PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Banksy Farm (FTM)

07 January 2022

paladinsec.co   info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Paladin is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Paladin may, at its discretion, claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Banksy Farm on the Fantom Opera network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	Banksy Farm
URL	https://ftm.banksy.farm/
Platform	Fantom Opera
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
BanksyTokenV3	0x17230A02f23722f5e2afb0fB1F359d6905c7a678	✓ MATCH
MasterChefV3	0x6daa10F9D8F3EBAc21BEcA9edC8b86EE32E33cD0	✓ MATCH
Timelock	0x116029CFA8CD098E7C1C67c8fC1533B40387125D	✓ MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
● High	0	-	-	-
● Medium	2	2	-	-
● Low	6	2	-	4
● Informational	6	-	-	6
Total	**14**	**4**	**-**	**10**

Classification of Issues

Severity	Description
● High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
● Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
● Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
● Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 BanksyTokenV3

ID	Severity	Summary	Status
01	MEDIUM	Contracts like Masterchef and LP need to be whitelisted	✓ RESOLVED
02	MEDIUM	Address can be forcefully blacklisted by sending tokens to the address, causing the address' balance to be greater than `_maxUserHoldAmount`	✓ RESOLVED
03	LOW	Incorrect values and events emitted in `updateMaxUserHoldAmountRate` and `updateMaxUserTransferAmountRate`	ACKNOWLEDGED
04	LOW	`antiBotWorking` can be turned on anytime by the owner	✓ RESOLVED
05	LOW	`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef	✓ RESOLVED

1.3.2 MasterChef

ID	Severity	Summary	Status
06	LOW	Minted amount can exceed `banksyMaximumSupply` due to lack of accounting for treasury mint amount	ACKNOWLEDGED
07	LOW	Initial reward emission set in constructor can be higher than `MAX_EMISSION_RATE`	ACKNOWLEDGED
08	LOW	Lack of non-zero address check in the constructor for treasury address	ACKNOWLEDGED
09	INFO	`pendingBanksy` will show inaccurate pending harvests on the dapp frontend if the pending rewards causes `totalSupply` to exceed `banksyMaximumSupply`	ACKNOWLEDGED
10	INFO	`banksyReward` is not directly set to zero even if `banksy.totalSupply()` is equal to `banksyMaximumSupply`	ACKNOWLEDGED
11	INFO	Gas optimization by caching `banksy.totalSupply()` as a local variable	ACKNOWLEDGED
12	INFO	Gas optimization by returning early if multiplier is 0	ACKNOWLEDGED
13	INFO	`pool.accBanksyPerShare` is updated even if `banksyReward` is zero	ACKNOWLEDGED
14	INFO	Unnecessary casting of `msg.sender` into address	ACKNOWLEDGED

1.3.3 Timelock

No issues found.

2 Findings

2.1 BanksyTokenV3

The Banksy token is an ERC20 token which will be used as the main reward token for the Masterchef. It allows for Banksy tokens to be minted when the `mint` function is called by the owner of the contract, which at the time of deployment would be the Banksy team. Users should therefore carefully inspect that ownership of this contract has been transferred to the Masterchef. There is a maximum supply of 1,000,000 Banksy tokens which is enforced in Masterchef.

On launch, the team has stated that they will set `antiBotWorking` to `true` to enable anti-bot measures. If enabled, every token transfer would check if the sender or recipient is in the blacklist mapping, and revert if either is in it. Next, it checks if the sender is in the `_excludedHoldersFromAntiBot` and `_excludedOperatorsFromAntiBot` mapping. If the sender is not in both, it will check if the sender's token balance exceeds `_maxUserHoldAmount`. If it exceeds this value, the sender will be added to the `_blacklist` mapping and the transfer will not be done.

On deployment, the following addresses are added to the `_excludedOperatorsFromAntiBot` mapping:

- Contract deployer
- The zero address
- The contract address
- The burn address

The `maxUserHoldAmountRate` is initialized at 9% of the total supply and can be modified within the range of 5% to 100%. The `maxUserTransferAmountRate` is initialized at 3% of the total supply and can be modified within the range of 0.5% to 100%.

2.1.1 Token Overview

Address	`0x17230A02f23722f5e2afb0fB1F359d6905c7a678`
Token Supply	1,000,000 (enforced in Masterchef)
Decimal Places	18
Transfer Max Size	3% (range of 0.5% to 100%)
Transfer Min Size	No minimum
Transfer Fees	None

2.1.2 Privileged Roles

The following functions can be called by the owner of the contract:

- `mint`
- `transferOwnership`
- `renounceOwnership`
- `updateOperatorsFromAntiBot`
- `updateHoldersFromAntiBot`
- `updateMaxUserHoldAmountRate`
- `updateMaxUserTransferAmountRate`
- `updateStatusAntiBotWorking`
- `addBotAddress`
- `addBotAddressBatch`

The following function can be called by the operator of the contract:

- `transferOperator`
- `removeBotAddress`
- `removeBotAddressBatch`

2.1.3 Issues & Recommendations

Issue #01	**Contracts like Masterchef and LP need to be whitelisted**
Severity	● MEDIUM SEVERITY
Description	When the anti-bot feature is active and if the Masterchef contract allows the staking of the Bansky token and is not whitelisted, users will not be able to withdraw once enough tokens have been staked. Similarly, the liquidity pool contract will not allow buys if it is not whitelisted and has enough Banksy tokens to exceed the `_maxUserHoldAmount`, users will not be able to purchase tokens.
Recommendation	Whitelist the above contracts in `_excludedHoldersFromAntiBot`.
Resolution	✓ RESOLVED `antiBotWorking` has been disabled. Ownership of the token has been transferred to the Masterchef so the this function cannot be re-enabled.

Issue #02	**Address can be forcefully blacklisted by sending tokens to the address, causing the address' balance to be greater than `_maxUserHoldAmount`**
Severity	MEDIUM SEVERITY
Description	While `antiBotWorking` is enabled, it is possible to send address tokens from multiple addresses until the address' balance exceeds `_maxUserHoldAmount`. When that address tries to transfer tokens, it will be blacklisted and not be able to transfer.
Recommendation	The check for `_maxUserHoldAmount` can be done on the recipient of a transfer, and revert if the transfer causes the recipient's balance after the transfer to exceed `_maxUserHoldAmount`. If this method is used, it would require whitelisting certain addresses such as the Masterchef and the LP contract.
Resolution	✓ RESOLVED `antiBotWorking` has been disabled. Ownership of the token has been transferred to the Masterchef so the this function cannot be re-enabled.

Issue #03	Incorrect values and events emitted in `updateMaxUserHoldAmountRate` and `updateMaxUserTransferAmountRate`
Severity	LOW SEVERITY

Location	Line 249~
	`function updateMaxUserHoldAmountRate(uint16 _maxUserHoldAmountRate) external onlyOwner {`
	`    require(_maxUserHoldAmountRate >= 500);`
	`    require(_maxUserHoldAmountRate <= 10000);`
	
	`    emit TransferTaxRateUpdated(_msgSender(), maxUserHoldAmountRate, _maxUserHoldAmountRate);`
	
	`    maxUserHoldAmountRate = _maxUserHoldAmountRate;`
	`}`
	
	Line 259~
	`function updateMaxUserTransferAmountRate(uint16 _maxUserTransferAmountRate) external onlyOwner {`
	`    require(_maxUserTransferAmountRate >= 50);`
	`    require(_maxUserTransferAmountRate <= 10000);`
	
	`    emit HoldingAmountRateUpdated(_msgSender(), maxUserHoldAmountRate, _maxUserTransferAmountRate);`
	
	`    maxUserTransferAmountRate = _maxUserTransferAmountRate;`
	`}`

Description	Incorrect events and values are being emitted in the `updateMaxUserHoldAmountRate` and `updateMaxUserTransferAmountRate`.

Recommendation

Correct the events and values emitted:

```
function updateMaxUserHoldAmountRate(uint16
_maxUserHoldAmountRate) external onlyOwner {
    require(_maxUserHoldAmountRate >= 500);
    require(_maxUserHoldAmountRate <= 10000);

    emit HoldingAmountRateUpdated(_msgSender(),
maxUserHoldAmountRate, _maxUserHoldAmountRate);

    maxUserHoldAmountRate = _maxUserHoldAmountRate;
}
function updateMaxUserTransferAmountRate(uint16
_maxUserTransferAmountRate) external onlyOwner {
    require(_maxUserTransferAmountRate >= 50);
    require(_maxUserTransferAmountRate <= 10000);

    emit TransferTaxRateUpdated(_msgSender(),
maxUserTransferAmountRate , _maxUserTransferAmountRate);

    maxUserTransferAmountRate = _maxUserTransferAmountRate;
}
```

Resolution

ACKNOWLEDGED

Issue #04	`antiBotWorking` can be turned on anytime by the owner
Severity	● LOW SEVERITY
Description	`antiBotWorking` can be modified anytime by the owner of the token contract using the `updateStatusAntiBotWorking` function. This can cause users who have a balance that exceeds the `maxUserHoldAmountRate` to be blacklisted when transferring.

Recommendation	`antiBotWorking` should be set to `true` in the constructor, and only allowed to be disabled the owner.
	
	Example:
	`function updateStatusAntiBotWorking() external onlyOwner {`
	`    require(antiBotWorking, "antiBotWorking is off");`
	`    emit AntiBotWorkingStatus(_msgSender(), antiBotWorking, false);`
	
	`    antiBotWorking = false;`
	`}`

Resolution	✓ RESOLVED `antiBotWorking` has been disabled. Ownership of the token has been transferred to the Masterchef so the this function cannot be re-enabled.

Issue #05	**`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef**
Severity	● LOW SEVERITY
Description	The `mint` function could be used to pre-mint tokens for legitimate uses including, but not limited to, the injection of initial liquidity, token presale, or airdrops; however, this function may also be used to pre-mint and dump tokens when the token contract has been deployed but before ownership is set to the Masterchef contract. This risk is prevalent amongst less-reputable projects, and any pre-mints can be prominently seen on the Blockchain.
Recommendation	Consider being forthright if this `mint` function is to be used by letting your community know how much was minted, where they are currently stored, if a vesting contract was used for token unlocking, and finally the purpose of the mints.
Resolution	✓ RESOLVED Ownership of the contract has been transferred to the Masterchef.

2.2 MasterChef

The Banksy Masterchef is an modified fork of Goose Finance's Masterchef where deposit fees have been capped to a maximum of 4.01%. A notable feature of forking the Goose Masterchef is the removal of the `migrator` function from Sushiswap, which can possibly be used to steal staked tokens.

Emissions have been modified to be per second instead of per block. The Masterchef has a check that ensures that only a maximum of 1,000,000 Banksy tokens are minted. The maximum emission that can be set is 1 token per second. 10% of emissions are minted to the treasury.

2.2.1 Privileged Roles

The following functions can be called by the owner of the Masterchef:

- `add`
- `set`
- `setFeeAddress`
- `setTreasuryAddress`
- `setEmissionRate`
- `setStartTime`
- `transferOwnership`
- `renounceOwnership`

2.2.2 Issues & Recommendations

Issue #06	Minted amount can exceed banksyMaximumSupply due to lack of accounting for treasury mint amount
Severity	● LOW SEVERITY

Description	The following check does not take into account the amount of minted tokens to the treasury, and can result in the second mint to cause the total supply to be more than banksyMaximumSupply.
	Line 206~
	`uint256 banksyReward = (multiplier * banksyPerSecond * pool.allocPoint) / totalAllocPoint;`
	
	`// This shouldn't happen, but just in case we stop rewards.`
	`if (banksy.totalSupply() > banksyMaximumSupply)`
	`    banksyReward = 0;`
	`else if ((banksy.totalSupply() + banksyReward) > banksyMaximumSupply)`
	`    banksyReward = banksyMaximumSupply - banksy.totalSupply();`
	
	`    if (banksyReward > 0){`
	`        banksy.mint(address(this), banksyReward);`
	`        banksy.mint(treasuryAddress, banksyReward / 10);`
	`    }`

Recommendation	Consider checking by using the total supply and sum of 110% of the rewards in the `else if` statement, and only minting the difference to the Masterchef if it exceeds the `banksyMaximumSupply`.

```
uint256 banksyReward = (multiplier * banksyPerSecond *
pool.allocPoint) / totalAllocPoint;
uint256 treasuryReward = banksyReward / 10;

// This shouldn't happen, but just in case we stop rewards.
if (banksy.totalSupply() > banksyMaximumSupply)
    banksyReward = 0;
    treasuryReward = 0;
else if ((banksy.totalSupply() + banksyReward +
treasuryReward) > banksyMaximumSupply) {
    banksyReward = banksyMaximumSupply -
banksy.totalSupply();
    treasuryReward = 0;
}

if (banksyReward > 0){
    banksy.mint(address(this), banksyReward);
}

if (treasuryReward > 0){
    banksy.mint(treasuryAddress, treasuryReward);
}
```

Resolution	● ACKNOWLEDGED

Issue #07	**Initial reward emission set in constructor can be higher than `MAX_EMISSION_RATE`**
Severity	● LOW SEVERITY
Description	Although the `setEmissionRate` function ensures that the emission does not exceed `MAX_EMISSION_RATE`, it is not done in the constructor.
Recommendation	Consider adding the same check for the emission to be less than the maximum emission rate in the constructor.
Resolution	● ACKNOWLEDGED

Issue #08	Lack of non-zero address check in the constructor for treasury address
Severity	● LOW SEVERITY
Description	Although there is a check in `setTreasuryAddress` to ensure that `treasuryAddress` cannot be set to the zero address, it is not done in the constructor. If the contract is deployed with the `treasuryAddress` set as the zero address, `updatePool` will revert as it will try to mint 10% of the emissions to the treasury address.
Recommendation	Consider adding the same non-zero address check in the constructor.
Resolution	● ACKNOWLEDGED

Issue #09	`pendingBanksy` will show inaccurate pending harvests on the dapp frontend if the pending rewards causes `totalSupply` to exceed `banksyMaximumSupply`
Severity	● INFORMATIONAL
Description	`pendingBanksy` does not check if the pending rewards will cause the `totalSupply` to exceed the `banksyMaximumSupply`. This can cause inaccurate pending harvests to be shown towards the end of token emissions.
Recommendation	Consider factoring in the `banksyMaximumSupply`, and set the pending reward to be the difference between `banksyMaximumSupply` and `totalSupply` if the pending reward causes `totalSupply` to exceed `banksyMaximumSupply`.
Resolution	● ACKNOWLEDGED

Issue #10	**banksyReward is not directly set to zero even if `banksy.totalSupply()` is equal to banksyMaximumSupply**
Severity	INFORMATIONAL

Description	The current check to set `banksyReward` as 0 is as follows:
	
	Line 1692
	`if (banksy.totalSupply() > banksyMaximumSupply)`
	`    banksyReward = 0;`
	
	In the case where `totalSupply` is equal to `banksyMaximumSupply`, it will not be set as zero directly, but instead enter to the `else if` statement.

Recommendation	Change it to include the case when `totalSupply` is equal to the `banksyMaximumSupply`.
	
	`if (banksy.totalSupply() >= banksyMaximumSupply)`
	`    banksyReward = 0;`

Resolution	ACKNOWLEDGED

Issue #11	**Gas optimization by caching `banksy.totalSupply()` as a local variable**
Severity	INFORMATIONAL
Description	As `banksy.totalSupply()` is called multiple times in the `updatePool` function, gas usage can be optimized by setting `banksy.totalSupply()` as a local variable, and reusing that local variable within the function.
Recommendation	Set and use the following local variable in `updatePool`: `uint256 totalSupply = banksy.totalSupply();`
Resolution	ACKNOWLEDGED

Issue #12	Gas optimization by returning early if multiplier is 0
Severity	INFORMATIONAL
Description	Even when `multiplier` will always return 0 after the total supply is done with minting, subsequent `updatePool` calls will still calculate `banksyReward` and attempt to update the `accBanksyPerShare`, even though `banksyReward` will be 0 when `multiplier` is 0.

Recommendation	Consider returning early if `multiplier` is 0.
	`uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.timestamp);`
	
	`if (pool.lpSupply == 0 || pool.allocPoint == 0 || multiplier == 0) {`
	`    pool.lastRewardTime = block.timestamp;`
	`    return;`
	`}`

Resolution	ACKNOWLEDGED

Issue #13	`pool.accBanksyPerShare` is updated even if `banksyReward` is zero
Severity	INFORMATIONAL
Description	In `updatePool`, even if `banksyReward` is 0, either due to the `totalSupply` exceeding the `banksyMaximumSupply`, or if the `updatePool` is called when `_to` and `_from` are the same, `pool.accBanksyPerShare` is updated.

Recommendation	Change it to only update `pool.accBanksyPerShare` when `banksyReward` is not zero.
	`if (banksyReward > 0)`
	`    banksy.mint(address(this), banksyReward);`
	`pool.accBanksyPerShare = pool.accBanksyPerShare + ((banksyReward * 1e18) / pool.lpSupply);`

Resolution	ACKNOWLEDGED

Issue #14	Unnecessary casting of msg.sender into address
Severity	● INFORMATIONAL
Location	(Example) Line 1721 `pool.lpToken.safeTransferFrom(address(msg.sender), address(this), _amount);`
Description	There are a number of instances where `msg.sender` is cast into an address, when it already is an address datatype.
Recommendation	Remove the casting of `msg.sender` as an address.
Resolution	● ACKNOWLEDGED

2.3 Timelock

The Timelock contract is a clean fork of Compound Finance's timelock. This is the most common contract used in DeFi to time lock governance access and is thus compatible with most third-party tools. This contract should be the owner of the Masterchef contract to time delay making sensitive changes such as adding a new pool, or changing the allocation for an existing pool.

Parameter	Value	Description
Delay	24 hours	The `delay` indicates the time the administrator has to wait after queuing a transaction to execute it.
Minimum Delay	6 hours	The `minDelay` indicates the lowest value that the `delay` can minimally be set. Sometimes, projects will queue a transaction that sets the `delay` to zero with the hope that nobody notices it. However, because of the minimum delay parameter, the value of `delay` can never be lower than that of the `minDelay` value. Note that the administrator could still queue a transaction to simply transfer the ownership back to their own account so it is still important to inspect every transaction carefully.
Maximum Delay	30 days	The maximum delay indicates the highest value that the `delay` can be set.
Grace Period	14 days	After the delay has expired after queueing a transaction, the administrator can only execute it within the grace period. This is to prevent them from hiding a malicious transaction among much earlier transactions, hoping that it goes unnoticed or buried, which can be executed in the future.

2.3.1 Issues & Recommendations

No issues found.

PALADIN

BLOCKCHAIN SECURITY