MythX

Analysis 79f44aad-ce35-4c8c-9de9-669270850061 MythX

Started	Wed Jul 05 2023 05:29:39 GMT+0000 (Coordinated Universal Time)
Finished	Wed Jul 05 2023 07:48:16 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	ElephantNFT_flattened.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	23

ISSUES

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
57
58  // multiply by 4/3 rounded up
59  uint256 encodedLen = 4 * ((data.length + 2) / 3);
60
61  // add some extra buffer at the end required for the writing
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
57
58  // multiply by 4/3 rounded up
59  uint256 encodedLen = 4 * ((data.length + 2) / 3);
60
61  // add some extra buffer at the end required for the writing
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
57
58  // multiply by 4/3 rounded up
59  uint256 encodedLen = 4 * ((data.length + 2) / 3);
60
61  // add some extra buffer at the end required for the writing
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
60
61  // add some extra buffer at the end required for the writing
62  string memory result = new string(encodedLen + 32);
63
64  assembly {
```

UNKNOWN Arithmetic operation "%" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
107
108 if (data.length == 0) return new bytes(0);
109 require(data.length % 4 == 0, "invalid base64 decoder input");
110
111 // load the table into memory
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
113
114 // every 4 characters represent 3 bytes
115 uint256 decodedLen = (data.length / 4) * 3;
116
117 // add some extra buffer at the end required for the writing
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
113
114 // every 4 characters represent 3 bytes
115 uint256 decodedLen = (data.length / 4) * 3;
116
117 // add some extra buffer at the end required for the writing
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
116
117 // add some extra buffer at the end required for the writing
118 bytes memory result = new bytes(decodedLen + 32);
119
120 assembly {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
186 if (x >= 0x100000000000000000000000000000000) {
187 x >>= 128;
188 r += 128;
189 }
190 if (x >= 0x10000000000000000) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
190 if (x >= 0x10000000000000000) {
191 x >>= 64;
192 r += 64;
193 }
194 if (x >= 0x100000000) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
194 if (x >= 0x100000000) {
195 x >>= 32;
196 r += 32;
197 }
198 if (x >= 0x10000) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
198 if (x >= 0x10000) {
199 x >>= 16;
200 r += 16;
201 }
202 if (x >= 0x100) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
202 if (x >= 0x100) {
203 x >>= 8;
204 r += 8;
205 }
206 if (x >= 0x10) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
206 if (x >= 0x10) {
207 x >>= 4;
208 r += 4;
209 }
210 if (x >= 0x4) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
210 if (x >= 0x4) {
211 x >>= 2;
212 r += 2;
213 }
214 if (x >= 0x2) r += 1;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
212 r += 2;
213 }
214 if (x >= 0x2) r += 1;
215 }
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
226 r = 255;
227 if (x & type(uint128).max > 0) {
228 r -= 128;
229 } else {
230 x >>= 128;
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
231 }
232 if (x & type(uint64).max > 0) {
233 r -= 64;
234 } else {
235 x >>= 64;
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
236 }
237 if (x & type(uint32).max > 0) {
238 r -= 32;
239 } else {
240 x >>= 32;
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
241 }
242 if (x & type(uint16).max > 0) {
243 r -= 16;
244 } else {
245 x >>= 16;
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
246 }
247 if (x & type(uint8).max > 0) {
248 r -= 8;
249 } else {
250 x >>= 8;
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
251 }
252 if (x & 0xf > 0) {
253 r -= 4;
254 } else {
255 x >>= 4;
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
256 }
257 if (x & 0x3 > 0) {
258 r -= 2;
259 } else {
260 x >>= 2;
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
260 x >>= 2;
261 }
262 if (x & 0x1 > 0) r -= 1;
263 }
264 }
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
293 function increment(Counter storage counter) internal {
294 unchecked {
295 counter._value += 1;
296 }
297 }
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
301 require(value > 0, "Counter: decrement overflow");
302 unchecked {
303 counter._value = value - 1;
304 }
305 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
432 function average(int256 a, int256 b) internal pure returns (int256) {
433 // Formula from the book "Hacker's Delight"
434 int256 x = (a & b) + ((a ^ b) >> 1);
435 return x + (int256(uint256(x) >> 255) & (a ^ b));
436 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
433 // Formula from the book "Hacker's Delight"
434 int256 x = (a & b) + ((a ^ b) >> 1);
435 return x + (int256(uint256(x) >> 255) & (a ^ b));
436 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
484 function average(uint256 a, uint256 b) internal pure returns (uint256) {
485 // (a + b) / 2 can overflow.
486 return (a & b) + (a ^ b) / 2;
487 }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
484 function average(uint256 a, uint256 b) internal pure returns (uint256) {
485 // (a + b) / 2 can overflow.
486 return (a & b) + (a ^ b) / 2;
487 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
495 function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
496 // (a + b - 1) / b can overflow on addition, so we distribute.
497 return a == 0 ? 0 : (a - 1) / b + 1;
498 }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
495 function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
496 // (a + b - 1) / b can overflow on addition, so we distribute.
497 return a == 0 ? 0 : (a - 1) / b + 1;
498 }
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
495 function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
496 // (a + b - 1) / b can overflow on addition, so we distribute.
497 return a == 0 ? 0 : (a - 1) / b + 1;
498 }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
521 // The surrounding unchecked block does not change this fact.
522 // See https://docs.soliditylang.org/en/latest/control-structures.html#checked-or-unchecked-arithmetic.
523 return prod0 / denominator;
524 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
546
547 // Does not overflow because the denominator cannot be zero at this stage in the function.
548 uint256 twos = denominator & (~denominator + 1);
549 assembly {
550 // Divide denominator by twos.
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
559
560 // Shift in bits from prod1 into prod0.
561 prod0 |= prod1 * twos;
562
563 // Invert denominator mod 2^256. Now that denominator is an odd number, it has an inverse modulo 2^256 such
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
564 // that denominator * inv = 1 mod 2^256. Compute the inverse by starting with a seed that is correct for
565 // four bits. That is, denominator * inv = 1 mod 2^4.
566 uint256 inverse = (3 * denominator) ^ 2;
567
568 // Use the Newton-Raphson iteration to improve the precision. Thanks to Hensel's lifting lemma, this also works
```

UNKNOWN Arithmetic operation "*=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
568 // Use the Newton-Raphson iteration to improve the precision. Thanks to Hensel's lifting lemma, this also works
569 // in modular arithmetic, doubling the correct bits in each step.
570 inverse *= 2 - denominator * inverse; // inverse mod 2^8
571 inverse *= 2 - denominator * inverse; // inverse mod 2^16
572 inverse *= 2 - denominator * inverse; // inverse mod 2^32
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
568 // Use the Newton-Raphson iteration to improve the precision. Thanks to Hensel's lifting lemma, this also works
569 // in modular arithmetic, doubling the correct bits in each step.
570 inverse *= 2 - denominator * inverse; // inverse mod 2^8
571 inverse *= 2 - denominator * inverse; // inverse mod 2^16
572 inverse *= 2 - denominator * inverse; // inverse mod 2^32
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
568 // Use the Newton-Raphson iteration to improve the precision. Thanks to Hensel's lifting lemma, this also works
569 // in modular arithmetic, doubling the correct bits in each step.
570 inverse *= 2 - denominator * inverse; // inverse mod 2^8
571 inverse *= 2 - denominator * inverse; // inverse mod 2^16
572 inverse *= 2 - denominator * inverse; // inverse mod 2^32
```

UNKNOWN Arithmetic operation "*=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
569 // in modular arithmetic, doubling the correct bits in each step.
570 inverse *= 2 - denominator * inverse; // inverse mod 2^8
571 inverse *= 2 - denominator * inverse; // inverse mod 2^16
572 inverse *= 2 - denominator * inverse; // inverse mod 2^32
573 inverse *= 2 - denominator * inverse; // inverse mod 2^64
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
569 // in modular arithmetic, doubling the correct bits in each step.
570 inverse *= 2 - denominator * inverse; // inverse mod 2^8
571 inverse *= 2 - denominator * inverse; // inverse mod 2^16
572 inverse *= 2 - denominator * inverse; // inverse mod 2^32
573 inverse *= 2 - denominator * inverse; // inverse mod 2^64
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
569 // in modular arithmetic, doubling the correct bits in each step.
570 inverse *= 2 - denominator * inverse; // inverse mod 2^8
571 inverse *= 2 - denominator * inverse; // inverse mod 2^16
572 inverse *= 2 - denominator * inverse; // inverse mod 2^32
573 inverse *= 2 - denominator * inverse; // inverse mod 2^64
```

UNKNOWN Arithmetic operation "*=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
570 inverse *= 2 - denominator * inverse; // inverse mod 2^8
571 inverse *= 2 - denominator * inverse; // inverse mod 2^16
572 inverse *= 2 - denominator * inverse; // inverse mod 2^32
573 inverse *= 2 - denominator * inverse; // inverse mod 2^64
574 inverse *= 2 - denominator * inverse; // inverse mod 2^128
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
570 inverse *= 2 - denominator * inverse; // inverse mod 2^8
571 inverse *= 2 - denominator * inverse; // inverse mod 2^16
572 inverse *= 2 - denominator * inverse; // inverse mod 2^32
573 inverse *= 2 - denominator * inverse; // inverse mod 2^64
574 inverse *= 2 - denominator * inverse; // inverse mod 2^128
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
570 inverse *= 2 - denominator * inverse; // inverse mod 2^8
571 inverse *= 2 - denominator * inverse; // inverse mod 2^16
572 inverse *= 2 - denominator * inverse; // inverse mod 2^32
573 inverse *= 2 - denominator * inverse; // inverse mod 2^64
574 inverse *= 2 - denominator * inverse; // inverse mod 2^128
```

UNKNOWN Arithmetic operation "*=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
571 inverse *= 2 - denominator * inverse; // inverse mod 2^16
572 inverse *= 2 - denominator * inverse; // inverse mod 2^32
573 inverse *= 2 - denominator * inverse; // inverse mod 2^64
574 inverse *= 2 - denominator * inverse; // inverse mod 2^128
575 inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
571 inverse *= 2 - denominator * inverse; // inverse mod 2^16
572 inverse *= 2 - denominator * inverse; // inverse mod 2^32
573 inverse *= 2 - denominator * inverse; // inverse mod 2^64
574 inverse *= 2 - denominator * inverse; // inverse mod 2^128
575 inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
571 inverse *= 2 - denominator * inverse; // inverse mod 2^16
572 inverse *= 2 - denominator * inverse; // inverse mod 2^32
573 inverse *= 2 - denominator * inverse; // inverse mod 2^64
574 inverse *= 2 - denominator * inverse; // inverse mod 2^128
575 inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN Arithmetic operation "*=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
572 | inverse *= 2 - denominator * inverse; // inverse mod 2^32
573 | inverse *= 2 - denominator * inverse; // inverse mod 2^64
574 | inverse *= 2 - denominator * inverse; // inverse mod 2^128
575 | inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
572 | inverse *= 2 - denominator * inverse; // inverse mod 2^32
573 | inverse *= 2 - denominator * inverse; // inverse mod 2^64
574 | inverse *= 2 - denominator * inverse; // inverse mod 2^128
575 | inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
572 | inverse *= 2 - denominator * inverse; // inverse mod 2^32
573 | inverse *= 2 - denominator * inverse; // inverse mod 2^64
574 | inverse *= 2 - denominator * inverse; // inverse mod 2^128
575 | inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN Arithmetic operation "*=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
573 | inverse *= 2 - denominator * inverse; // inverse mod 2^64
574 | inverse *= 2 - denominator * inverse; // inverse mod 2^128
575 | inverse *= 2 - denominator * inverse; // inverse mod 2^256
576 |
577 | // Because the division is now exact we can divide by multiplying with the modular inverse of denominator.
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
573 inverse *= 2 - denominator * inverse; // inverse mod 2^64
574 inverse *= 2 - denominator * inverse; // inverse mod 2^128
575 inverse *= 2 - denominator * inverse; // inverse mod 2^256
576
577 // Because the division is now exact we can divide by multiplying with the modular inverse of denominator.
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
573 inverse *= 2 - denominator * inverse; // inverse mod 2^64
574 inverse *= 2 - denominator * inverse; // inverse mod 2^128
575 inverse *= 2 - denominator * inverse; // inverse mod 2^256
576
577 // Because the division is now exact we can divide by multiplying with the modular inverse of denominator.
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
579 // less than 2^256, this is the final result. We don't need to compute the high bits of the result and prod1
580 // is no longer required.
581 result = prod0 * inverse;
582 return result;
583 }
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
590 uint256 result = mulDiv(x, y, denominator);
591 if (rounding == Rounding.Up && mulmod(x, y, denominator) > 0) {
592 result += 1;
593 }
594 return result;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
622 // into the expected uint128 result.
623 unchecked {
624 result = (result + a / result) >> 1;
625 result = (result + a / result) >> 1;
626 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
622 // into the expected uint128 result.
623 unchecked {
624 result = (result + a / result) >> 1;
625 result = (result + a / result) >> 1;
626 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
623 unchecked {
624 result = (result + a / result) >> 1;
625 result = (result + a / result) >> 1;
626 result = (result + a / result) >> 1;
627 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
623 unchecked {
624 result = (result + a / result) >> 1;
625 result = (result + a / result) >> 1;
626 result = (result + a / result) >> 1;
627 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
624 result = (result + a / result) >> 1;
625 result = (result + a / result) >> 1;
626 result = (result + a / result) >> 1;
627 result = (result + a / result) >> 1;
628 result = (result + a / result) >> 1;
```

UNKNOWN

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
624 result = (result + a / result) >> 1;
625 result = (result + a / result) >> 1;
626 result = (result + a / result) >> 1;
627 result = (result + a / result) >> 1;
628 result = (result + a / result) >> 1;
```

UNKNOWN

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
625 result = (result + a / result) >> 1;
626 result = (result + a / result) >> 1;
627 result = (result + a / result) >> 1;
628 result = (result + a / result) >> 1;
629 result = (result + a / result) >> 1;
```

UNKNOWN

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
625 result = (result + a / result) >> 1;
626 result = (result + a / result) >> 1;
627 result = (result + a / result) >> 1;
628 result = (result + a / result) >> 1;
629 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
626 result = (result + a / result) >> 1;
627 result = (result + a / result) >> 1;
628 result = (result + a / result) >> 1;
629 result = (result + a / result) >> 1;
630 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
626 result = (result + a / result) >> 1;
627 result = (result + a / result) >> 1;
628 result = (result + a / result) >> 1;
629 result = (result + a / result) >> 1;
630 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
627 result = (result + a / result) >> 1;
628 result = (result + a / result) >> 1;
629 result = (result + a / result) >> 1;
630 result = (result + a / result) >> 1;
631 return min(result, a / result);
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
627 result = (result + a / result) >> 1;
628 result = (result + a / result) >> 1;
629 result = (result + a / result) >> 1;
630 result = (result + a / result) >> 1;
631 return min(result, a / result);
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
628 result = (result + a / result) >> 1;
629 result = (result + a / result) >> 1;
630 result = (result + a / result) >> 1;
631 return min(result, a / result);
632 }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
628 result = (result + a / result) >> 1;
629 result = (result + a / result) >> 1;
630 result = (result + a / result) >> 1;
631 return min(result, a / result);
632 }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
629 result = (result + a / result) >> 1;
630 result = (result + a / result) >> 1;
631 return min(result, a / result);
632 }
633 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
639 unchecked {
640 uint256 result = sqrt(a);
641 return result + (rounding == Rounding.Up && result * result < a ? 1 : 0);
642 }
643 }
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
639 unchecked {
640 uint256 result = sqrt(a);
641 return result + (rounding == Rounding.Up && result * result < a ? 1 : 0);
642 }
643 }
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
652 if (value >> 128 > 0) {
653 value >>= 128;
654 result += 128;
655 }
656 if (value >> 64 > 0) {
657 value >>= 64;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
656 if (value >> 64 > 0) {
657 value >>= 64;
658 result += 64;
659 }
660 if (value >> 32 > 0) {
661 value >>= 32;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
660 if (value >> 32 > 0) {
661 value >>= 32;
662 result += 32;
663 }
664 if (value >> 16 > 0) {
665 value >>= 16;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
664 if (value >> 16 > 0) {
665 value >>= 16;
666 result += 16;
667 }
668 if (value >> 8 > 0) {
669 value >>= 8;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
668 if (value >> 8 > 0) {
669 value >>= 8;
670 result += 8;
671 }
672 if (value >> 4 > 0) {
673 value >>= 4;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
672 if (value >> 4 > 0) {
673 value >>= 4;
674 result += 4;
675 }
676 if (value >> 2 > 0) {
677 value >>= 2;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
676 if (value >> 2 > 0) {
677 value >>= 2;
678 result += 2;
679 }
680 if (value >> 1 > 0) {
681 result += 1;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
679 }
680 if (value >> 1 > 0) {
681 result += 1;
682 }
683 }
684 return result;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
692 unchecked {
693 uint256 result = log2(value);
694 return result + (rounding == Rounding.Up && 1 << result < value ? 1 : 0);
695 }
696 }
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
703 uint256 result = 0;
704 unchecked {
705 if (value >= 10 ** 64) {
706 value /= 10 ** 64; value /= 10 ** 64;
707 result += 64;
708 }
```

UNKNOWN Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
704 unchecked {
705 if (value >= 10 ** 64) {
706 value /= 10 ** 64;
707 result += 64;
708 }
709 if (value >= 10 ** 32) {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
704 unchecked {
705 if (value >= 10 ** 64) {
706 value /= 10 ** 64;
707 result += 64;
708 }
709 if (value >= 10 ** 32) {
```

UNKNOWN

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
705 if (value >= 10 ** 64) {
706 value /= 10 ** 64;
707 result += 64;
708 }
709 if (value >= 10 ** 32) {
710 value /= 10 ** 32;
```

UNKNOWN

Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
707 result += 64;
708 }
709 if (value >= 10 ** 32) {
710 value /= 10 ** 32; value /= 10 ** 32;
711 result += 32;
712 }
```

UNKNOWN

Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
708 }
709 if (value >= 10 ** 32) {
710 value /= 10 ** 32;
711 result += 32;
712 }
713 if (value >= 10 ** 16) {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
708 }
709 if (value >= 10 ** 32) {
710 value /= 10 ** 32;
711 result += 32;
712 }
713 if (value >= 10 ** 16) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
709 if (value >= 10 ** 32) {
710 value /= 10 ** 32;
711 result += 32;
712 }
713 if (value >= 10 ** 16) {
714 value /= 10 ** 16;
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
711 result += 32;
712 }
713 if (value >= 10 ** 16) {
714 value /= 10 ** 16; value /= 10 ** 16;
715 result += 16;
716 }
```

UNKNOWN Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
712 }
713 if (value >= 10 ** 16) {
714 value /= 10 ** 16;
715 result += 16;
716 }
717 if (value >= 10 ** 8) {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
712 }
713 if (value >= 10 ** 16) {
714 value /= 10 ** 16;
715 result += 16;
716 }
717 if (value >= 10 ** 8) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
713 if (value >= 10 ** 16) {
714 value /= 10 ** 16;
715 result += 16;
716 }
717 if (value >= 10 ** 8) {
718 value /= 10 ** 8;
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
715 result += 16;
716 }
717 if (value >= 10 ** 8) {
718 value /= 10 ** 8; value /= 10 ** 8;
719 result += 8;
720 }
```

UNKNOWN Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
716 }
717 if (value >= 10 ** 8) {
718 value /= 10 ** 8;
719 result += 8;
720 }
721 if (value >= 10 ** 4) {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
716 }
717 if (value >= 10 ** 8) {
718 value /= 10 ** 8;
719 result += 8;
720 }
721 if (value >= 10 ** 4) {
```

UNKNOWN

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
717 if (value >= 10 ** 8) {
718 value /= 10 ** 8;
719 result += 8;
720 }
721 if (value >= 10 ** 4) {
722 value /= 10 ** 4;
```

UNKNOWN

Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
719 result += 8;
720 }
721 if (value >= 10 ** 4) {
722 value /= 10 ** 4; value /= 10 ** 4;
723 result += 4;
724 }
```

UNKNOWN

Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
720 }
721 if (value >= 10 ** 4) {
722 value /= 10 ** 4;
723 result += 4;
724 }
725 if (value >= 10 ** 2) {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
720 }
721 if (value >= 10 ** 4) {
722 value /= 10 ** 4;
723 result += 4;
724 }
725 if (value >= 10 ** 2) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
721 if (value >= 10 ** 4) {
722 value /= 10 ** 4;
723 result += 4;
724 }
725 if (value >= 10 ** 2) {
726 value /= 10 ** 2;
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
723 result += 4;
724 }
725 if (value >= 10 ** 2) {
726 value /= 10 ** 2; value /= 10 ** 2;
727 result += 2;
728 }
```

UNKNOWN Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
724 }
725 if (value >= 10 ** 2) {
726 value /= 10 ** 2;
727 result += 2;
728 }
729 if (value >= 10 ** 1) {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
724 }
725 if (value >= 10 ** 2) {
726 value /= 10 ** 2;
727 result += 2;
728 }
729 if (value >= 10 ** 1) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
725 if (value >= 10 ** 2) {
726 value /= 10 ** 2;
727 result += 2;
728 }
729 if (value >= 10 ** 1) {
730 result += 1;
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
727 result += 2;
728 }
729 if (value >= 10 ** 1) {
730 result += 1; result += 1;
731 }
732 }
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
728 }
729 if (value >= 10 ** 1) {
730 result += 1;
731 }
732 }
733 return result;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
741 unchecked {
742 uint256 result = log10(value);
743 return result + (rounding == Rounding.Up && 10 ** result < value ? 1 : 0);
744 }
745 }
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
741 unchecked {
742 uint256 result = log10(value);
743 return result + (rounding == Rounding.Up && 10 ** result < value ? 1 : 0);
744 }
745 }
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
756 if (value >> 128 > 0) {
757 value >>= 128;
758 result += 16;
759 }
760 if (value >> 64 > 0) {
761 value >>= 64;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
760 if (value >> 64 > 0) {
761 value >>= 64;
762 result += 8;
763 }
764 if (value >> 32 > 0) {
765 value >>= 32;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
764 if (value >> 32 > 0) {
765 value >>= 32;
766 result += 4;
767 }
768 if (value >> 16 > 0) {
769 value >>= 16;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
768 if (value >> 16 > 0) {
769 value >>= 16;
770 result += 2;
771 }
772 if (value >> 8 > 0) {
773 result += 1;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
771 }
772 if (value >> 8 > 0) {
773 result += 1;
774 }
775 }
776 return result;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
784 unchecked {
785 uint256 result = log256(value);
786 return result + (rounding == Rounding.Up && 1 << (result << 3) < value ? 1 : 0);
787 }
788 }
789 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
810 function toString(uint256 value) internal pure returns (string memory) {
811 unchecked {
812 uint256 length = Math.log10(value) + 1;
813 string memory buffer = new string(length);
814 uint256 ptr;
815 /// @solidity memory-safe-assembly
```

UNKNOWN Arithmetic operation "--" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
818 }
819 while (true) {
820 ptr--;
821 /// @solidity memory-safe-assembly
822 assembly {
823 mstore8(ptr, byte(mod(value, 10), _SYMBOLS))
```

UNKNOWN Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
823 mstore8(ptr, byte(mod(value, 10), _SYMBOLS))
824 }
825 value /= 10;
826 if (value == 0) break;
827 }
828 return buffer;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
842 function toHexString(uint256 value) internal pure returns (string memory) {
843 unchecked {
844 return toHexString(value, Math.log256(value) + 1);
845 }
846 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
850 */
851 function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
852 bytes memory buffer = new bytes(2 * length + 2);
853 buffer[0] = "0";
854 buffer[1] = "x";
855 for (uint256 i = 2 * length + 1; i > 1; --i) {
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
850 */
851 function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
852 bytes memory buffer = new bytes(2 * length + 2);
853 buffer[0] = "0";
854 buffer[1] = "x";
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
853 buffer[0] = "0";
854 buffer[1] = "x";
855 for (uint256 i = 2 * length + 1; i > 1; --i) {
856 buffer[i] = _SYMBOLS[value & 0xf];
857 value >>= 4;
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
853 buffer[0] = "0";
854 buffer[1] = "x";
855 for (uint256 i = 2 * length + 1; i > 1; --i) {
856 buffer[i] = _SYMBOLS[value & 0xf];
857 value >>= 4;
```

UNKNOWN Arithmetic operation "--" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
853 buffer[0] = "0";
854 buffer[1] = "x";
855 for (uint256 i = 2 * length + 1; i > 1; --i) {
856 buffer[i] = _SYMBOLS[value & 0xf]; buffer[i] = _SYMBOLS[value & 0xf];
857 value >>= 4;
858 }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
908 function isRare(uint256 tokenId, address collection) internal pure returns (bool) {
909 bytes32 h = keccak256(abi.encodePacked(tokenId, collection));
910 return uint256(h) < type(uint256).max / (1 + BitMath.mostSignificantBit(tokenId) * 2);
911 }
912
913 function generateSVGDefs(uint256 tokenId, address collection) private pure returns (string memory svg) {
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
908 function isRare(uint256 tokenId, address collection) internal pure returns (bool) {
909 bytes32 h = keccak256(abi.encodePacked(tokenId, collection));
910 return uint256(h) < type(uint256).max / (1 + BitMath.mostSignificantBit(tokenId) * 2);
911 }
912
913 function generateSVGDefs(uint256 tokenId, address collection) private pure returns (string memory svg) {
```

UNKNOWN

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
908 function isRare(uint256 tokenId, address collection) internal pure returns (bool) {
909 bytes32 h = keccak256(abi.encodePacked(tokenId, collection));
910 return uint256(h) < type(uint256).max / (1 + BitMath.mostSignificantBit(tokenId) * 2);
911 }
912
913 function generateSVGDefs(uint256 tokenId, address collection) private pure returns (string memory svg) {
```

UNKNOWN

Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
956 string[] memory dDefs = new string[](defs.length);
957
958 for (uint i = 0; i < defs.length; i++) {
959 dDefs[i] = defs[i]; dDefs[i] = defs[i];
960 }
```

UNKNOWN

Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1010
1011 for (uint i = 0; i < borderInfo.length; i++) {
1012 dBorderInfo[i] = borderInfo[i];
1013 }
```

UNKNOWN

Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1035
1036 for (uint i = 0; i < bodyText.length; i++) {
1037 dBodyText[i] = bodyText[i];
1038 }
1039
```

UNKNOWN

Arithmetic operation "%" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1050 uint256 hue = h % 360;
1051
1052 uint256 sat = 50 + (h % 50);
1053
1054 uint256 secondOffset = 5 + (h % 60);
```

UNKNOWN

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1050 uint256 hue = h % 360;
1051
1052 uint256 sat = 50 + (h % 50);
1053
1054 uint256 secondOffset = 5 + (h % 60);
1055
1056 uint256 lumin1 = h % 30;
```

UNKNOWN Arithmetic operation "%" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1052 uint256 sat = 50 + (h % 50);
1053
1054 uint256 secondOffset = 5 + (h % 60);
1055
1056 uint256 lumin1 = h % 30;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1052 uint256 sat = 50 + (h % 50);
1053
1054 uint256 secondOffset = 5 + (h % 60);
1055
1056 uint256 lumin1 = h % 30;
1057
1058 uint256 lumin2 = lumin1 * 2;
```

UNKNOWN Arithmetic operation "%" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1054 uint256 secondOffset = 5 + (h % 60);
1055
1056 uint256 lumin1 = h % 30;
1057
1058 uint256 lumin2 = lumin1 * 2;
```

UNKNOWN Arithmetic operation "%" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1056 uint256 lumin1 = h % 30;
1057
1058 uint256 lumin2 = lumin1 * 2;
1059
1060 if (isRare(tokenId, collection)) {
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1057
1058 uint256 lumin2 = lumin1 * 2;
1059
1060 if (isRare(tokenId, collection)) {
1061 svg = string(
1062 abi.encodePacked(
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1097 ) private pure returns (string memory svg) {
1098 string memory sequence = string(abi.encodePacked(tokenId.toString(), unicode' • ', round.toString()));
1099 uint256 str1length = bytes(sequence).length + 4;
1100 uint256 rectWidth = 7 * (str1length + 4);
1101 uint256 xLoc = 290 - 32 - rectWidth; //290 total width, 32 padding
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1098 string memory sequence = string(abi.encodePacked(tokenId.toString(), unicode' • ', round.toString()));
1099 uint256 str1length = bytes(sequence).length + 4;
1100 uint256 rectWidth = 7 * (str1length + 4);
1101 uint256 xLoc = 290 - 32 - rectWidth; //290 total width, 32 padding
1102
1103 svg = string(
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1098 string memory sequence = string(abi.encodePacked(tokenId.toString(), unicode' • ', round.toString()));
1099 uint256 str1length = bytes(sequence).length + 4;
1100 uint256 rectWidth = 7 * (str1length + 4);
1101 uint256 xLoc = 290 - 32 - rectWidth; //290 total width, 32 padding
1102
1103 svg = string(
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1099 uint256 str1length = bytes(sequence).length + 4;
1100 uint256 rectWidth = 7 * (str1length + 4);
1101 uint256 xLoc = 290 - 32 - rectWidth; //290 total width, 32 padding
1102
1103 svg = string(
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1099 uint256 str1length = bytes(sequence).length + 4;
1100 uint256 rectWidth = 7 * (str1length + 4);
1101 uint256 xLoc = 290 - 32 - rectWidth; //290 total width, 32 padding
1102
1103 svg = string(
```

UNKNOWN Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1122
1123 for (uint i = 0; i < segments.length; i++) {
1124 result = string(abi.encodePacked(result, segments[i]));
1125 }
1126 }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1817 }
1818
1819 uint256 royaltyAmount = (salePrice * royalty.royaltyFraction) / _feeDenominator();
1820
1821 return (royalty.receiver, royaltyAmount);
1822 }
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
1817 }
1818
1819 uint256 royaltyAmount = (salePrice * royalty.royaltyFraction) / _feeDenominator();
1820
1821 return (royalty.receiver, royaltyAmount);
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2348 // this ever happens. Might change if we allow batch minting.
2349 // The ERC fails to describe this case.
2350 _balances[to] += 1;
2351 }
2352
2353 _owners[tokenId] = to;
2354
2355 emit Transfer(address(0), to, tokenId);
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2383 // Cannot overflow, as that would require more tokens to be burned/transferred
2384 // out than the owner initially received through minting and transferring in.
2385 _balances[owner] -= 1;
2386 }
2387 delete _owners[tokenId];
2388
2389 emit Transfer(owner, address(0), tokenId);
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2421 // `_balances[to]` could overflow in the conditions described in `_mint`. That would require
2422 // all 2**256 token ids to be minted, which in practice is impossible.
2423 _balances[from] -= 1;
2424 _balances[to] += 1;
2425 }
2426 _owners[tokenId] = to;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2422 // all 2**256 token ids to be minted, which in practice is impossible.
2423 _balances[from] -= 1;
2424 _balances[to] += 1;
2425 }
2426 _owners[tokenId] = to;
2427
2428 emit Transfer(from, to, tokenId);
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2534 // solhint-disable-next-line func-name-mixedcase
2535 function __unsafe_increaseBalance(address account, uint256 amount) internal {
2536 _balances[account] += amount;
2537 }
2538 }
2539
2540 // File: @openzeppelin/contracts/token/ERC721/extensions/ERC721Royalty.sol
2541
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2697 // then delete the last slot (swap and pop).
2698
2699 uint256 lastTokenIndex = ERC721.balanceOf(from) - 1;
2700 uint256 tokenIndex = _ownedTokensIndex[tokenId];
2701
2702 // When the token to delete is the last token, the swap operation is unnecessary
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2722 // then delete the last slot (swap and pop).
2723
2724 uint256 lastTokenIndex = _allTokens.length - 1;
2725 uint256 tokenIndex = _allTokensIndex[tokenId];
2726
2727 // When the token to delete is the last token, the swap operation is unnecessary. However, since this occurs so
```

UNKNOWN Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2785 _tokenIdCounter.increment(); //next
2786 current++;
2787 tokenId = _tokenIdCounter.current();
2788 }
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2791
2792  function price() external view returns (uint256) {
2793  return BASE_PRICE * (2 ** (totalSupply() / ROUND_SIZE));
2794  }
2795
2796  function round() external view returns (uint256) {
2797  return (totalSupply() / ROUND_SIZE) + 1;
2798  }
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2791
2792  function price() external view returns (uint256) {
2793  return BASE_PRICE * (2 ** (totalSupply() / ROUND_SIZE));
2794  }
2795
2796  function round() external view returns (uint256) {
2797  return (totalSupply() / ROUND_SIZE) + 1;
2798  }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2791
2792  function price() external view returns (uint256) {
2793  return BASE_PRICE * (2 ** (totalSupply() / ROUND_SIZE));
2794  }
2795
2796  function round() external view returns (uint256) {
2797  return (totalSupply() / ROUND_SIZE) + 1;
2798  }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2795
2796  function round() external view returns (uint256) {
2797  return (totalSupply() / ROUND_SIZE) + 1;
2798  }
2799
2800  function roundOf(uint256 tokenId) public pure returns (uint256) {
2801  require(tokenId > 0, "tokenId must be non-zero");
2802  return ((tokenId - 1) / ROUND_SIZE) + 1;
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2795
2796  function round() external view returns (uint256) {
2797  return (totalSupply() / ROUND_SIZE) + 1;
2798  }
2799
2800  function roundOf(uint256 tokenId) public pure returns (uint256) { function roundOf(uint256 tokenId) public pure returns (uint256) {
2801  require(tokenId > 0, "tokenId must be non-zero");
2802  return ((tokenId - 1) / ROUND_SIZE) + 1;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2800  function roundOf(uint256 tokenId) public pure returns (uint256) {
2801  require(tokenId > 0, "tokenId must be non-zero");
2802  return ((tokenId - 1) / ROUND_SIZE) + 1;
2803  }
2804
2805
2806  function safeMint(address to) public onlyRole(MINTER_ROLE) {
2807  uint256 tokenId = _tokenIdCounter.current();
2808  _tokenIdCounter.increment();
```

UNKNOWN

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2800 function roundOf(uint256 tokenId) public pure returns (uint256) {
2801 require(tokenId > 0, "tokenId must be non-zero");
2802 return ((tokenId - 1) / ROUND_SIZE) + 1;
2803 }
2804
2805
2806 function safeMint(address to) public onlyRole(MINTER_ROLE) {
```

UNKNOWN

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2800 function roundOf(uint256 tokenId) public pure returns (uint256) {
2801 require(tokenId > 0, "tokenId must be non-zero");
2802 return ((tokenId - 1) / ROUND_SIZE) + 1;
2803 }
```

UNKNOWN

Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2826 tokenIds = new uint[](balance);
2827 for (uint i=0; i < balance; i++) {
2828 tokenIds[i] = tokenOfOwnerByIndex(owner, i);
2829 }
2830 }
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
301 require(value > 0, "Counter: decrement overflow");
302 unchecked {
303 counter._value = value - 1;
304 }
305 }
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
495 function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
496 // (a + b - 1) / b can overflow on addition, so we distribute.
497 return a == 0 ? 0 : (a - 1) / b + 1;
498 }
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

ElephantNFT_flattened.sol

Locations

```
2697 // then delete the last slot (swap and pop).
2698
2699 uint256 lastTokenIndex = ERC721.balanceOf(from) - 1;
2700 uint256 tokenIndex = _ownedTokensIndex[tokenId];
2701
2702 // When the token to delete is the last token, the swap operation is unnecessary
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

ElephantNFT_flattened.sol

Locations

```
2722 // then delete the last slot (swap and pop).
2723
2724 uint256 lastTokenIndex = _allTokens.length - 1;
2725 uint256 tokenIndex = _allTokensIndex[tokenId];
2726
2727 // When the token to delete is the last token, the swap operation is unnecessary. However, since this occurs so
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

ElephantNFT_flattened.sol

Locations

```
2800 function roundOf(uint256 tokenId) public pure returns (uint256) {
2801 require(tokenId > 0, "tokenId must be non-zero");
2802 return ((tokenId - 1) / ROUND_SIZE) + 1;
2803 }
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
38
39
40 pragma solidity ^0.6.0;
41
42 /// @title Base64
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
170
171
172 pragma solidity >=0.5.0;
173
174 /// @title BitMath
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
269 // OpenZeppelin Contracts v4.4.1 (utils/Counters.sol)
270
271 pragma solidity ^0.8.0;
272
273 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
315 // OpenZeppelin Contracts v4.4.1 (access/IAccessControl.sol)
316
317 pragma solidity ^0.8.0;
318
319 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
406 // OpenZeppelin Contracts (last updated v4.8.0) (utils/math/SignedMath.sol)
407
408 pragma solidity ^0.8.0;
409
410 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
452 // OpenZeppelin Contracts (last updated v4.9.0) (utils/math/Math.sol)
453
454 pragma solidity ^0.8.0;
455
456 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
794 // OpenZeppelin Contracts (last updated v4.9.0) (utils/Strings.sol)
795
796 pragma solidity ^0.8.0;
797
798
799
800 /**/**
801  * @dev String operations.
802  */
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.9"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
880
881 //pragma solidity >=0.7.6;
882 pragma solidity ^0.8.9;
883
884
885
886
887 /// @title NFTSVG
888 /// @notice Provides a function for generating an SVG associated with a Uniswap NFT
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
1132 // OpenZeppelin Contracts v4.4.1 (utils/Context.sol)
1133
1134 pragma solidity ^0.8.0;
1135
1136 /**
1137 * @dev Provides information about the current execution context, including the
1138 * sender of the transaction and its data. While these are generally available
1139 * via msg.sender and msg.data, they should not be accessed in such a direct
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.1"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
1159 // OpenZeppelin Contracts (last updated v4.9.0) (utils/Address.sol)
1160
1161 pragma solidity ^0.8.1;
1162
1163 /**
1164 * @dev Collection of functions related to the address type
1165 */
1166 library Address {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
1406 // OpenZeppelin Contracts (last updated v4.6.0) (token/ERC721/IERC721Receiver.sol)
1407
1408 pragma solidity ^0.8.0;
1409
1410 /**
1411 * @title ERC721 token receiver interface
1412 * @dev Interface for any contract that wants to support safeTransfers
1413 * from ERC721 asset contracts.
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
1436 // OpenZeppelin Contracts v4.4.1 (utils/introspection/IERC165.sol)
1437
1438 pragma solidity ^0.8.0;
1439
1440 /**
1441 * @dev Interface of the ERC165 standard, as defined in the
1442 * https://eips.ethereum.org/EIPS/eip-165[EIP].
1443 *
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
1464 // OpenZeppelin Contracts (last updated v4.9.0) (interfaces/IERC2981.sol)
1465
1466 pragma solidity ^0.8.0;
1467
1468
1469 /**
1470 * @dev Interface for the NFT Royalty Standard.
1471 *
1472 * A standardized way to retrieve royalty payment information for non-fungible tokens (NFTs) to enable universal
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
1491 // OpenZeppelin Contracts v4.4.1 (utils/introspection/ERC165.sol)
1492
1493 pragma solidity ^0.8.0;
1494
1495
1496 /**
1497 * @dev Implementation of the {IERC165} interface.
1498 *
1499 * Contracts that want to implement ERC165 should inherit from this contract and override {supportsInterface} to check
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
1522 // OpenZeppelin Contracts (last updated v4.9.0) (access/AccessControl.sol)
1523
1524 pragma solidity ^0.8.0;
1525
1526
1527
1528
1529
1530 /**
1531 * @dev Contract module that allows children to implement role-based access
1532 * control mechanisms. This is a lightweight version that doesn't allow enumerating role
1533 * members except through off-chain means by accessing the contract event logs. Some
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
1772 // OpenZeppelin Contracts (last updated v4.9.0) (token/common/ERC2981.sol)
1773
1774 pragma solidity ^0.8.0;
1775
1776
1777
1778 /**
1779 * @dev Implementation of the NFT Royalty Standard, a standardized way to retrieve royalty payment information.
1780 *
1781 * Royalty information can be specified globally for all token ids via {_setDefaultRoyalty}, and/or individually for
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
1881 // OpenZeppelin Contracts (last updated v4.9.0) (token/ERC721/IERC721.sol)
1882
1883 pragma solidity ^0.8.0;
1884
1885
1886 /**
1887 * @dev Required interface of an ERC721 compliant contract.
1888 */
1889 interface IERC721 is IERC165 {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
2015 // OpenZeppelin Contracts (last updated v4.5.0) (token/ERC721/extensions/IERC721Enumerable.sol)
2016
2017 pragma solidity ^0.8.0;
2018
2019
2020 /**
2021 * @title ERC-721 Non-Fungible Token Standard, optional enumeration extension
2022 * @dev See https://eips.ethereum.org/EIPS/eip-721
2023 */
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
2046 // OpenZeppelin Contracts v4.4.1 (token/ERC721/extensions/IERC721Metadata.sol)
2047
2048 pragma solidity ^0.8.0;
2049
2050
2051 /**
2052 * @title ERC-721 Non-Fungible Token Standard, optional metadata extension
2053 * @dev See https://eips.ethereum.org/EIPS/eip-721
2054 */
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
2075 // OpenZeppelin Contracts (last updated v4.9.0) (token/ERC721/ERC721.sol)
2076
2077 pragma solidity ^0.8.0;
2078
2079
2080
2081
2082
2083
2084
2085
2086 /**
2087 * @dev Implementation of https://eips.ethereum.org/EIPS/eip-721[ERC721] Non-Fungible Token Standard, including
2088 * the Metadata extension, but not including the Enumerable extension, which is available separately as
2089 * {ERC721Enumerable}.
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
2543 // OpenZeppelin Contracts (last updated v4.8.0) (token/ERC721/extensions/ERC721Royalty.sol)
2544
2545 pragma solidity ^0.8.0;
2546
2547
2548
2549
2550 /**
2551 * @dev Extension of ERC721 with the ERC2981 NFT Royalty Standard, a standardized way to retrieve royalty payment
2552 * information.
2553 *
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
2583 // OpenZeppelin Contracts (last updated v4.8.0) (token/ERC721/extensions/ERC721Enumerable.sol)
2584
2585 pragma solidity ^0.8.0;
2586
2587
2588
2589 /**
2590 * @dev This implements an optional extension of {ERC721} defined in the EIP that adds
2591 * enumerability of all the token ids in the contract as well as all token ids owned by each
2592 * account.
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.9"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

ElephantNFT_flattened.sol

Locations

```
2742
2743
2744 pragma solidity ^0.8.9;
2745
2746
2747
2748
2749
2750
2751
2752
2753 /// @custom:security-contact contact@elephant.money
2754 contract ElephantMoneyUnlimitedNFT is ERC721Royalty, ERC721Enumerable, AccessControl {
2755 using Counters for Counters.Counter;
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

ElephantNFT_flattened.sol

Locations

```
851 function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
852 bytes memory buffer = new bytes(2 * length + 2);
853 buffer[0] = "0";
854 buffer[1] = "x";
855 for (uint256 i = 2 * length + 1; i > 1; --i) {
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

ElephantNFT_flattened.sol

Locations

```
852 bytes memory buffer = new bytes(2 * length + 2);
853 buffer[0] = "0";
854 buffer[1] = "x";
855 for (uint256 i = 2 * length + 1; i > 1; --i) {
856 buffer[i] = _SYMBOLS[value & 0xf];
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
854 buffer[1] = "x";
855 for (uint256 i = 2 * length + 1; i > 1; --i) {
856 buffer[i] = _SYMBOLS[value & 0xf];
857 value >>= 4;
858 }
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
854 buffer[1] = "x";
855 for (uint256 i = 2 * length + 1; i > 1; --i) {
856 buffer[i] = _SYMBOLS[value & 0xf];
857 value >>= 4;
858 }
859 require(value == 0, "Strings: hex length insufficient");
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
957
958 for (uint i = 0; i < defs.length; i++) {
959 dDefs[i] = defs[i];
960 }
961
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
957
958 for (uint i = 0; i < defs.length; i++) {
959 dDefs[i] = defs[i];
960 }
961
962 svg = packStrings(dDefs);
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
1010
1011 for (uint i = 0; i < borderInfo.length; i++) {
1012 dBorderInfo[i] = borderInfo[i];
1013 }
1014
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
1010
1011 for (uint i = 0; i < borderInfo.length; i++) {
1012 dBorderInfo[i] = borderInfo[i];
1013 }
1014
1015 svg = packStrings(dBorderInfo);
1016 }
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
1035
1036  for (uint i = 0; i < bodyText.length; i++) {
1037  dBodyText[i] = bodyText[i];
1038  }
1039
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
1035
1036  for (uint i = 0; i < bodyText.length; i++) {
1037  dBodyText[i] = bodyText[i];
1038  }
1039
1040  svg = packStrings(dBodyText);
1041
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
1122
1123  for (uint i = 0; i < segments.length; i++) {
1124  result = string(abi.encodePacked(result, segments[i]));
1125  }
1126  } }
1127
1128  }
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
2632 function tokenByIndex(uint256 index) public view virtual override returns (uint256) {
2633 require(index < ERC721Enumerable.totalSupply(), "ERC721Enumerable: global index out of bounds");
2634 return _allTokens[index];
2635 }
2636
2637 /**
2638 * @dev See {ERC721-_beforeTokenTransfer}.
2639 */
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
2728 // rarely (when the last minted token is burnt) that we still do the swap here to avoid the gas cost of adding
2729 // an 'if' statement (like in _removeTokenFromOwnerEnumeration)
2730 uint256 lastTokenId = _allTokens[lastTokenIndex];
2731
2732 _allTokens[tokenIndex] = lastTokenId; // Move the last token to the slot of the to-delete token
2733 _allTokensIndex[lastTokenId] = tokenIndex; // Update the moved token's index
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
2730 uint256 lastTokenId = _allTokens[lastTokenIndex];
2731
2732 _allTokens[tokenIndex] = lastTokenId; // Move the last token to the slot of the to-delete token
2733 _allTokensIndex[lastTokenId] = tokenIndex; // Update the moved token's index
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

ElephantNFT_flattened.sol

Locations

```
2826 tokenIds = new uint[](balance);
2827 for (uint i=0; i < balance; i++) {
2828 tokenIds[i] = tokenOfOwnerByIndex(owner, i);
2829 }
2830 }
```