StackFi Avax Security Review

Introduction

A time-boxed security review of the **StackFi Avax** contracts was conducted by **Stonewall**, with a focus on the security aspects of the smart contract implementation.

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

About Stonewall

Stonewall is an independent smart contract security firm delivering immovable protection for Web3 protocols. Our team brings deep expertise in DeFi security, having reviewed DEXs, yield farming protocols, gaming contracts, and complex financial systems.

About StackFi Avax

StackFi Avax is a **fork of Gearbox Protocol** adapted for the Avalanche network. It provides leveraged yield farming through:

- **Credit Accounts**: Isolated smart contract accounts for leveraged positions
- **Credit Facade**: User-facing interface for opening/managing positions
- **Pool V3**: Lending pools providing liquidity for leverage
- **Trading Adapters**: Connectors to DEXs and lending protocols (Aave V3)
- **Price Oracles**: Asset price feeds for health factor calculation

Key Differences from Gearbox

StackFi is a minimal fork with the following customizations:

1. **Network Adaptation**: Configured for Avalanche C-Chain
2. **Pool Addresses**: Updated pool and oracle addresses for AVAX ecosystem
3. **Aave V3 Adapter**: Custom adapter for Aave V3 lending on Avalanche
4. **Chainlink Oracles**: Configured for Avalanche price feeds
5. **Quota Mechanism Removed**: The original Gearbox V3 quota system for managing collateral limits has been removed to simplify the protocol. This reduces complexity but removes granular collateral management.
6. **Position Tracking**: Custom `PositionTracker` and `TradingPositionTracker` contracts added for leverage trading position management

Privileged Roles & Actors

Role	Description
Configurator	Can modify pool parameters, add collateral tokens
Controller	Emergency pause, risk parameter adjustments
Adapters	Whitelisted contracts for executing trades

Observations

- Core credit logic unchanged from audited Gearbox V3
- Custom AaveV3_PoolAdapter for Avalanche lending integration

- Standard ERC4626-style pool implementation
- Chainlink price feeds configured for all supported tokens

Risk Classification

	High Impact	Medium Impact	Low Impact
High Likelihood	Critical	High	Medium
Medium Likelihood	High	Medium	Low
Low Likelihood	Medium	Low	Low

Security Assessment Summary

Review Details	
Protocol Name	StackFi Avax
Repository	Private
Commit	`ef64f69`
Review Date	January 2026
Methods	Manual review, static analysis
Network	Avalanche C-Chain (43114)
Base Protocol	Gearbox Protocol V3

Project Links

Platform	Link
Twitter	@StackFi_
Telegram	Join Telegram
Discord	Join Discord
Farcaster	@stackfi

Base Protocol Security

Gearbox Protocol V3 has undergone extensive auditing:

- **ChainSecurity** - Full protocol audit
- **Sigma Prime** - Credit account and pool review
- **Code4rena** - Competitive audit with $200K+ bounty
- **Immunefi** - Active bug bounty program ($1M+ max)

Scope

Contract	SLOC	Notes
`AaveV3_PoolAdapter.sol`	~200	Aave V3 lending adapter
`PriceOracleV3.sol`	~300	Price feed router (custom)

`PositionTracker.sol`	~450	Position tracking for trades
`TradingPositionTracker.sol`	~400	Trading adapter with position tracking
`CreditManagerV3.sol`	~800	Core credit logic (forked)
`CreditFacadeV3.sol`	~600	User interface (forked)
`PoolV3.sol`	~500	Lending pool (forked)
Deployment Scripts	~1000	Configuration scripts

Deployed Contracts (Avalanche C-Chain)

Core Infrastructure

Contract	Address
AddressProviderV3	`0x044D6f017615deE9C06ebEb1E64b11284d0d59b5`
ACL	`0x649Db9c5b7d3A5EB2C91745E40e70f2F535A3476`
PriceOracleV3	`0x2EBcc4EdF9F119Eb517e7d3c685Fc5BA2947c381`
ContractsRegister	`0xBaBeB29dA07B8953c47b770A99cbE0F82120BE14`
AccountFactoryV3	`0xa4a33b1Dd7D6E3AaB184B2cfD66363C9D4eBfbA2`
BotListV3	`0xF71f28aEBcDAfEA828e317f4c9E76FD16D74AB8c`
DataCompressorV3	`0xB025C844F8F7c2E28a54d62CB38688DA3c9DEc47`

Lending Pools & Credit Managers

Pool	Underlying	Pool Address	Credit Manager	
Pool 1	WAVAX	`0xBF4d48eB8307983150460B74722535Ba148549B8`	`0x25CD84310b71fbaD56D32489Fb98a5D75929083e`	`0`
Pool 2	BTC.b	`0xBDA7e41133fD17743ede8b33D7A20F49810106A8`	`0x8E0217C76d0cbAf5De443a16c3A2B02212A375a7`	`0`
Pool 3	WETH	`0x08f2B4801A9cF53b9fE1dF7fF3E744A9DC1cDC62`	`0x3D95f59d62472B410E2F354037D9276222A0fBFC`	-
Pool 4	DAI	-	`0x4d3A7e73BEBB9cd934d470B4F770406b371951fe`	-
Pool 5	USDC	-	`0x0416CEa0153f3c454978cc0Df24407D457e92027`	-
Pool 6	USDT	-	`0xED2D8640269fc99793b7c070E927986B9F0e6128`	-
Pool 7	LINK	-	`0xbfA7EE14EBbBBE825917e01a9b91AE96105e8518`	-

Aave V3 Adapters

Credit Manager	Adapter Address
WAVAX Pool	`0xA008B0FB38F9f90175834F0C28f6dA34ed7bc894`
BTC.b Pool	`0xD668B2e70e6c752953a0F9E2e609ff1EFB04C7D8`
WETH Pool	`0x7002b55f162d508a70AF644465e60Acd49726a2c`
DAI Pool	`0x0FEFaB571c8E69f465103BeC22DEA6cf46a30f12`
USDC Pool	`0xA83cEA2E789dCFb3394164Cc468e671d133Ae433`
USDT Pool	`0x9F7f51390F7b8Fa2Ea5d40904296C126aeA2F8d2`
LINK Pool	`0xC13A5441E7790A8aA749Cae7E39b639A32920696`

Price Feeds (Chainlink on Avalanche)

Token	Price Feed Address
WAVAX/USD	`0x0A77230d17318075983913bC2145DB16C7366156`
WETH/USD	`0x976B3D034E162d8bD72D6b9C989d545b839003b0`
USDC/USD	`0xF096872672F44d6EBA71458D74fe67F9a77a23B9`
USDT/USD	`0xEBE676ee90Fe1112671f19b6B7459bC678B67e8a`
DAI/USD	`0x51D7180edA2260cc4F6e4EebB82FEF5c3c2B8300`
LINK/USD	`0x49ccd9ca821EfEab2b98c60dC60F518E765EDe9a`
BTC/USD	`0x2779D32d5166BAaa2B2b658333bA7e6Ec0C65743`

Supported Collateral Tokens

Token	Address	Decimals
WAVAX	`0xB31f66AA3C1e785363F0875A1B74E27b85FD66c7`	18
WETH	`0x49D5c2BdFfac6CE2BFdB6640F4F80f226bc10bAB`	18
USDC	`0xB97EF9Ef8734C71904D8002F8b6Bc66Dd9c48a6E`	6
USDT	`0x9702230A8Ea53601f5cD2dc00fDBc13d4dF4A8c7`	6
DAI	`0xd586E7F844cEa2F87f50152665BCbc2C279D8d70`	18
LINK	`0x5947BB275c521040051D82396192181b413227A3`	18
BTC.b	`0x152b9d0FdC40C096757F570A51E494bd4b943E50`	8

Aave V3 aTokens (Collateral)

aToken	Address	Underlying
aAvaWAVAX	`0x6d80113e533a2C0fe82EaBD35f1875DcEA89Ea97`	WAVAX
aAvaWETH	`0xe50fA9b3c56FfB159cB0FCA61F5c9D750e8128c8`	WETH
aAvaUSDC	`0x625E7708f30cA75bfd92586e17077590C60eb4cD`	USDC

aAvaUSDT	`0x6ab707Aca953eDAeFBc4fD23bA73294241490620`	USDT
aAvaDAI	`0x82E64f49Ed5EC1bC6e43DAD4FC8Af9bb3A2312EE`	DAI
aAvaLINK	`0x191c10Aa4AF7C30e871E70C95dB0E4eb77237530`	LINK
aAvaBTC.b	`0x078f358208685046a11C85e8ad32895DED33A249`	BTC.b

Findings Summary

ID	Title	Severity	Status
[H-01]	TradingPositionTracker uses hardcoded Base chain addresses	High	Fixed
[M-01]	Aave adapter withdraw sends tokens to arbitrary address	Medium	Acknowledged
[M-02]	PositionTracker.executeLong external self-call bypasses access control	Medium	Fixed
[M-03]	Input validation disabled in TradingPositionTracker.openPosition	Medium	Fixed
[L-01]	No staleness check bypass for trusted price feeds	Low	Acknowledged
[L-02]	Missing slippage protection in Aave adapter	Low	Open
[L-03]	Liquidation threshold set uniformly for all aTokens	Low	Acknowledged
[L-04]	O(n) loop in _removeFromOpenPositions could cause gas issues	Low	Acknowledged
[L-05]	Price feeds set with skipCheck=true bypasses all validation	Low	Acknowledged
[L-06]	Quota mechanism removal reduces collateral management granularity	Low	Acknowledged
[I-01]	Debug function and unused variable in PriceOracleV3	Informational	Open
[I-02]	Consider adding emergency withdraw in adapters	Informational	Open
[I-03]	Missing events for critical state changes	Informational	Open
[I-04]	Commented out isContract check in _validateToken	Informational	Acknowledged
[I-05]	P&L calculations could overflow with extreme values	Informational	Open

---

Findings

[H-01] TradingPositionTracker uses hardcoded Base chain addresses

Severity: High

Status: Fixed

Location: `contracts/core/TradingPositionTracker.sol:71, 78`

Description:

The `TradingPositionTracker` contract contained hardcoded addresses for Base chain instead of Avalanche:

```
constructor(address _creditManager) AbstractAdapter(_creditManager,
0x2626664c2603336E57B271c5C0b26F421741e481) {
```

```
    // Target contract set to Uniswap V3 SwapRouter on Base for trading operations
}

function getPriceOracle() internal view returns (IPriceOracleV3) {
    return IPriceOracleV3(0x84DE1ee49306822Bf2f71Dcb4980fec8A28ddd0F);
}
```

These addresses (`0x2626664c...` and `0x84DE1ee4...`) are Base chain contracts, not Avalanche. Using them on Avalanche would cause transactions to fail or interact with wrong contracts.

Resolution:

The team has updated the addresses to use Avalanche-specific contracts and dynamically fetch the price oracle from the credit manager.

[M-01] Aave adapter withdraw sends tokens to arbitrary address

Severity: Medium

Location: `scripts/helpers/adapters/AAVE_v3.sol:282-307`

Description:

The `withdraw` function in `AaveV3_PoolAdapter` accepts a `to` parameter that could potentially be set to an arbitrary address:

```
function withdraw(
    address asset,
    uint256 amount,
    address to  // Can be any address
)
```

While this is called through the CreditFacade which has access controls, the pattern differs from the safer `withdrawDiff` which always sends to the credit account.

Recommendation:

Consider always withdrawing to the credit account address:

```
function withdraw(address asset, uint256 amount, address /* to */) {
    address creditAccount = _creditAccount();
    // Always withdraw to creditAccount, not arbitrary 'to'
}
```

[M-02] PositionTracker.executeLong external self-call bypasses access control

Severity: Medium

Status: Fixed

Location: `contracts/core/PositionTracker.sol:179`

Description:

The `executeLong` function makes an external call to `this.openPosition()` :

```
function executeLong(...) external returns (uint256 positionIndex) {
    address creditAccount = msg.sender;
```

```
    // ...
    positionIndex = this.openPosition(  // External self-call
        longOrShortToken,
        Direction.Long,
        positionSize,
        priceOracle.getPrice(longOrShortToken),
        leverage,
        creditAccount
    );
}
```

Since `openPosition` has a `creditFacadeOnly` modifier and `this.openPosition()` changes `msg.sender` to the contract's address, this call would always fail because the contract itself is not the credit facade.

Resolution:

The team changed `this.openPosition()` to an internal function call `_openPosition()`.

[M-03] Input validation disabled in TradingPositionTracker.openPosition

Severity: Medium

Status: Fixed

Location: `contracts/core/TradingPositionTracker.sol:172-176`

Description:

Critical input validation was commented out in the `openPosition` function:

```
function openPosition(...) internal returns (uint256 positionIndex) {
    // require(_creditAccount != address(0), "Invalid credit account address");
    // require(_asset != address(0), "Invalid asset address");
    // require(_quantity > 0, "Quantity must be greater than 0");
    // require(_openPrice > 0, "Open price must be greater than 0");
    // require(_leverage >= 100, "Leverage must be at least 1x (100 basis points)");
```

This allows creation of invalid positions with zero addresses, zero quantities, or zero prices which could corrupt the position tracking state.

Resolution:

The team has re-enabled the input validation checks.

[L-01] No staleness check bypass for trusted price feeds

Severity: Low

Location: `contracts/core/PriceOracleV3.sol:40`

Description:

The `revertOnZeroPrice` flag is hardcoded to `true` but there's no mechanism to bypass staleness checks for trusted feeds in emergency situations. If a Chainlink feed becomes temporarily stale, the entire protocol could be blocked.

Recommendation:

Consider implementing an emergency mode or backup oracle system for critical operations.

[L-02] Missing slippage protection in Aave adapter

Severity: Low

Location: `scripts/helpers/adapters/AAVE_v3.sol:201-224`

Description:

The `supply` and `withdraw` functions in the Aave adapter don't include slippage protection parameters. While Aave V3 supply/withdraw typically have 1:1 exchange rates, market conditions or protocol upgrades could introduce slippage.

Recommendation:

Consider adding minimum output amount parameters:

```
function supply(
    address asset,
    uint256 amount,
    uint256 minSharesOut  // Add slippage protection
) external;
```

[L-03] Liquidation threshold set uniformly for all aTokens

Severity: Low

Location: Deployment scripts

Description:

All Aave aTokens were added with the same liquidation threshold (85%). Different assets have different risk profiles and should have different liquidation thresholds:

- Stablecoins (aUSDC, aUSDT, aDAI): Could have higher LT (~90%)
- Volatile assets (aWAVAX, aWETH): Should have lower LT (~75-80%)

Recommendation:

Configure asset-specific liquidation thresholds based on volatility and liquidity.

[L-04] O(n) loop in _removeFromOpenPositions could cause gas issues

Severity: Low

Status: Acknowledged

Location: `contracts/core/TradingPositionTracker.sol:409-417`, `contracts/core/PositionTracker.sol:459-468`

Description:

The `_removeFromOpenPositions` function uses a linear search through the entire `openPositionIndexes` array:

```
function _removeFromOpenPositions(uint256 _positionIndex) internal {
    for (uint256 i = 0; i < openPositionIndexes.length; i++) {
        if (openPositionIndexes[i] == _positionIndex) {
            openPositionIndexes[i] = openPositionIndexes[openPositionIndexes.length - 1];
            openPositionIndexes.pop();
            break;
        }
```

```
    }
}
```

With many open positions, this could exceed gas limits when closing positions.

Recommendation:

Consider using a mapping to track position indices for O(1) removal, or limit the maximum number of open positions.

[L-05] Price feeds set with skipCheck=true bypasses all validation

Severity: Low

Status: Acknowledged

Location: `contracts/core/PriceOracleV3.sol:319-328`

Description:

The `setPriceFeed` function always sets `skipCheck = true` and `trusted = true`:

```
function setPriceFeed(address token, address priceFeed) external {
    // ...
    bool skipCheck = true;
    _priceFeedsParams[token] = PriceFeedParams({
        priceFeed: priceFeed,
        stalenessPeriod: 0,
        skipCheck: skipCheck,  // Always skips validation
        decimals: decimals,
        useReserve: false,
        trusted: true  // Always trusted
    });
}
```

This bypasses staleness checks and allows potentially stale prices to be used without validation.

Recommendation:

Consider allowing configurator to specify `skipCheck` and `trusted` parameters, or implement automatic staleness detection.

[L-06] Quota mechanism removal reduces collateral management granularity

Severity: Low

Status: Acknowledged

Location: Fork-wide change

Description:

The original Gearbox V3 protocol includes a quota mechanism that allows fine-grained control over how much of each collateral type can be held across the protocol. This was removed in StackFi to simplify the codebase.

Impact:

- No per-token caps on collateral exposure
- Reduced ability to manage risk from concentrated positions in a single asset
- Simplified protocol but less control over systemic risk

Recommendation:

Consider implementing simplified collateral caps at the pool level if concentration risk becomes a concern.

[I-01] Debug function and unused variable in PriceOracleV3

Severity: Informational

Location: `contracts/core/PriceOracleV3.sol:274-277`

Description:

The PriceOracleV3 contract contains test/debug code that should be removed:

```
function sasasasaadasd() public view returns (uint256 aa) {
    aa = 12121;
}
uint256 lelf = 1312;
```

This includes:

1. A test function with a meaningless name (`sasasasaadasd`)
2. An unused state variable (`lelf`)

These waste gas on deployment and reduce code clarity.

Recommendation:

Remove all test/debug code before mainnet deployment.

[I-02] Consider adding emergency withdraw in adapters

Severity: Informational

Description:

The Aave adapter doesn't include an emergency withdrawal mechanism. If the Aave protocol experiences issues, users might not be able to withdraw their funds.

Recommendation:

Consider implementing an emergency withdrawal function that can be called by the configurator role.

[I-03] Missing events for critical state changes

Severity: Informational

Description:

Some state-changing operations in the adapter don't emit events, making it harder to track protocol activity off-chain.

Recommendation:

Add events for all supply and withdraw operations in the adapter.

[I-04] Commented out isContract check in _validateToken

Severity: Informational

Status: Acknowledged

Location: `contracts/core/PriceOracleV3.sol:382`

Description:

The `isContract` check in `_validateToken` is commented out:

```
function _validateToken(address token) internal view returns (uint8 decimals) {
    // if (!Address.isContract(token)) revert AddressIsNotContractException(token);
    try ERC20(token).decimals() returns (uint8 _decimals) {
```

While the `decimals()` call will fail for EOAs, explicitly checking `isContract` provides clearer error messages and follows defense-in-depth principles.

Recommendation:

Consider re-enabling the `isContract` check for clearer error handling.

[I-05] P&L calculations could overflow with extreme values

Severity: Informational

Location: `contracts/core/TradingPositionTracker.sol:353-364`, `contracts/core/PositionTracker.sol:413-427`

Description:

The P&L calculation functions perform multiplication before division:

```
function calculateUnrealizedPnL(...) public pure returns (int256) {
    if (_direction == Direction.Long) {
        int256 priceDiff = int256(_currentPrice) - int256(_openPrice);
        return (priceDiff * int256(_quantity)) / int256(_openPrice);  // Could overflow
    }
    // ...
}
```

With very large quantities and price differences, the multiplication `priceDiff * int256(_quantity)` could overflow before the division.

Recommendation:

Consider using a library for safe fixed-point math, or restructure calculations to avoid overflow:

```
return (int256(_quantity) * priceDiff) / int256(_openPrice);
// Could be restructured as:
return int256(_quantity) * (priceDiff / int256(_openPrice));
```

Inherited Security Properties

As a Gearbox V3 fork, StackFi inherits these security properties:

Positive (from Gearbox)

- Isolated credit accounts prevent cross-account attacks
- Health factor system prevents under-collateralization
- Multi-sig and timelock on critical operations
- Comprehensive access control via ACL
- Pause functionality for emergencies
- Price oracle redundancy

- Reentrancy protection via nonReentrant modifier

Considerations for Fork

- Oracle addresses correctly point to Avalanche Chainlink feeds ✓
- Adapter allowlists configured for Aave V3 on AVAX ✓
- Pool parameters configured for AVAX market conditions ✓
- Collateral tokens verified for AVAX network ✓
- aTokens added as collateral with price feeds ✓

Fork-Specific Recommendations

1. **Oracle Monitoring**
    - Set up alerts for Chainlink feed staleness
    - Monitor price deviations between feeds
    - Consider implementing a fallback oracle (e.g., Pyth)
2. **Adapter Security**
    - Add slippage protection parameters
    - Implement emergency withdrawal functions
    - Add comprehensive event logging
3. **Risk Parameters**
    - Review liquidation thresholds per asset class
    - Set appropriate debt ceilings per pool
    - Configure liquidation incentives for AVAX gas costs
4. **Operational Security**
    - Use multi-sig for all admin functions
    - Implement timelock for parameter changes
    - Set up monitoring for large position changes

Conclusion

StackFi Avax is a fork of the well-audited Gearbox Protocol V3 with custom Aave V3 adapter integration and position tracking for Avalanche. The core protocol logic inherits the security properties of multiple professional audits. The team has removed the quota mechanism to simplify the protocol and added custom position tracking contracts for leverage trading.

Key Findings:

- 1 High severity issue (hardcoded wrong chain addresses) - **Fixed**
- 3 Medium severity issues (arbitrary withdraw, access control bypass, disabled validation) - **2 Fixed, 1 Acknowledged**
- 6 Low severity issues (staleness, slippage, uniform LT, gas, skipCheck, quota removal)
- 5 Informational issues (debug code, emergency withdraw, events, isContract, overflow)

Fixed Issues:

- [H-01] TradingPositionTracker hardcoded Base addresses → Updated to Avalanche
- [M-02] External self-call in executeLong → Changed to internal call
- [M-03] Commented out validation → Re-enabled

Deployed Infrastructure:

- 7 lending pools (WAVAX, WETH, USDC, USDT, DAI, LINK, BTC.b)

- 7 Aave V3 adapters for leveraged farming
- 14 tokens configured with Chainlink price feeds
- Full access control via ACL system
- Position tracking for leverage trades

Overall Risk Assessment: Low-Medium

The protocol is production-ready with the following considerations:

1. Address the withdraw pattern in Aave adapter (Acknowledged)
2. Configure asset-specific liquidation thresholds
3. Remove debug code from PriceOracleV3
4. Set up operational monitoring
5. Monitor for gas issues with large position counts

This security review was conducted by Stonewall. For questions or clarifications, contact our team.

Audit completed: January 2026