HALBORN

Solana Foundation - Address Lookup Table and Versioned Transactions

Solana Program Security Audit

Prepared by: **Halborn**

Date of Engagement: **August 4th, 2022 - August 26th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	08/19/2022	Guillermo Álvarez
0.2	Final Draft	08/26/2022	Guillermo Álvarez
0.3	Draft Review	08/26/2022	Piotr Cielas
0.4	Draft Review	08/26/2022	Gabi Urrutia
1.0	Remediation Plan	08/30/2022	Guillermo Álvarez
1.1	Remediation Plan Review	08/30/2022	Piotr Cielas
1.2	Remediation Plan Review	08/30/2022	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
Guillermo Álvarez	Halborn	Guillermo.Alvarez@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Solana's messages are limited to an MTU size of 1280 bytes; therefore, the current cap is about 35 accounts per transaction after accounting for signatures and other metadata. In order to increase the number of accounts that can be used in a single transaction, Solana introduced a new program which manages on-chain address lookup tables and a new transaction format to leverage on-chain lookup tables and efficiently load more accounts in a single transaction.

Halborn conducted a security audit on the Address Lookup Table program and the new Versioned Transaction feature, beginning on August 4th, 2022 and ending on August 26th, 2022 . The security assessment was scoped to the programs 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 3 weeks for the engagement and assigned 2 full-time security engineers to audit the security of the program and code in scope. The security engineers are blockchain and Solana program security experts with advanced penetration testing and Solana program hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Identify potential security issues within the programs

In summary, Halborn did not identify any vulnerability affecting the Address Lookup Tables program or the Versioned Transactions, only three informational recommendations were presented, which were acknowledged by the Solana Foundation.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual review 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
- Manual program code review and walkthrough to identify 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 cluster deploying (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.

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
- Pull Requests and Commit IDs:
  - Version transaction message and add new message format
    - 1daad38b47d9e41827add682f6ba02ef9d680d38
    - 51921fa9d72c86265b0494f49e905ba10a3350b1
    - 51921fa9d72c86265b0494f49e905ba10a3350b1
    - a5ef863c3fac7db4db8bb731382266424047d466
    - 955456e75f81d48f9ebf2adf77056520dcf45610
    - 2e3f4264a6a28aa80ca2e0ecb40f9a46207a3375
    - 5fefb81d5162d04c22d747fbeb5276f431aed192
    - fa2df7ebf99c9d6bbc19894223620f816ce69454
    - 5ad1a3dc4d9649efb1382fa607d858898f4b373b
  - Store versioned transactions in the ledger, disabled by default
    - d444634c0c16448aa20d0cd24073b47950cd48e5
    - 9ff7c32a3dd6e93e5bb89dcb4aadcbf7b5f67060
    - cd022267b137d42242de5ca785e0372dd6ea7603
    - 13407b1bac69c99d29cc8af5807965b37d8f989d
    - 3bd97f64b517d8e469b0b68c00a18c53aac385dd
    - fc2cc2acb7638d6664e656f21b5dbe46b6c8626c
    - 929aa3feb80cb9b9bfacd11255a844dd4422360d
    - 96d6cec04802b5059db04e1dd5c67005eb0dc7ad
    - 62d0da90e0474b79c8e43ec7d11713c93c983da0
    - cc3592567f5f7ed0374c70892f7d1ec850665af0
    - b16fb60bc440c7dc141337368b0f35dfa5fb2f47
    - 539fafea45de0bea3be33db5e9c00758ba82da62
  - Add runtime support for address table lookups
    - dce2c73f283fc202b67342bca1349bcdf2e11b0e
    - 8e68edb9ff02167031f92b82d31ff349530dbf89
    - 678c08c0f7eeaaf51f5f0cfce97b03a2a441c93f

    - 0c488054ff1902601858a27744f9391fe52c816b
  - Limit number of accounts that a transaction can lock
    - 5cecb4c0207f8e2dc981dee4afa6f79f3e1c0367
  - Add address lookup table program
    - 34d4ebb23b61b8847edfeecb9759740940d2187c
    - 7045e8069931dbcd84ad277beef5250d2a6403ca
  - Add deactivation cooldown before address lookup tables can be closed
    - 55d6fd310f9a36ba51a794f225d68ad3c0ebfaf0
  - Prevent lookup tables from being closed during deactivation slot
    - 75a294c2df6cb05f8f169e7b87b9beb8ee9f8320
  - Make payer and system program optional when extending lookup tables
    - eef847a25590e51b74588998cc71d697f6870ffe
  - Fail tx sanitization when ix program id uses lookup table
    - 209c564c2babcd115e232278ce872462adf8ae01
    - 4ea6166fa6fe6c39d00ccf731450ae9436774f22

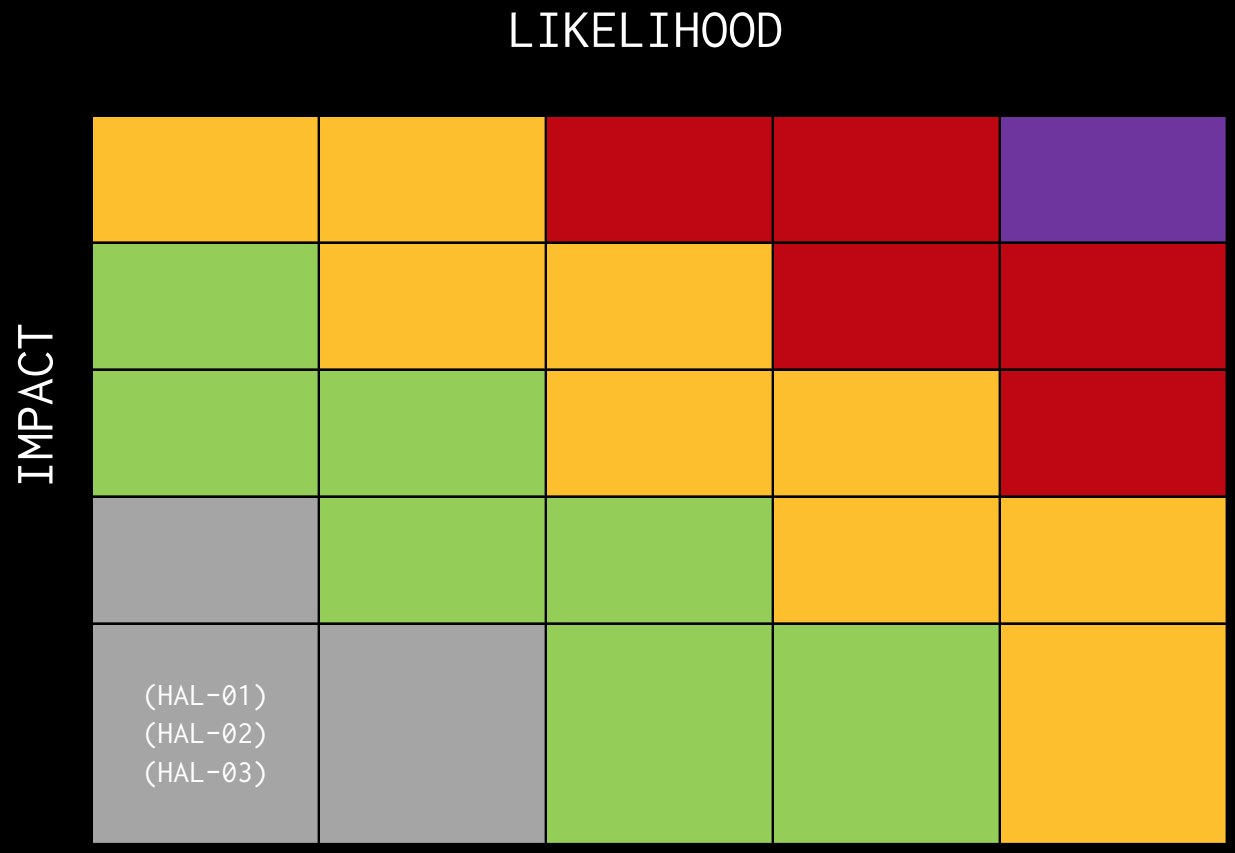
- Programs in scope:

  1. Address Lookup Table (programs/address-lookup-table/)

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

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	0	3



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) LACK OF CONSISTENCY	Informational	ACKNOWLEDGED
(HAL-02) MISSING CARGO OVERFLOW CHECKS	Informational	ACKNOWLEDGED
(HAL-03) POSSIBLE RUST PANICS DUE TO UNSAFE UNWRAP USAGE	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) LACK OF CONSISTENCY - INFORMATIONAL

Description:

Both CreateLookupTable and DeactivateLookupTable instructions follow the same approach. First, all validations, such as checking if payer and authority accounts are signers or if the Address Lookup Table account is in the correct state, are performed. Then the amount of required lamports for the Address Lookup Table account is calculated and if the account does not hold enough rent, necessary lamports are transferred from the payer's account. Finally, all changes are saved into the provided Address Lookup Table.

However, the ExtendLookupTable instruction does not implement the same flow. Once the initial accounts' validation, signature, authority and state requirements are met, the Address Lookup Table is directly extended, without first checking if additional lamports to cover the rent-exempt balance are required. Only then, if the Address Lookup Table does not hold enough lamports, the payer's account funds the table account. Consequently, if the payer account is missing or is not signer, then the transaction fails and changes made to the Address Lookup Table are reverted by the runtime.

Code Location:

Listing 1: programs/address-lookup-table/src/processor.rs (Lines 297,318)

```
288 {
289     let mut lookup_table_account_ref_mut = lookup_table_account.
 ↳ try_account_ref_mut()?;
290     AddressLookupTable::overwrite_meta_data(
291         lookup_table_account_ref_mut.data_as_mut_slice(),
292         lookup_table_meta,
293     )?;
294
```

```
295     let table_data = lookup_table_account_ref_mut.data_mut();
296     for new_address in new_addresses {
297         table_data.extend_from_slice(new_address.as_ref());
298     }
299 }
300
301 let rent = invoke_context.get_sysvar_cache().get_rent()?;
302 let required_lamports = rent
303     .minimum_balance(new_table_data_len)
304     .max(1)
305     .saturating_sub(lookup_table_account.lamports()?);
306
307 let table_key = *lookup_table_account.unsigned_key();
308 if required_lamports > 0 {
309     let payer_account =
310         keyed_account_at_index(keyed_accounts, checked_add(
 ↳ first_instruction_account, 2)?)?;
311     let payer_key = if let Some(payer_key) = payer_account.
 ↳ signer_key() {
312         *payer_key
313     } else {
314         ic_msg!(invoke_context, "Payer account must be a signer");
315         return Err(InstructionError::MissingRequiredSignature);
316     };
317
318     invoke_context.native_invoke(
319         system_instruction::transfer(&payer_key, &table_key,
 ↳ required_lamports),
320         &[payer_key],
321     )?;
322 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Instead of relying on the runtime to rollback account changes when the transaction fails, it is recommended to implement the same flow for all

instructions that might require lamports transfers. First all validations should be performed, then required lamports should be transferred from the payer's account and only after that the Address Lookup Table account should be modified.

Remediation Plan:

ACKNOWLEDGED: The Solana Foundation team reviewed and acknowledged this finding.

3.2 (HAL-02) MISSING CARGO OVERFLOW CHECKS - INFORMATIONAL

Description:

It was observed that there is no overflow-checks=true in Cargo.toml. By default, overflow checks are disabled in optimized release builds. Hence, if there is an overflow on release builds, it will be silenced, leading to unexpected behavior of an application. Even if checked arithmetic is used through checked_* or saturating_*, it is recommended to have that additional validation in Cargo.toml.

Code Location:

- programs/address-lookup-table/Cargo.toml

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to add overflow-checks=true under your release profile in Cargo.toml.

Remediation Plan:

ACKNOWLEDGED: The Solana Foundation team reviewed and acknowledged this finding.

3.3 (HAL-03) 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 intruduced by the changes in scope are listed, justified usages such as in tests were excluded.

Listing 2

```
   ./sdk/program/src/message/legacy.rs:69    keys.iter().position(|k|
↳  k == key).unwrap() as u8
   ./sdk/program/src/message/versions/v0/mod.rs:290       bincode::
↳ serialize(&(MESSAGE_VERSION_PREFIX, self)).unwrap()
   ./sdk/program/src/message/versions.rs:80    bincode::serialize(
↳ self).unwrap()
   ./sdk/program/src/message/versions.rs:96     Hash(<[u8; crate::
↳ hash::HASH_BYTES]>::try_from(hasher.finalize().as_slice()).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 the contract 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 Foundation team reviewed and acknowledged this finding.

AUTOMATED TESTING

4.1 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.

Results:

No vulnerabilities were identified.

4.2 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	Short Description
RUSTSEC-2020-0071	time 0.1.44	Potential segfault in the time crate
RUSTSEC-2021-0139	ansi_term 0.12.1	ansi_term is unmaintained
RUSTSEC-2021-0139	net2 0.2.37	net2 crate has been deprecated, use socket2 instead

4.3 UNSAFE RUST CODE DETECTION

Description:

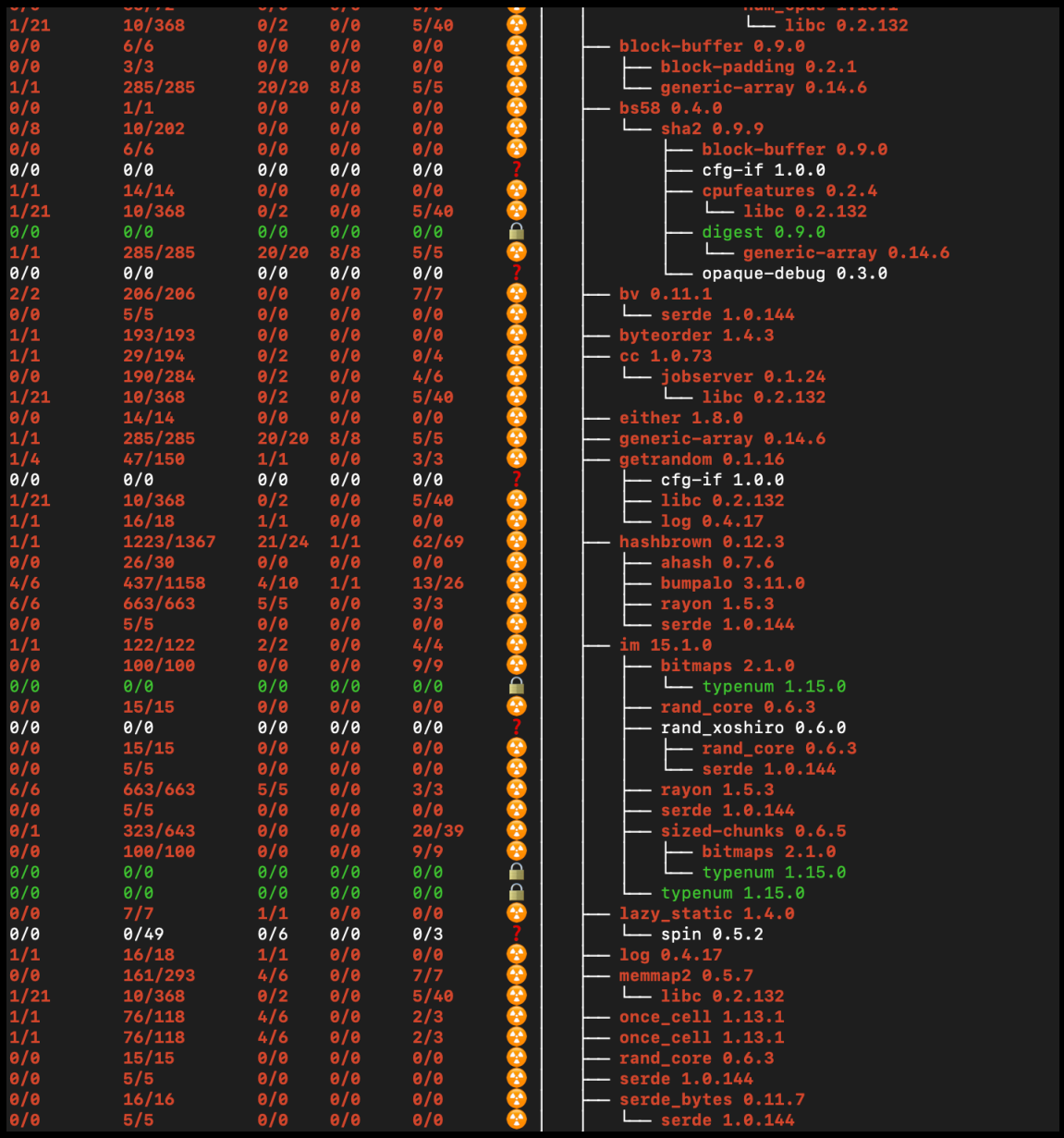
Halborn used automated security scanners to assist with the detection of well-known security issues and vulnerabilities. Among the tools used was cargo-geiger, a security tool that lists statistics related to the usage of unsafe Rust code in a core Rust codebase and all its dependencies.

Results:

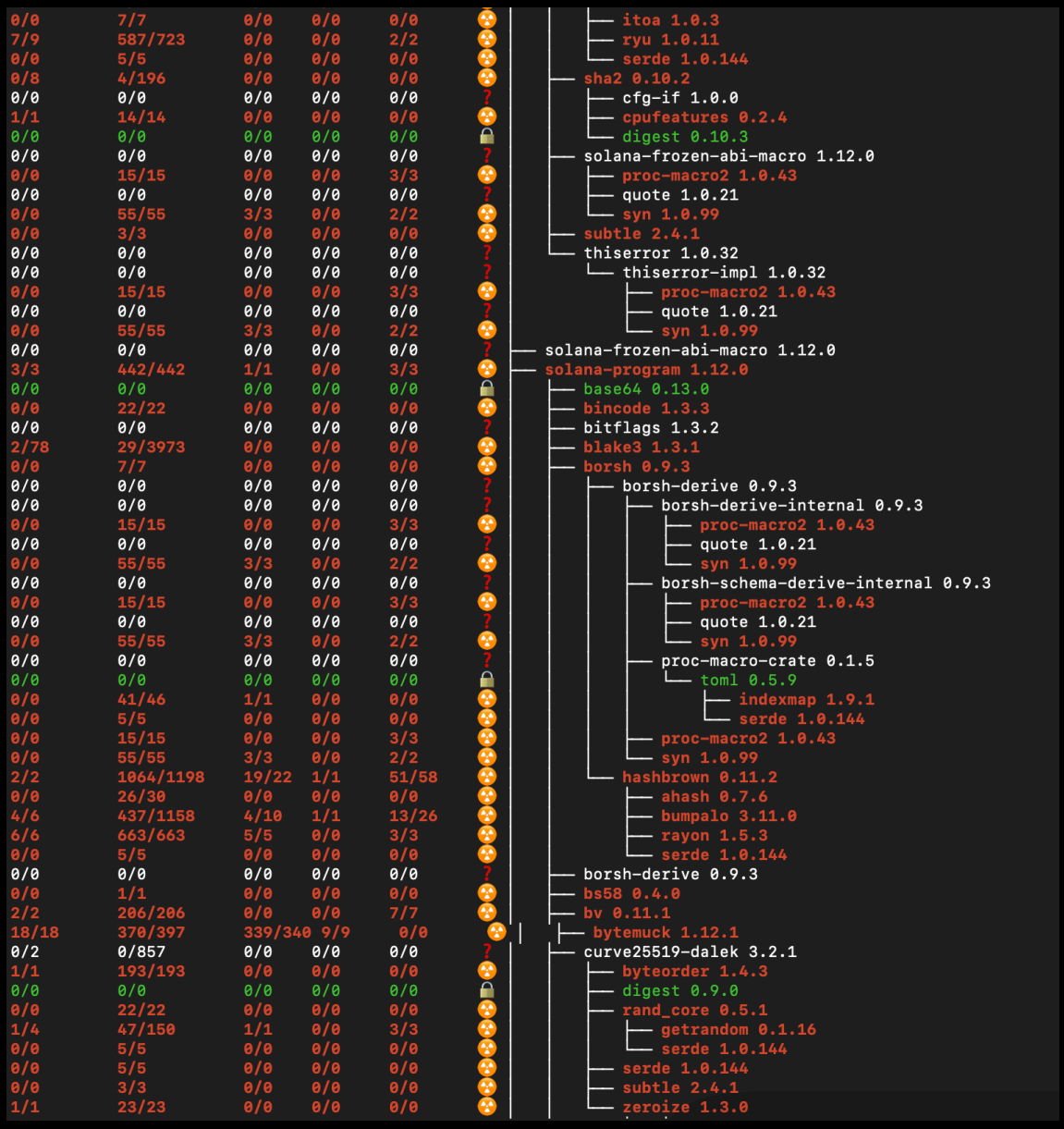
```
Symbols:
    🔒  = No `unsafe` usage found, declares #![forbid(unsafe_code)]
    ❓  = No `unsafe` usage found, missing #![forbid(unsafe_code)]
    ☢️  = `unsafe` usage found

Functions  Expressions  Impls  Traits  Methods  Dependency

0/0        0/0          0/0    0/0     0/0      ❓ solana-address-lookup-table-program 1.12.0
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 1.0.144
0/0        0/0          0/0    0/0     0/0      ❓ │       └── serde_derive 1.0.144
0/0        15/15        0/0    0/0     3/3      ☢️ │           ├── proc-macro2 1.0.43
0/0        4/4          0/0    0/0     0/0      ☢️ │           │   └── unicode-ident 1.0.3
0/0        0/0          0/0    0/0     0/0      ❓ │           ├── quote 1.0.21
0/0        15/15        0/0    0/0     3/3      ☢️ │           │   └── proc-macro2 1.0.43
0/0        55/55        3/3    0/0     2/2      ☢️ │           └── syn 1.0.99
0/0        15/15        0/0    0/0     3/3      ☢️ │               ├── proc-macro2 1.0.43
0/0        0/0          0/0    0/0     0/0      ❓ │               ├── quote 1.0.21
0/0        4/4          0/0    0/0     0/0      ☢️ │               └── unicode-ident 1.0.3
18/18      370/397      339/340 9/9     0/0      ☢️ ├── bytemuck 1.12.1
0/0        0/0          0/0    0/0     0/0      ❓ │   └── bytemuck_derive 1.2.1
0/0        15/15        0/0    0/0     3/3      ☢️ │       ├── proc-macro2 1.0.43
0/0        0/0          0/0    0/0     0/0      ❓ │       ├── quote 1.0.21
0/0        55/55        3/3    0/0     2/2      ☢️ │       └── syn 1.0.99
1/1        16/18        1/1    0/0     0/0      ☢️ ├── log 0.4.17
0/0        0/0          0/0    0/0     0/0      ❓ │   ├── cfg-if 1.0.0
0/0        5/5          0/0    0/0     0/0      ☢️ │   └── serde 1.0.144
0/0        0/0          0/0    0/0     0/0      ❓ ├── num-derive 0.3.3
0/0        15/15        0/0    0/0     3/3      ☢️ │   ├── proc-macro2 1.0.43
0/0        0/0          0/0    0/0     0/0      ❓ │   ├── quote 1.0.21
0/0        55/55        3/3    0/0     2/2      ☢️ │   └── syn 1.0.99
0/0        6/12         0/0    0/0     0/0      ☢️ ├── num-traits 0.2.15
0/0        5/5          0/0    0/0     0/0      ☢️ ├── serde 1.0.144
0/0        0/0          0/0    0/0     0/0      ❓ ├── solana-frozen-abi 1.12.0
0/0        26/30        0/0    0/0     0/0      ☢️ │   ├── ahash 0.7.6
1/4        49/166       1/1    0/0     3/3      ☢️ │   │   ├── getrandom 0.2.7
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 0.2.132
1/1        76/118       4/6    0/0     2/3      ☢️ │   │   ├── once_cell 1.13.1
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 0.2.132
0/1        0/399        0/7    0/1     0/13     ❓ │   │       └── smallvec 1.9.0
0/0        5/5          0/0    0/0     0/0      ☢️ │   │           └── serde 1.0.144
0/0        5/5          0/0    0/0     0/0      ☢️ │   └── serde 1.0.144
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 1.0.144
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
```

AUTOMATED TESTING

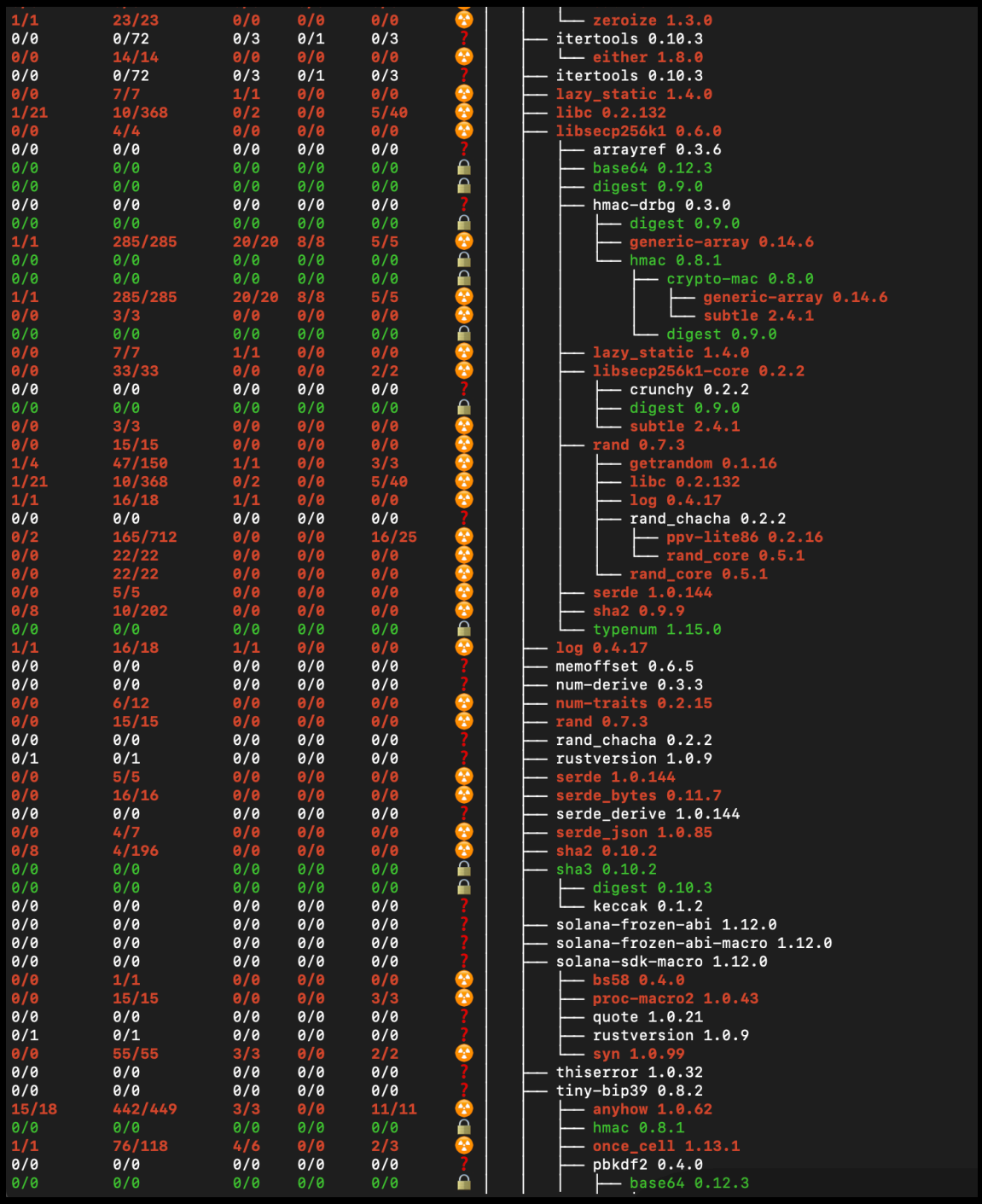
AUTOMATED TESTING



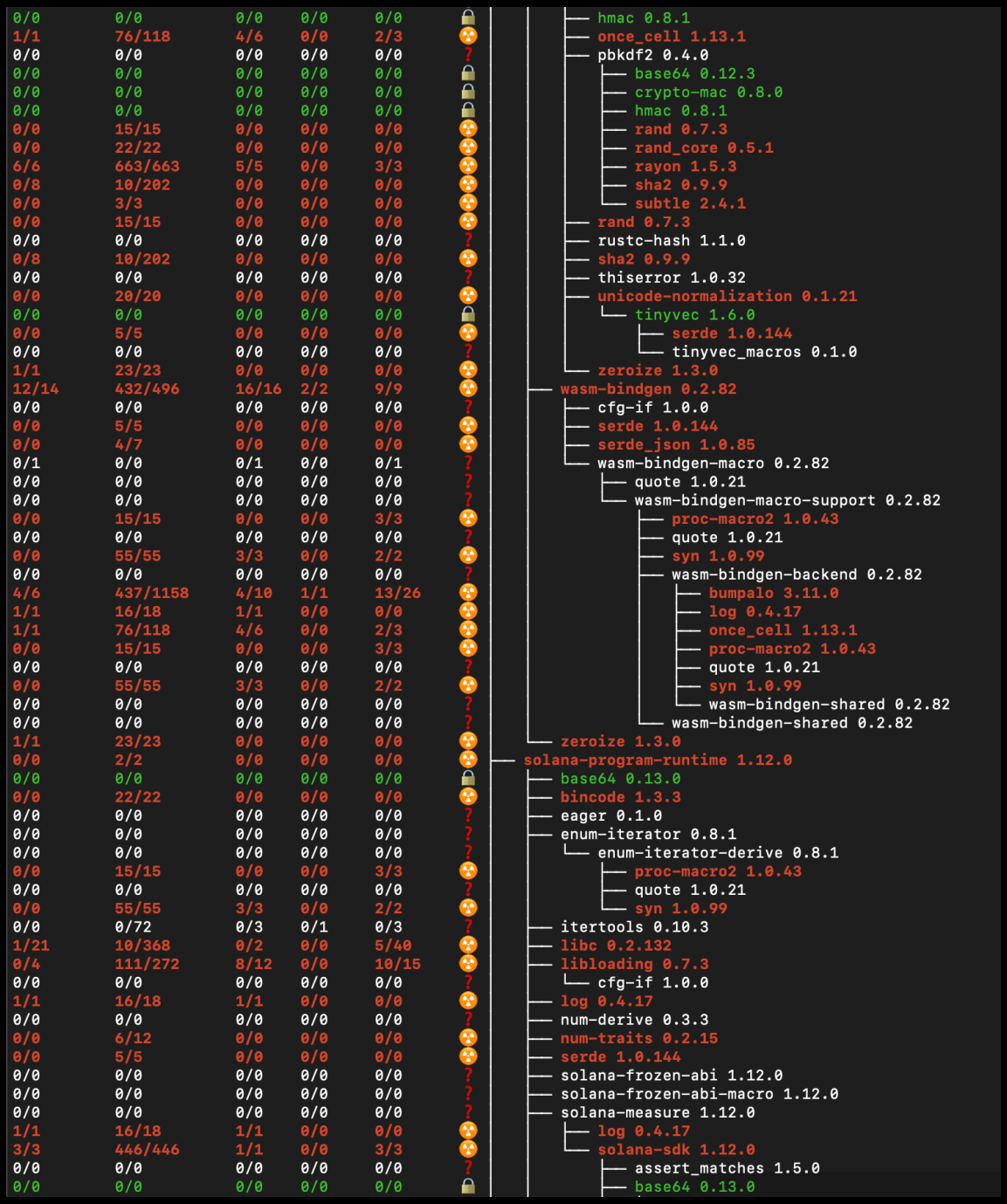
AUTOMATED TESTING



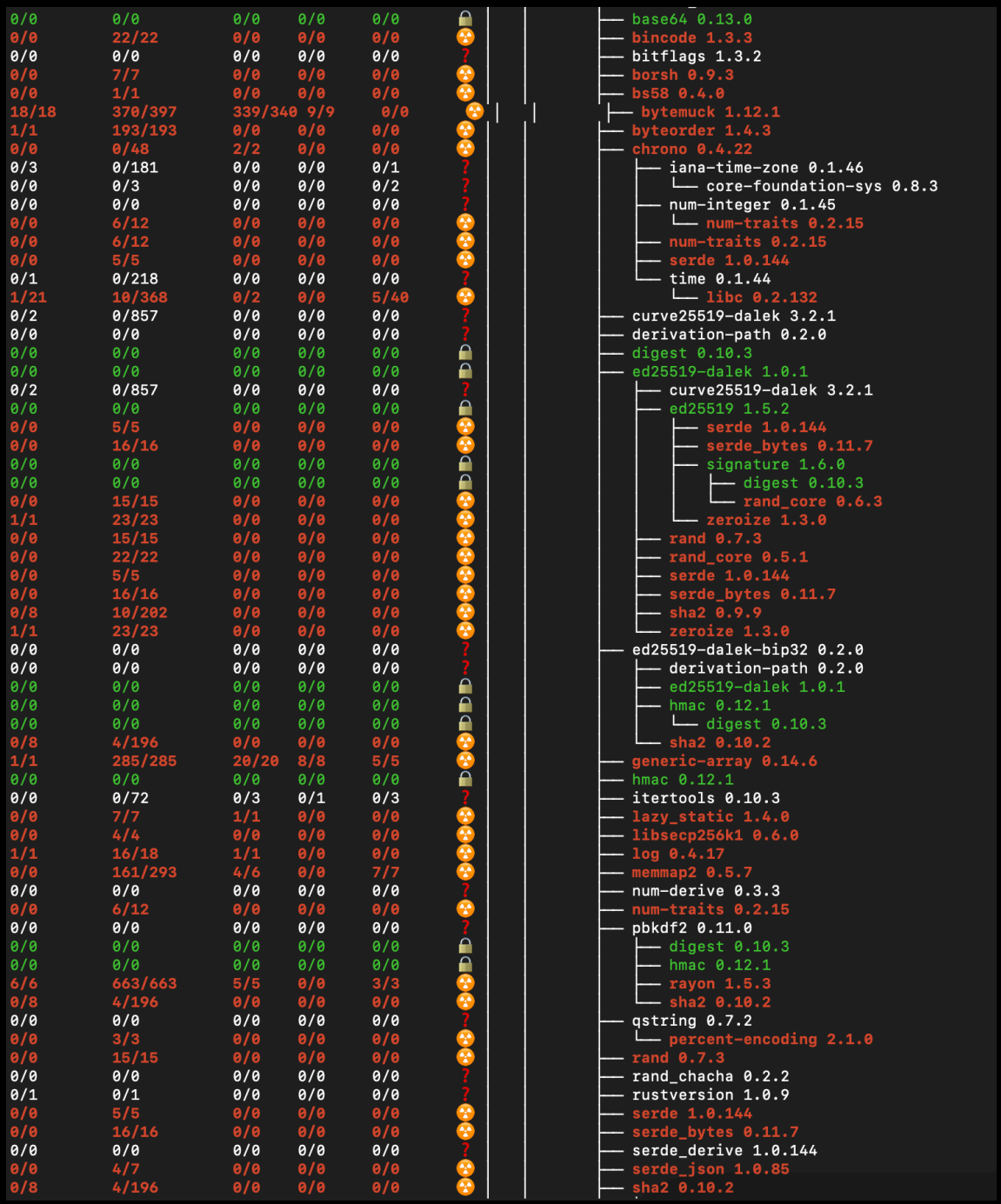
AUTOMATED TESTING



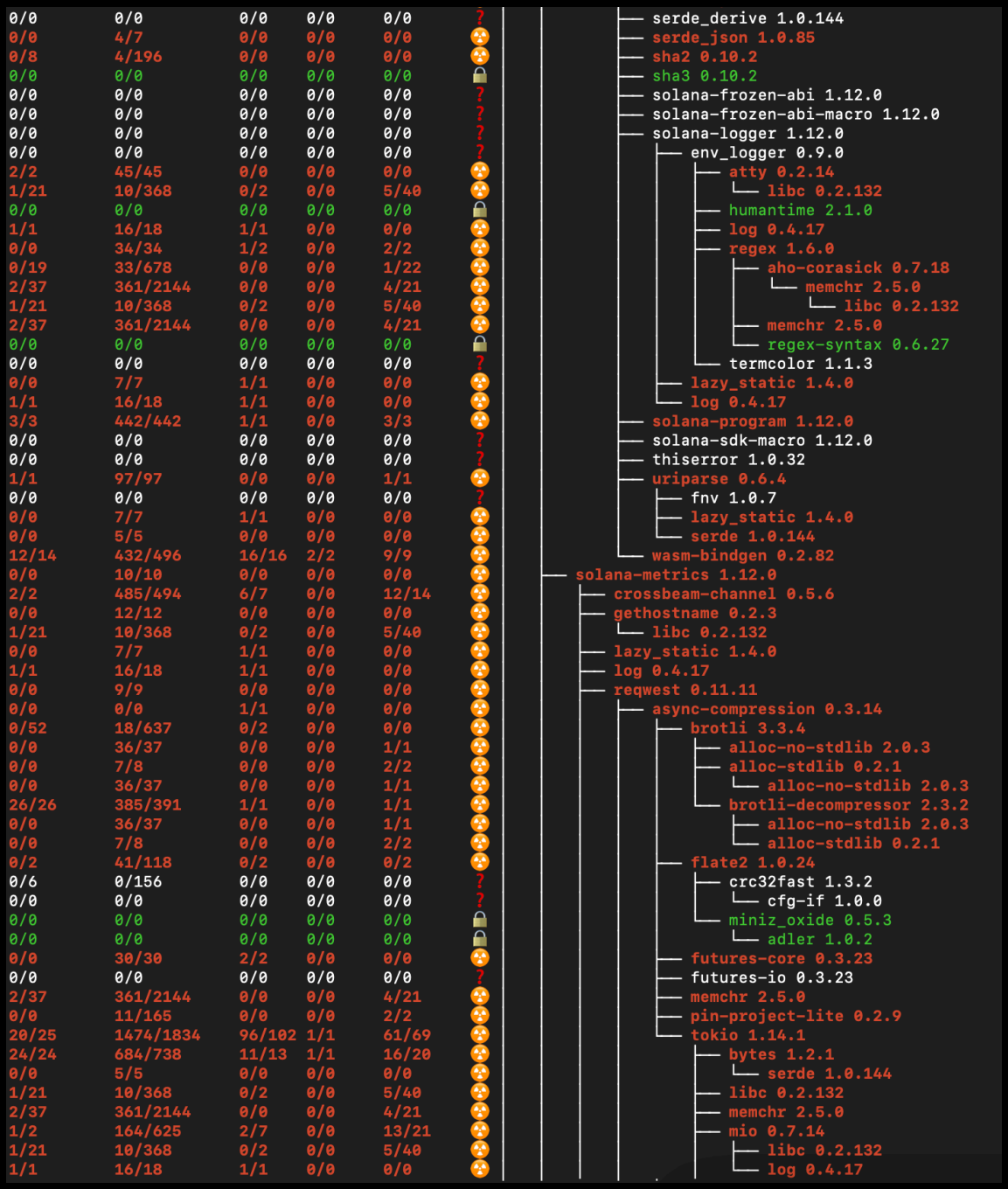
AUTOMATED TESTING



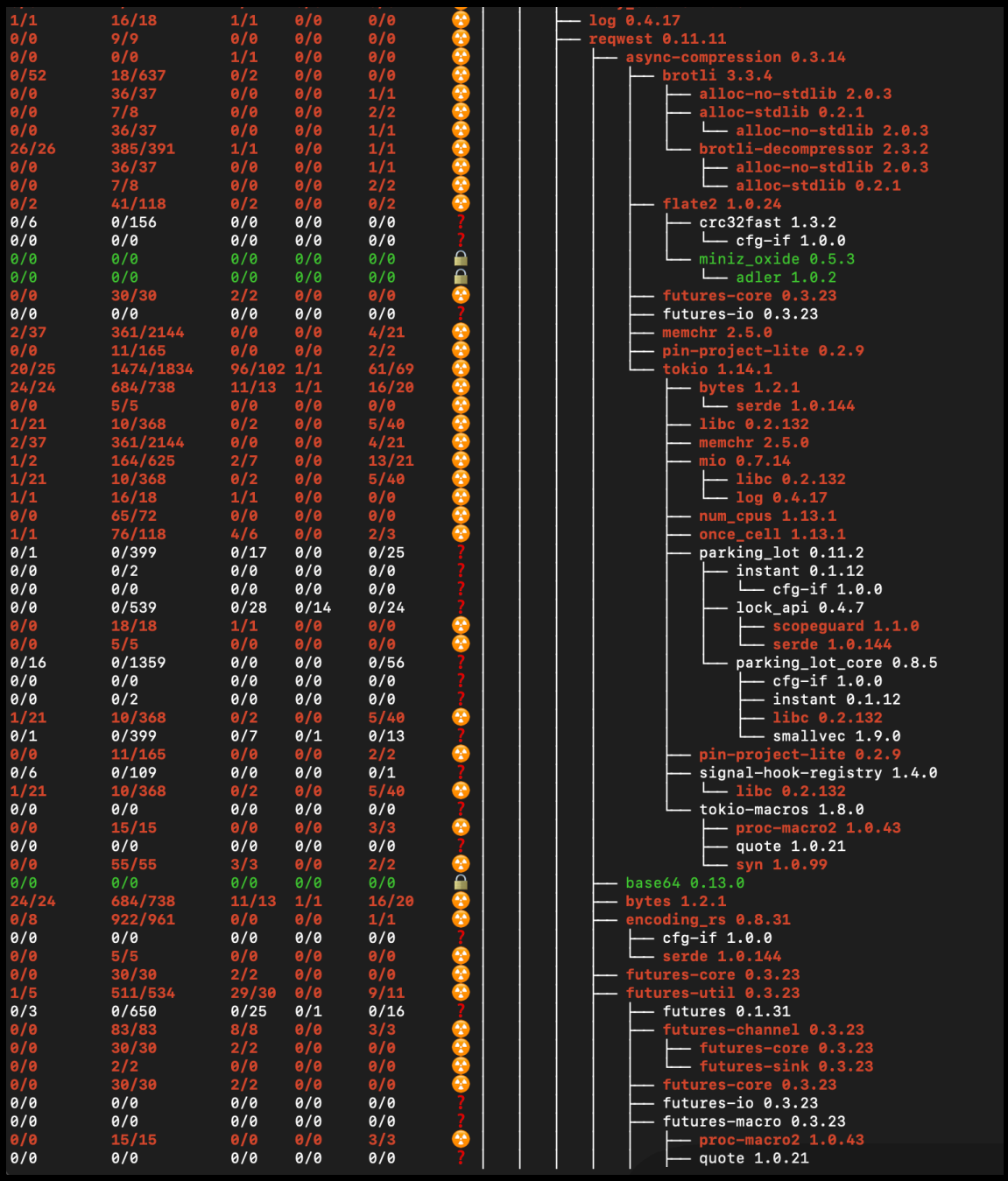
AUTOMATED TESTING



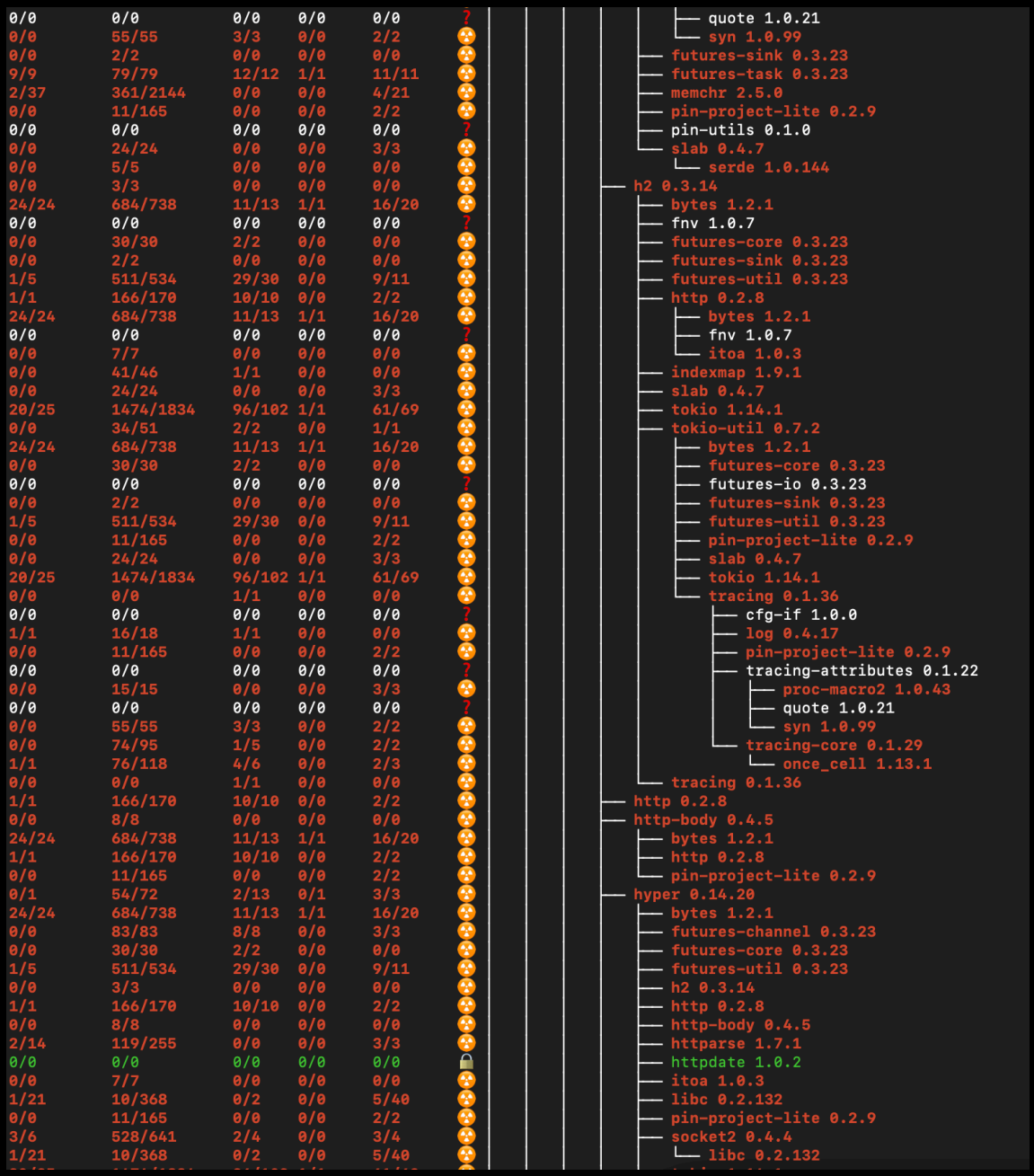
AUTOMATED TESTING



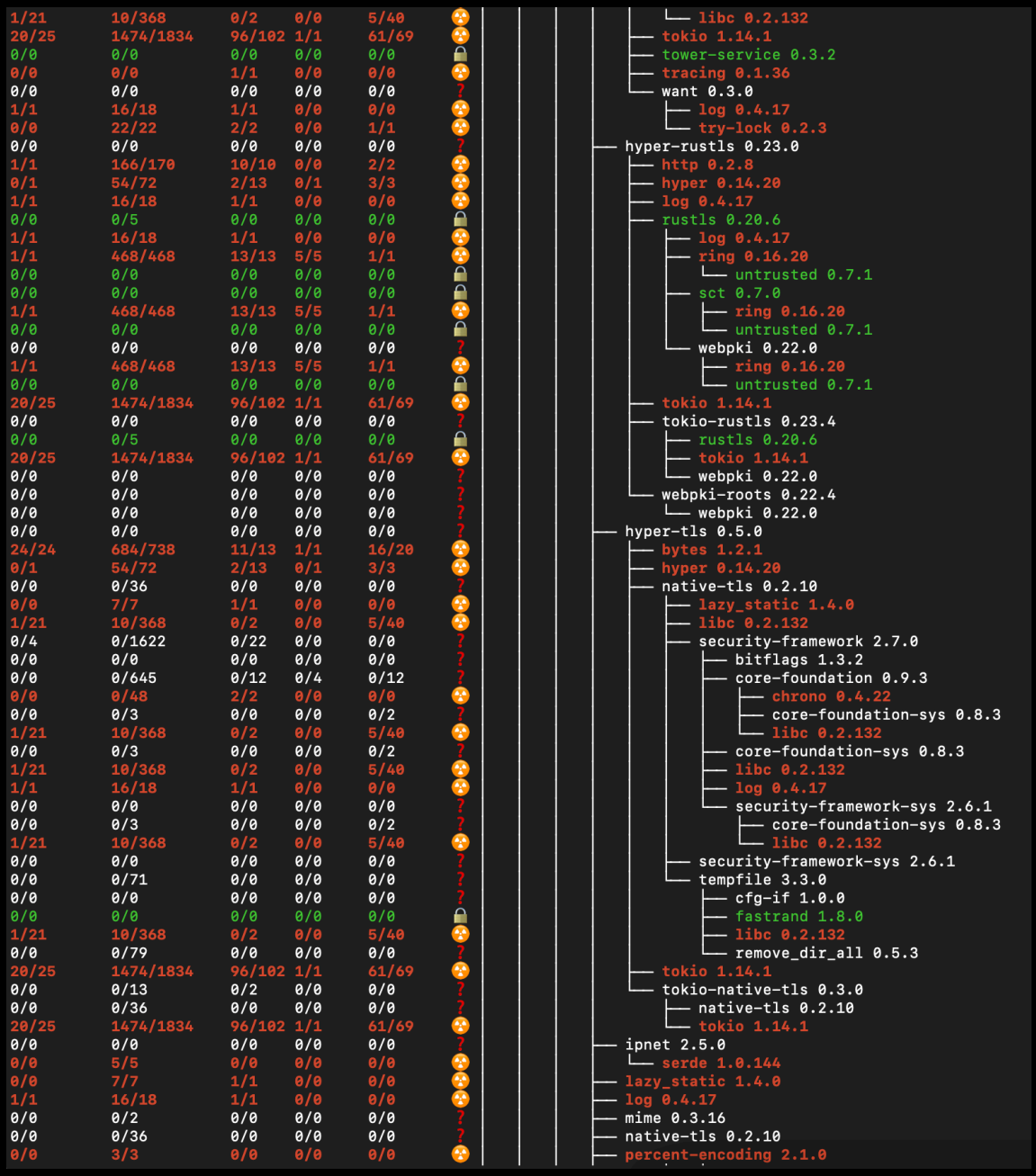
AUTOMATED TESTING



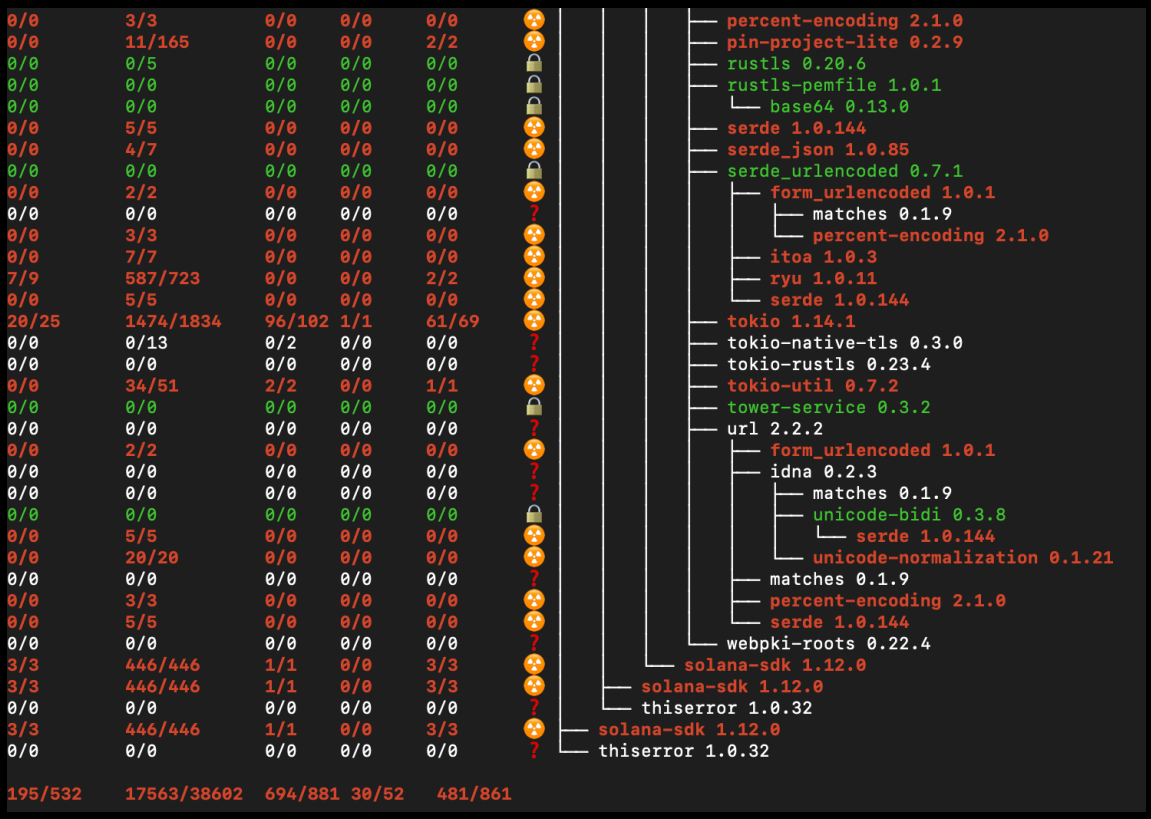
AUTOMATED TESTING



AUTOMATED TESTING



AUTOMATED TESTING



THANK YOU FOR CHOOSING

// HALBORN