OdinScan

SECURITY AUDIT REPORT

bank_two (Solana Developer Bootcamp)

Date: September 02, 2026

Report Version: 1.0

Report ID: `example`

Commit: 8aa304d1

Table of Contents

1. Introduction

1.1 Scope Functionality

This report covers the security audit of the smart contract repository **bank_two (Solana Developer Bootcamp)**. The assessment identifies vulnerabilities, insecure patterns, and code quality issues using a combination of static analysis engines and AI-powered detection.

1.2 Submitted Codebase

Item	Details
Repository	bank_two (Solana Developer Bootcamp)
Repository URL	https://github.com/solana-developers/developer-bootcamp-2024/tree/main/project-12-attack-the-bank/bank_two
Branch	-
Commit	`8aa304d1`

1.3 Methodologies

- **Static Code Analysis** — pattern matching and AST-based vulnerability detection across contract source files
- **AI-Powered Detection** — large language models specialised in smart contract security review each file for semantic vulnerabilities
- **Cross-Validation** — findings are merged, deduplicated, and scored for confidence
- **Verification Pass** — a second AI pass reviews flagged findings and marks likely false positives

1.4 Code Criteria

Severity	Description	Action Required
CRITICAL	Immediate risk of fund loss, contract takeover, or protocol insolvency	Fix before deployment
HIGH	Significant security issue with realistic exploit path	Fix as priority
MEDIUM	Exploitable under specific conditions or causes meaningful harm	Should be addressed
LOW	Minor security concern or best-practice deviation	Consider fixing
INFORMATIONAL	Code quality, documentation, or non-security observation	For reference

2. Findings Summary

Total findings: **6** — 2 CRITICAL 2 HIGH 1 MEDIUM 1 LOW 0 INFO

ID	Summary Title	Risk Impact	Status
C-01	Missing Signer Verification in update_authority	CRITICAL	CONFIRMED
C-02	First Depositor Can Become Bank Authority	CRITICAL	CONFIRMED
H-03	Unchecked Arithmetic Operations on bank_balance	HIGH	CONFIRMED
H-04	Missing Balance Validation in withdraw	HIGH	CONFIRMED
M-05	Missing Zero-Amount Validation in deposit and withdraw	MEDIUM	CONFIRMED
L-06	State Update Before External Transfer in deposit	LOW	CONFIRMED

3. Findings — Technical Details

C-01 CRITICAL CONFIRMED

Missing Signer Verification in update_authority

DESCRIPTION

The `UpdateAuthority` accounts struct declares `authority` as `SystemAccount` instead of `Signer`. Combined with `has_one = authority`, this allows any user to invoke the instruction by including the current authority's public key as a non-signer account in the transaction. The `has_one` constraint is satisfied without requiring an actual signature from the authority wallet, enabling anyone to change the bank authority to an address they control and subsequently drain funds.

LOCATION

`src/lib.rs:78–88`

RECOMMENDATION

Change `authority: SystemAccount` to `authority: Signer` in `UpdateAuthority` and keep the `has_one` constraint. The Signer constraint makes Anchor require the current bank authority's signature, so only the true owner can rotate it.

PROOF OF CONCEPT

```
await program.methods.updateAuthority().accounts({ newAuthority: attacker }).instruction();
await program.methods.withdraw(new anchor.BN(amount)).accounts({ authority: attacker }).instruction
();
```

VERIFICATION NOTES

Verifier confirmed the finding: `authority` is passed as a non-signer account and `has_one` only checks key equality, so any wallet can call `updateAuthority`.

C-02 CRITICAL CONFIRMED

First Depositor Can Become Bank Authority

DESCRIPTION

The `deposit` instruction uses `init_if_needed` and, when `is_initialized` is false, sets `bank.authority` to the caller's public key. Any user can become the bank authority simply by being the first to deposit (e.g., with 1 lamport). Once in control, combined with the missing signer check in `update_authority` and the unrestricted `withdraw` instruction, an attacker can drain all funds from the bank.

LOCATION

`src/lib.rs:10–32`

RECOMMENDATION

Replace `init_if_needed` with `init` plus an explicit initialization check, or use `init_if_needed` only when the account is guaranteed fresh. Alternatively, only honor the `is_initialized` flag when it was set by a genuine initializer (e.g., require the first depositor's signature once, via a dedicated `initialize` instruction).

H-03 HIGH CONFIRMED

Unchecked Arithmetic Operations on bank_balance

DESCRIPTION

The `bank_balance` field is modified using unchecked `+=` and `-=` operators in `deposit` and `withdraw`. While the Solana runtime reverts on overflow/underflow, using checked arithmetic (`checked_add`, `checked_sub`) is recommended for financial code to provide explicit error handling and clearer error reporting instead of relying on runtime panics.

LOCATION

`src/lib.rs:12-35`

RECOMMENDATION

Use checked arithmetic: `bank.bank_balance = bank.bank_balance.checked_add(amount) .ok_or(BankError::Overflow)?` in `deposit`, and checked_sub in `withdraw`.

H-04 HIGH CONFIRMED

Missing Balance Validation in withdraw

DESCRIPTION

The `withdraw` instruction does not verify that `bank_balance >= amount` before performing the subtraction and the lamport transfer. This allows users to attempt withdrawals larger than the available balance, causing transaction failures at the arithmetic underflow step, wasting compute and producing a poor user experience. The lamport transfer via `sub_lamports` would also fail if insufficient balance exists.

LOCATION

`src/lib.rs:34-40`

RECOMMENDATION

Add `require!(bank.balance >= amount, BankError::InsufficientFunds)` before the subtraction and transfer, and map the error to a user-friendly failure.

M-05 MEDIUM CONFIRMED

Missing Zero-Amount Validation in deposit and withdraw

DESCRIPTION

Neither `deposit` nor `withdraw` validates that the `amount` parameter is greater than zero. Zero-amount operations are accepted, which wastes compute, pollutes on-chain state and event logs (via `msg!`), and can confuse off-chain indexers and UIs that track deposits and withdrawals.

LOCATION

`src/lib.rs:10-40`

RECOMMENDATION

Add `require!(amount > 0, BankError::ZeroAmount)` to both `deposit` and `withdraw`.

L-06 LOW CONFIRMED

State Update Before External Transfer in deposit

DESCRIPTION

In the `deposit` instruction, `bank_balance += amount` is executed before the CPI `transfer` of lamports to the bank PDA. Although Solana transactions are atomic (a CPI failure reverts all state changes), modifying internal state before performing the external interaction violates the checks-effects-interactions pattern and is considered risky practice. If any non-CPI failure path were introduced later, the state could become desynchronized from actual lamport holdings.

LOCATION

`src/lib.rs:11–30`

RECOMMENDATION

Perform the CPI transfer first, then update `bank_balance` (checks-effects-interactions).

Document Control

Date	September 02, 2026
Report Version	1.0
Produced by	Odin Scan — AI-Enhanced Smart Contract Security
Contact	security@odinscan.ai
Website	odinscan.ai

Appendix A: Risk Assessment Methodology

Odin Scan assigns a severity level to each finding based on the combination of the potential impact and the likelihood of exploitation.

CRITICAL	Vulnerabilities where exploitation is straightforward and the impact is catastrophic — complete loss of funds, permanent denial of service, or full contract takeover. These must be resolved before any deployment.
HIGH	Serious vulnerabilities with a realistic attack path that results in significant financial loss or privilege escalation. Should be fixed as a top priority.
MEDIUM	Vulnerabilities exploitable under specific conditions or that cause meaningful but non-catastrophic harm. Should be addressed before production.
LOW	Minor security issues or deviations from best practices that present low risk in isolation but may combine with other issues to increase overall risk.
INFO	Observations about code quality, documentation gaps, or architectural notes that do not represent an immediate security risk but are worth addressing.

Appendix B: Disclaimer

> This report was generated automatically by Odin Scan.

This security audit report is provided for informational purposes only. While Odin Scan employs state-of-the-art static analysis and AI-powered detection techniques, no automated tool can guarantee the discovery of all vulnerabilities in a codebase.

The findings presented in this report represent the output of automated analysis at the time of the audit. New vulnerabilities may be introduced through subsequent code changes. Odin Scan recommends continuous monitoring and periodic re-audits as the codebase evolves.

This report does not constitute legal, financial, or professional advice. Deployment decisions should be made in conjunction with qualified security professionals reviewing this report alongside any additional manual audit findings.

All findings marked as "Likely False Positive" have been flagged by the automated verification pass and should be manually reviewed before being dismissed.