BADGER FINANCE

SMART CONTRACT SECURITY ANALYSIS

FEBRUARY 22nd, 2021

Project Summary

Project Name	Badger Finance
Scope	A community DAO focused on developing products and infrastructure to bring Bitcoin to the DeFi space.
Platform	Ethereum, Solidity

Executive Summary

- Twelve smart contracts were analyzed to check the availability of code vulnerabilities associated with the process of funds staking:
  BADGER, **DIGG**, **UFragmentsPolicy**, **Staking Reward**, **StrategyBadgerRewards**, **Controller**, **SmartVesting**, **SimpleTimelockWithVoting**, **RewardsEscrow**, **BadgerTree**, **BadgerHunt**, **Sett**.
- No significant security issues were revealed in the aforementioned contracts.

Smart Contract Ownership	Team Reward	Total Supply	Minting Function	Migration Function	Funds Lock Period	Contract Pause	Suspicious Functions
Specific for each contract	10% of the total BADGER supply	Specific for each token	Available via voting	Available through Aragon	None	Available for staking	Not found

External Smart Contract Audits:

- Zokyo smart contract **audit**
- HAECHI smart contract audit **report**

Manual Check Results

Ownership structure:

Smart contract	Owner	Description
BADGER	badger.aragonid.eth	This is a token contract that was created and is managed by DAOFactory. The contract: • invokes the mint and burn functions after appropriate voting results; • creates new votes by token holders; • manages voting by voting; • manages the project through voting.
DIGG	GnosisSafe	The following functions can be invoked by the owner: • **renounceOwnership** / **transferOwnership**; • **setMonetaryPolicy** installs a contract address of a new *monetary policy*; • Monetary policy is UFragmentsPolicy`s AdminUpgradeabilityProxy. It can call the rebase function.
UFragmentsPolicy	GnosisSafe	The following functions can be invoked by the owner: • **renounceOwnership** / **transferOwnership**; • **setMonetaryPolicy** installs a contract address of a new *monetary policy*; • **setCpiOracle**; • **setMarketOracle**; • **setOrchestrator**; • **setDeviationThreshold** sets the deviation threshold fraction. If the exchange rate given by the market oracle is within this fractional distance from the **targetRate**, then no supply modifications are made. • **setRebaseLag**; • **setRebaseTimingParameters**; • *cpiOracle* is MedianOracle; • *marketOracle* is MedianOracle[2].
Staking Reward	GnosisSafe	The following functions can be invoked by the owner: • **pause/unpause** allows or disallows to stake funds (withdrawals are always available); • **setRewardsDuration** updates reward duration period; • **recoverERC20** transfers any ERC20 token (except staking and reward tokens) from the contract to the *admin*; • **notifyRewardAmount** adds new reward tokens to the contract.
StrategyBadger Rewards	Controller Badger: Guardian EOA Badger.Deployer GnosisSafe	• *Geyser* is the StakingReward smart contract; • *Controller* is Controller`s AdminUpgradeabilityProxy; • *Controller* could call next functions: • **withdrawAll** withdraws the total supply of tokens from the contract to a relied address defined in *Controller*; • **withdraw** withdraws a specified amount of tokens from the contract to a relied address defined in *Controller* with applying a fee; • **withdrawOther** withdraws a specific amount of non-core tokens from the contract to *Controller*; • **deposit** transfers the total amount to Geyser (StakingReward). *Authorized Pausers* act like a *guardian* (Badger: Guardian EOA), a *strategist* (Badger.Deployer EOA) and *governance* (OwnersGnosisSafeMultisig). They can call the pause and unpause functions. Pause is applied to the deposit and withdraw functions of the contract? (to/from the *geyser* StakingReward contract)

		Governance is OwnersGnosisSafeMultisig. This address can call the following functions: • **setStrategist**; • **setKeeper**; • **setGovernance**; • **setGuardian**; • **setWithdrawalFee** can set any number as the withdrawal fee, the current one is 0; • **setPerformanceFeeStrategist** can set any number as the fee, the current one is 0; • **setPerformanceFeeGovernance** can set any number as the fee, the current one is 0; • **setController**.
Controller	GnosisSafe	The following functions can be invoked by the owner: • **approveStrategy** approves a strategy to be applied to tokens based on voting results; • **revokeStrategy** removes a strategy that was bound to tokens based on voting results; • **setRewards** sets receiver addresses for a fee that is applied on withdrawals from vaults; • **setSplit** sets a fee percentage that's transferred to the **oneSplit** audit contract. The current fee is 5%; • **setOneSplit** - the *oneSplit* audit contract address; • **setVault** defines *vault* address for a specific token; • **setStrategy** migrates tokens from the existing strategy to a newly approved strategy; • **setConverter** sets the contract used to convert between two given tokens; • **withdrawAll** calls the **withdrawAll** function in the strategy contract that is bound to a selected token; • **inCaseTokensGetStuck** transfers a specified amount of tokens from Controller to a sender; • **inCaseStrategyTokenGetStuck** calls the **withdrawOther** function in the strategy contract that is bound to a selected token. *Strategist* is a Badger.Deployer and can call: • **setVault** defines *vault* address for a specific token; • **setStrategy** migrates tokens from the existing strategy to a newly approved strategy; • **setConverter** sets the contract used to convert between two given tokens; • **withdrawAll** calls the **withdrawAll** function in the strategy contract that is bound to a selected token; • **inCaseTokensGetStuck** transfers a specified amount of tokens from *Controller* to a sender • **inCaseStrategyTokenGetStuck** calls the **withdrawOther** function in the strategy contract that is bound to a selected token;
SmartVesting	GnosisSafe	The following functions can be invoked by the owner: • **call** allows Timelock to call arbitrary contracts, as long as it does not reduce it's locked token balance; • **claimToken** transfers selected ERC20 token from the contract to a *beneficiary*, except locked tokens; • **claimEther** transfers all ETHs from the contract to a *beneficiary*. The *governor* is an Aragon`s Agent. He could call **approveTransfer** and **revokeTransfer** that allow or disallow transfers from this contract by specified addresses. • The token lock duration is 1 year (until January 2, 2022); • Linear token releasement.

SimpleTimelock WithVoting	Agent	• *The beneficiary* is an Aragon`s Agent; • The release time is January 2, 2021; • The release target is an Aragon`s Agent.
RewardsEscrow	GnosisSafe	The following functions can be invoked by the owner: • **call** allows RewardsEscrow to call arbitrary contracts, as long as it does not reduce it's token balance; • **transfer** transfers tokens to *approved recipients* (distribution pools); • **signalTokenLock** calls **signalTokenLock** function in a selected *geyser* contract; • **approveRecipient** / **revokeRecipient** allows and disallows transfers for *approved recipient*.
BadgerTree	Badger.Deployer Badger: Keeper Badger: Guardian	*Admin* is the Badger.Deployer EOA that can call the following functions: • **grantRole** adds new members for existing roles. *Root updater* is the Badger: Keeper EOA that can call the following functions: • **proposeRoot** proposes a new root and a content hash, which will be stored as pending until approved. *Guardian* is the Badger: Guardian EOA that can call following functions: • **approveRoot** approves the current pending root and content hash; • **pause** / **unpause** allows and disallows the claim function; • **claim** withdraws a selected amount of tokens to msg.sender. It requires the Merkle proof verification.
BadgerHunt	GnosisSafe	The following functions can be invoked by the owner: • **renounceOwnership** / **transferOwnership**; • **recycleExcess**. After a hunt is complete, it transfers excess funds to rewardsEscrow; • **setGracePeriod** influences epoch duration.
Sett	GnosisSafe	The following functions can be invoked by the owner: • **approveContractAccess** / **revokeContractAccess** makes a contract available on not available to be used in Sett; • **setMin** sets a minimum threshold of underlying that must be deposited in strategy; • **setController**; • **setStrategist**; • **setKeeper**; • **setGovernance**.

Total supply:
- DIGG: Variable due to the rebase functionality
- BADGER: 21 000 000

Minting function:
- Available through the governance voting

Migration function:
- Available and can be executed via the Aragon functionality

Team reward:
- 10% of the total BADGER supply

The risk of a quick token dump initiated by the team:
- **2** / 10

Funds lock period:
- None

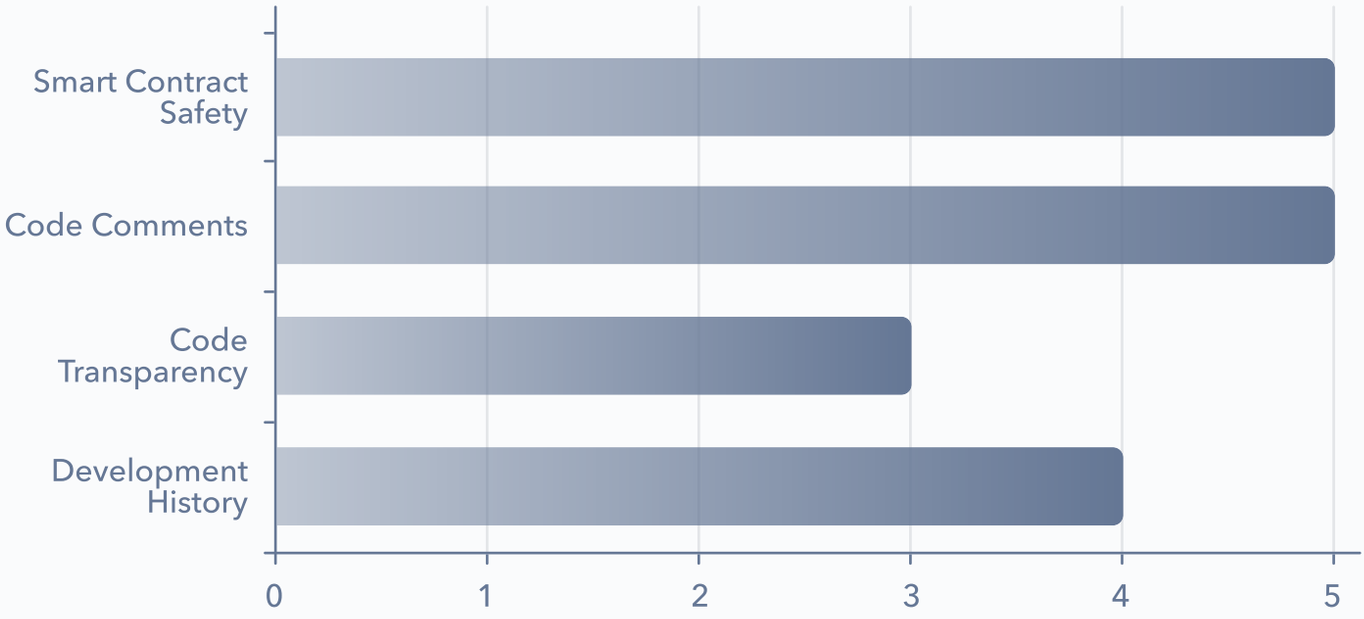
Possibility to pause the Smart Contracts:
- Available only for staking

Suspicious functions:
- Not found

Tornado cash connections:
- Not found

⚠ Risk Level LOW

Smart Contracts



Conclusion

Badger Finance is a community-driven protocol. The project has a complicated structure divided into a few ecosystem parts.

The governance of the project is run through an Aragon DAO, where voting rights are given to holders of a liquid governance token - $BADGER.

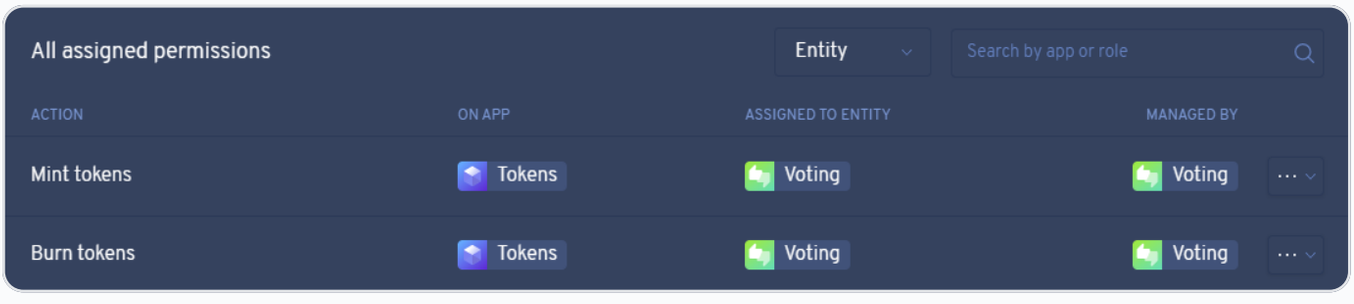
Another component of the ecosystem is Sett - a yield aggregator that proposes liquidity mining using Uniswap and Sushiswap.

The second protocol token is $DIGG - a BTC-pegged elastic supply token.
The total supply of the DIGG token is variable due to the rebase functionality.

etherscan.io/address/0xfabec03b04279c6e73f27aaf25866acc844448ae#code

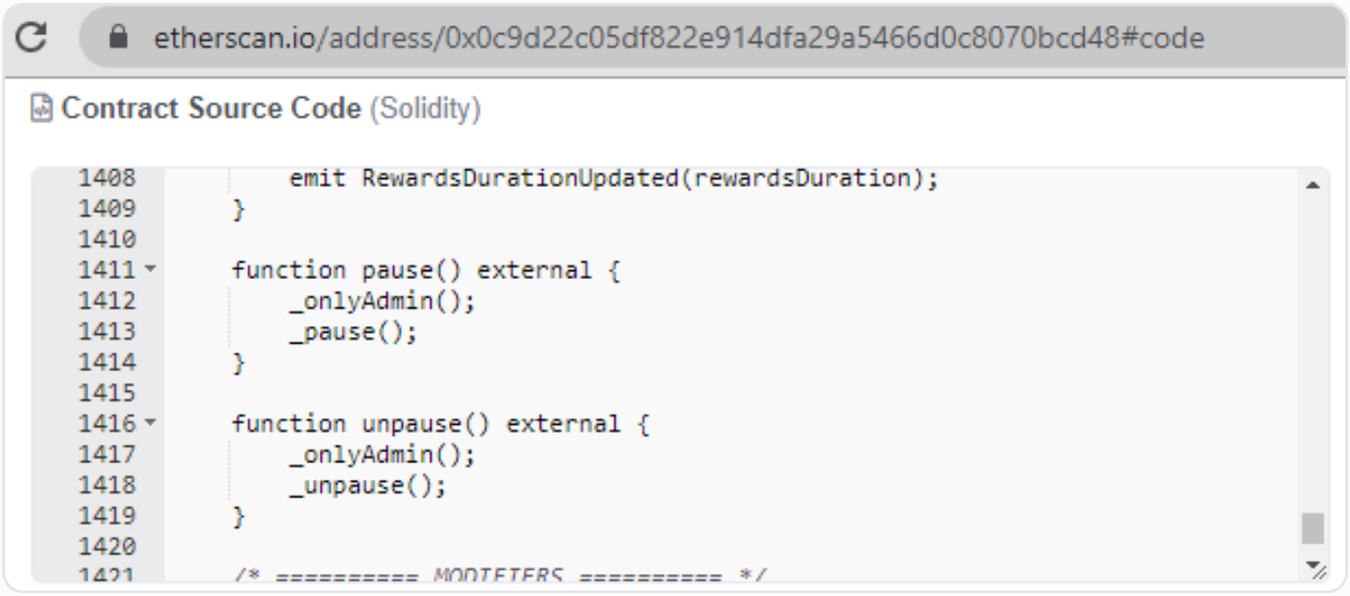
```
497     /**
498      * @dev Notifies Fragments contract about a new rebase cycle.
499      * @param supplyDelta The number of new fragment tokens to add into circulation via expansion.
500      * @return The total number of fragments after the supply adjustment.
501      */
502     function rebase(uint256 epoch, int256 supplyDelta) external onlyMonetaryPolicy onlyAfterRebaseStart returns (uint256) {
503         if (supplyDelta == 0) {
504             emit LogRebase(epoch, _totalSupply);
505             return _totalSupply;
506         }
507
508         if (supplyDelta < 0) {
509             _totalSupply = _totalSupply.sub(uint256(supplyDelta.abs()));
510         } else {
511             _totalSupply = _totalSupply.add(uint256(supplyDelta));
512         }
513
514         if (_totalSupply > MAX_SUPPLY) {
515             _totalSupply = MAX_SUPPLY;
516         }
517
518         _sharesPerFragment = TOTAL_SHARES.div(_totalSupply);
519
520         // From this point forward, _sharesPerFragment is taken as the source of truth.
521         // We recalculate a new _totalSupply to be in agreement with the _sharesPerFragment
522         // conversion rate.
523         // This means our applied supplyDelta can deviate from the requested supplyDelta,
524         // but this deviation is guaranteed to be < (_totalSupply^2)/(TOTAL_SHARES - _totalSupply).
525         //
526         // In the case of _totalSupply <= MAX_UINT64 (our current supply cap), this
527         // deviation is guaranteed to be < 1, so we can omit this step. If the supply cap is
528         // ever increased, it must be re-included.
529         // NB: Digg will likely never reach the total supply cap as the total supply of BTC is
530         // currently 21 million and MAX_UINT64 is many orders of magnitude greater.
531         // _totalSupply = TOTAL_SHARES.div(_sharesPerFragment)
532
533         emit LogRebase(epoch, _totalSupply);
534         return _totalSupply;
535     }
```

The total supply of $BADGER amounts to 21 million tokens and this number is fixed. According to the project's Aragon permissions page (https://client.aragon.org/?#/badger/permissions/), the mint function can be called, but it is controlled by voting:



Migration is possible as Aragon's proxy structure allows to upgrade smart contracts to new versions. Funds lock period isn't implemented.

Possibility of pausing is only implemented to the Staking Reward smart contract, meaning it only affects the staking functionality and doesn't block withdrawals of user funds.



Badger Finance has a well-built system of smart contracts with safe code. Therefore, the risk level of the project can be estimated as low.

Audit recommendations

- Changing **Controller`s** governance address to a real governance contract.
- Changing **Sett`s** governance address to a real governance contract.