IRONNODE

ironnode.io

FINAL SECURITY ASSESSMENT

JurassicFi

Solana Program Security Audit

by **IronNode** from **Jul 30 - Aug 11, 2026**

ON-CHAIN PROGRAM REVIEW / REMEDIATION VERIFICATION

01

SECTION 01

Executive overview

1 Executive Overview

1.1 Introduction

JurassicFi operates a token sale program on Solana. The program is written with the Anchor framework. One deployment holds many independent sales, and each sale is called a launch.

A launch has a fixed lifecycle. A single authorized administrator creates the launch. The program mints the full token supply into a vault that the program owns, and the program then removes its own mint authority in the same transaction. Contributors send a payment token into a payment vault until the total reaches an exact raise cap. USDC is the intended payment token. When the total reaches the cap, a settlement instruction sends the proceeds to a configured escrow address and a configured treasury address, and the same instruction releases a treasury token allocation. Contributors then claim their tokens in proportion to their contributions. When the total does not reach the cap before the deadline, each contributor requests a refund.

The program contains five instructions, two account types, and four lifecycle states. The on-chain scope is 1,266 lines of Rust across 15 files.

JurassicFi requested a security audit of this program. IronNode performed the audit, reported the findings, and then verified the remediation work. This report contains the architecture summary, the findings with impact and exploitability, the disposition of each finding, and the actions that remain.

1.2 Findings and Remediations

IronNode identified 24 findings. The set contains no critical severity finding, 1 high severity finding, 7 medium severity findings, 9 low severity findings, and 7 informational findings.

IronNode found no method for any party to steal funds. Value leaves the program through three routes only. Each route is constrained either to configuration that is fixed at creation, or to an account that the signer owns. Every serious finding in this report describes value that becomes unavailable, and not value that moves to an attacker.

JurassicFi remediated 14 of the 24 findings. IronNode verified each remediation against the updated source, a rebuilt program binary, and on-chain execution. The high severity finding is closed, and IronNode confirmed the result with a replay of the original attack.

JurassicFi accepted the risk of 6 findings because the reported behavior is intentional or controlled through an operational procedure. JurassicFi acknowledged 4 findings as release, deployment, or governance work that remains before production. The client dispositions in this report reflect the written response received on Aug 12, 2026.

1.3 Audit Objectives

1. Fund Safety And Settlement Integrity

- Every route that moves value out of the program, and the constraint that protects each route
- The state machine of a launch, and the reachability of every terminal state
- The recovery options when a precondition of settlement is not true

2. Claim And Refund Accounting

- The proportional claim calculation, the rounding behaviour, and the conservation of the token pool
- The refund conditions, and the states in which a refund is available
- Replay protection for a claim and for a refund

3. Authorization And Account Substitution

- The administrator control on launch creation, and the effect of a compromise of that key
- Program derived address derivation, and the isolation between launches
- Substitution of a vault, a position, a mint, or a payout account

4. Configuration Validation

- The parameter checks that the launch instruction applies, and the incorrect configurations that pass those checks
- The sale window, the allocation splits, and the metadata limits
- The options to correct an incorrect configuration after creation

5. Token Program Behaviour And Build Integrity

- The assumptions that the program makes about the legacy SPL Token program
- The exclusion of Token-2022, transfer fee mints, and transfer hook mints
- Build reproducibility, the interface definition, and the deployment procedure

1.4 Scope

Audit Scope:

`https://github.com/solbound-dev/jurassic_fi_token_sale`

Branch: **main**

Audited commit: `df7a938e231c397f531ff5be688fdfe00195ae48`

Remediation commit: `7fa3f94182f496da7e83f984aeea60fef0cab359`

Program identifier: `9Nw7vbXaMHhUgtTqKqFZmqEjErPZ4tis6D1o3t4W9adW`

In scope: the on-chain Rust program (15 files, 1,266 lines), the generated interface definition, the compiled SBF binary, and the build and deployment configuration.

Out of scope: the client application, any backend service, and the economic design of a launch.

1.5 Methodology

IronNode applied static review, dynamic testing, and formal methods. Each method covers a different class of defect, and the results of each method were compared against the results of the other methods.

Manual review

IronNode read all 1,266 lines two times. The first pass established semantics. The second pass was adversarial. IronNode also reviewed 9,529 lines of macro expanded code, so that each conclusion describes the code that Anchor generates and not the code that the attributes appear to declare.

Artifact reconciliation

IronNode compared the generated interface definition against the source element by element. IronNode also examined the sections and the syscall imports of the compiled binary.

Adversarial testing

IronNode executed 30 targeted tests against real program execution on a local validator. The tests covered account substitution, authorization, boundary values, time windows, and replay attempts.

Differential fuzzing

IronNode executed 374 randomized transactions and compared every balance and every counter against an independent reference model of the intended value semantics.

Model checking and proofs

IronNode performed exhaustive bounded model checking of the full state machine, discharged two machine proofs over the complete 64 bit input range, and enumerated the claim arithmetic exhaustively over small domains.

Exploit development

IronNode built nine proof of concept exploits. Each exploit is reproducible against a local validator.

Remediation verification

After the remediation work, IronNode rebuilt the program, ran the unit tests and the project test suite, replayed the original high severity attack, and confirmed the new control paths on chain.

1.6 Assessment Summary & Findings Overview

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	7	9	7

14
FIXED AND VERIFIED

6
RISK ACCEPTED

4
ACKNOWLEDGED

Ref	Security analysis	Risk level	Disposition
IRON-1	A Payout Recipient Can Stop Settlement And Lock The Full Raise	HIGH	FIXED
IRON-2	The Proceeds Recipient Can Remove All Refund Rights At No Net Cost	MEDIUM	RISK ACCEPTED
IRON-3	Floor Division Can Pay Zero Tokens And Close The Position	MEDIUM	FIXED
IRON-4	Claim Order Decides Which Contributor Receives The Rounding Remainder	MEDIUM	FIXED
IRON-5	The Payment Mint Has No Constraint, So A Freeze Authority Can Stop Refunds	MEDIUM	FIXED
IRON-6	The End Timestamp Has No Upper Bound And No Relation To The Clock	MEDIUM	FIXED
IRON-7	One Transaction Can Take The Full Sale	MEDIUM	RISK ACCEPTED

Ref	Security analysis	Risk level	Disposition
IRON-8	The Client Supplies The Mint Keypair, Which Is A Separate Trust Domain	MEDIUM	RISK ACCEPTED
IRON-9	A Failed Launch Keeps The Full Token Supply	LOW	FIXED
IRON-10	Tokens Sent To A Vault Outside The Sale Flow Stay In The Vault	LOW	RISK ACCEPTED
IRON-11	A Contribution Can Decrease To A Very Small Amount	LOW	FIXED
IRON-12	A Launch Address As A Payout Route Holds The Allocation	LOW	RISK ACCEPTED
IRON-13	A Launch At The Cap Has No Exit Until The Payout Accounts Exist	LOW	FIXED
IRON-14	A Person Can Take The Payment Vault Address Before The Launch	LOW	RISK ACCEPTED
IRON-15	The Injected Interface Instructions Have No Authentication	LOW	FIXED
IRON-16	The Launch Configuration Has Too Few Checks And Cannot Change	LOW	FIXED

Ref	Security analysis	Risk level	Disposition
IRON-17	The Default Deploy Directory Contained An Incorrect Binary And Keypair	LOW	ACKNOWLEDGED
IRON-18	One Hardcoded Administrator And An Unreviewed Upgrade Authority	INFORMATIONAL	ACKNOWLEDGED
IRON-19	The Instruction Handlers Use The Same Symbol Name	INFORMATIONAL	FIXED
IRON-20	Unused Error Variants, Checks That Are Too Strict, And A Test That Cannot Run	INFORMATIONAL	FIXED
IRON-21	Code Quality Items	INFORMATIONAL	FIXED
IRON-22	The Lockfile Does Not Build With The Project Toolchain	INFORMATIONAL	FIXED
IRON-23	No Verifiable Build, So The Binary Has No Comparison Against The Source	INFORMATIONAL	ACKNOWLEDGED
IRON-24	No Deployment Exists For Review	INFORMATIONAL	ACKNOWLEDGED

FIXED means that IronNode verified the remediation against the updated source and confirmed the corrected behavior. RISK ACCEPTED means that JurassicFi intentionally retains the behavior or controls it with an operational procedure. ACKNOWLEDGED means that JurassicFi agrees with the finding and plans to address it through release, deployment, or governance work.

02

SECTION 02

Findings

2 Findings

2.1 A Payout Recipient Can Stop Settlement And Lock The Full Raise

HIGH FIXED IRON-1

The settlement instruction identifies each of its three payout destinations in two ways at the same time. It compares the account address against the associated token account address that is derived from the configured recipient and the mint. It also compares the `owner` field that is stored inside that account.

The program treats the `owner` field as a constant value. The field is not constant. In the legacy SPL Token program, the current owner of a token account can assign a new owner at any time with `SetAuthority(AccountOwner)`. The `InitializeImmutableOwner` extension does not prevent this, because the legacy program records the extension as a no-operation instead of an error. The program uses `Program<'info, Token>` and rejects Token-2022 by type. Therefore the program can only operate with the one token program in which payout account owners remain changeable.

When a recipient assigns a new owner to its own payout account, the two constraints cannot both be true again. The address constraint prevents a different account. The token program prevents the account from being closed and created again, because the new owner must sign the close instruction, and the associated token account program refuses a target that exists. The `CapReachedPendingSettlement` state has no refund path, so no fallback exists.

2.1.1 Code location

programs/jurassic_fi_token_sale/src/instructions/settle_success.rs:47–53

```
#[account(
    mut,
    constraint = escrow_payment_account.mint == payment_mint.key() @
TokenSaleError::InvalidPaymentMint,
    constraint = escrow_payment_account.owner == escrow_authority.key() @
TokenSaleError::InvalidEscrowPaymentAccount,
    constraint = escrow_payment_account.key() == derive_associated_token_address(...) @
TokenSaleError::InvalidEscrowPaymentAccount,
)]
pub escrow_payment_account: Box<Account<'info, TokenAccount>>,
```

programs/jurassic_fi_token_sale/src/utils/validation.rs:62–69

```
pub fn require_refundable_launch(launch: &Launch, now: i64) -> Result<()> {
    let refundable = matches!(launch.terminal_state, TerminalState::Failed)
        || (launch.terminal_state == TerminalState::Open
            && now > launch.end_ts
            && launch.total_raised < launch.raise_cap);
    require!(refundable, TokenSaleError::LaunchNotRefundable);
    Ok(())
}
```

2.1.2 Impact

One transaction from the escrow recipient or the treasury recipient makes the full raise and the full token supply permanently unavailable. Settlement returns `InvalidEscrowPaymentAccount`. Refunds return `LaunchNotRefundable`. Claims return `LaunchNotSuccessful`. The program has no instruction that recovers the funds.

The attack can be prepared before the sale opens. IronNode confirmed the attack with an empty payment vault. Contributors then filled the sale and entered the locked state without a warning. The treasury recipient has two independent triggers, because both its payment route and its token route use the same constraint pattern.

2.1.3 Recommendation

1. Remove the three `token_account.owner == authority` constraints from the settlement instruction. The constraints are not necessary. The associated token account address proves the route for that authority and that mint. The mint constraint proves the asset.
2. Add an exit from the `CapReachedPendingSettlement` state to the `Failed` state. Control the exit with the administrator, a timeout, or both. The exit makes refunds available again after any settlement failure.
3. Validate the payout routes when the launch is created.
4. Use a different escrow address and a different treasury address for each launch.

2.1.4 Remediation verification

JurassicFi removed all three owner constraints and added the `fail_pending_settlement` instruction. IronNode replayed the original attack against the new build. The recipient assigned a new owner to its payout account, the sale filled, and settlement completed with the state `Successful` and an empty payment vault.

2.2 The Proceeds Recipient Can Remove All Refund Rights At No Net Cost

MEDIUM RISK ACCEPTED IRON-2

A refund is the only exit that a contributor controls. The refund is available only when the launch is still `Open`, the end timestamp has passed, and the total raised amount is below the cap.

Any party can contribute the remaining amount of the cap. This action moves the launch to `CapReachedPendingSettlement`. The state machine has no transition back from that state, so the refund path is removed permanently.

Settlement is permissionless and has no delay. Therefore the party that receives the proceeds can contribute the remaining amount and settle in one atomic transaction. The amount that this party contributes returns to it during settlement.

2.2.1 Code location

programs/jurassic_fi_token_sale/src/instructions/contribute.rs:72, 116–118

```
let accepted_amount = requested_amount.min(remaining_cap);
// ...
if launch.total_raised == launch.raise_cap {
    launch.terminal_state = TerminalState::CapReachedPendingSettlement;
}
```

2.2.2 Impact

The party that receives the proceeds can complete a sale that is below the cap at no net cost. IronNode measured the result. An unrelated third party paid 600,000 base units for the shortfall and received 2,400,000 tokens, which is a purchase at the configured price. An administrator whose escrow address and treasury address were its own account paid the same 600,000 base units and finished with a gain of 400,000 base units. The gain is equal to the contribution of the other participant.

The capital is necessary for one transaction only. Contributors receive the published allocation, but they lose an exit that they expected to control.

2.2.3 Recommendation

1. Make the refund available after the end timestamp in all states, and not only in the `Open` state.
2. Permit settlement only after the end timestamp.
3. As an alternative, keep refunds available during a fixed delay after the launch reaches the cap.

2.2.4 Client disposition

> JurassicFi confirmed that this behavior is intentional. A contribution is a binding purchase commitment and not a refundable reservation. A refund is available only when the launch remains underfunded after the sale window.
>
> The protocol does not treat an affiliated wallet differently because on-chain logic cannot reliably infer the beneficial owner of a wallet. JurassicFi accepts this risk and relies on the off-chain escrow process to apply the full settled proceeds to the RWA purchase.

2.3 Floor Division Can Pay Zero Tokens And Close The Position

MEDIUM FIXED IRON-3

The claim instruction calculates a proportional amount. It multiplies the claimable token amount by the contribution, then divides by the total raised amount. The division discards the remainder.

The program does not compare the claimable token pool against the raise cap. When the pool is small in relation to the cap, the calculated amount for a small contributor is zero. The program then marks the position as claimed and increases the counters. The contributor receives no tokens, and the position is closed.

2.3.1 Code location

programs/jurassic_fi_token_sale/src/instructions/claim.rs:81–96

```
let proportional_claim = (u128::from(launch.claimable_token_amount))
    .checked_mul(u128::from(user_position.contributed_amount))?
    .checked_div(u128::from(launch.total_raised))?;   // remainder is discarded

// ...
if claim_amount > 0 {
    // transfer
}
user_position.claimed = true;                        // set even when claim_amount == 0
```

2.3.2 Impact

A contributor can pay the full contribution and receive no tokens. The position changes to the claimed state, so the contributor cannot try again. The launch is in the `Successful` state, so no refund is available. The result is a total loss for that contributor.

2.3.3 Recommendation

1. Reject a configuration in which the claimable token amount is too small for the raise cap.
2. Make the claim instruction fail when the calculated amount is zero, and do not change the position.
3. Record the remainder instead of discarding it.

2.3.4 Remediation verification

JurassicFi added `require_claimable_precision`, which rejects a launch when the claimable token amount is below the raise cap. This rule makes a zero payment impossible. The claim instruction also returns the new `ZeroClaimAmount` error as a second control.

2.4 Claim Order Decides Which Contributor Receives The Rounding Remainder

MEDIUM FIXED IRON-4

The claim instruction gives the full remaining pool to the last contributor that claims. This rule makes the pool close at exactly zero. The rule also gives the accumulated remainder of all earlier claims to one participant.

The participant that receives the remainder is decided by the claim order and not by the contribution. Contributors with equal contributions can therefore receive different amounts.

2.4.1 Code location

programs/jurassic_fi_token_sale/src/instructions/claim.rs:81–82

```
let claim_amount = if user_position.contributed_amount == remaining_contribution_total {
    remaining_claimable_token_total     // the last claimant receives all that remains
} else {
    /* floored pro rata */
};
```

2.4.2 Impact

The distribution is not equal for equal contributions. IronNode measured a difference of a factor of 2.22 for an identical contribution in a low precision configuration. Twelve equal contributions against a pool of 119 tokens with one decimal paid 9 tokens to eleven participants and 20 tokens to the twelfth participant. With the usual precision of six decimals, the measured difference was zero.

2.4.3 Recommendation

1. Send the remainder to a destination that is fixed in the configuration, and not to a contributor.
2. Keep the rule that makes the pool close at exactly zero.
3. Record the remainder in the launch account so that operators can audit the amount.

2.4.4 Remediation verification

JurassicFi added a `dust_token_account` that is fixed when the launch is created. The remainder now goes to that account. The claim order gives no advantage. The pool continues to close at exactly zero.

2.5 The Payment Mint Has No Constraint, So A Freeze Authority Can Stop Refunds

MEDIUM FIXED IRON-5

The program accepts the payment mint with no address constraint. The program does not examine the authorities of the mint. Therefore an operator can select any legacy SPL mint.

When the selected mint has a freeze authority, that authority can freeze the payment vault of the launch. The SPL Token program rejects a transfer when one side of the transfer is frozen. The sale program examines the state of the launch and not the state of the account, so the program continues to accept transactions that then fail.

2.5.1 Code location

```
programs/jurassic_fi_token_sale/src/instructions/initialize_launch.rs:66
pub payment_mint: Account<'info, Mint>,   // no address constraint, no authority check
```

2.5.2 Impact

An external freeze authority can stop all refunds and all settlements for a launch. The funds are not lost, but they are unavailable until the authority removes the freeze. The intended asset for this program is USDC, and USDC has a freeze authority.

2.5.3 Recommendation

1. Constrain the payment mint to an approved address, or to a list of approved addresses that only the administrator can change.
2. Reject a mint that has a freeze authority, or record the acceptance of that risk for each approved mint.
3. Continue to reject Token-2022 mints, which prevents transfer fee mints and transfer hook mints.

2.5.4 Remediation verification

JurassicFi added a `PaymentMintAllowlist` account. The launch instruction now requires the allowlist account for the mint, and only the administrator can create or remove an entry. Removal of an entry does not affect a launch that already exists, because no other instruction reads the allowlist.

2.6 The End Timestamp Has No Upper Bound And No Relation To The Clock

MEDIUM FIXED IRON-6

The only check on the sale window compares the start timestamp against the end timestamp. The program does not compare either value against the current clock, and the program does not limit the length of the window.

A refund requires the end timestamp to pass. Therefore an incorrect end timestamp makes the refund unavailable until that time. A value in milliseconds instead of seconds, which is a common client error, produces a window of approximately 55,000 years.

2.6.1 Code location

programs/jurassic_fi_token_sale/src/utils/validation.rs:18–21

```
pub fn require_timestamp_order(start_ts: i64, end_ts: i64) -> Result<()> {
    require!(start_ts < end_ts, TokenSaleError::InvalidTimeWindow);   // no clock comparison
    Ok(())
}
```

2.6.2 Impact

Contributor funds can become unavailable for a period that is longer than a human lifetime. The program has no instruction that corrects the timestamp and no instruction that closes the launch.

2.6.3 Recommendation

1. Compare the end timestamp against the current clock when the launch is created.
2. Apply a maximum window length, for example 90 days.
3. Add an exit to the failed state so that operators can recover a launch that is stopped for any reason.

2.6.4 Remediation verification

JurassicFi added `require_timestamp_window`. The function rejects an end timestamp that is at or before the current clock. The function also limits the window to 60 days, measured from the later of the start timestamp and the current clock.

2.7 One Transaction Can Take The Full Sale

MEDIUM RISK ACCEPTED IRON-7

The program applies no maximum contribution for one participant. The program reduces a request that is larger than the remaining cap, but the program does not reject the request.

The cap transition is immediate. Therefore one wallet can contribute the full cap in one transaction and then settle and claim in the same transaction. All other participants then receive `LaunchNotOpen`.

2.7.1 Code location

programs/jurassic_fi_token_sale/src/instructions/contribute.rs:52, 72

```
require_non_zero(requested_amount)?;              // no maximum
let accepted_amount = requested_amount.min(remaining_cap);
```

2.7.2 Impact

One participant can take the full token distribution and exclude all other participants. IronNode reproduced the result with a 714 byte transaction and 17 account keys, signed by a wallet with no special permission. The wallet paid 1,000,000 base units and received 4,000,000 tokens, which is the full claimable allocation. No participant loses funds. The distribution of the sale is destroyed.

2.7.3 Recommendation

1. Apply a maximum contribution for each position, either as a configured value or as a percentage of the cap.
2. Require the start timestamp to be in the future so that a launch cannot be captured in the slot in which it is created.
3. Add a delay between the cap transition and settlement.
4. As an alternative, add an allowlist for contributors.

2.7.4 Client disposition

> JurassicFi confirmed that first-come, first-served fractional ownership is the intended model. The protocol does not guarantee broad distribution, a per-wallet allocation, or an oversubscription window. A participant can buy all available allocation by paying the configured price.
>
> JurassicFi accepts this product-design risk. A per-contributor maximum, vesting, and a settlement delay will not be added unless the distribution requirements change. The first-come, first-served behavior will be stated in the participant documentation.

2.8 The Client Supplies The Mint Keypair, Which Is A Separate Trust Domain

MEDIUM RISK ACCEPTED IRON-8

The token mint uses Anchor `init` with no seeds. Therefore the mint account must sign the transaction that creates it, and the client must supply a keypair. The mint address is not derived from the program.

The holder of that keypair has two options before the launch instruction runs. The holder can create a plain system account at the address, which makes the address unusable forever. The holder can also create a mint at the address with its own authorities and its own metadata.

2.8.1 Code location

programs/jurassic_fi_token_sale/src/instructions/initialize_launch.rs:59–65

```
#[account(
    init,
    payer = admin,
    mint::decimals = params.decimals,
    mint::authority = launch,
)]
pub token_mint: Account<'info, Mint>,     // no seeds: the account itself must sign
```

2.8.2 Impact

A person that holds the mint keypair can make the intended token address permanently unusable. The rent of that account is not recoverable, because a legacy mint has no close instruction and the account data cannot become smaller.

The same person can also create a counterfeit token at the address that the project published. The counterfeit token can hold the same name, the same symbol, and the same URI, and the person keeps the mint authority.

2.8.3 Recommendation

1. Derive the token mint as a program derived address from the launch identifier. This change removes the keypair and the trust domain.
2. Until that change, generate the mint keypair in the same process that sends the launch transaction, and delete the keypair after the transaction is confirmed.
3. Do not publish a token address before the launch transaction is confirmed.

2.8.4 Client disposition

> JurassicFi accepts the operational key-custody risk because support for vanity mint addresses is an explicit product requirement. The mint keypair is treated as a launch-critical secret before initialization.
>
> The mint address will not be published before initialization is confirmed. Post-launch checks will confirm that the mint authority is revoked and that the metadata and other authorities match the launch configuration.

2.9 A Failed Launch Keeps The Full Token Supply

LOW FIXED IRON-9

The program mints the full token supply when the launch is created. Two instructions move tokens out of the token vault. Both instructions require the `Successful` state.

The `Failed` state has no path to the `Successful` state. Therefore the full supply stays in the vault after a failed launch, and the rent of the vault is also not recoverable.

2.9.1 Code location

```
programs/jurassic_fi_token_sale/src/instructions/refund.rs:49-100
// The first refund moves the launch to Failed. The token vault is not changed.
launch.terminal_state = TerminalState::Failed;
```

2.9.2 Impact

The full token supply of a failed launch is permanently unavailable. Contributors receive the full refund of their payments, so no contributor loses funds. The token supply data remains incorrect for all external systems that read the mint.

2.9.3 Recommendation

1. Burn the token vault balance when the launch moves to the failed state.
2. As an alternative, add an administrator instruction that moves the tokens to the treasury and closes the vault.
3. Close the launch account and the position accounts after all refunds are complete, and return the rent.

2.9.4 Remediation verification

JurassicFi burns the full token vault balance when the launch moves to the failed state. Both the refund instruction and the new `fail_pending_settlement` instruction perform the burn. IronNode measured a mint supply that changed from 5,000,000 to 0.

2.10 Tokens Sent To A Vault Outside The Sale Flow Stay In The Vault

LOW RISK ACCEPTED IRON-10

The two vaults are standard associated token accounts. Any party can transfer tokens to them at any time.

Settlement transfers the amounts that the configuration holds. Settlement does not read the balance of the vault. Refunds transfer the recorded contribution. Therefore no instruction moves a balance that the program did not record.

2.10.1 Code location

programs/jurassic_fi_token_sale/src/instructions/settle_success.rs:88-131

```
token::transfer(/* ... */, launch.escrow_usdc_amount)?;      // configured constant
token::transfer(/* ... */, launch.treasury_usdc_amount)?;    // configured constant
```

2.10.2 Impact

A payment that a contributor sends directly to the vault, instead of through the contribute instruction, is permanently unavailable. The program creates no position for that payment. This is a user error and not an attack, but the result is permanent.

2.10.3 Recommendation

1. Add an administrator instruction that transfers any balance above the recorded amount to the treasury after the launch reaches a terminal state.
2. State in the user documentation that a direct transfer to a vault address is not a contribution.

2.10.4 Client disposition

> JurassicFi accepts this as an operational risk. A direct SPL token transfer does not create a position, increase the total raised amount, change an allocation, or change a refund right. Unsolicited deposits therefore do not corrupt the sale accounting.
>
> JurassicFi chose not to sweep these balances because a sweep would give accidentally transferred user funds to the project. The participant documentation will state that a direct transfer to a vault is not a contribution and is not credited or refunded.

2.11 A Contribution Can Decrease To A Very Small Amount

LOW FIXED IRON-11

The contribute instruction reduces a request to the remaining amount of the cap. The instruction has no parameter for a minimum acceptable amount.

Therefore a participant that sends a large request can receive a very small position when other participants fill the cap first. The transaction is successful, and the participant pays the transaction fee and the rent of the position account.

2.11.1 Code location

```
programs/jurassic_fi_token_sale/src/instructions/contribute.rs:51, 66–84
let accepted_amount = requested_amount.min(remaining_cap);
// The instruction has no minimum acceptance parameter.
```

2.11.2 Impact

A participant can receive a position that is much smaller than the request. The participant pays approximately 0.0016 SOL of rent for the position account, and that rent is not recoverable. IronNode measured a request of 1,000,000 base units that produced a position of 1 base unit.

2.11.3 Recommendation

1. Add a minimum acceptable amount parameter to the contribute instruction, and reject the transaction when the reduced amount is below that value.
2. Return the rent of the position account when a refund completes.

2.11.4 Remediation verification

JurassicFi added the `min_accepted_amount` parameter. The instruction returns `AcceptedAmountBelowMinimum` when the reduced amount is below the value that the caller supplies.

2.12 A Launch Address As A Payout Route Holds The Allocation

LOW RISK ACCEPTED IRON-12

The program validates the escrow address and the treasury address against the default public key only. The program does not compare the addresses against the launch address or against each other.

When an operator configures the launch address as a payout route, settlement transfers the amount from a vault of the launch to another account of the same launch. The transfer is successful. No instruction moves the tokens out of that account again.

2.12.1 Code location

programs/jurassic_fi_token_sale/src/instructions/initialize_launch.rs:209–210

```
require_non_default_pubkey(params.escrow_address)?;      // only Pubkey::default() is
rejected
require_non_default_pubkey(params.treasury_address)?;
```

2.12.2 Impact

The affected allocation is permanently unavailable. Settlement reports success, so no error indicates the condition. IronNode confirmed the result on chain. Only the configured amount is affected, and contributor claims continue to operate.

2.12.3 Recommendation

1. Reject an escrow address or a treasury address that is equal to the launch address.
2. Reject a payout address that is equal to another payout address of the same launch, unless the operator states that this is intended.
3. Confirm that each payout destination is different from its source before each transfer.

2.12.4 Client disposition

> JurassicFi accepts this as a trusted-administrator configuration risk. The escrow and treasury routes are intentionally flexible launch parameters.
>
> The client position is that the program cannot reject every address that is semantically wrong while it preserves flexible payout configuration. Launch review procedures will control this risk.

2.13 A Launch At The Cap Has No Exit Until The Payout Accounts Exist

LOW FIXED IRON-13

Settlement requires all three payout destinations to exist as initialized token accounts. Settlement does not create them.

The cap transition removes the refund path immediately. Therefore a launch whose payout accounts do not exist has no exit. Settlement fails because the accounts are not initialized, and contributors cannot request a refund.

The treasury token account is the most probable account to be absent. That account is an associated token account for a mint that does not exist before the launch instruction runs.

2.13.1 Code location

```
programs/jurassic_fi_token_sale/src/instructions/settle_success.rs:53, 63, 70
pub escrow_payment_account: Box<Account<'info, TokenAccount>>,    // not init_if_needed
pub treasury_payment_account: Box<Account<'info, TokenAccount>>,  // not init_if_needed
pub treasury_token_account: Box<Account<'info, TokenAccount>>,    // not init_if_needed
```

2.13.2 Impact

A full raise has no exit during this period. No value is permanently lost. Any party can correct the condition, because the creation of an associated token account is permissionless and costs approximately 0.002 SOL.

2.13.3 Recommendation

1. Declare the three payout destinations with `init_if_needed` and an explicit payer, so that settlement creates an account that is absent.
2. Until that change, require the settlement procedure to include idempotent account creation instructions in the same transaction.
3. Add the exit from the pending settlement state that IRON-1 recommends.

2.13.4 Remediation verification

JurassicFi added the `fail_pending_settlement` instruction. The administrator can move a launch that is stopped in the pending settlement state to the failed state after the timeout, and contributors then receive refunds. IronNode confirmed the full sequence on chain.

2.14 A Person Can Take The Payment Vault Address Before The Launch

LOW RISK ACCEPTED IRON-14

The launch address is predictable. The administrator is a constant in the program, and the launch identifier is an integer.

The payment vault uses Anchor `init`, which sends the non-idempotent associated token account creation instruction. The creation needs no signature from the owner. Therefore any person can create the payment vault for a predictable launch address and an existing payment mint before the launch transaction. The launch instruction then fails.

This condition does not apply to the token vault in the reported form. The sale token mint is created during the launch transaction. The token mint does not exist before that transaction, so a third party cannot create its associated token account in advance.

2.14.1 Code location

programs/jurassic_fi_token_sale/src/instructions/initialize_launch.rs:67–82

```
#[account(init, payer = admin, associated_token::mint = payment_mint,
          associated_token::authority = launch, /* ... */)]
pub payment_vault: Account<'info, TokenAccount>,      // init, not init_if_needed
```

2.14.2 Impact

A person can prevent the creation of a launch for a selected launch identifier and payment mint. The cost is approximately 0.002 SOL for each account. No funds are at risk, and a different launch identifier operates correctly. When launch identifiers are sequential, a person can block a full schedule of future launches for approximately two SOL for one thousand launches.

2.14.3 Recommendation

1. Declare the payment vault with `init_if_needed`. This change selects the idempotent creation path. The change also enables the verification block for the mint, the authority, and the derived address.
2. Select launch identifiers as random 64 bit values instead of sequential values. This operational change needs no change to the program.

2.14.4 Client disposition

> JurassicFi accepts the limited payment-vault initialization risk. The token vault cannot be taken in advance because the sale token mint does not exist before the launch transaction. The payment vault can be taken only when the payment mint already exists.
>
> The accepted operational control is to select random 64 bit launch identifiers instead of sequential identifiers. This makes advance selection of the payment vault address impractical. JurassicFi can use `init_if_needed` later if additional hardening becomes necessary.

2.15 The Injected Interface Instructions Have No Authentication

LOW FIXED IRON-15

The Anchor `#[program]` attribute adds a group of interface definition instructions to the dispatcher of the program. The group uses a tag prefix and not one of the declared instruction identifiers. The group is present by default, because the crate declares the `no-idl` feature but does not select it.

The account validation for the creation handler confirms four items only. The payer signs. The target account is writable. The base account is the expected program derived address. The program account is executable. The handler does not compare the signer against the upgrade authority of the program, and the handler then records the signer as the authority of the interface definition account.

2.15.1 Code location

programs/jurassic_fi_token_sale/Cargo.toml:16

```
[features]
default = []          # the no-idl feature exists but is not selected
```

2.15.2 Impact

No funds are at risk. The first signer becomes the permanent owner of the interface definition account, and the legitimate operator can never replace or close that account. The affected asset is the address that block explorers and client tools read to build and decode transactions for this program. The result is a risk to tool integrity and a possible aid to a phishing operation.

2.15.3 Recommendation

1. Publish the interface definition in the same operation that deploys the program, so that no period exists in which another party can claim it.
2. Confirm the authority of the resulting account immediately after publication.
3. As an alternative, build the program with the `no-idl` feature so that the instruction group is not present, or set the authority to the erased value after publication.

2.15.4 Remediation verification

JurassicFi made `no-idl` a default feature, so the instruction group is not compiled into the program. IronNode confirmed the change in the binary. The `anchor:idl` seed is present one time in the audited binary and zero times in the rebuilt binary.

2.16 The Launch Configuration Has Too Few Checks And Cannot Change

LOW FIXED IRON-16

Parameter validation covers the timestamp order, non-zero fields, the two allocation sums, non-default payout addresses, and string lengths. Several checks that prevent permanent errors are absent.

The program does not limit the token decimal count. A launch can therefore create a token with a very large decimal count. This program operates correctly with that token, but almost all external market makers, pricing services, and display integrations cannot process it.

The local string limits of 64, 16, and 256 bytes are larger than the Metaplex limits of 32, 10, and 200 bytes. A value with an intermediate length passes the local check and then fails inside the metadata cross-program invocation after approximately 108,000 compute units. The program also applies no minimum window length.

2.16.1 Code location

programs/jurassic_fi_token_sale/src/constants.rs:12–14

```
pub const MAX_TOKEN_NAME_LEN: usize = 64;     // Metaplex enforces 32
pub const MAX_TOKEN_SYMBOL_LEN: usize = 16;   // Metaplex enforces 10
pub const MAX_TOKEN_URI_LEN: usize = 256;     // Metaplex enforces 200
```

2.16.2 Impact

An operator error becomes permanent. Each individual gap needs an administrator action and gives no path to an attacker. The combination means that an incorrect launch cannot be corrected and cannot be removed. Several of these gaps are also the conditions that other findings in this report need.

2.16.3 Recommendation

1. Align the metadata string limits with the Metaplex limits.
2. Apply a maximum token decimal count. Nine is the usual maximum for Solana tokens.
3. Apply a minimum window length so that a launch cannot have a window that is shorter than one slot.
4. Add administrator instructions that close a launch in a terminal state and return the rent.

2.16.4 Remediation verification

JurassicFi aligned the string limits with the Metaplex limits of 32, 10, and 200 bytes. The program now limits the decimal count to nine and returns `InvalidTokenDecimals`. The sale window now has an upper bound, and the new exit from the pending settlement state gives a recovery path.

2.17 The Default Deploy Directory Contained An Incorrect Binary And Keypair

LOW ACKNOWLEDGED IRON-17

During the review, IronNode found two incorrect files in the default deployment directory. Version control excludes this directory, so a routine status check does not show its contents.

The first file was a compiled binary that was identical to an instrumented audit build. That build replaces the launch administrator with a test key. The second file was a program keypair whose public key did not match the program identifier in the source.

The Anchor deployment command uses the files that exist in that directory and does not build the program again. The audit harness build produced both files, because the build must write through the default output directory. IronNode deleted both files immediately.

2.17.1 Code location

```
target/deploy/  (excluded by .gitignore:6)

jurassic_fi_token_sale.so        # instrumented audit build, not the pristine program
jurassic_fi_token_sale-keypair.json   # public key does not match declare_id!
```

2.17.2 Impact

No deployment occurred, and the condition is corrected, so no technical exposure remains. The process gap is real and can occur again. An excluded directory can hold a binary that is ready to deploy, and no source review or version control check shows it. This repository has no build provenance check and no artifact hash check.

2.17.3 Recommendation

1. Do not deploy from a developer working directory.
2. Produce the release binary from a clean checkout with a verifiable build, and compare its hash against an approved value.
3. Supply the program keypair through a separate channel, and not from the build output directory.
4. Confirm that the program identifier in the source, the identifier in `Anchor.toml`, the address in the interface definition, and the public key of the deployment keypair are all equal. Implement this check as an automated release gate.

2.17.4 Client response

> JurassicFi acknowledged the release-process gap. Future releases will start from a clean checkout and will be rebuilt immediately before deployment.
>
> The release procedure will confirm that the source program identifier, the `Anchor.toml` identifier, the generated interface address, and the deployment keypair public key are equal.

2.18 One Hardcoded Administrator And An Unreviewed Upgrade Authority

INFORMATIONAL ACKNOWLEDGED IRON-18

The authorized launch administrator is a constant in the program. IronNode confirmed the value in the compiled bytecode. The constant is the only control on the creation of a launch. A change to the administrator needs a new build and a new deployment.

The upgrade authority of the program can replace all program logic. That action can make every vault of every launch available to one party in one transaction. The configuration declares a local network program entry only, so no deployment, loader, program data account, or upgrade authority exists for review.

2.18.1 Code location

programs/jurassic_fi_token_sale/src/constants.rs:3

```
pub const AUTHORIZED_LAUNCH_ADMIN: Pubkey =
    pubkey!("F1KYkkZJ9J7xmv3Mx7YNKV8Jk7x6SDyw9dYxaKssWUD7");
```

2.18.2 Impact

This is intended centralization and not a defect. IronNode records it because it is the largest remaining risk in the system, and because the interface definition of the program does not show it. The upgrade authority is the only control that can cause protocol wide loss in one transaction.

A compromised administrator key cannot extract value from a launch that already exists, because no instruction permits a change after creation.

2.18.3 Recommendation

1. Hold the administrator key and the upgrade authority in a multiple signature wallet.
2. Add a timelock to the upgrade authority so that a pending change is visible to the public before it becomes active.
3. Record a rotation procedure for the administrator key.
4. Publish a schedule to remove the upgrade authority or to transfer it to immutable governance.
5. Use a different escrow address and treasury address for each launch.

2.18.4 Client response

> JurassicFi acknowledged the centralized authority risk and confirmed that a multisignature wallet will hold the relevant authority.

2.19 The Instruction Handlers Use The Same Symbol Name

INFORMATIONAL FIXED IRON-19

Each instruction module declared a function with the name `handler`. The instruction module then re-exported all of these modules with a glob export, and the crate root re-exported that module again.

The result is a set of definitions with the same name in one namespace. The current code is correct, because each call uses the full module path. A future call that uses the short name would select a handler that the author did not intend.

2.19.1 Code location

```
programs/jurassic_fi_token_sale/src/instructions/mod.rs
pub use claim::*;          // each module exports a function named `handler`
pub use contribute::*;
pub use initialize_launch::*;
pub use refund::*;
pub use settle_success::*;
```

2.19.2 Impact

No effect on the current program. The condition is a risk for future maintenance, because a handler with the wrong name can pass a review.

2.19.3 Recommendation

1. Give each handler a unique name that states the instruction, for example `handle_claim`.
2. As an alternative, remove the glob re-exports and export only the types that Anchor needs.
3. Treat ambiguous name warnings as errors in continuous integration.

2.19.4 Remediation verification

JurassicFi renamed each handler to a unique `handle_*` name. The glob re-export no longer produces definitions with the same name.

2.20 Unused Error Variants, Checks That Are Too Strict, And A Test That Cannot Run

INFORMATIONAL FIXED IRON-20

The program declares two error variants that no code constructs. The variants occupy positions in the generated interface definition. A later removal would change the number of every following error and would break the error mapping of every client.

Validation rejects a zero token decimal count, a zero launch identifier, and zero treasury allocations. All three are valid configurations that the program cannot express.

The only multiple party rounding test in the repository cannot run. The test sets the token decimal count to zero, and validation rejects that value before the test reaches any claim assertion. Therefore the project has no active test coverage of claim rounding. The program also emits no events, so an indexer must compare account state to find activity.

2.20.1 Code location

programs/jurassic_fi_token_sale/src/errors.rs and utils/validation.rs:13–15

```
NotImplemented,        // never constructed
InvalidLaunchState,    // never constructed

pub fn require_non_zero_u8(value: u8) -> Result<()> {
    require!(value > 0, TokenSaleError::ZeroValueNotAllowed);   // rejects decimals = 0
    Ok(())
}
```

2.20.2 Impact

No direct security effect. The test that cannot run is the reason that the project did not find the two medium severity claim accounting findings in this report.

2.20.3 Recommendation

1. Remove the unused error variants before the first mainnet deployment, while the renumbering has no cost.
2. Remove the zero value restrictions for the configurations that are valid.
3. Correct the rounding test to use a non-zero decimal count so that its assertions run.
4. Emit events for contributions, settlement, claims, and refunds.
5. Add dependency scanning to continuous integration.

2.20.4 Remediation verification

JurassicFi removed both unused variants, removed the zero value restrictions on the launch identifier, the decimal count, and the treasury allocations, corrected the rounding test to use one decimal, and added four events. Dependency scanning is listed in Section 3.

2.21 Code Quality Items

INFORMATIONAL FIXED IRON-21

The contribute instruction sends its token transfer before it validates the position flags. The current code is safe, because a Solana transaction is atomic. A future change that adds an effect that cannot be reversed before those checks would become a defect.

The administrator authorization check is the last constraint that Anchor evaluates and not the first. This is a result of the two pass code generation of Anchor. The check runs after the program creates the launch account, the mint, and both vaults. The condition is not exploitable, but a reader who expects the check to run first can reach an incorrect conclusion.

The settlement instruction marks the token mint as mutable, and no code writes it. This marking locks the account for writing without a reason. The repository also contains no README, no architecture note, and no deployment procedure.

2.21.1 Code location

```
programs/jurassic_fi_token_sale/src/instructions/contribute.rs:74–104
token::transfer(/* ... */)?;          // transfer runs first
// ...
require_open_position(user_position)?;   // validation runs after the transfer
```

2.21.2 Impact

No current effect. Each item either has no result, or the item depends on a property that is true today and that the code does not enforce.

2.21.3 Recommendation

1. Validate all inputs before any state change and before any cross-program invocation.
2. Remove constraints that repeat another constraint, so that each error code has one cause.
3. Remove mutability markers that no code needs.
4. Add documentation for the launch lifecycle, the purpose of the escrow address and the treasury address, the intended payment mint, and the deployment procedure.

2.21.4 Remediation verification

JurassicFi moved the transfer in the contribute instruction after the position validation, and removed the unnecessary mutability marker from the token mint in the settlement instruction. The remaining items are listed in Section 3.

2.22 The Lockfile Does Not Build With The Project Toolchain

INFORMATIONAL FIXED IRON-22

A build of the program fails at dependency resolution with the Solana command line tools that the project targets. The bundled platform tools contain a Cargo release that is older than the 2024 edition, and the committed lockfile needs that edition through a transitive dependency.

A successful build needs platform tools version 1.52 or later. Interface definition generation separately needs a newer host compiler. The repository pins neither. The Anchor configuration declares an Anchor version and no Solana version, and no Rust toolchain file exists.

2.22.1 Code location

Anchor.toml:1–2 and Cargo.lock

```
[toolchain]
anchor_version = "0.32.1"     # no solana_version, and no rust-toolchain.toml exists
```

2.22.2 Impact

The build is not reproducible, which is a supply chain risk. Two engineers with nominally identical toolchains can produce different results, and one of them cannot build the program at all.

2.22.3 Recommendation

1. Pin the Solana version beside the Anchor version in the toolchain section of the Anchor configuration.
2. Add a Rust toolchain file that declares the minimum supported compiler version.
3. Build with the locked flag so that dependency resolution cannot change.
4. Run the documented build command in continuous integration from a clean checkout.

2.22.4 Remediation verification

JurassicFi added `solana_version` to the Anchor configuration, added a `rust-toolchain.toml` file, and added the locked flag to the build scripts. Section 3 records the verification step that remains.

2.23 No Verifiable Build, So The Binary Has No Comparison Against The Source

INFORMATIONAL ACKNOWLEDGED IRON-23

Docker was not available in the review environment. Therefore IronNode could not run the verifiable build command and the verification command.

All artifact hashes that this review records come from a native build. Those hashes are valid for that host only.

2.23.1 Code location

```
Not applicable. This is an environment and process gap.
```

2.23.2 Impact

The relation between the reviewed source and any binary that JurassicFi deploys later is not confirmed. This is a prerequisite for release and not a defect in the program.

2.23.3 Recommendation

1. Produce the deployment binary with a deterministic build in a container.
2. Publish the hash of that binary.
3. Compare the deployed bytecode against the source with the verification tool before the program is considered live, and repeat the comparison after each upgrade.
4. Record the published hash in the repository so that other parties can confirm the program that operates.

2.23.4 Client response

> JurassicFi acknowledged this release requirement. Before a production deployment, a clean deterministic environment will build the program with the pinned Anchor, Solana, and Rust versions.
>
> The release gate will record the binary hash and compare the deployed bytecode against the reviewed source. The same process will apply to each future upgrade.

2.24 No Deployment Exists For Review

INFORMATIONAL ACKNOWLEDGED IRON-24

The Anchor configuration defines a local network program entry only, and it selects the local cluster. Therefore the configuration records no devnet identifier and no mainnet identifier.

IronNode confirmed that the declared program identifier exists on neither mainnet nor devnet. Therefore the loader type, the program data account, the deployment slot, the upgrade authority, and the on-chain interface definition were not available for examination.

2.24.1 Code location

Anchor.toml:8–9, 14–16

```
[programs.localnet]
jurassic_fi_token_sale = "9Nw7vbXaMHhUgtTqKqFZmqEjErPZ4tis6D1o3t4W9adW"

[provider]
cluster = "Localnet"
```

2.24.2 Impact

The upgrade authority remains a trust boundary that no review covered. That authority can replace all program logic. No verification after deployment was possible during this review.

2.24.3 Recommendation

1. Record the intended deployment network and program identifier in the configuration before release.
2. After deployment, record the loader type, the program data account, the deployment slot, and the upgrade authority, and review them.
3. Compare the on-chain interface definition against the definition that the source generates.
4. Request a follow-up review of the deployed state when these artifacts exist.

2.24.4 Client response

> JurassicFi acknowledged this deployment-readiness item. The deployment target and program identifier will be recorded before release. After deployment, the loader type, program data account, deployment slot, upgrade authority, and bytecode hash will be captured and reviewed.
>
> An on-chain interface definition is optional for this program. If JurassicFi publishes one, the deployed definition and its authority will be compared against the definition that the reviewed source generates.

03

SECTION 03

Remaining recommendations

3 Remaining Recommendations

The findings in this section are closed. Each finding contained more than one item, and JurassicFi remediated the item that carried the security risk. The actions below are the items that remain. None of them is exploitable. Each one reduces the probability of a future defect, or improves the assurance that a release provides.

3.1 Document Atomic Payout Account Creation

FROM IRON-13

JurassicFi added an exit from the pending settlement state, and that exit removes the permanent condition. The cause remains. The three payout destinations in the settlement instruction are still plain accounts and not `init_if_needed` accounts.

JurassicFi accepted the client-side control. A settlement transaction will include idempotent associated token account creation instructions for all payout routes before `settle_success`. The instructions are harmless when the accounts exist and create the accounts atomically when they do not exist.

1. Add the atomic account-creation sequence to the settlement runbook.
2. Add an integration test that settles a launch when all three payout accounts are absent before the transaction.

3.2 Document Trusted-Administrator Configuration Decisions

FROM IRON-16

The program now limits the maximum sale window and rejects an end timestamp that has passed. The program does not apply a minimum window length. A sale of one second is therefore valid, and the clock can pass the window between two consecutive slots. Contribution is then not possible.

JurassicFi accepts short windows, immutable configuration, and unrecoverable legacy mint rent as trusted-administrator configuration risks. The fixed configuration is intentional because contributors must be able to rely on terms that cannot change after launch creation.

1. Add a peer-review checklist for each launch configuration before initialization.
2. State the minimum practical sale window in the operator runbook.
3. State which accounts and rent cannot be recovered after a terminal state.

3.3 Add Continuous Integration With Dependency Scanning

FROM IRON-20

JurassicFi corrected the error variants, the validation rules, the rounding test, and the absence of events. The repository has no continuous integration configuration. Therefore no automated process runs the test suite, the linters, or a dependency scan.

Dependency scanning during this review found no vulnerability that affects on-chain execution. Four advisories were limited to host tooling and build tooling. Fourteen JavaScript advisories were limited to the development toolchain and the test toolchain. These results are valid for the review date only.

1. Add a continuous integration workflow that builds the program, runs the unit tests, and runs the integration tests.
2. Add `cargo audit` and `npm audit` to that workflow.
3. Make the workflow a required check before a merge.

3.4 Complete Operator Documentation

FROM IRON-21

JurassicFi corrected the order of the transfer and the validation, and removed the unnecessary mutability marker. JurassicFi intentionally retains overlapping account constraints as defense-in-depth checks. The exact error code can therefore depend on the constraint evaluation order of Anchor. JurassicFi accepts this behavior.

The repository still contains no README, no architecture note, and no deployment procedure. Therefore the purpose of the payout addresses, the intended payment mint, and the launch lifecycle must be read from the source.

1. Add a README that states the launch lifecycle, the purpose of the escrow address and the treasury address, and the intended payment mint.
2. Document that overlapping constraints can produce different errors for equivalent substitution attempts.
3. Add a deployment procedure that includes the release gate that IRON-17 recommends.

3.5 Confirm The Pinned Toolchain From A Clean Checkout

FROM IRON-22

JurassicFi pinned the Solana version, added a Rust toolchain file, and added the locked flag to the build scripts. The pinned configuration is not confirmed from end to end.

On the review host, interface definition generation fails with the pinned compiler, because `anchor-syn` 0.32.1 calls an unstable procedural macro method. Only a nightly toolchain completed that step. Therefore the documented build command can still fail for a new engineer.

1. Run the documented build command from a clean checkout on a host that holds the pinned toolchain only.
2. Record the exact compiler version that interface definition generation needs.
3. Add that build to continuous integration so that a toolchain difference is visible immediately.

IRONNODE

ironnode.io

ASSESSMENT COMPLETE

Thank you for choosing IronNode.

JURASSICFI / SOLANA PROGRAM SECURITY AUDIT / 2026