Neon Labs - EVM

Solana Program Security Audit

Prepared by: **Halborn**

Date of Engagement: **June 1st, 2022 - July 20th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	6/1/2022	Piotr Cielas
0.2	Final Draft	7/20/2022	Piotr Cielas
0.3	Draft Review	7/20/2022	Gabi Urrutia
1.0	Remediation Plan	9/19/2022	Mateusz Garncarek
1.1	Remediation Plan Review	9/21/2022	Piotr Cielas
1.2	Remediation Plan Review	9/21/2022	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
Piotr Cielas	Halborn	Piotr.Cielas@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Neon Labs engaged Halborn to conduct a security audit on their program, beginning on June 1st, 2022 and ending on July 20th, 2022 .

Neon EVM is a tool that allows for Ethereum-like transactions to be processed on Solana, taking full advantage of the functionality native to Solana, including parallel execution of transactions. As such, Neon EVM allows dApps to operate with the low gas fees, high transaction speed, and high throughput of Solana, as well as offering access to the growing Solana market.

The security assessment was scoped to the programs provided in the neon-evm 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 seven weeks for the engagement and assigned a full-time security engineer to audit the security of the program in scope. The security engineer is a blockchain and smart contract security expert with advanced penetration testing and smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Identify potential security issues within the program

In summary, Halborn identified some security risks that were mostly addressed by Neon Labs .

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 smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts 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.
- Smart contract manual 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 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)

RISK METHODOLOGY:

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

RISK SCALE - LIKELIHOOD

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

RISK SCALE - IMPACT

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

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

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

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

1.4 SCOPE

Code repositories:

1. Neon EVM loader

- Repository: neon-evm
- Commit ID: 0659980fb03ee1adc8f89f8adc7fa19279d83496
- Programs in scope:

    1. evm-loader (evm_loader/program)

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

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	1	1	5	4

LIKELIHOOD

IMPACT

		(HAL-02)		(HAL-01)
(HAL-05) (HAL-08)		(HAL-03)		
	(HAL-06) (HAL-07)	(HAL-04)		
(HAL-09) (HAL-10) (HAL-11) (HAL-12)				

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
STEALING NEON TOKEN DEPOSITS	Critical	SOLVED - 09/16/2022
OVERWRITING HOLDER ACCOUNT DATA	High	SOLVED - 09/16/2022
OPERATOR ACCOUNTS CAN SELFDESTRUCT	Medium	SOLVED - 09/16/2022
INCORRECTLY GENERATED HOLDER SEED	Low	SOLVED - 09/16/2022
HARDCODED GOVERNANCE ADDRESSES	Low	RISK ACCEPTED
SUSCEPTIBLE TO INTEGER UNDERFLOW	Low	SOLVED - 09/16/2022
SUSCEPTIBLE TO ACCESS OUT-OF-BOUNDS	Low	RISK ACCEPTED
OPERATOR ETHEREUM NEON TOKEN ADDRESS NOT VALIDATED	Low	RISK ACCEPTED
DUPLICATE INSTRUCTION HANDLERS	Informational	ACKNOWLEDGED
DUPLICATE SECURITY CHECKS	Informational	ACKNOWLEDGED
REDUNDANT INSTRUCTION DATA	Informational	ACKNOWLEDGED
REDUNDANT INSTRUCTION ACCOUNTS	Informational	SOLVED - 09/16/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) STEALING NEON TOKEN DEPOSITS - CRITICAL

Description:

To compensate operators for processing NEON transactions, users pay fees in NEON tokens. Users deposit those tokens to the official vault with the Deposit instruction, providing some accounts, including the destination ETH account.

The instruction handler expects the source account owner to have delegated some NEON to the pool authority, and when invoked, it transfers the delegated amount to the pool.

Because neither the handler validates the transaction signer nor the destination ETH account is linked to the sender in any verifiable way, an anonymous user can deposit the entire delegated amount from any NEON token account to an arbitrary ETH account.

Code Location:

Listing 1: evm_loader/program/src/account/ether_account.rs (Line 38)

```
30 pub struct Data {
31     /// Ethereum address
32     pub address: H160,
33     /// Solana account nonce
34     pub bump_seed: u8,
35     /// Ethereum account nonce
36     pub trx_count: u64,
37     /// Neon token balance
38     pub balance: U256,
39     /// Address of solana account that stores code data (for contract accounts)
40     pub code_account: Option<Pubkey>,
41     /// Read-write lock
42     pub rw_blocked: bool,
43     /// Read-only lock counter
44     pub ro_blocked_count: u8,
```

```
45 }
```

Listing 2: evm_loader/program/src/instruction/account_create.rs (Lines 41-43)

```
36 fn validate(program_id: &Pubkey, accounts: &Accounts, address: &
   ↳ H160, bump_seed: u8) -> ProgramResult {
37     if !solana_program::system_program::check_id(accounts.
   ↳ ether_account.owner) {
38         return Err!(ProgramError::InvalidArgument; "Account {} -
   ↳ expected system owned", accounts.ether_account.key);
39     }
40
41     let program_seeds = [ &[ACCOUNT_SEED_VERSION], address.
   ↳ as_bytes()];
42     let (expected_address, expected_bump_seed) = Pubkey::
   ↳ find_program_address(&program_seeds, program_id);
43     if expected_address != *accounts.ether_account.key {
44         return Err!(ProgramError::InvalidArgument; "Account {} -
   ↳ expected PDA address {}", accounts.ether_account.key,
   ↳ expected_address);
45     }
```

Listing 3: evm_loader/program/src/instruction/neon_tokens_deposit.rs (Line 27)

```
24 struct Accounts<'a> {
25     source: token::State<'a>,
26     pool: token::State<'a>,
27     ethereum_account: EthereumAccount<'a>,
28     authority: &'a AccountInfo<'a>,
29     token_program: program::Token<'a>,
30 }
```

Listing 4: evm_loader/program/src/instruction/neon_tokens_deposit.rs (Line 58)

```
50 if accounts.source.mint != crate::config::token_mint::id() {
51         return Err!(ProgramError::InvalidArgument; "Account {} -
   ↳ expected Neon Token account", accounts.source.info.key);
52     }
```

```

    if accounts.pool.mint != crate::config::token_mint::id() {
        return Err!(ProgramError::InvalidArgument; "Account {} - expected Neon Token account", accounts.pool.info.key);
    }

    if !accounts.source.delegate.contains(accounts.authority.key) {
        return Err!(ProgramError::InvalidArgument; "Account {} - expected tokens delegated to authority account", accounts.source.info.key);
    }
```

Listing 5: evm_loader/program/src/instruction/neon_tokens_deposit.rs (Lines 69,73,81,83)

```
66 let amount = accounts.source.delegated_amount;
67
68 {
69     let signers_seeds: &[&[&[u8]]] = &[&[b"Deposit", &[bump_seed]]];
70
71     let instruction = spl_token::instruction::transfer(
72         accounts.token_program.key,
73         accounts.source.info.key,
74         accounts.pool.info.key,
75         accounts.authority.key,
76         &[],
77         amount
78     )?;
79
80     let account_infos: &[AccountInfo] = &[
81         accounts.source.info.clone(),
82         accounts.pool.info.clone(),
83         accounts.authority.clone(),
84         accounts.token_program.clone(),
85     ];
86
87     invoke_signed(&instruction, account_infos, signers_seeds)?;
```

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

On deposit, verify the signature of the source NEON token account owner.

Remediation Plan:

SOLVED: The Neon team solved this issue in commit
62ecbc30372651486ed352df10af73484c410e02
Now each user account has its ETH account as the delegate.

3.2 (HAL-02) OVERWRITING TRANSACTION DATA STORED IN HOLDER ACCOUNTS - HIGH

Description:

Holder accounts are used to store such EVM transactions that cannot be sent in one Solana instruction because they do not fit in one UDP datagram. Instead, such transactions are first uploaded in chunks and saved in Holder accounts and then those accounts are provided as EVM transaction data source to Neon EVM.

The evm-loader program introduces tags that identify account types. The first byte in the account's data space is parsed as tag. Instruction handlers verify those tags and accept or reject the account based on verification results. Uninitialized accounts are of the TAG_EMPTY type.

The WriteHolder instruction handler expects the operator to provide an empty Holder account to which it writes an Ethereum transaction. On each write, the account data is updated, but the first byte always stays 0. This means the programs always sees every Holder account as uninitialized, making it possible for them to be overwritten at any time, regardless of whether the original transaction has already been written in full or not.

Another scenario in which a Holder account may get overwritten mistakenly is when a new Storage account is initialized. The is_new_transaction defined in evm_loader/program/src/instruction/transaction.rs function checks the account tag and allows its caller to initialize a new Storage account if the tag is EMPTY, which it will be for every Holder account.

One more scenario in which a Holder account may get overwritten unintentionally is when a Contract account is resized.e The ResizeContractAccount instruction handler checks if the tag of the new contract account matches the empty tag, which it will for every Holder account, and if it does, it moves the contract to the new account, creating a new Contract account.

Code Location:

Listing 6: evm_loader/program/src/instruction/transaction_write_to_holder.rs (Line 25)

```
20 let data = &instruction[data_begin..data_end];
21
22 let operator = Operator::from_account(&accounts[1])?;
23 let mut holder = Holder::from_account(program_id, holder_id, &accounts[0], &operator)?;
24
25 holder.write(offset, data)
```

Listing 7: evm_loader/program/src/account/holder.rs (Line 32)

```
27 let expected_key = Pubkey::create_with_seed(operator.key, seed, program_id)?;
28 if *info.key != expected_key {
29     return Err!(ProgramError::InvalidArgument; "Account {} - expected holder key {}", info.key, expected_key);
30 }
31
32 Self::from_account_unchecked(program_id, info)
```

Listing 8: evm_loader/program/src/account/holder.rs (Line 36)

```
35 pub fn from_account_unchecked(program_id: &Pubkey, info: &'a AccountInfo<'a>) -> Result<Self, ProgramError> {
36     if account::tag(program_id, info)? != account::TAG_EMPTY {
37         return Err!(ProgramError::InvalidAccountData; "Account {} - expected empty tag", info.key)
38     }
39     msg!("holder account is empty!!!");
40
41     Ok(Self { info })
42 }
```

Listing 9: evm_loader/program/src/account/holder.rs (Line 46)

```
44 pub fn write(&mut self, offset: u32, bytes: &[u8]) -> Result<(), ProgramError> {
45     let mut data = self.info.try_borrow_mut_data()?;
```

```rust
    let begin = 1_usize/*TAG_EMPTY*/ + offset as usize;
    let end = begin.checked_add(bytes.len())
        .ok_or_else(|| E!(ProgramError::InvalidArgument; "Account data index overflow"))?;

    if data.len() < end {
        return Err!(ProgramError::AccountDataTooSmall; "Account data too small data.len()={:?}, offset={:?}, bytes.len()={:?}", data.len(), offset, bytes.len());
    }
    data[begin..end].copy_from_slice(bytes);

    Ok(())
}
```

Listing 10: evm_loader/program/src/instruction/transaction_step_from_-instruction.rs (Lines 47,49)

```rust
39 let mut account_storage = ProgramAccountStorage::new(
40     program_id,
41     accounts.remaining_accounts,
42     crate::config::token_mint::id(),
43     chain_id().as_u64(),
44 )?;
45
46
47 if is_new_transaction(program_id, storage_info, signature, &caller)? {
48     let trx = UnsignedTransaction::from_rlp(unsigned_msg)?;
49     let storage = Storage::new(program_id, storage_info, &accounts, caller, &trx, signature)?;
50
51     do_begin(step_count, accounts, storage, &mut account_storage, trx, caller)
```

Listing 11: evm_loader/program/src/instruction/transaction.rs (Line 32)

```rust
25 pub fn is_new_transaction<'a>(
26     program_id: &'a Pubkey,
27     storage_info: &'a AccountInfo<'a>,
28     signature: &[u8; 65],
```

```
    caller: &H160,
) -> Result<bool, ProgramError> {
    match account::tag(program_id, storage_info)? {
        account::TAG_EMPTY => Ok(true),
        FinalizedStorage::TAG => {
            if FinalizedStorage::from_account(program_id,
↳ storage_info)?.is_outdated(signature, caller) {
                Ok(true)
            } else {
                return Err!(EvmLoaderError::
↳ StorageAccountFinalized.into(); "Transaction already finalized")
            }
        },
        Storage::TAG => Ok(false),
        _ => return Err!(ProgramError::InvalidAccountData; "
↳ Account {} - expected storage or empty", storage_info.key)
    }
}
```

Risk Level:

Likelihood - 3
Impact - 5

Recommendation:

Introduce a new tag to distinguish uninitialized Holder accounts from initialized to prevent account data overwriting.

Remediation Plan:

SOLVED: The Neon team solved this issue in commit 62ecbc30372651486ed352df10af73484c410e02
A new tag was introduced, and the programs ensures that only the owner of the holder account can overwrite account data.

3.3 (HAL-03) OPERATOR ACCOUNTS CAN SELFDESTRUCT - MEDIUM

Description:

Operator accounts forward Ethereum transactions to Neon EVM. They charge transaction senders a NEON fee, and on each transaction execution they pay a fixed SOL fee to the Neon treasury.

In Solana, all custom accounts are required to pay rent to stay onchain. Rent is charged every epoch and if an account's balance drops below the required level, the account is at risk of being purged, i.e. all its properties are reset to default.

None of the Ethereum transaction handlers in the neon-evm program validate if the Operator account balance after transfer is rent-exempt, which puts every operator at risk of being purged.

Code Location:

Listing 12: evm_loader/program/src/instruction/transaction.rs (Line 54)

```
45 pub fn do_begin<'a>(
46     step_count: u64,
47     accounts: Accounts<'a>,
48     mut storage: Storage<'a>,
49     account_storage: &mut ProgramAccountStorage<'a>,
50     trx: UnsignedTransaction,
51     caller: H160,
52 ) -> ProgramResult {
53     debug_print!("do_begin");
54     accounts.system_program.transfer(&accounts.operator, &accounts
   ↳ .treasury, crate::config::PAYMENT_TO_TREASURE)?;
55     check_ethereum_transaction(account_storage, &caller, &trx)?;
```

Listing 13: evm_loader/program/src/instruction/transaction_execute_-from_instruction.rs (Line 82)

```
76 fn execute<'a>(
77     accounts: Accounts<'a>,
78     account_storage: &mut ProgramAccountStorage<'a>,
79     trx: UnsignedTransaction,
80     caller_address: H160,
81 ) -> ProgramResult {
82     accounts.system_program.transfer(&accounts.operator, &accounts
   ↳ .treasury, crate::config::PAYMENT_TO_TREASURE)?;
83
84     let (exit_reason, return_value, apply_state, used_gas) = {
```

Listing 14: evm_loader/program/src/instruction/transaction.rs (Line 94)

```
88 pub fn do_continue<'a>(
89     step_count: u64,
90     accounts: Accounts<'a>,
91     mut storage: Storage<'a>,
92     account_storage: &mut ProgramAccountStorage<'a>,
93 ) -> ProgramResult {
94     accounts.system_program.transfer(&accounts.operator, &accounts
   ↳ .treasury, crate::config::PAYMENT_TO_TREASURE)?;
95
96     let (results, used_gas) = {
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

On each transaction processing step, validate if the operator account balance is above the rent-exempt level.

Remediation Plan:

SOLVED: The Neon team solved this issue in commit 62ecbc30372651486ed352df10af73484c410e02.
The program verifies Operator accounts do not contain any data.

3.4 (HAL-04) INCORRECTLY GENERATED HOLDER SEED - LOW

Description:

Holder accounts are used to store such EVM transactions that cannot be sent in one Solana instruction because they do not fit in one UDP datagram. Instead, such transactions are first uploaded in chunks and saved in Holder accounts and then those accounts are provided as EVM transaction data source to Neon EVM.

Holder account addresses are created for the evm_loader program with the Pubkey::generate_from_seed method based on the operator address and a seed generated from a u64 holder ID. The seed is a keccak256 of the significant bytes of the provide holder ID.

The number of significant bytes for an ID is calculated as

$$(bitsize_{u64} - index_{most\_significant\_1} + 7)/8$$

The / operator denotes integer division, which truncates the remainder.

This formula increases this number by 1 even when the index of the most significant 1 is a multiple of 8, leading to incorrect seed generation and increasing the likelihood of collisions.

Code Location:

Listing 15: evm_loader/program/src/instruction/transaction_write_to_-holder.rs (Line 23)

```
19 let data_end: usize = data_begin + data_len;
20 let data = &instruction[data_begin..data_end];
21
22 let operator = Operator::from_account(&accounts[1])?;
23 let mut holder = Holder::from_account(program_id, holder_id, &
   ↳ accounts[0], &operator)?;
24
```

```
25 holder.write(offset, data)
```

Listing 16: evm_loader/program/src/account/holder.rs (Lines 18-22)

```
13 impl<'a> Holder<'a> {
14     pub fn from_account(program_id: &Pubkey, id: u64, info: &'a AccountInfo<'a>, operator: &Operator) -> Result<Self, ProgramError> {
15         // WTF!?
16         let bytes_count = std::mem::size_of_val(&id);
17         let bits_count = bytes_count * 8;
18         let holder_id_bit_length = bits_count - id.leading_zeros() as usize;
19         let significant_bytes_count = (holder_id_bit_length + 7) / 8;
20         let mut hasher = solana_program::keccak::Hasher::default();
21         hasher.hash(b"holder");
22         hasher.hash(&id.to_be_bytes()[bytes_count - significant_bytes_count..]);
23         let output = hasher.result();
24         let seed = &hex::encode(output)[..32];
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

Calculate the number of significant bytes as

$$index_{most\_significant\_1}/8 + index_{most\_significant\_1} mod(8)/7$$

Remediation Plan:

SOLVED: The Neon team has solved this issue in commit 30505ed23c692183995591e1bd479f76c6533caa. Holder seed was removed from the code.

3.5 (HAL-05) HARDCODED GOVERNANCE ADDRESSES - LOW

Description:

Important governance account addresses are hardcoded in evm_loader/program/src/config.rs. In case those addresses are compromised, the program authority has to redeploy the program to update them.

Code Location:

Listing 17: evm_loader/program/src/config.rs (Line 20)

```
17 // NOTE: when expanding this list, add same addresses to the
18 // alpha configuration as well
19 pubkey_array!(
20     AUTHORIZED_OPERATOR_LIST,
21     [
22         "NeonPQFrw5stVvs1rFLDxALWUBDCnSPsWBP83RfNUKK",
23         "NeoQM3utcHGxhKT41Nq81g8t4xGcPNFpkAgYj1N2N8v",
24         "Gw3Xiwve6HdvpJeQguhwT23cpK9nRjSy1NpNYCFY4XU9",
25         "DSRVyWpSVLEcHih9CVND2aGNBZxNW5bt34GEaK4aDk5i",
26     ]
27 );
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Consider making those governance addresses mutable and implement a function to update them in case they are compromised.

Remediation Plan:

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

3.6 (HAL-06) SUSCEPTIBLE TO INTEGER UNDERFLOW - LOW

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.

The evm-loader program has one integer underflows that could cause legitimate transactions to fail and thus cause a denial of service for the user. The ecrecover precompile subtracts a fixed value from a user-supplied v and if the result is negative, the transaction will fail.

Code Location:

Listing 18: evm_loader/program/src/precompile_contracts.rs (Line 543)

```
538 let v: u8 = if let Ok(v) = U256::from(v).as_u32().try_into() {
539     v
540 } else {
541     return Capture::Exit((ExitReason::Succeed(evm::ExitSucceed::
  ↳ Returned), vec![0; 32]));
542 };
543 let recovery_id = v - 27;
544 let public_key = match secp256k1_recover(&msg[..], recovery_id, &
  ↳ sig[..]) {
545     Ok(key) => key,
546     Err(_) => {
547         return Capture::Exit((ExitReason::Succeed(evm::ExitSucceed
  ↳ ::Returned), vec![0; 32]))
548     }
549 };
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to use safe and verified math libraries, such as checked_sub, for consistent arithmetic operations throughout the Solana program system. Consider using Rust safe arithmetic functions for primitives instead of standard arithmetic operators.

Remediation Plan:

SOLVED: The Neon team has solved this issue in commit 62ecbc30372651486ed352df10af73484c410e02.
The value of v must be in a specific range and cannot be lesser than 27.

3.7 (HAL-07) SUSCEPTIBLE TO ACCESS OUT-OF-BOUNDS - LOW

Description:

The EvmInstruction::CreateAccountV02 instruction handler expects the transaction sender to provide 3 required accounts and one extra:
- operator account
- system program account
- ethereum account
- ethereum smart contract account

Those accounts are retrieved from the AccountInfo slice by directly accessing the slice at relevant indices. Because the length of AccountInfo is not validated, the program may end up accessing the slice out-of-bounds if the transaction sender provides fewer accounts than expected and trigger a crash.

The instruction data processed by this handler is also susceptible to out-of-bounds access because the program assumes the provided slice is of required length when it generates the array reference.

This issue was also identified in the WriteHolder instruction handler in parsing input accounts and instruction data.

Code Location:

Listing 19: evm_loader/program/src/instruction/account_create.rs (Lines 20,22,23,36)

```
 9 struct Accounts<'a> {
10     operator: Operator<'a>,
11     system_program: program::System<'a>,
12     ether_account: &'a AccountInfo<'a>,
13     ether_contract: Option<&'a AccountInfo<'a>>,
14 }
15
16 pub fn process<'a>(program_id: &'a Pubkey, accounts: &'a [
```

```
AccountInfo<'a>], instruction: &[u8]) -> ProgramResult {
    solana_program::msg!("Instruction: Create Account");

    let parsed_accounts = Accounts {
        operator: unsafe { Operator::from_account_not_whitelisted(&accounts[0]) }?,
        system_program: program::System::from_account(&accounts[1])?,
        ether_account: &accounts[2],
        ether_contract: accounts.get(3),
    };

    let instruction = array_ref![instruction, 0, 20 + 1];
    let (address, bump_seed) = array_refs![instruction, 20, 1];
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Consider retrieving with the solana_program::account_info::next_account_info utility function to prevent Rust panics when accessing OOB.

Remediation Plan:

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

3.8 (HAL-08) OPERATOR ETHEREUM NEON TOKEN ADDRESS NOT VALIDATED - LOW

Description:

To compensate operators for processing NEON transactions, users pay fees in NEON tokens to operator nodes. The program transfers user tokens to the destination operator Ethereum account.

The CallFromRawEthereumTX instruction handler requires the operator to provide some accounts, including their Ethereum address, to transfer the transaction fees too. The handler does not verify if that Ethereum address actually belongs to the operator, which means the fees that should be paid to the operator might be transferred to an arbitrary Ethereum address instead.

Code Location:

Listing 20: evm_loader/program/src/instruction/transaction_execute_-from_instruction.rs (Line 119)

```
118 let gas_cost = used_gas.saturating_mul(trx.gas_price);
119 let payment_result = account_storage.transfer_gas_payment(
  ↳ caller_address, accounts.operator_ether_account, gas_cost);
120 let (exit_reason, return_value, apply_state) = match
  ↳ payment_result {
121     Ok(()) => {
122         (exit_reason, return_value, apply_state)
```

Listing 21: evm_loader/program/src/instruction/transaction_execute_-from_instruction.rs (Lines 18,37)

```
14 struct Accounts<'a> {
15     sysvar_instructions: sysvar::Instructions<'a>,
16     operator: Operator<'a>,
17     treasury: Treasury<'a>,
18     operator_ether_account: EthereumAccount<'a>,
19     system_program: program::System<'a>,
```

```rust
    neon_program: program::Neon<'a>,
    remaining_accounts: &'a [AccountInfo<'a>],
}

/// Execute Ethereum transaction in a single Solana transaction
/// Can only be used for function call or transfer
/// SOLANA TRANSACTION FAILS IF `trx.to` IS EMPTY
pub fn process<'a>(program_id: &'a Pubkey, accounts: &'a [AccountInfo<'a>], instruction: &[u8]) -> ProgramResult {
    solana_program::msg!("Instruction: Execute Transaction from Instruction");

    let treasury_index = u32::from_le_bytes(*array_ref![instruction, 0, 4]);
    let caller_address = H160::from(*array_ref![instruction, 4, 20]);
    let _signature = array_ref![instruction, 4 + 20, 65];
    let unsigned_msg = &instruction[4 + 20 + 65..];

    let accounts = Accounts {
        sysvar_instructions: sysvar::Instructions::from_account(&accounts[0])?,
        operator: Operator::from_account(&accounts[1])?,
        treasury: Treasury::from_account(program_id, treasury_index, &accounts[2])?,
```

Listing 22: evm_loader/program/src/instruction/transaction_execute_from_instruction.rs (Line 119)

```rust
118 let gas_cost = used_gas.saturating_mul(trx.gas_price);
119 let payment_result = account_storage.transfer_gas_payment(caller_address, accounts.operator_ether_account, gas_cost);
120 let (exit_reason, return_value, apply_state) = match payment_result {
121     Ok(()) => {
122         (exit_reason, return_value, apply_state)
123     },
124     Err(ProgramError::InsufficientFunds) => {
```

Listing 23: evm_loader/program/src/account_storage/apply.rs (Lines 23,28)

```
20 pub fn transfer_gas_payment(
21     &mut self,
22     origin: H160,
23     mut operator: EthereumAccount<'a>,
24     value: U256,
25 ) -> Result<(), ProgramError> {
26     let origin_balance = self.balance(&origin);
27     if origin_balance < value {
28         self.transfer_gas_payment(origin, operator, origin_balance)?;
29         return Err!(ProgramError::InsufficientFunds; "Account {} - insufficient funds", origin);
30     }
31
32
33     if self.ethereum_accounts.contains_key(&operator.address) {
34         self.transfer_neon_tokens(origin, operator.address, value)?;
35         core::mem::drop(operator);
36     } else {
37         let origin_account = self.ethereum_account_mut(&origin)
38             .ok_or_else(|| E!(ProgramError::InvalidArgument; "Account {} - expect initialized", origin))?;
39
40         let origin_balance = origin_account.balance.checked_sub(value)
41             .ok_or_else(|| E!(ProgramError::InsufficientFunds; "Account {} - insufficient funds, balance = {}", origin, origin_account.balance))?;
42
43         let operator_balance = operator.balance.checked_add(value)
44             .ok_or_else(|| E!(ProgramError::InvalidArgument; "Account {} - balance overflow", operator.address))?;
45
46         origin_account.balance = origin_balance;
47         operator.balance = operator_balance;
48     }
49
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Consider validating the destination operator Ethereum address to prevent operators from transferring NEON processing fees to accounts they do not own.

Remediation Plan:

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

3.9 (HAL-09) DUPLICATE INSTRUCTION HANDLERS - INFORMATIONAL

Description:

The evm-loader can process Ethereum transactions pre-saved in dedicated Holder accounts. The ExecuteTrxFromAccountDataIterativeOrContinueNoChainId and ExecuteTrxFromAccountDataIterativeOrContinue instruction handlers verify if the provided transaction is a fresh one or is it already being processed.

The only difference between those two handlers is the ExecuteTrxFromAccountDataIterati handler rejects transactions that are assigned some chain_id and sets a fixed transaction gas limit. The code can be optimized to decrease the program size and lower deployment and maintenance cost.

Code Location:

Listing 24

```
neon-evm/evm_loader/program/src/instruction$ diff transaction_step_from_account.rs transaction_step_from_account_no_chainid.rs
4a5
> use solana_program::program_error::ProgramError;
9a11
> use evm::U256;
14c16
<     solana_program::msg!("Instruction: Begin or Continue Transaction from Account");
---
>     solana_program::msg!("Instruction: Begin or Continue Transaction from Account Without ChainId");
45c47,52
<         let storage = Storage::new(program_id, storage_info, &accounts, caller, &trx, &signature)?;
---
>         if trx.chain_id.is_some() {
>             return Err!(ProgramError::InvalidArgument; "Expected
```

```
 transaction without chain id");
            }

            let mut storage = Storage::new(program_id, storage_info, &accounts, caller, &trx, &signature)?;
            storage.gas_limit = storage.gas_limit.saturating_mul(U256::from(crate::config::GAS_LIMIT_MULTIPLIER_NO_CHAINID));
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider removing the ExecuteTrxFromAccountDataIterativeOrContinueNoChainId instruction and its handler and modify the ExecuteTrxFromAccountDataIterativeOrContin instruction handler to a case when the transaction is missing chain ID:

Listing 25

```
let mut storage = Storage::new(program_id, storage_info, &accounts, caller, &trx, &signature)?;

if trx.chain_id.is_none() {
    storage.gas_limit = storage.gas_limit.saturating_mul(U256::from(crate::config::GAS_LIMIT_MULTIPLIER_NO_CHAINID));
}
```

Remediation Plan:

ACKNOWLEDGED: The Neon team acknowledged this finding.

3.10 (HAL-10) DUPLICATE SECURITY CHECKS - INFORMATIONAL

Description:

With the EvmInstruction::ResizeContractAccount instruction, users can migrate their Ethereum contracts to accounts with more data space. The handler expects the transaction sender to provide the "old" and the "new" account. The "old" account is zeorized and purged, and all data is migrated to the "new" account.

Because the handler writes to the "new" account, the Solana runtime is going to throw an error if an externally-owned account is provided instead of one owned by the evm_loader program, so verifying the "new" account owner is unnecessary.

Similarly, the check_secp256k1_instruction validates the address of the instruction sysvar account directly in line #56 in evm_loader/program/src/transaction.rs and then indirectly in line #62.

Code Location:

Listing 26: evm_loader/program/src/instruction/account_resize.rs (Line 62)

```
57 let expected_address = Pubkey::create_with_seed(operator.key, seed
   , program_id)?;
58 if *new_code_account.key != expected_address {
59     return Err!(ProgramError::InvalidArgument; "Account {} -
   expected key {}", new_code_account.key, expected_address);
60 }
61
62 if new_code_account.owner != program_id {
63     return Err!(ProgramError::InvalidArgument; "Account {} -
   expected program owned", new_code_account.key);
64 }
```

Listing 27: evm_loader/program/src/instruction/account_resize.rs (Lines 97,98)

```
80 let Accounts {
81     mut ethereum_account,
82     code_account,
83     new_code_account,
84     operator,
85 } = accounts;
86
87 if *code_account.key == Pubkey::default() {
88     EthereumContract::init(new_code_account, account::
 ↳ ether_contract::Data {
89         owner: *ethereum_account.info.key,
90         code_size: 0_u32,
91     })?;
92 } else {
93     {
94         let source = code_account.try_borrow_mut_data()?;
95         let mut dest = new_code_account.try_borrow_mut_data()?;
96
97         dest[..source.len()].copy_from_slice(&source);
98         dest[source.len()..].fill(0);
```

Listing 28: evm_loader/program/src/transaction.rs (Lines 56,62)

```
54 pub fn check_secp256k1_instruction(sysvar_info: &AccountInfo,
 ↳ message_len: usize, data_offset: u16) -> ProgramResult
55 {
56     if !solana_program::sysvar::instructions::check_id(sysvar_info
 ↳ .key) {
57         return Err!(ProgramError::InvalidAccountData; "Invalid
 ↳ sysvar instruction account {}", sysvar_info.key);
58     }
59
60     let message_len = u16::try_from(message_len).map_err(|e| E!(
 ↳ ProgramError::InvalidInstructionData; "TryFromIntError={:?}", e))
 ↳ ?;
61
62     let current_instruction = load_current_index_checked(
 ↳ sysvar_info)?;
63     let current_instruction = u8::try_from(current_instruction).
 ↳ map_err(|e| E!(ProgramError::InvalidInstructionData; "
 ↳ TryFromIntError={:?}", e))?;
```

```
    let index = current_instruction - 1;
```

Listing 29: solana-program-1.9.12/src/sysvar/instructions.rs (Line 119)

```
116 pub fn load_current_index_checked(
117     instruction_sysvar_account_info: &AccountInfo,
118 ) -> Result<u16, ProgramError> {
119 if !check_id(instruction_sysvar_account_info.key) {
120     return Err(ProgramError::UnsupportedSysvar);
121 }
122
123 let instruction_sysvar = instruction_sysvar_account_info.
  ↳ try_borrow_data()?;
124
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider removing the new_code_account ownership check from the EvmInstruction::ResizeContractAccount instruction handler and the instruction sysvar account ID validation from the check_secp256k1_instruction function.

Remediation Plan:

ACKNOWLEDGED: The Neon team acknowledged this finding.

3.11 (HAL-11) REDUNDANT INSTRUCTION DATA - INFORMATIONAL

Description:

The `EvmInstruction::CreateAccountV02` instruction handler expects the transaction sender to provide the Ethereum address for which a Solana account will be created and a `u8` bump. The program expects those two parameters to be serialized and provided as a 21-byte vector.

The Solana account address and the bump provided are verified to match the output of the `Pubkey::find_program_address` function called with the parsed Ethereum address and a static string as seeds.

Because the bump is expected to match the one generated with `Pubkey::find_program_address` it does not need to be provided externally as it's calculated on-chain.

Code Location:

Listing 30: evm_loader/program/src/instruction/account_create.rs (Lines 30,32)

```
26 let instruction = array_ref![instruction, 0, 20 + 1];
27 let (address, bump_seed) = array_refs![instruction, 20, 1];
28
29 let address = H160::from(address);
30 let bump_seed = u8::from_le_bytes(*bump_seed);
31
32 validate(program_id, &parsed_accounts, &address, bump_seed)?;
```

Listing 31: evm_loader/program/src/instruction/account_create.rs (Lines 42,46)

```
41 let program_seeds = [ &[ACCOUNT_SEED_VERSION], address.as_bytes()
   ↳ ];
42 let (expected_address, expected_bump_seed) = Pubkey::
   ↳ find_program_address(&program_seeds, program_id);
```

```
if expected_address != *accounts.ether_account.key {
    return Err!(ProgramError::InvalidArgument; "Account {} -
↳ expected PDA address {}", accounts.ether_account.key,
↳ expected_address);
}
if expected_bump_seed != bump_seed {
    return Err!(ProgramError::InvalidArgument; "Invalid bump seed,
↳  expected = {} found = {}", expected_bump_seed, bump_seed);
}
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider removing the bump from the EvmInstruction::CreateAccountV02 instruction data.

Remediation Plan:

ACKNOWLEDGED: The Neon team acknowledged this finding.

3.12 (HAL-12) REDUNDANT INSTRUCTION ACCOUNTS - INFORMATIONAL

Description:

The PartialCallOrContinueFromRawEthereumTX instruction handler expects the transaction sender, including the instruction sysvar account. This sysvar account stores all instruction included in the currently processed transaction, and the evm-loader program usually uses this sysvar account to verify secp256k1 signatures with the native solana-secp256k1-program program.

This instruction handler, however, implements custom secp256k1 signature verification, which renders the sysvar account unnecessary.

Code Location:

Listing 32: evm_loader/program/src/instruction/transaction_step_from_-instruction.rs (Line 28)

```
23 if caller != verify_tx_signature(signature, unsigned_msg)? {
24     return Err!(ProgramError::InvalidInstructionData; "Invalid signature");
25 }
26
27 let storage_info = &accounts[0];
28 let _sysvar_instructions = sysvar::Instructions::from_account(&accounts[1])?; // TODO remove it
29
30 let accounts = Accounts {
31     operator: Operator::from_account(&accounts[2])?,
32     treasury: Treasury::from_account(program_id, treasury_index, &accounts[3])?,
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Consider removing the instruction sysvar from the required EvmInstruction ::CreateAccountV02 instruction accounts.

Remediation Plan:

SOLVED: The Neon team fixed this issue in commit 62ecbc30372651486ed352df10af73484c410
Redundant instructions are removed.

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:

The issues identified were verified to be false positives.

```
Analyzing /root/audits/evm_loader/program/.coderrect/build/bpfel-unknown-unknown/release/deps/evm_loader.ll ...
Cargo.toml: spl_token version: 3.1.0
anchor_lang_version: 3.1.0 anchorVersionTooOld: 0
 - ✔ [00m:03s] Loading IR From File
 - ▘ [00m:00s] Running Compiler Optimization Passes
EntryPoints:
entrypoint
 - ✔ [00m:00s] Running Compiler Optimization Passes
 - ✔ [00m:02s] Running Pointer Analysis
=============This arithmetic operation may be UNSAFE!=================
Found a potential vulnerability at line 101, column 20 in program/src/gasometer.rs
The sub operation may result in underflows:

 95|        if !state.storage(address, key).is_zero() {
 96|            return;
 97|        }
 98|
 99|        let rent = self.rent.minimum_balance(STORAGE_ENTRY_BYTES);
 100|        let overhead = self.rent.minimum_balance(0);
>101|        let cost = rent - overhead;
 102|
 103|        self.gas = self.gas.saturating_add(cost);
 104|    }
 105|
 106|    pub fn record_deploy<B>(&mut self, state: &ExecutorState<B>, address: H160)
 107|    where
>>>Stack Trace:
>>>evm_loader::instruction::transaction_execute_from_instruction::process::h1cb5525fc9c6244f [program/src/entrypoint.rs:60]
>>>  evm_loader::instruction::transaction_execute_from_instruction::execute::h2b8601309dacd5ca [program/src/instruction/transaction_execute_from_instruction.rs:57]
>>>    evm_loader::executor::Machine$LT$B$GT$::execute::h24d46936a946ac0b [program/src/instruction/transaction_execute_from_instruction.rs:97]
>>>      evm_loader::executor::Machine$LT$B$GT$::execute_n_steps::h99ab3aba7f935351 [program/src/executor.rs:683]
>>>        evm_loader::executor::Machine$LT$B$GT$::run::he92a840135c6e474 [program/src/executor.rs:703]
>>>          evm_runtime::Runtime::run::h7304068fa04a2a73 [program/src/executor.rs:534]
>>>            evm_runtime::eval::eval::hf1f99b5d91f348e4 [/root/audits/evm_loader/rust-evm/runtime/src/lib.rs:200]
>>>              evm_runtime::eval::system::sstore::h2b4d7ad69a557cc1 [/root/audits/evm_loader/rust-evm/runtime/src/eval/mod.rs:52]
>>>                _$LT$evm_loader..executor..Executor$LT$B$GT$$u20$as$u20$evm_runtime..handler..Handler$GT$::set_storage::h7b0ca849865751a9 [/root/audits/evm_loader/rust-evm/runtime/src/eval/system.rs:201]
>>>                  evm_loader::gasometer::Gasometer::record_storage_write::hc5971d3f75ecdaaa [program/src/executor.rs:188]
```

```
=============This arithmetic operation may be UNSAFE!=================
Found a potential vulnerability at line 79, column 47 in program/src/query.rs
The sub operation may result in underflows:

 73|            Some(v) => {
 74|                if offset < v.offset || length == 0 {
 75|                    return Err(Error::InvalidArgument);
 76|                }
 77|                match &v.data {
 78|                    None => Err(Error::InvalidArgument),
>79|                    Some(d) => clone_chunk(d, offset - v.offset, length).map_or_else(
 80|                        || Err(Error::InvalidArgument),
 81|                        Ok
 82|                    ),
 83|                }
 84|            }
 85|        }
>>>Stack Trace:
>>>evm_loader::instruction::transaction_execute_from_instruction::process::h1cb5525fc9c6244f [program/src/entrypoint.rs:60]
>>>  evm_loader::instruction::transaction_execute_from_instruction::execute::h2b8601309dacd5ca [program/src/instruction/transaction_execute_from_instruction.rs:57]
>>>    evm_loader::executor::Machine$LT$B$GT$::execute::h24d46936a946ac0b [program/src/instruction/transaction_execute_from_instruction.rs:97]
>>>      evm_loader::executor::Machine$LT$B$GT$::execute_n_steps::h99ab3aba7f935351 [program/src/executor.rs:683]
>>>        evm_loader::executor::Machine$LT$B$GT$::run::he92a840135c6e474 [program/src/executor.rs:703]
>>>          evm_runtime::Runtime::run::h7304068fa04a2a73 [program/src/executor.rs:534]
>>>            evm_runtime::eval::eval::hf1f99b5d91f348e4 [/root/audits/evm_loader/rust-evm/runtime/src/lib.rs:200]
>>>              evm_runtime::eval::system::call::h51706f9fe10c7eb6 [/root/audits/evm_loader/rust-evm/runtime/src/eval/mod.rs:62]
>>>                _$LT$evm_loader..executor..Executor$LT$B$GT$$u20$as$u20$evm_runtime..handler..Handler$GT$::call::ha31d4d104c3af85b [/root/audits/evm_loader/rust-evm/runtime/src/eval/system.rs:369]
>>>                  evm_loader::precompile_contracts::call_precompile::hb88d7f68ef513362 [program/src/executor.rs:307]
>>>                    evm_loader::precompile_contracts::query_account::h860419d1261d4914 [program/src/precompile_contracts.rs:66]
>>>                      evm_loader::query::AccountCache::data::h079d9a13afef1044 [program/src/precompile_contracts.rs:505]
```

```
=============This arithmetic operation may be UNSAFE!================
Found a potential vulnerability at line 93, column 52 in program/src/storage_account.rs
The sub operation may result in underflows:

 87|        }
 88|
 89|        let mut storage = Storage::from_account(program_id, info)?;
 90|        storage.check_accounts(accounts)?;
 91|
 92|        let clock = Clock::get()?;
>93|        if (*operator.key != storage.operator) && ((clock.slot - storage.slot) <= OPERATOR_PRIORITY_SLOTS) {
 94|            return Err!(ProgramError::InvalidAccountData; "operator.key != storage.operator");
 95|        }
 96|
 97|        if storage.operator != *operator.key {
 98|            storage.operator = *operator.key;
 99|            storage.slot = clock.slot;
>>>Stack Trace:
>>>evm_loader::instruction::transaction_continue::process::h796ffa4b00a3d44e [program/src/entrypoint.rs:69]
>>>  evm_loader::storage_account::_$LT$impl$u20$evm_loader..account..AccountData$LT$evm_loader..account..storage..Data$GT$$GT$::restore::hf495697b0a5b6f28 [program/src/instruction/transaction_continue.rs:32]
```

```
=============This arithmetic operation may be UNSAFE!================
Found a potential vulnerability at line 22, column 39 in program/src/account/holder.rs
The sub operation may result in underflows:

 16|        let bytes_count = std::mem::size_of_val(&id);
 17|        let bits_count = bytes_count * 8;
 18|        let holder_id_bit_length = bits_count - id.leading_zeros() as usize;
 19|        let significant_bytes_count = (holder_id_bit_length + 7) / 8;
 20|        let mut hasher = solana_program::keccak::Hasher::default();
 21|        hasher.hash(b"holder");
>22|        hasher.hash(&id.to_be_bytes()[bytes_count - significant_bytes_count..]);
 23|        let output = hasher.result();
 24|        let seed = &hex::encode(output)[..32];
 25|
 26|        let expected_key = Pubkey::create_with_seed(operator.key, seed, program_id)?;
 27|        if *info.key != expected_key {
 28|            return Err!(ProgramError::InvalidArgument; "Account {} - expected holder key {}", info.key, expected_key);
>>>Stack Trace:
>>>evm_loader::instruction::transaction_write_to_holder::process::hb582b2a2db729dea [program/src/entrypoint.rs:54]
>>>  evm_loader::account::holder::Holder::from_account::h185344affa78156d [program/src/instruction/transaction_write_to_holder.rs:23]

 - ✔ [00m:05s] Building Static Happens-Before Graph
 - ✔ [00m:00s] Detecting Vulnerabilities
detected 0 untrustful accounts in total.
detected 4 unsafe math operations in total.

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

        4 unsafe arithmetic issues
```

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.

Results:

ID	package	Short Description
RUSTSEC-2020-0071	time	Potential segfault in the time crate

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:

```
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

1/1        117/117      1/1    0/0     5/5      ☢  evm-loader 0.7.6-dev
0/0        0/0          0/0    0/0     0/0      ?  ├── arrayref 0.3.6
0/0        22/22        0/0    0/0     0/0      ☢  ├── bincode 1.3.3
0/0        5/5          0/0    0/0     0/0      ☢  │   └── serde 1.0.139
0/0        0/0          0/0    0/0     0/0      ?  │       └── serde_derive 1.0.139
0/0        12/12        0/0    0/0     3/3      ☢  │           ├── proc-macro2 1.0.40
0/0        4/4          0/0    0/0     0/0      ☢  │           │   └── unicode-ident 1.0.2
0/0        0/0          0/0    0/0     0/0      ?  │           ├── quote 1.0.20
0/0        12/12        0/0    0/0     3/3      ☢  │           │   └── proc-macro2 1.0.40
0/0        50/50        3/3    0/0     2/2      ☢  │           └── syn 1.0.98
0/0        12/12        0/0    0/0     3/3      ☢  │               ├── proc-macro2 1.0.40
0/0        0/0          0/0    0/0     0/0      ?  │               ├── quote 1.0.20
0/0        4/4          0/0    0/0     0/0      ☢  │               └── unicode-ident 1.0.2
0/0        1/1          0/0    0/0     0/0      ☢  ├── bs58 0.3.1
0/0        0/0          0/0    0/0     0/0      ?  ├── cfg-if 1.0.0
0/0        0/12         0/0    0/0     0/1      ?  ├── const_format 0.2.26
0/0        0/0          0/0    0/0     0/0      ?  │   └── const_format_proc_macros 0.2.22
0/0        12/12        0/0    0/0     3/3      ☢  │       ├── proc-macro2 1.0.40
0/0        0/0          0/0    0/0     0/0      ?  │       ├── quote 1.0.20
0/0        50/50        3/3    0/0     2/2      ☢  │       ├── syn 1.0.98
0/0        0/0          0/0    0/0     0/0      🔒 │       └── unicode-xid 0.2.3
0/0        13/13        0/0    0/0     0/0      ☢  ├── evm 0.18.0
0/0        13/13        0/0    0/0     0/0      ☢  │   ├── evm-core 0.18.2
0/0        0/0          0/0    0/0     0/0      ?  │   │   ├── fixed-hash 0.7.0
1/1        193/193      0/0    0/0     0/0      ☢  │   │   │   ├── byteorder 1.4.3
0/0        0/32         0/0    0/0     0/0      ?  │   │   │   ├── rand 0.8.5
0/21       12/368       0/2    0/0     2/40     ☢  │   │   │   │   ├── libc 0.2.126
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 1.0.139
0/0        0/0          0/0    0/0     0/0      ?  │   │   │   │   ├── rand_chacha 0.3.1
2/2        636/712      0/0    0/0     17/25    ☢  │   │   │   │   │   ├── ppv-lite86 0.2.16
0/0        0/15         0/0    0/0     0/0      ?  │   │   │   │   │   ├── rand_core 0.6.3
2/4        52/166       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
0/21       12/368       0/2    0/0     2/40     ☢  │   │   │   │   │   │   │   └── libc 0.2.126
0/0        5/5          0/0    0/0     0/0      ☢  │   │   │   │   │   │   └── serde 1.0.139
0/0        5/5          0/0    0/0     0/0      ☢  │   │   │   │   │   └── serde 1.0.139
0/0        0/15         0/0    0/0     0/0      ?  │   │   │   │   ├── rand_core 0.6.3
0/0        5/5          0/0    0/0     0/0      ☢  │   │   │   │   └── serde 1.0.139
0/0        0/0          0/0    0/0     0/0      ?  │   │   │   ├── rustc-hex 2.1.0
0/0        0/0          0/0    0/0     0/0      ?  │   │   │   └── static_assertions 1.1.0
0/0        0/0          0/0    0/0     0/0      ?  │   │   ├── impl-rlp 0.3.0
0/0        0/0          0/0    0/0     0/0      ?  │   │   │   └── rlp 0.5.1
17/17      538/630      11/13  1/1     15/19    ☢  │   │   │       ├── bytes 1.1.0
0/0        5/5          0/0    0/0     0/0      ☢  │   │   │       │   └── serde 1.0.139
0/0        0/0          0/0    0/0     0/0      ?  │   │   │       └── rustc-hex 2.1.0
1/1        16/18        1/1    0/0     0/0      ☢  │   │   ├── log 0.4.17
0/0        0/0          0/0    0/0     0/0      ?  │   │   ├── rlp 0.5.1
0/0        5/5          0/0    0/0     0/0      ☢  │   │   ├── serde 1.0.139
0/0        16/16        0/0    0/0     0/0      ☢  │   │   ├── serde_bytes 0.11.6
0/0        5/5          0/0    0/0     0/0      ☢  │   │   │   └── serde 1.0.139
0/0        0/0          0/0    0/0     0/0      ?  │   │   └── uint 0.9.1
1/1        193/193      0/0    0/0     0/0      ☢  │   │       ├── byteorder 1.4.3
0/0        0/0          0/0    0/0     0/0      ?  │   │       ├── crunchy 0.2.2
0/0        0/0          0/0    0/0     0/0      ?  │   │       ├── hex 0.4.3
0/0        5/5          0/0    0/0     0/0      ☢  │   │       │   └── serde 1.0.139
0/0        15/15        0/0    0/0     0/0      ☢  │   │       ├── rand 0.7.3
2/4        50/150       1/1    0/0     3/3      ☢  │   │       │   ├── getrandom 0.1.16
```

AUTOMATED TESTING

```
0/0        0/0          0/0    0/0    0/0      ?  │   │       │   │   ├── cfg-if 1.0.0                    [2
0/21       12/368       0/2    0/0    2/40     ☢  │   │       │   │   ├── libc 0.2.126
1/1        16/18        1/1    0/0    0/0      ☢  │   │       │   │   └── log 0.4.17
0/21       12/368       0/2    0/0    2/40     ☢  │   │       │   ├── libc 0.2.126
1/1        16/18        1/1    0/0    0/0      ☢  │   │       │   ├── log 0.4.17
0/0        0/0          0/0    0/0    0/0      ?  │   │       │   ├── rand_chacha 0.2.2
2/2        636/712      0/0    0/0    17/25    ☢  │   │       │   │   ├── ppv-lite86 0.2.16
0/0        22/22        0/0    0/0    0/0      ☢  │   │       │   │   └── rand_core 0.5.1
2/4        50/150       1/1    0/0    3/3      ☢  │   │       │   │       ├── getrandom 0.1.16
0/0        5/5          0/0    0/0    0/0      ☢  │   │       │   │       └── serde 1.0.139
0/0        22/22        0/0    0/0    0/0      ☢  │   │       │   ├── rand_core 0.5.1
0/0        0/0          0/0    0/0    0/0      ?  │   │       │   └── rand_pcg 0.2.1
0/0        22/22        0/0    0/0    0/0      ☢  │   │       │       ├── rand_core 0.5.1
0/0        5/5          0/0    0/0    0/0      ☢  │   │       │       └── serde 1.0.139
0/0        0/0          0/0    0/0    0/0      ?  │   │       └── static_assertions 1.1.0
0/0        0/0          0/0    0/0    0/0      🔒 │   ├── evm-runtime 0.18.0
0/0        13/13        0/0    0/0    0/0      ☢  │   │   ├── evm-core 0.18.2
0/0        5/5          0/0    0/0    0/0      ☢  │   │   ├── serde 1.0.139
0/0        16/16        0/0    0/0    0/0      ☢  │   │   ├── serde_bytes 0.11.6
0/0        14/14        0/0    0/0    0/0      ☢  │   │   └── sha3 0.8.2
1/1        8/8          0/0    0/0    0/0      ☢  │   │       ├── block-buffer 0.7.3
0/0        0/0          0/0    0/0    0/0      ?  │   │       │   ├── block-padding 0.1.5
0/0        10/10        0/0    0/0    0/0      ☢  │   │       │   │   └── byte-tools 0.3.1
0/0        10/10        0/0    0/0    0/0      ☢  │   │       │   ├── byte-tools 0.3.1
1/1        193/193      0/0    0/0    0/0      ☢  │   │       │   ├── byteorder 1.4.3
1/1        223/223      18/18  8/8    5/5      ☢  │   │       │   └── generic-array 0.12.4
0/0        5/5          0/0    0/0    0/0      ☢  │   │       │       ├── serde 1.0.139
0/0        0/0          0/0    0/0    0/0      🔒 │   │       │       └── typenum 1.15.0
0/0        10/10        0/0    0/0    0/0      ☢  │   │       ├── byte-tools 0.3.1
0/0        0/0          0/0    0/0    0/0      ?  │   │       ├── digest 0.8.1
1/1        223/223      18/18  8/8    5/5      ☢  │   │       │   └── generic-array 0.12.4
0/0        0/0          0/0    0/0    0/0      ?  │   │       ├── keccak 0.1.2
0/0        0/0          0/0    0/0    0/0      ?  │   │       └── opaque-debug 0.2.3
1/1        16/18        1/1    0/0    0/0      ☢  │   ├── log 0.4.17
0/0        0/0          0/0    0/0    0/0      ?  │   ├── rlp 0.5.1
0/0        5/5          0/0    0/0    0/0      ☢  │   ├── serde 1.0.139
0/0        16/16        0/0    0/0    0/0      ☢  │   ├── serde_bytes 0.11.6
0/0        14/14        0/0    0/0    0/0      ☢  │   └── sha3 0.8.2
0/0        0/0          0/0    0/0    0/0      🔒 ├── evm-runtime 0.18.0
2/4        50/150       1/1    0/0    3/3      ☢  ├── getrandom 0.1.16
0/0        0/0          0/0    0/0    0/0      ?  ├── hex 0.4.3
0/0        0/0          0/0    0/0    0/0      ?  ├── impl-serde 0.1.1
0/0        0/0          0/0    0/0    0/0      ?  │   ├── rustc-hex 2.1.0
0/0        5/5          0/0    0/0    0/0      ☢  │   └── serde 1.0.139
1/1        16/18        1/1    0/0    0/0      ☢  ├── log 0.4.17
0/0        0/0          0/0    0/0    0/0      ?  ├── num-derive 0.3.3
0/0        12/12        0/0    0/0    3/3      ☢  │   ├── proc-macro2 1.0.40
0/0        0/0          0/0    0/0    0/0      ?  │   ├── quote 1.0.20
0/0        50/50        3/3    0/0    2/2      ☢  │   └── syn 1.0.98
0/0        6/12         0/0    0/0    0/0      ☢  ├── num-traits 0.2.15
0/0        0/0          0/0    0/0    0/0      ?  ├── num_enum 0.5.7
0/0        0/0          0/0    0/0    0/0      ?  │   └── num_enum_derive 0.5.7
0/0        0/0          0/0    0/0    0/0      ?  │       ├── proc-macro-crate 1.1.3
0/0        0/0          0/0    0/0    0/0      ?  │       │   ├── thiserror 1.0.31
0/0        0/0          0/0    0/0    0/0      ?  │       │   │   └── thiserror-impl 1.0.31
0/0        12/12        0/0    0/0    3/3      ☢  │       │   │       ├── proc-macro2 1.0.40
0/0        0/0          0/0    0/0    0/0      ?  │       │   │       ├── quote 1.0.20
0/0        50/50        3/3    0/0    2/2      ☢  │       │   │       └── syn 1.0.98
0/0        0/0          0/0    0/0    0/0      🔒 │       │   └── toml 0.5.9
0/0        0/46         0/1    0/0    0/0      ?  │       │       ├── indexmap 1.9.1
0/1        0/1367       0/24   0/1    0/69     ?  │       │       │   ├── hashbrown 0.12.3
0/0        26/30        0/0    0/0    0/0      ☢  │       │       │   │   ├── ahash 0.7.6
2/4        52/166       1/1    0/0    3/3      ☢  │       │       │   │   │   ├── getrandom 0.2.7
1/1        75/117       4/6    0/0    2/3      ☢  │       │       │   │   │   ├── once_cell 1.13.0
0/16       0/1341       0/0    0/0    0/56     ?  │       │       │   │   │   │   └── parking_lot_core 0.9.3
0/22       0/791        0/6    0/0    0/5      ?  │       │       │   │   │   │       ├── backtrace 0.3.66
0/0        0/26         0/0    0/0    0/0      ?  │       │       │   │   │   │       │   ├── addr2line 0.17.0
0/0        0/48         0/3    0/1    0/0      ?  │       │       │   │   │   │       │   │   ├── gimli 0.26.2
0/0        0/46         0/1    0/0    0/0      ?  │       │       │   │   │   │       │   │   │   ├── indexmap 1.9.1
0/0        0/0          0/18   0/2    0/0      ?  │       │       │   │   │   │       │   │   │   └── stable_deref_trait 1.2.0
```

```
0/0    0/0        0/0   0/0  0/0    ?                                              ├── rustc-demangle 0.1.21
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 1.0.139
0/0    0/0        0/0   0/0  0/0    ?                                          ├── cfg-if 1.0.0
0/21   12/368     0/2   0/0  2/40   ☢                                          ├── libc 0.2.126
0/0    0/0        0/0   0/0  0/0    🔒                                         ├── miniz_oxide 0.5.3
0/0    0/0        0/0   0/0  0/0    🔒                                             └── adler 1.0.2
0/0    0/26       0/0   0/1  0/0    ?                                          ├── object 0.29.0
0/6    0/156      0/0   0/0  0/0    ?                                              ├── crc32fast 1.3.2
0/0    0/0        0/0   0/0  0/0    ?                                                  └── cfg-if 1.0.0
0/2    0/118      0/2   0/0  0/2    ?                                              ├── flate2 1.0.24
0/6    0/156      0/0   0/0  0/0    ?                                                  ├── crc32fast 1.3.2
0/0    0/0        0/0   0/0  0/0    🔒                                                 └── miniz_oxide 0.5.3
0/1    0/1367     0/24  0/1  0/69   ?                                              ├── hashbrown 0.12.3
0/0    0/46       0/1   0/0  0/0    ?                                              ├── indexmap 1.9.1
36/37  2067/2144  0/0   0/0  21/21  ☢                                              └── memchr 2.5.0
0/21   12/368     0/2   0/0  2/40   ☢                                                  └── libc 0.2.126
0/0    0/0        0/0   0/0  0/0    ?                                          ├── rustc-demangle 0.1.21
0/0    5/5        0/0   0/0  0/0    ☢                                          └── serde 1.0.139
0/0    0/0        0/0   0/0  0/0    ?                                      ├── cfg-if 1.0.0
0/21   12/368     0/2   0/0  2/40   ☢                                      ├── libc 0.2.126
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 1.0.139
4/6    445/1159   4/10  1/1  12/25  ☢                          ├── bumpalo 3.10.0
0/6    0/663      0/5   0/0  0/3    ?                          ├── rayon 1.5.3
0/0    0/451      0/6   0/0  0/6    ?                              ├── crossbeam-deque 0.8.1
0/0    0/0        0/0   0/0  0/0    ?                                  ├── cfg-if 1.0.0
0/3    0/430      0/9   0/0  0/26   ?                                  ├── crossbeam-epoch 0.9.9
0/0    0/0        0/0   0/0  0/0    ?                                      ├── cfg-if 1.0.0
0/4    0/81       0/14  0/0  0/2    ?                                      ├── crossbeam-utils 0.8.10
0/0    0/0        0/0   0/0  0/0    ?                                          ├── cfg-if 1.0.0
1/1    75/117     4/6   0/0  2/3    ☢                                          └── once_cell 1.13.0
0/0    0/0        0/0   0/0  0/0    ?                                      ├── memoffset 0.6.5
1/1    75/117     4/6   0/0  2/3    ☢                                      ├── once_cell 1.13.0
0/0    0/18       0/1   0/0  0/0    ?                                      └── scopeguard 1.1.0
0/4    0/81       0/14  0/0  0/2    ?                                  └── crossbeam-utils 0.8.10
0/0    0/0        0/0   0/0  0/0    ?                              ├── either 1.7.0
0/0    5/5        0/0   0/0  0/0    ☢                                  └── serde 1.0.139
0/5    0/488      0/2   0/0  0/20   ?                              └── rayon-core 1.9.3
0/2    0/498      0/7   0/0  0/14   ?                                  ├── crossbeam-channel 0.5.5
0/0    0/0        0/0   0/0  0/0    ?                                      ├── cfg-if 1.0.0
0/4    0/81       0/14  0/0  0/2    ?                                      └── crossbeam-utils 0.8.10
0/0    0/451      0/6   0/0  0/6    ?                                  ├── crossbeam-deque 0.8.1
0/4    0/81       0/14  0/0  0/2    ?                                  ├── crossbeam-utils 0.8.10
0/0    0/72       0/0   0/0  0/0    ?                                  └── num_cpus 1.13.1
0/21   12/368     0/2   0/0  2/40   ☢                                      └── libc 0.2.126
0/0    5/5        0/0   0/0  0/0    ☢                          └── serde 1.0.139
0/6    0/663      0/5   0/0  0/3    ?                      ├── rayon 1.5.3
0/0    5/5        0/0   0/0  0/0    ☢                      └── serde 1.0.139
0/0    5/5        0/0   0/0  0/0    ☢                  └── serde 1.0.139
0/0    12/12      0/0   0/0  3/3    ☢          ├── proc-macro2 1.0.40
0/0    0/0        0/0   0/0  0/0    ?          ├── quote 1.0.20
0/0    50/50      3/3   0/0  2/2    ☢          └── syn 1.0.98
0/0    0/79       0/0   0/0  0/0    ?  ├── remove_dir_all 0.5.0
0/0    0/0        0/0   0/0  0/0    ?  ├── ripemd160 0.9.1
0/0    6/6        0/0   0/0  0/0    ☢      ├── block-buffer 0.9.0
0/0    3/3        0/0   0/0  0/0    ☢          ├── block-padding 0.2.1
1/1    292/292    20/20 8/8  5/5    ☢          └── generic-array 0.14.5
0/0    5/5        0/0   0/0  0/0    ☢              ├── serde 1.0.139
0/0    0/0        0/0   0/0  0/0    🔒             └── typenum 1.15.0
0/0    0/0        0/0   0/0  0/0    🔒     ├── digest 0.9.0
1/1    292/292    20/20 8/8  5/5    ☢          └── generic-array 0.14.5
0/0    0/0        0/0   0/0  0/0    ?      └── opaque-debug 0.3.0
0/0    0/0        0/0   0/0  0/0    ?  ├── rlp 0.5.1
0/0    5/5        0/0   0/0  0/0    ☢  ├── serde 1.0.139
0/0    16/16      0/0   0/0  0/0    ☢  ├── serde_bytes 0.11.6
0/0    0/0        0/0   0/0  0/0    ?  ├── serde_derive 1.0.139
3/3    405/405    1/1   0/0  2/2    ☢  ├── solana-program 1.9.12
0/0    0/0        0/0   0/0  0/0    🔒     ├── base64 0.13.0
0/0    22/22      0/0   0/0  0/0    ☢      ├── bincode 1.3.3
```

AUTOMATED TESTING

```
0/0      0/0        0/0     0/0   0/0    ?  │ ├── bitflags 1.3.2
10/78    71/3973    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 1.0.139
0/0      0/0        0/0     0/0   0/0    ?  │ │   ├── cfg-if 1.0.0
0/0      0/0        0/0     0/0   0/0    ?  │ │   ├── constant_time_eq 0.1.5
0/0      0/0        0/0     0/0   0/0    🔒 │ │   ├── digest 0.10.3
0/0      16/16      0/0     0/0   0/0    ☢  │ │   │   ├── block-buffer 0.10.2
1/1      292/292    20/20   8/8   5/5    ☢  │ │   │   │   └── generic-array 0.14.5
0/0      0/0        0/0     0/0   0/0    🔒 │ │   │   ├── crypto-common 0.1.6
1/1      292/292    20/20   8/8   5/5    ☢  │ │   │   │   ├── generic-array 0.14.5
0/0      0/15       0/0     0/0   0/0    ?  │ │   │   │   ├── rand_core 0.6.3
0/0      0/0        0/0     0/0   0/0    🔒 │ │   │   │   └── typenum 1.15.0
0/0      3/3        0/0     0/0   0/0    ☢  │ │   │   └── subtle 2.4.1
0/6      0/663      0/5     0/0   0/3    ?  │ │   └── rayon 1.5.3
0/0      7/7        0/0     0/0   0/0    ☢  │ ├── borsh 0.9.3
0/0      0/0        0/0     0/0   0/0    ?  │ │   ├── borsh-derive 0.9.3
0/0      0/0        0/0     0/0   0/0    ?  │ │   │   ├── borsh-derive-internal 0.9.3
0/0      12/12      0/0     0/0   3/3    ☢  │ │   │   │   ├── proc-macro2 1.0.40
0/0      0/0        0/0     0/0   0/0    ?  │ │   │   │   ├── quote 1.0.20
0/0      50/50      3/3     0/0   2/2    ☢  │ │   │   │   └── syn 1.0.98
0/0      0/0        0/0     0/0   0/0    ?  │ │   │   ├── borsh-schema-derive-internal 0.9
0/0      12/12      0/0     0/0   3/3    ☢  │ │   │   │   ├── proc-macro2 1.0.40
0/0      0/0        0/0     0/0   0/0    ?  │ │   │   │   ├── quote 1.0.20
0/0      50/50      3/3     0/0   2/2    ☢  │ │   │   │   └── syn 1.0.98
0/0      0/0        0/0     0/0   0/0    ?  │ │   │   ├── proc-macro-crate 0.1.5
0/0      0/0        0/0     0/0   0/0    🔒 │ │   │   │   └── toml 0.5.9
0/0      12/12      0/0     0/0   3/3    ☢  │ │   │   ├── proc-macro2 1.0.40
0/0      50/50      3/3     0/0   2/2    ☢  │ │   │   └── syn 1.0.98
2/2      1082/1198  19/22   1/1   51/58  ☢  │ │   └── hashbrown 0.11.2
0/0      26/30      0/0     0/0   0/0    ☢  │ │       ├── ahash 0.7.6
4/6      445/1159   4/10    1/1   12/25  ☢  │ │       ├── bumpalo 3.10.0
0/6      0/663      0/5     0/0   0/3    ?  │ │       ├── rayon 1.5.3
0/0      5/5        0/0     0/0   0/0    ☢  │ │       └── serde 1.0.139
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
8/8      202/202    0/0     0/0   0/0    ☢  │ │   └── sha2 0.9.9
0/0      6/6        0/0     0/0   0/0    ☢  │ │       ├── block-buffer 0.9.0
0/0      0/0        0/0     0/0   0/0    ?  │ │       ├── cfg-if 1.0.0
0/1      0/14       0/0     0/0   0/0    ?  │ │       ├── cpufeatures 0.2.2
0/0      0/0        0/0     0/0   0/0    🔒 │ │       ├── digest 0.9.0
0/0      0/0        0/0     0/0   0/0    ?  │ │       └── opaque-debug 0.3.0
2/2      206/206    0/0     0/0   7/7    ☢  │ ├── bv 0.11.1
0/0      5/5        0/0     0/0   0/0    ☢  │ │   └── serde 1.0.139
18/18    370/391    339/340 9/9    0/0   ☢  │  ├── bytemuck 1.10.0
0/0      0/0        0/0     0/0   0/0    ?  │ │   └── bytemuck_derive 1.1.0
0/0      12/12      0/0     0/0   3/3    ☢  │ │       ├── proc-macro2 1.0.40
0/0      0/0        0/0     0/0   0/0    ?  │ │       ├── quote 1.0.20
0/0      50/50      3/3     0/0   2/2    ☢  │ │       └── syn 1.0.98
0/2      0/857      0/0     0/0   0/0    ?  │ ├── curve25519-dalek 3.2.1
1/1      193/193    0/0     0/0   0/0    ☢  │ │   ├── byteorder 1.4.3
0/0      0/0        0/0     0/0   0/0    🔒 │ │   ├── digest 0.9.0
0/0      22/22      0/0     0/0   0/0    ☢  │ │   ├── rand_core 0.5.1
0/0      5/5        0/0     0/0   0/0    ☢  │ │   ├── serde 1.0.139
0/0      3/3        0/0     0/0   0/0    ☢  │ │   ├── subtle 2.4.1
1/1      23/23      0/0     0/0   0/0    ☢  │ │   └── zeroize 1.3.0
0/0      0/0        0/0     0/0   0/0    🔒 │ │       └── zeroize_derive 1.3.2
0/0      12/12      0/0     0/0   3/3    ☢  │ │           ├── proc-macro2 1.0.40
0/0      0/0        0/0     0/0   0/0    ?  │ │           ├── quote 1.0.20
0/0      50/50      3/3     0/0   2/2    ☢  │ │           ├── syn 1.0.98
0/0      0/0        1/1     0/0   0/0    ☢  │ │           └── synstructure 0.12.6
0/0      12/12      0/0     0/0   3/3    ☢  │ │               ├── proc-macro2 1.0.40
0/0      0/0        0/0     0/0   0/0    ?  │ │               ├── quote 1.0.20
0/0      50/50      3/3     0/0   2/2    ☢  │ │               ├── syn 1.0.98
0/0      0/0        0/0     0/0   0/0    🔒 │ │               └── unicode-xid 0.2.3
0/0      0/72       0/3     0/1   0/3    ?  │ ├── itertools 0.10.3
0/0      0/0        0/0     0/0   0/0    ?  │ │   └── either 1.7.0
0/0      0/72       0/3     0/1   0/3    ?  │ ├── itertools 0.10.3
0/0      7/7        1/1     0/0   0/0    ☢  │ ├── lazy_static 1.4.0
```

AUTOMATED TESTING

AUTOMATED TESTING

```
0/0    0/49       0/6    0/0  0/3    ?   │   └── spin 0.5.2
0/0    4/4        0/0    0/0  0/0    ☢   ├── libsecp256k1 0.6.0
0/0    0/0        0/0    0/0  0/0    ?   │   ├── arrayref 0.3.6
0/0    0/0        0/0    0/0  0/0    🔒  │   ├── base64 0.12.3
0/0    0/0        0/0    0/0  0/0    🔒  │   ├── digest 0.9.0
0/0    0/0        0/0    0/0  0/0    ?   │   ├── hmac-drbg 0.3.0
0/0    0/0        0/0    0/0  0/0    🔒  │   │   ├── digest 0.9.0
1/1    292/292    20/20  8/8  5/5    ☢   │   │   ├── generic-array 0.14.5
0/0    0/0        0/0    0/0  0/0    🔒  │   │   └── hmac 0.8.1
0/0    0/0        0/0    0/0  0/0    🔒  │   │       ├── crypto-mac 0.8.0
1/1    292/292    20/20  8/8  5/5    ☢   │   │       │   ├── generic-array 0.14.5
0/0    3/3        0/0    0/0  0/0    ☢   │   │       │   └── subtle 2.4.1
0/0    0/0        0/0    0/0  0/0    🔒  │   │       └── digest 0.9.0
0/0    7/7        1/1    0/0  0/0    ☢   │   ├── lazy_static 1.4.0
0/0    33/33      0/0    0/0  2/2    ☢   │   ├── libsecp256k1-core 0.2.2
0/0    0/0        0/0    0/0  0/0    ?   │   │   ├── crunchy 0.2.2
0/0    0/0        0/0    0/0  0/0    🔒  │   │   ├── digest 0.9.0
0/0    3/3        0/0    0/0  0/0    ☢   │   │   └── subtle 2.4.1
0/0    15/15      0/0    0/0  0/0    ☢   │   ├── rand 0.7.3
0/0    5/5        0/0    0/0  0/0    ☢   │   ├── serde 1.0.139
8/8    202/202    0/0    0/0  0/0    ☢   │   ├── sha2 0.9.9
0/0    0/0        0/0    0/0  0/0    🔒  │   └── typenum 1.15.0
1/1    16/18      1/1    0/0  0/0    ☢   ├── log 0.4.17
0/0    0/0        0/0    0/0  0/0    ?   ├── num-derive 0.3.3
0/0    6/12       0/0    0/0  0/0    ☢   ├── num-traits 0.2.15
0/0    15/15      0/0    0/0  0/0    ☢   ├── rand 0.7.3
0/1    0/1        0/0    0/0  0/0    ?   ├── rustversion 1.0.8
0/0    5/5        0/0    0/0  0/0    ☢   ├── serde 1.0.139
0/0    16/16      0/0    0/0  0/0    ☢   ├── serde_bytes 0.11.6
0/0    0/0        0/0    0/0  0/0    ?   ├── serde_derive 1.0.139
8/8    202/202    0/0    0/0  0/0    ☢   ├── sha2 0.9.9
0/0    14/14      0/0    0/0  0/0    ☢   ├── sha3 0.9.1
0/0    6/6        0/0    0/0  0/0    ☢   │   ├── block-buffer 0.9.0
0/0    0/0        0/0    0/0  0/0    🔒  │   ├── digest 0.9.0
0/0    0/0        0/0    0/0  0/0    ?   │   ├── keccak 0.1.2
0/0    0/0        0/0    0/0  0/0    ?   │   └── opaque-debug 0.3.0
0/0    0/0        0/0    0/0  0/0    ?   ├── solana-frozen-abi 1.9.12
0/0    1/1        0/0    0/0  0/0    ☢   │   ├── bs58 0.4.0
2/2    206/206    0/0    0/0  7/7    ☢   │   ├── bv 0.11.1
1/1    292/292    20/20  8/8  5/5    ☢   │   ├── generic-array 0.14.5
1/1    16/18      1/1    0/0  0/0    ☢   │   ├── log 0.4.17
0/0    147/282    4/6    0/0  7/7    ☢   │   ├── memmap2 0.5.5
0/21   12/368     0/2    0/0  2/40   ☢   │   │   ├── libc 0.2.126
0/0    0/0        0/18   0/2  0/0    ?   │   │   └── stable_deref_trait 1.2.0
0/0    5/5        0/0    0/0  0/0    ☢   │   ├── serde 1.0.139
0/0    0/0        0/0    0/0  0/0    ?   │   ├── serde_derive 1.0.139
8/8    202/202    0/0    0/0  0/0    ☢   │   ├── sha2 0.9.9
0/0    0/0        0/0    0/0  0/0    ?   │   ├── solana-frozen-abi-macro 1.9.12
0/0    12/12      0/0    0/0  3/3    ☢   │   │   ├── proc-macro2 1.0.40
0/0    0/0        0/0    0/0  0/0    ?   │   │   ├── quote 1.0.20
0/0    50/50      3/3    0/0  2/2    ☢   │   │   └── syn 1.0.98
0/0    0/0        0/0    0/0  0/0    ?   │   ├── solana-logger 1.9.12
0/0    0/0        0/0    0/0  0/0    ?   │   │   ├── env_logger 0.9.0
2/2    45/45      0/0    0/0  0/0    ☢   │   │   │   ├── atty 0.2.14
0/21   12/368     0/2    0/0  2/40   ☢   │   │   │   │   └── libc 0.2.126
0/0    0/0        0/0    0/0  0/0    🔒  │   │   │   ├── humantime 2.1.0
1/1    16/18      1/1    0/0  0/0    ☢   │   │   │   ├── log 0.4.17
0/0    34/34      1/2    0/0  2/2    ☢   │   │   │   ├── regex 1.6.0
19/19  678/678    0/0    0/0  22/22  ☢   │   │   │   │   ├── aho-corasick 0.7.18
36/37  2067/2144  0/0    0/0  21/21  ☢   │   │   │   │   │   └── memchr 2.5.0
36/37  2067/2144  0/0    0/0  21/21  ☢   │   │   │   │   ├── memchr 2.5.0
0/0    0/0        0/0    0/0  0/0    🔒  │   │   │   │   └── regex-syntax 0.6.27
0/0    0/0        0/0    0/0  0/0    ?   │   │   │   └── termcolor 1.1.3
0/0    7/7        1/1    0/0  0/0    ☢   │   │   ├── lazy_static 1.4.0
1/1    16/18      1/1    0/0  0/0    ☢   │   │   └── log 0.4.17
0/0    0/0        0/0    0/0  0/0    ?   │   └── thiserror 1.0.31
0/0    0/0        0/0    0/0  0/0    ?   ├── solana-frozen-abi-macro 1.9.12
0/0    0/0        0/0    0/0  0/0    ?   ├── solana-logger 1.9.12
0/0    0/0        0/0    0/0  0/0    ?   ├── solana-sdk-macro 1.9.12
0/0    1/1        0/0    0/0  0/0    ☢   │   ├── bs58 0.4.0
```

```
0/0      12/12      0/0     0/0   3/3     ☢️  │   │   ├── proc-macro2 1.0.40
0/0      0/0        0/0     0/0   0/0     ?   │   │   ├── quote 1.0.20
0/1      0/1        0/0     0/0   0/0     ?   │   │   ├── rustversion 1.0.8
0/0      50/50      3/3     0/0   2/2     ☢️  │   │   └── syn 1.0.98
0/0      0/0        0/0     0/0   0/0     ?   │   ├── thiserror 1.0.31
12/14    432/496    16/16   2/2   9/9     ☢️  │   └── wasm-bindgen 0.2.81
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 1.0.139
0/0      0/7        0/0     0/0   0/0     ?   │       ├── serde_json 1.0.82
0/0      0/46       0/1     0/0   0/0     ?   │       │   ├── indexmap 1.9.1
0/0      0/7        0/0     0/0   0/0     ?   │       │   ├── itoa 1.0.2
0/9      0/723      0/0     0/0   0/2     ?   │       │   ├── ryu 1.0.10
0/0      5/5        0/0     0/0   0/0     ☢️  │       │   └── serde 1.0.139
0/1      0/0        0/1     0/0   0/1     ?   │       └── wasm-bindgen-macro 0.2.81
0/0      0/0        0/0     0/0   0/0     ?   │           ├── quote 1.0.20
0/0      0/0        0/0     0/0   0/0     ?   │           └── wasm-bindgen-macro-support 0.2.81
0/0      12/12      0/0     0/0   3/3     ☢️  │               ├── proc-macro2 1.0.40
0/0      0/0        0/0     0/0   0/0     ?   │               ├── quote 1.0.20
0/0      50/50      3/3     0/0   2/2     ☢️  │               ├── syn 1.0.98
0/0      0/0        0/0     0/0   0/0     ?   │               ├── wasm-bindgen-backend 0.2.81
4/6      445/1159   4/10    1/1   12/25   ☢️  │               │   ├── bumpalo 3.10.0
0/0      7/7        1/1     0/0   0/0     ☢️  │               │   ├── lazy_static 1.4.0
1/1      16/18      1/1     0/0   0/0     ☢️  │               │   ├── log 0.4.17
0/0      12/12      0/0     0/0   3/3     ☢️  │               │   ├── proc-macro2 1.0.40
0/0      0/0        0/0     0/0   0/0     ?   │               │   ├── quote 1.0.20
0/0      50/50      3/3     0/0   2/2     ☢️  │               │   ├── syn 1.0.98
0/0      0/0        0/0     0/0   0/0     ?   │               │   └── wasm-bindgen-shared 0.2.81
0/0      0/0        0/0     0/0   0/0     ?   │               └── wasm-bindgen-shared 0.2.81
0/0      0/0        0/0     0/0   0/0     🔒  ├── spl-associated-token-account 1.0.3
3/3      405/405    1/1     0/0   2/2     ☢️  │   ├── solana-program 1.9.12
0/0      0/0        0/0     0/0   0/0     🔒  │   └── spl-token 3.2.0
0/0      0/0        0/0     0/0   0/0     ?   │       ├── arrayref 0.3.6
0/0      0/0        0/0     0/0   0/0     ?   │       ├── num-derive 0.3.3
0/0      6/12       0/0     0/0   0/0     ☢️  │       ├── num-traits 0.2.15
0/0      0/0        0/0     0/0   0/0     ?   │       ├── num_enum 0.5.7
3/3      405/405    1/1     0/0   2/2     ☢️  │       ├── solana-program 1.9.12
0/0      0/0        0/0     0/0   0/0     ?   │       └── thiserror 1.0.31
0/0      0/0        0/0     0/0   0/0     🔒  ├── spl-token 3.2.0
0/0      0/0        0/0     0/0   0/0     ?   └── thiserror 1.0.31

149/329  9092/23800 448/582 30/37  204/501
```

AUTOMATED TESTING

THANK YOU FOR CHOOSING

HALBORN