Smart Contract Audit Report

Contract Name: USDTFlash (USDTF)
Network: TRON
Compiler: Solidity 0.5.10
Repository: GitHub - trotoksud/USDTF
Audit Date: June 27, 2025
Post-Audit Review: July 2025

Audit Scope

This audit covers the full review of `USDTF.sol`, which implements a custom, educational-purpose token mimicking stablecoin behavior with flash minting and expiration logic.

Strengths

Feature	Description
Time-bound tokens	Tokens are minted with expiry timestamps
Flash minting	`flashMint()` limited to `onlyOwner`
Expiry enforcement	Expired lots removed via `burnExpired()` and `_cleanExpired()`
Access control	All admin actions gated behind `onlyOwner`
Transparency	Code and whitepaper are open-source and publicly documented

---

Observations & Recommendations

Area	Finding	Risk	Recommendation
Solidity Version	Uses `^0.5.10`	Moderate	Consider upgrading to `^0.8.x`
Expiry Timestamps	Uses `now`	Low	Use `block.timestamp` for clarity
Token Standard	Not fully ERC20-compliant	Medium	Add interfaces (`name`, `symbol`, etc.)
Unlimited Minting	Owner can mint infinitely	High	Add cap or throttle
Expired Token Cleanup	Only burns `from`, not `to`	Low	Burn for both ends or clarify intent
Data Structure Growth	`TokenLot[]` unbounded per user	Medium	Migrate to mappings or batch-cleanup logic
Circuit Breaker	No `pause()` or	Medium	Add pausable modifier

	failsafe		

□ Security Risk Review

Category	Status	Notes
Access Control	□ Safe	All writes protected with `onlyOwner`
Reentrancy	□ Safe	No external calls after state changes
Arithmetic Safety	□ Manual	Solidity 0.5.10 lacks SafeMath (none observed)
Storage Collisions	□ None	No overlapping or unsafe slot usage
Self-Destruct / Proxy	□ Absent	No self-destruct or upgradability patterns present

---

□ Post-Audit Update – Mythril Scan

Tool: Mythril
Scan Date: July 2025
Issue: Exception State (SWC-110) – Potential out-of-bounds access in public array `holdings[address][index]`.

□ Risk Context

- Only affects **public read access** to the `holdings` mapping
- Anyone querying a bad index gets a **revert**, not a security leak
- **No write vulnerability**, no impact to balances or expiry logic

□ Risk Mitigation

- All `holdings` writes restricted to `onlyOwner`
- Front-end validates index bounds before calls
- No contract logic is influenced by `holdings` reads

Conclusion: Low-risk symbolic finding. Not exploitable in practice.

□ Suggested Tests (Post-Deployment)

- □ Minting with varied expiries
- □ Transfer before and after expiry
- □ Front-end bounds-checks for `holdings[index]`
- □ Allowance + `transferFrom` checks
- □ Simulated time advancement for burn testing

□ Final Verdict

This contract is intended for **educational and non-commercial use**. It is secure within its defined scope and makes no attempt to be a production-grade standard token. Risks are documented and known.