OMNISCIA

June 3, 2025

SMART CONTRACT AUDIT REPORT

Fermion Protocol

V1.1.0 Upgrade Security Audit

omniscia.io

info@omniscia.io

Online report: fermion-protocol-v1.1.0-upgrade-security-audit

Omniscia.io is one of the fastest growing and most trusted blockchain security firms and has rapidly become a true market leader. To date, our team has collectively secured over 370+ clients, detecting 1,500+ high-severity issues in widely adopted smart contracts.

Founded in France at the start of 2020, and with a track record spanning back to 2017, our team has been at the forefront of auditing smart contracts, providing expert analysis and identifying potential vulnerabilities to ensure the highest level of security of popular smart contracts, as well as complex and sophisticated decentralized protocols.

Our clients, ecosystem partners, and backers include leading ecosystem players such as L'Oréal, Polygon, AvaLabs, Gnosis, Morpho, Vesta, Gravita, Olympus DAO, Fetch.ai, and LimitBreak, among others.

To keep up to date with all the latest news and announcements follow us on twitter @omniscia_sec.


omniscia.io


info@omniscia.io

Online report: fermion-protocol-v1.1.0-upgrade-security-audit

V1.1.0 Upgrade Security Audit Security Audit

Audit Report Revisions

Commit Hash	Date	Audit Report Hash
a71bb28f92	April 26th 2025	a6a65e6546
c332d7a9ee	May 20th 2025	346b977291
de8e96c89c	May 28th 2025	b24db1764d
de8e96c89c	June 3rd 2025	8291cd4cdb

Audit Overview

We were tasked with performing an audit of the Fermion Protocol codebase and in particular their V1.1.0 system update introducing changes and refactors across several contracts.

The system is meant to closely integrate with the Boson Protocol to offer an enhanced settlement layer that permits Real World Assets (RWAs) to be validated with significant scrutiny so as to facilitate high-value exchanges.

The features of the original implementation are inherited (i.e. fractionalization of assets, integration with OpenSea for settlement), and the system was upgraded to be compatible with the latest version of the Boson Protocol as well as support several new features like token-based fees, fixed-price OpenSea settlements, and digital assets accompanying physical ones termed "phygitals".

Over the course of the audit, we identified two significant issues around the auction and bidding process in addition to several issues involving transaction ordering and assumptions the system makes when it comes to the contract's state when a transaction is processed.

We advise the Fermion Protocol team to closely evaluate all minor-and-above findings identified in the report and promptly remediate them as well as consider all optimizational exhibits identified in the report.

Post-Audit Conclusion

The Fermion Protocol team iterated through all findings within the report and provided us with a revised commit hash to evaluate all exhibits on.

We evaluated all alleviations performed by Fermion Protocol and have identified that certain exhibits have not been adequately dealt with. We advise the Fermion Protocol team to revisit the following exhibits: **CYD-02M**, **EIP-01M**

Additionally, the following `informational` findings remain unaddressed and should be revisited: **FMR-02C**

Post-Audit Conclusion (de8e96c89c)

The Fermion Protocol team evaluated our follow-up recommendations and provided us with a commit hash to evaluate the resolution of all three aforementioned exhibits.

We confirmed that the manual review exhibits have been properly rectified and that the optimization outlined in `FMR-02C` had been safely extended to more calculations.

We consider all outputs of the audit report properly consumed by the Fermion Protocol team with no outstanding remediative actions remaining.

Audit Synopsis

Severity	Identified	Alleviated	Partially Alleviated	Acknowledged
Unknown	0	0	0	0
Informational	31	30	0	1
Minor	16	14	0	2
Medium	7	7	0	0
Major	2	2	0	0

During the audit, we filtered and validated a total of **2 findings utilizing static analysis** tools as well as identified a total of **54 findings during the manual review** of the codebase. We strongly recommend that any minor severity or higher findings are dealt with promptly prior to the project's launch as they can introduce potential misbehaviours of the system as well as exploits.

- **Scope**
- **Compilation**
- **Static Analysis**
- **Manual Review**
- **Code Style**

Scope

The audit engagement encompassed a specific list of contracts that were present in the commit hash of the repository that was in scope. The tables below detail certain meta-data about the target of the security assessment and a navigation chart is present at the end that links to the relevant findings per file.

Target

- Repository: **https://github.com/fermionprotocol/contracts**
- Commit: a71bb28f92a17df90f1a7ee50bafb974feac8bd1
- Language: Solidity
- Network: Ethereum, Polygon POS, Base, Optimism, Arbitrum, Boson, Fermion
- Revisions: **a71bb28f92**, **c332d7a9ee**, **de8e96c89c**

Contracts Assessed

File	Total Finding(s)
contracts/protocol/bases/mixins/Access.sol (ASS)	1
contracts/protocol/facets/backfill/BackfillingV1_1_0.sol (BV0)	0
contracts/protocol/clients/Common.sol (CNO)	1
contracts/protocol/facets/Config.sol (CGI)	1
contracts/protocol/bases/mixins/Context.sol (CTX)	1
contracts/protocol/bases/mixins/Custody.sol (CYD)	3
contracts/protocol/facets/Custody.sol (CYO)	2
contracts/protocol/domain/Constants.sol (CST)	1
contracts/protocol/libs/ContextLib.sol (CLB)	0
contracts/protocol/clients/CreatorToken.sol (CTN)	1

contracts/protocol/facets/CustodyVault.sol (CVT)	3
contracts/protocol/clients/oracle/ChainlinkPriceOracle.sol (CPO)	1
contracts/protocol/libs/EIP712.sol (EIP)	2
contracts/protocol/facets/Entity.sol (EYT)	6
contracts/protocol/domain/Errors.sol (ESR)	0
contracts/protocol/libs/EntityLib.sol (ELB)	0
contracts/protocol/facets/Funds.sol (FSD)	1
contracts/protocol/libs/FeeLib.sol (FLB)	1
contracts/protocol/clients/FermionFNFT.sol (FFN)	0
contracts/protocol/bases/mixins/FundsManager.sol (FMR)	4
contracts/protocol/libs/FermionFNFTLib.sol (FFF)	0
contracts/protocol/clients/FermionWrapper.sol (FWR)	1
contracts/protocol/clients/FermionFNFTBase.sol (FFT)	1
contracts/protocol/clients/FermionFractions.sol (FFS)	0

contracts/protocol/clients/FermionBuyoutAuction.sol (FBA)	2
contracts/protocol/clients/FermionFractionsMint.sol (FFM)	1
contracts/protocol/clients/FermionFractionsERC20.sol (FFE)	1
contracts/protocol/clients/FermionFNFTPriceManager.sol (FFP)	4
contracts/protocol/facets/Initialization.sol (INO)	1
contracts/protocol/libs/MathLib.sol (MLB)	0
contracts/protocol/facets/MetaTransaction.sol (MTN)	2
contracts/protocol/facets/Offer.sol (ORE)	10
contracts/protocol/facets/Pause.sol (PES)	1
contracts/protocol/facets/PriceOracleRegistry.sol (POR)	0
contracts/protocol/facets/Royalties.sol (RSE)	1
contracts/protocol/libs/RoyaltiesLib.sol (RLB)	0
contracts/protocol/bases/mixins/ReentrancyGuard.sol (RGD)	2
contracts/protocol/libs/Storage.sol (SEG)	0

contracts/protocol/clients/SeaportWrapper.sol (SWR)	0
contracts/protocol/domain/Types.sol (TSE)	0
contracts/protocol/facets/Verification.sol (VNO)	0

Compilation

The project utilizes `hardhat` as its development pipeline tool, containing an array of tests and scripts coded in TypeScript.

To compile the project, the `compile` command needs to be issued via the `npx` CLI tool to `hardhat`:

```
BASH

npx hardhat compile
```

The `hardhat` tool automatically selects Solidity version `0.8.24` based on the version specified within the `hardhat.config.ts` file.

The project contains discrepancies with regards to the Solidity version used, however, they are solely contained in external dependencies and can thus be safely ignored.

The `pragma` statements have been locked to `0.8.24` (`=0.8.24`), the same version utilized for our static analysis as well as optimizational review of the codebase.

During compilation with the `hardhat` pipeline, no errors were identified that relate to the syntax or bytecode size of the contracts.

Static Analysis

The execution of our static analysis toolkit identified **123 potential issues** within the codebase of which **119 were ruled out to be false positives** or negligible findings.

The remaining **4 issues** were validated and grouped and formalized into the **2 exhibits** that follow:

ID	Severity	Addressed	Title
CTN-01S	Informational	Yes	Multiple Top-Level Declarations
FWR-01S	Informational	Nullified	Inexistent Sanitization of Input Address

Manual Review

A **thorough line-by-line review** was conducted on the codebase to identify potential malfunctions and vulnerabilities in Fermion Protocol's code overhaul and update.

As the project at hand implements an iteration of the original Fermion Protocol, intricate care was put into ensuring that the **flow of funds within the system conforms to the specifications and restrictions** laid forth within the protocol's specification, that the Boson Protocol integration has been performed properly, and that the code delta introduced has not regressed any of the security features in the original implementation.

We validated that **all state transitions of the system occur within sane criteria** and that all rudimentary formulas within the system execute as expected. We **pinpointed multiple significant vulnerabilities** within the system which could have had **moderate-to-severe ramifications** to its overall operation; for more information, kindly consult all minor-and-above findings within the audit report as well as the audit report's summary.

Additionally, the system was investigated for any other commonly present attack vectors such as re-entrancy attacks, mathematical truncations, logical flaws and **ERC / EIP** standard inconsistencies. The documentation of the project was satisfactory to a great extent, containing extensive in-line documentation throughout the system.

A total of **54 findings** were identified over the course of the manual review of which **28 findings** concerned the behaviour and security of the system. The non-security related findings, such as optimizations, are included in the separate **Code Style** chapter.

The finding table below enumerates all these security / behavioural findings:

ID	Severity	Addressed	Title
CGI-01M	Minor	Yes	Inexistent Enforcement of Pause Region
CYD-01M	Minor	Nullified	Incorrect Caller Utilization
CYD-02M	Minor	Yes	Inexistent Accommodation of Equality Case
CYO-01M	Minor	Nullified	Potential Overcharge of Custodian Fee
CYO-02M	Medium	Yes	Checkout Request Clearance Race Condition

CVT-01M	Minor	Yes	Ambiguous Auction State Timestamp
EIP-01M	Minor	Yes	Incorrect Handling of Smart Contract Signers
EIP-02M	Minor	Yes	Inexistent Reset of Chain ID
EYT-01M	Minor	Yes	Inexistent Requirement of Successful Operation
EYT-02M	Medium	Yes	Improper Removal of Previous Administrator Roles
FBA-01M	Major	Yes	Bid Re-Entrancy Attack
FBA-02M	Major	Yes	Insecure Refund of Bid
FFP-01M	Medium	Yes	Inexistent Validation of Proposal or Epoch
FFP-02M	Medium	Yes	Insufficient Restriction of Price Proposal Creation

FFE-01M	Medium	Yes	Incorrect Restriction of Vote Adjustments
FMR-01M	Informational	Acknowledged	Potential Token Incompatibility
FMR-02M	Minor	Yes	Insecure Evaluation of Trusted Forwarder
INO-01M	Minor	Yes	Inexistent Validation of Call Success
MTN-01M	Minor	Acknowledged	Inexistent Expiry of Meta Transactions
ORE-01M	Informational	Yes	Improperly Commented Code
ORE-02M	Informational	Yes	Insecure Unchecked Code Block
ORE-03M	Minor	Yes	Fixed Resolution Period
ORE-04M	Minor	Yes	Incorrect Break Logic

ORE-05M	Minor	Yes	Potential Loss of Native Funds
ORE-06M	Medium	Nullified	Incorrect Wrap Recipient
ORE-07M	Medium	Nullified	Inexistent Validation of Successful Redemption of Twins
PES-01M	Minor	Acknowledged	Improper Unpause Restriction
RSE-01M	Minor	Yes	Incorrect Break Logic

Code Style

During the manual portion of the audit, we identified **26 optimizations** that can be applied to the codebase that will decrease the operational cost associated with the execution of a particular function and generally ensure that the project complies with the latest best practices and standards in Solidity.

Additionally, this section of the audit contains any opinionated adjustments we believe the code should make to make it more legible as well as truer to its purpose.

These optimizations are enumerated below:

ID	Severity	Addressed	Title
ASS-01C	Informational	Yes	Inefficient Import Paths
CPO-01C	Informational	Yes	Inefficient Event Emissions
CNO-01C	Informational	Yes	Redundant Import of Logic Implementation
CST-01C	Informational	Yes	Generic Typographic Mistake
CTX-01C	Informational	Yes	Redundant Constant Declaration
CYD-01C	Informational	Yes	Ineffectual Usage of Safe Arithmetics
CVT-01C	Informational	Nullified	Incorrect Revert Documentation
CVT-02C	Informational	Yes	Inefficient Re-Assignment of Vault Parameters
EYT-01C	Informational	Yes	Generic Typographic Mistakes
EYT-02C	Informational	Yes	Ineffectual Usage of Safe Arithmetics

EYT-03C	Informational	Yes	Inexplicable Literal Enum Cast
EYT-04C	Informational	Yes	Optimization of Modifier Application
FLB-01C	Informational	Yes	Ineffectual Usage of Safe Arithmetics
FFT-01C	Informational	Yes	Non-Standard Reservation of Storage Slot
FFP-01C	Informational	Yes	Ineffectual Usage of Safe Arithmetics
FFP-02C	Informational	Yes	Redundant Named Argument
FFM-01C	Informational	Nullified	Optimization of Migration Statements
FSD-01C	Informational	Yes	Non-Standard Explicit Contract Invocations
FMR-01C	Informational	Yes	Generic Typographic Mistake
FMR-02C	Informational	Yes	Ineffectual Usage of Safe Arithmetics
MTN-01C	Informational	Yes	Generic Typographic Mistake
ORE-01C	Informational	Yes	Generic Typographic Mistake
ORE-02C	Informational	Yes	Redundant Assignment of Boson Seller ID
ORE-03C	Informational	Yes	Redundant Input Address Validation

RGD-01C	Informational	Yes	Generic Typographic Mistake
RGD-02C	Informational	Yes	Repetitive Value Literals

CreatorToken Static Analysis Findings

CTN-01S: Multiple Top-Level Declarations

Type	Severity	Location
Code Style	Informational	CreatorToken.sol:L7, L45, L102

Description:

The referenced file contains multiple top-level declarations that decrease the legibility of the codebase.

Example:

contracts/protocol/clients/CreatorToken.sol

SOL

```sol
interface ICreatorToken {
    event TransferValidatorUpdated(address oldValidator, address newValidator);
    error SameTransferValidator();

    /**
     * @notice Sets a new transfer validator
     *
     * Emits a TransferValidatorUpdated event if successful
     *
     * Reverts if:
```

Example (Cont.):

```sol
     * - Caller is not the Contract owner
     * - The new validator is the same as the current one
     *
     * @param _newValidator The new transfer validator address.
     */
    function setTransferValidator(address _newValidator) external;

    /**
     * @notice Gets the current transfer validator
     *
     * @return transferValidator The current transfer validator address.
     */
    function getTransferValidator() external view returns (address transferValidator);

    /**
     * @notice Returns the transfer validation function used.
     *
     * @return functionSignature The function signature of the transfer validation function.
     * @return isViewFunction True if the function is a view function, false otherwise.
     */
    function getTransferValidationFunction() external view returns (bytes4 functionSignature, bool isViewFunction);
}

/**
 * @title CreatorToken
 * @notice Enforces transfer validation for creator tokens
 *
 */
```

Example (Cont.):

```sol
contract CreatorToken is Ownable, ICreatorToken {
```

Recommendation:

We advise all highlighted top-level declarations to be split into their respective code files, avoiding unnecessary imports as well as increasing the legibility of the codebase.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The referenced `ICreatorToken` and `ITransferValidator721` interfaces have been relocated to a dedicated file each and are now imported properly.

FermionWrapper Static Analysis Findings

FWR-01S: Inexistent Sanitization of Input Address

Type	Severity	Location
Input Sanitization	Informational	FermionWrapper.sol:L46-L51

Description:

The linked function accepts an `address` argument yet does not properly sanitize it.

Impact:

The presence of zero-value addresses, especially in `constructor` implementations, can cause the contract to be permanently inoperable. These checks are advised as zero-value inputs are a common side-effect of off-chain software related bugs.

Example:

contracts/protocol/clients/FermionWrapper.sol

SOL

```sol
46  constructor(address _seaportWrapper, address
_strictAuthorizedTransferSecurityRegistry, address _wrappedNative) {
47      if (_wrappedNative == address(0) || _seaportWrapper == address(0)) revert
FermionGeneralErrors.InvalidAddress();
48      WRAPPED_NATIVE = IWrappedNative(_wrappedNative);
49      SEAPORT_WRAPPER = _seaportWrapper;
50      STRICT_AUTHORIZED_TRANSFER_SECURITY_REGISTRY =
_strictAuthorizedTransferSecurityRegistry;
51  }
```

Recommendation:

We advise some basic sanitization to be put in place by ensuring that the `address` specified is non-zero.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The Fermion Protocol team evaluated this exhibit and specified that the referenced input argument is not meant to be sanitized as a zero-address value is permitted for it.

Config Manual Review Findings

CGI-01M: Inexistent Enforcement of Pause Region

Type	Severity	Location
Logical Fault	Minor	Config.sol:L112

Description:

The `ConfigFacet::setProtocolFeeTable` function will not enforce the `Config` pausable region restriction.

Impact:

The `Config::setProtocolFeeTable` function will remain accessible even if the `Config` region has been paused which is incorrect.

Example:

contracts/protocol/facets/Config.sol

SOL

```
74  /**
75   * @notice Sets the protocol fee percentage.
76   *
77   * Emits a ProtocolFeePercentageChanged event if successful.
78   *
79   * Reverts if:
80   * - The caller is not a protocol admin
81   * - The _protocolFeePercentage is greater than 10000
82   *
83   * @param _protocolFeePercentage - the percentage that will be taken as a fee
from the net of a Boson Protocol sale or auction (after royalties)
```

Example (Cont.):

```sol
84   *
85   * N.B. Represent percentage value as an unsigned int by multiplying the
percentage by 100:
86   * e.g, 1.75% = 175, 100% = 10000
87   */
88  function setProtocolFeePercentage(
89      uint16 _protocolFeePercentage
90  ) external onlyRole(ADMIN) notPaused(FermionTypes.PausableRegion.Config)
nonReentrant {
91      setProtocolFeePercentageInternal(_protocolFeePercentage);
92  }
93
94  /**
95   * @notice Sets the feeTable for a specific token given price ranges and fee
tiers for
96   * the corresponding price ranges.
97   *
98   * Reverts if the number of fee percentages does not match the number of price
ranges.
99   * Reverts if the price ranges are not in ascending order.
100  * Reverts if any of the fee percentages value is above 100%.
101  *
102  * @dev Caller must have ADMIN role.
103  *
104  * @param _tokenAddress - the address of the token
105  * @param _priceRanges - array of token price ranges
106  * @param _feePercentages - array of fee percentages corresponding to each price
range
107  */
108 function setProtocolFeeTable(
109     address _tokenAddress,
110     uint256[] calldata _priceRanges,
111     uint16[] calldata _feePercentages
```

Example (Cont.):

```sol
112 ) external onlyRole(ADMIN) nonReentrant {
```

Recommendation:

We advise it to be imposed, ensuring consistency across the `ConfigFacet` implementation.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The pause region is now properly enforced in the **`Config::setProtocolFeeTable`** function, alleviating this exhibit.

Custody Manual Review Findings

CYO-01M: Potential Overcharge of Custodian Fee

Type	Severity	Location
Logical Fault	Minor	Custody.sol:L521

Description:

The `Custody::_processCustodianUpdate` function will charge custody fees up to the current `block.timestamp` even though a custodian update might have been requested earlier due to the custodian not fulfilling their duties.

Impact:

A custodian is not incentivized to act as expected given that they will be paid until they are replaced regardless of the fulfilment of their duties.

Example:

contracts/protocol/facets/Custody.sol

SOL

```sol
521 uint256 custodianPayoff = ((block.timestamp - vault.period) * currentCustodianFee)
/ currentCustodianPeriod;
```

Recommendation:

As the custodian would be misbehaving in such circumstances, it might be fairer to charge a custodian fee up to the last time they were considered active.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The Fermion Protocol team evaluated this exhibit and consider the potential overcharge of a validator to be acceptable as the proposed solution would open up more severe griefing attack vectors for malicious sellers.

As such, we consider the original exhibit to be incorrect.

CYO-02M: Checkout Request Clearance Race Condition

Type	Severity	Location
Logical Fault	Medium	Custody.sol:L193-L231

Description:

A race condition exists when clearing a particular checkout request whereby a user might detect their checkout request being cleared and might update their taxes to an abnormally high amount as long as the caller of the checkout request's clearance has approved the protocol sufficiently.

Impact:

A user is able to consume an arbitrary amount of funds from a checkout request clearer by updating the tax due while a `CustodyFacet::clearCheckoutRequest` transaction is pending execution.

Example:

contracts/protocol/facets/Custody.sol

SOL

```sol
193 function clearCheckoutRequest(
194     uint256 _tokenId
195 ) external payable notPaused(FermionTypes.PausableRegion.Custody) nonReentrant {
196     FermionStorage.ProtocolLookups storage pl = FermionStorage.protocolLookups();
197     FermionTypes.CheckoutRequest storage checkoutRequest =
getValidCheckoutRequest(
198         _tokenId,
199         FermionTypes.CheckoutRequestStatus.CheckOutRequested,
200         pl
201     );
202
```

Example (Cont.):

```sol
    (, FermionTypes.Offer storage offer) = FermionStorage.getOfferFromTokenId(_tokenId);

    uint256 taxAmount = checkoutRequest.taxAmount;
    address buyer = checkoutRequest.buyer;
    if (taxAmount == 0) {
        // Seller is finalizing the checkout
        EntityLib.validateSellerAssistantOrFacilitator(offer.sellerId, offer.facilitatorId);
    } else {
        // Buyer is finalizing the checkout
        address msgSender = _msgSender();
        if (buyer != msgSender) {
            revert NotTokenBuyer(_tokenId, buyer, msgSender);
        }

        address exchangeToken = offer.exchangeToken;
        validateIncomingPayment(exchangeToken, taxAmount);
        increaseAvailableFunds(offer.sellerId, exchangeToken, taxAmount);
    }

    checkoutRequest.status = FermionTypes.CheckoutRequestStatus.CheckOutRequestCleared;

    pl.tokenLookups[_tokenId].phygitalsRecipient = EntityLib.getOrCreateEntityId(
        buyer,
        FermionTypes.EntityRole.Buyer,
        pl
    );

    emit CheckOutRequestCleared(offer.custodianId, _tokenId);
```

Example (Cont.):

```sol
}
```

Recommendation:

We advise the code to permit the caller to specify an upper bound of taxes they are willing to pay, preventing them from paying an abnormal tax amount due to a last minute change or a malicious alteration.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The `Custody::clearCheckoutRequest` function was updated to accept a `_maxTaxAmount` argument that acts as a control variable for the tax that the caller is willing to pay, effectively alleviating this exhibit in full.

CustodyVault Manual Review Findings

CVT-01M: Ambiguous Auction State Timestamp

Type	Severity	Location
Logical Fault	Minor	CustodyVault.sol:L339, L386

Description:

The auction of a particular offer ID is considered simultaneously ongoing and ended when the `block.timestamp` equates its `auctionEndTime`.

Impact:

An auction's state is inconsistent when the `block.timestamp` equates its `endTime` as both bidding on and ending an auction is permitted.

Example:

contracts/protocol/facets/CustodyVault.sol

SOL

```sol
330 function bid(
331     uint256 _offerId,
332     uint256 _bidAmount
333 ) external payable notPaused(FermionTypes.PausableRegion.CustodyVault)
nonReentrant {
334     FermionStorage.ProtocolLookups storage pl = FermionStorage.protocolLookups();
335     FermionTypes.FractionAuction storage fractionAuction =
pl.offerLookups[_offerId].fractionAuction;
336
337     uint256 auctionEndTime = fractionAuction.endTime;
338     if (auctionEndTime == 0) revert AuctionNotStarted(_offerId);
339     if (auctionEndTime < block.timestamp) revert AuctionEnded(_offerId,
auctionEndTime);
```

Example (Cont.):

```sol
    if (auctionEndTime - block.timestamp < AUCTION_END_BUFFER)
        fractionAuction.endTime = block.timestamp + AUCTION_END_BUFFER;

    uint256 previousBid = fractionAuction.maxBid;
    uint256 minimalBid = (previousBid * (HUNDRED_PERCENT + MINIMAL_BID_INCREMENT)) / HUNDRED_PERCENT;
    if (_bidAmount < minimalBid) revert InvalidBid(_offerId, minimalBid, _bidAmount);

    address exchangeToken = FermionStorage.protocolEntities().offer[_offerId].exchangeToken;

    validateIncomingPayment(exchangeToken, _bidAmount);

    // release funds to the previous bidder
    increaseAvailableFunds(fractionAuction.bidderId, exchangeToken, previousBid);

    address msgSender = _msgSender();
    uint256 bidderId = EntityLib.getOrCreateEntityId(msgSender, FermionTypes.EntityRole.Buyer, pl);

    fractionAuction.maxBid = _bidAmount;
    fractionAuction.bidderId = bidderId;

    emit BidPlaced(_offerId, msgSender, bidderId, _bidAmount);
}

/**
 * @notice Ends and settles the fractional auction.
 * The winner gets the fractions, and the funds are stored in the vault.
 * If the end price is lower than the target price, a new auction is started.
 *
```

Example (Cont.):

SOL

```sol
368  * Emits AuctionFinished event if successful.
369  *
370  * Reverts if:
371  * - Custody region is paused
372  * - Auction is not available
373  * - Auction is still ongoing
374  * - Bid is too low
375  * - Caller does not provide enough funds
376  *
377  * @param _offerId - offer ID associated with the vault
378  */
379 function endAuction(uint256 _offerId) external
notPaused(FermionTypes.PausableRegion.CustodyVault) nonReentrant {
380     FermionStorage.ProtocolLookups storage pl = FermionStorage.protocolLookups();
381     FermionStorage.OfferLookups storage offerLookups = pl.offerLookups[_offerId];
382     FermionTypes.FractionAuction storage fractionAuction =
offerLookups.fractionAuction;
383
384     uint256 auctionEndTime = fractionAuction.endTime;
385     if (auctionEndTime == 0) revert AuctionNotStarted(_offerId);
386     if (auctionEndTime > block.timestamp) revert AuctionOngoing(_offerId,
fractionAuction.endTime);
```

Recommendation:

We advise either the bidding or the finalization process to become inclusive in its comparison, ensuring that the equality case permits only a single of the two actions to be performed.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The `CustodyVault::endAuction` function was updated to evaluate the end time in an inclusive manner, addressing this exhibit.

EIP712 Manual Review Findings

EIP-01M: Incorrect Handling of Smart Contract Signers

Type	Severity	Location
Logical Fault	Minor	EIP712.sol:L134-L136

Description:

The `EIP712::verify` function is presently incompatible with **EIP-7702** smart accounts as it will fail when attempting to validate the signature through **EIP-1271** even though the signature might have been a proper ECDSA signature.

Impact:

Any **EIP-7702** enabled account will be unable to interact with the `EIP712` implementation if it does not satisfy the **EIP-1271** standard even if it has generated valid ECDSA signatures.

Example:

contracts/protocol/libs/EIP712.sol

SOL

```sol
111 function verify(address _user, bytes32 _hashedMessage, Signature memory _sig)
internal view {
112     bytes32 typedMessageHash = toTypedMessageHash(_hashedMessage);
113
114     // Check if user is a contract implementing ERC1271
115     if (_user.code.length > 0) {
116         (bool success, bytes memory returnData) = _user.staticcall(
117             abi.encodeCall(IERC1271.isValidSignature, (typedMessageHash,
abi.encode(_sig)))
118         );
119         if (success) {
120             if (returnData.length != SLOT_SIZE) {
```

Example (Cont.):

SOL

```sol
                revert FermionGeneralErrors.UnexpectedDataReturned(returnData);
            } else {
                // Make sure that the lowest 224 bits (28 bytes) are not set
                if (uint256(bytes32(returnData)) & type(uint224).max != 0) {
                    revert
FermionGeneralErrors.UnexpectedDataReturned(returnData);
                }

                if (abi.decode(returnData, (bytes4)) !=
IERC1271.isValidSignature.selector)
                    revert SignatureValidationFailed();

                return;
            }
        } else {
            if (returnData.length == 0) {
                revert SignatureValidationFailed();
            } else {
                /// @solidity memory-safe-assembly
                assembly {
                    revert(add(SLOT_SIZE, returnData), mload(returnData))
                }
            }
        }
    }

    // Ensure signature is unique
    // See https://github.com/OpenZeppelin/openzeppelin-
contracts/blob/04695aecbd4d17dddfd55de766d10e3805d6f42f/contracts/cryptography/ECDSA.
sol#63
    if (
        uint256(_sig.s) >
0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0 ||
```

Example (Cont.):

```sol
        (_sig.v != 27 && _sig.v != 28)
    ) revert InvalidSignature();

    address signer = ecrecover(typedMessageHash, _sig.v, _sig.r, _sig.s);
    if (signer == address(0)) revert InvalidSignature();
    if (signer != _user) revert SignatureValidationFailed();
}
```

Recommendation:

We advise the code to not revert if the call failed to execute as a particular address containing code no longer implies that it is not able to sign payloads.

Alleviation (c332d7a9ee):

While the code was remediated in PR#485, the final commit hash we evaluated did not reflect this change rendering this exhibit to remain open.

Alleviation (de8e96c89c):

The code was properly updated to handle **EIP-1271** verification flow failures gracefully (i.e. when the `_user` does not support the interface yet is a smart contract), alleviating this exhibit in full.

EIP-02M: Inexistent Reset of Chain ID

Type	Severity	Location
Logical Fault	Minor	EIP712.sol:L70, L96

Description:

The `EIP712::buildDomainSeparator` function will incorrectly utilize the `CHAIN_ID_CACHED` value instead of a fresh `block.chainid` value, resulting in the same domain separator being computed if the `block.chainid` changes and the `address(this)` evaluation remains the same.

Impact:

The **EIP-712** domain separator of the contract will be incorrect if the `block.chainid` value changes (i.e. during a fork), permitting signature re-use across two distinct chains.

Example:

contracts/protocol/libs/EIP712.sol

SOL

```sol
65  function getDomainSeparator() internal view returns (bytes32) {
66      if (address(this) == FERMION_PROTOCOL_ADDRESS && block.chainid ==
CHAIN_ID_CACHED) {
67          return DOMAIN_SEPARATOR_CACHED;
68      }
69
70      return buildDomainSeparator(PROTOCOL_NAME, PROTOCOL_VERSION, address(this));
71  }
72
73  /**
74   * @notice Generates the domain separator hash.
```

Example (Cont.):

SOL

```sol
75   * @dev Using the chainId as the salt enables the client to be active on one
chain
76   * while a metatx is signed for a contract on another chain. That could happen if
the client is,
77   * for instance, a metaverse scene that runs on one chain while the contracts it
interacts with are deployed on another chain.
78   *
79   * @param _name - the name of the protocol
80   * @param _version -  The version of the protocol
81   * @param _contract - the address of the contract
82   * @return the domain separator hash
83   */
84  function buildDomainSeparator(
85      string memory _name,
86      string memory _version,
87      address _contract
88  ) internal view returns (bytes32) {
89      return
90          keccak256(
91              abi.encode(
92                  EIP712_DOMAIN_TYPEHASH,
93                  keccak256(bytes(_name)),
94                  keccak256(bytes(_version)),
95                  _contract,
96                  CHAIN_ID_CACHED
97              )
98          );
99  }
```

Recommendation:

We advise the code to properly fetch the `block.chainid` dynamically.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The code now queries the `block.chainid` value dynamically addressing this exhibit.

Entity Manual Review Findings

EYT-01M: Inexistent Requirement of Successful Operation

Type	Severity	Location
Input Sanitization	Minor	Entity.sol:L737-L747, L750

Description:

The `Entity::addOrRemoveAssociatedEntities` function will not ensure the removal operation has succeeded, permitting incorrect removals to be processed successfully.

Impact:

It is presently possible to remove a facilitator that was never present in the first place which is considered incorrect.

Example:

contracts/protocol/facets/Entity.sol

SOL

```sol
703 function addOrRemoveAssociatedEntities(
704     FermionTypes.AssociatedRole _associatedRole,
705     uint256 _sellerId,
706     uint256[] calldata _associatedEntitiesIds,
707     bool _add
708 ) internal {
709     FermionStorage.ProtocolLookups storage pl = FermionStorage.protocolLookups();
710     EntityLib.validateEntityId(_sellerId, pl);
711     validateAdmin(_sellerId, pl);
712
```

Example (Cont.):

```sol
    (
        uint256[] storage associatedEntities,
        mapping(uint256 => bool) storage isAssociatedRole
    ) = getAssociatedLookups(_sellerId, _associatedRole, pl);

    mapping(uint256 => FermionTypes.EntityData) storage entityData = FermionStorage.protocolEntities().entityData;
    for (uint256 i; i < _associatedEntitiesIds.length; ++i) {
        uint256 associatedEntityId = _associatedEntitiesIds[i];
        if (_add) {
            if (isAssociatedRole[associatedEntityId])
                revert AssociatedEntityAlreadyExists(_associatedRole, _sellerId, associatedEntityId);

            EntityLib.validateEntityRole(
                associatedEntityId,
                entityData[associatedEntityId].roles,
                _associatedRole == FermionTypes.AssociatedRole.Facilitator
                    ? FermionTypes.EntityRole.Seller
                    : FermionTypes.EntityRole.RoyaltyRecipient
            );

            associatedEntities.push(associatedEntityId);

            emit AssociatedEntityAdded(_associatedRole, _sellerId, associatedEntityId);
        } else {
            uint256 facilitatorsLength = associatedEntities.length;
            for (uint256 j; j < facilitatorsLength; ++j) {
                if (associatedEntities[j] == associatedEntityId) {
                    if (j != facilitatorsLength - 1)
```

Example (Cont.):

```sol
                        associatedEntities[j] = associatedEntities[facilitatorsLength - 1];
                    associatedEntities.pop();

                    emit AssociatedEntityRemoved(_associatedRole, _sellerId, associatedEntityId);
                    break;
                }
            }
        }

        isAssociatedRole[associatedEntityId] = _add;
    }
}
```

Recommendation:

We advise the code to ensure that at least one `associatedEntities` data entry was removed, reverting otherwise.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The code was updated to `revert` if no valid entity to remove has been supplied, addressing this exhibit.

EYT-02M: Improper Removal of Previous Administrator Roles

Type	Severity	Location
Logical Fault	Medium	Entity.sol:L679

Description:

The `Entity::acceptAdminRole` function is meant to remove all roles from the `previousAdmin` yet adds them due to an incorrect `true` flag.

Impact:

The transition of the administrator role is presently incorrect as the `previousAdmin` will not be stripped of all privileges despite what the code's documentation details.

Example:

contracts/protocol/facets/Entity.sol

SOL

```sol
651 function acceptAdminRole(uint256 _entityId, address _account,
FermionStorage.ProtocolLookups storage pl) internal {
652     FermionStorage.EntityLookups storage entityLookups =
pl.entityLookups[_entityId];
653     if (entityLookups.pendingAdmin != _account) revert NotAdmin(_entityId,
_account);
654
655     delete entityLookups.pendingAdmin;
656
657     FermionTypes.EntityData storage entityData =
EntityLib.fetchEntityData(_entityId);
658     address previousAdmin = entityData.admin;
659     delete pl.entityId[previousAdmin];
660
```

Example (Cont.):

```sol
    entityData.admin = _account;
    pl.entityId[_account] = _entityId;

    // add new admin account
    EntityLib.storeCompactAccountRole(
        _entityId,
        _account,
        0xff << (31 * BYTE_SIZE),
        true,
        pl,
        FermionStorage.protocolEntities()
    ); // compact role for all current and potential future roles

    // strip old account of all privileges
    EntityLib.storeCompactAccountRole(
        _entityId,
        previousAdmin,
        0xff << (31 * BYTE_SIZE),
        true,
        pl,
        FermionStorage.protocolEntities()
    );

    EntityLib.emitManagerAccountAddedOrRemoved(_entityId, _account, true);
    EntityLib.emitManagerAccountAddedOrRemoved(_entityId, previousAdmin, false);
}
```

Recommendation:

We advise the flag to be updated to `false`, ensuring roles are correctly removed from the previous account.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The previous administrative roles are now removed correctly alleviating this exhibit.

FermionBuyoutAuction Manual Review Findings

FBA-01M: Bid Re-Entrancy Attack

Type	Severity	Location
Logical Fault	Major	FermionBuyoutAuction.sol:L87-L171

Description:

The `FermionBuyoutAuction::bid` function does not impose any re-entrancy protections and nor does the `FermioinFractions::bid` function that forwards calls to it.

As the function will release the previous bid prior to updating several storage variables, the code is susceptible to re-entrancy vulnerabilities.

As an example, the bidder might re-enter the contract to cause the auction to flow into the `Reserved` state even though the actual bid data that will ultimately be written would not trigger such a state. Alternatively, the previous bidder could re-enter the contract with a higher bid, effectively capturing the refund from the newly deposited funds of the victim outbidder.

Impact:

It is presently possible for a malicious bidder to re-enter the contract and outbid a victim outbidder, re-capturing a refund to their own address erroneously out of the victim's funds.

Example:

contracts/protocol/clients/FermionBuyoutAuction.sol

SOL

```sol
87  function bid(uint256 _tokenId, uint256 _price, uint256 _fractions) external
payable {
88      FermionTypes.BuyoutAuctionStorage storage $ =
Common._getBuyoutAuctionStorage(
89          Common._getFermionFractionsStorage().currentEpoch
90      );
91      if (!$.tokenInfo[_tokenId].isFractionalised) revert
TokenNotFractionalised(_tokenId);
92
93      FermionTypes.Auction storage auction = Common.getLastAuction(_tokenId, $);
94      FermionTypes.AuctionDetails storage auctionDetails = auction.details;
95      if (auctionDetails.state == FermionTypes.AuctionState.Reserved) revert
AuctionReserved(_tokenId);
96
```

Example (Cont.):

```sol
97      uint256 minimalBid;
98      {
99          uint256 maxBid = auctionDetails.maxBid;
100         minimalBid = (maxBid * (HUNDRED_PERCENT + MINIMAL_BID_INCREMENT)) /
HUNDRED_PERCENT;
101
102         // due to rounding errors, the minimal bid can be equal to the max bid.
Ensure strict increase.
103         if (minimalBid == maxBid) minimalBid += 1;
104     }
105
106     if (_price < minimalBid) {
107         revert InvalidBid(_tokenId, _price, minimalBid);
108     }
109
110     uint256 fractionsPerToken;
111     {
112         FermionTypes.BuyoutAuctionParameters storage auctionParameters =
$.auctionParameters;
113         if (auctionDetails.state >= FermionTypes.AuctionState.Ongoing) {
114             if (block.timestamp > auctionDetails.timer) revert
AuctionEnded(_tokenId, auctionDetails.timer);
115
116             fractionsPerToken = auctionDetails.totalFractions;
117         } else {
118             fractionsPerToken =
Common.liquidSupply(Common._getFermionFractionsStorage().currentEpoch) / $.nftCount;
119             if (_price >= auctionParameters.exitPrice &&
auctionParameters.exitPrice > 0) {
120                 // If price is above the exit price, the cutoff date is set
121                 startAuctionInternal(_tokenId);
122             } else {
123                 // reset ticker for Unbidding
124                 auctionDetails.timer = block.timestamp +
auctionParameters.topBidLockTime;
```

Example (Cont.):

SOL

```sol
            }
        }
    }

    // Return to the previous bidder the fractions and the bid
    address exchangeToken = $.exchangeToken;
    payOutLastBidder(auctionDetails, exchangeToken);

    FermionTypes.Votes storage votes = auction.votes;
    uint256 availableFractions = fractionsPerToken - votes.total; // available fractions to additionaly be used in bid

    address msgSender = _msgSender();
    uint256 bidAmount;
    if (_fractions >= availableFractions) {
        // Bidder has enough fractions to claim the remaining fractions. In this case they win the auction at the current price.
        // If the locked fractions belong to other users, the bidder must still pay the corresponding price.
        _fractions = availableFractions;

        if (auctionDetails.state == FermionTypes.AuctionState.NotStarted) startAuctionInternal(_tokenId);
        auctionDetails.state = FermionTypes.AuctionState.Reserved;
    }

    uint256 totalLockedFractions;
    unchecked {
        totalLockedFractions = _fractions + votes.individual[msgSender]; // cannot overflow, since _fractions <= availableFractions = fractionsPerToken - votes.total
        bidAmount = ((fractionsPerToken - totalLockedFractions) * _price) / fractionsPerToken; // cannot overflow, since fractionsPerToken >= totalLockedFractions
    }

```

Example (Cont.):

```sol
    auctionDetails.maxBidder = msgSender;
    auctionDetails.lockedFractions = _fractions; // locked in addition to the votes. If outbid, this is released back to the bidder
    auctionDetails.maxBid = _price;

    if (_fractions > 0) {
        Common._transferFractions(
            msgSender,
            address(this),
            _fractions,
            Common._getFermionFractionsStorage().currentEpoch
        );
    }
    if (bidAmount > 0) {
        validateIncomingPayment(exchangeToken, bidAmount);
    }

    auctionDetails.lockedBidAmount = bidAmount;
    emit Bid(_tokenId, msgSender, _price, totalLockedFractions, bidAmount);
}
```

Recommendation:

We advise the code to impose re-entrancy protections throughout its functions, preventing such misbehaviours from arising.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The `ReentrancyGuard::nonReentrant` modifier is now applied to the `FermionFractions::bid` function, alleviating this exhibit.

FBA-02M: Insecure Refund of Bid

Type	Severity	Location
Logical Fault	Major	FermionBuyoutAuction.sol:L551

Description:

The referenced function will `revert` if a native fund bid refund fails.

Impact:

It is presently possible to prevent any outbid from occurring in an auction using native funds by reverting on specific fund receipt scenarios.

Example:

contracts/protocol/clients/FermionBuyoutAuction.sol

SOL

```sol
536 function payOutLastBidder(FermionTypes.AuctionDetails storage _auction, address
_exchangeToken) internal {
537     address bidder = _auction.maxBidder;
538     if (bidder == address(0)) return; // no previous bidder
539
540     uint256 lockedFractions = _auction.lockedFractions;
541
542     // transfer to previus bidder if they used some of the fractions. Do not
transfer the locked votes.
543     if (lockedFractions > 0) {
544         Common._transferFractions(
545             address(this),
```

Example (Cont.):

```sol
            bidder,
            lockedFractions,
            Common._getFermionFractionsStorage().currentEpoch
        );
    }
    transferERC20FromProtocol(_exchangeToken, payable(bidder),
_auction.lockedBidAmount);
```

Recommendation:

We advise the code to credit the refund to the bidder who can then claim it at a secondary transaction, preventing any vulnerabilities from arising due to the native refund such as return bombs or transaction failures.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

A credit mechanism was introduced for outbid refunds as advised, ensuring refunds are processed at a secondary transaction and thus alleviating this exhibit.

FermionFNFTPriceManager Manual Review Findings

FFP-01M: Inexistent Validation of Proposal or Epoch

Type	Severity	Location
Logical Fault	Medium	FermionFNFTPriceManager.sol:L242

Description:

The `FermionFNFTPriceManager::voteOnProposal` function does not validate the proposal being voted on which might change if the previous one was finalized and replaced while the transaction was processing or if the epoch advanced and a new proposal was introduced.

Impact:

The impact of a particular yes/no vote of a proposal entirely depends on transaction ordering as well as the blockchain's state which might change between a transaction's submission and its execution within the blockchain network.

Example:

contracts/protocol/clients/FermionFNFTPriceManager.sol

SOL

```sol
242 function voteOnProposal(bool _voteYes) external {
243     FermionTypes.FermionFractionsStorage storage fractionStorage =
Common._getFermionFractionsStorage();
244     uint256 currentEpoch = fractionStorage.currentEpoch;
245     FermionTypes.PriceUpdateProposal storage proposal = Common
246         ._getBuyoutAuctionStorage(currentEpoch)
247         .currentProposal;
248     address erc20Clone = fractionStorage.epochToClone[currentEpoch];
249     address msgSender = _msgSender();
250     uint256 fractionsBalance = IERC20(erc20Clone).balanceOf(msgSender);
251
```

Example (Cont.):

```sol
    if (proposal.state != FermionTypes.PriceUpdateProposalState.Active) {
        revert FractionalisationErrors.ProposalNotActive(proposal.proposalId);
    }

    if (!_finalizeProposal(proposal, Common.liquidSupply(currentEpoch))) {
        if (fractionsBalance == 0) revert FractionalisationErrors.NoVotingPower(msgSender);

        FermionTypes.PriceUpdateVoter storage voter = proposal.voters[msgSender];
        uint256 additionalVotes;

        if (voter.proposalId == proposal.proposalId) {
            if (voter.votedYes != _voteYes) revert FractionalisationErrors.ConflictingVote();
            unchecked {
                additionalVotes = fractionsBalance > voter.voteCount ? fractionsBalance - voter.voteCount : 0;
            }
            if (additionalVotes == 0) revert FractionalisationErrors.AlreadyVoted();
        } else {
            voter.proposalId = proposal.proposalId;
            voter.votedYes = _voteYes;
            additionalVotes = fractionsBalance;
        }
        voter.voteCount = fractionsBalance;

        if (_voteYes) proposal.yesVotes += additionalVotes;
        else proposal.noVotes += additionalVotes;

        emit IFermionFractionsEvents.PriceUpdateVoted(proposal.proposalId, msgSender, fractionsBalance, _voteYes);
    }
```

Example (Cont.):

```sol
280 }
```

Recommendation:

We advise the code to accept an additional price argument indicating the new exit price that the user is willing to accept, preventing proposal votes from being ambiguous.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

Proposal votes are now correctly attached to a specific ID alleviating this exhibit.

FFP-02M: Insufficient Restriction of Price Proposal Creation

Type	Severity	Location
Logical Fault	Medium	FermionFNFTPriceManager.sol:L198-L200

Description:

The `FermionFNFTPriceManager::updateExitPrice` function requires the caller to simply retain at least `1 wei` of the current epoch's fraction balance.

This permits a user with just `1 wei` to hijack all upcoming exit price updates as long as an oracle has not been confirmed by submitting a price update with a `HUNDRED_PERCENT` quorum that will never be met if they do not vote.

Impact:

It is possible for a fraction holder of just `1 wei` to hijack all exit price updates by submitting an unfulfillable quorum.

Example:

contracts/protocol/clients/FermionFNFTPriceManager.sol

SOL

```sol
198 if (IERC20(erc20Clone).balanceOf(_msgSender()) == 0) {
199     revert FractionalisationErrors.OnlyFractionOwner();
200 }
201
202 if (_quorumPercent < MIN_QUORUM_PERCENT || _quorumPercent > HUNDRED_PERCENT) {
203     revert FermionGeneralErrors.InvalidPercentage(_quorumPercent);
204 }
```

Recommendation:

We advise the code to impose other restrictions or conditions, such as permitting an existing proposal to be overwritten by another user with a higher balance or by restricting the maximum quorum as well.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The way proposals are processed was completely refactored permitting multiple proposals to be active concurrently.

As such, it is no longer possible to hijack upcoming price updates with any balance.

FermionFractionsERC20 Manual Review Findings

FFE-01M: Incorrect Restriction of Vote Adjustments

Type	Severity	Location
Logical Fault	Medium	FermionFractionsERC20.sol:L65

Description:

The `FermionFractionsERC20::_update` function will adjust votes on transfer solely when those votes are being transferred, causing burn operations (such as in `FermionBuyoutAuction::claim`) to not update the votes of a particular exit price update even though the `Common::liquidSupply` the update's quorum relies on would be affected.

Impact:

It is presently possible for a user to vote on an exit price proposal in the `FermionFNFTPriceManager` implementation and burn their fractions, causing the proposal's finalization to rely on an incorrect total and thus quorum.

Example:

contracts/protocol/clients/FermionFractionsERC20.sol

SOL

```sol
function _update(address from, address to, uint256 value) internal override {
    super._update(from, to, value);

    if (from != address(0) && to != address(0)) {
        IFermionFNFTPriceManager(owner()).adjustVotesOnTransfer(from, value);
    }
}
```

Recommendation:

We advise vote updates to be carried out on burn operations as well, ensuring that a user cannot vote on a proposal and burn their fractions to manipulate the quorum of a particular exit price proposal.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The code was updated to adjust the voting power on-transfer for burn operations as well, alleviating this exhibit.

FundsManager Manual Review Findings

FMR-01M: Potential Token Incompatibility

Type	Severity	Location
Standard Conformity	Informational	FundsManager.sol:L319-L321, L339-L345

Description:

The `FundsManager::isERC721Contract` function will contain several checks to evaluate whether a particular smart contract implementation is an **EIP-20** or **EIP-721** token which may be over-restrictive in certain circumstances.

As an example, an **EIP-20** token behind a Diamond will be incompatible due to yielding an "incompatible function signature" error and an **EIP-20** token with an empty `fallback` function would cause `SLOT_SIZE` validations to fail.

Impact:

As the code will error solely for non-conventional EIP tokens, the severity of this exhibit is informational and can be safely acknowledged.

Example:

contracts/protocol/bases/mixins/FundsManager.sol

SOL

```sol
313 function isERC721Contract(address _tokenAddress, bool _erc721expected) internal
view {
314     (bool success, bytes memory returnData) = _tokenAddress.staticcall(
315         abi.encodeCall(IERC165.supportsInterface, (type(IERC721).interfaceId))
316     );
317
318     if (success) {
319         if (returnData.length != SLOT_SIZE) {
320             revert FermionGeneralErrors.UnexpectedDataReturned(returnData);
321         } else {
322             // If returned value equals 1 (= true), the contract is ERC721 and we
should revert
```

Example (Cont.):

```sol
            uint256 result = abi.decode(returnData, (uint256)); // decoding into uint256 not bool to cover all cases

            if (result > 1) revert FermionGeneralErrors.UnexpectedDataReturned(returnData);

            // If we expect ERC721 and the contract is not ERC721, revert.
            // If we do not expect ERC721 and the contract is ERC721, revert.
            if ((result == 1) != _erc721expected)
                revert FundsErrors.ERC721CheckFailed(_tokenAddress, _erc721expected);
        }
    } else {
        if (returnData.length == 0) {
            if (_erc721expected) {
                revert FundsErrors.ERC721CheckFailed(_tokenAddress, _erc721expected);
            }

            // If ERC721 is not expected, do nothing. ERC20 not implementing IERC721 interface is expected to revert without reason.
        } else {
            // If an actual error message is returned, revert with it
            /// @solidity memory-safe-assembly
            assembly {
                revert(add(SLOT_SIZE, returnData), mload(returnData))
            }
        }
    }
}
```

Recommendation:

We advise the Fermion Protocol team to evaluate the range of tokens they wish to support, and to potentially introduce a manually updated list of token types to support non-canonical **EIP-721** and **EIP-20** implementations.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The Fermion Protocol team evaluated this exhibit, assessed its ramifications, and opted to acknowledge it.

FMR-02M: Insecure Evaluation of Trusted Forwarder

Type	Severity	Location
Language Specific	Minor	FundsManager.sol:L185-L189

Description:

The referenced evaluation of a token's trusted forwarder is potentially insecure as it will not catch decoding errors if the `_tokenAddress` supports the `ERC2771Context::trustedForwarder` function selector yet does not yield any data.

Examples of such implementations include contracts with empty `fallback` functions or colliding function signatures, the former of which is more likely.

Impact:

Any EIP token that has an empty `fallback` implementation will be unsupported by the `FundsManager` and thus Fermion Protocol system due to an unhandled decoding error.

Example:

contracts/protocol/bases/mixins/FundsManager.sol

SOL

```sol
177 /**
178  * @notice Checks if the contract at the token address is FNFN or not.
179  *
180  * @dev override this function in the child contract does not need this check
181  *
182  * @param _tokenAddress - address of the token to be transferred
183  */
184 function checkTrustedForwarder(address _tokenAddress) internal view virtual
returns (bool) {
185     try ERC2771Context(_tokenAddress).trustedForwarder() returns (address
trustedForwarder) {
186         return trustedForwarder == address(this);
```

Example (Cont.):

```sol
    } catch {
        return false;
    }
}
```

Recommendation:

We advise the code to utilize a low-level `staticcall` instead, manually decoding the yielded payload if it satisfies the `SLOT_SIZE` limitation similarly to the `FundsManager::isERC721Contract` function implementation.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The decoding error edge case outlined is no longer possible in the codebase as the trusted forwarder query is being safely performed with appropriate length checks for the yielded payload.

Initialization Manual Review Findings

INO-01M: Inexistent Validation of Call Success

Type	Severity	Location
Logical Fault	Minor	Initialization.sol:L91

Description:

The `InitializationFacet::initializeDiamond` function will succeed even if its internal `AccessController::initialize` call fails as the return data of the low-level `delegatecall` operation remains unchecked.

Impact:

The `InitializationFacet::initializeDiamond` function will not revert even if the initialization of the `AccessController` implementation fails.

Example:

contracts/protocol/facets/Initialization.sol

SOL

```
85  function initializeDiamond(
86      address _accessController,
87      address _defaultAdmin,
88      address _bosonProtocolAddress,
89      address _fermionFNFTImplementation
90  ) external noDirectInitialization {
91      _accessController.delegatecall(abi.encodeCall(AccessController.initialize,
(_defaultAdmin)));
92      initializeBosonSellerAndBuyerAndDR(_bosonProtocolAddress,
_fermionFNFTImplementation);
93
94      LibDiamond.DiamondStorage storage ds = LibDiamond.diamondStorage();
```

Example (Cont.):

```sol
95      ds.supportedInterfaces[type(IERC165).interfaceId] = true;
96      ds.supportedInterfaces[type(IDiamondCut).interfaceId] = true;
97      ds.supportedInterfaces[type(IDiamondLoupe).interfaceId] = true;
98      ds.supportedInterfaces[type(IERC173).interfaceId] = true;
99      ds.supportedInterfaces[type(IAccessControl).interfaceId] = true;
100 }
```

Recommendation:

We advise the return data to be explicitly validated, preventing incorrect successful initializations of the Diamond.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The Fermion Protocol team evaluated this exhibit and opted to acknowledge it as they considered initialization to occur only once during the lifetime of the diamond; an action whose outcome is immediately validated by the Fermion Protocol team.

MetaTransaction Manual Review Findings

MTN-01M: Inexistent Expiry of Meta Transactions

Type	Severity	Location
Logical Fault	Minor	MetaTransaction.sol:L66-L87

Description:

The `MetaTransaction::executeMetaTransaction` function does not contain any expiry validation, causing signed transactions to be valid as long as the nonce has not been consumed yet.

Impact:

Any signed meta-transaction does not expire, resulting in potentially untimely transaction execution.

Example:

contracts/protocol/facets/MetaTransaction.sol

SOL

```
66  function executeMetaTransaction(
67      address _userAddress,
68      string calldata _functionName,
69      bytes calldata _functionSignature,
70      uint256 _nonce,
71      Signature calldata _sig,
72      uint256 _offerIdWithEpoch
73  ) external payable notPaused(FermionTypes.PausableRegion.MetaTransaction)
nonReentrant returns (bytes memory) {
74      address userAddress = _userAddress; // stack too deep workaround.
75      validateTx(_functionName, _functionSignature, _nonce, userAddress);
```

Example (Cont.):

```sol

    FermionTypes.MetaTransaction memory metaTx;
    metaTx.nonce = _nonce;
    metaTx.from = userAddress;
    metaTx.contractAddress = getContractAddress(_offerIdWithEpoch);
    metaTx.functionName = _functionName;
    metaTx.functionSignature = _functionSignature;

    verify(userAddress, hashMetaTransaction(metaTx), _sig);

    return executeTx(metaTx.contractAddress, userAddress, _functionName,
_functionSignature, _nonce);
}
```

Recommendation:

We advise the code to either permit a nonce to be consumed explicitly, to introduce an expiry to signed transactions, or to introduce both security features.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The Fermion Protocol team evaluated this exhibit and clarified that meta-transactions are expected to be executed immediately and that they will consider introducing an expiry in the future.

Offer Manual Review Findings

ORE-01M: Improperly Commented Code

Type	Severity	Location
Logical Fault	Informational	**Offer.sol:L103, L105**

Description:

The code in scope of the audit is meant to be compatible with Boson Protocol's version `2.4.2` yet the code to accommodate for it is commented out.

Impact:

The commented-out code functions as expected when it is re-introduced, rendering this to be an informational notice.

Example:

contracts/protocol/facets/Offer.sol

SOL

```sol
103 // bosonOffer.price = _offer.verifierFee; // Boson currently requires price to be 0; this will be enabled with 2.4.2 release
104 bosonOffer.sellerDeposit = _offer.sellerDeposit;
105 // bosonOffer.buyerCancelPenalty = _offer.verifierFee; // Boson currently requires buyerCancelPenalty to be 0; this will be enabled with 2.4.2 release
```

Recommendation:

We advise the code to be re-introduced, ensuring consistency across the two codebases.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The code was revised in a different way ensuring that it is properly compatible with the 2.4.2 release, addressing this exhibit.

ORE-02M: Insecure Unchecked Code Block

Type	Severity	Location
Mathematical Operations	Informational	Offer.sol:L547-L552

Description:

The referenced `unchecked` code block is insecure as there might be more considerations (i.e. royalties) than the ones explicitly validated.

Impact:

Although the code should fail downstream in the OpenSea call, it is still advised to avoid corrupt variables internally as relying on an external integrator to validate these values is bad practice.

Example:

contracts/protocol/facets/Offer.sol

SOL

```
546 _priceDiscovery.price = _buyerOrder.parameters.offer[0].startAmount;
547 unchecked {
548     for (uint256 i = 1; i < _buyerOrder.parameters.consideration.length; i++) {
549         // reduce the price by the openSea fee and the royalties
550         _priceDiscovery.price -=
_buyerOrder.parameters.consideration[i].startAmount;
551     }
552 }
```

Recommendation:

We advise the `unchecked` code block to be omitted, preventing a subtraction underflow from occurring for the `_priceDiscovery.price` value.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The `unchecked` code block has been properly removed from the reference operation, addressing this exhibit.

ORE-03M: Fixed Resolution Period

Type	Severity	Location
Logical Fault	Minor	Offer.sol:L121

Description:

The resolution period when creating a Boson offer via the `OfferFacet::createOffer` is fixed to `7 days` even though its permitted values are dynamically configured in the Boson Protocol system.

Impact:

Should the limits change on the Boson Protocol system, offer creations will consistently fail until the contracts are upgraded.

Example:

contracts/protocol/facets/Offer.sol

SOL

```sol
118 IBosonProtocol.OfferDurations memory bosonOfferDurations;
119 bosonOfferDurations.disputePeriod = type(uint256).max; // TBD: how to limit the time verifier has to respond
120 bosonOfferDurations.voucherValid = 1; // It could be 0, since in fermion offers, commit and redeem happen atomically, but Boson does not allow it
121 bosonOfferDurations.resolutionPeriod = 7 days; // Not needed for fermion, but Boson requires it
```

Recommendation:

We advise the code to either query the current limits and utilize either one, or to permit the default resolution period to be re-configured by the administrators of the Fermion Protocol system.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The Fermion Protocol team evaluated this exhibit and specified that while it is technically correct, they wish to acknowledge it as they do not anticipate configurational changes in the Boson Protocol system.

ORE-04M: Incorrect Break Logic

Type	Severity	Location
Logical Fault	Minor	Offer.sol:L302-L303

Description:

The `Offer::listFixedPriceOrdersInternal` function will attempt to replace the `address(0)` with the `admin` of a particular `sellerId`, however, it will `break` after identifying the first `address(0)` instance even though duplicate entries are permitted in the `royaltyInfo` of a particular offer.

Impact:

While the vulnerability is correct, the likelihood of a user introducing the zero-address twice in the same royalty information configuration is low rendering this issue to be minor.

Example:

contracts/protocol/facets/Offer.sol

SOL

```sol
286 function listFixedPriceOrdersInternal(
287     uint256 _offerId,
288     uint256[] calldata _prices,
289     uint256[] calldata _endTimes,
290     address wrapperAddress,
291     uint256 startingNFTId
292 ) internal {
293     if (_prices.length != _endTimes.length)
294         revert FermionGeneralErrors.ArrayLengthMismatch(_prices.length,
_endTimes.length);
295
```

Example (Cont.):

```sol
296     FermionStorage.ProtocolEntities storage pe =
FermionStorage.protocolEntities();
297     FermionTypes.Offer storage offer = pe.offer[_offerId];
298     FermionTypes.RoyaltyInfo memory royaltyInfo = offer.royaltyInfo;
299     // If some of the royalty recipient is set to 0, replace it with entity admin
300     for (uint256 i; i < royaltyInfo.recipients.length; ++i) {
301         if (royaltyInfo.recipients[i] == address(0)) {
302             royaltyInfo.recipients[i] =
payable(pe.entityData[offer.sellerId].admin);
303             break;
304         }
305     }
306
307     wrapperAddress.listFixedPriceOrders(startingNFTId, _prices, _endTimes,
royaltyInfo, offer.exchangeToken);
308 }
```

Recommendation:

We advise the code to not `break` early, ensuring all `address(0)` royalty recipients are properly replaced.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The `break` statement was removed ensuring all royalty recipients are properly assigned.

ORE-05M: Potential Loss of Native Funds

Type	Severity	Location
Input Sanitization	Minor	Offer.sol:L423

Description:

The `Offer::unwrapNFT` and `Offer::unwrapNFTAndSetVerificationTimeout` functions may result in fund loss if an NFT that is either an OpenSea auction or an OpenSea fixed price listing is unwrapped with non-zero native funds.

Impact:

Native funds might be lost if they are sent alongside an OpenSea-reliant NFT unwrapping operation.

Example:

contracts/protocol/facets/Offer.sol

SOL

```
332 /**
333  * @notice Unwraps F-NFT, uses seaport to sell the NFT
334  * Reverts if:
335  * - Offer region is paused
336  * - Caller is not the seller's assistant or facilitator
337  * - If seller deposit is non zero and there are not enough funds to cover it
338  * - The price is not high enough to cover the verification fee
339  *
340  * @param _tokenId - the token ID
341  * @param _wrapType - the wrap type
```

Example (Cont.):

SOL

```sol
 * @param _data - additional data, depending on the wrap type
 */
function unwrapNFT(uint256 _tokenId, FermionTypes.WrapType _wrapType, bytes calldata _data) external payable {
    unwrapNFT(_tokenId, _wrapType, _data, 0);
}

/**
 * @notice Same as unwrapNFT, but also sets the verification timeout
 *
 * @param _tokenId - the token ID
 * @param _wrapType - the wrap type
 * @param _data - additional data, depending on the wrap type
 * @param _verificationTimeout - the verification timeout in UNIX timestamp
 */
function unwrapNFTAndSetVerificationTimeout(
    uint256 _tokenId,
    FermionTypes.WrapType _wrapType,
    bytes calldata _data,
    uint256 _verificationTimeout
) external payable {
    unwrapNFT(_tokenId, _wrapType, _data, _verificationTimeout);
}
```

Recommendation:

We advise the code of the `Offer::openSeaAuction` and `Offer::openSeaFixedPrice` functions to validate that the native funds supplied alongside the call are zero.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The Fermion Protocol team evaluated the EXHIBIT and opted to instead prevent native funds at the `Offer` level if a non-self-sale unwrapping type has been specified with a seller deposit of 0 signaling an OpenSea-type unwrapping operation.

ORE-06M: Incorrect Wrap Recipient

Type	Severity	Location
Logical Fault	Medium	Offer.sol:L807

Description:

The `Offer::wrapNFTS` function will wrap the NFTs to the `msgSender` solely when the `WrapType` has been specified as `OS_AUCTION` which is incorrect as the `OS_FIXED_PRICE` type (as evidenced by `FermionWrapper::unwrapFixedPriced`) expects the NFT to be owned by an address other than the contract itself.

Impact:

Wrapping functionality for the `OS_FIXED_PRICE` will fail when attempting to unwrap the NFTs via the `Offer::unwrapNFT` and `Offer::unwrapNFTAndSetVerificationTimeout` functions, effectively causing the NFTs to be lost.

Example:

contracts/protocol/facets/Offer.sol

SOL

```sol
802 // wrap NFTs
803 _bosonVoucher.setApprovalForAll(wrapperAddress, true);
804 wrapperAddress.wrap(
805     _startingNFTId,
806     _quantity,
807     _wrapType == FermionTypes.WrapType.OS_AUCTION ? msgSender : wrapperAddress
808 );
809 _bosonVoucher.setApprovalForAll(wrapperAddress, false);
```

Recommendation:

We advise the conditional to be corrected, minting the NFT to the `msgSender` solely in the `OS_FIXED_PRICE` wrap type.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The Fermion Protocol team evaluated this exhibit and clarified that the fund and NFT flows are being executed as expected in the `OS_FIXED_PRICE` scenario.

After evaluating the statements of the Fermion Protocol team, we confirmed that the fund flows are correct and 1st consider this exhibit invalid.

ORE-07M: Inexistent Validation of Successful Redemption of Twins

Type	Severity	Location
Logical Fault	Medium	Offer.sol:L440

Description:

The redemption of a voucher may result in an automatic dispute being raised for this particular voucher due to the failure of a twin's redemption or an out-of-gas error.

Such a case is not handled properly by the system, and instead moves the voucher to the validation phase, including configuring the item verification timeout even though verification should not begin yet.

Impact:

The system will incorrectly proceed to the item verification phase even though the voucher might have raised a dispute internally within the Boson Protocol system.

Example:

contracts/protocol/facets/Offer.sol

SOL

```sol
414 {
415     address exchangeToken = offer.exchangeToken;
416
417     // Check the caller is the seller's assistant
418     uint256 sellerId = offer.sellerId;
419     EntityLib.validateSellerAssistantOrFacilitator(sellerId,
offer.facilitatorId);
420     handleBosonSellerDeposit(sellerId, exchangeToken, offer.sellerDeposit);
421
422     // WrapType wrapType = _wrapType;
423     deriveAndValidatePriceDiscoveryData(tokenId, _priceDiscovery, exchangeToken,
_data);
```

Example (Cont.):

```sol

    uint256 bosonProtocolFee = getBosonProtocolFee(exchangeToken, _priceDiscovery.price);
    (uint256 fermionFeeAmount, uint256 facilitatorFeeAmount) = FeeLib.calculateAndValidateFees(
        _priceDiscovery.price,
        bosonProtocolFee,
        offer
    );

    // Store item full price along with all fees
    tokenLookups.itemPrice = _priceDiscovery.price;
    tokenLookups.bosonProtocolFee = bosonProtocolFee;
    tokenLookups.fermionFeeAmount = fermionFeeAmount;
    tokenLookups.verifierFee = offer.verifierFee;
    tokenLookups.facilitatorFeeAmount = facilitatorFeeAmount;

    BOSON_PROTOCOL.commitToPriceDiscoveryOffer(payable(address(this)), tokenId, _priceDiscovery);
    BOSON_PROTOCOL.redeemVoucher(tokenId & type(uint128).max); // Exchange id is in the lower 128 bits
}

// Verification timeout logic
uint256 itemVerificationTimeout;
uint256 maxItemVerificationTimeout;
{
    FermionStorage.ProtocolConfig storage pc = FermionStorage.protocolConfig();
    maxItemVerificationTimeout = block.timestamp + pc.maxVerificationTimeout;
    if (_verificationTimeout == 0) {
        itemVerificationTimeout = block.timestamp + pc.defaultVerificationTimeout;
    } else {
```

Example (Cont.):

```sol
        if (_verificationTimeout > maxItemVerificationTimeout) {
            revert VerificationErrors.VerificationTimeoutTooLong(
                _verificationTimeout,
                maxItemVerificationTimeout
            );
        }
        itemVerificationTimeout = _verificationTimeout;
    }
    tokenLookups.itemVerificationTimeout = itemVerificationTimeout;
    tokenLookups.itemMaxVerificationTimeout = maxItemVerificationTimeout;
}
```

Recommendation:

We advise this scenario to be handled properly by the code, either by not configuring the item verification timeout until the dispute is resolved or by mandating that the redemption of a voucher succeeds, either of which we consider valid.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The Fermion Protocol team evaluated this exhibit and clarified that since the Fermion Protocol is not able to introduce a twin or create a bundle for a particular offer, a twin redemption will never be performed and as such a dispute will never be automatically raised in that way the exhibit describes.

As a result, we consider this exhibit to be inapplicable.

Pause Manual Review Findings

PES-01M: Improper Unpause Restriction

Type	Severity	Location
Input Sanitization	Minor	**Pause.sol**: • I-1: **L29-L34** • I-2: **L45-L53**

Description:

The `PauseFacet::pause` and `PauseFacet::unpause` functions will permit duplicate regions to be specified.

Additionally, the `PauseFacet::unpause` function will permit an unpause of a region that is not paused to be processed.

Impact:

It is presently possible to specify the same region multiple times when processing pause operations and it is possible for an unpause operation to succeed despite what the `NotPaused` error would entail.

Example:

contracts/protocol/facets/Pause.sol

SOL

```
19  /**
20   * @notice Pauses some or all of the protocol.
21   *
22   * Emits a ProtocolPaused event if successful.
23   *
24   * Reverts if:
25   * - Caller is not the protocol pauser
26   *
27   * @param _regions - an array of regions to pause. See:
{FermionTypes.PausableRegion}
28   */
```

Example (Cont.):

```sol
29  function pause(FermionTypes.PausableRegion[] calldata _regions) external
onlyRole(PAUSER) {
30      togglePause(_regions, true);
31
32      // Notify watchers of state change
33      emit ProtocolPaused(_regions);
34  }
35
36  /**
37   * @notice Unpauses the protocol.
38   *
39   * Emits a ProtocolUnpaused event if successful.
40   *
41   * Reverts if:
42   * - Caller is not the protcol pauser
43   * - Protocol is not currently paused
44   */
45  function unpause(FermionTypes.PausableRegion[] calldata _regions) external
onlyRole(PAUSER) {
46      // Make sure the protocol is paused
47      if (FermionStorage.protocolStatus().paused == 0) revert NotPaused();
48
49      togglePause(_regions, false);
50
51      // Notify watchers of state change
52      emit ProtocolUnpaused(_regions);
53  }
```

Recommendation:

We advise the code to prevent duplicate regions from being specified by validating the existing status of each region, addressing both issues simultaneously.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The Fermion Protocol team evaluated this exhibit and specified that they wish to acknowledge it and that they will consider introducing this restriction in the future.

Royalties Manual Review Findings

RSE-01M: Incorrect Break Logic

Type	Severity	Location
Logical Fault	Minor	Royalties.sol:L108

Description:

The `Royalties::getRoyalties` function will attempt to replace the `address(0)` with the `defaultRecipient` of a particular `_tokenId`, however, it will `break` after identifying the first `address(0)` instance even though duplicate entries are permitted in the `royaltyInfo` of a particular offer.

Impact:

While the vulnerability is correct, the likelihood of a user introducing the zero-address twice in the same royalty information configuration is low rendering this issue to be minor.

Example:

contracts/protocol/facets/Royalties.sol

SOL

```sol
99  function getRoyalties(
100     uint256 _tokenId
101 ) external view returns (address payable[] memory recipients, uint256[] memory
bps) {
102     (FermionTypes.RoyaltyInfo memory royaltyInfo, address defaultRecipient) =
fetchRoyalties(_tokenId);
103
104     // replace default recipient with the default recipient (admin) address
105     for (uint256 i; i < royaltyInfo.recipients.length; ++i) {
106         if (royaltyInfo.recipients[i] == address(0)) {
107             royaltyInfo.recipients[i] = payable(defaultRecipient);
108             break;
```

Example (Cont.):

SOL

```
109         }
110     }
111
112     return (royaltyInfo.recipients, royaltyInfo.bps);
113 }
```

Recommendation:

We advise the code to not `break` early, ensuring all `address(0)` royalty recipients are properly replaced.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The `break` statement was removed ensuring all royalty recipients are properly assigned.

Access Code Style Findings

ASS-01C: Inefficient Import Paths

Type	Severity	Location
Code Style	Informational	Access.sol:L8, L9

Description:

The referenced `import` paths will traverse two levels upward and will re-enter the same folders.

Example:

contracts/protocol/bases/mixins/Access.sol

SOL

```sol
8  import { Context } from "../../bases/mixins/Context.sol";
9  import { ReentrancyGuard } from "../../bases/mixins/ReentrancyGuard.sol";
```

Recommendation:

We advise relative imports to be utilized instead, directly importing from the `./Context.sol` and `./ReentrancyGuard.sol` files.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The `import` paths have been corrected as advised.

ChainlinkPriceOracle Code Style Findings

CPO-01C: Inefficient Event Emissions

Type	Severity	Location
Gas Optimization	Informational	**ChainlinkPriceOracle.sol**: • I-1: **L64**, **L66** • I-2: **L75**, **L77**

Description:

The referenced event emissions are inefficient as they will store a temporary local variable redundantly.

Example:

contracts/protocol/clients/oracle/ChainlinkPriceOracle.sol

SOL

```sol
63  function setFeed(address _feed) external onlyOwner {
64      address previousFeed = feed;
65      _setFeedInternal(_feed);
66      emit FeedUpdated(previousFeed, _feed);
67  }
```

Recommendation:

We advise the event emissions to precede the state mutations, permitting the relevant storage entries to be utilized directly in the emission statement efficiently (i.e. `emit FeedUpdated(feed, _feed)`).

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The `event` emissions have been optimized per hour recommendation.

Common Code Style Findings

CNO-01C: Redundant Import of Logic Implementation

Type	Severity	Location
Code Style	Informational	Common.sol:L8

Description:

The referenced `import` operation is inefficient as it imports a logic implementation that is utilized as an `interface`.

Example:

contracts/protocol/clients/Common.sol

SOL

```sol
8  import { FermionFractionsERC20 } from "./FermionFractionsERC20.sol";
```

Recommendation:

We advise a proper `interface` to be introduced for the `FermionFractionsERC20` implementation that is consequently imported, optimizing the code's syntax.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The code was updated to properly utilize an `interface` instead of a logic implementation as advised.

Constants Code Style Findings

CST-01C: Generic Typographic Mistake

Type	Severity	Location
Code Style	Informational	Constants.sol:L38

Description:

The referenced line contains a typographical mistake (i.e. `private` variable without an underscore prefix, a non-`snake_case` module) or generic documentational error (i.e. copy-paste) that should be corrected.

Example:

contracts/protocol/domain/Constants.sol

```sol
38  // OpenSea
```

Recommendation:

We advise this to be done so to enhance the legibility of the codebase.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The referenced outdated documentation line has been removed addressing this exhibit.

Context Code Style Findings

CTX-01C: Redundant Constant Declaration

Type	Severity	Location
Code Style	Informational	Context.sol:L12

Description:

The referenced `constant` declaration is `private` and remains unused in the `Context` contract's context.

Example:

contracts/protocol/bases/mixins/Context.sol

SOL

```sol
contract Context {
    uint256 private constant ADDRESS_LENGTH = 20;

    /**
     * @notice Returns the message sender address.
     *
     * @dev Could be msg.sender or the message sender address from storage (in case of meta transaction).
     *
     * @return the message sender address
     */
```

Example (Cont.):

```sol
    function _msgSender() internal view virtual returns (address) {
        return ContextLib._msgSender();
    }
}
```

Recommendation:

We advise it to be omitted, minimizing the code's redundancy.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The `constant` declaration has been removed per our recommendation.

Custody Code Style Findings

CYD-01C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	Informational	**Custody.sol**: • I-1: **L93** • I-2: **L105** • I-3: **L107**

Description:

The linked mathematical operations are guaranteed to be performed safely by surrounding conditionals evaluated in either `require` checks or `if-else` constructs.

Example:

contracts/protocol/bases/mixins/Custody.sol

SOL

```sol
101 if (custodianPayoff + custodianFee.amount > balance) {
102     // In case of external fractionalisation, the caller can provide additional
funds to cover the custodian fee. If not enough, revert.
103     if (_externalCall) {
104         // Full custodian payoff must be paid in order to fractionalise
105         uint256 diff = custodianPayoff + custodianFee.amount - balance;
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.X`, we advise the linked statements to be wrapped in `unchecked` code blocks thereby optimizing their execution cost.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

All arithmetic operations referenced have been wrapped in an `unchecked` code block optimizing their gas cost.

CustodyVault Code Style Findings

CVT-01C: Incorrect Revert Documentation

Type	Severity	Location
Code Style	Informational	CustodyVault.sol:L562

Description:

The referenced condition under which the function is expected to revert is not fulfilled.

Example:

contracts/protocol/facets/CustodyVault.sol

SOL

```sol
561 * Reverts if:
562  * - Custody region is paused
563  * - Call is external and caller is not the F-NFT contract owning the token
564  *
565  * @param _firstTokenId - the lowest token ID to add to the vault
566  * @param _length - the number of tokens to add to the vault
567  * @param _depositAmount - the amount to deposit
568  * @param _depositAmount - the amount to deposit
569  * @param _custodianVaultParameters - the custodian vault parameters
570  * @param _externalCall - if true, the caller is checked to be the F-NFT contract
owning the token. Use false for internal calls.
```

Example (Cont.):

```sol
 * @return returnedAmount - the amount returned to the caller
 */
function setupCustodianOfferVault(
    uint256 _firstTokenId,
    uint256 _length,
    FermionTypes.CustodianVaultParameters memory _custodianVaultParameters,
    uint256 _depositAmount,
    bool _externalCall
) internal returns (uint256 returnedAmount) {
```

Recommendation:

We advise it to be omitted as it is effectively imposed by parent call chains that ultimately invoke the `CustodyVaultFacet::setupCustodianItemVault` function.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The Fermion Protocol team evaluated this exhibit and clarified that the revert condition occurs internally.

As such, we consider this exhibit nullified.

CVT-02C: Inefficient Re-Assignment of Vault Parameters

Type	Severity	Location
Gas Optimization	Informational	CustodyVault.sol:L508

Description:

The vault parameters are solely expected to change when `itemsInVault` are `0` as a new vault is created yet the `vaultParameters` variable is re-assigned unconditionally.

Furthermore, the assignment should not be necessary at all given that the `vaultParameters` variable is a `storage` pointer and thus consistently points to updated data entries.

Example:

contracts/protocol/facets/CustodyVault.sol

SOL

```
459 FermionTypes.CustodianVaultParameters storage vaultParameters =
offerLookups.custodianVaultParameters;
460
461 if (fractionAuction.endTime > block.timestamp) revert AuctionOngoing(_offerId,
fractionAuction.endTime);
462
463 uint256 fractionsToIssue;
464 uint256 itemsInVault = offerLookups.custodianVaultItems;
465 address fermionFNFTAddress = offerLookups.fermionFNFTAddress;
466 if (!_isOfferVault) {
467     // Forceful fractionalisation
468     if (itemsInVault > 0) {
```

Example (Cont.):

```sol
        // vault exist already
        fermionFNFTAddress.mintFractions(_tokenId, 1, 0);
        addItemToCustodianOfferVault(_tokenId, 1, 0, false, pl);
    } else {
        // no vault yet. Use the default parameters
        FermionTypes.CustodianVaultParameters memory _custodianVaultParameters;
        {
            FermionTypes.BuyoutAuctionParameters memory _buyoutAuctionParameters;
            FermionStorage.TokenLookups storage tokenLookups =
pl.tokenLookups[_tokenId];
            _buyoutAuctionParameters.exitPrice = tokenLookups.selfSaleItemPrice !=
0
                ? tokenLookups.selfSaleItemPrice
                : tokenLookups.itemPrice;
            uint256 partialAuctionThreshold = PARTIAL_THRESHOLD_MULTIPLIER *
_custodianFee.amount;
            uint256 newFractionsPerAuction = (partialAuctionThreshold *
DEFAULT_FRACTION_AMOUNT) /
                _buyoutAuctionParameters.exitPrice;
            _custodianVaultParameters = FermionTypes.CustodianVaultParameters({
                partialAuctionThreshold: partialAuctionThreshold,
                partialAuctionDuration: _custodianFee.period /
PARTIAL_AUCTION_DURATION_DIVISOR,
                liquidationThreshold: LIQUIDATION_THRESHOLD_MULTIPLIER *
_custodianFee.amount,
                newFractionsPerAuction: newFractionsPerAuction
            });

            fermionFNFTAddress.mintFractions(
                _tokenId,
                1,
                DEFAULT_FRACTION_AMOUNT,
                _buyoutAuctionParameters,
                _custodianVaultParameters,
```

Example (Cont.):

```sol
                0,
                address(0)
            );
        }
        // set the offer vault period in the past, so the auction covers the past expenses, too
        // Since this is the first fractionalisation, offer vault period should match the item vault period
        pl.tokenLookups[_offerId].vault.period = pl.tokenLookups[_tokenId].vault.period;
        setupCustodianOfferVault(_tokenId, 1, _custodianVaultParameters, 0, false);
    }

    itemsInVault++;
    vaultParameters = offerLookups.custodianVaultParameters;
```

Recommendation:

We advise the code to be optimized either by relocating the assignment to the `else` branch that precedes it or by omitting the assignment altogether.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The assignment has been omitted altogether as advised.

Entity Code Style Findings

EYT-01C: Generic Typographic Mistakes

Type	Severity	Location
Code Style	Informational	**Entity.sol**: • I-1: **L343** • I-2: **L547**

Description:

The referenced lines contain typographical mistakes (i.e. `private` variable without an underscore prefix) or generic documentational errors (i.e. copy-paste) that should be corrected.

Example:

contracts/protocol/facets/Entity.sol

```sol
343 * - Caller is not a account for any enitity
```

Recommendation:

We advise them to be corrected enhancing the legibility of the codebase.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The reference typographic errors have been corrected.

EYT-02C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	Informational	Entity.sol:L740, L741

Description:

The linked mathematical operation is guaranteed to be performed safely by logical inference, such as surrounding conditionals evaluated in `require` checks or `if-else` constructs.

Example:

contracts/protocol/facets/Entity.sol

SOL

```sol
738 for (uint256 j; j < facilitatorsLength; ++j) {
739     if (associatedEntities[j] == associatedEntityId) {
740         if (j != facilitatorsLength - 1)
741             associatedEntities[j] = associatedEntities[facilitatorsLength - 1];
742         associatedEntities.pop();
743 
744         emit AssociatedEntityRemoved(_associatedRole, _sellerId,
associatedEntityId);
745         break;
746     }
747 }
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.X`, we advise the linked statement to be wrapped in an `unchecked` code block thereby optimizing its execution cost.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

Both arithmetic operations have been wrapped in an `unchecked` code block, optimizing the codebase.

EYT-03C: Inexplicable Literal Enum Cast

Type	Severity	Location
Code Style	Informational	Entity.sol:L589

Description:

The referenced operation will cast the `0` literal to the `FermionTypes::EntityRole` enum which is illegible.

Example:

contracts/protocol/facets/Entity.sol

SOL

```
589 FermionTypes.EntityRole(0),
```

Recommendation:

We advise the `FermionTypes.EntityRole.Seller` value to be accessed directly, optimizing the code's legibility.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The code was updated to utilize a newly introduced `ANY_ENTITY_ROLE` constant therby addressing this exhibit.

EYT-04C: Optimization of Modifier Application

Type	Severity	Location
Gas Optimization	Informational	Entity.sol:L73, L99, L120-L126

Description:

The two referenced functions apply the exact same `modifier` implementations and invoke the same `internal` function.

Example:

contracts/protocol/facets/Entity.sol

SOL

```sol
/**
 * @notice Add entity accounts.
 *
 * Each address can have multiple account roles from FermionTypes.AccountRole
 * For each role that the entity has, the account roles are set independently.
 *
 * Emits an EntityAccountAdded event if successful.
 *
 * Reverts if:
 * - Entity region is paused
```

Example (Cont.):

```sol
  * - Entity does not exist
  * - Caller is not an admin for the entity role
  * - Length of _accounts, _entityRoles and _accountRoles do not match
  * - Entity does not have the role
  *
  * @param _accounts - list of accounts that acts on the seller's behalf
  * @param _entityRoles - list of corresponding roles, for which the address is given a certain account role. If entityRoles[i] is empty, the address is given the account role to all entity roles.
  * @param _accountRoles - list of account roles for each account and entity role
  */
 function addEntityAccounts(
     uint256 _entityId,
     address[] calldata _accounts,
     FermionTypes.EntityRole[][] calldata _entityRoles,
     FermionTypes.AccountRole[][][] calldata _accountRoles
 ) external notPaused(FermionTypes.PausableRegion.Entity) nonReentrant {
     addOrRemoveEntityAccounts(_entityId, _accounts, _entityRoles, _accountRoles, true);
 }

 /**
  * @notice Remove entity accounts.
  *
  * Emits an EntityAccountRemoved event if successful.
  *
  * Reverts if:
  * - Entity region is paused
  * - Entity does not exist
  * - Caller is not the admin for the entity role
  * - Length of _accounts, _entityRoles and _accountRoles do not match
```

Example (Cont.):

```sol
    * - Entity does not have the role
    *
    * @param _entityId - the entity ID
    * @param _accounts - list of accounts that acts on the seller's behalf
    * @param _entityRoles - list of corresponding roles, for which the address is given a certain account role. If entityRoles[i] is empty, the address is given the account role to all entity roles.
    * @param _accountRoles - list of account roles for each account and entity role
    */
    function removeEntityAccounts(
        uint256 _entityId,
        address[] calldata _accounts,
        FermionTypes.EntityRole[][] calldata _entityRoles,
        FermionTypes.AccountRole[][][] calldata _accountRoles
    ) external notPaused(FermionTypes.PausableRegion.Entity) nonReentrant {
        addOrRemoveEntityAccounts(_entityId, _accounts, _entityRoles, _accountRoles, false);
    }
```

Recommendation:

We advise the modifiers to be relocated to the `internal` function, optimizing the contract's bytecode size.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The `modifier` applications have been relocated to the common internal function optimizing the bytecode size of the contract.

FeeLib Code Style Findings

FLB-01C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	Informational	FeeLib.sol:L36, L43

Description:

The linked mathematical operation is guaranteed to be performed safely by logical inference, such as surrounding conditionals evaluated in `require` checks or `if-else` constructs.

Example:

contracts/protocol/libs/FeeLib.sol

SOL

```sol
35  if (priceRangesLength > 0) {
36      for (uint256 i; i < priceRangesLength - 1; ++i) {
37          if (_price <= priceRanges[i]) {
38              // Return the fee percentage for the matching price range
39              return feePercentages[i];
40          }
41      }
42      // If price exceeds all ranges, use the highest fee percentage
43      return feePercentages[priceRangesLength - 1];
44  }
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.X`, we advise the linked statement to be wrapped in an `unchecked` code block thereby optimizing its execution cost.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The first arithmetic operation has been wrapped in an `unchecked` code block whereas the second instance is no longer present in the codebase.

As such, we consider this exhibit addressed.

FermionFNFTBase Code Style Findings

FFT-01C: Non-Standard Reservation of Storage Slot

Type	Severity	Location
Standard Conformity	Informational	FermionFNFTBase.sol:L19

Description:

The referenced storage slot reservation is improper as it will reserve `160` bits instead of the full `256` bit slot, resulting in downstream errors if the `FermionWrapper` derivative contract adjusts its storage variables.

Example:

contracts/protocol/clients/FermionFNFTBase.sol

SOL

```
19  address internal emptySlot; // former fermionProtocol. Keep it to keep
`voucherAddress` in slot 1.
```

Recommendation:

We advise the full slot to be consumed, specifying the data type as `uint256`, `bytes32`, or a similar data type that will occupy the full slot.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The data type of the referenced variable was updated to `uint256` as advised.

FermionFNFTPriceManager Code Style Findings

FFP-01C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	Informational	FermionFNFTPriceManager.sol:L427

Description:

The linked mathematical operation is guaranteed to be performed safely by logical inference, such as surrounding conditionals evaluated in `require` checks or `if-else` constructs.

Example:

contracts/protocol/clients/FermionFNFTPriceManager.sol

SOL

```sol
423 if (remainingBalance >= voteCount) {
424     return;
425 }
426
427 uint256 votesToRemove = voteCount - remainingBalance;
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.X`, we advise the linked statement to be wrapped in an `unchecked` code block thereby optimizing its execution cost.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The referenced arithmetic operation has been properly wrapped in an `unchecked` code block as advised.

FFP-02C: Redundant Named Argument

Type	Severity	Location
Gas Optimization	Informational	FermionFNFTPriceManager.sol:L397

Description:

The `amount` argument that is explicitly named in the `FermionFNFTPriceManager::adjustVotesOnTransfer` function is redundant and unused.

Example:

contracts/protocol/clients/FermionFNFTPriceManager.sol

SOL

```sol
397 function adjustVotesOnTransfer(address from, uint256 amount) external {
```

Recommendation:

We advise its explicit name to be omitted, optimizing the code's gas cost as no memory is reserved for it.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The `amount` argument has been properly removed as advised.

FermionFractionsMint Code Style Findings

FFM-01C: Optimization of Migration Statements

Type	Severity	Location
Gas Optimization	Informational	FermionFractionsMint.sol:L252, L261-L264

Description:

The `FermionFractionsMint::migrateFractions` function is slightly inefficient as it will contain the same statement across two distinct `if-else` clauses.

Example:

contracts/protocol/clients/FermionFractionsMint.sol

SOL

```
249 address cloneAddress;
250 if (fractionStorage.epochToClone.length == 0) {
251     _advanceEpoch();
252     cloneAddress = fractionStorage.epochToClone[0];
253
254     uint256 fractionBalance = $._balances[address(this)];
255     if (fractionBalance > 0) {
256         FermionFractionsERC20(cloneAddress).mint(address(this), fractionBalance);
257         emit FractionsMigrated(address(this), fractionBalance);
258     }
```

Example (Cont.):

```sol
    fractionStorage.migrated[address(this)] = true;
} else {
    if (_owners.length == 0) {
        revert InvalidLength();
    }
    cloneAddress = fractionStorage.epochToClone[0];
}
```

Recommendation:

We advise the `cloneAddress` assignment to be relocated after the `if-else` structure, and the inner `if` of the `else` clause to be merged to it optimizing the code.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The Fermion Protocol team evaluated this exhibit and clarified that the optimization recommended is inapplicable because of the `true` execution branch of the referenced `if-else` clause as it will affect the `cloneAddress` assignment when the epoch is advanced.

Funds Code Style Findings

FSD-01C: Non-Standard Explicit Contract Invocations

Type	Severity	Location
Code Style	Informational	**Funds.sol**: • I-1: **L276** • I-2: **L280**

Description:

The referenced statements represent non-standard `contract` invocations that are not consistent with the rest of the codebase.

Example:

contracts/protocol/facets/Funds.sol

SOL

```sol
for (uint256 i; i < recipientsLength; i++) {
    uint256 _entityId;
    if (royaltyInfo.recipients[i] == address(0)) {
        _entityId = offer.sellerId;
    } else {
        _entityId = EntityLib.getOrCreateEntityId(
            royaltyInfo.recipients[i],
            FermionTypes.EntityRole.RoyaltyRecipient,
            pl
        );
```

Example (Cont.):

```sol
    }

    uint256 amount = MathLib.applyPercentage(_saleProceeds, royaltyInfo.bps[i]);
    royalties += amount;

    FundsManager.increaseAvailableFunds(_entityId, tokenAddress, amount);
}

// return the remainder
FundsManager.transferERC20FromProtocol(tokenAddress, payable(msg.sender),
_saleProceeds - royalties);
```

Recommendation:

We advise the referenced functions to be invoked without specifying the `contract` name explicitly, optimizing the code's style.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The code style of the reference function invocations was revised as advised.

FundsManager Code Style Findings

FMR-01C: Generic Typographic Mistake

Type	Severity	Location
Code Style	Informational	FundsManager.sol:L178

Description:

The referenced line contains a typographical mistake (i.e. `private` variable without an underscore prefix, a non-`snake_case` module) or generic documentational error (i.e. copy-paste) that should be corrected.

Example:

contracts/protocol/bases/mixins/FundsManager.sol

```sol
178 * @notice Checks if the contract at the token address is FNFN or not.
```

Recommendation:

We advise this to be done so to enhance the legibility of the codebase.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The reference typographic error has been corrected.

FMR-02C: Ineffectual Usage of Safe Arithmetics

Type	Severity	Location
Language Specific	Informational	FundsManager.sol:L236

Description:

The linked mathematical operation is guaranteed to be performed safely by logical inference, such as surrounding conditionals evaluated in `require` checks or `if-else` constructs.

Example:

contracts/protocol/bases/mixins/FundsManager.sol

SOL

```sol
226 uint256 index = entityTokens[_tokenAddress] - 1;
227
228 // if target is last index then only pop and delete are needed
229 // otherwise, we overwrite the target with the last token first
230 if (index != lastTokenIndex) {
231     // Need to fill gap caused by delete if more than one element in storage
array
232     address tokenToMove = tokenList[lastTokenIndex];
233     // Copy the last token in the array to this index to fill the gap
234     tokenList[index] = tokenToMove;
235     // Reset index mapping. Should be index in tokenList array + 1
```

Example (Cont.):

```sol
236     entityTokens[tokenToMove] = index + 1;
```

Recommendation:

Given that safe arithmetics are toggled on by default in `pragma` versions of `0.8.X`, we advise the linked statement to be wrapped in an `unchecked` code block thereby optimizing its execution cost.

Alleviation (c332d7a9ee):

An `unchecked` code block has been introduced albeit wrapping all arithmetic operations rather than just the one referenced by the exhibit rendering this exhibit partially addressed.

Alleviation (de8e96c89c):

The Fermion Protocol team clarified that they consider the `unchecked` code block applicable to all operations within it due to constraints the code enforces earlier in the code segment.

We confirmed that the operations are indeed inferred to be secure by preceding statements and thus consider this optimization properly applied.

MetaTransaction Code Style Findings

MTN-01C: Generic Typographic Mistake

Type	Severity	Location
Code Style	Informational	MetaTransaction.sol:L94

Description:

The referenced line contains a typographical mistake (i.e. `private` variable without an underscore prefix, a non-`snake_case` module) or generic documentational error (i.e. copy-paste) that should be corrected.

Example:

contracts/protocol/facets/MetaTransaction.sol

SOL

```
94  * If epoch is -1, returns the address of the FermionFNFT contract.
```

Recommendation:

We advise this to be done so to enhance the legibility of the codebase.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The documentation referenced was corrected to reflect the actual implementation of the function that follows it.

Offer Code Style Findings

ORE-01C: Generic Typographic Mistake

Type	Severity	Location
Code Style	Informational	Offer.sol:L769

Description:

The referenced line contains a typographical mistake (i.e. `private` variable without an underscore prefix, a non-`snake_case` module) or generic documentational error (i.e. copy-paste) that should be corrected.

Example:

contracts/protocol/facets/Offer.sol

SOL

```sol
769 function wrapNFTS(
```

Recommendation:

We advise this to be done so to enhance the legibility of the codebase.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The reference typographic error has been corrected.

ORE-02C: Redundant Assignment of Boson Seller ID

Type	Severity	Location
Gas Optimization	Informational	Offer.sol:L102

Description:

The referenced assignment is already performed within the Boson Protocol creation offer mechanism and thus is redundant.

Example:

contracts/protocol/facets/Offer.sol

SOL

```sol
101 IBosonProtocol.Offer memory bosonOffer;
102 bosonOffer.sellerId = bosonSellerId;
```

Recommendation:

We advise it to be omitted, optimizing the code's gas cost.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The redundant assignment of the `sellerId` has been removed as advised.

ORE-03C: Redundant Input Address Validation

Type	Severity	Location
Gas Optimization	Informational	Offer.sol:L39

Description:

The referenced input address validation is redundant as a call that would fail if no code exists is performed within the same block.

Example:

contracts/protocol/facets/Offer.sol

SOL

```sol
constructor(address _bosonProtocol) {
    if (_bosonProtocol == address(0)) revert FermionGeneralErrors.InvalidAddress();

    BOSON_PROTOCOL = IBosonProtocol(_bosonProtocol);
    BOSON_TOKEN = IBosonProtocol(_bosonProtocol).getTokenAddress();
}
```

Recommendation:

We advise it to be omitted, optimizing the code's gas cost.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The reference to input validation has been removed as advised.

ReentrancyGuard Code Style Findings

RGD-01C: Generic Typographic Mistake

Type	Severity	Location
Code Style	Informational	ReentrancyGuard.sol:L14

Description:

The referenced line contains a typographical mistake (i.e. `private` variable without an underscore prefix, a non-`snake_case` module) or generic documentational error (i.e. copy-paste) that should be corrected.

Example:

contracts/protocol/bases/mixins/ReentrancyGuard.sol

SOL

```
14  // NB: it's more optiomal to compare msg.sender to address(this) twice than storing it in a variable (e.g. _isSelf)
```

Recommendation:

We advise this to be done so to enhance the legibility of the codebase.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The reference typographic error has been corrected.

RGD-02C: Repetitive Value Literals

Type	Severity	Location
Code Style	Informational	ReentrancyGuard.sol:L23, L31

Description:

The linked value literals are repeated across the codebase multiple times.

Example:

contracts/protocol/bases/mixins/ReentrancyGuard.sol

SOL

```
23  tstore(GUARD_SLOT, 1)
```

Recommendation:

We advise each to be set to its dedicated `constant` variable instead, optimizing the legibility of the codebase.

In case some of the `constant` declarations have already been introduced, we advise them to be properly re-used across the code.

Alleviation (c332d7a9eee7ec78a1641122325f2075e6f6d49f):

The referenced value literals `1`, and `0` have all been relocated to contract-level `constant` declarations labelled `GUARD_LOCKED`, and `GUARD_UNLOCKED` respectively, optimizing the code's legibility.

Finding Types

A description of each finding type included in the report can be found below and is linked by each respective finding. A full list of finding types Omniscia has defined will be viewable at the central audit methodology we will publish soon.

Input Sanitization

As there are no inherent guarantees to the inputs a function accepts, a set of guards should always be in place to sanitize the values passed in to a particular function.

Indeterminate Code

These types of issues arise when a linked code segment may not behave as expected, either due to mistyped code, convoluted if blocks, overlapping functions / variable names and other ambiguous statements.

Language Specific

Language specific issues arise from certain peculiarities that the Circom language boasts that discerns it from other conventional programming languages.

Curve Specific

Circom defaults to using the BN128 scalar field (a 254-bit prime field), but it also supports BSL12-381 (which has a 255-bit scalar field) and Goldilocks (with a 64-bit scalar field). However, since there are no constants denoting either the prime or the prime size in bits available in the Circom language, some Circomlib templates like `Sign` (which returns the sign of the input signal), and `AliasCheck` (used by the strict versions of `Num2Bits` and `Bits2Num`), hardcode either the BN128 prime size or some other constant related to BN128. Using these circuits with a custom prime may thus lead to unexpected results and should be avoided.

Code Style

In these types of findings, we identify whether a project conforms to a particular naming convention and whether that convention is consistent within the codebase and legible. In case of inconsistencies, we point them out under this category. Additionally, variable shadowing falls under this category as well which is identified when a local-level variable contains the same name as a toplevel variable in the circuit.

Mathematical Operations

This category is used when a mathematical issue is identified. This implies an issue with the implementation of a calculation compared to the specifications.

Logical Fault

This category is a bit broad and is meant to cover implementations that contain flaws in the way they are implemented, either due to unimplemented functionality, unaccounted-for edge cases or similar extraordinary scenarios.

Privacy Concern

This category is used when information that is meant to be kept private is made public in some way.

Proof Concern

Under-constrained signals are one of the most common issues in zero-knowledge circuits. Issues with proof generation fall under this category.

Severity Definition

In the ever-evolving world of blockchain technology, vulnerabilities continue to take on new forms and arise as more innovative projects manifest, new blockchain-level features are introduced, and novel layer-2 solutions are launched. When performing security reviews, we are tasked with classifying the various types of vulnerabilities we identify into subcategories to better aid our readers in understanding their impact.

Within this page, we will clarify what each severity level stands for and our approach in categorizing the findings we pinpoint in our audits. To note, all severity assessments are performed **as if the contract's logic cannot be upgraded** regardless of the underlying implementation.

Severity Levels

There are five distinct severity levels within our reports; `unknown`, `informational`, `minor`, `medium`, and `major`. A TL;DR overview table can be found below as well as a dedicated chapter to each severity level:

	Impact (None)	Impact (Low)	Impact (Moderate)	Impact (High)
Likelihood (None)	Informational	Informational	Informational	Informational
Likelihood (Low)	Informational	Minor	Minor	Medium
Likelihood (Moderate)	Informational	Minor	Medium	Major
Likelihood (High)	Informational	Medium	Major	Major

Unknown Severity

The `unknown` severity level is reserved for misbehaviors we observe in the codebase that cannot be quantified using the above metrics. Examples of such vulnerabilities include potentially desirable system behavior that is undocumented, reliance on external dependencies that are out-of-scope but could result in some form of vulnerability arising, use of external out-of-scope contracts that appears incorrect but cannot be pinpointed, and other such vulnerabilities.

In general, `unknown` severity level vulnerabilities require follow-up information by the project being audited and are either adjusted in severity (if valid), or marked as nullified (if invalid).

Additionally, the `unknown` severity level is sometimes assigned to centralization issues that cannot be assessed in likelihood due to their exploitation being tied to the honesty of the project's team.

Informational Severity

The `informational` severity level is dedicated to findings that do not affect the code functionally and tend to be stylistic or optimizational in nature. Certain edge cases are also set under `informational` vulnerabilities, such as overflow operations that will not manifest in the lifetime of the contract but should be guarded against as a best practice, to give an example.

Minor Severity

The `minor` severity level is meant for vulnerabilities that require functional changes in the code but tend to either have little impact or be unlikely to be recreated in a production environment. These findings can be acknowledged except for findings with a moderate impact but low likelihood which must be alleviated.

Medium Severity

The `medium` severity level is assigned to vulnerabilities that must be alleviated and have an observable impact on the overall project. These findings can only be acknowdged if the project deems them desirable behavior and we disagree with their point-of-view, instead urging them to reconsider their stance while marking the exhibit as acknowledged given that the project has ultimate say as to what vulnerabilities they end up patching in their system.

Major Severity

The `major` severity level is the maximum that can be specified for a finding and indicates a significant flaw in the code that must be alleviated.

Likelihood & Impact Assessment

As the preface chapter specifies, the blockchain space is constantly reinventing itself meaning that new vulnerabilities take place and our understanding of what security means differs year-to-year.

In order to reliably assess the likelihood and impact of a particular vulnerability, we instead apply an abstract measurement of a vulnerability's impact, duration the impact is applied for, and probability that the vulnerability would be exploited in a production environment.

Our proposed definitions are inspired by multiple sources in the security community and are as follows:

- Impact (High): A core invariant of the protocol can be broken for an extended duration.
- Impact (Moderate): A non-core invariant of the protocol can be broken for an extended duration or at scale, or an otherwise major-severity issue is reduced due to hypotheticals or external factors affecting likelihood.
- Impact (Low): A non-core invariant of the protocol can be broken with reduced likelihood or impact.
- Impact (None): A code or documentation flaw whose impact does not achieve low severity, or an issue without theoretical impact; a valuable best-practice
- Likelihood (High): A flaw in the code that can be exploited trivially and is ever-present.
- Likelihood (Moderate): A flaw in the code that requires some external factors to be exploited that are likely to manifest in practice.
- Likelihood (Low): A flaw in the code that requires multiple external factors to be exploited that may manifest in practice but would be unlikely to do so.
- Likelihood (None): A flaw in the code that requires external factors proven to be impossible in a production environment, either due to mathematical constraints, operational constraints, or system-related factors (i.e. EIP-20 tokens not being re-entrant).

Disclaimer

The following disclaimer applies to all versions of the audit report produced (preliminary / public / private) and is in effect for all past, current, and future audit reports that are produced and hosted under Omniscia:

IMPORTANT TERMS & CONDITIONS REGARDING OUR SECURITY AUDITS/REVIEWS/REPORTS AND ALL PUBLIC/PRIVATE CONTENT/DELIVERABLES

Omniscia ("Omniscia") has conducted an independent security review to verify the integrity of and highlight any vulnerabilities, bugs or errors, intentional or unintentional, that may be present in the codebase that were provided for the scope of this Engagement.

Blockchain technology and the cryptographic assets it supports are nascent technologies. This makes them extremely volatile assets. Any assessment report obtained on such volatile and nascent assets may include unpredictable results which may lead to positive or negative outcomes.

In some cases, services provided may be reliant on a variety of third parties. This security review does not constitute endorsement, agreement or acceptance for the Project and technology that was reviewed. Users relying on this security review should not consider this as having any merit for financial advice or technological due diligence in any shape, form or nature.

The veracity and accuracy of the findings presented in this report relate solely to the proficiency, competence, aptitude and discretion of our auditors. Omniscia and its employees make no guarantees, nor assurance that the contracts are free of exploits, bugs, vulnerabilities, deprecation of technologies or any system / economical / mathematical malfunction.

This audit report shall not be printed, saved, disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Omniscia.

All the information/opinions/suggestions provided in this report does not constitute financial or investment advice, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report.

Information in this report is provided 'as is'. Omniscia is under no covenant to the completeness, accuracy or solidity of the contracts reviewed. Omniscia's goal is to help reduce the attack vectors/surface and the high level of variance associated with utilizing new and consistently changing technologies.

Omniscia in no way claims any guarantee, warranty or assurance of security or functionality of the technology that was in scope for this security review.

In no event will Omniscia, its partners, employees, agents or any parties related to the design/creation of this security review be ever liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this security review.

Cryptocurrencies and all other technologies directly or indirectly related to cryptocurrencies are not standardized, highly prone to malfunction and extremely speculative by nature. No due diligence and/or safeguards may be insufficient and users should exercise maximum caution when participating and/or investing in this nascent industry.

The preparation of this security review has made all reasonable attempts to provide clear and actionable recommendations to the Project team (the “client”) with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts in scope for this engagement.

It is the sole responsibility of the Project team to provide adequate levels of test and perform the necessary checks to ensure that the contracts are functioning as intended, and more specifically to ensure that the functions contained within the contracts in scope have the desired intended effects, functionalities and outcomes, as documented by the Project team.

All services, the security reports, discussions, work product, attack vectors description or any other materials, products or results of this security review engagement is provided "as is" and "as available" and with all faults, uncertainty and defects without warranty or guarantee of any kind.

Omniscia will assume no liability or responsibility for delays, errors, mistakes, or any inaccuracies of content, suggestions, materials or for any loss, delay, damage of any kind which arose as a result of this engagement/security review.

Omniscia will assume no liability or responsibility for any personal injury, property damage, of any kind whatsoever that resulted in this engagement and the customer having access to or use of the products, engineers, services, security report, or any other other materials.

For avoidance of doubt, this report, its content, access, and/or usage thereof, including any associated services or materials, shall not be considered or relied upon as any form of financial, investment, tax, legal, regulatory, or any other type of advice.