

Scanned Code Report

AUDIT AGENT

Code Info

[Advanced Scan](#) Scan ID
5 Date
November 16, 2025 Organization
CJ42 Repository
potato-tipper-contracts Branch
main Commit Hash
73c2fc81...7ee38d7a

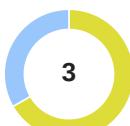
Contracts in scope

[src/Constants.sol](#) [src/Events.sol](#) [src/PotatoTipper.sol](#) [src/PotatoTipperConfig.sol](#)
[src/PotatoTipperSettingsLib.sol](#)

Code Statistics

 Findings
3 Contracts Scanned
5 Lines of Code
473

Findings Summary



Total Findings

 High Risk (0) Medium Risk (0) Low Risk (2) Info (1) Best Practices (0)

Code Summary

The Potato Tipper protocol is a utility designed for the LUKSO ecosystem that enables users to automatically reward new followers with `$POTATO` tokens. It functions as an automated incentive mechanism, leveraging LUKSO's standard smart contract architecture, including Universal Profiles (LSP0), the Universal Receiver standard (LSP1), Digital Assets (LSP7), and the Follower System (LSP26).

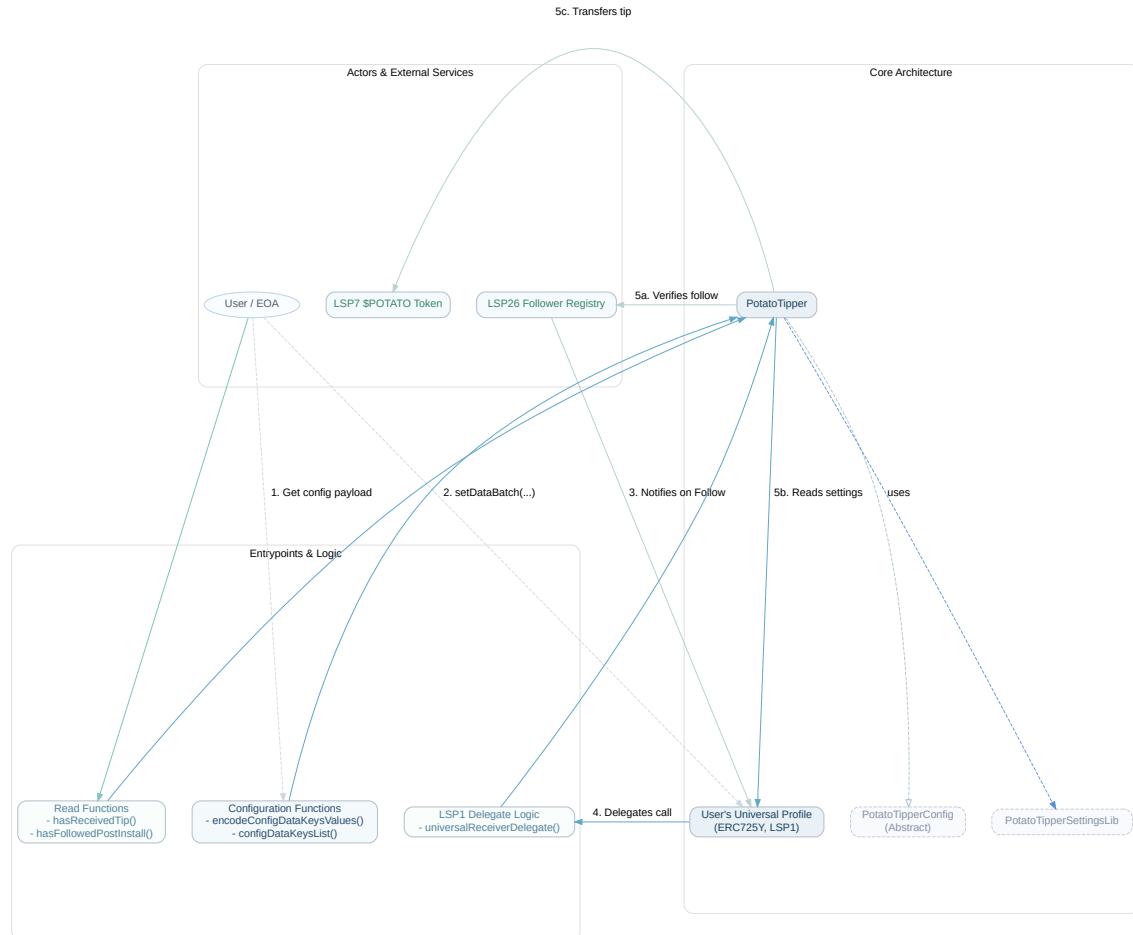
A user integrates the protocol by configuring their Universal Profile (UP) to delegate follow and unfollow notifications to the `PotatoTipper` contract. The user also sets specific tipping rules directly on their UP using ERC725Y data keys. These rules, defined in a `TipSettings` struct, include the amount of `$POTATO` to tip, and eligibility criteria for the new follower, such as their minimum follower count and minimum `$POTATO` token balance.

When a new follower follows the user, the LSP26 Follower Registry sends a notification to the user's UP. The UP's `universalReceiver` function then calls the `universalReceiverDelegate` function on the `PotatoTipper` contract. The contract verifies the legitimacy of the follow event, checks if the new follower meets the user-defined eligibility criteria, and ensures the follower has not been tipped before. To prevent gaming the system, the protocol also tracks followers who were present before its activation, making them ineligible for a tip if they unfollow and re-follow. If all conditions are met, the contract executes an LSP7 token transfer of `$POTATO` from the user's UP to the new follower's UP, for which the user must have granted prior authorization.

Main Entry Points and Actors

- `universalReceiverDelegate(address sender, uint256 value, bytes32 typeId, bytes data)`: This is the primary entry point, designed to be called by a user's Universal Profile. It processes notifications from the LSP26 Follower Registry. Based on the `typeId`, it handles either a new follow by triggering the tipping logic or an unfollow by updating its internal state to prevent abuse. It modifies the protocol's state by recording who has been tipped and tracking follower history.
- **Actors:** User's Universal Profile.

Code Diagram



 1 of 3 Findings src/PotatoTipper.sol**POTATO tokens can become stuck if forcibly sent to the delegate contract** • Low Risk

LSP7 supports transfers with `force=true`, allowing tokens to be sent to any address even if it does not implement the LSP1 receiver interface. The PotatoTipper contract does not implement a recovery mechanism and never initiates outbound transfers from its own balance (tips are executed with `from: msg.sender`). If someone mistakenly or maliciously transfers POTATO to `address(this)` with `force=true`, those tokens cannot be retrieved by the contract and may become permanently stuck.

```
function _transferTip(address follower, uint256 tipAmount) internal returns (bytes memory) {
    _tipped[msg.sender][follower] = true;
    try _POTATO_TOKEN.transfer({
        from: msg.sender,
        to: follower,
        amount: tipAmount,
        force: false,
        data: unicode"Thanks for following! Tipping you some 🥔"
    }) {
        emit PotatoTipSent({from: msg.sender, to: follower, amount: tipAmount});
        return abi.encodePacked(unicode"🥔✅ Successfully sent tip to new follower: ",
            follower.toHexString());
    } catch (bytes memory errorData) {
        delete _tipped[msg.sender][follower];
        emit PotatoTipFailed({from: msg.sender, to: follower, amount: tipAmount, errorData:
            errorData});
        return abi.encodePacked(
            unicode"🥔❌ Failed to send tip to ", follower.toHexString(), ". LSP7 transfer
            reverted");
    }
}
```

Because the contract holds no logic to move tokens from its own balance, any POTATO inadvertently held by the contract is effectively locked.

 2 of 3 Findings src/PotatoTipper.sol

Follower count manipulation through Sybil networks

 Low Risk

The `_validateTipEligibilityCriteria` function checks a follower's follower count to determine tip eligibility:

```
if (_FOLLOWER_REGISTRY.followerCount(follower) < minimumFollowerCountRequired) {  
    return (false, unicode"🔍❌ Not eligible for tip: minimum follower required not met");  
}
```

This validation only checks the raw count from the LSP26 Follower Registry without verifying the legitimacy or quality of those followers. An attacker can create a network of Sybil accounts (bot Universal Profiles) that follow each other to artificially inflate their follower counts:

1. Create multiple Universal Profile contracts controlled by the attacker
2. Have these profiles follow each other in a circular pattern
3. Each profile now meets the `minimumFollowerCountRequired` threshold
4. Use these profiles to follow target users and collect tips

This attack is economically viable when the cost of deploying and maintaining Sybil UPs is less than the accumulated tip rewards. The attack becomes more effective with lower eligibility thresholds and higher tip amounts. While the protocol includes the `minimumPotatoBalance` check as an additional barrier, this can be bypassed via flash loans (see related finding). The documentation acknowledges this limitation and mentions that future versions could implement oracles to verify follower age and legitimacy.

Severity Note:

- Creating and operating LSP0 UPs is cheap enough compared to the tip amount to be profitable.
- LSP26 follower registry counts unique addresses without Sybil resistance measures.
- Users have enabled PotatoTipper and granted a non-zero tipping allowance.

 3 of 3 Findings src/PotatoTipper.sol

Front-Running Vulnerability in Tipping Mechanism

 Info

The tipping mechanism operates on a first-come, first-served basis. When a legitimate user submits a transaction to follow a tipper-enabled profile, this transaction becomes visible in the public mempool before it is mined.

A malicious actor, such as an MEV bot, can monitor the mempool for such `follow` transactions. Upon detecting one, the bot can copy the transaction, replace the original follower's address with its own, and submit it with a higher gas fee. This ensures the bot's transaction is processed first, allowing it to claim the tip that was intended for the legitimate follower.

This is particularly impactful if the user's tipping budget (`authorizedAmountFor`) is limited, as the bot could deplete the available funds, leaving none for genuine followers.

Disclaimer

Kindly note, the Audit Agent is currently in beta stage and no guarantee is being given as to the accuracy and/or completeness of any of the outputs the Audit Agent may generate, including without limitation this Report. The results set out in this Report may not be complete nor inclusive of all vulnerabilities. The Audit Agent is provided on an 'as is' basis, without warranties or conditions of any kind, either express or implied, including without limitation as to the outputs of the code scan and the security of any smart contract verified using the Audit Agent.

Blockchain technology remains under development and is subject to unknown risks and flaws. This Report does not indicate the endorsement of any particular project or team, nor guarantee its security. Neither you nor any third party should rely on this Report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset.

To the fullest extent permitted by law, Nethermind disclaims any liability in connection with this Report, its content, and any related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Nethermind does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and Nethermind will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.