HALBORN

Bracket.fi - Channels and EpochChannels

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **December 19th, 2022 - December 28th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	12/29/2022	Manuel García
0.2	Document Updates	12/29/2022	Manuel García
0.3	Final Draft	12/29/2022	Manuel García
0.4	Draft Review	12/29/2022	Ataberk Yavuzer
0.5	Draft Review	01/02/2023	Piotr Cielas
0.6	Draft Review	01/02/2023	Gabi Urrutia
1.0	Remediation Plan	01/13/2023	Manuel García
1.1	Remediation Plan Review	01/16/2023	Roberto Reigada
1.2	Remediation Plan Review	01/16/2023	Piotr Cielas
1.3	Remediation Plan Review	01/16/2023	Gabi Urrutia
1.4	Epoch Channels Changes Review	01/28/2023	Manuel García

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Piotr Cielas	Halborn	Piotr.Cielas@halborn.com
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com
Manuel García	Halborn	Manuel.Diaz@halborn.com
Diogo Pereira	Halborn	Diogo.Pereira@halborn.com
Ataberk Yavuzer	Halborn	Ataberk.Yavuzer@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Bracket.fi Channels allows users to bet on the price of an asset to go out of the money (OTM) i.e. outside predefined volatility parameters. If the price stays within the range defined by the funder, the users can bet against it by paying a premium and receive rewards as long as the price stays in the money (ITM) for the duration of the channel.

Epoch Channels is based on Channels. The channels are defined by the contract owner and users are betting against each other, paying a predefined premium betting if the asset price stays ITM or goes OTM once the expiry time is reached. The premiums are then being redistributed to the winning side.

Bracket.fi engaged Halborn to conduct a security audit on their smart contracts beginning on December 19th, 2022 and ending on December 28th, 2022 . The security assessment was scoped to the programs provided in the bracketx-halborn GitLab repository. Commit hashes and further details can be found in the Scope section of this report.

1.2 AUDIT SUMMARY

The team at Halborn was provided 2 weeks for the engagement and assigned 2 full-time security engineers to audit the security of the programs in scope. The security engineers are blockchain and smart contract security experts with advanced penetration testing and smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of the audits is to:

- Identify potential security issues within the programs
- Ensure that smart contract functions operate as intended

In summary, Halborn identified some improvements to reduce the likelihood and impact of multiple risks, which has been successfully addressed by

Bracket.fi . The main ones are the following:

- When performing ERC20 token transfers, the returned value was not being checked. Some non-standard ERC20 tokens don't revert upon unsuccessful ERC20 transfer; however, they return false. In order to address this Bracket.fi wrapped the transfer functions in a require statement that reverted the transaction unless it returned true.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hot-spots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE, Ganache, Foundry)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW

3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

Code Repositories:

- BNPL Contracts Repository: bracketx-halborn
- Commit ID:
  - Audit: 0c8dc3237b7530e0cdbe0e68928adbc8b167971b
  - Remediation: 7aa426ee0a8f367aca5875267d9872bac05ec17d
  - Remediation: 6fe5621f23b7cd09bc4d3396b9573c18c154d36c
  - EpochChannels Changes: f9a1ffbfd4a65e9d68ea6d06d9c75d9007760c22
- Smart contracts in scope:
  1. Channel.sol
  2. CNFT.sol
  3. ChanOffers.sol
  4. ChanLib.sol
  5. ChanConfig.sol
  6. ChanMaxKeeper.sol
  7. ChanKeeper.sol
  8. EpochChannels.sol
  9. PriceProd.sol

Out-of-Scope:

- Third-party libraries and dependencies
- Economic attacks
- Attacks based on third-party oracle price manipulation.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	1	6

LIKELIHOOD

IMPACT

(HAL-01)				
(HAL-02)				
(HAL-03) (HAL-04) (HAL-05) (HAL-06) (HAL-07)				

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - ERC-20 TRANSFER IGNORING RETURN VALUE	Low	SOLVED - 01/13/2023
HAL02 - CENTRALIZATION RISK	Informational	SOLVED - 01/13/2023
HAL03 - IMMUTABLE DEPENDENCIES OF THE EPOCH CHANNELS CONTRACT	Informational	SOLVED - 01/13/2023
HAL04 - HARDCODED STATE VARIABLE	Informational	SOLVED - 01/13/2023
HAL05 - MISSING ZERO ADDRESS CHECK	Informational	SOLVED - 01/13/2023
HAL06 - FLOATING PRAGMA	Informational	SOLVED - 01/13/2023
HAL07 - > 0 CONSUMES MORE GAS THAN != 0 FOR UINTS	Informational	SOLVED - 01/13/2023

FINDINGS & TECH DETAILS

3.1 (HAL-01) ERC-20 TRANSFER IGNORING RETURN VALUE - LOW

Description:

In the Channel contract, when transferring the commission from the contract to the owner the transfer function is not wrapped with a require statement, meaning the return value of the transfer function is ignored.

While failed USDC transfers are reverted, it is important to note that some failed ERC-20 token transfers may return a boolean false instead, which may lead to processing transactions that otherwise should be reverted.

Listing 1: contracts/Channel.sol (Line 362)

```
362 IERC20(usdc).transfer(payable(owner()), v.commishUSDC);
```

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

Wrap the transfer function with a require statement to ensure failed ERC20 token transfers are always reverted.

Remediation Plan:

SOLVED: ERC20 token transfers were wrapped with a require statement which the transaction revert unless transfer returns true.
Commit ID: 6fe5621f23b7cd09bc4d3396b9573c18c154d36c.

3.2 (HAL-02) CENTRALIZATION RISK - INFORMATIONAL

Description:

The CNFT contract implements methods for the contract owner to mint and set policies. While the probability of this happening is very low, if the private keys of the owner account are compromised, a malicious user could potentially mint NFTs and use the setPolicy() function to steal other users' policies.

Listing 2: contracts/CNFT.sol (Line 56)

```
56 function setPolicy(uint256 tokenId, ChanLib.CPolicy memory policy)
57     external
58     onlyBrkt
59 {
60     Policies[tokenId] = policy;
61 }
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendation:

Remove the owner from the onlyBrkt() modifier.

Remediation Plan:

SOLVED: The contract owner was removed from the onlyBrkt modifier.
Commit ID: 7aa426ee0a8f367aca5875267d9872bac05ec17d.

3.3 (HAL-03) IMMUTABLE DEPENDENCIES OF THE EPOCH CHANNELS CONTRACT - INFORMATIONAL

Description:

In the EpochChannels contract, the addresses of the pricing contract and the Ethereum/USD Price Feed Oracle is set on initialization. However, these addresses cannot be changed later on if those contracts are deprecated or their private keys are compromised.

Listing 3: contracts/EpochChannels.sol (Line 70)

```
70 function initialize(address _pricing, address _eth) external
↳ initializer {
71     __Ownable_init();
72     __Pausable_init();
73     __ReentrancyGuard_init();
74     require(_pricing != address(0) && _eth != address(0), "ZADDR")
↳ ;
75     pricing = _pricing;
76     ETH = _eth;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Implement a function to change the dependencies addresses.

Remediation Plan:

SOLVED: The Bracket team added the cngLinks functions to change dependency addresses.

Commit ID: 7aa426ee0a8f367aca5875267d9872bac05ec17d.

3.4 (HAL-04) HARDCODED STATE VARIABLE - INFORMATIONAL

Description:

In the EpochChannels contract, the minimum USD amount a user can invest in an Epoch Channel MIN_BUY_USD is hardcoded into the initialize function. Moreover, unlike other parameters, there is no setter function to change this parameter. In case this value needs to be changed, the contract needs to be redeployed.

.

Listing 4: contracts/EpochChannels.sol (Line 70)

```
70 function initialize(address _pricing, address _eth) external
↳ initializer {
71     __Ownable_init();
72     __Pausable_init();
73     __ReentrancyGuard_init();
74     require(_pricing != address(0) && _eth != address(0), "ZADDR")
↳ ;
75     pricing = _pricing;
76     ETH = _eth;
77     COMMISH = 975e15;
78     // 1 - 2.5% in 18 digits
79     MAX_MULT = 19;
80     MAX_DELAY = 180;
81     // 3min
82     MIN_BUY_USD = 9e18;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Implement a setter function for the MIN_BUY_USD state variable.

Remediation Plan:

SOLVED: The Bracket team added the `setMinBuyUSD` function to change the `MIN_BUY_USD` state variable.
Commit ID: 7aa426ee0a8f367aca5875267d9872bac05ec17d.

3.5 (HAL-05) MISSING ZERO ADDRESS CHECK - INFORMATIONAL

Description:

Missing several zero address checks within the code base can lead to accidentally setting the `eth` or `channel` address to the `0` address.

Code Location:

- ChanConfig.sol: `setEth()`, `initialize()`
- CNFT.sol: `setChannel()`
- ChanMaxKeeper.sol: `initialize()`, `cngLinks`
- ChanKeeper.sol: `initialize()`, `cngLinks`
- ChanOffers.sol: `cngLinks()`
- PriceProd.sol: `initialize()`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Add a `require()` check for zero address for the addresses that impact core functionality and can lead to losses.

Remediation Plan:

SOLVED: The Bracket team implemented several zero address checks in above code locations.
Commit ID: 7aa426ee0a8f367aca5875267d9872bac05ec17d.

3.6 (HAL-06) FLOATING PRAGMA - INFORMATIONAL

Description:

The contracts in scope use the floating pragma `^0.8.12`. The contract should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, either an outdated compiler version that might introduce bugs that affect the contract system negatively or a pragma version too new which has not been extensively tested.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider locking the pragma version with known bugs for the compiler version. When possible, do not use floating pragma in the final live deployment. Specifying a fixed compiler version ensures that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Listing 5: contracts/EpochChannels.sol (Line 1)

```
1 // SPDX-License-Identifier: apache-2.0
2 pragma solidity 0.8.12;
```

Remediation Plan:

SOLVED: The pragma was modified to specify a fixed compiler version. Commit ID: 7aa426ee0a8f367aca5875267d9872bac05ec17d.

3.7 (HAL-07) > 0 CONSUMES MORE GAS THAN != 0 FOR UINTS - INFORMATIONAL

Description:

The use of `>` sign consumes more gas than `!=` sign. There are some cases where both can be used indistinctly, such as in unsigned integers where numbers cannot be negative, and as such, there is only a need to check that a number is not 0.

Code Location:

EpochChannel.sol:
- Line 133, 171 and 193: `require(_epoch > 0...`
- Line 230: `require(_invId > 0...`
- Line 248: `if (amt > 0){...`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Use `!=` instead of `>` in cases where both can be used.

Remediation Plans:

SOLVED: `> 0` was replaced by `!= 0` for uint comparison, where both can be used interchangeably.
Commit ID: 7aa426ee0a8f367aca5875267d9872bac05ec17d.

MANUAL TESTING

In the manual testing phase, the following scenarios were simulated. The scenarios listed below were selected based on the severity of the vulnerabilities Halborn was testing the program for.

4.1 RE-ENTRANCY ATTACKS

Description:

As the smart contracts transfer native assets on some of the functions tested, all functions were reviewed in depth to exclude the possibility of a re-entrancy attack.

Results:

All the functions were found to be using nonReentrant modifier from the OpenZeppelin's ReentrancyGuardUpgradeable library.

4.2 INVALID OFFERS AND CHANNELS

Description:

Both the Channels and Epoch channels setOffer and newChannel functions were fuzzed and tested manually to verify malformed offers cannot break functionality or impact user's funds in any way.

```
ADDING FUNDS:
Asset: 0x0000000000000000000000000000000000000001
Value: 50000000000.0
User: 0x33A4622B82D4c04a53e170c638B944ce27cffce3
Transaction sent: 0xf209ad06c4a7991823272089e71f8d44c1036d054c37d5a247965a619389c82b
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 19
  FakeUSDC.transfer confirmed   Block: 20   Gas used: 51067 (0.43%)

Transaction sent: 0x961798f7f00cffd1313e3966428a92a95a7d2e22b56e8f82599430df7259ecc7
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  FakeUSDC.increaseAllowance confirmed   Block: 21   Gas used: 45148 (0.38%)

Transaction sent: 0x748554460d1ce9e5ef730ee4bb64e8034a8fc40f45e0a7bee77e08280ff5536b
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  Channel.addAvailable confirmed   Block: 22   Gas used: 63816 (0.53%)

Current version: 1
CREATING OFFER:
Asset: [950, 950, 0, 0, 0, 0, 0, 0]
Amount: 0x0000000000000000000000000000000000000001
User: 0x33A4622B82D4c04a53e170c638B944ce27cffce3
Transaction sent: 0xa2151bc94f08d311bd289eb4a70e3048cda6a91958fa6f35f5bb38f3ce7a3791
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ChanOffers.setOffer confirmed (AMT)   Block: 23   Gas used: 34736 (0.29%)

CREATING NEW EPOCH:
Asset: 0x0000000000000000000000000000000000000001
Price: 5e+16
Time Open: 1672666161
Estimated Start: 1672666161
Time Expired: 1672680561
Transaction sent: 0x9e8676759a92f0ae458b41611756c158ecb03d440f1e399937e6d09e93e814bd
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 19
  EpochChannels.newChan confirmed (START)   Block: 20   Gas used: 30055 (0.25%)

CREATING NEW EPOCH:
Asset: 0x0000000000000000000000000000000000000001
Price: 5e+16
Time Open: 1672669832
Estimated Start: 1672669892
Time Expired: 1672670032
Transaction sent: 0x6b119bdb79bea729b87886e64ab45255be6b5d8b420e63fba751437d07ff10a2
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 19
  EpochChannels.newChan confirmed (EXP)   Block: 20   Gas used: 30241 (0.25%)
```

Results:

No Offers or Channels that could impact security or break any functionality in the contracts can be created. It was possible to create an offer with all amounts set to 0, however it did not have any impact as it could not be bought later on. It was also possible for the owner to create a new Channel with an opening time already past. However, this also did not have any impact as the start time was checked.

4.3 ADD AND REDUCE AVAILABLE FUNCTIONS

Description:

Both ‘addAvailable’ and ‘reduceAvailable’ as well as any function which change the FunderAvil state variable values were reviewed in-depth to ensure user’s balance is properly tracked and updated, and it cannot lead to loss of funds.

Results:

The funds were properly tracked over multiple calls to these functions. No security issues were found.

4.4 FUND MANAGEMENT

Description:

Halborn tested fund management to verify funder is unable to withdraw the money locked for a bought Policy.

```
Current version: 1
CREATING OFFER:
Asset: [100, 50, 0, 0, 0, 0, 0, 0]
Amount: 0x0000000000000000000000000000000000000001
User: 0x33A4622B82D4c04a53e170c638B944ce27cffce3
Transaction sent: 0x083941694ad32b4cfea836ea88ef5db5682b2284f869442d474201c721b0faa4
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ChanOffers.setOffer confirmed   Block: 23   Gas used: 130752 (1.09%)

BUYING OFFER:
Asset: 0x0000000000000000000000000000000000000001
Funder: 0x33A4622B82D4c04a53e170c638B944ce27cffce3
ID: 0
Max Price: 1e+21
Min Price: 1e+18
User: 0x0063046686E46Dc6F15918b61AE2B121458534a5
Value: 0.5ETH 500.0USD
Transaction sent: 0x2c8e8f389218c45fe881ea47391203f272b76a2f686c8a2f2dfba13e5aab9e2a
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  Channel.buyPolicy confirmed   Block: 24   Gas used: 465617 (3.88%)

Offer bought, NFT id: 1
REDUCING FUNDS:
Asset: 0x0000000000000000000000000000000000000001
Value: 50000000000.0
User: 0x33A4622B82D4c04a53e170c638B944ce27cffce3
Transaction sent: 0xff18142fed89518ede13a054c8ed7500e39c2a13a042a13b7bc6362d727cd46e
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 3
  Channel.reduceAvailable confirmed (RA1)   Block: 25   Gas used: 29496 (0.25%)
```

Results:

The funder is credited when the policy is bought and the balance is redistributed when the policy is claimed, making it impossible for the funder to retrieve those funds while the policy is open.

4.5 ACCESS CONTROLS

Description:

Access controls in every function in the contracts in scope were reviewed and checked.

Results:

All functions have proper access control. For example:

a) addAvailable and reduceAvailable functions only modify the state variable for the msg.sender address, preventing anyone from modifying other users balance.
b) The claim function in the Channels contract can be called by anyone. However, it can only be called once the policy is expired.
c) The startChan, claim and pay functions in the EpochChannels contract can be called by anyone. However, startChan can only be called after the estimated start time, while claim and pay can only be called once the channel is expired.
d) The mintCNFT and setPolicy function can only be called by the Channel contract or the contract owner.
e) The setChannel function can only be called by the contract owner.
f) The newChannel function can only be called by the contract owner.
g) All the non-view functions in the ChanConfig contract can only be called by the contract owner.

4.6 CLAIM AND PAY/WITHDRAW WORKFLOW

Description:

Claim and pay/withdraw workflows were reviewed in-depth. The simulated scenarios included trying to claim a non-expired policy, check proper fund returns upon canceling, etc.

```
ACCOUNT 0 BALANCE BEFORE BUY 100000000000000000000
ACCOUNT 1 BALANCE BEFORE BUY 100000000000000000000
ACCOUNT 2 BALANCE BEFORE BUY 100000000000000000000
ACCOUNT 3 BALANCE BEFORE BUY 100000000000000000000
ACCOUNT 4 BALANCE BEFORE BUY 100000000000000000000
ACCOUNT 5 BALANCE BEFORE BUY 100000000000000000000
ACCOUNT 6 BALANCE BEFORE BUY 100000000000000000000
ACCOUNT 7 BALANCE BEFORE BUY 100000000000000000000
ACCOUNT 8 BALANCE BEFORE BUY 100000000000000000000
CREATING NEW EPOCH:
Asset: 0x0000000000000000000000000000000000000001
Price: 5e+16
Time Open: 1672670914
Estimated Start: 1672674514
Time Expired: 1672681714
Transaction sent: 0x8b4b2a031204b7b6090397bd702ea8deb4e6783e3edd60bfe46ff4283f052219
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 19
  EpochChannels.newChan confirmed   Block: 20   Gas used: 143843 (1.20%)

Epoch Created with ID: 1
BETTING 5e+17 FOR IN THE CHANNEL ON EPOCH 1 WITH USER 0x33A4622B82D4c04a53e170c638B944ce27cffce3
Transaction sent: 0x6a93975d2ee04f8bf70fa5e86c622e38e386bdc435f15e3acad72890af58a17e
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  EpochChannels.buy confirmed   Block: 21   Gas used: 127835 (1.07%)

BETTING 5e+17 FOR IN THE CHANNEL ON EPOCH 1 WITH USER 0x0063046686E46Dc6F15918b61AE2B121458534a5
Transaction sent: 0xba304f9de93a75603b1799c50971b91bd1f98f388d33a0bf9fc6591412011f6d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  EpochChannels.buy confirmed   Block: 22   Gas used: 112835 (0.94%)

BETTING 5e+17 FOR IN THE CHANNEL ON EPOCH 1 WITH USER 0x21b42413bA931038f35e7A5224FaDb065d297Ba3
Transaction sent: 0x676d457cec414c35f0bc8260d780de8079ad6af52b008aa21c33fe2a579567d0
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  EpochChannels.buy confirmed   Block: 23   Gas used: 112835 (0.94%)

BETTING 5e+17 FOR IN THE CHANNEL ON EPOCH 1 WITH USER 0x46C0a5326E643E4f71D3149d50B48216e174Ae84
Transaction sent: 0x42af78287b75fa3f3f059c2d4b090c4874d802ffdfb4482f4a6c7805315a5b66
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  EpochChannels.buy confirmed   Block: 24   Gas used: 112835 (0.94%)

ACCOUNT 0 BALANCE AFTER BUY 100000000000000000000
ACCOUNT 1 BALANCE AFTER BUY 99950000000000000000
ACCOUNT 2 BALANCE AFTER BUY 99950000000000000000
ACCOUNT 3 BALANCE AFTER BUY 99950000000000000000
ACCOUNT 4 BALANCE AFTER BUY 99950000000000000000
ACCOUNT 5 BALANCE AFTER BUY 100000000000000000000
ACCOUNT 6 BALANCE AFTER BUY 100000000000000000000
ACCOUNT 7 BALANCE AFTER BUY 100000000000000000000
ACCOUNT 8 BALANCE AFTER BUY 100000000000000000000
Transaction sent: 0xe8c958a0cc03203ec7e8f253fe138a4d4bff6ffa4d22e709b6d4dfbda1305b56
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 20
  EpochChannels.startChan confirmed   Block: 25   Gas used: 120568 (1.00%)

PAYING 5 INVESTMENTS
Transaction sent: 0x3c14f52fea2c8a0b4ec25e6c81eb6e579b6964e00b09695ae0b2d22f5c10e30e
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 21
  EpochChannels.pay confirmed   Block: 26   Gas used: 53591 (0.45%)

Transaction sent: 0xa9e543ed52a7572fa4dfe4a899743042907bee0b2cde05b3d185b7f0610d58da
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 22
  EpochChannels.pay confirmed   Block: 27   Gas used: 53591 (0.45%)

Transaction sent: 0x3e47b377f7a692fbc179285695f59695f5f3442965d987bd57b2d0ae474c2b64
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 23
  EpochChannels.pay confirmed   Block: 28   Gas used: 53591 (0.45%)

Transaction sent: 0xc4a485b037e681e939019f3825bccbfca05bb055328de37ac53fbca7b5cc071c
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 24
  EpochChannels.pay confirmed   Block: 29   Gas used: 53591 (0.45%)

ACCOUNT 0 BALANCE AFTER CANCEL 100000000000000000000
ACCOUNT 1 BALANCE AFTER CANCEL 100000000000000000000
ACCOUNT 2 BALANCE AFTER CANCEL 100000000000000000000
ACCOUNT 3 BALANCE AFTER CANCEL 100000000000000000000
ACCOUNT 4 BALANCE AFTER CANCEL 100000000000000000000
ACCOUNT 5 BALANCE AFTER CANCEL 100000000000000000000
ACCOUNT 6 BALANCE AFTER CANCEL 100000000000000000000
ACCOUNT 7 BALANCE AFTER CANCEL 100000000000000000000
ACCOUNT 8 BALANCE AFTER CANCEL 100000000000000000000
```

Results:

All workflows work as intended and no security related issues were found.

4.7 UNCLAIMED POLICY TRANSFER

Description:

Transferring non-expired and unclaimed policies NFTs was tested as Bracket team requires it for regulatory compliance.

```
ADDING FUNDS:
Asset: 0x0000000000000000000000000000000000000001
Value: 50000000000.0
User: 0x33A4622B82D4c04a53e170c638B944ce27cffce3
Transaction sent: 0xf209ad06c4a7991823272089e71f8d44c1036d054c37d5a247965a619389c82b
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 19
  FakeUSDC.transfer confirmed   Block: 20   Gas used: 51067 (0.43%)

Transaction sent: 0x961798f7f00cffd1313e3966428a92a95a7d2e22b56e8f82599430df7259ecc7
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  FakeUSDC.increaseAllowance confirmed   Block: 21   Gas used: 45148 (0.38%)

Transaction sent: 0x748554460d1ce9e5ef730ee4bb64e8034a8fc40f45e0a7bee77e08280ff5536b
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  Channel.addAvailable confirmed   Block: 22   Gas used: 63816 (0.53%)

Current version: 1
CREATING OFFER:
Asset: [100, 50, 0, 0, 0, 0, 0, 0]
Amount: 0x0000000000000000000000000000000000000001
User: 0x33A4622B82D4c04a53e170c638B944ce27cffce3
Transaction sent: 0x083941694ad32b4cfea836ea88ef5db5682b2284f869442d474201c721b0faa4
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ChanOffers.setOffer confirmed   Block: 23   Gas used: 130752 (1.09%)

BUYING OFFER:
Asset: 0x0000000000000000000000000000000000000001
Funder: 0x33A4622B82D4c04a53e170c638B944ce27cffce3
ID: 0
Max Price: 1e+21
Min Price: 1e+18
User: 0x0063046686E46Dc6F15918b61AE2B121458534a5
Value: 0.5ETH 500.0USD
Transaction sent: 0x2c8e8f389218c45fe881ea47391203f272b76a2f686c8a2f2dfba13e5aab9e2a
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  Channel.buyPolicy confirmed   Block: 24   Gas used: 465617 (3.88%)

Offer bought, NFT id: 1
TRANSFERING NFT 1 FROM ACCOUNTS 2 TO ACCOUNTS 3
Transaction sent: 0xc2bbc5772dd5a21797bffab24921cbcfb86dc0385291c9249c9ca7680d6789d6
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 20
  CNFT.transferFrom confirmed (UNCLAIMED)   Block: 25   Gas used: 31047 (0.26%)
```

Results:

The CNFT contract successfully prevented users from transferring unclaimed policies in all transferFrom and safeTransferFrom functions.

4.8 PROPER CALCULATIONS

Description:

All calculations involving msg.value and user's balances, as well as price conversions from ETH to USD with the `multFactor` function were reviewed and fuzzed to detect miscalculations or value loss during decimal conversions.

Results:

Calculations were properly performed, no decimal precision loss or miscalculation was identified.

4.9 COMISSION HANDLING

Description:

Correct commission handling and collection was tested, including performing multiple actions while keeping track of the commissions collected.

Results:

All commissions are properly transferred to the contract owner.

AUTOMATED TESTING

5.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their ABIs and binary format, Slither was run against the contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Results:

Channel.sol

```
Reentrancy in Channel.buyPolicy(address,address,uint8,uint256,uint256) (contracts/Channel.sol#198-310):
        External calls:
        - (success2,data2) = address(ibnft).call(abi.encodeWithSignature(mintCNFT(address,uint256),msg.sender,v.premiumUSD)) (contracts/Channel.sol#254-256)
        External calls sending eth:
        - address(address(owner())).transfer(v.commishETH) (contracts/Channel.sol#247)
        State variables written after the call(s):
        - FunderAvil[_asset][_funder] = v.availUSD.div(1e12) (contracts/Channel.sol#284)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

Channel.claim(uint256) (contracts/Channel.sol#326-383) ignores return value by IERC20(usdc).transfer(address(owner()),v.commishUSDC) (contracts/Channel.sol#362)
Channel.withdraw() (contracts/Channel.sol#386-399) ignores return value by IERC20(usdc).transfer(msg.sender,amountUSDC) (contracts/Channel.sol#396)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Channel.claim(uint256) (contracts/Channel.sol#326-383) uses a dangerous strict equality:
        - require(bool,string)(v.claimTime == v.p.timeExpired || outOfChannel,TOOEARLY) (contracts/Channel.sol#338)
Channel.claim(uint256) (contracts/Channel.sol#326-383) uses a dangerous strict equality:
        - v.preClaimAmtUSD == 0 (contracts/Channel.sol#348)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Channel.buyPolicy(address,address,uint8,uint256,uint256).v (contracts/Channel.sol#205) is a local variable never initialized
Channel.claim(uint256).v (contracts/Channel.sol#327) is a local variable never initialized
ChanOffers.setOffer(address,uint16[8],uint16).i_scope_0 (contracts/ChanOffers.sol#96) is a local variable never initialized
ChanOffers.setOffer(address,uint16[8],uint16).i (contracts/ChanOffers.sol#87) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

CNFT.setChannel(address) (contracts/CNFT.sol#44-46) should emit an event for:
        - channel = _channel (contracts/CNFT.sol#45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

CNFT.setChannel(address)._channel (contracts/CNFT.sol#44) lacks a zero-check on :
                - channel = _channel (contracts/CNFT.sol#45)
ChanConfig.initialize(address).eth (contracts/ChanConfig.sol#33) lacks a zero-check on :
                - ETH = eth (contracts/ChanConfig.sol#35)
ChanConfig.setETH(address).eth (contracts/ChanConfig.sol#49) lacks a zero-check on :
                - ETH = eth (contracts/ChanConfig.sol#50)
ChanOffers.cngLinks(address,address)._pricing (contracts/ChanOffers.sol#52) lacks a zero-check on :
                - pricing = _pricing (contracts/ChanOffers.sol#53)
ChanOffers.cngLinks(address,address)._config (contracts/ChanOffers.sol#52) lacks a zero-check on :
                - config = _config (contracts/ChanOffers.sol#54)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Channel.addAvailable(address,uint256) (contracts/Channel.sol#151-162):
        External calls:
        - require(bool,string)(IERC20(usdc).transferFrom(msg.sender,address(this),_usdcValue),T1) (contracts/Channel.sol#154-157)
        State variables written after the call(s):
        - FunderAvil[asset][msg.sender] = priorAmt.add(_usdcValue) (contracts/Channel.sol#160)
Reentrancy in Channel.buyPolicy(address,address,uint8,uint256,uint256) (contracts/Channel.sol#198-310):
        External calls:
        - (success2,data2) = address(ibnft).call(abi.encodeWithSignature(mintCNFT(address,uint256),msg.sender,v.premiumUSD)) (contracts/Channel.sol#254-256)
        External calls sending eth:
        - address(address(owner())).transfer(v.commishETH) (contracts/Channel.sol#247)
        State variables written after the call(s):
        - pendingETHwithdrawals[owner()] = pendingETHwithdrawals[owner()].add(v.commishETH) (contracts/Channel.sol#275-277)
        - pendingETHwithdrawals[_funder] = pendingETHwithdrawals[_funder].add(v.funderETHpay) (contracts/Channel.sol#279-281)
Reentrancy in Channel.claim(uint256) (contracts/Channel.sol#326-383):
```

```
        External calls:
        - CNFT(ibnft).setClaimed(_policyNum) (contracts/Channel.sol#346)
        - IERC20(usdc).transfer(address(owner()),v.commishUSDC) (contracts/Channel.sol#362)
        State variables written after the call(s):
        - FunderAvil[v.p.asset][v.p.funder] = FunderAvil[v.p.asset][v.p.funder].add(v.addAvailUSDC) (contracts/Channel.sol#369)
        - pendingUSDCwithdrawals[v.owner] = pendingUSDCwithdrawals[v.owner].add(v.claimAmtUSDC) (contracts/Channel.sol#365)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Channel.addAvailable(address,uint256) (contracts/Channel.sol#151-162):
        External calls:
        - require(bool,string)(IERC20(usdc).transferFrom(msg.sender,address(this),_usdcValue),T1) (contracts/Channel.sol#154-157)
        Event emitted after the call(s):
        - CAvailableEv(asset,msg.sender,priorAmt,FunderAvil[asset][msg.sender]) (contracts/Channel.sol#161)
Reentrancy in Channel.buyPolicy(address,address,uint8,uint256,uint256) (contracts/Channel.sol#198-310):
        External calls:
        - (success2,data2) = address(ibnft).call(abi.encodeWithSignature(mintCNFT(address,uint256),msg.sender,v.premiumUSD)) (contracts/Channel.sol#254-256)
        - CNFT(ibnft).setPolicy(v.id,p) (contracts/Channel.sol#285)
        External calls sending eth:
        - address(address(owner())).transfer(v.commishETH) (contracts/Channel.sol#247)
        Event emitted after the call(s):
        - CPolicyxBuyEv1(v.id,msg.sender,_funder,_asset,_offerId,v.verData.mult[_offerId],v.hiPriceUSD,v.lowPriceUSD,msg.value,v.premiumUSD) (contracts/Channel.sol#287-297)
        - CPolicyxBuyEv2(v.id,msg.sender,_funder,_asset,_offerId,block.timestamp,v.timeExpired,v.commishETH,v.priceCoin,v.funderETHpay) (contracts/Channel.sol#299-309)
Reentrancy in Channel.claim(uint256) (contracts/Channel.sol#326-383):
        External calls:
        - CNFT(ibnft).setClaimed(_policyNum) (contracts/Channel.sol#346)
        - IERC20(usdc).transfer(address(owner()),v.commishUSDC) (contracts/Channel.sol#362)
        Event emitted after the call(s):
        - CClaimEv(v.p.funder,_policyNum,v.p.asset,v.owner,v.claimTime,v.claimPrice,v.claimAmtUSDC,v.commishUSDC,v.addAvailUSDC) (contracts/Channel.sol#372-382)
Reentrancy in Channel.withdraw() (contracts/Channel.sol#386-399):
        External calls:
        - IERC20(usdc).transfer(msg.sender,amountUSDC) (contracts/Channel.sol#396)
        External calls sending eth:
        - address(msg.sender).transfer(amountETH) (contracts/Channel.sol#392)
        Event emitted after the call(s):
        - CWithdrawEv(msg.sender,amountETH,amountUSDC) (contracts/Channel.sol#398)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Channel.buyPolicy(address,address,uint8,uint256,uint256) (contracts/Channel.sol#198-310) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(v.premiumUSD >= ChanConfig(config).MIN_PREM(),MIN) (contracts/Channel.sol#211)
        - require(bool,string)(v.o.pctOut[_offerId] != 0,NO OFFER1) (contracts/Channel.sol#218)
        - require(bool,string)(v.priceCoin <= _maxAssetPrice,SLIP1) (contracts/Channel.sol#224)
        - require(bool,string)(v.priceCoin >= _minAssetPrice,SLIP2) (contracts/Channel.sol#225)
        - require(bool,string)(v.verData.term[_offerId] != uint32(0),NO OFFER2) (contracts/Channel.sol#228)
        - require(bool,string)(v.availUSD >= v.maxClaimUSD,AVAIL) (contracts/Channel.sol#243)
        - require(bool,string)(success2,IMF1) (contracts/Channel.sol#257)
        - assert(bool)(v.id != 0) (contracts/Channel.sol#259)
Channel.claim(uint256) (contracts/Channel.sol#326-383) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(v.p.claimed == false,CLAIMED) (contracts/Channel.sol#329)
        - outOfChannel = (v.claimPrice < v.p.lowPriceUSD) || (v.claimPrice > v.p.hiPriceUSD) (contracts/Channel.sol#336)
        - require(bool,string)(v.claimTime >= v.p.timeExpired || outOfChannel,TOOEARLY) (contracts/Channel.sol#338)
        - require(bool,string)(v.preClaimAmtUSD <= v.p.maxClaimUSD,MAXCLAIM) (contracts/Channel.sol#343)
        - v.preClaimAmtUSD == 0 (contracts/Channel.sol#348)
        - v.addAvailUSDC != 0 (contracts/Channel.sol#367)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

CNFT.safeTransferFrom(address,address,uint256,bytes) (contracts/CNFT.sol#98-106) compares to a boolean constant:
        -require(bool,string)(tokenId >= 1 && tokenId <= _tokenIds.current() && Policies[tokenId].claimed == true,UNCLAIMED) (contracts/CNFT.sol#104)
Channel.claim(uint256) (contracts/Channel.sol#326-383) compares to a boolean constant:
        -require(bool,string)(v.p.claimed == false,CLAIMED) (contracts/Channel.sol#329)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Pragma version^0.8.12 (contracts/CNFT.sol#2) allows old versions
Pragma version^0.8.12 (contracts/ChanConfig.sol#3) allows old versions
Pragma version^0.8.12 (contracts/ChanLib.sol#3) allows old versions
Pragma version^0.8.12 (contracts/ChanOffers.sol#3) allows old versions
Pragma version^0.8.12 (contracts/Channel.sol#2) allows old versions
Pragma version^0.8.12 (contracts/IPricing.sol#3) allows old versions
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Channel.buyPolicy(address,address,uint8,uint256,uint256) (contracts/Channel.sol#198-310):
        - (success2,data2) = address(ibnft).call(abi.encodeWithSignature(mintCNFT(address,uint256),msg.sender,v.premiumUSD)) (contracts/Channel.sol#254-256)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter CNFT.setChannel(address)._channel (contracts/CNFT.sol#44) is not in mixedCase
Variable CNFT.Policies (contracts/CNFT.sol#23) is not in mixedCase
Variable CNFT._tokenIds (contracts/CNFT.sol#25) is not in mixedCase
Function ChanConfig.setMIN_AVAIL_NEW(uint256) (contracts/ChanConfig.sol#84-86) is not in mixedCase
Function ChanConfig.setMIN_PREM(uint256) (contracts/ChanConfig.sol#88-90) is not in mixedCase
Function ChanConfig.setPREM_COMMISH(uint256) (contracts/ChanConfig.sol#92-94) is not in mixedCase
Function ChanConfig.setCLAIM_COMMISH(uint256) (contracts/ChanConfig.sol#96-98) is not in mixedCase
Variable ChanConfig.MIN_AVAIL_NEW (contracts/ChanConfig.sol#12) is not in mixedCase
Variable ChanConfig.MIN_PREM (contracts/ChanConfig.sol#13) is not in mixedCase
Variable ChanConfig.PREM_COMMISH (contracts/ChanConfig.sol#14) is not in mixedCase
Variable ChanConfig.CLAIM_COMMISH (contracts/ChanConfig.sol#15) is not in mixedCase
Variable ChanConfig.ETH (contracts/ChanConfig.sol#16) is not in mixedCase
Variable ChanConfig.Vers (contracts/ChanConfig.sol#17) is not in mixedCase
Parameter ChanOffers.initialize(address,address)._pricing (contracts/ChanOffers.sol#42) is not in mixedCase
Parameter ChanOffers.initialize(address,address)._config (contracts/ChanOffers.sol#42) is not in mixedCase
Parameter ChanOffers.cngLinks(address,address)._pricing (contracts/ChanOffers.sol#52) is not in mixedCase
Parameter ChanOffers.cngLinks(address,address)._config (contracts/ChanOffers.sol#52) is not in mixedCase
Variable ChanOffers.Offers (contracts/ChanOffers.sol#25) is not in mixedCase
Struct Channel.buyLocals (contracts/Channel.sol#177-192) is not in CapWords
Struct Channel.claimLocals (contracts/Channel.sol#312-321) is not in CapWords
Parameter Channel.initialize(address,address,address,address,address)._ibnft (contracts/Channel.sol#89) is not in mixedCase
Parameter Channel.initialize(address,address,address,address,address)._usdc (contracts/Channel.sol#90) is not in mixedCase
Parameter Channel.initialize(address,address,address,address,address)._pricing (contracts/Channel.sol#91) is not in mixedCase
Parameter Channel.initialize(address,address,address,address,address)._offers (contracts/Channel.sol#92) is not in mixedCase
Parameter Channel.initialize(address,address,address,address,address)._config (contracts/Channel.sol#93) is not in mixedCase
Parameter Channel.cngLinks(address,address,address,address,address)._ibnft (contracts/Channel.sol#112) is not in mixedCase
Parameter Channel.cngLinks(address,address,address,address,address)._usdc (contracts/Channel.sol#113) is not in mixedCase
Parameter Channel.cngLinks(address,address,address,address,address)._pricing (contracts/Channel.sol#114) is not in mixedCase
Parameter Channel.cngLinks(address,address,address,address,address)._offers (contracts/Channel.sol#115) is not in mixedCase
Parameter Channel.cngLinks(address,address,address,address,address)._config (contracts/Channel.sol#116) is not in mixedCase
Parameter Channel.getWithdrawAmts(address)._acct (contracts/Channel.sol#140) is not in mixedCase
Parameter Channel.addAvailable(address,uint256)._usdcValue (contracts/Channel.sol#151) is not in mixedCase
Parameter Channel.reduceAvailable(address,uint256)._newValue (contracts/Channel.sol#167) is not in mixedCase
Parameter Channel.buyPolicy(address,address,uint8,uint256,uint256)._asset (contracts/Channel.sol#199) is not in mixedCase
Parameter Channel.buyPolicy(address,address,uint8,uint256,uint256)._funder (contracts/Channel.sol#200) is not in mixedCase
Parameter Channel.buyPolicy(address,address,uint8,uint256,uint256)._offerId (contracts/Channel.sol#201) is not in mixedCase
Parameter Channel.buyPolicy(address,address,uint8,uint256,uint256)._maxAssetPrice (contracts/Channel.sol#202) is not in mixedCase
Parameter Channel.buyPolicy(address,address,uint8,uint256,uint256)._minAssetPrice (contracts/Channel.sol#203) is not in mixedCase
Parameter Channel.claim(uint256)._policyNum (contracts/Channel.sol#326) is not in mixedCase
Variable Channel.FunderAvil (contracts/Channel.sol#28) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in Channel.buyPolicy(address,address,uint8,uint256,uint256) (contracts/Channel.sol#198-310):
        External calls:
        - address(address(owner())).transfer(v.commishETH) (contracts/Channel.sol#247)
        State variables written after the call(s):
        - FunderAvil[_asset][_funder] = v.availUSD.div(1e12) (contracts/Channel.sol#284)
        - pendingETHwithdrawals[owner()] = pendingETHwithdrawals[owner()].add(v.commishETH) (contracts/Channel.sol#275-277)
        - pendingETHwithdrawals[_funder] = pendingETHwithdrawals[_funder].add(v.funderETHpay) (contracts/Channel.sol#279-281)
        Event emitted after the call(s):
        - CPolicyxBuyEv1(v.id,msg.sender,_funder,_asset,_offerId,v.verData.mult[_offerId],v.hiPriceUSD,v.lowPriceUSD,msg.value,v.premiumUSD) (contracts/Channel.sol#287-297)
        - CPolicyxBuyEv2(v.id,msg.sender,_funder,_asset,_offerId,block.timestamp,v.timeExpired,v.commishETH,v.priceCoin,v.funderETHpay) (contracts/Channel.sol#299-309)
Reentrancy in Channel.withdraw() (contracts/Channel.sol#386-399):
        External calls:
        - address(msg.sender).transfer(amountETH) (contracts/Channel.sol#392)
        State variables written after the call(s):
        - pendingUSDCwithdrawals[msg.sender] = 0 (contracts/Channel.sol#395)
        Event emitted after the call(s):
        - CWithdrawEv(msg.sender,amountETH,amountUSDC) (contracts/Channel.sol#398)
```

CNFT.sol

```
CNFT.setChannel(address) (contracts/CNFT.sol#44-46) should emit an event for:
        - channel = _channel (contracts/CNFT.sol#45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

CNFT.setChannel(address)._channel (contracts/CNFT.sol#44) lacks a zero-check on :
                - channel = _channel (contracts/CNFT.sol#45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

CNFT.safeTransferFrom(address,address,uint256,bytes) (contracts/CNFT.sol#98-106) compares to a boolean constant:
        -require(bool,string)(tokenId >= 1 && tokenId <= _tokenIds.current() && Policies[tokenId].claimed == true,UNCLAIMED) (contracts/CNFT.sol#104)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

ChanLib.multFactor(uint256,uint256) (contracts/ChanLib.sol#37-43) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.12 (contracts/CNFT.sol#2) allows old versions
Pragma version^0.8.12 (contracts/ChanLib.sol#3) allows old versions
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter CNFT.setChannel(address)._channel (contracts/CNFT.sol#44) is not in mixedCase
Variable CNFT.Policies (contracts/CNFT.sol#23) is not in mixedCase
Variable CNFT._tokenIds (contracts/CNFT.sol#25) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
./contracts/CNFT.sol analyzed (19 contracts with 81 detectors), 10 result(s) found
```

ChanOffers.sol

```
ChanOffers.setOffer(address,uint16[8],uint16).i (contracts/ChanOffers.sol#87) is a local variable never initialized
ChanOffers.setOffer(address,uint16[8],uint16).i_scope_0 (contracts/ChanOffers.sol#96) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ChanConfig.initialize(address).eth (contracts/ChanConfig.sol#33) lacks a zero-check on :
                - ETH = eth (contracts/ChanConfig.sol#35)
ChanConfig.setETH(address).eth (contracts/ChanConfig.sol#49) lacks a zero-check on :
                - ETH = eth (contracts/ChanConfig.sol#50)
ChanOffers.cngLinks(address,address)._pricing (contracts/ChanOffers.sol#52) lacks a zero-check on :
                - pricing = _pricing (contracts/ChanOffers.sol#53)
ChanOffers.cngLinks(address,address)._config (contracts/ChanOffers.sol#52) lacks a zero-check on :
                - config = _config (contracts/ChanOffers.sol#54)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

ChanLib.multFactor(uint256,uint256) (contracts/ChanLib.sol#37-43) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.12 (contracts/ChanConfig.sol#3) allows old versions
Pragma version^0.8.12 (contracts/ChanLib.sol#3) allows old versions
Pragma version^0.8.12 (contracts/ChanOffers.sol#3) allows old versions
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function ChanConfig.setMIN_AVAIL_NEW(uint256) (contracts/ChanConfig.sol#84-86) is not in mixedCase
Function ChanConfig.setMIN_PREM(uint256) (contracts/ChanConfig.sol#88-90) is not in mixedCase
Function ChanConfig.setPREM_COMMISH(uint256) (contracts/ChanConfig.sol#92-94) is not in mixedCase
Function ChanConfig.setCLAIM_COMMISH(uint256) (contracts/ChanConfig.sol#96-98) is not in mixedCase
Variable ChanConfig.MIN_AVAIL_NEW (contracts/ChanConfig.sol#12) is not in mixedCase
Variable ChanConfig.MIN_PREM (contracts/ChanConfig.sol#13) is not in mixedCase
Variable ChanConfig.PREM_COMMISH (contracts/ChanConfig.sol#14) is not in mixedCase
Variable ChanConfig.CLAIM_COMMISH (contracts/ChanConfig.sol#15) is not in mixedCase
Variable ChanConfig.ETH (contracts/ChanConfig.sol#16) is not in mixedCase
Variable ChanConfig.Vers (contracts/ChanConfig.sol#17) is not in mixedCase
Parameter ChanOffers.initialize(address,address)._pricing (contracts/ChanOffers.sol#42) is not in mixedCase
Parameter ChanOffers.initialize(address,address)._config (contracts/ChanOffers.sol#42) is not in mixedCase
Parameter ChanOffers.cngLinks(address,address)._pricing (contracts/ChanOffers.sol#52) is not in mixedCase
Parameter ChanOffers.cngLinks(address,address)._config (contracts/ChanOffers.sol#52) is not in mixedCase
Variable ChanOffers.Offers (contracts/ChanOffers.sol#25) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
./contracts/ChanOffers.sol analyzed (11 contracts with 81 detectors), 26 result(s) found
```

ChanLib.sol

```
ChanLib.multFactor(uint256,uint256) (contracts/ChanLib.sol#37-43) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.12 (contracts/ChanLib.sol#3) allows old versions
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
./contracts/ChanLib.sol analyzed (3 contracts with 81 detectors), 3 result(s) found
```

ChanConfig.sol

```
ChanConfig.initialize(address).eth (contracts/ChanConfig.sol#33) lacks a zero-check on :
                - ETH = eth (contracts/ChanConfig.sol#35)
ChanConfig.setETH(address).eth (contracts/ChanConfig.sol#49) lacks a zero-check on :
                - ETH = eth (contracts/ChanConfig.sol#50)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

ChanLib.multFactor(uint256,uint256) (contracts/ChanLib.sol#37-43) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.12 (contracts/ChanConfig.sol#3) allows old versions
Pragma version^0.8.12 (contracts/ChanLib.sol#3) allows old versions
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function ChanConfig.setMIN_AVAIL_NEW(uint256) (contracts/ChanConfig.sol#84-86) is not in mixedCase
Function ChanConfig.setMIN_PREM(uint256) (contracts/ChanConfig.sol#88-90) is not in mixedCase
Function ChanConfig.setPREM_COMMISH(uint256) (contracts/ChanConfig.sol#92-94) is not in mixedCase
Function ChanConfig.setCLAIM_COMMISH(uint256) (contracts/ChanConfig.sol#96-98) is not in mixedCase
Variable ChanConfig.MIN_AVAIL_NEW (contracts/ChanConfig.sol#12) is not in mixedCase
Variable ChanConfig.MIN_PREM (contracts/ChanConfig.sol#13) is not in mixedCase
Variable ChanConfig.PREM_COMMISH (contracts/ChanConfig.sol#14) is not in mixedCase
Variable ChanConfig.CLAIM_COMMISH (contracts/ChanConfig.sol#15) is not in mixedCase
Variable ChanConfig.ETH (contracts/ChanConfig.sol#16) is not in mixedCase
Variable ChanConfig.Vers (contracts/ChanConfig.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
./contracts/ChanConfig.sol analyzed (7 contracts with 81 detectors), 16 result(s) found
```

ChanMaxKeeper.sol

```
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
./contracts/ChanMaxKeeper.sol analyzed (29 contracts with 81 detectors), 1 result(s) found
```

ChanKeeper.sol

```
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
./contracts/ChanKeeper.sol analyzed (29 contracts with 81 detectors), 1 result(s) found
```

PriceProd.sol

AUTOMATED TESTING

```
PriceProd.setGracePeriod(uint256) (contracts/PriceProd.sol#35-37) should emit an event for:
        - GRACE_PERIOD_TIME = gp (contracts/PriceProd.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

PriceProd.getLatestPrice(address) (contracts/PriceProd.sol#62-117) uses timestamp for comparisons
        Dangerous comparisons:
        - timeSinceUp <= GRACE_PERIOD_TIME (contracts/PriceProd.sol#81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

PriceProd.getLatestPrice(address) (contracts/PriceProd.sol#62-117) compares to a boolean constant:
        -Depeg[asset] == true && price >= 1e8 (contracts/PriceProd.sol#113)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Pragma version^0.8.12 (contracts/IPricing.sol#3) allows old versions
Pragma version^0.8.12 (contracts/PriceProd.sol#2) allows old versions
solc-0.8.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable PriceProd.GRACE_PERIOD_TIME (contracts/PriceProd.sol#15) is not in mixedCase
Variable PriceProd.Depeg (contracts/PriceProd.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
./contracts/PriceProd.sol analyzed (9 contracts with 81 detectors), 8 result(s) found
```

5.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the smart contracts and sent the compiled results to the analyzers in order to locate any vulnerabilities.

Results:

Channel.sol

Report for Jobs/Bracket/brownie/contracts/Channel.sol
https://dashboard.mythx.io/#/console/analyses/17b1ef46-0245-4663-acde-0ee55c34e9ef

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

CNFT.sol

Report for Jobs/Bracket/brownie/contracts/CNFT.sol
https://dashboard.mythx.io/#/console/analyses/33ae3cb5-4af7-4290-ac39-35417818f4c8

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

ChanOffers.sol

Report for Jobs/Bracket/brownie/contracts/ChanOffers.sol
https://dashboard.mythx.io/#/console/analyses/3d2f61da-9cbb-457a-b638-655118a71cb4

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
91	(SWC-113) DoS with Failed Call	Medium	Multiple calls are executed in the same transaction.

ChanLib.sol

Report for Jobs/Bracket/brownie/contracts/ChanLib.sol
https://dashboard.mythx.io/#/console/analyses/d15e26e9-d469-4da5-8bdf-c5450eed4ded

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

ChanConfig.sol

^[^[Report for Jobs/Bracket/brownie/contracts/ChanConfig.sol
https://dashboard.mythx.io/#/console/analyses/1e98413e-87df-4104-9b3b-be824f3c22b6

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
17	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

ChanMaxKeeper.sol

Report for Jobs/Bracket/brownie/contracts/ChanMaxKeeper.sol
https://dashboard.mythx.io/#/console/analyses/23ab30ff-dbbc-4954-bc62-6c998bdfb77f

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

ChanKeeper.sol

Report for Jobs/Bracket/brownie/contracts/ChanKeeper.sol
https://dashboard.mythx.io/#/console/analyses/f8f76c93-1114-4c4b-9ae2-76ae958ddb25

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

PriceProd.sol

Report for Jobs/Bracket/brownie/contracts/PriceProd.sol
https://dashboard.mythx.io/#/console/analyses/f8b83141-6918-4538-8832-d94f8670f1bb

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
11	(SWC-123) Requirement Violation	Low	Requirement violation.
70	(SWC-123) Requirement Violation	Low	Requirement violation.
81	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.

- No major issues were found by MythX.
- No issues were found by MythX on the remaining smart contracts.

THANK YOU FOR CHOOSING

// HALBORN