SALUS SECURITY

DEC 2022

CODE SECURITY ASSESSMENT

NSWAP

Overview

Project Summary

- Name: Nswap
- Platform: BSC; ETH; ETHW
- Language: Solidity
- Codebase: https://github.com/Nswap/NswapExchange/tree/feat/batch
- Audit Range: contracts/exchange

Project Dashboard

Application Summary

Name	Nswap
Version	v2
Type	Solidity
Dates	Dec 14 2022
Logs	Nov 23 2022; Dec 14 2022

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	5
Total informational issues	30
Total	36

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Insecure usage of transferFrom() method	Medium	Coding Practice	Resolved
2	Initialize() method can be called by anyone	Low	Coding Practice	Acknowledged
3	SetProtocolFee() issue	Low	Coding Practice	Resolved
4	Unmatched checkOwner() description and functionality	Low	Coding Practice	Resolved
5	Insufficient event handler for _calculatefillAmount() method	Low	Coding Practice	Resolved
6	Off-chain risk	Low	Coding Practice	Acknowledged
7	Reentrancy lock needed for withdrawETH method	Informational	Coding Practice	Resolved
8	Redundant reentrancy lock	Informational	Coding Practice	Resolved
9	__Ownable_init() redundantly called	Informational	Coding Practice	Acknowledged
10	Clarified inheritance logic	Informational	Coding Practice	Resolved
11	Removable __Context_init()	Informational	Coding Practice	Acknowledged
12	Storage Gap correction	Informational	Coding Practice	Acknowledged
13	File naming issue	Informational	Coding Practice	Acknowledged
14	Variable naming issue	Informational	Coding Practice	Resolved
15	Function naming issue	Informational	Coding Practice	Resolved
16	Redundant SafeMathUpgradeable	Informational	Coding Practice	Resolved
17	Modifiable OpenZeppelin EIP712 import route	Informational	Coding Practice	Acknowledged
18	OrderValidator naming convention	Informational	Coding Practice	Resolved

19	_updateFilledAmount should check newAmount is not UINT256_MAX	Informational	Coding Practice	Resolved
20	Shadow variable	Informational	Coding Practice	Resolved
21	EnumerableSetUpgradeable.AddressSet	Informational	Coding Practice	Resolved
22	__CurrencyManager_init_unchained(address[] memory currencies) lacks overlapping information check	Informational	Coding Practice	Resolved
23	Redundant Null return value	Informational	Coding Practice	Resolved
24	Redundant code	Informational	Coding Practice	Acknowledged
25	Missing visibility specifier	Informational	Coding Practice	Resolved
26	Centralization	Informational	Coding Practice	Acknowledged
27	Enumeration optimization	Informational	Coding Practice	Resolved
28	Redundant code logic	Informational	Coding Practice	Resolved
29	Questionable getRoyalties method	Informational	Business Logic	Acknowledged
30	Questionable processing fee logic	Informational	Business Logic	Acknowledged
31	Unused imports could be removed	Informational	Coding Practice	Resolved
32	Description is missing	Informational	Coding Practice	Resolved
33	Incorrect description	Informational	Coding Practice	Resolved
34	abi.decode substitute	Informational	Coding Practice	Resolved
35	tryCatch substitute	Informational	Coding Practice	Acknowledged
36	Spelling error	Informational	Coding Practice	Resolved

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Insecure usage of transferFrom() method	
Severity: Medium	Category: Coding Practice
Target: - TransferExecutor.sol	

Description

```
if (!token.transferFrom(from, to, value)) {
revert TransferExecutor_ERC20_Failed();
}
```

The code indicated below would ideally check the return value of the transferFrom() function. And if the return value is false then this indicates that the token is not transferred and therefore would revert. Also if the return value is true then it indicates that the token is transferred successfully. However, in reality, lots of tokens do not necessarily follow the ERC20 interface, hence the transferFrom() method from most of the tokens does not have the return value. Such as the USDT situation on the Ethereum platform, in which case even if the transferFrom() method successfully carried out token transfer actions, still, there will be no return value. And the below code would still revert. Recommended action is given below as well.

Recommendation

Consider using the SafeERC20 protocol of OpenZeppelin, which can deal with the non-return value situation. More specifically, import SafeERC20Upgradeable from OpenZeppelin in the TransferExecutor.sol file like below.

```
Import "@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol";
```

Then add the below code at the beginning of TransferExecutor contract

```
using SafeERC20Upgradeable for IERC20Upgradeable;
```

Then change the original code to below.

```
token.safeTransferFrom(from, to, value);
```

2. Initialize() method can be called by anyone	
Severity: Medium	Category: Coding Practice
Target: - NswapExchange.sol	

Description

```
function initialize(
address[] memory currencies,
uint256 defaultProtocolFee,
address defaultFeeRecipient,
uint256 newRoyaltiesLimit,
string memory EIP712Name,
string memory EIP712Version
) public initializer {
}
```

initialize() function can be called by anyone in the implementation contract.

Recommendation

Consider adding _disableInitializers() in constructor.

3. SetProtocolFee() issue	
Severity: Low	Category: Coding Practice
Target: - ProtocolFeeManager.sol	

Description

```
function getProtocolFee(address wallet) public view returns (uint256) {
uint256 newProtocolFee = protocolFee[wallet];
if (newProtocolFee != 0) {
return newProtocolFee;
}return defaultProtocolFee;
}
function setProtocolFee(address wallet, uint256 newProtocolFee)
external
override
onlyOwner
{
if (newProtocolFee > LibPayInfo.Max_Protocol_Share){
revert ProtocolFeeManager_Fee_Exceed_Max_Protocol_Share();
}
protocolFee[wallet] = newProtocolFee;
emit EventUpdatedCustomProtocolFee(wallet, newProtocolFee);
}
```

The getProtocolFee(wallet) method below will first get the corresponding “protocolFee” variable from the mapping of the wallet. If the “protocolFee” is not 0, then the value will be returned, and if it is 0 then the “defaultProtocolFee” will be returned. Therefore, if the protocolFee[wallet] is 0 and the “defaultProtocolFee” is not 0 then the return value will not be 0.

Furthermore, the owner can access the “setProtocolFee(wallet, 0)” method and set the wallet address protocolFee to be 0. By doing so, “getProtocolFee(wallet)” will return “defaultProtocolFee” and the value will not be 0.

Recommendation

Reasonable actions should be taken based on the business logic of the project. if the “protocolFee[wallet] == 0” represents the “defaultProtocolFee” of some specific wallet address then it is recommended to add some description note in the file and also consider addresses that do not need “protocolFee”.

if the “protocolFee[wallet] == 0” represents the wallet addresses from which the “protocolFee” can be except, then the business logic of the “getProtocolFee” should be reconsidered.

4. Unmatched checkOwner() description and functionality	
Severity: Low	Category: Coding Practice
Target: - RoyaltiesManager.sol	

Description

```
function checkOwner(address token) internal view {
if (
(owner() != _msgSender()) &&
(OwnableUpgradeable(token).owner() != _msgSender())
) {
revert RoyaltiesRegistry_Not_Owner();
}
}
```

“checkOwner” function gets the token address owner by calling the OwnableUpgradeable(token).owner(), but this is only valid when there is the presence of the “owner()” method in the token contract.

Recommendation

It is recommended to add descriptions to specify from which tokens the checkOwner() method can be used upon.

5. Insufficient event handler for _calculatefillAmount() method	
Severity: Low	Category: Coding Practice
Target: - NswapExchange.sol	

Description

```
function _calculatefillAmount(
LibOrder.Order memory sell,
LibOrder.Order memory buy,
uint256 sellFilledAmount,uint256 buyFilledAmount
) internal pure returns (uint256 sellMakeAmount, uint256 buyMakeAmount) {
if (sell.saleKind == LibOrder.SaleKind.FixedPrice) {
if (buy.price.value < sell.price.value) {
revert NSwapExchange_Price_Mismatch();
}
uint256 sellRemaining = sell.nft.value - sellFilledAmount;
uint256 buyRemaining = buy.nft.value - buyFilledAmount;
sellMakeAmount = (sellRemaining > buyRemaining)
? buyRemaining
: sellRemaining;
buyMakeAmount = sell.price.value * sellMakeAmount;
}
}
```

The original code below only handles the situation where saleKind is a FixedPrice situation. If another situation was not supposed to be considered in this situation then a default revert branch can be added. The recommended code version is also given below.

Recommendation

```
function _calculatefillAmount(
LibOrder.Order memory sell,
LibOrder.Order memory buy,
uint256 sellFilledAmount,
uint256 buyFilledAmount
) internal pure returns (uint256 sellMakeAmount, uint256 buyMakeAmount) {
if (sell.saleKind == LibOrder.SaleKind.FixedPrice) {
if (buy.price.value < sell.price.value) {
revert NSwapExchange_Price_Mismatch();
}
uint256 sellRemaining = sell.nft.value - sellFilledAmount;
uint256 buyRemaining = buy.nft.value - buyFilledAmount;
sellMakeAmount = (sellRemaining > buyRemaining)
? buyRemaining
: sellRemaining;buyMakeAmount = sell.price.value * sellMakeAmount;
} else {
revert(); // or use your custom error
}
}
```

6. Off-chain risk	
Severity: Low	Category: Coding Practice
Target: - NswapExchange.sol	

Description

```
/**
* @param signature
The counterparty order signature of the tx sender.
*/
function matchOrders(
LibOrder.Order memory sell,
LibOrder.Order memory buy,
bytes memory signature
) external payable nonReentrant whenNotPaused {
...
}
```

Like indicated in the comment section of the below source code, matchOrders() method needs the counterpart signature, the initiative party has to obtain the other party's maker signature, and such procedure would pose a risk taken place off the chain.

2.3 Informational Findings

7. Reentrancy lock needed for withdrawETH method	
Severity: Informational	Category: Coding Practice
Target: - NswapExchange.sol	

Description

```
function withdrawETH(address payable recipient, uint256 amount)
external
onlyOwner
{
LibTransfer.transferEth(recipient, amount);
emit EventWithdrawETH(recipient, amount);
}
```

The withdrawETH corresponds to the following

```
library LibTransfer {
function transferEth(address to, uint256 value) internal {
(bool success, ) = to.call{value: value}("");
if(!success){
revert Transfer_Eth_Fail();}
}
}
```

The owner contract can set its address to the recipient and reenter the withdrawETH method. Although this reentrancy issue will not be an issue for funds but it will make the sequence go backwards. For example, we imagine a situation where the owner contract reenter the withdrawETH method and sequentially transferred 1,2 and 3 eth, then the contract event will be EventWithdrawETH(owner,3eth), EventWithdrawETH(owner,2 eth), EventWithdrawETH(owner,1 eth), which is contrary to our expectation. LibTransfer.transferEthcan use the call{gas:50000}() limitation.

Recommendation

It is recommended to add in the reentrancy lock, and since the NSwapExchange already inheritedOpenzeppelin's ReentrancyGuardUpgradeable, therefore need to change the following codes to the later one.

```
function withdrawETH(address payable recipient, uint256 amount)
external
onlyOwner
{...}
```

Revised:

```
function withdrawETH(address payable recipient, uint256 amount)
external
nonReentrant
onlyOwner
{...}
```

8. Redundant reentrancy lock	
Severity: Informational	Category: Coding Practice
Target: - NswapExchange.sol	

Description

It is redundant for the cancel() function to have reentrancy lock, as it did not call any exterior function.

Recommendation

It is recommended that the nonReentrant modifier can be removed from the original code L165-L169.

9. __Ownable_init() redundantly called

Severity: Informational	Category: Coding Practice
Target: - NswapExchange.sol - CurrencyManager.sol - ProtocolFeeManager.sol - RoyaltiesManager.sol	

Description

NSwapExchange contract has been called four times during initialization, and it is demonstrated below. It will not affect the contract functionality but there will be four events in the OwnershipTransferred(address,address).

Recommendation

1. Modify the base contract to have the initialization logic put into __ContractName_init_unchained() method.

2. call the initialization method of the __ContractName_init_unchained() method from NSwapExchange initialization method

3. The initialization method of mutually relied base contract to be called by derived NSwapExchange contract

```
└─ NSwapExchange::initialize
├─ NSwapExchange::__NswapExchange_init
│ ├─ OwnableUpgradeable::__Ownable_init
│ ├─ NswapTransferManager::__NswapTransferManager_init
│ │ ├─ TransferExecutor::__TransferExecutor_init
│ │ │ ├─ CurrencyManager::__CurrencyManager_init
│ │ │ │ └─ OwnableUpgradeable::__Ownable_init
│ │ ├─ ProtocolFeeManager::__ProtocolFeeManager_init
│ │ │ └─ OwnableUpgradeable::__Ownable_init
│ │ ├─ RoyaltiesManager::__RoyaltiesManager_init
│ │ │ └─ OwnableUpgradeable::__Ownable_init
```

10. Clarified inheritance logic	
Severity: Informational	Category: Coding Practice
Target: - NswapExchange.sol	

Description

NSwapExchange declared the inheritance from Ownable Upgradeable / Pausable Upgradeable, but these contracts inherited the ContextUpgradeable and therefore NSwapExchange also inherited the ContextUpgradeable logic. Since the NSwapExchange uses the method in ContextUpgradeable, such as: NSwapExchange.sol:L170,L225,L241 uses _msgSender().

```
function withdrawETH(address payable recipient, uint256 amount)
external
onlyOwner
{...}
```

Recommendation

It is recommended to add ContextUpgradeable into the NSwapExchange inheritance declaration to make the inheritance logic clearer.

change the above codes to the following

```
function withdrawETH(address payable recipient, uint256 amount)
external
nonReentrant
onlyOwner
{...}
```

11. Removable __Context_init()	
Severity: Informational	Category: Coding Practice
Target: - NswapExchange.sol - OrderValidator.sol	

Description

__Context_init() is a NULL function based on the original OpenZepplin source code, it does not contain the actionable initialization logic. Therefore, for the contracts inherited from the ContextUpgradeable, the initialization logic does not have to be added into the __Context_init().

Recommendation

remove the __Context_init() from the following two places

1. contracts/exchange/NSwapExchange.sol:L142

2. contracts/exchange/OrderValidator.sol:L48

12. Storage Gap correction	
Severity: Informational	Category: Coding Practice
Target: - NswapExchange.sol - CurrencyManager.sol - ProtocolFeeManager.sol - RoyaltiesManager.sol	

Description

_based on the Storage Gap document from OpenZeppelin about upgrading the contracts, `__gap` data set should have the contract take up 50 storage slots.

However,

1. CurrencyManager contract `_approvedCurrencies` variable takes up 2 slots, `__gap` preoccupied 50 slots,which in total is 52 slots

2. OrderValidator contract `fillsStat` variable takes up 1 slot,`__gap` preoccupied 50 slots,which in total is 51 slots

3. ProtocolFeeManager contract `defaultProtocolFee`、`protocolFee` `defaultProtocolFeeReceiver` three variables takes up 1 slot each,`__gap` preoccupied 50 slots,which in total is 53 slots

4. RoyaltiesManager contract `royaltiesByToken`、`royaltiesType`、`royaltiesLimit` three variables takes up 1 slot each,`__gap` preoccupied 50 slots,which in total is 53 slots

Recommendation

Modify `__gap` data set length to make the contract takes up 50 slots, more specifically

```
Change CurrencyManage.sol:L120
uint256[50] private __gap;
to
uint256[48] private __gap;
change OrderValidator.sol:L184
uint256[50] private __gap;
to
uint256[49] private __gap;
change ProtocolFeeManager.sol:L106
uint256[50] private __gap;
to
uint256[47] private __gap;
change RoyaltiesManager.sol:L348
uint256[50] private __gap;
to
uint256[47] private __gap;
```

13. File naming issue	
Severity: Informational	Category: Coding Practice
Target: - NswapExchange.sol - NswapTransferManager.sol	

Description

NswapTransferManager second letter ‘s’ is lowercase, but NSwapExchange second ‘s’ uses the uppercase.

Recommendation

Consider using the NSwap as the prefix. Modify it to NSwapTransferManager.sol.

14. Variable naming issue	
Severity: Informational	Category: Coding Practice
Target: - LibPayInfo.sol	

Description

Based on the Solidity official document Style Guide, constant variable has to follow uppercase and underscore naming convention such as UPPERCASE_WITH_UNDERSCORE but the code from contracts/exchange/lib/LibPayInfo.sol:L7-L9 `Total_share`, `Max_Royalty_share`, `Max_Protocol_Share` do not comply with the convention.

Recommendation

Consider changing these three variables into:

- TOTAL_SHARE
- MAX_ROYALTY_SHARE
- MAX_PROTOCOL_SHARE

15. Function naming issue	
Severity: Informational	Category: Coding Practice
Target: - NswapExchange.sol	

Description

Modify __NswapExchange_init to “ __NSwapExchange_init” Also modify __NswapExchange_init_unchained to “__NSwapExchange_init_unchained”

16. Redundant SafeMathUpgradeable	
Severity: Informational	Category: Coding Practice
Target: - NswapExchange.sol - NswapTransferManager.sol - OrderValidator.sol	

Description

```
using SafeMathUpgradeable for uint256;
```

According to the Openzeppelin SafeMath, version above 0.8.0 has built-in overflow checking mechanism which means the SafeMath/SafeMathUpgradeable won’t be needed.

Recommendation

Consider removing the SafeMathUpgradeable.

17. Modifiable OpenZeppelin EIP712 import route	
Severity: Informational	Category: Coding Practice
Target: - OrderValidator.sol	

Description

```
import"@openzeppelin/contracts/utils/cryptography/EIP712Upgradeable.sol;
```

From version 4.8.0 Openzepplin, the import route for EIP712 can be modified as follows.

Recommendation

Consider changing this to the following:

```
import"@openzeppelin/contracts/utils/cryptography/EIP712Upgradeable.sol;
```

18. OrderValidator naming convention	
Severity: Informational	Category: Coding Practice
Target: - OrderValidator.sol	

Description

```
uint256 private constant UINT256_MAX = type(uint256).max;
function _cancel(bytes32 orderHash) internal virtual {
fillsStat[orderHash] = UINT256_MAX;
}
OrderValidator.sol:L160-L171:
function _getFilledAmount(bytes32 orderHash)
internal
view
returns (uint256 filledAmount)
{
// Get has completed fill amount.
filledAmount = fillsStat[orderHash];
// Cancelled order cannot be matched.
if (filledAmount == UINT256_MAX) {
revert OrderValidator_Has_Canceled();
}
}
```

By the logic described in the following code section, the `UINT256_MAX` variable represents the canceled order's filledAmount, but it is not reflected in the variable's name.

Recommendation

Consider changing the `UINT256_MAX` to `CANCELLED_ORDER_FILL` and two other places in OrderValidator.sol:L61,L168 where function calls were made from.

19. _updateFilledAmount should check newAmount is not UINT256_MAX	
Severity: Informational	Category: Coding Practice
Target: - OrderValidator.sol	

Description

```
function _updateFilledAmount(uint256 newAmount, bytes32 orderHash)
internal
{
fillsStat[orderHash] = newAmount;
```

_updateFilledAmount allows the caller to put in any newAmount value, but in OrderValidator contract, `UINT256_MAX` or type(uint256).max value is used to represent the canceled order.

Recommendation

Consider adding a checker to determine the newly put-in newAmount value is not specially reserved. change the above code to:

```
function _updateFilledAmount(uint256 newAmount, bytes32 orderHash)
internal
{
require(newAmount != UINT256_MAX, "invalid newAmount");
fillsStat[orderHash] = newAmount;}
```

20. Shadow variable

Severity: Informational	Category: Coding Practice
Target: - NswapExchange.sol - NswapTransferManager.sol	

Description

```
uint256 public defaultProtocolFee;
```

NSwapExchange.sol:initialize(...) L116-L123

```
function initialize(
address[] memory currencies,
uint256 defaultProtocolFee,
address defaultFeeRecipient,
uint256 newRoyaltiesLimit,
string memory EIP712Name,
string memory EIP712Version
) public initializer {...}
```

NSwapExchange.sol:__NswapExchange_init(...) L134-L141

```
function __NswapExchange_init(
address[] memory currencies,
uint256 defaultProtocolFee,
address defaultFeeRecipient,
uint256 newRoyaltiesLimit,
string memory EIP712Name,
string memory EIP712Version
) internal onlyInitializing {...}
```

NswapTransferManager.sol:__NswapTransferManager_init(...) L27-L32

```
function __NswapTransferManager_init(
address[] memory currencies,
uint256 defaultProtocolFee,
address defaultFeeRecipient,
uint256 newRoyaltiesLimit
) internal onlyInitializing {...}
```

ProtocolFeeManager.sol:L18 declared the defaultProtocolFee variable. Same variable name is used elsewhere too. codes are illustrated below.

Recommendation

This won’t affect the contract logic but it would be better to use a different variable name in the following places. For example: change `defaultProtocolFee` to`initDefaultProtocolFee`
contracts/exchange/NSwapExchange.sol:L118,L126,L136,L150
contracts/exchange/NswapTransferManager.sol:L29,L34

21. EnumerableSetUpgradeable.AddressSet	
Severity: Informational	Category: Coding Practice
Target: - CurrencyManager.sol	

Description

According to Openzepplin EnumerableSetUpgradeable

- when using set.add(value), it will check whether the value is in the set, if it is then it will return false.

- when using set.remove(value), whether value is in the set, if it is then it will return false.

Therefore, when using EnumerableSetUpgradeable.AddressSet, add and remove functionality, there is no need to call “contains” to determine whether value is in the set or not, instead it is better to use the return value from add and remove.

Recommendation

Change :

CurrencyManager.sol:L56-L60

```
if (_approvedCurrencies.contains(currency)) {
revert CurrencyManager_Already_Approved();
}
_approvedCurrencies.add(currency);
```

To:

```
bool success = _approvedCurrencies.add(currency);
require(success); // or use your custom error
```

Change:

CurrencyManager.sol:L70-L74

```
if (!_approvedCurrencies.contains(currency)) {
revert CurrencyManager_Not_Approved();
}
_approvedCurrencies.remove(currency);
```

To:

```
bool success = _approvedCurrencies.remove(currency);
require(success); // or use your custom error
```

22. __CurrencyManager_init_unchained(address[] memory currencies) lacks overlapping information check	
Severity: Informational	Category: Coding Practice
Target: - CurrencyManager.sol	

Description

```
function __CurrencyManager_init_unchained(address[] memory currencies)
internal
onlyInitializing
{
for (uint256 i = 0; i < currencies.length; i++) {
address currency = currencies[i];
if (currency == address(0)) {
revert CurrencyManager_Currency_Zero_Address();
}
_approvedCurrencies.add(currency);
emit EventCurrencyApproved(currency);
}
}
```

If during the initialization process currencyA has been put-in twice, then it will send same EventCurrencyApproved(currencyA) event.

Recommendation

Change :

CurrencyManager.sol:L42-L44

```
_approvedCurrencies.add(currency);
emit EventCurrencyApproved(currency);
```

To:

```
bool success = _approvedCurrencies.add(currency);
if(success) {
emit EventCurrencyApproved(currency);
}
```

23. Redundant Null return value	
Severity: Informational	Category: Coding Practice
Target: - NswapTransferManager.sol	

Description

NswapTransferManager.sol:L171-L187

```
function getRoyaltiesByAssetType(LibAsset.AssetType memory matchNft)
internal
returns (LibPayInfo.PayInfo[] memory royalties)
{
if (
matchNft.assetClass == LibAsset.ERC1155_ASSET_CLASS ||
matchNft.assetClass == LibAsset.ERC721_ASSET_CLASS
) {
(address token, uint256 tokenId) = (
matchNft.tokenAddress,
matchNft.tokenID
);
return getRoyalties(token, tokenId);
}
LibPayInfo.PayInfo[] memory empty;
return empty;
}
function getRoyaltiesByAssetType(LibAsset.AssetType memory matchNft)
internal
returns (LibPayInfo.PayInfo[] memory royalties)
{...}
```

getRoyaltiesByAssetType under default situation will return LibPayInfo.PayInfo[], thus there is no need to construct null return situation.

Recommendation

remove NswapTransferManager.sol:L185-L186

```
LibPayInfo.PayInfo[] memory empty;
return empty;
```

24. Redundant code

Severity: Informational	Category: Coding Practice

Target:
- LibPayInfo.sol

Description

```
function hash(PayInfo memory info) internal pure returns (bytes32) {
return keccak256(abi.encode(TYPE_HASH, info.receiver, info.share));
}
```

Below code never used, hence can be removed.

25. Missing visibility specifier

Severity: Informational	Category: Coding Practice

Target:
- LibAsset.sol
- LibOrder.sol
- LibPayInfo.sol

Description

The following code is missing the visibility specifier.
LibAsset.sol

```
bytes32 constant ASSET_TYPE_TYPEHASH
bytes32 constant ASSET_TYPEHASH
```

LibOrder.sol

```
bytes32 constant ORDER_TYPEHASH
```

LibPayInfo.sol

```
uint256 constant Total_Share
uint256 constant Max_Royalty_Share
uint256 constant Max_Protocol_Share
```

26. Centralization	
Severity: Informational	Category: Coding Practice
Target: - NswapExchange.sol - NswapTransferManager.sol - ProtocolFeeManager.sol	

Description

ProtocolFeeManager.sol

```
function setDefaultProtocolFee(uint256 newDefaultProtocolFee)
function setDefaultProtocolFeeReceiver(
address payable newDefaultProtocolFeeReceiver
)
function setProtocolFee(address wallet, uint256 newProtocolFee)
```

CurrencyManager.sol

```
function approveCurrency(address currency)
function revokeCurrency(address currency)
```

NSwapExchange.sol

```
function withdrawETH(address payable recipient, uint256 amount)
function pause()
function unpause()
```

Owner can call and modify the essential elements for the following functions.

Recommendation

It is recommended to safely store private key, use multi signature wallet or use DAO as owner.

27. Enumeration optimization	
Severity: Informational	Category: Coding Practice
Target: - RoyaltiesManager.sol	

Description

```
function setRoyaltiesType
if (newRoyaltiesType <= 0 || newRoyaltiesType > _TOTAL_ROYALTY_TYPE) {
revert RoyaltiesRegistry_Wrong_RoyaltiesType();
}
```

Enumeration is not properly used in the below function.

Recommendation

It is recommended to directly use RoyaltyType declaration, which will automatically check whether parameters are within the enumeration range.

28. Redundant code logic	
Severity: Informational	Category: Coding Practice
Target: - RoyaltiesManager.sol	

Description

```
function getRoyalties(address token, uint256 tokenId)
public
override
returns (LibPayInfo.PayInfo[] memory)
{
RoyaltyType royaltyType = _getRoyaltiesType(token);
if (royaltyType == RoyaltyType.UNSET) {
royaltyType = _calculateRoyaltiesType(token);
_setRoyaltiesType(token, royaltyType); //Line 168
}
```

Enumeration is not properly used in the below function.

Recommendation

It is recommended to directly use RoyaltyType declaration, which will automatically check whether parameters are within the enumeration range.

29. Questionable getRoyalties method	
Severity: Informational	Category: Business Logic
Target: - RoyaltiesManager.sol	

Description

```
function getRoyalties
if (royaltyType == RoyaltyType.EIP2981) {
if (royaltyPoint >= LibPayInfo.Total_Share) {
revert RoyaltiesRegistry_ERC2981_Royalties_Exceed();
}
}
```

In EIP2981 logic branch,

```
if (royaltyPoint >= LibPayInfo.Total_Share),
```

Recommendation

consider whether the following comparison is necessary

```
royaltyPoint > LibPayInfo.Max_Royalty_Share
```

30. Questionable processing fee logic

Severity: Informational	Category: Business Logic
Target: - NswapTransferManager.sol	

Description

```
function transferProtocolFee
if (assetType.assetClass != LibAsset.ETH_ASSET_CLASS && fee > 0)
transfer(LibAsset.Asset(assetType, fee), from, defaultProtocolFeeReceiver);
}
```

When assetClass is ETH, the processing fee will stay in the current contract.

Recommendation

consider whether it should be transferred to defaultProtocolFeeReceiver.

31. Unused imports could be removed

Severity: Informational	Category: Coding Practice
Target: - NswapExchange.sol	

Description

```
import "./TransferExecutor.sol";
import "./interface/ITransferManager.sol";
```

NSwapExchange, line 31: 'TransferExecutor' could be removed. It's not used.

NSwapExchange, line 35: 'ITransferManager' could be removed. It's not used.

32. Description is missing	
Severity: Informational	Category: Coding Practice
Target: - TransferExecutor.sol	

Description

```
/**
* @dev Parse order extra data.
* @param order Order to parse.
* @return extraData
*/
function parse(LibOrder.Order memory order)
internal
pure
returns (Data memory extraData)
{
...
}
```

LibOrderData, line 22: The description for 'extraData' is missing.

33. Incorrect description

Severity: Informational	Category: Coding Practice
Target: - TransferExecutor.sol	

Description

```
/**
* @notice Transfers `value` amount of `token` from the `from` address to the
`to` address specified.
* @param token The NFT address
* @param from Source address
* @param to Target address
* @param value Transfer amount
*/
function erc20safeTransferFrom(IERC20Upgradeable token,
address from,
address to,
uint256 value
) internal {}
```

TransferExecutor, line 38: 'token' is not the nft address. It's erc20 address.

34. abi.decode substitute	
Severity: Informational	Category: Coding Practice
Target: - NSwapExchange.sol	

Description

SliceUint can be replaced by abi.decode without customization.

```
if (success) {
    buyMakeAmount = sliceUint(data, 32);
    buyMakeETHAmount += buyMakeAmount;
} else {
```

35. tryCatch substitute	
Severity: Informational	Category: Coding Practice
Target: - NSwapExchange.sol	

Description

Delegatecall is the method of calling the current contract. Our guess is that the transaction will not be rolled back in the case of revert, and the corresponding operation can be performed through the return value success. Consider replacing delegatecall with try catch.

```
(bool success, bytes memory data) = address(this).delegatecall(
    abi.encodeWithSignature(

"matchOrdersWithoutPayback((uint8,uint8,address,address,((bytes4,address,uint256),uint25
6),((bytes4,address,uint256),uint256),uint256,uint256,uint256,bytes4,bytes),(uint8,uint8
,address,address,((bytes4,address,uint256),uint256),((bytes4,address,uint256),uint256),u
int256,uint256,uint256,bytes4,bytes),bytes,uint256)",
        matchDetail.sellOrder,
        matchDetail.buyOrder,
        matchDetail.signature,
        msg.value.sub(buyMakeETHAmount)
    )
);
```

36. Spelling error	
Severity: Informational	Category: Coding Practice
Target: - LibOrder.sol	

Description

Collectino should be collection

```
// Currently supported kinds of sale: fixed price for item, fixed price for collectino and
dutch auctions
```