JUNE.24

SECURITY REVIEW REPORT FOR SPOOL

CONTENTS

ABOUT HEXENS

Hexens is a cybersecurity company that strives to elevate the standards of security in Web 3.0, create a safer environment for users, and ensure mass Web 3.0 adoption.

Hexens has multiple top-notch auditing teams specialized in different fields of information security, showing extreme performance in the most challenging and technically complex tasks, including but not limited to: Infrastructure Audits, Zero Knowledge Proofs / Novel Cryptography, DeFi and NFTs. Hexens not only uses widely known methodologies and flows, but focuses on discovering and introducing new ones on a day-to-day basis.

In 2022, our team announced the closure of a $4.2 million seed round led by IOSG Ventures, the leading Web 3.0 venture capital. Other investors include Delta Blockchain Fund, Chapter One, Hash Capital, ImToken Ventures, Tenzor Capital, and angels from Polygon and other blockchain projects.

Since Hexens was founded in 2021, it has had an impressive track record and recognition in the industry: Mudit Gupta - CISO of Polygon Technology - the biggest EVM Ecosystem, joined the company advisory board after completing just a single cooperation iteration. Polygon Technology, 1inch, Lido, Hats Finance, Quickswap, Layerswap, 4K, RociFi, as well as dozens of DeFi protocols and bridges, have already become our customers and taken proactive measures towards protecting their assets.

EXECUTIVE SUMMARY

OVERVIEW

The audit targeted the Ethena strategy smart contract for the Spool Protocol V2 and the new smart vault factory contract, which is capable of minting fees to a beneficiary. The Spool Protocol V2 acts as a DeFi middleware enabling users to participate in a specific set of yield-generating strategies.

Our security assessment involved a comprehensive review of the strategy smart contracts over a total of 1 week. Throughout this audit, we identified several minor issues.

All of our reported issues were fixed by the development team and subsequently validated by us.

We can confidently say that the overall security and code quality of the project have increased after the completion of our audit.

SCOPE

The analyzed resources are located on:

https://github.com/solidant/spool-v2-core/pull/318
src/strategies/EthenaStrategy.sol
src/libraries/BytesUint256Lib.sol

https://github.com/solidant/spool-v2-core/pull/316
src/SmartVaultBeneficiary.sol
src/SmartVaultBeneficiaryFactoryHpf.sol

The issues described in this report were fixed in the following commit:

https://github.com/solidant/spool-v2-core/pull/318

commit: 37438f7c7cfaa8460387a2229f19c13012fde4e9

AUDITING DETAILS

STARTED
10.06.2024

DELIVERED
14.06.2024

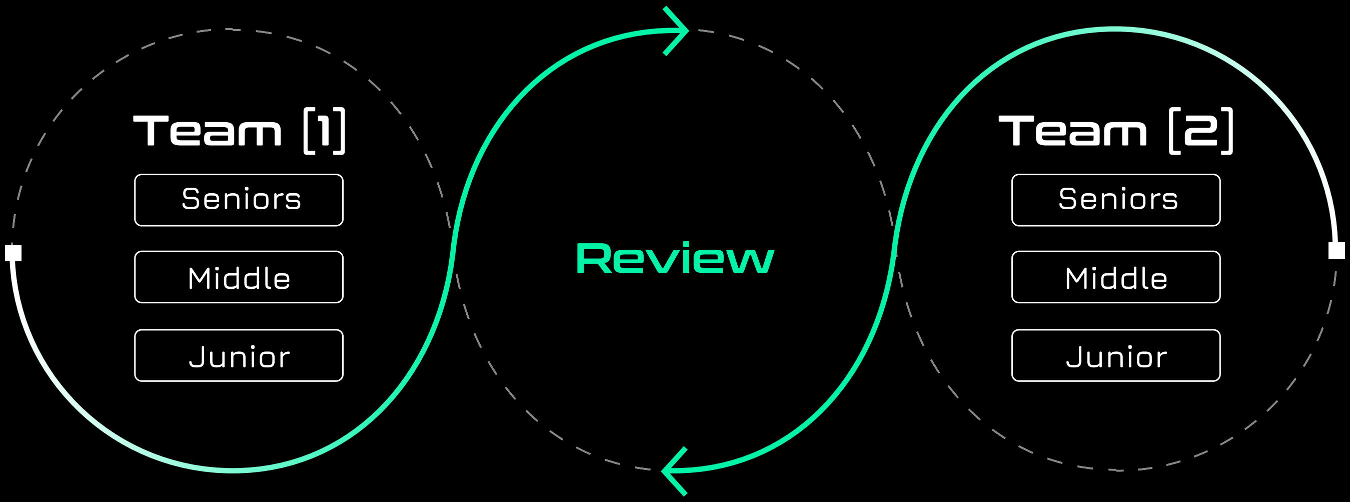
Review Led by

MIKHAIL EGOROV

Senior Security Researcher | Hexens

HEXENS METHODOLOGY

Hexens methodology involves 2 teams, including multiple auditors of different seniority, with at least 5 security engineers. This unique cross-checking mechanism helps us provide the best quality in the market.



SEVERITY STRUCTURE

The vulnerability severity is calculated based on two components

- Impact of the vulnerability
- Probability of the vulnerability

Impact	Probability			
	rare	unlikely	likely	very likely
Low/Info	Low/Info	Low/Info	Medium	Medium
Medium	Low/Info	Medium	Medium	High
High	Medium	Medium	High	Critical
Critical	Medium	High	Critical	Critical

SEVERITY CHARACTERISTICS

Smart contract vulnerabilities can range in severity and impact, and it's important to understand their level of severity in order to prioritize their resolution. Here are the different types of severity levels of smart contract vulnerabilities:

Critical

Vulnerabilities with this level of severity can result in significant financial losses or reputational damage. They often allow an attacker to gain complete control of a contract, directly steal or freeze funds from the contract or users, or permanently block the functionality of a protocol. Examples include infinite mints and governance manipulation.

High

Vulnerabilities with this level of severity can result in some financial losses or reputational damage. They often allow an attacker to directly steal yield from the contract or users, or temporarily freeze funds. Examples include inadequate access control integer overflow/underflow, or logic bugs.

Medium

Vulnerabilities with this level of severity can result in some damage to the protocol or users, without profit for the attacker. They often allow an attacker to exploit a contract to cause harm, but the impact may be limited, such as temporarily blocking the functionality of the protocol. Examples include uninitialized storage pointers and failure to check external calls.

Low

Vulnerabilities with this level of severity may not result in financial losses or significant harm. They may, however, impact the usability or reliability of a contract. Examples include slippage and front-running, or minor logic bugs.

Informational

Vulnerabilities with this level of severity are regarding gas optimizations and code style. They often involve issues with documentation, incorrect usage of EIP standards, best practices for saving gas, or the overall design of a contract. Examples include not conforming to ERC20, or disagreement between documentation and code.

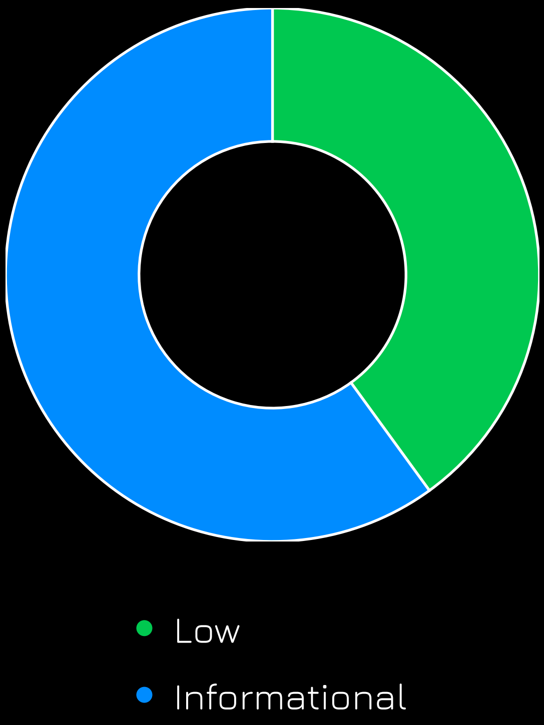
ISSUE SYMBOLIC CODES

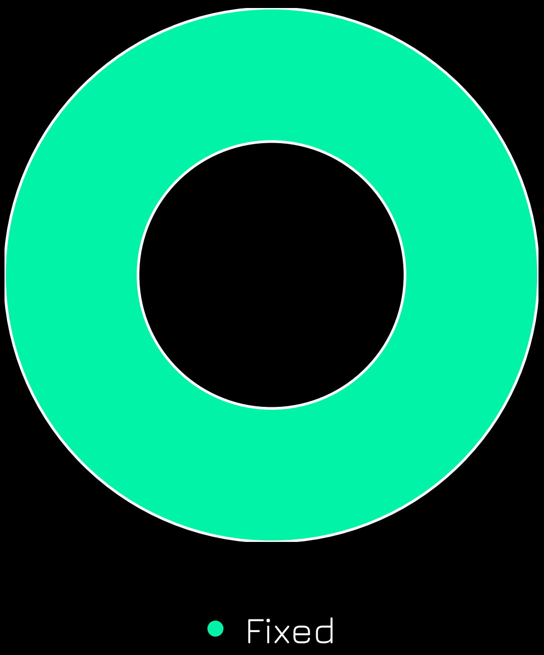
Every issue being identified and validated has its unique symbolic code assigned to the issue at the security research stage. Cause of the vulnerability reporting flow design, some of the rejected issues could be missing.

FINDINGS SUMMARY

Severity	Number of Findings
Critical	0
High	0
Medium	0
Low	2
Informational	3

Total: 5





WEAKNESSES

This section contains the list of discovered weaknesses.

SPOOL6-1

LACK OF USAGE _GETPROTOCOLREWARDSINTERNAL() FUNCTION

SEVERITY: Low

PATH:

src/strategies/EthenaStrategy.sol:_compound():L142-168

REMEDIATION:

Consider updating the _getProtocolRewardsInternal() function by adding this part:

```
if (amounts[0] > 0){
  ENAToken.safeTransfer(address(swapper), enaBalance);
```

Also, use the _getProtocolRewardsInternal() internal function within the _compound() function.

STATUS: Fixed

DESCRIPTION:

Function **_compound()** implements functionality for getting reward token balance and transferring it to the **swapper** contract. It is assumed that the protocol design should facilitate this functionality through the **_getProtocolRewardsInternal** function. Utilizing this internal function could improve gas efficiency and enhance code readability.

```
function _getProtocolRewardsInternal() internal virtual override returns
(address[] memory, uint256[] memory) {
        address[] memory tokens = new address[](1);
        tokens[0] = address(ENAToken);
        uint256[] memory amounts = new uint256[](1);
        amounts[0] = ENAToken.balanceOf(address(this));
        return (tokens, amounts);
    }
```

```
function _compound(address[] calldata, SwapInfo[] calldata swapInfo,
uint256[] calldata)
        internal
        virtual
        override
        returns (int256 compoundedYieldPercentage)
    {
        if (swapInfo.length > 0) {
            uint256 enaBalance = ENAToken.balanceOf(address(this));

            if (enaBalance > 0) {
                address[] memory tokensIn = new address[](1);
                tokensIn[0] = address(ENAToken);
                address[] memory tokensOut = new address[](1);
                tokensOut[0] = address(USDe);
                ENAToken.safeTransfer(address(swapper), enaBalance);
                    uint256 swappedAmount = swapper.swap(tokensIn, swapInfo,
tokensOut, address(this))[0];

                if (swappedAmount > 0) {
                                                    uint256 sUSDeBalanceBefore =
sUSDe.balanceOf(address(this));
                    sUSDe.deposit(swappedAmount, address(this));
                    compoundedYieldPercentage =
                                _calculateYieldPercentage(sUSDeBalanceBefore,
sUSDe.balanceOf(address(this)));
                }
            }
        }
    }
```

SPOOL6-4

AN INCORRECT ORIGINALLENGTH VALUE IN THE DECODE FUNCTION CAN LEAD TO UNEXPECTED DECODING RESULTS

SEVERITY: Low

PATH:

Path: src/libraries/BytesUint256Lib.sol

REMEDIATION:

Add the require() statement provided below to the beginning of the decode() function.

require((originalLength + 31) >> 5 == data.length);

STATUS: Fixed

DESCRIPTION:

Calling the **decode()** function with the wrong **originalLength** parameter can result in unexpected outcomes.

Example 1

- data = [0xFFFFFFFFFFFFFFFFFFFFFFFFFF, 0xAAAAAAAAAAAAAAAAAA]
- originalLength = 32
- result = 0x00000000000000000000000000000000000000ffffffffffffffffffffffffff (data[1] value isn't included)

Example 2

- data = [0xFFFFFFFFFFFFFFFFFFFFFFFFFF, 0xAAAAAAAAAAAAAAAAAA]
- originalLength = 100
- result = 0x00000000000000000000000000000000000000ffffffffffffffffffffffffff000 0000000000000000000000000000000000000000000aaaaaaaaaaaaaa aaaa0000000000000000000000000000000000000000000000000000 00000000000000000000 (unexpected trailing zeros)

To prevent this, it is recommended to add a **require()** statement inside the **decode()** function, as detailed in the remediation section.

```
function decode(
    uint256[] memory data,
    uint256 originalLength
) public pure returns (bytes memory) {
    bytes memory result = new bytes(originalLength);

    for (uint256 i = 0; i < data.length; i++) {
        uint256 chunk = data[i];
        assembly {
            mstore(add(result, add(32, mul(i, 32))), chunk)
        }
    }

    return result;
}
```

SPOOL6-3

ENCODE AND DECODE FUNCTIONS CAN BE OPTIMIZED

SEVERITY: Informational

PATH:

src/libraries/BytesUint256Lib.sol

REMEDIATION:

See description.

STATUS: Fixed

DESCRIPTION:

The **encode()** and **decode()** functions have the potential for optimization, which could result in gas savings when handling large amounts of data.

```
function encode(bytes memory data) internal pure returns (uint256[] memory)
{
    uint256 numChunks = (data.length + 31) / 32;
    uint256[] memory result = new uint256[](numChunks);

    for (uint256 i; i < numChunks; ++i) {
        uint256 chunk;
        assembly {
            chunk := mload(add(data, add(32, mul(i, 32))))
        }
        result[i] = chunk;
    }

    return result;
}
```

```solidity
function decode(
    uint256[] memory data,
    uint256 originalLength
) public pure returns (bytes memory) {
    bytes memory result = new bytes(originalLength);

    for (uint256 i = 0; i < data.length; i++) {
        uint256 chunk = data[i];
        assembly {
            mstore(add(result, add(32, mul(i, 32))), chunk)
        }
    }

    return result;
}
```

Replace the encode() and decode() functions with the following code snippets. There are two versions for each function: one optimized and the other even more optimized, entirely written in assembly.

```
// encode
function encode(
    bytes memory data
) internal pure returns (uint256[] memory result) {
    unchecked {
        uint256 numChunks = (data.length + 31) >> 5;
        result = new uint256[](numChunks);

        for (uint256 i = 1; i <= numChunks; ++i) {
            assembly {
                let pos := shl(5, i)
                mstore(add(result, pos), mload(add(data, pos)))
            }
        }
    }
}

// encode (more optimized(assembly))
function encode(
    bytes memory data
) internal pure returns (uint256[] memory result) {
    assembly {
        let numChunks := shr(5, add(mload(data), 63))
        let limit := shl(5, numChunks)
        result := mload(0x40)
        // update free memory pointer
        mstore(0x40, add(result, limit))

        mstore(result, sub(numChunks, 1))


        for { let pos := 32 } lt(pos, limit) { pos := add(pos, 32) } {
            mstore(add(result, pos), mload(add(data, pos)))
        }
    }
}
```

```
    // decode
    function decode(
        uint256[] memory data,
        uint256 originalLength
    ) public pure returns (bytes memory result) {
        result = new bytes(originalLength);
        uint256 len = data.length;
        unchecked {
            for (uint256 i = 1; i <= len; i++) {
                assembly {
                    let pos := shl(5, i)
                    mstore(add(result, pos), mload(add(data, pos)))
                }
            }
        }
    }

    // decode (more optimized(assembly))
    function decode(
        uint256[] memory data,
        uint256 originalLength
    ) public pure returns (bytes memory result) {
        assembly {
            let limit := shl(5, add(mload(data), 1))
            result := mload(0x40)
            // update free memory pointer
                mstore(0x40, add(result, shl(5, shr(5, add(originalLength,
63)))))


            mstore(result, originalLength)

            for { let pos := 32 } lt(pos, limit) { pos := add(pos, 32) } {
                mstore(add(result, pos), mload(add(data, pos)))
            }
        }

    }
```

SPOOL6-5

NO NEED OF SAFEAPPROVE USAGE

SEVERITY: Informational

PATH:

src/strategies/EthenaStrategy.sol:initialize()

REMEDIATION:

Considering using approve() instead of safeApprove() within initialize() for better gas efficiency.

STATUS: Fixed

DESCRIPTION:

Calling **safeApprove()** within **initialize()** is unnecessary as there is no risk of front-running. Since **safeApprove()** consumes more gas, it is beneficial to use **approve()** instead.

```
function initialize(string memory strategyName_, uint256 assetGroupId_)
external virtual initializer {
    __Strategy_init(strategyName_, assetGroupId_);
    USDe.safeApprove(address(sUSDe), type(uint256).max);
    USDe.safeApprove(address(swapper), type(uint256).max);
    sUSDe.safeApprove(address(swapper), type(uint256).max);
    lastPreviewRedeem = _previewConstantRedeem();
}
```

SPOOL6-7

MISSING ADDRESS ZERO CHECK

SEVERITY: Informational

PATH:

EthenaStrategy.sol:L73

REMEDIATION:

Consider adding a zero address check for the getPriceFeedManager_ variable in the constructor.

STATUS: Fixed

DESCRIPTION:

The **getPriceFeedManager_** variable in the constructor of the **EthenaStrategy** contract lacks zero address check, similar to other variables passed to the constructor.

```
constructor(
    IAssetGroupRegistry assetGroupRegistry_,
    ISpoolAccessControl accessControl_,
    IERC20Metadata USDe_,
    IsUSDe sUSDe_,
    IERC20Metadata ENAToken_,
    ISwapper swapper_,
    IUsdPriceFeedManager priceFeedManager_
) Strategy(assetGroupRegistry_, accessControl_, NULL_ASSET_GROUP_ID) {
    _disableInitializers();
    if (
         address(ENAToken_) == address(0) || address(swapper_) == address(0)
|| address(sUSDe_) == address(0)
            || address(USDe_) == address(0)
    ) {
        revert ConfigurationAddressZero();
    }
    USDe = USDe_;
    sUSDe = sUSDe_;
    ENAToken = ENAToken_;
    swapper = swapper_;
    priceFeedManager = priceFeedManager_;
    constantShareAmount = 10 ** (sUSDe.decimals() * 2);
}
```

hexens × Spool