BLOCKHAT
SECURITY

MADNFT 1.0

Smart Contract Security Audit

Prepared by BlockHat

January 25th, 2023 - February 1st, 2023

BlockHat.io

contact@blockhat.io

Document Properties

Client	Jacob Clay
Version	1.1
Classification	Public

Scope

The MADNFT 1.0 Contract in the MADNFT 1.0 Repository

Repo	Owner
https://github.com/madnfts/madnfts-solidity-contracts/tree/release/1.0	MADNFTs

Files	MD5 Hash
contracts/EventsAndErrors.sol	0fa9eed8d9e93cb5a2fbf90f003fbc33
contracts/MAD.sol	d74a6be382cd2f23a3ebadb8d79c3f08
contracts/MADFactory1155.sol	845dc037d6631ea23e92200c7ec76ad9
contracts/MADFactory721.sol	d16b2983efab3c4be945c6df9448389f
contracts/MADMarketplace1155.sol	5c3110d6a7d2fc33e9730b84e9d0fd6c
contracts/MADMarketplace721.sol	6c06e551f913204897484066abdb8151
contracts/MADRouter1155.sol	26944124b03433aa42e30a5dcb947725
contracts/MADRouter721.sol	8a5a2e5ab6a334379ecdd5d997bf03ac

contracts/Types.sol	211938fc82b786b14a391ee5f2a8ee5f
lib/utils/Counters.sol	d216f3aabd6e9fcf0c041a8edaa895df
lib/utils/CREATE3.sol	3a09bbdf98d7284b6585ea335ada4b8b
lib/utils/MerkleProof.sol	c964b36fee0132365c17d4a53aeca414
lib/utils/SafeTransferLib.sol	0b7b6b404e478867d79ed10c7d8d847b
lib/utils/Strings.sol	5ab70c6b68313b6d7b196dbca9f17c34
lib/tokens/ERC20.sol	6d05d741c4ffb579e455a4736e9e9041
lib/tokens/ERC721/Impl/ERC721Basic.sol	5618e75d6121d5458bf04a6147fba447
lib/tokens/ERC721/Impl/ERC721Lazy.sol	1bae6600438ace20cadc430f63038f1b
lib/tokens/ERC721/Impl/ERC721Minimal.sol	0fe039b83f1c2919b9139cd35efea205
lib/tokens/ERC721/Impl/ERC721Whitelist.sol	d67d93261e5426006494a64e9e0afc8b
lib/tokens/ERC721/Base/ERC721.sol	bf2f901d011ad5c0990e4831fc5148cc
lib/tokens/ERC721/Base/utils/ERC721Holder.sol	acc5d77cdf104884dd4d1158bf289d48
lib/tokens/ERC721/Base/interfaces/ERC721EventAndErrors.sol	5240053a3f2a57541ad8ef6a28560f98
lib/tokens/ERC721/Base/interfaces/IERC721.sol	7b54e6881e257c4c934dbb6dcc425b02
lib/tokens/ERC1155/Impl/ERC1155Basic.sol	348853cd9c321955ae879e939337b29a
lib/tokens/ERC1155/Impl/ERC1155Lazy.sol	4ab9ce5078e083cad982a72b9e5d9e66
lib/tokens/ERC1155/Impl/ERC1155Minimal.sol	ee19697195cf28b30e8acfda33750522
lib/tokens/ERC1155/Impl/ERC1155Whitelist.sol	bb5e42d2cc6180be1a14cba1ccfe6cdf
lib/tokens/ERC1155/Base/ERC1155B.sol	b09c1d1b5cc6e121f1acdb2ba40f73e3

lib/tokens/ERC1155/Base/utils/ERC1155Holder.sol	81c56017acbde380827c3f0ac97463df
lib/tokens/ERC1155/Base/interfaces/ERC1155EventAndErrors.sol	4b21c51b30b96d253b63ce4d90e1511a
lib/tokens/ERC1155/Base/interfaces/IERC1155.sol	bf24bd68d21bbe70e3874f9673e32870
lib/tokens/common/ERC2981.sol	ccd94fe2933c3a11a3fe5e661e10977d
lib/tokens/common/FeeOracle.sol	19e1b61b398275c51453133e7981bcfe
lib/test/erc1155-mock.sol	02fdd44bbe56a2fdfd659987ebb77048
lib/test/erc20-mock.sol	105a433834cb9fc1759df5b12ab1637b
lib/test/erc2981-mock.sol	92595f27f4b695874873df0dffdd68d2
lib/test/erc721-mock.sol	4b3024b5a73ab7f6aff779b6f738e9d5
lib/test/test-interfaces.sol	59a8c882c8fa1edbc28c99a04b7521f8
lib/splitter/SplitterEventsAndErrors.sol	367908e4e3cf487917bec89406e69fa1
lib/splitter/SplitterImpl.sol	c72f9d3da9ea65a97ac7a5767dd2981c
lib/security/DCPrevent.sol	3a2308d6e5759a1f81061109e1942ac8
lib/security/Pausable.sol	1286b0b6207ae026ec9922d7345c06f6
lib/security/ReentrancyGuard.sol	03355df147e2ef07cfb8c09d346a9cd2
lib/deployers/ERC1155Deployer.sol	0ea61a892fabec2a881a5034277856c9
lib/deployers/ERC721Deployer.sol	1f03487f68aea5b5c06ee10032e47cd3
lib/deployers/SplitterDeployer.sol	0c407b49fed828cc74553bccb81633c1
lib/auth/FactoryVerifier.sol	c7f3d59a47c84642f5ed386d1088944e

lib/auth/Owned.sol	a880f344c057b2682d5ec6f03db96abf

Contacts

COMPANY	CONTACT
BlockHat	contact@blockhat.io

Contents

1 Introduction

MADNFT 1.0 engaged BlockHat to conduct a security assessment on the MADNFT 1.0 beginning on January 25th, 2023 and ending February 1st, 2023. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About MADNFT 1.0

MADNFT is a nft marketplace that allows the minting of 721 and 1155 NFTs on the harmony blockchain. There is a configurable mint fee of 0.25ONE and configurable platform fee set at 10 % . User can trade other external harmony NFTs on the marketplace too.

Issuer	Jacob Clay
Website	`https://madnfts.io/`
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

BlockHat used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

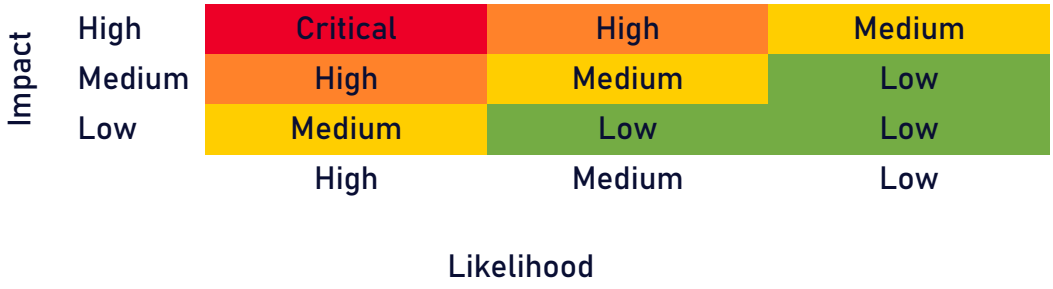
1.2.1 Risk Methodology

Vulnerabilities or bugs identified by BlockHat are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.



2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the MADNFT 1.0 implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include 4 critical-severity, 5 medium-severity, 5 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
Incorrect fees check	CRITICAL	Fixed
Incorrect fees check	CRITICAL	Fixed
Incorrect NFT Payments	CRITICAL	Fixed
Incorrect NFT Payments	CRITICAL	Fixed
Missing fees verification	MEDIUM	Fixed
Race Condition in Fees Value	MEDIUM	Acknowledged
Race Condition in Fees Value	MEDIUM	Acknowledged
address `_router` verification	MEDIUM	Acknowledged
address `_router` verification	MEDIUM	Acknowledged
Missing `_price` value verification	LOW	Fixed
Missing `_price` value verification	LOW	Fixed
Missing address verification	LOW	Acknowledged
Missing Value verification	LOW	Acknowledged
Approve race condition	LOW	Acknowledged

3 Finding Details

A MADRouter721.sol

A.1 Incorrect fees check [CRITICAL]

Description:

The fee price is initialized at 0.25 ether, which translates to a fee of 25 x 10 ** 16. If the owner sets Ethereum as ERC20 token payment method. The creator will pay 0.25 ethereum tokens: Which is a big amount. Also the fee setter only checks the limit in case of Harmony coin not other ERC20 tokens

Code:

Listing 1: MADRouter721

```
44      uint256 public feeMint = 0.25 ether;

46      /// @notice Burn fee store.
47      uint256 public feeBurn = 0;
```

Listing 2: MADRouter721

```
609     function setFees(uint256 _feeMint, uint256 _feeBurn)
610         external
611         onlyOwner
612     {
613         assembly {
614             sstore(feeBurn.slot, _feeBurn)
615             sstore(feeMint.slot, _feeMint)
616         }

618         emit FeesUpdated(_feeMint, _feeBurn);
619     }
```

Risk Level:

Likelihood – 4
Impact - 5

Recommendation:

We recommend calculating the fee value by converting Harmony to that ERC20 token. We advise to use Chainklink oracles also put a setter in Routercontract to upgrade the chainlink contract in case one day you need to add other tokens And add another limit in fee setters in case of erc20 != 0

Status - Fixed

The dev Team fixed the issue

A.2 Missing fees verification [MEDIUM]

Description:

Certain functions lack a safety check in the values, the values of the arguments should be verified to allow only the ones that go with the contract's logic. FeeMint and FeeBurn should be verified

Code:

Listing 3: MADRouter721

```
609     function setFees(uint256 _feeMint, uint256 _feeBurn)
610         external
611         onlyOwner
612     {
613         assembly {
614             sstore(feeBurn.slot, _feeBurn)
615             sstore(feeMint.slot, _feeMint)
616         }
```

```
618         emit FeesUpdated(_feeMint, _feeBurn);
619     }
```

Risk Level:

Likelihood – 3
Impact - 2

Recommendation:

We recommend to add require to set max value for the fees

Status - Fixed

The Dev team fixed the issue.

B MADRouter1155.sol

B.1 Incorrect fees check [CRITICAL]

Description:

The fee price is initialized at 0.25 ether, which translates to a fee of 25 x 10 ** 16. If the owner sets Ethereum as ERC20 token payment method. The creator will pay 0.25 ethereum tokens: Which is a big amount. Also the fee setter only checks the limit in case of Harmony coin not other ERC20 token

Code:

Listing 4: MADRouter1155

```
651     uint256 public feeMint = 0.25 ether;

653     /// @notice Burn fee store.
654     uint256 public feeBurn = 0;
```

Listing 5: MADRouter1155

```
651     function setFees(uint256 _feeMint, uint256 _feeBurn)
652         external
653         onlyOwner
654     {
655         require(
656             _feeMint < 50 ether && _feeBurn < 50 ether,
657             "Invalid Fees"
658         );
659         assembly {
660             sstore(feeBurn.slot, _feeBurn)
661             sstore(feeMint.slot, _feeMint)
662         }

664         emit FeesUpdated(_feeMint, _feeBurn);
665     }
```

Risk Level:

Likelihood – 4
Impact - 5

Recommendation:

We recommend calculating the fee value by converting Harmony to that ERC20 token. We advise to use Chainklink oracles also put a setter in Routercontract to upgrade the chainlink contract in case one day you need to add other tokens And add another limit in fee setters in case of erc20 != 0

Status - Fixed

The dev team fixed the issue

C MADMarketplace721.sol

C.1 Incorrect NFT Payments [CRITICAL]

Description:

A user may unintentionally purchase an NFT using incorrect payment method , In the buy function, attempting to purchase an NFT using Harmony may result an transaction with wrong method due to the ERC20 payment method being added by the owner while purchasing , The same issue occurs in the bid function.

Code:

Listing 6: MADMarketplace721

```
225     function buy(bytes32 _order)
226         external
227         payable
228         whenNotPaused
229      {
230         Types.Order721 storage order = orderInfo[_order];

232         _buyChecks(
233             order.endTime,
234             order.orderType,
235             order.isSold
236         );

238         uint256 currentPrice = getCurrentPrice(_order);
239         if (address(erc20) != address(0)) {
240             if (
241                 erc20.allowance(msg.sender, address(this)) <
242                 currentPrice
243             ) revert WrongPrice();
244             SafeTransferLib.safeTransferFrom(
245                 erc20,
```

```
246                 msg.sender,
247                 address(this),
248                 currentPrice
249             );
250         } else {
251             if (msg.value != currentPrice)
252                 revert WrongPrice();
253         }

255         order.isSold = true;

257         uint256 key = uint256(
258             uint160(address(order.token))
259         ) << 12;

261         // path for inhouse minted tokens
262         if (
263             !feeSelector[key][order.tokenId] &&
264             MADFactory721.creatorAuth(
265                 address(order.token),
266                 order.seller
267             ) ==
268             true
269         ) {
270             _intPath(
271                 order,
272                 currentPrice,
273                 _order,
274                 msg.sender,
275                 key
276             );
277         }
278         // path for external tokens
279         else {
```

```
280             // case for external tokens with ERC2981 support
281             if (
282                 ERC165Check(address(order.token)) &&
283                 interfaceCheck(
284                     address(order.token),
285                     0x2a55205a
286                 ) ==
287                 true
288             ) {
289                 _extPath0(
290                     order,
291                     currentPrice,
292                     _order,
293                     msg.sender // ,
294                     //key
295                 );
296             }
297             // case for external tokens without ERC2981 support
298             else {
299                 _extPath1(
300                     order,
301                     currentPrice,
302                     _order,
303                     msg.sender // ,
304                     // key
305                 );
306             }
307         }
308     }
```

Code:

Listing 7: MADMarketplace721

```
148     function bid(bytes32 _order)
```

```
149         external
150         payable
151         whenNotPaused
152     {
153         Types.Order721 storage order = orderInfo[_order];
154
155         uint256 lastBidPrice = order.lastBidPrice;
156         uint256 bidValue = address(erc20) != address(0)
157             ? erc20.allowance(msg.sender, address(this))
158             : msg.value;
159
160         _bidChecks(
161             order.orderType,
162             order.endTime,
163             order.seller,
164             lastBidPrice,
165             order.startPrice,
166             bidValue
167         );
168
169         if (address(erc20) != address(0)) {
170             SafeTransferLib.safeTransferFrom(
171                 erc20,
172                 msg.sender,
173                 address(this),
174                 bidValue
175             );
176         }
177
178         // 1s blocktime
179         assembly {
180             let endTime := and(
181                 sload(add(order.slot, 4)),
182                 shr(32, not(0))
```

```
183             )
184             if gt(
185                 timestamp(),
186                 sub(endTime, sload(minAuctionIncrement.slot))
187             ) {
188                 let inc := add(
189                     endTime,
190                     sload(minAuctionIncrement.slot)
191                 )
192                 sstore(add(order.slot, 4), inc)
193             }
194             sstore(add(order.slot, 6), caller())
195             sstore(add(order.slot, 5), bidValue)
196         }

198         if (lastBidPrice != 0) {
199             if (address(erc20) != address(0)) {
200                 SafeTransferLib.safeTransfer(
201                     erc20,
202                     order.lastBidder,
203                     lastBidPrice
204                 );
205             } else {
206                 SafeTransferLib.safeTransferETH(
207                     order.lastBidder,
208                     lastBidPrice
209                 );
210             }
211         }

213         emit Bid(
214             order.token,
215             order.tokenId,
216             _order,
```

```
217             msg.sender,
218             bidValue
219         );
220     }
```

Risk Level:

Likelihood – 4
Impact - 5

Recommendation:

We suggest either allowing users to choose their preferred payment method, or have a fixed one , or temporarily suspending the contract until all auctions are ended

Status - Fixed

The Dev team fixed the issue by setting the setPaymentToken function to private.

C.2 Race Condition in Fees Value [MEDIUM]

Description:

The feeVal2 , feeVal3 variables have a setter. If the user checks the value of this variable, then calls the buy or call function, and the owner updates the Fees Value , the order of the transaction might overturn and the user's transaction in this case will be executed with the new fees without him knowing about it.

Code:

Listing 8: MADMarketplace721

```
422     function setFees(uint256 _feeVal2, uint256 _feeVal3)
423         external
424         onlyOwner
425     {
```

```
426         // max fees, 15% for royalties, 5% for fees
427         require(
428             _feeVal2 <= 1.5e3 && _feeVal3 <= 5.0e2,
429             "Invalid Fees"
430         );
431         assembly {
432             sstore(feeVal2.slot, _feeVal2)
433             sstore(feeVal3.slot, _feeVal3)
434         }

436         emit FeesUpdated(_feeVal2, _feeVal3);
437     }
```

Risk Level:

Likelihood – 2
Impact - 3

Recommendation:

Consider adding the feeVal2 , feeVal3 in in the arguments of the _feeResolver function then add require statements that verifies that the values provided in the arguments are the same as the one that is stored in the smart contract .In the other hand, add FeeVal3 in the arguments of the _extPath0 , _extPath1 functions then a require statement that verifies that feeVal3 is the same as the one that is stored in the contract

Status - Acknowledged

The dev team acknowledged the issue

D MADMarketplace1155.sol

D.1 Incorrect NFT Payments [CRITICAL]

Description:

A user may unintentionally purchase an NFT using incorrect payment method , In the buy function, attempting to purchase an NFT using Harmony may result an transaction with wrong method due to the ERC20 payment method being added by the owner while purchasing , The same issue occurs in the bid function.

Code:

Listing 9: MADMarketplace1155

```
245     function buy(bytes32 _order)
246         external
247         payable
248         whenNotPaused
249      {
250         Types.Order721 storage order = orderInfo[_order];

252         _buyChecks(
253             order.endTime,
254             order.orderType,
255             order.isSold
256         );

258         uint256 currentPrice = getCurrentPrice(_order);
259         if (address(erc20) != address(0)) {
260             if (
261                 erc20.allowance(msg.sender, address(this)) <
262                 currentPrice
263             ) revert WrongPrice();
264             SafeTransferLib.safeTransferFrom(
265                 erc20,
```

```
266                 msg.sender,
267                 address(this),
268                 currentPrice
269             );
270         } else {
271             if (msg.value != currentPrice)
272                 revert WrongPrice();
273         }
274
275         order.isSold = true;
276
277         uint256 key = uint256(
278             uint160(address(order.token))
279         ) << 12;
280
281         // path for inhouse minted tokens
282         if (
283             !feeSelector[key][order.tokenId] &&
284             MADFactory721.creatorAuth(
285                 address(order.token),
286                 order.seller
287             ) ==
288             true
289         ) {
290             _intPath(
291                 order,
292                 currentPrice,
293                 _order,
294                 msg.sender,
295                 key
296             );
297         }
298         // path for external tokens
299         else {
```

```
300             // case for external tokens with ERC2981 support
301             if (
302                 ERC165Check(address(order.token)) &&
303                 interfaceCheck(
304                     address(order.token),
305                     0x2a55205a
306                 ) ==
307                 true
308             ) {
309                 _extPath0(
310                     order,
311                     currentPrice,
312                     _order,
313                     msg.sender // ,
314                     //key
315                 );
316             }
317             // case for external tokens without ERC2981 support
318             else {
319                 _extPath1(
320                     order,
321                     currentPrice,
322                     _order,
323                     msg.sender // ,
324                     // key
325                 );
326             }
327         }
328     }
```

Code:

Listing 10: MADMarketplace1155

```
168     function bid(bytes32 _order)
```

```
169         external
170         payable
171         whenNotPaused
172     {
173         Types.Order721 storage order = orderInfo[_order];
174 
175         uint256 lastBidPrice = order.lastBidPrice;
176         uint256 bidValue = address(erc20) != address(0)
177             ? erc20.allowance(msg.sender, address(this))
178             : msg.value;
179 
180         _bidChecks(
181             order.orderType,
182             order.endTime,
183             order.seller,
184             lastBidPrice,
185             order.startPrice,
186             bidValue
187         );
188 
189         if (address(erc20) != address(0)) {
190             SafeTransferLib.safeTransferFrom(
191                 erc20,
192                 msg.sender,
193                 address(this),
194                 bidValue
195             );
196         }
197 
198         // 1s blocktime
199         assembly {
200             let endTime := and(
201                 sload(add(order.slot, 4)),
202                 shr(32, not(0))
```

```
203             )
204             if gt(
205                 timestamp(),
206                 sub(endTime, sload(minAuctionIncrement.slot))
207             ) {
208                 let inc := add(
209                     endTime,
210                     sload(minAuctionIncrement.slot)
211                 )
212                 sstore(add(order.slot, 4), inc)
213             }
214             sstore(add(order.slot, 6), caller())
215             sstore(add(order.slot, 5), bidValue)
216         }

218         if (lastBidPrice != 0) {
219             if (address(erc20) != address(0)) {
220                 SafeTransferLib.safeTransfer(
221                     erc20,
222                     order.lastBidder,
223                     lastBidPrice
224                 );
225             } else {
226                 SafeTransferLib.safeTransferETH(
227                     order.lastBidder,
228                     lastBidPrice
229                 );
230             }
231         }

233         emit Bid(
234             order.token,
235             order.tokenId,
236             _order,
```

```
237             msg.sender,
238             bidValue
239         );
240     }
```

Risk Level:

Likelihood – 4
Impact - 5

Recommendation:

We suggest either allowing users to choose their preferred payment method, or have a fixed one , or temporarily suspending the contract until all auctions are ended

Status - Fixed

The Dev team fixed the issue by setting the function setPrivateToken to private.

D.2 Race Condition in Fees Value [MEDIUM]

Description:

The feeVal2 , feeVal3 variables have a setter. If the user checks the value of this variable, then calls the buy or call function, and the owner updates the Fees Value , the order of the transaction might overturn and the user's transaction in this case will be executed with the new fees without him knowing about it.

Code:

Listing 11: MADMarketplace1155

```
445     function setFees(uint256 _feeVal2, uint256 _feeVal3)
446         external
447         onlyOwner
448     {
```

```
449         require(
450             _feeVal2 <= 1.5e3 && _feeVal3 <= 5.0e2,
451             "Invalid Fees"
452         );

454         assembly {
455             sstore(feeVal2.slot, _feeVal2)
456             sstore(feeVal3.slot, _feeVal3)
457         }

459         emit FeesUpdated(_feeVal2, _feeVal3);
460     }
```

Risk Level:

Likelihood – 2
Impact - 3

Recommendation:

Consider adding the feeVal2 , feeVal3 in in the arguments of the _feeResolver function then add require statements that verifies that the values provided in the arguments are the same as the one that is stored in the smart contract .In the other hand, add FeeVal3 in the arguments of the _extPath0 , _extPath1 functions then a require statement that verifies that feeVal3 is the same as the one that is stored in the contract

Status - Acknowledged

The dev team Acknowledged the issue

E MADFactory721.sol

E.1 address _router verification [MEDIUM]

Description:

The constructor lack a safety check in _router address , the address-type argument should include a zero-address test, otherwise, some of the contract's functionality may become inaccessible.

Code:

Listing 12: MADFactory721

```
91      constructor
92      (
93          address _marketplace,
94          address _router,
95          address _signer,
96          address _paymentTokenAddress
97      )
98      {
99          setMarket(_marketplace);
100         setSigner(_signer);
101         if (_paymentTokenAddress != address(0)) {
102             setPaymentToken(_paymentTokenAddress);
103         }
104         router = _router;
105         emit RouterUpdated(_router);
106     }
```

Risk Level:

Likelihood – 3

Impact - 2

Recommendation:

We recommend to use SetRouter functon instead of affecting the _router address to router in the constructor

Status - Acknowledged

E.2 Missing _price value verification [LOW]

Description:

Certain functions lack a safety check in the values, the values of the arguments should be verified to allow only the ones that go with the contract's logic. The_price variable in create-Collection function should be different than zero.

Code:

Listing 13: MADFactory721

```
334     function createCollection(
335         uint8 _tokenType,
336         string memory _tokenSalt,
337         string memory _name,
338         string memory _symbol,
339         uint256 _price,
340         uint256 _maxSupply,
341         string memory _baseURI,
342         address _splitter,
343         uint256 _royalty
344     )
345         external
346         nonReentrant
347         isThisOg
348         whenNotPaused
349     {
350         _limiter(_tokenType, _splitter);
```

```
351         _royaltyLocker(_royalty);
352
353         if (_tokenType < 1) {
354             (bytes32 tokenSalt, address deployed) =
355                 ERC721MinimalDeployer._721MinimalDeploy(
356                     _tokenSalt,
357                     _name,
358                     _symbol,
359                     _baseURI,
360                     _price,
361                     _splitter,
362                     router,
363                     _royalty,
364                     erc20
365                 );
366
367             bytes32 colId = deployed.fillLast12Bytes();
368             userTokens[tx.origin].push(colId);
369
370             colInfo[colId] = Types.Collection721(
371                 tx.origin,
372                 Types.ERC721Type.ERC721Minimal,
373                 tokenSalt,
374                 block.number,
375                 _splitter
376             );
377
378             emit ERC721MinimalCreated(
379                 _splitter,
380                 deployed,
381                 _name,
382                 _symbol,
383                 _royalty,
384                 _maxSupply,
```

```
385                 _price
386             );
387         }
388         if (_tokenType == 1) {
389             (bytes32 tokenSalt, address deployed) =
390             ERC721BasicDeployer._721BasicDeploy(
391                 _tokenSalt,
392                 _name,
393                 _symbol,
394                 _baseURI,
395                 _price,
396                 _maxSupply,
397                 _splitter,
398                 router,
399                 _royalty,
400                 erc20
401             );
402
403             bytes32 colId = deployed.fillLast12Bytes();
404             userTokens[tx.origin].push(colId);
405
406             colInfo[colId] = Types.Collection721(
407                 tx.origin,
408                 Types.ERC721Type.ERC721Basic,
409                 tokenSalt,
410                 block.number,
411                 _splitter
412             );
413
414             emit ERC721BasicCreated(
415                 _splitter,
416                 deployed,
417                 _name,
418                 _symbol,
```

```
                _royalty,
                _maxSupply,
                _price
            );
        }
        if (_tokenType == 2) {
            (bytes32 tokenSalt, address deployed) =
            ERC721WhitelistDeployer._721WhitelistDeploy(
                _tokenSalt,
                _name,
                _symbol,
                _baseURI,
                _price,
                _maxSupply,
                _splitter,
                router,
                _royalty,
                erc20
            );

            bytes32 colId = deployed.fillLast12Bytes();
            userTokens[tx.origin].push(colId);

            colInfo[colId] = Types.Collection721(
                tx.origin,
                Types.ERC721Type.ERC721Whitelist,
                tokenSalt,
                block.number,
                _splitter
            );

            emit ERC721WhitelistCreated(
                _splitter,
                deployed,
```

```
453                 _name,
454                 _symbol,
455                 _royalty,
456                 _maxSupply,
457                 _price
458             );
459         }
460         if (_tokenType > 2) {
461             (bytes32 tokenSalt, address deployed) =
462             ERC721LazyDeployer._721LazyDeploy(
463                 _tokenSalt,
464                 _name,
465                 _symbol,
466                 _baseURI,
467                 _splitter,
468                 router,
469                 signer,
470                 _royalty,
471                 erc20
472             );
473
474             bytes32 colId = deployed.fillLast12Bytes();
475             userTokens[tx.origin].push(colId);
476
477             colInfo[colId] = Types.Collection721(
478                 tx.origin,
479                 Types.ERC721Type.ERC721Lazy,
480                 tokenSalt,
481                 block.number,
482                 _splitter
483             );
484
485             emit ERC721LazyCreated(
486                 _splitter,
```

```
487                 deployed,
488                 _name,
489                 _symbol,
490                 _royalty,
491                 _maxSupply,
492                 _price
493             );
494         }
495     }
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Add a 'require' statement to verify that '_price' is not equal to zero.

Status - Fixed

F MADFactory1155.sol

F.1 address `_router` verification [MEDIUM]

Description:

The constructor lack a safety check in _router address , the address-type argument should include a zero-address test, otherwise, some of the contract's functionality may become inaccessible.

Code:

Listing 14: MADFactory1155

```
91  constructor
```

```
92      (
93          address _marketplace,
94          address _router,
95          address _signer,
96          address _paymentTokenAddress
97      )
98      {
99          setMarket(_marketplace);
100         setSigner(_signer);
101         if (_paymentTokenAddress != address(0)) {
102             setPaymentToken(_paymentTokenAddress);
103         }

105         router = _router;
106         emit RouterUpdated(_router);
107     }
```

Risk Level:

Likelihood – 3
Impact - 2

Recommendation:

Use of the SetRouter function is recommended instead of assigning _router address to router in the constructor

Status - Acknowledged

F.2 Missing _price value verification [LOW]

Description:

Certain functions lack a safety check in the values, the values of the arguments should be verified to allow only the ones that go with the contract's logic. The _price variable in cre-

ateCollection function should be different than zero

Code:

Listing 15: MADFactory1155

```
331     function createCollection(
332         uint8 _tokenType,
333         string memory _tokenSalt,
334         string memory _name,
335         string memory _symbol,
336         uint256 _price,
337         uint256 _maxSupply,
338         string memory _uri,
339         address _splitter,
340         uint256 _royalty
341     )
342         external
343         nonReentrant
344         isThisOg
345         whenNotPaused
346     {
347         _limiter(_tokenType, _splitter);
348         _royaltyLocker(_royalty);

350         if (_tokenType < 1) {
351         (bytes32 tokenSalt, address deployed) =
352             ERC1155MinimalDeployer._1155MinimalDeploy(
353                 _tokenSalt,
354                 _uri,
355                 _price,
356                 _splitter,
357                 router,
358                 _royalty,
359                 erc20
```

```
360             );

362         bytes32 colId = deployed.fillLast12Bytes();
363         userTokens[tx.origin].push(colId);

365         colInfo[colId] = Types.Collection1155(
366             tx.origin,
367             Types.ERC1155Type.ERC1155Minimal,
368             tokenSalt,
369             block.number,
370             _splitter
371         );

373         emit ERC1155MinimalCreated(
374             _splitter,
375             deployed,
376             _name,
377             _symbol,
378             _royalty,
379             _maxSupply,
380             _price
381         );
382         }
383         if (_tokenType == 1) {
384             (bytes32 tokenSalt, address deployed) =
385             ERC1155BasicDeployer._1155BasicDeploy(
386                 _tokenSalt,
387                 _uri,
388                 _price,
389                 _maxSupply,
390                 _splitter,
391                 router,
392                 _royalty,
393                 erc20
```

```
394             );
395
396         bytes32 colId = deployed.fillLast12Bytes();
397         userTokens[tx.origin].push(colId);
398
399         colInfo[colId] = Types.Collection1155(
400             tx.origin,
401             Types.ERC1155Type.ERC1155Basic,
402             tokenSalt,
403             block.number,
404             _splitter
405         );
406
407         emit ERC1155BasicCreated(
408             _splitter,
409             deployed,
410             _name,
411             _symbol,
412             _royalty,
413             _maxSupply,
414             _price
415         );
416         }
417         if (_tokenType == 2) {
418             (bytes32 tokenSalt, address deployed) =
419             ERC1155WhitelistDeployer._1155WhitelistDeploy(
420                 _tokenSalt,
421                 _uri,
422                 _price,
423                 _maxSupply,
424                 _splitter,
425                 router,
426                 _royalty,
427                 erc20
```

```
428             );
429
430         bytes32 colId = deployed.fillLast12Bytes();
431         userTokens[tx.origin].push(colId);
432
433         colInfo[colId] = Types.Collection1155(
434             tx.origin,
435             Types.ERC1155Type.ERC1155Whitelist,
436             tokenSalt,
437             block.number,
438             _splitter
439         );
440
441         emit ERC1155WhitelistCreated(
442             _splitter,
443             deployed,
444             _name,
445             _symbol,
446             _royalty,
447             _maxSupply,
448             _price
449         );
450         }
451         if (_tokenType > 2) {
452             (bytes32 tokenSalt, address deployed) =
453                 ERC1155LazyDeployer._1155LazyDeploy(
454                     _tokenSalt,
455                     _uri,
456                     _splitter,
457                     router,
458                     signer,
459                     _royalty,
460                     erc20
461                 );
```

```
463         bytes32 colId = deployed.fillLast12Bytes();
464         userTokens[tx.origin].push(colId);

466         colInfo[colId] = Types.Collection1155(
467             tx.origin,
468             Types.ERC1155Type.ERC1155Lazy,
469             tokenSalt,
470             block.number,
471             _splitter
472         );

474         emit ERC1155LazyCreated(
475             _splitter,
476             deployed,
477             _name,
478             _symbol,
479             _royalty,
480             _maxSupply,
481             _price
482         );
483         }
484     }
```

Risk Level:

Likelihood – 2

Impact - 2

Recommendation:

Add a 'require' statement to verify that '_price' is not equal to zero.

Status - Fixed

G ERC20.sol

G.1 Missing address verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, some of the contract's functionality may become inaccessible.

Code:

Listing 16: ERC20.sol

```
77      function approve(address spender, uint256 amount)
78          public
79          virtual
80          returns (bool)
81      {
82          allowance[msg.sender][spender] = amount;

84          emit Approval(msg.sender, spender, amount);

86          return true;
87      }
```

Listing 17: ERC20.sol

```
89      function transfer(address to, uint256 amount)
90          public
91          virtual
92          returns (bool)
93      {
94          balanceOf[msg.sender] -= amount;
```

```
96          // Cannot overflow because the sum of all user
97          // balances can't exceed the max uint256 value.
98          unchecked {
99              balanceOf[to] += amount;
100         }
101
102         emit Transfer(msg.sender, to, amount);
103
104         return true;
105     }
```

Listing 18: ERC20.sol

```
107     function transferFrom(
108         address from,
109         address to,
110         uint256 amount
111     ) public virtual returns (bool) {
112         uint256 allowed = allowance[from][msg.sender]; // Saves gas for
                ↪ limited approvals.
113
114         if (allowed != type(uint256).max)
115             allowance[from][msg.sender] = allowed - amount;
116
117         balanceOf[from] -= amount;
118
119         // Cannot overflow because the sum of all user
120         // balances can't exceed the max uint256 value.
121         unchecked {
122             balanceOf[to] += amount;
123         }
124
125         emit Transfer(from, to, amount);
126
127         return true;
```

```
128     }
```

Listing 19: ERC20.sol

```
223     function _mint(address to, uint256 amount)
224         internal
225         virtual
226     {
227         totalSupply += amount;

229         // Cannot overflow because the sum of all user
230         // balances can't exceed the max uint256 value.
231         unchecked {
232             balanceOf[to] += amount;
233         }

235         emit Transfer(address(0), to, amount);
236     }
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

It is recommended to verify that the addresses "to", "spender" and "from" provided in the arguments are different from the address(0) .

Status - Acknowledged

The team Acknowledged the issue due the use of erc20 interface

G.2 Missing Value verification [LOW]

Description:

Certain functions lack a safety check in the values, the values of the arguments should be verified to allow only the ones that go with the contract's logic.

Code:

Listing 20: ERC20.sol

```
77      function approve(address spender, uint256 amount)
78          public
79          virtual
80          returns (bool)
81      {
82          allowance[msg.sender][spender] = amount;
83
84          emit Approval(msg.sender, spender, amount);
85
86          return true;
87      }
```

Listing 21: ERC20.sol

```
89      function transfer(address to, uint256 amount)
90          public
91          virtual
92          returns (bool)
93      {
94          balanceOf[msg.sender] -= amount;
95
96          // Cannot overflow because the sum of all user
97          // balances can't exceed the max uint256 value.
98          unchecked {
99              balanceOf[to] += amount;
100         }
```

```
102         emit Transfer(msg.sender, to, amount);

104         return true;
105     }
```

Listing 22: ERC20.sol

```
107     function transferFrom(
108         address from,
109         address to,
110         uint256 amount
111     ) public virtual returns (bool) {
112         uint256 allowed = allowance[from][msg.sender]; // Saves gas for
                ↪ limited approvals.

114         if (allowed != type(uint256).max)
115             allowance[from][msg.sender] = allowed - amount;

117         balanceOf[from] -= amount;

119         // Cannot overflow because the sum of all user
120         // balances can't exceed the max uint256 value.
121         unchecked {
122             balanceOf[to] += amount;
123         }

125         emit Transfer(from, to, amount);

127         return true;
128     }
```

Listing 23: ERC20.sol

```
223     function _mint(address to, uint256 amount)
224         internal
```

```
225         virtual
226     {
227         totalSupply += amount;
229         // Cannot overflow because the sum of all user
230         // balances can't exceed the max uint256 value.
231         unchecked {
232             balanceOf[to] += amount;
233         }
235         emit Transfer(address(0), to, amount);
236     }
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

Add a 'require' statement to verify that '_amount' is not equal to zero.

Status - Acknowledged

The team Acknowledged the issue due the use of erc20 interface

G.3 Approve race condition [LOW]

Description:

The standard ERC20 implementation contains a widely known racing condition in its approve function, wherein a spender can witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the

spender's transaction is validated before the owner's, the spender will be able to get both approval amounts of both transactions.

Code:

Listing 24: ERC20

```
77      function approve(address spender, uint256 amount)
78          public
79          virtual
80          returns (bool)
81      {
82          allowance[msg.sender][spender] = amount;

84          emit Approval(msg.sender, spender, amount);

86          return true;
87      }
```

Code:

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

We recommend using increaseAllowance and decreaseAllowance functions to modify the approval amount instead of using the approve function to modify it.

Status - Acknowledged

The team Acknowledged the issue due the use of erc20 interface

4 Best Practices

BP.1 Public functions can be external

Description:

Functions with a public scope that are not called inside the contract should be declared external to reduce the gas fees

Code:

Listing 25: MADMarketplace721.sol

```
134     function englishAuction(
135         IERC721 _token,
136         uint256 _id,
137         uint256 _startPrice,
138         uint256 _endTime
139     ) public whenNotPaused {
```

Listing 26: MADMarketplace721.sol

```
114     function dutchAuction(
115         IERC721 _token,
116         uint256 _id,
117         uint256 _startPrice,
118         uint256 _endPrice,
119         uint256 _endTime
120     ) public whenNotPaused {
```

Listing 27: MADMarketplace721.sol

```
103     function fixedPrice(
104         IERC721 _token,
105         uint256 _id,
106         uint256 _price,
107         uint256 _endTime
```

```
108     ) public whenNotPaused {
```

Listing 28: MADMarketplace1155.sol

```
145     function englishAuction(
146         IERC1155 _token,
147         uint256 _id,
148         uint256 _amount,
149         uint256 _startPrice,
150         uint256 _endTime
151     ) public whenNotPaused {
```

Listing 29: MADMarketplace1155.sol

```
123     function dutchAuction(
124         IERC1155 _token,
125         uint256 _id,
126         uint256 _amount,
127         uint256 _startPrice,
128         uint256 _endPrice,
129         uint256 _endTime
130     ) public whenNotPaused {
```

Listing 30: MADMarketplace1155.sol

```
123     function fixedPrice(
124         IERC1155 _token,
125         uint256 _id,
126         uint256 _amount,
127         uint256 _price,
128         uint256 _endTime
129     ) public whenNotPaused {
```

Listing 31: MAD.sol

```
28     function name()
29         public
30         pure
```

```
31          virtual
32          returns (string memory);
33  }
```

Listing 32: MADFactory721.sol

```
45      function name()
46          public
47          pure
48          override(MAD)
49          returns (string memory)
50      {
```

Listing 33: MADFactory1155.sol

```
46      function name()
47          public
48          pure
49          override(MAD)
50          returns (string memory)
51      {
```

Listing 34: MADMarketplace721.sol

```
24      function name()
25          public
26          pure
27          override(MAD)
28          returns (string memory)
29      {
```

Listing 35: MADMarketplace1155.sol

```
24      function name()
25          public
26          pure
27          override(MAD)
28          returns (string memory)
```

```
29      {
```

Listing 36: MADRouter721.sol

```
54      function name()
55          public
56          pure
57          override(MAD)
58          returns (string memory)
59      {
```

Listing 37: MADRouter1155.sol

```
54      function name()
55          public
56          pure
57          override(MAD)
58          returns (string memory)
59      {
```

Listing 38: MADFactory721.sol

```
530     function setRouter(address _router) public onlyOwner {
```

Listing 39: MADFactory1155.sol

```
518      function setRouter(address _router) public onlyOwner {
```

Listing 40: MADFactory721.sol

```
723     function creatorCheck(bytes32 _colID)
724     public
```

Listing 41: MADFactory1155.sol

```
710     function creatorCheck(bytes32 _colID)
711     public
```

Listing 42: MADFactory721.sol

```
820     function getDeployedAddr(string memory _salt)
```

Listing 43: MADFactory1155.sol

```
807     function getDeployedAddr(string memory _salt)
```

5 Tests

Results:

```
Compiled 48 Solidity files successfully
/// ... .. .. ..
/// x*8888x.:*8888: -"888: dF
/// X 48888X `8888H 8888 '88bu.
/// X8x. 8888X 8888X !888> u '*88888bu
/// X8888 X8888 88888 "*8%- us888u. ^"*8888N
/// '*888!X8888> X8888 xH8> .@88 "8888" beWE "888L
/// `?8 `8888 X888X X888> 9888 9888 888E 888E
/// -^ '888" X888 8888> 9888 9888 888E 888E
/// dx '88~x. !88~ 8888> 9888 9888 888E 888F
/// .8888Xf.888x:! X888X.: 9888 9888 .888N..888
/// :""888":~"888" `888*" "888*""888" `"888*""
/// "~' "~ "" ^Y" ^Y' "" MADNFTs © 2022.


  ERC1155Basic
    Init
      Splitter and ERC1155 should initialize (154ms)
      accounts have been funded
    Only owner setters
      Should set base URI, emit event and revert if not owner (127ms)
      Should set public mint state, emit event & revert if not owner
        ↪ (110ms)
    Mint
      Should revert if public mint is turned off (44ms)
      Should revert if max supply has reached max (6290ms)
      Should revert if price is wrong (45ms)
      Should mint, update storage and emit events (106ms)
      Should handle multiple mints (5492ms)
    Batch mint
```

```
      Should revert if supply has reached max (5812ms)
      Should revert if public mint is turned off
      Should revert if price is wrong (43ms)
      Should batch mint, update storage and emit events (221ms)
      Should handle multiple batch mints (254ms)
    Burn
      Should revert if not owner
      Should revert if id is already burnt/hasn't been minted (150ms)
      Should revert if ids length is less than 2 (65ms)
      Should burn tokens, update storage and emit event (221ms)
    Batch burn
      Should revert if caller is not the owner (80ms)
      Should revert if id is already burnt/hasn't been minted (189ms)
      Should batch burn tokens, update storage and emit event (226ms)
      Should handle multiple batch burns (358ms)
    Withdraw
      Should withdraw contract's funds (375ms)
      Should withdraw contract's ERC20s (230ms)
    Public getters
      Should query royalty info
      Should query token uri and revert if not yet minted (79ms)
      Should query total supply
      Should query base uri
    Interface IDs
      Should support interfaces (48ms)

  ERC1155Lazy
    Init
      Splitter and ERC1155 should initialize (69ms)
      accounts have been funded
    Lazy mint
      Should mint, update storage and emit events (470ms)
      Should revert if voucher has already been used (313ms)
      Should revert if signature is invalid
```

```
      Should revert if price is wrong (48ms)
    Lazy batch mint
      Should mint, update storage and emit events (166ms)
      Should revert if voucherId has already been used (80ms)
      Should revert if signature is invalid
      Should revert if price is wrong (38ms)
    Only owner functions
      Should set URI and emit event (54ms)
      Should withdraw and update balances (553ms)
    Burn
      Should revert if not owner (141ms)
      Should revert if id is already burnt/hasn't been minted (238ms)
      Should revert if ids length is less than 2 (48ms)
      Should burn update storage and emit events (366ms)
    Batch burn
      Should revert if caller is not the owner (241ms)
      Should revert if id is already burnt/hasn't been minted (220ms)
      Should batch burn tokens, update storage and emit event (306ms)
      Should handle multiple batch burns (452ms)
    Public getters
      Should query royalty info
      Should retrieve the domain separator
      Should retrive URI and total supply (295ms)
      Should retrive tokenURI and revert if not yet minted (229ms)
      Should support interfaces

  ERC1155Minimal
    Init
      Splitter and ERC1155 should initialize (60ms)
      accounts have been funded
    Safe Minting
      Should revert if not the owner
      Should mint, update storage and emit events (48ms)
      Should revert if already minted (70ms)
```

```
    Burning
      Should revert if has not been minted
      Should revert if not the owner (56ms)
      Should burn, update storage and emit events (104ms)
      Should revert if already burned (80ms)
    Public Minting
      Should update public mint state (65ms)
      Should revert if public mint is off
      Should revert if price is wrong (59ms)
      Should revert if already minted (80ms)
      Should mint, update storage and emit events (93ms)
    Withdrawing
      Should revert if not the owner (89ms)
      Should update balances of contract and owner (180ms)
      Should withdraw contract's ERC20s (287ms)
    Royalties
      Should retrive royalty info
    Token URI
      Should revert if ID is not 1
      Should revert if token was not minted
      Should retrieve tokenURI (49ms)
    Interface IDs
      Should support interfaces

  ERC1155Whitelist
    Init
      Splitter and ERC721 should initialize (157ms)
      accounts have been funded
    Only owner setters
      Should check for whitelist & freeclaim event emitting/error
        ↪ handling (85ms)
      Should set URI and emit event (52ms)
      Should set mint states (111ms)
    Public mint
```

```
      Should revert if value under/overflows
      Should revert if public mint state is off
      Should revert if available supply has reached max (6028ms)
      Should revert if price is wrong (67ms)
      Should mint, update storage and emit events (135ms)
    Batch mint
      Should revert if supply has reached max (6016ms)
      Should revert if public mint is turned off
      Should revert if price is wrong (53ms)
      Should batch mint, update storage and emit events (119ms)
      Should handle multiple batch mints (215ms)
    Whitelist mint
      Should revert if value under/overflows
      Should revert if whitelist mint state is off
      Should revert if whitelist supply has reached max (5999ms)
      Should revert if price is wrong (46ms)
      Should revert if address is not whitelisted (54ms)
      Should mint, update storage and emit events (140ms)
    Whitelist batch mint
      Should revert if value under/overflows
      Should revert if whitelist mint state is off
      Should revert if whitelist supply has reached max (5797ms)
      Should revert if price is wrong (51ms)
      Should revert if address is not whitelisted (51ms)
      Should mint, update storage and emit events (138ms)
    Free claim
      Should revert if free claim state is off
      Should revert if available supply has reached max (6589ms)
      Should revert if address is not whitelisted (41ms)
      Should revert if user has already claimed (71ms)
      Should mint, update storage and emit events (123ms)
      Should gift tokens (237ms)
    Mint and batch mint to creator
      Should mint to creator (192ms)
```

```
      Should batch mint to creator (177ms)
    Burn
      Should revert if not owner
      Should revert if id is already burnt/hasn't been minted (119ms)
      Should revert if ids length is less than 2
      Should burn tokens, update storage and emit event (221ms)
    Batch burn
      Should revert if caller is not the owner (80ms)
      Should revert if id is already burnt/hasn't been minted (120ms)
      Should batch burn tokens, update storage and emit event (207ms)
      Should handle multiple batch burns (365ms)
    Withdraw
      Should withdraw contract's funds (177ms)
      Should withdraw contract's ERC20s (197ms)
    Public getters
      Should query royalty info
      Should query token uri and revert if not yet minted (75ms)
      Should query total supply
      Should query base uri
    Interface IDs
      Should support interfaces

  ERC721Basic
    Init
      Splitter and ERC721 should initialize (91ms)
      accounts have been funded
    Only owner setters
      Should set base URI, emit event and revert if not owner (69ms)
      Should set public mint state, emit event & revert if not owner (51
        ↪ ms)
    Mint
      Should revert if public mint is turned off
      Should revert if max supply has reached max (6718ms)
      Should revert if price is wrong
```

```
      Should mint, update storage and emit events (89ms)
      Should handle multiple mints (6881ms)
    Burn
      Should revert if not owner
      Should revert if id is already burnt/hasn't been minted (119ms)
      Should revert if ids length is less than 2
      Should burn tokens, update storage and emit event (208ms)
    Withdraw
      Should withdraw contract's funds (146ms)
      Should withdraw contract's ERC20s (219ms)
    Public getters
      Should query royalty info
      Should query token uri and revert if not yet minted (80ms)
      Should query total supply
      Should query base uri
      Should support interfaces (52ms)

  ERC721Lazy
    Init
      Splitter and ERC721 should initialize (87ms)
      accounts have been funded
    Lazy mint
      Should mint, update storage and emit events (373ms)
      Should revert if voucher has already been used (221ms)
      Should revert if signature is invalid
      Should revert if price is wrong
    Only owner functions
      Should set baseURI and emit event (46ms)
      Should withdraw and update balances (641ms)
    Burn
      Should revert if not owner
      Should revert if id is already burnt/hasn't been minted (224ms)
      Should revert if ids length is less than 2
      Should burn update storage and emit events (317ms)
```

```
    Public getters
      Should retrieve the domain separator
      Should retrive baseURI and total supply (279ms)
      Should retrive tokenURI and revert if not yet minted (261ms)
      Should query royalty info
      Should support interfaces (82ms)

  ERC721Minimal
    Init
      Splitter and ERC721 should initialize (89ms)
      accounts have been funded
    Safe Minting
      Should revert if not the owner
      Should mint, update storage and emit events (64ms)
      Should revert if already minted (66ms)
    Burning
      Should revert if has not been minted
      Should revert if not the owner (51ms)
      Should burn, update storage and emit events (123ms)
      Should revert if already burned (74ms)
    Public Minting
      Should update public mint state (76ms)
      Should revert if public mint is off
      Should revert if price is wrong (49ms)
      Should revert if already minted (97ms)
      Should mint, update storage and emit events (81ms)
    Withdrawing
      Should revert if not the owner (151ms)
      Should update balances of contract and owner (125ms)
      Should withdraw contract's ERC20s (193ms)
    Royalties
      Should retrive royalty info
    Token URI
      Should revert if ID is not 1
```

```
      Should revert if token was not minted
      Should retrieve tokenURI (46ms)
    Interface IDs
      Should support interfaces

  ERC721Whitelist
    Init
      Splitter and ERC721 should initialize (210ms)
      accounts have been funded
    Only owner setters
      Should check for whitelist & freeclaim event emitting/error
        ↪ handling (101ms)
      Should set baseURI and emit event (44ms)
      Should set mint states (104ms)
    Public mint
      Should revert if value under/overflows
      Should revert if public mint state is off
      Should revert if available supply has reached max (6631ms)
      Should revert if price is wrong (38ms)
      Should mint, update storage and emit events (175ms)
    Whitelist mint
      Should revert if value under/overflows
      Should revert if whitelist mint state is off
      Should revert if whitelist supply has reached max (6995ms)
      Should revert if price is wrong (41ms)
      Should revert if address is not whitelisted
      Should mint, update storage and emit events (140ms)
    Free claim
      Should revert if free claim state is off
      Should revert if available supply has reached max (6939ms)
      Should revert if address is not whitelisted
      Should revert if user has already claimed (59ms)
      Should mint, update storage and emit events (201ms)
      Should mint to creator (208ms)
```

```
      Should gift tokens (249ms)
    Burn
      Should revert if not owner
      Should revert if id is already burnt/hasn't been minted (101ms)
      Should revert if ids length is less than 2
      Should burn update storage and emit events (213ms)
    Public getters
      Should retrive baseURI and total supply (153ms)
      Should retrive tokenURI and revert if not yet minted (56ms)
      Should support interfaces
    Withdrawing
      Should revert if not the owner (173ms)
      Should update balances of contract and owner (132ms)
      Should withdraw contract's ERC20s (204ms)

  MADFactory1155
    Init
      Factory should initialize (39ms)
    Splitter check
      Should revert if repeated salt is provided (343ms)
      Should deploy splitter without ambassador, update storage and emit
        ↪  events (199ms)
      Should deploy splitter with ambassador, update storage and emit
        ↪ events (208ms)
    Create collection
      Should deploy ERC1155Minimal, update storage and emit events (562
        ↪ ms)
      Should deploy ERC1155Basic, update storage and emit events (526ms)
        ↪
      Should deploy ERC1155Whitelist, update storage and emit events
        ↪ (636ms)
      Should deploy ERC1155Lazy, update storage and emit events (714ms)
    Only owner functions
      Should update contract's owner (51ms)
```

```
      Should set new marketplace instance (69ms)
      Should update ERC1155Lazy signer (41ms)
      Should update router's address (46ms)
      Should initialize paused and unpaused states (110ms)
    Helpers
      Should retrieve user's colID indexes (904ms)
      Should get collection ID from address
      Should retrieve collection type (472ms)
      Should enable marketplace no-fee listing (656ms)
      Should verify a collection's creator (522ms)

  MADFactory721
    Init
      Factory should initialize (54ms)
    Splitter check
      Should revert if repeated salt is provided (187ms)
      Should deploy splitter without ambassador, update storage and emit
        ↪  events (181ms)
      Should deploy splitter with ambassador, update storage and emit
        ↪ events (211ms)
      Should deploy splitter with ambassador and project, update storage
        ↪  and emit events (209ms)
    Create collection
      Should deploy ERC721Minimal, update storage and emit events (431ms
        ↪ )
      Should deploy ERC721Basic, update storage and emit events (448ms)
      Should deploy ERC721Whitelist, update storage and emit events (497
        ↪ ms)
      Should deploy ERC721Lazy, update storage and emit events (458ms)
    Only owner functions
      Should update contract's owner (56ms)
      Should set new marketplace instance (81ms)
      Should update ERC721Lazy signer (48ms)
      Should update router's address (49ms)
```

```
      Should initialize paused and unpaused states (113ms)
    Helpers
      Should retrieve user's colID indexes (794ms)
      Should get collection ID from address
      Should retrieve collection type (549ms)
      Should enable marketplace no-fee listing (686ms)
      Should verify a collection's creator (390ms)

  MADMarketplace1155
    Init
      Marketplace should initialize (48ms)
    Owner Functions
      Should update factory address (61ms)
      Should update marketplace settings (50ms)
      Should initialize paused and unpaused states (225ms)
      Should update recipient (50ms)
      Should update contract's owner (56ms)
      Should withdraw to owner (133ms)
      Should delete order (960ms)
    Fixed Price Listing
      Should revert if transaction approval hasn't been set (513ms)
      Should revert if duration is less than min allowed (481ms)
      Should revert if price is invalid (453ms)
      Should list fixed price order, update storage and emit event (623
        ↪ ms)
      Should handle multiple fixed price orders (1201ms)
    Dutch Auction Listing
      Should revert if transaction approval hasn't been set (488ms)
      Should revert if duration is less than min allowed (446ms)
      Should revert if startPrice is invalid (499ms)
      Should list dutch auction order, update storage and emit event
        ↪ (696ms)
      Should handle multiple dutch auction orders (1557ms)
    English Auction Listing
```

```
      Should revert if transaction approval hasn't been set (507ms)
      Should revert if duration is less than min allowed (454ms)
      Should revert if startPrice is invalid (431ms)
      Should list english auction order, update storage and emit event
         ↪ (612ms)
      Should handle multiple english auction orders (1311ms)
   Bidding
      Should revert if price is wrong (558ms)
      Should revert if not English Auction (536ms)
      Should revert if order was canceled (533ms)
      Should revert if order has timed out (529ms)
      Should revert if bidder is the seller (484ms)
      Should bid, update storage and emit events (573ms)
   Buying
      Should revert if price is wrong (512ms)
      Should revert if order is an English Auction (515ms)
      Should revert if order was canceled (776ms)
      Should revert if order has timed out (641ms)
      Should revert if token has already been sold (605ms)
      Should buy inhouse minted tokens, update storage and emit events
         ↪ (1423ms)
      Should verify inhouse minted tokens balance changes (1193ms)
      Should buy third party minted tokens with ERC2981 support (592ms)
      Should buy third party minted tokens without ERC2981 support (672
         ↪ ms)
      Should verify inhouse minted tokens balance changes - set fees
         ↪ (1175ms)
      Should buy third party minted tokens with ERC2981 support - set
         ↪ fees (604ms)
      Should buy third party minted tokens without ERC2981 support - set
         ↪  fees (607ms)
   Claim
      Should revert if caller is seller or bidder (535ms)
      Should revert if token has already been claimed (1116ms)
```

```
      Should revert if orderType is not an english auction (298ms)
      Should revert if auction hasn't ended (516ms)
      Should claim inhouse minted tokens, update storage and emit events
        ↪  (822ms)
      Should verify inhouse minted tokens balance changes (735ms)
      Should claim third party minted tokens with ERC2981 support (404ms
        ↪ )
      Should claim third party minted tokens without ERC2981 support
        ↪ (381ms)
    Order Cancelling
      Should revert due to already sold fixed price order (671ms)
      Should revert due to already sold dutch auction order (599ms)
      Should revert due to already sold english auction order (626ms)
      Should cancel fixed price order (671ms)
      Should cancel dutch auction order (625ms)
      Should cancel english auction order (591ms)
    Public Helpers
      Should fetch the length of orderIds for a token (819ms)
      Should fetch the length of orderIds for a seller (835ms)

  MADMarketplace721
    Init
      Marketplace should initialize (43ms)
    Owner Functions
      Should update factory address (45ms)
      Should update marketplace settings (46ms)
      Should initialize paused and unpaused states (187ms)
      Should update recipient (49ms)
      Should update contract's owner (44ms)
      Should withdraw to owner (114ms)
      Should delete order (720ms)
    Fixed Price Listing
      Should revert if transaction approval hasn't been set (511ms)
      Should revert if duration is less than min allowed (446ms)
```

```
      Should revert if price is invalid (475ms)
      Should list fixed price order, update storage and emit event (570
         ↪ ms)
      Should handle multiple fixed price orders (1324ms)
   Dutch Auction Listing
      Should revert if transaction approval hasn't been set (491ms)
      Should revert if duration is less than min allowed (514ms)
      Should revert if startPrice is invalid (550ms)
      Should list dutch auction order, update storage and emit event
         ↪ (772ms)
      Should handle multiple dutch auction orders (1434ms)
   English Auction Listing
      Should revert if transaction approval hasn't been set (498ms)
      Should revert if duration is less than min allowed (537ms)
      Should revert if startPrice is invalid (456ms)
      Should list english auction order, update storage and emit event
         ↪ (729ms)
      Should handle multiple english auction orders (1370ms)
   Bidding
      Should revert if price is wrong (667ms)
      Should revert if not English Auction (625ms)
      Should revert if order was canceled (549ms)
      Should revert if order has timed out (824ms)
      Should revert if bidder is the seller (736ms)
      Should bid, update storage and emit events (850ms)
   Buying
      Should revert if price is wrong (583ms)
      Should revert if order is an English Auction (577ms)
      Should revert if order was canceled (775ms)
      Should revert if order has timed out (526ms)
      Should revert if token has already been sold (625ms)
      Should buy inhouse minted tokens, update storage and emit events
         ↪ (1347ms)
      Should verify inhouse minted tokens balance changes (1101ms)
```

```
BigNumber { value: "347222222222222264" } BigNumber { value:
    ↪ "8680555555555556" }
      Should buy third party minted tokens with ERC2981 support (524ms)
      Should buy third party minted tokens without ERC2981 support (450
          ↪ ms)
      Should verify inhouse minted tokens balance changes - fee change
          ↪ update (1141ms)
BigNumber { value: "347222222222222264" } BigNumber { value:
    ↪ "17361111111111113" }
      Should buy third party minted tokens with ERC2981 support - fee
          ↪ change update (671ms)
      Should buy third party minted tokens without ERC2981 support - fee
          ↪  change update (655ms)
    Claim
      Should revert if caller is seller or bidder (694ms)
      Should revert if token has already been claimed (1072ms)
      Should revert if orderType is not an english auction (415ms)
      Should revert if auction hasn't ended (590ms)
      Should claim inhouse minted tokens, update storage and emit events
          ↪  (844ms)
      Should verify inhouse minted tokens balance changes (771ms)
      Should claim third party minted tokens with ERC2981 support (711ms
          ↪ )
      Should claim third party minted tokens without ERC2981 support
          ↪ (447ms)
    Order Cancelling
      Should revert due to already sold fixed price order (713ms)
      Should revert due to already sold dutch auction order (685ms)
      Should revert due to already sold english auction order (717ms)
      Should cancel fixed price order (621ms)
      Should cancel dutch auction order (704ms)
      Should cancel english auction order (734ms)
    Public Helpers
      Should fetch the length of orderIds for a token (906ms)
```

```
      Should fetch the length of orderIds for a seller (788ms)


  MADRouter1155
    Init
      Router should initialize
    Set URI
      Should revert for invalid collection type (396ms)
      Should set URI for 1155Basic collection type (508ms)
      Should set URI for 1155Whitelist collection type (598ms)
      Should set URI for 1155Lazy collection type (502ms)
    Whitelist Settings
      Should revert for invalid collection type (440ms)
      Should set whitelist config for 1155Whitelist collection type (494
        ↪ ms)
    FreeClaim Settings
      Should revert for invalid collection type (439ms)
      Should set freeClaim config for 1155Whitelist collection type (521
        ↪ ms)
    Minimal SafeMint
      Should revert for invalid collection type (462ms)
(node:1617) PromiseRejectionHandledWarning: Promise rejection was
  ↪ handled asynchronously (rejection id: 14)
(Use `node --trace-warnings ...` to show where the warning was created)
      Should call safeMint for 1155Minimal collection type (456ms)
    Burn
      Should burn token for 1155Minimal collection type (605ms)
      Should burn tokens for 1155Basic collection type (628ms)
      Should burn tokens for 1155Whitelist collection type (630ms)
      Should burn tokens for 1155Lazy collection type (627ms)
    Batch Burn
      Should revert for invalid collection type (432ms)
      Should batch burn token for 1155Basic collection type (611ms)
      Should batch burn tokens for 1155Whitelist collection type (730ms)
        ↪
```

```
      Should batch burn tokens for 1155Lazy collection type (612ms)
    Set MintState
      Should revert for invalid stateType
      Should revert for invalid tokenType (373ms)
      Should set publicMintState for minimal, basic and whitelist
        ↪ colTypes (1111ms)
      Should set whitelistMintState for whitelist colType (538ms)
      Should set freeClaimState for whitelist colType (549ms)
    Whitelist Creator Mint
      Should revert for invalid coltype (408ms)
      Should mint to creator (619ms)
    Whitelist Creator Batch Mint
      Should revert for invalid coltype (486ms)
      Should batch mint to creator (431ms)
      Should mint to creator (629ms)
    Whitelist token gifting
      Should revert for invalid coltype (416ms)
      Should gift tokens (613ms)
    Creator Withdraw
      Should withdraw balance and ERC20 for all colTypes (3472ms)
    Only Owner
      Should update contract's owner (43ms)
      Should initialize paused and unpaused states (219ms)
    Minimal SafeMint
      Should call safeMint for 1155Minimal collection type (510ms)
    Burn-setfees
      Should burn token for 1155Minimal collection type (551ms)
fee is BigNumber { value: "500000000000000000" }
      Should burn tokens for 1155Basic collection type (655ms)
      Should burn tokens for 1155Whitelist collection type (803ms)
      Should burn tokens for 1155Lazy collection type (708ms)
    Batch Burn
      Should revert for invalid collection type (507ms)
      Should batch burn token for 1155Basic collection type (758ms)
```

```
      Should batch burn tokens for 1155Whitelist collection type (947ms)
        ↪
      Should batch burn tokens for 1155Lazy collection type (680ms)
    Whitelist Creator Mint
      Should revert for invalid coltype (399ms)
      Should mint to creator (620ms)
    Whitelist Creator Batch Mint
      Should mint to creator (786ms)
    Whitelist token gifting
      Should gift tokens (799ms)

  MADRouter721
    Init
      Router should initialize
    Set baseURI
      Should revert for invalid collection type (363ms)
      Should set baseURI for 721Basic collection type (469ms)
      Should set baseURI for 721Whitelist collection type (491ms)
      Should set baseURI for 721Lazy collection type (640ms)
    Whitelist Settings
      Should revert for invalid collection type (462ms)
      Should set whitelist config for 721Whitelist collection type (583
        ↪ ms)
    FreeClaim Settings
      Should revert for invalid collection type (520ms)
      Should set freeClaim config for 721Whitelist collection type (575
        ↪ ms)
    Minimal SafeMint
      Should revert for invalid collection type (413ms)
(node:1617) PromiseRejectionHandledWarning: Promise rejection was
  ↪ handled asynchronously (rejection id: 15)
BigNumber { value: "2500000000000000000" }
minted successfully
      Should call safeMint for 721Minimal collection type (536ms)
```

```
Burn
  Should burn token for 721Minimal collection type (482ms)
  Should burn tokens for 721Basic collection type (661ms)
  Should burn tokens for 721Whitelist collection type (931ms)
  Should burn tokens for 721Lazy collection type (582ms)
Set MintState
  Should revert for invalid stateType
  Should revert for invalid tokenType (347ms)
  Should set publicMintState for minimal, basic and whitelist
    ↪ colTypes (1053ms)
  Should set whitelistMintState for whitelist colType (469ms)
  Should set freeClaimState for whitelist colType (465ms)
Whitelist Creator Mint
  Should revert for invalid coltype (375ms)
  Should mint to creator (563ms)
Whitelist token gifting
  Should revert for invalid coltype (377ms)
  Should gift tokens (580ms)
Creator Withdraw
  Should withdraw balance and ERC20 for all colTypes (2974ms)
Only Owner
  Should update contract's owner (40ms)
  Should initialize paused and unpaused states (195ms)
Minimal SafeMint-setBaseFee
  Should call safeMint for 721Minimal collection type (557ms)
Burn-setBaseFee
  Should burn tokens for 721Basic collection type (596ms)
  Should burn tokens for 721Whitelist collection type (611ms)
  Should burn tokens for 721Lazy collection type (617ms)
Whitelist Creator Mint-setBaseFee
  Should mint to creator (599ms)
Whitelist token gifting-setBaseFee
  Should gift tokens (599ms)
```

```
Royalties
   Royalties should initialize
   Should retrive royalty info
   Should accept recipient and fee change (84ms)
   Should support interfaces

Splitter
  Init
     Splitter should initialize (67ms)
     accounts have been funded
  Reverts
     should revert if no payees are provided
     should revert if more payees than shares are provided
     should revert if more shares than payees are provided
     should revert if dead address is provided as payee
     should revert if a share is set to zero
     should revert if a provided payees are duplicated (38ms)
     should revert if a provided payees are duplicated (41ms)
     should revert if account has no shares to claim
     should revert if there are no funds to claim
     should revert if account has no ERC20 shares to claim (86ms)
     should revert if there is no ERC20 to claim (93ms)
  Receive Payments
     should accept value and autodistribute to payees (265ms)
     should accept ERC20 (103ms)
  Release Payments
     should release value to payee (87ms)
     should release all pending balance to payees (243ms)
     should release ERC20 to payee (179ms)


471 passing (5m)
```

6 Static Analysis (Slither)

Description:

Block Hat expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
MADMarketplace721.feeSelector (MADMarketplace721.sol#69-70) is never
   ↪ initialized. It is used in:
       - MADMarketplace721.buy(bytes32) (MADMarketplace721.sol#204-267)
       - MADMarketplace721.claim(bytes32) (MADMarketplace721.sol
          ↪ #272-335)
       - MADMarketplace721._feeResolver(uint256,uint256) (
          ↪ MADMarketplace721.sol#725-745)
MADMarketplace721.minOrderDuration (MADMarketplace721.sol#72) is never
   ↪ initialized. It is used in:
       - MADMarketplace721.updateSettings(uint256,uint256,uint256) (
          ↪ MADMarketplace721.sol#405-429)
       - MADMarketplace721._makeOrderChecks(uint256,uint256) (
          ↪ MADMarketplace721.sol#789-818)
MADMarketplace721.minAuctionIncrement (MADMarketplace721.sol#73) is
   ↪ never initialized. It is used in:
       - MADMarketplace721.bid(bytes32) (MADMarketplace721.sol#148-199)
       - MADMarketplace721.updateSettings(uint256,uint256,uint256) (
          ↪ MADMarketplace721.sol#405-429)
MADMarketplace721.minBidValue (MADMarketplace721.sol#74) is never
   ↪ initialized. It is used in:
```

```
        - MADMarketplace721.updateSettings(uint256,uint256,uint256) (
            ↪ MADMarketplace721.sol#405-429)
        - MADMarketplace721._bidChecks(uint8,uint256,address,uint256,
            ↪ uint256) (MADMarketplace721.sol#844-899)
MADMarketplace721.recipient (MADMarketplace721.sol#76) is never
    ↪ initialized. It is used in:
        - MADMarketplace721.setRecipient(address) (MADMarketplace721.sol
            ↪ #445-457)
        - MADMarketplace721._intPath(Types.Order721,uint256,bytes32,
            ↪ address,uint256) (MADMarketplace721.sol#603-646)
        - MADMarketplace721._extPath0(Types.Order721,uint256,bytes32,
            ↪ address) (MADMarketplace721.sol#648-690)
        - MADMarketplace721._extPath1(Types.Order721,uint256,bytes32,
            ↪ address) (MADMarketplace721.sol#692-723)
MADMarketplace721.MADFactory721 (MADMarketplace721.sol#77) is never
    ↪ initialized. It is used in:
        - MADMarketplace721.buy(bytes32) (MADMarketplace721.sol#204-267)
        - MADMarketplace721.claim(bytes32) (MADMarketplace721.sol
            ↪ #272-335)
        - MADMarketplace721.setFactory(FactoryVerifier) (
            ↪ MADMarketplace721.sol#370-379)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-state-variables

MADMarketplace721.getCurrentPrice(bytes32) (MADMarketplace721.sol
    ↪ #967-1033) performs a multiplication on the result of a division:
        -_tick_getCurrentPrice_asm_0 = _startPrice_getCurrentPrice_asm_0
            ↪ - _endPrice_getCurrentPrice_asm_0 /
            ↪ _endTime_getCurrentPrice_asm_0 -
            ↪ _startTime_getCurrentPrice_asm_0 (MADMarketplace721.sol
            ↪ #1006-1009)
        -price = _startPrice_getCurrentPrice_asm_0 - timestamp()() -
            ↪ _startTime_getCurrentPrice_asm_0 *
            ↪ _tick_getCurrentPrice_asm_0 (MADMarketplace721.sol
```

```
            ↪ #1010-1013)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #divide-before-multiply

SplitterImpl.release(ERC20,address) (lib/splitter/SplitterImpl.sol
    ↪ #147-171) uses a dangerous strict equality:
        - payment == 0 (lib/splitter/SplitterImpl.sol#154)
SplitterImpl.release(address) (lib/splitter/SplitterImpl.sol#107-127)
    ↪ uses a dangerous strict equality:
        - payment == 0 (lib/splitter/SplitterImpl.sol#112)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dangerous-strict-equalities

Contract locking ether found:
        Contract MADMarketplace721 (MADMarketplace721.sol#22-1060) has
            ↪ payable functions:
         - MADMarketplace721.bid(bytes32) (MADMarketplace721.sol#148-199)
         - MADMarketplace721.buy(bytes32) (MADMarketplace721.sol#204-267)
         - MADMarketplace721.receive() (MADMarketplace721.sol#362)
        But does not have a function to withdraw the ether
Contract locking ether found:
        Contract SplitterImpl (lib/splitter/SplitterImpl.sol#27-297) has
            ↪ payable functions:
         - SplitterImpl.constructor(address[],uint256[]) (lib/splitter/
             ↪ SplitterImpl.sol#58-78)
         - SplitterImpl.receive() (lib/splitter/SplitterImpl.sol#92-98)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #contracts-that-lock-ether

SplitterImpl.releaseAll().i (lib/splitter/SplitterImpl.sol#132) is a
    ↪ local variable never initialized
SplitterImpl.constructor(address[],uint256[]).i (lib/splitter/
    ↪ SplitterImpl.sol#70) is a local variable never initialized
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-local-variables
Reentrancy in MADMarketplace721._extPath0(Types.Order721,uint256,bytes32
    ↪ ,address) (MADMarketplace721.sol#648-690):
        External calls:
        - _order.token.safeTransferFrom(address(this),_to,_order.tokenId)
            ↪  (MADMarketplace721.sol#677-681)
        Event emitted after the call(s):
        - Claim(_order.token,_order.tokenId,_orderId,_order.seller,_to,
            ↪ _price) (MADMarketplace721.sol#682-689)
Reentrancy in MADMarketplace721._extPath1(Types.Order721,uint256,bytes32
    ↪ ,address) (MADMarketplace721.sol#692-723):
        External calls:
        - _order.token.safeTransferFrom(address(this),_to,_order.tokenId)
            ↪  (MADMarketplace721.sol#710-714)
        Event emitted after the call(s):
        - Claim(_order.token,_order.tokenId,_orderId,_order.seller,_to,
            ↪ _price) (MADMarketplace721.sol#715-722)
Reentrancy in MADMarketplace721._intPath(Types.Order721,uint256,bytes32,
    ↪ address,uint256) (MADMarketplace721.sol#603-646):
        External calls:
        - _order.token.safeTransferFrom(address(this),_to,_order.tokenId)
            ↪  (MADMarketplace721.sol#633-637)
        Event emitted after the call(s):
        - Claim(_order.token,_order.tokenId,_orderId,_order.seller,_to,
            ↪ _price) (MADMarketplace721.sol#638-645)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (lib
    ↪ /tokens/ERC20.sol#134-185) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,
            ↪ PERMIT_DEADLINE_EXPIRED) (lib/tokens/ERC20.sol#143-146)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #block-timestamp

MADMarketplace721.name() (MADMarketplace721.sol#32-43) uses assembly

MADFactory1155.market (MADFactory1155.sol#88) is never initialized. It
    ↪ is used in:
        - MADFactory1155.setMarket(address) (MADFactory1155.sol#499-506)
        - MADFactory1155._isMarket() (MADFactory1155.sol#777-784)
MADFactory1155.signer (MADFactory1155.sol#94) is never initialized. It
    ↪ is used in:
        - MADFactory1155.createCollection(uint8,string,string,string,
            ↪ uint256,uint256,string,address,uint256) (MADFactory1155.
            ↪ sol#327-476)
        - MADFactory1155.setSigner(address) (MADFactory1155.sol#522-530)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-state-variables

SplitterImpl.release(address) (lib/splitter/SplitterImpl.sol#107-127)
    ↪ uses a dangerous strict equality:
        - payment == 0 (lib/splitter/SplitterImpl.sol#112)
SplitterImpl.release(ERC20,address) (lib/splitter/SplitterImpl.sol
    ↪ #147-171) uses a dangerous strict equality:
        - payment == 0 (lib/splitter/SplitterImpl.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dangerous-strict-equalities

Contract locking ether found:
        Contract SplitterImpl (lib/splitter/SplitterImpl.sol#27-297) has
            ↪ payable functions:
         - SplitterImpl.constructor(address[],uint256[]) (lib/splitter/
             ↪ SplitterImpl.sol#58-78)
         - SplitterImpl.receive() (lib/splitter/SplitterImpl.sol#92-98)
        But does not have a function to withdraw the ether
```

```
Contract locking ether found:
        Contract ERC1155Basic (lib/tokens/ERC1155/Impl/ERC1155Basic.sol
            ↪ #18-420) has payable functions:
         - ERC1155Basic.mintTo(address,uint256,uint256[]) (lib/tokens/
            ↪ ERC1155/Impl/ERC1155Basic.sol#105-129)
         - ERC1155Basic.mintBatchTo(address,uint256[],uint256[]) (lib/
            ↪ tokens/ERC1155/Impl/ERC1155Basic.sol#131-154)
         - ERC1155Basic.burn(address[],uint256[],uint256[]) (lib/tokens/
            ↪ ERC1155/Impl/ERC1155Basic.sol#157-176)
         - ERC1155Basic.burnBatch(address,uint256[],uint256[]) (lib/
            ↪ tokens/ERC1155/Impl/ERC1155Basic.sol#179-202)
         - ERC1155Basic.mint(uint256,uint256) (lib/tokens/ERC1155/Impl/
            ↪ ERC1155Basic.sol#263-286)
         - ERC1155Basic.mintBatch(uint256[],uint256[]) (lib/tokens/
            ↪ ERC1155/Impl/ERC1155Basic.sol#289-313)
        But does not have a function to withdraw the ether
Contract locking ether found:
        Contract ERC1155Lazy (lib/tokens/ERC1155/Impl/ERC1155Lazy.sol
            ↪ #18-498) has payable functions:
         - ERC1155Lazy.lazyMint(Types.Voucher,uint8,bytes32,bytes32) (lib
            ↪ /tokens/ERC1155/Impl/ERC1155Lazy.sol#99-117)
         - ERC1155Lazy.lazyMintBatch(Types.UserBatch,uint8,bytes32,
            ↪ bytes32) (lib/tokens/ERC1155/Impl/ERC1155Lazy.sol#120-142)
         - ERC1155Lazy.burn(address[],uint256[],uint256[]) (lib/tokens/
            ↪ ERC1155/Impl/ERC1155Lazy.sol#164-185)
         - ERC1155Lazy.burnBatch(address,uint256[],uint256[]) (lib/tokens
            ↪ /ERC1155/Impl/ERC1155Lazy.sol#188-212)
        But does not have a function to withdraw the ether
Contract locking ether found:
        Contract ERC1155Minimal (lib/tokens/ERC1155/Impl/ERC1155Minimal.
            ↪ sol#14-210) has payable functions:
         - ERC1155Minimal.safeMint(address,uint256) (lib/tokens/ERC1155/
            ↪ Impl/ERC1155Minimal.sol#60-66)
```

```
        - ERC1155Minimal.burn(address,uint256) (lib/tokens/ERC1155/Impl/
            ↪ ERC1155Minimal.sol#69-72)
        - ERC1155Minimal.publicMint(uint256) (lib/tokens/ERC1155/Impl/
            ↪ ERC1155Minimal.sol#142-149)
      But does not have a function to withdraw the ether
Contract locking ether found:
      Contract ERC1155Whitelist (lib/tokens/ERC1155/Impl/
         ↪ ERC1155Whitelist.sol#19-703) has payable functions:
        - ERC1155Whitelist.burn(address[],uint256[],uint256[]) (lib/
            ↪ tokens/ERC1155/Impl/ERC1155Whitelist.sol#228-247)
        - ERC1155Whitelist.burnBatch(address,uint256[],uint256[]) (lib/
            ↪ tokens/ERC1155/Impl/ERC1155Whitelist.sol#250-274)
        - ERC1155Whitelist.mintToCreator(uint256,uint256[],uint256) (lib
            ↪ /tokens/ERC1155/Impl/ERC1155Whitelist.sol#276-300)
        - ERC1155Whitelist.mintBatchToCreator(uint256[],uint256[],
            ↪ uint256) (lib/tokens/ERC1155/Impl/ERC1155Whitelist.sol
            ↪ #302-332)
        - ERC1155Whitelist.giftTokens(address[],uint256[],uint256) (lib/
            ↪ tokens/ERC1155/Impl/ERC1155Whitelist.sol#335-364)
        - ERC1155Whitelist.mint(uint256,uint256[],uint256) (lib/tokens/
            ↪ ERC1155/Impl/ERC1155Whitelist.sol#425-451)
        - ERC1155Whitelist.mintBatch(uint256[],uint256[]) (lib/tokens/
            ↪ ERC1155/Impl/ERC1155Whitelist.sol#453-480)
        - ERC1155Whitelist.whitelistMint(uint8,uint256[],uint256,bytes32
            ↪ []) (lib/tokens/ERC1155/Impl/ERC1155Whitelist.sol#482-517)
        - ERC1155Whitelist.whitelistMintBatch(uint256[],uint256[],
            ↪ bytes32[]) (lib/tokens/ERC1155/Impl/ERC1155Whitelist.sol
            ↪ #519-553)
      But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #contracts-that-lock-ether

Reentrancy in MADFactory1155.createCollection(uint8,string,string,string
   ↪ ,uint256,uint256,string,address,uint256) (MADFactory1155.sol
```

```
    ↪ #327-476):
        External calls:
        - (tokenSalt,deployed) = ERC1155MinimalDeployer.
            ↪ _1155MinimalDeploy(_tokenSalt,_uri,_price,_splitter,router
            ↪ ,_royalty) (MADFactory1155.sol#347-355)
        - (tokenSalt,deployed) = ERC1155BasicDeployer._1155BasicDeploy(
            ↪ _tokenSalt,_uri,_price,_maxSupply,_splitter,router,
            ↪ _royalty) (MADFactory1155.sol#379-388)
        State variables written after the call(s):
        - userTokens[tx.origin].push(colId_scope_2) (MADFactory1155.sol
            ↪ #391)
    Reentrancy in MADFactory1155.createCollection(uint8,string,string,string
    ↪ ,uint256,uint256,string,address,uint256) (MADFactory1155.sol
    ↪ #327-476):
        External calls:
        - (tokenSalt,deployed) = ERC1155MinimalDeployer.
            ↪ _1155MinimalDeploy(_tokenSalt,_uri,_price,_splitter,router
            ↪ ,_royalty) (MADFactory1155.sol#347-355)
        - (tokenSalt,deployed) = ERC1155BasicDeployer._1155BasicDeploy(
            ↪ _tokenSalt,_uri,_price,_maxSupply,_splitter,router,
            ↪ _royalty) (MADFactory1155.sol#379-388)
        - (tokenSalt,deployed) = ERC1155WhitelistDeployer.
            ↪ _1155WhitelistDeploy(_tokenSalt,_uri,_price,_maxSupply,
            ↪ _splitter,router,_royalty) (MADFactory1155.sol#412-421)
        State variables written after the call(s):
        - userTokens[tx.origin].push(colId_scope_5) (MADFactory1155.sol
            ↪ #424)
    Reentrancy in MADFactory1155.createCollection(uint8,string,string,string
    ↪ ,uint256,uint256,string,address,uint256) (MADFactory1155.sol
    ↪ #327-476):
        External calls:
        - (tokenSalt,deployed) = ERC1155MinimalDeployer.
            ↪ _1155MinimalDeploy(_tokenSalt,_uri,_price,_splitter,router
            ↪ ,_royalty) (MADFactory1155.sol#347-355)
```

```
        - (tokenSalt,deployed) = ERC1155BasicDeployer._1155BasicDeploy(
            ↪ _tokenSalt,_uri,_price,_maxSupply,_splitter,router,
            ↪ _royalty) (MADFactory1155.sol#379-388)
        - (tokenSalt,deployed) = ERC1155WhitelistDeployer.
            ↪ _1155WhitelistDeploy(_tokenSalt,_uri,_price,_maxSupply,
            ↪ _splitter,router,_royalty) (MADFactory1155.sol#412-421)
        - (tokenSalt,deployed) = ERC1155LazyDeployer._1155LazyDeploy(
            ↪ _tokenSalt,_uri,_splitter,router,signer,_royalty) (
            ↪ MADFactory1155.sol#445-453)
        State variables written after the call(s):
        - userTokens[tx.origin].push(colId_scope_8) (MADFactory1155.sol
            ↪ #456)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-1

MADFactory1155.creatorCheck(bytes32) (MADFactory1155.sol#736-759) uses
    ↪ tx.origin for authorization: creator == origin()() (
    ↪ MADFactory1155.sol#750-752)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dangerous-usage-of-txorigin

Counters.decrement(Counters.Counter) (lib/utils/Counters.sol#43-53)
    ↪ contains a tautology or contradiction:
        - ! _val_decrement_asm_0 | _val_decrement_asm_0 < 0x00 (lib/utils
            ↪ /Counters.sol#46-50)
Counters.decrement(Counters.Counter,uint256) (lib/utils/Counters.sol
    ↪ #55-65) contains a tautology or contradiction:
        - ! _val_decrement_asm_0 | _val_decrement_asm_0 < 0x00 |
            ↪ _val_decrement_asm_0 - amount < 0x00 (lib/utils/Counters.
            ↪ sol#58-62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #tautology-or-contradiction
```

```
ERC1155Basic.withdraw().j (lib/tokens/ERC1155/Impl/ERC1155Basic.sol#219)
   ↪  is a local variable never initialized
ERC1155Basic._sumAmounts(uint256[]).i (lib/tokens/ERC1155/Impl/
   ↪ ERC1155Basic.sol#332) is a local variable never initialized
ERC1155Whitelist.withdraw().j (lib/tokens/ERC1155/Impl/ERC1155Whitelist.
   ↪ sol#381) is a local variable never initialized
ERC1155Whitelist.withdrawERC20(ERC20).i (lib/tokens/ERC1155/Impl/
   ↪ ERC1155Whitelist.sol#397) is a local variable never initialized
ERC1155Basic.withdrawERC20(ERC20).j (lib/tokens/ERC1155/Impl/
   ↪ ERC1155Basic.sol#246) is a local variable never initialized
MADFactory1155.createCollection(uint8,string,string,string,uint256,
   ↪ uint256,string,address,uint256).tokenSalt_scope_0 (MADFactory1155
   ↪ .sol#379) is a local variable never initialized
ERC1155Lazy.lazyMintBatch(Types.UserBatch,uint8,bytes32,bytes32).i (lib/
   ↪ tokens/ERC1155/Impl/ERC1155Lazy.sol#131) is a local variable
   ↪ never initialized
ERC1155Lazy._userMint(uint256,uint256[],address).j (lib/tokens/ERC1155/
   ↪ Impl/ERC1155Lazy.sol#404) is a local variable never initialized
MADFactory1155.createCollection(uint8,string,string,string,uint256,
   ↪ uint256,string,address,uint256).deployed_scope_1 (MADFactory1155.
   ↪ sol#379) is a local variable never initialized
ERC1155Lazy.withdrawERC20(ERC20).j (lib/tokens/ERC1155/Impl/ERC1155Lazy.
   ↪ sol#256) is a local variable never initialized
MADFactory1155.createCollection(uint8,string,string,string,uint256,
   ↪ uint256,string,address,uint256).tokenSalt_scope_6 (MADFactory1155
   ↪ .sol#445) is a local variable never initialized
ERC1155Minimal.withdrawERC20(ERC20).i (lib/tokens/ERC1155/Impl/
   ↪ ERC1155Minimal.sol#114) is a local variable never initialized
ERC1155Lazy.lazyMint(Types.Voucher,uint8,bytes32,bytes32).i (lib/tokens/
   ↪ ERC1155/Impl/ERC1155Lazy.sol#109) is a local variable never
   ↪ initialized
ERC1155Lazy.withdraw().i (lib/tokens/ERC1155/Impl/ERC1155Lazy.sol#219)
   ↪ is a local variable never initialized
```

```
MADFactory1155.createCollection(uint8,string,string,string,uint256,
    ↪ uint256,string,address,uint256).deployed_scope_7 (MADFactory1155.
    ↪ sol#445) is a local variable never initialized
ERC1155Minimal.withdraw().i (lib/tokens/ERC1155/Impl/ERC1155Minimal.sol
    ↪ #88) is a local variable never initialized
ERC1155Whitelist.withdrawERC20(ERC20).j (lib/tokens/ERC1155/Impl/
    ↪ ERC1155Whitelist.sol#408) is a local variable never initialized
SplitterImpl.releaseAll().i (lib/splitter/SplitterImpl.sol#132) is a
    ↪ local variable never initialized
ERC1155Basic.withdraw().i (lib/tokens/ERC1155/Impl/ERC1155Basic.sol#209)
    ↪  is a local variable never initialized
ERC1155Basic.withdrawERC20(ERC20).i (lib/tokens/ERC1155/Impl/
    ↪ ERC1155Basic.sol#235) is a local variable never initialized
ERC1155Whitelist.withdraw().i (lib/tokens/ERC1155/Impl/ERC1155Whitelist.
    ↪ sol#371) is a local variable never initialized
ERC1155Lazy.withdraw().j (lib/tokens/ERC1155/Impl/ERC1155Lazy.sol#229)
    ↪ is a local variable never initialized
MADFactory1155.createCollection(uint8,string,string,string,uint256,
    ↪ uint256,string,address,uint256).tokenSalt_scope_3 (MADFactory1155
    ↪ .sol#412) is a local variable never initialized
ERC1155Minimal.withdrawERC20(ERC20).j (lib/tokens/ERC1155/Impl/
    ↪ ERC1155Minimal.sol#125) is a local variable never initialized
ERC1155Lazy.withdrawERC20(ERC20).i (lib/tokens/ERC1155/Impl/ERC1155Lazy.
    ↪ sol#245) is a local variable never initialized
MADFactory1155.createCollection(uint8,string,string,string,uint256,
    ↪ uint256,string,address,uint256).deployed_scope_4 (MADFactory1155.
    ↪ sol#412) is a local variable never initialized
SplitterImpl.constructor(address[],uint256[]).i (lib/splitter/
    ↪ SplitterImpl.sol#70) is a local variable never initialized
ERC1155Minimal.withdraw().j (lib/tokens/ERC1155/Impl/ERC1155Minimal.sol
    ↪ #98) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-local-variables
```

```
ERC1155Whitelist.mintToCreator(uint256,uint256[],uint256) (lib/tokens/
    ↪ ERC1155/Impl/ERC1155Whitelist.sol#276-300) should emit an event
    ↪ for:
        - freeSupply += balanceTotal (lib/tokens/ERC1155/Impl/
            ↪ ERC1155Whitelist.sol#286)
ERC1155Whitelist.mintBatchToCreator(uint256[],uint256[],uint256) (lib/
    ↪ tokens/ERC1155/Impl/ERC1155Whitelist.sol#302-332) should emit an
    ↪ event for:
        - freeSupply += balanceTotal (lib/tokens/ERC1155/Impl/
            ↪ ERC1155Whitelist.sol#313)
ERC1155Whitelist.giftTokens(address[],uint256[],uint256) (lib/tokens/
    ↪ ERC1155/Impl/ERC1155Whitelist.sol#335-364) should emit an event
    ↪ for:
        - freeSupply += amountGifted (lib/tokens/ERC1155/Impl/
            ↪ ERC1155Whitelist.sol#347)
ERC1155Whitelist.whitelistMint(uint8,uint256[],uint256,bytes32[]) (lib/
    ↪ tokens/ERC1155/Impl/ERC1155Whitelist.sol#482-517) should emit an
    ↪ event for:
        - whitelistMinted += balanceTotal (lib/tokens/ERC1155/Impl/
            ↪ ERC1155Whitelist.sol#507)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-events-arithmetic

MADFactory1155.constructor(address,address,address)._router (
    ↪ MADFactory1155.sol#103) lacks a zero-check on :
              - router = _router (MADFactory1155.sol#111)
Owned.setOwner(address).newOwner (lib/auth/Owned.sol#43) lacks a zero-
    ↪ check on :
              - owner = newOwner (lib/auth/Owned.sol#48)
ERC1155Lazy.constructor(string,SplitterImpl,uint96,address,address).
    ↪ _signer (lib/tokens/ERC1155/Impl/ERC1155Lazy.sol#74) lacks a zero
    ↪ -check on :
              - signer = _signer (lib/tokens/ERC1155/Impl/ERC1155Lazy.
                  ↪ sol#78)
```

```
ERC1155Lazy.setSigner(address)._signer (lib/tokens/ERC1155/Impl/
    ↪ ERC1155Lazy.sol#149) lacks a zero-check on :
            - signer = _signer (lib/tokens/ERC1155/Impl/ERC1155Lazy.
              ↪ sol#150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation


MADFactory721.market (MADFactory721.sol#87) is never initialized. It is
    ↪ used in:
        - MADFactory721.setMarket(address) (MADFactory721.sol#506-514)
        - MADFactory721._isMarket() (MADFactory721.sol#788-795)
MADFactory721.signer (MADFactory721.sol#93) is never initialized. It is
    ↪ used in:
        - MADFactory721.createCollection(uint8,string,string,string,
          ↪ uint256,uint256,string,address,uint256) (MADFactory721.sol
          ↪ #326-483)
        - MADFactory721.setSigner(address) (MADFactory721.sol#531-540)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-state-variables

SplitterImpl.release(address) (lib/splitter/SplitterImpl.sol#107-127)
    ↪ uses a dangerous strict equality:
        - payment == 0 (lib/splitter/SplitterImpl.sol#112)
SplitterImpl.release(ERC20,address) (lib/splitter/SplitterImpl.sol
    ↪ #147-171) uses a dangerous strict equality:
        - payment == 0 (lib/splitter/SplitterImpl.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dangerous-strict-equalities

Contract locking ether found:
        Contract SplitterImpl (lib/splitter/SplitterImpl.sol#27-297) has
          ↪ payable functions:
```

```
        - SplitterImpl.constructor(address[],uint256[]) (lib/splitter/
          ↪ SplitterImpl.sol#58-78)
        - SplitterImpl.receive() (lib/splitter/SplitterImpl.sol#92-98)
      But does not have a function to withdraw the ether
Contract locking ether found:
      Contract ERC721Basic (lib/tokens/ERC721/Impl/ERC721Basic.sol
        ↪ #18-332) has payable functions:
        - ERC721Basic.mintTo(address,uint256) (lib/tokens/ERC721/Impl/
          ↪ ERC721Basic.sol#110-134)
        - ERC721Basic.burn(uint256[]) (lib/tokens/ERC721/Impl/
          ↪ ERC721Basic.sol#136-157)
        - ERC721Basic.mint(uint256) (lib/tokens/ERC721/Impl/ERC721Basic.
          ↪ sol#218-243)
      But does not have a function to withdraw the ether
Contract locking ether found:
      Contract ERC721Lazy (lib/tokens/ERC721/Impl/ERC721Lazy.sol
        ↪ #18-433) has payable functions:
        - ERC721Lazy.lazyMint(Types.Voucher,uint8,bytes32,bytes32) (lib/
          ↪ tokens/ERC721/Impl/ERC721Lazy.sol#95-113)
        - ERC721Lazy.burn(uint256[]) (lib/tokens/ERC721/Impl/ERC721Lazy.
          ↪ sol#137-158)
      But does not have a function to withdraw the ether
Contract locking ether found:
      Contract ERC721Minimal (lib/tokens/ERC721/Impl/ERC721Minimal.sol
        ↪ #16-217) has payable functions:
        - ERC721Minimal.safeMint(address) (lib/tokens/ERC721/Impl/
          ↪ ERC721Minimal.sol#65-72)
        - ERC721Minimal.burn() (lib/tokens/ERC721/Impl/ERC721Minimal.sol
          ↪ #75-78)
        - ERC721Minimal.publicMint() (lib/tokens/ERC721/Impl/
          ↪ ERC721Minimal.sol#148-156)
      But does not have a function to withdraw the ether
Contract locking ether found:
```

```
        Contract ERC721Whitelist (lib/tokens/ERC721/Impl/ERC721Whitelist.
           ↪ sol#19-529) has payable functions:
         - ERC721Whitelist.burn(uint256[]) (lib/tokens/ERC721/Impl/
            ↪ ERC721Whitelist.sol#222-242)
         - ERC721Whitelist.mintToCreator(uint256) (lib/tokens/ERC721/Impl
            ↪ /ERC721Whitelist.sol#244-267)
         - ERC721Whitelist.giftTokens(address[]) (lib/tokens/ERC721/Impl/
            ↪ ERC721Whitelist.sol#270-294)
         - ERC721Whitelist.mint(uint256) (lib/tokens/ERC721/Impl/
            ↪ ERC721Whitelist.sol#355-379)
         - ERC721Whitelist.whitelistMint(uint8,bytes32[]) (lib/tokens/
            ↪ ERC721/Impl/ERC721Whitelist.sol#381-411)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #contracts-that-lock-ether

Reentrancy in MADFactory721.createCollection(uint8,string,string,string,
   ↪ uint256,uint256,string,address,uint256) (MADFactory721.sol
   ↪ #326-483):
        External calls:
        - (tokenSalt,deployed) = ERC721MinimalDeployer._721MinimalDeploy(
           ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_splitter,router,
           ↪ _royalty) (MADFactory721.sol#346-356)
        - (tokenSalt,deployed) = ERC721BasicDeployer._721BasicDeploy(
           ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_maxSupply,
           ↪ _splitter,router,_royalty) (MADFactory721.sol#380-391)
        State variables written after the call(s):
        - userTokens[tx.origin].push(colId_scope_2) (MADFactory721.sol
           ↪ #394)
Reentrancy in MADFactory721.createCollection(uint8,string,string,string,
   ↪ uint256,uint256,string,address,uint256) (MADFactory721.sol
   ↪ #326-483):
        External calls:
```

```
        - (tokenSalt,deployed) = ERC721MinimalDeployer._721MinimalDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_splitter,router,
            ↪ _royalty) (MADFactory721.sol#346-356)
        - (tokenSalt,deployed) = ERC721BasicDeployer._721BasicDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_maxSupply,
            ↪ _splitter,router,_royalty) (MADFactory721.sol#380-391)
        - (tokenSalt,deployed) = ERC721WhitelistDeployer.
            ↪ _721WhitelistDeploy(_tokenSalt,_name,_symbol,_baseURI,
            ↪ _price,_maxSupply,_splitter,router,_royalty) (
            ↪ MADFactory721.sol#415-426)
        State variables written after the call(s):
        - userTokens[tx.origin].push(colId_scope_5) (MADFactory721.sol
            ↪ #429)
    Reentrancy in MADFactory721.createCollection(uint8,string,string,string,
        ↪ uint256,uint256,string,address,uint256) (MADFactory721.sol
        ↪ #326-483):
        External calls:
        - (tokenSalt,deployed) = ERC721MinimalDeployer._721MinimalDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_splitter,router,
            ↪ _royalty) (MADFactory721.sol#346-356)
        - (tokenSalt,deployed) = ERC721BasicDeployer._721BasicDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_maxSupply,
            ↪ _splitter,router,_royalty) (MADFactory721.sol#380-391)
        - (tokenSalt,deployed) = ERC721WhitelistDeployer.
            ↪ _721WhitelistDeploy(_tokenSalt,_name,_symbol,_baseURI,
            ↪ _price,_maxSupply,_splitter,router,_royalty) (
            ↪ MADFactory721.sol#415-426)
        - (tokenSalt,deployed) = ERC721LazyDeployer._721LazyDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_splitter,router,signer,
            ↪ _royalty) (MADFactory721.sol#450-460)
        State variables written after the call(s):
        - userTokens[tx.origin].push(colId_scope_8) (MADFactory721.sol
            ↪ #463)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-1

MADFactory721.creatorCheck(bytes32) (MADFactory721.sol#747-770) uses tx.
    ↪ origin for authorization: creator == origin()() (MADFactory721.
    ↪ sol#761-763)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dangerous-usage-of-txorigin

Counters.decrement(Counters.Counter) (lib/utils/Counters.sol#43-53)
    ↪ contains a tautology or contradiction:
        - ! _val_decrement_asm_0 | _val_decrement_asm_0 < 0x00 (lib/utils
          ↪ /Counters.sol#46-50)
Counters.decrement(Counters.Counter,uint256) (lib/utils/Counters.sol
    ↪ #55-65) contains a tautology or contradiction:
        - ! _val_decrement_asm_0 | _val_decrement_asm_0 < 0x00 |
          ↪ _val_decrement_asm_0 - amount < 0x00 (lib/utils/Counters.
          ↪ sol#58-62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #tautology-or-contradiction

ERC721Lazy.withdrawERC20(ERC20).i (lib/tokens/ERC721/Impl/ERC721Lazy.sol
    ↪ #191) is a local variable never initialized
MADFactory721.createCollection(uint8,string,string,string,uint256,
    ↪ uint256,string,address,uint256).deployed_scope_1 (MADFactory721.
    ↪ sol#380) is a local variable never initialized
ERC721Lazy.withdraw().i (lib/tokens/ERC721/Impl/ERC721Lazy.sol#165) is a
    ↪  local variable never initialized
ERC721Lazy._userMint(uint256,address).j (lib/tokens/ERC721/Impl/
    ↪ ERC721Lazy.sol#336) is a local variable never initialized
MADFactory721.createCollection(uint8,string,string,string,uint256,
    ↪ uint256,string,address,uint256).tokenSalt_scope_6 (MADFactory721.
    ↪ sol#450) is a local variable never initialized
```

```
ERC721Minimal.withdrawERC20(ERC20).j (lib/tokens/ERC721/Impl/
    ↪ ERC721Minimal.sol#131) is a local variable never initialized
MADFactory721.createCollection(uint8,string,string,string,uint256,
    ↪ uint256,string,address,uint256).deployed_scope_7 (MADFactory721.
    ↪ sol#450) is a local variable never initialized
ERC721Minimal.withdraw().j (lib/tokens/ERC721/Impl/ERC721Minimal.sol
    ↪ #104) is a local variable never initialized
ERC721Whitelist.withdrawERC20(ERC20).j (lib/tokens/ERC721/Impl/
    ↪ ERC721Whitelist.sol#338) is a local variable never initialized
ERC721Lazy.withdraw().j (lib/tokens/ERC721/Impl/ERC721Lazy.sol#175) is a
    ↪  local variable never initialized
ERC721Lazy.withdrawERC20(ERC20).j (lib/tokens/ERC721/Impl/ERC721Lazy.sol
    ↪ #202) is a local variable never initialized
ERC721Whitelist.withdraw().i (lib/tokens/ERC721/Impl/ERC721Whitelist.sol
    ↪ #301) is a local variable never initialized
ERC721Basic.withdrawERC20(ERC20).i (lib/tokens/ERC721/Impl/ERC721Basic.
    ↪ sol#190) is a local variable never initialized
ERC721Whitelist.withdrawERC20(ERC20).i (lib/tokens/ERC721/Impl/
    ↪ ERC721Whitelist.sol#327) is a local variable never initialized
ERC721Basic.withdraw().i (lib/tokens/ERC721/Impl/ERC721Basic.sol#164) is
    ↪  a local variable never initialized
MADFactory721.createCollection(uint8,string,string,string,uint256,
    ↪ uint256,string,address,uint256).tokenSalt_scope_3 (MADFactory721.
    ↪ sol#415) is a local variable never initialized
MADFactory721.createCollection(uint8,string,string,string,uint256,
    ↪ uint256,string,address,uint256).deployed_scope_4 (MADFactory721.
    ↪ sol#415) is a local variable never initialized
SplitterImpl.constructor(address[],uint256[]).i (lib/splitter/
    ↪ SplitterImpl.sol#70) is a local variable never initialized
ERC721Minimal.withdrawERC20(ERC20).i (lib/tokens/ERC721/Impl/
    ↪ ERC721Minimal.sol#120) is a local variable never initialized
ERC721Whitelist.withdraw().j (lib/tokens/ERC721/Impl/ERC721Whitelist.sol
    ↪ #311) is a local variable never initialized
```

```
ERC721Lazy.lazyMint(Types.Voucher,uint8,bytes32,bytes32).i (lib/tokens/
   ↪ ERC721/Impl/ERC721Lazy.sol#105) is a local variable never
   ↪ initialized
ERC721Basic.withdraw().j (lib/tokens/ERC721/Impl/ERC721Basic.sol#174) is
   ↪  a local variable never initialized
ERC721Basic.withdrawERC20(ERC20).j (lib/tokens/ERC721/Impl/ERC721Basic.
   ↪ sol#201) is a local variable never initialized
SplitterImpl.releaseAll().i (lib/splitter/SplitterImpl.sol#132) is a
   ↪ local variable never initialized
ERC721Minimal.withdraw().i (lib/tokens/ERC721/Impl/ERC721Minimal.sol#94)
   ↪  is a local variable never initialized
MADFactory721.createCollection(uint8,string,string,string,uint256,
   ↪ uint256,string,address,uint256).tokenSalt_scope_0 (MADFactory721.
   ↪ sol#380) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #uninitialized-local-variables

MADFactory721.constructor(address,address,address)._router (
   ↪ MADFactory721.sol#102) lacks a zero-check on :
              - router = _router (MADFactory721.sol#110)
Owned.setOwner(address).newOwner (lib/auth/Owned.sol#43) lacks a zero-
   ↪ check on :
              - owner = newOwner (lib/auth/Owned.sol#48)
ERC721Lazy.constructor(string,string,string,SplitterImpl,uint96,address,
   ↪ address)._signer (lib/tokens/ERC721/Impl/ERC721Lazy.sol#70) lacks
   ↪  a zero-check on :
              - signer = _signer (lib/tokens/ERC721/Impl/ERC721Lazy.sol
                 ↪ #74)
ERC721Lazy.setSigner(address)._signer (lib/tokens/ERC721/Impl/ERC721Lazy
   ↪ .sol#120) lacks a zero-check on :
              - signer = _signer (lib/tokens/ERC721/Impl/ERC721Lazy.sol
                 ↪ #121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #missing-zero-address-validation MADFactory721.sol analyzed (41
```

```
    ↪ contracts with 78 detectors), 352 result(s) found

        - MADFactory721.setMarket(address) (MADFactory721.sol#506-514)
        - MADFactory721._isMarket() (MADFactory721.sol#788-795)
MADFactory721.signer (MADFactory721.sol#93) is never initialized. It is
    ↪ used in:
        - MADFactory721.createCollection(uint8,string,string,string,
            ↪ uint256,uint256,string,address,uint256) (MADFactory721.sol
            ↪ #326-483)
```

```
MADFactory7
    ↪ .
    ↪ mark
    ↪
    ↪ (
    ↪ MADF
    ↪ .
    ↪ sol
    ↪ #87)
    ↪
    ↪ is
    ↪
    ↪ neve
    ↪
    ↪ init
    ↪ .
    ↪
    ↪ It
    ↪
    ↪ is
    ↪
    ↪ used
    ↪
    ↪ in
    ↪ :
    ↪
```

```
        - MADFactory721.setSigner(address) (MADFactory721.sol#531-540)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-state-variables

SplitterImpl.release(address) (lib/splitter/SplitterImpl.sol#107-127)
    ↪ uses a dangerous strict equality:
        - payment == 0 (lib/splitter/SplitterImpl.sol#112)
SplitterImpl.release(ERC20,address) (lib/splitter/SplitterImpl.sol
    ↪ #147-171) uses a dangerous strict equality:
        - payment == 0 (lib/splitter/SplitterImpl.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dangerous-strict-equalities

Contract locking ether found:
        Contract SplitterImpl (lib/splitter/SplitterImpl.sol#27-297) has
            ↪ payable functions:
         - SplitterImpl.constructor(address[],uint256[]) (lib/splitter/
             ↪ SplitterImpl.sol#58-78)
         - SplitterImpl.receive() (lib/splitter/SplitterImpl.sol#92-98)
        But does not have a function to withdraw the ether
Contract locking ether found:
        Contract ERC721Basic (lib/tokens/ERC721/Impl/ERC721Basic.sol
            ↪ #18-332) has payable functions:
         - ERC721Basic.mintTo(address,uint256) (lib/tokens/ERC721/Impl/
             ↪ ERC721Basic.sol#110-134)
         - ERC721Basic.burn(uint256[]) (lib/tokens/ERC721/Impl/
             ↪ ERC721Basic.sol#136-157)
         - ERC721Basic.mint(uint256) (lib/tokens/ERC721/Impl/ERC721Basic.
             ↪ sol#218-243)
        But does not have a function to withdraw the ether
Contract locking ether found:
        Contract ERC721Lazy (lib/tokens/ERC721/Impl/ERC721Lazy.sol
            ↪ #18-433) has payable functions:
```

```
        - ERC721Lazy.lazyMint(Types.Voucher,uint8,bytes32,bytes32) (lib/
            ↪ tokens/ERC721/Impl/ERC721Lazy.sol#95-113)
        - ERC721Lazy.burn(uint256[]) (lib/tokens/ERC721/Impl/ERC721Lazy.
            ↪ sol#137-158)
      But does not have a function to withdraw the ether
Contract locking ether found:
      Contract ERC721Minimal (lib/tokens/ERC721/Impl/ERC721Minimal.sol
         ↪ #16-217) has payable functions:
        - ERC721Minimal.safeMint(address) (lib/tokens/ERC721/Impl/
            ↪ ERC721Minimal.sol#65-72)
        - ERC721Minimal.burn() (lib/tokens/ERC721/Impl/ERC721Minimal.sol
            ↪ #75-78)
        - ERC721Minimal.publicMint() (lib/tokens/ERC721/Impl/
            ↪ ERC721Minimal.sol#148-156)
      But does not have a function to withdraw the ether
Contract locking ether found:
      Contract ERC721Whitelist (lib/tokens/ERC721/Impl/ERC721Whitelist.
         ↪ sol#19-529) has payable functions:
        - ERC721Whitelist.burn(uint256[]) (lib/tokens/ERC721/Impl/
            ↪ ERC721Whitelist.sol#222-242)
        - ERC721Whitelist.mintToCreator(uint256) (lib/tokens/ERC721/Impl
            ↪ /ERC721Whitelist.sol#244-267)
        - ERC721Whitelist.giftTokens(address[]) (lib/tokens/ERC721/Impl/
            ↪ ERC721Whitelist.sol#270-294)
        - ERC721Whitelist.mint(uint256) (lib/tokens/ERC721/Impl/
            ↪ ERC721Whitelist.sol#355-379)
        - ERC721Whitelist.whitelistMint(uint8,bytes32[]) (lib/tokens/
            ↪ ERC721/Impl/ERC721Whitelist.sol#381-411)
      But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #contracts-that-lock-ether

Reentrancy in MADFactory721.createCollection(uint8,string,string,string,
   ↪ uint256,uint256,string,address,uint256) (MADFactory721.sol
```

```
    ↪ #326-483):
        External calls:
        - (tokenSalt,deployed) = ERC721MinimalDeployer._721MinimalDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_splitter,router,
            ↪ _royalty) (MADFactory721.sol#346-356)
        - (tokenSalt,deployed) = ERC721BasicDeployer._721BasicDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_maxSupply,
            ↪ _splitter,router,_royalty) (MADFactory721.sol#380-391)
        State variables written after the call(s):
        - userTokens[tx.origin].push(colId_scope_2) (MADFactory721.sol
            ↪ #394)
    Reentrancy in MADFactory721.createCollection(uint8,string,string,string,
    ↪ uint256,uint256,string,address,uint256) (MADFactory721.sol
    ↪ #326-483):
        External calls:
        - (tokenSalt,deployed) = ERC721MinimalDeployer._721MinimalDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_splitter,router,
            ↪ _royalty) (MADFactory721.sol#346-356)
        - (tokenSalt,deployed) = ERC721BasicDeployer._721BasicDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_maxSupply,
            ↪ _splitter,router,_royalty) (MADFactory721.sol#380-391)
        - (tokenSalt,deployed) = ERC721WhitelistDeployer.
            ↪ _721WhitelistDeploy(_tokenSalt,_name,_symbol,_baseURI,
            ↪ _price,_maxSupply,_splitter,router,_royalty) (
            ↪ MADFactory721.sol#415-426)
        State variables written after the call(s):
        - userTokens[tx.origin].push(colId_scope_5) (MADFactory721.sol
            ↪ #429)
    Reentrancy in MADFactory721.createCollection(uint8,string,string,string,
    ↪ uint256,uint256,string,address,uint256) (MADFactory721.sol
    ↪ #326-483):
        External calls:
        - (tokenSalt,deployed) = ERC721MinimalDeployer._721MinimalDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_splitter,router,
```

```
            ↪ _royalty) (MADFactory721.sol#346-356)
        - (tokenSalt,deployed) = ERC721BasicDeployer._721BasicDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_maxSupply,
            ↪ _splitter,router,_royalty) (MADFactory721.sol#380-391)
        - (tokenSalt,deployed) = ERC721WhitelistDeployer.
            ↪ _721WhitelistDeploy(_tokenSalt,_name,_symbol,_baseURI,
            ↪ _price,_maxSupply,_splitter,router,_royalty) (
            ↪ MADFactory721.sol#415-426)
        - (tokenSalt,deployed) = ERC721LazyDeployer._721LazyDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_splitter,router,signer,
            ↪ _royalty) (MADFactory721.sol#450-460)
        State variables written after the call(s):
        - userTokens[tx.origin].push(colId_scope_8) (MADFactory721.sol
            ↪ #463)
    Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
        ↪ #reentrancy-vulnerabilities-1

    MADFactory721.creatorCheck(bytes32) (MADFactory721.sol#747-770) uses tx.
        ↪ origin for authorization: creator == origin()() (MADFactory721.
        ↪ sol#761-763)
    Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
        ↪ #dangerous-usage-of-txorigin

    Counters.decrement(Counters.Counter) (lib/utils/Counters.sol#43-53)
        ↪ contains a tautology or contradiction:
            - ! _val_decrement_asm_0 | _val_decrement_asm_0 < 0x00 (lib/utils
                ↪ /Counters.sol#46-50)
    Counters.decrement(Counters.Counter,uint256) (lib/utils/Counters.sol
        ↪ #55-65) contains a tautology or contradiction:
            - ! _val_decrement_asm_0 | _val_decrement_asm_0 < 0x00 |
                ↪ _val_decrement_asm_0 - amount < 0x00 (lib/utils/Counters.
                ↪ sol#58-62)
    Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
        ↪ #tautology-or-contradiction
```

```
ERC721Lazy.withdrawERC20(ERC20).i (lib/tokens/ERC721/Impl/ERC721Lazy.sol
   ↪ #191) is a local variable never initialized
MADFactory721.createCollection(uint8,string,string,string,uint256,
   ↪ uint256,string,address,uint256).tokenSalt_scope_6 (MADFactory721.
   ↪ sol#450) is a local variable never initialized
ERC721Lazy.withdraw().i (lib/tokens/ERC721/Impl/ERC721Lazy.sol#165) is a
   ↪  local variable never initialized
MADFactory721.createCollection(uint8,string,string,string,uint256,
   ↪ uint256,string,address,uint256).deployed_scope_7 (MADFactory721.
   ↪ sol#450) is a local variable never initialized
ERC721Lazy._userMint(uint256,address).j (lib/tokens/ERC721/Impl/
   ↪ ERC721Lazy.sol#336) is a local variable never initialized
SplitterImpl.releaseAll().i (lib/splitter/SplitterImpl.sol#132) is a
   ↪ local variable never initialized
ERC721Minimal.withdraw().j (lib/tokens/ERC721/Impl/ERC721Minimal.sol
   ↪ #104) is a local variable never initialized
ERC721Basic.withdraw().i (lib/tokens/ERC721/Impl/ERC721Basic.sol#164) is
   ↪  a local variable never initialized
ERC721Whitelist.withdrawERC20(ERC20).j (lib/tokens/ERC721/Impl/
   ↪ ERC721Whitelist.sol#338) is a local variable never initialized
ERC721Minimal.withdrawERC20(ERC20).j (lib/tokens/ERC721/Impl/
   ↪ ERC721Minimal.sol#131) is a local variable never initialized
ERC721Whitelist.withdrawERC20(ERC20).i (lib/tokens/ERC721/Impl/
   ↪ ERC721Whitelist.sol#327) is a local variable never initialized
ERC721Lazy.withdraw().j (lib/tokens/ERC721/Impl/ERC721Lazy.sol#175) is a
   ↪  local variable never initialized
MADFactory721.createCollection(uint8,string,string,string,uint256,
   ↪ uint256,string,address,uint256).tokenSalt_scope_3 (MADFactory721.
   ↪ sol#415) is a local variable never initialized
ERC721Lazy.withdrawERC20(ERC20).j (lib/tokens/ERC721/Impl/ERC721Lazy.sol
   ↪ #202) is a local variable never initialized
ERC721Basic.withdrawERC20(ERC20).i (lib/tokens/ERC721/Impl/ERC721Basic.
   ↪ sol#190) is a local variable never initialized
```

```
ERC721Whitelist.withdraw().i (lib/tokens/ERC721/Impl/ERC721Whitelist.sol
   ↪ #301) is a local variable never initialized
MADFactory721.createCollection(uint8,string,string,string,uint256,
   ↪ uint256,string,address,uint256).deployed_scope_4 (MADFactory721.
   ↪ sol#415) is a local variable never initialized
ERC721Basic.withdraw().j (lib/tokens/ERC721/Impl/ERC721Basic.sol#174) is
   ↪  a local variable never initialized
ERC721Lazy.lazyMint(Types.Voucher,uint8,bytes32,bytes32).i (lib/tokens/
   ↪ ERC721/Impl/ERC721Lazy.sol#105) is a local variable never
   ↪ initialized
ERC721Minimal.withdraw().i (lib/tokens/ERC721/Impl/ERC721Minimal.sol#94)
   ↪  is a local variable never initialized
MADFactory721.createCollection(uint8,string,string,string,uint256,
   ↪ uint256,string,address,uint256).tokenSalt_scope_0 (MADFactory721.
   ↪ sol#380) is a local variable never initialized
ERC721Whitelist.withdraw().j (lib/tokens/ERC721/Impl/ERC721Whitelist.sol
   ↪ #311) is a local variable never initialized
ERC721Basic.withdrawERC20(ERC20).j (lib/tokens/ERC721/Impl/ERC721Basic.
   ↪ sol#201) is a local variable never initialized
MADFactory721.createCollection(uint8,string,string,string,uint256,
   ↪ uint256,string,address,uint256).deployed_scope_1 (MADFactory721.
   ↪ sol#380) is a local variable never initialized
ERC721Minimal.withdrawERC20(ERC20).i (lib/tokens/ERC721/Impl/
   ↪ ERC721Minimal.sol#120) is a local variable never initialized
SplitterImpl.constructor(address[],uint256[]).i (lib/splitter/
   ↪ SplitterImpl.sol#70) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #uninitialized-local-variables


SplitterImpl.release(address) (lib/splitter/SplitterImpl.sol#107-127)
   ↪ uses a dangerous strict equality:
       - payment == 0 (lib/splitter/SplitterImpl.sol#112)
```

```
SplitterImpl.release(ERC20,address) (lib/splitter/SplitterImpl.sol
   ↪ #147-171) uses a dangerous strict equality:
       - payment == 0 (lib/splitter/SplitterImpl.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #dangerous-strict-equalities

Contract locking ether found:
       Contract SplitterImpl (lib/splitter/SplitterImpl.sol#27-297) has
          ↪ payable functions:
        - SplitterImpl.constructor(address[],uint256[]) (lib/splitter/
           ↪ SplitterImpl.sol#58-78)
        - SplitterImpl.receive() (lib/splitter/SplitterImpl.sol#92-98)
       But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #contracts-that-lock-ether

SplitterImpl.constructor(address[],uint256[]).i (lib/splitter/
   ↪ SplitterImpl.sol#70) is a local variable never initialized
SplitterImpl.releaseAll().i (lib/splitter/SplitterImpl.sol#132) is a
   ↪ local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #uninitialized-local-variables

MADFactory721.signer (MADFactory721.sol#93) is never initialized. It is
   ↪ used in:
       - MADFactory721.createCollection(uint8,string,string,string,
          ↪ uint256,uint256,string,address,uint256) (MADFactory721.sol
          ↪ #326-483)
       - MADFactory721.setSigner(address) (MADFactory721.sol#531-540)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #uninitialized-state-variables

SplitterImpl.release(address) (lib/splitter/SplitterImpl.sol#107-127)
   ↪ uses a dangerous strict equality:
```

```
        - payment == 0 (lib/splitter/SplitterImpl.sol#112)
SplitterImpl.release(ERC20,address) (lib/splitter/SplitterImpl.sol
    ↪ #147-171) uses a dangerous strict equality:
        - payment == 0 (lib/splitter/SplitterImpl.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dangerous-strict-equalities

Contract locking ether found:
        Contract SplitterImpl (lib/splitter/SplitterImpl.sol#27-297) has
            ↪ payable functions:
         - SplitterImpl.constructor(address[],uint256[]) (lib/splitter/
             ↪ SplitterImpl.sol#58-78)
         - SplitterImpl.receive() (lib/splitter/SplitterImpl.sol#92-98)
        But does not have a function to withdraw the ether
Contract locking ether found:
        Contract ERC721Basic (lib/tokens/ERC721/Impl/ERC721Basic.sol
            ↪ #18-332) has payable functions:
         - ERC721Basic.mintTo(address,uint256) (lib/tokens/ERC721/Impl/
             ↪ ERC721Basic.sol#110-134)
         - ERC721Basic.burn(uint256[]) (lib/tokens/ERC721/Impl/
             ↪ ERC721Basic.sol#136-157)
         - ERC721Basic.mint(uint256) (lib/tokens/ERC721/Impl/ERC721Basic.
             ↪ sol#218-243)
        But does not have a function to withdraw the ether
Contract locking ether found:
        Contract ERC721Lazy (lib/tokens/ERC721/Impl/ERC721Lazy.sol
            ↪ #18-433) has payable functions:
         - ERC721Lazy.lazyMint(Types.Voucher,uint8,bytes32,bytes32) (lib/
             ↪ tokens/ERC721/Impl/ERC721Lazy.sol#95-113)
Reentrancy in MADFactory721.createCollection(uint8,string,string,string,
    ↪ uint256,uint256,string,address,uint256) (MADFactory721.sol
    ↪ #326-483):
        External calls:
```

```
        - (tokenSalt,deployed) = ERC721MinimalDeployer._721MinimalDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_splitter,router,
            ↪ _royalty) (MADFactory721.sol#346-356)
        - (tokenSalt,deployed) = ERC721BasicDeployer._721BasicDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_price,_maxSupply,
            ↪ _splitter,router,_royalty) (MADFactory721.sol#380-391)
        - (tokenSalt,deployed) = ERC721WhitelistDeployer.
            ↪ _721WhitelistDeploy(_tokenSalt,_name,_symbol,_baseURI,
            ↪ _price,_maxSupply,_splitter,router,_royalty) (
            ↪ MADFactory721.sol#415-426)
        - (tokenSalt,deployed) = ERC721LazyDeployer._721LazyDeploy(
            ↪ _tokenSalt,_name,_symbol,_baseURI,_splitter,router,signer,
            ↪ _royalty) (MADFactory721.sol#450-460)
        State variables written after the call(s):
        - userTokens[tx.origin].push(colId_scope_8) (MADFactory721.sol
            ↪ #463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-1


supportsInterface(bytes4) should be declared external:
        - ERC721.supportsInterface(bytes4) (lib/tokens/ERC721/Base/ERC721
            ↪ .sol#199-209)
        - ERC721Basic.supportsInterface(bytes4) (lib/tokens/ERC721/Impl/
            ↪ ERC721Basic.sol#315-331)
        - ERC721Lazy.supportsInterface(bytes4) (lib/tokens/ERC721/Impl/
            ↪ ERC721Lazy.sol#416-432)
        - ERC721Minimal.supportsInterface(bytes4) (lib/tokens/ERC721/Impl
            ↪ /ERC721Minimal.sol#200-216)
        - ERC721Whitelist.supportsInterface(bytes4) (lib/tokens/ERC721/
            ↪ Impl/ERC721Whitelist.sol#512-528)
setSigner(address) should be declared external:
        - ERC721Lazy.setSigner(address) (lib/tokens/ERC721/Impl/
            ↪ ERC721Lazy.sol#120-124)
```

```
_verifyVoucher(Types.Voucher,uint8,bytes32,bytes32) should be declared
    ↪ external:
        - ERC721Lazy._verifyVoucher(Types.Voucher,uint8,bytes32,bytes32)
            ↪ (lib/tokens/ERC721/Impl/ERC721Lazy.sol#240-278)
royaltyInfo(uint256,uint256) should be declared external:
        - ERC2981.royaltyInfo(uint256,uint256) (lib/tokens/common/ERC2981
            ↪ .sol#15-23)
supportsInterface(bytes4) should be declared external:
        - ERC2981.supportsInterface(bytes4) (lib/tokens/common/ERC2981.
            ↪ sol#25-34)
        - ERC721Basic.supportsInterface(bytes4) (lib/tokens/ERC721/Impl/
            ↪ ERC721Basic.sol#315-331)
        - ERC721Lazy.supportsInterface(bytes4) (lib/tokens/ERC721/Impl/
            ↪ ERC721Lazy.sol#416-432)
        - ERC721Minimal.supportsInterface(bytes4) (lib/tokens/ERC721/Impl
            ↪ /ERC721Minimal.sol#200-216)
        - ERC721Whitelist.supportsInterface(bytes4) (lib/tokens/ERC721/
            ↪ Impl/ERC721Whitelist.sol#512-528)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external
MADFactory721.sol analyzed (41 contracts with 78 detectors), 352 result(
    ↪ s) found
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review.

7 Conclusion

In this audit, we examined the design and implementation of MADNFT 1.0 contract and discovered several issues of varying severity. Jacob Clay team addressed 7 issues raised in the initial report and implemented the necessary fixes, while classifying the rest as a risk with low-probability of occurrence. BlockHat' auditors advised Jacob Clay Team to maintain a high level of vigilance and to keep those findings in mind in order to avoid any future complications.

BLOCKHAT
SECURITY

For a Smart Contract Audit, contact us at contact@blockhat.io