Art Gobblers contest
Findings & Analysis Report

2022-12-09

Table of contents

- Overview
  - About C4
  - Wardens
- Summary
- Scope
- Severity Criteria
- High Risk Findings (1)
  - [H-01] Can Recover Gobblers Burnt In Legendary Mint
- Medium Risk Findings (3)
  - [M-01] Possible centralization issue around RandProvider
  - [M-02] The reveal process could brick if `randProvider` stops working
  - [M-03] Wrong `balanceOf` user after minting legendary gobbler
- Low Risk and Non-Critical Issues
  - Summary
  - 01 Don't roll your own crypto

- 02 Chainlink's VRF V1 is deprecated
- 03 Rinkeby is not supported for Chainlink's VRF
- 04 Missing checks for `address(0x0)` when assigning values to `address` state variables
- 05 `requestId` is always zero
- 06 Don't use periods with fragments
- 07 Contract implements interface without extending the interface
- 08 `public` functions not called by the contract should be declared `external` instead
- 09 `constant` s should be defined rather than using magic numbers
- 10 Use a more recent version of solidity
- 11 Use a more recent version of solidity
- 12 Constant redefined elsewhere
- 13 Variable names that consist of all capital letters should be reserved for `constant` / `immutable` variables
- 14 File is missing NatSpec
- 15 NatSpec is incomplete
- 16 Event is missing `indexed` fields
- 17 Not using the named return variables anywhere in the function is confusing
- 18 Duplicated `require()` / `revert()` checks should be refactored to a modifier or function

- Gas Optimizations

  - Gas Optimizations Summary
  - G-01 Save gas by not updating seed
  - G-02 State variables only set in the constructor should be declared `immutable`
  - G-03 Avoid contract existence checks by using solidity version 0.8.10 or later

- G-04 State variables should be cached in stack variables rather than re-reading them from storage
- G-05 Multiple accesses of a mapping/array should use a local variable cache
- G-06 `<x> += <y>` costs more gas than `<x> = <x> + <y>` for state variables
- G-07 `<array>.length` should not be looked up in every loop of a `for`-loop
- G-08 Optimize names to save gas
- G-09 Using `bool`s for storage incurs overhead
- G-10 Use a more recent version of solidity
- G-11 `++i` costs less gas than `i++`, especially when it's used in `for`-loops (`--i` / `i--` too)
- G-12 Using `private` rather than `public` for constants, saves gas
- G-13 Division by two should use bit shifting
- G-14 Stack variable used as a cheaper cache for a state variable is only used once
- G-15 Use custom errors rather than `revert()` / `require()` strings to save gas
- G-16 Functions guaranteed to revert when called by normal users can be marked `payable`

- Disclosures

Overview

About C4

Code4rena (C4) is an open organization consisting of security researchers, auditors, developers, and individuals with domain expertise in smart contracts.

A C4 audit contest is an event in which community participants, referred to as Wardens, review, audit, or analyze smart contract logic in exchange for a bounty provided by sponsoring projects.

During the audit contest outlined in this document, C4 conducted an analysis of the Art Gobblers smart contract system written in Solidity. The audit contest took place between September 20—September 27 2022.

Wardens

117 Wardens contributed reports to the Art Gobblers contest:

1. IIIIIII
2. rbserver
3. 0xSmartContract
4. minhtrng
5. zzykxx
6. wagmi
7. Lambda
8. CertoraInc (egjlmn1, OriDabush, ItayG, shakedwinder, and RoiEvenHaim)
9. 0x52
10. arcoun
11. RaymondFam
12. philogy
13. bin2chen
14. hansfriese
15. cccz
16. ladboy233
17. m9800
18. pedroais
19. ronnyx2017
20. auditor0517
21. hyh
22. KIntern_NA (TrungOre and duc)
23. pauliax

24. wastewa
25. shung
26. 8olidity
27. berndartmueller
28. devtooligan
29. obront
30. tonisives
31. zkhorse (karmacoma and horsefacts)
32. __141345__
33. Deivitto
34. ReyAdmirado
35. bytehat
36. imare
37. zdhu
38. Ch_301
39. csanuragjain
40. 0x1f8b
41. 0xNazgul
42. aviggiano
43. catchup
44. djxploit
45. pfapostol
46. Tadashi
47. V_B (Barichek and vlad_bochok)
48. ElKu
49. Atarpara
50. Deathstore
51. gogo
52. MiloTruck

53. SnowMan
54. 0x4non
55. 0x5rings
56. 0xdeadbeef
57. 0xRobocop
58. a12jmx
59. asutorufos
60. Aymen0909
61. B2
62. B353N
63. bharg4v
64. brgltd
65. bulej93
66. c3phas
67. ch0bu
68. CodingNameKiki
69. cryptonue
70. cryptphi
71. delfin454000
72. durianSausage
73. eighty
74. erictee
75. exd0tpy
76. fatherOfBlocks
77. Funen
78. giovannidisiena
79. ignacio
80. JC
81. JohnnyTime

82. Kresh
83. lukris02
84. malinariy
85. martin
86. Noah3o6
87. oyc_109
88. pedr02b2
89. prasantgupta52
90. RockingMiles (robee and pants)
91. Rolezn
92. rotcivegaf
93. rvierdiiev
94. sach1r0
95. simon135
96. Sm4rty
97. SuldaanBeegsi
98. tnevler
99. Tomio
100. TomJ
101. Waze
102. zzzitron
103. yixxas
104. 0xsanson
105. ak1
106. Amithuddar
107. Chom
108. joestakey
109. throttle

This contest was judged by Alex the Entreprenerd.

Final report assembled by liveactionllama.

Summary

The C4 analysis yielded an aggregated total of 4 unique vulnerabilities. Of these vulnerabilities, 1 received a risk rating in the category of HIGH severity and 3 received a risk rating in the category of MEDIUM severity.

Additionally, C4 analysis included 96 reports detailing issues with a risk rating of LOW severity or non-critical. There were also 22 reports recommending gas optimizations.

All of the issues presented here are linked back to their original finding.

Scope

The code under review can be found within the C4 Art Gobblers contest repository, and is composed of 22 smart contracts written in the Solidity programming language and includes 1,498 lines of Solidity code.

Severity Criteria

C4 assesses the severity of disclosed vulnerabilities according to a methodology based on OWASP standards.

Vulnerabilities are divided into three primary risk categories: high, medium, and low/non-critical.

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious Input Handling
- Escalation of privileges
- Arithmetic
- Gas use

Further information regarding the severity criteria referenced throughout the submission review process, please refer to the documentation provided on the C4

website.

High Risk Findings (1)

[H-01] Can Recover Gobblers Burnt In Legendary Mint

Submitted by philogy, also found by auditor0517, bin2chen, cccz, hansfriese, hyh, KIntern_NA, ladboy233, m9800, pauliax, pedroais, ronnyx2017, wagmi, wastewa, and zzykxx

ArtGobblers.sol#L432
ArtGobblers.sol#L890

Allows users to mint legendary Gobblers for free assuming they have the necessary amount of Gobblers to begin with. This is achieved by "reviving" sacrificed Gobblers after having called `mintLegendaryGobbler`.

Severity Justification

This vulnerability allows the violation of the fundamental mechanics of in-scope contracts, allowing buyers to purchase legendary Gobblers at almost no cost outside of temporary liquidity requirements which can be reduced via the use of NFT flashloans.

Proof of Concept

Add the following code to the `ArtGobblersTest` contract in `test/ArtGobblers.t.sol` and run the test via `forge test --match-test testCanReuseSacrificedGobblers -vvv`:

```
    function testCanReuseSacrificedGobblers() public {
            address user = users[0];

            // setup legendary mint
            uint256 startTime = block.timestamp + 30 days;
            vm.warp(startTime);
            mintGobblerToAddress(user, gobblers.LEGENDARY_AUCTION_IN
            uint256 cost = gobblers.legendaryGobblerPrice();
            assertEq(cost, 69);
```

```
        setRandomnessAndReveal(cost, "seed");

        for (uint256 curId = 1; curId <= cost; curId++) {
                ids.push(curId);
                assertEq(gobblers.ownerOf(curId), users[0]);
        }

        // do token approvals for vulnerability exploit
        vm.startPrank(user);
        for (uint256 i = 0; i < ids.length; i++) {
                gobblers.approve(user, ids[i]);
        }
        vm.stopPrank();

        // mint legendary
        vm.prank(user);
        uint256 mintedLegendaryId = gobblers.mintLegendaryGobble

        // confirm user owns legendary
        assertEq(gobblers.ownerOf(mintedLegendaryId), user);

        // show that contract initially thinks tokens are burnt
        for (uint256 i = 0; i < ids.length; i++) {
                hevm.expectRevert("NOT_MINTED");
                gobblers.ownerOf(ids[i]);
        }

        // "revive" burnt tokens by transferring from zero addre
        // which was not reset
        vm.startPrank(user);
        for (uint256 i = 0; i < ids.length; i++) {
                gobblers.transferFrom(address(0), user, ids[i]);
                assertEq(gobblers.ownerOf(ids[i]), user);
        }
        vm.stopPrank();
    }
```

Recommended Mitigation Steps

Ensure token ownership is reset in the for-loop of the `mintLegendaryGobbler` method. Alternatively to reduce the gas cost of `mintLegendaryGobbler` by saving on the approval deletion, simply check the `from` address in `transferFrom`, revert if it's `address(0)`. Note that the latter version would also require changing the

`getApproved` view method such that it checks the owner of the token and returns the zero-address if the owner is zero, otherwise the `getApproved` method would return the old owner after the underlying Gobbler was sacrificed.

FrankieIsLost (Art Gobblers) confirmed and commented:

> Great find. Agree with severity. Here is the fix:
> https://github.com/artgobblers/art-gobblers/pull/151

Alex the Entreprenerd (judge) commented:

> The Warden has shown how, because the approvals field was not cleared, an owner could "bring back a Gobbler from the dead", allowing them to mint legendary Gobblers for free.

> The sponsor has mitigated the issue by clearing the `getApproved` field.

> Because this finding breaks protocol invariants, and would allow to sidestep the cost to mint a Legendary Gobbler, I agree with High Severity.

Medium Risk Findings (3)

[M-01] Possible centralization issue around RandProvider

Submitted by minhtrng, also found by 0xSmartContract, IIIIIII, and rbserver

ArtGobblers.sol#L560-L567

While it is very common for web3 projects to have privileged functions that can only be called by an admin address, special thought should be given to functions that can break core functionality of a project.

One such function is `ArtGobblers.upgradeRandProvider()`. If this is called passing a non-compatible contract address or EOA, requesting new seeds will be bricked and as a consequence reveals will not be possible. Also the seed could be controlled using a custom contract which would allow a malicious actor to set seeds in his favor.

Naturally, the assumption that the deployer or a multisig (likely that ownership will probably be transferred to such) go rogue and perform a malicious action is unlikely to happen as they have a stake in the project (monetary and reputation wise).

However, as this is a project that will be unleashed on launch without any further development and left to form its own ecosystem for many years to come, less centralized options should be considered.

Proof of Concept

The function `ArtGobblers.upgradeRandProvider()`, allows the owner to arbitrarily pass a RandProvider with the only restriction being that there is currently no seed requested from the current RandProvider:

```
    function upgradeRandProvider(RandProvider newRandProvider) exter
        // Revert if waiting for seed, so we don't interrupt rec
        if (gobblerRevealsData.waitingForSeed) revert SeedPendir

        randProvider = newRandProvider; // Update the randomness

        emit RandProviderUpgraded(msg.sender, newRandProvider);
    }
```

The RandProvider is the only address eligible (as well as responsible) to call `ArtGobblers.acceptRandomSeed()`, which is required to perform reveals of minted Gobblers:

```
    function acceptRandomSeed(bytes32, uint256 randomness) external
        // The caller must be the randomness provider, revert ir
        if (msg.sender != address(randProvider)) revert NotRandF

        // The unchecked cast to uint64 is equivalent to moduloi
        gobblerRevealsData.randomSeed = uint64(randomness); // (

        gobblerRevealsData.waitingForSeed = false; // We have th

        emit RandomnessFulfilled(randomness);
    }
```

This could be abused with the consequences outlined above.

Recommended Mitigation Steps

The inclusion of a voting and governance mechanism should be considered for protocol critical functions. This could for example take the form of each Gobbler representing 1 vote, with legendary Gobblers having more weight (literally) based on the amount of consumed Gobblers.

Alex the Entreprenerd (judge) commented:

> The warden has shown a few possible risks for end users, because of the privileged function `upgradeRandProvider`, a malicious owner could set the `randProvider` to either a malicious implementation or a faulty implementation.
>
> This would prevent reveals which, as we know from other findings could cause the inability to mint legendary gobblers with non-zero emission factors.
>
> Because this is contingent on a malicious Admin, which could deny reveals and hence deny a key aspect of the protocol, I believe Medium Severity to be appropriate

FrankieIsLost (Art Gobblers) disagreed with severity and commented:

> We disagree with severity. This type of griefing attack by admin does not provide any economic benefits. Additionally, there doesn't seem to be any viable alternatives here, as introducing a governance system just to upgrade the rand provider (which should only happen once or twice during the lifetime of the project) seems like overkill

Alex the Entreprenerd (judge) commented:

> I agree with the Sponsors unwillingness to create a complex system to maintain the VRF provider.
>
> I also must concede that griefing the mint is of dubious economic benefit.
>
> However, per our rules and historical context I believe this is an example of Admin Privilege, the Admin can change the implementation of the Randomness Provider

> to their advantage and can deny the mint from continuing.

> I have to agree with a nofix, beside recommending the use of a strong Multisig to avoid any issues in the future.

> However, the in-scope system has no way of:
>
> - Ensuring a multisig will be used
> - Ensuring that a randomness provider which is fair is going to be used.

> C4 has historically flagged these type of risks as Medium, and for those reasons I believe the correct judgement is Medium Severity.

> We will be discussing these types of findings to provide consistent and transparent judging rules in the future, however, given the historical track record of C4, I believe the right move is to keep it as Medium Severity.

[M-02] The reveal process could brick if `randProvider` stops working

Submitted by zzykxx, also found by 8olidity, berndartmueller, bin2chen, bytehat, devtooligan, hansfriese, IIIIIII, imare, Lambda, obront, philogy, shung, tonisives, zdhu, and zkhorse

It could become impossible to reveal gobblers which leads any new minted gobbler to have an `emissionMultiple` of `0` forever, preventing them from minting any goo.

Proof of Concept

In `ArtGobblers.sol` calling `requestRandomSeed()` sets `gobblerRevealsData.waitingForSeed = true`, which makes both `revealGobblers()` and `upgradeRandProvider()` revert.

The only way to set `gobblerRevealsData.waitingForSeed = false` is by calling `acceptRandomSeed()` which can only be called by the `randProvider` itself.

This means that if `randProvider` stops working and `requestRandomSeed()` is called (which sets `gobblerRevealsData.waitingForSeed = true`) there is no way for `upgradeRandProvider()` and `revealGobblers()` to be ever called again.

Copy the test in `RandProvider.t.sol` and run it with `forge test -m testRandomnessBrick`:

```
    function testRandomnessBrick() public {
        vm.warp(block.timestamp + 1 days);
        mintGobblerToAddress(users[0], 1);

        bytes32 requestId = gobblers.requestRandomSeed();

        /*
            At this point we should have
            vrfCoordinator.callBackWithRandomness(requestId, randomr

            But that doesn't happen because we are assuming
            randProvider stopped responding
        /*

        /*
            Can't reveal gobblers
        */
        vm.expectRevert(ArtGobblers.SeedPending.selector);
        gobblers.revealGobblers(1);

        /*
            Can't update provider
        */
        RandProvider newRandProvider = new ChainlinkV1RandProvider(
            ArtGobblers(address(gobblers)),
            address(vrfCoordinator),
            address(linkToken),
            keyHash,
            fee
        );

        vm.expectRevert(ArtGobblers.SeedPending.selector);
        gobblers.upgradeRandProvider(newRandProvider);
    }
```

Recommended Mitigation Steps

A potential fix is to reset some variables in `upgradeRandProvider` instead of reverting:

```
function upgradeRandProvider(RandProvider newRandProvider) exter
    if (gobblerRevealsData.waitingForSeed) {
        gobblerRevealsData.waitingForSeed = false;
        gobblerRevealsData.toBeRevealed = 0;
        gobblerRevealsData.nextRevealTimestamp = uint64(nextReve
    }

    randProvider = newRandProvider; // Update the randomness prc

    emit RandProviderUpgraded(msg.sender, newRandProvider);
}
```

This gives a little extra powers to the owner, but I don't think it's a big deal considering that he can just plug in any provider he wants anyway.

Alex the Entreprenerd (judge) commented:

> The Warden has highlighted a risk with the state handling that concerns receiving randomness from the `randProvider`. As detailed, if no callback is performed, this would put the contract in a state where the faulty provider could not be replaced.

> While no specific way for the provider to be bricked was given, if we assume that the provider was retired or deprecated, the contract may fall in such a situation.

> Because this is contingent on an externality, but would deny a core functionality of the protocol, I agree with Medium Severity.

FrankieIsLost (Art Gobblers) commented:

> Agree, thanks. Fixed here: https://github.com/artgobblers/art-gobblers/pull/154

[M-03] Wrong `balanceOf` user after minting legendary gobbler

Submitted by wagmi, also found by 0x52, arcoun, CertoraInc, Lambda, RaymondFam, and zzykxx

[ArtGobblers.sol#L458]

In `ArtGobblers.mintLegendaryGobbler()` function, line 458 calculates the number of gobblers user owned after minting.

```
// We subtract the amount of gobblers burned, and then add 1 to
getUserData[msg.sender].gobblersOwned = uint32(getUserData[msg.s
```

It added 1 to factor in the new legendary. But actually, this new legendary is accounted in `_mint()` function already

```
function _mint(address to, uint256 id) internal {
    // Does not check if the token was already minted or the rec
    // because ArtGobblers.sol manages its ids in such a way tha
    // double mint and will only mint to safe addresses or msg.s

    unchecked {
        ++getUserData[to].gobblersOwned;
    }

    getGobblerData[id].owner = to;

    emit Transfer(address(0), to, id);
}
```

So the result is `gobblersOwned` is updated incorrectly. And `balanceOf()` will return wrong value.

Proof of Concept

Script modified from `testMintLegendaryGobbler()`

```
    function testMintLegendaryGobbler() public {
        uint256 startTime = block.timestamp + 30 days;
        vm.warp(startTime);
        // Mint full interval to kick off first auction.
        mintGobblerToAddress(users[0], gobblers.LEGENDARY_AUCTION_IN
        uint256 cost = gobblers.legendaryGobblerPrice();
        assertEq(cost, 69);
        setRandomnessAndReveal(cost, "seed");
        uint256 emissionMultipleSum;
        for (uint256 curId = 1; curId <= cost; curId++) {
            ids.push(curId);
            assertEq(gobblers.ownerOf(curId), users[0]);
            emissionMultipleSum += gobblers.getGobblerEmissionMultip
        }

        assertEq(gobblers.getUserEmissionMultiple(users[0]), emissio

        uint256 beforeSupply = gobblers.balanceOf(users[0]);
        vm.prank(users[0]);
        uint256 mintedLegendaryId = gobblers.mintLegendaryGobbler(id

        // Check balance
        assertEq(gobblers.balanceOf(users[0]), beforeSupply - cost +
    }
```

Tools Used

Foundry

Recommended Mitigation Steps

Consider remove adding 1 when calculating `gobblersOwned`

```
    getUserData[msg.sender].gobblersOwned = uint32(getUserData[msg.s
```

transmissions11 (Art Gobblers) commented:

> Great find!

FrankieIsLost (Art Gobblers) confirmed and commented:

> Good find, fixed here: https://github.com/artgobblers/art-gobblers/pull/153

Alex the Entreprenerd (judge) commented:

> The warden has demonstrated an accounting issue in the system, the Sponsor has mitigated.

> Because the finding is valid and the behaviour shown diverges from the intended one, without a severe risk of loss, I agree with Medium Severity.

Low Risk and Non-Critical Issues

For this contest, 96 reports were submitted by wardens detailing low risk and non-critical issues. The report highlighted below by IIIIIII received the top score from the judge.

The following wardens also submitted reports: __141345__, 0x1f8b, 0x4non, 0x5rings, 0xdeadbeef, 0xNazgul, 0xRobocop, 0xSmartContract, a12jmx, arcoun, asutorufos, aviggiano, Aymen0909, B2, B353N, bharg4v, bin2chen, brgltd, bulej93, c3phas, catchup, cccz, CertoraInc, Ch_301, ch0bu, CodingNameKiki, cryptonue, cryptphi, csanuragjain, Deivitto, delfin454000, devtooligan, djxploit, durianSausage, eighty, erictee, exd0tpy, fatherOfBlocks, Funen, giovannidisiena, hansfriese, ignacio, JC, JohnnyTime, Kresh, ladboy233, Lambda, lukris02, malinariy, martin, Noah3o6, oyc_109, pedr02b2, pedroais, pfapostol, philogy, prasantgupta52, rbserver, ReyAdmirado, RockingMiles, Rolezn, ronnyx2017, rotcivegaf, rvierdiiev, sach1r0, shung, simon135, Sm4rty, SuldaanBeegsi, Tadashi, tnevler, Tomio, TomJ, tonisives, V_B, wagmi, Waze, zkhorse, zzykxx, zzzitron, yixxas, 0x52, 0xsanson, 8olidity, ak1, Amithuddar, berndartmueller, Chom, ElKu, joestakey, m9800, minhtrng, obront, RaymondFam, *and* throttle.

Summary

	Issue	Instances
[01]	Don't roll your own crypto	1
[02	Chainlink's VRF V1 is deprecated	1

	Issue	Instances
]		
[03]	Rinkeby is not supported for Chainlink's VRF	1
[04]	Missing checks for `address(0x0)` when assigning values to `address` state variables	10
[05]	`requestId` is always zero	1
[06]	Don't use periods with fragments	1
[07]	Contract implements interface without extending the interface	1
[08]	`public` functions not called by the contract should be declared `external` instead	1
[09]	`constant` s should be defined rather than using magic numbers	112
[10]	Use a more recent version of solidity	2
[11]	Use a more recent version of solidity	2
[12]	Constant redefined elsewhere	8
[13]	Variable names that consist of all capital letters should be reserved for `constant` / `immutable` variables	3
[14]	File is missing NatSpec	2
[15]	NatSpec is incomplete	10
[16]	Event is missing `indexed` fields	16
[17]	Not using the named return variables anywhere in the function is confusing	3
[18]	Duplicated `require()` / `revert()` checks should be refactored to a modifier or function	2

[01] Don't roll your own crypto

The general advice when it comes to cryptography is **don't roll your own crypto**. The Chainlink VRF best practices page says that in order to get multiple values from a single random value, one should **hash a counter along with the random value**. This project does not follow that guidance, and instead recursively hashes previous

hashes. Logically, if you have a source of entropy, then truncate, and hash again, over and over, you're going to lose entropy if there's a weakness in the hashing function, and every hashing function will eventually be found to have a weakness. Unless there is a very good reason not to, the code should follow the best practice Chainlink outlines. Furthermore, the current scheme wastes gas updating the random seed in every function call, as is outlined in my separate gas report.

There is 1 instance of this issue:

```
File: /src/ArtGobblers.sol

664                 // Update the random seed to choose a new c
665                 // It is critical that we cast to uint64 he
666                 // set after calling revealGobblers(1) thri
667                 // after calling revealGobblers(3) a single
668                 // to choose from a number of different see
669                 // Equivalent to randomSeed = uint64(uint25
670                 assembly {
671                     mstore(0, randomSeed) // Store the ranc
672
673                     // Moduloing by 1 << 64 (2 ** 64) is ec
674                     randomSeed := mod(keccak256(0, 32), shl
675                 }
676             }
677
678             // Update all relevant reveal state.
679:            gobblerRevealsData.randomSeed = uint64(randomSe
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L664-L679

[02] Chainlink's VRF V1 is deprecated

VRF V1 is deprecated, so new projects should not use it

There is 1 instance of this issue:

```
File: /src/utils/rand/ChainlinkV1RandProvider.sol
```

```
13   /// @notice RandProvider wrapper around Chainlink VRF v1.
14:  contract ChainlinkV1RandProvider is RandProvider, VRFConsum
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/rand/ChainlinkV1RandProvider.sol#L13-L14

[03] Rinkeby is not supported for Chainlink's VRF

Rinkeby is deprecated and not listed as supported for Chainlink. The documentation states `Once the Ethereum mainnet transitions to proof-of-stake, Rinkeby will no longer be an accurate staging environment for mainnet. ... Developers who currently use Rinkeby as a staging/testing environment should prioritize migrating to Goerli or Sepolia` link

There is 1 instance of this issue:

```
File: /script/deploy/DeployRinkeby.s.sol

6:   contract DeployRinkeby is DeployBase {
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/script/deploy/DeployRinkeby.s.sol#L6

[04] Missing checks for `address(0x0)` when assigning values to `address` state variables

There are 10 instances of this issue:

```
File: src/ArtGobblers.sol

316:          team = _team;

317:          community = _community;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L316

```
File: script/deploy/DeployBase.s.sol

49:             teamColdWallet = _teamColdWallet;

52:             vrfCoordinator = _vrfCoordinator;

53:             linkToken = _linkToken;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/script/deploy/DeployBase.s.sol#L49

```
File: src/Pages.sol

181:             community = _community;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Pages.sol#L181

```
File: src/Goo.sol

83:             artGobblers = _artGobblers;

84:             pages = _pages;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Goo.sol#L83

```
File: lib/solmate/src/auth/Owned.sol

30:             owner = _owner;
```

```
40:           owner = newOwner;
```

https://github.com/transmissions11/solmate/blob/265728027431 01f160f2d07556edfc162896115e/src/auth/Owned.sol#L30

[05] `requestId` is always zero

Chainlink suggests to have a unique `requestId` for every separate randomness request. By always using the same value, it's not possible to tell whether Chainlink returned data for a valid request, or if there was some Chainlink bug that triggered a callback for a request that was never made

There is 1 instance of this issue:

```
File: /src/utils/rand/ChainlinkV1RandProvider.sol

62        function requestRandomBytes() external returns (bytes32
63            // The caller must be the ArtGobblers contract, rev
64            if (msg.sender != address(artGobblers)) revert NotG
65
66:           emit RandomBytesRequested(requestId);
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/rand/ChainlinkV1RandProvider.sol#L62-L66

[06] Don't use periods with fragments

Throughout the files, most of the comments have fragments that end with periods. They should either be converted to actual sentences with both a noun phrase and a verb phrase, or the periods should be removed.

There is 1 instance of this issue:

```
File: /src/ArtGobblers.sol
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol

[07] Contract implements interface without extending the interface

Not extending the interface may lead to the wrong function signature being used, leading to unexpected behavior. If the interface is in fact being implemented, use the `override` keyword to indicate that fact

There is 1 instance of this issue:

```
File: src/utils/token/GobblersERC1155B.sol

/// @audit IERC1155MetadataURI.balanceOf(), IERC1155MetadataURI.
8:    abstract contract GobblersERC1155B {
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC1155B.sol#L8

[08] `public` functions not called by the contract should be declared `external` instead

Contracts are allowed to override their parents' functions and change the visibility from `external` to `public`.

There is 1 instance of this issue:

```
File: lib/goo-issuance/src/LibGOO.sol

17        function computeGOOBalance(
18            uint256 emissionMultiple,
19            uint256 lastBalanceWad,
20            uint256 timeElapsedWad
21:       ) public pure returns (uint256) {
```

https://github.com/transmissions11/goo-issuance/blob/648e65e66e43ff5c19681427e300ece9c0df1437/src/LibGOO.sol#L17-L21

[09] `constant`s should be defined rather than using magic numbers

Even assembly can benefit from using readable constants instead of hex/numeric literals

There are 112 instances of this issue. For full details, please see the warden's original submission.

[10] Use a more recent version of solidity

Use a solidity version of at least 0.8.13 to get the ability to use `using for` with a list of free functions

There are 2 instances of this issue:

```
File: src/Pages.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Pages.sol#L2

```
File: lib/goo-issuance/src/LibGOO.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/transmissions11/goo-issuance/blob/648e65e66e43ff5c19681427e300ece9c0df1437/src/LibGOO.sol#L2

[11] Use a more recent version of solidity

Use a solidity version of at least 0.8.4 to get `bytes.concat()` instead of `abi.encodePacked(<bytes>,<bytes>)` Use a solidity version of at least 0.8.12 to get `string.concat()` instead of `abi.encodePacked(<str>,<str>)`

There are 2 instances of this issue:

```
File: src/ArtGobblers.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L2

```
File: script/deploy/DeployRinkeby.s.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/script/deploy/DeployRinkeby.s.sol#L2

[12] Constant redefined elsewhere

Consider defining in only one contract so that values cannot become out of sync when only one location is updated. A cheap way to store constants in a single location is to create an `internal constant` in a `library`. If the variable is a local cache of another contract's value, consider making the cache variable internal or private, which will require external users to query the contract with the source of truth, so that callers don't get out of sync.

There are 8 instances of this issue:

```
File: src/Pages.sol
```

```
/// @audit seen in src/ArtGobblers.sol
86:       Goo public immutable goo;

/// @audit seen in src/ArtGobblers.sol
89:       address public immutable community;

/// @audit seen in src/ArtGobblers.sol
103:      uint256 public immutable mintStart;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Pages.sol#L86

```
File: src/utils/rand/ChainlinkV1RandProvider.sol

/// @audit seen in src/utils/token/PagesERC721.sol
20:       ArtGobblers public immutable artGobblers;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/rand/ChainlinkV1RandProvider.sol#L20

```
File: script/deploy/DeployRinkeby.s.sol

/// @audit seen in src/Pages.sol
11:       uint256 public immutable mintStart = 1656369768;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/script/deploy/DeployRinkeby.s.sol#L11

```
File: src/Goo.sol

/// @audit seen in src/utils/rand/ChainlinkV1RandProvider.sol
64:       address public immutable artGobblers;

/// @audit seen in src/ArtGobblers.sol
67:       address public immutable pages;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Goo.sol#L64

```
File: src/utils/GobblerReserve.sol

/// @audit seen in src/Goo.sol
18:       ArtGobblers public immutable artGobblers;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/GobblerReserve.sol#L18

[13] Variable names that consist of all capital letters should be reserved for `constant` / `immutable` variables

If the variable needs to be different based on which class it comes from, a `view` / `pure` *function* should be used instead (e.g. like this).

There are 3 instances of this issue:

```
File: src/ArtGobblers.sol

136:      string public UNREVEALED_URI;

139:      string public BASE_URI;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L136

```
File: src/Pages.sol

96:       string public BASE_URI;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Pages.sol#L96

[14] File is missing NatSpec

There are 2 instances of this issue:

```
File: script/deploy/DeployBase.s.sol
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/script/deploy/DeployBase.s.sol

```
File: script/deploy/DeployRinkeby.s.sol
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/script/deploy/DeployRinkeby.s.sol

[15] NatSpec is incomplete

There are 10 instances of this issue:

```
File: src/ArtGobblers.sol

/// @audit Missing: '@param _pages'
278       /// @notice Sets VRGDA parameters, mint config, releva
279       /// @param _merkleRoot Merkle root of mint mintlist.
280       /// @param _mintStart Timestamp for the start of the \
281       /// @param _goo Address of the Goo contract.
282       /// @param _team Address of the team reserve.
283       /// @param _community Address of the community reserve
284       /// @param _randProvider Address of the randomness prc
285       /// @param _baseUri Base URI for revealed gobblers.
286       /// @param _unrevealedUri URI for unrevealed gobblers.
287       constructor(
288           // Mint config:
```

```
289             bytes32 _merkleRoot,
290             uint256 _mintStart,
291             // Addresses:
292             Goo _goo,
293             Pages _pages,
294             address _team,
295             address _community,
296             RandProvider _randProvider,
297             // URIs:
298             string memory _baseUri,
299             string memory _unrevealedUri
300         )
301             GobblersERC721("Art Gobblers", "GOBBLER")
302             Owned(msg.sender)
303             LogisticVRGDA(
304                 69.42e18, // Target price.
305                 0.31e18, // Price decay percent.
306                 // Max gobblers mintable via VRGDA.
307                 toWadUnsafe(MAX_MINTABLE),
308                 0.0023e18 // Time scale.
309:            )

/// @audit Missing: '@param bytes32'
544         /// @notice Callback from rand provider. Sets randomSe
545         /// @param randomness The 256 bits of verifiable randc
546:        function acceptRandomSeed(bytes32, uint256 randomness)

/// @audit Missing: '@return'
691         /// @notice Returns a token's URI if it has been minte
692         /// @param gobblerId The id of the token to get the UF
693:        function tokenURI(uint256 gobblerId) public view virtu

/// @audit Missing: '@return'
755         /// @notice Calculate a user's virtual goo balance.
756         /// @param user The user to query balance for.
757:        function gooBalance(address user) public view returns

/// @audit Missing: '@return'
837         /// @dev Gobblers minted to reserves cannot comprise m
838         /// the supply of goo minted gobblers and the supply c
839:        function mintReservedGobblers(uint256 numGobblersEach)

/// @audit Missing: '@return'
864         /// @notice Convenience function to get emissionMultip
865         /// @param gobblerId The gobbler to get emissionMultip
866:        function getGobblerEmissionMultiple(uint256 gobblerId)
```

```
/// @audit Missing: '@return'
870       /// @notice Convenience function to get emissionMultip
871       /// @param user The user to get emissionMultiple for.
872:      function getUserEmissionMultiple(address user) externa
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L278-L309

```
File: src/Pages.sol

/// @audit Missing: '@return'
237       /// @dev Pages minted to the reserve cannot comprise m
238       /// supply of goo minted pages and the supply of pages
239:      function mintCommunityPages(uint256 numPages) external

/// @audit Missing: '@return'
263       /// @notice Returns a page's URI if it has been minted
264       /// @param pageId The id of the page to get the URI fo
265:      function tokenURI(uint256 pageId) public view virtual
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Pages.sol#L237-L239

```
File: lib/goo-issuance/src/LibGOO.sol

/// @audit Missing: '@return'
15        /// @param lastBalanceWad The last checkpointed balanc
16        /// @param timeElapsedWad The time elapsed since the l
17        function computeGOOBalance(
18            uint256 emissionMultiple,
19            uint256 lastBalanceWad,
20            uint256 timeElapsedWad
21:       ) public pure returns (uint256) {
```

https://github.com/transmissions11/goo-issuance/blob/648e65e66e43ff5c19681427e300ece9c0df1437/src/LibGOO.sol#

L15-L21

[16] Event is missing `indexed` fields

Index event fields make the field more quickly accessible to off-chain tools that parse events. However, note that each index field costs extra gas during emission, so it's not necessarily best to index the maximum allowed per event (three fields). Each `event` should use three `indexed` fields if there are three or more fields, and gas usage is not particularly of concern for the events in question. If there are fewer than three fields, all of the fields should be indexed.

There are 16 instances of this issue:

```
File: src/ArtGobblers.sol

229:      event GooBalanceUpdated(address indexed user, uint256

232:      event GobblerPurchased(address indexed user, uint256 i

233:      event LegendaryGobblerMinted(address indexed user, uir

234:      event ReservedGobblersMinted(address indexed user, uir

236:      event RandomnessFulfilled(uint256 randomness);

237:      event RandomnessRequested(address indexed user, uint25

240:      event GobblersRevealed(address indexed user, uint256 r
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L229

```
File: lib/solmate/src/tokens/ERC721.sol

15:       event ApprovalForAll(address indexed owner, address ir
```

https://github.com/transmissions11/solmate/blob/26572802743101f160f2d07556edfc162896115e/src/tokens/ERC721.sol#L15

```
File: src/utils/token/GobblersERC1155B.sol

29:       event ApprovalForAll(address indexed owner, address ir

31:       event URI(string value, uint256 indexed id);
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC1155B.sol#L29

```
File: src/utils/token/GobblersERC721.sol

17:       event ApprovalForAll(address indexed owner, address ir
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC721.sol#L17

```
File: src/utils/token/PagesERC721.sol

18:       event ApprovalForAll(address indexed owner, address ir
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/PagesERC721.sol#L18

```
File: src/Pages.sol

134:      event PagePurchased(address indexed user, uint256 inde

136:      event CommunityPagesMinted(address indexed user, uint2
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Pages.sol#L134

```
File: src/utils/rand/RandProvider.sol

13:         event RandomBytesRequested(bytes32 requestId);

14:         event RandomBytesReturned(bytes32 requestId, uint256 r
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/rand/RandProvider.sol#L13

[17] Not using the named return variables anywhere in the function is confusing

Consider changing the variable to be an unnamed one

There are 3 instances of this issue:

```
File: src/utils/token/GobblersERC1155B.sol

/// @audit owner
55:         function ownerOf(uint256 id) public view virtual retur
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC1155B.sol#L55

```
File: src/utils/token/PagesERC721.sol

/// @audit isApproved
72:         function isApprovedForAll(address owner, address opera
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/PagesERC721.sol#L72

```
File: src/utils/rand/ChainlinkV1RandProvider.sol
```

```
/// @audit requestId
62:       function requestRandomBytes() external returns (bytes3
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/rand/ChainlinkV1RandProvider.sol#L62

[18] Duplicated `require()` / `revert()` checks should be refactored to a modifier or function

The compiler will inline the function, which will avoid `JUMP` instructions usually associated with functions

There are 2 instances of this issue:

```
File: src/ArtGobblers.sol

705:          if (gobblerId < FIRST_LEGENDARY_GOBBLER_ID) revert
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L705

```
File: lib/solmate/src/tokens/ERC721.sol

158:          require(to != address(0), "INVALID_RECIPIENT");
```

https://github.com/transmissions11/solmate/blob/265728027431o1f160f2d07556edfc162896115e/src/tokens/ERC721.sol#L158

Alex the Entreprenerd (judge) commented:

> **[01] Don't roll your own crypto**
> `Refactoring` for this specific case as I'm not convinced the entropy to be misused. Recommend following up with sponsor if you find further info.

[02] Chainlink's VRF V1 is deprecated

`Refactoring` , as system enables swapping.

[03] Rinkeby is not supported for Chainlink's VRF

`Refactoring` , nice find

[04] Missing checks for address(0x0) when assigning values to address state variables

`Low`

[05] requestId is always zero

`Non-critical`

[06] Don't use periods with fragments

I don't have an opinion about this one

[07] Contract implements interface without extending the interface

`Refactoring`

[08] public functions not called by the contract should be declared external instead

`Refactoring`

[09] constants should be defined rather than using magic numbers

`Refactoring`

[10] Use a more recent version of solidity

`Refactoring`

[12] Constant redefined elsewhere

This one looks off as those are immutable and not known until deploy time.

[13] Variable names that consist of all capital letters should be reserved for constant/immutable variables

`Refactoring`

> **[14] File is missing NatSpec**
>
> `Non-Critical`

> **[15] NatSpec is incomplete**
>
> `Non-Critical`

> **[16] Event is missing indexed fields**
>
> I still don't get why you'd index "gibberish" values if it doesn't even save gas.

> **[17] Not using the named return variables anywhere in the function is confusing**
>
> `Refactoring`

> **[18] Duplicated require()/revert() checks should be refactored to a modifier or function**
>
> `Refactoring`

> Overall, pretty good report with some interesting ideas. Of the automated ones, definitely the best.

Gas Optimizations

For this contest, 22 reports were submitted by wardens detailing gas optimizations. The report highlighted below by IIIIIII received the top score from the judge.

The following wardens also submitted reports: __141345__, V_B, 0x1f8b, 0xNazgul, 0xSmartContract, Atarpara, aviggiano, catchup, CertoraInc, Deathstore, Deivitto, djxploit, ElKu, gogo, MiloTruck, pfapostol, philogy, ReyAdmirado, shung, SnowMan, *and* Tadashi.

Gas Optimizations Summary

	Issue	Instances	Total Gas Saved
[G-01]	Save gas by not updating seed	1	2897
[G-02]	State variables only set in the constructor should be declared `immutable`	7	12582

	Issue	Instances	Total Gas Saved
[G-03]	Avoid contract existence checks by using solidity version 0.8.10 or later	11	1100
[G-04]	State variables should be cached in stack variables rather than re-reading them from storage	4	388
[G-05]	Multiple accesses of a mapping/array should use a local variable cache	25	1050
[G-06]	`<x> += <y>` costs more gas than `<x> = <x> + <y>` for state variables	2	226
[G-07]	`<array>.length` should not be looked up in every loop of a `for` -loop	2	6
[G-08]	Optimize names to save gas	10	220
[G-09]	Using `bool` s for storage incurs overhead	5	68400
[G-10]	Use a more recent version of solidity	22	-
[G-11]	`++i` costs less gas than `i++` , especially when it's used in `for` -loops (`--i` / `i--` too)	9	45
[G-12]	Using `private` rather than `public` for constants, saves gas	18	-
[G-13]	Division by two should use bit shifting	1	20
[G-14]	Stack variable used as a cheaper cache for a state variable is only used once	2	6
[G-15]	Use custom errors rather than `revert()` / `require()` strings to save gas	36	-
[G-16]	Functions guaranteed to revert when called by normal users can be marked `payable`	3	63

Total: 158 instances over 16 issues with **87003 gas** saved

Gas totals use lower bounds of ranges and count two iterations of each `for` -loop. All values above are runtime, not deployment, values; deployment values are listed in the individual issue descriptions

[G-01] Save gas by not updating seed

The code that determines a set of random values based on the seed does not follow Chainlink's stated [best practice](#) for doing so. By updating the seed rather than including a counter in what's hashed (e.g. `currentId`), the code incurs an unnecessary Gsreset **2900 gas**.

There is 1 instance of this issue:

File: /src/ArtGobblers.sol

```
664                 // Update the random seed to choose a new c
665                 // It is critical that we cast to uint64 he
666                 // set after calling revealGobblers(1) thri
667                 // after calling revealGobblers(3) a single
668                 // to choose from a number of different see
669                 // Equivalent to randomSeed = uint64(uint25
670                 assembly {
671                     mstore(0, randomSeed) // Store the ranc
672
673                     // Moduloing by 1 << 64 (2 ** 64) is ec
674                     randomSeed := mod(keccak256(0, 32), shl
675:                }
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L664-L675

[G-02] State variables only set in the constructor should be declared `immutable`

Avoids a Gsset (**20000 gas**) in the constructor, and replaces the first access in each transaction (Gcoldsload - **2100 gas**) and each access thereafter (Gwarmacces - **100 gas**) with a `PUSH32` (**3 gas**).

While `string`s are not value types, and therefore cannot be `immutable` / `constant` if not hard-coded outside of the constructor, the same behavior can be achieved by making the current contract `abstract` with `virtual`

functions for the `string` accessors, and having a child contract override the functions with the hard-coded implementation-specific values.

There are 7 instances of this issue:

```
File: src/ArtGobblers.sol

/// @audit UNREVEALED_URI (constructor)
321:           UNREVEALED_URI = _unrevealedUri;

/// @audit UNREVEALED_URI (access)
702:           if (gobblerId <= currentNonLegendaryId) return UNR

/// @audit BASE_URI (constructor)
320:           BASE_URI = _baseUri;

/// @audit BASE_URI (access)
698:               return string.concat(BASE_URI, uint256(getGobb

/// @audit BASE_URI (access)
709:               return string.concat(BASE_URI, gobblerId.toStr
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L321

```
File: src/Pages.sol

/// @audit BASE_URI (constructor)
183:           BASE_URI = _baseUri;

/// @audit BASE_URI (access)
268:           return string.concat(BASE_URI, pageId.toString());
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Pages.sol#L183

[G-03] Avoid contract existence checks by using solidity version 0.8.10 or later

Prior to 0.8.10 the compiler inserted extra code, including `EXTCODESIZE` (**100 gas**), to check for contract existence for external calls. In more recent solidity versions, the compiler will not insert these checks if the external call has a return value.

There are 11 instances of this issue:

```
File: src/ArtGobblers.sol

/// @audit toString()
698:                 return string.concat(BASE_URI, uint256(getGobb
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L698

```
File: lib/solmate/src/tokens/ERC721.sol

/// @audit onERC721Received()
120:                 ERC721TokenReceiver(to).onERC721Received(m

/// @audit onERC721Received()
136:                 ERC721TokenReceiver(to).onERC721Received(m

/// @audit onERC721Received()
198:                 ERC721TokenReceiver(to).onERC721Received(m

/// @audit onERC721Received()
213:                 ERC721TokenReceiver(to).onERC721Received(m
```

https://github.com/transmissions11/solmate/blob/265728027431o1f160f2d07556edfc162896115e/src/tokens/ERC721.sol#L120

```
File: src/utils/token/GobblersERC1155B.sol

/// @audit onERC1155Received()
150:                 ERC1155TokenReceiver(to).onERC1155Receivec
```

```
/// @audit onERC1155BatchReceived()
186:                 ERC1155TokenReceiver(to).onERC1155BatchRec
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC1155B.sol#L150

```
File: src/utils/token/GobblersERC721.sol

/// @audit onERC721Received()
123:                 ERC721TokenReceiver(to).onERC721Received(m

/// @audit onERC721Received()
139:                 ERC721TokenReceiver(to).onERC721Received(m
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC721.sol#L123

```
File: src/utils/token/PagesERC721.sol

/// @audit onERC721Received()
136:                 ERC721TokenReceiver(to).onERC721Received(m

/// @audit onERC721Received()
152:                 ERC721TokenReceiver(to).onERC721Received(m
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/PagesERC721.sol#L136

[G-04] State variables should be cached in stack variables rather than re-reading them from storage

The instances below point to the second+ access of a state variable within a function. Caching of a state variable replaces each Gwarmaccess (**100 gas**) with a much cheaper stack read. Other less obvious fixes/optimizations include having local

memory caches of state variable structs, or having local caches of state variable contracts/addresses.

There are 4 instances of this issue:

```
File: src/ArtGobblers.sol

/// @audit BASE_URI on line 698
709:               return string.concat(BASE_URI, gobblerId.toStr

/// @audit numMintedFromGoo on line 482
493:               uint256 numMintedSinceStart = numMintedFromGoo

/// @audit getGobblerData[swapId].idx on line 615
617:                     : getGobblerData[swapId].idx; // Shuff

/// @audit getGobblerData[currentId].idx on line 623
625:                     : getGobblerData[currentId].idx; // Sh
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L709

[G-05] Multiple accesses of a mapping/array should use a local variable cache

The instances below point to the second+ access of a value inside a mapping/array, within a function. Caching a mapping's value in a local `storage` or `calldata` variable when the value is accessed multiple times, saves **~42 gas per access** due to not having to recalculate the key's keccak256 hash (Gkeccak256 - **30 gas**) and that calculation's associated stack operations. Caching an array's struct avoids recalculating the array offsets into memory/calldata.

There are 25 instances of this issue:

```
File: src/ArtGobblers.sol

/// @audit getGobblerData[id] on line 437
439:                 burnedMultipleTotal += getGobblerData[id].
```

```
/// @audit getGobblerData[id] on line 439
441:                 emit Transfer(msg.sender, getGobblerData[i

/// @audit getUserData[<etc>] on line 454
455:             getUserData[msg.sender].lastTimestamp = uint64

/// @audit getUserData[<etc>] on line 455
456:             getUserData[msg.sender].emissionMultiple += ui

/// @audit getUserData[<etc>] on line 456
/// @audit getUserData[<etc>] on line 458
458:             getUserData[msg.sender].gobblersOwned = uint32

/// @audit getGobblerData[swapId] on line 615
617:                     : getGobblerData[swapId].idx; // Shuff

/// @audit getGobblerData[currentId] on line 620
623:                 uint64 currentIndex = getGobblerData[curre

/// @audit getGobblerData[currentId] on line 623
625:                     : getGobblerData[currentId].idx; // Sh

/// @audit getGobblerData[currentId] on line 625
649:                 getGobblerData[currentId].idx = swapIndex;

/// @audit getGobblerData[currentId] on line 649
650:                 getGobblerData[currentId].emissionMultiple

/// @audit getGobblerData[swapId] on line 617
653:                 getGobblerData[swapId].idx = currentIndex;

/// @audit getUserData[currentIdOwner] on line 660
661:                 getUserData[currentIdOwner].lastTimestamp

/// @audit getUserData[currentIdOwner] on line 661
662:                 getUserData[currentIdOwner].emissionMultip

/// @audit getUserData[user] on line 761
762:             getUserData[user].lastBalance,

/// @audit getUserData[user] on line 762
763:             uint(toDaysWadUnsafe(block.timestamp - getUser

/// @audit getUserData[user] on line 825
826:         getUserData[user].lastTimestamp = uint64(block.tim
```

```
/// @audit getGobblerData[id] on line 885
896:          getGobblerData[id].owner = to;

/// @audit getGobblerData[id] on line 896
899:              uint32 emissionMultiple = getGobblerData[id].e

/// @audit getUserData[from] on line 903
904:              getUserData[from].lastTimestamp = uint64(block

/// @audit getUserData[from] on line 904
905:              getUserData[from].emissionMultiple -= emissior

/// @audit getUserData[from] on line 905
906:              getUserData[from].gobblersOwned -= 1;

/// @audit getUserData[to] on line 910
911:              getUserData[to].lastTimestamp = uint64(block.t

/// @audit getUserData[to] on line 911
912:              getUserData[to].emissionMultiple += emissionMu

/// @audit getUserData[to] on line 912
913:              getUserData[to].gobblersOwned += 1;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L439

[G-06] `<x> += <y>` costs more gas than `<x> = <x> + <y>` for state variables

Using the addition operator instead of plus-equals saves 113 gas.

There are 2 instances of this issue:

```
File: src/ArtGobblers.sol

844:              uint256 newNumMintedForReserves = numMintedFor
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L844

```
File: src/Pages.sol

244:              uint256 newNumMintedForCommunity = numMintedFc
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Pages.sol#L244

[G-07] `<array>.length` should not be looked up in every loop of a `for`-loop

The overheads outlined below are *PER LOOP*, excluding the first loop

- storage arrays incur a Gwarmaccess (**100 gas**)
- memory arrays use `MLOAD` (**3 gas**)
- calldata arrays use `CALLDATALOAD` (**3 gas**)

Caching the length changes each of these to a `DUP<N>` (**3 gas**), and gets rid of the extra `DUP<N>` needed to store the stack offset.

There are 2 instances of this issue:

```
File: src/utils/token/GobblersERC1155B.sol

114:              for (uint256 i = 0; i < owners.length; ++i) {
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC1155B.sol#L114

```
File: src/utils/GobblerReserve.sol
```

```
37:                 for (uint256 i = 0; i < ids.length; i++) {
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/GobblerReserve.sol#L37

[G-08] Optimize names to save gas

`public` / `external` function names and `public` member variable names can be optimized to save gas. See this link for an example of how it works. Below are the interfaces/abstract contracts that can be optimized so that the most frequently-called functions use the least amount of gas possible during method lookup. Method IDs that have two leading zero bytes can save **128 gas** each during deployment, and renaming functions to have lower method IDs will save **22 gas** per call, per sorted position shifted.

There are 10 instances of this issue:

```
File: src/ArtGobblers.sol

/// @audit claimGobbler(), mintFromGoo(), gobblerPrice(), mintLe
83:   contract ArtGobblers is GobblersERC721, LogisticVRGDA, Owr
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L83

```
File: script/deploy/DeployBase.s.sol

/// @audit run()
16:   abstract contract DeployBase is Script {
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/script/deploy/DeployBase.s.sol#L16

```
File: src/Pages.sol

/// @audit mintFromGoo(), pagePrice(), mintCommunityPages()
78:   contract Pages is PagesERC721, LogisticToLinearVRGDA {
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Pages.sol#L78

```
File: src/utils/rand/ChainlinkV1RandProvider.sol

/// @audit requestRandomBytes()
14:   contract ChainlinkV1RandProvider is RandProvider, VRFConsu
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/rand/ChainlinkV1RandProvider.sol#L14

```
File: src/Goo.sol

/// @audit mintForGobblers(), burnForGobblers(), burnForPages()
58:   contract Goo is ERC20("Goo", "GOO", 18) {
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Goo.sol#L58

```
File: lib/VRGDAs/src/VRGDA.sol

/// @audit getVRGDAPrice(), getTargetSaleTime()
10:   abstract contract VRGDA {
```

https://github.com/transmissions11/VRGDAs/blob/f4dec0611641e344339b2c78e5f733bba3b532e0/src/VRGDA.sol#L10

```
File: lib/goo-issuance/src/LibGOO.sol

/// @audit computeGOOBalance()
10:   library LibGOO {
```

https://github.com/transmissions11/goo-issuance/blob/648e65e66e43ff5c19681427e300ece9c0df1437/src/LibGOO.sol#L10

```
File: lib/solmate/src/auth/Owned.sol

/// @audit setOwner()
6:    abstract contract Owned {
```

https://github.com/transmissions11/solmate/blob/265728027431o1f160f2d07556edfc162896115e/src/auth/Owned.sol#L6

```
File: src/utils/GobblerReserve.sol

/// @audit withdraw()
12:   contract GobblerReserve is Owned {
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/GobblerReserve.sol#L12

```
File: src/utils/rand/RandProvider.sol

/// @audit requestRandomBytes()
8:    interface RandProvider {
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/rand/RandProvider.sol#L8

[G-09] Using `bool`s for storage incurs overhead

```
// Booleans are more expensive than uint256 or any type that
// word because each write operation emits an extra SLOAD to
// slot's contents, replace the bits taken up by the boolean
// back. This is the compiler's defense against contract upg
// pointer aliasing, and it cannot be disabled.
```

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/58f635312aa21f947cae5f8578638a85aa2519f5/contracts/security/ReentrancyGuard.sol#L23-L27 Use `uint256(1)` and `uint256(2)` for true/false to avoid a Gwarmaccess (100 gas) for the extra SLOAD, and to avoid Gsset (**20000 gas**) when changing from `false` to `true`, after having been `true` in the past

There are 5 instances of this issue:

```
File: src/ArtGobblers.sol

149:     mapping(address => bool) public hasClaimedMintlistGobb
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L149

```
File: lib/solmate/src/tokens/ERC721.sol

51:     mapping(address => mapping(address => bool)) public is
```

https://github.com/transmissions11/solmate/blob/265728027431o1f160f2d07556edfc162896115e/src/tokens/ERC721.sol#L51

```
File: src/utils/token/GobblersERC1155B.sol

37:     mapping(address => mapping(address => bool)) public is
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC1155B.sol#L37

```
File: src/utils/token/GobblersERC721.sol

77:         mapping(address => mapping(address => bool)) public is
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC721.sol#L77

```
File: src/utils/token/PagesERC721.sol

70:         mapping(address => mapping(address => bool)) internal
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/PagesERC721.sol#L70

[G-10] Use a more recent version of solidity

Use a solidity version of at least 0.8.2 to get simple compiler automatic inlining
Use a solidity version of at least 0.8.3 to get better struct packing and cheaper multiple storage reads
Use a solidity version of at least 0.8.4 to get custom errors, which are cheaper at deployment than `revert()/require()` strings
Use a solidity version of at least 0.8.10 to have external calls skip contract existence checks if the external call has a return value

There are 22 instances of this issue:

```
File: src/ArtGobblers.sol

2:     pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L2

```
File: lib/solmate/src/utils/FixedPointMathLib.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/transmissions11/solmate/blob/265728027431 01f160f2d07556edfc162896115e/src/utils/FixedPointMathLib.sol#L2

```
File: lib/solmate/src/tokens/ERC721.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/transmissions11/solmate/blob/265728027431 01f160f2d07556edfc162896115e/src/tokens/ERC721.sol#L2

```
File: src/utils/token/GobblersERC1155B.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC1155B.sol#L2

```
File: lib/solmate/src/utils/SignedWadMath.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/transmissions11/solmate/blob/265728027431 01f160f2d07556edfc162896115e/src/utils/SignedWadMath.sol#L2

```
File: src/utils/token/GobblersERC721.sol
```

```
2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC721.sol#L2

```
File: src/utils/token/PagesERC721.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/PagesERC721.sol#L2

```
File: script/deploy/DeployBase.s.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/script/deploy/DeployBase.s.sol#L2

```
File: src/Pages.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Pages.sol#L2

```
File: src/utils/rand/ChainlinkV1RandProvider.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/rand/ChainlinkV1RandProvider.sol#L2

```
File: lib/VRGDAs/src/LogisticToLinearVRGDA.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/transmissions11/VRGDAs/blob/f4dec0611641e344339b2c78e5f733bba3b532e0/src/LogisticToLinearVRGDA.sol#L2

```
File: lib/solmate/src/utils/MerkleProofLib.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/transmissions11/solmate/blob/265728027431 01f160f2d07556edfc162896115e/src/utils/MerkleProofLib.sol#L2

```
File: script/deploy/DeployRinkeby.s.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/script/deploy/DeployRinkeby.s.sol#L2

```
File: src/Goo.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Goo.sol#L2

```
File: lib/VRGDAs/src/LogisticVRGDA.sol
```

```
2:    pragma solidity >=0.8.0;
```

https://github.com/transmissions11/VRGDAs/blob/f4dec0611641e344339b2c78e5f733bba3b532e0/src/LogisticVRGDA.sol#L2

```
File: lib/solmate/src/utils/LibString.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/transmissions11/solmate/blob/265728027431 01f160f2d07556edfc162896115e/src/utils/LibString.sol#L2

```
File: lib/VRGDAs/src/VRGDA.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/transmissions11/VRGDAs/blob/f4dec0611641e344339b2c78e5f733bba3b532e0/src/VRGDA.sol#L2

```
File: lib/goo-issuance/src/LibGOO.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/transmissions11/goo-issuance/blob/648e65e66e43ff5c19681427e300ece9c0df1437/src/LibGOO.sol#L2

```
File: lib/solmate/src/auth/Owned.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/transmissions11/solmate/blob/265728027431 01f160f2d07556edfc162896115e/src/auth/Owned.sol#L2

```
File: lib/VRGDAs/src/LinearVRGDA.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/transmissions11/VRGDAs/blob/f4dec0611641e344339b2c78e5f733bba3b532e0/src/LinearVRGDA.sol#L2

```
File: src/utils/GobblerReserve.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/GobblerReserve.sol#L2

```
File: src/utils/rand/RandProvider.sol

2:    pragma solidity >=0.8.0;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/rand/RandProvider.sol#L2

[G-11] `++i` costs less gas than `i++`, especially when it's used in `for`-loops (`--i` / `i--` too)

Saves **5 gas per loop**

There are 9 instances of this issue:

```
File: lib/solmate/src/tokens/ERC721.sol

99:               _balanceOf[from]--;

101:              _balanceOf[to]++;
```

```
164:                _balanceOf[to]++;

179:                _balanceOf[owner]--;
```

https://github.com/transmissions11/solmate/blob/265728027431o1f160f2d07556edfc162896115e/src/tokens/ERC721.sol#L99

```
File: src/utils/token/PagesERC721.sol

115:                _balanceOf[from]--;

117:                _balanceOf[to]++;

181:                _balanceOf[to]++;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/PagesERC721.sol#L115

```
File: src/Pages.sol

251:                for (uint256 i = 0; i < numPages; i++) _mint(c
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Pages.sol#L251

```
File: src/utils/GobblerReserve.sol

37:                 for (uint256 i = 0; i < ids.length; i++) {
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/GobblerReserve.sol#L37

[G-12] Using `private` rather than `public` for constants, saves gas

If needed, the values can be read from the verified contract source code, or if there are multiple values there can be a single getter function that [returns a tuple](#) of the values of all currently-public constants. Saves **3406-3606 gas** in deployment gas due to the compiler not having to create non-payable getter functions for deployment calldata, not having to store the bytes of the value outside of where it's used, and not adding another entry to the method ID table.

There are 18 instances of this issue:

```
File: src/ArtGobblers.sol

112:      uint256 public constant MAX_SUPPLY = 10000;

115:      uint256 public constant MINTLIST_SUPPLY = 2000;

118:      uint256 public constant LEGENDARY_SUPPLY = 10;

122:      uint256 public constant RESERVED_SUPPLY = (MAX_SUPPLY

126       uint256 public constant MAX_MINTABLE = MAX_SUPPLY
127           - MINTLIST_SUPPLY
128           - LEGENDARY_SUPPLY
129:          - RESERVED_SUPPLY;

146:      bytes32 public immutable merkleRoot;

156:      uint256 public immutable mintStart;

177:      uint256 public constant LEGENDARY_GOBBLER_INITIAL_STAR

180:      uint256 public constant FIRST_LEGENDARY_GOBBLER_ID = M

184:      uint256 public constant LEGENDARY_AUCTION_INTERVAL = M
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L112

```
File: src/Pages.sol

103:      uint256 public immutable mintStart;
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/Pages.sol#L103

```
File: script/deploy/DeployRinkeby.s.sol

11:       uint256 public immutable mintStart = 1656369768;

13:       string public constant gobblerBaseUri = "https://testr

14:       string public constant gobblerUnrevealedUri = "https:/

15:       string public constant pagesBaseUri = "https://testnet
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/script/deploy/DeployRinkeby.s.sol#L11

```
File: lib/VRGDAs/src/LogisticVRGDA.sol

20:       int256 public immutable logisticLimit;

25:       int256 public immutable logisticLimitDoubled;
```

https://github.com/transmissions11/VRGDAs/blob/f4dec0611641e344339b2c78e5f733bba3b532e0/src/LogisticVRGDA.sol#L20

```
File: lib/VRGDAs/src/VRGDA.sol

17:       int256 public immutable targetPrice;
```

https://github.com/transmissions11/VRGDAs/blob/f4dec0611641e344339b2c78e5f733bba3b532e0/src/VRGDA.sol#L17

[G-13] Division by two should use bit shifting

`<x> / 2` is the same as `<x> >> 1`. While the compiler uses the `SHR` opcode to accomplish both, the version that uses division incurs an overhead of [20 gas](https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L462) due to `JUMP`s to and from a compiler utility function that introduces checks which can be avoided by using `unchecked {}` around the division by two.

There is 1 instance of this issue:

```
File: src/ArtGobblers.sol

462:                 cost <= LEGENDARY_GOBBLER_INITIAL_START_PR
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L462

[G-14] Stack variable used as a cheaper cache for a state variable is only used once

If the variable is only accessed once, it's cheaper to use the state variable directly that one time, and save the **3 gas** the extra stack assignment would spend.

There are 2 instances of this issue:

```
File: src/ArtGobblers.sol

480:         uint256 startPrice = legendaryGobblerAuctionData.s
481:         uint256 numSold = legendaryGobblerAuctionData.numS
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L480

[G-15] Use custom errors rather than `revert()` / `require()` strings to save gas

Custom errors are available from solidity version 0.8.4. Custom errors save ~50 gas each time they're hit by avoiding having to allocate and store the revert string. Not defining the strings also save deployment gas.

There are 36 instances of this issue:

```
File: src/ArtGobblers.sol

437:                 require(getGobblerData[id].owner == msg.se

885:         require(from == getGobblerData[id].owner, "WRONG_F

887:         require(to != address(0), "INVALID_RECIPIENT");

889          require(
890              msg.sender == from || isApprovedForAll[from][m
891              "NOT_AUTHORIZED"
892:          );
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L437

```
File: lib/solmate/src/tokens/ERC721.sol

36:          require((owner = _ownerOf[id]) != address(0), "NOT

40:          require(owner != address(0), "ZERO_ADDRESS");

69:          require(msg.sender == owner || isApprovedForAll[ow

87:          require(from == _ownerOf[id], "WRONG_FROM");

89:          require(to != address(0), "INVALID_RECIPIENT");

91           require(
92               msg.sender == from || isApprovedForAll[from][m
93               "NOT_AUTHORIZED"
```

```
94:         );

118         require(
119             to.code.length == 0 ||
120                 ERC721TokenReceiver(to).onERC721Received(m
121                 ERC721TokenReceiver.onERC721Received.selec
122             "UNSAFE_RECIPIENT"
123:        );

134         require(
135             to.code.length == 0 ||
136                 ERC721TokenReceiver(to).onERC721Received(m
137                 ERC721TokenReceiver.onERC721Received.selec
138             "UNSAFE_RECIPIENT"
139:        );

158:        require(to != address(0), "INVALID_RECIPIENT");

160:        require(_ownerOf[id] == address(0), "ALREADY_MINTE

175:        require(owner != address(0), "NOT_MINTED");

196         require(
197             to.code.length == 0 ||
198                 ERC721TokenReceiver(to).onERC721Received(m
199                 ERC721TokenReceiver.onERC721Received.selec
200             "UNSAFE_RECIPIENT"
201:        );

211         require(
212             to.code.length == 0 ||
213                 ERC721TokenReceiver(to).onERC721Received(m
214                 ERC721TokenReceiver.onERC721Received.selec
215             "UNSAFE_RECIPIENT"
216:        );
```

https://github.com/transmissions11/solmate/blob/265728027431o1f160f2d07556edfc162896115e/src/tokens/ERC721.sol#L36

File: src/utils/token/GobblersERC1155B.sol

```
107:        require(owners.length == ids.length, "LENGTH_MISMA

149             require(
```

```
150                 ERC1155TokenReceiver(to).onERC1155Receivec
151                     ERC1155TokenReceiver.onERC1155Receivec
152                 "UNSAFE_RECIPIENT"
153:            );

185             require(
186                 ERC1155TokenReceiver(to).onERC1155BatchRec
187                     ERC1155TokenReceiver.onERC1155BatchRec
188                 "UNSAFE_RECIPIENT"
189:            );
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC1155B.sol#L107

```
File: lib/solmate/src/utils/SignedWadMath.sol

142:          require(x > 0, "UNDEFINED");
```

https://github.com/transmissions11/solmate/blob/265728027431o1f160f2d07556edfc162896115e/src/utils/SignedWadMath.sol#L142

```
File: src/utils/token/GobblersERC721.sol

62:           require((owner = getGobblerData[id].owner) != addr

66:           require(owner != address(0), "ZERO_ADDRESS");

95:           require(msg.sender == owner || isApprovedForAll[ow

121           require(
122               to.code.length == 0 ||
123                   ERC721TokenReceiver(to).onERC721Received(m
124                   ERC721TokenReceiver.onERC721Received.selec
125               "UNSAFE_RECIPIENT"
126:          );

137           require(
138               to.code.length == 0 ||
139                   ERC721TokenReceiver(to).onERC721Received(m
140                   ERC721TokenReceiver.onERC721Received.selec
```

```
141                 "UNSAFE_RECIPIENT"
142:            );
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/GobblersERC721.sol#L62

```
File: src/utils/token/PagesERC721.sol

55:             require((owner = _ownerOf[id]) != address(0), "NOT

59:             require(owner != address(0), "ZERO_ADDRESS");

85:             require(msg.sender == owner || isApprovedForAll(ow

103:            require(from == _ownerOf[id], "WRONG_FROM");

105:            require(to != address(0), "INVALID_RECIPIENT");

107             require(
108                 msg.sender == from || isApprovedForAll(from, m
109                 "NOT_AUTHORIZED"
110:            );

135                 require(
136                     ERC721TokenReceiver(to).onERC721Received(m
137                         ERC721TokenReceiver.onERC721Received.s
138                     "UNSAFE_RECIPIENT"
139:                );

151                 require(
152                     ERC721TokenReceiver(to).onERC721Received(m
153                         ERC721TokenReceiver.onERC721Received.s
154                     "UNSAFE_RECIPIENT"
155:                );
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/token/PagesERC721.sol#L55

```
File: lib/VRGDAs/src/VRGDA.sol
```

```
32:             require(decayConstant < 0, "NON_NEGATIVE_DECAY_CON
```

https://github.com/transmissions11/VRGDAs/blob/f4dec0611641e344339b2c78e5f733bba3b532e0/src/VRGDA.sol#L32

```
File: lib/solmate/src/auth/Owned.sol

20:             require(msg.sender == owner, "UNAUTHORIZED");
```

https://github.com/transmissions11/solmate/blob/265728027431o1f160f2d07556edfc162896115e/src/auth/Owned.sol#L20

[G-16] Functions guaranteed to revert when called by normal users can be marked `payable`

If a function modifier such as `onlyOwner` is used, the function will revert if a normal user tries to pay the function. Marking the function as `payable` will lower the gas cost for legitimate callers because the compiler will not include checks for whether a payment was provided. The extra opcodes avoided are `CALLVALUE` (2), `DUP1` (3), `ISZERO` (3), `PUSH2` (3), `JUMPI` (10), `PUSH1` (3), `DUP1` (3), `REVERT` (0), `JUMPDEST` (1), `POP` (2), which costs an average of about **21 gas per call** to the function, in addition to the extra deployment cost.

There are 3 instances of this issue:

```
File: src/ArtGobblers.sol

560:     function upgradeRandProvider(RandProvider newRandProvi
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/ArtGobblers.sol#L560

```
File: lib/solmate/src/auth/Owned.sol
```

```
39:       function setOwner(address newOwner) public virtual onl
```

https://github.com/transmissions11/solmate/blob/265728027431 01f160f2d07556edfc162896115e/src/auth/Owned.sol#L39

```
File: src/utils/GobblerReserve.sol

34:       function withdraw(address to, uint256[] calldata ids)
```

https://github.com/code-423n4/2022-09-artgobblers/blob/d2087c5a8a6a4f1b9784520e7fe75afa3a9cbdbe/src/utils/GobblerReserve.sol#L34

Alex the Entreprenerd (judge) commented:

> **[G-01] Save gas by not updating seed**
> I'm not sure what this finding implies as if we were to re-use the same randomness input and then use a iterator, we'd need to store a new uint of the values we've revealed, incurring still a 5k gas cost for setting that (as we may want to re-use the same randomness value until exhausted)

> I'll give 100 gas, would ask to send a coded test next time to get a better score

> **[G-02] State variables only set in the constructor should be declared immutable**
> 6.3k for the three URI variables

> **[G-03] Avoid contract existence checks by using solidity version 0.8.10 or later**
> Will accept because explained, but will cap at 500
> 500

> **[G-04] State variables should be cached in stack variables rather than re-reading them from storage**
> 300 gas, ignoring the URI as it's in 2 separate mutually exclusive returns

> **[G-05] Multiple accesses of a mapping/array should use a local variable cache**
> Will give 500 in lack of benchmark (Foundry Output), mostly valid but unsure of

> the effective savings

> Rest will give you a ballpark 150 gas saved, better than the average report by presentation and detail, however those are not as high impact as the ones above.

> Overall one of the best reports, I think adding custom benchmarks would make it the best by far.

> 7850

Disclosures

C4 is an open organization governed by participants in the community.

C4 Contests incentivize the discovery of exploits, vulnerabilities, and bugs in smart contracts. Security researchers are rewarded at an increasing rate for finding higher-risk issues. Contest submissions are judged by a knowledgeable security researcher and solidity developer and disclosed to sponsoring developers. C4 does not conduct formal verification regarding the provided code but instead provides final verification.

C4 does not provide any guarantee or warranty regarding the security of this project. All smart contract software should be used at the sole risk and responsibility of users.

Top