Paladin Program Audit Report

– Mad Shield

David P – david@fomo3d.app david@madshield.xyz
BlueWolf – wolf@madshield.xyz

Table of Contents

1. Introduction

This audit focuses on the Paladin (P3) protocol, a Solana-based validator enhancement designed to revolutionize transaction processing through an innovative priority and bundle execution system. Paladin addresses the challenges of MEV (Maximal Extractable Value), transaction ordering, and front-running in the Solana ecosystem by implementing a specialized UDP-based transaction ingestion system (P3) and a sophisticated bundle processing mechanism.The audit was conducted between the 19th of december 2024 and the 25th of december 2024.

This report outlines the audit process, describes the methodology used, and examines the protocol's security while in beta phase. Special consideration is given to Paladin's beta status and planned production features, including the transition from IP whitelisting to $PAL token-gating for validator participation.

2. Findings & Recommendations

Our severity classification system adheres to the criteria outlined here.

Severity Level	Exploitability	Potential Impact	Examples
Critical	Low to moderate difficulty, 3rd-party attacker	Irreparable financial harm	Direct theft of funds, permanent freezing of tokens/NFTs
High	High difficulty, external attacker or specific user interactions	Recoverable financial harm	Temporary freezing of assets
Medium	Unexpected behavior, potential for misuse	Limited to no financial harm, non-critical disruption	Escalation of non-sensitive privilege, program malfunctions, inefficient execution
Low	Implementation varia(nce, uncommon scenarios	Zero financial implications, minor inconvenience	Program crashes in rare situations, parameter adjustments by authorized entities
Informational	N/A	Recommendations for improvement	Design enhancements, best practices, usability suggestions

No, Critical, High, Medium or Low severities were identified.

3. Protocol Overview

Paladin represents a significant advancement in Solana's transaction processing ecosystem, designed to enhance both validator profitability and user transaction security. The protocol introduces innovative mechanisms to combat sandwich attacks while simultaneously increasing validator rewards, creating a mutually beneficial environment for all participants.

During the beta phase, this system employs IP whitelisting for security, with plans to transition to a $PAL token-gating mechanism in production. This careful approach ensures both security and controlled growth of the network.

The protocol's economic model introduces a revolutionary concept in the DeFi space: making it more profitable for validators to prevent sandwich attacks than to execute them. This creates a virtuous cycle where users, trusting in the protection against sandwich attacks, preferentially route their high-priority transactions through Paladin-enabled validators ("Palidators"). As a result, these validators process more transactions and earn higher rewards, reinforcing the economic incentives for honest behavior.

Built on top of Jito-Solana, the protocol requires no additional DevOps overhead, allowing validators to maintain their current operational workflows while gaining access to enhanced revenue streams. This compatibility ensures that validators can adopt Paladin without sacrificing their existing setups or revenue sources.

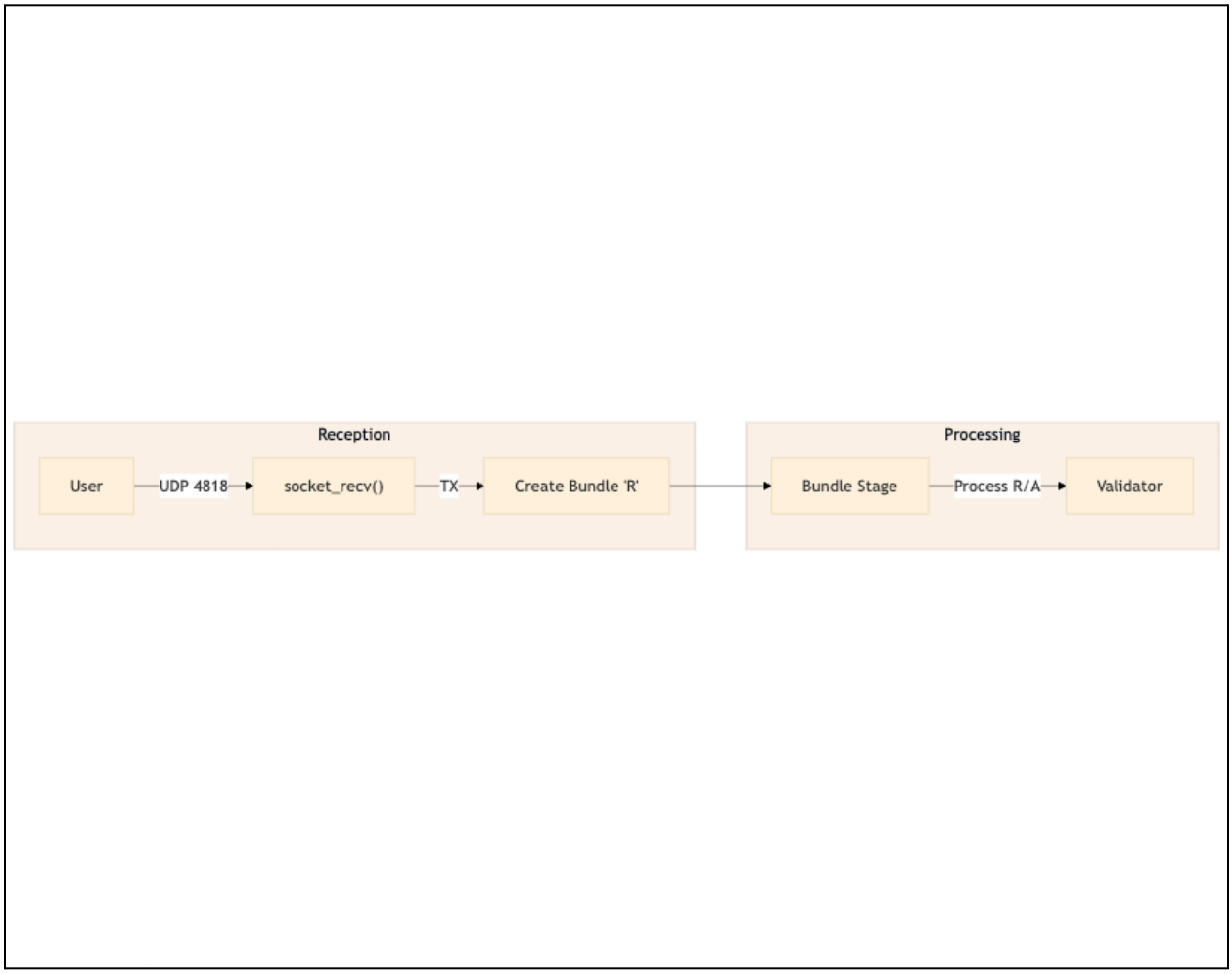
The protocol has demonstrated remarkable effectiveness in its initial deployment, achieving a 95% reduction in sandwich attack_bundles while delivering a 12.5% increase in validator rewards. These impressive metrics are particularly significant given the protocol's integration with Solana's $3.8B network stake. Currently, approximately 5% of validators have implemented Paladin, with adoption growing steadily as the benefits become increasingly apparent.

Paladin is positioned to fundamentally transform the DeFi landscape by aligning validator incentives with user interests. The protocol's success in reducing sandwich attacks while increasing validator profitability demonstrates the viability of creating systems where security and profitability reinforce each other, rather than existing in opposition.

Program Charts

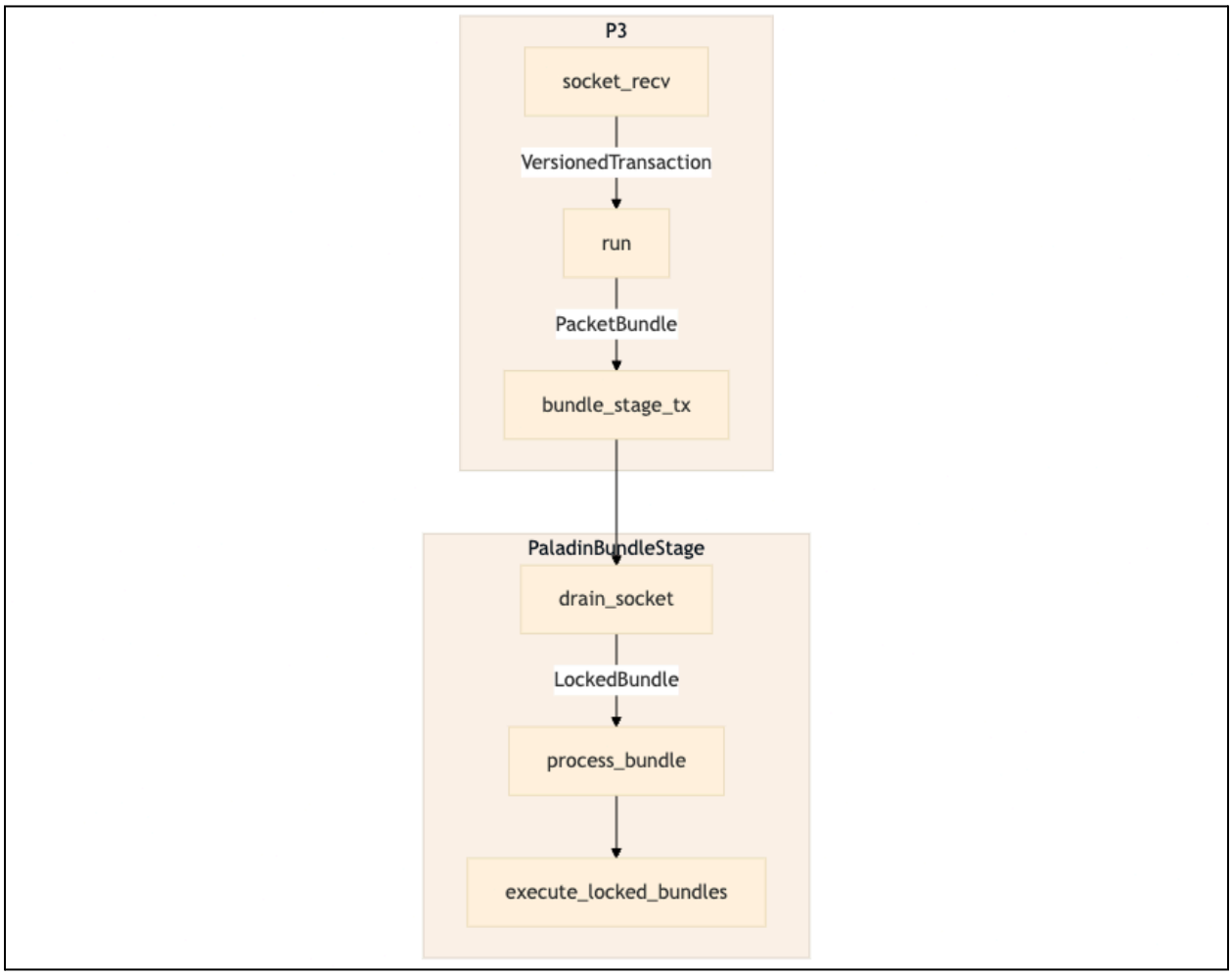
Transaction Flow

Paladin operates as a specialized transaction processing system for Solana validators, designed to enhance transaction prioritization and execution. The protocol's core flow begins at UDP port 4818, where high-priority transactions are received through `socket_recv()`. Each incoming transaction is wrapped into a bundle and marked with an 'R' prefix for regular transactions. These bundles are then forwarded to the Bundle Stage, which manages different bundle types (marked as 'R' or 'A') and processes them according to priority rules. Finally, the validator processes these prioritized bundles during block production.



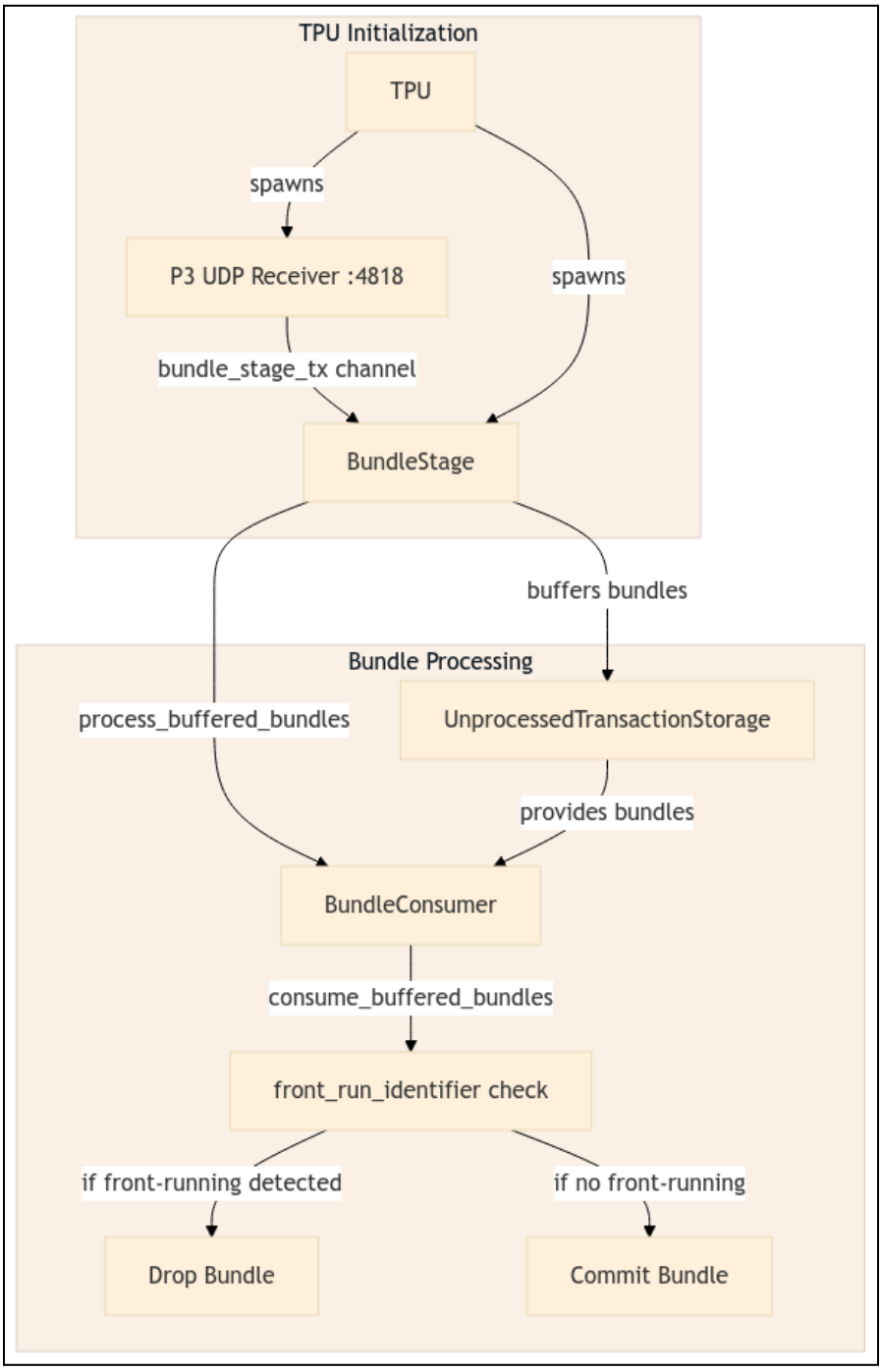
Bundle Processing Architecture

The Paladin protocol implements a multi-stage bundle processing system that ensures secure and efficient transaction handling. The architecture consists of two main components: the initial P3 reception layer and the PaladinBundleStage processing layer.



Bundle Processing Pipeline

The pipeline starts with P3, a UDP receiver on port 4818, which forwards transaction bundles to `BundleStage` through a channel. `BundleStage` buffers these in `UnprocessedTransactionStorage` before processing them via `BundleConsumer`. The BundleConsumer runs front-running checks - if detected, the bundle is dropped; if clean, it's committed. This creates a streamlined pipeline for secure bundle processing.



Front-Running Detection System

The front-running detection system in Paladin is designed to identify and prevent Maximal Extractable Value (MEV) attacks, particularly sandwich attacks and front-running in DeFi transactions. The system analyzes transaction bundles for suspicious patterns involving AMM (Automated Market Maker) interactions.

4. Methodology

Given that Paladin is a priority transaction protocol managing validator transaction processing and MEV protection, we employed a comprehensive and systematic methodology to test the protocol's constraints and behavior.

Our primary approach involved analyzing the UDP transaction ingestion system and bundle processing mechanisms to validate the protocol's security and reliability.This methodology allowed us to observe how the protocol would operate in real-world validator conditions, especially focusing on the transaction prioritization implementation and bundle processing systems.

This security-focused approach was essential for evaluating the safety and reliability of the protocol's core components, particularly the UDP port exposure and transaction handling.We conducted detailed code analysis of transaction reception and bundle processing functions. This included reviewing the implementation of UDP packet handling, bundle creation, account locking mechanisms, and the security of transaction processing across different scenarios.

We documented security observations and potential improvements to enhance the protocol's reliability and safety, ensuring robust operation in production validator environments. Particular emphasis was placed on the protocol's claims of sandwich attack prevention and increased validator rewards through proper transaction prioritization.

5. Scope and Objectives

The primary objectives of the audit are defined as:

- Minimizing the possible presence of any critical vulnerabilities in the program. This would include detailed examination of the code and edge case scrutinization to find as many vulnerabilities.

- 2-way communication during the audit process. This included for Mad Shield to reach a perfect understanding of the design of the system and the goals of the team.

- Provide clear and thorough explanations of all vulnerabilities discovered during the process with potential suggestions and recommendations for fixes and code improvements.

- Clear attention to the documentation of the vulnerabilities with an eventual publication of a comprehensive audit report to the public audience for all stakeholders to understand the security status of the programs.

Paladin has delivered the program to Mad Shield at the following Github repositories.

Repository URL	https://github.com/paladin-bladesmith/paladin-solana
Commit (start of audit)	684eba2f61a97633c21ac34e89d20562ec100e24
Commit (end of audit)	TBD

Tab1. Audit Marks of Paladin

6. Conclusion

Mad Shield conducted an extensive audit of Paladin, utilizing a hands-on methodology that prioritizes a detailed, immersive review of the program. Our team's approach is rooted in active collaboration, working closely with each unique project to identify potential security risks and mitigate vulnerabilities effectively.

Mad Shield's dedication to advancing auditing techniques is clear throughout our process. We consistently apply innovative strategies, allowing us to analyze the code at a granular level, simulate real-world scenarios, and uncover potential risks that traditional audits might miss.

No critical vulnerabilities were identified during the Paladin audit, and all findings were promptly communicated to the development team. Our recommendations focus on strengthening the codebase to enhance long-term security and resilience. Mad Shield remains committed to setting new standards in smart contract auditing.