SMART CONTRACTS

TABLE OF CONTENTS

ABOUT TRUTH SECURITY

Truth Security Audits conducts a comprehensive security review of blockchain applications, using modern tools and employing only the most experienced solidity experts on the market. During the elaboration of the audit, auditors analyze possible attack vectors both from the project owners and its users and rate them by Severity (see Appendix A) from Informational to Critical as per common reason, giving an adequate explanation in the body of the audit. Our Audits are versioned, and clients get a grace period to alleviate or comment on all of our findings.

METHODOLOGY

As per standard code review practices, we use manual and static analysis. During the manual phase, auditor(s) review source code line-by-line, studying its intended and actual behavior, referencing known vulnerabilities (including SWC Registry https://swcregistry.io/), comparing the code to common contracts, and noting all things that are out of the ordinary for confirmation or rejection. Static analysis refers to a computer-aided analysis of the code, providing automatized and powerful insight into additional subtle issues possibly present in the code.

Some auditors additionally write their own test cases and try to break the contracts in their own local simulated blockchain environment. This is called dynamic analysis and is often employed in the more complex contracts, where consolidated testing scenarios help assess the completeness of contract logic or reversely give a proof-of-concept for potential security vulnerability.

Finally, all auditors do on-chain verification of live contracts. This crucial step confirms contracts were not swapped for malicious only after audit, or that parameters are set based on reasonable expectations. Issues such as non-renounced ownerships are also assessed in this step.

PROJECT DETAILS

Project Name	Animal Farm
Description	Animal Farm is the only platform to have dynamic emission rates based on the current demand for underlying assets, reducing the supply in times of low demand to allow for optimal price performance in all market conditions. On top of this Animal Farm is the only yield aggregating lending protocol which allows users to secure profits in the form of a BUSD & BNB dividend, generated from both liquidity provider fees and low risk lending, without ever needing to sell their AFD (yield asset) or AFP (governance asset) tokens.
Links	https://animalfarm.app/ https://t.me/The_Animal_Farm https://twitter.com/DRIPcommunity
Code Language	Solidity
Chain	Binance Smart Chain

VERSION DETAILS

Version	Based on status at	Published at	Elaborated by	Notes
V	October 28,2022	October 28,2022	October 28,2022	Revised October 30th, 2022

OBJECTS OF REVIEW

In Versions	Source	Contents
V.1		Contracts
		DDSCAPIGS deployed: 0xa92Af2d72d5214206B9c15Ab4c46705BE68eb37B DDSCADOGS deployed: 0x321e03e005f4910ceB29476956084B20cC7B6C61 DogPound deployed: 0x6dA8227Bc7B576781ffCac69437e17b8D4F4aE41 DogPoundLinearPool deployed: 0xf2f4382C8F88f33F78f710354D62B13bA0200954 Stakemanager: 0x25A959dDaEcEb50c1B724C603A57fe7b32eCbEeA DogPoundAutoPool deployed: 0x7993A5A830A158961efF96280538960C875f6807 FeeManager deployed: 0xcdb16E645d34E9c8bd57C6062225B34C2EA5f7a7 FeeManagerDogs deployed: 0x0AB59C36367A5188Ed391342C6c4400e1648CC20 devfeem address: 0xFe44479A11Cf491DA91BE1f3d7b2727Dd2df7424 MC pigs: 0x8536178222fC6Ec5fac49BbfeBd74CA3051c638f MC dogs: 0x78205CE1a7e714CAE95a32e65B6dA7b2dA8D8A10 RewardsvaultBNB: 0x4c004C4fB925Be396F902DE262F2817dEeBC22Ec

SUMMARY

AUDIT PROCEDURE

Auditors	Robert Morgan III & Mark
Audited as	Truth Seekers
Methodology	Static analysis examines code to identify issues within the logic and techniques. This type of analysis addresses weaknesses in source code that might lead to vulnerabilities. Static analysis supports development operations by creating an automated feedback loop. Static review can lead to false positives/negatives which Truth Seekers confirms or denies with penetration testing. Static analysis is being used on Animal Farm because it specifically identifies defects before you run a program (e.g, between coding and unit testing)
Tools	Dynamic analysis finds vulnerabilities in a runtime environment. Automated tools analyze the input and output of an application for potential threats like SQL injection. Tools can also search for other application-specific issues and analyze server configuration errors. The purpose of dynamic analysis is to analyze the program as an attacker would, looking for entry points and vulnerable sections during program execution. Hence, we create a well written suite of manually ran unit and regression tests.

Truth Security Audits has no control over website UI projects provide. Always double check you are signing a contract matching one of the contracts in section Objects of Review.

Truth Security Audits concerns itself exclusively with code quality and smart contract security.
We have not audited tokenomics, nor a general likelihood of making money with this project. Truth Security report is not financial advice.

FINDINGS

Finding ID	DDSCA.sol	Severity	Informational
Type	Gas savings//Masterchef	Status	Resolved
Location	**L11: Gas Savings** **L147: Masterchef**		
Description	**L11**: **Token can be scoped as immutable, as it never changes after being initialized in the constructor** **L147: MasterChef can be changed by contract owner, meaning owner could grant himself permission to alter the emissions for the contract, although emissions are constrained by its upper and lower bounds**		
Recommendation	MasterChef: Resolved		
Alleviation	Developers will put 48 hour delay on Emission Charges.		

Finding ID	DevFeeManager.sol	Severity	Informational / Low
Type	**Best Practices / Gas Error**	Status	Resolved
Location	**L20-27: State Variables** **L75: Potential Out Of Gas Error**		
Description	L20-27: **Make sure to scope all state variables** L75: **Loops are dangerous as the platform scales, if the number of LP tokens grows too large, no one will not be able to call updateEarned() in one transaction.**		
Recommendation	Resolved		
Alleviation	Added token list indexing -		

Finding ID	DogPoundAutoPool.sol	Severity	Informational
Type	Unused Variables / Logic Error	Status	Resolved

Location	L190+L215: Question For Dev: L195-198+L220-223: Best Practices L287-288: Unused Variables
Description	L190 + L215: pigsClaimedTotal is set equal to the `tempAmount`, which is re-set per loop iteration and does not add onto itself. Was this instead meant to be set to `pigsPending`, since this is incremented each step of the loop? If not, find a better naming convention for `pigsClaimedTotal`, as this is not showcasing the total number of pigs being claimed L195-198+L220-223: Make sure you update state before transferring assets. Move L195, where you transfer PigsToken, to be below L198 where you finishing updating the state that relies on the Pigs Tokens being transferred L287-288: Variables `unusedTokenA` and `unusedTokenB` are never used or returned, and should
Recommendation	Above.
Alleviation	Resolved.

Finding ID	DogsClaim.sol	Severity	Informational
Type	Best Practices	Status	Resolved

Location	L26-28: Best Practices
Description	L26-28: Make sure you update state before transferring assets. Move L26 to the line after you update the users claim amount (L28).
Recommendation	Above
Alleviation	N/A

Finding ID	N/A	Severity	
Type		Status	
Location			
Description			
Recommendation			
Alleviation	N/A		

Finding ID	DogsTokenV2.sol	Severity	Medium/Informational
Type	Centralization Risks	Status	Resolved
Location	N/A		
Description	**Project Owner can mint himself new tokens. The comments and revert arguments claim the MasterChef will own the contract, but a function present in MasterChef allows the owner to mint tokens directly to the Master Chef owner's address, with no upper limit or bound as to how many tokens.** **Informational: Centralization Risks: Project Owner Can update tax rates for buys, sells, and transfers;** **blacklist addresses; disable selling; disable transferring; limit buy amount. However, the ability to disable selling, blacklist addresses, disable transferring, or limit the buy amount can be disabled and never re-enabled.**		
Recommendation	**I suggest adding some limitation to master chef to limit how many tokens can be minted per day or block. This functionality can be toggled inside of master chef, but should be used carefully until disabled.**		
Alleviation	Will remove the ability to mint after launch.		

Finding ID	N/A	Severity	
Type		Status	
Location			
Description			
Recommendation			
Alleviation	N/A		

Finding ID	MasterChefDogs.sol	Severity	Informational/Low
Type	Best Practices/Centralization	Status	Resolved
Location	L30/31 - Best Practices L417 - Centralization		
Description	L30/31 - Make sure to scope all state variables L417: Owner is able to mint new dog tokens as Master Chef with the owner function `increaseDogsSupply`, which mints tokens directly to the `msg.sender`		
Recommendation	L417: Add some limitation here to prevent vast numbers of tokens coming into existence if the owner key was ever compromised.		
Alleviation	Resolved		

Finding ID	MasterChefPigs.sol	Severity	Informational/Low
Type	Best Practices/Centralization	Status	Resolved
Location	L24/29 - Best Practices L410 - Centralization		
Description	L24/29 -**Make sure to scope all state variables** L410: Owner is able to mint new Pig V2 tokens as Master Chef with the owner function `increasePigsSupply`, which mints tokens directly to the `msg.sender`.		
Recommendation	**L410: Add some limitation here to prevent vast numbers of tokens coming into existence if the owner key was ever compromised.**		
Alleviation	Resolved		

Finding ID	N/A	Severity	
Type		Status	
Location			
Description	N/A		
Recommendation	**N/A**		
Alleviation	N/A		

APPENDIX A: ISSUE SEVERITY CLASSIFICATION

The Auditor(s) assign one of the following Severities to every finding as per common practice.

Critical. Such issues may result in significant loss of funds, complete contract logic breakdown, or the ability of project owners to withdraw liquidity in an unreasonable way. Code snippets proving the project owner's malicious intent are flagged as critical as well. Critical issues require immediate attention, and investing in projects with critical issues is extremely risky. Examples include unguarded mint functions or their executions, provably illicit pool drainage logic, or potential flash-loan vulnerabilities.

Major. Although not proving malicious intent by themselves, major issues may still be exploited by project owners or users for a significant loss of funds or very irregular contract behavior. Examples include centralized ownership without Timelocks and multi-signatures, potential reentrancy vulnerabilities, and concentrated holdings of tokens.

Medium. Such issues do not pose an immediate and severe risk but may pose a risk of partial loss of funds or irregular contract behavior. Examples include susceptibility to obviously unintended investment strategies, high-impact integer overflows, or high-impact standardization faults such as library usage,

Minor. These issues pose a low risk to contract logic or investor funds but may be convenient to consider. Examples include integer overflows in non-essential places, nonversioned libraries, missing or faulty licensing, misleading function names, or low-impact standardization mistakes.

Informational. These issues do not pose any risk to contract logic and investor funds, Examples include tokenomics clarifications, gas optimizations, redundant code, misleading comments, style, and convention.

Confirmational. In specific situations, we issue these findings, which confirm some of the universally-concerning facts that many investors seek. Examples include contract renounces and confirmation of a contract being fork of another protocol. Note: These points are not actual issues. Obviously, only a small subset of tests ran in an audit suite receives its Confirmational Finding.

APPENDIX B: DISCLAIMER

This audit is for informational purposes only and does not provide any financial or investment advice. This report does not substitute, in any way, due diligence and your own research. This report represents result extensive process intending to help our customers improve quality of their code and readers to assess quality of customers' code, but should not be used in any way to make decisions around involvement in any particular project.

Audit has been done in accordance to methodology as outlined in AboutTruth Security and Audit Procedure sections Unless explicitly and specifically stated, only code quality has been reviewed, focusing on security flaws which could cause loss of funds or logical breakdowns within the contracts. Unless explicitly stated, tokenomics have not been reviewed (although in cases of forks of one project, Auditor may point out cases of significant deviations from common settings). Website UI has not been reviewed, as it is impossible for any auditing body to assure security of domains which are under absolute control of owners - always check you are signing correct contracts.

The report does not signify an approval," „endorsement," or ,,disapproval of the Project The audit does not indicate in any way your likelihood of making, or not losing, money in the project, as we have no control over issues such as general viability of financial primitives presented, their tokenomics, and actions of project owners including, but not limited to, selling their positions or abandoning the project.

The audit has been based on status dated in section Version Details, on artifacts detailed in Objects of Review. Specifically, we have no control nor knowledge of changes made after the date, or on different artifacts. In case the Objects of Review are not live contracts, but private code or GitHub repositories, we expect these artifacts to be full, unaltered, unabridged, and not misleading.

The audit has been elaborated by paid professional(s) as mentioned in section Audit Procedure. Please note that all statements made in this report are Auditor(s)' and do not reflect stance of © Truth Security Audits itself.