MythX

# REPORT 61F78D844C951D0019A26336

| | |
|---|---|
| Created | Mon Jan 31 2022 07:19:32 GMT+0000 (Coordinated Universal Time) |
| Number of analyses | 1 |
| User | 61f52e351fd393a0c51a34fe |

## REPORT SUMMARY

| Analyses ID | Main source file | Detected vulnerabilities |
|---|---|---|
| b0d5eb1c-7dd6-429e-bb76-821039942e70 | ve.sol | 13 |

Analysis b0d5eb1c-7dd6-429e-bb76-821039942e70 MythX

| | |
|---|---|
| Started | Mon Jan 31 2022 07:19:41 GMT+0000 (Coordinated Universal Time) |
| Finished | Mon Jan 31 2022 08:05:35 GMT+0000 (Coordinated Universal Time) |
| Mode | Deep |
| Client Tool | Remythx |
| Main Source File | Ve.Sol |

## DETECTED VULNERABILITIES

| HIGH | MEDIUM | LOW |
|---|---|---|
| 0 | 0 | 13 |

## ISSUES

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

ve.sol

Locations

```
token = token_addr;
voter = msg.sender;
point_history[0].blk = block.number;
point_history[0].ts = block.timestamp;

```

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

ve.sol

Locations

```
_addTokenTo(_to, _tokenId);
// Set the block of ownership transfer (for Flash NFT protection)
ownership_change[_tokenId] = block.number;
// Log the transfer
emit Transfer(_from, _to, _tokenId);
```

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

`ve.sol`

Locations

```
}

Point memory last_point = Point({bias: 0, slope: 0, ts: block.timestamp, blk: block.number});
if (_epoch > 0) {
last_point = point_history[_epoch];
```

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

`ve.sol`

Locations

```
uint block_slope = 0; // dblock/dt
if (block.timestamp > last_point.ts) {
block_slope = (MULTIPLIER * (block.number - last_point.blk)) / (block.timestamp - last_point.ts);
}
// If last point is already recorded in this block, slope=0
```

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

`ve.sol`

Locations

```
_epoch += 1;
if (t_i == block.timestamp) {
last_point.blk = block.number;
break;
} else {
```

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

`ve.sol`

Locations

```
user_point_epoch[_tokenId] = user_epoch;
u_new.ts = block.timestamp;
u_new.blk = block.number;
user_point_history[_tokenId][user_epoch] = u_new;
}
```

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

`ve.sol`

Locations

```

function block_number() external view returns (uint) {
return block.number;
}

```

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

`ve.sol`

Locations

```

function balanceOfNFT(uint _tokenId) external view returns (uint) {
if (ownership_change[_tokenId] == block.number) return 0;
return _balanceOfNFT(_tokenId, block.timestamp);
}
```

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

`ve.sol`

Locations

```
// Copying and pasting totalSupply code because Vyper cannot pass by
// reference yet
assert(_block <= block.number);

// Binary search
```

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

`ve.sol`

Locations

```
d_t = point_1.ts - point_0.ts;
} else {
d_block = block.number - point_0.blk;
d_t = block.timestamp - point_0.ts;
}
```

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

`ve.sol`

Locations

```
/// @return Total voting power at `_block`
function totalSupplyAt(uint _block) external view returns (uint) {
assert(_block <= block.number);
uint _epoch = epoch;
uint target_epoch = _find_block_epoch(_block, _epoch);
```

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

`ve.sol`

Locations

```
}
} else {
if (point.blk != block.number) {
dt = ((_block - point.blk) * (block.timestamp - point.ts)) / (block.number - point.blk);
}
```

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

`ve.sol`

Locations

```
} else {
if (point.blk != block.number) {
dt = ((_block - point.blk) * (block.timestamp - point.ts)) / (block.number - point.blk);
}
}
```