Nafter

Marketplace

SMART CONTRACT AUDIT

01.09.2021

<u>**Made in Germany by Chainsulting.de**</u>

Table of contents

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Nafter Project. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (30.08.2021)	Layout
0.2 (31.08.2021)	Test Deployment
0.5 (31.08.2021)	Automated Security Testing Manual Security Testing
0.6 (31.08.2021)	Testing SWC Checks
0.7 (01.09.2021)	Verify Claims
0.9 (01.09.2021)	Summary and Recommendation
1.0 (01.09.2021)	Final document
1.1 (02.09.2021)	Adding deployed contract address

2. About the Project and Company

Company address:

Helios Ltd.
c/o Nafter Project
Lavida Plus Officetel, Nguyen Van Linh Street,
Tan Phong Ward, District 7
Ho Chi Minh City
Vietnam

Website: https://nafter.io

Twitter: https://twitter.com/Nafterapp

Telegram: https://t.me/nafterapp

Instagram: https://www.instagram.com/nafterapp

Medium: https://nafterapp.medium.com

2.1 Project Overview

With NFT trade volumes soaring and more use-cases emerging, Nafter allows influencers to make the most of this exciting new industry and creatively respond to fan demand. Through the platform, these influencers can convert memorable photos into digital collectibles that can be acquired by followers.

No technical knowledge is required: Nafter automates the whole process, allowing influencers to transform stunning snaps into collectible NFTs at the touch of a button. A one-of-a-kind platform, Nafter helps to drive engagement among fan bases by enabling well-known figures to create NFTs that capture a moment in time. In turn, followers get to lay their hands on unique tokens that they can cherish for years, sell or trade.

Nafter isn't just for influencers. It's for photographers, artists, athletes, musicians, celebrities, and creators. It's for anyone with a following that wants to unlock value by commoditizing special moments. On Nafter, fans get to decide which NFTs are valuable and which are not. The people have the power, and through social engagement, they will communicate the type of content they want to see. In this way, influencers will learn to build better connections with their audience and deliver content that hits the spot.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

## 4.2 Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.2/contracts/access/Ownable.sol
@openzeppelin/contracts/access/AccessControl.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.2/contracts/access/AccessControl.sol
@openzeppelin/contracts/math/SafeMath.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.2/contracts/math/SafeMath.sol
@openzeppelin/contracts/token/ERC1155/ERC1155.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.2/contracts/token/ERC1155/ERC1155.sol
@openzeppelin/contracts/token/ERC1155/IERC1155.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.2/contracts/token/ERC1155/IERC1155.sol
@openzeppelin/contracts/payment/PullPayment.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.2/contracts/payment/PullPayment.sol

4.3 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

| File	Fingerprint (MD5)
./IERC1155TokenCreator.sol	858ff28e048350991dea9471412d32dd
./IMarketplaceSettings.sol	d7e9a9f99b29c6e5cd43a3837ae14d64
./INafter.sol	6157ce21fc37ae942fb66126f7131602
./INafterMarketAuction.sol	b3f3c9ae3b33e23c8d1ec8f32aeb8cc9
./INafterRoyaltyRegistry.sol	0ed8cd80cbf3290d99d0485d5d914d6f
./INafterTokenCreatorRegistry.sol	861c4ae34ef4d30f94c2bcfbfeb3d27c
./ISendValueProxy.sol	11073b2b1646b6793ce0171f57e6ff6b
./MarketplaceSettings.sol	e9589208ef72f06efdf4030884bef536
./MaybeSendValue.sol	d5a393da9034d0a82de39384c4017728
./Nafter.sol	99b8708e97ea76b1b949bb2217105286
./NafterMarketAuction.sol	59ff366587890602921fe1d8c6093622
./NafterRoyaltyRegistry.sol	9071fc28ccbc879e73f373ac2f6a4d04
./NafterTokenCreatorRegistry.sol	7ff0b7298093ded2c9aa80f6afa4e8be
./Payments.sol	a41f606e36aefacdee7362feab28de96
./SendValueOrEscrow.sol	f16faabd02a01cd230e33f7b9875e3a5
./SendValueProxy.sol	3da17e39a806ea2c3b7e641246b710fe

4.4 Metrics / CallGraph
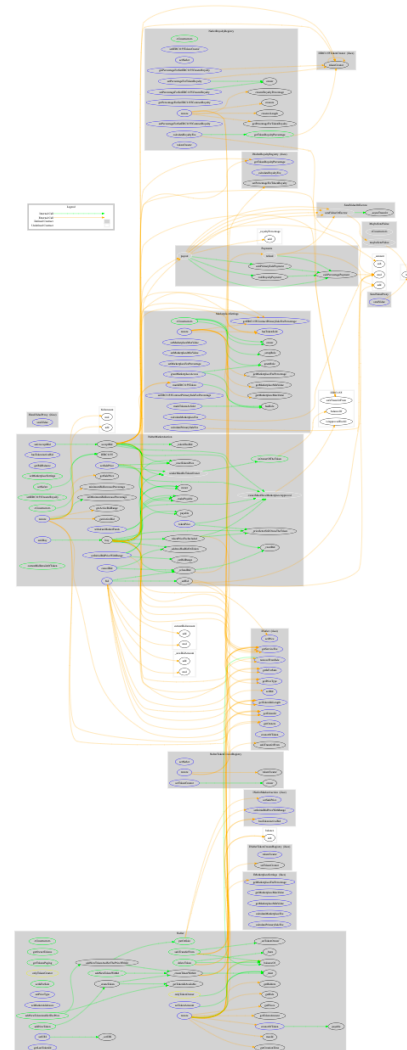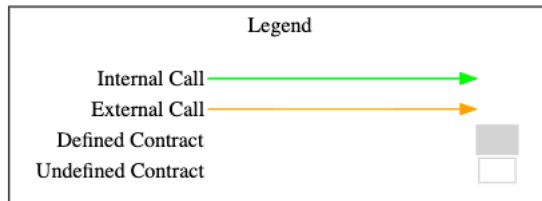
View full version: https://chainsulting.de/wp-content/uploads/2021/09/solidity-metrics_nafter.html

4.5 Metrics / Source Lines & Risk
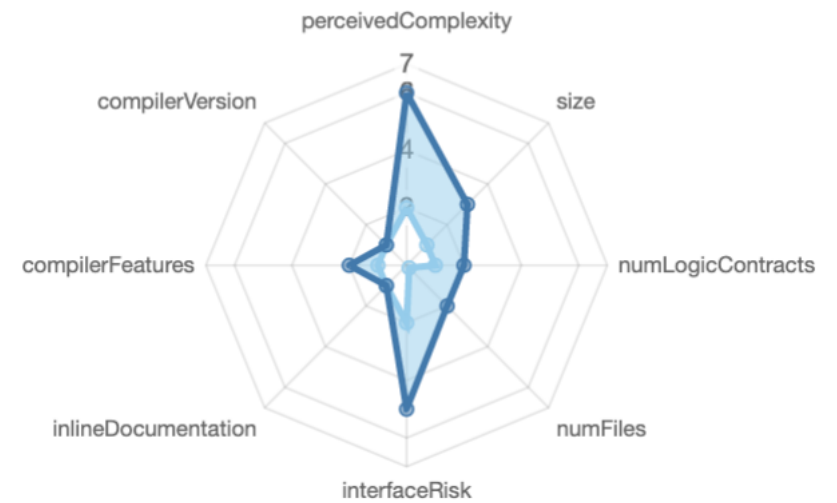
4.6 Metrics / Capabilities

🔧 Solidity Versions observed	✏️ Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`0.6.12`	`ABIEncoderV2`	`yes`	**** (0 asm blocks)	

📥 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🎰 Uses Hash Functions	📝 ECRecover	🌀 New/Create/Create2
`yes`					`yes` → `NewContract:SendValueProxy`

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
120	5

External	Internal	Private	Pure	View
95	89	2	4	64

StateVariables

Total	🌐Public
35	18

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/active/IERC1155TokenCreator.sol		1	18	14	3	9	3	
📝	contracts/active/NafterRoyaltyRegistry.sol	1		306	277	125	124	90	
📝	contracts/active/NafterMarketAuction.sol	1		950	885	413	360	271	💰🔺🔵
📝	contracts/active/MarketplaceSettings.sol	1		323	291	106	151	90	
📝	contracts/active/MaybeSendValue.sol	1		32	29	12	9	14	🔺🔵🌀
📝	contracts/active/NafterTokenCreatorRegistry.sol	1		103	95	42	38	38	
🔍	contracts/active/INafterTokenCreatorRegistry.sol		1	28	14	3	14	5	
📝	contracts/active/Payments.sol	1		203	177	78	83	36	
🔍	contracts/active/ISendValueProxy.sol		1	6	5	3	1	6	💰

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/active/INafterRoyaltyRegistry.sol		1	44	16	4	24	9	
🔍	contracts/active/INafter.sol		1	92	16	3	54	21	
📝	contracts/active/SendValueOrEscrow.sol	1		33	33	13	17	9	
📝	contracts/active/Nafter.sol	1		620	578	291	217	295	🖊️
📝	contracts/active/SendValueProxy.sol	1		19	19	7	10	9	💰📤
🔍	contracts/active/IMarketplaceSettings.sol		1	89	16	3	54	17	
🔍	contracts/active/INafterMarketAuction.sol		1	40	11	3	19	7	
📝🔍	**Totals**	**9**	**7**	**2906**	**2476**	**1109**	**1184**	**920**	🖊️💰📤🌀

Legend: [━]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

5. Scope of Work

The Nafter Team provided us with the files that needs to be tested. The scope of the audit are the marketplace contracts.

The team put forward the following assumptions regarding the security, usage of the contracts:

- Fees/Royalty and auctions are working as expected
- Access control is correctly working
- Restore data from old contract is correctly working
- Owner cannot burn or lock user funds
- Owner cannot pause the contract
- Owner cannot front-run or manipulate auctions
- The smart contract is coded according to the newest standards and in a secure way.

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.
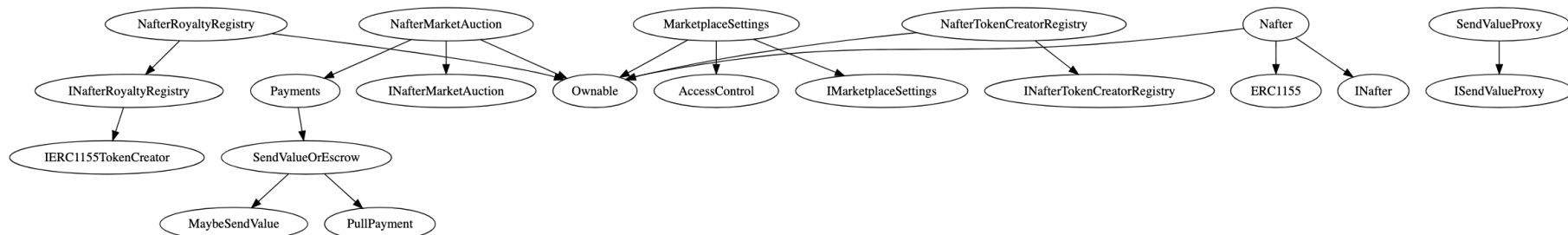
5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES
During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES
During the audit, Chainsulting's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES
During the audit, Chainsulting's experts found **no Medium issues** in the code of the smart contract

LOW ISSUES
During the audit, Chainsulting's experts found **no Low issues** in the code of the smart contract

5.2. SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✅
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✅
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✅
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✅
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✅
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✅
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✅
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✅

ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	☑
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	☑
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	☑
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	☑
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	☑
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	☑
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	☑
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	☑
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	☑

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✅
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✅
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✅
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✅
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✅
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✅
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✅
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✅

ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	☑
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	☑
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	☑
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	☑

5.3. Verify Claims

5.3.1 Fees/Royalty and auctions are working as expected
Status: tested and verified ✅

5.3.2 Access control is correctly working
Status: tested and verified ✅

5.3.3 Restore data from old contract is correctly working
Status: tested and verified ✅

5.3.4 Owner cannot burn or lock user funds
Status: tested and verified ✅
There is no function to burn or lock user funds

Line: 484 – 492
File: Nafter.sol
Status: Only the token owner (different to contract owner) is able to burn.

```
/**
 * @dev Deletes the token with the provided ID.
 * @param _tokenId uint256 ID of the token.
 * @param _amount amount of the token to delete
 */
function deleteToken(uint256 _tokenId, uint256 _amount) public onlyTokenOwner(_tokenId) {
    bool activeBid = marketAuction.hasTokenActiveBid(_tokenId, msg.sender);
    uint256 balance = balanceOf(msg.sender, _tokenId);
    //2
    if (activeBid == true)
        require(balance.sub(_amount) > 0, "you have the active bid");
```

```
    _burn(msg.sender, _tokenId, _amount);
    DeleteTokens(msg.sender, _tokenId, _amount);
}
```

5.3.5 Owner cannot pause the contract
  Status: tested and verified ✅
  There is no function to pause the contract

5.3.6 Owner cannot front-run or manipulate auctions
  Status: tested and verified ✅

5.3.7 The smart contract is coded according to the newest standards and in a secure way.
  Status: tested and verified ✅

5.4. Unit Test

```
Using network 'development'.

Compiling your contracts...
===========================
> Compiling ./contracts/active/IERC1155TokenCreator.sol
> Compiling ./contracts/active/IMarketplaceSettings.sol
> Compiling ./contracts/active/INafter.sol
> Compiling ./contracts/active/INafterMarketAuction.sol
> Compiling ./contracts/active/INafterRoyaltyRegistry.sol
> Compiling ./contracts/active/INafterTokenCreatorRegistry.sol
> Compiling ./contracts/active/ISendValueProxy.sol
> Compiling ./contracts/active/MarketplaceSettings.sol
> Compiling ./contracts/active/MaybeSendValue.sol
> Compiling ./contracts/active/Migrations.sol
> Compiling ./contracts/active/Nafter.sol
> Compiling ./contracts/active/NafterMarketAuction.sol
> Compiling ./contracts/active/NafterRoyaltyRegistry.sol
```

```
> Compiling ./contracts/active/NafterTokenCreatorRegistry.sol
> Compiling ./contracts/active/Payments.sol
> Compiling ./contracts/active/SendValueOrEscrow.sol
> Compiling ./contracts/active/SendValueProxy.sol
> Compiling ./contracts/active/TestAssertFailOnPay.sol
> Compiling ./contracts/active/TestExpensiveWallet.sol
> Compiling ./contracts/active/TestRequireFailOnPay.sol
> Compiling ./contracts/active/TestRevertOnPay.sol
> Compiling @openzeppelin/contracts/access/AccessControl.sol
> Compiling @openzeppelin/contracts/access/Ownable.sol
> Compiling @openzeppelin/contracts/introspection/ERC165.sol
> Compiling @openzeppelin/contracts/introspection/IERC165.sol
> Compiling @openzeppelin/contracts/math/SafeMath.sol
> Compiling @openzeppelin/contracts/payment/PullPayment.sol
> Compiling @openzeppelin/contracts/payment/escrow/Escrow.sol
> Compiling @openzeppelin/contracts/token/ERC1155/ERC1155.sol
> Compiling @openzeppelin/contracts/token/ERC1155/IERC1155.sol
> Compiling @openzeppelin/contracts/token/ERC1155/IERC1155MetadataURI.sol
> Compiling @openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol
> Compiling @openzeppelin/contracts/utils/Address.sol
> Compiling @openzeppelin/contracts/utils/Context.sol
> Compiling @openzeppelin/contracts/utils/EnumerableSet.sol
> Artifacts written to /var/folders/hx/tlqhgx850gl4qd91d42yfb5c0000gn/T/test--9777-y7VYCzVnzYF0
> Compiled successfully using:
   - solc: 0.6.12+commit.27d51765.Emscripten.clang

marketplaceSettings address => 0xAC55c411cBAe806bb040FDA7204fab86b45258Bc
Nafter address => 0x507B3F1B3db8cE1801Ee3cec11b428852870186e
NafterTokenCreatorRegistry address => 0x07E129C3cd09a3E8b60656EDA7e9DD8C20271d03
NafterMarketAuctionV2 address => 0xF37C43D1b32C6fbEEB85BFD5846f3932261d47eA
NafterRoyaltyRegistry address => 0xcac9916cFd512Df054494890Da1F2A1DE4b18e89


   Contract: Nafter test
     ✓ should create new token (328ms)
```

```
✓ should not create new token if not a creator (209ms)
✓ should set a sale price (862ms)
✓ should create a bid on a token (810ms)
✓ should create a bid on a token and balance be reflected in bidder's account (580ms)
✓ should not bid on token if token owner (785ms)
✓ should not bid on token if token has bid greater than proposed bid (919ms)
✓ should accept a bid on a token and payout all parties (1867ms)
✓ should buy a token (1244ms)
✓ should not buy a token if the owner (1389ms)
✓ should not buy a token if not for sale (890ms)
✓ should not buy a token if not enough money sent (937ms)
✓ should buy a token and current bidder gets money returned (1671ms)
✓ should not set creator percentage if not Nafter owner (57ms)
✓ should other than creater place the bid (1609ms)
✓ should work with multiple licenses (1815ms)
✓ should update the nafter uri (190ms)
✓ should upgrade contracts (12839ms)


18 passing (30s)
```

6. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs took place on the September 01, 2021.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the functions. During the audit, no critical issues were found after the manual and automated security testing and the claims been successfully verified. We have been very satisfied with the codebase quality and NatSpec documentation.

7. Deployed Smart Contract

VERIFIED

ETH:
Verifying MarketplaceSettings
Pass - Verified: https://etherscan.io/address/0x4c4Edc5694f252459fCce235968c359cE717E961#contracts

Verifying Nafter
Pass - Verified: https://etherscan.io/address/0x045E2254eA5853b65880EDbd9e81118748b06592#contracts

Verifying NafterTokenCreatorRegistry
Pass - Verified: https://etherscan.io/address/0x94060EB92F711A5A88596b91aec4d964C01C31BB#contracts

Verifying NafterMarketAuction
Pass - Verified: https://etherscan.io/address/0xC40999f4Ba28aDC523f06623815B3F2A33bfD6Ba#contracts

Verifying NafterRoyaltyRegistry
Pass - Verified: https://etherscan.io/address/0x7cE3600ee30641E04e99EaE29ac30Fc861fbBB53#contracts

Successfully verified 5 contract(s).

BSC:
Verifying MarketplaceSettings
Pass - Verified: https://bscscan.com/address/0x19fb3e8AA418C3675A488FBE6f3371875aA67617#contracts

Verifying Nafter
Pass - Verified: https://bscscan.com/address/0xb33b95835589583375369693043b6aa2e6dAafe#contracts

Verifying NafterTokenCreatorRegistry
Pass - Verified: https://bscscan.com/address/0xfaf291208346054E2899f0FB9bf5303a0cF0B0F7#contracts

Verifying NafterMarketAuction
Pass - Verified: https://bscscan.com/address/0x20425f68C031e5A5dCb5a55cC81cF8DB1fE30BDA#contracts

Verifying NafterRoyaltyRegistry
Pass - Verified: https://bscscan.com/address/0xAaa27CF7E5331adE8Ae76213352666ACDcfc56dC#contracts

Successfully verified 5 contract(s).