Vulnerability disclosure: the discovery and the rescue

MICHAEL EGOROV
25 JAN 2020 • 3 MIN READ

At 3 a.m. I woke up to 20 messages from Robert Leshner and Sam Sun. The news was that the Curve contract had a critical (but not exploited) vulnerability which allowed anyone to drain the smart contract.

The bug wasn't anything a linter could catch: it was hidden deep into Curve's algorithm. According to yet-to-be-shipped whitepaper, the solution of the *master equation*, which describes the link between the balances and the invariant, is given by the following method:

$$x_j^2 + x_j \left(\sum_{i \neq j} x_i - \frac{(An^n - 1)\, D}{An^n} \right) = \frac{D^{n+1}}{n^{2n} A \prod_{i \neq j} x_i}. \tag{13}$$

This equation can be solved iteratively as:

$$x_{j,k+1} = \frac{x_{j,k}^2 + c}{2x_{j,k} + b}, \tag{14}$$

where k is the number of iteration, $b = \left(\sum_{i \neq j} x_i - \frac{(An^n - 1)D}{An^n} \right)$, $c = \frac{D^{n+1}}{n^{2n} A \prod_{i \neq j} x_i}$. Itera-

This method has, well, a product of all balances for *i* not equal to *j*. While *x_i* is the "incoming" balance and *x_j* is the "outgoing" balance,

th component). Which is fine when we exchange i-th asset into j-th asset. But what if $i == j$? And here comes a problem.

```
def get_y(i: int128, j: int128, x: uint256, _xp: uint256[N_COINS]) -> uint256:
    # x in the input is converted to the same price/precision
    D: uint256 = self.get_D(_xp)
    c: uint256 = D
    S_: uint256 = 0
    Ann: uint256 = convert(self.A, uint256) * N_COINS

    _x: uint256 = 0
    for _i in range(N_COINS):
        if _i == i:
            _x = x
        elif _i != j:
            _x = _xp[_i]
        else:
            continue
        S_ += _x
        c = c * D / (_x * N_COINS)
    c = c * D / (Ann * N_COINS)
    b: uint256 = S_ + D / Ann  # - D
```

The situation when $i == j$ is not accounted for, and submitting an exchange of some asset into the same asset, essentially, drains this asset. *Anyone* could do it.

So, what sort of action should be done in such a case?

One way could be to announce that there is a vulnerability and let people quickly withdraw funds. However, that could bring too much attention of black hat hackers who could find this vulnerability after knowing it exists.

Another way could be to attack the contract as a white hat, drain it and bring funds back to liquidity providers. The problem is though that there are front-running bots who look for transactions which turn with profit (not only for hacks, but for arbitrage, too), and execute a competing transaction automatically. And really, there was no good way around them.

Curve Finance – Vulnerability disclosure: the discovery and the rescue

custody over funds) except for asking all LPs to withdraw and/or migrate funds over.

Reaching all LPs privately was impossible, too, because of the overwhelming success of Curve growing up to 100% a day.

In this situation, it was decided to deploy a new version (which was brewing anyway) with newer and better parameters and other good changes, such as more advanced logging, and the fix in question, however without disclosing the fix in question publicly on github. Or, at least, before LPs migrate the funds. As most trades were going from 1inch.exchange, they were able to switch to the new contract within 10 minutes, leaving the old, vulnerable, contract without profits beyond Compound interest.

Immediately after deploying the new contract and UI, more than 50% of funds were migrated over even before the official announcement. The rest of the migration took 3 days.

In conclusion, the contract was fixed. There was a kill switch which stops everything except withdrawals (however works only for the first two months after the deployment) which was added. The audit of the (fixed) contract is going to happen at the end of January. ETH Zurich have provided early access to their fantastic tool to do formal verification for Vyper contracts which would've prevented this situation completely.

Although the situation was well handled, be careful with unaudited contracts and only deposit what you can afford to lose before audits happen.

Curve wants to thank Sam Sun who discovered the vulnerability and helped handle the situation - a bug bounty was awarded, Robert Leshner (Compound), Lev Livnev, Julien Bouteloup (Stake Capital), and Richard Burton.

recent. We are in the unknown territory which hasn't ever been explored. The journey to rebuild the World Financial System can be dangerous at times. And exciting.