

REPORT 6195B6F3E8463F00194F9D16

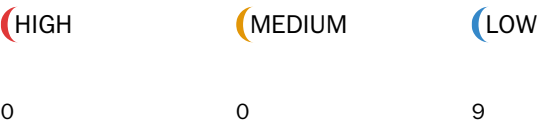
Created Thu Nov 18 2021 02:14:11 GMT+0000 (Coordinated Universal Time)
Number of analyses 1
User 5f50e9c4f992e6001848d9db

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
a39b913e-89cd-4b66-9686-d7074f4b7161	bevy.sol	9

Started	Thu Nov 18 2021 02:14:16 GMT+0000 (Coordinated Universal Time)
Finished	Thu Nov 18 2021 02:59:19 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	Bevy.sol

DETECTED VULNERABILITIES



ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is `""^0.7.0""`. 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

bevy.sol

Locations

```
5 | // SPDX-License-Identifier: MIT
6 |
7 | pragma solidity ^0.7.0;
8 |
9 | /*
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is `""^0.7.0""`. 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

bevy.sol

Locations

```
29 |
30 |
31 | pragma solidity ^0.7.0;
32 |
33 | /**
```

LOW

A floating pragma is set.

SWC-103

The current pragma Solidity directive is `""^0.7.0""`. 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

bevy.sol

Locations

```
188 |
189 |
190 | pragma solidity ^0.7.0
191 |
192 | /**
```

LOW

A floating pragma is set.

SWC-103

The current pragma Solidity directive is `""^0.7.0""`. 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

bevy.sol

Locations

```
265 |
266 |
267 | pragma solidity ^0.7.0
268 |
269 | /**
```

LOW

A floating pragma is set.

SWC-103

The current pragma Solidity directive is `""^0.7.0""`. 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

bevy.sol

Locations

```
430 |
431 |
432 | pragma solidity ^0.7.0
433 |
434 | // Due to compiling issues, _name, _symbol, and _decimals were removed
```

LOW

A floating pragma is set.

SWC-103

The current pragma Solidity directive is `""^0.7.0""`. 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

bevy.sol

Locations

```
711 |
712 |
713 | pragma solidity ^0.7.0
714 |
715 | /**
```

LOW

A floating pragma is set.

SWC-103

The current pragma Solidity directive is `"">=0.6.7""`. 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

bevy.sol

Locations

```
1025 |
1026 |
1027 | pragma solidity >=0.6.7
1028 |
1029 | interface AggregatorV3Interface {
```

LOW

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

SWC-120

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

bevy.sol

Locations

```
1637 | */
1638 | function getPriorVotes(address account, uint blockNumber) public view returns (uint96) {
1639 |     require(blockNumber < block.number, "BEVY::getPriorVotes: not yet determined");
1640 |
1641 |     uint32 nCheckpoints = numCheckpoints[account];
```

LOW

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

SWC-120

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

bevy.sol

Locations

```
1693 |  
1694 | function _writeCheckpoint(address voter, uint32 nCheckpoints, uint96 oldVotes, uint96 newVotes) internal {  
1695 |     uint32 blockNumber = safe32(block.number, "BEVY::_writeCheckpoint: block number exceeds 32 bits");  
1696 |  
1697 |     if (nCheckpoints > 0 && checkpoints[voter][nCheckpoints - 1].fromBlock == blockNumber) {
```