Bug bounty submission

Reporter: Devan Purhar | Staked; devan@staked.us
Date: 10 June 2020

Response: Alex Scott | mStable; alex@mstable.org
Date: 11th June 2020

Reported Issue

Context

The SAVE feature reflects value accrual through an exchange rate, which is updated by default every time a deposit to the SavingsContract.sol occurs if more than 30 minutes has passed since it was last updated. By natural flow, upon withdrawals, the exchange rate is never updated, however the function responsible for updating the exchange rate is unprotected, and can be called at any time and will update the exchange rate as long as the contract isn't "paused" and it's been greater than 30 minutes since the last update. Any party can see and get the "last collection" timestamp.

Vulnerability

Any party can exploit the stale exchange rate mechanism by depositing at an artificially lower value, manually updating it, then withdrawing at the updated value. I focus on the scenario where the attacking parties already hold an mAsset. They do as they wish with their mAsset for 97% (29/30 minutes) of the time, and the other 3% carry out this attack. However, it's also possible they're not in the mStable system at all for 97% of the time, but in other external assets such as Dai, USDC, etc. The only change that makes is increasing the cost of the attack, as we need to include the cost for mint and burn transactions of the mAsset.

Full report available here

mStable Response

Reporter is noted to have understood both the mStable system and wider ecosystem at an expert level, and has demonstrated significant abilities at system analysis and exposing attack vectors. mStable notes that this discovery is tied to one of our 'areas of interest': 'Unfair payouts through SAVE, MINT, REDEEM or SWAP functionalities that results in an under-collateralized or affected system'.

We detail below additional context around the area of the reported issue, an analysis and a resolution method.

Context

Reason why the 30 minute window exists

Limiting the APY gained from the basket to avoid accidental hyper inflation - for example in the case of an erroneous 'balance' produced through the 'PlatformIntegration' contracts. This would be devastating to the mAsset itself, and not just those invested in the SAVE contract.

Because the mAsset yield is so volatile (both from SWAP fees and from the Lending platform integrations), there is a large chance that this rate fluctuates highly and averages out over the longer term. Ideally we would impose a sort of 150% max APY, however we kept the rate as 1500%, and extrapolated the APY based on the previous timestamp (>=30 minutes ago) and the current time, with the value gained.

This window was not implemented to reduce gas costs for the saver as pointed out in the reporter's document, this was a side-effect.

Why does the SAVE contract exist?

This mechanism allows mAsset holders (essentially liquidity providers to the system) to get a return for their contribution through the yield and SWAP fees generated from their collateral. Technically speaking anybody holding an mAsset is a liquidity provider, although only those holding in the SAVE contract get the native interest generated through the mAsset.

Notes on attack

Definition of AUM, TotalSupply and APY

In the reported attack, the term AUM is used to describe the amount held in the mAsset SAVE contract.
TotalSupply is the total amount of mAsset in circulation.
APY is used to describe the yield benefiting the actors in the 'SAVE" contract (i.e. to those AUM).
Note that the APY will only remain high if there is sufficient yield being generated on the underlying, meaning that if the attacker were to exit out of the mAsset in between attacks, the overall collateral used to generate yield would be reduced.

mAsset holders are liquidity providers

As mentioned above ("Why does the SAVE contract exist"), all mAsset holders are essentially liquidity providers and can choose to receive the native interest through SAVE. This is important, as it affects the 'fairness' of attack scenarios, depending on the utility derived from the mAsset when not being used for attack.

What is the mAsset doing when it is not in the savings contract?

This attack assumes that the mAsset is being profited from whilst not in the SAVE contract (i.e. for the 'other' 29 minutes of the window). This may be true - however it's utilisation would come at a cost to the user. For this attack to be a net-negative to the mAsset, the attacker would need to be exiting **out** of the mAsset between attacks, **or** claiming other rewards available in the ecosystem (for example, contributing to a Uniswap pool).

Scenario 1: Attacker holds mUSD outside of savings contract
Result: This is acceptable to the system as the attacker is still providing liquidity to the system and contributing proportionately to the yield generated through the underlying.
Conclusion: Not necessarily negative - but should investigated for profitability

Scenario 2: Attacker is depositing into an ecosystem pool (e.g. mUSD/ETH on Uniswap, mUSD on Aave etc)
Result: Attacker would incur significant gas costs (comparable to executing base attack) which decreases profit from attack. Attackers mAsset units would still generate interest to the benefit of other savers and to contribute to liquidity.

Note: *No pools of this nature currently exist for the mAssets however will do in the immediate future*
Conclusion: Unfair to savers and ecosystem participants - needs investigated for profitability (adding roughly 700k gas to base costs) i.e. at what APY in an external market would it be profitable for them

Scenario 3: Attacker exits out of the mAsset and holds a proportionate amount of bAssets and then buys back into the mAsset before executing the attack.
Result: This attack would only be possible if the attacker were to 'redeemMasset', which comes at a large gas cost (roughly 400k per bAsset). They would then need to mint with everything (roughly 200k per bAsset). No interest would be gained from their collateral during that time.
Conclusion: Negative for mAsset - needs investigated for profitability, adding roughly 2.5m gas to base costs and reducing size of yield for other savers.

Scenario 4: Attacker exits out of the mAsset into a specific bAsset, incurring a SWAP fee (0.3% at time of writing)
Result: Attacker contributes to SWAP fees in a large way, at the benefit of other savers.
Conclusion: Highly unlikely this will be profitable for the attacker. 0.3% fee per 30 minutes equates to 5000% per year

Is this attackable through flash loan?

No - as it is requiring 2 blocks & 2 tx at minimum, where flash loans must be repaid in the same transaction.

Profitability analysis

Running through the scenarios from above, to determine at what point this attack becomes profitable.

Fixed costs

- Base attack gas consumption: ~580k (~60k for deposit, ~450k for exchange rate update, ~70k for redemption)
- Average gas (fast) over past 12 months: 13 gwei (Although currently 46 gwei)
- Assumed ETH price: $242
- Attacker capital (assume 100% of totalSupply)

Variables

- AUM (Assets Under Management)
- Total Supply
- APY

Scenario 1

profit = (interest stolen - gas fees)

Base attack costs ~$1.83 (at 13 gwei), assuming the attacker is using an amount equal to AUM in order to attack.

AUM (w/o attacker)	APY	Yield collected	“Profit” per 30min
$1m	25%	~$13.7/2	~$4.3
$10m	25%	~$137/2	~$61
$25m	20%	~$274/2	~$134

Scenario 2

Adds ~700k to base gas cost. This number is industry average for depositing and then withdrawing from a lending market. Profit is then gained from the lending market or reward program.

Assuming the attacker is rewarded on a **perfectly pro-rata basis** for those assets he has in the third party market, with a 29/30 utilisation rate (~96.7%).
Assuming **modest** 700k gas cost for going in/out every time (~$2.20 at $242 ETH), with a cost per year of ~$38000 at 13 gwei.
Total gas cost per 30 min window: ~$4.03.
This calculation represents the absolute best case scenario for an attacker, in terms of capital efficiency.

Attacker funds	Required APY
$1m	~7.1%
$10m	~0.71%
$50m	~0.14%

Conclusion: Even though the above is an absolute best case scenario in terms of capital efficiency, it is potentially profitable for the attacker to pivot into third party markets and capitalise from them in between dipping in and out of the SAVE contract. Even though this mAsset holder is still acting as a liquidity provider, this circumvents the intention of the SAVE contract.

Scenario 3

Adds ~2.5m to base gas cost, taking total to ~3,080k gas. At a (historically) fast gas price of 13, this would cost ~$10 per 30 min window. Current conditions see 42 as a fast price. Assume that the attacker has capital equivalent to the current AUM (for example, if AUM is $1m, assume attacker is using $1m).

AUM (w/o attacker)	APY	Yield collected	Gas price: 13 Profit per 30m	Gas price: 42 Profit per 30m
$1m	25%	~$13.7/2	Neg $3.7	Neg $25.7
$5m	20%	~$57/2	~$18	Neg $3.5
$10m	25%	~$137/2	~$53	~$31.5
$25m	20%	~$274/2	~$127	~$105

Conclusion: Vector becomes more profitable at higher total mAsset supply and lower gas costs.

Risk analysis

This is only deemed a net negative to the system if it is profitable for the attacker to profit from dipping in and out of the SAVE contract if they are fully exiting from the system (scenario 3), or are circumventing the SAVE mechanism and concurrently profiting from other third party platforms, at the expense of other ecosystem participants (scenario 2).

Topic	Analysis
Motive	Strong motives for this to take place given that it is a no loss profit opportunity at high AUM.
Opportunity	Exposure increases with increase in mUSD total supply (as this is what is being used to generate the yield). **Does not affect mStable under current**

	market conditions, but taking projected growth into consideration, is **highly likely to become profitable**/exploitable within the next 6 months.
Ease of discovery	As noted in the initial report, contracts are open source and well documented.
Ease of exploit	Relies on sufficient technical knowledge and large amounts of available capital. Execution itself is trivial at that stage, needing basic infrastructure and custom smart contracts.
Scope of affected users	Other participants in the SAVE contract.
Financial damage	Loss of trust in the SAVE mechanism would likely see negative economic actions occur throughout the mStable system (exiting o the system).
Reputation damage	Dependant on length of attack.

Resolution

Removes the blocker on collecting interest more than once in 30 minute period.

Retains existing 'extrapolatedAPY' calculations when it has been longer than 30 minutes since the last interest collection.

If it has been less than 30 minutes, it simply checks that the supply has not inflated by more than 0.1% (or 1e15) during that PERIOD (30 mins). At a 0.1% 'SWAP' fee, this would mean that the total supply of the mAsset would need to be swapped between the two collections, if this were to be hit.

Full solution is implemented here:
https://github.com/mstable/mStable-contracts/pull/98

Downsides to new implementation:

- Higher gas costs for all users (as pointed out by reporter, all users would collected yield from mAsset and incur ~310k gas, as opposed to some of them savings gas as currently is)

- Marginally higher gas costs for all savers, due to the addition of new properties in the above (per tx increase of ~1500, one saver per 24h will incur +20000)

Procedure:

- Write and test the new code
- Deploy the updated `SavingsManager` to both Mainnet and Ropsten
- Perform a module upgrade request through mStable governance via the Nexus on both Mainnet and Ropsten
- Wait 1 week
- Accept the resulting upgrade on Ropsten
- Perform beta testing on Ropsten
- Accept the update on Mainnet

Conclusion

The reported vulnerability is complex in that the execution of it does not always produce a net negative effect. The important factor is”'what is the mAsset doing when it's not in the SAVE contract”. With that in mind, scenarios 2 and 3 outlined above highlight circumstances (based on market / asset conditions) that present opportunities for an attacker that are both profitable and either have a negative affect on other users (scenario 3) or circumvent the intended behaviour of our SAVE mechanism (scenario 2).

This attack does not become profitable until AUM is high (projected in the coming months), assuming that the APY remains consistently high (if lower, AUM must compensate). Additional factors that may delay the profitability of this attack include ETH price, network congestion and Basket Composition (thus Redeem gas costs).

Severity: Low-Moderate

Likelihood: Likely-Almost certain

Bounty: **$1,000**