MythX

Analysis 788e2b98-c7c6-4d66-a35b-f07ea3394827 MythX

Started	Mon Feb 26 2024 22:50:55 GMT+0000 (Coordinated Universal Time)
Finished	Mon Feb 26 2024 22:51:00 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Vscode-Extension
Main Source File	/Hx_fees/Exampleerc20hxfees.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	27

ISSUES

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
589 function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
590 address owner = _msgSender();
591 _approve(owner, spender, allowance(owner, spender) + addedValue);
592 return true;
593 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
612 require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");
613 unchecked {
614 _approve(owner, spender, currentAllowance - subtractedValue);
615 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
641 require(fromBalance >= amount, "ERC20: transfer amount exceeds balance");
642 unchecked {
643 _balances[from] = fromBalance - amount;
644 // Overflow not possible: the sum of all balances is capped by totalSupply, and the sum is preserved by
645 // decrementing then incrementing.
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
644 // Overflow not possible: the sum of all balances is capped by totalSupply, and the sum is preserved by
645 // decrementing then incrementing.
646 _balances[to] += amount;
647 }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
666 _beforeTokenTransfer(address(0), account, amount);
667
668 _totalSupply += amount;
669 unchecked {
670 // Overflow not possible: balance + amount is at most totalSupply + amount, which is checked above.
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
669 unchecked {
670 // Overflow not possible: balance + amount is at most totalSupply + amount, which is checked above.
671 _balances[account] += amount;
672 }
673 emit Transfer(address(0), account, amount);
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
695 require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
696 unchecked {
697 _balances[account] = accountBalance - amount;
698 // Overflow not possible: amount <= accountBalance <= totalSupply.
699 _totalSupply -= amount;
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
697 _balances[account] = accountBalance - amount;
698 // Overflow not possible: amount <= accountBalance <= totalSupply.
699 _totalSupply -= amount;
700 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
739 require(currentAllowance >= amount, "ERC20: insufficient allowance");
740 unchecked {
741 _approve(owner, spender, currentAllowance - amount);
742 }
743 }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
942 function increment(Counter storage counter) internal {
943 unchecked {
944 counter._value += 1;
945 }
946 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
950 require(value > 0, "Counter: decrement overflow");
951 unchecked {
952 counter._value = value - 1;
953 }
954 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
997  function average(uint256 a, uint256 b) internal pure returns (uint256) {
998  // (a + b) / 2 can overflow.
999  return (a & b) + (a ^ b) / 2;
1000 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
997  function average(uint256 a, uint256 b) internal pure returns (uint256) {
998  // (a + b) / 2 can overflow.
999  return (a & b) + (a ^ b) / 2;
1000 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1008 function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
1009 // (a + b - 1) / b can overflow on addition, so we distribute.
1010 return a == 0 ? 0 : (a - 1) / b + 1;
1011 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1008 function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
1009 // (a + b - 1) / b can overflow on addition, so we distribute.
1010 return a == 0 ? 0 : (a - 1) / b + 1;
1011 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1008 function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
1009 // (a + b - 1) / b can overflow on addition, so we distribute.
1010 return a == 0 ? 0 : (a - 1) / b + 1;
1011 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1034 // The surrounding unchecked block does not change this fact.
1035 // See https://docs.soliditylang.org/en/latest/control-structures.html#checked-or-unchecked-arithmetic.
1036 return prod0 / denominator;
1037 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1059
1060 // Does not overflow because the denominator cannot be zero at this stage in the function.
1061 uint256 twos = denominator & (~denominator + 1);
1062 assembly {
1063 // Divide denominator by twos.
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1072
1073 // Shift in bits from prod1 into prod0.
1074 prod0 |= prod1 * twos;
1075
1076 // 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1077 // that denominator * inv = 1 mod 2^256. Compute the inverse by starting with a seed that is correct for
1078 // four bits. That is, denominator * inv = 1 mod 2^4.
1079 uint256 inverse = (3 * denominator) ^ 2;
1080
1081 // 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1081 // Use the Newton-Raphson iteration to improve the precision. Thanks to Hensel's lifting lemma, this also works
1082 // in modular arithmetic, doubling the correct bits in each step.
1083 inverse *= 2 - denominator * inverse; // inverse mod 2^8
1084 inverse *= 2 - denominator * inverse; // inverse mod 2^16
1085 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1081 // Use the Newton-Raphson iteration to improve the precision. Thanks to Hensel's lifting lemma, this also works
1082 // in modular arithmetic, doubling the correct bits in each step.
1083 inverse *= 2 - denominator * inverse; // inverse mod 2^8
1084 inverse *= 2 - denominator * inverse; // inverse mod 2^16
1085 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1081 // Use the Newton-Raphson iteration to improve the precision. Thanks to Hensel's lifting lemma, this also works
1082 // in modular arithmetic, doubling the correct bits in each step.
1083 inverse *= 2 - denominator * inverse; // inverse mod 2^8
1084 inverse *= 2 - denominator * inverse; // inverse mod 2^16
1085 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1082 // in modular arithmetic, doubling the correct bits in each step.
1083 inverse *= 2 - denominator * inverse; // inverse mod 2^8
1084 inverse *= 2 - denominator * inverse; // inverse mod 2^16
1085 inverse *= 2 - denominator * inverse; // inverse mod 2^32
1086 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1082 // in modular arithmetic, doubling the correct bits in each step.
1083 inverse *= 2 - denominator * inverse; // inverse mod 2^8
1084 inverse *= 2 - denominator * inverse; // inverse mod 2^16
1085 inverse *= 2 - denominator * inverse; // inverse mod 2^32
1086 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1082 // in modular arithmetic, doubling the correct bits in each step.
1083 inverse *= 2 - denominator * inverse; // inverse mod 2^8
1084 inverse *= 2 - denominator * inverse; // inverse mod 2^16
1085 inverse *= 2 - denominator * inverse; // inverse mod 2^32
1086 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1083 inverse *= 2 - denominator * inverse; // inverse mod 2^8
1084 inverse *= 2 - denominator * inverse; // inverse mod 2^16
1085 inverse *= 2 - denominator * inverse; // inverse mod 2^32
1086 inverse *= 2 - denominator * inverse; // inverse mod 2^64
1087 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1083 inverse *= 2 - denominator * inverse; // inverse mod 2^8
1084 inverse *= 2 - denominator * inverse; // inverse mod 2^16
1085 inverse *= 2 - denominator * inverse; // inverse mod 2^32
1086 inverse *= 2 - denominator * inverse; // inverse mod 2^64
1087 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1083 inverse *= 2 - denominator * inverse; // inverse mod 2^8
1084 inverse *= 2 - denominator * inverse; // inverse mod 2^16
1085 inverse *= 2 - denominator * inverse; // inverse mod 2^32
1086 inverse *= 2 - denominator * inverse; // inverse mod 2^64
1087 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1084 inverse *= 2 - denominator * inverse; // inverse mod 2^16
1085 inverse *= 2 - denominator * inverse; // inverse mod 2^32
1086 inverse *= 2 - denominator * inverse; // inverse mod 2^64
1087 inverse *= 2 - denominator * inverse; // inverse mod 2^128
1088 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1084 inverse *= 2 - denominator * inverse; // inverse mod 2^16
1085 inverse *= 2 - denominator * inverse; // inverse mod 2^32
1086 inverse *= 2 - denominator * inverse; // inverse mod 2^64
1087 inverse *= 2 - denominator * inverse; // inverse mod 2^128
1088 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1084 inverse *= 2 - denominator * inverse; // inverse mod 2^16
1085 inverse *= 2 - denominator * inverse; // inverse mod 2^32
1086 inverse *= 2 - denominator * inverse; // inverse mod 2^64
1087 inverse *= 2 - denominator * inverse; // inverse mod 2^128
1088 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1085 inverse *= 2 - denominator * inverse; // inverse mod 2^32
1086 inverse *= 2 - denominator * inverse; // inverse mod 2^64
1087 inverse *= 2 - denominator * inverse; // inverse mod 2^128
1088 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1085 inverse *= 2 - denominator * inverse; // inverse mod 2^32
1086 inverse *= 2 - denominator * inverse; // inverse mod 2^64
1087 inverse *= 2 - denominator * inverse; // inverse mod 2^128
1088 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1085 inverse *= 2 - denominator * inverse; // inverse mod 2^32
1086 inverse *= 2 - denominator * inverse; // inverse mod 2^64
1087 inverse *= 2 - denominator * inverse; // inverse mod 2^128
1088 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1086 inverse *= 2 - denominator * inverse; // inverse mod 2^64
1087 inverse *= 2 - denominator * inverse; // inverse mod 2^128
1088 inverse *= 2 - denominator * inverse; // inverse mod 2^256
1089
1090 // 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1086 inverse *= 2 - denominator * inverse; // inverse mod 2^64
1087 inverse *= 2 - denominator * inverse; // inverse mod 2^128
1088 inverse *= 2 - denominator * inverse; // inverse mod 2^256
1089
1090 // 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1086 inverse *= 2 - denominator * inverse; // inverse mod 2^64
1087 inverse *= 2 - denominator * inverse; // inverse mod 2^128
1088 inverse *= 2 - denominator * inverse; // inverse mod 2^256
1089
1090 // 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1092 // less than 2^256, this is the final result. We don't need to compute the high bits of the result and prod1
1093 // is no longer required.
1094 result = prod0 * inverse;
1095 return result;
1096 }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1103 uint256 result = mulDiv(x, y, denominator);
1104 if (rounding == Rounding.Up && mulmod(x, y, denominator) > 0) {
1105 result += 1;
1106 }
1107 return result;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1135 // into the expected uint128 result.
1136 unchecked {
1137 result = (result + a / result) >> 1;
1138 result = (result + a / result) >> 1;
1139 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1135 // into the expected uint128 result.
1136 unchecked {
1137 result = (result + a / result) >> 1;
1138 result = (result + a / result) >> 1;
1139 result = (result + a / result) >> 1;
```

UNKNOWN

Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1136 unchecked {
1137 result = (result + a / result) >> 1;
1138 result = (result + a / result) >> 1;
1139 result = (result + a / result) >> 1;
1140 result = (result + a / result) >> 1;
```

UNKNOWN

Arithmetic operation "/" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1136 unchecked {
1137 result = (result + a / result) >> 1;
1138 result = (result + a / result) >> 1;
1139 result = (result + a / result) >> 1;
1140 result = (result + a / result) >> 1;
```

UNKNOWN

Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1137 result = (result + a / result) >> 1;
1138 result = (result + a / result) >> 1;
1139 result = (result + a / result) >> 1;
1140 result = (result + a / result) >> 1;
1141 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1137 result = (result + a / result) >> 1;
1138 result = (result + a / result) >> 1;
1139 result = (result + a / result) >> 1;
1140 result = (result + a / result) >> 1;
1141 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1138 result = (result + a / result) >> 1;
1139 result = (result + a / result) >> 1;
1140 result = (result + a / result) >> 1;
1141 result = (result + a / result) >> 1;
1142 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1138 result = (result + a / result) >> 1;
1139 result = (result + a / result) >> 1;
1140 result = (result + a / result) >> 1;
1141 result = (result + a / result) >> 1;
1142 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1139 result = (result + a / result) >> 1;
1140 result = (result + a / result) >> 1;
1141 result = (result + a / result) >> 1;
1142 result = (result + a / result) >> 1;
1143 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1139 result = (result + a / result) >> 1;
1140 result = (result + a / result) >> 1;
1141 result = (result + a / result) >> 1;
1142 result = (result + a / result) >> 1;
1143 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1140 result = (result + a / result) >> 1;
1141 result = (result + a / result) >> 1;
1142 result = (result + a / result) >> 1;
1143 result = (result + a / result) >> 1;
1144 return min(result, a / result);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1140 result = (result + a / result) >> 1;
1141 result = (result + a / result) >> 1;
1142 result = (result + a / result) >> 1;
1143 result = (result + a / result) >> 1;
1144 return min(result, a / result);
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1141 result = (result + a / result) >> 1;
1142 result = (result + a / result) >> 1;
1143 result = (result + a / result) >> 1;
1144 return min(result, a / result);
1145 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1141 result = (result + a / result) >> 1;
1142 result = (result + a / result) >> 1;
1143 result = (result + a / result) >> 1;
1144 return min(result, a / result);
1145 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1142 result = (result + a / result) >> 1;
1143 result = (result + a / result) >> 1;
1144 return min(result, a / result);
1145 }
1146 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1152 unchecked {
1153 uint256 result = sqrt(a);
1154 return result + (rounding == Rounding.Up && result * result < a ? 1 : 0);
1155 }
1156 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1152 unchecked {
1153 uint256 result = sqrt(a);
1154 return result + (rounding == Rounding.Up && result * result < a ? 1 : 0);
1155 }
1156 }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1165 if (value >> 128 > 0) {
1166 value >>= 128;
1167 result += 128;
1168 }
1169 if (value >> 64 > 0) {
1170 value >>= 64;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1169 if (value >> 64 > 0) {
1170 value >>= 64;
1171 result += 64;
1172 }
1173 if (value >> 32 > 0) {
1174 value >>= 32;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1173 if (value >> 32 > 0) {
1174 value >>= 32;
1175 result += 32;
1176 }
1177 if (value >> 16 > 0) {
1178 value >>= 16;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1177 if (value >> 16 > 0) {
1178 value >>= 16;
1179 result += 16;
1180 }
1181 if (value >> 8 > 0) {
1182 value >>= 8;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1181 if (value >> 8 > 0) {
1182 value >>= 8;
1183 result += 8;
1184 }
1185 if (value >> 4 > 0) {
1186 value >>= 4;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1185 if (value >> 4 > 0) {
1186 value >>= 4;
1187 result += 4;
1188 }
1189 if (value >> 2 > 0) {
1190 value >>= 2;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1189 if (value >> 2 > 0) {
1190 value >>= 2;
1191 result += 2;
1192 }
1193 if (value >> 1 > 0) {
1194 result += 1;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1192 }
1193 if (value >> 1 > 0) {
1194 result += 1;
1195 }
1196 }
1197 return result;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1205 unchecked {
1206 uint256 result = log2(value);
1207 return result + (rounding == Rounding.Up && 1 << result < value ? 1 : 0);
1208 }
1209 }
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1216 uint256 result = 0;
1217 unchecked {
1218 if (value >= 10 ** 64) {
1219 value /= 10 ** 64; value /= 10 ** 64;
1220 result += 64;
1221 }
```

UNKNOWN Arithmetic operation "/=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1217 unchecked {
1218 if (value >= 10 ** 64) {
1219 value /= 10 ** 64;
1220 result += 64;
1221 }
1222 if (value >= 10 ** 32) {
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1217 unchecked {
1218 if (value >= 10 ** 64) {
1219 value /= 10 ** 64;
1220 result += 64;
1221 }
1222 if (value >= 10 ** 32) {
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1218 if (value >= 10 ** 64) {
1219 value /= 10 ** 64;
1220 result += 64;
1221 }
1222 if (value >= 10 ** 32) {
1223 value /= 10 ** 32;
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1220 result += 64;
1221 }
1222 if (value >= 10 ** 32) {
1223 value /= 10 ** 32; value /= 10 ** 32;
1224 result += 32;
1225 }
```

UNKNOWN Arithmetic operation "/=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1221 }
1222 if (value >= 10 ** 32) {
1223 value /= 10 ** 32;
1224 result += 32;
1225 }
1226 if (value >= 10 ** 16) {
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1221 }
1222 if (value >= 10 ** 32) {
1223 value /= 10 ** 32;
1224 result += 32;
1225 }
1226 if (value >= 10 ** 16) {
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1222 if (value >= 10 ** 32) {
1223 value /= 10 ** 32;
1224 result += 32;
1225 }
1226 if (value >= 10 ** 16) {
1227 value /= 10 ** 16;
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1224 result += 32;
1225 }
1226 if (value >= 10 ** 16) {
1227 value /= 10 ** 16; value /= 10 ** 16;
1228 result += 16;
1229 }
```

UNKNOWN Arithmetic operation "/=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1225 }
1226 if (value >= 10 ** 16) {
1227 value /= 10 ** 16;
1228 result += 16;
1229 }
1230 if (value >= 10 ** 8) {
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1225 }
1226 if (value >= 10 ** 16) {
1227 value /= 10 ** 16;
1228 result += 16;
1229 }
1230 if (value >= 10 ** 8) {
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1226 if (value >= 10 ** 16) {
1227 value /= 10 ** 16;
1228 result += 16;
1229 }
1230 if (value >= 10 ** 8) {
1231 value /= 10 ** 8;
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1228 result += 16;
1229 }
1230 if (value >= 10 ** 8) {
1231 value /= 10 ** 8; value /= 10 ** 8;
1232 result += 8;
1233 }
```

UNKNOWN Arithmetic operation "/=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1229 }
1230 if (value >= 10 ** 8) {
1231 value /= 10 ** 8;
1232 result += 8;
1233 }
1234 if (value >= 10 ** 4) {
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1229 }
1230 if (value >= 10 ** 8) {
1231 value /= 10 ** 8;
1232 result += 8;
1233 }
1234 if (value >= 10 ** 4) {
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1230 if (value >= 10 ** 8) {
1231 value /= 10 ** 8;
1232 result += 8;
1233 }
1234 if (value >= 10 ** 4) {
1235 value /= 10 ** 4;
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1232 result += 8;
1233 }
1234 if (value >= 10 ** 4) {
1235 value /= 10 ** 4; value /= 10 ** 4;
1236 result += 4;
1237 }
```

UNKNOWN Arithmetic operation "/=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1233 }
1234 if (value >= 10 ** 4) {
1235 value /= 10 ** 4;
1236 result += 4;
1237 }
1238 if (value >= 10 ** 2) {
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1233 }
1234 if (value >= 10 ** 4) {
1235 value /= 10 ** 4;
1236 result += 4;
1237 }
1238 if (value >= 10 ** 2) {
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1234 if (value >= 10 ** 4) {
1235 value /= 10 ** 4;
1236 result += 4;
1237 }
1238 if (value >= 10 ** 2) {
1239 value /= 10 ** 2;
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1236 result += 4;
1237 }
1238 if (value >= 10 ** 2) {
1239 value /= 10 ** 2; value /= 10 ** 2;
1240 result += 2;
1241 }
```

UNKNOWN Arithmetic operation "/=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1237 }
1238 if (value >= 10 ** 2) {
1239 value /= 10 ** 2;
1240 result += 2;
1241 }
1242 if (value >= 10 ** 1) {
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1237 }
1238 if (value >= 10 ** 2) {
1239 value /= 10 ** 2;
1240 result += 2;
1241 }
1242 if (value >= 10 ** 1) {
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1238 if (value >= 10 ** 2) {
1239 value /= 10 ** 2;
1240 result += 2;
1241 }
1242 if (value >= 10 ** 1) {
1243 result += 1;
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1240  result += 2;
1241  }
1242  if (value >= 10 ** 1) {
1243  result += 1; result += 1;
1244  }
1245  }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1241  }
1242  if (value >= 10 ** 1) {
1243  result += 1;
1244  }
1245  }
1246  return result;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1254  unchecked {
1255  uint256 result = log10(value);
1256  return result + (rounding == Rounding.Up && 10 ** result < value ? 1 : 0);
1257  }
1258  }
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1254 unchecked {
1255 uint256 result = log10(value);
1256 return result + (rounding == Rounding.Up && 10 ** result < value ? 1 : 0);
1257 }
1258 }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1269 if (value >> 128 > 0) {
1270 value >>= 128;
1271 result += 16;
1272 }
1273 if (value >> 64 > 0) {
1274 value >>= 64;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1273 if (value >> 64 > 0) {
1274 value >>= 64;
1275 result += 8;
1276 }
1277 if (value >> 32 > 0) {
1278 value >>= 32;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1277 if (value >> 32 > 0) {
1278 value >>= 32;
1279 result += 4;
1280 }
1281 if (value >> 16 > 0) {
1282 value >>= 16;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1281 if (value >> 16 > 0) {
1282 value >>= 16;
1283 result += 2;
1284 }
1285 if (value >> 8 > 0) {
1286 result += 1;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1284 }
1285 if (value >> 8 > 0) {
1286 result += 1;
1287 }
1288 }
1289 return result;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1297 unchecked {
1298 uint256 result = log256(value);
1299 return result + (rounding == Rounding.Up && 1 << (result << 3) < value ? 1 : 0);
1300 }
1301 }
1302 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1334 function average(int256 a, int256 b) internal pure returns (int256) {
1335 // Formula from the book "Hacker's Delight"
1336 int256 x = (a & b) + ((a ^ b) >> 1);
1337 return x + (int256(uint256(x) >> 255) & (a ^ b));
1338 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1335 // Formula from the book "Hacker's Delight"
1336 int256 x = (a & b) + ((a ^ b) >> 1);
1337 return x + (int256(uint256(x) >> 255) & (a ^ b));
1338 }
1339
1340 /**
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1370 function toString(uint256 value) internal pure returns (string memory) {
1371 unchecked {
1372 uint256 length = Math.log10(value) + 1;
1373 string memory buffer = new string(length);
1374 uint256 ptr;
1375 /// @solidity memory-safe-assembly
```

UNKNOWN Arithmetic operation "--" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1378 }
1379 while (true) {
1380 ptr--;
1381 /// @solidity memory-safe-assembly
1382 assembly {
1383 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1383 mstore8(ptr, byte(mod(value, 10), _SYMBOLS))
1384 }
1385 value /= 10;
1386 if (value == 0) break;
1387 }
1388 return buffer;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1402 function toHexString(uint256 value) internal pure returns (string memory) {
1403 unchecked {
1404 return toHexString(value, Math.log256(value) + 1);
1405 }
1406 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1410 */
1411 function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
1412 bytes memory buffer = new bytes(2 * length + 2);
1413 buffer[0] = "0";
1414 buffer[1] = "x";
1415 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1410 */
1411 function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
1412 bytes memory buffer = new bytes(2 * length + 2);
1413 buffer[0] = "0";
1414 buffer[1] = "x";
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1413  buffer[0] = "0";
1414  buffer[1] = "x";
1415  for (uint256 i = 2 * length + 1; i > 1; --i) {
1416  buffer[i] = _SYMBOLS[value & 0xf];
1417  value >>= 4;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1413  buffer[0] = "0";
1414  buffer[1] = "x";
1415  for (uint256 i = 2 * length + 1; i > 1; --i) {
1416  buffer[i] = _SYMBOLS[value & 0xf];
1417  value >>= 4;
```

UNKNOWN Arithmetic operation "--" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1413  buffer[0] = "0";
1414  buffer[1] = "x";
1415  for (uint256 i = 2 * length + 1; i > 1; --i) {
1416  buffer[i] = _SYMBOLS[value & 0xf]; buffer[i] = _SYMBOLS[value & 0xf];
1417  value >>= 4;
1418  }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1539 function tryRecover(bytes32 hash, bytes32 r, bytes32 vs) internal pure returns (address, RecoverError) {
1540 bytes32 s = vs & bytes32(0x7fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff);
1541 uint8 v = uint8((uint256(vs) >> 255) + 27);
1542 return tryRecover(hash, v, r, s);
1543 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2373
2374 function mintNFTFrom(address from, address to, uint256 amountNFT) public {
2375 uint256 amountFT = amountNFT * 10 ** decimals();
2376 require(
2377 allowance(from, _msgSender()) >= amountFT,
2378 "ERC20HX: Insufficient allowance"
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2373
2374 function mintNFTFrom(address from, address to, uint256 amountNFT) public {
2375 uint256 amountFT = amountNFT * 10 ** decimals();
2376 require(
2377 allowance(from, _msgSender()) >= amountFT,
2378 "ERC20HX: Insufficient allowance"
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2392 uint256 amountNFT
2393 ) virtual internal {
2394 uint256 amountFT = amountNFT * 10 ** decimals();
2395
2396 ERC20._burn(spender, amountFT);
2397 nftContract.mint(to, amountNFT);
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2392 uint256 amountNFT
2393 ) virtual internal {
2394 uint256 amountFT = amountNFT * 10 ** decimals();
2395
2396 ERC20._burn(spender, amountFT);
2397 nftContract.mint(to, amountNFT);
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2423 uint256[] calldata nftIds
2424 ) virtual internal {
2425 uint256 amountFT = (nftIds.length) * 10 ** decimals();
2426
2427 nftContract.burn(owner, nftIds);
2428 ERC20._mint(to, amountFT);
```

UNKNOWN

Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2423 uint256[] calldata nftIds
2424 ) virtual internal {
2425 uint256 amountFT = (nftIds.length) * 10 ** decimals();
2426
2427 nftContract.burn(owner, nftIds);
2428 ERC20._mint(to, amountFT);
```

UNKNOWN

Arithmetic operation "++" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2452 function _totalWeight() internal view virtual returns (uint96) {
2453 uint96 totalWeight = 0;
2454 for (uint i = 0; i < feesInfo.length; i++) totalWeight += feesInfo[i].weight;
2455
2456 return totalWeight;
```

UNKNOWN

Arithmetic operation "+=" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2452 function _totalWeight() internal view virtual returns (uint96) {
2453 uint96 totalWeight = 0;
2454 for (uint i = 0; i < feesInfo.length; i++) totalWeight += feesInfo[i].weight;
2455
2456 return totalWeight;
2457 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2466 require(receivers.length == weights.length, "ERC20HXFees: receivers length must be equal to feeumerators");
2467
2468 for (uint i = 0; i < receivers.length; i++) _pushFeeInfo(receivers[i], weights[i]);
2469 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2478
2479 function burnFee(uint256 amount) public view virtual returns (uint256) {
2480 uint256 burnFeeAmount = (amount * feeUnitAmount);
2481
2482 return burnFeeAmount;
2483 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2487
2488 if(msg.value > _burnFee) {
2489 uint256 refund = (msg.value - _burnFee);
2490 (bool sent, ) = _msgSender().call{value: refund}("");
2491
2492 require(sent, "ERC20HXFees: Failure to refund");
```

UNKNOWN

Arithmetic operation "++" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2495 uint256 totalWeight = _totalWeight();
2496 for(uint i = 0; i < feesInfo.length; i++) {
2497 uint256 recv = (_burnFee * feesInfo[i].weight) / totalWeight;
2498 (bool sent, ) = (feesInfo[i].receiver).call{value: recv}("");
```

UNKNOWN

Arithmetic operation "/" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2495 uint256 totalWeight = _totalWeight();
2496 for(uint i = 0; i < feesInfo.length; i++) {
2497 uint256 recv = (_burnFee * feesInfo[i].weight) / totalWeight;
2498 (bool sent, ) = (feesInfo[i].receiver).call{value: recv}("");
2499
2500 require(sent, "ERC20HXFees: Failure to transfer ETH");
```

UNKNOWN

Arithmetic operation "*" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2495 uint256 totalWeight = _totalWeight();
2496 for(uint i = 0; i < feesInfo.length; i++) {
2497 uint256 recv = (_burnFee * feesInfo[i].weight) / totalWeight;
2498 (bool sent, ) = (feesInfo[i].receiver).call{value: recv}("");
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2522 address _nftContractAddress
2523 ) ERC20("ExampleERC20HXFees", "ExampleERC20HXFees") ERC20Permit("ExampleERC20HXFees") {
2524 _mint(msg.sender, 10000 * 10 ** decimals());
2525 _setNFTContract(_nftContractAddress);
2526 }
2527 }
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2522 address _nftContractAddress
2523 ) ERC20("ExampleERC20HXFees", "ExampleERC20HXFees") ERC20Permit("ExampleERC20HXFees") {
2524 _mint(msg.sender, 10000 * 10 ** decimals());
2525 _setNFTContract(_nftContractAddress);
2526 }
2527 }
```

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

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

SWC-101

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
950 require(value > 0, "Counter: decrement overflow");
951 unchecked {
952 counter._value = value - 1;
953 }
954 }
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
1008 function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
1009 // (a + b - 1) / b can overflow on addition, so we distribute.
1010 return a == 0 ? 0 : (a - 1) / b + 1;
1011 }
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
8  // OpenZeppelin Contracts (last updated v4.9.4) (utils/Context.sol)
9
10 pragma solidity ^0.8.0;
11
12 /**
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
40 // OpenZeppelin Contracts (last updated v4.9.0) (access/Ownable.sol)
41
42 pragma solidity ^0.8.0;
43
44 /**
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
125 // OpenZeppelin Contracts (last updated v4.9.0) (token/ERC20/IERC20.sol)
126
127 pragma solidity ^0.8.0;
128
129 /**
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
207 // OpenZeppelin Contracts v4.4.1 (interfaces/IERC20.sol)
208
209 pragma solidity ^0.8.0;
210
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
215 // OpenZeppelin Contracts v4.4.1 (utils/introspection/IERC165.sol)
216
217 pragma solidity ^0.8.0;
218
219 /**
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
244 // OpenZeppelin Contracts (last updated v4.9.0) (token/ERC721/IERC721.sol)
245
246 pragma solidity ^0.8.0;
247
248 /**
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
378 // OpenZeppelin Contracts v4.4.1 (interfaces/IERC721.sol)
379
380 pragma solidity ^0.8.0;
381
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
386 // OpenZeppelin Contracts v4.4.1 (token/ERC20/extensions/IERC20Metadata.sol)
387
388 pragma solidity ^0.8.0;
389
390 /**
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
416 // OpenZeppelin Contracts (last updated v4.9.0) (token/ERC20/ERC20.sol)
417
418 pragma solidity ^0.8.0;
419
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
783 // OpenZeppelin Contracts (last updated v4.5.0) (token/ERC20/extensions/ERC20Burnable.sol)
784
785 pragma solidity ^0.8.0;
786
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
824 // OpenZeppelin Contracts (last updated v4.9.4) (token/ERC20/extensions/IERC20Permit.sol)
825
826 pragma solidity ^0.8.0;
827
828 /**
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
918 // OpenZeppelin Contracts v4.4.1 (utils/Counters.sol)
919
920 pragma solidity ^0.8.0;
921
922 /**
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
965 // OpenZeppelin Contracts (last updated v4.9.0) (utils/math/Math.sol)
966
967 pragma solidity ^0.8.0;
968
969 /**
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1308 // OpenZeppelin Contracts (last updated v4.8.0) (utils/math/SignedMath.sol)
1309
1310 pragma solidity ^0.8.0;
1311
1312 /**
1313 * @dev Standard signed math utilities missing in the Solidity language.
1314 */
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1355 // OpenZeppelin Contracts (last updated v4.9.0) (utils/Strings.sol)
1356
1357 pragma solidity ^0.8.0;
1358
1359
1360 /**
1361 * @dev String operations.
1362 */
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1442 // OpenZeppelin Contracts (last updated v4.9.0) (utils/cryptography/ECDSA.sol)
1443
1444 pragma solidity ^0.8.0;
1445
1446 /**
1447 * @dev Elliptic Curve Digital Signature Algorithm (ECDSA) operations.
1448 *
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1661 // OpenZeppelin Contracts (last updated v4.9.0) (interfaces/IERC5267.sol)
1662
1663 pragma solidity ^0.8.0;
1664
1665 interface IERC5267 {
1666 /**
1667 * @dev MAY be emitted to signal that the domain could have changed.
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1694 // This file was procedurally generated from scripts/generate/templates/StorageSlot.js.
1695
1696 pragma solidity ^0.8.0;
1697
1698 /**
1699 * @dev Library for reading and writing primitive types to specific storage slots.
1700 *
1701 * Storage slots are often used to avoid storage conflict when dealing with upgradeable contracts.
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.8"". 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1835 // OpenZeppelin Contracts (last updated v4.9.0) (utils/ShortStrings.sol)
1836
1837 pragma solidity ^0.8.8;
1838
1839 // | string | 0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA |
1840 // | length | 0x BB |
1841 type ShortString is bytes32;
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.8"". 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1959 // OpenZeppelin Contracts (last updated v4.9.0) (utils/cryptography/EIP712.sol)
1960
1961 pragma solidity ^0.8.8;
1962
1963
1964
1965 /**
1966 * @dev https://eips.ethereum.org/EIPS/eip-712[EIP 712] is a standard for hashing and signing of typed structured data.
1967 *
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
2103 // OpenZeppelin Contracts (last updated v4.9.4) (token/ERC20/extensions/ERC20Permit.sol)
2104
2105 pragma solidity ^0.8.0;
2106
2107
2108
2109
2110
2111 /**
2112 * @dev Implementation of the ERC20 Permit extension allowing approvals to be made via signatures, as defined in
2113 * https://eips.ethereum.org/EIPS/eip-2612[EIP-2612].
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
2200 // OpenZeppelin Contracts (last updated v4.9.0) (security/ReentrancyGuard.sol)
2201
2202 pragma solidity ^0.8.0;
2203
2204 /**
2205 * @dev Contract module that helps prevent reentrant calls to a function.
2206 *
2207 * Inheriting from `ReentrancyGuard` will make the {nonReentrant} modifier
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.24"". 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
2279
2280 // Original license: SPDX_License_Identifier: MIT
2281 pragma solidity ^0.8.24;
2282
2283 interface IERC721HX is IERC721 {
2284 function MINTER_ROLE() external returns (bytes32);
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.24"". 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
2319
2320 // Original license: SPDX_License_Identifier: MIT
2321 pragma solidity ^0.8.24;
2322
2323
2324 interface IERC20HX is IERC20 {
2325 // Events
2326 event MintNFT(address indexed from, uint256 amount);
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.24"". 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
2348
2349 // Original license: SPDX_License_Identifier: MIT
2350 pragma solidity ^0.8.24;
2351
2352
2353
2354
2355
2356
2357 abstract contract ERC20HX is IERC20HX, ERC20, ERC20Permit, Ownable {abstract contract ERC20HX is IERC20HX, ERC20, ERC20Permit, Ownable {
2358 IERC721HX nftContract;
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.24"". 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
2436
2437 // Original license: SPDX_License_Identifier: MIT
2438 pragma solidity ^0.8.24;
2439
2440
2441 /* assume fees to be paid in ETH */
2442 abstract contract ERC20HXFees is ERC20HX, ReentrancyGuard {
2443 uint256 feeUnitAmount;
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.24"". 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
2516
2517 // Original license: SPDX_License_Identifier: MIT
2518 pragma solidity ^0.8.24;
2519
2520 contract ExampleERC20HXFees is ERC20HXFees {
2521 constructor(
2522 address _nftContractAddress
```

UNKNOWN

SWC-110

Public state variable with array type causing reacheable exception by default.

The public state variable "feesInfo" in "ERC20HXFees" contract has type "struct ERC20HXFees.FeesInfo[]" and can cause an exception in case of use of invalid array index value.

Source file

/hx_fees/exampleerc20hxfees.sol

Locations

```
2448 }
2449
2450 FeesInfo[] public feesInfo;
2451
2452 function _totalWeight() internal view virtual returns (uint96) {
2453 uint96 totalWeight = 0;
2454 for (uint i = 0; i < feesInfo.length; i++) totalWeight += feesInfo[i].weight;
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1411 function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
1412 bytes memory buffer = new bytes(2 * length + 2);
1413 buffer[0] = "0";
1414 buffer[1] = "x";
1415 for (uint256 i = 2 * length + 1; i > 1; --i) {
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1412 bytes memory buffer = new bytes(2 * length + 2);
1413 buffer[0] = "0";
1414 buffer[1] = "x";
1415 for (uint256 i = 2 * length + 1; i > 1; --i) {
1416 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1414 buffer[1] = "x";
1415 for (uint256 i = 2 * length + 1; i > 1; --i) {
1416 buffer[i] = _SYMBOLS[value & 0xf];
1417 value >>= 4;
1418 }
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
1414 buffer[1] = "x";
1415 for (uint256 i = 2 * length + 1; i > 1; --i) {
1416 buffer[i] = _SYMBOLS[value & 0xf];
1417 value >>= 4;
1418 }
1419 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

/hx_fees/exampleerc20hxfees.sol

Locations

```
2452 function _totalWeight() internal view virtual returns (uint96) {
2453 uint96 totalWeight = 0;
2454 for (uint i = 0; i < feesInfo.length; i++) totalWeight += feesInfo[i].weight;
2455
2456 return totalWeight;
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
2466 require(receivers.length == weights.length, "ERC20HXFees: receivers length must be equal to feeumerators");
2467
2468 for (uint i = 0; i < receivers.length; i++) _pushFeeInfo(receivers[i], weights[i]);
2469 }
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
2466 require(receivers.length == weights.length, "ERC20HXFees: receivers length must be equal to feeumerators");
2467
2468 for (uint i = 0; i < receivers.length; i++) _pushFeeInfo(receivers[i], weights[i]);
2469 }
2470
2471 function deleteFeeInfo() public virtual onlyOwner {
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
2495 uint256 totalWeight = _totalWeight();
2496 for(uint i = 0; i < feesInfo.length; i++) {
2497 uint256 recv = (_burnFee * feesInfo[i].weight) / totalWeight;
2498 (bool sent, ) = (feesInfo[i].receiver).call{value: recv}("");
```

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

/hx_fees/exampleerc20hxfees.sol

Locations

```
2496 for(uint i = 0; i < feesInfo.length; i++) {
2497 uint256 recv = (_burnFee * feesInfo[i].weight) / totalWeight;
2498 (bool sent, ) = (feesInfo[i].receiver).call{value: recv}("");
2499
2500 require(sent, "ERC20HXFees: Failure to transfer ETH");
```