MythX

Analysis 3ac5c7ca-deb1-495c-92dd-12346925f763

MythX

Started	Mon Feb 19 2024 21:53:50 GMT+0000 (Coordinated Universal Time)
Finished	Mon Feb 19 2024 21:53:55 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Vscode-Extension
Main Source File	/Flatten/Simpleerc20hx.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	25

ISSUES

UNKNOWN SWC-101 Arithmetic operation "+" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN SWC-101 Arithmetic operation "-" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
_beforeTokenTransfer(address(0), account, amount);

_totalSupply += amount;
unchecked {
// Overflow not possible: balance + amount is at most totalSupply + amount, which is checked above.
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
function increment(Counter storage counter) internal {
unchecked {
counter._value += 1;
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```

// Shift in bits from prod1 into prod0.
prod0 |= prod1 * twos;

// Invert denominator mod 2^256. Now that denominator is an odd number, it has an inverse modulo 2^256 such
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

// Use the Newton-Raphson iteration to improve the precision. Thanks to Hensel's lifting lemma, this also works
```

UNKNOWN

SWC-101

Arithmetic operation "*=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
inverse *= 2 - denominator * inverse; // inverse mod 2^8
inverse *= 2 - denominator * inverse; // inverse mod 2^16
inverse *= 2 - denominator * inverse; // inverse mod 2^32
inverse *= 2 - denominator * inverse; // inverse mod 2^64
inverse *= 2 - denominator * inverse; // inverse mod 2^128
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
inverse *= 2 - denominator * inverse; // inverse mod 2^8
inverse *= 2 - denominator * inverse; // inverse mod 2^16
inverse *= 2 - denominator * inverse; // inverse mod 2^32
inverse *= 2 - denominator * inverse; // inverse mod 2^64
inverse *= 2 - denominator * inverse; // inverse mod 2^128
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
inverse *= 2 - denominator * inverse; // inverse mod 2^8
inverse *= 2 - denominator * inverse; // inverse mod 2^16
inverse *= 2 - denominator * inverse; // inverse mod 2^32
inverse *= 2 - denominator * inverse; // inverse mod 2^64
inverse *= 2 - denominator * inverse; // inverse mod 2^128
```

UNKNOWN

SWC-101

Arithmetic operation "*=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
inverse *= 2 - denominator * inverse; // inverse mod 2^16
inverse *= 2 - denominator * inverse; // inverse mod 2^32
inverse *= 2 - denominator * inverse; // inverse mod 2^64
inverse *= 2 - denominator * inverse; // inverse mod 2^128
inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
inverse *= 2 - denominator * inverse; // inverse mod 2^16
inverse *= 2 - denominator * inverse; // inverse mod 2^32
inverse *= 2 - denominator * inverse; // inverse mod 2^64
inverse *= 2 - denominator * inverse; // inverse mod 2^128
inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
inverse *= 2 - denominator * inverse; // inverse mod 2^16
inverse *= 2 - denominator * inverse; // inverse mod 2^32
inverse *= 2 - denominator * inverse; // inverse mod 2^64
inverse *= 2 - denominator * inverse; // inverse mod 2^128
inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN

SWC-101

Arithmetic operation "*=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
inverse *= 2 - denominator * inverse; // inverse mod 2^32
inverse *= 2 - denominator * inverse; // inverse mod 2^64
inverse *= 2 - denominator * inverse; // inverse mod 2^128
inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
inverse *= 2 - denominator * inverse; // inverse mod 2^32
inverse *= 2 - denominator * inverse; // inverse mod 2^64
inverse *= 2 - denominator * inverse; // inverse mod 2^128
inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
inverse *= 2 - denominator * inverse; // inverse mod 2^32
inverse *= 2 - denominator * inverse; // inverse mod 2^64
inverse *= 2 - denominator * inverse; // inverse mod 2^128
inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN

SWC-101

Arithmetic operation "*=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
inverse *= 2 - denominator * inverse; // inverse mod 2^64
inverse *= 2 - denominator * inverse; // inverse mod 2^128
inverse *= 2 - denominator * inverse; // inverse mod 2^256

// Because the division is now exact we can divide by multiplying with the modular inverse of denominator.
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
inverse *= 2 - denominator * inverse; // inverse mod 2^64
inverse *= 2 - denominator * inverse; // inverse mod 2^128
inverse *= 2 - denominator * inverse; // inverse mod 2^256

// Because the division is now exact we can divide by multiplying with the modular inverse of denominator.
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
inverse *= 2 - denominator * inverse; // inverse mod 2^64
inverse *= 2 - denominator * inverse; // inverse mod 2^128
inverse *= 2 - denominator * inverse; // inverse mod 2^256

// Because the division is now exact we can divide by multiplying with the modular inverse of denominator.
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
// into the expected uint128 result.
unchecked {
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
// into the expected uint128 result.
unchecked {
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
unchecked {
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
unchecked {
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
return min(result, a / result);
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
return min(result, a / result);
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
return min(result, a / result);
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
return min(result, a / result);
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
result = (result + a / result) >> 1;
result = (result + a / result) >> 1;
return min(result, a / result);
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
if (value >> 128 > 0) {
value >>= 128;
result += 128;
}
if (value >> 64 > 0) {
value >>= 64;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
if (value >> 64 > 0) {
value >>= 64;
result += 64;
}
if (value >> 32 > 0) {
value >>= 32;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
if (value >> 32 > 0) {
value >>= 32;
result += 32;
}
if (value >> 16 > 0) {
value >>= 16;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
if (value >> 16 > 0) {
value >>= 16;
result += 16;
}
if (value >> 8 > 0) {
value >>= 8;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
if (value >> 8 > 0) {
value >>= 8;
result += 8;
}
if (value >> 4 > 0) {
value >>= 4;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
if (value >> 4 > 0) {
value >>= 4;
result += 4;
}
if (value >> 2 > 0) {
value >>= 2;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
if (value >> 2 > 0) {
value >>= 2;
result += 2;
}
if (value >> 1 > 0) {
result += 1;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
}
if (value >> 1 > 0) {
result += 1;
}
}
return result;
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
uint256 result = 0;
unchecked {
if (value >= 10 ** 64) {
value /= 10 ** 64; value /= 10 ** 64;
result += 64;
}
```

UNKNOWN

SWC-101

Arithmetic operation "/=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
unchecked {
if (value >= 10 ** 64) {
value /= 10 ** 64;
result += 64;
}
if (value >= 10 ** 32) {
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
unchecked {
if (value >= 10 ** 64) {
value /= 10 ** 64;
result += 64;
}
if (value >= 10 ** 32) {
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
if (value >= 10 ** 64) {
value /= 10 ** 64;
result += 64;
}
if (value >= 10 ** 32) {
value /= 10 ** 32;
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
result += 64;
}
if (value >= 10 ** 32) {
value /= 10 ** 32; value /= 10 ** 32;
result += 32;
}
```

UNKNOWN

SWC-101

Arithmetic operation "/=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
}
if (value >= 10 ** 32) {
value /= 10 ** 32;
result += 32;
}
if (value >= 10 ** 16) {
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
}
if (value >= 10 ** 32) {
value /= 10 ** 32;
result += 32;
}
if (value >= 10 ** 16) {
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
if (value >= 10 ** 32) {
value /= 10 ** 32;
result += 32;
}
if (value >= 10 ** 16) {
value /= 10 ** 16;
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
result += 32;
}
if (value >= 10 ** 16) {
value /= 10 ** 16; value /= 10 ** 16;
result += 16;
}
```

UNKNOWN

SWC-101

Arithmetic operation "/=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
}
if (value >= 10 ** 16) {
value /= 10 ** 16;
result += 16;
}
if (value >= 10 ** 8) {
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
}
if (value >= 10 ** 16) {
value /= 10 ** 16;
result += 16;
}
if (value >= 10 ** 8) {
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
if (value >= 10 ** 16) {
value /= 10 ** 16;
result += 16;
}
if (value >= 10 ** 8) {
value /= 10 ** 8;
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
result += 16;
}
if (value >= 10 ** 8) {
value /= 10 ** 8; value /= 10 ** 8;
result += 8;
}
```

UNKNOWN

SWC-101

Arithmetic operation "/=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
}
if (value >= 10 ** 8) {
value /= 10 ** 8;
result += 8;
}
if (value >= 10 ** 4) {
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
}
if (value >= 10 ** 8) {
value /= 10 ** 8;
result += 8;
}
if (value >= 10 ** 4) {
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
if (value >= 10 ** 8) {
value /= 10 ** 8;
result += 8;
}
if (value >= 10 ** 4) {
value /= 10 ** 4;
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
result += 8;
}
if (value >= 10 ** 4) {
value /= 10 ** 4; value /= 10 ** 4;
result += 4;
}
```

UNKNOWN

SWC-101

Arithmetic operation "/=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
}
if (value >= 10 ** 4) {
value /= 10 ** 4;
result += 4;
}
if (value >= 10 ** 2) {
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
}
if (value >= 10 ** 4) {
value /= 10 ** 4;
result += 4;
}
if (value >= 10 ** 2) {
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
if (value >= 10 ** 4) {
value /= 10 ** 4;
result += 4;
}
if (value >= 10 ** 2) {
value /= 10 ** 2;
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
result += 4;
}
if (value >= 10 ** 2) {
value /= 10 ** 2; value /= 10 ** 2;
result += 2;
}
```

UNKNOWN

SWC-101

Arithmetic operation "/=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
}
if (value >= 10 ** 2) {
value /= 10 ** 2;
result += 2;
}
if (value >= 10 ** 1) {
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
}
if (value >= 10 ** 2) {
value /= 10 ** 2;
result += 2;
}
if (value >= 10 ** 1) {
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
if (value >= 10 ** 2) {
value /= 10 ** 2;
result += 2;
}
if (value >= 10 ** 1) {
result += 1;
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
if (value >> 128 > 0) {
value >>= 128;
result += 16;
}
if (value >> 64 > 0) {
value >>= 64;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
if (value >> 64 > 0) {
value >>= 64;
result += 8;
}
if (value >> 32 > 0) {
value >>= 32;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
if (value >> 32 > 0) {
value >>= 32;
result += 4;
}
if (value >> 16 > 0) {
value >>= 16;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
if (value >> 16 > 0) {
value >>= 16;
result += 2;
}
if (value >> 8 > 0) {
result += 1;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

```
}
if (value >> 8 > 0) {
result += 1;
}
}
return result;
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

/**
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "--" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
}
while (true) {
ptr--;
/// @solidity memory-safe-assembly
assembly {
mstore8(ptr, byte(mod(value, 10), _SYMBOLS))
```

UNKNOWN

SWC-101

Arithmetic operation "/=" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
mstore8(ptr, byte(mod(value, 10), _SYMBOLS))
}
value /= 10;
if (value == 0) break;
}
return buffer;
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "--" discovered

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

Source file

`/flatten/simpleerc20hx.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
uint256 amountNFT
) internal {
uint256 amountFT = amountNFT * 10 ** decimals();

ERC20._burn(spender, amountFT);
nftContract.mint(to, amountNFT);
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
uint256 amountNFT
) internal {
uint256 amountFT = amountNFT * 10 ** decimals();

ERC20._burn(spender, amountFT);
nftContract.mint(to, amountNFT);
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
uint256[] calldata nftIds
) internal {
uint256 amountFT = (nftIds.length) * 10 ** decimals();

nftContract.burn(owner, nftIds);
ERC20._mint(to, amountFT);
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
uint256[] calldata nftIds
) internal {
uint256 amountFT = (nftIds.length) * 10 ** decimals();

nftContract.burn(owner, nftIds);
ERC20._mint(to, amountFT);
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
address _nftContractAddress
) ERC20("SimpleERC20HX", "SimpleERC20HX") ERC20Permit("SimpleERC20HX") {
_mint(msg.sender, 10000 * 10 ** decimals());
_setNFTContract(_nftContractAddress);
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

```
address _nftContractAddress
) ERC20("SimpleERC20HX", "SimpleERC20HX") ERC20Permit("SimpleERC20HX") {
_mint(msg.sender, 10000 * 10 ** decimals());
_setNFTContract(_nftContractAddress);
}
}
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

/flatten/simpleerc20hx.sol

Locations

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

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

/flatten/simpleerc20hx.sol

Locations

```
// OpenZeppelin Contracts (last updated v4.9.4) (utils/Context.sol)

pragma solidity ^0.8.0;

/**
```

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

/flatten/simpleerc20hx.sol

Locations

```
// OpenZeppelin Contracts (last updated v4.9.0) (access/Ownable.sol)

pragma solidity ^0.8.0;

/**
```

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

/flatten/simpleerc20hx.sol

Locations

```
// OpenZeppelin Contracts (last updated v4.9.0) (token/ERC20/IERC20.sol)

pragma solidity ^0.8.0;

/**
```

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

/flatten/simpleerc20hx.sol

Locations

```
// OpenZeppelin Contracts v4.4.1 (interfaces/IERC20.sol)

pragma solidity ^0.8.0;

```

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

/flatten/simpleerc20hx.sol

Locations

```
// OpenZeppelin Contracts v4.4.1 (utils/introspection/IERC165.sol)

pragma solidity ^0.8.0;

/**
```

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

/flatten/simpleerc20hx.sol

Locations

```
// OpenZeppelin Contracts (last updated v4.9.0) (token/ERC721/IERC721.sol)

pragma solidity ^0.8.0;

/**
```

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

/flatten/simpleerc20hx.sol

Locations

```
// OpenZeppelin Contracts v4.4.1 (interfaces/IERC721.sol)

pragma solidity ^0.8.0;

```

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

`/flatten/simpleerc20hx.sol`

Locations

```
// OpenZeppelin Contracts v4.4.1 (token/ERC20/extensions/IERC20Metadata.sol)

pragma solidity ^0.8.0;

/**
```

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

`/flatten/simpleerc20hx.sol`

Locations

```
// OpenZeppelin Contracts (last updated v4.9.0) (token/ERC20/ERC20.sol)

pragma solidity ^0.8.0;

```

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

`/flatten/simpleerc20hx.sol`

Locations

```
// OpenZeppelin Contracts (last updated v4.5.0) (token/ERC20/extensions/ERC20Burnable.sol)

pragma solidity ^0.8.0;

```

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

`/flatten/simpleerc20hx.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

`/flatten/simpleerc20hx.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

`/flatten/simpleerc20hx.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

/flatten/simpleerc20hx.sol

Locations

```
// OpenZeppelin Contracts (last updated v4.8.0) (utils/math/SignedMath.sol)

pragma solidity ^0.8.0;

/**
* @dev Standard signed math utilities missing in the Solidity language.
*/
```

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

/flatten/simpleerc20hx.sol

Locations

```
// OpenZeppelin Contracts (last updated v4.9.0) (utils/Strings.sol)

pragma solidity ^0.8.0;


/**
* @dev String operations.
*/
```

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

/flatten/simpleerc20hx.sol

Locations

```
// OpenZeppelin Contracts (last updated v4.9.0) (utils/cryptography/ECDSA.sol)

pragma solidity ^0.8.0;

/**
* @dev Elliptic Curve Digital Signature Algorithm (ECDSA) operations.
*
```

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

/flatten/simpleerc20hx.sol

Locations

```
// OpenZeppelin Contracts (last updated v4.9.0) (interfaces/IERC5267.sol)

pragma solidity ^0.8.0;

interface IERC5267 {
/**
* @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

/flatten/simpleerc20hx.sol

Locations

```
// This file was procedurally generated from scripts/generate/templates/StorageSlot.js.

pragma solidity ^0.8.0;

/**
* @dev Library for reading and writing primitive types to specific storage slots.
*
* 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

/flatten/simpleerc20hx.sol

Locations

```
// OpenZeppelin Contracts (last updated v4.9.0) (utils/ShortStrings.sol)

pragma solidity ^0.8.8;

// | string | 0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA |
// | length | 0x BB |
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

`/flatten/simpleerc20hx.sol`

Locations

```
// OpenZeppelin Contracts (last updated v4.9.0) (utils/cryptography/EIP712.sol)

pragma solidity ^0.8.8;




/**
* @dev https://eips.ethereum.org/EIPS/eip-712[EIP 712] is a standard for hashing and signing of typed structured data.
*
```

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

`/flatten/simpleerc20hx.sol`

Locations

```
// OpenZeppelin Contracts (last updated v4.9.4) (token/ERC20/extensions/ERC20Permit.sol)

pragma solidity ^0.8.0;





/**
* @dev Implementation of the ERC20 Permit extension allowing approvals to be made via signatures, as defined in
* https://eips.ethereum.org/EIPS/eip-2612[EIP-2612].
```

LOW

SWC-103

A floating pragma is set.

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

`/flatten/simpleerc20hx.sol`

Locations

```

// Original license: SPDX_License_Identifier: MIT
pragma solidity ^0.8.24;

interface IERC721HX is IERC721 {
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

/flatten/simpleerc20hx.sol

Locations

```

// Original license: SPDX_License_Identifier: MIT
pragma solidity ^0.8.24;


interface IERC20HX is IERC20 {
// Events
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

/flatten/simpleerc20hx.sol

Locations

```

// Original license: SPDX_License_Identifier: MIT
pragma solidity ^0.8.24;






abstract contract ERC20HX is IERC20HX, ERC20, ERC20Permit, Ownable {abstract contract ERC20HX is IERC20HX, ERC20, ERC20Permit, Ownable {
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

/flatten/simpleerc20hx.sol

Locations

```

// Original license: SPDX_License_Identifier: MIT
pragma solidity ^0.8.24;

contract SimpleERC20HX is ERC20HX {
constructor(
address _nftContractAddress
```

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

/flatten/simpleerc20hx.sol

Locations

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

/flatten/simpleerc20hx.sol

Locations

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

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

/flatten/simpleerc20hx.sol

Locations

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

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

`/flatten/simpleerc20hx.sol`

Locations

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