

Curve DAO

Security Assessment

July 10th 2020

Prepared For:
Michael Egorov | *Swiss Stake*
michael@swiss-stake.com

Prepared By:
Josselin Feist | *Trail of Bits*
josselin@trailofbits.com

Gustavo Grieco | *Trail of Bits*
gustavo.grieco@trailofbits.com

Michael Colburn | *Trail of Bits*
michael.colburn@trailofbits.com

[Executive Summary](#)

[Project Dashboard](#)

[Code Maturity Evaluation](#)

[Engagement Goals](#)

[Coverage](#)

[Recommendations Summary](#)

[Short term](#)

[Long term](#)

[Findings Summary](#)

- [1. LiquidityGauge does not account for VotingEscrow's balance updates](#)
- [2. LiquidityGauge does not account for VotingEscrow's totalSupply updates](#)
- [3. Early users will have a unfair advantage](#)
- [4. GaugeController allows for quick vote and withdraw voting strategy](#)
- [5. Adding the same gauge multiple times will lead to incorrect sum of weights](#)
- [6. Spam attack would prevent LiquidityGauge's integral from being updated](#)
- [7. Minter user can confiscate any user tokens](#)
- [8. Mint and Burn events cannot be trusted](#)
- [9. VotingEscrow's Admin can take whitelisted accounts hostage](#)
- [10. ERC20CRV is not initiated correctly with large name and symbol](#)
- [11. Lack of two-step procedure for critical operations is error-prone](#)
- [12. Lack of value verification on decimals is error-prone](#)
- [13. Lack of events is error-prone](#)
- [14. Race condition in removing addresses from whitelist and withdrawing](#)
- [15. Lack of documentation is error-prone](#)
- [16. VotingEscrow's balanceOfAt and totalSupplyAt can return different values for the same block](#)
- [17. No incentive to vote early in GaugeController](#)
- [18. Several loops are not executable due to gas limitation](#)
- [19. Testing smart contract code in Brownie can be unreliable](#)
- [20. Aragon's voting does not follow voting best practices](#)
- [21. Race condition may result in users earning less interest than expected](#)

[A. Vulnerability Classifications](#)

[B. Code Maturity Classifications](#)

[C. Code Quality](#)

[D. Token Integration Checklist](#)

[ERC Conformity](#)

[Contract Composition](#)

[Owner privileges](#)

[Token Scarcity](#)

[E. Fix Log](#)

[Detailed Fix Log](#)

Executive Summary

From June 22 through July 10, 2020, Swiss-Stake engaged Trail of Bits to review the security of Curve DAO. We conducted this assessment over the course of six person-weeks with three engineers working from [iggiigg](#) from the [G.L.H.H.H.G.I.J.I.L.L.G.L.L.](#) repository.

In the first two weeks, we focused on understanding the codebase and reviewing the contracts against the most common smart contract flaws. In the final week, we reviewed the checkpoint functions and [L.I.K.L.I.H.I.L.N.G.G.L.I.H](#) bookkeeping, and looked for corner cases in the most complex contract's interactions.

Our review resulted in 21 findings ranging from high to informational severity. The most significant findings are related to incorrect updating of the [L.I.K.L.I.H.I.L.N.G.G.L.I.H](#) bonus, which can allow attackers to earn unfair interest. Moreover, we found that the code would benefit from better documentation, function composition, and code readability. We also found potential risks related to out-of-gas consumption, and external risk introduced by the use of Aragon's contracts. See additional code quality issues in [Appendix C](#), and see recommendations to follow when adding arbitrary tokens in [Appendix D](#).

Overall, the codebase meets most of its security expectations. A significant effort has been made to identify potential risks and to develop suitable mitigations and tests. However, the codebase is very complex, numerous behaviors are not documented, and the arithmetic operations would benefit from high-level clarifications.

Moving forward, Trail of Bits recommends addressing the findings presented and increasing the documentation. Curve Dao must be careful with the deployment of the contracts and the interactions of its early users and their advantages. We also recommend considering an alternative to the Aragon voting contract. Finally, we recommend performing an economic assessment to make sure the monetary incentives are properly designed.

Project Dashboard

Application Summary

Name	Curve Dao
Version	igGigg
Type	Vyper contracts
Platforms	Ethereum

Engagement Summary

Dates	June 22–July 10
Method	Whitebox
Consultants Engaged	3
Level of Effort	6 person-weeks

Vulnerability Summary

Total High-Severity Issues	4	■ ■ ■ ■
Total Medium-Severity Issues	8	■ ■ ■ ■ ■ ■ ■ ■
Total Low-Severity Issues	4	■ ■ ■ ■
Total Informational-Severity Issues	4	■ ■ ■ ■
Total Undetermined-Severity Issues	1	■
Total	21	

Category Breakdown

Access Controls	2	■ ■
Auditing and Logging	3	■ ■ ■
Data Validation	13	■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
Patching	1	■
Timing	2	■ ■
Total	21	

Code Maturity Evaluation

Category Name	Description
Access Controls	Satisfactory. The codebase has a strong access controls mechanism and we found only minor concerns.
Arithmetic	Moderate. The system relies on complex arithmetic. While the use of Vyper prevents overflow and underflow flaws, we found several issues related to interest computation.
Centralization	Moderate. The contracts' owners have significant privileges. Additionally, the deployer of ERCgfiCRV will own all the tokens at deployment and will have a significant advantage.
Upgradeability	Not Applicable.
Function Composition	Moderate. Some components are written multiple times, and the codebase would benefit from code reuse.
Front-Running	Satisfactory. Most functions are not impacted by front-running, or the impact is expected. We found only one minor issue.
Monitoring	Weak. We found that Mint and Burn events could be compromised. Additionally, several components do not emit events. Finally, we were not aware of any off-chain components that monitor the contracts.
Specification	Moderate. The provided documentation omitted several behaviors, and the codebase would benefit from more thorough documentation.
Testing & Verification	Moderate. The codebase has several unit tests, but it is missing gas evaluation. No code verification was present.

Engagement Goals

The engagement was scoped to provide a security assessment of Curve DAO protocol smart contracts in the `github.com/curvefi/curve-dao-contracts` repository.

Specifically, we sought to answer the following questions:

- Are appropriate access controls set for the admin/user roles?
- Does arithmetic for internal bookkeeping operations hold?
- Is there any arithmetic overflow or underflow affecting the code?
- Can participants manipulate or block gauge or voting operations?
- Is it possible to manipulate gauges or voting by front-running transactions?
- Is it possible for participants to steal or lose tokens?
- Can participants perform denial-of-service attacks against any of the gauges or voting escrow?

Coverage

The engagement focused on the following components:

- **Liquidity gauges:** These allow users to deposit liquidity using different ERC20 tokens and get CRV tokens based on the amount locked and other factors. We reviewed the contract's interactions with users depositing liquidity to ensure proper behavior. We looked for flaws that would allow an attacker to withdraw more than deposited and prevent users from withdrawing their assets. We also focused on interest rate computation and history catch-up.
- **Controller gauge:** Liquidity gauges are created and managed by a special contract called the controller gauge. We reviewed the access control of this contract as well as interaction with the gauges once deployed. We looked for flaws in voting and checked for the proper increase of period and epoch.
- **Voting escrow:** Once users deposit liquidity, they can use mint tokens locked for a period of time in the voting escrow contract. We reviewed the consistency and the corner cases in computation of weights and verified that the locks are held in each case. We looked for flaws that would allow an attacker to unlock a deposit early, withdraw more than deposited, or prevent users from withdrawing their deposits.
- **CRV Token and Minter:** The liquidity gauge mints a CRV token every time it adds liquidity to the gauge. This contract implements a standard ERC20 token. We verified that all the expected properties are correctly implemented. We also looked for flaws that would allow a minter to mint more than the time-limited supply, and we reviewed the CRV token for its conformity to the ERC20 standard.
- **Access controls.** Many parts of the system expose privileged functionality, i.e., setting protocol parameters or managing gauges. We reviewed these functions to ensure they can only be triggered by the intended actors and that they do not contain unnecessary privileges that may be abused.
- **Arithmetic.** We reviewed calculations for logical consistency, as well as rounding issues and scenarios where reverts due to overflow may negatively impact use of the protocol.

We briefly reviewed the Curve DAO external interactions with the [Aragon contracts](#), however, their upgradability and external dependency risks were considered out of scope.

Additionally, we briefly reviewed the [Airdrop contract](#) and looked for the most common smart contract flaws.

Off-chain code components were outside the scope of this assessment.

Adding the same gauge multiple times will corrupt the gauges' weights ([TOB-CURVE-DAO-005](#)).

Ensure that the `MIN_WEIGHT` parameters always lead `weight` to be greater than `0`. Rounding to zero will allow attackers to spam the gauge and prevent users from earning interest ([TOB-CURVE-DAO-006](#)).

Remove the minter's permission to take tokens from other users, or properly document why this is necessary. This will prevent users from distrusting the contracts ([TOB-CURVE-DAO-007](#)).

Use dedicated events for minting and burning, or don't allow users to fake `mint` events. This will prevent confusion when events are used by off-chain components ([TOB-CURVE-DAO-008](#)).

Make sure users are aware that admin privileges can take whitelisted accounts hostage. This will help users better understand the risks of interacting with this contract ([TOB-CURVE-DAO-009](#)).

Check the length of the token's name and symbol in `ERC20`. This will prevent the contract from returning an unexpected name or symbol ([TOB-CURVE-DAO-010](#)).

Use a two-step procedure for all non-recoverable critical operations. This will reduce the possibility of mistakes when the users are executing critical operations ([TOB-CURVE-DAO-011](#)).

Either use a bit mask on the return of decimals, or revert if the value is greater than 255 in `DECIMALS`. This will prevent the contract from returning an unexpected number of decimals ([TOB-CURVE-DAO-012](#)).

Add events for all critical operations to monitor the contracts and detect suspicious behavior. Missing events are listed in [TOB-CURVE-DAO-013](#).

Document how to deal with whitelist removal. Consider:

- Calling `unlock` when tokens are still locked (so the attacker cannot withdraw them, even after the lock expires).
- Increase the amount of gas when calling `unlock` to reduce the window of opportunity for this issue.

This will help reduce an attacker's window of opportunity to move their tokens ([TOB-CURVE-DAO-014](#)).

❑ **Increase the documentation, including [all the identified](#) missing behavior descriptions.** This will help users and auditors understand the system better ([TOB-CURVE-DAO-015](#)).

❑ **Document `ĞĠĠĞİĞH0İAŁ` and `ŁIJŁĞİSŁĴĴİŃAŁ` must not be called on the current block.** This will prevent users from misusing the `ĞĠĠĞİĞH0İAŁ` and `ŁIJŁĞİSŁĴĴİŃAŁ` functions ([TOB-CURVE-DAO-016](#)).

❑ **Create an incentive to vote early in `ĞĠŁÍHCUIŁŁIJİİHL`.** Consider using either:

- A decreasing weight to create an advantage for early voters, or
- A blind vote.

The lack of an incentive encourages voting at the very last minute and penalizes early voters ([TOB-CURVE-DAO-017](#)).

❑ **Reduce the risks associated with out-of-gas issues.**

- Allow users to execute the history catch-up in `VIJŁİİÍELĞLIJŃGÇĞİHGİĴIJİİİŁ` without depositing or withdrawing the lock.
- Create a bot that will call `LİKŁİİİŁŃĞĠÍHgŁLHLÇĞİHGİĴIJİİİŁ` and the `VIJŁİİÍELĞLIJŃnL` history catch-up function at least once per week.
- Consider allowing iteration over the periods in multiple transactions in `ĞĠŁÍHCUIŁŁIJİİHLg`

Several contracts can be trapped if they are not called for a long time, or if `ĞĠŁÍHCUIŁŁIJİİHL` lists too many gauges ([TOB-CURVE-DAO-018](#)).

❑ **Improve Brownie test capabilities:**

- Modify Brownie to disallow automatic increase of the block timestamp and number.
- Set a reasonable default for the maximum gas used per transaction during tests.

This will improve testing of corner cases in the code where operations are executed in the same block or use a large amount of gas ([TOB-CURVE-DAO-019](#)).

❑ **Do not use the original Aragon contract. Consider:**

- Improving Aragon's voting to mitigate the issues listed in [TOB-CURVE-DAO-020](#).
- Implementing a voting contract to replace Aragon's. Perform a security assessment on the contract before deployment.

Aragon's voting contract does not meet the security requirements for Curve Dao ([TOB-CURVE-DAO-020](#)).

❑ **Add a parameter to `LİKŁİİİİŁŃĞĠÍHgHĴIJLİİŁ` to specify the minimal amount of interest to receive, or make sure off-chain components take changes in the bonus into account.** This will prevent users from receiving less interest than expected ([TOB-CURVE-DAO-021](#)).

Long term

- Write clear documentation of the different components' interactions and the dependencies of the assets, and consider an economical assessment.** This will help users and auditors to better understand how the contracts work ([TOB-CURVE-DAO-001](#), [TOB-CURVE-DAO-002](#), [TOB-CURVE-DAO-003](#)).
- Properly document the GŁÍHCIIŁŁÍJĪĨHL's voting process.** This will help prevent misconceptions of how users are allowed to use their voting weight ([TOB-CURVE-DAO-004](#), [TOB-CURVE-DAO-017](#) [TOB-CURVE-DAO-020](#)).
- Follow closely the progress made by the community on on-chain voting.** Blockchain-based online voting is a known challenge. No perfect solution has been found so far and the domain evolves quickly ([TOB-CURVE-DAO-004](#), [TOB-CURVE-DAO-017](#) [TOB-CURVE-DAO-020](#)).
- Identify, review, and minimize the permissions assigned to each privileged user, and make sure users can access the information.** This will mitigate any potential private key compromise and increase the trust users have in your contracts ([TOB-CURVE-DAO-007](#), [TOB-CURVE-DAO-009](#), [TOB-CURVE-DAO-011](#)).
- Use a blockchain monitoring system to track any suspicious behavior in the contracts.** The system relies on the correct behavior of several contracts. A monitoring system that tracks critical events and upfront-running would quickly detect any compromised system components ([TOB-CURVE-DAO-008](#), [TOB-CURVE-DAO-013](#), [TOB-CURVE-DAO-014](#)).
- Carefully review Vyper's security advisories, open issues, and the current language limitations.** This will mitigate the risk of introducing issues caused by the compiler ([TOB-CURVE-DAO-010](#), [TOB-CURVE-DAO-012](#)).
- Create an incident response plan.** This will help reduce response time in case of security incidents ([TOB-CURVE-DAO-014](#)).
- Review the contract's complete documentation and simplify its use.** This will mitigate the possibility of function misuse ([TOB-CURVE-DAO-015](#)).
- Properly test system properties when functions are called in the same block or within a short period.** This will prevent unexpected results when functions are called with a small time interval ([TOB-CURVE-DAO-016](#)).
- Improve the support of out-of-gas scenarios due to loop iterations:**

- Test the functions for their gas limit.
 - Use `GHASH` with the `high` flag.
 - Use the Echidna [gas fuzzing feature](#).
- Update `CurveDAO`'s logic to work with a large number of periods. This will help detect issues caused by very high gas consumption before deployment ([TOB-CURVE-DAO-018](#)).

☐ Carefully consider the unpredictable nature of Ethereum transactions and design your contracts so they don't depend on the transaction's ordering. An attacker can control the order of the transactions to attack the system ([TOB-CURVE-DAO-021](#)).

☐ Use a lower or higher bound on asset conversions. An attacker can control the order of the transactions to change the outcome of asset conversion ([TOB-CURVE-DAO-021](#)).

☐ Use [Echidna](#) and [Manticore](#) to test and verify:

- Time-dependent code ([TOB-CURVE-DAO-006](#), [TOB-CURVE-DAO-019](#))
- High-gas-consuming code ([TOB-CURVE-DAO-019](#))
- Gauge administration functions ([TOB-CURVE-DAO-005](#))

Several issues were found in these areas, and automated testing and verification will prevent similar issues.

Findings Summary

#	Title	Type	Severity
1	LĪKĻĪHĪĻŅĢĻĪH does not account for VIJĻĪĪĪELĢĻIJŅ's balance updates	Data Validation	Medium
2	LĪKĻĪHĪĻŅĢĻĪH does not account for VIJĻĪĪĪELĢĻIJŅ's ĻIJĢĪSĻĴĴĪŅ updates	Data Validation	Medium
3	Early users will have a unfair advantage	Data Validation	Medium
4	ĢĢĻĪHCĪJĪĻĻIJĪĪHĻ allows for quick vote and withdraw voting strategy	Data Validation	Medium
5	Adding the same gauge multiple times will lead to incorrect sum of weights	Data Validation	Medium
6	Spam attack would prevent LĪKĻĪHĪĻŅĢĻĪH's integratal from being updated	Timing	Medium
7	MĪĪĻHĻ user can confiscate any user tokens	Access Controls	High
8	MĪĪĻ and BĻĻĪ events cannot be trusted	Auditing and Logging	Low
9	VIJĻĪĪĪELĢĻIJŅ's Admin can take whitelisted accounts hostage	Access Controls	Medium
10	ERCġfCRV is not initiated correctly with large name and symbol	Data Validation	Low
11	Lack of two-step procedure for critical operations is error-prone	Data Validation	High
12	Lack of value verification on decimals is error-prone	Data Validation	Low
13	Lack of events is error-prone	Auditing and Logging	Informational
14	Race condition in removing addresses from whitelist and withdrawing	Timing	Informational
15	Lack of documentation is error-prone	Auditing and Logging	Informational

16	VIJLĪĪĪEL'GLIJŅ'S ĢĢĪĢĪGH0ĪAL and LIJĢĪSĶĶĪŅĀL can return different values for the same block	Data Validation	Low
17	No incentive to vote early in ĢĢĪĪHCĪJĪĻĻĪJĪĪĪĻĻĪ	Data Validation	Medium
18	Several loops are not executable due to gas limitation	Data Validation	High
19	Testing smart contract code in Brownie can be unreliable	Patching	Undetermined
20	Aragon's voting does not follow voting best practices	Data Validation	High
21	Race condition may result in users earning less interest than expected	Data Validation	Informational

1. LIKLEILNGGLIH does not account for VIJLIIIELEGLIJN's balance updates

Severity: Medium

Difficulty: Low

Type: Data Validation

Finding ID: TOB-CURVE-DAO-001

Target: LIKLEILNGGLIHgLN

Description

VIJLIIIELEGLIJN's balance update is not accounted for in LIKLEILNGGLIH, so an attacker can earn more interest than they should by moving their VIJLIIIELEGLIJN tokens.

LIKLEILNGGLIH computes the interest earned by users. A bonus is applied for VIJLIIIELEGLIJN token holders:

```

HMI  ctJHGLHcIILNcIILaGHALF  GHALHLd  If  tILghhd  Lf  tILghhf
n  TUJ  GH  GGIHA  GILHL  LJLGISJIN  IL  tJHGH
cLJLIIIELEGLIJNf  GHALHL  z  LHIgLLIIIELEGLIJN
LJLIIIELEGLIJNf  tILghh  z  ERCglacLJLIIIELEGLIJNagGGIGIGHOiaGHAla
LJLIIIELEGLIJNf  tILghh  z  ERCglacLJLIIIELEGLIJNagLJLGISJINaa

IIf  tILghh  z  i  a  gl  w  gfl
II  LJLIIIELEGLIJN  n  fl
    III  kl  L  a  LJLIIIELEGLIJN  w  LJLIIIELEGLIJN  a  if  w  gfl

Ii  z  IIAid  Iia

```

Figure 1.1: LIKLEILNGGLIHgLNfLihhLi.

Users receive VIJLIIIELEGLIJN tokens by locking their CRV tokens for a given period of time. Once the locking period is complete, they can withdraw their tokens.

The withdrawal of VIJLIIIELEGLIJN tokens does not decrease the bonus applied to the interest rate in LIKLEILNGGLIH. As a result, an attacker can make a profit by re-using the tokens in the system to earn more interest, or by selling them while still earning the interest.

Exploit Scenario

The system has four users. Three of them have the same amount of liquidity tokens (100) and CRV locked (100):

- Alice: 100 LT, 100 Locked: working_balance = 60
- Bob: 100 LT, 100 Locked: working_balance = 60
- Eve 1: 50 LT, 100 Locked: working_balance = 50
- Eve 2: 50 LT, 0 Locked: working_balance = 10
- Carl: 0 LT, 300 Locked: working_balance = 60

Once the lock on Eve's first account ends, she deposits the CRV tokens in her second account. As a result, she has two accounts with a total working balance of 100 units when she should earn only 60 units.

Recommendation

Short term, consider either:

1. Removing the bonus based on the locked tokens,
2. Adding watchers that will penalize users cheating the system, or
3. Integrating the locking end time in the bonus computation.

Solutions (2) and (3) require significant modifications in the codebase and should be implemented with caution. Issues [TOB-CURVE-DAO-002](#) and [TOB-CURVE-DAO-003](#) must be considered when implementing the fix.

Long term, write clear documentation of the different components' interactions and the dependencies of the assets. Consider an economical assessment.

2. LIKLEILNGGLIH does not account for VIJLEL'GLIJN's LIJGISLJIN updates

Severity: Medium

Difficulty: Low

Type: Data Validation

Finding ID: TOB-CURVE-DAO-002

Target: LIKLEILNGGLIH

Description

VIJLEL'GLIJN's LIJGISLJIN update is not accounted for in LIKLEILNGGLIH. As a result, users will not earn the expected interest.

LIKLEILNGGLIH computes the interest earned by users. A bonus is applied to VIJLEL'GLIJN token's holder:

```

HAI ctJAGLHCILKLEILNGGLIHILaGHALF GHALHL'ld if tILghhd Lf tILghhaf
n TUJ GH GGIHHA GILHL LIJGISLJIN IL tJAGLHA
cLJLEIL'GLIJNf GHALHL' z LHILJLEIL'GLIJN
LJLEIL'GLIJNf tILghh z ERCglacLJLEIL'GLIJNagGGIGIHOiaGHALa
LJLEIL'GLIJNf tILghh z ERCglacLJLEIL'GLIJNagLIJGISLJINaa

iif tILghh z i a gfl w gfl
i LJLEIL'GLIJN n fl
i k L a LJLEIL'GLIJNf w LJLEIL'GLIJN a ifl w gfl

i z iiaid iia

```

Figure 2.1: LIKLEILNGGLIHHLiLi.

The bonus is based on a percentage of a user's VIJLEL'GLIJN's tokens. VIJLEL'GLIJN can be minted and burned at any moment, changing LIJGISLJIN.

As a result, the interest bonus given when LIKLEILNGGLIH is called does not reflect the real percentage over time. This might result in unexpected opportunities.

Exploit Scenario

Bob has 20% of the VIJLEL'GLIJN locked tokens. Bob starts earning interest in LIKLEILNGGLIH. After a few days, the other users unlock their tokens. Bob now has 40% of the locked tokens, but he continues to earn interest based on 20%.

Recommendation

Short term, consider either:

1. Removing the bonus based on locked tokens, or
2. Updating the formulas to account for the total supply updates.

The second option may not be straightforward to implement and may require significant change. Issues [TOB-CURVE-DAO-001](#) and [TOB-CURVE-DAO-003](#) must be considered when implementing the fix.

Long term, write clear documentation of the different components' interactions and the asset dependencies. Consider an economical assessment.

3. Early users will have a unfair advantage

Severity: Medium

Type: Data Validation

Target: LIKtIHILNGGtIHgLN

Difficulty: Medium

Finding ID: TOB-CURVE-DAO-003

Description

The VIJLIIEELGLIJN's bonus for earned interest gives an unfair advantage to early users.

LIKtIHILNGGtIH distributes a bonus based on the user's VIJLIIEELGLIJN token percentage:

```
ÀÈÌ ÇtJHGLHçIİKtIHILNçIiIIL aGHALF GHALHL'ld İf tIILghhd Lf tIILghhaf
n TUJ GH GGIIHA GILHL LJLGIStJIN IL tJHGLHA
çLJLIIEELGLIJNf GHALHL' t LHIIGLJLIIEELGLIJN
LJLIIEELGLIJNf tIILghh t ERCgfaçLJLIIEELGLIJNagGGIGIGHOiaGHALa
LJLIIEELGLIJNf tIILghh t ERCgfaçLJLIIEELGLIJNagLJLGIStJINaa

Iiif tIILghh t İ á gfl w gflf
Ii LJLIIEELGLIJNf n flf
IiI kL L á LJLIIEELGLIJNf w LJLIIEELGLIJNf á ífl w gflf

IiI t IiIaid IiIa
```

Figure 3.1: LIKtIHILNGGtIHgLNfLiLihLi.

At launch, the ERCgflCRV contract has 100% of the token supply, so it and the first token receivers can receive a significant and unfair bonus on their interest.

Combined with [TOB-CURVE-DAO-001](#), this issue will allow early users to earn significant profits.

Exploit Scenario

Eve deploys the system, locks half of the supply, and only puts the other half in distribution. As a result, Eve earns significantly more interest than any other user.

Recommendation

Short term, consider either:

- Removing the bonus based on the locked tokens, or
- Clearly documenting that early users will have an advantage in the system.

Issues [TOB-CURVE-DAO-001](#) and [TOB-CURVE-DAO-002](#) must be considered when implementing the fix.

Long term, write clear documentation of the different components' interactions and the asset dependencies. Consider an economical assessment.

6. Spam attack would prevent LIKtIHILNGGtIH's integral from being updated

Severity: Medium

Difficulty: High

Type: Timing

Finding ID: TOB-CURVE-DAO-006

Target: LIKtIHILNGGtIHgLn

Description

An attacker spamming LIKtIHILNGGtIH can prevent the integral from being updated. As a result, users will not earn interest.

On every balance's update, LIKtIHILNGGtIHgCgIHGiJUIl is executed and updates the integral based on the time elapsed since the last update:

```
function updateIntegral() {
    let timeElapsed = Date.now() - lastUpdateTime;
    let interest = timeElapsed * rate;
    integral += interest;
    lastUpdateTime = Date.now();
}
```

Figure 6.1: LIKtIHILNGGtIHgLnfiLghLgh.

If $\Delta t > \frac{1}{rate}$, the integral will not be updated. Δt is the time elapsed since the last call to `updateIntegral` and is directly controllable by the caller.

An attacker can prevent the integral from being updated by calling the contract frequently. The attack is partially mitigated by the gas cost, but miners can perform the attack without paying any gas.

Exploit Scenario

Eve is a malicious miner, and adds a call to `LiquidityGauge` on every block. As a result, Eve prevents the `LIKtIHILNGGtIH` from earning interest.

Recommendation

Short term, ensure that the system's parameters always make $\Delta t > \frac{1}{rate}$ greater than $\frac{1}{rate}$.

Long term, take in consideration short and long times period increase in the tests, and consider using [Echidna](#) and [Manticore](#) to identify unexpected behaviors allowed by these increases.


```

L'HİİgGĞİGİGHOİäçLJÜä kİ çL-GİtH
İJİgTŁGİLİHŁàZEROçADDRESSd çLJÜd çL-GİtHä

```

Figure 7.2: ERCğfCRVgLNİfLğğfHhLğğfH

However, it is also possible to use the İİİLHŁ to take tokens from other user accounts, since the ŁŁGİLİHŁFŁJİ function has an allowance bypass hardcoded for the İİİLHŁ user:

```

dJtGİİG
ÄHİ ŁŁGİLİHŁFŁJİäçİLJÜİ f GÄÄLHŁL'çLJÜ f GÄÄLHŁL'çL-GİtH f tİİLğğhã hñ GÜJİf
  ððð
    dÄHŁ TŁGİLİHŁ ŁJİHİŁ İLJÜİ JİH GÄÄLHŁL ŁJ GİJLİHŁg
      NUJH ŁİGŁ NİİİH ŁİİL İtİGŁİJÜJ HİİLŁ Ğ TŁGİLİHŁ HŁHİŁd ŁİİL İL İJÜŁ ŁHŁİLİHÄ ĞL JHŁ
ŁİH ŁJHĞİİİGĞLİJÜİd
    ĞİÄ JLİHŁ GÜJİJİİGİŁ İİJİHİHİŁĞLİJÜİŁ İĞŃ İJÜŁ HİİL ŁİH HŁHİŁg
dJĞLĞİ çİLJÜİ GÄÄLHŁL TİH GÄÄLHŁL NİİGİ NUŁ NĞİŁ ŁJ L'HİÄ ŁJİHİŁ İLJÜİ
dJĞLĞİ çLJÜ GÄÄLHŁL TİH GÄÄLHŁL NİİGİ NUŁ NĞİŁ ŁJ ŁŁGİLİHŁ ŁJ
dJĞLĞİ çL-GİtH tİİLğğhã ŁİH ĞİJLİŁ Jİ ŁJİHİŁ ŁJ ĞH ŁŁGİLİHŁLİHÄ
  ððð
  ñ NOTEf LŃJHŁ ÄJHŁ İJÜŁ ĞİİJŃ tİÄHŁİİJŃL'
  ñ      ŁJ ŁİH İJİİJŃİİİ ŁtĞŁLĞĞLİJÜİ NUŁİÄ LÄHŁLŁ Jİ İİLtİİİGİHİŁ ĞİİJŃGİG
L'HİİgGĞİGİGHOİäçİLJÜİä hİ çL-GİtH
L'HİİgGĞİGİGHOİäçLJÜä kİ çL-GİtH
İİ İLİgL'HİÄHŁ İİ L'HİİgİİİLHŁf ñ İİİLHŁ İL ĞİİJŃHÄ ŁJ ŁŁGİLİHŁ ĞİŃLİİİİ
  ñ NOTEf LŃJHŁ ÄJHŁ İJÜŁ ĞİİJŃ tİÄHŁİİJŃL'
  ñ      ŁJ ŁİH İJİİJŃİİİ ŁtĞŁLĞĞLİJÜİ NUŁİÄ LÄHŁLŁ Jİ İİLtİİİGİHİŁ ĞİİJŃGİG
L'HİİgĞİİJŃGİGHL'äçİLJÜİääİLİgL'HİÄHŁä hİ çL-GİtH
İJİgTŁGİLİHŁäçİLJÜİd çLJÜd çL-GİtHä
LÄŁtLİ TŁtH

```

Figure 7.3: ERCğfCRVgLNİfLğğfHhLğğg

Exploit Scenario

A malicious admin can silently change the minter address to steal tokens from users.

Recommendation

Short term, remove the İİİLHŁ's permission to take tokens from other users or properly document why this is necessary.

Long term, review and minimize the permissions assigned to each privileged user. This will mitigate any potential private key compromise and increase the trust from users in your contracts.

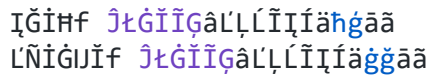
10. ERCğfCRV is not initiated correctly with large name and symbol

Severity: Low
Type: Data Validation
Target: ERCğfCRVgŁÑ

Difficulty: High
Finding ID: TOB-CURVE-DAO-010

Description

Vyper does not check the length of the string it receives and only keeps the destination size's number of elements. As a result, if ERCğfCRV is initiated with a large ŁĠİĤ or ŁÑİĠJĪ, it will have an incorrect value.



```
ŁĠİĤf JŁĠİĠĠĞaL'ŁŁİİİäĥgää
ŁÑİĠJĪf JŁĠİĠĠĞaL'ŁŁİİİäğğgää
```

Figure 10.1: ERCğfCRVgŁÑfLğğhLğğ.

Exploit Scenario

Bob deploys ERCğfCRV with a name of 65 characters, but only the first 64 characters are kept, so the token is deployed incorrectly.

Recommendation

Short term, check the length of the string.

Long term, carefully review Vyper's open issues and current language limitations.

References

- [ŁÑJĪLfiğíğfi](#)

12. Lack of value verification on decimals is error-prone

Severity: Low

Type: Data Validation

Target: `VIJḶİİÍÉḶĠUJ̣ṆĠ`

Difficulty: High

Finding ID: TOB-CURVE-DAO-012

Description

The lack of `đİİĹı` type in Vyper requires that all return values of `erc20.ĤĤĠİİĠİĠĹã` calls are checked.

`VIJḶİİÍÉḶĠUJ̣ṆĠ` calls `ĤĤĠİİĠİĠĹã` without checking the return value:

```
ḶĤİİġĤĤĠİİĠİĠĹ 1 ERCġflâıııİĤİıçĠĤĤĹăġĤĤĠİİĠİĠĹã
```

Figure 12.1: `VIJḶİİÍÉḶĠUJ̣ṆĠ` call.

`ERCġflĤĤĠİİĠİĠĹã` returns a `đİİĹı`, but this type is not handled by Vyper. As a result, the decimal value used could be invalid.

Exploit Scenario

Eve deploys a token with decimals of 520. Its decimals are read as 8 by the Solidity contract, but 520 by `VIJḶİİÍÉḶĠUJ̣ṆĠ`. As a result, `VIJḶİİÍÉḶĠUJ̣ṆĠ`'s usage is incorrect.

Recommendation

Short term, either use a bit mask on the return of decimals, or revert if the value is greater than 255.

Long term, carefully review Vyper's security advisories and the current language limitations.

References

- [VVE-2020-0001: Interfaces returning integer types less than 256 bits can be manipulated if `đİİĹġĥ` is used](#)

13. Lack of events is error-prone

Severity: Informational
Type: Auditing and Logging
Target: All contracts

Difficulty: Low
Finding ID: TOB-CURVE-DAO-013

Description

Several critical operations do not trigger events. As a result, it will be difficult to review the correct behavior of the contracts once deployed.

Critical operations that would benefit from triggering events include:

- `PIUJIPLIJNNGLHLCGHIIIL` `APIUJIPLIJNNGLNfLgfla`
- `PIUJIPLIJNNGLHLCGLLIHL` `APIUJIPLIJNNGLNfhfla`
- `ERCgflCRVgkJHGHCIIIIICJGLGIHLHL` `ERCgflCRVgLNfliiga`
- `ERCgflCRVgHLIILHL` `ERCgflCRVgLNfignga`
- `ERCgflCRVgHLIIL` `ERCgflCRVgLNfignfla`
- `GGLIHCIJILLIJIIHLGLGLIHLIHLIHLIIJ` `AGGLIHCIJILLIJIIHLGLNfLifla`
- `GGLIHCIJILLIJIIHLGLGLIHLIHLIHLIIJ` `AGGLIHCIJILLIJIIHLGLNfLggga`
- `GGLIHCIJILLIJIIHLGLGLIHLIHLIHLIIJ` `AGGLIHCIJILLIJIIHLGLNfLgiga`
- `GGLIHCIJILLIJIIHLGLJLHCIIJLCIGLIHLIHLIHLIIJ` `AGGLIHCIJILLIJIIHLGLNfLghia`
- `LIIKLIHILNGLIHLGHCIIKLIHILNIIIIIL` `LIIKLIHILNGLIHLIHLIHLIIJ`
- `VIJLIIEELGLIJNGGLIHLIHLIHLIIJ` `AVIJLIIEELGLIJNGLNfLihã`
- `VIJLIIEELGLIJNGGHHCIIJNIIHLIIIL` `AVIJLIIEELGLIJNGLNfLgflga`
- `VIJLIIEELGLIJNGHLIHLIHLIHLIIIL` `AVIJLIIEELGLIJNGLNfLgghfla`

Users and blockchain monitoring systems can't easily detect suspicious behaviors without events.

Exploit Scenario

Eve compromises the `PIUJIPLIJNNG` contract. Bob does not notice the compromise and Eve is able to change the parameter of the pool.

Recommendation

Short term, add events for all critical operations to help monitor the contracts and detect suspicious behavior.

Long term, consider using a blockchain monitoring system to track any suspicious behavior in the contracts. The system relies on the correct behavior of several contracts. A monitoring system that tracks critical events would allow quick detection of any compromised system components.

- Call `lockTokens` when tokens are still locked (so the attacker cannot withdraw them, even after the locked expires).
- Increase the amount of gas when calling `lockTokens` in order to reduce the window of opportunity.

Long term, carefully monitor the blockchain to prevent and mitigate these kinds of front-running attacks, and create an incident response plan.

15. Lack of documentation is error-prone

Severity: Informational

Difficulty: Low

Type: Auditing and Logging

Finding ID: TOB-CURVE-DAO-015

Target: L'HL'HL'G'Ī ĞIJL'L'G'G'L' ĞI'Ĥ L'ĤĜĤI'Ĥ

Description

The overall codebase lacks code documentation, high-level description, and examples. As a result, the contracts are difficult to review and the likelihood of user mistakes is high.

Several behaviors are not documented, including:

- VIJL'ĪI'ĪEL'G'L'IJN'gN'ĪL'ĪĤL'ĜN'âçL'ĜĪçĤĤ will withdraw the whole balance if çL'ĜĪçĤĤ is zero.
 - Additionally, allowing the withdrawal of only part of the locked amount is error-prone and it is unclear whether this functionality is needed.
- VIJL'ĪI'ĪEL'G'L'IJN'gĤĤĪJL'ĪL'âL'ĜĪçĤĤd çĪĪIJĜĪçL'ĪĪĤ has no documentation regarding the expected value for çĪĪIJĜĪçL'ĪĪĤ. It also fails if used with a L'ĜĪçĤĤ larger L'ĪĜĪĝ'ââĝĝĪ because the locked amounts are internally converted to ĪĪL'ĝĝĪ.
- ççL'HL'çN'ĤĪĪĪL' in GĜçĪĤCIJL'L'IJĪĪHL'gL'JL'ĤçĪIJL'çĪĜçĪĤçN'ĤĪĪĪL'âçĪĜçĪĤçĪĤd ççL'HL'çN'ĤĪĪĪL'â should be between 0 and 10,000.
- The lock time in VIJL'ĪI'ĪEL'G'L'IJN'gĤĤĪJL'ĪL' is rounded down to weeks.
- L'ĜL'ççĪIJĪL'gĜĪĜL' in VIJL'ĪI'ĪEL'G'L'IJN'gçĜĪĤĜĪĪIJĪL' can be negative due to arithmetic rounding.

The current [high-level documentation](#) would benefit from more details, including:

- User-level examples that describe who the different users are, how they interact with the contracts, and concrete scenarios highlighting usage.
- The reasoning behind some design choices, such as:
 - EL'G'L'IJN'VIJL'ĪI'Ī must not be tokenized.
 - Partial withdrawals from escrow are possible.

Exploit Scenario

Bob develops a ĪçĪL'ĪL'ĪĪ contract that calls VIJL'ĪI'ĪEL'G'L'IJN'gN'ĪL'ĪĤL'ĜN'. Bob is not aware that N'ĪL'ĪĤL'ĜN'âflâ withdraws the whole balance. As a result, Bob's contract does not work as expected.

Recommendation

Short term, review and properly document these corner cases.

Long term, review the complete documentation of the contract and simplify it to prevent misuse.

16. VIJLĪĪĪEL'ĠLĪJŃ's ĠĠĪĠĪĠHOĪAĻ and ĻĪJĻĠĪSĻĴĴĪĪŃAĻ can return different values for the same block

Severity: Low

Difficulty: Low

Type: Data Validation

Finding ID: TOB-CURVE-DAO-016

Target: VIJLĪĪĪEL'ĠLĪJŃŅĠÑ

Description

VIJLĪĪĪEL'ĠLĪJŃ's ĠĠĪĠĪĠHOĪAĻ and ĻĪJĻĠĪSĻĴĴĪĪŃAĻ return their corresponding values for a given block. Because the balance and supply can vary within the same block, these functions can return different values when called on the current block.

VIJLĪĪĪEL'ĠLĪJŃ's ĠĠĪĠĪĠHOĪAĻăĠĠĠĻđ ĠĪJĠĪă and ĻĪJĻĠĪSĻĴĴĪĪŃAĻăĠĪJĠĪă use a binary search to return their values associated with the ĠĪJĠĪ:

```
fi BĪĪĠĻŃ ĻĠĠĠĪ
çĪĪĪf ĪĪĻġġĪ ȳ fi
çĪĠŃf ĪĪĻġġĪ ȳ ĻĠĪĪġĻĠĻĴĴĪĪĪĻĴçĪĪĪĴçĪĪĪĴĪăĠĠĠĻă
ĪJĻ Ī ĪĪ ĻĠĪĪĠăġġĪăf fi wĪĪĪ ĠĤ ĠĪŃĠŃĻ ĤĪJĻĪĪ ĪJĻ ġġĪĤĠĪĻ ĪĻĪĠĠĻĻĻ
ĪĪ çĪĪĪ ņȳ çĪĠŃf
ĠĠĠĠĪ
çĪĪĪf ĪĪĻġġĪ ȳ âçĪĪĪ κ çĪĠŃ κ ġă w ġ
ĪĪ ĻĠĪĪġĻĠĻĴçĪĪĪĻĻĻĻĻĻĻĻĻĻăçĪĪĪĠăġġĪĪ 1ȳ çĠĪJĠĪf
çĪĪĪ ȳ çĪĪĪ
ĤĪĻĻf
çĪĠŃ ȳ çĪĪĪ h ġ
```

Figure 16.1: VIJLĪĪĪEL'ĠLĪJŃŅĠÑfiĻĠġġĪĠĠĠĪ.

```
çĪĪĪf ĪĪĻġġĪ ȳ fi
çĪĠŃf ĪĪĻġġĪ ȳ ĪĠŃçĪĪJĠĪ
ĪJĻ Ī ĪĪ ĻĠĪĪĠăġġĪăf fi wĪĪĪ ĠĤ ĠĪŃĠŃĻ ĤĪJĻĪĪ ĪJĻ ġġĪĤĠĪĻ ĪĻĪĠĠĻĻĻ
ĪĪ çĪĪĪ ņȳ çĪĠŃf
ĠĠĠĠĪ
çĪĪĪf ĪĪĻġġĪ ȳ âçĪĪĪ κ çĪĠŃ κ ġă w ġ
ĪĪ ĻĠĪĪġĻĠĻĻĻĻĻĻĻĻĻĻĻĻăçĪĪĪĠăġġĪĪ 1ȳ çĠĪJĠĪf
çĪĪĪ ȳ çĪĪĪ
ĤĪĻĻf
çĪĠŃ ȳ çĪĪĪ h ġ
ĻĻĻĻĻ çĪĪĪ
```

Figure 16.2: VIJLĪĪĪEL'ĠLĪJŃŅĠÑfiĻġġġġĠġġġĠ.

If a block is contained in ĴĪĪĪĻĻĻĻĻĻĻĻĻĻ, the latest one will be used.

Points on the current block can be added indefinitely in `vote`. As a result, a user calling `vote` or `propose` on the current block might not receive the latest value.

The issue does not impact Aragon's usage, as vote creation uses the previous block number for its snapshot:

```

    1. vote
    2. propose
    3. vote
    4. propose
    5. vote
    6. propose
    7. vote
    8. propose
    9. vote
    10. propose
    11. vote
    12. propose
    13. vote
    14. propose
    15. vote
    16. propose
    17. vote
    18. propose
    19. vote
    20. propose
    21. vote
    22. propose
    23. vote
    24. propose
    25. vote
    26. propose
    27. vote
    28. propose
    29. vote
    30. propose
    31. vote
    32. propose
    33. vote
    34. propose
    35. vote
    36. propose
    37. vote
    38. propose
    39. vote
    40. propose
    41. vote
    42. propose
    43. vote
    44. propose
    45. vote
    46. propose
    47. vote
    48. propose
    49. vote
    50. propose
    51. vote
    52. propose
    53. vote
    54. propose
    55. vote
    56. propose
    57. vote
    58. propose
    59. vote
    60. propose
    61. vote
    62. propose
    63. vote
    64. propose
    65. vote
    66. propose
    67. vote
    68. propose
    69. vote
    70. propose
    71. vote
    72. propose
    73. vote
    74. propose
    75. vote
    76. propose
    77. vote
    78. propose
    79. vote
    80. propose
    81. vote
    82. propose
    83. vote
    84. propose
    85. vote
    86. propose
    87. vote
    88. propose
    89. vote
    90. propose
    91. vote
    92. propose
    93. vote
    94. propose
    95. vote
    96. propose
    97. vote
    98. propose
    99. vote
    100. propose

```

Figure 16.2: [Voting.sol#L284g](#)

Exploit Scenario

Bob creates a voting contract that relies on `vote` and `propose`. Eve creates a vote using `vote` as a snapshot and corrupts the quorum percentage.

Recommendation

Short term, document that `vote` and `propose` must not be called on the current block.

Long term, properly test system properties when functions called in the same block or within a short period.

17. No incentive to vote early in GŁŁÍĤCIJİŁŁÍJİİĤŁ

Severity: Medium

Difficulty: Medium

Type: Data Validation

Finding ID: TOB-CURVE-DAO-017

Target: GŁŁÍĤCIJİŁŁÍJİİĤŁgŁÑ

Description

GŁŁÍĤCIJİŁŁÍJİİĤŁ voting offers no incentive to vote early, so late-voting users have a benefit over early voters.

Since all the votes are public, users who vote earlier are penalized because their votes are known by the other participants. An attacker can learn exactly how many tokens are necessary to change the outcome of the voting just before it ends.

Exploit Scenario

Bob votes for a vote gauge with half of its weight. His vote is winning, so he does not put in the other half of its weight. Eve votes at the last second and changes the outcome of the vote. As a result, Bob loses the vote.

Recommendation

Blockchain-based online voting is a known challenge. No perfect solution has been found so far.

Short term consider either:

- Using a decreasing weight to create an early voting advantage
- Using a blind vote

Long term, properly document and test the voting process and closely follow the community's progress regarding on-chain voting.

References

- [Aragon vote shows the perils of on-chain governance](#)

Exploit Scenario

Bob adds hundreds of gauges. As a result, most of the functions in `GĞŁÍĤCIUİŁŁÍJİİĤŁ` cannot be executed anymore.

Recommendations

Short term

- Allow users to execute the history catch-up in `VIJŁİİÍÉŁĞÍJŃĞÇĞÎĤĞİŶJİİİŁ` without depositing or withdrawing the lock.
- Create a bot that will call `LİĶŁİĤİŁŃGĞŁÍĤĞŁŁĤÇĞÎĤĞİŶJİİİŁ` and the `VIJŁİİÍÉŁĞÍJŃŃŁ` history catch-up function at least once per week.
- Consider allowing iteration over the periods in multiple transactions in `GĞŁÍĤCIUİŁŁÍJİİĤŁ` and make sure the partial updates are sound.

Long term:

- Test functions for their gas limit:
 - Use `ĞÍJŃİİĤ ŁĤŁŁ` with the `hhÍĞŁ` flag.
 - Use Echidna's [gas fuzzing feature](#).
- Update `GĞŁÍĤCIUİŁŁÍJİİĤŁ`'s logic to work with a large number of periods.

19. Testing smart contract code in Brownie can be unreliable

Severity: Undetermined

Difficulty: Medium

Type: Patching

Finding ID: TOB-CURVE-DAO-019

Target: All the smart contracts and tests

Description

The Brownie testing system should be improved to make it more robust when dealing with time-dependent and high-consumption gas tests.

When Brownie tests code that depends on the block number and timestamp in smart contracts, it provides [specific functions to simulate how they're produced](#) by the simulated blockchain.

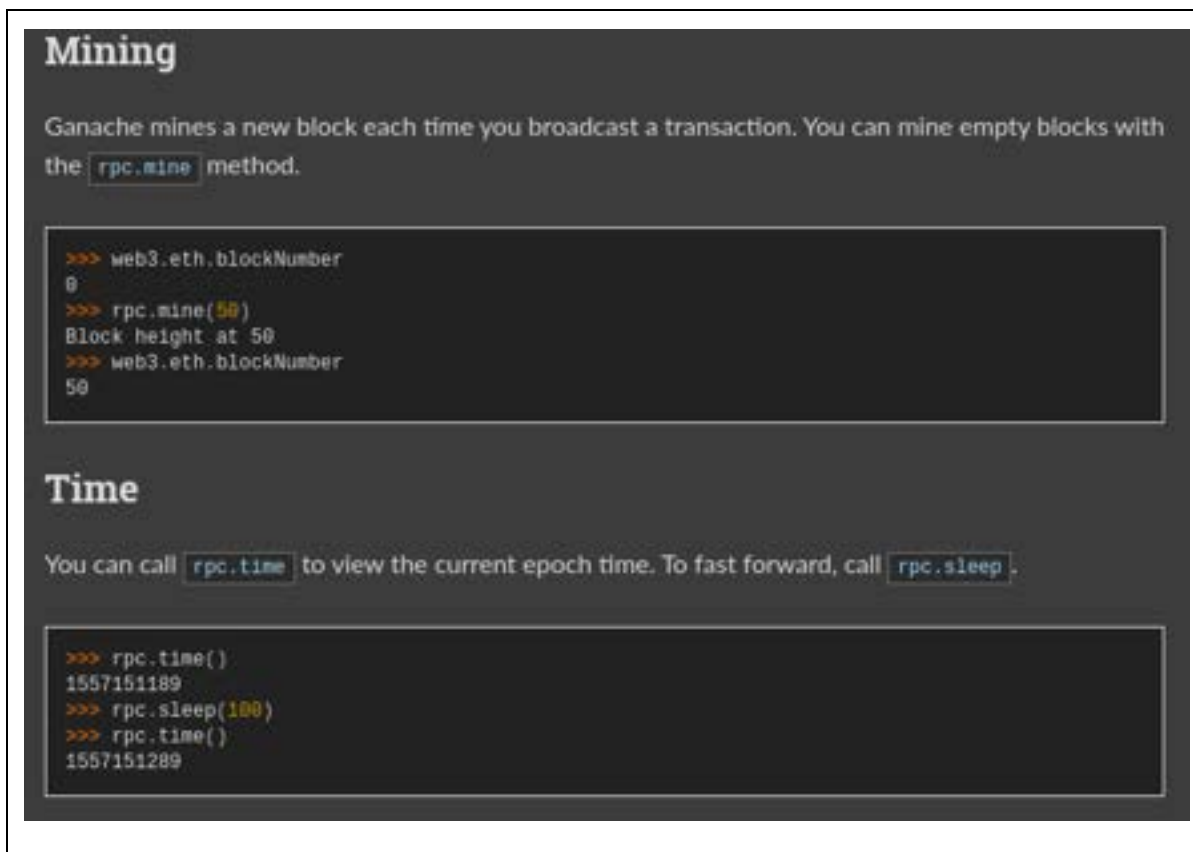


Figure 19.1: Simulating blocks in Brownie tests.

However, we found that the timestamp and block number increase even if the developer does not use the instrumentation functions. This means any test that requires checking whether the code can be executed correctly in the same block will not operate reliably.

Additionally, during testing, Brownie uses a default value for maximum gas which is determined using the [EIP198](#) function. This estimate could allow tests to pass

even if they consume a very large amount of gas, making them impractical to use when deployed.

Exploit Scenario

Curve DAO contracts are developed without proper testing and as a result, the code is deployed with a critical bug in it.

Recommendation

Short term:

- Modify Brownie to disallow automatic block timestamp and number increases.
- Set a reasonable default for the maximum gas used per transaction during tests.

Long term, use [Echidna](#) and [Manticore](#) to test your time-dependent and high-gas-consuming code.

20. Aragon's voting does not follow voting best practices

Severity: High

Difficulty: Medium

Type: Data Validation

Finding ID: TOB-CURVE-DAO-020

Target: Aragon's [Voting Contract](#)

Description

Curve Dao uses [Aragon for voting](#). Its voting logic is simple, but does not prevent several abuses that can occur with on-chain voting.

In particular, the voting contract has the following issues:

- No mitigation for quick vote and withdraw (similar to issue [TOB-CURVE-DAO-004](#)).
- No incentive to vote earlier (similar to issue [TOB-CURVE-DAO-017](#)).
- No mitigation for spam attacks. An attacker with vote creation rights can create hundreds of thousands of votes, and will need only one to pass to succeed.

Exploit Scenario

Eve is a miner. She creates new votes to set a new [minting function](#) on ERC20 on every block. The other users cannot vote on all the votes. As a result, one vote is accepted, and Eve takes control of ERC20's minting.

Recommendation

Blockchain-based online voting is a known challenge. No perfect solution has been found so far.

Short term, consider either:

- Improving Aragon's voting to mitigate the listed issues, or
- Implementing a voting contract to replace Aragon's. Perform a security assessment on the contract before deployment.

Long term, properly document and test the voting process. Closely Follow the community's progress regarding on-chain voting.

References

- [Security Disclosure: Aragon 0.6 Voting \("Voting v1"\)](#)
- [Aragon vote shows the perils of on-chain governance](#)

21. Race condition may result in users earning less interest than expected

Severity: Informational

Difficulty: Medium

Type: Data Validation

Finding ID: TOB-CURVE-DAO-021

Target: LIKLEIILNGGELIHGLN

Description

The absence of a minimal interest rate might return a lower bonus for users than expected.

LIKLEIILNGGELIH computes the interest earned by users. A bonus is applied to VIJLEILELGLIJN token’s holder:

```

    AHI cJAGLHcIILKLEIILNGGELIILaGHALF GHALHLLd If tIILghhd Lf tIILghhf
    n TU GH GGIIMH GILHL LJLGISLJJIN IL tJAGLHA
    cLJLEIILcHLGLIJNf GHALHLL 1 LHILgLJLEIILcHLGLIJN
    LjLEIILcGGIGIGHf tIILghh 1 ERCgflacLjLEIILcHLGLIJNagGGIGIGHOiaGHALa
    LjLEIILcLJLGIf tIILghh 1 ERCgflacLjLEIILcHLGLIJNagLJLGISLJJINaa

    IIf tIILghh 1 i a gfl w gffl
    II LjLEIILcLJLGIf n flf
    III kl L a LjLEIILcGGIGIGH w LjLEIILcLJLGIf a ifl w gffl

    Iii 1 IiIaId Iiia
  
```

Figure 2.1: LIKLEIILNGGELIHGLNfiLihhLi.

The bonus depends on VIJLEILELGLIJN’s total supply, which can increase over time. If a user makes a deposit in LIKLEIILNGGELIH and his transaction is mined after the total supply is increased, they can receive less bonus as expected.

Exploit Scenario

Bob calls LIKLEIILNGGELIH and expects to receive a bonus of 10%. At the same time, Alice locks a significant amount of tokens in VIJLEILELGLIJN. Alice’s transaction is accepted before Bob’s, so Bob receives a bonus of only 9%.

Recommendation

Short term, add a parameter to LIKLEIILNGGELIHgHJIL specifying the minimal amount of interest to be received, or make sure off-chain components take this scenario into account.

Long term, carefully consider the unpredictable nature of Ethereum transactions and design your contracts so they don’t depend on transactions order. Additionally, always use a lower or higher bound on asset conversions.

A. Vulnerability Classifications

Vulnerability Classes	
Class	Description
Access Controls	Related to authorization of users and assessment of rights
Auditing and Logging	Related to auditing of actions or logging of problems
Authentication	Related to the identification of users
Configuration	Related to security configurations of servers, devices, or software
Cryptography	Related to protecting the privacy or integrity of data
Data Exposure	Related to unintended exposure of sensitive information
Data Validation	Related to improper reliance on the structure or values of data
Denial of Service	Related to causing system failure
Error Reporting	Related to the reporting of error conditions in a secure fashion
Patching	Related to keeping software up to date
Session Management	Related to the identification of authenticated users
Timing	Related to race conditions, locking, or order of operations
Undefined Behavior	Related to undefined behavior triggered by the program

Severity Categories	
Severity	Description
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or Defense in Depth
Undetermined	The extent of the risk was not determined during this engagement
Low	The risk is relatively small or is not a risk the customer has indicated is important
Medium	Individual user's information is at risk, exploitation would be bad for

	client's reputation, moderate financial impact, possible legal implications for client
High	Large numbers of users, very bad for client's reputation, or serious legal or financial implications

Difficulty Levels	
Difficulty	Description
Undetermined	The difficulty of exploit was not determined during this engagement
Low	Commonly exploited, public tools exist or can be scripted that exploit this flaw
Medium	Attackers must write an exploit, or need an in-depth knowledge of a complex system
High	The attacker must have privileged insider access to the system, may need to know extremely complex technical details, or must discover other weaknesses in order to exploit this issue

B. Code Maturity Classifications

Code Maturity Classes	
Category Name	Description
Access Controls	Related to the authentication and authorization of components.
Arithmetic	Related to the proper use of mathematical operations and semantics.
Assembly Use	Related to the use of inline assembly.
Centralization	Related to the existence of a single point of failure.
Upgradeability	Related to contract upgradeability.
Function Composition	Related to separation of the logic into functions with clear purpose.
Front-Running	Related to resilience against front-running.
Key Management	Related to the existence of proper procedures for key generation, distribution, and access.
Monitoring	Related to use of events and monitoring procedures.
Specification	Related to the expected codebase documentation.
Testing & Verification	Related to the use of testing techniques (unit tests, fuzzing, symbolic execution, etc.).

Rating Criteria	
Rating	Description
Strong	The component was reviewed and no concerns were found.
Satisfactory	The component had only minor issues.
Moderate	The component had some issues.
Weak	The component led to multiple issues; more issues might be present.
Missing	The component was missing.

Not Applicable	The component is not applicable.
Not Considered	The component was not reviewed.
Further Investigation Required	The component requires further investigation.

C. Code Quality

The following recommendations are not associated with specific vulnerabilities. However, they enhance code readability and may prevent the introduction of vulnerabilities in the future.

General suggestions:

- **Do not use one-letter variable names.** The smart contract code uses variables with very short names that can be difficult to parse when the code is modified or reviewed. Use full names, e.g., `NHIIIL` instead of `N`.
- **Split large functions into internal functions.** Large functions such as `LIXLIIILNGGLIHGCGIHGIJUIIL` and `VIJLIIIELGLIJNGCGIHGIJUIIL` can be split into internal functions (e.g., history catch-up, user value update, etc.). Having smaller and simpler functions will simplify review and verification of the code.
- **Do not use unnamed constants.** The smart contract code uses certain constants without naming them. Use proper names, e.g., `BASE` instead of `gfI áá gí`.

ERC9fCRVgLÑ:

- **Consider correcting the RATEÇREDUCTIONÇCOEFFICIENT constant to be more accurate.** The exact coefficient used is `ggggggghhggiñfiñhñid` and the comment accompanying its declaration indicates it should be equal to `LXLâgã á gHgí`. However, a more accurate approximation of `LXLâgã á gHgí` would actually be `ggggggghhggiñfiñhñí`, which differs in the last three decimal places.

VotingEscrow.vy:

- **Split the deposit functions into deposit creation, amount increase, and time increase functions.** `DHJULIL` handles the creation and increase of a deposit's amount and time simultaneously. As a result, the function has to handle too many cases and is error-prone.
- **Use `IILHÇGIJGIÇHJIGI` in `GGIÇIÇHOIAL`.** `BGIÇIÇHOIAL` duplicates the code of `IILHÇGIJGIÇHJIGI`.

D. Token Integration Checklist

The following checklist provides recommendations when interacting with arbitrary tokens. Every unchecked item should be justified and its associated risks understood.

For convenience, all [Slither](#) utilities can be run directly on a token address, such as:

```
slither --code-verify --code-verify-args '0x0000000000000000000000000000000000000000' --code-verify-args '0x0000000000000000000000000000000000000000'
```

General Security Considerations

- The contract has a security review.** Avoid interacting with contracts that lack a security review. Check the length of the assessment (aka “level of effort”), the reputation of the security firm, and the number and severity of the findings.
- You have contacted the developers.** You may need to alert their team to an incident. Look for appropriate contacts on [blockchain-security-contacts](#).
- They have a security mailing list for critical announcements.** Their team should advise users (like you!) when critical issues are found or when upgrades occur.

ERC Conformity

Slither includes a utility, [slither-check-erc](#), that reviews the conformance of a token to many related ERC standards. Use `slither-check-erc` to review that:

- `transferFrom` and `transferFromAndApproval` return a boolean.** Several tokens do not return a boolean on these functions. As a result, their calls in the contract might fail.
- `approve`, `approveAndCall`, and `approveAndCallBatch` functions are present if used.** These functions are optional in the ERC20 standard and might not be present.
- `approve` returns a `uint256`.** Several tokens incorrectly return a `bool`. If this is the case, ensure the value returned is below 255.
- The token mitigates the [known ERC20 race condition](#).** The ERC20 standard has a known ERC20 race condition that must be mitigated to prevent attackers from stealing tokens.
- The token is not an ERC777 token and has no external function call in `transferFrom` and `transferFromAndApproval`.** External calls in the transfer functions can lead to reentrancies.

Slither includes a utility, [slither-prop](#), that generates unit tests and security properties that can discover many common ERC flaws. Use `slither-prop` to review that:

- The contract passes all unit tests and security properties from `slither-prop`.** Run the generated unit tests, then check the properties with [Echidna](#) and [Manticore](#).

Finally, there are certain characteristics that are difficult to identify automatically. Review for these conditions by hand:

- TLĠĠL'İHL and LĠĠL'İHLFLUJİ should not take a fee.** Deflationary tokens can lead to unexpected behavior.
- Potential interest earned from the token is taken into account.** Some tokens distribute interest to token holders. This interest might be trapped in the contract if not taken into account.

Contract Composition

- The contract avoids unneeded complexity.** The token should be a simple contract; a token with complex code requires a higher standard of review. Use Slither's [İĠİĠİhL'İİİĠN](#) printer to identify complex code.
- The contract uses SĠİHMĠİ.** Contracts that do not use SĠİHMĠİ require a higher standard of review. Inspect the contract by hand for SĠİHMĠİ usage.
- The contract has only a few non-token-related functions.** Non-token-related functions increase the likelihood of an issue in the contract. Use Slither's [GUJLĠĠĠLhL'İİİĠN](#) printer to broadly review the code used in the contract.

Owner privileges

- The token is not upgradeable.** Upgradeable contracts might change their rules over time. Use Slither's [İĠİĠİhL'İİİĠN](#) printer to determine if the contract is upgradeable.
- The owner has limited minting capabilities.** Malicious or compromised owners can abuse minting capabilities. Use Slither's [İĠİĠİhL'İİİĠN](#) printer to review minting capabilities, and consider manually reviewing the code.
- The token is not pausable.** Malicious or compromised owners can trap contracts relying on pausable tokens. Identify pauseable code by hand.
- The owner cannot blacklist the contract.** Malicious or compromised owners can trap contracts relying on tokens with a blacklist. Identify blacklisting features by hand.
- The team behind the token is known and can be held responsible for abuse.** Contracts with anonymous development teams, or that reside in legal shelters should require a higher standard of review.

Token Scarcity

Reviews for issues of token scarcity requires manual review. Check for these conditions:

- No user owns most of the supply.** If a few users own most of the tokens, they can influence operations based on the token's repartition.

- ❑ **The total supply is sufficient.** Tokens with a low total supply can be easily manipulated.
- ❑ **The tokens are located in more than a few exchanges.** If all the tokens are in one exchange, a compromise of the exchange can compromise the contract relying on the token.
- ❑ **Users understand the associated risks of large funds or flash loans.** Contracts relying on the token balance must carefully take in consideration attackers with large funds or attacks through flash loans.

E. Fix Log

Swiss-Stake addressed issues TOB-CURVE-DAO-001 to TOB-CURVE-DAO-013 in their codebase as a result of the assessment. Each of the fixes was verified by Trail of Bits. The reviewed code is available in git revision [iGǾffliǾhflǾǾHǾhHhǾflǾGǾhǾǾiǾǾǾflǾiǾflǾHhflǾǾ](#).

ID	Title	Severity	Status
01	LĪKŁĪĤĪĻŃGǾĪH does not account for VIJĻĪĪĪELǾLIJŃ's balance updates	Medium	Mitigated
02	LĪKŁĪĤĪĻŃGǾĪH does not account for VIJĻĪĪĪELǾLIJŃ's ŁIJĻǾISŁĴĴĪŃ updates	Medium	Not fixed
03	Early users have a unfair advantage	Medium	Not fixed
04	GǾĪHĈIJĻŁIJĪHŁ allows for quick vote and withdraw voting strategy	Medium	Mitigated
05	Adding the same gauge multiple times will lead to incorrect sum of weights	Medium	Fixed
06	Spam attack would prevent LĪKŁĪĤĪĻŃGǾĪH's integral from being updated	Medium	Risk accepted
07	MĪĪŁHŁ user can confiscate any user tokens	High	Fixed
08	MĪĪŁ and BŁĻĪ events cannot be trusted	Low	Fixed
09	VIJĻĪĪĪELǾLIJŃ's Admin can take whitelisted accounts hostage	Medium	Fixed
10	ERCǾfCRV is not initiated correctly with large name and symbol	Low	Fixed
11	Lack of two-step procedure for critical operations is error-prone	High	Fixed
12	Lack of value verification on decimals is error-prone	Low	Fixed
13	Lack of events is error-prone	Informational	Mitigated
14	Race condition in removing addresses from whitelist and withdrawing	Informational	WIP
15	Lack of documentation is error-prone	Informational	WIP
16	VIJĻĪĪĪELǾLIJŃ's GǾĪǾIǾHŌIĀŁ and ŁIJĻǾISŁĴĴĪŃĀŁ can	Low	WIP

	return different values for the same block		
17	No incentive to vote early in GŁŁÍHCJIŁŁÍJİİİHL	Medium	WIP
18	Several loops will not be executable due to gas limitation	High	WIP
19	Testing smart contract code in Brownie can be unreliable	Undetermined	WIP
20	Aragon's voting does not follow voting best practices	High	WIP
21	Race condition can lead users to earn less interest than expected	Informational	WIP

Detailed Fix Log

This section includes brief descriptions of fixes implemented by Swiss-Stake after the end of this assessment that were reviewed by Trail of Bits.

Finding 1: LIKLIHILNGGLIH does not account for VIJLIILIELGLIJN's balance updates

This issue is mitigated by:

- Reducing the bonus created by the vote locks from 5 to 2.5.
- Adding a public IIGI function to adjust the working balance of any user abusing the bonus.

We recommend updating the documentation to ensure users are aware of IIGI. Curve DAO should consider developing a bot that will scan the account and call IIGI when appropriate. This bot should be publically available to prevent [TOB-CURVE-DAO-001](#) being exploited.

Finding 2: LIKLIHILNGGLIH does not account for VIJLIILIELGLIJN's LIJGISLJJIIN updates

This issue is not fixed.

Finding 3: Early users have a unfair advantage

To fix the issue, Curve DAO added a check preventing the bonus from being applies during the first two weeks:

```
âGIIUGIGLIIHLGIIJ n LHIIGJHLIIJHÇGÎHGIIJIIIL'äflā κ BOOSTÇWARMUPä
```

LIKLIHILNGGLIHGLNfLgflgg

sHIIGJHLIIJHÇGÎHGIIJIIIL'äflā will be zero if the liquidity gauge is deployed when the period on the gauge controller is greater than or equal to 1. As a result, the check is incorrectly implemented.

Additionally, the delay in the bonus activation will only work if early users share their tokens enough to create a well-distributed repartition.

Finding 4: GGLIHCUJLJIJHHL allows for quick vote and withdraw voting strategy

This appears to be mitigated by disallowing changing weight votes more often than once in 10 days.

Finding 5: Adding the same gauge multiple times leads to incorrect sum of weights

This appears to be fixed by disallowing adding the same gauge twice.

Finding 6: Spam attack would prevent LIKLIHILNNGGLIH's integral from being updated

The client estimated the impact of this issue and accepted the risk.

Finding 7: MIILHL user can confiscate any user tokens

This appears to be fixed by:

- Disallowing the transfer of unapproved tokens by the IILHL.
- Disallowing setting the IILHL address more than once.

Finding 8: MIIL and BLIL events cannot be trusted

This appears to be fixed by:

- Disallowing transfer of unapproved tokens by the IILHL.
- Disallowing users to transfer to flnl.

Finding 9: VIILIELGLIJN's Admin can take whitelisted accounts hostage

This appears to be fixed by allowing un-whitelisted addresses to withdraw from the voting escrow contract.

Finding 10: ERCglCRV is not initiated correctly with large name and symbol

This appears to be fixed by requiring the use of Vyper 0.2.0 to resolve this issue.

Finding 11: Lack of two-step procedure for critical operations is error-prone

This appears to be fixed by implementing a two-step procedure in the following functions:

- VIILIELGLIJNgLGLILHLIJNHLIL
- PIJJIPLIJNngLHLGLILHL
- GGLIHCIJILIJILHLGLILHLIJNHLIL

Finding 12: Lack of value verification on decimals is error-prone

This appears to be fixed by validating the values obtained from calling the HGLIIL function.

Finding 13: Lack of events is error-prone

This appears to be mitigated by adding suitable events in the following functions:

- PIJJIPLIJNngLHLGLILHL
- PIJJIPLIJNngLHLGLILHL
- ERCglCRVgkJHGLHLIILHLILGLGLHLHL
- ERCglCRVgLHLGLILHL
- ERCglCRVgLHLGLILHL

