracts/superfluid/Superfluid.sol#277)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Superfluid._updateContext(ISuperfluid.Context) (contracts/superfluid/Superfluid.sol#846-871) uses timestamp for comparisons
        Dangerous comparisons:
```

```
        - require(bool,string)(context.appLevel < MAX_APP_LEVEL_5F: APP_RULE_MAX_APP_LEVEL_REACHED) (contracts/superfluid/Superfluid.sol#850)
Superfluid._isCtxValid(bytes) (contracts/superfluid/Superfluid.sol#913-915) uses timestamp for comparisons
        Dangerous comparisons:
        - ctx.length == 0 && keccak256(bytes)(ctx) == _ctxStamp (contracts/superfluid/Superfluid.sol#914)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Proxy._delegate(address) (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#21-41) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#23-40)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
CallUtils.getRevertMsg(bytes) (contracts/libs/CallUtils.sol#14-23) uses assembly
        - INLINE ASM (contracts/libs/CallUtils.sol#18-21)
CallUtils.parseSelector(bytes) (contracts/libs/CallUtils.sol#31-37) uses assembly
        - INLINE ASM (contracts/libs/CallUtils.sol#34-36)
CallUtils.isValidAbiEncodedBytes(bytes) (contracts/libs/CallUtils.sol#53-63) uses assembly
        - INLINE ASM (contracts/libs/CallUtils.sol#58)
        - INLINE ASM (contracts/libs/CallUtils.sol#60)
Superfluid.mapAgreementClasses(uint256) (contracts/superfluid/Superfluid.sol#193-209) uses assembly
        - INLINE ASM (contracts/superfluid/Superfluid.sol#208)
Superfluid.registerAppByFactory(ISuperApp,uint256) (contracts/superfluid/Superfluid.sol#322-347) uses assembly
        - INLINE ASM (contracts/superfluid/Superfluid.sol#332)
Superfluid._registerApp(uint256,ISuperApp,bool) (contracts/superfluid/Superfluid.sol#349-368) uses assembly
        - INLINE ASM (contracts/superfluid/Superfluid.sol#357)
Superfluid._replacePlaceholderCtx(bytes,bytes) (contracts/superfluid/Superfluid.sol#985-1011) uses assembly
        - INLINE ASM (contracts/superfluid/Superfluid.sol#996)
        - INLINE ASM (contracts/superfluid/Superfluid.sol#1002)
UUPSUtils.implementation() (contracts/upgradability/UUPSUtils.sol#18-22) uses assembly
        - INLINE ASM (contracts/upgradability/UUPSUtils.sol#19-21)
UUPSUtils.setImplementation(address) (contracts/upgradability/UUPSUtils.sol#25-33) uses assembly
        - INLINE ASM (contracts/upgradability/UUPSUtils.sol#26-32)
BaseRelayRecipient._getTransactionSigner() (contracts/ux/BaseRelayRecipient.sol#26-34) uses assembly
        - INLINE ASM (contracts/ux/BaseRelayRecipient.sol#31-33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.7.6', '>0.4.24<0.8.0', '>0.5.0', '>0.6.0<0.8.0', '>0.6.2<0.8.0', '>0.7.0']
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#3)
        - >0.4.24<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#4)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777.sol#3)
        - >0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#3)
        - >0.7.0 (contracts/interfaces/superfluid/Definitions.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperAgreement.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperApp.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperToken.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperTokenFactory.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperfluid.sol#2)
        - ABIEncoderV2 (contracts/interfaces/superfluid/ISuperfluid.sol#4)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperfluidGovernance.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperfluidToken.sol#2)
        - >0.5.0 (contracts/interfaces/tokens/ERC20WithTokenInfo.sol#2)
        - >0.5.0 (contracts/interfaces/tokens/TokenInfo.sol#2)
        - >0.7.0 (contracts/interfaces/ux/IRelayRecipient.sol#2)
        - 0.7.6 (contracts/libs/CallUtils.sol#2)
        - 0.7.6 (contracts/superfluid/Superfluid.sol#2)
        - ABIEncoderV2 (contracts/superfluid/Superfluid.sol#3)
        - 0.7.6 (contracts/upgradability/UUPSProxiable.sol#2)
        - 0.7.6 (contracts/upgradability/UUPSProxy.sol#2)
        - 0.7.6 (contracts/upgradability/UUPSUtils.sol#2)
        - 0.7.6 (contracts/ux/BaseRelayRecipient.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) is never used and should be removed
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#89-91) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#129-132) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
Proxy._implementation() (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#47) is never used and should be removed
SafeCast.toInt128(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#123-126) is never used and should be removed
SafeCast.toInt16(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#177-180) is never used and should be removed
SafeCast.toInt256(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#207-210) is never used and should be removed
SafeCast.toInt32(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#159-162) is never used and should be removed
SafeCast.toInt64(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#141-144) is never used and should be removed
SafeCast.toInt8(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#195-198) is never used and should be removed
SafeCast.toUint16(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#78-81) is never used and should be removed
SafeCast.toUint256(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#185-188) is never used and should be removed
SafeCast.toUint32(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#63-66) is never used and should be removed
SafeCast.toUint64(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#48-51) is never used and should be removed
SafeCast.toUint8(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#93-96) is never used and should be removed
SafeMath.div(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#135-138) is never used and should be removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#190-193) is never used and should be removed
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#152-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#210-213) is never used and should be removed
SafeMath.mul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#116-121) is never used and should be removed
SafeMath.sub(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#101-104) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#170-173) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#24-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#60-63) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#70-73) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#45-53) is never used and should be removed
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#35-38) is never used and should be removed
SignedSafeMath.div(int256,int256) (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#50-57) is never used and should be removed
SignedSafeMath.mul(int256,int256) (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#22-36) is never used and should be removed
SignedSafeMath.sub(int256,int256) (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#69-74) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3) is too complex
Pragma version>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#3) is too complex
Pragma version>0.4.24<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#4) is too complex
Pragma version>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#3) is too complex
Pragma version>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777.sol#3) is too complex
Pragma version>0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#3) is too complex
Pragma version>0.7.0 (contracts/interfaces/superfluid/Definitions.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperAgreement.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperApp.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperToken.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperTokenFactory.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperfluid.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperfluidGovernance.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/superfluid/ISuperfluidToken.sol#2) allows old versions
Pragma version>0.5.0 (contracts/interfaces/tokens/ERC20WithTokenInfo.sol#2) allows old versions
Pragma version>0.5.0 (contracts/interfaces/tokens/TokenInfo.sol#2) allows old versions
Pragma version>0.7.0 (contracts/interfaces/ux/IRelayRecipient.sol#2) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#167)
Low level call in Superfluid._callExternalWithReplacedCtx(address,bytes,bytes) (contracts/superfluid/Superfluid.sol#917-938):
        - (success,returnedData) = target.call(callData) (contracts/superfluid/Superfluid.sol#933)
Low level call in Superfluid._callCallback(ISuperApp,bool,bool,bytes,bytes) (contracts/superfluid/Superfluid.sol#948-960):
        - (success,returnedData) = address(app).staticcall{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#957)
        - (success,returnedData) = address(app).call{gas: CALLBACK_GAS_LIMIT}(callData) (contracts/superfluid/Superfluid.sol#960)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Constant SuperfluidGovernanceConfigs.CFAV1_LIQUIDATION_PERIOD_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#167-168) is not in UPPER_CASE_WITH_UNDERSCORES
Variable Superfluid.NON_UPGRADABLE_DEPLOYMENT (contracts/superfluid/Superfluid.sol#47) is not in mixedCase
Variable Superfluid.APP_WHITE_LISTING_ENABLED (contracts/superfluid/Superfluid.sol#50) is not in mixedCase
Variable Superfluid.MAX_APP_LEVEL (contracts/superfluid/Superfluid.sol#60) is not in mixedCase
Variable Superfluid.CALLBACK_GAS_LIMIT (contracts/superfluid/Superfluid.sol#63) is not in mixedCase
Variable Superfluid._gov (contracts/superfluid/Superfluid.sol#70) is not in mixedCase
Variable Superfluid._agreementClasses (contracts/superfluid/Superfluid.sol#73) is not in mixedCase
Variable Superfluid._agreementClassIndices (contracts/superfluid/Superfluid.sol#75) is not in mixedCase
Variable Superfluid._superTokenFactory (contracts/superfluid/Superfluid.sol#78) is not in mixedCase
Variable Superfluid._appManifests (contracts/superfluid/Superfluid.sol#81) is not in mixedCase
Variable Superfluid._compositeApps (contracts/superfluid/Superfluid.sol#83) is not in mixedCase
Variable Superfluid._ctxStamp (contracts/superfluid/Superfluid.sol#86) is not in mixedCase
Variable Superfluid._appKeysUsed (contracts/superfluid/Superfluid.sol#88) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Superfluid.slitherConstructorVariables() (contracts/superfluid/Superfluid.sol#32-1044) uses literals with too many digits:
        - CALLBACK_GAS_LIMIT = 3000000 (contracts/superfluid/Superfluid.sol#63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

SuperAppDefinitions.APP_LEVEL_FINAL (contracts/interfaces/superfluid/Definitions.sol#22) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_LEVEL_SECOND (contracts/interfaces/superfluid/Definitions.sol#25) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_JAIL_BIT (contracts/interfaces/superfluid/Definitions.sol#31) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AGREEMENT_CALLBACK_NOOP_BITMASKS (contracts/interfaces/superfluid/Definitions.sol#39) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_CREATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#40) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_CREATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#41) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_UPDATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#42) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_UPDATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#43) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_TERMINATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#44) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_TERMINATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#45) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_REGISTRATION_ONLY_IN_CONSTRUCTOR (contracts/interfaces/superfluid/Definitions.sol#51) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_REGISTRATION_FOR_EOA (contracts/interfaces/superfluid/Definitions.sol#52) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_REVERT_ON_TERMINATION_CALLBACK (contracts/interfaces/superfluid/Definitions.sol#53) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_CRITICAL_SENDER_ACCOUNT (contracts/interfaces/superfluid/Definitions.sol#54) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_CRITICAL_RECEIVER_ACCOUNT (contracts/interfaces/superfluid/Definitions.sol#55) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_READONLY (contracts/interfaces/superfluid/Definitions.sol#56) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_NOT_CLEAN (contracts/interfaces/superfluid/Definitions.sol#57) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_MALFORMATED (contracts/interfaces/superfluid/Definitions.sol#58) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_COMPOSITE_APP_IS_NOT_WHITELISTED (contracts/interfaces/superfluid/Definitions.sol#59) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_COMPOSITE_APP_IS_JAILED (contracts/interfaces/superfluid/Definitions.sol#60) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_MAX_APP_LEVEL_REACHED (contracts/interfaces/superfluid/Definitions.sol#61) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
ContextDefinitions.CALL_INFO_CALL_TYPE_AGREEMENT (contracts/interfaces/superfluid/Definitions.sol#79) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
ContextDefinitions.CALL_INFO_CALL_TYPE_APP_ACTION (contracts/interfaces/superfluid/Definitions.sol#80) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
ContextDefinitions.CALL_INFO_CALL_TYPE_APP_CALLBACK (contracts/interfaces/superfluid/Definitions.sol#81) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
SuperfluidGovernanceConfigs.SUPERFLUID_REWARD_ADDRESS_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#164-165) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
SuperfluidGovernanceConfigs.CFAV1_LIQUIDATION_PERIOD_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#167-168) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
SuperfluidGovernanceConfigs.SUPERTOKEN_MINIMUM_DEPOSIT_KEY (contracts/interfaces/superfluid/Definitions.sol#170-171) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

isAppJailed(ISuperApp) should be declared external:
        - Superfluid.isAppJailed(ISuperApp) (contracts/superfluid/Superfluid.sol#396-403)
getCodeAddress() should be declared external:
        - UUPSProxiable.getCodeAddress() (contracts/upgradability/UUPSProxiable.sol#15-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SuperfluidGovernanceII.sol SuperfluidGovernanceBase.sol

```
Contract locking ether found:
        Contract SuperfluidGovernanceIIProxy (contracts/gov/SuperfluidGovernanceII.sol#15) has payable functions:
         - Proxy.fallback() (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#63-65)
         - Proxy.receive() (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#71-73)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

SuperfluidGovernanceBase.updateContracts(ISuperfluid,address,address[],address) (contracts/gov/SuperfluidGovernanceBase.sol#54-72) has external calls inside a loop: host.updateAgreementClass(ISuperAgreement(agreementClassNewLogics[i
])) (contracts/gov/SuperfluidGovernanceBase.sol#67)
SuperfluidGovernanceBase.batchUpdateSuperTokenLogic(ISuperfluid,ISuperToken[]) (contracts/gov/SuperfluidGovernanceBase.sol#74-84) has external calls inside a loop: host.updateSuperTokenLogic(tokens[i]) (contracts/gov/SuperfluidGovern
anceBase.sol#82)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop

Proxy._delegate(address) (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#21-41) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#23-40)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
SuperfluidGovernanceBase.authorizeAppFactory(ISuperfluid,address) (contracts/gov/SuperfluidGovernanceBase.sol#436-454) uses assembly
        - INLINE ASM (contracts/gov/SuperfluidGovernanceBase.sol#446)
UUPSUtils.implementation() (contracts/upgradability/UUPSUtils.sol#18-22) uses assembly
        - INLINE ASM (contracts/upgradability/UUPSUtils.sol#19-21)
UUPSUtils.setImplementation(address) (contracts/upgradability/UUPSUtils.sol#25-33) uses assembly
        - INLINE ASM (contracts/upgradability/UUPSUtils.sol#26-32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.7.6', '>0.4.24<0.8.0', '>0.5.0', '>0.6.0<0.8.0', '>0.6.2<0.8.0', '>0.7.0']
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - >0.4.24<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#4)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777.sol#3)
        - >0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - 0.7.6 (contracts/gov/SuperfluidGovernanceBase.sol#2)
        - 0.7.6 (contracts/gov/SuperfluidGovernanceII.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/Definitions.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperAgreement.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperApp.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperToken.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperTokenFactory.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperfluid.sol#2)
        - ABIEncoderV2 (contracts/interfaces/superfluid/ISuperfluid.sol#4)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperfluidGovernance.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperfluidToken.sol#2)
        - >0.5.0 (contracts/interfaces/tokens/ERC20WithTokenInfo.sol#2)
        - >0.5.0 (contracts/interfaces/tokens/TokenInfo.sol#2)
        - 0.7.6 (contracts/upgradability/UUPSProxiable.sol#2)
        - 0.7.6 (contracts/upgradability/UUPSProxy.sol#2)
        - 0.7.6 (contracts/upgradability/UUPSUtils.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) is never used and should be removed
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#89-91) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145) is never used and should be removed
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-23) is never used and should be removed
ContextDefinitions.decodeCallInfo(uint256) (contracts/interfaces/superfluid/Definitions.sol#83-89) is never used and should be removed
ContextDefinitions.encodeCallInfo(uint8,uint8) (contracts/interfaces/superfluid/Definitions.sol#91-96) is never used and should be removed
```

AUTOMATED TESTING

```
Initializable._isConstructor() (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#52-54) is never used and should be removed
Proxy._implementation() (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#47) is never used and should be removed
SuperAppDefinitions.getAppLevel(uint256) (contracts/interfaces/superfluid/Definitions.sol#27-29) is never used and should be removed
SuperAppDefinitions.isAppJailed(uint256) (contracts/interfaces/superfluid/Definitions.sol#32-34) is never used and should be removed
SuperfluidGovernanceBase._requireAuthorised(ISuperfluid) (contracts/gov/SuperfluidGovernanceBase.sol#477) is never used and should be removed
SuperfluidGovernanceConfigs.getAppRegistrationConfigKey(address,string) (contracts/interfaces/superfluid/Definitions.sol#179-184) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) is too complex
Pragma version^0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#4) is too complex
Pragma version^0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#3) is too complex
Pragma version^0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version^0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777.sol#3) is too complex
Pragma version^0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version^0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) is too complex
Pragma version^0.7.0 (contracts/interfaces/superfluid/Definitions.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperAgreement.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperApp.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperToken.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperTokenFactory.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperfluid.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperfluidGovernance.sol#2) allows old versions
Pragma version^0.7.0 (contracts/interfaces/superfluid/ISuperfluidToken.sol#2) allows old versions
Pragma version^0.5.0 (contracts/interfaces/tokens/ERC20WithTokenInfo.sol#2) allows old versions
Pragma version^0.5.0 (contracts/interfaces/tokens/TokenInfo.sol#2) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable SuperfluidGovernanceBase._configs (contracts/gov/SuperfluidGovernanceBase.sol#28) is not in mixedCase
Constant SuperfluidGovernanceConfigs.CFAv1_LIQUIDATION_PERIOD_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#167-168) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (node_modules/@openzeppelin/contracts/utils/Context.sol#21)" inContext (node_modules/@openzeppelin/contracts/utils/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

SuperAppDefinitions.APP_LEVEL_FINAL (contracts/interfaces/superfluid/Definitions.sol#22) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_LEVEL_SECOND (contracts/interfaces/superfluid/Definitions.sol#25) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_JAIL_BIT (contracts/interfaces/superfluid/Definitions.sol#31) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AGREEMENT_CALLBACK_NOOP_BITMASKS (contracts/interfaces/superfluid/Definitions.sol#39) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_CREATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#40) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_CREATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#41) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_UPDATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#42) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_UPDATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#43) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_TERMINATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#44) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_TERMINATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#45) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_REGISTRATION_ONLY_IN_CONSTRUCTOR (contracts/interfaces/superfluid/Definitions.sol#51) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_REGISTRATION_FOR_EOA (contracts/interfaces/superfluid/Definitions.sol#52) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_REVERT_ON_TERMINATION_CALLBACK (contracts/interfaces/superfluid/Definitions.sol#53) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_CRITICAL_SENDER_ACCOUNT (contracts/interfaces/superfluid/Definitions.sol#54) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_CRITICAL_RECEIVER_ACCOUNT (contracts/interfaces/superfluid/Definitions.sol#55) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_READONLY (contracts/interfaces/superfluid/Definitions.sol#56) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_NOT_CLEAN (contracts/interfaces/superfluid/Definitions.sol#57) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_MALFORMATED (contracts/interfaces/superfluid/Definitions.sol#58) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_COMPOSITE_APP_IS_NOT_WHITELISTED (contracts/interfaces/superfluid/Definitions.sol#59) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_COMPOSITE_APP_IS_JAILED (contracts/interfaces/superfluid/Definitions.sol#60) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_MAX_APP_LEVEL_REACHED (contracts/interfaces/superfluid/Definitions.sol#61) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
ContextDefinitions.CALL_INFO_CALL_TYPE_AGREEMENT (contracts/interfaces/superfluid/Definitions.sol#79) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
ContextDefinitions.CALL_INFO_CALL_TYPE_APP_ACTION (contracts/interfaces/superfluid/Definitions.sol#80) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
ContextDefinitions.CALL_INFO_CALL_TYPE_APP_CALLBACK (contracts/interfaces/superfluid/Definitions.sol#81) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
SuperfluidGovernanceConfigs.SUPERFLUID_REWARD_ADDRESS_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#164-165) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)

SuperfluidGovernanceConfigs.CFAv1_LIQUIDATION_PERIOD_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#167-168) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
SuperfluidGovernanceConfigs.SUPERTOKEN_MINIMUM_DEPOSIT_KEY (contracts/interfaces/superfluid/Definitions.sol#170-171) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#63-67)
getRewardAddress(ISuperfluid,ISuperfluidToken) should be declared external:
        - SuperfluidGovernanceBase.getRewardAddress(ISuperfluid,ISuperfluidToken) (contracts/gov/SuperfluidGovernanceBase.sol#221-230)
setRewardAddress(ISuperfluid,ISuperfluidToken,address) should be declared external:
        - SuperfluidGovernanceBase.setRewardAddress(ISuperfluid,ISuperfluidToken,address) (contracts/gov/SuperfluidGovernanceBase.sol#232-244)
clearRewardAddress(ISuperfluid,ISuperfluidToken) should be declared external:
        - SuperfluidGovernanceBase.clearRewardAddress(ISuperfluid,ISuperfluidToken) (contracts/gov/SuperfluidGovernanceBase.sol#246-256)
getCFAv1LiquidationPeriod(ISuperfluid,ISuperfluidToken) should be declared external:
        - SuperfluidGovernanceBase.getCFAv1LiquidationPeriod(ISuperfluid,ISuperfluidToken) (contracts/gov/SuperfluidGovernanceBase.sol#265-275)
setCFAv1LiquidationPeriod(ISuperfluid,ISuperfluidToken,uint256) should be declared external:
        - SuperfluidGovernanceBase.setCFAv1LiquidationPeriod(ISuperfluid,ISuperfluidToken,uint256) (contracts/gov/SuperfluidGovernanceBase.sol#277-289)
clearCFAv1LiquidationPeriod(ISuperfluid,ISuperfluidToken) should be declared external:
        - SuperfluidGovernanceBase.clearCFAv1LiquidationPeriod(ISuperfluid,ISuperfluidToken) (contracts/gov/SuperfluidGovernanceBase.sol#291-301)
getSuperTokenMinimumDeposit(ISuperfluid,ISuperfluidToken) should be declared external:
        - SuperfluidGovernanceBase.getSuperTokenMinimumDeposit(ISuperfluid,ISuperfluidToken) (contracts/gov/SuperfluidGovernanceBase.sol#310-318)
clearSuperTokenMinimumDeposit(ISuperfluid,ISuperToken) should be declared external:
        - SuperfluidGovernanceBase.clearSuperTokenMinimumDeposit(ISuperfluid,ISuperToken) (contracts/gov/SuperfluidGovernanceBase.sol#329-336)
isTrustedForwarder(ISuperfluid,ISuperfluidToken,address) should be declared external:
        - SuperfluidGovernanceBase.isTrustedForwarder(ISuperfluid,ISuperfluidToken,address) (contracts/gov/SuperfluidGovernanceBase.sol#346-357)
enableTrustedForwarder(ISuperfluid,ISuperfluidToken,address) should be declared external:
        - SuperfluidGovernanceBase.enableTrustedForwarder(ISuperfluid,ISuperfluidToken,address) (contracts/gov/SuperfluidGovernanceBase.sol#359-371)
disableTrustedForwarder(ISuperfluid,ISuperfluidToken,address) should be declared external:
        - SuperfluidGovernanceBase.disableTrustedForwarder(ISuperfluid,ISuperfluidToken,address) (contracts/gov/SuperfluidGovernanceBase.sol#373-385)
clearTrustedForwarder(ISuperfluid,ISuperfluidToken,address) should be declared external:
        - SuperfluidGovernanceBase.clearTrustedForwarder(ISuperfluid,ISuperfluidToken,address) (contracts/gov/SuperfluidGovernanceBase.sol#387-398)
isAuthorizedAppFactory(ISuperfluid,address) should be declared external:
        - SuperfluidGovernanceBase.isAuthorizedAppFactory(ISuperfluid,address) (contracts/gov/SuperfluidGovernanceBase.sol#420-430)
authorizeAppFactory(ISuperfluid,address) should be declared external:
        - SuperfluidGovernanceBase.authorizeAppFactory(ISuperfluid,address) (contracts/gov/SuperfluidGovernanceBase.sol#436-454)
unauthorizeAppFactory(ISuperfluid,address) should be declared external:
        - SuperfluidGovernanceBase.unauthorizeAppFactory(ISuperfluid,address) (contracts/gov/SuperfluidGovernanceBase.sol#460-469)
getCodeAddress() should be declared external:
        - UUPSProxiable.getCodeAddress() (contracts/upgradability/UUPSProxiable.sol#15-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SuperToken.sol SuperfluidToken.sol

```
SuperfluidToken._inactiveAgreementBitmap (contracts/superfluid/SuperfluidToken.sol#37) is never initialized. It is used in:
        - SuperfluidToken.getAccountActiveAgreements(address) (contracts/superfluid/SuperfluidToken.sol#178-181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

SuperToken._toUnderlyingAmount(uint256) (contracts/superfluid/SuperToken.sol#616-637) performs a multiplication on the result of a division:
        -underlyingAmount = amount / factor (contracts/superfluid/SuperToken.sol#625)
        -adjustedAmount = underlyingAmount * factor (contracts/superfluid/SuperToken.sol#627)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

SuperToken._downgrade(address,address,uint256,bytes,bytes) (contracts/superfluid/SuperToken.sol#589-611) uses a dangerous strict equality:
        - require(bool,string)(underlyingAmount == actualDowngradedAmount,SuperToken: inflationary/deflationary tokens not supported) (contracts/superfluid/SuperToken.sol#606-608)
SuperToken._upgrade(address,address,address,uint256,bytes,bytes) (contracts/superfluid/SuperToken.sol#562-587) uses a dangerous strict equality:
        - require(bool,string)(underlyingAmount == actualUpgradedAmount,SuperToken: inflationary/deflationary tokens not supported) (contracts/superfluid/SuperToken.sol#578-580)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

SuperfluidToken.realtimeBalanceOf(address,uint256) (contracts/superfluid/SuperfluidToken.sol#78-111) has external calls inside a loop: (agreementDynamicBalance,agreementDeposit,agreementOwedDeposit) = activeAgreements[i].realtimeBala
nceOf(this,account,timestamp) (contracts/superfluid/SuperfluidToken.sol#91-99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop

Reentrancy in SuperToken._burn(address,address,uint256,bytes,bytes) (contracts/superfluid/SuperToken.sol#254-271):
        External calls:
        - _callTokensToSend(operator,from,address(0),amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#265)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#318)
        State variables written after the call(s):
        - SuperfluidToken._burn(from,amount) (contracts/superfluid/SuperToken.sol#267)
                - _balances[account] = _balances[account].sub(amount.toInt256()) (contracts/superfluid/SuperfluidToken.sol#207)
        - SuperfluidToken._burn(from,amount) (contracts/superfluid/SuperToken.sol#267)
                - _totalSupply = _totalSupply.sub(amount) (contracts/superfluid/SuperfluidToken.sol#208)
Reentrancy in SuperToken._send(address,address,address,uint256,bytes,bytes,bool) (contracts/superfluid/SuperToken.sol#173-192):
        External calls:
        - _callTokensToSend(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#187)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#318)
        State variables written after the call(s):
        - _move(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#189)
                - _balances[from] = _balances[from].sub(amount) (contracts/superfluid/SuperfluidToken.sol#220)
                - _balances[to] = _balances[to].add(amount) (contracts/superfluid/SuperfluidToken.sol#221)
Reentrancy in SuperToken._transferFrom(address,address,address,uint256) (contracts/superfluid/SuperToken.sol#139-161):
        External calls:
        - _callTokensToSend(operator,holder,recipient,amount,,) (contracts/superfluid/SuperToken.sol#147)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#318)
        State variables written after the call(s):
        - _approve(holder,spender,_allowances[holder][spender].sub(amount,SuperToken: transfer amount exceeds allowance)) (contracts/superfluid/SuperToken.sol#152-155)
                - _allowances[account][spender] = amount (contracts/superfluid/SuperToken.sol#292)
        - _move(operator,holder,recipient,amount,,) (contracts/superfluid/SuperToken.sol#149)
                - _balances[from] = _balances[from].sub(amount) (contracts/superfluid/SuperfluidToken.sol#220)
                - _balances[to] = _balances[to].add(amount) (contracts/superfluid/SuperfluidToken.sol#221)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SuperToken._burn(address,address,uint256,bytes,bytes) (contracts/superfluid/SuperToken.sol#254-271):
        External calls:
        - _callTokensToSend(operator,from,address(0),amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#265)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#318)
        Event emitted after the call(s):
        - Burned(operator,from,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#269)
        - Transfer(from,address(0),amount) (contracts/superfluid/SuperToken.sol#270)
Reentrancy in SuperToken._downgrade(address,address,uint256,bytes,bytes) (contracts/superfluid/SuperToken.sol#589-611):
        External calls:
        - _burn(operator,account,adjustedAmount,data,operatorData) (contracts/superfluid/SuperToken.sol#600)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#318)
        - _underlyingToken.safeTransfer(account,underlyingAmount) (contracts/superfluid/SuperToken.sol#603)
        Event emitted after the call(s):
        - TokenDowngraded(account,adjustedAmount) (contracts/superfluid/SuperToken.sol#610)
Reentrancy in SuperToken._mint(address,address,uint256,bool,bytes,bytes) (contracts/superfluid/SuperToken.sol#227-245):
        External calls:
        - _callTokensReceived(operator,address(0),account,amount,userData,operatorData,requireReceptionAck) (contracts/superfluid/SuperToken.sol#241)
                - IERC777Recipient(implementer).tokensReceived(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#347)
        Event emitted after the call(s):
        - Minted(operator,account,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#243)
        - Transfer(address(0),account,amount) (contracts/superfluid/SuperToken.sol#244)
Reentrancy in SuperToken._send(address,address,address,uint256,bytes,bytes,bool) (contracts/superfluid/SuperToken.sol#173-192):

        External calls:
        - _callTokensToSend(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#187)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#318)
        Event emitted after the call(s):
        - Sent(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#286)
                - _move(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#189)
        - Transfer(from,to,amount) (contracts/superfluid/SuperToken.sol#287)
                - _move(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#189)
Reentrancy in SuperToken._transferFrom(address,address,address,uint256) (contracts/superfluid/SuperToken.sol#139-161):
        External calls:
        - _callTokensToSend(operator,holder,recipient,amount,,) (contracts/superfluid/SuperToken.sol#147)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#318)
        Event emitted after the call(s):
        - Approval(account,spender,amount) (contracts/superfluid/SuperToken.sol#293)
                - _approve(holder,spender,_allowances[holder][spender].sub(amount,SuperToken: transfer amount exceeds allowance)) (contracts/superfluid/SuperToken.sol#152-155)
        - Sent(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#286)
                - _move(operator,holder,recipient,amount,,) (contracts/superfluid/SuperToken.sol#149)
        - Transfer(from,to,amount) (contracts/superfluid/SuperToken.sol#287)
                - _move(operator,holder,recipient,amount,,) (contracts/superfluid/SuperToken.sol#149)
Reentrancy in SuperToken._upgrade(address,address,address,uint256,bytes,bytes) (contracts/superfluid/SuperToken.sol#562-587):
        External calls:
        - _underlyingToken.safeTransferFrom(account,address(this),underlyingAmount) (contracts/superfluid/SuperToken.sol#575)
        - _mint(operator,to,adjustedAmount,account == to,userData,operatorData) (contracts/superfluid/SuperToken.sol#582-584)
                - IERC777Recipient(implementer).tokensReceived(operator,from,to,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#347)
        Event emitted after the call(s):
        - Minted(operator,account,amount,userData,operatorData) (contracts/superfluid/SuperToken.sol#243)
                - _mint(operator,to,adjustedAmount,account == to,userData,operatorData) (contracts/superfluid/SuperToken.sol#582-584)
        - TokenUpgraded(to,adjustedAmount) (contracts/superfluid/SuperToken.sol#586)
        - Transfer(address(0),account,amount) (contracts/superfluid/SuperToken.sol#244)
                - _mint(operator,to,adjustedAmount,account == to,userData,operatorData) (contracts/superfluid/SuperToken.sol#582-584)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

SuperfluidToken.realtimeBalanceOf(address,uint256) (contracts/superfluid/SuperfluidToken.sol#78-111) uses timestamp for comparisons
        Dangerous comparisons:
        - agreementDeposit > agreementOwedDeposit (contracts/superfluid/SuperfluidToken.sol#104-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
FixedSizeData.storeData(bytes32,bytes32[]) (contracts/libs/FixedSizeData.sol#22-27) uses assembly
        - INLINE ASM (contracts/libs/FixedSizeData.sol#25)
FixedSizeData.hasData(bytes32,uint256) (contracts/libs/FixedSizeData.sol#29-36) uses assembly
        - INLINE ASM (contracts/libs/FixedSizeData.sol#32)
FixedSizeData.loadData(bytes32,uint256) (contracts/libs/FixedSizeData.sol#41-48) uses assembly
        - INLINE ASM (contracts/libs/FixedSizeData.sol#45)
FixedSizeData.eraseData(bytes32,uint256) (contracts/libs/FixedSizeData.sol#53-57) uses assembly
        - INLINE ASM (contracts/libs/FixedSizeData.sol#55)
UUPSUtils.implementation() (contracts/upgradability/UUPSUtils.sol#16-22) uses assembly
        - INLINE ASM (contracts/upgradability/UUPSUtils.sol#19-21)
UUPSUtils.setImplementation(address) (contracts/upgradability/UUPSUtils.sol#25-33) uses assembly
        - INLINE ASM (contracts/upgradability/UUPSUtils.sol#26-32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.7.6', '>0.4.24<0.8.0', '>0.5.0', '>0.6.0<0.8.0', '>0.6.2<0.8.0', '>0.7.0']
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/introspection/IERC1820Registry.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/Math.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#3)
        - >0.4.24<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#4)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777Recipient.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777Sender.sol#3)
        - >0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#3)
        - >0.7.0 (contracts/interfaces/superfluid/Definitions.sol#2)
```

```
  - ⩾0.7.0 (contracts/interfaces/superfluid/ISuperAgreement.sol#2)
  - ⩾0.7.0 (contracts/interfaces/superfluid/ISuperApp.sol#2)
  - ⩾0.7.0 (contracts/interfaces/superfluid/ISuperToken.sol#2)
  - ⩾0.7.0 (contracts/interfaces/superfluid/ISuperTokenFactory.sol#2)
  - ⩾0.7.0 (contracts/interfaces/superfluid/ISuperfluid.sol#2)
  - ABIEncoderV2 (contracts/interfaces/superfluid/ISuperfluid.sol#4)
  - ⩾0.7.0 (contracts/interfaces/superfluid/ISuperfluidGovernance.sol#2)
  - ⩾0.7.0 (contracts/interfaces/superfluid/ISuperfluidToken.sol#2)
  - ⩾0.5.0 (contracts/interfaces/tokens/ERC20WithTokenInfo.sol#2)
  - ⩾0.5.0 (contracts/interfaces/tokens/TokenInfo.sol#2)
  - 0.7.6 (contracts/libs/ERC777Helper.sol#2)
  - 0.7.6 (contracts/libs/FixedSizeData.sol#2)
  - 0.7.6 (contracts/superfluid/SuperToken.sol#2)
  - 0.7.6 (contracts/superfluid/SuperfluidToken.sol#3)
  - 0.7.6 (contracts/upgradability/UUPSProxiable.sol#2)
  - 0.7.6 (contracts/upgradability/UUPSUtils.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
ContextDefinitions.decodeCallInfo(uint256) (contracts/interfaces/superfluid/Definitions.sol#83-89) is never used and should be removed
ContextDefinitions.encodeCallInfo(uint8,uint8) (contracts/interfaces/superfluid/Definitions.sol#91-96) is never used and should be removed
ERC777Helper.setupDefaultOperators(ERC777Helper.Operators,address[]) (contracts/libs/ERC777Helper.sol#66-73) is never used and should be removed
Math.average(uint256,uint256) (node_modules/@openzeppelin/contracts/math/Math.sol#27-30) is never used and should be removed
Math.max(uint256,uint256) (node_modules/@openzeppelin/contracts/math/Math.sol#12-14) is never used and should be removed
Math.min(uint256,uint256) (node_modules/@openzeppelin/contracts/math/Math.sol#19-21) is never used and should be removed
SafeCast.toInt128(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#123-126) is never used and should be removed
SafeCast.toInt16(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#177-180) is never used and should be removed
SafeCast.toInt32(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#159-162) is never used and should be removed
SafeCast.toInt64(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#141-144) is never used and should be removed
SafeCast.toInt8(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#195-198) is never used and should be removed
SafeCast.toUint128(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#33-36) is never used and should be removed
SafeCast.toUint16(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#78-81) is never used and should be removed
SafeCast.toUint256(int256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#185-188) is never used and should be removed
SafeCast.toUint32(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#63-66) is never used and should be removed
SafeCast.toUint64(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#48-51) is never used and should be removed
SafeCast.toUint8(uint256) (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#93-96) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#37-44) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#53-56) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#46-51) is never used and should be removed
SafeMath.div(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#135-138) is never used and should be removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#190-193) is never used and should be removed
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#152-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#210-213) is never used and should be removed
SafeMath.mul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#116-121) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#24-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#60-63) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#70-73) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#45-51) is never used and should be removed
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#35-38) is never used and should be removed
SignedSafeMath.div(int256,int256) (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#58-57) is never used and should be removed
SignedSafeMath.mul(int256,int256) (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#22-36) is never used and should be removed
SuperAppDefinitions.getAppLevel(uint256) (contracts/interfaces/superfluid/Definitions.sol#27-29) is never used and should be removed
SuperAppDefinitions.isAppJailed(uint256) (contracts/interfaces/superfluid/Definitions.sol#32-34) is never used and should be removed
SuperToken._setupDefaultOperators(address[]) (contracts/superfluid/SuperToken.sol#474-476) is never used and should be removed
SuperfluidGovernanceConfigs.getAppFactoryConfigKey(address) (contracts/interfaces/superfluid/Definitions.sol#186-190) is never used and should be removed
SuperfluidGovernanceConfigs.getAppRegistrationConfigKey(address,string) (contracts/interfaces/superfluid/Definitions.sol#179-184) is never used and should be removed
SuperfluidGovernanceConfigs.getTrustedForwarderConfigKey(address) (contracts/interfaces/superfluid/Definitions.sol#173-177) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version⩾0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/introspection/IERC1820Registry.sol#3) is too complex
Pragma version⩾0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/Math.sol#3) is too complex
Pragma version⩾0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3) is too complex
Pragma version⩾0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SignedSafeMath.sol#3) is too complex
Pragma version⩾0.4.24<0.8.0 (node_modules/@openzeppelin/contracts/proxy/Initializable.sol#4) is too complex

Pragma version⩾0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version⩾0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3) is too complex
Pragma version⩾0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777.sol#3) is too complex
Pragma version⩾0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777Recipient.sol#3) is too complex
Pragma version⩾0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777Sender.sol#3) is too complex
Pragma version⩾0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version⩾0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#3) is too complex
Pragma version⩾0.7.0 (contracts/interfaces/superfluid/Definitions.sol#2) allows old versions
Pragma version⩾0.7.0 (contracts/interfaces/superfluid/ISuperAgreement.sol#2) allows old versions
Pragma version⩾0.7.0 (contracts/interfaces/superfluid/ISuperApp.sol#2) allows old versions
Pragma version⩾0.7.0 (contracts/interfaces/superfluid/ISuperToken.sol#2) allows old versions
Pragma version⩾0.7.0 (contracts/interfaces/superfluid/ISuperTokenFactory.sol#2) allows old versions
Pragma version⩾0.7.0 (contracts/interfaces/superfluid/ISuperfluid.sol#2) allows old versions
Pragma version⩾0.7.0 (contracts/interfaces/superfluid/ISuperfluidGovernance.sol#2) allows old versions
Pragma version⩾0.7.0 (contracts/interfaces/superfluid/ISuperfluidToken.sol#2) allows old versions
Pragma version⩾0.5.0 (contracts/interfaces/tokens/ERC20WithTokenInfo.sol#2) allows old versions
Pragma version⩾0.5.0 (contracts/interfaces/tokens/TokenInfo.sol#2) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Constant SuperfluidGovernanceConfigs.CFAv1_LIQUIDATION_PERIOD_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#167-168) is not in UPPER_CASE_WITH_UNDERSCORES
Variable SuperToken._underlyingToken (contracts/superfluid/SuperToken.sol#53) is not in mixedCase
Variable SuperToken._underlyingDecimals (contracts/superfluid/SuperToken.sol#56) is not in mixedCase
Variable SuperToken._name (contracts/superfluid/SuperToken.sol#59) is not in mixedCase
Variable SuperToken._symbol (contracts/superfluid/SuperToken.sol#62) is not in mixedCase
Variable SuperToken._allowances (contracts/superfluid/SuperToken.sol#65) is not in mixedCase
Variable SuperToken._operators (contracts/superfluid/SuperToken.sol#68) is not in mixedCase
Variable SuperToken._reserve2 (contracts/superfluid/SuperToken.sol#72) is not in mixedCase
Variable SuperToken._reserve31 (contracts/superfluid/SuperToken.sol#81) is not in mixedCase
Variable SuperfluidToken._host (contracts/superfluid/SuperfluidToken.sol#34) is not in mixedCase
Variable SuperfluidToken._inactiveAgreementBitmap (contracts/superfluid/SuperfluidToken.sol#37) is not in mixedCase
Variable SuperfluidToken._balances (contracts/superfluid/SuperfluidToken.sol#40) is not in mixedCase
Variable SuperfluidToken._totalSupply (contracts/superfluid/SuperfluidToken.sol#43) is not in mixedCase
Variable SuperfluidToken._reserve4 (contracts/superfluid/SuperfluidToken.sol#47) is not in mixedCase
Variable SuperfluidToken._reserve13 (contracts/superfluid/SuperfluidToken.sol#56) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SuperAppDefinitions.APP_LEVEL_FINAL (contracts/interfaces/superfluid/Definitions.sol#22) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_LEVEL_SECOND (contracts/interfaces/superfluid/Definitions.sol#25) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_JAIL_BIT (contracts/interfaces/superfluid/Definitions.sol#31) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AGREEMENT_CALLBACK_NOOP_BITMASKS (contracts/interfaces/superfluid/Definitions.sol#39) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_CREATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#40) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_CREATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#41) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_UPDATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#42) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_UPDATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#43) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_TERMINATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#44) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_TERMINATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#45) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_REGISTRATION_ONLY_IN_CONSTRUCTOR (contracts/interfaces/superfluid/Definitions.sol#51) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_REGISTRATION_FOR_EOA (contracts/interfaces/superfluid/Definitions.sol#52) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_REVERT_ON_TERMINATION_CALLBACK (contracts/interfaces/superfluid/Definitions.sol#53) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_CRITICAL_SENDER_ACCOUNT (contracts/interfaces/superfluid/Definitions.sol#54) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_CRITICAL_RECEIVER_ACCOUNT (contracts/interfaces/superfluid/Definitions.sol#55) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_READONLY (contracts/interfaces/superfluid/Definitions.sol#56) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_NOT_CLEAN (contracts/interfaces/superfluid/Definitions.sol#57) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_MALFORMATED (contracts/interfaces/superfluid/Definitions.sol#58) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_COMPOSITE_APP_IS_NOT_WHITELISTED (contracts/interfaces/superfluid/Definitions.sol#59) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_COMPOSITE_APP_IS_JAILED (contracts/interfaces/superfluid/Definitions.sol#60) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_MAX_APP_LEVEL_REACHED (contracts/interfaces/superfluid/Definitions.sol#61) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
ContextDefinitions.CALL_INFO_CALL_TYPE_AGREEMENT (contracts/interfaces/superfluid/Definitions.sol#79) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
ContextDefinitions.CALL_INFO_CALL_TYPE_APP_ACTION (contracts/interfaces/superfluid/Definitions.sol#80) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
ContextDefinitions.CALL_INFO_CALL_TYPE_APP_CALLBACK (contracts/interfaces/superfluid/Definitions.sol#81) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
```

```
SuperfluidGovernanceConfigs.SUPERFLUID_REWARD_ADDRESS_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#164-165) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
SuperfluidGovernanceConfigs.CFAV1_LIQUIDATION_PERIOD_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#167-168) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
SuperfluidGovernanceConfigs.SUPERTOKEN_MINIMUM_DEPOSIT_KEY (contracts/interfaces/superfluid/Definitions.sol#170-171) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
ERC777Helper._TOKENS_SENDER_INTERFACE_HASH (contracts/libs/ERC777Helper.sol#13-14) is never used in ERC777Helper (contracts/libs/ERC777Helper.sol#7-75)
ERC777Helper._TOKENS_RECIPIENT_INTERFACE_HASH (contracts/libs/ERC777Helper.sol#17-18) is never used in ERC777Helper (contracts/libs/ERC777Helper.sol#7-75)
SuperfluidToken._reserve4 (contracts/superfluid/SuperfluidToken.sol#47) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperfluidToken._reserve5 (contracts/superfluid/SuperfluidToken.sol#48) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperfluidToken._reserve6 (contracts/superfluid/SuperfluidToken.sol#49) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperfluidToken._reserve7 (contracts/superfluid/SuperfluidToken.sol#50) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperfluidToken._reserve8 (contracts/superfluid/SuperfluidToken.sol#51) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperfluidToken._reserve9 (contracts/superfluid/SuperfluidToken.sol#52) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperfluidToken._reserve10 (contracts/superfluid/SuperfluidToken.sol#53) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperfluidToken._reserve11 (contracts/superfluid/SuperfluidToken.sol#54) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperfluidToken._reserve12 (contracts/superfluid/SuperfluidToken.sol#55) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperfluidToken._reserve13 (contracts/superfluid/SuperfluidToken.sol#56) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperToken._reserve22 (contracts/superfluid/SuperToken.sol#72) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperToken._reserve23 (contracts/superfluid/SuperToken.sol#73) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperToken._reserve24 (contracts/superfluid/SuperToken.sol#74) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperToken._reserve25 (contracts/superfluid/SuperToken.sol#75) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperToken._reserve26 (contracts/superfluid/SuperToken.sol#76) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperToken._reserve27 (contracts/superfluid/SuperToken.sol#77) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperToken._reserve28 (contracts/superfluid/SuperToken.sol#78) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperToken._reserve29 (contracts/superfluid/SuperToken.sol#79) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperToken._reserve30 (contracts/superfluid/SuperToken.sol#80) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
SuperToken._reserve31 (contracts/superfluid/SuperToken.sol#81) is never used in SuperToken (contracts/superfluid/SuperToken.sol#32-689)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

SuperToken._reserve22 (contracts/superfluid/SuperToken.sol#72) should be constant
SuperToken._reserve23 (contracts/superfluid/SuperToken.sol#73) should be constant
SuperToken._reserve24 (contracts/superfluid/SuperToken.sol#74) should be constant
SuperToken._reserve25 (contracts/superfluid/SuperToken.sol#75) should be constant
SuperToken._reserve26 (contracts/superfluid/SuperToken.sol#76) should be constant
SuperToken._reserve27 (contracts/superfluid/SuperToken.sol#77) should be constant
SuperToken._reserve28 (contracts/superfluid/SuperToken.sol#78) should be constant
SuperToken._reserve29 (contracts/superfluid/SuperToken.sol#79) should be constant
SuperToken._reserve30 (contracts/superfluid/SuperToken.sol#80) should be constant
SuperToken._reserve31 (contracts/superfluid/SuperToken.sol#81) should be constant
SuperfluidToken._reserve10 (contracts/superfluid/SuperfluidToken.sol#53) should be constant
SuperfluidToken._reserve11 (contracts/superfluid/SuperfluidToken.sol#54) should be constant
SuperfluidToken._reserve12 (contracts/superfluid/SuperfluidToken.sol#55) should be constant
SuperfluidToken._reserve13 (contracts/superfluid/SuperfluidToken.sol#56) should be constant
SuperfluidToken._reserve4 (contracts/superfluid/SuperfluidToken.sol#47) should be constant
SuperfluidToken._reserve5 (contracts/superfluid/SuperfluidToken.sol#48) should be constant
SuperfluidToken._reserve6 (contracts/superfluid/SuperfluidToken.sol#49) should be constant
SuperfluidToken._reserve7 (contracts/superfluid/SuperfluidToken.sol#50) should be constant
SuperfluidToken._reserve8 (contracts/superfluid/SuperfluidToken.sol#51) should be constant
SuperfluidToken._reserve9 (contracts/superfluid/SuperfluidToken.sol#52) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

totalSupply() should be declared external:
        - SuperToken.totalSupply() (contracts/superfluid/SuperToken.sol#359-363)
transfer(address,uint256) should be declared external:
        - SuperToken.transfer(address,uint256) (contracts/superfluid/SuperToken.sol#378-382)
allowance(address,address) should be declared external:
        - SuperToken.allowance(address,address) (contracts/superfluid/SuperToken.sol#384-388)
approve(address,uint256) should be declared external:
        - SuperToken.approve(address,uint256) (contracts/superfluid/SuperToken.sol#390-396)
transferFrom(address,address,uint256) should be declared external:
        - SuperToken.transferFrom(address,address,uint256) (contracts/superfluid/SuperToken.sol#398-402)
increaseAllowance(address,uint256) should be declared external:
        - SuperToken.increaseAllowance(address,uint256) (contracts/superfluid/SuperToken.sol#404-408)
decreaseAllowance(address,uint256) should be declared external:
        - SuperToken.decreaseAllowance(address,uint256) (contracts/superfluid/SuperToken.sol#410-415)
getCodeAddress() should be declared external:
        - UUPSProxiable.getCodeAddress() (contracts/upgradability/UUPSProxiable.sol#15-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

TOGA.sol

```
Reentrancy in TOGA._becomePIC(ISuperToken,address,uint256,int96) (contracts/utils/TOGA.sol#246-309):
        External calls:
        - _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.deleteFlow.selector,token,address(this),currentPICAddr,new bytes(0)),0x) (contracts/utils/TOGA.sol#260-270)
        - token.send(gas: ERC777_SEND_GAS_LIMIT)(currentPICAddr,currentPICBond,0x) (contracts/utils/TOGA.sol#277-284)
        - token.send(address(custodian),currentPICBond,abi.encode(currentPICAddr)) (contracts/utils/TOGA.sol#283)
        State variables written after the call(s):
        - _currentPICs[token].addr = newPIC (contracts/utils/TOGA.sol#289)
Reentrancy in TOGA._becomePIC(ISuperToken,address,uint256,int96) (contracts/utils/TOGA.sol#246-309):
        External calls:
        - _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.deleteFlow.selector,token,address(this),currentPICAddr,new bytes(0)),0x) (contracts/utils/TOGA.sol#260-270)
        - token.send(gas: ERC777_SEND_GAS_LIMIT)(currentPICAddr,currentPICBond,0x) (contracts/utils/TOGA.sol#277-284)
        - token.send(address(custodian),currentPICBond,abi.encode(currentPICAddr)) (contracts/utils/TOGA.sol#283)
        - _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.createFlow.selector,token,newPIC,exitRate,new bytes(0)),0x) (contracts/utils/TOGA.sol#293-303)
        State variables written after the call(s):
        - _currentPICs[token].lock = false (contracts/utils/TOGA.sol#307)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

TOGA.changeExitRate(ISuperToken,int96) (contracts/utils/TOGA.sol#181-230) ignores return value by _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.updateFlow.selector,token,currentPICAddr,newExitRate,new bytes(0)),0x) (contracts/utils/TOGA.sol#198-208)
TOGA.changeExitRate(ISuperToken,int96) (contracts/utils/TOGA.sol#181-230) ignores return value by _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.createFlow.selector,token,currentPICAddr,newExitRate,new bytes(0)),0x) (contracts/utils/TOGA.sol#203-213)
TOGA.changeExitRate(ISuperToken,int96) (contracts/utils/TOGA.sol#181-230) ignores return value by _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.deleteFlow.selector,token,address(this),currentPICAddr,new bytes(0)),0x) (contracts/utils/TOGA.sol#216-226)
TOGA._becomePIC(ISuperToken,address,uint256,int96) (contracts/utils/TOGA.sol#246-309) ignores return value by _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.deleteFlow.selector,token,address(this),currentPICAddr,new bytes(0)),0x) (contracts/utils/TOGA.sol#260-270)
TOGA._becomePIC(ISuperToken,address,uint256,int96) (contracts/utils/TOGA.sol#246-309) ignores return value by _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.createFlow.selector,token,newPIC,exitRate,new bytes(0)),0x) (contracts/utils/TOGA.sol#293-303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Reentrancy in TOGA.constructor(ISuperfluid,uint256,TokenCustodian) (contracts/utils/TOGA.sol#133-149):
        External calls:
        - _ERC1820_REG.setInterfaceImplementer(address(this),erc777TokensRecipientHash,address(this)) (contracts/utils/TOGA.sol#140)
        - _ERC1820_REG.setInterfaceImplementer(address(this),keccak256(bytes)(TOGAv1),address(this)) (contracts/utils/TOGA.sol#141)
        - _ERC1820_REG.setInterfaceImplementer(address(this),keccak256(bytes)(TOGAv2),address(this)) (contracts/utils/TOGA.sol#142)
        State variables written after the call(s):
        - custodian = custodian_ (contracts/utils/TOGA.sol#148)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in TOGA._becomePIC(ISuperToken,address,uint256,int96) (contracts/utils/TOGA.sol#246-309):
        External calls:
        - _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.deleteFlow.selector,token,address(this),currentPICAddr,new bytes(0)),0x) (contracts/utils/TOGA.sol#260-270)
        - token.send(gas: ERC777_SEND_GAS_LIMIT)(currentPICAddr,currentPICBond,0x) (contracts/utils/TOGA.sol#277-284)
        - token.send(address(custodian),currentPICBond,abi.encode(currentPICAddr)) (contracts/utils/TOGA.sol#283)
        - _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.createFlow.selector,token,newPIC,exitRate,new bytes(0)),0x) (contracts/utils/TOGA.sol#293-303)
        Event emitted after the call(s):
        - NewPIC(token,newPIC,amount,exitRate) (contracts/utils/TOGA.sol#308)
Reentrancy in TOGA.changeExitRate(ISuperToken,int96) (contracts/utils/TOGA.sol#181-230):
        External calls:
        - _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.updateFlow.selector,token,currentPICAddr,newExitRate,new bytes(0)),0x) (contracts/utils/TOGA.sol#198-208)
        - _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.createFlow.selector,token,currentPICAddr,newExitRate,new bytes(0)),0x) (contracts/utils/TOGA.sol#203-213)
        - _host.callAgreement(_cfa,abi.encodeWithSelector(_cfa.deleteFlow.selector,token,address(this),currentPICAddr,new bytes(0)),0x) (contracts/utils/TOGA.sol#216-226)
        Event emitted after the call(s):
        - ExitRateChanged(token,newExitRate) (contracts/utils/TOGA.sol#229)
Reentrancy in TokenCustodian.flush(IERC777,address) (contracts/utils/TokenCustodian.sol#22-30):
        External calls:
        - token.send(recipient,amount,0x0) (contracts/utils/TokenCustodian.sol#27)
        Event emitted after the call(s):
        - CustodianWithdrawal(token,recipient,amount) (contracts/utils/TokenCustodian.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Different versions of Solidity is used:
        - Version used: ['0.7.6', '>0.5.0', '>0.6.0<0.8.0', '>0.7.0']
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/introspection/IERC1820Registry.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777.sol#3)
        - >0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777Recipient.sol#3)
        - >0.7.0 (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/Definitions.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperAgreement.sol#2)
        - >0.7.0 (contracts/interfaces/superfluid/ISuperApp.sol#2)
```

```
        - ≥0.7.0 (contracts/interfaces/superfluid/ISuperToken.sol#2)
        - ≥0.7.0 (contracts/interfaces/superfluid/ISuperTokenFactory.sol#2)
        - ≥0.7.0 (contracts/interfaces/superfluid/ISuperfluid.sol#2)
        - ABIEncoderV2 (contracts/interfaces/superfluid/ISuperfluid.sol#4)
        - ^0.7.0 (contracts/interfaces/superfluid/ISuperfluidGovernance.sol#2)
        - ≥0.7.0 (contracts/interfaces/superfluid/ISuperfluidToken.sol#2)
        - ≥0.5.0 (contracts/interfaces/tokens/ERC20WithTokenInfo.sol#2)
        - ^0.5.0 (contracts/interfaces/tokens/TokenInfo.sol#2)
        - 0.7.6 (contracts/utils/TOGA.sol#2)
        - 0.7.6 (contracts/utils/TokenCustodian.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

ContextDefinitions.decodeCallInfo(uint256) (contracts/interfaces/superfluid/Definitions.sol#83-89) is never used and should be removed
ContextDefinitions.encodeCallInfo(uint8,uint8) (contracts/interfaces/superfluid/Definitions.sol#93-96) is never used and should be removed
SuperAppDefinitions.getAppLevel(uint256) (contracts/interfaces/superfluid/Definitions.sol#27-29) is never used and should be removed
SuperAppDefinitions.isAppJailed(uint256) (contracts/interfaces/superfluid/Definitions.sol#32-34) is never used and should be removed
SuperfluidGovernanceConfigs.getAppFactoryConfigKey(address) (contracts/interfaces/superfluid/Definitions.sol#186-190) is never used and should be removed
SuperfluidGovernanceConfigs.getAppRegistrationConfigKey(address,string) (contracts/interfaces/superfluid/Definitions.sol#179-184) is never used and should be removed
SuperfluidGovernanceConfigs.getTrustedForwarderConfigKey(address) (contracts/interfaces/superfluid/Definitions.sol#173-177) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version≥0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/introspection/IERC1820Registry.sol#3) is too complex
Pragma version≥0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version≥0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777.sol#3) is too complex
Pragma version≥0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC777/IERC777Recipient.sol#3) is too complex
Pragma version≥0.7.0 (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#2) allows old versions
Pragma version≥0.7.0 (contracts/interfaces/superfluid/Definitions.sol#2) allows old versions
Pragma version≥0.7.0 (contracts/interfaces/superfluid/ISuperAgreement.sol#2) allows old versions
Pragma version≥0.7.0 (contracts/interfaces/superfluid/ISuperApp.sol#2) allows old versions
Pragma version≥0.7.0 (contracts/interfaces/superfluid/ISuperToken.sol#2) allows old versions
Pragma version≥0.7.0 (contracts/interfaces/superfluid/ISuperTokenFactory.sol#2) allows old versions
Pragma version≥0.7.0 (contracts/interfaces/superfluid/ISuperfluid.sol#2) allows old versions
Pragma version≥0.7.0 (contracts/interfaces/superfluid/ISuperfluidGovernance.sol#2) allows old versions
Pragma version≥0.7.0 (contracts/interfaces/superfluid/ISuperfluidToken.sol#2) allows old versions
Pragma version≥0.5.0 (contracts/interfaces/tokens/ERC20WithTokenInfo.sol#2) allows old versions
Pragma version≥0.5.0 (contracts/interfaces/tokens/TokenInfo.sol#2) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Constant SuperfluidGovernanceConfigs.CFAv1_LIQUIDATION_PERIOD_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#167-168) is not in UPPER_CASE_WITH_UNDERSCORES
Variable TOGA._currentPICs (contracts/utils/TOGA.sol#123) is not in mixedCase
Variable TOGA._host (contracts/utils/TOGA.sol#124) is not in mixedCase
Variable TOGA._cfa (contracts/utils/TOGA.sol#125) is not in mixedCase
Variable TOGA.ERC777_SEND_GAS_LIMIT (contracts/utils/TOGA.sol#129) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

TOGA.slitherConstructorVariables() (contracts/utils/TOGA.sol#117-334) uses literals with too many digits:
        - ERC777_SEND_GAS_LIMIT = 3000000 (contracts/utils/TOGA.sol#129)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

IConstantFlowAgreementV1 (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#13-219) does not implement functions:
        - IConstantFlowAgreementV1.createFlow(ISuperfluidToken,address,int96,bytes) (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#61-68)
        - IConstantFlowAgreementV1.deleteFlow(ISuperfluidToken,address,address,bytes) (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#190-197)
        - IConstantFlowAgreementV1.getAccountFlowInfo(ISuperfluidToken,address) (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#148-157)
        - IConstantFlowAgreementV1.getDepositRequiredForFlowRate(ISuperfluidToken,int96) (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#39-43)
        - IConstantFlowAgreementV1.getFlow(ISuperfluidToken,address,address) (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#109-120)
        - IConstantFlowAgreementV1.getFlowByID(ISuperfluidToken,bytes32) (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#131-141)
        - IConstantFlowAgreementV1.getMaximumFlowRateFromDeposit(ISuperfluidToken,uint256) (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#24-28)
        - IConstantFlowAgreementV1.getNetFlow(ISuperfluidToken,address) (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#165-170)
        - ISuperAgreement.realtimeBalanceOf(ISuperfluidToken,address,uint256) (contracts/interfaces/superfluid/ISuperAgreement.sol#26-37)
        - IConstantFlowAgreementV1.updateFlow(ISuperfluidToken,address,int96,bytes) (contracts/interfaces/agreements/IConstantFlowAgreementV1.sol#89-96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

SuperAppDefinitions.APP_LEVEL_FINAL (contracts/interfaces/superfluid/Definitions.sol#22) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_LEVEL_SECOND (contracts/interfaces/superfluid/Definitions.sol#25) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_JAIL_BIT (contracts/interfaces/superfluid/Definitions.sol#31) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AGREEMENT_CALLBACK_NOOP_BITMASKS (contracts/interfaces/superfluid/Definitions.sol#39) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_CREATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#40) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_CREATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#41) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_UPDATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#42) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_UPDATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#43) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.BEFORE_AGREEMENT_TERMINATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#44) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.AFTER_AGREEMENT_TERMINATED_NOOP (contracts/interfaces/superfluid/Definitions.sol#45) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_REGISTRATION_ONLY_IN_CONSTRUCTOR (contracts/interfaces/superfluid/Definitions.sol#51) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_REGISTRATION_FOR_EOA (contracts/interfaces/superfluid/Definitions.sol#52) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_REVERT_ON_TERMINATION_CALLBACK (contracts/interfaces/superfluid/Definitions.sol#53) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_CRITICAL_SENDER_ACCOUNT (contracts/interfaces/superfluid/Definitions.sol#54) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_NO_CRITICAL_RECEIVER_ACCOUNT (contracts/interfaces/superfluid/Definitions.sol#55) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_READONLY (contracts/interfaces/superfluid/Definitions.sol#56) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_NOT_CLEAN (contracts/interfaces/superfluid/Definitions.sol#57) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_CTX_IS_MALFORMATED (contracts/interfaces/superfluid/Definitions.sol#58) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_COMPOSITE_APP_IS_NOT_WHITELISTED (contracts/interfaces/superfluid/Definitions.sol#59) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_COMPOSITE_APP_IS_JAILED (contracts/interfaces/superfluid/Definitions.sol#60) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
SuperAppDefinitions.APP_RULE_MAX_APP_LEVEL_REACHED (contracts/interfaces/superfluid/Definitions.sol#61) is never used in SuperAppDefinitions (contracts/interfaces/superfluid/Definitions.sol#7-62)
ContextDefinitions.CALL_INFO_CALL_TYPE_AGREEMENT (contracts/interfaces/superfluid/Definitions.sol#79) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
ContextDefinitions.CALL_INFO_CALL_TYPE_APP_ACTION (contracts/interfaces/superfluid/Definitions.sol#80) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
ContextDefinitions.CALL_INFO_CALL_TYPE_APP_CALLBACK (contracts/interfaces/superfluid/Definitions.sol#81) is never used in ContextDefinitions (contracts/interfaces/superfluid/Definitions.sol#67-98)
SuperfluidGovernanceConfigs.SUPERFLUID_REWARD_ADDRESS_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#164-165) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
SuperfluidGovernanceConfigs.CFAv1_LIQUIDATION_PERIOD_CONFIG_KEY (contracts/interfaces/superfluid/Definitions.sol#167-168) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
SuperfluidGovernanceConfigs.SUPERTOKEN_MINIMUM_DEPOSIT_KEY (contracts/interfaces/superfluid/Definitions.sol#170-171) is never used in SuperfluidGovernanceConfigs (contracts/interfaces/superfluid/Definitions.sol#162-191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

flush(IERC777,address) should be declared external:
        - TokenCustodian.flush(IERC777,address) (contracts/utils/TokenCustodian.sol#22-30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- No major issues were found by Slither. High-risk vulnerabilities detected are false positives.

ERC20 checks:

SuperToken.sol

The SuperToken contract is a standard ERC20 and ERC777 token.

```
# Check SuperToken

## Check functions
[v] totalSupply() is present
        [v] totalSupply() → () (correct return value)
        [v] totalSupply() is view
[v] balanceOf(address) is present
        [v] balanceOf(address) → () (correct return value)
        [v] balanceOf(address) is view
[v] transfer(address,uint256) is present
        [v] transfer(address,uint256) → () (correct return value)
        [v] Transfer(address,address,uint256) is emitted
[v] transferFrom(address,address,uint256) is present
        [v] transferFrom(address,address,uint256) → () (correct return value)
        [v] Transfer(address,address,uint256) is emitted
[v] approve(address,uint256) is present
        [v] approve(address,uint256) → () (correct return value)
        [v] Approval(address,address,uint256) is emitted
[v] allowance(address,address) is present
        [v] allowance(address,address) → () (correct return value)
        [v] allowance(address,address) is view
[v] name() is present
        [v] name() → () (correct return value)
        [v] name() is view
[v] symbol() is present
        [v] symbol() → () (correct return value)
        [v] symbol() is view
[v] decimals() is present
        [v] decimals() → () (correct return value)
        [v] decimals() is view

## Check events
[v] Transfer(address,address,uint256) is present
        [v] parameter 0 is indexed
        [v] parameter 1 is indexed
[v] Approval(address,address,uint256) is present
        [v] parameter 0 is indexed
        [v] parameter 1 is indexed


        [v] SuperToken has increaseAllowance(address,uint256)
```

As we can see above, the token properly follows the standards.

5.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

BatchLiquidator.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

ConstantFlowAgreementV1.sol

Line	SWC Title	Severity	Short Description
24	(SWC-123) Requirement Violation	Low	Requirement violation.
109	(SWC-123) Requirement Violation	Low	Requirement violation.

Definitions.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
40	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
41	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
42	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
43	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
44	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
45	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
130	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
139	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
149	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
159	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered

InstandDistributionAgreementV1.sol

Line	SWC Title	Severity	Short Description
25	(SWC-123) Requirement Violation	Low	Requirement violation.
1000	(SWC-123) Requirement Violation	Low	Requirement violation.

MaticBridgedNativeSuperToken.sol

Line	SWC Title	Severity	Short Description
35	(SWC-123) Requirement Violation	Low	Requirement violation.

NativeSuperToken.sol

Line	SWC Title	Severity	Short Description
26	(SWC-123) Requirement Violation	Low	Requirement violation.
32	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.

Superfluid.sol

Line	SWC Title	Severity	Short Description
301	(SWC-115) Authorization through tx.origin	Low	Use of "tx.origin" as a part of authorization control.
352	(SWC-115) Authorization through tx.origin	Low	Use of "tx.origin" as a part of authorization control.

SuperfluidGovernanceBase.sol

Line	SWC Title	Severity	Short Description
41	(SWC-123) Requirement Violation	Low	Requirement violation.
41	(SWC-107) Reentrancy	Low	A call to a user-supplied address is executed.

SuperfluidOwnableGovernance.sol

Line	SWC Title	Severity	Short Description
13	(SWC-123) Requirement Violation	Low	Requirement violation.

Superupgrader.sol

Line	SWC Title	Severity	Short Description
36	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
37	(SWC-110) Assert Violation	Unknown	Out of bounds array access
38	(SWC-110) Assert Violation	Unknown	Out of bounds array access
105	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
106	(SWC-110) Assert Violation	Unknown	Out of bounds array access

TOGA.sol

Line	SWC Title	Severity	Short Description
171	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
171	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
178	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
185	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
241	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
249	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
251	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered

TokenCustodian.sol

Line	SWC Title	Severity	Short Description
47	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered

- The floating pragma flagged by MythX is a false positive, as the pragma is set in the truffle-config.js file to the 0.7.6 version.

THANK YOU FOR CHOOSING

// HALBORN