SHERLOCK

SHERLOCK SECURITY REVIEW FOR

Prepared for:	Sense
Prepared by:	Sherlock
Lead Security Expert:	0x52
Dates Audited:	March 17 - March 23, 2023
Prepared on:	May 1, 2023

Introduction

Sense is decentralized permissionless infrastructure, where teams can build and develop new yield primitives for DeFi.

Scope

Repository: sense-finance/sense-v1

Branch: dev

Commit: 82abac25404d83b7aefaaeb46631f1d050dc4a4e

Repository: sense-finance/auto-roller

Branch: main

Commit: 60b8b4d56346f053becafb6a9f50f75cebafcafa

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	0

Issues not fixed or acknowledged

Medium	High
0	0

Security experts who found valid issues

spyrosonic10	0xAgro	Breeje
0x52	martin	Saeedalipoor01988
Bauer	sayan_	tsvetanovv

Issue M-1: Refund of protocol fee is being to wrong user

Source: https://github.com/sherlock-audit/2023-03-sense-judging/issues/40

Found by

spyrosonic10

Summary

There is one function, _fillQuote(), which is handling swap from `0x`. Ideally If there is any remaining protocol fee (in ETH) then it will be returned to sender aka msg.sender. There are scenarios when fee can be sent to receiver of swap instead.

Vulnerability Detail

Periphery and RollerPeriphery both are using almost identical logic in `_fillQuote()` hence this vulnerability affect both contracts. It exist if qupte.buyToken is ETH and there is any remaining protocol fee.

Here are pieces of puzzle

1. After swap if buyToken == ETH then store contract ETH balance in `boughtAmount`

```
// RollerPeriphery.sol
251:   boughtAmount = address(quote.buyToken) == ETH ? address(this).balance :
↪  quote.buyToken.balanceOf(address(this));
```

2. Next it store refundAmt

```
// RollerPeriphery.sol
257:        // Refund any unspent protocol fees (paid in ether) to the sender.
258:        uint256 refundAmt = address(this).balance;
```

3. Calculate actual refundAmt and transfer to sender

```
259:        if (address(quote.buyToken) == ETH) refundAmt = refundAmt -
↪  boughtAmount;
260:        payable(msg.sender).transfer(refundAmt);
```

4. This is clear that due to line 251, 258 and 259, refundAmt is 0. So sender is not getting refund.
5. Later on in logic flow buyToken will be transferred to receiver

```
        address(quote.buyToken) == ETH
                ? payable(receiver).transfer(amtOut)
                : ERC20(address(quote.buyToken)).safeTransfer(receiver,
↪  amtOut); // transfer bought tokens to receiver
```

Impact

Sender is not getting protocol fee refund.

Code Snippet

RollerPeriphery.sol#L251-L260

Periphery.sol#L921-L930

Tool used

Manual Review

Recommendation

Consider intercepting refund amount properly when buyToken is ETH or else just handle refund when buyToken is NOT ETH and write some explanation around it.

Discussion

jparklev

While it is valid that when `quote.buyToken` is ETH, there's an error. This line:

`if (address(quote.buyToken) == ETH) refundAmt = refundAmt - boughtAmount;`

Always returns 0.

However, since the `boughtAmount` is set with the Periphery's ETH balance, the user will receive not only the bought amount from 0x but also the refunded fees.

As a result, we don't think there is any value at risk. However, it's a great callout and we're open to judge's perspective on this

hrishibhat

Considering this issue as low as the `boughtAmount` is inclusive of the refunded fees.

spyrosonic10

Escalate for 10 USDC

Intention in code is to send refund fee back to `msg.sender` and send `boughtAmount`(NOT inclusive refund fee) to receiver(check point 5 from my report). So this is actually an issue if bounghtAmount is inclusive of refund fee. Different entities are supposed to get fee and boughtAmount. Assume by mistake a `msg.sender` send 1000x of fee, who should get remaining fee? Definitely not receiver.

For example:

- redeem() which take receiver as a param
- later in the logic in `fillQuote()` refundAmt/fee should be refunded to `msg.sender`
- msg.sender is different than recipient of boughtAmount
- as last step in redeem() (https://github.com/sherlock-audit/2023-03-sense/blob/main/auto-roller/src/RollerPeriphery.sol#L107-L113) boughtAmount aka amtOut is being send to receiver

So in the end if boughtAmount is inclusive of fee then that's a loss for msg.sender. So this issue deserve to be `medium`

sherlock-admin

> Escalate for 10 USDC
>
> Intention in code is to send refund fee back to `msg.sender` and send `boughtAmount`(NOT inclusive refund fee) to receiver(check point 5 from my report). So this is actually an issue if bounghtAmount is inclusive of refund fee. Different entities are supposed to get fee and boughtAmount. Assume by mistake a `msg.sender` send 1000x of fee, who should get remaining fee? Definitely not receiver.
>
> For example:
>
> - redeem() which take receiver as a param
> - later in the logic in `fillQuote()` refundAmt/fee should be refunded to `msg.sender`
> - msg.sender is different than recipient of boughtAmount
> - as last step in redeem() (https://github.com/sherlock-audit/2023-03-sense/blob/main/auto-roller/src/RollerPeriphery.sol#L107-L113) boughtAmount aka amtOut is being send to receiver
>
> So in the end if boughtAmount is inclusive of fee then that's a loss for msg.sender. So this issue deserve to be `medium`

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.

hrishibhat

Escalation accepted

Issue is a valid medium After looking into this further, agree with the escalation. Given that the refunded fee is being sent to the receiver instead of the msg.sender, this is a valid issue, considering this issue as a valid medium.

sherlock-admin

> Escalation accepted
>
> Issue is a valid medium After looking into this further, agree with the escalation. Given that the refunded fee is being sent to the receiver instead of the msg.sender, this is a valid issue, considering this issue as a valid medium.

This issue's escalations have been accepted!

Contestants' payouts and scores will be updated according to the changes made on this issue.

jparklev

Fixed here: https://github.com/sense-finance/sense-v1/pull/348

We removed the protocol fee refunds entirely, since 0x no longer uses them, which obviated the need another fix

IAm0x52

Fix looks good. Since protocol fees have been removed, the refund has also been remove. It seems an occurrence was missed here:

https://github.com/sherlock-audit/2023-03-sense/blob/52049213bdff31138c43a28f061fea493b3d7e14/auto-roller/src/RollerPeriphery.sol#L258-L260

MLON33

fedealconada added a fix to take care of the occurrence missed:

remove protocolFee refunds

https://github.com/sense-finance/auto-roller/pull/34

IAm0x52

Fix looks good. protocolFee refunds have been removed from RollerPeriphery

Issue M-2: sponsorSeries() method fails when user want to swap for stake token using

Source: https://github.com/sherlock-audit/2023-03-sense-judging/issues/36

Found by

spyrosonic10

Summary

`sponsorSeries()` fails when user want to use `swapQuote` to swap for stake token to sponsor a series.

Vulnerability Detail

stake is token that user need to deposit (technically is pulled) to be able to sponsor a series for a given target. User has option to send `SwapQuote calldata quote` and swap any ERC20 token for stake token. Below is the code that doing transferFrom() of stakeToken not sellToken()

```
if (address(quote.sellToken) != ETH) _transferFrom(permit, stake, stakeSize);
 if (address(quote.sellToken) != stake) _fillQuote(quote);
```

Expected behaviour of this function is to pull `sellToken` from msg.sender when `address(quote.sellToken) != stake`. For example- stake token is WETH. User want to swap DAI for WETH in `sponsorSeries()`. In this case, user would be sending SwapQuote.sellToken = DAI and swapQuote.buyToke = WETH and expect that fillQuote() would swap it for WETH. This method will fail because sellToken not transferred from msg.sender.

Impact

sponsorSeries() fails when `address(quote.sellToken) != stake`

Code Snippet

https://github.com/sherlock-audit/2023-03-sense/blob/main/sense-v1/pkg/core/src/Periphery.sol#L116-L128

Tool used

Manual Review

Recommendation

Consider implementation of functionality to transferFrom `sellToken` from msg.sender with actual amount that is require to get exact amountOut greater or equal to `stakeSize`

Discussion

jparklev

Accepted:

This bug is valid but the below statement

> `sponsorSeries()` fails when user want to use `swapQuote` to swap for stake token to sponsor a series.

is not quite accurate.

The problem here is that here:

```
if (address(quote.sellToken) != ETH) _transferFrom(permit, stake, stakeSize);
```

we are sending wrong params to `_transferFrom`.

If we are making use of the `permit` feature, this would work fine because the `_transferFrom` **ignores** the params on that case.

On the contrary, if we want to make use of the traditional approval, this would revert since we will be trying to pull a the `stake` which has not been approved by the user.

Fix:

```
if (address(quote.sellToken) != ETH) _transferFrom(permit, quote.sellToken,
↪   quote.amount);
// quote.amount does not exist so we may need to add this param to the struct
```

jparklev

Fixed here: https://github.com/sense-finance/sense-v1/pull/347

We used the fix mentioned above

IAm0x52

Fix looks good. _transferFrom now correctly pulls the sellToken instead of stake

Issue M-3: fillQuote uses transfer instead of call which can break with future updates to gas costs

Source: https://github.com/sherlock-audit/2023-03-sense-judging/issues/33

Found by

0x52, 0xAgro, Bauer, Breeje, Saeedalipoor01988, martin, sayan_, tsvetanovv

Summary

Transfer will always send ETH with a 2300 gas. This can be problematic for interacting smart contracts if gas cost change because their interaction may abruptly break.

Vulnerability Detail

See summary.

Impact

Changing gas costs may break integrations in the future

Code Snippet

https://github.com/sherlock-audit/2023-03-sense/blob/main/sense-v1/pkg/core/src/Periphery.sol#L902-L932

Tool used

Manual Review

Recommendation

Use call instead of transfer. Reentrancy isn't a concern since the contract should only ever contain the callers funds.

Discussion

jparklev

Accepted: we should use .call instead of transfer when transferring ETH, specifically if the receiver is a contract that is integrating Sense.

jparklev

Fixed here: https://github.com/sense-finance/sense-v1/pull/348

We ended up removing the protocol fees entirely, which obviated the need for the suggested fix here

IAm0x52

Fix looks good, transfer has ben changed to call but the occurrence is rollerPeriphery appears to have been missed:

https://github.com/sherlock-audit/2023-03-sense/blob/52049213bdff31138c43a28f061fea493b3d7e14/auto-roller/src/RollerPeriphery.sol#L111

IAm0x52

Additional missed transfers:

https://github.com/sense-finance/sense-v1/pull/346/files#r1167351042
https://github.com/sense-finance/auto-roller/pull/32#discussion_r1167351433

fedealconada

> Fix looks good, transfer has ben changed to call but the occurrence is rollerPeriphery appears to have been missed:
>
> https://github.com/sherlock-audit/2023-03-sense/blob/52049213bdff31138c43a28f061fea493b3d7e14/auto-roller/src/RollerPeriphery.sol#L111

fixed here

fedealconada

> Additional missed transfers:
>
> https://github.com/sense-finance/sense-v1/pull/346/files#r1167351042
> [sense-finance/auto-roller#32 (comment)]

fixed here

> (https://github.com/sense-finance/auto-roller/pull/32#discussion_r1167351433)

fixed here

IAm0x52

Fixes look good. All three occurrences have been address. 2 occurrences in RollerPeriphery addressed in PR#32 and the other in Periphery addressed in PR#346

Issue M-4: Periphery#_swapPTsForTarget won't work correctly if PT is mature but redeem is restricted

Source: https://github.com/sherlock-audit/2023-03-sense-judging/issues/32

Found by

0x52

Summary

Periphery#_swapPTsForTarget doesn't properly account for mature PTs that have their redemption restricted

Vulnerability Detail

Periphery.sol#L531-L551

```
function _swapPTsForTarget(
    address adapter,
    uint256 maturity,
    uint256 ptBal,
    PermitData calldata permit
) internal returns (uint256 tBal) {
    _transferFrom(permit, divider.pt(adapter, maturity), ptBal);

    if (divider.mscale(adapter, maturity) > 0) {
        tBal = divider.redeem(adapter, maturity, ptBal); <- @audit-issue always
         ↪  tries to redeem even if restricted
    } else {
        tBal = _balancerSwap(
            divider.pt(adapter, maturity),
            Adapter(adapter).target(),
            ptBal,
            BalancerPool(spaceFactory.pools(adapter, maturity)).getPoolId(),
            0,
            payable(address(this))
        );
    }
}
```

Adapters can have their redeem restricted meaning the even when they are mature they can't be redeemed. In the scenario that it is restricted Periphery#_swapPTsForTarget simply won't work.

Impact

Redemption will fail when redeem is restricted because it tries to redeem instead of swapping

Code Snippet

Tool used

Manual Review

Recommendation

Use the same structure as _removeLiquidity:

```
if (divider.mscale(adapter, maturity) > 0) {
    if (uint256(Adapter(adapter).level()).redeemRestricted()) {
        ptBal = _ptBal;
    } else {
        // 2. Redeem PTs for Target
        tBal += divider.redeem(adapter, maturity, _ptBal);
    }
```

Discussion

jparklev

Accepted: This is valid and is indeed something we should fix

jparklev

Fixed here: https://github.com/sense-finance/sense-v1/pull/352

We removed the `redeem` code path if redeem is disabled

IAm0x52

Fix looks good. Periphery will swap instead of redeeming if redeem is restricted

Issue M-5: Multiple functions may leave excess funds in the contract that should be returned

Source: https://github.com/sherlock-audit/2023-03-sense-judging/issues/29

Found by

0x52, Bauer, spyrosonic10

Summary

Periphery#combine may leave excess underlying in the contract due to _fromTarget unwrapping to underlying and the quote may not swap them all.

When using arbitrary tokens to swap to underlying the contract always moves in the full amount specified. There is no guarantee that the quote will consume all tokens. As a result the contract may leave excess sell tokens in the contract but it should return then to the receiver.

These functions include:

RollerPeriphery

1) deposit

Periphery

1) swapForPTs
2) addLiquidity
3) issue

RollerPeriphery#RollermintFromUnderlying uses adapter.scale and previewMint to determine the amount of underlying to transfer. The roller code will mean that previewMint will always perfectly reflect the exact exchange rate into the roller. However adapter.scale varies by adapter and isn't guaranteed to be exact. The result is that _transferFrom may take too much underlying. Since this underlying is wrapped to target the contract should return all excess target to receiver.

Vulnerability Detail

See summary.

Impact

Token may be left in the contract and lost

Code Snippet

https://github.com/sherlock-audit/2023-03-sense/blob/main/auto-roller/src/RollerPeriphery.sol#L175-L186

https://github.com/sherlock-audit/2023-03-sense/blob/main/auto-roller/src/RollerPeriphery.sol#L196

https://github.com/sherlock-audit/2023-03-sense/blob/main/sense-v1/pkg/core/src/Periphery.sol#L178

https://github.com/sherlock-audit/2023-03-sense/blob/main/sense-v1/pkg/core/src/Periphery.sol#L325

https://github.com/sherlock-audit/2023-03-sense/blob/main/sense-v1/pkg/core/src/Periphery.sol#L409

Tool used

Manual Review

Recommendation

Return excess tokens at the end of the function

Discussion

jparklev

We have this feature, for example, on `_swapSenseToken`, but not on the cases mentioned.

Our fix will be: Transfer non-used tokens back to the user.

jparklev

Fixed here for sense-v1: https://github.com/sense-finance/sense-v1/pull/346 And here for the auto-roller: https://github.com/sense-finance/auto-roller/pull/32

We returned excess `sellTokens` funds in the from and to target functions

IAm0x52

Fixes look good. Excess sell tokens are now returned inside _toTarget and _fromTarget for both periphery and rollerPeriphery

Issue M-6: Multiple functions aren't payable so quotes that require protocol fees won't work correctly

Source: https://github.com/sherlock-audit/2023-03-sense-judging/issues/28

Found by

0x52, Bauer

Summary

There are multiple functions that use quotes but that aren't payable. This breaks their compatibility with some quotes. As the 0x docs state: `Certain quotes require a protocol fee, in ETH, to be attached to the swap call.`

The following flows use a quote but the external/public starting function isn't payable:

RollerPeriphery

1) redeem

Periphery

1) removeLiquidity
2) combine
3) swapPT
4) swapYT
5) issue

Vulnerability Detail

See summary.

Impact

Functions won't be compatible with certain quotes causing wasted gas fees or bad rates for users

Code Snippet

https://github.com/sherlock-audit/2023-03-sense/blob/main/auto-roller/src/RollerPeriphery.sol#L104 https://github.com/sherlock-audit/2023-03-sense/blob/main/sense-v1/pkg/core/src/Periphery.sol#L325

https://github.com/sherlock-audit/2023-03-sense/blob/main/sense-v1/pkg/core/src/Periphery.sol#L409

https://github.com/sherlock-audit/2023-03-sense/blob/main/sense-v1/pkg/core/src/Periphery.sol#L433

https://github.com/sherlock-audit/2023-03-sense/blob/main/sense-v1/pkg/core/src/Periphery.sol#L240

https://github.com/sherlock-audit/2023-03-sense/blob/main/sense-v1/pkg/core/src/Periphery.sol#L263

Tool used

Manual Review

Recommendation

Add payable to these external/public functions

Discussion

jparklev

Confirmed: We've forgotten to add payable to the functions mentioned

jparklev

Fixed here for sense-v1: https://github.com/sense-finance/sense-v1/pull/345 And here for the auto-roller: https://github.com/sense-finance/auto-roller/pull/33

We took the suggested fix and added payable to the mentioned functions

IAm0x52

Fixes look good. Payable has been added to Periphery#removeLiquidity, combine, swapPT, swapYT and issue. Also adds payable to RollerPeriphery#redeem