// HALBORN

Solana Labs - Solana v1.11.3 - v1.14.1

Solana Program Security Audit

Prepared by: **Halborn**

Date of Engagement: **September 19th, 2022 - November 21st, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/25/2022	Guillermo Alvarez
0.2	Document Updates	11/21/2022	Michael Smith
0.3	Final Draft	11/21/2022	Guillermo Alvarez
0.4	Draft Review	11/22/2022	Piotr Cielas
0.5	Draft Review	11/22/2022	Gabi Urrutia
1.0	Remediation Plan	01/05/2023	Guillermo Alvarez
1.1	Remediation Plan Review	01/05/2023	Isabel Burruezo
1.2	Remediation Plan Review	01/05/2023	Piotr Cielas
1.3	Remediation Plan Review	01/05/2023	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
Piotr Cielas	Halborn	Piotr.Cielas@halborn.com
Isabel Burruezo	Halborn	Isabel.Burruezo@halborn.com
Guillermo Alvarez	Halborn	Guillermo.Alvarez@halborn.com
Michael Smith	Halborn	Michael.Smith@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Solana is an open-source project implementing a new, high-performance, permissionless blockchain. Changes in scope affected several modules, the most important ones are briefly described. Sealevel, Solana's parallel smart contracts runtime, is a concurrent transaction processor. Transactions specify their data dependencies upfront, and dynamic memory allocation is explicit. By separating program code from the state it operates on, the runtime is able to choreograph concurrent access. Gulf Stream the transaction forwarding protocol, which is Solana's mempool-less solution for forwarding and storing transactions before processing them. The Gossip Service acts as a gateway to nodes in the control plane. Validators use the service to ensure information is available to all other nodes in a cluster. TPU (Transaction Processing Unit) is the logic of the validator responsible for block production.

Halborn conducted a security audit on the Solana v1.11.3 to v1.14.1 changes beginning on September 19th, 2022 and ending on November 21st, 2022 . The security assessment was scoped to the implementation of the updates up to v1.14.1 provided in the solana GitHub repository. Commit hashes and further details can be found in the **Scope** section of this report.

1.2 AUDIT SUMMARY

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

The purpose of this audit is to:

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

In summary, Halborn did not identify any security risk affecting the new updates introduced from version 1.11.3 to version 1.14.1. An informational finding was presented and acknowledged by the Solana team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of a manual review of the source 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 business logic, processes, 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.
- Manual program source code review to identify business logic issues.
- Mapping out possible attack vectors
- 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 runtime testing (solana-test-framework)

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.

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

1.4 SCOPE

Code repositories:

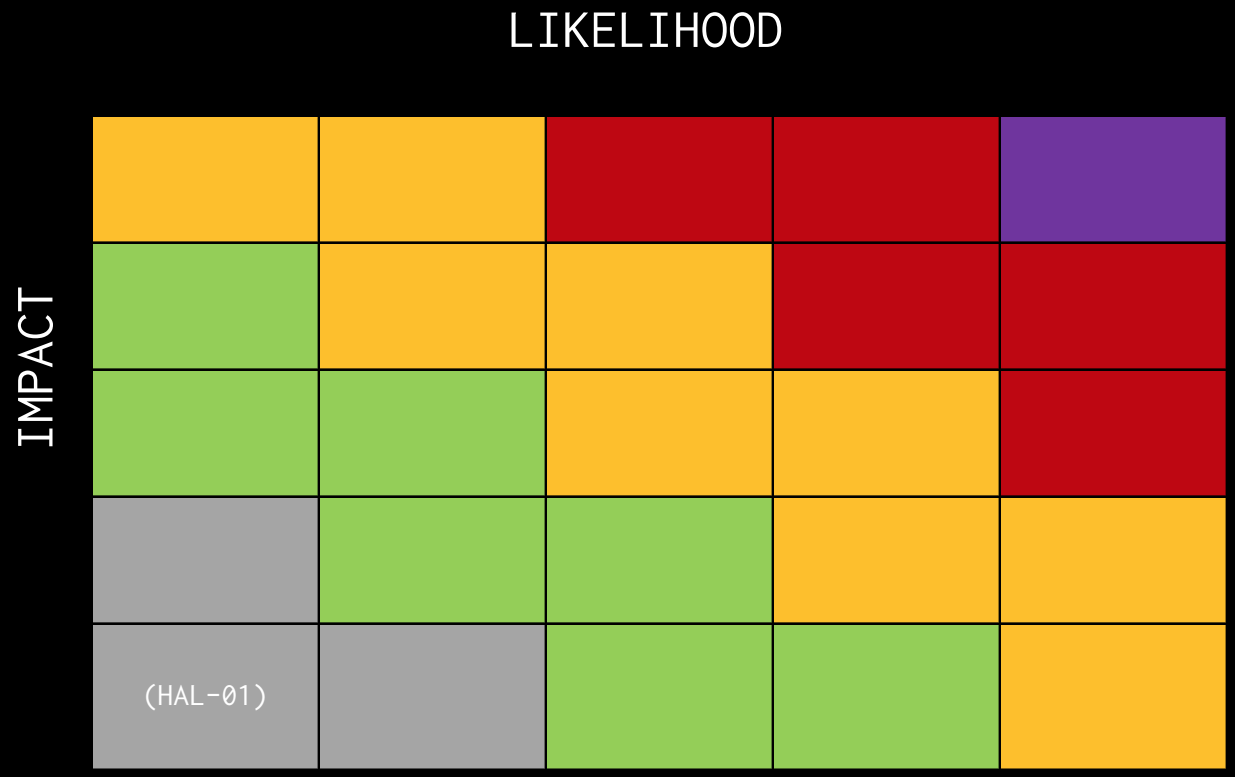
1. Solana

- Repository: solana
- Commits v1.11.3-v1.14.1:
  - start: 9798e8b1f5ba774ff700a2136f7a5531f6f4dfb9
  - end: fd5df1cf25999a50c6c2616340f95c2115b585df

Out-of-scope:
- third-party libraries and dependencies
- financial-related attacks

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	0	1



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) POSSIBLE RUST PANICS DUE TO UNSAFE UNWRAP USAGE	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) POSSIBLE RUST PANICS DUE TO UNSAFE UNWRAP USAGE - INFORMATIONAL

Description:

The use of helper methods in Rust, such as unwrap, is allowed in dev and testing environment because those methods are supposed to throw an error (also known as panic!) when called on Option::None or a Result which is not Ok. However, keeping unwrap functions in production environment is considered bad practice because they may lead to program crashes, which are usually accompanied by insufficient or misleading error messages.

Code Location:

Note: only unwraps introduced by the changes in scope are listed, justified usages such as in tests were excluded.

Listing 1

```
./banks-server/src/banks_server.rs:110              .unwrap();
./bucket_map/src/bucket.rs:283            self.index.allocate(
↳ elem_ix, elem_uid, is_resizing).unwrap();
./bucket_map/src/bucket_storage.rs:161                hdr.as_mut()
↳ .unwrap()
./core/src/broadcast_stage/broadcast_utils.rs:36     32 *
↳ ShredData::capacity(/*merkle_proof_size*/ None).unwrap() as u64;
./core/src/broadcast_stage/standard_broadcast_run.rs:72
↳     .unwrap();
./core/src/broadcast_stage/standard_broadcast_run.rs:101     let (
↳ slot, parent_slot) = self.current_slot_and_parent.unwrap();
./core/src/broadcast_stage/standard_broadcast_run.rs:120 Shredder
↳ ::new(slot, parent_slot, reference_tick, self.shred_version)
↳ .unwrap();
./core/src/cluster_info_vote_listener.rs:278    .unwrap();
./core/src/cluster_info_vote_listener.rs:392  .unwrap()
./core/src/cluster_info_vote_listener.rs:396   .unwrap()
./core/src/ledger_cleanup_service.rs:98   .unwrap();
```

```
./core/src/ledger_cleanup_service.rs:144    for (i, (slot, meta))
↳ in blockstore.slot_meta_iterator(0).unwrap().enumerate() {
./core/src/ledger_cleanup_service.rs:244          *blockstore.
↳ lowest_cleanup_slot.write().unwrap() = lowest_cleanup_slot;
./core/src/replay_stage.rs:102          .unwrap();
./core/src/replay_stage.rs:1717         let mut w_replay_stats =
↳ replay_stats.write().unwrap();
./core/src/replay_stage.rs:1718         let mut w_replay_progress =
↳  replay_progress.write().unwrap();
./core/src/replay_stage.rs:2253       let mut progress_lock =
↳ progress.write().unwrap();
./core/src/replay_stage.rs:2265       let bank = &bank_forks.read()
↳ .unwrap().get(bank_slot).unwrap();
./core/src/replay_stage.rs:2349         let bank = &bank_forks.read
↳ ().unwrap().get(bank_slot).unwrap();
./core/src/replay_stage.rs:2427         let bank = &bank_forks.read
↳ ().unwrap().get(bank_slot).unwrap();
./core/src/replay_stage.rs:2436           bank_forks.read().unwrap().
↳ root(),
./core/src/replay_stage.rs:2462                 let r_replay_stats
↳ = replay_stats.read().unwrap();
./core/src/replay_stage.rs:2464                 let
↳ r_replay_progress = replay_progress.read().unwrap();
./core/src/replay_stage.rs:2533            let
↳ block_metadata_notifier = block_metadata_notifier.read().unwrap();
./core/src/replay_stage.rs:2601          let active_bank_slots =
↳ bank_forks.read().unwrap().active_bank_slots();
./core/src/replay_stage.rs:2612                     .unwrap()
./core/src/replay_stage.rs:2614                     .unwrap()
./core/src/replay_stage.rs:2901                  &bank_forks.read()
↳ .unwrap(),
./core/src/replay_stage.rs:3127             .get(
↳ current_leader_slot.unwrap())
./core/src/replay_stage.rs:3272       .unwrap()
./core/src/retransmit_stage.rs:369        .unwrap();
./core/src/serve_repair.rs:493     let root_bank = self.bank_forks
↳ .read().unwrap().root_bank();
./core/src/serve_repair.rs:599      .unwrap()
./core/src/serve_repair.rs:1192 slot = meta.parent_slot.unwrap();
./core/src/snapshot_packager_service.rs:54    renice_this_thread(
↳ snapshot_config.packager_thread_niceness_adj).unwrap();
./core/src/system_monitor_service.rs:218 .map(|(label, val)| (
↳ label.to_string(), val.parse::<u64>().unwrap()))
./core/src/system_monitor_service.rs:376      .unwrap();
```

```
  ./core/src/tpu.rs:125       Some(bank_forks.read().unwrap().
↳ get_vote_only_mode_signal()),
  ./core/src/tpu.rs:170      .unwrap();
  ./core/src/tpu.rs:184      .unwrap();
  ./core/src/validator.rs:307     .unwrap(),
  ./core/src/validator.rs:713       *start_progress.write().unwrap() =
↳  ValidatorStartProgress::StartingServices;
  ./core/src/validator.rs:1614            let previous_start_process
↳  = *self.start_progress.read().unwrap();
  ./core/src/validator.rs:1615            *self.start_progress.write
↳ ().unwrap() = ValidatorStartProgress::LoadingLedger;
  ./core/src/validator.rs:1627                                let slot =
↳  bank_forks.read().unwrap().working_bank().slot();
  ./core/src/validator.rs:1628                                *
↳ start_progress.write().unwrap() =
  ./core/src/validator.rs:1633                      .unwrap();
  ./core/src/validator.rs:1909          let bank = bank_forks.read()
↳ .unwrap().working_bank();
  ./core/src/validator.rs:1947       *start_progress.write().unwrap()
↳ =
  ./core/src/verified_vote_packets.rs:112      let vote_account_key
↳ = self.vote_account_keys.pop().unwrap();
  ./core/src/verified_vote_packets.rs:225      let slot = vote.
↳ last_voted_slot().unwrap();
  ./core/src/verified_vote_packets.rs:283
↳ validator_votes.remove(&smallest_key).unwrap();
  ./core/src/voting_service.rs:52         let rooted_bank =
↳ bank_forks.read().unwrap().root_bank().clone();
  ./banks-server/src/banks_server.rs:110       .unwrap();
  ./gossip/src/cluster_info.rs:1687        .unwrap();
  ./gossip/src/cluster_info.rs:2565        .unwrap();
  ./gossip/src/cluster_info.rs:2580             Builder::new().name(
↳ thread_name).spawn(run_consume).unwrap()
  ./ledger/src/bank_forks_utils.rs:149    .unwrap()
  ./ledger/src/bigtable_upload.rs:197     .unwrap()
  ./ledger/src/blockstore.rs:95        .unwrap();
  ./ledger/src/blockstore.rs:100        .unwrap();
  ./ledger/src/shredder.rs:24        .unwrap();
  ./ledger/src/shredder.rs:81        .unwrap();
  ./rpc/src/rpc_service.rs:387      .on_thread_start(move ||
↳ renice_this_thread(rpc_niceness_adj).unwrap())
  ./rpc/src/rpc_service.rs:485 renice_this_thread(rpc_niceness_adj)
↳ .unwrap();
```

```
./rpc/src/rpc_subscriptions.rs:574     let blockstore = Blockstore
↳ ::open(&ledger_path).unwrap();
./runtime/src/accounts_background_service.rs:248       ).unwrap();
./runtime/src/accounts_db.rs:1713    .unwrap()
./runtime/src/accounts_db.rs:1959         .unwrap(),
./runtime/src/accounts_db.rs:3376          let mut list =
↳ slot_stores.write().unwrap();
./runtime/src/accounts_index.rs:1361        lock.as_ref().unwrap().
↳ get_internal(pubkey, |entry| {
./runtime/src/bank/sysvar_cache.rs:8        let mut sysvar_cache =
↳  self.sysvar_cache.write().unwrap();
./runtime/src/bank_client.rs:317       .unwrap();
./runtime/src/hardened_unpack.rs:341       sender.send(
↳ entry_path_buf).unwrap();
./runtime/src/in_mem_accounts_index.rs:1056       let mut
↳ possible_evictions = self.possible_evictions.write().unwrap();
./runtime/src/in_mem_accounts_index.rs:1067        let map = self.
↳ map_internal.read().unwrap();
./runtime/src/in_mem_accounts_index.rs:1089
↳ possible_evictions.get_possible_evictions().unwrap()
./runtime/src/read_only_accounts_cache.rs:114     let (pubkey,
↳ slot) = match self.queue.lock().unwrap().get_first() {
./runtime/src/snapshot_utils.rs:1572    let open_file = || File::
↳ open(&snapshot_tar).unwrap();
./runtime/src/snapshot_utils.rs:1577            zstd::stream::read
↳ ::Decoder::new(BufReader::new(open_file())).unwrap(),
./runtime/src/snapshot_utils.rs:1580            SharedBuffer::new(
↳ lz4::Decoder::new(BufReader::new(open_file())).unwrap())
./runtime/src/verify_accounts_hash_in_background.rs:45      *self.
↳ thread.lock().unwrap() = Some(start());
./runtime/src/verify_accounts_hash_in_background.rs:64     let mut
↳  lock = self.thread.lock().unwrap();
./runtime/src/verify_accounts_hash_in_background.rs:68     let
↳ result = lock.take().unwrap().join().unwrap();
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended not to use the unwrap function in the production environment because its use causes panic! and may crash any affected module, program or in the worst case the runtime without verbose error messages. Crashing the system will result in a loss of availability and, in some cases, even private information stored in the state. Some alternatives are possible, such as propagating the error with ? instead of unwrapping, or using the error-chain crate for errors.

Remediation Plan:

ACKNOWLEDGED : The Solana team acknowledged this issue.

MANUAL TESTING

In the manual testing phase, the following scenarios were simulated. The scenarios listed below were selected based on the severity of the vulnerabilities Halborn was testing the program for.

4.1 LOSS OF FUNDS

Description:

Commit 48862c575a1f62f2d010d7568cbe0279b2e51f62 introduced a new Redelegate instruction to the pre-existing Stake program. When a stake account is redelegated, the delegated lamports from the source stake account are transferred to a new stake account. The new instruction was tested to ensure that accounts were properly validated, staked lamports cannot be locked in edge-cases and implemented business logic cannot be bypassed.

Results:

```
*  Executing task: cargo test --package solana-stake-program --lib -- stake_instruction::tests::hal_test_redelegate --nocapture

   Compiling solana-stake-program v1.14.1 (/Users/guillermo/halborn/solana-pr-v1.14.1/programs/stake)
    Finished test [unoptimized + debuginfo] target(s) in 2.18s
     Running unittests src/lib.rs (target/debug/deps/solana_stake_program-934dcf6f164b3c9b)

running 2 tests
To redelegate without staker signature
Is expected result `InstructionError::MissingRequiredSignature`: true
To redelegate without staker signature
Is expected result `InstructionError::MissingRequiredSignature`: true
To redelegate with new invalid vote account
Is expected result `InstructionError::AccountAlreadyInitialized)`: true
To redelegate with new invalid vote account
Is expected result `InstructionError::AccountAlreadyInitialized)`: true
```

No code vulnerabilities were identified.

4.2 CONSENSUS

Description:

Commit 8d69e8d44772d2c2be77e99cca8b685e2ce4ba66 introduced CompactVoteStateUpdate and CompactUpdateVoteStateSwitch instructions. Several tests were performed to ensure that malicious compacted vote state updates did not have any impact over the consensus mechanism.

Results:

```
running 1 test
Vote ix: Instruction { program_id: Vote111111111111111111111111111111111111111, accounts: [AccountMeta { pubkey: 1111111QLbz7JHiBTspS962RLKV8GndWFwiEaqKM, is_signer: false, is_writable: true }, AccountMeta { pubkey: 1111111ogCyDbaRMvkdsHB3qfdyFYaG1WtRUAfdh, is_signer: true, is_writable: false }], data: [12, 0, 0, 0, 255, 255, 255, 255, 255, 255, 255, 255, 3, 0, 3, 1, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] }
Vote ix state switch: Instruction { program_id: Vote111111111111111111111111111111111111111, accounts: [AccountMeta { pubkey: 11111112D1oxKts8YPdTJRG5FzxTNpMtWmq8hkVx3, is_signer: false, is_writable: true }, AccountMeta { pubkey: 11111112cMQwSC9qirWGjZM6gLGwW69X22mqwLLGP, is_signer: true, is_writable: false }], data: [13, 0, 0, 0, 255, 255, 255, 255, 255, 255, 255, 255, 3, 0, 3, 1, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2] }
Vote: Ok(ParsedInstructionEnum { instruction_type: "compactupdatevotestate", info: Object({"voteAccount": String("1111111QLbz7JHiBTspS962RLKV8GndWFwiEaqKM"), "voteAuthority": String("1111111ogCyDbaRMvkdsHB3qfdyFYaG1WtRUAfdh"), "voteStateUpdate": Object({"hash": String("11111111111111111111111111111111"), "lockouts": Array([Object({"confirmation_count": Number(3), "slot": Number(0)}), Object({"confirmation_count": Number(2), "slot": Number(1)}), Object({"confirmation_count": Number(1), "slot": Number(2)})]), "root": Null, "timestamp": Null})}) })
Is `InstructionError::InvalidInstructionData`: true
test parse_vote::test::hal_compacted_vote_state ... ok
```

No code vulnerabilities were identified.

4.3 BUSINESS PROCESS DESIGN

Description:

Commit bf225bae738c071bd02ec79ec0b8ddfc1679a766 implements additional restrictions over RentPaying accounts. Before RentPaying accounts could continue as rent paying unless they were resized, in such case they would need to be rent exempt. However, there was still an edge-case that was possible, RentPaying accounts stay as rent paying if they were credited below the rent-exempt threshold. We performed several tests to ensure that the implemented workflow could not be circumvented.

Results:

```
halborn/solana-pr-v1.14.1 [ver1.14.1●] » export RUST_LOG=trace && cargo test --package solana-runtime --lib -- account_rent_state::tests::hal_test_rent_paying --exact --nocapture
   Compiling solana-runtime v1.14.1 (/Users/guillermo/halborn/solana-pr-v1.14.1/runtime)
    Finished test [unoptimized + debuginfo] target(s) in 3.99s
     Running unittests src/lib.rs (target/debug/deps/solana_runtime-348aa2a29ecc9a70)

running 1 test
Transition with Account resize and lamports balance change
Is Rent Paying allowed: false
test account_rent_state::tests::hal_test_rent_paying ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 802 filtered out; finished in 0.00s
```

No code vulnerabilities were identified.

4.4 DENIAL OF SERVICE

Description:

In pull request 26851 it was identified that the runtime may misidentify stake accounts as newly created accounts and delete them causing the bank to crash. This and similar scenarios we're tested to ensure the fix

resolved the issue.

Results:

```
running 1 test
Rent collected from: SnsXTGj4nbyLFFVxVZzQ81vMwRrjjZHXwjxnryxgiuk
Rent not collected from: AKevGDK2kUxtoFvou6RoKgvqGPrWLnR7WHHtih38y7AB
Rent not collected from: B9Vw8R42AwnkTD7FcQdAa8u82ox7WzAaVzkihq9zqgu9
Rent not collected from: FZkBd8BG2NcVFVAnbNkF7ixsYdTA2zjaeoWKmKPtF9ZW
test accounts::tests::halborn_collect_accounts_to_store_not_preserve_rent_epoch ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 805 filtered out; finished in 0.01s

 *  Terminal will be reused by tasks, press any key to close it.
```

No code vulnerabilities were identified.

AUTOMATED TESTING

5.1 AUTOMATED ANALYSIS

Description:

Halborn used automated security scanners to assist with the 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.

Results:

ID	package	Short Description
RUSTSEC-2020-0071	time	Potential segfault in the time crate
RUSTSEC-2021-0139	ansi_term	ansi_term is unmaintained
RUSTSEC-2020-0016	net2	net2 crate has been deprecated
RUSTSEC-2021-0127	serde_cbor	serde_cbor is unmaintained

5.2 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 usage of unsafe Rust code in a core Rust codebase and all its dependencies.

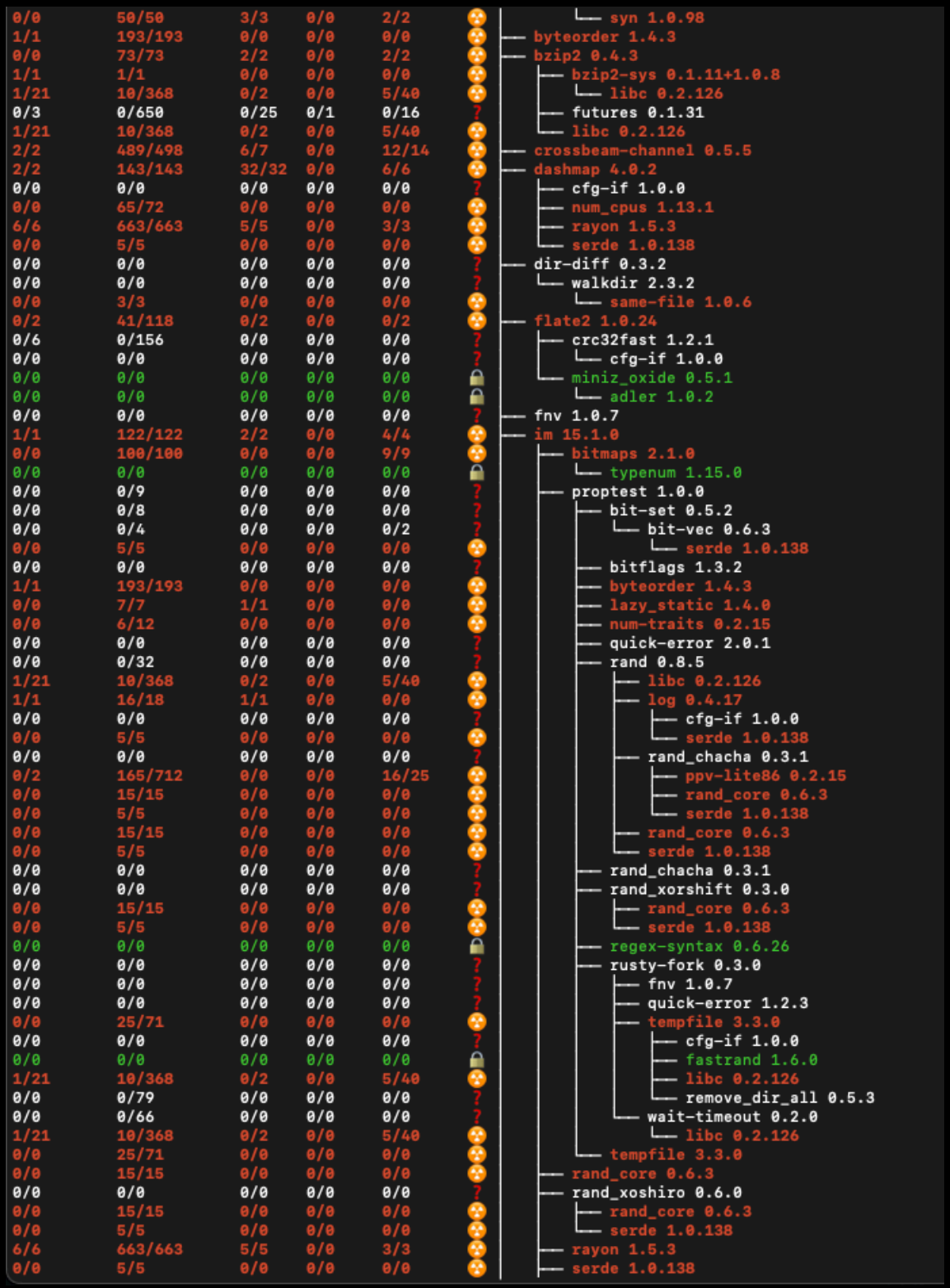
Results:

Runtime

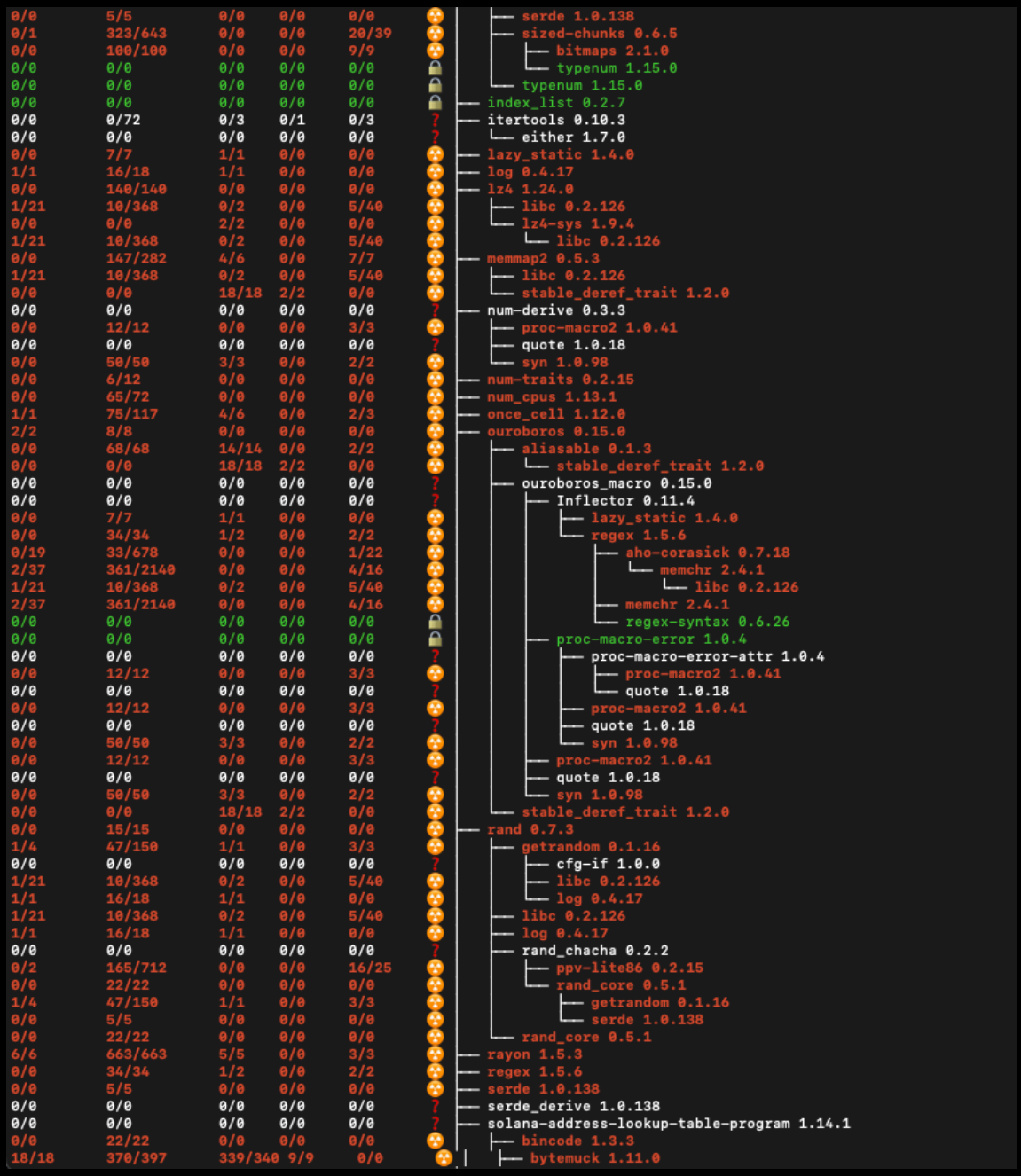
AUTOMATED TESTING

Functions	Expressions	Impls	Traits	Methods	Dependency
0/0	35/35	0/0	0/0	0/0	solana-runtime 1.14.1
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 1.0.138
0/0	0/0	0/0	0/0	0/0	│ └── serde_derive 1.0.138
0/0	12/12	0/0	0/0	3/3	│ ├── proc-macro2 1.0.41
0/0	4/4	0/0	0/0	0/0	│ │ └── unicode-ident 1.0.2
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 1.0.41
0/0	50/50	3/3	0/0	2/2	│ └── syn 1.0.98
0/0	12/12	0/0	0/0	3/3	│ ├── proc-macro2 1.0.41
0/0	0/0	0/0	0/0	0/0	│ ├── quote 1.0.18
0/0	4/4	0/0	0/0	0/0	│ └── unicode-ident 1.0.2
2/78	29/3973	0/0	0/0	0/0	├── blake3 1.3.1
0/0	0/0	0/0	0/0	0/0	│ ├── arrayref 0.3.6
2/2	350/350	2/2	0/0	7/7	│ ├── arrayvec 0.7.2
0/0	5/5	0/0	0/0	0/0	│ │ └── serde 1.0.138
0/0	0/0	0/0	0/0	0/0	│ ├── cfg-if 1.0.0
0/0	0/0	0/0	0/0	0/0	│ ├── constant_time_eq 0.1.5
0/0	0/0	0/0	0/0	0/0	│ ├── digest 0.10.3
0/0	16/16	0/0	0/0	0/0	│ │ ├── block-buffer 0.10.2
1/1	292/292	20/20	8/8	5/5	│ │ │ └── generic-array 0.14.5
0/0	5/5	0/0	0/0	0/0	│ │ │ ├── serde 1.0.138
0/0	0/0	0/0	0/0	0/0	│ │ │ └── typenum 1.15.0
0/0	0/0	0/0	0/0	0/0	│ │ ├── crypto-common 0.1.3
1/1	292/292	20/20	8/8	5/5	│ │ │ ├── generic-array 0.14.5
0/0	15/15	0/0	0/0	0/0	│ │ │ ├── rand_core 0.6.3
1/4	47/149	1/1	0/0	3/3	│ │ │ │ ├── getrandom 0.2.3
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 0.2.126
0/0	5/5	0/0	0/0	0/0	│ │ │ │ └── serde 1.0.138
0/0	0/0	0/0	0/0	0/0	│ │ │ └── typenum 1.15.0
0/0	3/3	0/0	0/0	0/0	│ │ └── subtle 2.4.1
6/6	663/663	5/5	0/0	3/3	│ └── rayon 1.5.3
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	421/433	9/9	0/0	26/26	│ │ ├── crossbeam-epoch 0.9.5
0/0	0/0	0/0	0/0	0/0	│ │ │ ├── cfg-if 1.0.0
4/4	85/85	14/14	0/0	2/2	│ │ │ ├── crossbeam-utils 0.8.8
0/0	0/0	0/0	0/0	0/0	│ │ │ │ ├── cfg-if 1.0.0
0/0	7/7	1/1	0/0	0/0	│ │ │ │ └── lazy_static 1.4.0
0/0	0/49	0/6	0/0	0/3	│ │ │ │ └── spin 0.5.2
0/0	7/7	1/1	0/0	0/0	│ │ │ ├── lazy_static 1.4.0
0/0	0/0	0/0	0/0	0/0	│ │ │ ├── memoffset 0.6.4
0/0	18/18	1/1	0/0	0/0	│ │ │ └── scopeguard 1.1.0
4/4	85/85	14/14	0/0	2/2	│ │ └── crossbeam-utils 0.8.8
0/0	0/0	0/0	0/0	0/0	│ │ ├── either 1.7.0
0/0	5/5	0/0	0/0	0/0	│ │ │ └── serde 1.0.138
5/5	485/488	2/2	0/0	20/20	│ │ └── rayon-core 1.9.2
2/2	489/498	6/7	0/0	12/14	│ │ ├── crossbeam-channel 0.5.5
0/0	0/0	0/0	0/0	0/0	│ │ │ ├── cfg-if 1.0.0
4/4	85/85	14/14	0/0	2/2	│ │ │ └── crossbeam-utils 0.8.8
0/0	451/451	6/6	0/0	6/6	│ │ ├── crossbeam-deque 0.8.1
4/4	85/85	14/14	0/0	2/2	│ │ ├── crossbeam-utils 0.8.8
0/0	65/72	0/0	0/0	0/0	│ │ ├── num_cpus 1.13.1
1/21	10/368	0/2	0/0	5/40	│ │ │ └── libc 0.2.126
2/2	206/206	0/0	0/0	7/7	├── bv 0.11.1
0/0	5/5	0/0	0/0	0/0	│ └── serde 1.0.138
18/18	370/397	339/340	9/9	0/0	├── bytemuck 1.11.0
0/0	0/0	0/0	0/0	0/0	│ └── bytemuck_derive 1.1.0
0/0	12/12	0/0	0/0	3/3	│ ├── proc-macro2 1.0.41
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 1.0.98
1/1	193/193	0/0	0/0	0/0	├── byteorder 1.4.3

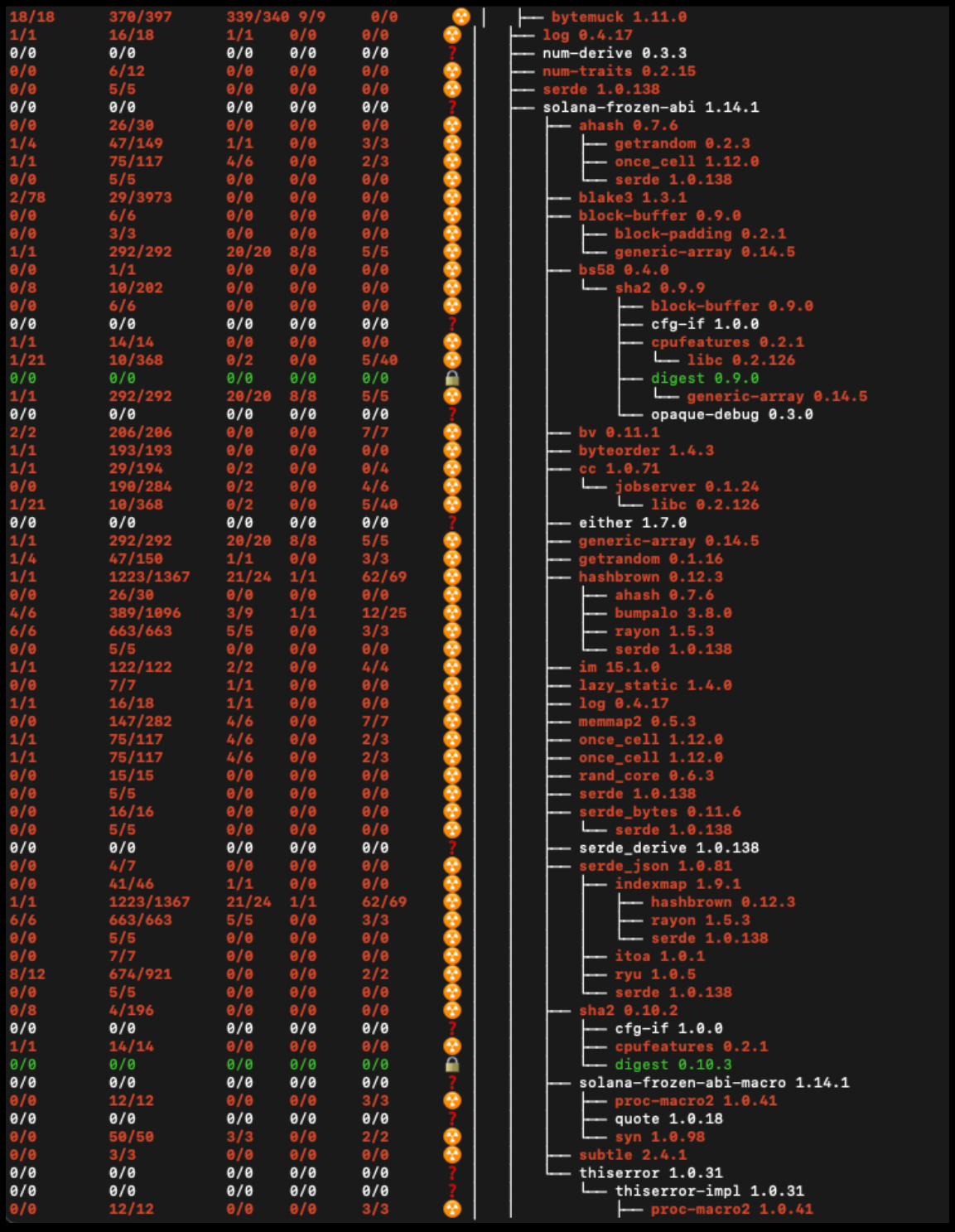
AUTOMATED TESTING



AUTOMATED TESTING

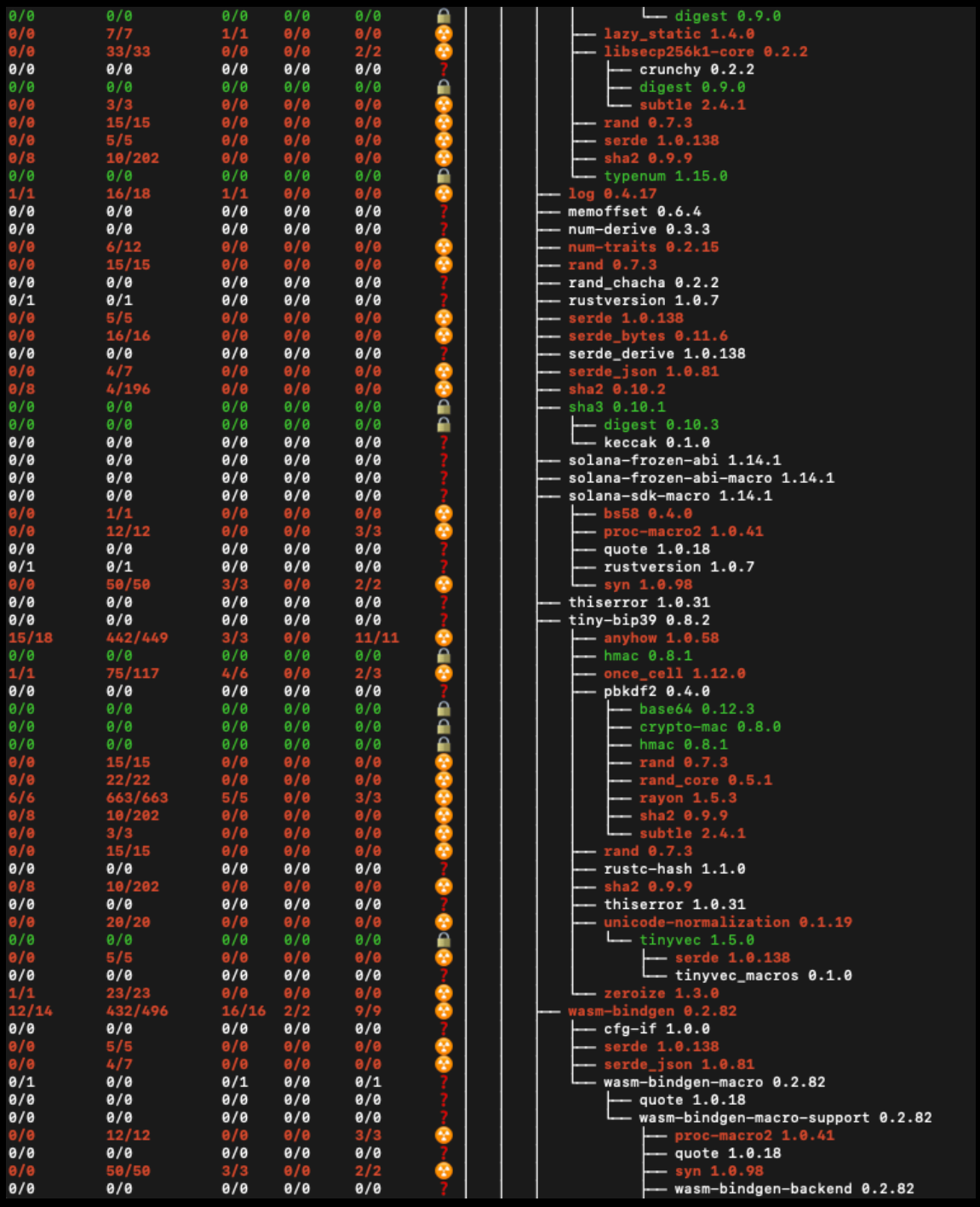


AUTOMATED TESTING

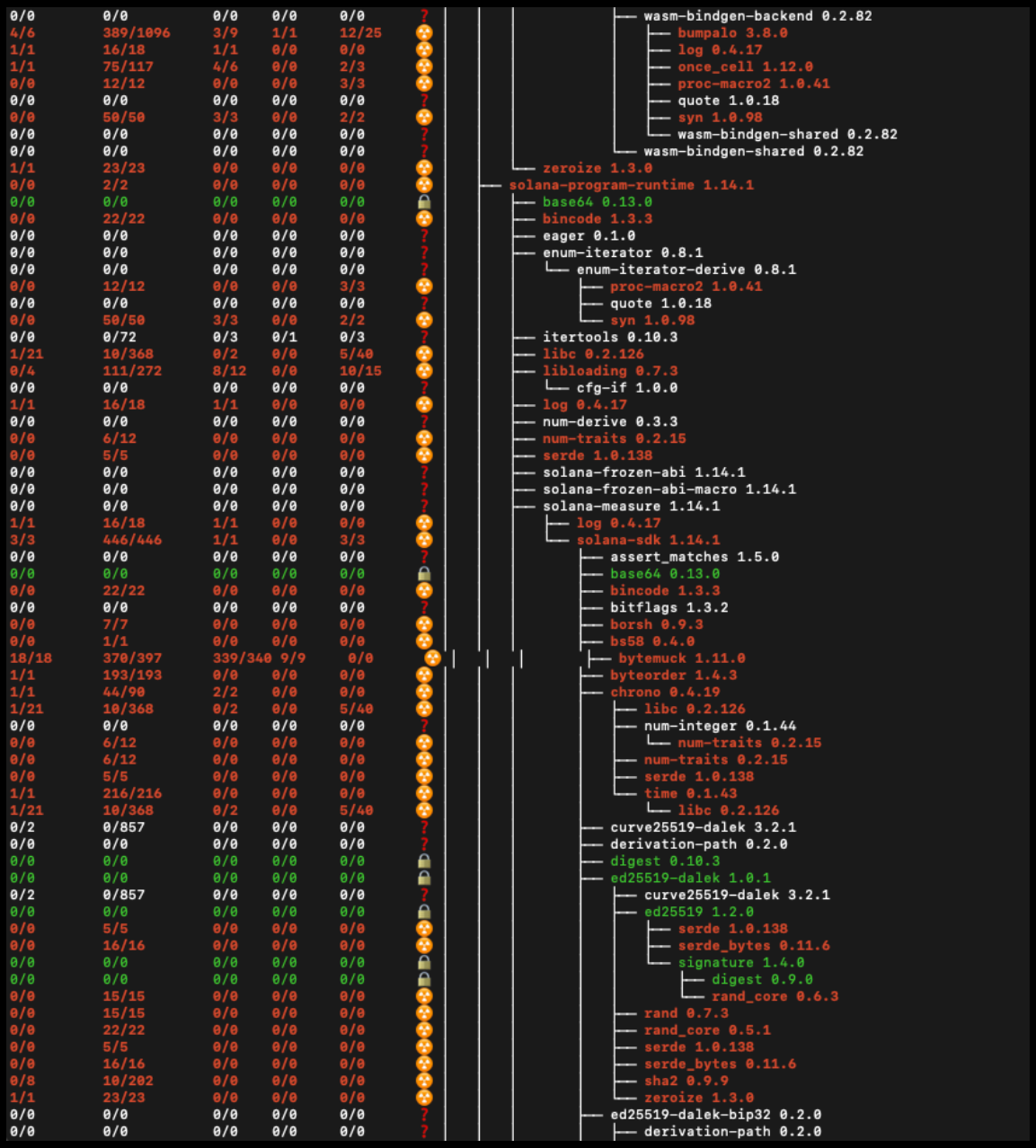


AUTOMATED TESTING

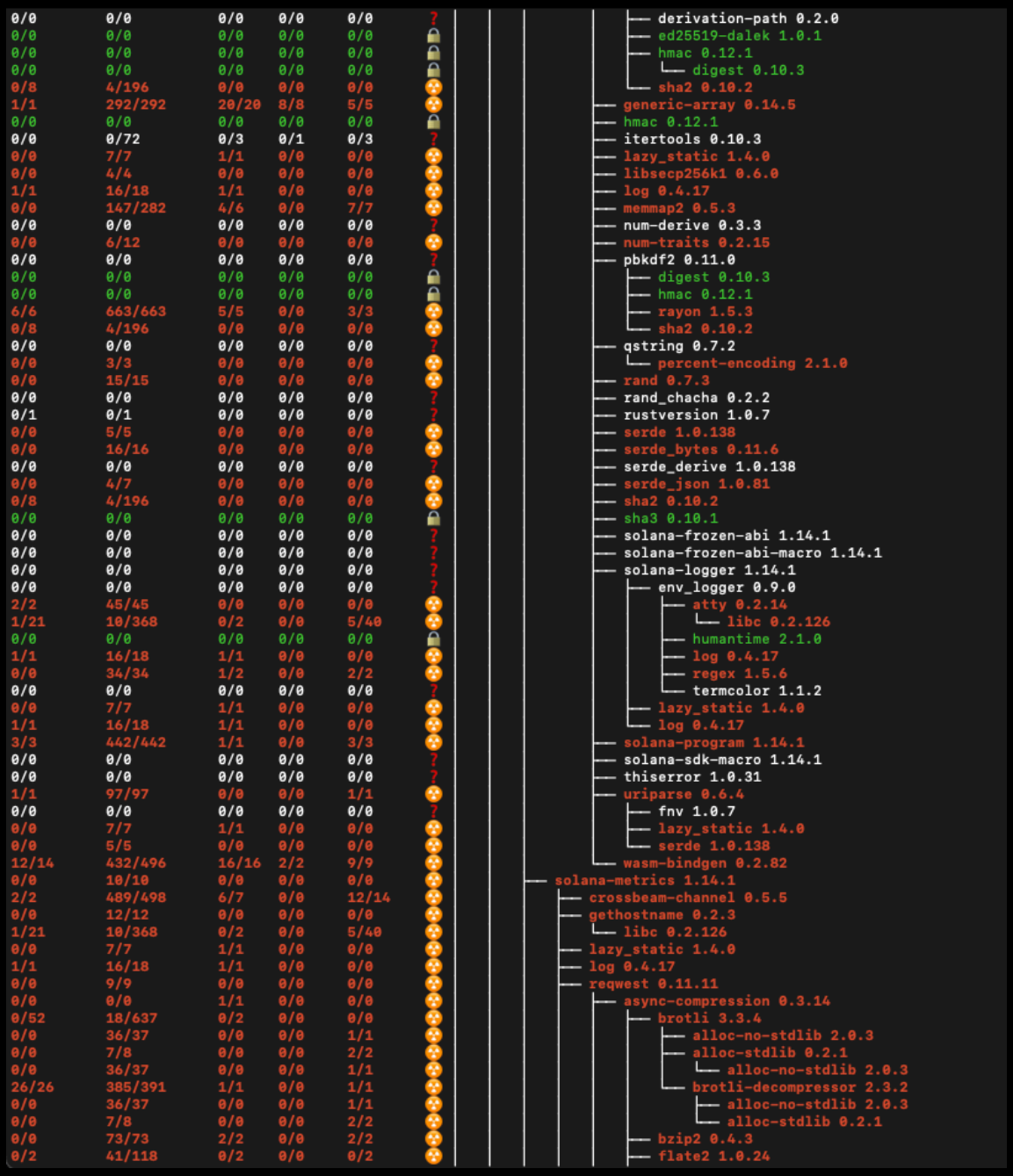
AUTOMATED TESTING



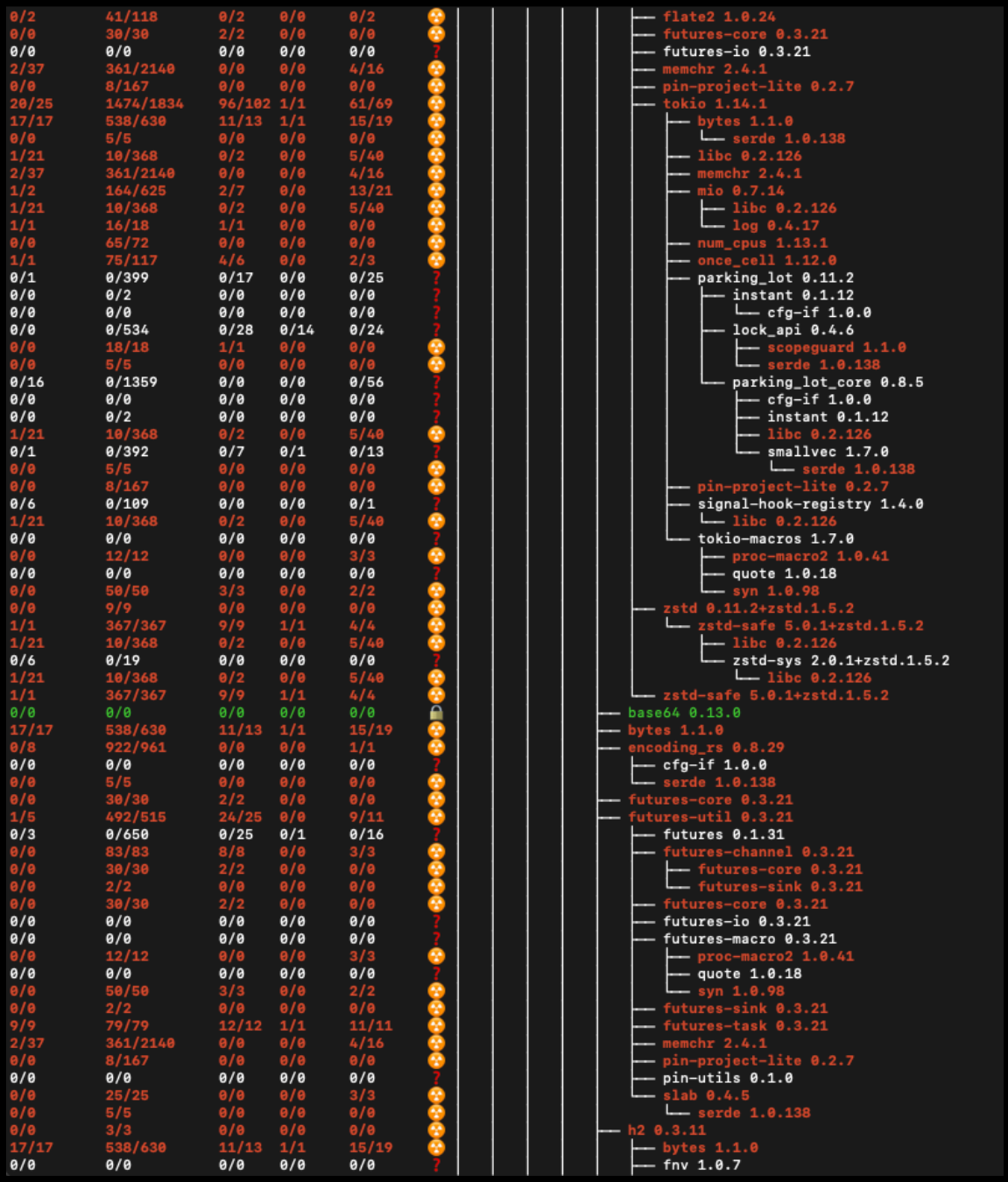
AUTOMATED TESTING



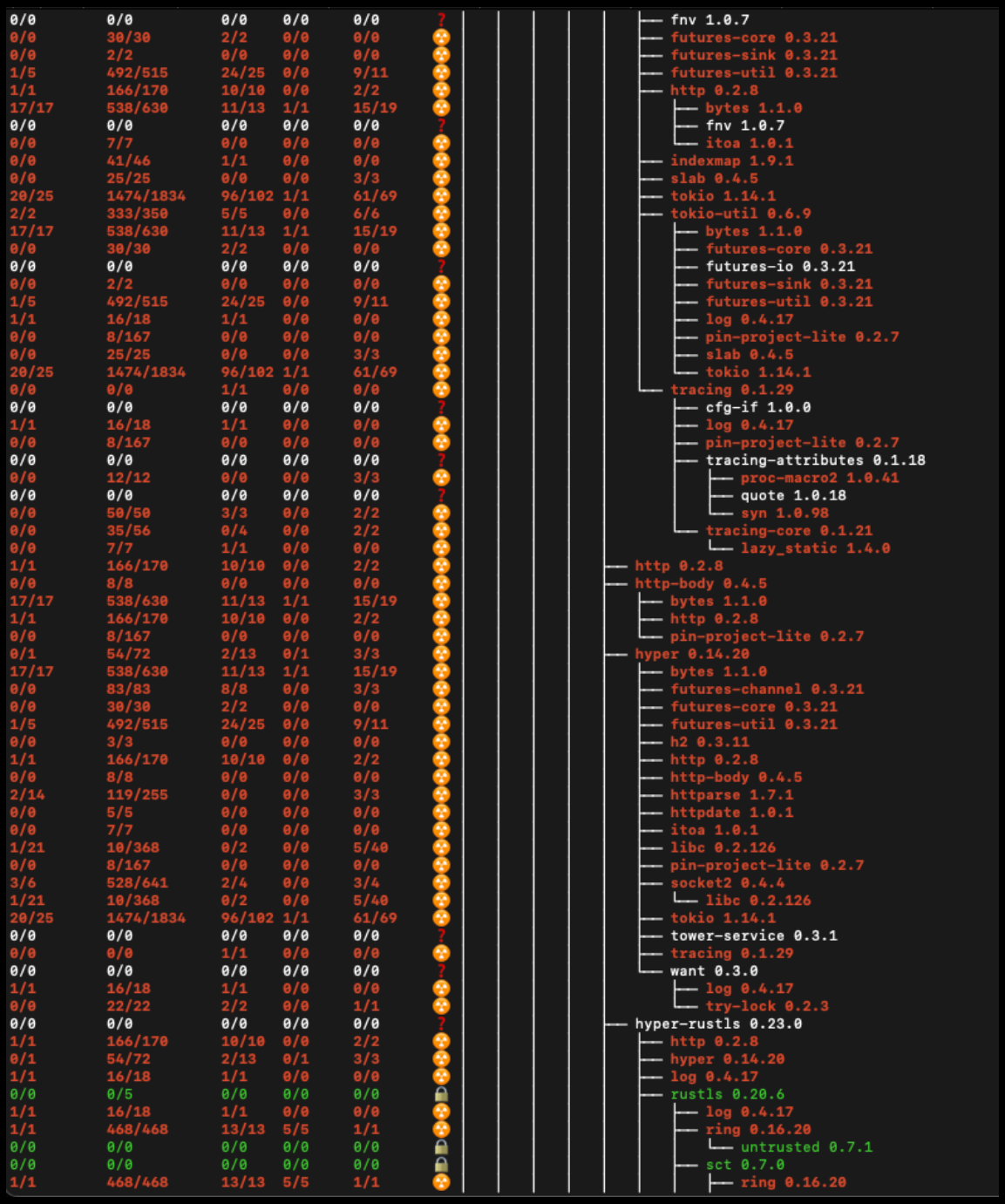
AUTOMATED TESTING



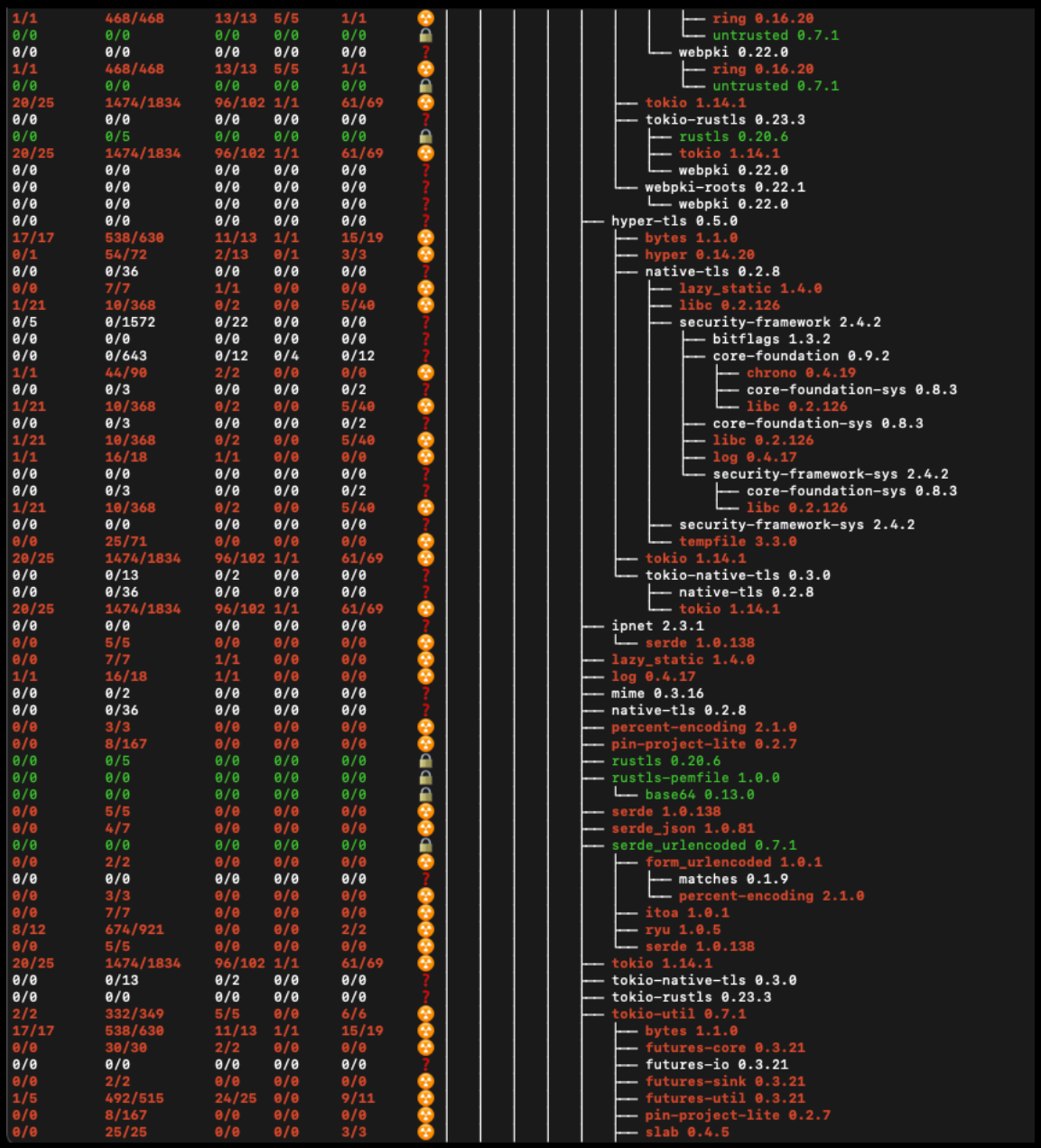
AUTOMATED TESTING



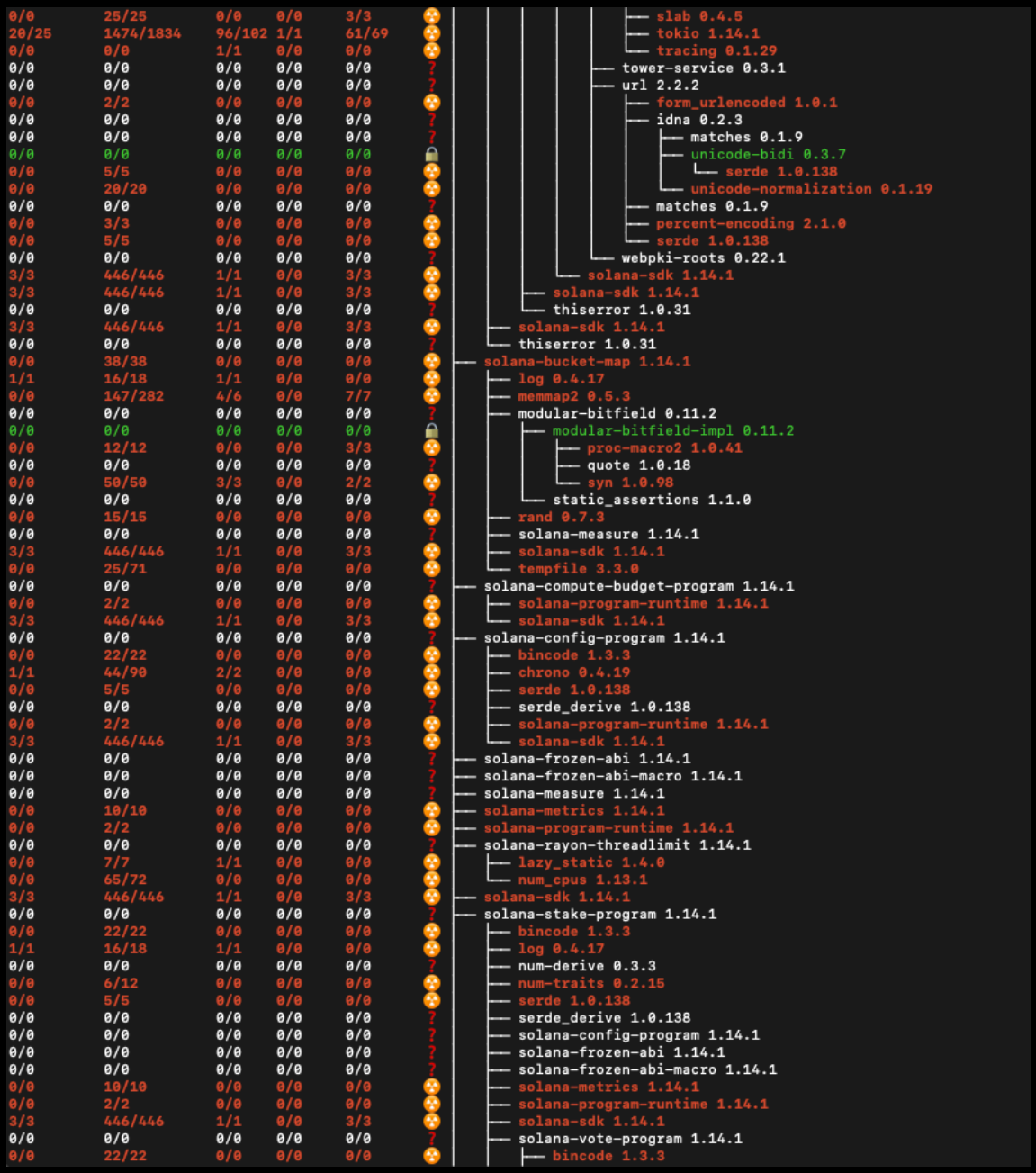
AUTOMATED TESTING



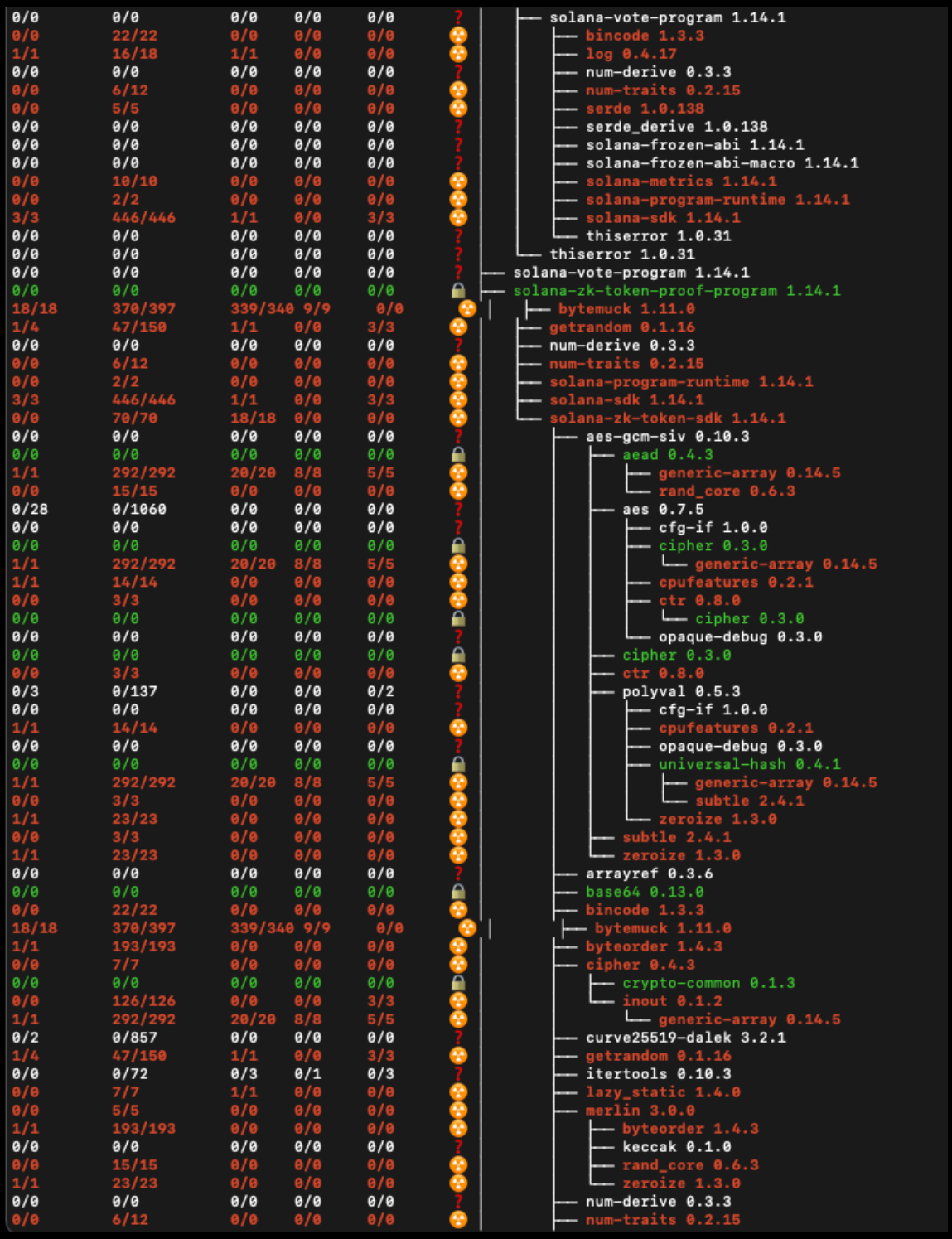
AUTOMATED TESTING



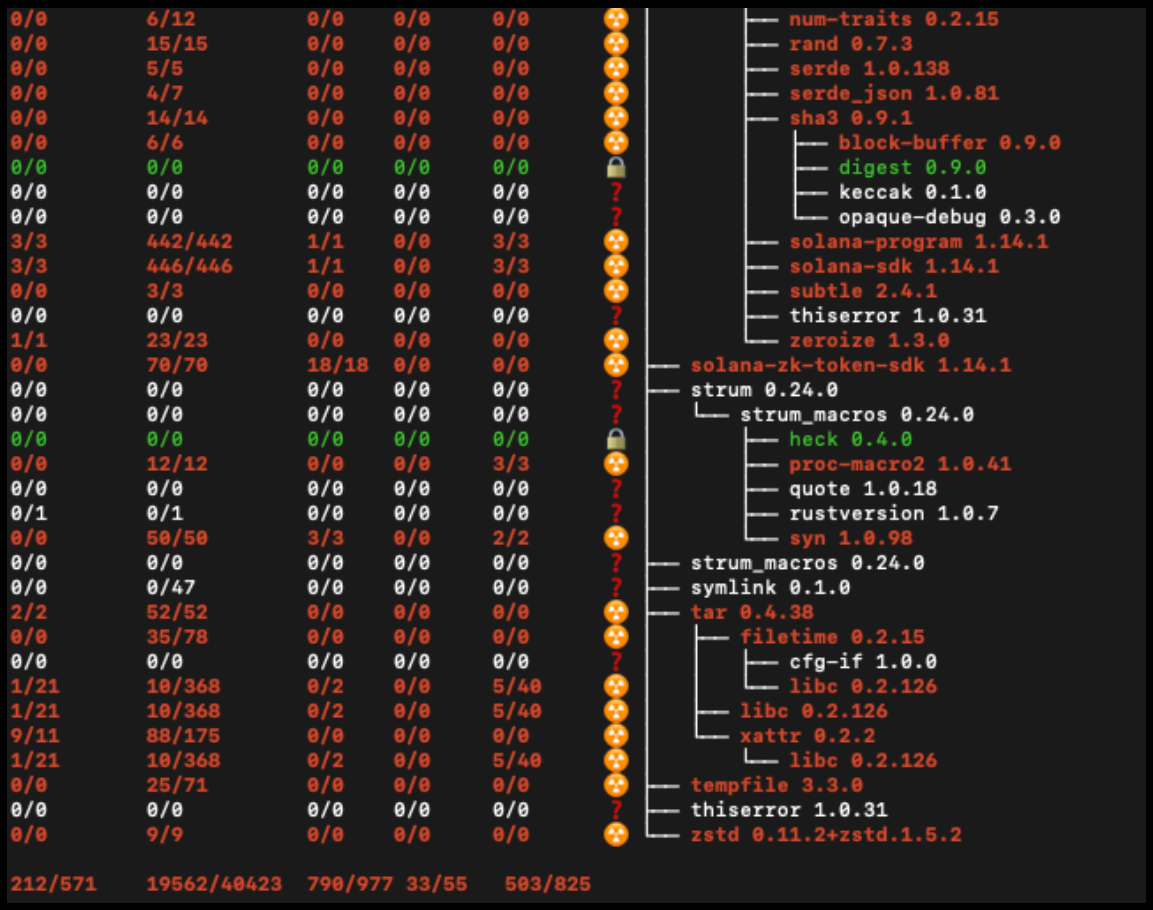
AUTOMATED TESTING



AUTOMATED TESTING



AUTOMATED TESTING



THANK YOU FOR CHOOSING

// HALBORN