Syndicate
Protocol v2

SMART CONTRACT AUDIT

18.08.2022

Made in Germany by Chainsulting.de

Table of contents

1. Disclaimer 4
2. About the Project and Company 5
   2.1 Project Overview 6
3. Vulnerability & Risk Level 7
4. Auditing Strategy and Techniques Applied 8
   4.1 Methodology 8
5. Metrics 9
   5.1 Tested Contract Files 9
   5.2 Used Code from other Frameworks/Smart Contracts 11
   5.3 CallGraph 14
   5.4 Inheritance Graph 17
   5.5 Source Lines & Risk 18
   5.6 Capabilities 19
   5.7 Source Unites in Scope 21
6. Scope of Work 25
   6.1 Findings Overview 26
   6.2 Manual and Automated Vulnerability Test 27
   6.2.1 Overpowered Owner Rights 27
   6.2.2 Maximum Member Count 28
   6.2.3 Redundant Overflow Check 29
   6.2.4 Unused Function Parameter 30
   6.2.5 Missing Value Verification 31

6.2.6 Uninitialized Variables 34
6.3 SWC Attacks 35
6.4 Verify Claims 39
7. Executive Summary 40
8. Deployed Smart Contract 40
9. About the Auditor 41

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of SYNDICATE INC. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (25.07.2022)	Layout
0.4 (29.07.2022)	Automated Security Testing Manual Security Testing
0.5 (30.07.2022)	Verify Claims and Test Deployment
0.6 (03.08.2022)	Testing SWC Checks
0.9 (05.08.2022)	Summary and Recommendation
1.0 (06.08.2022)	Final document
1.1 (18.08.2022)	Re-check
1.2 (TBA)	Deployed contracts

2. About the Project and Company

Company address:
SYNDICATE INC.
1049 El Monte Avenue Ste C #560
Mountain View, CA 94040
USA

Website: https://syndicate.io

Twitter: https://twitter.com/SyndicateDAO

Discord: https://discord.gg/aB89kn7bvV

Documentation: https://guide.syndicate.io

2.1 Project Overview

Syndicate is a decentralized investing protocol and social network that's transforming the investing world.

Within the next decade, investing will be decentralized, democratized, and community-driven. Syndicate is building infrastructure that will empower communities to raise, coordinate, and invest capital like never before. This will have a profound impact by expanding what gets funded and built in the world—by whom, for whom, and where.

It'll level the playing field, empower communities of builders, creatives, influencers, and friends to invest in new ideas and world-changing technologies, and introduce radical new models for investing to society that are fundamentally more open, free, and fair. Syndicate is building Web3-native tools to create and manage Investing DAOs. Our first product is Web3 Investment Clubs, which is just the first step in developing a full suite of DAO tools for Web3 communities to invest and allocate capital in Web3-native ways.

For the first time, we can democratize investing with Syndicate's integrated platform and ecosystem of applications, tools, services, and Web3 protocols.

The first product is Web3 Investment Clubs, which allows anyone to create and manage an investment club on Ethereum in seconds.

- An unlimited number of on-chain investment clubs can be created instantly, on-demand
- Run as DAOs on Ethereum with a wallet (Metamask, Gnosis) and Syndicate's smart contracts
- Syndicate's smart contracts automate the management of deposits, cap tables, reporting, distributions, and more
- Optional legal infrastructure enables legal entity creation for off-chain investments and supports compliance
- Composable with other Syndicate tools and the broader web3 ecosystem

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

5. Metrics

The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

5.1 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
src/contracts/guards/common/GuardMixinManager.sol	ef6484417342585d39ce87e6b6527ef2
src/contracts/guards/mixins/LockableSettings.sol	e4b0e3022965ebb4bd431f3b0356ece5
src/contracts/guards/mixins/TokenGated.sol	0f7747762ae33297b18ba62308ad657d
src/contracts/modules/ERC721Collective/mint/MerkleDistributorModuleERC721.sol	dadbb188de4f969026c23ec812f2a238
src/contracts/modules/ERC721Collective/mint/EthPriceMintModule.sol	b97770df127fc9a009e52028245cf60c
src/contracts/modules/utils/FeeCalculator.sol	a21c1591e48feebc9d788a7a6ea0cc00
src/contracts/modules/utils/Constants.sol	726f2237afbfe2e8666da90d956b8de3
src/contracts/modules/utils/IFeeManager.sol	29ed08d419f80bec0874d1dc63f0751a
src/contracts/modules/utils/FeeManager.sol	707375a8ad419792540c8a184df02800
src/contracts/modules/ERC20Club/mint/DepositExchangeTokenMintModule.sol	5ee6d6977167d1836a574e9dc57d941b
src/contracts/modules/ERC20Club/mint/BatchMintModuleERC20.sol	da591fd9fa44488759f8e79b0be5314d
src/contracts/modules/ERC20Club/mint/DepositTokenMintModule.sol	c5665624a1276853317e8bd698c4901a

src/contracts/modules/ERC20Club/mint/EthMintModule.sol	26f4e7aafcbfd97cf36e94e59188dace
src/contracts/ERC721Collective/factory/ERC721CollectiveFactory.sol	f7db56e398666d7b30f22885772a77bc
src/contracts/ERC721Collective/ERC721UpgradeableFork.sol	2bcb9a0d31758f00b64f4cbb00406069
src/contracts/ERC721Collective/ERC721Collective.sol	d81744cbfec070c9a6247e93f9bff47f
src/contracts/common/TokenEnforceableUpgradeable.sol	a2fc1f69e288ee125930d899486f0fab
src/contracts/common/ITokenRecoverable.sol	e220259816471e345be5692b0dfe1ccf
src/contracts/common/ITokenEnforceable.sol	0696c80de6afd18f2aa42261ad6d4c45
src/contracts/common/TokenRecoverable.sol	284a88dab5a80ea7376ccb80a648e3e0
src/contracts/common/IERC1644.sol	f4d1e96eff815cac35863da1867e628b
src/contracts/utils/TokenOwnerChecker.sol	ba24e63c3976f642982772fb392eb84e
src/contracts/ERC20Club/factory/ERC20ClubFactoryEth.sol	69b69b6039aa9114b22a9b797c6d051e
src/contracts/ERC20Club/factory/ERC20ClubFactoryCustom.sol	1b61e9823ec3b36e6966d51a149799c8
src/contracts/ERC20Club/factory/ERC20ClubFactoryBase.sol	c2fbee9759a6cfbc7f4d4795db1c9f8c
src/contracts/ERC20Club/factory/ERC20ClubFactory.sol	0bd295698cd1dbc050dfce6be305d216
src/contracts/ERC20Club/factory/ERC20ClubFactoryDepositToken.sol	8889eb5af239470778c1fed73fb1440f
src/contracts/ERC20Club/ERC20Club.sol	d3383ba9be8d7ae27d6bd90d3bdcf8c7
src/contracts/ERC20Club/ERC165CheckerERC20Club.sol	2517a83b8f1d666267a766108c38dd3e

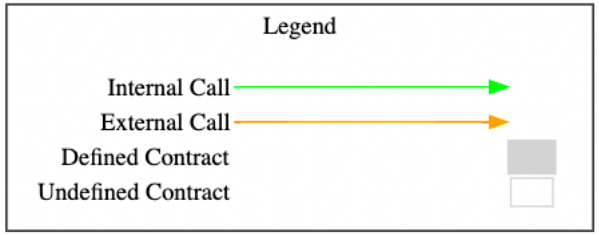
5.2 Used Code from other Frameworks/Smart Contracts (direct imports)

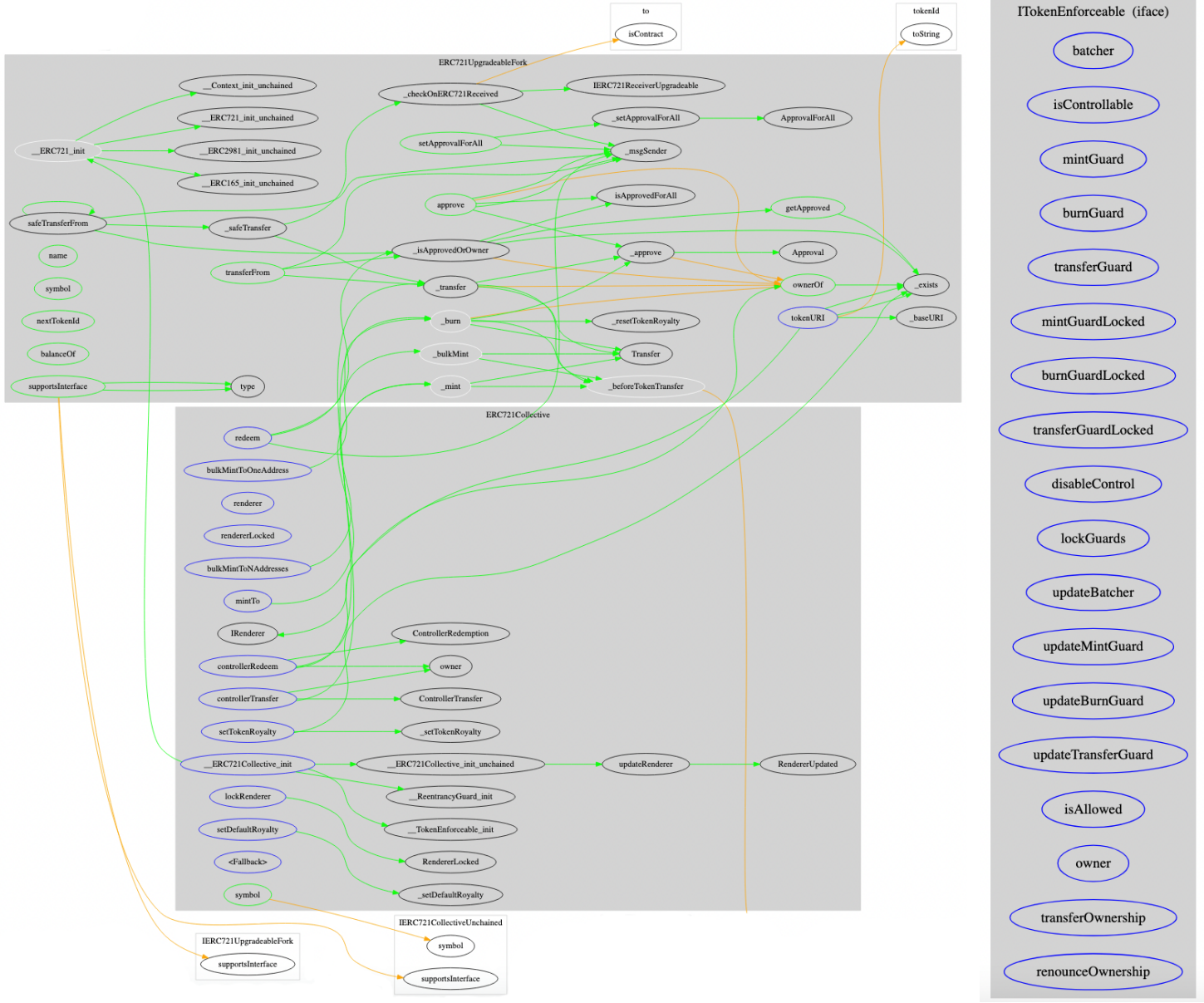
Dependency / Import Path	Source
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/access/OwnableUpgradeable.sol
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/proxy/utils/Initializable.sol
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/security/ReentrancyGuardUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/token/ERC721/IERC721ReceiverUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/common/ERC2981Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/token/common/ERC2981Upgradeable.sol
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/utils/AddressUpgradeable.sol
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/utils/ContextUpgradeable.sol

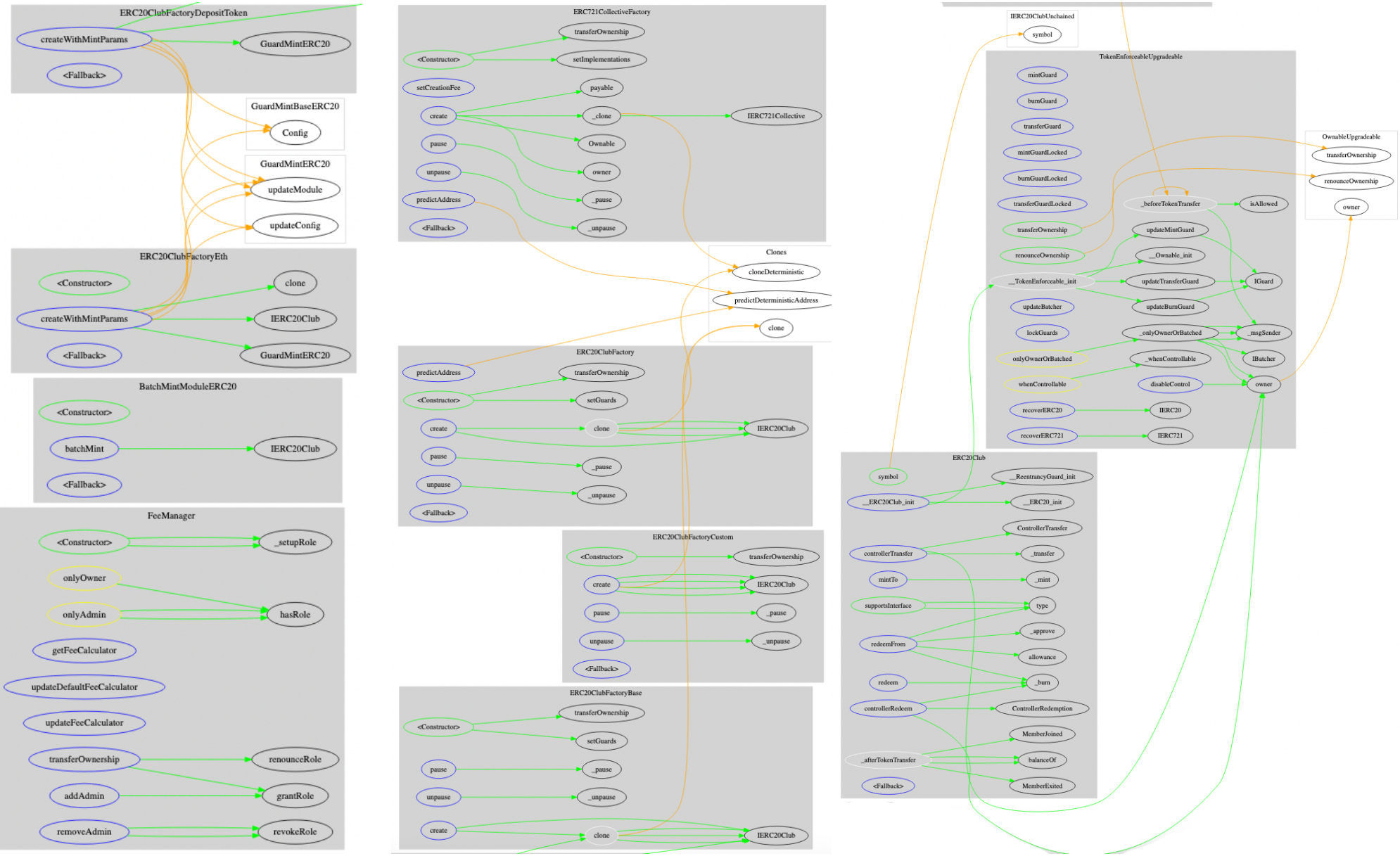
Dependency / Import Path	Source
@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/utils/StringsUpgradeable.sol
@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/utils/introspection/ERC165Upgradeable.sol
@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/utils/introspection/IERC165Upgradeable.sol
@openzeppelin/contracts/access/AccessControl.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/access/AccessControl.sol
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/access/Ownable.sol
@openzeppelin/contracts/proxy/Clones.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/proxy/Clones.sol
@openzeppelin/contracts/security/Pausable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/security/Pausable.sol
@openzeppelin/contracts/security/ReentrancyGuard.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/security/ReentrancyGuard.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/token/ERC20/utils/SafeERC20.sol

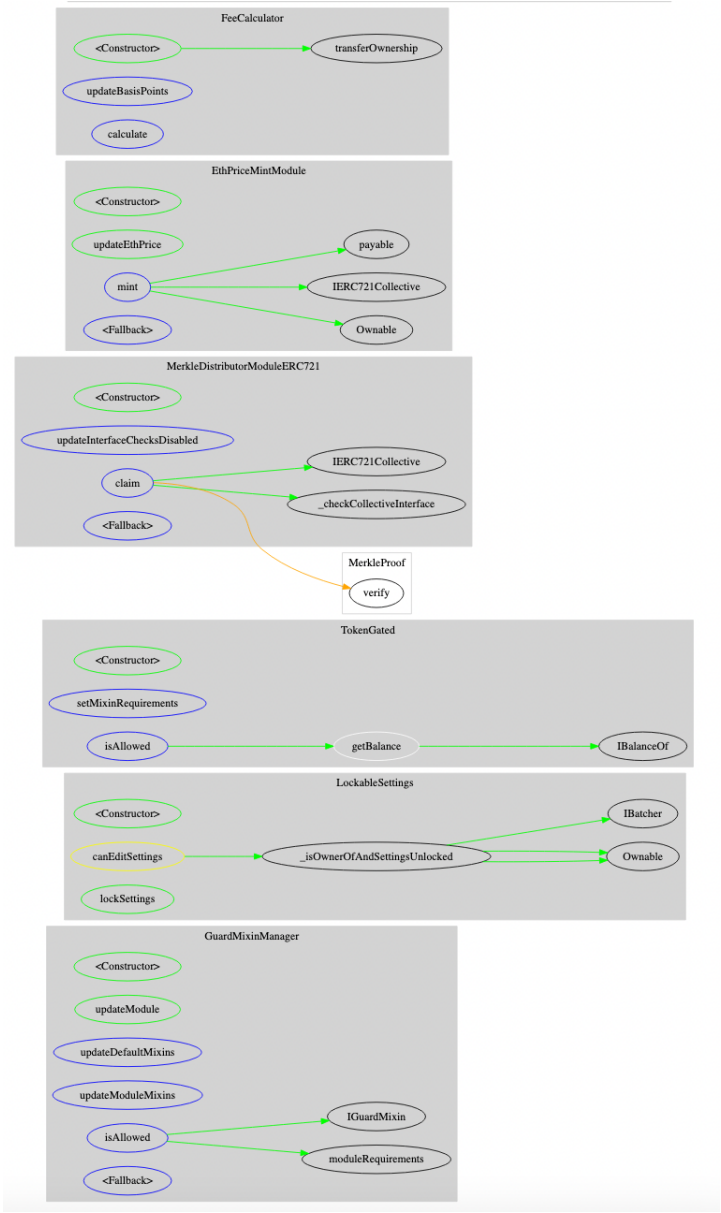
Dependency / Import Path	Source
@openzeppelin/contracts/token/ERC721/IERC721.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/token/ERC721/IERC721.sol
@openzeppelin/contracts/utils/cryptography/MerkleProof.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/utils/cryptography/MerkleProof.sol
@openzeppelin/contracts/utils/introspection/ERC165Checker.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/utils/introspection/ERC165Checker.sol
@openzeppelin/contracts/utils/structs/BitMaps.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/utils/structs/BitMaps.sol

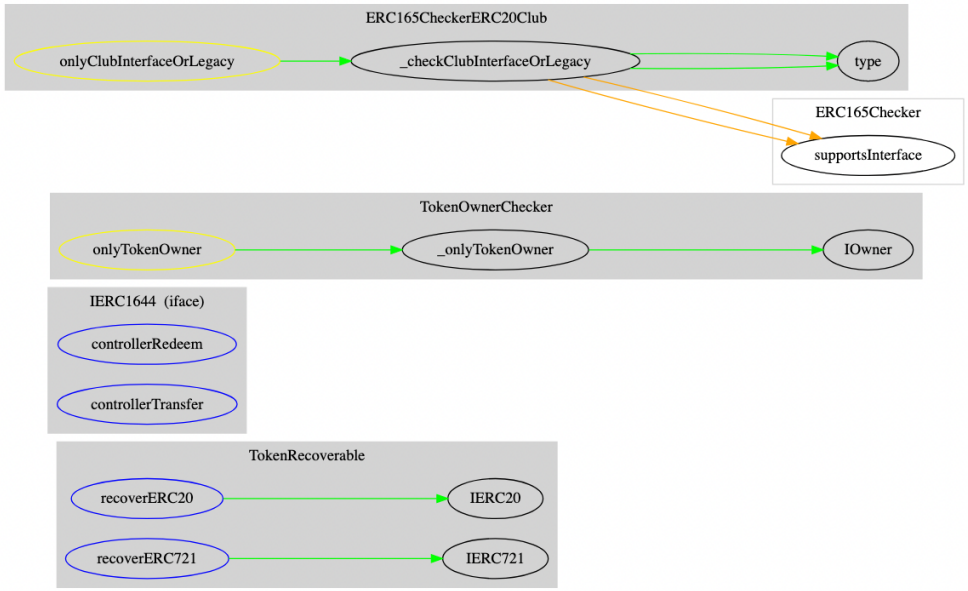
5.3 CallGraph



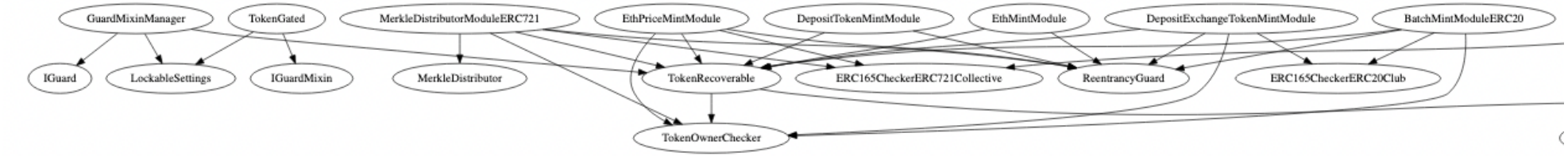


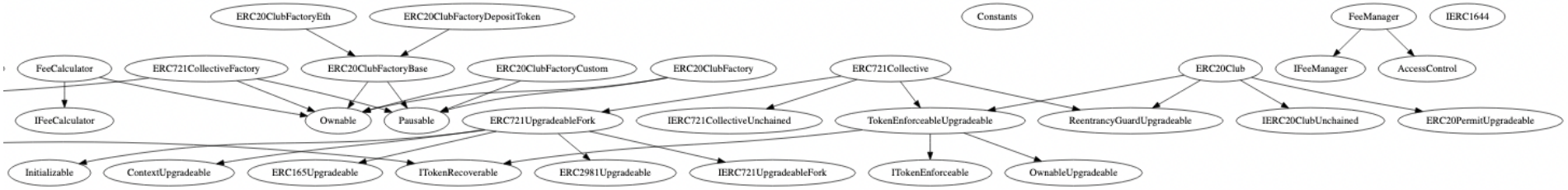

ERC20ClubFactoryDepositToken
createWithMintParams
GuardMintERC20
<Fallback>
GuardMintBaseERC20
Config
GuardMintERC20
updateModule
updateConfig
ERC20ClubFactoryEth
<Constructor>
createWithMintParams
<Fallback>
clone
IERC20Club
GuardMintERC20
BatchMintModuleERC20
<Constructor>
batchMint
<Fallback>
IERC20Club
FeeManager
<Constructor>
onlyOwner
onlyAdmin
getFeeCalculator
updateDefaultFeeCalculator
updateFeeCalculator
transferOwnership
addAdmin
removeAdmin
_setupRole
hasRole
renounceRole
grantRole
revokeRole
ERC721CollectiveFactory
<Constructor>
setCreationFee
create
pause
unpause
predictAddress
<Fallback>
transferOwnership
setImplementations
payable
_clone
Ownable
owner
_pause
_unpause
IERC721Collective
Clones
cloneDeterministic
predictDeterministicAddress
clone
ERC20ClubFactory
predictAddress
<Constructor>
create
pause
unpause
<Fallback>
transferOwnership
setGuards
clone
_pause
_unpause
IERC20Club
ERC20ClubFactoryCustom
<Constructor>
create
pause
unpause
<Fallback>
transferOwnership
IERC20Club
_pause
_unpause
ERC20ClubFactoryBase
<Constructor>
pause
unpause
create
transferOwnership
setGuards
_pause
_unpause
clone
IERC20Club
IERC20ClubUnchained
symbol
TokenEnforceableUpgradeable
mintGuard
burnGuard
transferGuard
mintGuardLocked
burnGuardLocked
transferGuardLocked
transferOwnership
renounceOwnership
__TokenEnforceable_init
updateBatcher
lockGuards
onlyOwnerOrBatched
whenControllable
recoverERC20
recoverERC721
_beforeTokenTransfer
updateMintGuard
__Ownable_init
updateTransferGuard
updateBurnGuard
_onlyOwnerOrBatched
_whenControllable
disableControl
isAllowed
IGuard
_msgSender
IBatcher
owner
IERC20
IERC721
OwnableUpgradeable
transferOwnership
renounceOwnership
owner
ERC20Club
symbol
__ERC20Club_init
controllerTransfer
minTo
supportsInterface
redeemFrom
redeem
controllerRedeem
_afterTokenTransfer
<Fallback>
__ReentrancyGuard_init
__ERC20_init
ControllerTransfer
_transfer
_mint
type
_approve
allowance
_burn
ControllerRedemption
MemberJoined
balanceOf
MemberExited


FeeCalculator
<Constructor>
transferOwnership
updateBasisPoints
calculate
EthPriceMintModule
<Constructor>
updateEthPrice
payable
mint
IERC721Collective
<Fallback>
Ownable
MerkleDistributorModuleERC721
<Constructor>
updateInterfaceChecksDisabled
IERC721Collective
claim
_checkCollectiveInterface
<Fallback>
MerkleProof
verify
TokenGated
<Constructor>
setMixinRequirements
isAllowed
getBalance
IBalanceOf
LockableSettings
<Constructor>
IBatcher
canEditSettings
_isOwnerOfAndSettingsUnlocked
Ownable
lockSettings
GuardMixinManager
<Constructor>
updateModule
updateDefaultMixins
updateModuleMixins
IGuardMixin
isAllowed
moduleRequirements
<Fallback>

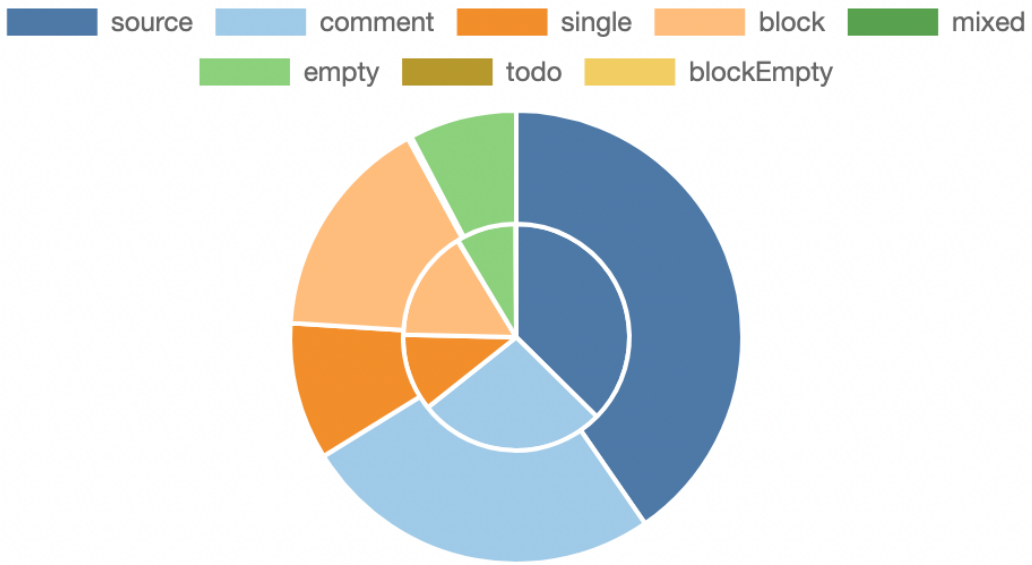

ERC165CheckerERC20Club
onlyClubInterfaceOrLegacy
_checkClubInterfaceOrLegacy
type
ERC165Checker
supportsInterface
TokenOwnerChecker
onlyTokenOwner
_onlyTokenOwner
IOwner
IERC1644 (iface)
controllerRedeem
controllerTransfer
TokenRecoverable
recoverERC20
IERC20
recoverERC721
IERC721

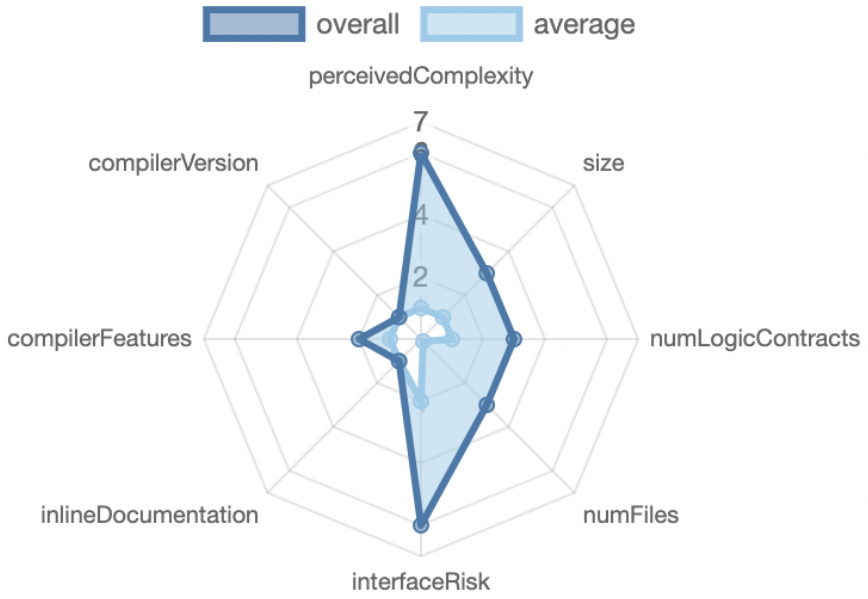
5.4 Inheritance Graph





5.5 Source Lines & Risk





5.6 Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
`0.8.15` `^0.8.0`		`yes`	`yes` (1 asm blocks)	

Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
			`yes`		

TryCatch	Σ Unchecked
`yes`	`yes`

Inline Documentation

- **Comment-to-Source Ratio:** On average there are `1.56` code lines per comment (lower=better).
- **ToDo's:** `0`

Components

Contracts	Libraries	Interfaces	Abstract
17	1	4	7

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
149	3

External	Internal	Private	Pure	View
112	161	1	1	58

StateVariables

Total	🌐Public
62	48

5.7 Source Unites in Scope

Source: https://github.com/SyndicateProtocol/protocol-v2
Last commit: 2969c6f552393bcbed57ab5b8386101f8fb4b2cb

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	src/contracts/guards/common/GuardMixinManager.sol	1		143	122	71	33	54	Σ
🎨	src/contracts/guards/mixins/LockableSettings.sol	1		64	61	22	27	15	
📝	src/contracts/guards/mixins/TokenGated.sol	1		129	114	76	26	38	Σ
📝	src/contracts/modules/ERC721Collective/mint/MerkleDistributorModuleERC721.sol	1		94	85	58	15	36	🧱
📝	src/contracts/modules/ERC721Collective/mint/EthPriceMintModule.sol	1		77	67	41	16	33	💰
📝	src/contracts/modules/utils/FeeCalculator.sol	1		47	42	23	9	17	
📚	src/contracts/modules/utils/Constants.sol	1		8	8	5	1	2	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	src/contracts/modules/utils/IFeeManager.sol		1	7	6	3	1	3	🔆
🎨	src/contracts/modules/utils/FeeManager.sol	1		112	106	67	24	50	
📝	src/contracts/modules/ERC20Club/mint/DepositExchangeTokenMintModule.sol	1		226	206	134	49	74	
📝	src/contracts/modules/ERC20Club/mint/BatchMintModuleERC20.sol	1		60	55	37	11	25	Σ
📝	src/contracts/modules/ERC20Club/mint/DepositTokenMintModule.sol	1		141	132	80	35	41	
📝	src/contracts/modules/ERC20Club/mint/EthMintModule.sol	1		104	94	57	23	37	💰
📝	src/contracts/ERC721Collective/factory/ERC721CollectiveFactory.sol	1		232	216	126	66	74	💰Σ
📝	src/contracts/ERC721Collective/ERC721UpgradeableFork.sol	1		720	634	250	316	169	💻♻️
📝	src/contracts/ERC721Collective/ERC721Collective.sol	1		408	350	136	192	108	Σ

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🎨	src/contracts/common/TokenEnforceableUpgradeable.sol	1		405	361	146	180	113	
🔍	src/contracts/common/ITokenRecoverable.sol		1	55	31	13	28	5	
🔍	src/contracts/common/ITokenEnforceable.sol		1	194	25	16	129	37	
🎨	src/contracts/common/TokenRecoverable.sol	1		54	46	17	24	16	
🔍	src/contracts/common/IERC1644.sol		1	57	36	14	31	5	
🎨	src/contracts/utils/TokenOwnerChecker.sol	1		28	28	14	9	5	
📝	src/contracts/ERC20Club/factory/ERC20ClubFactoryEth.sol	1		105	95	53	32	22	
📝	src/contracts/ERC20Club/factory/ERC20ClubFactoryCustom.sol	1		89	77	44	22	35	
🎨	src/contracts/ERC20Club/factory/ERC20ClubFactoryBase.sol	1		132	116	74	28	43	
📝	src/contracts/ERC20Club/factory/ERC20ClubFactory.sol	1		170	150	93	40	57	**Σ**

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	src/contracts/ERC20Club/factory/ERC20ClubFactoryDepositToken.sol	1		113	102	60	34	24	
📝	src/contracts/ERC20Club/ERC20Club.sol	1		196	160	73	70	69	**Σ**
🎨	src/contracts/ERC20Club/ERC165CheckerERC20Club.sol	1		35	35	26	5	8	
📝📚🔍🎨	**Totals**	**25**	**4**	**4205**	**3560**	**1829**	**1476**	**1215**	🖥️💰🧱🔆♻️**Σ**

Legend: [➖]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

6. Scope of Work

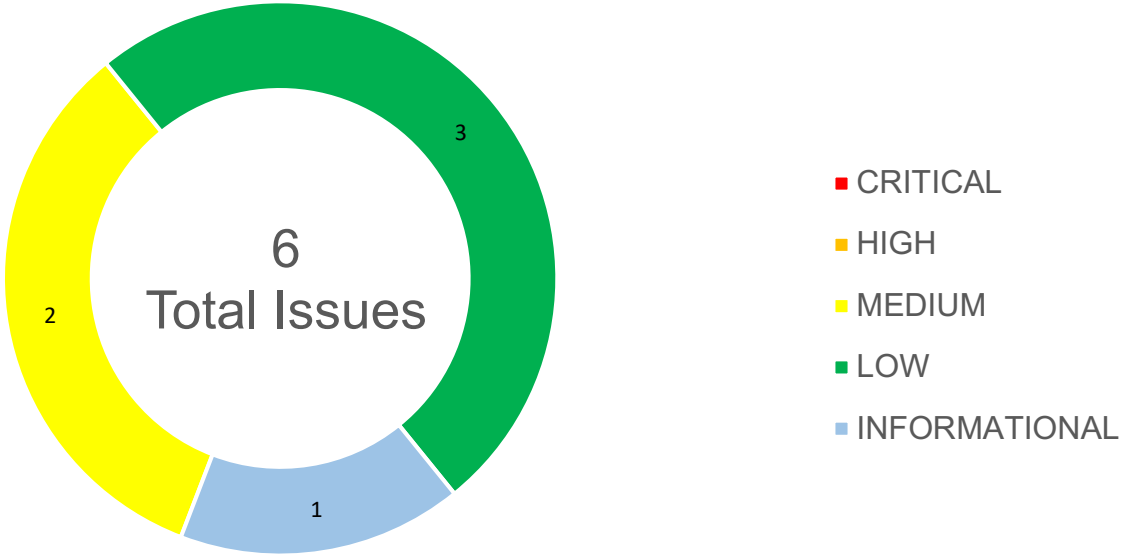
The Syndicate DAO Team provided us with the files that needs to be tested. The scope of the audit are the syndicate protocol v2 contracts.

The team put forward the following assumptions regarding the security, usage of the contracts:

- ERC20 Club is working as expected
- ERC721 Collective is working as expected
- Fees are working as expected
- The smart contract is coded according to the newest standards and in a secure way.

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.

6.1 Findings Overview



No	Title	Severity	Status
6.2.1	Overpowered Owner Rights	MEDIUM	ACKNOWLEDGED
6.2.2	Maximum Member Count	MEDIUM	FIXED
6.2.3	Redundant Overflow Check	LOW	FIXED
6.2.4	Unused Function Parameter	LOW	ACKNOWLEDGED
6.2.5	Missing Value Verification	LOW	ACKNOWLEDGED
6.2.6	Uninitialized Variables	INFORMATIONAL	ACKNOWLEDGED

6.2 Manual and Automated Vulnerability Test

CRITICAL ISSUES
During the audit, Chainsulting‘s experts found **0 Critical issues** in the code of the smart contract.

HIGH ISSUES
During the audit, Chainsulting’s experts found **0 High issues** in the code of the smart contract.

MEDIUM ISSUES
During the audit, Chainsulting’s experts found **2 Medium issues** in the code of the smart contract.

6.2.1 Overpowered Owner Rights
Severity: MEDIUM
Status: ACKNOWLEDGED
Code: CWE-284
File(s) affected: ERC20Club/ERC20Club.sol
Update: Acknowledged, however we want to keep the controller functions there by design. Additionally, if the club owner is malicious (or got hacked) then there are no security guarantees for that club anyways. Lastly, isControllable can be irreversibly set to false, so at least controller functions can be basically disabled.

Attack/Description	The ERC20Club token holds functions, that permits specific addresses to transfer or burn tokens of other users, when the contract is in controllable state.
Code	Line 65 - 85 (ERC20Club/ERC20Club.sol) `function controllerTransfer(` `address sender,` `address recipient,`

	uint256 amount) external onlyOwnerOrBatched whenControllable { _transfer(sender, recipient, amount); emit ControllerTransfer(owner(), sender, recipient, amount); } function controllerRedeem(address account, uint256 amount) external onlyOwnerOrBatched whenControllable { _burn(account, amount); emit ControllerRedemption(owner(), account, amount); }
Result/Recommendation	It is recommended to remove such overpowered owner functionalities to ensure the users full control over their funds.

6.2.2 Maximum Member Count
Severity: MEDIUM
Status: FIXED
Code: NA
File(s) affected: guard/mixins/MaxMemberCount.sol
Update: https://github.com/SyndicateProtocol/protocol-v2/commit/02e1309f0f4c1b3a73cae33816820f58f9462f84

Attack/Description	The isAllowed function is missing a require, as at the moment it's not allowed for member to send all their tokens to a new member, in the case the member count is at the maximum.
Code	Line 40 - 52 (guard/mixins/MaxMemberCount.sol)

	`` function isAllowed(address token_, address, // operator address, // from address to, uint256 // value) external view override returns (bool) { uint256 max = maxMemberCount[token_] == 0 ? DEFAULT_MEMBER_COUNT : maxMemberCount[token_]; IERC20Club token = IERC20Club(token_); return token.balanceOf(to) > 0 || token.memberCount() < max; } ```
Result/Recommendation	It is recommended to change the require to token.balanceOf(to) > 0 || token.memberCount() < max || token.balanceOf(from) == value

LOW ISSUES

During the audit, Chainsulting's experts found **3 Low issues** in the code of the smart contract.

6.2.3 Redundant Overflow Check
Severity: LOW
Status: FIXED
Code: CWE-190
File(s) affected: modules/ERC20Club/mint/EthMintModule.sol
Update: https://github.com/SyndicateProtocol/protocol-v2/commit/3d2e7ce95e15fd51f49852b38f55bcc9c134dbec

Attack/Description	The getMintAmount function checks for an potential overflow before calling a safe multiply operation. Since Solidity version 0.8.0, arithmetical operation are automatically checking for over- and underflows.
Code	Line 35 - 45 (modules/ERC20Club/mint/EthMintModule.sol) `function getMintAmount(uint256 depositAmount)` `public` `pure` `returns (uint256)` `{` `require(` `depositAmount < type(uint256).max / ETH_CLUB_ERC20_EXCHANGE_RATE,` `"EthMintModule: Deposit amount will cause overflow"` `);` `return (depositAmount * ETH_CLUB_ERC20_EXCHANGE_RATE);` `}`
Result/Recommendation	It is recommended to remove the redundant overflow check by removing either the require statement or adding an unchecked block for the multiplication.

6.2.4 Unused Function Parameter

Severity: LOW
Status: ACKNOWLEDGED
Code: CWE-1164, SWC-131
File(s) affected: modules/utils/IFeeCalculator.sol
Update: Acknowledged, in the future we would find it useful to have the parameter since there may be different fee calculation logic based on the msg.sender.

Attack/Description	The calculate function takes two parameters: minter and depositAmount. The minter address is never used in the implementation of the function and can be removed.
Code	Line 5 - 10 (modules/utils/IFeeCalculator.sol) `interface IFeeCalculator {` `  function calculate(address minter, uint256 depositAmount)` `  external` `  view` `  returns (uint256);` `}`
Result/Recommendation	It is recommended to remove all unused code to keep the projects clean and well arranged. Remove the minter parameter of the calculate function.

6.2.5 Missing Value Verification
Severity: LOW
Status: ACKNOWLEDGED
Code: CWE-345
File(s) affected: common/TokenEnforceableUpgradeable.sol, modules/ERC20Club/mint/DepositTokenMintModule.sol, modules/utils/FeeManager.sol, guards/mixins/LockableSettings.sol
Update: Acknowledged, would be helpful but not sure if worth the implementation for two reasons: 1) Some of the previously deployed guards & mixins contracts are not EIP-165 compatible so there may be some backwards compatibility issues, 2) The contracts calls for those contracts are view functions, i.e. isAllowed() and calculateFee() so state changes cannot occur there anyways

Attack / Description	Certain functions lack a value safety check. Therefore, only values that are consistent with the logic of the contract should be permitted.
Code	Line 224 - 268 (common/TokenEnforceableUpgradeable.sol) `function updateMintGuard(address implementation) public onlyOwnerOrBatched {`

```
        require(!_mintGuard.isLocked, "TokenEnforceable: mint guard is locked");
        _mintGuard.guard = IGuard(implementation);
        emit GuardUpdated(GuardType.Mint, implementation);
    }


function updateBurnGuard(address implementation) public onlyOwnerOrBatched {
        require(!_burnGuard.isLocked, "TokenEnforceable: burn guard is locked");
        _burnGuard.guard = IGuard(implementation);
        emit GuardUpdated(GuardType.Burn, implementation);
    }

function updateTransferGuard(address implementation)
        public
        onlyOwnerOrBatched
    {
        require(
            !_transferGuard.isLocked,
            "TokenEnforceable: transfer guard is locked"
        );
        _transferGuard.guard = IGuard(implementation);
        emit GuardUpdated(GuardType.Transfer, implementation);
    }
```

Line 96 - 102 (modules/ERC20Club/mint/DepositTokenMintModule.sol)

```
  function setTreasuryAddress(address club, address treasuryAddress)
        external
        onlyTokenOwner(club)
    {
        clubTreasuryAddress[club] = treasuryAddress;
        emit TreasuryAddressUpdated(club, treasuryAddress);
    }
```

	Line 23 - 27 (modules/utils/FeeManager.sol) `constructor(address owner_, address defaultFeeCalculator_) {` `  defaultFeeCalculator = defaultFeeCalculator_;` `  _setupRole(OWNER, owner_);` `  _setupRole(ADMIN, owner_);` `  }` Line 57 - 74 (modules/utils/FeeManager.sol) `function updateDefaultFeeCalculator(address defaultFeeCalculator_)` `  external` `  onlyOwner` `  {` `  defaultFeeCalculator = defaultFeeCalculator_;` `  emit DefaultFeeCalculatorUpdated(defaultFeeCalculator_);` `  }` `function updateFeeCalculator(address token, address fee)` `  external` `  onlyAdmin` `  {` `  feeCalculator[token] = fee;` `  emit FeeCalculatorUpdated(token, fee);` `  }` Line 21 - 23 (guards/mixins/LockableSettings.sol) `constructor(address _batcher) {` `  batcher = _batcher;` `  }`
Result/Recommendation	It is recommended to check address values for correctness. This can be done in first stage to exclude zero address in a require statement. In second stage to check if an address is a contract. and most specific if an address implements a specified interface (EIP-165).

INFORMATIONAL ISSUES

During the audit, Chainsulting's experts found **1 Informational issue** in the code of the smart contract.

6.2.6 Uninitialized Variables
Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: NA
File(s) affected: ERC20Club/factory/ERC20ClubFactory.sol, ERC721Collective/factory/ERC721CollectiveFactory.sol, guards/mixins/TokenGated.sol
Update: Acknowledged, we feel like this isn't necessary for now.

Attack/Description	Certain functions lack of explicit variable initialization. Implicitly, uint variables are set to zero.
Code	Line 138 (ERC20Club/factory/ERC20ClubFactory.sol) `for (uint256 i; i < length; ) {` Line 121 (ERC721Collective/factory/ERC721CollectiveFactory.sol) `for (uint256 i; i < length; ) {` Line 54, 112 (guards/mixins/TokenGated.sol) for (uint256 i; i < length;) { ... }
Result/Recommendation	It is recommended to set all variables explicitly to enhance code readability and avoid unintended behaviour. It is recommended to set the *i* variable to zero explicitly.

6.3 SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	X
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✅
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✅
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✅
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✅
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✅
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✅
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✅

ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✅
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✅
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✅
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✅
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✅
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✅

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✅
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✅
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✅
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✅
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✅
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✅
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✅
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✅

ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✅
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✅
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✅
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅

6.4 Verify Claims

6.4.1 ERC20 Club is working as expected
Status: tested and verified ✅

6.4.2 ERC721 Collective is working as expected
Status: tested and verified ✅

6.4.3 Fees are working as expected
Status: tested and verified ✅

6.4.4 The smart contract is coded according to the newest standards and in a secure way.
Status: tested and verified ✅

7. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase.

The main goal of the audit was to verify the claims regarding the security and functions of the smart contract. During the audit, no critical, no high, two medium, three low and one Informational issue have been found, after the manual and automated security testing. We advise the syndicate team to implement the recommendations to further enhance the code's security and readability.

8. Deployed Smart Contract

PENDING

9. About the Auditor

Chainsulting is a professional software development firm, founded in 2017 and based in Germany. They show ways, opportunities, risks and offer comprehensive blockchain solutions. Some of their services include blockchain development, smart contract audits and consulting.

Chainsulting conducts code audits on market-leading blockchains such as Hyperledger, Tezos, Ethereum, Binance Smart Chain, and Solana to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secure the smart contracts of 1Inch, POA Network, Unicrypt, Amun, Furucombo among numerous other top DeFi projects.

Chainsulting currently secures $100 billion in user funds locked in multiple DeFi protocols. The team behind the leading audit firm relies on their robust technical know-how in the blockchain sector to deliver top-notch smart contract audit solutions, tailored to the clients' evolving business needs.

Check our website for further information: https://chainsulting.de