Introduction

A security review of the GumV1 protocol, done by Backseats focusing on the gas and security aspects of their smart contract.

Disclaimer

This security review is not a guarantee of a lack of vulnerabilities. It is my best effort within the time spent to identify, reproduce, and report errors and improvements that can be made to the project or protocol. As always, the more eyes on the project the better so I encourage others to give this protocol a security review as well.

About Backseats

Backseats is the founder of Backseats Studio and Solidity developer. We help people build and launch products and protocols. He is also the founder of ContractReader.io, a site to read and understand smart contracts that includes live data right inline with the code to give you a better picture of what's happening onchain. You can reach out to him on Twitter @backseats.

About GumV1

GumV1 is a protocol developed by Outland Art that allows anyone to bet on the future potential of an artist on their platform. By creating a `Network`, others can `subscribe` and enjoy earnings from the `pool` of that network. Additionally, the final subscriber can claim the entire `pool` if the `block.timestamp >= network.nextDrawing`.

Observations

`GumV1` uses `Ownable`, to set fee percentages, fee destinations, and time needed for the last subscriber to claim the `pool`. Usage is minimal.

`GumV1` is non-upgradeable so should a new contract be deployed, it would be `GumV2` and would potentially read existing data, if applicable, from the `GumV1` contract while expanding the features or patching any issues rolled out in the V2.

Security Roles and Actors

`GumV1` is deployed using an `initialOwner` address that controls the protocol. The team has told me that the deployer and owner will be different addresses.

Anyone can call `subscribe` with another person's address, presumably the artist's wallet in question to start a `Network` for them and become a fan. However this is an open protocol so you can `subscribe` to any address that is not the `msg.sender`.

Severity Classification

- Critical – Leading to sigificant financial loss and reputational risk if attacked
- High – A very bad but not entirely devastating vector of attack
- Medium – Significant but again, not devastating
- Low – Minimal damage to be done by exploiting

Scope

The `GumV1.sol` file was in scope of this audit.

`Ownable.sol` and `ReentrancyGuard.sol` were not in scope as they've been rigorously audited by OpenZeppelin and the Solidity community.

Findings

The following issues were found, categorized by severity:

- Critical: 0 issues
- High: 1 issue
- Medium: 0 issues
- Low: 3 issues

ID	Title	Severity	Status
[H-01]	`payable` function does not check `msg.value`	High	Fixed
[L-01]	Ensure `setFeeDestination` isn't used with the 0 address	Low	Fixed
[L-02]	`setWinTimeSpan` should have note about how to set units	Low	Acknowledged
[L-03]	Incorrect operator: `>=` used when `==` will do	Low	Fixed

[H-01] `payable` function does not check `msg.value`

`upgradeToLeader` was a `payable` function but nowhere in the function was `msg.value` checked.

[L-01] Ensure `setFeeDestination` isn't used with the 0 address

Low probability of issue here as the team will likely set these on deploy but bears calling out so that this address is not ever set to the 0 address.

[L-02] `setWinTimeSpan` should have note about how to set units

The current comment is confusing and not sufficient for properly setting the value and may lead to the protocol not behaving as intended. Since it's being compared added to the `block.timestamp` in various functions, it's expected to be a UNIX timestamp amount. For example, a value of `3600` would be 1 hour. `86_400` would be 1 day and `604_800` would be 1 week.

[L-03] Incorrect operator: `>=` used when `==` will do

The `subscribe` function originally had a `require` that checked if `msg.value >= price + protocolFee + subjectFee`. `==` is the appropriate operator here since the protocol shouldn't accept a larger amount than called for to satisfy the various fees. That would lead to the user paying more than necessary, fixing it is basic good UX practice.