MythX

REPORT 604C4A36F31D800018A83911

Created	Sat Mar 13 2021 05:14:30 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	pyxis.cto@gmail.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
d570bb89-c6ba-4af4-8c52-f132965d4088	PYXToken.sol	5

Analysis d570bb89-c6ba-4af4-8c52-f132965d4088 MythX

Started	Sat Mar 13 2021 05:14:40 GMT+0000 (Coordinated Universal Time)
Finished	Sat Mar 13 2021 06:00:02 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	PYXToken.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	5

ISSUES

MythX

REPORT 604CFAC78F58E800195E43BD

Created	Sat Mar 13 2021 17:47:51 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	pyxis.cto@gmail.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
759cb471-cb55-469e-bfeb-ae4d8efa6799	CMPSToken.sol	2

Analysis 759cb471-cb55-469e-bfeb-ae4d8efa6799 MythX

Started	Sat Mar 13 2021 17:47:55 GMT+0000 (Coordinated Universal Time)
Finished	Sat Mar 13 2021 18:33:13 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	CMPSToken.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
2	0	0

ISSUES

MythX

REPORT 604CFAD7F62C32001842B046

Created	Sat Mar 13 2021 17:48:07 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	pyxis.cto@gmail.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
fceffd72-1819-4a67-be8a-88682417f422	CMPSReservation.sol	4

Analysis `fceffd72-1819-4a67-be8a-88682417f422` MythX

Started	Sat Mar 13 2021 17:48:15 GMT+0000 (Coordinated Universal Time)
Finished	Sat Mar 13 2021 18:33:33 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	`CMPSReservation.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	4

ISSUES

MythX

REPORT 604CFAE35F8BEB001989AE07

Created	Sat Mar 13 2021 17:48:19 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	pyxis.cto@gmail.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
24921ecd-8934-4b86-ad05-af3aeb357d1a	CMPSToPYXSwapper.sol	14

Analysis 24921ecd-8934-4b86-ad05-af3aeb357d1a MythX

Started	Sat Mar 13 2021 17:48:25 GMT+0000 (Coordinated Universal Time)
Finished	Sat Mar 13 2021 18:33:43 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	CMPSToPYXSwapper.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	1	13

ISSUES

MythX

REPORT 604C4A17F31D800018A8390D

Created	Sat Mar 13 2021 05:13:59 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	pyxis.cto@gmail.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
9aebaf68-67a9-4300-a92e-430a45579863	PYXStaking.sol	4

Analysis 9aebaf68-67a9-4300-a92e-430a45579863 MythX

Started	Sat Mar 13 2021 05:14:00 GMT+0000 (Coordinated Universal Time)
Finished	Sat Mar 13 2021 05:59:46 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	PYXStaking.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	4

ISSUES

MythX

REPORT 604C4A2301C00600186CFB5D

Created	Sat Mar 13 2021 05:14:11 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	pyxis.cto@gmail.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
a921f243-b283-4398-9f29-13d74c83a94e	ETHAutostaking.sol	19

Analysis a921f243-b283-4398-9f29-13d74c83a94e MythX

Started	Sat Mar 13 2021 05:14:20 GMT+0000 (Coordinated Universal Time)
Finished	Sat Mar 13 2021 05:59:41 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	ETHAutostaking.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	4	15

ISSUES

MEDIUM

Incorrect function "getRoleMemberCount" state mutability

Function "getRoleMemberCount" state mutability is considered "view" by compiler, but should be set to non-payable (default).

SWC-000

Source file
ETHAutostaking.sol
Locations

```
570 * together with {getRoleMember} to enumerate all bearers of a role.
571 */
572 function getRoleMemberCount(bytes32 role) public view returns (uint256) {
573 return _roles[role].members.length();
574 }
575
576 /**
```

MEDIUM

Function could be marked as external.

The function definition of "getRoleAdmin" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

SWC-000

Source file
ETHAutostaking.sol
Locations

```
596 * To change a role's admin, use {_setRoleAdmin}.
597 */
598 function getRoleAdmin(bytes32 role) public view returns (bytes32) {
599 return _roles[role].adminRole;
600 }
601
602 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "grantRole" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

ETHAutostaking.sol

Locations

```
610 | * - the caller must have ``role```s admin role.
611 | */
612 | function grantRole(bytes32 role, address account) public virtual {
613 | require(hasRole(_roles[role].adminRole, _msgSender()), "AccessControl: sender must be an admin to grant");
614 |
615 | _grantRole(role, account);
616 | }
617 |
618 | /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "revokeRole" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

ETHAutostaking.sol

Locations

```
625 | * - the caller must have ``role```s admin role.
626 | */
627 | function revokeRole(bytes32 role, address account) public virtual {
628 | require(hasRole(_roles[role].adminRole, _msgSender()), "AccessControl: sender must be an admin to revoke");
629 |
630 | _revokeRole(role, account);
631 | }
632 |
633 | /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.6.0<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

ETHAutostaking.sol

Locations

```
5 | // SPDX-License-Identifier: MIT
6 |
7 | pragma solidity >=0.6.0 <0.8.0;
8 |
9 | /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.6.2<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

ETHAutostaking.sol

Locations

```
283
284
285 pragma solidity >=0.6.2 <0.8.0;
286
287 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.6.0<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

ETHAutostaking.sol

Locations

```
452
453
454 pragma solidity >=0.6.0 <0.8.0;
455
456 /*
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.6.0<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

ETHAutostaking.sol

Locations

```
480
481
482 pragma solidity >=0.6.0 <0.8.0;
483
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.6.0<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

ETHAutostaking.sol

Locations

```
699
700
701 pragma solidity >=0.6.0 <0.8.0;
702
703 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.6.2"". 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

ETHAutostaking.sol

Locations

```
860 // File @uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol@v1.1.0-beta.0
861
862 pragma solidity >=0.6.2;
863
864 interface IUniswapV2Router01 {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.6.2"". 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

ETHAutostaking.sol

Locations

```
959 // File @uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol@v1.1.0-beta.0
960
961 pragma solidity >=0.6.2;
962
963 interface IUniswapV2Router02 is IUniswapV2Router01 {
```

LOW

SWC-103

A floating pragma is set.

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

Source file

ETHAutostaking.sol

Locations

```
1007
1008
1009 pragma solidity ^0.6.0;
1010
1011 interface IAutoStaking {
```

LOW

SWC-103

A floating pragma is set.

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

Source file

ETHAutostaking.sol

Locations

```
1018
1019
1020 pragma solidity ^0.6.0;
1021
1022 interface IPYXStaking {
```

LOW

SWC-103

A floating pragma is set.

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

Source file

ETHAutostaking.sol

Locations

```
1037
1038
1039 pragma solidity ^0.6.0;
1040
1041 interface IPYXToken {
```

LOW

SWC-103

A floating pragma is set.

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

Source file

ETHAutostaking.sol

Locations

```
1052
1053
1054 pragma solidity ^0.6.0;
1055
```

LOW

SWC-107

Read of persistent state following external call.

The contract account state is accessed after an external call. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

ETHAutostaking.sol

Locations

```
1264 ADDRESSES.PYX_TOKEN.burn(msg.sender, _pyx);
1265
1266 totalForSalePYX = totalForSalePYX.add(_pyx);
1267
1268 // [event]
```

LOW

SWC-107

Write to persistent state following external call.

The contract account state is accessed after an external call. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

ETHAutostaking.sol

Locations

```
1264 ADDRESSES.PYX_TOKEN.burn(msg.sender, _pyx);
1265
1266 totalForSalePYX = totalForSalePYX.add(_pyx);
1267
1268 // [event]
```

LOW

SWC-110

An assertion violation was triggered.

It is possible to cause an assertion violation. Note that Solidity assert() statements should only be used to check invariants. Review the transaction trace generated for this issue and either make sure your program logic is correct, or use require() instead of assert() if your goal is to constrain user inputs or enforce preconditions. Remember to validate inputs from both callers (for instance, via passed arguments) and callees (for instance, via return values).

Source file

ETHAutostaking.sol

Locations

```
1350 */
1351 function getNumDayInWeek() public view returns (uint256) {
1352 return (block.timestamp / SETTINGS.STEP_SECONDS) % 7;
1353 }
```

LOW

Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

SWC-123

Source file

ETHAutostaking.sol

Locations

```
1262
1263 function addForSalePYX(uint256 _pyx) external {
1264 ADDRESSES.PYX_TOKEN.burn(msg.sender, _pyx);
1265
1266 totalForSalePYX = totalForSalePYX.add(_pyx);
```

Source file

ETHAutostaking.sol

Locations

```
1062 * the PYX is purchased from pancakeswap for the market price and then staked.
1063 */
1064 contract ETHAutoStaking is AccessControl, IAutoStaking {
1065 using SafeMath for uint256;
1066
1067 struct Settings {
1068 uint256 MIN_AUTO_STAKE_STEPS; // 90
1069 uint256 STAKE_BONUS; // 20
1070 uint256 INFLATION_RATE; // 12
1071 uint256 INFLATION_RATE_DIVIDER; // 364
1072 uint256 STEP_SECONDS; // 86400
1073 }
1074
1075 struct Addresses {
1076 IPYXToken PYX_TOKEN;
1077 IPYXStaking PYX_STAKING;
1078 IUniswapV2Router02 UNISWAP;
1079 address RECIPIENT;
1080 }
1081
1082 event ETHStake(
1083 address indexed account,
1084 uint256 indexed eth,
1085 uint256 indexed buyBackPYX,
1086 uint256 pyx
1087 );
1088
1089 event AddForSalePYX(
1090 address indexed account,
1091 uint256 indexed pyx,
1092 uint256 indexed totalForSalePYX
1093 );
1094
1095 event AddInflatedForSalePYX(
1096 address indexed account,
1097 uint256 indexed pyx,
1098 uint256 indexed totalForSalePYX
1099 );
1100
1101 event WithdrawSlippagePYX(
1102 uint256 indexed pyx,
1103 address indexed recipient,
1104 address indexed account
1105 );
1106
```

```
event UpdateSettings(
bytes32 indexed setting,
uint256 indexed newValue,
address indexed caller
);

// constants
bytes32 public constant SETTER_ROLE = keccak256('SETTER_ROLE');
bytes32 public constant PYX_ADDER_ROLE = keccak256('PYX_ADDER_ROLE');
bytes32 public constant SETTINGS_MANAGER_ROLE =
keccak256('SETTINGS_MANAGER_ROLE');

// settings
Settings public SETTINGS;
Addresses public ADDRESSES;
// numDay => open
mapping(uint256 => bool) public IS_OPEN_OF;

// states
uint256 public totalForSalePYX;
uint256 public totalSoldPYX;

uint256 public totalSlippagePYX;
uint256 public withdrawnSlippagePYX;

modifier onlySetter() {
require(
hasRole(SETTER_ROLE, msg.sender),
'ETHAutoStaking: Caller is not a setter'
);
_;
}

modifier onlyPYXAdder() {
require(
hasRole(PYX_ADDER_ROLE, msg.sender),
'ETHAutoStaking: Caller is not a PYX adder'
);
_;
}

modifier onlySettingsManager() {
require(
hasRole(SETTINGS_MANAGER_ROLE, msg.sender),
'ETHAutoStaking: Caller is not a settings manager'
);
_;
}

constructor() public {
_setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
_setupRole(SETTER_ROLE, msg.sender);
}

function init(
uint256 _minimumAutoStakeSteps,
uint256 _stakeBonus,
uint256 _inflationRate,
uint256 _inflationRateDivider,
uint256 _stepSeconds,
address _pyxToken,
address _pyxStaking,
address _uniswap,
```

```
address _recipient,
address[] calldata _pyxAdderAccounts
) external onlySetter {
SETTINGS.MIN_AUTO_STAKE_STEPS = _minimumAutoStakeSteps;
SETTINGS.STAKE_BONUS = _stakeBonus;
SETTINGS.INFLATION_RATE = _inflationRate;
SETTINGS.INFLATION_RATE_DIVIDER = _inflationRateDivider;
SETTINGS.STEP_SECONDS = _stepSeconds;

IS_OPEN_OF[6] = true; // wednesday
IS_OPEN_OF[1] = true; // friday
IS_OPEN_OF[3] = true; // sunday

ADDRESSES.PYX_TOKEN = IPYXToken(_pyxToken);
ADDRESSES.PYX_STAKING = IPYXStaking(_pyxStaking);
ADDRESSES.UNISWAP = IUniswapV2Router02(_uniswap);
ADDRESSES.RECIPIENT = _recipient;

for (
uint256 idx = 0;
idx < _pyxAdderAccounts.length;
idx = idx.add(1)
) {
_setupRole(PYX_ADDER_ROLE, _pyxAdderAccounts[idx]);
}

renounceRole(SETTER_ROLE, msg.sender);
}

function ethStake(uint256 _stakeSteps, uint256 _pyxToGet) external payable {
require(
_stakeSteps >= SETTINGS.MIN_AUTO_STAKE_STEPS,
'ETHAutoStaking[ethStake]: _stakeSteps < SETTINGS.MIN_AUTO_STAKE_STEPS'
);

uint256 numDayInWeek = getNumDayInWeek();
require(
IS_OPEN_OF[numDayInWeek],
'ETHAutoStaking[ethStake]: staking closed'
);

uint256 pyxAvailableForSale = totalForSalePYX.sub(totalSoldPYX);
require(
_pyxToGet <= pyxAvailableForSale,
'ETHAutoStaking[ethStake]: _pyxToGet > pyxAvailableForSale'
);

// use eth to buy from uniswap
uint256 buyBackPYX = _buyBack(msg.value, _pyxToGet);

// calculate fees and ref bonus
uint256 slippagePYX = buyBackPYX.sub(_pyxToGet);

// transfer the slippage token to the recipient
if (slippagePYX > 0) {
totalSlippagePYX = totalSlippagePYX.add(slippagePYX);
}

// state - update
totalSoldPYX = totalSoldPYX.add(_pyxToGet);

ADDRESSES.PYX_STAKING.contractStake(
msg.sender,
```

```
_stakeSteps,
_pyxToGet,
_pyxToGet,
SETTINGS.STAKE_BONUS
);

_addInflatedForSalePYX(_stakeSteps, _pyxToGet);

// [event]
emit ETHStake(msg.sender, msg.value, buyBackPYX, _pyxToGet);
}

function withdrawSlippagePYX() external {
uint256 slippagePYXLeft = totalSlippagePYX.sub(withdrawnSlippagePYX);
require(
slippagePYXLeft > 0,
'ETHAutoStaking[withdrawSlippagePYX]: slippagePYXLeft <= 0'
);
ADDRESSES.PYX_TOKEN.mint(ADDRESSES.RECIPIENT, slippagePYXLeft);

withdrawnSlippagePYX = withdrawnSlippagePYX.add(slippagePYXLeft);

// [event]
emit WithdrawSlippagePYX(
slippagePYXLeft,
ADDRESSES.RECIPIENT,
msg.sender
);
}

function addForSalePYX(uint256 _pyx) external {
ADDRESSES.PYX_TOKEN.burn(msg.sender, _pyx);

totalForSalePYX = totalForSalePYX.add(_pyx);

// [event]
emit AddForSalePYX(msg.sender, _pyx, totalForSalePYX);
}

/* settings */
function addStakedDay(uint256 _day) external onlySettingsManager {
IS_OPEN_OF[_day] = true;
emit UpdateSettings('IS_OPEN_OF:add', _day, msg.sender);
}

function removeStakedDay(uint256 _day) external onlySettingsManager {
delete IS_OPEN_OF[_day];
emit UpdateSettings('IS_OPEN_OF:remove', _day, msg.sender);
}

function setMinAutoStakeSteps(uint256 _steps) external onlySettingsManager {
SETTINGS.MIN_AUTO_STAKE_STEPS = _steps;
emit UpdateSettings('MIN_AUTO_STAKE_STEPS', _steps, msg.sender);
}

function setStakeBonus(uint256 _bonus) external onlySettingsManager {
SETTINGS.STAKE_BONUS = _bonus;
emit UpdateSettings('STAKE_BONUS', _bonus, msg.sender);
}

function setInflationRate(uint256 _inflationRate)
external
onlySettingsManager
```

```
{
SETTINGS.INFLATION_RATE = _inflationRate;
emit UpdateSettings('INFLATION_RATE', _inflationRate, msg.sender);
}

function setInflationRateDivider(uint256 _inflationRateDivider)
external
onlySettingsManager
{
SETTINGS.INFLATION_RATE_DIVIDER = _inflationRateDivider;
emit UpdateSettings(
'INFLATION_RATE_DIVIDER',
_inflationRateDivider,
msg.sender
);
}

/* end settings */

function addInflatedForSalePYX(uint256 _stakeSteps, uint256 _pyx)
external
override
onlyPYXAdder
{
_addInflatedForSalePYX(_stakeSteps, _pyx);
}

function _addInflatedForSalePYX(uint256 _stakeSteps, uint256 _pyx) private {
uint256 inflatedPYX = getInflatedPYXAmount(_stakeSteps, _pyx);
totalForSalePYX = totalForSalePYX.add(inflatedPYX);

// [event]
emit AddInflatedForSalePYX(msg.sender, inflatedPYX, totalForSalePYX);
}

function getInflatedPYXAmount(uint256 _stakeSteps, uint256 _pyx)
public
view
returns (uint256)
{
return
(_pyx.mul(_stakeSteps).mul(SETTINGS.INFLATION_RATE)).div(
SETTINGS.INFLATION_RATE_DIVIDER.mul(100)
);
}

/** timestamp 0 = 00:00:00 UTC Thursday, 1 January 1970
* 0 - Thursday
* 1 - Friday
* 2 - Saturday
* 3 - Sunday
* 4 - Monday
* 5 - Tuesday
* 6 - Wednesday
*/
function getNumDayInWeek() public view returns (uint256) {
return (block.timestamp / SETTINGS.STEP_SECONDS) % 7;
}

function _buyBack(uint256 _eth, uint256 _pyxOutMin)
private
returns (uint256)
{
```

```
1359  uint256 deadline = block.timestamp.add(uint256(60).mul(30)); // + 30 minutes
1360
1361  address[] memory path = new address[](2);
1362  path[0] = ADDRESSES.UNISWAP.WETH();
1363  path[1] = address(ADDRESSES.PYX_TOKEN);
1364
1365  ADDRESSES.UNISWAP.swapExactETHForTokens{value: _eth}(
1366  _pyxOutMin,
1367  path,
1368  address(this),
1369  deadline
1370  );
1371
1372  uint256 buyBackPYX = ADDRESSES.PYX_TOKEN.getBalanceOf(address(this));
1373
1374  // pyx - burn
1375  ADDRESSES.PYX_TOKEN.burn(address(this), buyBackPYX);
1376
1377  return buyBackPYX;
1378  }
1379  }
```