HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Apartchain
Date: 18 April, 2023

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Apartchain
Approved By	Marcin Ugarenko | Lead Solidity SC Auditor at Hacken OU
Type	ERC1155; Marketplace
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://apartchain.io/
Changelog	28.02.2023 - Initial Review 18.04.2023 - Second Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Apartchain (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is review and security analysis of smart contracts in the repository:

Initial review scope

Repository	https://github.com/apartchain/smartcontracts
Commit	da78677e1ff9175c19e3e0fdda98110c9d172855
Technical Requirements	Technical documentation - Confluence.pdf
Contracts	File: ./contracts/Fee.sol SHA3: 64f88681e6073b6af5c4186fedd55a7e96dbd4a2ade344e867d2e293d323f34a File: ./contracts/Marketplace.sol SHA3: ff446d95ee8f27481297f66a7ea6fa5849500670cf0a227835023d1d91f95e29 File: ./contracts/opengsn/Forwarder.sol SHA3: cfe74c8ea0fd4d541319a1fb6c41651a548fbd0f6c6e77fc3ba85484d5494555 File: ./contracts/opengsn/VerifyingPaymaster.sol SHA3: e425c12f8b203736b74de8d7c7212f458efda5ef64ee9555b2c8e356f0af2437 File: ./contracts/RealEstate.sol SHA3: a0d266b6dd43155dc121f4936fc087ce5add10b5e5d2123d42c213065d64d636 File: ./contracts/Referral.sol SHA3: 5f3f4a3d2b0ee47d22bbec15bb0528f7069b34770981dc86c802d74e8daaaf03 File: ./contracts/Verifier.sol SHA3: c0796102d6e2ad75708926b461e1bd6a28b2a4457431937bc4aa9a526a34d64e

Second review scope

Repository	https://github.com/apartchain/smartcontracts
Commit	d46d3c3541ef1241450c2e55327a87369246d212
Functional Requirements	Aparchain_sc_doc.pdf
Technical Requirements	Aparchain_sc_doc.pdf

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Contracts	File: contracts/Fee.sol SHA3: 88af0ebc6e696eb8b68078159d37d3e249a92c687d8068f3ea998c0e06b05f20 File: contracts/Forwarder.sol SHA3: c8f6166c4c643cb0bf66f68e0c02721087932d8ae30d7c17dc1d9300a69d53b0 File: contracts/Marketplace.sol SHA3: 2c6d7c84252d7ea602a9bdd2f6f9e15288c9def22aa9b9f9f9d22826ab2a5a64 File: contracts/RealEstate.sol SHA3: c9e3855b30a2964fef2d48f287599c86afe47865a8d4a3430e8cf093f3cf76f2 File: contracts/Referral.sol SHA3: e6366b135d3abcb0213891b03efd24ca3b6e19b62a0036a30ebe9e73c9cdb1f0 File: contracts/Verifier.sol SHA3: f352e21ae426fdec0eb1434b94c71abcd5026bfd58813129f7e8a86b70843d1c

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor gas optimization. These issues won't have a significant impact on code execution but affect code quality

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional requirements are provided.
- Technical description is provided.
- Natspec is present and sufficient.

Code quality

The total Code Quality score is **10** out of **10**.

- The development environment is configured.

Test coverage

Code coverage of the project is **98.68%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Negative cases coverage is missing.
- Interactions with several users are not tested thoroughly.

Security score

As a result of the audit, the code contains **1** low severity issue. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **10**

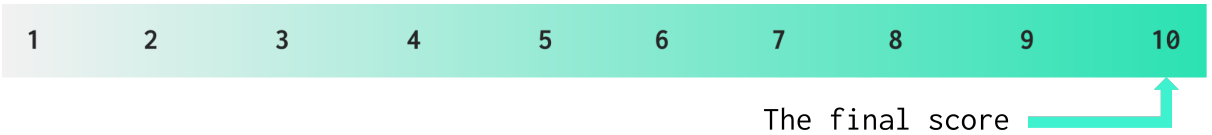

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
28 February 2023	23	10	4	1
18 April 2023	1	0	0	0

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

System Overview

Apartchain is a Real Estate marketplace. It allows agencies to create properties (ERC1155), and users to book, buy and fulfill the buy order for the property.

The project contains the following contracts:

- **Fee.sol**: used to manage and calculate most of the fees inside the protocol.
- **Marketplace.sol**: used to manage the bookings and the orders.
- **RealEstate.sol**: used to mint and burn the ERC1155 tokens.
- **Referral.sol**: used for the referral system, users can refer other users and gain a 0.2% fee on fulfilled orders from referred users.
- **Verifier.sol**: used to manage the list of whitelisted users and agencies that can interact with the protocol.

Privileged roles

- **MARKETPLACE_MANAGER_ROLE**: can cancel the booking, cancel the trade, fulfill the buy and change the signedAllDoc boolean inside the Marketplace.sol contract.
- **FEE_CHANGER_ROLE**: can change various fees in the Fee.sol contract.
- **VERIFIER_ROLE**: can set a user or agency as verified in the Verifier.sol contract.
- **MARKETPLACE_CONTRACT_ROLE**: is the address of the marketplace that can mint and burn ERC1155 inside the RealEstate.sol contract.
- **SERVICE_ROLE**: can manage the referral inside the Referral.sol contract.
- **DEFAULT_ADMIN_ROLE:** there are 5 different default admin roles, one for the marketplace, one for the fee contract, one for the real estate contract, one for the verifier and one for the referral, this default admin role will be able to manage the various roles inside the contracts.

Risks

- The project is highly centralized
- The roles have to be setted in the correct way for the protocol to function properly.
- Certain functionalities are off-chain and the correct execution of the entire flow cannot be ensured.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Not Relevant
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Not Relevant
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Passed
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Not Relevant
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155 EIP-712	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Passed
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of Unused Variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP Standards Violation	EIP	EIP standards should not be violated.	Not Relevant
Assets Integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed
User Balances Manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Not Relevant
Token Supply Manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Passed
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Not Relevant
Style Guide Violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Not Relevant
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed
Stable Imports	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Findings

■■■■ Critical

C01. Requirements Violations

There is a significant number of requirements violations.

The technical documentation for the *createProperty()* function does not match the code:

- The function *createProperty()* does not check for *MARKETPLACE_MANAGER_ROLE*, as stated in the documentation.
- It does not check if the *_seller* is a verified user
- It does not use *_uri* except for the event emission.

The technical documentation for the *bookProperty()* function does not match the code:

- The function *bookProperty()* checks if the *isBooked* flag is false, but this is not reflected in the documentation.

The technical documentation for the *cancelBooking()* function does not match the code:

- The function *cancelBooking()* has the *onlyRole(MARKETPLACE_MANAGER_ROLE)* modifier, but this is not reflected in the documentation.
- *toUser* is not a documented parameter.
- the caller is not checked as explained in the documentation.

The technical documentation for the *cancelTrade()* function does not match the code:

- The function *cancelTrade()* does not burn *ERC1155*.
- It has the *onlyRole(MARKETPLACE_MANAGER_ROLE)* modifier instead of verifying that the caller is a verified agency.
- It does not check *isOnSale*, but does check if the booking has been paid.
- *isOnSale* is set to true and not false.
- The *paid* boolean is not changed and instead the booked flag is set to false.

The technical documentation for the *buyProperty()* function does not match the code:

- The function *buyProperty()* does not have *_amount* as one of the two input parameters.
- It does not call *getReferrer* and does not send any fees to the *referrer*.
- It does not burn *ERC1155*.
- It does a lot of *require* checks that are not documented.

The technical documentation for the *fulfilBuy()* function does not match the code:

- The function *fulfilBuy()* has the *onlyRole(MARKETPLACE_MANAGER_ROLE)* modifier.
- The *signedAllDoc* variable is not checked.
- It does not check if the caller is a verified user.
- It does not transfer the amount from the buyer to the seller.
- The *paid* boolean is never set to true.

All of these requirements violations make the scope of the code unclear.

Path:
./contracts/Marketplace.sol

Recommendation: Fix the requirement violations, the documentation should match the code.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

High

H01. Undocumented Behavior

In the documentation, *platformFee* is described as a flat fee, but this is not reflected in the code, platformFee is a result of a calculation with a variable called *factor* and the calculation is not documented.

The number 18 in the for loop is also a magic number.

Path:
./contracts/Fee.sol : getPlatformFee()

Recommendation: Document the calculations inside *platformFee* or fix the mismatch in the code.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

H02. Denial Of Service

There is a possibility of a DoS inside the *fulfillBuy()* function, when the *sellerFee* and *buyerFee* are set to 0 and there is a referral. The *referralFee* will be calculated as *pt.price * 20 / 10000*, the *platformFee* will be 0 and when performing the calculation *platformFee -= referralFee* there will be an underflow.
At this point the only callable function will be *cancelTrade()*.

Path:
./contracts/Marketplace.sol : fulFillBuy()

Recommendation: If there is a referral, *platformFee* should be able to cover that expense or the referral should not be paid.

Additionally *poaFee* should be added after the deduction of the *referralFee* to avoid taking *referralFee* from *poaFee*.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

H03. Funds Lock

If a buyer calls the function *buyProperty()* with the option to pay for a POA and then the trade is canceled, *poaFee* will be locked in the contract.

Path:
./contracts/Marketplace.sol : cancelTrade()

Recommendation: Allow *poaFee* to be withdrawn if the trade is canceled.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

H04. Denial Of Service

In case there is a *sellerFee* higher than 98% there will be a DoS. The equation *uint256 sellerPart = pt.price - bk.sellerFee - agencyFee*; will revert due to underflow.

Path:
./contracts/Marketplace.sol : fulFillBuy()

Recommendation: Do not allow the *sellerFee* to be higher than 98%.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

Medium

M01. Best Practice Violation

The functions do not use the *SafeERC20* library to check the result of *ERC20* token transfers. Tokens may not follow the *ERC20* standard and return false in case of a transfer failure or not return any value at all.

Path:
./contracts/MarketPlace.sol : bookProperty(), cancelBooking(), cancelTrade(), buyProperty(), fulfilBuy()

Recommendation: Use the *SafeERC20* library to interact with tokens safely.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

M02. Contradiction

The use of *ERC2771* for the gasless meta transactions is not documented.

Path:
./contracts/Marketplace.sol

Recommendation: Remove the functionality or mention it in the documentation.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

M03. Contradiction

The function *signedAllDoc()* is not documented.

Path:
./contracts/Marketplace.sol : signedAllDoc()

Recommendation: Remove the functionality or mention it in the documentation.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

M04. Contradiction

There is no functionality to remove a property from the contract, nor to put it as not for sale

Path:
./contracts/Marketplace.sol

Recommendation: Add the possibility to remove a property or put it as not for sale.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

M05. Insufficient Gas Model

All the fees should be calculated in a single function and there should only be one request to the fee contract in *bookProperty()*.

Paths:
./contracts/Marketplace.sol : bookProperty()
./contracts/Fee.sol

Recommendation: Remove the redundant calls to the *Fee.sol* contract and perform all the fee calculations inside a single function.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

M06. Insufficient Gas Model

In *cancelBooking()* there is an unnecessary storage operation, the variable *uint256 bookingFee* is declared, but the variable *bk.fee* is already available.

Path:
./contracts/Marketplace.sol : cancelBooking()

Recommendation: Use the variable *bk.fee* and remove the newly created variable *bookingFee.*

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

M07. Invalid Calculations

In *cancelBooking()* due to a rounding error in percentage calculations, the *agencyFee* should be equal to *bookingFee - sellerFee - platformFee*.

Path:
./contracts/Marketplace.sol : cancelBooking()

Recommendation: Change the way the *agencyFee* is calculated.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

M08. Insufficient Gas Model

The *mapping(address => bool) private set;* is redundant.

Path:
./contracts/Referral.sol

Recommendation: The *!set[_referral]* check can be replaced by *referrals[_referral] == address(0)*.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

M09. Contradiction

There is a contradiction in the NatSpec of *cancelBooking()* and *bookProperty()*, where the *bookingFee* is stated as 10%, but the fee can be different than 10%.

Path:
./contracts/Marketplace.sol : cancelBooking(), bookProperty()

Recommendation: Fix the mismatch.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

Low

L01. Floating Pragma

The project uses floating pragmas *^0.8.9*, *^0.8.1*, *^0.8.0*.

Paths:
./contracts/Marketplace.sol
./contracts/Verifier.sol
./contracts/Fee.sol
./contracts/RealEstate.sol
./contracts/Referral.sol
./contracts/Forwarder.sol
./contracts/VerifyingPaymaster.sol

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

L02. Unindexed Events

Having indexed parameters in the events makes it easier to search for these events using indexed parameters as filters.

Paths:
./contracts/Marketplace.sol
./contracts/Verifier.sol

Recommendation: Use the “indexed” keyword for relevant, trackable event parameters.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

L03. Missing Events

Events for critical state changes should be emitted for tracking things off-chain.

Path:
./contracts/VerifyingPaymaster.sol : setSigner()

Recommendation: Create and emit related events.

Found in: da78677

Status: Mitigated (The contract VerifyingPaymaster is out of scope in the second review)

L04. Missing Zero Address Validation

Address parameters are used without checking against the possibility of 0x0.

This can lead to unwanted external calls to 0x0.

Paths:
./contracts/Marketplace.sol : constructor()
./contracts/VerifyingPaymaster.sol : setSigner()

Recommendation: Implement zero address checks.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

L05. Empty Contract

The contract *Forwarder.sol* is empty.

Path:
./contracts/Forwarder.sol

Recommendation: Remove the contract or implement it.

Found in: da78677

Status: Fixed (Revised commit: d46d3c3)

L06. Redundant Override Keyword

Since *solidity 0.8.8*, a function that overrides only a single interface function does not require the *override* specifier.

Path:
./contracts/Marketplace.sol : onERC1155Received(), onERC1155BatchReceived()

Recommendation: Remove redundant code.

Status: Fixed (Revised commit: d46d3c3)

L07. Style Guide Violation

The provided projects should follow the official guidelines.

Paths:
./contracts/Marketplace.sol
./contracts/Fee.sol
./contracts/RealEstate.sol
./contracts/Referral.sol
./contracts/Verifier.sol

Recommendation: Follow the official Solidity guidelines.

Status: Fixed (Revised commit: d46d3c3)

L08. Redundant Block

The usage of *bk.buyer = sender* is unnecessary for the contract, the check *require(bk.buyer == sender, "not your booking")* already made sure that the bk.buyer is equal to the sender.

Path:
./contracts/Marketplace.sol : buyProperty()

Recommendation: Remove the redundant code block.

Status: Fixed (Revised commit: d46d3c3)

L09. Inefficient Gas Model

Inside the declaration of the struct *Booking* changing the order of the variable would save gas.

Packing variables in a 32 byte block would allow the contract to save gas.

Path:
./contracts/Marketplace.sol

Recommendation: Change the order of the variables inside the struct *Booking*.

Status: Fixed (Revised commit: d46d3c3)

L10. Deprecated Function

The function *_setupRole()* has been deprecated in favor of *_grantRole()*.

Paths:
./contracts/Verifier.sol : constructor(), setVerifier()
./contracts/Referral.sol : constructor(), setService()

./contracts/RealEstate.sol : constructor(), setMarketplaceContract()
./contracts/Marketplace.sol : setMarketplace()
./contracts/Fee.sol : constructor(), setFeeChanger()

Recommendation: Change the instances where *_setupRole()* is used.

Status: Fixed (Revised commit: d46d3c3)

L11. Function That Can Be Declared External

In order to save Gas, public functions that are never called in the contract should be declared as external.

Paths:
./contracts/Verifier.sol : isVerifiedUser(), isVerifiedAgency(), setVerifier(), setVerificationAgency(), setVerificationUser()
./contracts/Referral.sol : getReferrer(), setService(), setReferral()
./contracts/Fee.sol : getBuyerFee(), getSellerFee(), getPoaFee(), getBookingFee(), setFeeChanger()
./contracts/RealEstate.sol : setMarketplaceContract()

Recommendation: Use the external attribute for functions never called from the contract.

Status: Fixed (Revised commit: d46d3c3)

L12. Redundant Import

The usage of *@opengsn/contracts/src/forwarder/IForwarder.sol* is unnecessary for the contract.

Paths:
./contracts/opengsn/VerifyingPaymaster.sol

Recommendation: Remove the redundant import.

Status: Mitigated (The contract VerifyingPaymaster is out of scope in the second review)

L13. Unfinished NatSpec

In most contracts the NatSpec is missing, in the *Marketplace.sol* contract it is insufficient.

Paths:
./contracts/Marketplace.sol
./contracts/Fee.sol
./contracts/Referral.sol
./contracts/RealEstate.sol
./contracts/Verifier.sol

Recommendation: Add a meaningful NatSpec.

Status: Fixed (Revised commit: d46d3c3)

L14. Redundant Block

In the *checkPercentage()* modifier there is a check for the fee to be *>= 0*, this is redundant since the fee is a uint256.

Paths:
./contracts/Fee.sol : checkPercentage()

Recommendation: Remove the redundant piece of code.

Status: Fixed (Revised commit: d46d3c3)

L15. Redundant Pragma

pragma experimental ABIEncoderV2; is the default for the solidity version 0.8.0.

Paths:
./contracts/opengsn/VerifyingPaymaster.sol

Recommendation: Remove the redundant pragma.

Status: Mitigated (The contract VerifyingPaymaster is out of scope in the second review)

L16. Redundant Block

In many top level functions there is still a virtual modifier.

Paths:
./contracts/RealEstate.sol : supportsInterface()
./contracts/Marketplace.sol : _msgSender(), _msgData(), supportsInterface()
./contracts/oepngsn/VerifyingPaymaster.sol : preRelayedCall(), postRelayedCall(), versionPaymaster();

Recommendation: Remove the redundant piece of code.

Status: Fixed (Revised commit: d46d3c3)

L17. Typos In The Comments

There are various typos in the comments.

Paths:
./contracts/Referral.sol
./contracts/Marketplace.sol

Recommendation: Change *setted* with set, *USDc* with USDC, *usdC* with USDC, *transfering* with transferring and *reffered* with referred.

Status: Fixed (Revised commit: d46d3c3)

L18. Missing Error Message

Some require statements are missing error messages.

This makes the code harder to test and debug.

Paths:
./contracts/RealEstate.sol : setMarketplaceContract()

Recommendation: Add error messages to require conditions.

Status: Mitigated (The require statements have been removed)

L19. Variables Can Be Declared Immutable

The following variables' values are set in the constructor: *realEstate*, *verifier*, *fee*, *usdC*, *platform*. These variables can be declared as immutable.

This will lower the Gas cost.

Paths:
./contracts/Marketplace.sol

Recommendation: Declare mentioned variables as immutable.

Status: Fixed (Revised commit: d46d3c3)

L20. Redundant Mapping

In the *mapping(address => mapping(uint256 => Property)) public properties*, the *address => mapping* section is redundant.

Paths:
./contracts/Marketplace.sol

Recommendation: Remove the redundant piece of code.

Status: Fixed (Revised commit: d46d3c3)

L21. Best Practice Violation

There are many CEI pattern violations mitigated with the *noReentrant* modifier, the CEI could be followed and the modifier could be dropped to save gas.

Path:
./contracts/Marketplace.sol : bookProperty(), cancelBooking(), cancelTrade(), buyProperty(), fulfillBuy()

Recommendation: Remove the modifier and follow the CEI pattern.

Status: Fixed (Revised commit: d46d3c3)

L22. Redundant Require

require(usdC.allowance(sender, address(this)) >= bookingFee, "not enough allowance"); is redundant because it is already checked in the *transferFrom()* function.

Path:
./contracts/Marketplace.sol : bookProperty(), buyProperty()

Recommendation: Remove the redundant piece of code.

Status: Fixed (Revised commit: d46d3c3)

L23. Redundant Timestamp In Events

The *timestamp* in events increases gas cost, and can be omitted as it can be retrieved off-chain from the block with an extra off-chain operation.

Paths:
./contracts/Marketplace.sol
./contracts/Fee.sol

Recommendation: Remove the *timestamp* from the events.

Status: Fixed (Revised commit: d46d3c3)

L24. Strict Condition

In the *fulfillBuy()* function there is an `if` statement that checks if there is a referrer and if the *plaformFee* is higher than the *referralFee*, this is done to avoid a DoS in case the *referralFee* is higher, but there could be a cases where the *platformFee* is equal to the *referralFee*, and since this checks only passes if the *plaformFee* is strictly higher, then the referrer doesn't get paid.

Path:
./contracts/Marketplace.sol : fulfillBuy()

Recommendation: Add the equal sign in the if check.

Found in: d46d3c3

Status: New

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.