Introduction & Scope

This audit looks into the contract `PirexGmxDepositRouter.sol` in commit hash 7020431 as seen here and the diff of `AutoPxGmx` between the commit hash 570385b and a7514fa.

This audit was conducted by kebabsec members sai, FlameHorizon, okkothejawa and shung.

Note: This report does not provide any guarantee or warranty of security for the project.

Executive Summary

Table of Contents

Findings:

1. [MEDIUM] Camelot pair's fee percent may change

Description: The function `_getAmountIn` assumes that the fee percent of the WETH/pxGMX Camelot pool is 0.3% by hard-coding it.

```
uint256 numerator = reserveIn * amountOut * 1000;
        uint256 denominator = (reserveOut - amountOut) * 997;
        uint256 amountIn = (numerator / denominator) + 1;

        return amountIn;
```

Yet the Camelot pair has a `setFeePercent` function which can change the fee percent of the pair and rendering the fee calculation done in `_getAmountIn` invalid.

```
  function setFeePercent(uint16 newToken0FeePercent, uint16 newToken1FeePercent) external lock {
    require(msg.sender == ICamelotFactory(factory).feePercentOwner(), "CamelotPair: only factory's feeAmountOwner");
    require(newToken0FeePercent <= MAX_FEE_PERCENT && newToken1FeePercent <= MAX_FEE_PERCENT, "CamelotPair: feePercent mustn't excee
    require(newToken0FeePercent > 0 && newToken1FeePercent > 0, "CamelotPair: feePercent mustn't exceed the minimum");
    token0FeePercent = newToken0FeePercent;
    token1FeePercent = newToken1FeePercent;
    emit FeePercentUpdated(newToken0FeePercent, newToken1FeePercent);
  }
```

Such an invalid calculation may result in getting a revert in the try block as the calculated minimum amount out expected from the swap would exceed the actual possible minimum amount out in the case of the fee percent of the pair being larger than 0.3%.

Recommendation: Do not hard-code the fee percent and either read it directly from the pair to make the calculation, or have a permissioned setter for the fee percent.

2. [MEDIUM] Setting `_platform` through `setPlatform` doesn't update approvals, causing DOS for `compound` of `AutoPxGmx`

Description: The commit AutoPxGmx.sol introduced a method to set depositor router address, initially intended to route depositing through `PirexGmxDepositRouter`. However after the changes, `setPlatform` no longer updates approvals for the newly set platforms:

```
    function setPlatform(address _platform) external onlyOwner {
        if (_platform == address(0)) revert ZeroAddress();

-         // Update GMX transfer allowance for the old and new platforms
-         gmx.safeApprove(platform, 0);
-         gmx.safeApprove(_platform, type(uint256).max);

        platform = _platform;

        emit PlatformUpdated(_platform);
    }

+    function SetDepositRouter(address _router) external onlyOwner {
+        if (_router == address(0)) revert ZeroAddress();
+
+        // Update GMX transfer allowance for the old and new platforms
+        gmx.safeApprove(depositRouter, 0);
+        gmx.safeApprove(_router, type(uint256).max);
+
+        depositRouter = _router;
+    }
```

Essentially breaking `compound` of AutoPxGmx.sol as it deposits GMX into PirexGmx platform directly here. As the platforms that are set after the initial one are not approved of GMX anymore, the following code path will cause revert for `compound` and cause denial of service requiring redeployment to fix.

```
(pxGmxAmountOut, ) = PirexGmx(platform).depositGmx(
                    gmx.balanceOf(address(this)),
                    address(this)
                );
```

In addition, `PirexGmxDepositRouter` is set only at construction and will require a new deployment.

Recommendation: Revert `setPlatform` back so that it approves the new platform for GMX.

3. [LOW] `depositGMX` does not have a deadline and slippage check

Description: An EOA directly calling `PirexGmxDepositRouter.depositGmx` has no option to provide deadline and slippage checks. Lack of slippage check can be an issue if a user considers the limit provided by direct depositing to be too low, and wants to ensure they swap at a more profitable rate than directly depositing.

Recommendation: Add slippage and deadline checks.

4. [INFO] Typographical errors

Description: In line 36 of AutoPxGmx.sol, the address variable for camelot is currently named calemotReferral this is also the case in the comment above that same line, the comment also spels the word "referral" wrong, as it's currently spelt "refferal". This is incorrect as the name of the exchange is Camelot.

In line 211 of AutoPxGmx.sol the name of the function does not follow the camel case standard of naming, as the first letter of SetDepositRouter is capitalized.

Recommendation: Fix the typos.

5. [INFO] Do not use magic numbers

Description: The following lines uses magic numbers:

AutoPxGmx

PirexGmxDepositRouter

Recommendation: Use constant variable instead of `1_000_000_`.

6. [INFO] Unused error

Description: The following error `AlreadySet` is unused.

Recommendation: Remove unused error if not needed.

7. [MED] `compound` lacks access control

Description: `AutoPxGmx.compound` does not have an access control. This can be abused by a malicious user providing a low `amountOutMinimum` value and atomically sandwiching the swap.

Recommendation: Consider limiting who can call `compound` to trusted users only.