SHERLOCK

SHERLOCK SECURITY REVIEW FOR

Prepared for: Unitas
Prepared by: Sherlock
Lead Security Expert: stopthecap
Dates Audited: June 5 - June 12, 2023
Prepared on: July 6, 2023

Introduction

Unitized stablecoins serving as units of account representing emerging market currencies. A new currency revolution.

Scope

Repository: xrex-inc/Unitas-Protocol

Branch: main

Commit: 9ef6847c5437bfe5e178355f36f9ebb19c1d0468

For the detailed scope, see the contest details.

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Issues found

Medium	High
6	1

Issues not fixed or acknowledged

Medium	High
0	0

Security experts who found valid issues

thekmj
moneyversed
PokemonAuditSimulator
0xyPhilic
ast3ros
0xGoodess
Ruhum
sashik_eth
0x4non
toshii
tsvetanovv
kutugu

shogoki
PRAISE
stopthecap
Norah
Juntao
PawelK
Jiamin
n33k
DevABDee
0xJuda
Dug
0x00ffDa
okolicodes
vagrant
carrotsmuggler
qpzm
ctf_sec
mau
circlelooper
radev_sw

Issue H-1: Unitas swap function is vulnerable to Sandwich Attack at oracle price update

Source: https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/67

Found by

DevABDee, Dug, n33k, qpzm, sashik_eth, shogoki, vagrant

Summary

WHen the oracle price is updates, an attacker can sandwich the update adderss with 2 swap transactions to gain a profit and drain the collateral.

Vulnerability Detail

Inside the `swap` function of the `Unitas` contract the `getLatestPrice` function of the `XOracle` contract is used to fetch the current price of the USDx token to be swapped to/from. The price has to be updated periodically because the currencies fluctuate against the USD price. A user could use these fluctuations to speculate on prices and gain a profit or loss.

However,as the price for the oracle is updated by a transaction that is publicly visible inside the mempool, a malicious user or attacker can see the new price before it is active. As the oracle price is the only thing, which has influence of the token number to mint/burn on a swap call, an attacker can easily exploit a temporarily appreciation of an USDx token agains the USD1 (US Dollar price). This can be achieved by "sandwiching" the price update transaction with a transaction to swap USD1 into the relevant USDx token first, and swap it back after the price update.

Example: THe price of USD91 (INR) is to be increased from 0.012 to 0.013 Attacker already holds 10,000 of USD1 tokens.

- The Feeder sends the transaction to update the oracle price, and it gets placed in the mempool.
- Attacker sees these transaction, and sends himself 2 transactions.

1. Swap 10,000 possible mount of USD1 to USD91
2. Swap all USD91 to USD1

- The attacker sets the gas to ensure that the first tx gets included before the price update, and the second one after the price update.
- The executed Transactions in order will be:

1. Attacker swaps 10,000 USD1 to USD91 (should receive: 10,000 / 0.012 = 833,333.333)
2. FEEDER updates oracle price of USD91 to 0.013
3. Attacker swaps all USD91 tokens to USD1 (should receive: 833,333.333 * 0.013 = 10,833.333)

The attacker just made 833 USD1 profit in these 2 transactions, and can redeem the USD1 tokens for USDT, which will deduct the collateral by this amount.

Impact

Attacker can gain profit and "steal" collateral

Code Snippet

https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/main/Unitas-Protocol/src/Unitas.sol#L439

https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/main/Unitas-Protocol/src/XOracle.sol#L26-L32

Tool used

Manual Review

Recommendation

The 2 way minting/burning mechanism by the orcale price might be dangerous. However to prevent the specific attack vector, maybe the minting can be paused and unpaused before and after the price update. (in separate transactions)

Discussion

SunXrex

Sun & Lucas: To update prices, we can consider using an RPC (Remote Procedure Call) that helps prevent MEV (Miner Extractable Value).

Risk should be **medium**.

Usually, updates to the exchange rate for stablecoins result in less than a 1~1.6% change. Additionally, we apply a fee for minting and burning. This makes arbitrage difficult. To provide a perspective from the auditors' case, an increase from 0.012 to 0.013 represents an approximate growth of 8.33% which is almost impossible.

Aditya: Price update frequency is much lower for Unitas compared to other defi swaps like Uniswaps. Unitas update frequency could be as low as once per day.

Also the expected users/minters of the protocol are low frequency but high ticket size. Chances of users mint request being in the mempool and Unitas price update happening at the same time given are fairly low. Also, unlike Uniswap any new trade does not move the price. The price is only updated at certain periods of the day or once a day.

ctf-sec

Can be a valid medium

0xffff11

I agree with a medium. Due to the current architecture of having a fee and the stability of the assets, the value it is quite limited.

hrishibhat

Considering this a medium issue based on the above comments.

Shogoki

Escalate for 10USDC This should be considered as High:

I was thinking for a longer time on the comments and the reasons for this one to be downgraded to Medium. I even created another escalation on #88 referencing this reasoning. However, after i saw this comment

> Hi @Shogoki glad to know your thoughts.
>
> This issue is a valid medium, because price is based on currency exchange rates and it is not fair to say the price is stable or not (depends on the currency itself).
>
> When EMC appriciates / depreciates, the sponsor **HAVE TO** update price or the protocol will not work properly.

i tend to agree with that. Given that it should bot be a lowering factor, that the prices **should** be stable. Therefore this should be a high, as reported!

———- If this gets accepted, i believe #88 is a valid medium.

However, If judges decide to reject this escalation here, i think #88 should be rated as low, as stated in my other escalation.

sherlock-admin

> Escalate for 10USDC This should be considered as High:
>
> I was thinking for a longer time on the comments and the reasons for this one to be downgraded to Medium. I even created another escalation on #88 referencing this reasoning. However, after i saw this comment
>
> > Hi @Shogoki glad to know your thoughts.

> > This issue is a valid medium, because price is based on currency exchange rates and it is not fair to say the price is stable or not (depends on the currency itself).
> >
> > When EMC appriciates / depreciates, the sponsor **HAVE TO** update price or the protocol will not work properly.
>
> i tend to agree with that. Given that it should bot be a lowering factor, that the prices **should** be stable. Therefore this should be a high, as reported!
>
> ———- If this gets accepted, i believe #88 is a valid medium.
>
> However, If judges decide to reject this escalation here, i think #88 should be rated as low, as stated in my other escalation.

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

DevABDee

Escalate for 10 USDC

I agree with @Shogoki that this should be considered as a High.

I believe It's a High issue even if prices are not gonna updated frequently because It's not only that the attacker can make a profit from that. Tokens holders can prevent loss as well. And not just prevent the loss, they can turn it into profits by exploiting this loophole (I have provided a more detailed explanation of this in my submission #105) As the protocol team said "updates to the exchange rate for stablecoins result in less than a 1~1.6% change", I agree but we must also take into account unusual scenarios, particularly when dealing with EMC Markets. Who knows the market? Maybe the prices get updated frequently!

I completely agree with the points @Shogoki has raised, If this issue is not deemed High, then it would be reasonable to classify #88 as a low priority.

Thanks!

sherlock-admin

> Escalate for 10 USDC
>
> I agree with @Shogoki that this should be considered as a High.
>
> I believe It's a High issue even if prices are not gonna updated frequently because It's not only that the attacker can make a profit from that. Tokens holders can prevent loss as well. And not just prevent the loss, they can turn it into profits by exploiting this loophole (I have provided a

> more detailed explanation of this in my submission #105) As the protocol team said "updates to the exchange rate for stablecoins result in less than a 1~1.6% change", I agree but we must also take into account unusual scenarios, particularly when dealing with EMC Markets. Who knows the market? Maybe the prices get updated frequently!
>
> I completely agree with the points @Shogoki has raised, If this issue is not deemed High, then it would be reasonable to classify #88 as a low priority.
>
> Thanks!

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

ctf-sec

Agree with high

jacksanford1

The case for Medium is that the frequency of being able to steal a material amount of funds seems very rare:

- will be used for stablecoins (which can have large price swings, but the frequency of large prices swings is normally low)
- there will be a fee for minting/burning which eats into profitability

The case for High is that the sponsor is:

- not assuming prices are relatively stable in #88
- Sponsor comment in #88 that "sponsor HAVE TO update price or the protocol will not work properly" means that the protocol's hand could be forced in making a large price update even if there's a scenario where it will get sandwiched

I am fine with High unless someone makes a stronger case for Medium.

SunXrex

No code change in this fix. We will update the price over Flashbot (private transaction).

jacksanford1

Understood. In that case, we'll change the label for this issue to "Won't Fix". cc @SunXrex

hrishibhat

Result: High Has duplicates

sherlock-admin

Escalations have been resolved successfully!

Escalation status:

- Shogoki: accepted
- DevABDee: accepted

jacksanford1

Acknowledged by protocol team (won't fix).

Issue M-1: In case the portfolio makes a loss, the total reserves and reserve ratio will be inflated.

Source: https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/16

Found by

ast3ros

Summary

The pool balance is transferred to the portfolio for investment, for example sending USDT to Curve/Aave/Balancer etc. to generate yield. However, there are risks associated with those protocols such as smart contract risks. In case a loss happens, it will not be reflected in the pool balance and the total reserve and reserve ratio will be inflated.

Vulnerability Detail

The assets in the pool can be sent to the portfolio account to invest and earn yield. The amount of assets in the insurance pool and Unitas pool is tracked by the `_balance` variable. This amount is used to calculate the total reserve and total collateral, which then are used to calculate the reserve ratio.

```
uint256 tokenReserve = _getBalance(token);
uint256 tokenCollateral = IInsurancePool(insurancePool).getCollateral(token);
```

When there is a loss to the portfolio, there is no way to write down the `_balance` variable. This leads to an overstatement of the total reserve and reserve ratio.

Impact

Overstatement of the total reserve and reserve ratio can increase the risk for the protocol because of undercollateralization of assets.

Code Snippet

https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/main/Unitas-Protocol/src/Unitas.sol#L508-L509 https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/main/Unitas-Protocol/src/PoolBalances.sol#L111-L113

Tool used

Manual Review

Recommendation

Add function to allow admin to write off the `_balance` in case of investment lost. Example:

```
function writeOff(address token, uint256 amount) external onlyGuardian {

    uint256 currentBalance = IERC20(token).balanceOf(address(this));

    // Require that the amount to write off is less than or equal to the current
↪   balance
    require(amount <= currentBalance, "Amount exceeds balance");
    _balance[token] -= amount;

    emit WriteOff(token, amount);
}
```

Discussion

Adityaxrex

Aditya: This will need to be manually updated. The likelyhood is small but can be contained by diversification and risk management.

SunXrex

in phase 1, reserve ratio will not impact by portfolio impact

ctf-sec

> This will need to be manually updated. The likelyhood is small but can be contained by diversification and risk management.

Intend to maintain the medium severity

0xffff11

I agree with a medium. Despite being very unlikely, the risk exists.

Shogoki

Escalate for 10 USDC I think this should be rated as low, as per the sponsors comments. I think it is similiar to #95 the design decision to use the portfolio together with the collateral & accepted risk of the protocol.

sherlock-admin

> Escalate for 10 USDC I think this should be rated as low, as per the sponsors comments. I think it is similiar to #95 the design decision to use

> the portfolio together with the collateral & accepted risk of the protocol.

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

thangtranth

Escalate for 10 USDC

@Shogoki thank you for your idea. However those are two different issues:

In #95, the sponsors explained their design choice of “taking a small portion to invest in delta neutral defi pools.” This implies that there are enough funds for users to withdraw at any time.

In this issue, the sponsors mention above how to “contain” the risks, meaning how the protocol will try to reduce the financial loss in case of a negative event. However, the loss still happens and the reserve accounting of the protocol contract is incorrect. Therefore this is a valid medium issue.

sherlock-admin

> Escalate for 10 USDC
>
> @Shogoki thank you for your idea. However those are two different issues:
>
> In #95, the sponsors explained their design choice of “taking a small portion to invest in delta neutral defi pools.” This implies that there are enough funds for users to withdraw at any time.
>
> In this issue, the sponsors mention above how to “contain” the risks, meaning how the protocol will try to reduce the financial loss in case of a negative event. However, the loss still happens and the reserve accounting of the protocol contract is incorrect. Therefore this is a valid medium issue.

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

Shogoki

I think in both cases the sent out fund are not directly available for users to withdraw.

thangtranth

No, it is not. Just one simple example that shows the difference:

As design, the protocol is at 130% reserve ratio (110% stays in pool for redemption, 20% for yield generation in portfolio). The portfolio makes a loss, which drive the reserve ratio down to 115% (110% still stays in the pool for redemption, 5% left in portfolio).

In this case, all the funds are still available for users. But the reserve ratio is hugely overstated (130% instead of 115%).

ctf-sec

Can keep the medium severity, inflated total reserves can be considered as accounting issue if the invested portfolio make a loss (countless example of rug and hack, certainly we do not hope the protocol, our sponsor deal with such situation)

jacksanford1

Agree with Medium. The case for Low is:

> I think this should be rated as low, as per the sponsors comments.

Sponsor is not saying it should be low imo. Sponsor is saying it's unlikely, but it can still be a Medium even if that's true.

> I think it is similiar to https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/95 the design decision to use the portfolio together with the collateral & accepted risk of the protocol.

It seems like the problem is that a loss of assets would not be properly reflected in the pool balance, resulting in bad outcomes. If the pool balance is not functioning properly, I don't think that's an intentional design decision.

0xffff11

It is a complicated issue so I see the case for both. Though I tend more to medium. Imo a medium makes sense on this one.

jacksanford1

This falls into the "breaks unconditional exit" category but also points out something novel which is that the _balance variable doesn't reflect a loss in the portfolio.

Valid Medium.

Adityaxrex

@jacksanford1 Thank you

hrishibhat

Result: Medium Unique

sherlock-admin

Escalations have been resolved successfully!

Escalation status:

- thangtranth: accepted
- Shogoki: rejected

jacksanford1

Acknowledged by protocol team (won't fix).

Issue M-2: USD1 is priced as $1 instead of being pegged to USDT

Source: https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/48

Found by

Ruhum, carrotsmuggler

Summary

The system treats 1 USD1 = $1 instead of 1 USD1 = 1 USDT which allows arbitrage opportunities.

Vulnerability Detail

To swap from one token to another Unitas first get's the price of the quote token and then calculates the swap result. Given that we want to swap 1 USD1 for USDT, we have USDT as the quote token:

Since `amountType == AmountType.In`, it executes `_calculateSwapResultByAmountIn()`:

Given that the price is 0.99e18, i.e. 1 USDT is worth $0.99, it calculates the amount of USDT we should receive as:

Given that:

- toBase = 10**6 = 1e6 (USDT has 6 decimals)
- fromBase = 10**18 = 1e18 (USD1 has 18 decimals)
- priceBase = 1e18
- price = 0.99e18 (1 USDT = $0.99)
- fromAmount = 1e18 (we swap 1 USD1) we get:
  $1e18 * 0.99e18 * 1e6/(1e18 * 1e18) = 0.99e6$

So by redeeming 1 USD1 I only get back 0.99 USDT. The other way around, trading USDT for USD1, would get you 1.01 USD1 for 1 USDT:
$1e6 * 1e18 * 1e18/(0.99e18 * 1e6) = 1.01e18$

The contract values USD1 at exactly $1 while USDT's price is variable. But, in reality, USD1 is not pegged to $1. It's pegged to USDT the only underlying asset.

That allows us to do the following:

- Given that USDT loses its peg slightly, e.g. it goes down to 0.997, which happens quite regularly: https://coinmarketcap.com/currencies/tether/
- we can swap a lot of USDT for USD1, wait for USDT to recover and swap back to USDT $100,000e6 * 1e18 * 1e18/(0.997e18 * 1e6) = 1.003009e+23$

With USDT back to $1weget$:1.003009e+23 * 1e18 * 1e6 / (1e18 * 1e18) = 100300.9e6$

That's a profit of 300 USDT. The profit is taken from other users of the protocol who deposited USDT to get access to the other stablecoins.

Impact

An attacker can abuse the price variation of USDT to buy USD1 for cheap.

Code Snippet

https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/main/Unitas-Protocol/src/SwapFunctions.sol#L28 https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/main/Unitas-Protocol/src/SwapFunctions.sol#L63 https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/main/Unitas-Protocol/src/SwapFunctions.sol#L203-L239

Tool used

Manual Review

Recommendation

1 USDT should always be 1 USD1. You treat 1 USD1 as $1 but that's not the case.

Discussion

Adityaxrex

Aditya: USD1 is pegged to USDT instead of 1 USD. This means the liability and asset accounting is done in USDT value.

SunXrex

This should be invalid. because we design converting rate always USD1:USDT to 1:1.

0xruhum

Escalate for 10 USDC

You don't treat 1 USDT as 1 USD1. That's exactly what I'm describing here. If 1 USDT = 1 USD1 you wouldn't have to query the price of USDT to calculate the

swap. The example calculation in the original issue shows how USDT slightly de-pegging opens up arbitrage opportunities. By USDT losing value you get more USD1. If USD1 were pegged to USDT you'd always get 1 USD1 for 1 USDT no matter the price of USDT.

Also, #79, #102, and #147 are not duplicates of this issue. Only #141 is a valid duplicate.

sherlock-admin

> Escalate for 10 USDC
>
> You don't treat 1 USDT as 1 USD1. That's exactly what I'm describing here. If 1 USDT = 1 USD1 you wouldn't have to query the price of USDT to calculate the swap. The example calculation in the original issue shows how USDT slightly de-pegging opens up arbitrage opportunities. By USDT losing value you get more USD1. If USD1 were pegged to USDT you'd always get 1 USD1 for 1 USDT no matter the price of USDT.
>
> Also, #79, #102, and #147 are not duplicates of this issue. Only #141 is a valid duplicate.

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

ctf-sec

I actually agree with the escalation, https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/79, https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/102, and https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/147 are not duplicates of this issue and this issue itself can be a valid medium

Adityaxrex

Liability and asset are both calculated in USDT. Case: 1 USDT = 0.5 USD. Mint 100 USD1 from 100 USDT. Protocol liability = 100 USDT. In nowhere in our calculation we consider dollar value of USDT. If someone is able to make profit by getting cheap USDT, it is not at the cost of the protocol but USDT. It still remains not a valid issue.

jacksanford1

@Adityaxrex Why is `oracle.getLatestPrice(priceQuoteToken)` being used in the swap request shown in @0xruhum's example above if it isn't being taken into account to swap between USD1 and USDT?

@0xffff11 What is your understanding of this situation?

Adityaxrex

@jacksanford1 in this version our protocol's main backing asset is USDT. In all practical purpose we fix 1USDT=1USD1. Even if USDT depegs, we keep 1USDT=1USD1. It does not change accounting whether we check the price or not. This helps us keep the design simple. However, we eventually move towards diversifying to other assets.

0xruhum

As shown in the code snippets and the example calculation, the price **does** have an effect on the swap. While the protocol team's intention might have been to treat 1 USD1 as 1 USDT, the actual implementation doesn't do that.

jacksanford1

Ok, it seems we have a disagreement about how the actual implementation works.

@ctf-sec @0xffff11 Can you please verify if the swap function above is swapping USD1 to USDT (or vice versa) using a fixed 1:1 ratio or if the swap is being done based on the price of USDT returned by Chainlink?

0xffff11

@jacksanford1 I did a deep research on the issue. It seems like it is not redeeming paired 1:1 usd1 for usdt or vice versa. The price of the `quote` token, in this case `usdt`, is used for the calculation of the amountOut instead of a 1:1 ratio. So the depeg of usdt, in my opinion, would affect the exchange rate. I think it is a valid issue and as a comment above, most dups are invalid

SunWeb3Sec

We keep the swap logic consistent, so instead of hard-coding a 1:1 ratio in the code, we maintain it through our Oracle price updates.

Adityaxrex

As mentioned by @SunWeb3Sec we keep the logic consistent and check the price for any pair periodically. However, our feeder logic keeps updating the price for USD1 and USDT price just like any other pair. As mentioned earlier we plan to keep USDT:USD1 = 1:1 for now through our feeder logic. This can be changed in future to reflect the true market price. Thank you

0xruhum

That wouldn't work. You price USDT as 1$ in that case without it being worth 1$ if it depegs.

The oracle price is used to calculate the protocol's collateral value: https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/main/Unitas-Protocol/src/Unitas.sol#L512

If you don't update the price to keep the 1:1 ratio with USD1, `totalReserves` won't reflect the actual value of your collateral. That breaks the reserve ratio calculation.

Adityaxrex

@jacksanford1 @SunWeb3Sec the above argument is similar to issue #145. Upto you to decide whether to accept this as duplicate or the other one. Thank you

0xruhum

The argument for not keeping the price of USDT hardcoded as $1 is the same yeah. But the two reported issues are different. One describes the issue of assets being frozen because the swap functionality breaks in case the price outside the min/max values. The other shows how price movement in USDT causes the system to mint either more or less USD1 than it should. It's not a duplicate of #145 IMO.

The next time, these kinds of assumptions regarding off-chain mechanics, like the oracle values in this case, should be mentioned in the contest's README. Having them come up during the judging period is annoying for all parties.

Adityaxrex

Please see that when user tries to redeem USDT, we also use 1:1. This way we avoid any loss to the protocol. If the user is able to obtain cheaper USDT compared to 1 USD it is at the cost of the USDT seller not the protocol. Thank you

Adityaxrex

@0xruhum essentially the minter is able to make profit but the loss is coming from anyone selling USDT for lesser than 1 USD. In terms of USD1:USDT, mint and burn are both at 1:1 so there is no need to worry from the protocol point of view. It can create a scenario that protocol owns a lot of USDT which is worth less because of USDT depeg but to the protocol that is not an issue because we only redeem USDT not USD

0xruhum

The value of your tokens depends on the value of your underlying collateral. If the value of USDT falls below $1, your derivative tokens become unreserved. At that point, they will depeg as well. You can't just hardcode the price of USDT as $1. That's what I'm trying to explain here.

Here's a small example: Given that you have a reserve ratio threshold of 100%, e.g. you need 1$ of collateral to mint $1 of your tokens.

Alice mints 100 USD1 using 100 USDT. She then swaps 100 USD1 for 8200 INR (the current exchange rate is 1:82). At this point you have 100 USDT covering 8200 INR. The INR token only has that value because I can use the Unitas protocol to get back the underlying USDT. Otherwise, INR is worth nothing because there's no asset backing it. Now let's say USDT depegs and is only worth $0.9. At this point, you have $90 worth of USDT backing 8200 INR tokens. That doesn't work, right? Because USDT the underlying asset lost 10% of its value, your INR token will also lose value. Its value will now move to 8200 * 0.9 = 7380 INR while the token amount stays the same. Since your whole protocol depends on these derivative

tokens mimicking the price of the asset they try to represent, e.g. INR, everything breaks. The actual USD/INR ratio hasn't changed at all. But because your underlying asset depegged, your derivative token also depegs.

Keeping USDT:USD1 at 1:1 won't protect your derivative tokens from depegging. What you do with that is to keep the reserve ratio artificially at a healthy ratio while the **real** reserve ratio keeps going down. So all the reserve ratio shenanigans of what tokens are supposed to be mintable and burnable at different stages won't really matter.

To sum up, if we assume that your oracle just says that USDT is worth $1 at any time, you've got the problem I've described above. If your oracle returns the correct value for USDT, whether it depegs or not, your swap calculation causes you to mint the wrong amount of USD1 as seen in the original issue.

So no matter what you do, the current design is flawed.

jacksanford1

Agree that this is not a duplicate of #145, it describes a different part of the code as @0xruhum explains above.

Based on @0xffff11's research, it seems like it was not obvious that a 1:1 ratio would be kept here. It was reasonable to assume the oracle would be using the latest trading price of the asset. So that makes this a Medium issue imo because the functioning of this oracle doesn't seem to have been clearly stated as being hardcoded 1:1. Open to correction on this. Here's a statement from the docs which makes it sound like the conversion would be value-based instead of 1:1:

> USD91 minters need to provide USDT, the protocol will mint equivalent in value USD91

I'm unclear as to whether the 1:1 hardcoding will cause additional issues in the protocol, but the team is acknowledging that the protocol should function in a way that takes into account the ramifications of hardcoding the price.

0xruhum

Yeah, the 1:1 conversion wasn't clearly stated. But, why does that change the severity of the issue?

If we go with the original assumption that the oracle will use the correct trading price of USDT, the swap calculation is broken. Because while using the real trading price for USDT, you continue to treat USD1 as worth $1. As described in the original issue, when the price of USDT drops you're able to mint more USD1 for any given amount of USDT. At the same time you're able to swap USD1 for any of the other derivative tokens at the actual conversion rate of those two assets, e.g. USD/INR. That's a HIGH. The core mechanics of the protocol are broken.

If we assume that the price is hardcoded to 1:1, we have a totally different situation. The reserve calculation breaks which causes the protocol to believe it's fully

collateralized while it isn't. Even if USDT drops to $0.9, the protocol continues to treat it as $1 which keeps the reserve ratio above the threshold. But, that doesn't stop the derivative tokens from depegging. Since the value of each derivative token depends on the value of the underlying asset. Again, a HIGH severity issue since another core mechanic of the protocol (distribution of derivative tokens that represent currencies like INR and TRY) is broken.

jacksanford1

@Adityaxrex @lucas-xrex @SunXrex Any response to the second paragraph about the downside of hardcoding the oracle at a 1:1 ratio?

@0xruhum If USD1 is hardcoded 1:1 to USDT, it wouldn't mean that USD91 depegs right? Because the conversion ratio between USD91 and USDT would take into account the fact that USDT is now trading at $0.90?

0xruhum

There are no swaps from USDT to any of the other derivative tokens. There's only USDT/USD1. USD1 is then used to trade to the other derivative tokens. The conversion between USD1 and the other derivative tokens will represent the current exchange rate between USD and those other currencies not the actual conversion rate between the tokens themselves.

USDT depegging will cause all the derivative tokens to depeg. The only reason these tokens have any value is because they are backed by an asset with value. USDT is worth $1 because you can exchange it for $1 through Tether no matter what happens (at least that's the idea). That's the reason USDT is pegged at $1.

USD1 is only worth something because it's backed by USDT. I know my USD1 is worth something because I can trade it for USDT which the market agrees is worth X US Dollars. If the price of USDT drops, the price of USD1 will also drop.

USD91 is only worth something because it's backed by USD1. If USD1 loses value, e.g. through USDT depegging, USD91 will also lose value and depeg. Even if I trade my USD91 at the same rate I initially bought it for back to USD1, the asset I get is worth less than at the time of my trade. It doesn't matter that I get the same amount of USD1 back. It's simply not worth the same amount as back then. The same logic applies to all the other derivative tokens as well.

The system here is just one long chain: $1 -> USDT -> USD1 -> USD91. If any of the links break, the whole system will blow up. Meaning, as soon as USDT depegs and doesn't trade for $1, all of your tokens will depeg as well. USDT is the only thing that gave them value.

Adityaxrex

We have discussed this internally and concluded that USDT depeg is not an issue for us. Thank you

On Sat, Jul 1, 2023 at 5:23 PM ruhum **@**.***> wrote:

There are no swaps from USDT to any of the other derivative tokens. There's only USDT/USD1. USD1 is then used to trade to the other derivative tokens. The conversion between USD1 and the other derivative tokens will represent the current exchange rate between USD and those other currencies not the actual conversion rate between the tokens themselves.

USDT depegging will cause all the derivative tokens to depeg. The only reason these tokens have any value is because they are backed by an asset with value. USDT is worth $1 because you can exchange it for $1 through Tether no matter what happens (at least that's the idea). That's the reason USDT is pegged at $1.

USD1 is only worth something because it's backed by USDT. I know my USD1 is worth something because I can trade it for USDT which the market agrees is worth X US Dollars. If the price of USDT drops, the price of USD1 will also drop.

USD91 is only worth something because it's backed by USD1. If USD1 loses value, e.g. through USDT depegging, USD91 will also lose value and depeg. Even if I trade my USD91 at the same rate I initially bought it for back to USD1, the asset I get is worth less than at the time of my trade. It doesn't matter that I get the same amount of USD1 back. It's simply not worth the same amount as back then. The same logic applies to all the other derivative tokens as well.

The system here is just one long chain: $1 -> USDT -> USD1 -> USD91. If any of the links break, the whole system will blow up. Meaning, as soon as USDT depegs and doesn't trade for $1, all of your tokens will depeg as well. USDT is the only thing that gave them value.

— Reply to this email directly, view it on GitHub https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/48#issuecomment-1615773614, or unsubscribe https://github.com/notifications/unsubscribe-auth/AXN5N655U56CPSLAAC4NNRLXN7UAPANCNFSM6AAAAAAZDTL374 . You are receiving this because you were mentioned.Message ID: **@**.*** .com>

jacksanford1

@0xruhum @Adityaxrex @SunXrex Last question from me:

When USD1 is being converted into USD91 and vice versa, how is the price determined? Is it using the exchange rate between USD and rupees? Or the exchange rate between USDT and rupees?

Adityaxrex

It is with with exchange rate using USDT and INR. Thank you

On Tue, Jul 4, 2023 at 10:50 AM Jack Sanford @.***> wrote:

> @0xruhum https://github.com/0xruhum @Adityaxrex https://github.com/Adityaxrex @SunXrex https://github.com/SunXrex Last question from me:
>
> When USD1 is being converted into USD91 and vice versa, how is the price determined? Is it using the exchange rate between USD and rupees? Or the exchange rate between USDT and rupees?
>
> — Reply to this email directly, view it on GitHub https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/48#issuecomment-1619388479, or unsubscribe https://github.com/notifications/unsubscribe-auth/AXN5N65W5EKTA3L2WCBBTPLXOOAG7ANCNFSM6AAAAAAZDTL374 . You are receiving this because you were mentioned.Message ID: @.*** .com>

jacksanford1

Ok, in that case I think @0xruhum's statement below would not be true:

> The conversion between USD1 and the other derivative tokens will represent the current exchange rate between USD and those other currencies not the actual conversion rate between the tokens themselves.

I think the problem is that @0xruhum assumes everything is based around the value of USD ($1) when in reality it's based around USDT (1 USDT). And so if USDT depegs, then it's correct that everything else (USD1, USD91) depegs from $1 but the system doesn't care because it's all based in USDT. So there's no problem.

This chain from @0xruhum would not be correct:

> The system here is just one long chain: $1 -> USDT -> USD1 -> USD91. If any of the links break, the whole system will blow up.

And instead the chain is simply USDT -> USD1 -> USD91.

However, because it was not clear in the code or docs that the Chainlink oracle for USD/USDT would not be used in this specific case, the issue should stay a valid Medium imo. With #141 as a duplicate.

hrishibhat

Result: Medium Has duplicates

sherlock-admin

Escalations have been resolved successfully!

Escalation status:

- 0xruhum: accepted

jacksanford1

Acknowledged by protocol team (won't fix).

Issue M-3: No slippage or deadline control for swapping while stability burning

Source: https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/88

Found by

0x4non, 0xGoodess, Jiamin, Juntao, PawelK, circlelooper, ctf_sec, moneyversed, okolicodes, radev_sw, stopthecap, tsvetanovv

Summary

No slippage or deadline control for swapping while stability burning

Vulnerability Detail

Even though Unitas claims there will not be slippage because if it is burned from one side, it is minted in the other one, there is an edge-case where it does create a slight slippage depending on the burn amount. Unitas intends to make stability burns when the reserve ratio is below 130% to try and get it back to normal levels. This burn, is a one sided burn of `usd1`, which reduces the `totalSupply` of `usd1` unilaterally. https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/main/Unitas-Protocol/src/Unitas.sol#L208

If any user is trying to swap `usd1` while there is a stability burn being conducted, they will be affected by that slippage.

The other recommendation is the usage of a deadline param. Without a deadline parameter, the transaction may sit in the mempool and be executed at a much later time potentially resulting in a swap after/before a stability burn.

Impact

User will be affected by unintended and unhandled slippage, potentially affecting the funds they get back from the swap

Code Snippet

https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/main/Unitas-Protocol/src/Unitas.sol#L208

Tool used

Manual Review

Recommendation

Allow a user to specify to key parameters, a `deadline` and a `minOutAmount`. And make a check for both at start and end of the execution in the swap function.

Discussion

SunXrex

Sun: In this phase we don't support stability burn.

Aditya: Slippage on a swap DEX like Uniswap is part of the design. This is to ensure that the pool does not get fully empty. So if the user places a large order, they see a slippage in price. Every order on a DEX like Uniswap, moves the price. Large orders move the price in higher degree. As the order grows, the slippage also grows. This is inherent property of xy=k bonding curve on which Uniswap operate.

In case of Unitas, every swap order is essentially mint/burn order. As long as there is enough USDT in reserve and insurance pool, protocol can mint and burn any amount. This leads to 1:1 value transfer without slippage. Unitas does not mint on a bondig curve, instead the protocol simply mints new tokens or burns the existing. Eg: On uniswap if I buy 1 ETH, I will face low slippage but if I buy 1000 ETH, the slippage will be much higher. However, in case of Unitas whether the user mints 1000 USD91 or 1million USD91, the user will get all the tokens at Oracle price.

In case of Uniswap, the DEX does not have the authority to mint/burn but only facilitate a swap. In case of Unitas, the protocol has the authority to mint and burn. This leads to no slippage for minters/burners.

Adityaxrex

This style of slippage is low probability for our design. We may fix it in the future but edge case scenario for now.

SunXrex

It should be considered a low-medium risk because the chances of encountering it are very low.

ctf-sec

> This style of slippage is low probability for our design. We may fix it in the future but edge case scenario for now.

medium

hrishibhat

Considering this issue as a medium based on the above comments

Shogoki

Escalate for 10 USDC I think this issue should be rated as a valid low instead of medium. Reasons:

- Sponsor stated here already that they consider it low probability.
- In the contest Readme: “getting different price for price update in the same block” was excluded, which is similiar.
- according to the sponsor price updates are not that frequent, so the tx would have to stay a long time in the mempool, for this issue
- Issue #67 was downgraded because the prices are in general quite stable, same goes here and diminishes the impact of the issue

sherlock-admin

> Escalate for 10 USDC I think this issue should be rated as a valid low instead of medium. Reasons:
>
> - Sponsor stated here already that they consider it low probability.
> - In the contest Readme: “getting different price for price update in the same block” was excluded, which is similiar.
> - according to the sponsor price updates are not that frequent, so the tx would have to stay a long time in the mempool, for this issue
> - Issue #67 was downgraded because the prices are in general quite stable, same goes here and diminishes the impact of the issue

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

juntzhan

Escalate for 10 USDC

Hi @Shogoki glad to know your thoughts.

This issue is a valid medium, because price is based on currency exchange rates and it is not fair to say the price is stable or not (depends on the currency itself).

When EMC appriciates / depreciates, the sponsor **HAVE TO** update price or the protocol will not work properly.

Please see #96 for a reference.

sherlock-admin

> Escalate for 10 USDC
>
> Hi @Shogoki glad to know your thoughts.

> This issue is a valid medium, because price is based on currency exchange rates and it is not fair to say the price is stable or not (depends on the currency itself).
>
> When EMC appriciates / depreciates, the sponsor **HAVE TO** update price or the protocol will not work properly.
>
> Please see #96 for a reference.

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

ctf-sec

Can keep the medium severity

0xffff11

I strongly believe it should keep a med too.

jacksanford1

In order to be Medium severity, the magnitude of the loss has to be quite material.

I don't see anything that points to the magnitude of loss with this slippage issue being very material. @Shogoki makes a good point that there also needs to be proof that this issue doesn't fall into the "known issues" category from the README:

> In the contest Readme: “getting different price for price update in the same block” was excluded, which is similiar.

I also agree with @juntzhan that we can't rely on the idea that the price is stable:

> because price is based on currency exchange rates and it is not fair to say the price is stable or not (depends on the currency itself).

The probability of this loss seems fairly low, but even if the probability were medium (instead of low) the impact needs to be large enough to be considered material. And it seems hard to prove that the magnitude of loss could be large, especially considering the stability burn mechanism is probably set up in a way where it shouldn't cause large slippage (someone can check me on that).

@0xffff11 @ctf-sec In order for this to be considered Medium I think two things need to be cleared up:

1) This technically does not fall under the README known issue above
2) The magnitude of loss can be large enough to warrant a Medium

juntzhan

In order to be Medium severity, the magnitude of the loss has to be quite material.

I don't see anything that points to the magnitude of loss with this slippage issue being very material. @Shogoki makes a good point that there also needs to be proof that this issue doesn't fall into the "known issues" category from the README:

> In the contest Readme: “getting different price for price update in the same block” was excluded, which is similiar.

I also agree with @juntzhan that we can't rely on the idea that the price is stable:

> because price is based on currency exchange rates and it is not fair to say the price is stable or not (depends on the currency itself).

The probability of this loss seems fairly low, but even if the probability were medium (instead of low) the impact needs to be large enough to be considered material. And it seems hard to prove that the magnitude of loss could be large, especially considering the stability burn mechanism is probably set up in a way where it shouldn't cause large slippage (someone can check me on that).

@0xffff11 @ctf-sec In order for this to be considered Medium I think two things need to be cleared up:

1. This technically does not fall under the README known issue above
2. The magnitude of loss can be large enough to warrant a Medium

1. the README known issue "getting different price for price update in the same block" assumes the price update and swap happen at the same **POINT** of time and the possibility is very low, while this issue decribes a **PERIOD** of time during which the price update happens, the possibility is much higher, so they are different issues
2. similar to traditional DEX, user may suffer a large loss without slippage protection, (assume USDEMC/USD1 is 1 : 10) a swap with 10000 USDEMC could result in 100 LESS USD1 due to 10% slippage caused by depreciation, and user could lose more if he swaps more or USDEMC depreciates more

UPDATE: I don't think stability burn mechanism does much help to migitate slippage, as the price would be significantly influenced by the currency exchage rates in external financial markets (as confirmed by sponsor, price data is from multiple cex)

jacksanford1

@Shogoki What do you think about juntzhan's points?

Shogoki

@Shogoki What do you think about juntzhan's points?

Thanks for enquiring my feedback. Appreciate it :-)

Regarding 1. the mentioned issue in the Readme: It does not say, it happens at the same time, but only “in the same block”. One might say it is probably assumed like this, but it is not clearly stated. . .

However, as also stated in my escalation on #67, if we do not count the argument from the sponsor of stable prices, which was the reason why #67 was downgraded, i believe this one can also be counted as Medium.

hrishibhat

Result: Medium Has duplicates

sherlock-admin

Escalations have been resolved successfully!

Escalation status:

- juntzhan: accepted
- Shogoki: rejected

jacksanford1

Acknowledged by protocol team (won't fix).

Issue M-4: Users may not be able to fully redeem USD1 into USDT even when reserve ratio is above 100%

Source: https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/95

Found by

0x00ffDa, 0xyPhilic, Jiamin, Juntao, PokemonAuditSimulator, Ruhum, sashik_eth

Summary

Users may not be able to fully redeem USDT even when reserve ratio is above 100%, because of portfolio being taken into the account for calculation.

Vulnerability Detail

Reserve ratio shows how many liabilities is covered by reserves, a reserve ratio above 100% guarantees protocol has enough USDT to redeem, the way of calculating reserve ratio is `Reserve Ratio = allReserves / liabilities` and is implemented in Unitas#_getReserveStatus(...) function:

```
reserveRatio = ScalingUtils.scaleByBases(
    allReserves * valueBase / liabilities,
    valueBase,
    tokenManager.RESERVE_RATIO_BASE()
);
```

`allReserves` is the sum of the balance of Unitas and InsurancePool, calculated in Unitas#_getTotalReservesAndCollaterals() function:

```
for (uint256 i; i < tokenCount; i++) {
    address token = tokenManager.tokenByIndex(tokenTypeValue, i);
    uint256 tokenReserve = _getBalance(token);
    uint256 tokenCollateral = IInsurancePool(insurancePool).getCollateral(token);


    if (tokenReserve > 0 || tokenCollateral > 0) {
        uint256 price = oracle.getLatestPrice(token);


        reserves += _convert(
            token,
            baseToken,
            tokenReserve,
```

```
            MathUpgradeable.Rounding.Down,
            price,
            priceBase,
            token
        );


        collaterals += _convert(
            token,
            baseToken,
            tokenCollateral,
            MathUpgradeable.Rounding.Down,
            price,
            priceBase,
            token
        );
    }
}
```

`liabilities` is the total value of USD1 and USDEMC tokens, calculated in Unitas#_getTotalLiabilities() function:

```
for (uint256 i; i < tokenCount; i++) {
    address token = tokenManager.tokenByIndex(tokenTypeValue, i);
    uint256 tokenSupply = IERC20Token(token).totalSupply();


    if (token == baseToken) {
        // Adds up directly when the token is USD1
        liabilities += tokenSupply;
    } else if (tokenSupply > 0) {
        uint256 price = oracle.getLatestPrice(token);


        liabilities += _convert(
            token,
            baseToken,
            tokenSupply,
            MathUpgradeable.Rounding.Down,
            price,
            priceBase,
            token
        );
    }
}
```

Some amount of USDT in both Unitas and InsurancePool is portfolio, which

represents the current amount of assets used for strategic investments, it is worth noting that after sending portfolio, `balance` remains the same, which means `portfolio` is taken into account in the calculation of reserve ratio.

This is problematic because `portfolio` is not available when user redeems, and user may not be able to fully redeem for USDT even when protocols says there is sufficient reserve ratio.

Let's assume :

> Unitas's balance is 10000 USD and its portfolio is 2000 USD, avaliable balance is 8000 USD InsurancePool's balance is 3000 USD and its portfolio is 600 USD, available balance is 2400 USD AllReserves value is 13000 USD Liabilities (USDEMC) value is 10000 USD Reserve Ratio is (10000 + 3000) / 10000 = 130%.

Later on, USDEMC appreciates upto 10% and we can get:

> AllReserves value is still 13000 USD Liabilities (USDEMC) value is 11000 USD Reserve Ratio is (10000 + 3000) / 11000 = 118%.

The available balance in Unitas is 8000 USD so there is 3000 USD in short, it needs to be obtain from InsurancePool, however, the available balance in InsurancePool is 2400 USD, transaction will be reverted and users cannot redeem.

There would also be an extreme situation when reserve ratio is above 100% but there is no available balance in protocol because all the `balance` is `portfolio` (this is possible when InsurancePool is drained out), users cannot redeem any USDT in this case.

Impact

Users may not be able to fully redeem USD1 into USDT even when reserve ratio is above 100%, this defeats the purpose of reserve ratio and breaks the promise of the protocol, users may be mislead and lose funds.

Code Snippet

https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/main/Unitas-Protocol/src/Unitas.sol#L500-L535

Tool used

Manual Review

Recommendation

Portfolio should not be taken into account for the calculation of reserve ratio.

```
    function _getTotalReservesAndCollaterals() internal view returns (uint256
↪   reserves, uint256 collaterals) {
        ...
-           uint256 tokenReserve = _getBalance(token);
+           uint256 tokenReserve = _getBalance(token) - _getPortfolio(token);
-           uint256 tokenCollateral =
↪   IInsurancePool(insurancePool).getCollateral(token);
+           uint256 tokenCollateral =
↪   IInsurancePool(insurancePool).getCollateral(token) -
↪   IInsurancePool(insurancePool).getPortfolio(token);
        ...
    }
```

Discussion

SunXrex

We think it's invalid. it's design decision.

Aditya: The purpose of insurance pool is to support additional USDT requirements in the scenario the reserve pool does not have required amount. To simplify this, we keep only a small portion of assets for yield generation. Priorities : User redemption above yield generation. Ie: Only when there is enough capital in reserve and insurance pool, we take a small portion to invest in delta nuetral defi pools.

ctf-sec

Protocol's design choice, not a issue

0xruhum

Escalate for 10 USDC This is not just a design choice. There are no specifics on what a "small portion" or "enough capital in reserve and insurance pool" means. Allowing the protocol team to remove collateral from the system to earn yield is a security risk. In the current version, USDT is the collateral for USD1. Without USDT, USD1 is worth nothing. The reserve ratio is used to keep the protocol healthy at all times. By including the funds inside the portfolio in the calculation you allow the protocol to become under-collateralized under certain circumstances.

Considering that #16 describes the risk of the portfolio incurring a loss which is deemed a valid issue, it's reckless to account for the funds inside the portfolio when calculating the reserve ratio. From a protocol perspective, these funds are not part of the system anymore. They were moved into a different protocol to earn yield. That is fine as long as you keep the reserve ratio of Unitas above 130%. For that to be the case, the funds inside the portfolio have to be **excluded** from the reserve ratio calculation.

To sum up, any funds inside the portfolio should not be used to calculate the reserve ratio. They can be gone at any moment due to the risks described in #16.

EDIT:

- #14 is not a valid duplicate. If portfolio > collateral the subtraction would revert so there's no underflow there.
- #43 is not a valid duplicate. Warden argues that `_getPortfolio()` is not called although it clearly is in the code snipped they provide. The submission makes no sense
- #90 is not a valid duplicate. Warden argues that a swap will fail if there are not enough funds in the insurance pool. That's desired behavior. You don't want to execute the swap if there are not enough funds for it.
- #127 is not a valid duplicate. Warden argues that the portfolio amount is accounted for multiple times. But, the insurance pool is a separate contract. The portfolio amount would be 0 for the insurance pool. Thus you don't account for it twice.
- #129 is not a valid duplicate. The issue itself is valid. The admin is able to leave the protocol under-collateralized by sending to many funds to the portfolio. That's a duplicate of #40
- #142 is not a valid duplicate. Tokens that are not registered in the TokenManager are not used as collateral. Even if the insurance pool holds these tokens they shouldn't be accounted for since they are not registered collateral tokens. Also, the admin has to make a mistake and send these tokens to the insurance pool since they are the only ones that are allowed to deposit.

I checked the docs on whether each of these duplication issues had to be escalated separately. I didn't find any rules for that. So I decided to bring them up in a single comment. Otherwise, I'd had to create 6 additional escalations without even knowing whether this original issue here is deemed valid. If that's not the case you don't care about the wrong duplication anyways.

sherlock-admin

> Escalate for 10 USDC This is not just a design choice. There are no specifics on what a "small portion" or "enough capital in reserve and insurance pool" means. Allowing the protocol team to remove collateral from the system to earn yield is a security risk. In the current version, USDT is the collateral for USD1. Without USDT, USD1 is worth nothing. The reserve ratio is used to keep the protocol healthy at all times. By including the funds inside the portfolio in the calculation you allow the protocol to become under-collateralized under certain circumstances.
>
> Considering that #16 describes the risk of the portfolio incurring a loss

which is deemed a valid issue, it's reckless to account for the funds inside the portfolio when calculating the reserve ratio. From a protocol perspective, these funds are not part of the system anymore. They were moved into a different protocol to earn yield. That is fine as long as you keep the reserve ratio of Unitas above 130%. For that to be the case, the funds inside the portfolio have to be **excluded** from the reserve ratio calculation.

To sum up, any funds inside the portfolio should not be used to calculate the reserve ratio. They can be gone at any moment due to the risks described in #16.

EDIT:

- #14 is not a valid duplicate. If portfolio > collateral the subtraction would revert so there's no underflow there.
- #43 is not a valid duplicate. Warden argues that `_getPortfolio()` is not called although it clearly is in the code snipped they provide. The submission makes no sense
- #90 is not a valid duplicate. Warden argues that a swap will fail if there are not enough funds in the insurance pool. That's desired behavior. You don't want to execute the swap if there are not enough funds for it.
- #127 is not a valid duplicate. Warden argues that the portfolio amount is accounted for multiple times. But, the insurance pool is a separate contract. The portfolio amount would be 0 for the insurance pool. Thus you don't account for it twice.
- #129 is not a valid duplicate. The issue itself is valid. The admin is able to leave the protocol under-collateralized by sending to many funds to the portfolio. That's a duplicate of #40
- #142 is not a valid duplicate. Tokens that are not registered in the TokenManager are not used as collateral. Even if the insurance pool holds these tokens they shouldn't be accounted for since they are not registered collateral tokens. Also, the admin has to make a mistake and send these tokens to the insurance pool since they are the only ones that are allowed to deposit.

I checked the docs on whether each of these duplication issues had to be escalated separately. I didn't find any rules for that. So I decided to bring them up in a single comment. Otherwise, I'd had to create 6 additional escalations without even knowing whether this original issue here is deemed valid. If that's not the case you don't care about the wrong duplication anyways.

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

ctf-sec

Will bring for sponsor review, I think this can be a valid medium

thangtranth

Escalate for 10 USDC

Reading the original report, I see no mentioning of the potential path or risk that leads to the issue. However in the escalation, the arguments are built upon the insights and finding of another report after submission. If the original report stands alone, I am not sure that it is convincing enough.

This escalation also aims to clarify the rules for other Watsons who may do the same in the future.

sherlock-admin

> Escalate for 10 USDC
>
> Reading the original report, I see no mentioning of the potential path or risk that leads to the issue. However in the escalation, the arguments are built upon the insights and finding of another report after submission. If the original report stands alone, I am not sure that it is convincing enough.
>
> This escalation also aims to clarify the rules for other Watsons who may do the same in the future.

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

jacksanford1

@0xruhum I see what you're getting at. But the only way this can be a valid issue is if the protocol has intended for users to always be able to redeem USD1 into USDT when the reserve ratio is 100% or greater. I don't think Unitas has made that claim anywhere? If you see a claim like this (in comments or in code), please show me.

jacksanford1

Update @0xruhum: Based on the following sentence in the README, the broader point of this issue should be true:

> The Unitas protocol guarantees unrestricted and unconditional conversion of its unitized stablecoins “back” to USD-pegged stablecoins.

The issue correct gets that "users may not be able to fully redeem USD1 into USDT even when the reserve ratio is above 100%" so this issue should be a valid Medium (maybe even High). But the portfolio aspect is not super unique, so it should be duplicated with any other issues that get the general idea of this vulnerability.

jacksanford1

Escalation accepted

Duplicate of #13

0xruhum

> But the portfolio aspect is not super unique, so it should be duplicated with any other issues that get the general idea of this vulnerability.

Yep. Just wanted to mention again that the issues I've linked in the original escalation comment aren't valid duplicates tho. That should be taken into account.

jacksanford1

Ok @0xruhum, as far as I'm aware none of them are considered to be duplicates right now.

Jiamincoin

> Ok @0xruhum, as far as I'm aware none of them are considered to be duplicates right now.

Hey I think #114 should be taken into account too. I didn't create escalation because it was considered duplicate of the main report in the first place.

jacksanford1

Ok @Jiamincoin, I agree that #114 is a duplicate of this issue.

0x00ffDa

I do not see any discussion of duplicate #118. Please consider it before ending the escalation evaluations.

hrishibhat

Result: Medium Has duplicates

sherlock-admin

Escalations have been resolved successfully!

Escalation status:

- 0xruhum: accepted

- thangtranth: rejected

jacksanford1

Acknowledged by protocol team (won't fix).

Issue M-5: If any stable depegs, oracle will fail, disabling swaps

Source: https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/145

Found by

Ruhum, stopthecap

Summary

If any stable depegs, oracle will fail, disabling swaps

Vulnerability Detail

When swapping, the price of the asset/stable is fetched from OracleX. After fetching the price, the deviation is checked in the `_checkPrice` function.

https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/d5328421bea80e3b0fd4595e4eb6b732a40e421e/Unitas-Protocol/src/Unitas.sol#L595-L600

If the price of an asset/stable depegs, the following require will fail:

Due to the fail in the deviation, any swapping activity will be disabled by default and transactions will not go through

Impact

Core functionality of the protocol will fail to work if any token they fetch depegs and its price goes outside the bounds.

Code Snippet

https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/d5328421bea80e3b0fd4595e4eb6b732a40e421e/Unitas-Protocol/src/Unitas.sol#L440

https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/d5328421bea80e3b0fd4595e4eb6b732a40e421e/Unitas-Protocol/src/Unitas.sol#L595-L600

Tool used

Manual Review

Recommendation

Use a secondary oracle when the first one fails and wrap the code in a try catch and store the last fetched price in a variable

Discussion

SunXrex

Sun: minPrice and maxPrice are used as a precautionary measure in case the feeder role is compromised. Additionally, the min and max prices are set to values that are not expected to be reached in the market.

We think shoud be invaild.

Aditya: Hyper change in EMC price is not accepted beyond a limit by the oracle. Oracle will reject the new price and retain the old price. This will ensure swaps happen. Although this may not happen at the true market rate.

Adityaxrex

before the price is sent to Oracle, it is processed by oracle feeder. Feeder rejects hyper change to the prices. Instead the feeder will send maximum allowed change per update.

ctf-sec

The minPrice and maxPrice are protocol's design choice to add oracle protection. not a valid issue

0xffff11

If as @Adityaxrex explains, the feeder "rejects hyper change to the prices. Instead the feeder will send maximum allowed change per update." , I agree invalid. It is pretty much the same as the proposed fixed of providing a valid price (what they are doing under the hood). So in theory they already safeguard for this according to sponsor comments

0xruhum

Escalate for 10 USDC

> Hyper change in EMC price is not accepted beyond a limit by the oracle. Oracle will reject the new price and retain the old price.
>
> Feeder rejects hyper change to the prices. Instead the feeder will send maximum allowed change per update.

This was not communicated by the protocol team. Wardens assumed that the oracle would at all times try to represent the true price of the token.

Limiting the oracle to certain values is in itself a security issue. Even if USDT loses its peg you want to continue using $1 for its price? If INR loses value against USD

you don't update it either? That would break the whole system since you open up tons of arbitrage opportunities with other protocols.

If you have a strict limit on prices, you run into the issue that big movements in the market will leave you with outdated price data since it moves outside of your limits. By setting a broader limit you allow more accurate pricing if these movements happen but don't protect against the feeder being compromised. If an attacker is able to move the price even by 10% they are able to drain the protocol's collateral (move price up -> take loan -> move price down -> repay loan).

These price limits offer no protection against the feeder being compromised. You have to find a different solution for that. For example, instead of having just 1 EOA that's allowed to push price data, you set up multiple ones in different locations and then take the avg price. That would mean that the attacker has to compromise a majority of the EOA to have any real impact on pricing. Instead, you implement a naive price limit that will just cause your system to break whenever any big price movements happen. Just take USDC as an example. A stablecoin that was generally regarded as the safest out there. Even DAI uses it as collateral. Because of the banking crisis in the USA back in March/April, the price dropped below 90 cents. USDT is also full of uncertainty.

To update the price limits you have to go through the timelock. Quick reactions to these unexpected events will be impossible.

If you push price data that's outside the price limit you'll cause swaps to halt. That means users can't repay their loans to take out their collateral. Effectively you freeze their funds. If you don't push price data that's outside the price limit, you force the protocol to work with outdated price data which opens up arbitrage opportunities that will break the protocol. Either way, you have a security issue here.

sherlock-admin

> Escalate for 10 USDC
>
> > Hyper change in EMC price is not accepted beyond a limit by the oracle. Oracle will reject the new price and retain the old price.
> >
> > Feeder rejects hyper change to the prices. Instead the feeder will send maximum allowed change per update.
>
> This was not communicated by the protocol team. Wardens assumed that the oracle would at all times try to represent the true price of the token.
>
> Limiting the oracle to certain values is in itself a security issue. Even if USDT loses its peg you want to continue using $1 for its price? If INR loses value against USD you don't update it either? That would break the

whole system since you open up tons of arbitrage opportunities with other protocols.

If you have a strict limit on prices, you run into the issue that big movements in the market will leave you with outdated price data since it moves outside of your limits. By setting a broader limit you allow more accurate pricing if these movements happen but don't protect against the feeder being compromised. If an attacker is able to move the price even by 10% they are able to drain the protocol's collateral (move price up -> take loan -> move price down -> repay loan).

These price limits offer no protection against the feeder being compromised. You have to find a different solution for that. For example, instead of having just 1 EOA that's allowed to push price data, you set up multiple ones in different locations and then take the avg price. That would mean that the attacker has to compromise a majority of the EOA to have any real impact on pricing. Instead, you implement a naive price limit that will just cause your system to break whenever any big price movements happen. Just take USDC as an example. A stablecoin that was generally regarded as the safest out there. Even DAI uses it as collateral. Because of the banking crisis in the USA back in March/April, the price dropped below 90 cents. USDT is also full of uncertainty.

To update the price limits you have to go through the timelock. Quick reactions to these unexpected events will be impossible.

If you push price data that's outside the price limit you'll cause swaps to halt. That means users can't repay their loans to take out their collateral. Effectively you freeze their funds. If you don't push price data that's outside the price limit, you force the protocol to work with outdated price data which opens up arbitrage opportunities that will break the protocol. Either way, you have a security issue here.

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

Adityaxrex

This issue is valid and accepted. We will need to find a reasonable solution for this. Thank you

SunXrex

1. Again, the min and max prices are set to values that are not expected to be reached in the market.

2. I agree with this statement~ However, by doing so, it would be detected by our monitor, and attackers might not use a single transaction to drain the pool. `If an attacker is able to move the price even by 10% they are able to drain the protocol's collateral (move price up -> take loan -> move price down -> repay loan).`
3. Apologies for not disclosing too much about how we protect our Feeder EOA. We use Shamir's secret sharing, following an M of N scheme, across different containers.
4. Thank you for escalating this. It makes us pay more attention to this issue.
5. It should be Medium.

jacksanford1

Generally agree that:

1) Human cannot be relied upon for making this a non-issue
2) The max and min safeguards could create problems on their own (if min price is 95 cents and USDT is trading at 70 cents and people are able to use the 95 cent price for real transactions)
3) Stablecoins can be fairly stable, but they can be very unstable as well and there needs to be a plan for handling extreme instability

Will call this a Medium for now but I could see a case for a High (stablecoins depeg a lot depending on the magnitude of depeg needed to cause an issue). And will need more information from someone in order to downgrade it back to low/invalid.

hrishibhat

Result: Medium Has duplicates

sherlock-admin

Escalations have been resolved successfully!

Escalation status:

- 0xruhum: accepted

jacksanford1

Acknowledged by protocol team (won't fix).

Issue M-6: No clear threshold on when the oracle is updated will cause stale prices to be accepted

Source: https://github.com/sherlock-audit/2023-04-unitasprotocol-judging/issues/150

Found by

0xGoodess, 0xJuda, Juntao, Norah, PRAISE, PawelK, carrotsmuggler, ctf_sec, kutugu, mau, stopthecap, thekmj, toshii

Summary

No clear threshold on when the oracle is updated will cause stale prices to be accepted

Vulnerability Detail

According to the team, the oracle is updated depending on the gas fees.

"" the oracle price update is limited by cost of gas fee during updating. we want to have a balance between reaching the true price vs saving the cost. Once we move to cheaper chains like arbitrum or matic, we will update the oracle a lot more frequently just fyi ""

Not having a clear point in time when the oracle will be updated, will cause that in times when the network is very expensive to include transactions, stale prices of assets/stables will be accepted as the current price, causing wrong/stale prices be fetched as if they were the latest.

Impact

Wrong/stale prices will be used in times when gas fees are very expensive

Code Snippet8

https://github.com/sherlock-audit/2023-04-unitasprotocol/blob/d5328421bea80e3b0fd4595e4eb6b732a40e421e/Unitas-Protocol/src/XOracle.sol#L34-L46

Tool used

Manual Review

Recommendation

Add a clear threshold when the prices should be updated (ex. each 2 minutes), potentially not using the gas fees as an indicator to when updating the oracle.

Discussion

SunXrex

Fixed: https://github.com/xrex-inc/Unitas-Protocol/pull/9

0xffff11

Fix looks good.

The new implementation includes a default threshold of 1 day and a threshold according to each token that unitas uses, which can be changed with a specific setter function. If a threshold is not added for whatever reason or token, it will return the default 1 day threshold.

If the threshold is less than `type(uint32).max` , basically always, the stale price will be checked and validated:

```
if (threshold < type(uint32).max) {
            uint64 priceTimestamp = prices[asset].timestamp;
            _require(
                priceTimestamp > block.timestamp || block.timestamp -
↪  priceTimestamp <= threshold,
                Errors.PRICE_STALE
            );
```