MythX

REPORT 615D0D822626FC001837E9F1

Created	Wed Oct 06 2021 02:44:18 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	615cfffe43f2c3497112e999

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
00c0751a-b79f-45bb-a4cd-ee3d83835051	PumpkinToken.sol	13

Analysis 00c0751a-b79f-45bb-a4cd-ee3d83835051 MythX

Started	Wed Oct 06 2021 02:44:23 GMT+0000 (Coordinated Universal Time)
Finished	Wed Oct 06 2021 03:29:25 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	PumpkinToken.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	13

ISSUES

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
PumpkinToken.sol
Locations

```
7
8
9  pragma solidity >=0.6.0 <0.8.0;
10
11 /**
```

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
PumpkinToken.sol
Locations

```
220 }
221
222 pragma solidity >=0.6.2 <0.8.0;
223
224 /**
```

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

PumpkinToken.sol

Locations

```
409
410
411 pragma solidity >=0.6.0 <0.8.0;
412
413 /*
```

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

PumpkinToken.sol

Locations

```
433
434
435 pragma solidity >=0.6.0 <0.8.0;
436
437 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.4.0"". 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

PumpkinToken.sol

Locations

```
500
501
502 pragma solidity >=0.4.0;
503
504 interface IBEP20 {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.5.0"". 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

PumpkinToken.sol

Locations

```
597 }
598
599 pragma solidity ^0.5.0;
600
601 interface IUniswapV2ERC20 {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.5.0"". 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

PumpkinToken.sol

Locations

```
621 }
622
623 pragma solidity ^0.5.0;
624
625 interface IUniswapV2Pair {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.5.0"". 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

PumpkinToken.sol

Locations

```
674 }
675
676 pragma solidity ^0.5.0;
677
678 interface IUniswapV2Factory {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.4.0"". 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

PumpkinToken.sol

Locations

```
693
694
695 pragma solidity ^0.4.0;
696
697 /**
```

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

PumpkinToken.sol

Locations

```
1017 }
1018
1019 pragma solidity ^0.6.2;
1020
1021 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

PumpkinToken.sol

Locations

```
1114
1115
1116 pragma solidity ^0.6.2;
1117
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

PumpkinToken.sol

Locations

```
1602  returns (uint256)
1603  {
1604  require(blockNumber < block.number, "PUMPKIN::getPriorVotes: not yet determined");
1605
1606  uint32 nCheckpoints = numCheckpoints[account];
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

PumpkinToken.sol

Locations

```
1675  internal
1676  {
1677  uint32 blockNumber = safe32(block.number, "PUMPKIN::_writeCheckpoint: block number exceeds 32 bits");
1678
1679  if (nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber) {
```