HALBORN

Solana Labs - Permissionless TWAMM

Solana Program Security Audit

Prepared by: **Halborn**

Date of Engagement: **April 4th, 2023 - April 27th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	04/25/2023	Michael Smith
0.2	Document Updates	04/26/2023	Michael Smith
0.3	Final Draft	04/26/2023	Michael Smith
0.4	Draft Review	04/26/2023	Isabel Burruezo
0.5	Draft Review	04/26/2023	Piotr Cielas
0.6	Draft Review	04/28/2023	Gabi Urrutia
1.0	Remediation Plan	05/02/2023	Michael Smith
1.1	Remediation Plan Review	05/03/2023	Isabel Burruezo
1.2	Remediation Plan Review	05/04/2023	Piotr Cielas
1.3	Remediation Plan Review	05/05/2023	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
Michael Smith	Halborn	Michael.Smith@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

The Permisonless TWAMM (time-weighted average price market maker) program assists traders to efficiently execute large orders by pooling together a large order, breaking them down into small pieces and executing them over a specified time interval.

Solana Labs engaged Halborn to conduct a security audit on their Solana programs, beginning on April 4th, 2023 and ending on April 27th, 2023 . The security assessment was scoped to the programs provided in the twamm 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 1 full-time security engineer to audit the security of the programs in scope. The security engineer is a blockchain and Solana program security expert 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 identified some improvements to reduce the likelihood and impact of multiple risks, which have been mostly addressed by Solana Labs . The main ones are the following:

(HAL-01) EXPIRED ORDERS CAN BE SETTLED

Solana Labs acknowledged this finding and will resolve it in a **future release**. They plan to add a check to prevent pools that have expired past a certain threshold from being settled.

(HAL-06) POSSIBLE RUST PANICS DUE TO UNSAFE UNWRAP USAGE

Solana Labs **successfully** remediated this issue by removing all the unwraps throughout the program

(HAL-07) INTEGER OVERFLOW

Solana Labs **successfully** remediated this issue by replacing arithmetic operation with checked math

Solana Labs acknowledged and accepted the risk of the rest of the findings since their impact were low or informational.

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 lead to arithmetic vulnerabilities.
- Finding unsafe Rust code usage (cargo-geiger)
- Scanning dependencies for known vulnerabilities (cargo audit).

- Local runtime testing (solana-test-framework)
- Scanning for common Solana vulnerabilities (soteria)

2. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

2.1 Exploitability

Attack Origin (AO):

Captures whether the attack requires compromising a specific account.

Attack Cost (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

Attack Complexity (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

Metrics:

Exploitability Metric (m_E)	Metric Value	Numerical Value
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

2.2 Impact

Confidentiality (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

Integrity (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

Availability (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

Deposit (D):

Measures the impact to the deposits made to the contract by either users or owners.

Yield (Y):

Measures the impact to the yield generated by the contract for either users or owners.

Metrics:

Impact Metric (m_I)	Metric Value	Numerical Value
Confidentiality (C)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium: (Y:M)	0.5
	High: (Y:H)	0.75
	Critical (Y:H)	1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

2.3 Severity Coefficient

Reversibility (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

Scope (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

Coefficient (C)	Coefficient Value	Numerical Value
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

Severity	Score Value Range
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

2.4 SCOPE

Code repositories:

1. TWAMM

- Repository: twamm
- Commit ID: 1350c9c15f581acd83f4e97035b69d9336b8f9f7
- Programs in scope:
    1. twamm (programs/twamm)

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

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	2	5

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
EXPIRED ORDERS CAN BE SETTLED	Low (3.7)	FUTURE RELEASE
LACK OF EMERGENCY-STOP PATTERN	Low (2.5)	RISK ACCEPTED
MINIMUM MULTISIG THRESHOLD CHECK MISSING	Informational (0.9)	ACKNOWLEDGED
ORACLE OWNER VALIDATION	Informational (0.1)	ACKNOWLEDGED
ZERO AMOUNT PROCESSING	Informational (0.0)	ACKNOWLEDGED
POSSIBLE RUST PANICS DUE TO UNSAFE UNWRAP USAGE	Informational (0.0)	SOLVED - 04/2/2023
INTEGER OVERFLOW	Informational (0.0)	SOLVED - 04/2/2023

FINDINGS & TECH DETAILS

4.1 (HAL-01) EXPIRED ORDERS CAN BE SETTLED - LOW (3.7)

The twamm program allows traders to execute large orders efficiently by pooling them together and executing them over a specified time interval. Each user order is tied to a pool that has an expiry timestamp based on the wall clock. Before an pool expires, orders can be settled in two different methods.

Internal Settlement - Orders with opposite matching sides (buy and sell) are settled internally using an oracle's price feed, optionally net outstanding liquidity can be settled via Jupiter, so long as the swap price falls within a threshold. This is done with a permission-less crank instruction, which is executed by a crank job or a user, periodically through the life of the order to try to fulfill it close to the oracle's price. Successful cranks are rewarded with a crank reward, incentivizing cranks to be executed.

Arbitrage - Market Makers can settle orders by swapping tokens directly with the twamm programs pools, this is done with a settle instruction.

Both the crank and settle instructions will change a pools' status from active to expired once the current time surpasses the pools' expiration date. Normally, this is not an issue as users are incentivized to crank and market makers will settle profitable swaps, thus expired pools will be changed to expired not long after they expire. However, if the chain was to halt for a long period of time, token prices could change drastically and once un-halted market makers could front run a crank or a user's cancel order instruction & settle the order at the new token price. Newer orders would be more susceptible to this as they have more outstanding liquidity, a whole older order should have less.

Code Location:

Listing 1: programs/twamm/src/instructions/settle.rs

```
83
84
85 #[derive(AnchorSerialize, AnchorDeserialize)]
86 pub struct SettleParams {
87     pub supply_side: MatchingSide,
88     pub min_token_amount_in: u64,
89     pub max_token_amount_in: u64,
90     pub worst_exchange_rate: u64,
91 }
92
93 pub fn settle(ctx: Context<Settle>, params: &SettleParams) ->
   ↳ Result<i64> {
94     // validate inputs
95     require_gt!(
96         params.max_token_amount_in,
97         0u64,
98         TwammError::InvalidTokenAmount
99     );
100
101     let token_pair = ctx.accounts.token_pair.as_mut();
102     require!(
103         token_pair.allow_settlements,
104         TwammError::SettlementsNotAllowed
105     );
106     require_neq!(
107         params.supply_side,
108         MatchingSide::Internal,
109         TwammError::InvalidSettlementSide
110     );
111
112     // collect and validate pools
113     msg!("Load pools");
114     let mut pools = token_pair.load_pools(ctx.remaining_accounts)
   ↳ ?.0;
115     require!(!pools.is_empty(), TwammError::NothingToSettle);
116
117     // compute token balance changes
118     msg!("Compute token balance changes");
119     let oracle_price = token_pair
120         .get_token_pair_oracle_price(&ctx.accounts.oracle_token_a,
    ↳  &ctx.accounts.oracle_token_b)?;
```

```
121     let token_a_change;
122     let token_b_change;
123     let settlement_side;
124     if params.supply_side == MatchingSide::Buy {
125         token_a_change = token_pair.get_token_a_amount(params.
 ↳ max_token_amount_in, oracle_price)?;
126         require_gte!(
127             token_a_change,
128             params.worst_exchange_rate,
129             TwammError::MaxSlippage
130         );
131         token_b_change = params.max_token_amount_in;
132         settlement_side = MatchingSide::Sell;
133     } else {
134         token_a_change = params.max_token_amount_in;
135         token_b_change = token_pair.get_token_b_amount(params.
 ↳ max_token_amount_in, oracle_price)?;
136         require_gte!(
137             token_b_change,
138             params.worst_exchange_rate,
139             TwammError::MaxSlippage
140         );
141         settlement_side = MatchingSide::Buy;
142     }
143 
144     // settle pools
145     msg!("Settle pools");
146     let current_time = token_pair.get_time()?;
147 
148     // settle pools function takes raw pool refs for easier
 ↳ testing
149     let mut pool_refs: Vec<&mut Pool> = Vec::with_capacity(pools.
 ↳ len());
150     for pool in pools.iter_mut() {
151         pool_refs.push(pool);
152     }
153 
154     let res = token_pair.settle_pools(
155         &mut pool_refs,
156         params.supply_side,
157         token_a_change,
158         token_b_change,
159         oracle_price,
160         oracle_price,
```

```
161         current_time,
162     )?;
163 
164     msg!("Validate settled amounts");
165     require!(
166         res.settlement_side == settlement_side,
167         TwammError::InvalidSettlementSide
168     );
169     require!(
170         res.net_amount_settled > 0,
171         TwammError::SettlementAmountTooLarge
172     );
173 
174     // check that spent amount against the limits
175     require_gte!(
176         params.max_token_amount_in,
177         res.source_amount_received,
178         TwammError::SettlementError
179     );
180     require_gte!(
181         res.net_amount_required,
182         res.net_amount_settled,
183         TwammError::SettlementError
184     );
185     require_gte!(
186         res.source_amount_received,
187         params.min_token_amount_in,
188         TwammError::SettlementAmountTooSmall
189     );
190 
191     // check received amount against worst_exchange_rate
192     let min_expected_amount = math::checked_as_u64(math::
 ↳ checked_div(
193         math::checked_mul(
194             res.source_amount_received as u128,
195             params.worst_exchange_rate as u128,
196         )?,
197         params.max_token_amount_in as u128,
198     )?)?;
199     require_gte!(
200         res.net_amount_settled,
201         min_expected_amount,
202         TwammError::MaxSlippage
203     );
```

```
204
205     // transfer tokens to/from the user
206     msg!("Transfer tokens to/from the user");
207     let settle_fee = math::checked_as_u64(math::checked_div(
208         math::checked_mul(
209             res.net_amount_settled as u128,
210             token_pair.settle_fee_numerator as u128,
211         )?,
212         token_pair.settle_fee_denominator as u128,
213     )?)?;
214     let net_amount_settled_after_fees = math::checked_sub(res.
 ↳ net_amount_settled, settle_fee)?;
215     if params.supply_side == MatchingSide::Buy {
216         let context = CpiContext::new(
217             ctx.accounts.token_program.to_account_info(),
218             Transfer {
219                 from: ctx.accounts.user_account_token_b.
 ↳ to_account_info(),
220                 to: ctx.accounts.custody_token_b.to_account_info()
 ↳ ,
221                 authority: ctx.accounts.owner.to_account_info(),
222             },
223         );
224         anchor_spl::token::transfer(context, res.
 ↳ source_amount_received)?;
225         token_pair.transfer_tokens(
226             ctx.accounts.custody_token_a.to_account_info(),
227             ctx.accounts.user_account_token_a.to_account_info(),
228             ctx.accounts.transfer_authority.clone(),
229             ctx.accounts.token_program.to_account_info(),
230             net_amount_settled_after_fees,
231         )?;
232         token_pair.stats_a.fees_collected =
233             token_pair.stats_a.fees_collected.wrapping_add(
 ↳ settle_fee);
234     } else {
235         let context = CpiContext::new(
236             ctx.accounts.token_program.to_account_info(),
237             Transfer {
238                 from: ctx.accounts.user_account_token_a.
 ↳ to_account_info(),
239                 to: ctx.accounts.custody_token_a.to_account_info()
 ↳ ,
240                 authority: ctx.accounts.owner.to_account_info(),
```

```
241                 },
242             );
243             anchor_spl::token::transfer(context, res.
 ↳ source_amount_received)?;
244             token_pair.transfer_tokens(
245                 ctx.accounts.custody_token_b.to_account_info(),
246                 ctx.accounts.user_account_token_b.to_account_info(),
247                 ctx.accounts.transfer_authority.clone(),
248                 ctx.accounts.token_program.to_account_info(),
249                 net_amount_settled_after_fees,
250             )?;
251             token_pair.stats_b.fees_collected =
252                 token_pair.stats_b.fees_collected.saturating_add(
 ↳ settle_fee);
253         }
254
255         // update pool states
256         msg!("Update pool states");
257         for pool in pools.iter_mut() {
258             pool.update_state(token_pair.min_time_till_expiration,
 ↳ current_time)?;
259             // if pool is complete, switch to the future pool
260             if pool.is_complete(current_time)? {
261                 token_pair.finalize_pool(
262                     pool,
263                     &pool.to_account_info(),
264                     &ctx.accounts.transfer_authority,
265                 )?;
266             }
267         }
268         token_pair.save_pools(&pools)?;
269
270         // update token pair stats
271         msg!("Update token pair stats");
272         token_pair.update_trade_stats(
273             &res,
274             SettlementType::Settlement,
275             &ctx.accounts.oracle_token_a,
276             &ctx.accounts.oracle_token_b,
277         )?;
278
279         // return net unsettled amount
280         let net_amount_required = if res.net_amount_required >= i64::
 ↳ MAX as u64 {
```

```
281         i64::MAX
282     } else {
283         res.net_amount_required as i64
284     };
285     match res.settlement_side {
286         MatchingSide::Internal => Ok(0),
287         MatchingSide::Buy => Ok(net_amount_required),
288         MatchingSide::Sell => Ok(-net_amount_required),
289     }
290 }
291
```

Listing 2: programs/twamm/src/instructions/crank.rs

```
91 pub struct CrankParams {
92     router_instruction_data: Vec<u8>,
93 }
94
95 pub fn crank(ctx: Context<Crank>, params: &CrankParams) -> Result<
↳ i64> {
96     // validate inputs
97     let token_pair = ctx.accounts.token_pair.as_mut();
98     require!(token_pair.allow_cranks, TwammError::CranksNotAllowed
↳ );
99
100     if token_pair.crank_authority != Pubkey::default()
101         && token_pair.crank_authority != ctx.accounts.owner.key()
102     {
103         msg!("Error: Permissionless cranks are not enabled for
↳ this token pair");
104         return err!(TwammError::CranksNotAllowed);
105     }
106
107     // collect and validate pools
108     msg!("Load pools");
109     let (mut pools, router_program) = token_pair.load_pools(ctx.
↳ remaining_accounts)?;
110     require!(!pools.is_empty(), TwammError::NothingToSettle);
```

Listing 3: programs/twamm/src/state/token_pair.rs

```
289     pub fn load_pools<'a>(
290         &self,
```

```
291         accounts: &[AccountInfo<'a>],
292     ) -> Result<(Vec<Account<'a, Pool>>, Pubkey)> {
293         let mut pools: Vec<Account<Pool>> = Vec::with_capacity(
 ↳ accounts.len());
294         let mut pools_found: [bool; TokenPair::MAX_POOLS] = [false
 ↳ ; TokenPair::MAX_POOLS];
295         let mut router_program = Pubkey::default();
296 
297         for (idx, account) in accounts.iter().enumerate() {
298             if account.key == &crate::jupiter::ID || account.key
 ↳ == &Pubkey::default() {
299                 router_program = *account.key;
300                 break;
301             }
302             if state::is_empty_account(account)? {
303                 continue;
304             }
305 
306             // validate account
307             if idx >= TokenPair::MAX_POOLS {
308                 msg!("Error: Unexpected number of pool accounts");
309                 return err!(TwammError::InvalidPoolAddress);
310             }
311             if account.owner != &crate::ID {
312                 return Err(ProgramError::IllegalOwner.into());
313             }
314             if account.try_data_len()? != Pool::LEN {
315                 return Err(ProgramError::InvalidAccountData.into()
 ↳ );
316             }
317             // deserialize pool
318             let pool = Account::<Pool>::try_from(account)?;
319             // validate pool address
320             let pool_address = Pubkey::create_program_address(
321                 &[
322                     b"pool",
323                     self.config_a.custody.as_ref(),
324                     self.config_b.custody.as_ref(),
325                     pool.time_in_force.to_le_bytes().as_slice(),
326                     pool.counter.to_le_bytes().as_slice(),
327                     &[pool.bump],
328                 ],
329                 &crate::ID,
330             )
```

```
331             .map_err(|_| TwammError::InvalidPoolAddress)?;
332
333             if &pool_address != account.key {
334                 msg!("Error: Invalid pool address: Doesn't belong
 ↳ to the given token pair");
335                 return err!(TwammError::InvalidPoolAddress);
336             }
337
338             // validate pool
339             let tif_idx = self.get_tif_index(pool.time_in_force)?;
340             if pools_found[tif_idx] {
341                 msg!("Error: Invalid pool address: Pool with the
 ↳ same TIF already processed");
342                 return err!(TwammError::InvalidPoolAddress);
343             }
344             if pool.counter != self.pool_counters[tif_idx] {
345                 msg!("Error: Invalid pool address: Pool is not
 ↳ current");
346                 return err!(TwammError::InvalidPoolAddress);
347             }
348             pools_found[tif_idx] = true;
349
350             pools.push(pool);
351         }
352
353         // check all current pools have been provided
354         if pools_found != self.current_pool_present {
355             msg!("Error: Not all current pools provided in
 ↳ accounts");
356             return err!(TwammError::InvalidPoolAddress);
357         }
358
359         Ok((pools, router_program))
360     }
```

BVSS:

AO:A/AC:L/AX:H/C:N/I:N/A:N/D:C/Y:M/R:N/S:U - 3.7 - Low (3.7)

Recommendation:

Market makers should be prevented from settling expired pools that pass a certain threshold. Users should be informed that there is no guarantee that their order will be fulfilled and that increasing the orders' duration can increase the chances of fulfillment.

Remediation Plan:

FUTURE RELEASE: The Solana Labs team acknowledged the issue and will resolve it in a future release. The planned solution is to prevent expired pools that have passed a certain threshold from being settled.

4.2 (HAL-02) LACK OF EMERGENCY-STOP PATTERN - LOW (2.5)

Description:

DeFi protocols are high-value targets for malicious users and are subject to constant scrutiny. They can be vulnerable not only to security vulnerabilities, but also to other dependencies such as oracles, vulnerabilities in the chain or chain downtime. Due to this, we suggest that the twamm program implement functionality to pause the contract.

In the event of an incident, the owner would not be able to stop any possible malicious actions. In addition, if an incident were to take place, it allows the team more time to consider the best course of action.

BVSS:

AO:A/AC:L/AX:L/C:L/I:L/A:L/D:L/Y:L/R:P/S:U (2.5)

Recommendation:

Consider adding a pause function to the program.

Remediation Plan:

RISK ACCEPTED: The Solana Labs team accepted the risk of this finding.

4.3 (HAL-03) MINIMUM MULTISIG THRESHOLD CHECK MISSING - INFORMATIONAL (0.9)

Description:

The Init instruction allows the upgrade authority to initialize the multisig account and set the multisig's signers. The instruction handler requires the transaction sender to provide a selection of accounts and parameters, including min_signatures. This parameter sets the signatures threshold required for a transaction to be considered valid. Likewise, the SetAdminSigners instruction handler allows setting a new signers list and new min_signatures.

However, both instructions handlers allow setting the min_signatures field of multisig with a value equal to 1. Setting the threshold to 1 results in having no multisig functionality at all because only one user controls the account.

Code Location:

Listing 4: src/instruction/init.rs

```
48 pub struct InitParams {
49     pub min_signatures: u8,
50 }
51
52 pub fn init(ctx: Context<Init>, params: &InitParams) -> Result<()>
↳  {
53     // initialize multisig, this will fail if account is already
↳ initialized
54     let mut multisig = ctx.accounts.multisig.load_init()?;
55
56     multisig.set_signers(ctx.remaining_accounts, params.
↳ min_signatures)?;
57
58     // record multisig PDA bump
59     multisig.bump = *ctx
```

```
60          .bumps
61          .get("multisig")
62          .ok_or(ProgramError::InvalidSeeds)?;
63
64      Ok(())
65  }
```

Listing 5: programs/twamm/src/instructions/set_admin_signers.rs

```
25  pub struct SetAdminSignersParams {
26      pub min_signatures: u8,
27  }
28
29  pub fn set_admin_signers<'info>(
30      ctx: Context<'_, '_, '_, 'info, SetAdminSigners<'info>>,
31      params: &SetAdminSignersParams,
32  ) -> Result<u8> {
33      // validate signatures
34      let mut multisig = ctx.accounts.multisig.load_mut()?;
35
36      let signatures_left = multisig.sign_multisig(
37          &ctx.accounts.admin,
38          &Multisig::get_account_infos(&ctx)[1..],
39          &Multisig::get_instruction_data(AdminInstruction::
↳ SetAdminSigners, params)?,
40      )?;
41      if signatures_left > 0 {
42          msg!(
43              "Instruction has been signed but more signatures are
↳ required: {}",
44              signatures_left
45          );
46          return Ok(signatures_left);
47      }
48
49      // set new admin signers
50      multisig.set_signers(ctx.remaining_accounts, params.
↳ min_signatures)?;
51
52      Ok(0)
53  }
```

Listing 6: programs/twamm/src/state/multisig.rs

```
13 pub struct Multisig {
14     pub num_signers: u8,
15     pub num_signed: u8,
16     pub min_signatures: u8,
17     pub instruction_accounts_len: u8,
18     pub instruction_data_len: u16,
19     pub instruction_hash: u64,
20     pub signers: [Pubkey; 6], // Multisig::MAX_SIGNERS
21     pub signed: [bool; 6],    // Multisig::MAX_SIGNERS
22     pub bump: u8,
23 }
```

BVSS:

AO:S/AC:L/AX:L/C:N/I:N/A:N/D:H/Y:H/R:P/S:U (0.9)

Recommendation:

It is recommended to add a check to verify that the value of `min_signatures` passed as a parameter is equal to or greater than three, as well as the number of remaining accounts provided as admins.

Remediation Plan:

ACKNOWLEDGED: The Solana Labs team acknowledged this finding.

4.4 (HAL-04) ORACLE OWNER VALIDATION - INFORMATIONAL (0.1)

Description:

The Twamm program uses Pyth oracle price feeds to retrieve accurate pricing information to match and settle orders at a fair market value, due to this, secure and reliable price feeds are critical to the security of the program. Unfortunately, the program fails to validate the owner of the oracles.

However, when admins initialize the token pair with the init_token_pair instruction and update the oracle configuration with set_oracle_config they provide the instruction with oracle_account_token_a & oracle_account_token_b pubkeys. Neither of these pubkeys are checked to be owned by the Pyth program, if the incorrect oracle account is provided this could lead to a temporary denial of service.

Code Location:

Listing 7: programs/twamm/src/instructions/init_token_pair.rs (Lines 160,165)

```
112 pub fn init_token_pair<'info>(
113     ctx: Context<'_, '_, '_, 'info, InitTokenPair<'info>>,
114     params: &InitTokenPairParams,
115 ) -> Result<u8> {
116     // validate signatures
117     let mut multisig = ctx.accounts.multisig.load_mut()?;
118 
119     let signatures_left = multisig.sign_multisig(
120         &ctx.accounts.admin,
121         &Multisig::get_account_infos(&ctx)[1..],
122         &Multisig::get_instruction_data(AdminInstruction::
 ↳ InitTokenPair, params)?,
123     )?;
124     if signatures_left > 0 {
125         msg!(
126             "Instruction has been signed but more signatures are
```

```
 ↳ required: {}",
127                 signatures_left
128             );
129             return Ok(signatures_left);
130         }
131
132         // record token pair data
133         let token_pair = ctx.accounts.token_pair.as_mut();
134         if token_pair.config_a.mint != Pubkey::default() {
135             // return error if token pair is already initialized
136             return Err(ProgramError::AccountAlreadyInitialized.into())
 ↳ ;
137         }
138
139         token_pair.allow_deposits = params.allow_deposits;
140         token_pair.allow_withdrawals = params.allow_withdrawals;
141         token_pair.allow_cranks = params.allow_cranks;
142         token_pair.allow_settlements = params.allow_settlements;
143
144         token_pair.fee_numerator = params.fee_numerator;
145         token_pair.fee_denominator = params.fee_denominator;
146         token_pair.settle_fee_numerator = params.settle_fee_numerator;
147         token_pair.settle_fee_denominator = params.
 ↳ settle_fee_denominator;
148         token_pair.config_a.crank_reward = params.crank_reward_token_a
 ↳ ;
149         token_pair.config_b.crank_reward = params.crank_reward_token_b
 ↳ ;
150
151         token_pair.config_a.min_swap_amount = params.
 ↳ min_swap_amount_token_a;
152         token_pair.config_b.min_swap_amount = params.
 ↳ min_swap_amount_token_b;
153         token_pair.max_swap_price_diff = params.max_swap_price_diff;
154         token_pair.max_unsettled_amount = params.max_unsettled_amount;
155         token_pair.min_time_till_expiration = params.
 ↳ min_time_till_expiration;
156
157         token_pair.config_a.max_oracle_price_error = params.
 ↳ max_oracle_price_error_token_a;
158         token_pair.config_a.max_oracle_price_age_sec = params.
 ↳ max_oracle_price_age_sec_token_a;
159         token_pair.config_a.oracle_type = params.oracle_type_token_a;
160         token_pair.config_a.oracle_account = params.
```

```
↳ oracle_account_token_a;
161
162     token_pair.config_b.max_oracle_price_error = params.
↳ max_oracle_price_error_token_b;
163     token_pair.config_b.max_oracle_price_age_sec = params.
↳ max_oracle_price_age_sec_token_b;
164     token_pair.config_b.oracle_type = params.oracle_type_token_b;
165     token_pair.config_b.oracle_account = params.
↳ oracle_account_token_b;
```

Listing 8: programs/twamm/src/instructions/set_oracle_config.rs (Lines 74,79)

```
49 pub fn set_oracle_config<'info>(
50     ctx: Context<'_, '_, '_, 'info, SetOracleConfig<'info>>,
51     params: &SetOracleConfigParams,
52 ) -> Result<u8> {
53     // validate signatures
54     let mut multisig = ctx.accounts.multisig.load_mut()?;
55
56     let signatures_left = multisig.sign_multisig(
57         &ctx.accounts.admin,
58         &Multisig::get_account_infos(&ctx)[1..],
59         &Multisig::get_instruction_data(AdminInstruction::
↳ SetOracleConfig, params)?,
60     )?;
61     if signatures_left > 0 {
62         msg!(
63             "Instruction has been signed but more signatures are
↳ required: {}",
64             signatures_left
65         );
66         return Ok(signatures_left);
67     }
68
69     // update permissions
70     let token_pair = ctx.accounts.token_pair.as_mut();
71     token_pair.config_a.max_oracle_price_error = params.
↳ max_oracle_price_error_token_a;
72     token_pair.config_a.max_oracle_price_age_sec = params.
↳ max_oracle_price_age_sec_token_a;
73     token_pair.config_a.oracle_type = params.oracle_type_token_a;
74     token_pair.config_a.oracle_account = params.
```

```
↳ oracle_account_token_a;
75
76      token_pair.config_b.max_oracle_price_error = params.
↳ max_oracle_price_error_token_b;
77      token_pair.config_b.max_oracle_price_age_sec = params.
↳ max_oracle_price_age_sec_token_b;
78      token_pair.config_b.oracle_type = params.oracle_type_token_b;
79      token_pair.config_b.oracle_account = params.
↳ oracle_account_token_b;
```

BVSS:

AO:S/AC:L/AX:L/C:N/I:N/A:L/D:N/Y:N/R:F/S:U (0.1)

Recommendation:

Validate that the oracle accounts are owned by the Pyth program.

Remediation Plan:

ACKNOWLEDGED: The Solana Labs team acknowledged this finding.

4.5 (HAL-05) ZERO AMOUNT PROCESSING - INFORMATIONAL (0.0)

Description:

The withdraw_fees instruction allows admins to withdraw fees collected from users interacting with the program, such as canceling orders, settling orders and cranking. However, the withdraw_fees instruction allows for zero amount processing, It takes a WithdrawFeesParams struct and uses it for further processing. However, no sanity check is done on the parameter's value; therefore it is possible to process any arbitrary amounts, including 0.

Code Location:

Listing 9: programs/twamm/src/instructions/withdraw_fees.rs

```
75 pub struct WithdrawFeesParams {
76     pub amount_token_a: u64,
77     pub amount_token_b: u64,
78     pub amount_sol: u64,
79 }
80
81 pub fn withdraw_fees<'info>(
82     ctx: Context<'_, '_, '_, 'info, WithdrawFees<'info>>,
83     params: &WithdrawFeesParams,
84 ) -> Result<u8> {
85     // validate signatures
86     let mut multisig = ctx.accounts.multisig.load_mut()?;
87
88     let signatures_left = multisig.sign_multisig(
89         &ctx.accounts.admin,
90         &Multisig::get_account_infos(&ctx)[1..],
91         &Multisig::get_instruction_data(AdminInstruction::
↳ WithdrawFees, params)?,
92     )?;
93     if signatures_left > 0 {
94         msg!(
95             "Instruction has been signed but more signatures are
↳ required: {}",
```

```
96              signatures_left
97          );
98          return Ok(signatures_left);
99      }
100
101     // transfer token fees from the custody to the receiver
102     let token_pair = ctx.accounts.token_pair.as_mut();
103
104     if params.amount_token_a > 0 {
105         msg!(
106             "Withdraw token A fees: {} / {}",
107             params.amount_token_a,
108             token_pair.stats_a.fees_collected
109         );
110         if token_pair.stats_a.fees_collected < params.
 ↳ amount_token_a {
111             return Err(ProgramError::InsufficientFunds.into());
112         }
113         token_pair.stats_a.fees_collected =
114             math::checked_sub(token_pair.stats_a.fees_collected,
 ↳ params.amount_token_a)?;
115
116         token_pair.transfer_tokens(
117             ctx.accounts.custody_token_a.to_account_info(),
118             ctx.accounts.receiver_token_a.to_account_info(),
119             ctx.accounts.transfer_authority.clone(),
120             ctx.accounts.token_program.to_account_info(),
121             params.amount_token_a,
122         )?;
123     }
124
125     if params.amount_token_b > 0 {
126         msg!(
127             "Withdraw token B fees: {} / {}",
128             params.amount_token_b,
129             token_pair.stats_b.fees_collected
130         );
131         if token_pair.stats_b.fees_collected < params.
 ↳ amount_token_b {
132             return Err(ProgramError::InsufficientFunds.into());
133         }
134         token_pair.stats_b.fees_collected =
135             math::checked_sub(token_pair.stats_b.fees_collected,
 ↳ params.amount_token_b)?;
```

```
136
137         token_pair.transfer_tokens(
138             ctx.accounts.custody_token_b.to_account_info(),
139             ctx.accounts.receiver_token_b.to_account_info(),
140             ctx.accounts.transfer_authority.clone(),
141             ctx.accounts.token_program.to_account_info(),
142             params.amount_token_b,
143         )?;
144     }
145
146     // transfer sol fees from the custody to the receiver
147     if params.amount_sol > 0 {
148         let balance = ctx.accounts.transfer_authority.try_lamports
 ↳ ()?;
149         let min_balance = sysvar::rent::Rent::get().unwrap().
 ↳ minimum_balance(0);
150         let available_balance = if balance > min_balance {
151             math::checked_sub(balance, min_balance)?
152         } else {
153             0
154         };
155         msg!(
156             "Withdraw SOL fees: {} / {}",
157             params.amount_sol,
158             available_balance
159         );
160         if available_balance < params.amount_sol {
161             return Err(ProgramError::InsufficientFunds.into());
162         }
163
164         state::transfer_sol_from_owned(
165             ctx.accounts.transfer_authority.to_account_info(),
166             ctx.accounts.receiver_sol.to_account_info(),
167             params.amount_sol,
168         )?;
169     }
170
171     Ok(0)
172 }
```

BVSS:

AO:S/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:F/S:U (0.0)

Recommendation:

Validate the amount_token_a, amount_token_b & amount_sol before processing the instruction further in order to avoid doing unnecessary calculations when the amount is 0.

Remediation Plan:

ACKNOWLEDGED: The Solana Labs team acknowledged this finding.

4.6 (HAL-06) POSSIBLE RUST PANICS DUE TO UNSAFE UNWRAP USAGE - INFORMATIONAL (0.0)

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 the 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:

Listing 10: programs/twamm/src/instructions/withdraw_fees.rs (Line 149)

```
81 pub fn withdraw_fees<'info>(
82     ctx: Context<'_, '_, '_, 'info, WithdrawFees<'info>>,
83     params: &WithdrawFeesParams,
84 ) -> Result<u8> {
85     // validate signatures
86     let mut multisig = ctx.accounts.multisig.load_mut()?;
87
88     let signatures_left = multisig.sign_multisig(
89         &ctx.accounts.admin,
90         &Multisig::get_account_infos(&ctx)[1..],
91         &Multisig::get_instruction_data(AdminInstruction::
↳ WithdrawFees, params)?,
92     )?;
93     if signatures_left > 0 {
94         msg!(
95             "Instruction has been signed but more signatures are
↳ required: {}",
96             signatures_left
97         );
98         return Ok(signatures_left);
99     }
```

```
100
101     // transfer token fees from the custody to the receiver
102     let token_pair = ctx.accounts.token_pair.as_mut();
103
104     if params.amount_token_a > 0 {
105         msg!(
106             "Withdraw token A fees: {} / {}",
107             params.amount_token_a,
108             token_pair.stats_a.fees_collected
109         );
110         if token_pair.stats_a.fees_collected < params.
 ↳ amount_token_a {
111             return Err(ProgramError::InsufficientFunds.into());
112         }
113         token_pair.stats_a.fees_collected =
114             math::checked_sub(token_pair.stats_a.fees_collected,
 ↳ params.amount_token_a)?;
115
116         token_pair.transfer_tokens(
117             ctx.accounts.custody_token_a.to_account_info(),
118             ctx.accounts.receiver_token_a.to_account_info(),
119             ctx.accounts.transfer_authority.clone(),
120             ctx.accounts.token_program.to_account_info(),
121             params.amount_token_a,
122         )?;
123     }
124
125     if params.amount_token_b > 0 {
126         msg!(
127             "Withdraw token B fees: {} / {}",
128             params.amount_token_b,
129             token_pair.stats_b.fees_collected
130         );
131         if token_pair.stats_b.fees_collected < params.
 ↳ amount_token_b {
132             return Err(ProgramError::InsufficientFunds.into());
133         }
134         token_pair.stats_b.fees_collected =
135             math::checked_sub(token_pair.stats_b.fees_collected,
 ↳ params.amount_token_b)?;
136
137         token_pair.transfer_tokens(
138             ctx.accounts.custody_token_b.to_account_info(),
139             ctx.accounts.receiver_token_b.to_account_info(),
```

```
140             ctx.accounts.transfer_authority.clone(),
141             ctx.accounts.token_program.to_account_info(),
142             params.amount_token_b,
143         )?;
144     }
145
146     // transfer sol fees from the custody to the receiver
147     if params.amount_sol > 0 {
148         let balance = ctx.accounts.transfer_authority.try_lamports
 ↳ ()?;
149         let min_balance = sysvar::rent::Rent::get().unwrap().
 ↳ minimum_balance(0);
```

BVSS:

AO:S/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:F/S:U (0.0)

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:

SOLVED: The Solana Labs team solved this issue in commit bc586779c70f4806f6d2e167d33834dd46816ccb by removing all unwrap functions.

4.7 (HAL-07) INTEGER OVERFLOW - INFORMATIONAL (0.0)

Description:

Integer overflow/underflow occurs when an arithmetic operation attempts to create a numeric value that is outside the range that can be represented by a given number of bits, either greater than the maximum or less than the minimum representable value. Although integer overflows and underflows do not cause Rust to panic in the release mode, the consequences could be dire if the result of those operations is used in financial calculations.

Code Location:

Listing 11: programs/twamm/src/state/token_pair.rs (Lines 506,507,508,509,518,519,539,540,550,551)

```
472
473     pub fn settle_pools(
474         &self,
475         pools: &mut [&mut Pool],
476         supply_side: MatchingSide,
477         token_a_change: u64,
478         token_b_change: u64,
479         exchange_rate: OraclePrice,
480         oracle_price: OraclePrice,
481         current_time: i64,
482     ) -> Result<Settlement> {
483         let mut outstanding_a: [u64; TokenPair::MAX_POOLS] = [0;
 ↳ TokenPair::MAX_POOLS];
484         let mut outstanding_b: [u64; TokenPair::MAX_POOLS] = [0;
 ↳ TokenPair::MAX_POOLS];
485         let mut total_outstanding_a = 0;
486         let mut total_outstanding_b = 0;
487         let mut res = Settlement {
488             net_amount_settled: 0,
489             net_amount_required: 0,
490             source_amount_received: 0,
491             total_amount_settled_a: 0,
492             total_amount_settled_b: 0,
```

```
493             settlement_side: MatchingSide::Internal,
494         };
495
496         // calculate outstanding amount per pool and in total
497         // try settle each pool with itself along the way
498         assert!(!pools.is_empty() && pools.len() < TokenPair::
 ↳ MAX_POOLS);
499         for (idx, pool) in pools.iter_mut().enumerate() {
500             let outstanding_sell = pool
501                 .sell_side
502                 .get_unsettled_amount(pool.expiration_time,
 ↳ current_time)?;
503             let outstanding_buy = pool
504                 .buy_side
505                 .get_unsettled_amount(pool.expiration_time,
 ↳ current_time)?;
506             if outstanding_sell > 0 || outstanding_buy > 0 {
507                 total_outstanding_a += outstanding_sell;
508                 total_outstanding_b += outstanding_buy;
509                 outstanding_a[idx] += outstanding_sell;
510                 outstanding_b[idx] += outstanding_buy;
511
512                 let (settled_a, settled_b) = self.settle_sides(
513                     &mut pool.sell_side,
514                     &mut pool.buy_side,
515                     outstanding_sell,
516                     outstanding_buy,
517                     oracle_price,
518                 )?;
519                 outstanding_a[idx] -= settled_a;
520                 outstanding_b[idx] -= settled_b;
521             }
522         }
523
524         if total_outstanding_a == 0 && total_outstanding_b == 0 {
525             return Ok(res);
526         }
527
528         // settle pools internally
529         for idx in 0..(pools.len() - 1) {
530             if outstanding_a[idx] != 0 || outstanding_b[idx] != 0
 ↳ {
531                 for other_idx in (idx + 1)..pools.len() {
```

```
532                     if outstanding_a[idx] != 0 && outstanding_b[
 ↳ other_idx] != 0 {
533                         let (settled_a, settled_b) = self.
 ↳ settle_sides(
534                             &mut pools[idx].sell_side,
535                             &mut pools[other_idx].buy_side,
536                             outstanding_a[idx],
537                             outstanding_b[other_idx],
538                             oracle_price,
539                         )?;
540                         outstanding_a[idx] -= settled_a;
541                         outstanding_b[other_idx] -= settled_b;
542                     }
543                     if outstanding_b[idx] != 0 && outstanding_a[
 ↳ other_idx] != 0 {
544                         let (settled_a, settled_b) = self.
 ↳ settle_sides(
545                             &mut pools[other_idx].sell_side,
546                             &mut pools[idx].buy_side,
547                             outstanding_a[other_idx],
548                             outstanding_b[idx],
549                             oracle_price,
550                         )?;
551                         outstanding_a[other_idx] -= settled_a;
552                         outstanding_b[idx] -= settled_b;
553                     }
554                 }
555             }
556         }
```

BVSS:

AO:S/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:F/S:U (0.0)

Recommendation:

These findings were determined to be a false positive due to overflow-checks being enabled in cargo.toml, but the Solana Labs team should be aware if overflow-checks are ever removed, the program will be vulnerable to integer under/overflows.

Remediation Plan:

SOLVED: The Solana Labs team solved this issue in commit bc586779c70f4806f6d2e167d33834dd46816ccb by replacing all arithmetic operations with checked math functions.

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.

5.1 MULTISIG BYPASS

Description:

The Twamm program has several instructions that require multiple admins to sign the same transaction before it takes effect. If an attacker can bypass this, they could execute instructions that could negatively affect users. Such as changing the oracles, changing fees, withdrawing fees and more.

Results:

No vulnerabilities were identified.

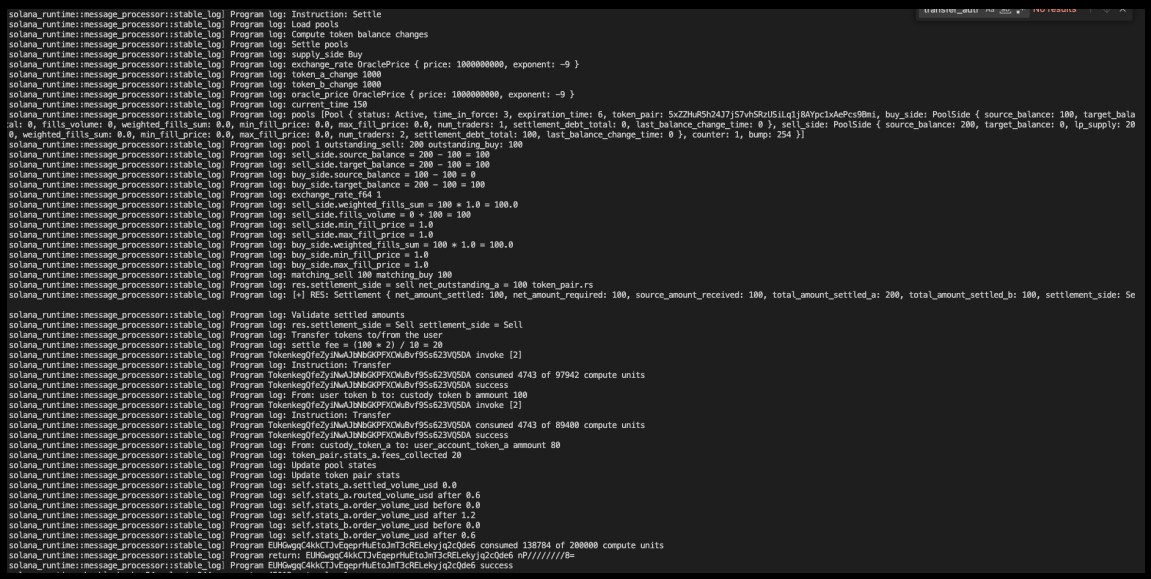
5.2 SETTLE INTERNAL ORDERS WITH ARBITRAGE

Description:

Users can create orders on twamm giving market makers the opportunity to settle them directly with the programs pools. However, if there are two trades with opposite positions, market makers should only be able to settle the outstanding net balance between them. If a market maker was able to accidentally bypass this and settle the whole balance, it could lead to a loss of funds.

Results:

No vulnerabilities were identified.



5.3 WITHDRAW MORE LIQUIDITY THAN ADDED

Description:

When users place an order, an order account is created to track the details of their order. The order contains an lp_balance field, which represents how much liquidity they've provided to the pool. When executing a cancel_order instruction, users can specify how much liquidity they would like to withdraw, and the program will debit the lp_balance & credit them with tokens from its pools. If a user is able to withdraw more liquidity than they provided or if the program ignores other users lp_balance a user could drain the pools.

However, the program prevents this by:

1) When withdrawing liquidity from a completed order, using their lp_balance instead of the amount specified by the user
2) When withdrawing liquidity from an order that isn't completed, checking if the amount specified by the user is less than or equal to their lp_balance

Results:

No vulnerabilities were identified

MANUAL TESTING

```
DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: CancelOrder
DEBUG solana_runtime::message_processor::stable_log] Program log: is pool complete? true
DEBUG solana_runtime::message_processor::stable_log] Program log: Compute balance changes
DEBUG solana_runtime::message_processor::stable_log] Program log: withdraw_amount_fees 20
DEBUG solana_runtime::message_processor::stable_log] Program log: Update order data
DEBUG solana_runtime::message_processor::stable_log] Program log: Update pool data
DEBUG solana_runtime::message_processor::stable_log] Program log: Transfer tokens to the user
DEBUG solana_runtime::message_processor::stable_log] Program log: transfer 0 from custody_token_a to user_account_token_a
DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Transfer
DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 4838 of 171103 compute units
DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
DEBUG solana_runtime::message_processor::stable_log] Program log: transfer 80 from custody_token_b to user_account_token_b
DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Transfer
DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 4743 of 162442 compute units
DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
DEBUG solana_runtime::message_processor::stable_log] Program log: Update token pair stats
DEBUG solana_runtime::message_processor::stable_log] Program log: token_pair.stats_b.fees_collected = 20
DEBUG solana_runtime::message_processor::stable_log] Program log: Close order account
DEBUG solana_runtime::message_processor::stable_log] Program log: Update pool state
DEBUG solana_runtime::message_processor::stable_log] Program EUHGwgqC4kkCTJvEqeprHuEtoJmT3cRELekyjq2cQde6 consumed 51536 of 200000 compute units
DEBUG solana_runtime::message_processor::stable_log] Program EUHGwgqC4kkCTJvEqeprHuEtoJmT3cRELekyjq2cQde6 success
DEBUG solana_runtime::bank] check: 22us load: 300us execute: 23348us txs_len=1
DEBUG solana_runtime::bank] store: 472us txs_len=1
DEBUG solana_runtime::accounts] bank unlock accounts
DEBUG solana_runtime::bank] processing transactions: 1
DEBUG solana_runtime::message_processor::stable_log] Program EUHGwgqC4kkCTJvEqeprHuEtoJmT3cRELekyjq2cQde6 invoke [1]
DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: CancelOrder
DEBUG solana_runtime::message_processor::stable_log] Program log: is pool complete? true
DEBUG solana_runtime::message_processor::stable_log] Program log: Compute balance changes
DEBUG solana_runtime::message_processor::stable_log] Program log: withdraw_amount_fees 20
DEBUG solana_runtime::message_processor::stable_log] Program log: Update order data
DEBUG solana_runtime::message_processor::stable_log] Program log: Update pool data
DEBUG solana_runtime::message_processor::stable_log] Program log: Transfer tokens to the user
DEBUG solana_runtime::message_processor::stable_log] Program log: transfer 0 from custody_token_a to user_account_token_a
DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Transfer
DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 4838 of 171103 compute units
DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
DEBUG solana_runtime::message_processor::stable_log] Program log: transfer 80 from custody_token_b to user_account_token_b
DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Transfer
DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 4743 of 162442 compute units
DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
DEBUG solana_runtime::message_processor::stable_log] Program log: Update token pair stats
DEBUG solana_runtime::message_processor::stable_log] Program log: token_pair.stats_b.fees_collected = 40
DEBUG solana_runtime::message_processor::stable_log] Program log: Close order account
DEBUG solana_runtime::message_processor::stable_log] Program log: Update pool state
DEBUG solana_runtime::message_processor::stable_log] Program log: Close pool account
DEBUG solana_runtime::message_processor::stable_log] Program EUHGwgqC4kkCTJvEqeprHuEtoJmT3cRELekyjq2cQde6 consumed 50316 of 200000 compute units
DEBUG solana_runtime::message_processor::stable_log] Program EUHGwgqC4kkCTJvEqeprHuEtoJmT3cRELekyjq2cQde6 success
```

5.4 ORACLE TRADING STATUS MANIPULATION

Description:

Twamm relies on Pyth oracle PriceFeeds when matching, settling trades and cranking. This creates a critical point of failure, as the program needs continuous and reliable pricing data. If the program was to use old or outdated pricing data, an arbitrager could take advantage of this and drain the pool.

Often times developers are unaware Pyth oracles have statuses and can range from trading, unknown, halted, auction and ignored. The status of the oracle will determine the publish_time of the PriceFeed and if developers don't validate that the publish_time is within an acceptable threshold they're at risk of using outdated pricing info.

Results:

Twamm ensures the oracles publish-time meets the admins' threshold

```
DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Crank
DEBUG solana_runtime::message_processor::stable_log] Program log: Load pools
DEBUG solana_runtime::message_processor::stable_log] Program log: Error: Pyth oracle price is stale
DEBUG solana_runtime::message_processor::stable_log] Program log: AnchorError thrown in programs/twamm/src/oracle.rs:141. Error Code: StaleOraclePrice. Error Number: 6021. Error Message: Stale oracle price.
DEBUG solana_runtime::message_processor::stable_log] Program EUHGwgqC4kkCTJvEqeprHuEtoJmT3cRELekyjq2cQde6 consumed 25664 of 200000 compute units
DEBUG solana_runtime::message_processor::stable_log] Program EUHGwgqC4kkCTJvEqeprHuEtoJmT3cRELekyjq2cQde6 failed: custom program error: 0x1785
DEBUG solana_runtime::bank] check: 29us load: 328us execute: 10090us txs_len=1
```

5.5 ORACLE CONFIDENCE INTERVAL MANIPULATION

Description:

Similar to the ORACLE TRADING STATUS MANIPULATION test Pricefeeds have a confidence interval and during times of market stress or volatility, it can vary to a large degree and lead to inaccurate pricing.

Results:

Twamm ensures the oracles confidence threshold meets the admins' threshold

```
DEBUG solana_runtime::message_processor::stable_log] Program EUHGwgqC4kkCTJvEqeprHuEtoJmT3cRELekyjq2cQde6 invoke [1]
DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Crank
DEBUG solana_runtime::message_processor::stable_log] Program log: Load pools
DEBUG solana_runtime::message_processor::stable_log] Program log: Error: Pyth oracle price is out of bounds
DEBUG solana_runtime::message_processor::stable_log] Program log: AnchorError thrown in programs/twamm/src/oracle.rs:149. Error Code: InvalidOraclePrice. Error Number: 6022. Error Message: Invalid oracle price.
DEBUG solana_runtime::message_processor::stable_log] Program EUHGwgqC4kkCTJvEqeprHuEtoJmT3cRELekyjq2cQde6 consumed 26427 of 200000 compute units
DEBUG solana_runtime::message_processor::stable_log] Program EUHGwgqC4kkCTJvEqeprHuEtoJmT3cRELekyjq2cQde6 failed: custom program error: 0x1786
DEBUG solana_runtime::bank] check: 33us load: 331us execute: 10949us txs_len=1
```

AUTOMATED TESTING

6.1 AUTOMATED VULNERABILITY SCANNING

Description:

Halborn used automated security scanners to assist with the 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 in scope and sent the compiled results to analyzers to locate well-known vulnerabilities.

Results:

No vulnerabilities were identified, overflows are prevented with the release feature overflow-checks in cargo.toml

```
==========This arithmetic operation may be UNSAFE!==========
Found a potential vulnerability at line 506, column 17 in programs/twamm/src/state/token_pair.rs
The add operation may result in overflows:

 500|                .sell_side
 501|                .get_unsettled_amount(pool.expiration_time, current_time)?;
 502|            let outstanding_buy = pool
 503|                .buy_side
 504|                .get_unsettled_amount(pool.expiration_time, current_time)?;
 505|            if outstanding_sell > 0 || outstanding_buy > 0 {
>506|                total_outstanding_a += outstanding_sell;
 507|                total_outstanding_b += outstanding_buy;
 508|                outstanding_a[idx] += outstanding_sell;
 509|                outstanding_b[idx] += outstanding_buy;
 510|
 511|                let (settled_a, settled_b) = self.settle_sides(
 512|                    &mut pool.sell_side,
>>>Stack Trace:
>>>twamm::try_entry::hc06acca7e89bade9 [programs/twamm/src/lib.rs:37]
>>>  twamm::dispatch::hf7aa41276ed49a8e [programs/twamm/src/lib.rs:37]
>>>    twamm::__private::__global::settle::h5dcd1b0add718cbf [programs/twamm/src/lib.rs:37]
>>>      twamm::twamm::settle::h46cd4315f58a6f44 [programs/twamm/src/lib.rs:37]
>>>        twamm::instructions::settle::settle::hcd37cff6f01e4594 [programs/twamm/src/lib.rs:152]
>>>          twamm::state::token_pair::TokenPair::settle_pools::hdaff4210dcccf2ac [programs/twamm/src/instructions/settle.rs:152]

==========This arithmetic operation may be UNSAFE!==========
Found a potential vulnerability at line 507, column 17 in programs/twamm/src/state/token_pair.rs
The add operation may result in overflows:

 501|                .get_unsettled_amount(pool.expiration_time, current_time)?;
 502|            let outstanding_buy = pool
 503|                .buy_side
 504|                .get_unsettled_amount(pool.expiration_time, current_time)?;
 505|            if outstanding_sell > 0 || outstanding_buy > 0 {
 506|                total_outstanding_a += outstanding_sell;
>507|                total_outstanding_b += outstanding_buy;
 508|                outstanding_a[idx] += outstanding_sell;
 509|                outstanding_b[idx] += outstanding_buy;
 510|
 511|                let (settled_a, settled_b) = self.settle_sides(
 512|                    &mut pool.sell_side,
 513|                    &mut pool.buy_side,
>>>Stack Trace:
>>>twamm::try_entry::hc06acca7e89bade9 [programs/twamm/src/lib.rs:37]
>>>  twamm::dispatch::hf7aa41276ed49a8e [programs/twamm/src/lib.rs:37]
>>>    twamm::__private::__global::settle::h5dcd1b0add718cbf [programs/twamm/src/lib.rs:37]
>>>      twamm::twamm::settle::h46cd4315f58a6f44 [programs/twamm/src/lib.rs:37]
>>>        twamm::instructions::settle::settle::hcd37cff6f01e4594 [programs/twamm/src/lib.rs:152]
>>>          twamm::state::token_pair::TokenPair::settle_pools::hdaff4210dcccf2ac [programs/twamm/src/instructions/settle.rs:152]

==========This arithmetic operation may be UNSAFE!==========
Found a potential vulnerability at line 508, column 17 in programs/twamm/src/state/token_pair.rs
The add operation may result in overflows:

 502|            let outstanding_buy = pool
 503|                .buy_side
 504|                .get_unsettled_amount(pool.expiration_time, current_time)?;
 505|            if outstanding_sell > 0 || outstanding_buy > 0 {
 506|                total_outstanding_a += outstanding_sell;
 507|                total_outstanding_b += outstanding_buy;
>508|                outstanding_a[idx] += outstanding_sell;
```

AUTOMATED TESTING

```
>508|                outstanding_a[idx] += outstanding_sell;
 509|                outstanding_b[idx] += outstanding_buy;
 510|
 511|                let (settled_a, settled_b) = self.settle_sides(
 512|                    &mut pool.sell_side,
 513|                    &mut pool.buy_side,
 514|                    outstanding_sell,
>>>Stack Trace:
>>>twamm::try_entry::hc06acca7e89bade9 [programs/twamm/src/lib.rs:37]
>>>  twamm::dispatch::hf7aa41276ed49a8e [programs/twamm/src/lib.rs:37]
>>>    twamm::__private::__global::settle::h5dcd1b0add718cbf [programs/twamm/src/lib.rs:37]
>>>      twamm::twamm::settle::h46cd4315f58a6f44 [programs/twamm/src/lib.rs:37]
>>>        twamm::instructions::settle::settle::hcd37cff6f01e4594 [programs/twamm/src/lib.rs:152]
>>>          twamm::state::token_pair::TokenPair::settle_pools::hdaff4210dcccf2ac [programs/twamm/src/instructions/settle.rs:152]

==========This arithmetic operation may be UNSAFE!==========
Found a potential vulnerability at line 509, column 17 in programs/twamm/src/state/token_pair.rs
The add operation may result in overflows:

 503|                .buy_side
 504|                .get_unsettled_amount(pool.expiration_time, current_time)?;
 505|            if outstanding_sell > 0 || outstanding_buy > 0 {
 506|                total_outstanding_a += outstanding_sell;
 507|                total_outstanding_b += outstanding_buy;
 508|                outstanding_a[idx] += outstanding_sell;
>509|                outstanding_b[idx] += outstanding_buy;
 510|
 511|                let (settled_a, settled_b) = self.settle_sides(
 512|                    &mut pool.sell_side,
 513|                    &mut pool.buy_side,
 514|                    outstanding_sell,
 515|                    outstanding_buy,
>>>Stack Trace:
>>>twamm::try_entry::hc06acca7e89bade9 [programs/twamm/src/lib.rs:37]
>>>  twamm::dispatch::hf7aa41276ed49a8e [programs/twamm/src/lib.rs:37]
>>>    twamm::__private::__global::settle::h5dcd1b0add718cbf [programs/twamm/src/lib.rs:37]
>>>      twamm::twamm::settle::h46cd4315f58a6f44 [programs/twamm/src/lib.rs:37]
>>>        twamm::instructions::settle::settle::hcd37cff6f01e4594 [programs/twamm/src/lib.rs:152]
>>>          twamm::state::token_pair::TokenPair::settle_pools::hdaff4210dcccf2ac [programs/twamm/src/instructions/settle.rs:152]

==========This arithmetic operation may be UNSAFE!==========
Found a potential vulnerability at line 518, column 17 in programs/twamm/src/state/token_pair.rs
The sub operation may result in underflows:

 512|                    &mut pool.sell_side,
 513|                    &mut pool.buy_side,
 514|                    outstanding_sell,
 515|                    outstanding_buy,
 516|                    oracle_price,
 517|                )?;
>518|                outstanding_a[idx] -= settled_a;
 519|                outstanding_b[idx] -= settled_b;
 520|            }
 521|        }
 522|
 523|        if total_outstanding_a == 0 && total_outstanding_b == 0 {
 524|            return Ok(res);
>>>Stack Trace:
>>>twamm::try_entry::hc06acca7e89bade9 [programs/twamm/src/lib.rs:37]
>>>  twamm::dispatch::hf7aa41276ed49a8e [programs/twamm/src/lib.rs:37]
>>>    twamm::__private::__global::settle::h5dcd1b0add718cbf [programs/twamm/src/lib.rs:37]
>>>      twamm::twamm::settle::h46cd4315f58a6f44 [programs/twamm/src/lib.rs:37]
>>>        twamm::instructions::settle::settle::hcd37cff6f01e4594 [programs/twamm/src/lib.rs:152]
>>>          twamm::state::token_pair::TokenPair::settle_pools::hdaff4210dcccf2ac [programs/twamm/src/instructions/settle.rs:152]

==========This arithmetic operation may be UNSAFE!==========
Found a potential vulnerability at line 519, column 17 in programs/twamm/src/state/token_pair.rs
The sub operation may result in underflows:

 513|                    &mut pool.buy_side,
 514|                    outstanding_sell,
 515|                    outstanding_buy,
 516|                    oracle_price,
 517|                )?;
```

AUTOMATED TESTING

```
 517|                )?;
 518|                outstanding_a[idx] -= settled_a;
>519|                outstanding_b[idx] -= settled_b;
 520|            }
 521|        }
 522|
 523|        if total_outstanding_a == 0 && total_outstanding_b == 0 {
 524|            return Ok(res);
 525|        }
>>>Stack Trace:
>>>twamm::try_entry::hc06acca7e89bade9 [programs/twamm/src/lib.rs:37]
>>>  twamm::dispatch::hf7aa41276ed49a8e [programs/twamm/src/lib.rs:37]
>>>    twamm::__private::__global::settle::h5dcd1b0add718cbf [programs/twamm/src/lib.rs:37]
>>>      twamm::twamm::settle::h46cd4315f58a6f44 [programs/twamm/src/lib.rs:37]
>>>        twamm::instructions::settle::settle::hcd37cff6f01e4594 [programs/twamm/src/lib.rs:152]
>>>          twamm::state::token_pair::TokenPair::settle_pools::hdaff4210dcccf2ac [programs/twamm/src/instructions/settle.rs:152]

==========This arithmetic operation may be UNSAFE!==========
Found a potential vulnerability at line 539, column 25 in programs/twamm/src/state/token_pair.rs
The sub operation may result in underflows:

 533|                            &mut pools[idx].sell_side,
 534|                            &mut pools[other_idx].buy_side,
 535|                            outstanding_a[idx],
 536|                            outstanding_b[other_idx],
 537|                            oracle_price,
 538|                        )?;
>539|                        outstanding_a[idx] -= settled_a;
 540|                        outstanding_b[other_idx] -= settled_b;
 541|                    }
 542|                    if outstanding_b[idx] != 0 && outstanding_a[other_idx] != 0 {
 543|                        let (settled_a, settled_b) = self.settle_sides(
 544|                            &mut pools[other_idx].sell_side,
 545|                            &mut pools[idx].buy_side,
>>>Stack Trace:
>>>twamm::try_entry::hc06acca7e89bade9 [programs/twamm/src/lib.rs:37]
>>>  twamm::dispatch::hf7aa41276ed49a8e [programs/twamm/src/lib.rs:37]
>>>    twamm::__private::__global::settle::h5dcd1b0add718cbf [programs/twamm/src/lib.rs:37]
>>>      twamm::twamm::settle::h46cd4315f58a6f44 [programs/twamm/src/lib.rs:37]
>>>        twamm::instructions::settle::settle::hcd37cff6f01e4594 [programs/twamm/src/lib.rs:152]
>>>          twamm::state::token_pair::TokenPair::settle_pools::hdaff4210dcccf2ac [programs/twamm/src/instructions/settle.rs:152]

==========This arithmetic operation may be UNSAFE!==========
Found a potential vulnerability at line 540, column 25 in programs/twamm/src/state/token_pair.rs
The sub operation may result in underflows:

 534|                            &mut pools[other_idx].buy_side,
 535|                            outstanding_a[idx],
 536|                            outstanding_b[other_idx],
 537|                            oracle_price,
 538|                        )?;
 539|                        outstanding_a[idx] -= settled_a;
>540|                        outstanding_b[other_idx] -= settled_b;
 541|                    }
 542|                    if outstanding_b[idx] != 0 && outstanding_a[other_idx] != 0 {
 543|                        let (settled_a, settled_b) = self.settle_sides(
 544|                            &mut pools[other_idx].sell_side,
 545|                            &mut pools[idx].buy_side,
 546|                            outstanding_a[other_idx],
>>>Stack Trace:
>>>twamm::try_entry::hc06acca7e89bade9 [programs/twamm/src/lib.rs:37]
>>>  twamm::dispatch::hf7aa41276ed49a8e [programs/twamm/src/lib.rs:37]
>>>    twamm::__private::__global::settle::h5dcd1b0add718cbf [programs/twamm/src/lib.rs:37]
>>>      twamm::twamm::settle::h46cd4315f58a6f44 [programs/twamm/src/lib.rs:37]
>>>        twamm::instructions::settle::settle::hcd37cff6f01e4594 [programs/twamm/src/lib.rs:152]
>>>          twamm::state::token_pair::TokenPair::settle_pools::hdaff4210dcccf2ac [programs/twamm/src/instructions/settle.rs:152]

==========This arithmetic operation may be UNSAFE!==========
Found a potential vulnerability at line 550, column 25 in programs/twamm/src/state/token_pair.rs
The sub operation may result in underflows:

 544|                            &mut pools[other_idx].sell_side,
 545|                            &mut pools[idx].buy_side,
 546|                            outstanding_a[other_idx],
```

AUTOMATED TESTING

```
 546|                            outstanding_a[other_idx],
 547|                            outstanding_b[idx],
 548|                            oracle_price,
 549|                        )?;
>550|                        outstanding_a[other_idx] -= settled_a;
 551|                        outstanding_b[idx] -= settled_b;
 552|                    }
 553|                }
 554|            }
 555|        }
 556|
>>>Stack Trace:
>>>twamm::try_entry::hc06acca7e89bade9 [programs/twamm/src/lib.rs:37]
>>>  twamm::dispatch::hf7aa41276ed49a8e [programs/twamm/src/lib.rs:37]
>>>    twamm::__private::__global::settle::h5dcd1b0add718cbf [programs/twamm/src/lib.rs:37]
>>>      twamm::twamm::settle::h46cd4315f58a6f44 [programs/twamm/src/lib.rs:37]
>>>        twamm::instructions::settle::settle::hcd37cff6f01e4594 [programs/twamm/src/lib.rs:152]
>>>          twamm::state::token_pair::TokenPair::settle_pools::hdaff4210dcccf2ac [programs/twamm/src/instructions/settle.rs:152]

==========This arithmetic operation may be UNSAFE!==========
Found a potential vulnerability at line 551, column 25 in programs/twamm/src/state/token_pair.rs
The sub operation may result in underflows:

 545|                            &mut pools[idx].buy_side,
 546|                            outstanding_a[other_idx],
 547|                            outstanding_b[idx],
 548|                            oracle_price,
 549|                        )?;
 550|                        outstanding_a[other_idx] -= settled_a;
>551|                        outstanding_b[idx] -= settled_b;
 552|                    }
 553|                }
 554|            }
 555|        }
 556|
 557|        // compute net amounts
>>>Stack Trace:
>>>twamm::try_entry::hc06acca7e89bade9 [programs/twamm/src/lib.rs:37]
>>>  twamm::dispatch::hf7aa41276ed49a8e [programs/twamm/src/lib.rs:37]
>>>    twamm::__private::__global::settle::h5dcd1b0add718cbf [programs/twamm/src/lib.rs:37]
>>>      twamm::twamm::settle::h46cd4315f58a6f44 [programs/twamm/src/lib.rs:37]
>>>        twamm::instructions::settle::settle::hcd37cff6f01e4594 [programs/twamm/src/lib.rs:152]
>>>          twamm::state::token_pair::TokenPair::settle_pools::hdaff4210dcccf2ac [programs/twamm/src/instructions/settle.rs:152]

 - ✔ [00m:01s] Building Static Happens-Before Graph
 - ✔ [00m:00s] Detecting Vulnerabilities
detected 0 untrustful accounts in total.
detected 10 unsafe math operations in total.

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

       10 unsafe arithmetic issues
```

6.2 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-2023-0001	tokio	reject_remote_clients Configuration corruption
RUSTSEC-2021-0139	ansi term	Unmaintained

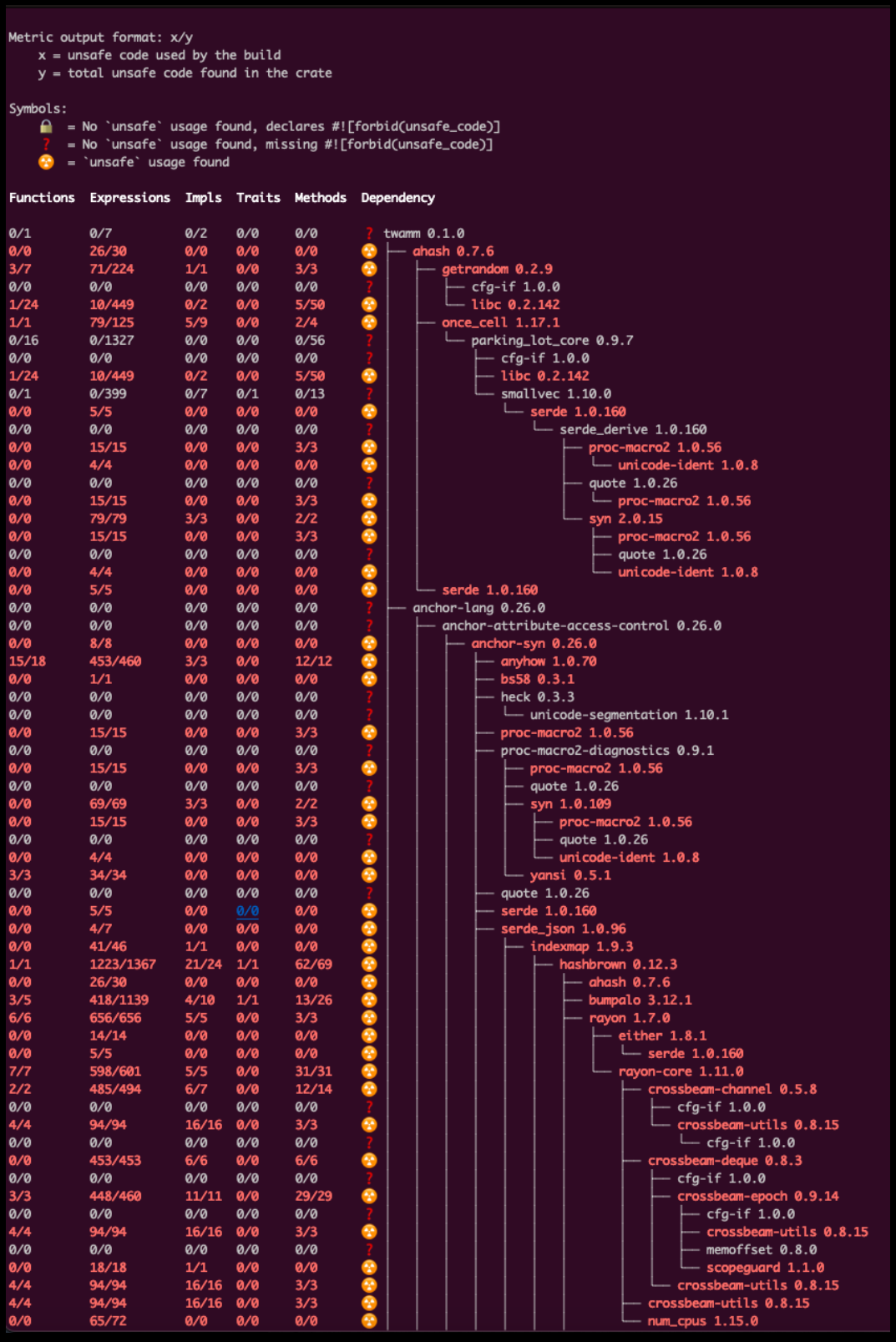
6.3 UNSAFE RUST CODE DETECTION

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.

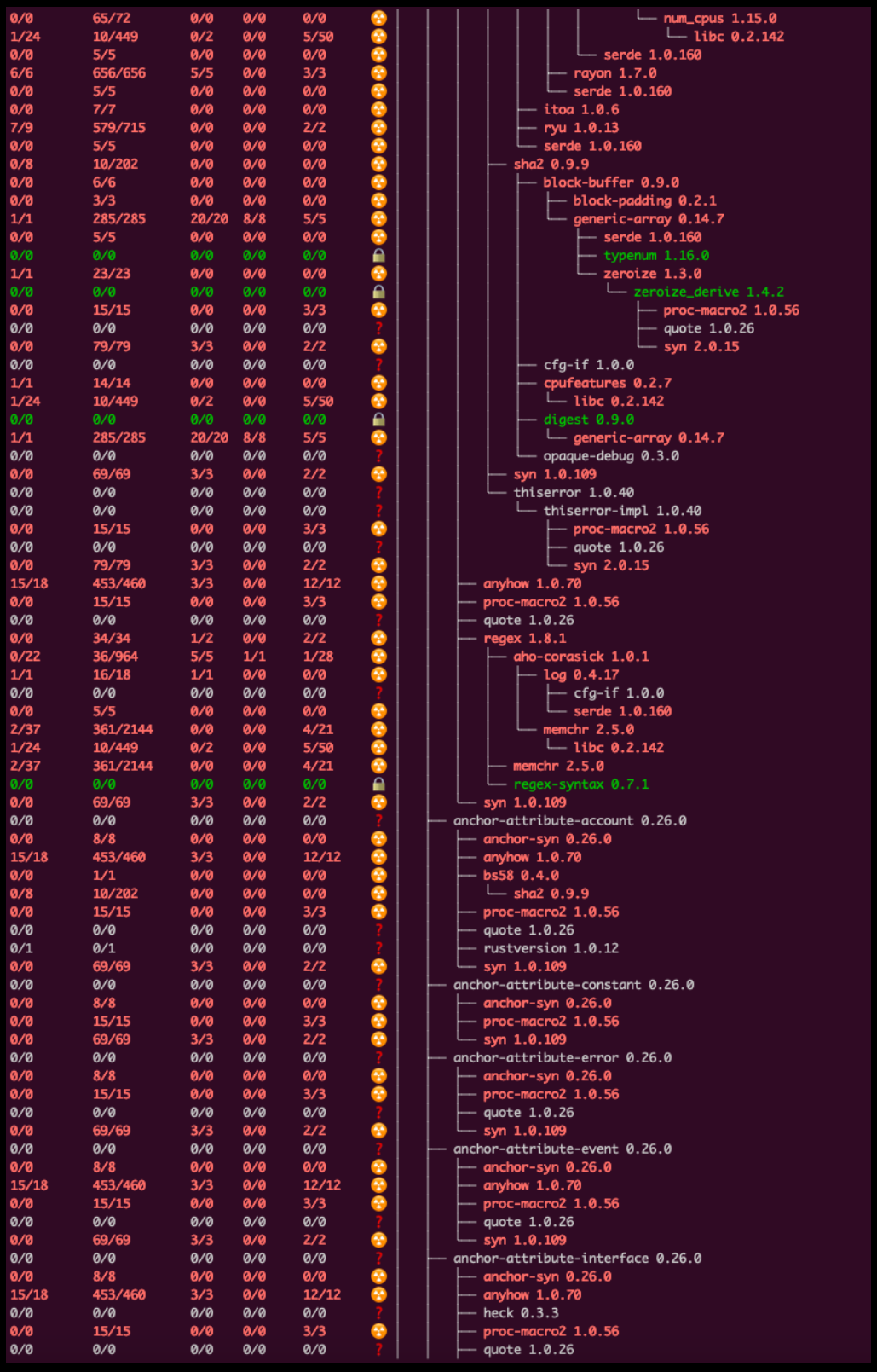
Results:

AUTOMATED TESTING



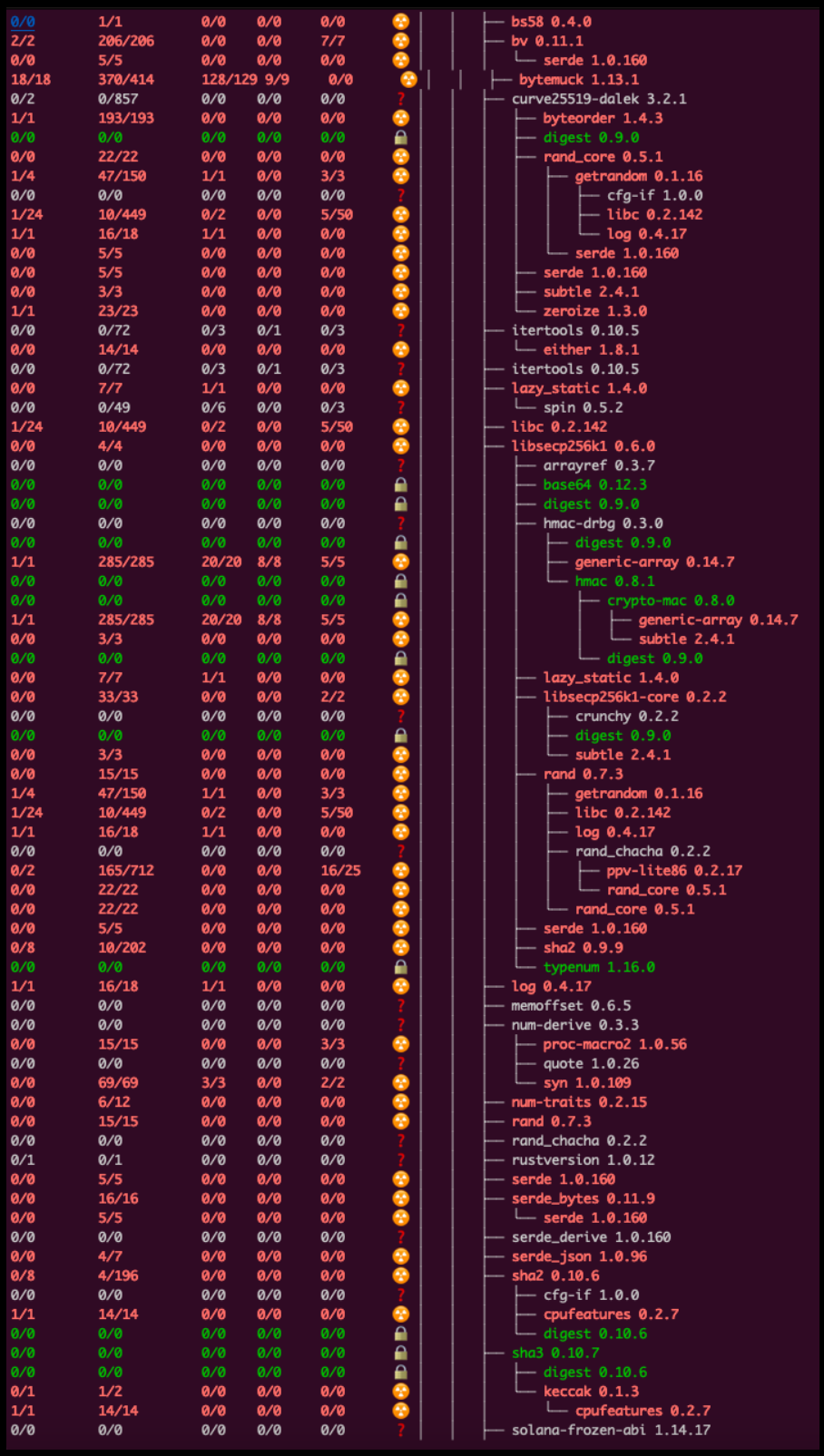
```
Metric output format: x/y
    x = unsafe code used by the build
    y = total unsafe code found in the crate

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

0/1        0/7          0/2    0/0     0/0      ❓ twamm 0.1.0
0/0        26/30        0/0    0/0     0/0      ☢️ ├── ahash 0.7.6
3/7        71/224       1/1    0/0     3/3      ☢️ │   ├── getrandom 0.2.9
0/0        0/0          0/0    0/0     0/0      ❓ │   │   ├── cfg-if 1.0.0
1/24       10/449       0/2    0/0     5/50     ☢️ │   │   └── libc 0.2.142
1/1        79/125       5/9    0/0     2/4      ☢️ │   ├── once_cell 1.17.1
0/16       0/1327       0/0    0/0     0/56     ❓ │   │   └── parking_lot_core 0.9.7
0/0        0/0          0/0    0/0     0/0      ❓ │   │       ├── cfg-if 1.0.0
1/24       10/449       0/2    0/0     5/50     ☢️ │   │       ├── libc 0.2.142
0/1        0/399        0/7    0/1     0/13     ❓ │   │       └── smallvec 1.10.0
0/0        5/5          0/0    0/0     0/0      ☢️ │   │           └── serde 1.0.160
0/0        0/0          0/0    0/0     0/0      ❓ │   │               └── serde_derive 1.0.160
0/0        15/15        0/0    0/0     3/3      ☢️ │   │                   ├── proc-macro2 1.0.56
0/0        4/4          0/0    0/0     0/0      ☢️ │   │                   │   └── unicode-ident 1.0.8
0/0        0/0          0/0    0/0     0/0      ❓ │   │                   ├── quote 1.0.26
0/0        15/15        0/0    0/0     3/3      ☢️ │   │                   │   └── proc-macro2 1.0.56
0/0        79/79        3/3    0/0     2/2      ☢️ │   │                   └── syn 2.0.15
0/0        15/15        0/0    0/0     3/3      ☢️ │   │                       ├── proc-macro2 1.0.56
0/0        0/0          0/0    0/0     0/0      ❓ │   │                       ├── quote 1.0.26
0/0        4/4          0/0    0/0     0/0      ☢️ │   │                       └── unicode-ident 1.0.8
0/0        5/5          0/0    0/0     0/0      ☢️ │   └── serde 1.0.160
0/0        0/0          0/0    0/0     0/0      ❓ ├── anchor-lang 0.26.0
0/0        0/0          0/0    0/0     0/0      ❓ │   ├── anchor-attribute-access-control 0.26.0
0/0        8/8          0/0    0/0     0/0      ☢️ │   │   ├── anchor-syn 0.26.0
15/18      453/460      3/3    0/0     12/12    ☢️ │   │   │   ├── anyhow 1.0.70
0/0        1/1          0/0    0/0     0/0      ☢️ │   │   │   ├── bs58 0.3.1
0/0        0/0          0/0    0/0     0/0      ❓ │   │   │   ├── heck 0.3.3
0/0        0/0          0/0    0/0     0/0      ❓ │   │   │   │   └── unicode-segmentation 1.10.1
0/0        15/15        0/0    0/0     3/3      ☢️ │   │   │   ├── proc-macro2 1.0.56
0/0        0/0          0/0    0/0     0/0      ❓ │   │   │   ├── proc-macro2-diagnostics 0.9.1
0/0        15/15        0/0    0/0     3/3      ☢️ │   │   │   │   ├── proc-macro2 1.0.56
0/0        0/0          0/0    0/0     0/0      ❓ │   │   │   │   ├── quote 1.0.26
0/0        69/69        3/3    0/0     2/2      ☢️ │   │   │   │   ├── syn 1.0.109
0/0        15/15        0/0    0/0     3/3      ☢️ │   │   │   │   │   ├── proc-macro2 1.0.56
0/0        0/0          0/0    0/0     0/0      ❓ │   │   │   │   │   ├── quote 1.0.26
0/0        4/4          0/0    0/0     0/0      ☢️ │   │   │   │   │   └── unicode-ident 1.0.8
3/3        34/34        0/0    0/0     0/0      ☢️ │   │   │   │   └── yansi 0.5.1
0/0        0/0          0/0    0/0     0/0      ❓ │   │   │   ├── quote 1.0.26
0/0        5/5          0/0    0/0     0/0      ☢️ │   │   │   ├── serde 1.0.160
0/0        4/7          0/0    0/0     0/0      ☢️ │   │   │   ├── serde_json 1.0.96
0/0        41/46        1/1    0/0     0/0      ☢️ │   │   │   │   ├── indexmap 1.9.3
1/1        1223/1367    21/24  1/1     62/69    ☢️ │   │   │   │   │   ├── hashbrown 0.12.3
0/0        26/30        0/0    0/0     0/0      ☢️ │   │   │   │   │   │   ├── ahash 0.7.6
3/5        418/1139     4/10   1/1     13/26    ☢️ │   │   │   │   │   │   ├── bumpalo 3.12.1
6/6        656/656      5/5    0/0     3/3      ☢️ │   │   │   │   │   │   ├── rayon 1.7.0
0/0        14/14        0/0    0/0     0/0      ☢️ │   │   │   │   │   │   │   ├── either 1.8.1
0/0        5/5          0/0    0/0     0/0      ☢️ │   │   │   │   │   │   │   │   └── serde 1.0.160
7/7        598/601      5/5    0/0     31/31    ☢️ │   │   │   │   │   │   │   └── rayon-core 1.11.0
2/2        485/494      6/7    0/0     12/14    ☢️ │   │   │   │   │   │   │       ├── crossbeam-channel 0.5.8
0/0        0/0          0/0    0/0     0/0      ❓ │   │   │   │   │   │   │       │   ├── cfg-if 1.0.0
4/4        94/94        16/16  0/0     3/3      ☢️ │   │   │   │   │   │   │       │   └── crossbeam-utils 0.8.15
0/0        0/0          0/0    0/0     0/0      ❓ │   │   │   │   │   │   │       │       └── cfg-if 1.0.0
0/0        453/453      6/6    0/0     6/6      ☢️ │   │   │   │   │   │   │       ├── crossbeam-deque 0.8.3
0/0        0/0          0/0    0/0     0/0      ❓ │   │   │   │   │   │   │       │   ├── cfg-if 1.0.0
3/3        448/460      11/11  0/0     29/29    ☢️ │   │   │   │   │   │   │       │   ├── crossbeam-epoch 0.9.14
0/0        0/0          0/0    0/0     0/0      ❓ │   │   │   │   │   │   │       │   │   ├── cfg-if 1.0.0
4/4        94/94        16/16  0/0     3/3      ☢️ │   │   │   │   │   │   │       │   │   ├── crossbeam-utils 0.8.15
0/0        0/0          0/0    0/0     0/0      ❓ │   │   │   │   │   │   │       │   │   ├── memoffset 0.8.0
0/0        18/18        1/1    0/0     0/0      ☢️ │   │   │   │   │   │   │       │   │   └── scopeguard 1.1.0
4/4        94/94        16/16  0/0     3/3      ☢️ │   │   │   │   │   │   │       │   └── crossbeam-utils 0.8.15
4/4        94/94        16/16  0/0     3/3      ☢️ │   │   │   │   │   │   │       ├── crossbeam-utils 0.8.15
0/0        65/72        0/0    0/0     0/0      ☢️ │   │   │   │   │   │   │       └── num_cpus 1.15.0
```

AUTOMATED TESTING

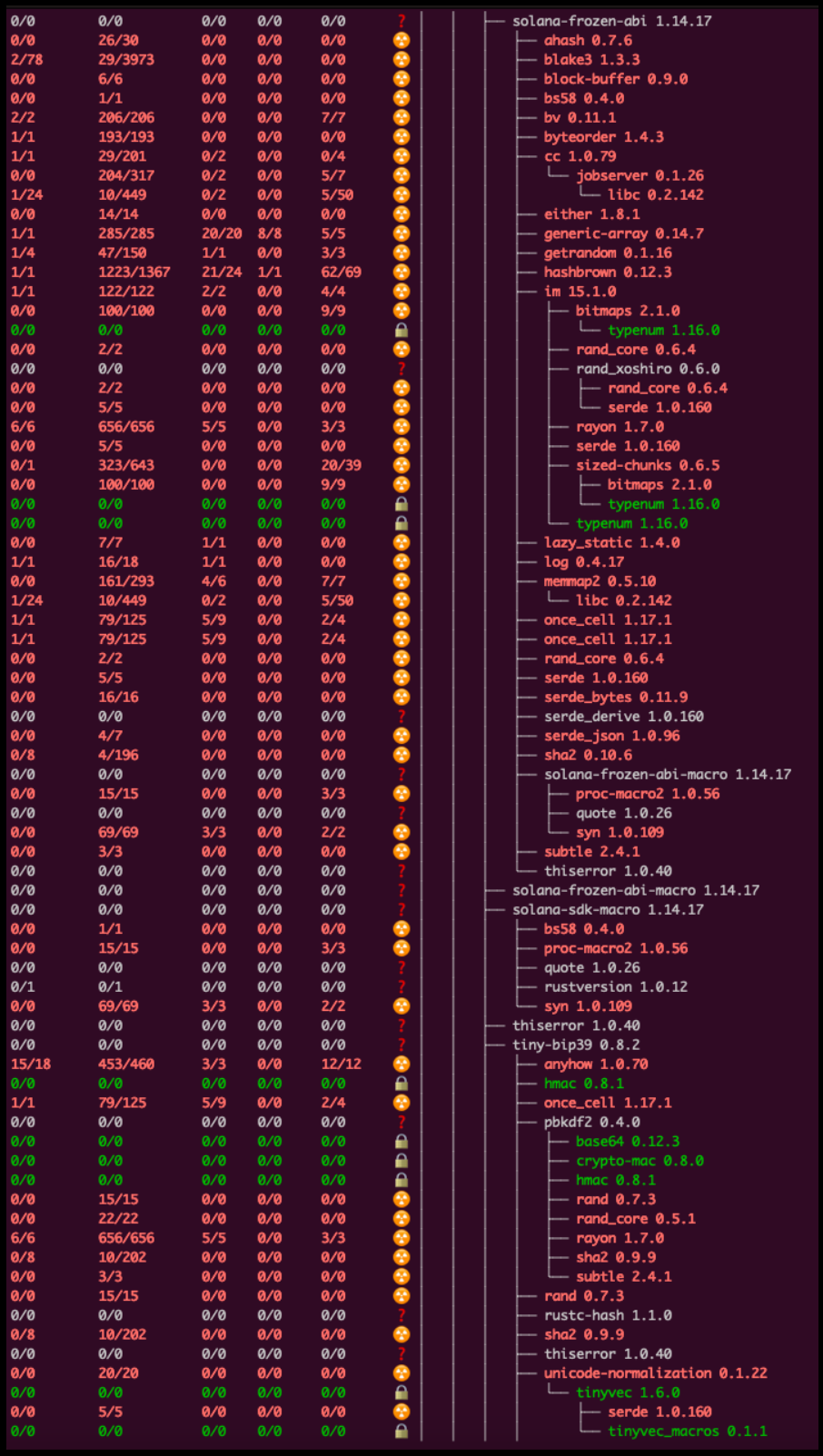


```
0/0    0/0        0/0     0/0  0/0    ?  │  │   ├── quote 1.0.26
0/0    69/69      3/3     0/0  2/2    ☢  │  │   └── syn 1.0.109
0/0    0/0        0/0     0/0  0/0    ?  │  ├── anchor-attribute-program 0.26.0
0/0    8/8        0/0     0/0  0/0    ☢  │  │   ├── anchor-syn 0.26.0
15/18  453/460    3/3     0/0  12/12  ☢  │  │   ├── anyhow 1.0.70
0/0    15/15      0/0     0/0  3/3    ☢  │  │   ├── proc-macro2 1.0.56
0/0    0/0        0/0     0/0  0/0    ?  │  │   ├── quote 1.0.26
0/0    69/69      3/3     0/0  2/2    ☢  │  │   └── syn 1.0.109
0/0    0/0        0/0     0/0  0/0    ?  │  ├── anchor-attribute-state 0.26.0
0/0    8/8        0/0     0/0  0/0    ☢  │  │   ├── anchor-syn 0.26.0
15/18  453/460    3/3     0/0  12/12  ☢  │  │   ├── anyhow 1.0.70
0/0    15/15      0/0     0/0  3/3    ☢  │  │   ├── proc-macro2 1.0.56
0/0    0/0        0/0     0/0  0/0    ?  │  │   ├── quote 1.0.26
0/0    69/69      3/3     0/0  2/2    ☢  │  │   └── syn 1.0.109
0/0    0/0        0/0     0/0  0/0    ?  │  ├── anchor-derive-accounts 0.26.0
0/0    8/8        0/0     0/0  0/0    ☢  │  │   ├── anchor-syn 0.26.0
15/18  453/460    3/3     0/0  12/12  ☢  │  │   ├── anyhow 1.0.70
0/0    15/15      0/0     0/0  3/3    ☢  │  │   ├── proc-macro2 1.0.56
0/0    0/0        0/0     0/0  0/0    ?  │  │   ├── quote 1.0.26
0/0    69/69      3/3     0/0  2/2    ☢  │  │   └── syn 1.0.109
0/0    0/0        0/0     0/0  0/0    ?  │  ├── arrayref 0.3.7
0/0    0/0        0/0     0/0  0/0    🔒  │  ├── base64 0.13.1
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 1.0.160
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    15/15      0/0     0/0  3/3    ☢  │  │   │   │   ├── proc-macro2 1.0.56
0/0    0/0        0/0     0/0  0/0    ?  │  │   │   │   ├── quote 1.0.26
0/0    69/69      3/3     0/0  2/2    ☢  │  │   │   │   └── syn 1.0.109
0/0    0/0        0/0     0/0  0/0    ?  │  │   │   ├── borsh-schema-derive-internal 0.9.3
0/0    15/15      0/0     0/0  3/3    ☢  │  │   │   │   ├── proc-macro2 1.0.56
0/0    0/0        0/0     0/0  0/0    ?  │  │   │   │   ├── quote 1.0.26
0/0    69/69      3/3     0/0  2/2    ☢  │  │   │   │   └── syn 1.0.109
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 0.5.11
0/0    41/46      1/1     0/0  0/0    ☢  │  │   │   │       ├── indexmap 1.9.3
0/0    5/5        0/0     0/0  0/0    ☢  │  │   │   │       └── serde 1.0.160
0/0    15/15      0/0     0/0  3/3    ☢  │  │   │   ├── proc-macro2 1.0.56
0/0    69/69      3/3     0/0  2/2    ☢  │  │   │   └── syn 1.0.109
2/2    1064/1198  19/22   1/1  51/58  ☢  │  │   └── hashbrown 0.11.2
0/0    26/30      0/0     0/0  0/0    ☢  │  │       ├── ahash 0.7.6
3/5    418/1139   4/10    1/1  13/26  ☢  │  │       ├── bumpalo 3.12.1
6/6    656/656    5/5     0/0  3/3    ☢  │  │       ├── rayon 1.7.0
0/0    5/5        0/0     0/0  0/0    ☢  │  │       └── serde 1.0.160
18/18  370/414    128/129 9/9  0/0    ☢  │  ├── bytemuck 1.13.1
0/0    0/0        0/1     0/0  0/0    ?  │  │   └── bytemuck_derive 1.4.1
0/0    15/15      0/0     0/0  3/3    ☢  │  │       ├── proc-macro2 1.0.56
0/0    0/0        0/0     0/0  0/0    ?  │  │       ├── quote 1.0.26
0/0    79/79      3/3     0/0  2/2    ☢  │  │       └── syn 2.0.15
3/3    442/442    1/1     0/0  3/3    ☢  │  ├── solana-program 1.14.17
0/0    0/0        0/0     0/0  0/0    🔒  │  │   ├── base64 0.13.1
0/0    22/22      0/0     0/0  0/0    ☢  │  │   ├── bincode 1.3.3
0/0    0/0        0/0     0/0  0/0    ?  │  │   ├── bitflags 1.3.2
2/78   29/3973    0/0     0/0  0/0    ☢  │  │   ├── blake3 1.3.3
0/0    0/0        0/0     0/0  0/0    ?  │  │   │   ├── arrayref 0.3.7
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 1.0.160
0/0    0/0        0/0     0/0  0/0    ?  │  │   │   ├── cfg-if 1.0.0
0/0    4/4        0/0     0/0  0/0    ☢  │  │   │   ├── constant_time_eq 0.2.5
0/0    0/0        0/0     0/0  0/0    🔒  │  │   │   ├── digest 0.10.6
0/0    16/16      0/0     0/0  0/0    ☢  │  │   │   │   ├── block-buffer 0.10.4
1/1    285/285    20/20   8/8  5/5    ☢  │  │   │   │   │   └── generic-array 0.14.7
0/0    0/0        0/0     0/0  0/0    🔒  │  │   │   │   ├── crypto-common 0.1.6
1/1    285/285    20/20   8/8  5/5    ☢  │  │   │   │   │   ├── generic-array 0.14.7
0/0    2/2        0/0     0/0  0/0    ☢  │  │   │   │   │   ├── rand_core 0.6.4
3/7    71/224     1/1     0/0  3/3    ☢  │  │   │   │   │   │   ├── getrandom 0.2.9
0/0    5/5        0/0     0/0  0/0    ☢  │  │   │   │   │   │   └── serde 1.0.160
0/0    0/0        0/0     0/0  0/0    🔒  │  │   │   │   │   └── typenum 1.16.0
0/0    3/3        0/0     0/0  0/0    ☢  │  │   │   │   └── subtle 2.4.1
6/6    656/656    5/5     0/0  3/3    ☢  │  │   │   └── rayon 1.7.0
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    1/1        0/0     0/0  0/0    ☢  │  │   ├── bs58 0.4.0
```

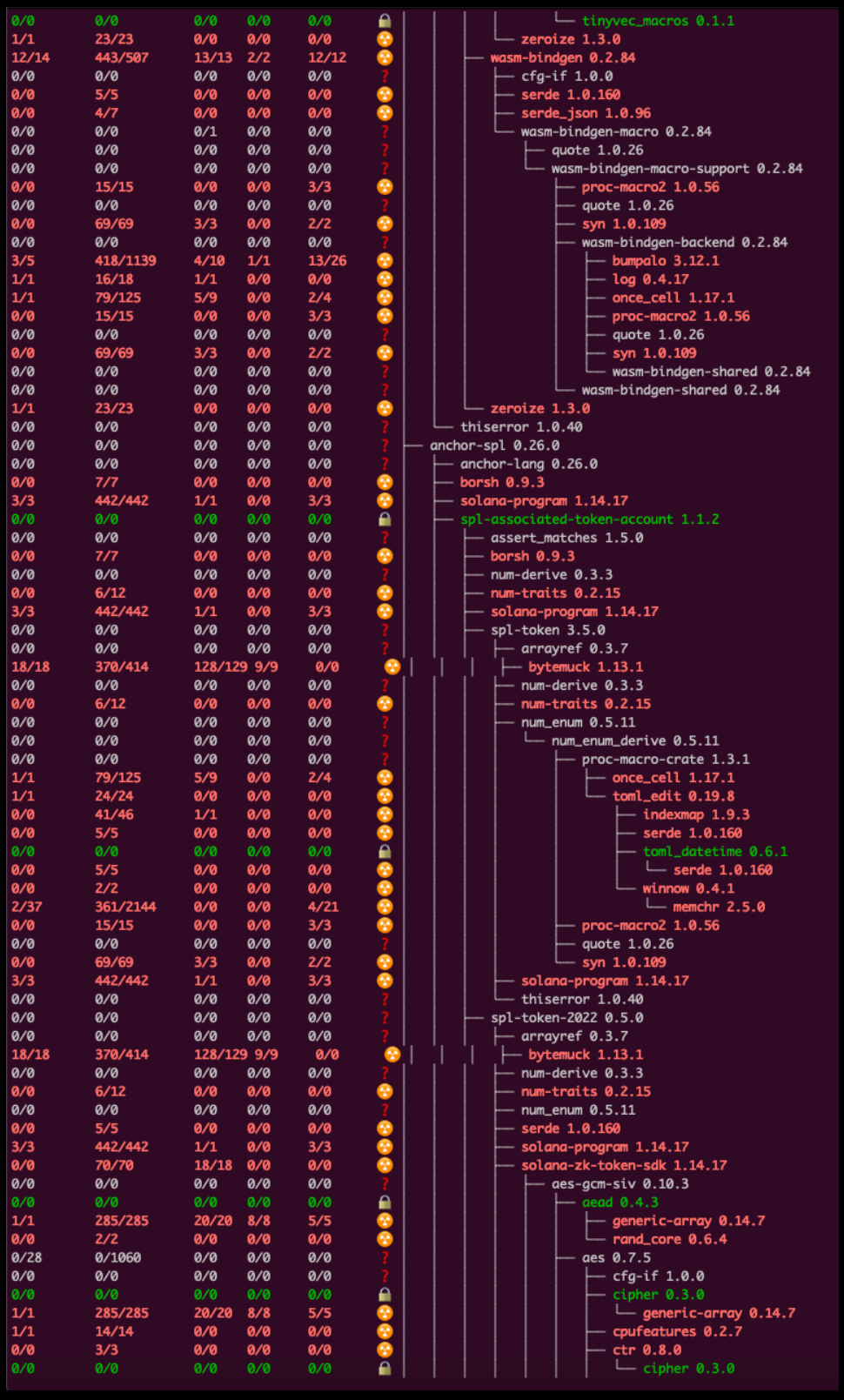
AUTOMATED TESTING



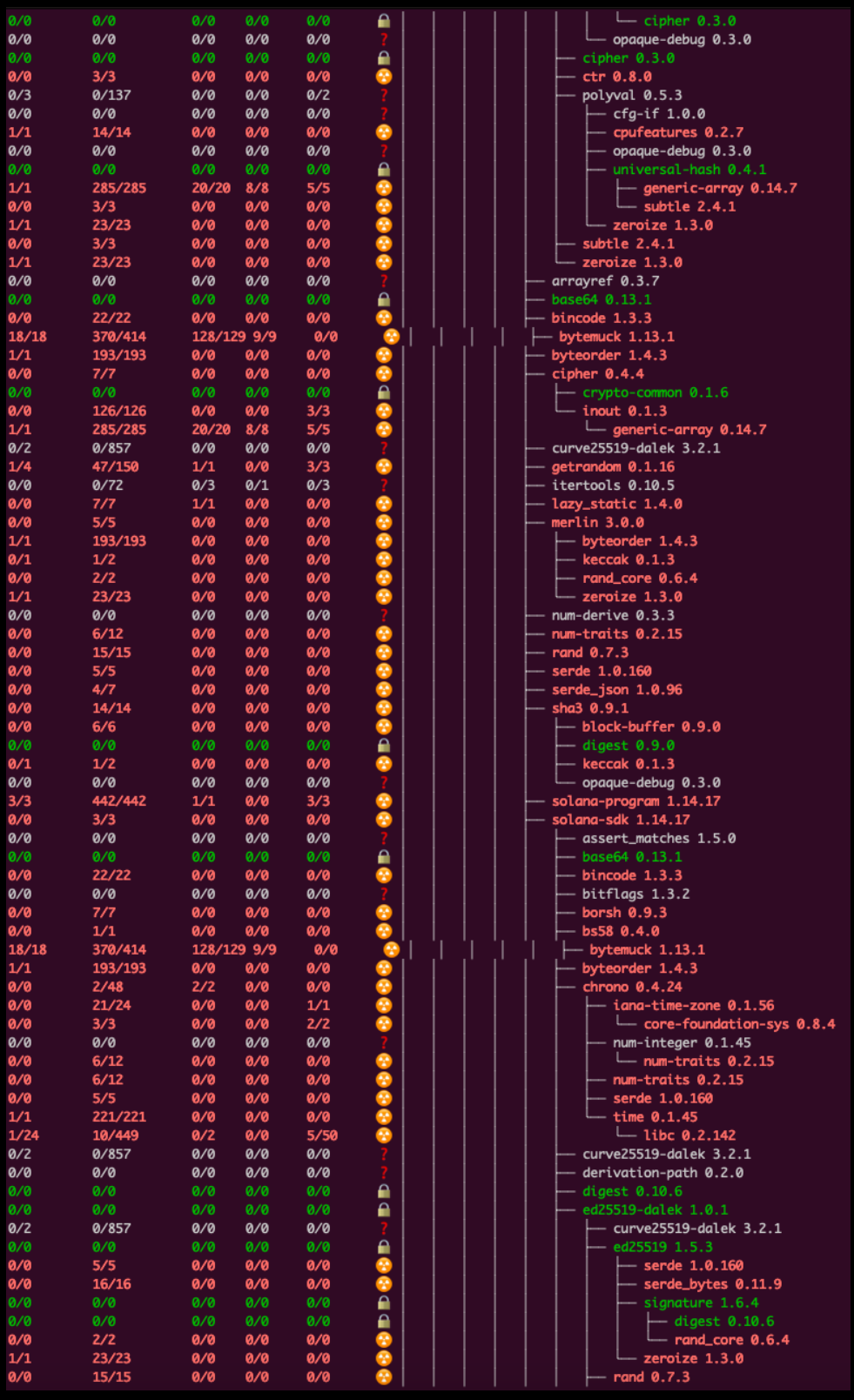
AUTOMATED TESTING



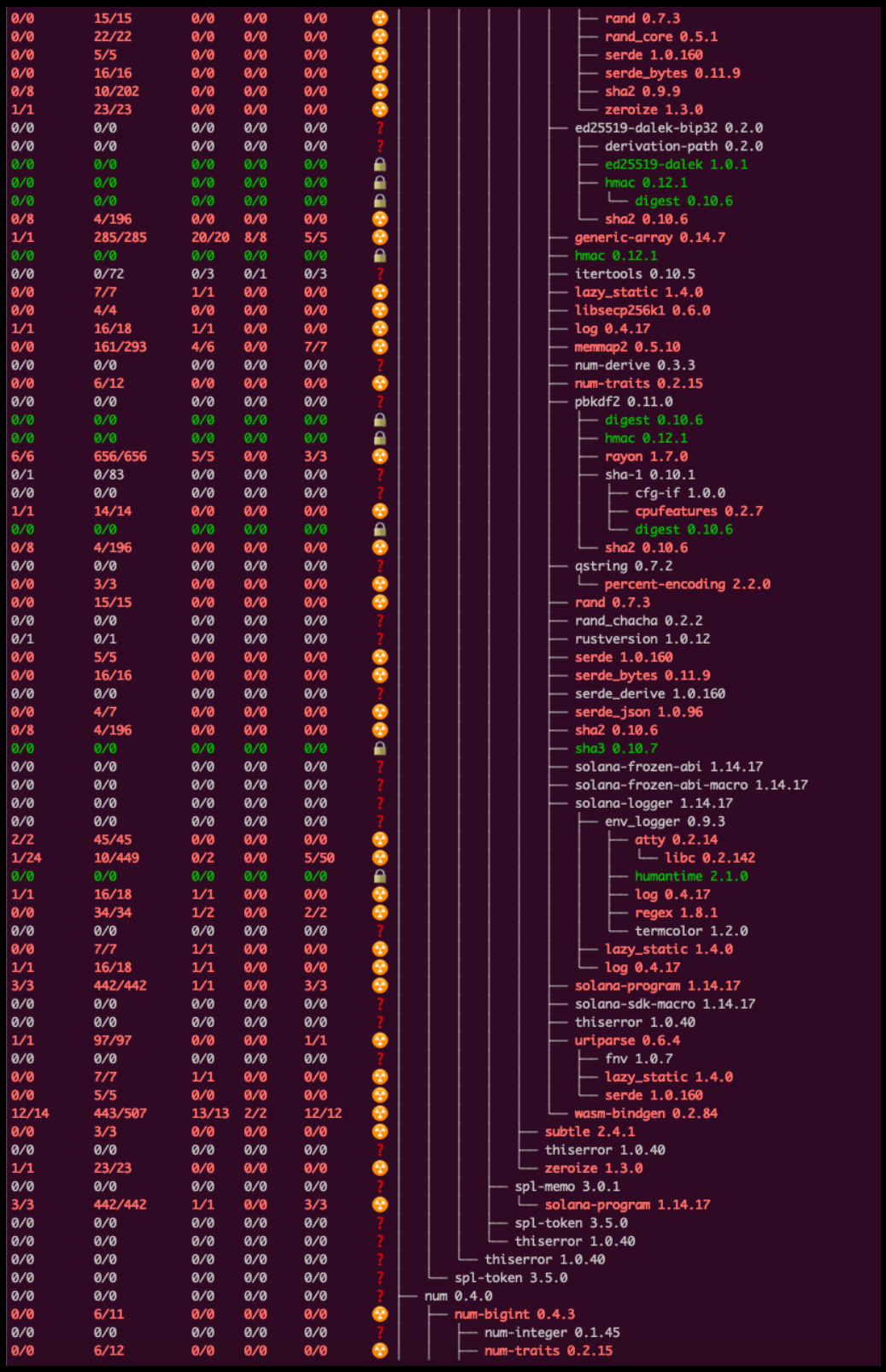
AUTOMATED TESTING



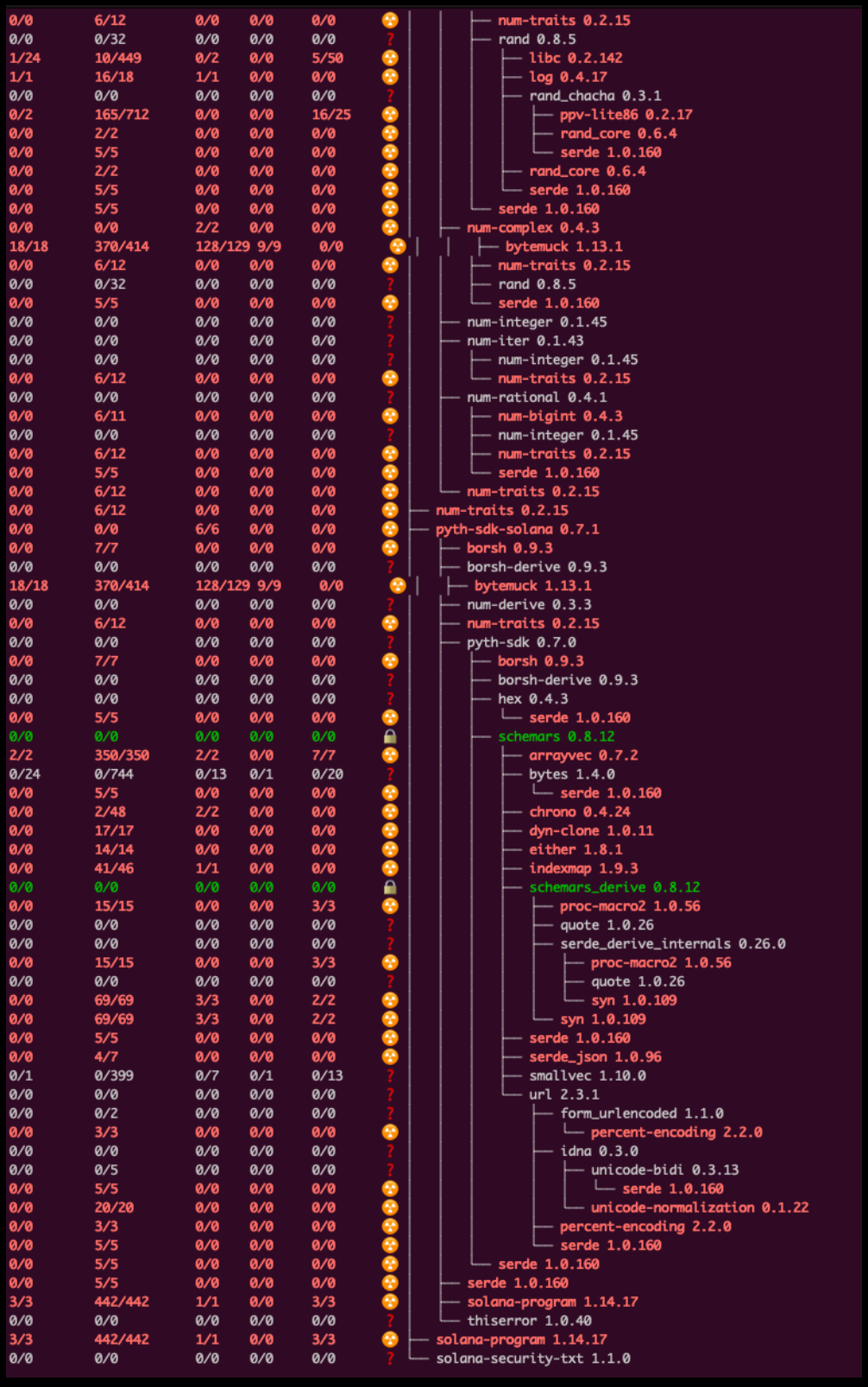
AUTOMATED TESTING

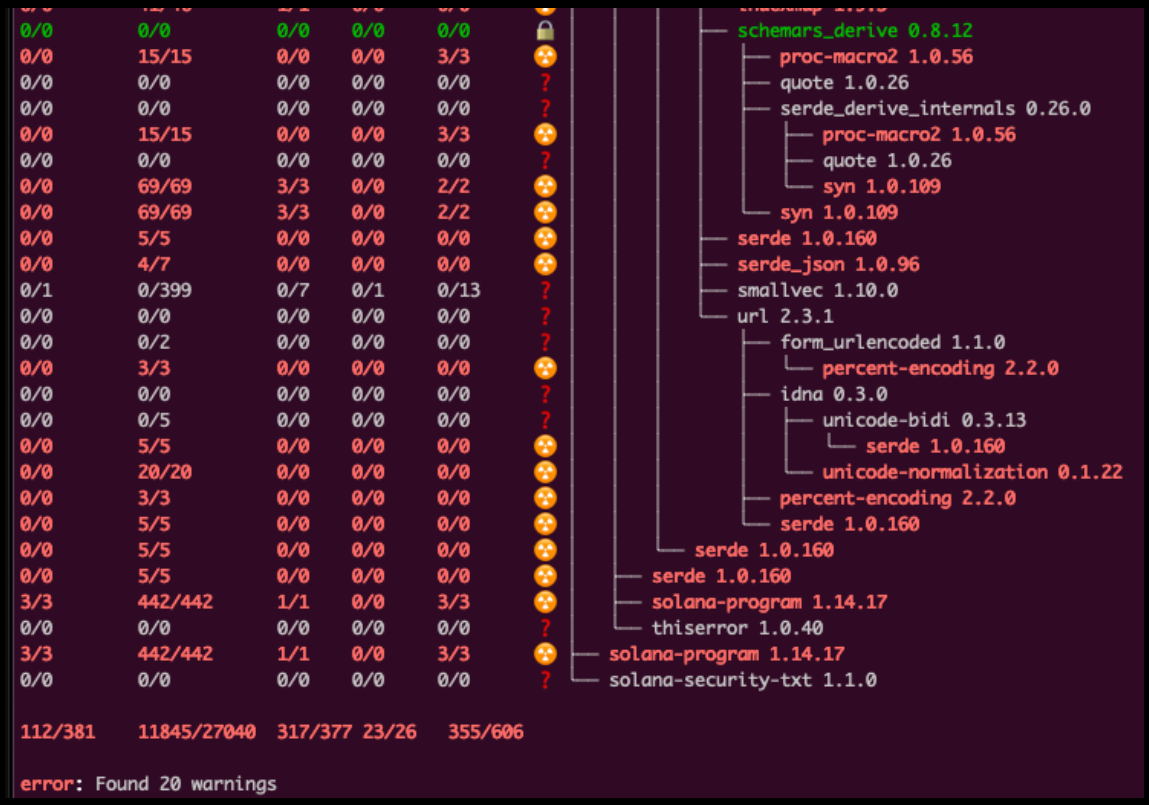


AUTOMATED TESTING



AUTOMATED TESTING




0/0 0/0 0/0 0/0 0/0
0/0 15/15 0/0 0/0 3/3
0/0 0/0 0/0 0/0 0/0
0/0 0/0 0/0 0/0 0/0
0/0 15/15 0/0 0/0 3/3
0/0 0/0 0/0 0/0 0/0
0/0 69/69 3/3 0/0 2/2
0/0 69/69 3/3 0/0 2/2
0/0 5/5 0/0 0/0 0/0
0/0 4/7 0/0 0/0 0/0
0/1 0/399 0/7 0/1 0/13
0/0 0/0 0/0 0/0 0/0
0/0 0/2 0/0 0/0 0/0
0/0 3/3 0/0 0/0 0/0
0/0 0/0 0/0 0/0 0/0
0/0 0/5 0/0 0/0 0/0
0/0 5/5 0/0 0/0 0/0
0/0 20/20 0/0 0/0 0/0
0/0 3/3 0/0 0/0 0/0
0/0 5/5 0/0 0/0 0/0
0/0 5/5 0/0 0/0 0/0
0/0 5/5 0/0 0/0 0/0
3/3 442/442 1/1 0/0 3/3
0/0 0/0 0/0 0/0 0/0
3/3 442/442 1/1 0/0 3/3
0/0 0/0 0/0 0/0 0/0
112/381 11845/27040 317/377 23/26 355/606
error: Found 20 warnings
schemars_derive 0.8.12
proc-macro2 1.0.56
quote 1.0.26
serde_derive_internals 0.26.0
proc-macro2 1.0.56
quote 1.0.26
syn 1.0.109
syn 1.0.109
serde 1.0.160
serde_json 1.0.96
smallvec 1.10.0
url 2.3.1
form_urlencoded 1.1.0
percent-encoding 2.2.0
idna 0.3.0
unicode-bidi 0.3.13
serde 1.0.160
unicode-normalization 0.1.22
percent-encoding 2.2.0
serde 1.0.160
serde 1.0.160
serde 1.0.160
solana-program 1.14.17
thiserror 1.0.40
solana-program 1.14.17
solana-security-txt 1.1.0

THANK YOU FOR CHOOSING

// HALBORN