



VRust

Security Assessment

O2Lab VRust Team

11/02/2022 20:55:15

Contents

Summary	3
Overview	4
Project Summary	4
Audit Summary	4
Vulnerability Summary	4
Findings	5
Finding Statistic	6
Issue: 0: IntegerFlow	7
Issue: 1: MissingKeyCheck	9
Issue: 2: CrossProgramInvocation	11
Issue: 3: CrossProgramInvocation	14
Issue: 4: CrossProgramInvocation	17
Issue: 5: CrossProgramInvocation	20
Issue: 6: CrossProgramInvocation	23
Issue: 7: CrossProgramInvocation	27
Issue: 8: CrossProgramInvocation	31
Issue: 9: CrossProgramInvocation	35
Appendix	38
Finding Categories	38
Gas Optimization	38
Mathematical Operations	38
Logical Issue	38
Language Specific	38
Coding Style	38
Checksum Calculation Method	38
Disclaimer	40

Summary

This report has been prepared for O2Lab VRust Team to discover issues and vulnerabilities in the source code of the O2Lab VRust Team project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques. The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	O2Lab VRust Team
Platform	Ethereum
Language	Solana
Crate	token_bridge
GitHub Location	https://github.com/parasol-aser/vrust
sha256	Unknown

Audit Summary

Delivery Date	11/02/2022
Audit Methodology	Static Analysis
Key Components	

Vulnerability Summary

Vulnerability Level	Total
Critical	10
Major	0
Medium	0
Minor	0
Informational	0
Discussion	0

Findings

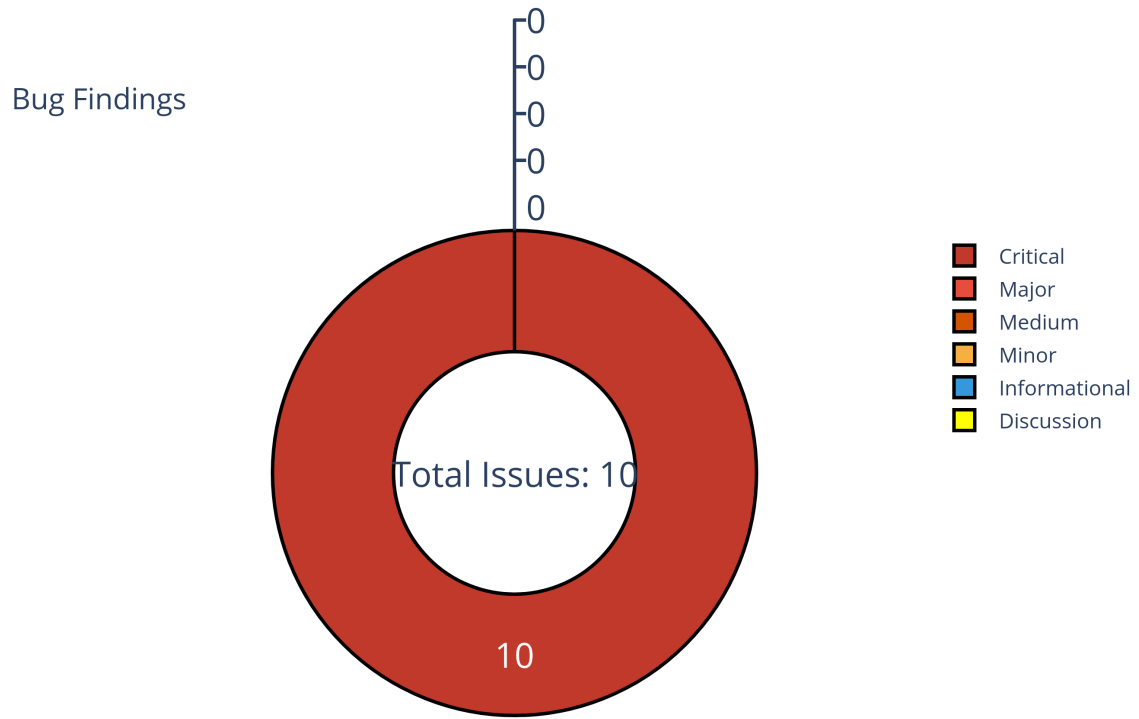


Figure 1: Findings

Finding Statistic

Category	Count
IntegerFlow	1
MissingKeyCheck	1
CrossProgramInvocation	8

ID	Category	Severity	Status
0	IntegerFlow	Critical	UnResolved
1	MissingKeyCheck	Critical	UnResolved
2	CrossProgramInvocation	Critical	UnResolved
3	CrossProgramInvocation	Critical	UnResolved
4	CrossProgramInvocation	Critical	UnResolved
5	CrossProgramInvocation	Critical	UnResolved
6	CrossProgramInvocation	Critical	UnResolved
7	CrossProgramInvocation	Critical	UnResolved
8	CrossProgramInvocation	Critical	UnResolved
9	CrossProgramInvocation	Critical	UnResolved

Issue: 0: IntegerFlow

Category	Severity	Status
IntegerFlow	Critical	UnResolved

- Location

program/src/api/transfer.rs:174:23: 174:50

```
174 data.amount / trunc_divisor
175
```

- Code Context

Vulnerability at Line: 174

```
169         invoke_signed(&init_ix, ctx.accounts, &[])?;
170     }
171
172     let trunc_divisor = 10u64.pow(8.max(accs.mint.decimals as u32) - 8);
173     // Truncate to 8 decimals
174     let amount: u64 = data.amount / trunc_divisor;
175     let fee: u64 = data.fee / trunc_divisor;
176     // Untruncate the amount to drop the remainder so we don't "burn"
177     // ↳ user's funds.
178     let amount_trunc: u64 = amount * trunc_divisor;
179
```

- Call Stack

```
1 fn entrypoint() { // /home/yifei/.cargo/registry/src/github.com-
↳ 1ecc6299db9ec823/solana-program-1.7.0/src/entrypoint.rs:46:9: 53:10
↳ }
2 fn instruction::solitaire() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 108:14 }
3 fn instruction::dispatch() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 99:14 }
```

4
5
6

```
fn instruction::TransferNative::execute(){//  
    ↪ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/m  
    ↪ 74:22 }  
fn api::transfer::transfer_native(){//  
    ↪ program/src/api/transfer.rs:127:1: 234:2 }
```

- description:
- link:
- alleviation:

Issue: 1: MissingKeyCheck

Category	Severity	Status
MissingKeyCheck	Critical	UnResolved

- Location

/home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/processors/peel.rs:211:22:
211:50

```
211 ctx.info().lamports.borrow()  
212
```

- Code Context

– Function Definition:

```
192 fn peel<I>(ctx: &'c mut Context<'a, 'b, 'c, I>) -> Result<Self>  
193
```

Vulnerability at Line: 202

```
197     }  
198  
199     // If we're initializing the type, we should emit system/rent as  
200     ↪ deps.  
201     let (initialized, data): (bool, T) = match IsInitialized {  
202         AccountState::Uninitialized => {  
203             if **ctx.info().lamports.borrow() != 0 {  
204                 return  
205                 ↪ Err(SolitaireError::AlreadyInitialized(*ctx.info().key));  
206             }  
207             (false, T::default())  
208         }  
209     }
```

Other Use Case for Variable: ctx.info().lamports.borrow()

211

```
if **ctx.info().lamports.borrow() == 0 {
```

- Call Stack

1
2
3
4
5
6
7

```
fn entrypoint() { // /home/yifei/.cargo/registry/src/github.com-
↳ 1ecc6299db9ec823/solana-program-1.7.0/src/entrypoint.rs:46:9: 53:10
↳ }
fn instruction::solitaire() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 108:14 }
fn instruction::dispatch() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 99:14 }
fn instruction::AttestToken::execute() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 74:22 }
fn <api::attest::AttestToken<'b> as
↳ solitaire::FromAccounts<'a, 'b, 'c>::from() { //
↳ program/src/api/attest.rs:68:10: 68:22 }
fn <solitaire::Data<'b, T, IsInitialized> as
↳ solitaire::Peel<'a, 'b, 'c>::peel() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 236:6 }
```

- description:
- link:
- alleviation:

Issue: 2: CrossProgramInvocation

Category	Severity	Status
CrossProgramInvocation	Critical	UnResolved

- Location

```
program/src/api/complete_transfer.rs
```

- Code Context

```
85 pub fn complete_native(  
86     ctx: &ExecutionContext,  
87     accs: &mut CompleteNative,  
88     data: CompleteNativeData,  
89 ) -> Result<()> {  
90     // Verify the chain registration  
91     let derivation_data: EndpointDerivationData = (&*accs).into();  
92     accs.chain_registration  
93         .verify_derivation(ctx.program_id, &derivation_data)?;  
94  
95     // Verify that the custody account is derived correctly  
96     let derivation_data: CustodyAccountDerivationData = (&*accs).into();  
97     accs.custody  
98         .verify_derivation(ctx.program_id, &derivation_data)?;  
99  
100    // Verify mints  
101    if *accs.mint.info().key != accs.to.mint {  
102        return Err(InvalidMint.into());  
103    }  
104    if *accs.mint.info().key != accs.to_fees.mint {  
105        return Err(InvalidMint.into());  
106    }  
107    if *accs.mint.info().key != accs.custody.mint {  
108        return Err(InvalidMint.into());  
109    }  
110    if *accs.custody_signer.key != accs.custody.owner {  
111        return Err(WrongAccountOwner.into());
```

```
112     }
113
114     // Verify VAA
115     if accs.vaa.token_address != accs.mint.info().key.to_bytes() {
116         return Err(InvalidMint.into());
117     }
118     if accs.vaa.token_chain != 1 {
119         return Err(InvalidChain.into());
120     }
121     if accs.vaa.to_chain != CHAIN_ID_SOLANA {
122         return Err(InvalidChain.into());
123     }
124     if accs.vaa.to != accs.to.info().key.to_bytes() {
125         return Err(InvalidRecipient.into());
126     }
127
128     // Prevent vaa double signing
129     accs.vaa.verify(ctx.program_id)?;
130     accs.vaa.claim(ctx, accs.payer.key)?;
131
132     let mut amount = accs.vaa.amount.as_u64();
133     let mut fee = accs.vaa.fee.as_u64();
134
135     // Wormhole always caps transfers at 8 decimals; un-truncate if the
136     ↳ local token has more
137     if accs.mint.decimals > 8 {
138         amount *= 10u64.pow((accs.mint.decimals - 8) as u32);
139         fee *= 10u64.pow((accs.mint.decimals - 8) as u32);
140     }
141
142     // Transfer tokens
143     let transfer_ix = spl_token::instruction::transfer(
144         &spl_token::id(),
145         accs.custody.info().key,
146         accs.to.info().key,
147         accs.custody_signer.key,
148         &[],
149         amount.checked_sub(fee).unwrap(),
150     )?;
151     invoke_seeded(&transfer_ix, ctx, &accs.custody_signer, None)?;
152
153     // Transfer fees
```

```

153     let transfer_ix = spl_token::instruction::transfer(
154         &spl_token::id(),
155         accs.custody.info().key,
156         accs.to_fees.info().key,
157         accs.custody_signer.key,
158         &[],
159         fee,
160     )?;
161     invoke_seeded(&transfer_ix, ctx, &accs.custody_signer, None)?;
162
163     Ok(())
164 }
165

```

- Call Stack

```

1  fn entrypoint() { // /home/yifei/.cargo/registry/src/github.com-
    ↪ 1ecc6299db9ec823/solana-program-1.7.0/src/entrypoint.rs:46:9: 53:10
    ↪ }
2  fn instruction::solitaire() { //
    ↪ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
    ↪ 108:14 }
3  fn instruction::dispatch() { //
    ↪ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
    ↪ 99:14 }
4  fn instruction::CompleteNative::execute() { //
    ↪ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
    ↪ 74:22 }
5      fn api::complete_transfer::complete_native() { //
    ↪ program/src/api/complete_transfer.rs:85:1: 164:2 }
6

```

- description:
- link:
- alleviation:

Issue: 3: CrossProgramInvocation

Category	Severity	Status
CrossProgramInvocation	Critical	UnResolved

- Location

```
program/src/api/complete_transfer.rs
```

- Code Context

```
85 pub fn complete_native(  
86     ctx: &ExecutionContext,  
87     accs: &mut CompleteNative,  
88     data: CompleteNativeData,  
89 ) -> Result<()> {  
90     // Verify the chain registration  
91     let derivation_data: EndpointDerivationData = (&*accs).into();  
92     accs.chain_registration  
93         .verify_derivation(ctx.program_id, &derivation_data)?;  
94  
95     // Verify that the custody account is derived correctly  
96     let derivation_data: CustodyAccountDerivationData = (&*accs).into();  
97     accs.custody  
98         .verify_derivation(ctx.program_id, &derivation_data)?;  
99  
100    // Verify mints  
101    if *accs.mint.info().key != accs.to.mint {  
102        return Err(InvalidMint.into());  
103    }  
104    if *accs.mint.info().key != accs.to_fees.mint {  
105        return Err(InvalidMint.into());  
106    }  
107    if *accs.mint.info().key != accs.custody.mint {  
108        return Err(InvalidMint.into());  
109    }  
110    if *accs.custody_signer.key != accs.custody.owner {  
111        return Err(WrongAccountOwner.into());
```

```
112     }
113
114     // Verify VAA
115     if accs.vaa.token_address != accs.mint.info().key.to_bytes() {
116         return Err(InvalidMint.into());
117     }
118     if accs.vaa.token_chain != 1 {
119         return Err(InvalidChain.into());
120     }
121     if accs.vaa.to_chain != CHAIN_ID_SOLANA {
122         return Err(InvalidChain.into());
123     }
124     if accs.vaa.to != accs.to.info().key.to_bytes() {
125         return Err(InvalidRecipient.into());
126     }
127
128     // Prevent vaa double signing
129     accs.vaa.verify(ctx.program_id)?;
130     accs.vaa.claim(ctx, accs.payer.key)?;
131
132     let mut amount = accs.vaa.amount.as_u64();
133     let mut fee = accs.vaa.fee.as_u64();
134
135     // Wormhole always caps transfers at 8 decimals; un-truncate if the
136     ↪ local token has more
137     if accs.mint.decimals > 8 {
138         amount *= 10u64.pow((accs.mint.decimals - 8) as u32);
139         fee *= 10u64.pow((accs.mint.decimals - 8) as u32);
140     }
141
142     // Transfer tokens
143     let transfer_ix = spl_token::instruction::transfer(
144         &spl_token::id(),
145         accs.custody.info().key,
146         accs.to.info().key,
147         accs.custody_signer.key,
148         &[],
149         amount.checked_sub(fee).unwrap(),
150     )?;
151     invoke_seeded(&transfer_ix, ctx, &accs.custody_signer, None)?;
152
153     // Transfer fees
```

```

153     let transfer_ix = spl_token::instruction::transfer(
154         &spl_token::id(),
155         accs.custody.info().key,
156         accs.to_fees.info().key,
157         accs.custody_signer.key,
158         &[],
159         fee,
160     )?;
161     invoke_seeded(&transfer_ix, ctx, &accs.custody_signer, None)?;
162
163     Ok(())
164 }
165

```

- Call Stack

```

1  fn entrypoint() { // /home/yifei/.cargo/registry/src/github.com-
    ↳ 1ecc6299db9ec823/solana-program-1.7.0/src/entrypoint.rs:46:9: 53:10
    ↳ }
2  fn instruction::solitaire() { //
    ↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
    ↳ 108:14 }
3  fn instruction::dispatch() { //
    ↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
    ↳ 99:14 }
4  fn instruction::CompleteNative::execute() { //
    ↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
    ↳ 74:22 }
5      fn api::complete_transfer::complete_native() { //
        ↳ program/src/api/complete_transfer.rs:85:1: 164:2 }
6

```

- description:
- link:
- alleviation:

Issue: 4: CrossProgramInvocation

Category	Severity	Status
CrossProgramInvocation	Critical	UnResolved

- Location

```
program/src/api/complete_transfer.rs
```

- Code Context

```
208 pub fn complete_wrapped(  
209     ctx: &ExecutionContext,  
210     accs: &mut CompleteWrapped,  
211     data: CompleteWrappedData,  
212 ) -> Result<()> {  
213     // Verify the chain registration  
214     let derivation_data: EndpointDerivationData = (&*accs).into();  
215     accs.chain_registration  
216         .verify_derivation(ctx.program_id, &derivation_data)?;  
217  
218     // Verify mint  
219     accs.wrapped_meta.verify_derivation(  
220         ctx.program_id,  
221         &WrappedMetaDerivationData {  
222             mint_key: *accs.mint.info().key,  
223         },  
224     )?;  
225     if accs.wrapped_meta.token_address != accs.vaa.token_address  
226         || accs.wrapped_meta.chain != accs.vaa.token_chain  
227     {  
228         return Err(InvalidMint.into());  
229     }  
230  
231     // Verify mints  
232     if *accs.mint.info().key != accs.to.mint {  
233         return Err(InvalidMint.into());  
234     }
```

```
235     if *accs.mint.info().key != accs.to_fees.mint {
236         return Err(InvalidMint.into());
237     }
238
239     // Verify VAA
240     if accs.vaa.to_chain != CHAIN_ID_SOLANA {
241         return Err(InvalidChain.into());
242     }
243     if accs.vaa.to != accs.to.info().key.to_bytes() {
244         return Err(InvalidRecipient.into());
245     }
246
247     accs.vaa.verify(ctx.program_id)?;
248     accs.vaa.claim(ctx, accs.payer.key)?;
249
250     // Mint tokens
251     let mint_ix = spl_token::instruction::mint_to(
252         &spl_token::id(),
253         accs.mint.info().key,
254         accs.to.info().key,
255         accs.mint_authority.key,
256         &[],
257         accs.vaa
258             .amount
259             .as_u64()
260             .checked_sub(accs.vaa.fee.as_u64())
261             .unwrap(),
262     )?;
263     invoke_seeded(&mint_ix, ctx, &accs.mint_authority, None)?;
264
265     // Mint fees
266     let mint_ix = spl_token::instruction::mint_to(
267         &spl_token::id(),
268         accs.mint.info().key,
269         accs.to_fees.info().key,
270         accs.mint_authority.key,
271         &[],
272         accs.vaa.fee.as_u64(),
273     )?;
274     invoke_seeded(&mint_ix, ctx, &accs.mint_authority, None)?;
275
276     Ok(())
```

277
278

```
}
}
```

- Call Stack

1
2
3
4
5
6

```
fn entrypoint(){// /home/yifei/.cargo/registry/src/github.com-
↳ 1ecc6299db9ec823/solana-program-1.7.0/src/entrypoint.rs:46:9: 53:10
↳ }
fn instruction::solitaire(){//
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 108:14 }
fn instruction::dispatch(){//
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 99:14 }
fn instruction::CompleteWrapped::execute(){//
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 74:22 }
fn api::complete_transfer::complete_wrapped(){//
↳ program/src/api/complete_transfer.rs:208:1: 277:2 }
```

- description:
- link:
- alleviation:

Issue: 5: CrossProgramInvocation

Category	Severity	Status
CrossProgramInvocation	Critical	UnResolved

- Location

```
program/src/api/complete_transfer.rs
```

- Code Context

```
208 pub fn complete_wrapped(  
209     ctx: &ExecutionContext,  
210     accs: &mut CompleteWrapped,  
211     data: CompleteWrappedData,  
212 ) -> Result<()> {  
213     // Verify the chain registration  
214     let derivation_data: EndpointDerivationData = (&*accs).into();  
215     accs.chain_registration  
216         .verify_derivation(ctx.program_id, &derivation_data)?;  
217  
218     // Verify mint  
219     accs.wrapped_meta.verify_derivation(  
220         ctx.program_id,  
221         &WrappedMetaDerivationData {  
222             mint_key: *accs.mint.info().key,  
223         },  
224     )?;  
225     if accs.wrapped_meta.token_address != accs.vaa.token_address  
226         || accs.wrapped_meta.chain != accs.vaa.token_chain  
227     {  
228         return Err(InvalidMint.into());  
229     }  
230  
231     // Verify mints  
232     if *accs.mint.info().key != accs.to.mint {  
233         return Err(InvalidMint.into());  
234     }
```

```
235     if *accs.mint.info().key != accs.to_fees.mint {
236         return Err(InvalidMint.into());
237     }
238
239     // Verify VAA
240     if accs.vaa.to_chain != CHAIN_ID_SOLANA {
241         return Err(InvalidChain.into());
242     }
243     if accs.vaa.to != accs.to.info().key.to_bytes() {
244         return Err(InvalidRecipient.into());
245     }
246
247     accs.vaa.verify(ctx.program_id)?;
248     accs.vaa.claim(ctx, accs.payer.key)?;
249
250     // Mint tokens
251     let mint_ix = spl_token::instruction::mint_to(
252         &spl_token::id(),
253         accs.mint.info().key,
254         accs.to.info().key,
255         accs.mint_authority.key,
256         &[],
257         accs.vaa
258             .amount
259             .as_u64()
260             .checked_sub(accs.vaa.fee.as_u64())
261             .unwrap(),
262     )?;
263     invoke_seeded(&mint_ix, ctx, &accs.mint_authority, None)?;
264
265     // Mint fees
266     let mint_ix = spl_token::instruction::mint_to(
267         &spl_token::id(),
268         accs.mint.info().key,
269         accs.to_fees.info().key,
270         accs.mint_authority.key,
271         &[],
272         accs.vaa.fee.as_u64(),
273     )?;
274     invoke_seeded(&mint_ix, ctx, &accs.mint_authority, None)?;
275
276     Ok(())
```

277
278

}

- Call Stack

1
2
3
4
5
6

```
fn entrypoint() { // /home/yifei/.cargo/registry/src/github.com-
↳ 1ecc6299db9ec823/solana-program-1.7.0/src/entrypoint.rs:46:9: 53:10
↳ }
fn instruction::solitaire() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 108:14 }
fn instruction::dispatch() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 99:14 }
fn instruction::CompleteWrapped::execute() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 74:22 }
fn api::complete_transfer::complete_wrapped() { //
↳ program/src/api/complete_transfer.rs:208:1: 277:2 }
```

- description:
- link:
- alleviation:

Issue: 6: CrossProgramInvocation

Category	Severity	Status
CrossProgramInvocation	Critical	UnResolved

- Location

program/src/api/transfer.rs

- Code Context

```
295 pub fn transfer_wrapped(  
296     ctx: &ExecutionContext,  
297     accs: &mut TransferWrapped,  
298     data: TransferWrappedData,  
299 ) -> Result<()> {  
300     // Prevent transferring to the same chain.  
301     if data.target_chain == CHAIN_ID_SOLANA {  
302         return Err(InvalidChain.into());  
303     }  
304  
305     // Verify that the from account is owned by the from_owner  
306     if &accs.from.owner != accs.from_owner.key {  
307         return Err(WrongAccountOwner.into());  
308     }  
309  
310     // Verify mints  
311     if accs.mint.info().key != &accs.from.mint {  
312         return Err(TokenBridgeError::InvalidMint.into());  
313     }  
314  
315     // Fee must be less than amount  
316     if data.fee > data.amount {  
317         return Err(InvalidFee.into());  
318     }  
319  
320     // Verify that meta is correct  
321     let derivation_data: WrappedMetaDerivationData = (&*accs).into();
```

```
322     accs.wrapped_meta
323         .verify_derivation(ctx.program_id, &derivation_data)?;
324
325     // Burn tokens
326     let burn_ix = spl_token::instruction::burn(
327         &spl_token::id(),
328         accs.from.info().key,
329         accs.mint.info().key,
330         accs.authority_signer.key,
331         &[],
332         data.amount,
333     )?;
334     invoke_seeded(&burn_ix, ctx, &accs.authority_signer, None)?;
335
336     // Pay fee
337     let transfer_ix = solana_program::system_instruction::transfer(
338         accs.payer.key,
339         accs.fee_collector.key,
340         accs.bridge.config.fee,
341     );
342
343     invoke(&transfer_ix, ctx.accounts)?;
344
345     // Post message
346     let payload = PayloadTransfer {
347         amount: U256::from(data.amount),
348         token_address: accs.wrapped_meta.token_address,
349         token_chain: accs.wrapped_meta.chain,
350         to: data.target_address,
351         to_chain: data.target_chain,
352         fee: U256::from(data.fee),
353     };
354     let params = (
355         bridge::instruction::Instruction::PostMessage,
356         PostMessageData {
357             nonce: data.nonce,
358             payload: payload.try_to_vec()?,
359             consistency_level: ConsistencyLevel::Finalized,
360         },
361     );
362
363     let ix = Instruction::new_with_bytes(
```



```

364     accs.config.wormhole_bridge,
365     params.try_to_vec()?.as_slice(),
366     vec![
367         AccountMeta::new(*accs.bridge.info().key, false),
368         AccountMeta::new(*accs.message.key, true),
369         AccountMeta::new_readonly(*accs.emitter.key, true),
370         AccountMeta::new(*accs.sequence.key, false),
371         AccountMeta::new(*accs.payer.key, true),
372         AccountMeta::new(*accs.fee_collector.key, false),
373         AccountMeta::new_readonly(*accs.clock.info().key, false),
374         AccountMeta::new_readonly(solana_program::system_program::id(),
375             ↪ false),
376         AccountMeta::new_readonly(solana_program::sysvar::rent::ID,
377             ↪ false),
378     ],
379 );
380 invoke_seeded(&ix, ctx, &accs.emitter, None)?;
381
382 Ok(())
383 }

```

• Call Stack

```

1  fn entrypoint() { // /home/yifei/.cargo/registry/src/github.com-
    ↪ 1ecc6299db9ec823/solana-program-1.7.0/src/entrypoint.rs:46:9: 53:10
    ↪ }
2  fn instruction::solitaire() { //
    ↪ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
    ↪ 108:14 }
3  fn instruction::dispatch() { //
    ↪ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
    ↪ 99:14 }
4  fn instruction::TransferWrapped::execute() { //
    ↪ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
    ↪ 74:22 }
5  fn api::transfer::transfer_wrapped() { //
    ↪ program/src/api/transfer.rs:295:1: 381:2 }
6

```

• description:

- link:
- alleviation:

Issue: 7: CrossProgramInvocation

Category	Severity	Status
CrossProgramInvocation	Critical	UnResolved

- Location

program/src/api/transfer.rs

- Code Context

```
127 pub fn transfer_native(  
128     ctx: &ExecutionContext,  
129     accs: &mut TransferNative,  
130     data: TransferNativeData,  
131 ) -> Result<()> {  
132     // Prevent transferring to the same chain.  
133     if data.target_chain == CHAIN_ID_SOLANA {  
134         return Err(InvalidChain.into());  
135     }  
136  
137     // Verify that the custody account is derived correctly  
138     let derivation_data: CustodyAccountDerivationData = (&*accs).into();  
139     accs.custody  
140         .verify_derivation(ctx.program_id, &derivation_data)?;  
141  
142     // Verify mints  
143     if accs.from.mint != *accs.mint.info().key {  
144         return Err(TokenBridgeError::InvalidMint.into());  
145     }  
146  
147     // Fee must be less than amount  
148     if data.fee > data.amount {  
149         return Err(InvalidFee.into());  
150     }  
151  
152     // Verify that the token is not a wrapped token  
153     if let COption::Some(mint_authority) = accs.mint.mint_authority {
```

```
154         if mint_authority == MintSigner::key(None, ctx.program_id) {
155             return Err(TokenBridgeError::TokenNotNative.into());
156         }
157     }
158
159     if !accs.custody.is_initialized() {
160         accs.custody
161             .create(&(&*accs).into(), ctx, accs.payer.key, Exempt)?;
162
163         let init_ix = spl_token::instruction::initialize_account(
164             &spl_token::id(),
165             accs.custody.info().key,
166             accs.mint.info().key,
167             accs.custody_signer.key,
168         )?;
169         invoke_signed(&init_ix, ctx.accounts, &[])?;
170     }
171
172     let trunc_divisor = 10u64.pow(8.max(accs.mint.decimals as u32) - 8);
173     // Truncate to 8 decimals
174     let amount: u64 = data.amount / trunc_divisor;
175     let fee: u64 = data.fee / trunc_divisor;
176     // Untruncate the amount to drop the remainder so we don't "burn"
177     // ↪ user's funds.
178     let amount_trunc: u64 = amount * trunc_divisor;
179
180     // Transfer tokens
181     let transfer_ix = spl_token::instruction::transfer(
182         &spl_token::id(),
183         accs.from.info().key,
184         accs.custody.info().key,
185         accs.authority_signer.key,
186         &[],
187         amount_trunc,
188     )?;
189     invoke_seeded(&transfer_ix, ctx, &accs.authority_signer, None)?;
190
191     // Pay fee
192     let transfer_ix = solana_program::system_instruction::transfer(
193         accs.payer.key,
194         accs.fee_collector.key,
195         accs.bridge.config.fee,
```

```
195     );
196     invoke(&transfer_ix, ctx.accounts)?;
197
198     // Post message
199     let payload = PayloadTransfer {
200         amount: U256::from(amount),
201         token_address: accs.mint.info().key.to_bytes(),
202         token_chain: CHAIN_ID_SOLANA,
203         to: data.target_address,
204         to_chain: data.target_chain,
205         fee: U256::from(fee),
206     };
207     let params = (
208         bridge::instruction::Instruction::PostMessage,
209         PostMessageData {
210             nonce: data.nonce,
211             payload: payload.try_to_vec()?,
212             consistency_level: ConsistencyLevel::Finalized,
213         },
214     );
215
216     let ix = Instruction::new_with_bytes(
217         accs.config.wormhole_bridge,
218         params.try_to_vec()?.as_slice(),
219         vec![
220             AccountMeta::new(*accs.bridge.info().key, false),
221             AccountMeta::new(*accs.message.key, true),
222             AccountMeta::new_readonly(*accs.emitter.key, true),
223             AccountMeta::new(*accs.sequence.key, false),
224             AccountMeta::new(*accs.payer.key, true),
225             AccountMeta::new(*accs.fee_collector.key, false),
226             AccountMeta::new_readonly(*accs.clock.info().key, false),
227             AccountMeta::new_readonly(solana_program::system_program::id(),
228                 ↪ false),
229             AccountMeta::new_readonly(solana_program::sysvar::rent::ID,
230                 ↪ false),
231         ],
232     );
233     invoke_seeded(&ix, ctx, &accs.emitter, None)?;
234
235     Ok(())
236 }
```

235

- Call Stack

1
2
3
4
5
6

```
fn entrypoint() { // /home/yifei/.cargo/registry/src/github.com-
↳ 1ecc6299db9ec823/solana-program-1.7.0/src/entrypoint.rs:46:9: 53:10
↳ }
fn instruction::solitaire() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 108:14 }
fn instruction::dispatch() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 99:14 }
fn instruction::TransferNative::execute() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 74:22 }
fn api::transfer::transfer_native() { //
↳ program/src/api/transfer.rs:127:1: 234:2 }
```

- description:
- link:
- alleviation:

Issue: 8: CrossProgramInvocation

Category	Severity	Status
CrossProgramInvocation	Critical	UnResolved

- Location

program/src/api/transfer.rs

- Code Context

```
127 pub fn transfer_native(  
128     ctx: &ExecutionContext,  
129     accs: &mut TransferNative,  
130     data: TransferNativeData,  
131 ) -> Result<()> {  
132     // Prevent transferring to the same chain.  
133     if data.target_chain == CHAIN_ID_SOLANA {  
134         return Err(InvalidChain.into());  
135     }  
136  
137     // Verify that the custody account is derived correctly  
138     let derivation_data: CustodyAccountDerivationData = (&*accs).into();  
139     accs.custody  
140         .verify_derivation(ctx.program_id, &derivation_data)?;  
141  
142     // Verify mints  
143     if accs.from.mint != *accs.mint.info().key {  
144         return Err(TokenBridgeError::InvalidMint.into());  
145     }  
146  
147     // Fee must be less than amount  
148     if data.fee > data.amount {  
149         return Err(InvalidFee.into());  
150     }  
151  
152     // Verify that the token is not a wrapped token  
153     if let COption::Some(mint_authority) = accs.mint.mint_authority {
```

```
154         if mint_authority == MintSigner::key(None, ctx.program_id) {
155             return Err(TokenBridgeError::TokenNotNative.into());
156         }
157     }
158
159     if !accs.custody.is_initialized() {
160         accs.custody
161             .create(&(&*accs).into(), ctx, accs.payer.key, Exempt)?;
162
163         let init_ix = spl_token::instruction::initialize_account(
164             &spl_token::id(),
165             accs.custody.info().key,
166             accs.mint.info().key,
167             accs.custody_signer.key,
168         )?;
169         invoke_signed(&init_ix, ctx.accounts, &[])?;
170     }
171
172     let trunc_divisor = 10u64.pow(8.max(accs.mint.decimals as u32) - 8);
173     // Truncate to 8 decimals
174     let amount: u64 = data.amount / trunc_divisor;
175     let fee: u64 = data.fee / trunc_divisor;
176     // Untruncate the amount to drop the remainder so we don't "burn"
177     // ↪ user's funds.
178     let amount_trunc: u64 = amount * trunc_divisor;
179
180     // Transfer tokens
181     let transfer_ix = spl_token::instruction::transfer(
182         &spl_token::id(),
183         accs.from.info().key,
184         accs.custody.info().key,
185         accs.authority_signer.key,
186         &[],
187         amount_trunc,
188     )?;
189     invoke_seeded(&transfer_ix, ctx, &accs.authority_signer, None)?;
190
191     // Pay fee
192     let transfer_ix = solana_program::system_instruction::transfer(
193         accs.payer.key,
194         accs.fee_collector.key,
195         accs.bridge.config.fee,
```



```
195     );
196     invoke(&transfer_ix, ctx.accounts)?;
197
198     // Post message
199     let payload = PayloadTransfer {
200         amount: U256::from(amount),
201         token_address: accs.mint.info().key.to_bytes(),
202         token_chain: CHAIN_ID_SOLANA,
203         to: data.target_address,
204         to_chain: data.target_chain,
205         fee: U256::from(fee),
206     };
207     let params = (
208         bridge::instruction::Instruction::PostMessage,
209         PostMessageData {
210             nonce: data.nonce,
211             payload: payload.try_to_vec()?,
212             consistency_level: ConsistencyLevel::Finalized,
213         },
214     );
215
216     let ix = Instruction::new_with_bytes(
217         accs.config.wormhole_bridge,
218         params.try_to_vec()?.as_slice(),
219         vec![
220             AccountMeta::new(*accs.bridge.info().key, false),
221             AccountMeta::new(*accs.message.key, true),
222             AccountMeta::new_readonly(*accs.emitter.key, true),
223             AccountMeta::new(*accs.sequence.key, false),
224             AccountMeta::new(*accs.payer.key, true),
225             AccountMeta::new(*accs.fee_collector.key, false),
226             AccountMeta::new_readonly(*accs.clock.info().key, false),
227             AccountMeta::new_readonly(solana_program::system_program::id(),
228                 ↪ false),
229             AccountMeta::new_readonly(solana_program::sysvar::rent::ID,
230                 ↪ false),
231         ],
232     );
233     invoke_seeded(&ix, ctx, &accs.emitter, None)?;
234
235     Ok(())
236 }
```

235

- Call Stack

1
2
3
4
5
6

```
fn entrypoint() { // /home/yifei/.cargo/registry/src/github.com-
↳ 1ecc6299db9ec823/solana-program-1.7.0/src/entrypoint.rs:46:9: 53:10
↳ }
fn instruction::solitaire() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 108:14 }
fn instruction::dispatch() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 99:14 }
fn instruction::TransferNative::execute() { //
↳ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
↳ 74:22 }
fn api::transfer::transfer_native() { //
↳ program/src/api/transfer.rs:127:1: 234:2 }
```

- description:
- link:
- alleviation:

Issue: 9: CrossProgramInvocation

Category	Severity	Status
CrossProgramInvocation	Critical	UnResolved

- Location

```
program/src/api/create_wrapped.rs
```

- Code Context

```
140 pub fn create_accounts(  
141     ctx: &ExecutionContext,  
142     accs: &mut CreateWrapped,  
143     data: CreateWrappedData,  
144 ) -> Result<()> {  
145     // Create mint account  
146     accs.mint  
147         .create(&((&*accs).into()), ctx, accs.payer.key, Exempt)?;  
148  
149     // Initialize mint  
150     let init_ix = spl_token::instruction::initialize_mint(  
151         &spl_token::id(),  
152         accs.mint.info().key,  
153         accs.mint_authority.key,  
154         None,  
155         min(8, accs.vaa.decimals), // Limit to 8 decimals, truncation is  
156     ↪ handled on the other side  
157     )?;  
158     invoke_signed(&init_ix, ctx.accounts, &[])?;  
159  
160     // Create meta account  
161     accs.meta  
162         .create(&((&*accs).into()), ctx, accs.payer.key, Exempt)?;  
163  
164     // Initialize spl meta  
165     accs.spl_metadata.verify_derivation(  
166         &spl_token_metadata::id(),
```

```

166         &SplTokenMetaDerivationData {
167             mint: *accs.mint.info().key,
168         },
169     )?;
170
171     // Normalize Token Metadata.
172     let name = truncate_utf8(&accs.vaa.name, 32 - 11) + " (Wormhole)";
173     let symbol = truncate_utf8(&accs.vaa.symbol, 10);
174
175     let spl_token_metadata_ix =
176     ↪ spl_token_metadata::instruction::create_metadata_accounts(
177         spl_token_metadata::id(),
178         *accs.spl_metadata.key,
179         *accs.mint.info().key,
180         *accs.mint_authority.info().key,
181         *accs.payer.info().key,
182         *accs.mint_authority.info().key,
183         name,
184         symbol,
185         String::from(""),
186         None,
187         0,
188         false,
189         true,
190     );
191     invoke_seeded(&spl_token_metadata_ix, ctx, &accs.mint_authority,
192     ↪ None)?;
193
194     // Populate meta account
195     accs.meta.chain = accs.vaa.token_chain;
196     accs.meta.token_address = accs.vaa.token_address;
197     accs.meta.original_decimals = accs.vaa.decimals;
198
199     Ok(())
200 }

```

- Call Stack

```

1 fn entrypoint(){// /home/yifei/.cargo/registry/src/github.com-
  ↪ 1ecc6299db9ec823/solana-program-1.7.0/src/entrypoint.rs:46:9: 53:10
  ↪ }

```

2
3
4
5
6
7

```
fn instruction::solitaire() { //
    ↪ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
    ↪ 108:14 }

fn instruction::dispatch() { //
    ↪ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
    ↪ 99:14 }

fn instruction::CreateWrapped::execute() { //
    ↪ /home/yifei/open/vrust/examples2/wormhole/solana/solitaire/program/src/macros.rs
    ↪ 74:22 }

fn api::create_wrapped::create_wrapped() { //
    ↪ program/src/api/create_wrapped.rs:108:1: 138:2 }

fn api::create_wrapped::create_accounts() { //
    ↪ program/src/api/create_wrapped.rs:140:1: 198:2
    ↪ }
```

- description:
- link:
- alleviation:

Appendix

Copied from <https://leaderboard.certik.io/projects/aave>

Finding Categories

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The “Checksum” field in the “Audit Scope” section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum” command against the target file.

Disclaimer

Copied from <https://leaderboard.certik.io/projects/aave>

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.