Neon Labs - Governance Contracts

Solana Program Security Audit

Prepared by: **Halborn**

Date of Engagement: **Apr 17th, 2022 - May 8th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	05/09/2022	Pablo Gómez
0.2	Draft Review	05/11/2022	Gabi Urrutia
1.0	Remediation Plan	05/16/2022	Pablo Gómez
1.1	Remediation Plan Review	05/20/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Maciej Rak	Halborn	Maciej.rak@halborn.com
Pablo Gómez	Halborn	Pablo.gomez@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Neon Labs engaged Halborn to conduct a security audit on their Solana programs, beginning on Apr 17th, 2022 and ending on May 8th, 2022 . The security assessment was scoped to the spl-governance-addin-fixed-weights and spl-governance-addin-vesting Solana programs provided to the Halborn team. Neon Labs programs in scope are related with the general project governance and rely on spl-governance and other Solana libraries to implement custom governance process.

1.2 AUDIT SUMMARY

The team at Halborn was provided three weeks for the engagement and assigned two full-time security engineers to audit the security of the Solana programs. The security engineers are blockchain, smart contract and Solana program security experts with advanced penetration testing and smart-contract hacking skills, and deep knowledge of multiple blockchain protocols..

The purpose of this audit is to:

- Ensure that Solana program functions operate as intended
- Identify potential security issues with the Solana programs

In summary, Halborn identified some improvements to reduce the likelihood and impact of multiple risks, which has been mostly addressed by Neon Labs . The main ones are the following:

- (HAL-01): The NEON Labs team fixed the issue by adding overflows-check=true in the workspace Cargo.toml file
- (HAL-02): The NEON Labs team fixed the issue by upgrading solana-program and spl-token dependencies to their latest versions.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual view of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the program audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of programs and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into the architecture, purpose, and use of the platform.
- Solana program manual code review and walkthrough to identify any logic issue.
- Thorough assessment of safety and usage of critical Rust variables and functions in scope that could led to arithmetic vulnerabilities.
- Finding unsafe Rust code usage (cargo-geiger).
- Scanning dependencies for known vulnerabilities (cargo audit).
- Local cluster deployment (solana-test-validator)
- Scanning for common Solana vulnerabilities (soteria)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

Code repositories:

1. Neon Solana Programs

- Repository: nonlabsorg/neon-spl-governance
    - Commit ID: c0c3732cf0aa0b90527f54a0068367d8d03af748
    - Programs in scope:
        1. spl-governance-addin-vesting
        2. spl-governance-addin-fixed-weights

Out-of-scope: External libraries and financial related attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	1	1

LIKELIHOOD

IMPACT

🟨	🟨	🟥	🟥	🟪
🟩	🟨	🟨	🟥	🟥
🟩	🟩	🟨	🟨	🟥
⬜	🟩	🟩 (HAL-01)	🟨	🟨
⬜ (HAL-02)	⬜	🟩	🟩	🟨

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) CARGO OVERFLOW CHECKS MISSING	Low	SOLVED - 05/16/2022
(HAL-02) OUTDATED DEPENDENCIES VERSION	Informational	SOLVED - 05/16/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) CARGO OVERFLOW CHECKS MISSING - LOW

Description:

It was observed that there is no overflow-checks=true in any Cargo.toml file. By default, overflow checks are disabled in optimized release builds. Therefore, if there is an overflow in the release build, it will pass silently, causing unexpected behavior of an application. Even when checked arithmetic is used (checked_*), it is still recommended to have those checks in Cargo.toml.

This vulnerability was identified in the addin-vesting and the spl-governance-addin-fixed-weights programs.

Code Location:

- spl-governance-addin-vesting: neon-spl-governance/addin-vesting/program/Cargo.toml
- spl-governance-addin-fixed-weights: neon-spl-governance//program/Cargo.toml

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

Add overflow-checks=true in every Cargo.toml file.

Remediation Plan:

SOLVED: The NEON Labs team fixed the issue by adding overflows-check=true in the workspace Cargo.toml file in commit 5425078d1c45c62f92b5bb90492bbaeac751ec7b

3.2 (HAL-02) OUTDATED DEPENDENCIES VERSION - INFORMATIONAL

Description:

Software is continuously updated for multiple reasons: introducing new features, removing old ones and patching bug and vulnerabilities, to name a few. The Halborn team detected that the versions of spl-governance-addin-vesting and spl-governance-addin-fixed-wights packages used reference outdated versions of solana-program and spl-token, which could cause logic flows to malfunction.

Code Location:

Listing 1: Cargo.toml

```
[dependencies]
...
solana-program = "1.9.9"
spl-token = { version = "3.2", features = ["no-entrypoint"] }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Upgrade to the latest versions of crates:
- solana-program:1.10.13
- spl-token:3.3.0

Remediation Plan:

SOLVED: The NEON Labs team fixed the issue by updating solana-program and spl-token to their latest versions in commit 5425078d1c45c62f92b5bb90492bbaeac751ec7b

AUTOMATED TESTING

4.1 AUTOMATED ANALYSIS

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and vulnerabilities. Among the tools used was cargo-audit, a security scanner for vulnerabilities reported to the RustSec Advisory Database. All vulnerabilities published in https://crates.io are stored in a repository named The RustSec Advisory Database. cargo audit is a human-readable version of the advisory database which performs a scanning on Cargo.lock. Security Detections are only in scope. All vulnerabilities shown here were already disclosed in the above report. However, to better assist the developers maintaining this code, the auditors are including the output with the dependencies tree, and this is included in the cargo audit output to better know the dependencies affected by unmaintained and vulnerable crates.

id	package	categories
RUSTSEC-2020-0159	chrono	memory-corruption
RUSTSEC-2020-0071	time	segmentation fault

Remediation Plan:

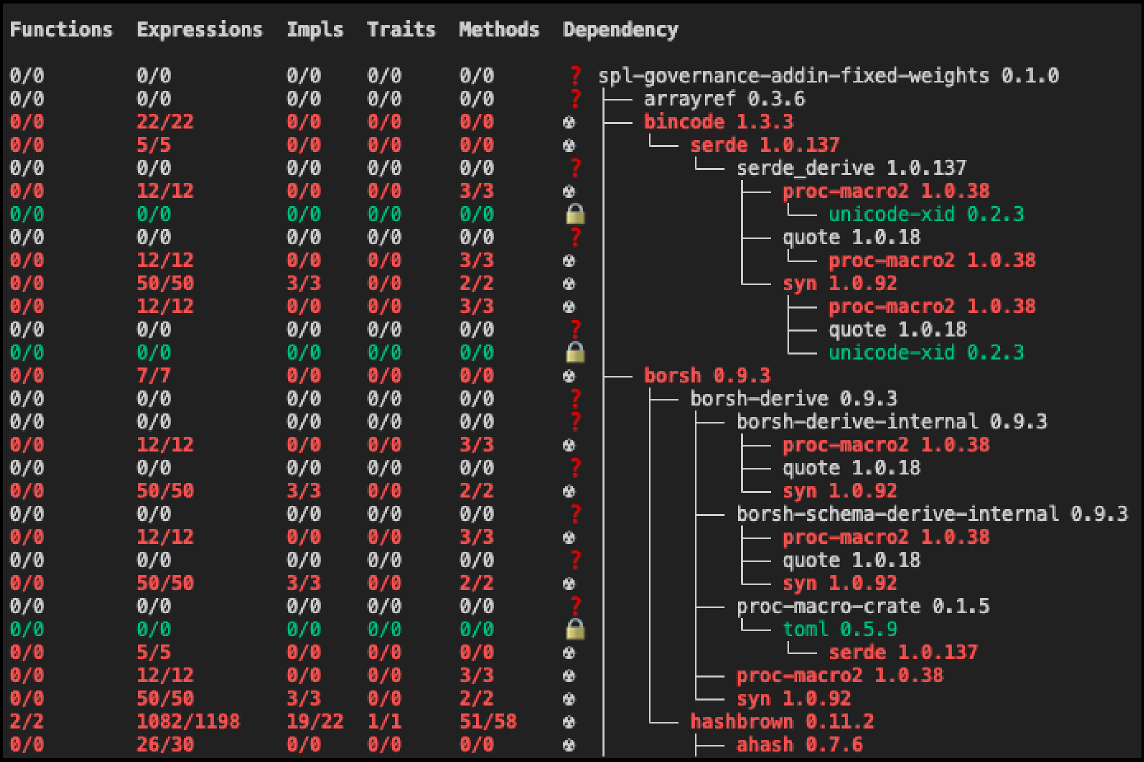
SOLVED: The NEON Labs team fixed the issue by updating affected dependencies in commit 5425078d1c45c62f92b5bb90492bbaeac751ec7b

4.2 UNSAFE RUST CODE DETECTION

Description:

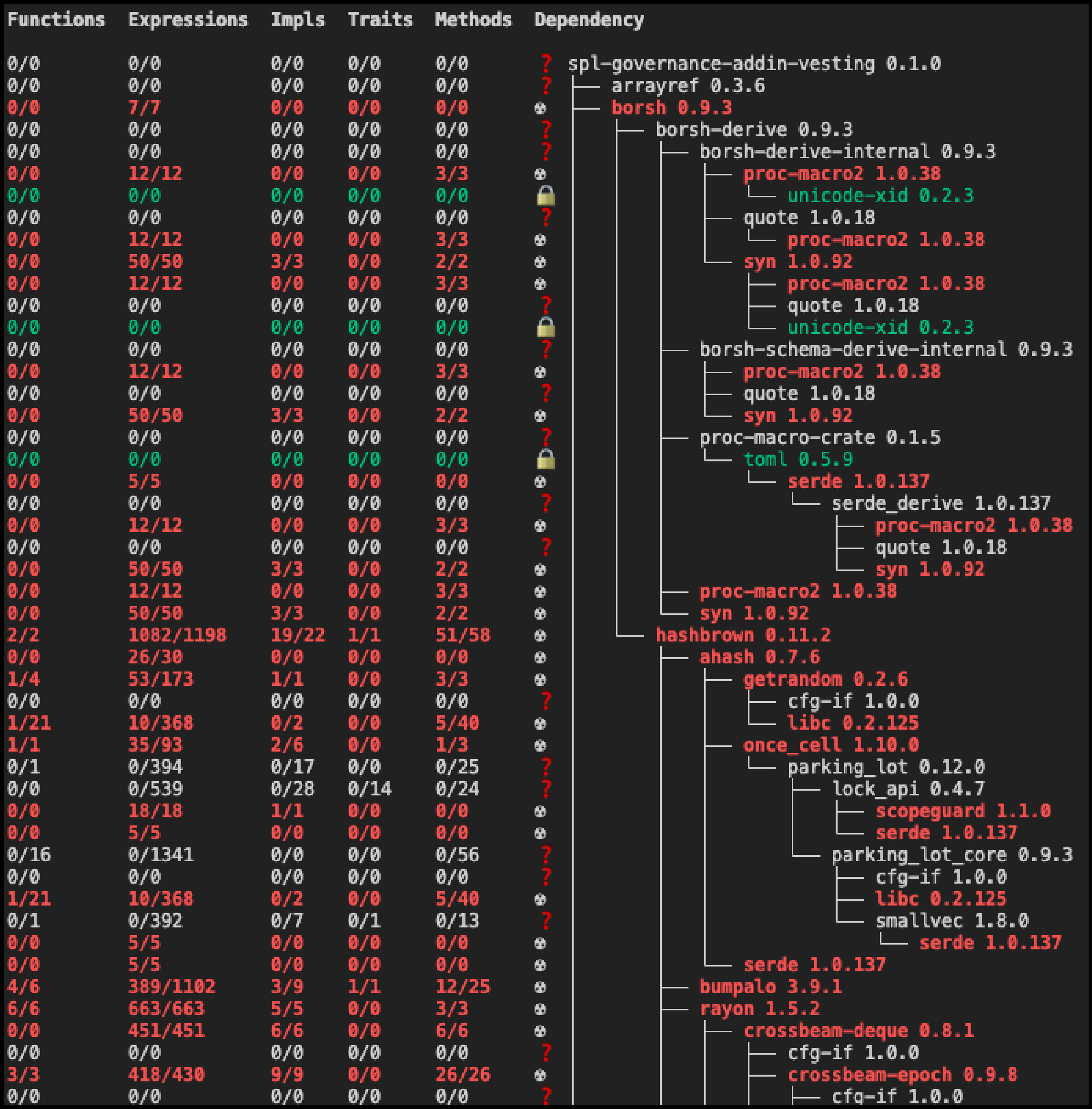
Halborn used automated security scanners to assist with the detection of well-known security issues and vulnerabilities. Among the tools used was cargo-geiger, a security tool that lists statistics related to the use of unsafe Rust code in a core Rust codebase and all its dependencies.

spl-governance-addin-fixed-weights:



```
Functions  Expressions  Impls  Traits  Methods  Dependency

0/0        0/0          0/0    0/0     0/0      ?  spl-governance-addin-fixed-weights 0.1.0
0/0        0/0          0/0    0/0     0/0      ?  ├── arrayref 0.3.6
0/0        22/22        0/0    0/0     0/0      ☢  ├── bincode 1.3.3
0/0        5/5          0/0    0/0     0/0      ☢  │   └── serde 1.0.137
0/0        0/0          0/0    0/0     0/0      ?  │       └── serde_derive 1.0.137
0/0        12/12        0/0    0/0     3/3      ☢  │           ├── proc-macro2 1.0.38
0/0        0/0          0/0    0/0     0/0      🔒 │           │   └── unicode-xid 0.2.3
0/0        0/0          0/0    0/0     0/0      ?  │           ├── quote 1.0.18
0/0        12/12        0/0    0/0     3/3      ☢  │           │   └── proc-macro2 1.0.38
0/0        50/50        3/3    0/0     2/2      ☢  │           └── syn 1.0.92
0/0        12/12        0/0    0/0     3/3      ☢  │               ├── proc-macro2 1.0.38
0/0        0/0          0/0    0/0     0/0      ?  │               ├── quote 1.0.18
0/0        0/0          0/0    0/0     0/0      🔒 │               └── unicode-xid 0.2.3
0/0        7/7          0/0    0/0     0/0      ☢  ├── borsh 0.9.3
0/0        0/0          0/0    0/0     0/0      ?  │   ├── borsh-derive 0.9.3
0/0        0/0          0/0    0/0     0/0      ?  │   │   ├── borsh-derive-internal 0.9.3
0/0        12/12        0/0    0/0     3/3      ☢  │   │   │   ├── proc-macro2 1.0.38
0/0        0/0          0/0    0/0     0/0      ?  │   │   │   ├── quote 1.0.18
0/0        50/50        3/3    0/0     2/2      ☢  │   │   │   └── syn 1.0.92
0/0        0/0          0/0    0/0     0/0      ?  │   │   ├── borsh-schema-derive-internal 0.9.3
0/0        12/12        0/0    0/0     3/3      ☢  │   │   │   ├── proc-macro2 1.0.38
0/0        0/0          0/0    0/0     0/0      ?  │   │   │   ├── quote 1.0.18
0/0        50/50        3/3    0/0     2/2      ☢  │   │   │   └── syn 1.0.92
0/0        0/0          0/0    0/0     0/0      ?  │   │   ├── proc-macro-crate 0.1.5
0/0        0/0          0/0    0/0     0/0      🔒 │   │   │   └── toml 0.5.9
0/0        5/5          0/0    0/0     0/0      ☢  │   │   │       └── serde 1.0.137
0/0        12/12        0/0    0/0     3/3      ☢  │   │   ├── proc-macro2 1.0.38
0/0        50/50        3/3    0/0     2/2      ☢  │   │   └── syn 1.0.92
2/2        1082/1198    19/22  1/1     51/58    ☢  │   └── hashbrown 0.11.2
0/0        26/30        0/0    0/0     0/0      ☢  │       ├── ahash 0.7.6
```

spl-governance-addin-vesting:



```
Functions  Expressions  Impls  Traits  Methods  Dependency

0/0        0/0          0/0    0/0     0/0       ?  spl-governance-addin-vesting 0.1.0
0/0        0/0          0/0    0/0     0/0       ?  ├── arrayref 0.3.6
0/0        7/7          0/0    0/0     0/0       ☢  ├── borsh 0.9.3
0/0        0/0          0/0    0/0     0/0       ?  │   ├── borsh-derive 0.9.3
0/0        0/0          0/0    0/0     0/0       ?  │   │   ├── borsh-derive-internal 0.9.3
0/0        12/12        0/0    0/0     3/3       ☢  │   │   │   ├── proc-macro2 1.0.38
0/0        0/0          0/0    0/0     0/0       🔒 │   │   │   │   └── unicode-xid 0.2.3
0/0        0/0          0/0    0/0     0/0       ?  │   │   │   ├── quote 1.0.18
0/0        12/12        0/0    0/0     3/3       ☢  │   │   │   │   └── proc-macro2 1.0.38
0/0        50/50        3/3    0/0     2/2       ☢  │   │   │   └── syn 1.0.92
0/0        12/12        0/0    0/0     3/3       ☢  │   │   │       ├── proc-macro2 1.0.38
0/0        0/0          0/0    0/0     0/0       ?  │   │   │       ├── quote 1.0.18
0/0        0/0          0/0    0/0     0/0       🔒 │   │   │       └── unicode-xid 0.2.3
0/0        0/0          0/0    0/0     0/0       ?  │   │   ├── borsh-schema-derive-internal 0.9.3
0/0        12/12        0/0    0/0     3/3       ☢  │   │   │   ├── proc-macro2 1.0.38
0/0        0/0          0/0    0/0     0/0       ?  │   │   │   ├── quote 1.0.18
0/0        50/50        3/3    0/0     2/2       ☢  │   │   │   └── syn 1.0.92
0/0        0/0          0/0    0/0     0/0       ?  │   │   ├── proc-macro-crate 0.1.5
0/0        0/0          0/0    0/0     0/0       🔒 │   │   │   └── toml 0.5.9
0/0        5/5          0/0    0/0     0/0       ☢  │   │   │       └── serde 1.0.137
0/0        0/0          0/0    0/0     0/0       ?  │   │   │           └── serde_derive 1.0.137
0/0        12/12        0/0    0/0     3/3       ☢  │   │   │               ├── proc-macro2 1.0.38
0/0        0/0          0/0    0/0     0/0       ?  │   │   │               ├── quote 1.0.18
0/0        50/50        3/3    0/0     2/2       ☢  │   │   │               └── syn 1.0.92
0/0        12/12        0/0    0/0     3/3       ☢  │   │   ├── proc-macro2 1.0.38
0/0        50/50        3/3    0/0     2/2       ☢  │   │   └── syn 1.0.92
2/2        1082/1198    19/22  1/1     51/58     ☢  │   └── hashbrown 0.11.2
0/0        26/30        0/0    0/0     0/0       ☢  │       ├── ahash 0.7.6
1/4        53/173       1/1    0/0     3/3       ☢  │       │   ├── getrandom 0.2.6
0/0        0/0          0/0    0/0     0/0       ?  │       │   │   ├── cfg-if 1.0.0
1/21       10/368       0/2    0/0     5/40      ☢  │       │   │   └── libc 0.2.125
1/1        35/93        2/6    0/0     1/3       ☢  │       │   ├── once_cell 1.10.0
0/1        0/394        0/17   0/0     0/25      ?  │       │   │   └── parking_lot 0.12.0
0/0        0/539        0/28   0/14    0/24      ?  │       │   │       ├── lock_api 0.4.7
0/0        18/18        1/1    0/0     0/0       ☢  │       │   │       │   ├── scopeguard 1.1.0
0/0        5/5          0/0    0/0     0/0       ☢  │       │   │       │   └── serde 1.0.137
0/16       0/1341       0/0    0/0     0/56      ?  │       │   │       └── parking_lot_core 0.9.3
0/0        0/0          0/0    0/0     0/0       ?  │       │   │           ├── cfg-if 1.0.0
1/21       10/368       0/2    0/0     5/40      ☢  │       │   │           ├── libc 0.2.125
0/1        0/392        0/7    0/1     0/13      ?  │       │   │           └── smallvec 1.8.0
0/0        5/5          0/0    0/0     0/0       ☢  │       │   │               └── serde 1.0.137
0/0        5/5          0/0    0/0     0/0       ☢  │       │   └── serde 1.0.137
4/6        389/1102     3/9    1/1     12/25     ☢  │       ├── bumpalo 3.9.1
6/6        663/663      5/5    0/0     3/3       ☢  │       ├── rayon 1.5.2
0/0        451/451      6/6    0/0     6/6       ☢  │       │   ├── crossbeam-deque 0.8.1
0/0        0/0          0/0    0/0     0/0       ?  │       │   │   ├── cfg-if 1.0.0
3/3        418/430      9/9    0/0     26/26     ☢  │       │   │   ├── crossbeam-epoch 0.9.8
0/0        0/0          0/0    0/0     0/0       ?  │       │   │   │   ├── cfg-if 1.0.0
```

More information:

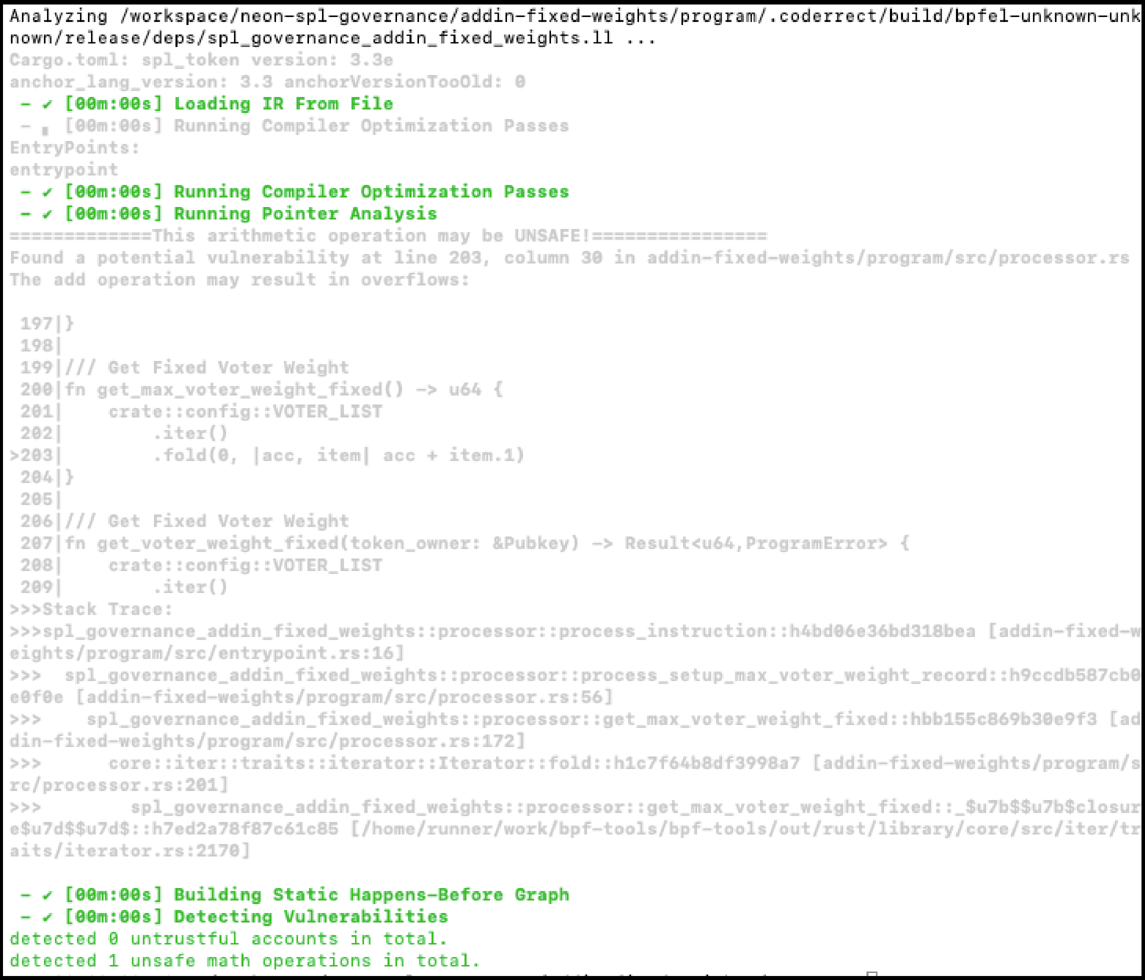
Geiger Github

4.3 AUTOMATED VULNERABILITY SCANNING

Description:

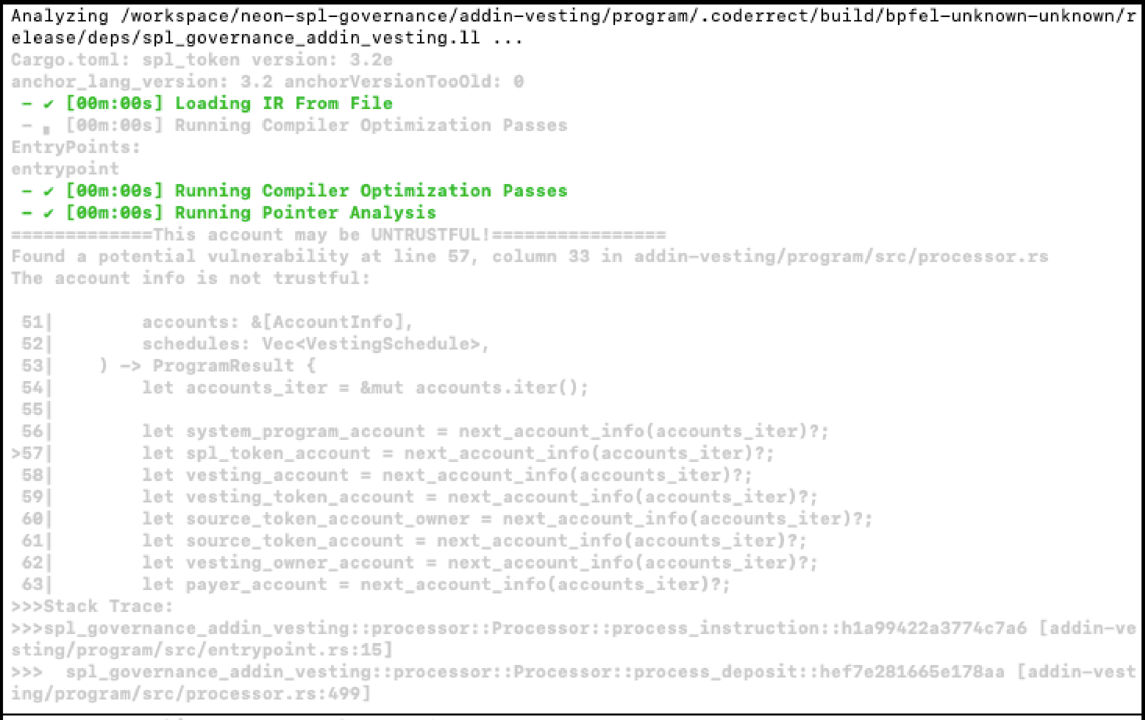
Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was Soteria, a security analysis service for Solana programs. Soteria performed a scan on all the programs and sent the compiled results to the analyzers to locate any vulnerabilities.

addin-fixed-weights:

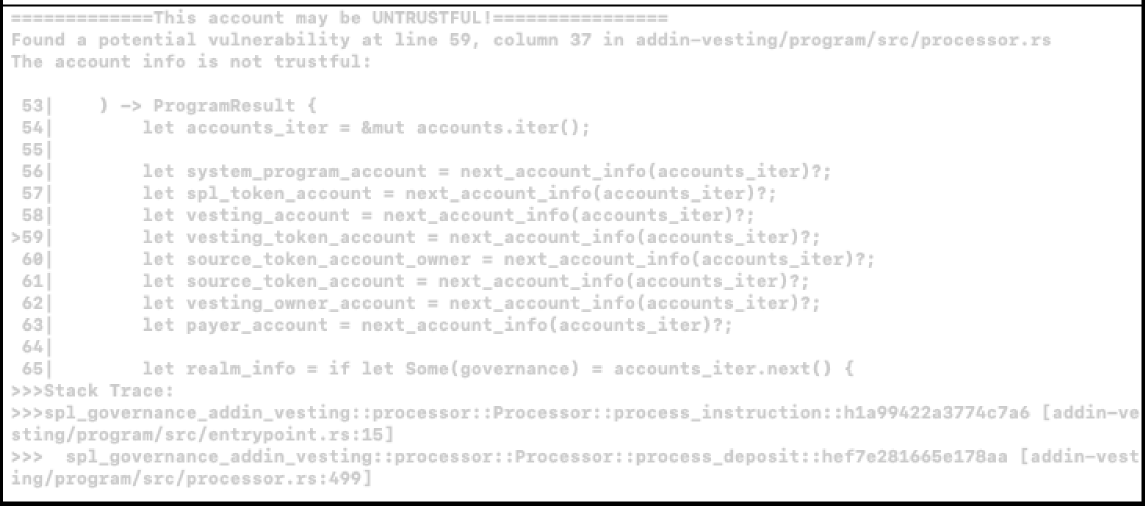


```
Analyzing /workspace/neon-spl-governance/addin-fixed-weights/program/.coderrect/build/bpfel-unknown-unk
nown/release/deps/spl_governance_addin_fixed_weights.ll ...
Cargo.toml: spl_token version: 3.3e
anchor_lang_version: 3.3 anchorVersionTooOld: 0
 - ✓ [00m:00s] Loading IR From File
 - ▖ [00m:00s] Running Compiler Optimization Passes
EntryPoints:
entrypoint
 - ✓ [00m:00s] Running Compiler Optimization Passes
 - ✓ [00m:00s] Running Pointer Analysis
=============This arithmetic operation may be UNSAFE!=================
Found a potential vulnerability at line 203, column 30 in addin-fixed-weights/program/src/processor.rs
The add operation may result in overflows:

 197|}
 198|
 199|/// Get Fixed Voter Weight
 200|fn get_max_voter_weight_fixed() -> u64 {
 201|    crate::config::VOTER_LIST
 202|        .iter()
>203|        .fold(0, |acc, item| acc + item.1)
 204|}
 205|
 206|/// Get Fixed Voter Weight
 207|fn get_voter_weight_fixed(token_owner: &Pubkey) -> Result<u64,ProgramError> {
 208|    crate::config::VOTER_LIST
 209|        .iter()
>>>Stack Trace:
>>>spl_governance_addin_fixed_weights::processor::process_instruction::h4bd06e36bd318bea [addin-fixed-w
eights/program/src/entrypoint.rs:16]
>>>  spl_governance_addin_fixed_weights::processor::process_setup_max_voter_weight_record::h9ccdb587cb0
e0f0e [addin-fixed-weights/program/src/processor.rs:56]
>>>    spl_governance_addin_fixed_weights::processor::get_max_voter_weight_fixed::hbb155c869b30e9f3 [ad
din-fixed-weights/program/src/processor.rs:172]
>>>      core::iter::traits::iterator::Iterator::fold::h1c7f64b8df3998a7 [addin-fixed-weights/program/s
rc/processor.rs:201]
>>>        spl_governance_addin_fixed_weights::processor::get_max_voter_weight_fixed::_$u7b$$u7b$closur
e$u7d$$u7d$::h7ed2a78f87c61c85 [/home/runner/work/bpf-tools/bpf-tools/out/rust/library/core/src/iter/tr
aits/iterator.rs:2170]

 - ✓ [00m:00s] Building Static Happens-Before Graph
 - ✓ [00m:00s] Detecting Vulnerabilities
detected 0 untrustful accounts in total.
detected 1 unsafe math operations in total.
```

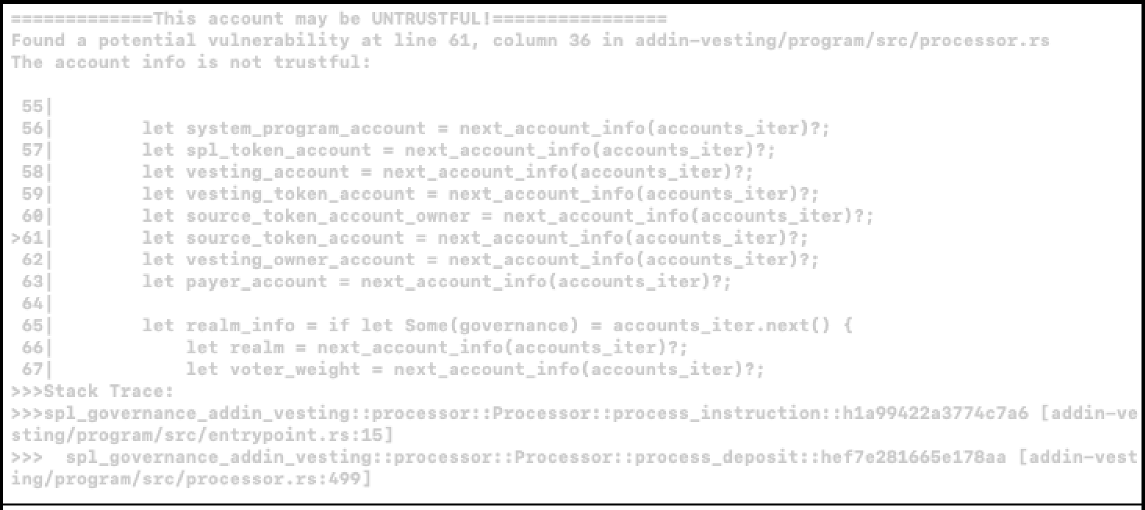
spl-governance-addin-vesting:



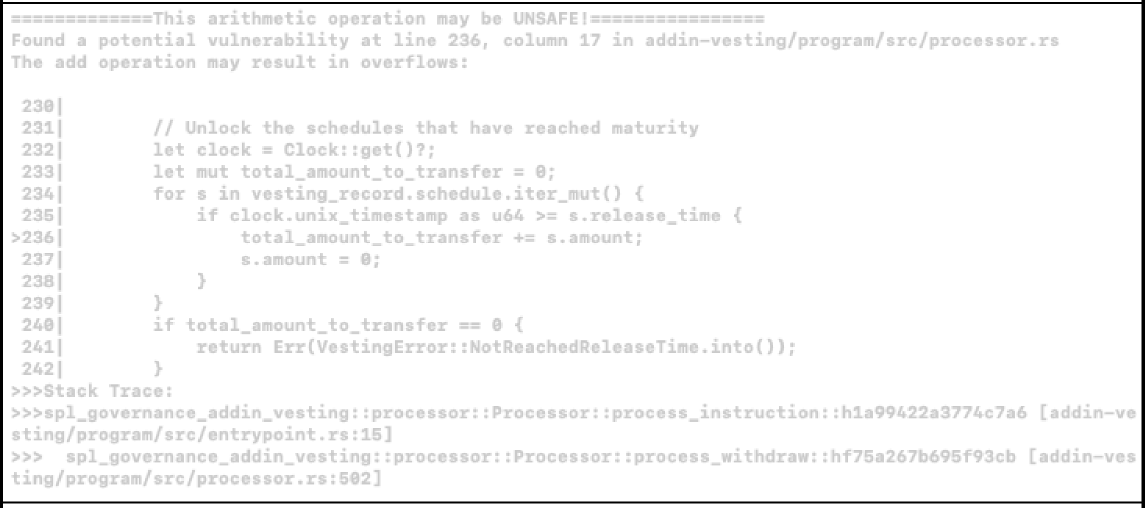
```
Analyzing /workspace/neon-spl-governance/addin-vesting/program/.coderrect/build/bpfel-unknown-unknown/r
elease/deps/spl_governance_addin_vesting.ll ...
Cargo.toml: spl_token version: 3.2e
anchor_lang_version: 3.2 anchorVersionTooOld: 0
 - ✔ [00m:00s] Loading IR From File
 - ▖ [00m:00s] Running Compiler Optimization Passes
EntryPoints:
entrypoint
 - ✔ [00m:00s] Running Compiler Optimization Passes
 - ✔ [00m:00s] Running Pointer Analysis
=============This account may be UNTRUSTFUL!================
Found a potential vulnerability at line 57, column 33 in addin-vesting/program/src/processor.rs
The account info is not trustful:

 51|        accounts: &[AccountInfo],
 52|        schedules: Vec<VestingSchedule>,
 53|    ) -> ProgramResult {
 54|        let accounts_iter = &mut accounts.iter();
 55|
 56|        let system_program_account = next_account_info(accounts_iter)?;
>57|        let spl_token_account = next_account_info(accounts_iter)?;
 58|        let vesting_account = next_account_info(accounts_iter)?;
 59|        let vesting_token_account = next_account_info(accounts_iter)?;
 60|        let source_token_account_owner = next_account_info(accounts_iter)?;
 61|        let source_token_account = next_account_info(accounts_iter)?;
 62|        let vesting_owner_account = next_account_info(accounts_iter)?;
 63|        let payer_account = next_account_info(accounts_iter)?;
>>>Stack Trace:
>>>spl_governance_addin_vesting::processor::Processor::process_instruction::h1a99422a3774c7a6 [addin-ve
sting/program/src/entrypoint.rs:15]
>>>  spl_governance_addin_vesting::processor::Processor::process_deposit::hef7e281665e178aa [addin-vest
ing/program/src/processor.rs:499]
```



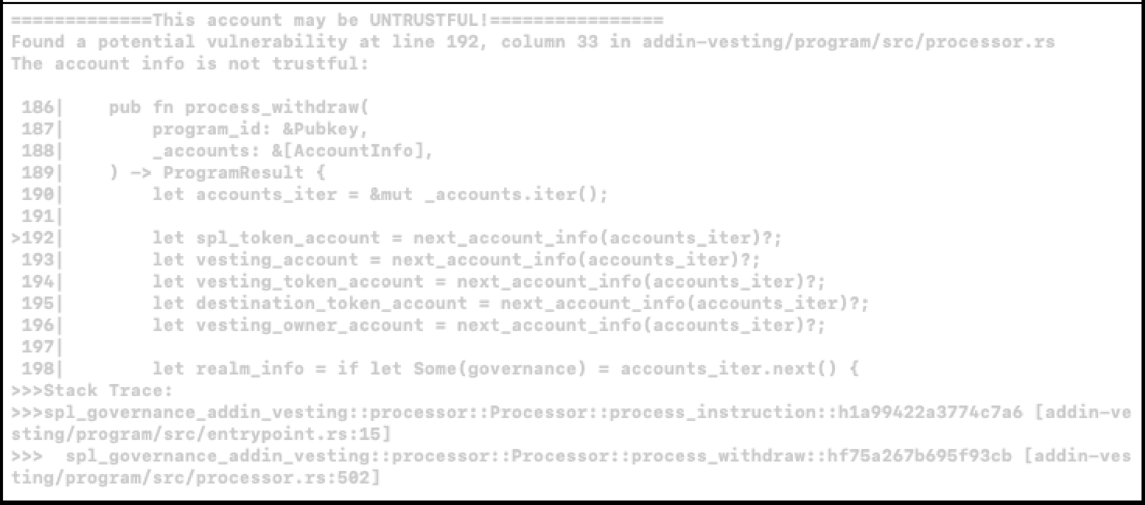
```
=============This account may be UNTRUSTFUL!================
Found a potential vulnerability at line 59, column 37 in addin-vesting/program/src/processor.rs
The account info is not trustful:

 53|    ) -> ProgramResult {
 54|        let accounts_iter = &mut accounts.iter();
 55|
 56|        let system_program_account = next_account_info(accounts_iter)?;
 57|        let spl_token_account = next_account_info(accounts_iter)?;
 58|        let vesting_account = next_account_info(accounts_iter)?;
>59|        let vesting_token_account = next_account_info(accounts_iter)?;
 60|        let source_token_account_owner = next_account_info(accounts_iter)?;
 61|        let source_token_account = next_account_info(accounts_iter)?;
 62|        let vesting_owner_account = next_account_info(accounts_iter)?;
 63|        let payer_account = next_account_info(accounts_iter)?;
 64|
 65|        let realm_info = if let Some(governance) = accounts_iter.next() {
>>>Stack Trace:
>>>spl_governance_addin_vesting::processor::Processor::process_instruction::h1a99422a3774c7a6 [addin-ve
sting/program/src/entrypoint.rs:15]
>>>  spl_governance_addin_vesting::processor::Processor::process_deposit::hef7e281665e178aa [addin-vest
ing/program/src/processor.rs:499]
```



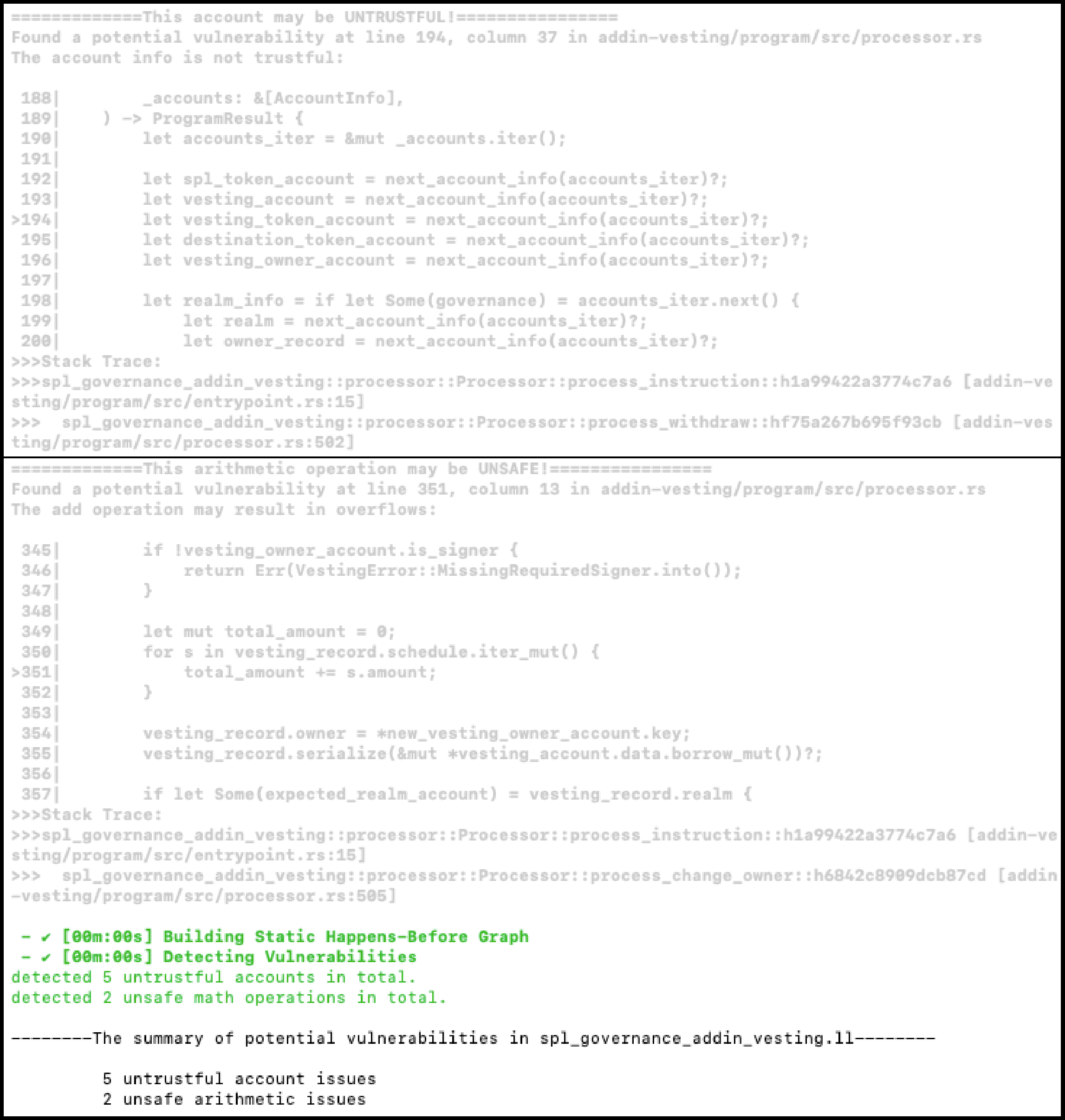
```
=============This account may be UNTRUSTFUL!================
Found a potential vulnerability at line 61, column 36 in addin-vesting/program/src/processor.rs
The account info is not trustful:

 55|
 56|        let system_program_account = next_account_info(accounts_iter)?;
 57|        let spl_token_account = next_account_info(accounts_iter)?;
 58|        let vesting_account = next_account_info(accounts_iter)?;
 59|        let vesting_token_account = next_account_info(accounts_iter)?;
 60|        let source_token_account_owner = next_account_info(accounts_iter)?;
>61|        let source_token_account = next_account_info(accounts_iter)?;
 62|        let vesting_owner_account = next_account_info(accounts_iter)?;
 63|        let payer_account = next_account_info(accounts_iter)?;
 64|
 65|        let realm_info = if let Some(governance) = accounts_iter.next() {
 66|            let realm = next_account_info(accounts_iter)?;
 67|            let voter_weight = next_account_info(accounts_iter)?;
>>>Stack Trace:
>>>spl_governance_addin_vesting::processor::Processor::process_instruction::h1a99422a3774c7a6 [addin-ve
sting/program/src/entrypoint.rs:15]
>>>  spl_governance_addin_vesting::processor::Processor::process_deposit::hef7e281665e178aa [addin-vest
ing/program/src/processor.rs:499]
```



```
=============This arithmetic operation may be UNSAFE!================
Found a potential vulnerability at line 236, column 17 in addin-vesting/program/src/processor.rs
The add operation may result in overflows:

 230|
 231|        // Unlock the schedules that have reached maturity
 232|        let clock = Clock::get()?;
 233|        let mut total_amount_to_transfer = 0;
 234|        for s in vesting_record.schedule.iter_mut() {
 235|            if clock.unix_timestamp as u64 >= s.release_time {
>236|                total_amount_to_transfer += s.amount;
 237|                s.amount = 0;
 238|            }
 239|        }
 240|        if total_amount_to_transfer == 0 {
 241|            return Err(VestingError::NotReachedReleaseTime.into());
 242|        }
>>>Stack Trace:
>>>spl_governance_addin_vesting::processor::Processor::process_instruction::h1a99422a3774c7a6 [addin-ve
sting/program/src/entrypoint.rs:15]
>>>  spl_governance_addin_vesting::processor::Processor::process_withdraw::hf75a267b695f93cb [addin-ves
ting/program/src/processor.rs:502]
```



```
=============This account may be UNTRUSTFUL!================
Found a potential vulnerability at line 192, column 33 in addin-vesting/program/src/processor.rs
The account info is not trustful:

 186|    pub fn process_withdraw(
 187|        program_id: &Pubkey,
 188|        _accounts: &[AccountInfo],
 189|    ) -> ProgramResult {
 190|        let accounts_iter = &mut _accounts.iter();
 191|
>192|        let spl_token_account = next_account_info(accounts_iter)?;
 193|        let vesting_account = next_account_info(accounts_iter)?;
 194|        let vesting_token_account = next_account_info(accounts_iter)?;
 195|        let destination_token_account = next_account_info(accounts_iter)?;
 196|        let vesting_owner_account = next_account_info(accounts_iter)?;
 197|
 198|        let realm_info = if let Some(governance) = accounts_iter.next() {
>>>Stack Trace:
>>>spl_governance_addin_vesting::processor::Processor::process_instruction::h1a99422a3774c7a6 [addin-ve
sting/program/src/entrypoint.rs:15]
>>>  spl_governance_addin_vesting::processor::Processor::process_withdraw::hf75a267b695f93cb [addin-ves
ting/program/src/processor.rs:502]
```



```
=============This account may be UNTRUSTFUL!================
Found a potential vulnerability at line 194, column 37 in addin-vesting/program/src/processor.rs
The account info is not trustful:

 188|         _accounts: &[AccountInfo],
 189|     ) -> ProgramResult {
 190|         let accounts_iter = &mut _accounts.iter();
 191|
 192|         let spl_token_account = next_account_info(accounts_iter)?;
 193|         let vesting_account = next_account_info(accounts_iter)?;
>194|         let vesting_token_account = next_account_info(accounts_iter)?;
 195|         let destination_token_account = next_account_info(accounts_iter)?;
 196|         let vesting_owner_account = next_account_info(accounts_iter)?;
 197|
 198|         let realm_info = if let Some(governance) = accounts_iter.next() {
 199|             let realm = next_account_info(accounts_iter)?;
 200|             let owner_record = next_account_info(accounts_iter)?;
>>>Stack Trace:
>>>spl_governance_addin_vesting::processor::Processor::process_instruction::h1a99422a3774c7a6 [addin-ve
sting/program/src/entrypoint.rs:15]
>>>  spl_governance_addin_vesting::processor::Processor::process_withdraw::hf75a267b695f93cb [addin-ves
ting/program/src/processor.rs:502]
=============This arithmetic operation may be UNSAFE!================
Found a potential vulnerability at line 351, column 13 in addin-vesting/program/src/processor.rs
The add operation may result in overflows:

 345|         if !vesting_owner_account.is_signer {
 346|             return Err(VestingError::MissingRequiredSigner.into());
 347|         }
 348|
 349|         let mut total_amount = 0;
 350|         for s in vesting_record.schedule.iter_mut() {
>351|             total_amount += s.amount;
 352|         }
 353|
 354|         vesting_record.owner = *new_vesting_owner_account.key;
 355|         vesting_record.serialize(&mut *vesting_account.data.borrow_mut())?;
 356|
 357|         if let Some(expected_realm_account) = vesting_record.realm {
>>>Stack Trace:
>>>spl_governance_addin_vesting::processor::Processor::process_instruction::h1a99422a3774c7a6 [addin-ve
sting/program/src/entrypoint.rs:15]
>>>  spl_governance_addin_vesting::processor::Processor::process_change_owner::h6842c8909dcb87cd [addin
-vesting/program/src/processor.rs:505]

 - ✔ [00m:00s] Building Static Happens-Before Graph
 - ✔ [00m:00s] Detecting Vulnerabilities
detected 5 untrustful accounts in total.
detected 2 unsafe math operations in total.

--------The summary of potential vulnerabilities in spl_governance_addin_vesting.ll--------

         5 untrustful account issues
         2 unsafe arithmetic issues
```

THANK YOU FOR CHOOSING

HALBORN