Adevar Labs

Prepared for:

Tempest Labs

January 12, 2026

Ignition Stake Pool Program Audit Report

Version: Audit report after fix review

Table of Contents

1. Executive Summary

About Ignition

Ignition is a liquid staking protocol for SOL developed by Tempest Labs, forked from the original SPL stake-pool program. The protocol was modified to integrate Fogo Sessions and WSOL wrapping/unwrapping, a session key mechanism allowing users to execute protocol interactions without per-transaction signature requirements while maintaining defined security boundaries.

Audit Summary

The table below summarizes identified vulnerabilities by risk level and remediation status.

Risk Level	Count	Fixed	Acknowledged
Critical	0	–	–
High	1	1	0
Medium	0	–	–
Low	3	3	0

Enhancement opportunities: 3

Overall Security Posture

The following sections describe the security posture before and after the fix review. The fix review is the stage of the audit where Adevar Labs checked the fixes implemented by the Tempest Labs development team for the issues found during the audit.

The Tempest Labs team demonstrated a solid understanding of the Solana SPL program. By restricting the modifications to two wrapper functions around the deposit and withdraw functions, the team employed a prudent development strategy, thereby reducing risk and supporting a secure deployment by leaving all core financial logic unchanged.

After Fix Review Summary

The fix review confirmed successful resolution of both the High and Low severity findings. The Tempest Labs team's implementation of most enhancement opportunities further demonstrates their commitment to code quality and security best practices.

All mandatory and strongly recommended issues have been addressed. Post-launch, the team should maintain alignment with the upstream SPL stake-pool program by monitoring and incorporating relevant security updates as they become available.

Before Fix Review Summary

The core architecture directly inheriting from the SPL stake-pool program ensures a robust foundation for the protocol. The clean Fogo sessions and WSOL integration demonstrated sound

design principles and great understanding of the Solana and Fogo environment.

The audit identified one High severity issue related to the use of an outdated version of the SPL stake-pool program, and three Low severity issues: unconditional rent transfer on stake account creation, inability to specify minimum output amounts (minAmountOut) in deposit/withdraw wSOL functions when using Fogo Sessions, and users burning more pool tokens than the stake lamports received when paying for stake account rent themselves. Additionally, three enhancement opportunities were identified to improve code quality and protocol flexibility.

I. Mandatory before launch:

- Fix the High severity issue:
  - Include fixes from the v2.0.4 of the SPL stake-pool program.

II. Strongly recommended:

- Fix all Low severity issues.
- Implement proposed enhancement opportunities

III. Post-Launch Monitoring:

- Monitor every update of the SPL stake-pool program and ensure that the Ignition version is always up-to-date with security fixes.

Audit Scope

The initial audit scope was performed on the diff between the forked commit hash and before-fix review commit hash.

- **Repository:** https://github.com/Tempest-Finance/spl-stake-pool
- **Before-Fix Review Commit Hash:** 82dbce2ba5f9e31134f88a7c01a5f18c56b9886f
- **After-Fix Review Commit Hash:** 544c269ce395a65e4615f2153b51c8e8dfd14213
- **Files/Modules in Scope:**
  - *program/src*
- **Forked repository:** https://github.com/solana-program/stake-pool
- **Forked commit hash:** 8f705b3cb1a80e49ef4037a593d10713499ea8d7

Note: During the engagement, the Tempest Labs team requested the inclusion of additional changes (commit ac5d7dc07194982fd19f4ea56ec8e1be2fe3275e), which were integrated into this report.

It should be noted that this 2nd PR has introduced new functionality that wasn't present in an initial scope review.

2. Assumptions and Considerations

Audit Assumptions

Ignition is a fork of the SPL stake-pool program. Only changes and their impact on the protocol have been reviewed during this audit.

Trust Assumptions

The protocol's correct functioning relies on the fogo-sessions and token programs of the Fogo Foundation.

3. Severity Definitions

Each issue identified in this report is assigned a severity level based on two dimensions: **Impact** and **Likelihood**. These dimensions help project our team's understanding of both the potential consequences of a vulnerability and how likely a vulnerability is to be discovered and exploited in the real world.

Note: Enhancements represent non-blocking improvements–typically usability, observability, or defense-in-depth tweaks that do not pose an immediate asset risk but would improve the product's reliability and/or user experience if implemented.

Impact

Impact reflects the potential consequences of the issue–particularly on **project funds**, **user funds**, and the **availability or integrity** of the protocol.

- **High Impact**: Successful exploitation could result in a complete loss of user or protocol funds, disruption of core protocol functionality, or permanent loss of control over critical components.
- **Medium Impact**: Exploitation could cause significant disruption or partial loss of funds, but not a total compromise. May impact some users or non-core functionality.
- **Low Impact**: The issue has minor or negligible consequences. It may affect edge cases, expose metadata, or degrade performance slightly without putting funds or core logic at serious risk.

Likelihood

Likelihood reflects how easy a vulnerability is to discover and exploit by an attacker, as well as how economically attractive the exploit is to an attacker.

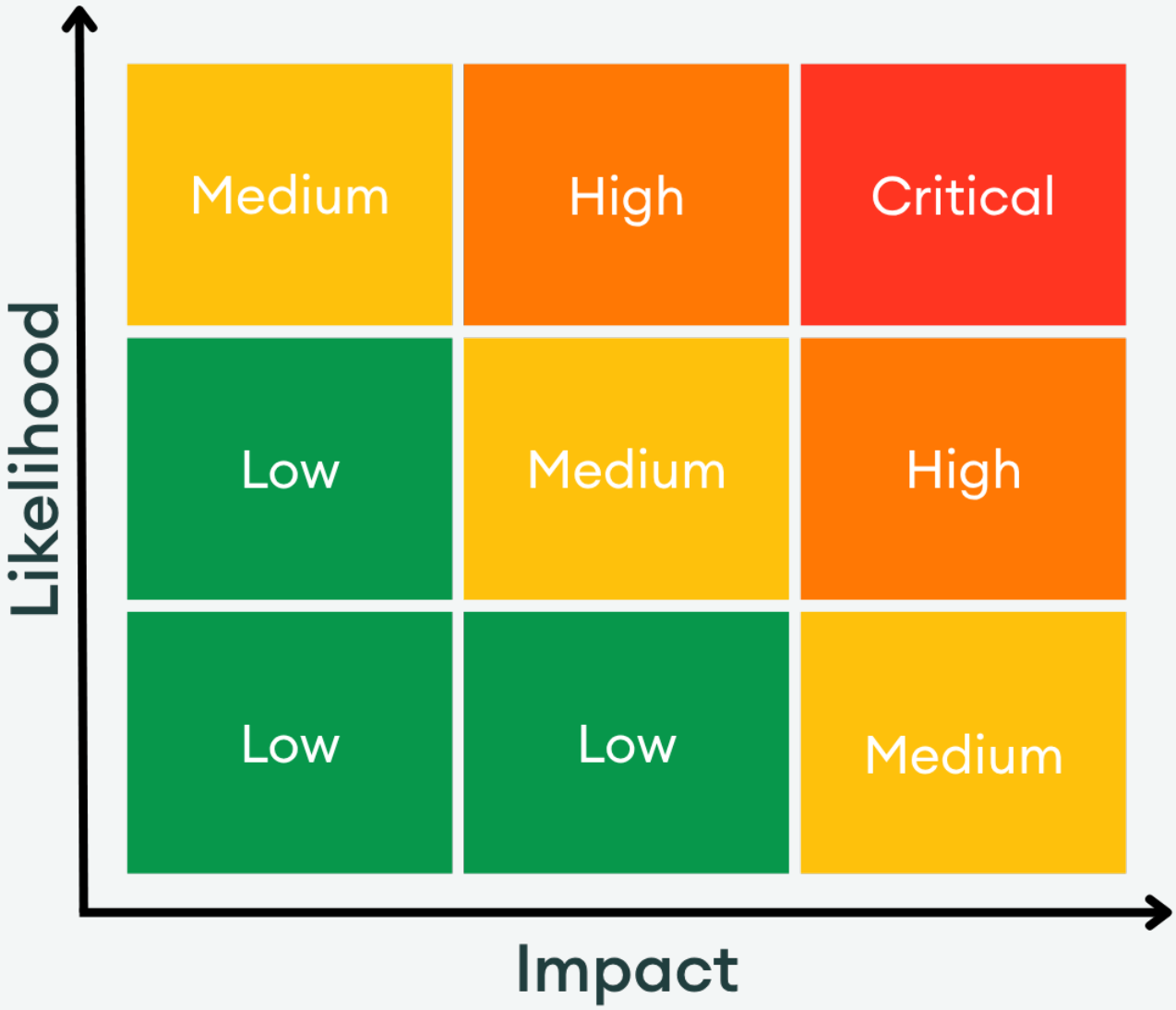
- **High Likelihood**: The vulnerability is trivially exploitable. This means it can be exploited by a wide range of actors without privileged access rights, with minimal capital requirements and low financial risks.
- **Medium Likelihood**: This type of vulnerability can be found and exploited with moderate effort. It might require a significant capital investment, but with manageable financial risk.
- **Low Likelihood**: Exploitation of these vulnerabilities is often technically unfeasible or requires highly specialized conditions. They may require extraordinary effort or a significant financial risk for an attacker, with a high chance of failure and minimal potential return.

Severity Classification Matrix

By combining **Impact** and **Likelihood**, we assign a severity level using the matrix below:

- **Critical**: High impact + high likelihood (e.g. a bug that could allow anyone to drain a substansial amount of protocol funds with minimal effort)
- **High**: High impact with medium likelihood, or medium impact with high likelihood
- **Medium**: Moderate impact and/or discoverability
- **Low**: Minimal impact or unlikely to be exploited

This structured approach helps teams prioritize fixes and mitigate the most dangerous threats first.



Severity Matrix

4. Findings

H01: The program is forked from an outdated version of the stake pool program

Status: Resolved

Impact:

Likelihood:

Severity: High

Location: https://github.com/Tempest-Finance/spl-stake-pool

Description:

The repository was forked from an upstream commit 864ba3c that predates seventeen security fixes patched in release program@v2.0.4.

These fixes address numerous security issues including stake account hijacking during cluster restarts, unauthorized stake manipulation on non-active validators, reserve depletion attacks, and validator accounting inconsistencies.

Since the fork occurred before these patches, the current codebase inherits all these vulnerabilities in their original unpatched state.

Review the complete diff for implementation details.

Recommendation:

Rebase the fork to include commit 0e562954cc280185fcc87ef01d7b and all subsequent security patches from the upstream repository.

Developer Response:

Fixed according to the recommendation.
The program now integrates all the new security changes from the v2.0.4 Stake Pool program.

L01: Unconditional Rent Transfer on Stake Account Creation

Status: Resolved

Impact: Low Medium High

Likelihood: Low Medium High

Severity: Low

Location: https://github.com/AdevarLabs/ignition-spl-stake-pool-20251215/blob/b9d44a8162fe9e4f4cf4e665282bc829d16cb89f/program/src/processor.rs#L3401-L3405

>_ processor.rs RUST

```
3399:
3400:                // Fund the PDA with rent from payer
3401:                invoke(
3402:                    &system_instruction::transfer(
3403:                        payer_info.key,
3404:                        stake_split_to.key,
3405:                        stake_rent,
3406:                    ),
3407:                    &[
```

Description:

When creating a new stake account, the code:
1. Creates the stake account via create_stake_account()
2. Transfers the full stake_rent amount from payer to the newly created account
3. **Does not check** if the account already has any lamports before transferring

This means that if the stake account already contains lamports, the code will still transfer the full stake_rent amount.

Recommendation:

Update to only transfer missing rent

Developer Response:

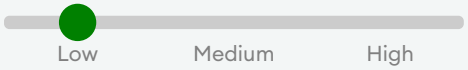
Resolved. Updated to only transfer missing rent.
User can't profit by pre-funding, they'd just pay rent themselves instead of paymaster.

L02: Users burn more pool tokens than stake lamports received when paying for the stake account rent themselves

Status: Resolved

Impact:

Likelihood:



Severity: Low

Location: https://github.com/AdevarLabs/ignition-spl-stake-pool-20251215/blob/b9d44a8162fe9e4f4cf4e665282bc829d16cb89f/program/src/processor.rs#L3454-L3463

>_ processor.rs RUST

```
3452:             let program_signer_seeds: &[&[u8]] = &[PROGRAM_SIGNER_SEED, &[program_sign
          er_bump]];
3453:
3454:             // Burn using session
3455:             invoke_signed(
3456:                 &burn(
3457:                     token_program_info.key,
3458:                     burn_from_pool_info.key,
3459:                     pool_mint_info.key,
3460:                     signer_or_session_info.key,
3461:                     Some(program_signer_info.key),
3462:                     pool_tokens_burnt,
3463:                 )?,
3464:                 &[
3465:                     burn_from_pool_info.clone(),
```

Description:

In the process_withdraw_stake function, when using the session-based withdrawal path, users that fund the stake_split_to stake account by themselves burn pool tokens based on withdraw_lamports but only receive split_lamports worth of stake, where split_lamports = withdraw_lamports - stake_rent.

This means effectively users pay pool tokens for withdraw_lamports worth of stake but only receive split_lamports worth of actual stake.

Recommendation:

Check if payer_info.key == &user;_pubkey, if that's the case, only burn split_amount.
Else, burn split_amount + required_rent.
Doing this:
1. When the user funds the PDA, they only pay for the received stake.
2. When the paymaster funds the PDA, users pay for the received stake and paymaster funding.

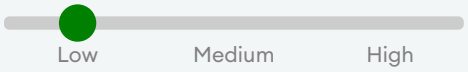
Developer Response:

Fixed according to the recommendation. The user now gets the correct amount of shares burnt.

L03: minAmountOut tokens cannot be specified in deposit/withdraw wSOL functions

Status: Resolved

Impact:



Likelihood:

Severity: Low

Location: https://github.com/AdevarLabs/ignition-spl-stake-pool-20251215/blob/5b5540e73576ff6ac846f3981de6eca3ce3cd9db/program/src/processor.rs#L4033-L4039

>_ processor.rs RUST

```
4031:                 )
4032:             }
4033:             StakePoolInstruction::DepositWsolWithSession(lamports) => {
4034:                 msg!("Instruction: DepositWsolWithSession");
4035:                 Self::process_deposit_wsol_with_session(program_id, accounts, lamports
          , None)
4036:             }
4037:             StakePoolInstruction::WithdrawWsolWithSession(amount) => {
4038:                 msg!("Instruction: WithdrawWsolWithSession");
4039:                 Self::process_withdraw_wsol_with_session(program_id, accounts, amount,
           None)
4040:             }
4041:         }
```

Description:

The DepositWsolWithSession and WithdrawWsolWithSession instructions hard-code slippage parameters to None, preventing users from setting minimum output amounts.

Unlike the regular deposit/withdraw instructions that support slippage via WithSlippage variants, Fogo session operations always pass None, exposing users to slippage risk.

Recommendation:

Allow users to provide custom minAmountOut.

Developer Response:

Fixed according to the recommendation.
The instructions now allow the caller to provide a minimum amount out.

5. Enhancement Opportunities

E01: Redundant lamport check after account closure

Location: https://github.com/AdevarLabs/ignition-spl-stake-pool-20251215/blob/5b5540e73576ff6ac846f3981de6eca3ce3cd9db/program/src/processor.rs#L2708-L2712

>_ processor.rs RUST

```
2706:             .minimum_balance(spl_token::state::Account::LEN)
2707:             .max(1)
2708:             .saturating_sub(if wsol_transient_info.lamports() > 0 {
2709:                 wsol_transient_info.lamports()
2710:             } else {
2711:                 0
2712:             });
2713:
2714:         invoke_signed(
```

Description:

After closing the transient wSOL account, the code checks if it has remaining lamports to subtract from the rent calculation.

This check is redundant because close_account transfers all lamports to the destination, leaving the closed account with zero lamports.

The conditional always evaluates to 0 and can be removed, simplifying the rent calculation logic.

Potential Benefit:

Simplifies the rent refund calculation logic and slightly reduces CUs usage.

Recommendation:

Remove the logic from the calculation.

Developer Response:

Fixed according to the recommendation.
The redundant check for remaining lamports has been removed.

E02: Removing wSOL associated token account restriction will enhance protocol flexibility

Location: https://github.com/AdevarLabs/ignition-spl-stake-pool-20251215/blob/5b5540e73576ff6ac846f3981de6eca3ce3cd9db/program/src/processor.rs#L2619-L2621

>_ processor.rs RUST

```
2617:         );
2618:
2619:         if *wsol_token_info.key != expected_wsol_ata {
2620:             msg!("`wsol_token` is not the expected ATA for the user");
2621:             return Err(ProgramError::InvalidAccountData);
2622:         }
2623:
```

Description:

The protocol currently has restrictions in place regarding the source/destination token accounts in the process_deposit_wsol_with_session and process_withdraw_wsol_with_session functions.

These restrictions do not exist in the already existing process_deposit_sol and process_withdraw_sol functions.

• In process_deposit_wsol_with_session: The wsol_token_info representing the source token account of the wSOL must be the associated token account of the session's user.

• In process_withdraw_wsol_with_session: The destination_account_info representing the recipient of the pool share tokens must be the associated token account of the session's user.

Potential Benefit:

Improve protocol flexibility and composability with other DeFi strategies that are reliant on non-associated token accounts.

Recommendation:

Remove the associated token account restrictions to allow users to use any of their token accounts, not just the ATA.

The data of the destination_account_info account can be unpacked into an Account and ownership can be checked against the session user to prevent sending stake pool shares to an unintended user.

The Mint and Authority of wsol_token_info will be checked by the SPL token processor.

Developer Response:

Fixed according to the recommendation.
The program now checks that the token account mint is WSOL, and that the owner is the user.

E03: Removing unused import will improve code clarity

Location: https://github.com/AdevarLabs/ignition-spl-stake-pool-20251215/blob/5b5540e73576ff6ac846f3981de6eca3ce3cd9db/program/src/processor.rs#L2569

>_ processor.rs RUST

```
2567:         use fogo_sessions_sdk::token::instruction::transfer_checked;
2568:         use fogo_sessions_sdk::{session::Session, token::PROGRAM_SIGNER_SEED};
2569:         use solana_program::program_pack::Pack;
2570:         use spl_associated_token_account::{
2571:             get_associated_token_address_with_program_id, tools::account::create_pda_a
          ccount,
```

Description:

The process_deposit_wsol_with_session function imports the Solana program Pack but does not use it.

Potential Benefit:

Improve code clarity.

Recommendation:

Remove the unused import.

Developer Response:

Acknowledged, as no risk is associated with this enhancement.

About Us

About Adevar Labs

Adevar Labs is a boutique blockchain security firm specializing in web3 audits.

Built by a mix of experienced professionals in traditional enterprise and crypto natives who have contributed to some of the most critical projects in blockchain infrastructure.

Our team's background spans companies like Bitdefender, Asymmetric Research, Quantstamp, Chainproof, and Juicebox, and includes experience securing smart contracts, bridges, and L1 and L2 protocols across ecosystems like Solana, Ethereum, Polkadot, Cosmos, and MultiversX.

With over 100 audits completed and a portfolio that includes custom fuzzers, exploit modeling, and runtime testing frameworks, Adevar Labs brings both depth and precision to every engagement.

Our auditors have discovered critical vulnerabilities, built high-impact tooling, and placed in top positions in premier audit competitions including Code4rena and Sherlock.

Team members hold distinctions such as PhDs in software protection, and key roles at Fortune 500 companies, and leadership of flagship conferences like ETH Bucharest.

With team members having publications with over 1,100 academic citations and trusted by projects securing over $500M in on-chain value, Adevar Labs blends elite technical rigor with real-world security impact.

We also collaborate with some of the best independent security researchers in the web3 space.

Projects may optionally request specific contributors to be part of their audit.

Audit Methodology

Our audit methodology is specialized to provide thorough security assessments of Solana programs. We focus explicitly on rigorous manual analysis, detailed threat modeling, and careful validation of implemented fixes to ensure your programs operate securely and as intended.

1. Program Context and Architecture Analysis

Our auditors begin by deeply examining your Solana program's documentation, intended functionality, and account design. We meticulously map out program interactions, instruction processing flows, and state management logic. Special attention is given to understanding how your program interfaces with critical Solana system programs, such as the SPL Token Program, Stake Program, and System Program. Last but not least we check external integrations with other Solana projects and verify if the inputs and return values are handled properly.

2. Threat Modeling

We conduct targeted threat modeling tailored specifically for Solana's execution environment. We carefully define attacker capabilities and identify potential vulnerabilities that may arise from Solana-specific issues, including but not limited to:

- Unauthorized account data manipulation
- Improper ownership or signer verification
- Misuse of Program-Derived Addresses (PDAs)

- Incorrect use of Cross-Program Invocations (CPI)
- Failure to adequately handle account privileges or account states
- Risks stemming from rent-exemption and account initialization logic

3. In-depth Manual Security Review

Our experienced auditors perform an extensive manual security review of your Rust-based Solana programs. This involves a comprehensive line-by-line inspection of source code, focusing on common Solana vulnerabilities including, but not limited to:

- Missing or insufficient ownership checks
- Inadequate signer checks
- Incorrect handling of CPI calls and invocation privileges
- Arithmetic and integer overflow or underflow errors
- Unsafe deserialization and serialization of account data structures
- Improper token transfers and SPL-token logic issues
- Logic flaws in financial operations or state transitions
- Edge-case handling in instruction input validation
- Potential denial-of-service vectors related to transaction execution and account handling

During this stage, we clearly document any discovered vulnerabilities, including detailed descriptions, precise severity ratings, and recommendations for secure implementation. In situations where it is not clear how the vulnerability might be exploited we may also include a detailed proof-of-concept exploit including code snippets and instructions on how the exploit could be performed.

4. Detailed Fix Review and Validation

After the initial audit and your team's subsequent remediation efforts, we perform a comprehensive fix review to ensure the vulnerabilities identified have been effectively resolved.

We verify each fix individually, confirming that:

- Corrections effectively eliminate the security risks
- Changes do not inadvertently introduce new vulnerabilities or regressions
- The fixes align closely with Solana best practices and secure coding guidelines

Our detailed validation ensures that security improvements are robust, complete, and aligned with best practices specific to the Solana development ecosystem.

Confidentiality Notice

This report, including its content, data, and underlying methodologies, is subject to the confidentiality and feedback provisions in your agreement with Adevar Labs. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Adevar Labs.

Legal Disclaimer

The review and this report are provided by Adevar Labs on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Adevar Labs disclaims all warranties, expressed or implied, in

connection with this report, its content, and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

You agree that access to and/or use of the report and other results of the review, including any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

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. This report is based on the scope of materials and documentation provided for a limited review at the time provided.

You acknowledge that blockchain technology remains under development and is subject to unknown risks and flaws. Adevar Labs does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open-source or third-party software, code, libraries, materials, or information accessible through the report. As with the purchase or use of a product or service in any environment, you should use your best judgment and exercise caution where appropriate.