# CroVegas - Security Review


06.07.2025

Conducted by:

**Kann**, Lead Security Researcher

# Table of Contents

# 1 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

# 2 Risk classification

| Severity | Impact: High | Impact: Medium | Impact: Low |
|---|---|---|---|
| Likelihood: High | Critical | High | Medium |
| Likelihood: Medium | High | Medium | Low |
| Likelihood: Low | Medium | Low | Low |

## 2.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

## 2.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

# 3 Executive summary

**Overview**

| | |
|---|---|
| Project Name | CroVegas |
| Repository | 0xba5b3AA6ED9f5dCE991F53a07D446FF1B5B8E589 |
| Commit hash | |
| Resolution | Fixed |
| Documentation | |
| Methods | Manual review |

**Scope**

| |
|---|
| /src/MultiTokenLottery v2 |

**Issues Found**

| | |
|---|---|
| Critical risk | 0 |
| High risk | 0 |
| Medium risk | 0 |
| Low risk | 0 |
| Informational | 2 |

# 4 Findings

## 4.1 Informational

### 4.1.1 Protocol May Inefficiently Trigger Randomness Already Generated by Witnet

**Severity:** *Informational*

**Description:** The protocol currently interacts with the globally deployed WitnetRandomness contract on Cronos to obtain randomness via the requestRandomness() function. Internally, this calls WITNET_RANDOMNESS.randomize, which requires sending a small amount of wei to trigger randomness generation.

However, since this contract is shared and accessible by everyone, another actor might have already triggered randomize() for the same block, making the additional call by the protocol unnecessary and wasteful.

Before calling WITNET_RANDOMNESS.randomize(), the protocol should first check whether randomness for the target block has already been generated using:

if (!WITNET_RANDOMNESS.isRandomized(pool.randomizingBlock)) { WITNET_RANDOMNESS.randomize{value: X}(); // Only call if not already done }

This conditional check helps avoid redundant randomize() calls, saving wei and gas.

**Resolution:** Acknowledged

### 4.1.2 removeWhitelistedToken() Does Not Update Whitelist Array

**Severity:** *Informational*

**Description:** When a token is removed via removeWhitelistedToken(), it is correctly marked as no longer whitelisted in the mapping:

```
whitelistedTokens[token] = false;
```

However, the token is not removed from the whitelistedTokenList array. This leads to an inconsistent state:

The array still contains the token

But the mapping says it's no longer whitelisted

**Resolution:** Fixed