arbitrary./execution

PROTOCOL SECURITY ASSESSMENT

March 9, 2022

Prepared For:

Gamma Strategies

Prepared By:

Chris Masden

Jasper Clark

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Gamma Strategies hired Arbitrary Execution to perform a code security audit of the Gamma Strategies Hypervisor. This report contains the results of the assessment of the smart contracts that comprise the Gamma Strategies Hypervisor, which acts as a non-custodial, automated, concentrated liquidity manager.

Two Arbitrary Execution engineers conducted this review over a 2-week period, from January 14, 2022 to January 28, 2022. The audited commit was `1c0ebc1adcfa3c92c42d8db1fa2c2f697c3b752c` in the `stable` branch of the `GammaStrategies/hypervisor` repository. The Solidity files in scope for this audit included all contracts in the `contracts` directory with the exception of the mocks and test contracts. The complete list of files is in Appendix A.

There are tests which execute some of the basic functionality of the project. However, many areas do not have test coverage. It would be beneficial to write additional unit tests around each function present in the contracts, including administrative functions. This would help catch any unexpected behavior and would help prevent regressions in future code changes. There is also a lack of function documentation for most of the functions in the contracts. We recommend inserting comments for every function in the NatSpec format.

Update

Of all the issues that were reported, 16 were resolved and 8 were partially resolved, and 1 non-security issue was left unresolved. Unrelated changes introduced during the fix process were disregarded.

VULNERABILITY STATISTICS

TOTAL ISSUES BY SEVERITY

Critical (C)	0
High (H)	3
Medium (M)	2
Low (L)	7
Informational Notes (N)	13
Undetermined (U)	0
Total	25

FINDINGS

Our findings appear below, in order of importance.

HIGH SEVERITY

[H1] DANGEROUS USE OF SHADOWED VARIABLES

In `Hypervisor.sol` and `admin.sol` the `pullLiquidity` function, four `uint256` variables (`base0`, `base1`, `limit0`, and `limit1`) are declared in the returns function prototype. Then, in the function body, four `uint256` variables are declared with the same name. These shadow the existing declaration and will cause the function to always return four zeros (the default initialized value).

In `Hypervisor.sol` the withdraw function has a `uint256` variable called `totalSupply` that shadows the `totalSupply` public function of the contract. Once the `uint256` variable is declared, the `totalSupply` function is unable to be called until the `uint256` variable goes out of scope.

In `UniProxy.sol` the `checkPriceChange` function has a `uint256` price variable that is declared inside of the function prototype. Then, in the function body, another `uint256` variable is declared with the same name. This shadows the existing declaration and will cause the function to always return a zero (the default initialized value).

Recommendation

Remove the `uint256` variable declarations in the function body for variables that are already defined in the function prototype and rename the `totalSupply` variable.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. Variable declarations were kept in the function prototype and removed in the function bodies.

[H2] DANGEROUS USE OF USER-SUPPLIED FROM ADDRESS

The `from` address in the `deposit` function in `Hypervisor.sol` can be set to any arbitrary address by a user calling the function. There is a whitelist check, however it checks to see if the `from` address is in the whitelist, not `msg.sender`. The whitelist check is also only in effect if the whitelist is enabled. This behavior can be exploited as a malicious user could send tokens from one account to the Hypervisor (if there is a prior allowance set for the address to the Hypervisor). The malicious user could then send the newly minted LP tokens to an arbitrary address specified by the caller.

Recommendation

Remove the user-specified `from` address and use `msg.sender` instead. Use OpenZeppelin's role-based access control functions (https://docs.openzeppelin.com/contracts/4.x/access-control). A "depositor" role could then be created on the `Hypervisor.sol` contract and granted to the deployed `UniProxy.sol` contract.

Update

Partially resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The `deposit` function in the `Uniproxy` contract was changed to remove the `from` address parameter and now uses `msg.sender` as the `from` address parameter when calling the `deposit` function on the `Hypervisor` contract. The Gamma team reported the following in relation to this issue:

> *We found that introducing OpenZeppelin's `AccessControl` contract pushed `Hypervisor.sol` over the deployment kb limit. We will maintain an operational policy of only exposing `Hypervisor.deposit` through `UniProxy.sol` by means of its whitelist array.*

As long as the whitelist is enabled on the `Hypervisor` contract, this addresses the security concern.

[H3] IMPOSSIBLE TO OVERRIDE TWAPINTERVAL

The `setTwapOverride` function inside of `UniProxy.sol` is used to set the `boolean twapOverride` and the `uint32 twapInterval` for a specified position. The function uses the global `twapInterval` to set the `twapInterval` on a position instead of the `_twapInterval` function parameter. This makes it impossible to set the `twapInterval` for a position due to the incorrect variable being used.

Recommendation

Change the function body so that when the `twapInterval` for a position is being assigned, it uses the user specified `_twapInterval` variable.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The `_twapInterval` function parameter is now used to set the `twapInterval` of a given position.

MEDIUM SEVERITY

[M1] UNABLE TO REMOVE WHITELISTED ADDRESSES

`Hypervisor.sol` uses a whitelist to determine who is allowed to use the `deposit` function on the contract. The `removeListed` function removes addresses from the whitelist, however this is unable to be called because it's protected with the `onlyOwner` modifier, and the owner of the contract is `admin.sol`. The `Admin` contract lacks the ability to call `removeListed` on the hypervisor. This could lead to an address being added to the whitelist that is then later compromised by an attacker. The owner of the `Hypervisor` contract would be unable to remove that compromised address from the whitelist.

Recommendation

If the whitelist is removed and OpenZeppelin's role-based access control functions are used, the `removeListed` function can be removed. See the [L1] ACCESS CONTROL CHANGES issue for more details.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. A `removeListed` function was added to `admin.sol` which allows the `removeListed` function to be called on the `Hypervisor` contract.

[M2] ERROR MIGRATING FUNDS FROM OLD HYPERVISOR

The `migrate` function in `HypervisorV3Migrator.sol` cannot be executed as intended due to an external call on the `getDepositAmount` function in `UniProxy.sol`. The `migrate` function attempts to call `getDepositAmount` with two function parameters. This is incompatible with `UniProxy.sol` as it requires three function parameters. The result of a user calling this when attempting to migrate to the V3 Hypervisor will be a reverted transaction and the user will lose the gas cost.

Recommendation

Supply the additional address argument to the `getDepositAmount` function.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The `getDepositAmount` function is now supplied with the correct number of arguments.

LOW SEVERITY

[L1] ACCESS CONTROL CHANGES

In `Hypervisor.sol` the `deposit` function uses the `list` storage variable as a whitelist of addresses that are permitted to execute the function. After communicating with the Gamma team, the intent was stated that only one address should be allowed to call the `deposit` function: The address of the deployed `UniProxy.sol` contract.

In `admin.sol` there are two custom modifiers, `onlyAdvisor` and `onlyOwner`.

In `Swap.sol` there is a custom `onlyOwner` modifier.

In `UniProxy.sol` there is a custom `onlyOwner` modifier.

Recommendation

Use OpenZeppelin's role-based access control functions (https://docs.openzeppelin.com/contracts/4.x/access-control). A "depositor" role could then be created on the `Hypervisor.sol` contract and granted to the deployed UniProxy.sol contract. Another role, the "DEFAULT_ADMIN_ROLE", could also be used for administrative purposes and the custom `onlyOwner` modifier could be removed from the `Hypervisor.sol` contract. An "advisor" role could be created on the `admin.sol` contract and the `onlyAdvisor` and `onlyOwner` modifiers could then be removed. The `Swap.sol` and `UniProxy.sol` contract could inherit from OpenZeppelin's Ownable contract and the `onlyOwner` modifier could be removed. Using existing and well vetted flexible access control libraries is best practice.

Update

Partially resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The Gamma team reported the following in relation to this issue:

> *We implemented access control recommendations for contracts outside of `Hypervisor.sol`. We found that introducing OpenZeppelin's `AccessControl` contract pushed `Hypervisor.sol` over the deployment kb limit.*

The Gamma team was unable to use OpenZeppelin's access control contracts in `Hypervisor.sol` but they were able to implement them in the other noted contracts. The `Hypervisor` contract uses a list storage variable and an `onlyOwner` modifier to restrict access to certain functions and is adequate as long as the `whitelisted` storage variable is set to `true`.

[L2] INTERFACE DOES NOT MATCH CONTRACT

The `getDepositAmount` function in `UniProxy.sol` takes three parameter types: `address`, `address`, and `uint256`. The interface for this contract, `IUniProxy.sol`, has a `getDepositAmount` function that takes two parameter types: `address` and `uint256`.

Recommendation

Add in the missing address parameter to the `getDepositAmount` function inside of the `IUniProxy.sol` contract.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The `getDepositAmount` function now has the correct number of function parameters required to interface with the `UniProxy` contract.

[L3] POSSIBLE REENTRANCY

In the `addPosition` function in `UniProxy.sol` there are 2 external calls to contracts before the storage variable `p` is written. This could lead to a new position being saved to a storage slot multiple times.

In the rebalance function in `Hypervisor.sol`, there are multiple external calls before state is written. This could lead to unintended behavior. In the `withdraw` function in `Hypervisor.sol` there are multiple external calls before state is written.

Recommendation

In the `addPosition` function, make the storage variable modification before the external calls to `approve` to follow the checks-effects-interactions pattern. In the rebalance and withdraw functions, make these reentrant safe by using the OpenZeppelin reentrant guard library (https://docs.openzeppelin.com/contracts/4.x/api/security#ReentrancyGuard).

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The `nonReentrant` function modifier was added to the `rebalance` and `withdraw` functions in `Hypervisor.sol`. The `addPosition` function in `UniProxy.sol` now follows the checks-effects-interactions pattern.

[L4] ADD CONFIG EVENTS

The Gamma contracts have various configuration functions to change the owner, advisor, max supply, etc. These functions do not emit events when called and thus it is possible for them to be changed without easily being noticed. Adding events to major configuration changes can help in the administration of the contracts.

Recommendation

We recommend that any major configuration changes emit an associated event. Specifically, the following functions should emit an event:

In `Hypervisor.sol`:

- `setMaxTotalSupply`
- `setDepositMax`

In `UniProxy.sol`:

- `setPriceThreshold`
- `setDepositDelta`
- `setDeltaScale`
- `setTwapInterval`
- `setTwapOverride`
- `toggleDepositFree`
- `toggleDepositFreeOverride`
- `toggleTwap`
- `appendList`
- `removeListed`
- `addPosition`
- `customDeposit`

If the OpenZeppelin `Ownable` contract is used, no event needs to be written for the transfer of ownership as the `Ownable` contract already does this.

Update

Partially resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The following events were added:

In `Hypervisor.sol`:

- `MaxTotalSupplySet` in `setMaxTotalSupply`
- `DepositMaxSet` in `setDepositMax`

In `UniProxy.sol`:

- `PriceThresholdSet` in `setPriceThreshold`
- `DepositDeltaSet` in `setDepositDelta`

- `DeltaScaleSet` in `setDeltaScale`
- `TwapIntervalSet` in `setTwapInterval`
- `TwapOverrideSet` in `setTwapOverride`
- `DepositOverrideToggled` in `toggleDepositFree`
- `DepositFreeOverrideToggled` in `toggleDepositFreeOverride`
- `TwapToggled` in `toggleTwap`
- `ListAppended` in `appendList`
- `ListRemoved` in `removeListed`
- `PositionAdded` in `addPosition`
- `CustomDeposit` in `customDeposit`

Because the `Hypervisor` contract was not modified to inherit from OpenZeppelin's `Ownable` contract, we still recommend adding an event to signal transfer of ownership of the contract.

[L5] UNASSIGNED RETURN VARIABLE

The function `getDepositAmount` in `UniProxy.sol` declares two `uint256` variables in the returns statement, `amountStart` and `amountEnd`. These two variables are not used in the function body.

The function `pendingFees` in `admin.sol` declares two `uint256` variables in the returns statement: `fees0` and `fees1`. These two variables are not used in the function body. Any transactions that rely on the return values from this function will not operate correctly as the return values are invalid.

Recommendation

Either assign values to the declared return values in the function body or modify the `returns` statement to only specify the two `uint256` types.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The `amountStart` and `amountEnd` variables are now assigned in the `getDepositAmount` function. The `pendingFees` function was removed.

[L6] UPDATE OLD DEPENDENCIES

In `package.json` many libraries specify old versions, for example `@openzepplin/contracts targets 3.4.1-solc-0.7.2` which was released on March 3rd, 2021.

In `package.json` the Uniswap libraries specify a version wildcard which could introduce bugs and/or unknown behavior if a new version is released that contains breaking changes.

Recommendation

Pin all dependencies to the latest trusted versions. Pin to the specific version of Uniswap libraries used for testing.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. Packages in `package.json` are now pinned. The `@openzeppelin/contracts package` was not updated to latest as the project uses the 0.7.6 solidity compiler because of the reliance on the `@uniswap` packages.

[L7] CHANGE THE WAY PRAGMA VERSION STATEMENTS ARE SPECIFIED

The pragma version specifiers in Gamma contracts aren't consistent; the `UniProxy` contract is missing a pragma statement, and other Gamma contracts target 0.7.6 which is an older version. Locking the compiler version prevents accidentally deploying the contracts with an older Solidity version that lacks bug fixes or behaves differently than the version used for testing.

Recommendation

Lock each contract to the latest trusted version (0.8.4). We also recommend using a consistent method for specifying the version, such as "`pragma 0.8.4`". This will provide built-in access to safe math features and allow the removal of the `_uint128Safe` function.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The choice was made to keep using the 0.7.6 solidity compiler due to its reliance on Uniswap packages. The Gamma team stated the following:

> *In regards to Solidity 0.8.4, we implemented the recommended update to our packages and contracts in a development branch, however uniswap-v3-core `TickMath.sol`, `FullMath.sol` used in `UniswapV3Pool` is incompatible with >= 0.8 (https://github.com/Uniswap/v3-core/issues/489) and we would like to avoid modification of these dependencies so we have kept to 0.7.6 (Uniswap's).*

All of the pragma versions are now consistent across all the Solidity files and we agree that continuing to use 0.7.6 to enable use of Uniswap libraries is a good approach.

INFORMATIONAL NOTES

[N1] LACK OF NATSPEC FORMATTED COMMENTS

Incomplete documentation of input and output parameters makes the code more difficult to follow and could lead to errors if developers and users fill in their own assumptions in the absence of information.

Recommendation

Ensure that every function follows NatSpec format and includes the appropriate `@param` and `@return` tags, without exception.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The comments added greatly increase the readability of the code and make understanding the purpose of specific functions much easier.

[N2] UNUSED VISIBILITY ON CONSTRUCTOR

The constructor in `admin.sol` explicitly sets the visibility of the constructor to public. With the release of solc 0.7.0, this is no longer needed and is obsolete.

Recommendation

Remove the public keyword from the constructor.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The public visibility was removed from the constructor in `admin.sol`.

[N3] UNCHECKED ADDRESS PARAMETERS IN FUNCTIONS

The constructor in `Hypervisor.sol` takes the `pool` and `owner` addresses as input parameters and derives the `token0` and `token1` state variables from the pool address.

The function `transferOwnership` in `Hypervisor`.sol takes a new owner address as an input parameter.

The function rebalance in `Hypervisor.sol` takes the `feeRecipient` address as an input parameter.

The constructor in `HypervisorFactory.sol` takes the `uniswapV3Factory` address as an input parameter.

The constructor in `HypervisorV3Migrator.sol` takes the `uniswapV2Factory` and `uniProxy` addresses as input parameters.

The constructor in `Swap.sol` takes the `owner`, `router`, and `VISR` addresses as input parameters.

The function `changeRecipient` in `Swap.sol` takes a new recipient address as an input parameter.

The function `transferOwnership` in `Swap.sol` takes a new owner address as an input parameter.

The constructor in `admin.sol` takes the `owner` and `advisor` addresses as input parameters.

The function `transferAdmin` in `admin.sol` takes the `newAdmin` as an input parameter.

The function `transferAdvisor` in `admin.sol` takes the `newAdvisor` as an input parameter.

The function `rescueERC20` in `admin.sol` takes the `recipient` as an input parameter.

The function `transferOwnership` in `UniProxy.sol` takes a new owner address as an input parameter.

The function deposit in `UniProxy.sol` takes the `to` address as an input parameter.

Recommendation

Add `require` statements as needed within these functions to ensure that any address variables being used are non-zero.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. All of the recommendations were followed by adding `require` statements that check address parameters against 0.

[N4] POTENTIAL LOSS OF PRECISION

In `UniProxy.sol` the `getDepositAmount` function has several instances where a divide occurs before a multiply, which can potentially result in a loss of precision.

Recommendation

Use the FullMath library to avoid the potential loss of precision (https://docs.uniswap.org/protocol/reference/core/libraries/FullMath).

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The `FullMath` library is now used for all mathematical operations in the `getDepositAmount` function.

[N5] GAS SAVINGS

In `UniProxy.sol` the `getDepositAmount` function is not used internally to the contract and can be marked external.

In `UniProxy.sol` the `MAX_INT` variable is never changed and can be marked as a constant.

In `Hypervisor.sol` the `setDepositMax` function can potentially reassign an unchanged value.

In `Hypervisor.sol` the `deposit` function executes the public function `totalSupply` three times.

In `UniProxy.sol` the `positions` mapping is accessed directly in many places rather than via a local storage variable.

Recommendation

Use external when possible to save gas.

Mark variables as constant when possible to save gas.

Check before costly variable assignments, such as in the `setDepositMax` function, when possible to save gas.

Access a mapping via a local storage variable when possible to save gas. We also recommend using a consistent style for accessing storage mappings to improve code clarity.

Save the result of the `totalSupply` function to a local variable. This would exceed the max number of local variables, however, using curly braces to denote a block scope will avoid the issue.

Either the `deposit0PricedInToken1` or the `pool0PricedInToken1` variables could be placed into their own block scope.

Update

Partially resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. All of the recommendations were following with the exception of making `token0, token1, fee, tickSpacing, uniswapV3Factory, uniswapV2Factory, uniProxy, GAMMA,` and `router` storage variables `immutable`.

[N6] CHECK BALANCES AGAINST ZERO

The `addBaseLiquidity` and `addLimitLiquidity` functions in `Hypervisor.sol` perform an external call to a uniswap contract to determine how much liquidity will be returned in exchange for the two tokens in the pool. It is possible to specify an amount for both tokens which the caller does not have. This will cause a reverted transaction.

Recommendation

Save the balance of `token0` and `token1` as local variables and add in a `require` check to ensure the amount specified in the function parameter is less than or equal to the amount of tokens the contract currently holds. The local variables can then be used in the ternary statement to help reduce gas costs.

Update

This issue was not fixed. The Gamma team reported the following in relation to this issue:

> *This introduction pushed `Hypervisor.sol` over its deployment kb limit.*
> *We will maintain an operational policy of owner account checking these parameters before calling the function.*

As this is not a security issue, this mitigation is sufficient.

[N7] POSITION EXISTS MODIFIER

The `UniProxy.sol` deposit function has a require check that enforces the existence of a position. This will prevent potential misconfiguration issues that could occur by bypassing the `addPosition` function and setting configuration on addresses that have yet to be added as valid positions.

Recommendation

We recommend creating a modifier that could be used to enforce this check on all public or external functions that use the `positions` storage variable.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The recommended fix was followed and an `onlyAddedPosition` modifier was added to the following functions:

- `deposit`
- `customDeposit`
- `toggleDepositOverride`
- `toggleDepositFreeOverride`
- `setTwapOverride`
- `appendList`
- `removeListed`

[N8] CHANGE THE WAY MAXTOTALSUPPLY INDICATES A NO CAPACITY LIMIT

In `Hypervisor.sol` the `maxTotalSupply` storage variable is used to indicate a capacity limit. Setting `maxTotalSupply` to 0 indicates that there is no capacity limit in effect.

Recommendation

Set the max unsigned integer value to indicate there is no capacity limit in effect. This will simplify the require statement on line 138 and reduce the gas cost of the `deposit` function. Having 0 indicate no capacity limit may lead to unintended behavior on external contracts that do not take the special case of 0 into effect.

Update

Partially resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The `maxTotalSupply` variable now defaults to `type(uint256).max`, but the `require` statement at the end of the `deposit` function still treats 0 as meaning no capacity limit.

[N9] IMPROVE CODE CLARITY

In `Hypervisor.sol` variables use `uint256(-1)` to set a maximum value.

In `UniProxy.sol` a variable uses `2**256-1` to set a maximum value.

In `Hypervisor.sol` the `rebalance` function duplicates code from the `pendingFees` function.

In `Hypervisor.sol` the `deposit` and `rebalance` functions duplicate code from `addBaseLiquidity` and `addLimitLiquidity`.

In `HypervisorV3Migrator.sol` there is an unnecessary double casting of an address.

In `HypervisorFactory.sol` there is an extra newline after a function declaration.

In `IHypervisor.sol` there is an extra space in the returns portion of the function parameter.

In `IHypervisor.sol` there is an extra newline after the interface declaration.

In `IUniProxy.sol` there is an extra newline after the interface declaration.

In `HypervisorFactory.sol` the import statements have an inconsistent style compared to the existing import statements.

In `Swap.sol` there are variables and events that reference an old ERC20 token (VISR).

In `IHypervisor.sol` standard ERC20 functions are declared that could be inherited from the existing IERC20 interfaces.

Recommendation

Use `type(uint256).max` to set the max value for `uint256` types.

Remove duplicate code.

Avoid double casting values.

Remove extra newlines.

Remove extra spaces.

Update the import statements in `HypervisorFactory.sol` to match the style used elsewhere.

Update any references to the existing VISR token to mention the new GAMMA token instead.

Update the `IHypervisor` interface to remove function definitions that already exist in OpenZeppelin interfaces.

Update

Partially resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. All of the above recommendations were followed with two exceptions:

- The `Hypervisor` contract still contains a reference to the VISR token
- The `IHypervisor` interface still defines `totalSupply` and `transfer` instead of inheriting them from the OpenZeppelin IERC20 interface

[N10] UNUSED FUNCTION PARAMETER

The function `deposit` in UniProxy.sol requires a `from` address as an input parameter. The `from` address is not used inside of the function body.

Recommendation

Remove the `from` address from the input parameters.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The `from` function parameter was removed from the `deposit` function.

[N11] TYPOGRAPHICAL ERRORS

In `Hypervisor.sol` change “update fess for inclusion" to "update fees for inclusion".

In `Hypervisor.sol` change `feesLimit0` and `feesLimit1` to `feesBase0` and `feesBase1` when the `_position` function is called with the `baseLower` and `baseUpper` parameters in the `rebalance` and `pendingFees` functions.

In `Hypervisor.sol` change `feesBase0` and `feesBase1` to `feesLimit0` and `feesLimit1` when the `_position` function is called with the `limitLower` and `limitUpper` parameters in the `rebalance` and `pendingFees` functions.

In `HypervisorFactory.sol` change toke0, token1, fee" to "token0, token1, fee".

In `HypervisorV3Migrator.sol` change "@title HperVisor V3 Migrator" to "@title HyperVisor V3 Migrator".

Rename the file `admin.sol` to `Admin.sol`.

Recommendation

For clarity, consider making the suggested changes.

Update

Partially resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The following recommended changes were not made:

- Change `feesLimit0` and `feesLimit1` to `feesBase0` and `feesBase1` when assigning return values from the `_position` function when using the `baseLower` and `baseUpper` arguments
- Change `feesBase0` and `feesBase1` to `feesLimit0` and `feesLimit1` when assigning return values from the `_position` function when using the `limitLower` and `limitUpper` arguments
- In `HypervisorFactory.sol`, on line 11 change `toke0` to `token0`
- Rename the file `admin.sol` to `Admin.sol`

[N12] FIX MISLEADING DOCUMENTATION

In `Hypervisor.sol` a comment references the VISR ERC20 token that is no longer the correct token (GAMMA).

In `Hypervisor.sol` the `getTotalAmounts` function docstring contains the phrase "unused in the Hypervisor" without specifying what is unused (the liquidity).

In `Hypervisor.sol` the `pendingFees` function has a comment that mentions that it withdraws all liquidity when the function does not do so.

Recommendation

Update references from VISR to GAMMA and update the docstring for `getTotalAmounts`. Fix the comment in `pendingFees` and change the name to more accurately reflect its usage, such as `updateFees`.

Update

Partially resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The `pendingFees` function was removed and part of its functionality was replaced with a new `zeroBurn` function. The other recommended changes were not implemented.

[N13] MISSING LICENSE IDENTIFIERS

The `UniProxy.sol`, `IUniProxy.sol`, and `admin.sol` contracts do not specify a license.

Recommendation

Use the `BUSL-1.1` license that is used in `Hypervisor.sol`.

Update

Resolved as of commit `487feb4b30320b9565642260393bb1ea9775dff8`. The `BUSL-1.1` license was added to the three aforementioned files.

GLOSSARY

SEVERITY DEFINITIONS

Critical	This issue is straightforward to exploit and is likely to lead to catastrophic impact for client's reputation and can lead to financial loss for client or users.
High	This issue is difficult to exploit and is likely to lead to catastrophic impact for client's reputation and can lead to financial loss for client or users.
Medium	This issue is important to fix and puts a subset of users' data at risk and is possible to lead to moderate financial impact.
Low	This issue is not exploitable on a recurring basis and cannot have a significant impact on execution.
Informational (Note)	This issue does not pose an immediate risk but is relevant to security best practices.
Undetermined	The extent of the risk was not determined during this audit.

APPENDIX A – LIST OF FILES IN SCOPE

contracts/Hypervisor.sol
contracts/HypervisorFactory.sol
contracts/UniProxy.sol
contracts/HypervisorV3Migrator.sol
contracts/Swap.sol
contracts/proxy/admin.sol
contracts/interfaces/IHypervisor.sol
contracts/interfaces/IUniversalVault.sol
contracts/interfaces/IVault.sol
contracts/interfaces/IUniProxy.sol