Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 22 June 2026

Z. Luo
OraSRS Protocol

19 December 2025

Decentralized Threat Signaling Protocol (DTSP) using OraSRS
draft-luo-orasrs-decentralized-threat-signaling-01

Abstract

This document specifies the Decentralized Threat Signaling Protocol (DTSP), a mechanism for distributed edge clients to collaboratively detect, report, and mitigate network threats. The protocol defines a state machine for threat lifecycle management (T0-T3), a standardized data format for threat signaling, and security mechanisms to prevent abuse in a permissionless environment.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 June 2026.

Copyright Notice

Table of Contents

1. Introduction

The increasing sophistication of network attacks requires a collaborative defense mechanism that operates at the edge. Centralized systems introduce unacceptable latency and single points of failure. DTSP shifts the paradigm from ’Filter then Verify’ to ’Verify then Filter’, enabling < 1ms response times at the edge while maintaining global consensus integrity.

DTSP enables a decentralized network of edge clients to share threat intelligence in real-time, leveraging a blockchain-based consensus mechanism for verification.

See Figure 1 (Architecture Diagram) for the interaction between Kernel Space, User Space, and the Consensus Layer. [Reference: https://github.com/srs-protocol/OraSRS-protocol/docs/]

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Protocol Logic

The lifecycle of a threat in DTSP is modeled as a Finite State Machine (FSM). Each edge client maintains the state of detected threats.

3.1. State Machine

The FSM consists of the following states:

* *IDLE*: No threat detected.
* *OPTIMISTIC_BLOCK (T0)*: Local detection of suspicious activity. Immediate local mitigation.
* *REPORTED (T1)*: Threat evidence submitted to the network.
* *VERIFIED (T2)*: Network consensus reached. Threat confirmed.
* *GLOBAL_ENFORCE (T3)*: Global propagation of the threat signature.
* *EXPIRED*: Threat entry validity period ended.

3.2. State Transitions

3.2.1. T0: Local Detection

The Edge Client MUST transition to the OPTIMISTIC_BLOCK state immediately upon detection of traffic matching local heuristic rules (e.g., high-frequency connection attempts).

In this state, the client:

1. MUST block the source IP locally.
2. SHOULD generate a ThreatEvidence package.

3.2.1.1. Outbound C2 Interception

In addition to inbound defense, the Edge Client MUST inspect outbound traffic for patterns matching known C2 (Command & Control) signatures.

* *Pre-Release Query*: Outbound connections to suspicious domains/IPs trigger a synchronous lookup in the Local Cache.

* *T0 Block*: If a match is found, the connection is terminated immediately at the kernel level to prevent data exfiltration.

This mechanism protects against private key exfiltration from compromised nodes and botnet command reception. Implementation MUST use kernel-space filtering (e.g., Netfilter/eBPF) to achieve sub-millisecond interception latency.

3.2.2. T1: Signaling

The Edge Client MUST transition to the REPORTED state after generating valid evidence. The client sends a ThreatSignal message to the network.

To prevent front-running, the client MUST use a Commit-Reveal scheme:

1. *Commit*: Send Hash(Evidence + Salt).

2. *Reveal*: Send Evidence + Salt after a random delay.

3.2.3. T2: Verification

The state transitions to VERIFIED when the network (via Smart Contract or Oracle) validates the evidence. A threat is considered verified if: Sum(Reputation(Reporters)) > Threshold

3.2.4. T3: Global Enforcement

Upon entering the GLOBAL_ENFORCE state, the threat signature is added to the Global Blocklist. All participating clients MUST synchronize this list and enforce blocking rules via their local kernel datapath (e.g., eBPF).

4. Data Format

4.1. Threat Signal

The ThreatSignal message is used to report a detected threat. It is serialized using JSON.

Field	Type	Description
version	uint8	Protocol version (e.g., 1)
type	uint8	Threat type (0=DDoS, 1=Scanning, 2=Malware)
source_ip	bytes	IP address of the attacker (4 or 16 bytes for IPv4/IPv6)
target_ip	bytes	IP address of the victim (4 or 16 bytes for IPv4/IPv6, optional)
timestamp	uint64	Unix timestamp of detection (ms)
evidence	bytes	Cryptographic proof or log snippet
signature	bytes	Digital signature of the reporting client

Table 1

4.2. Evidence Package

Field	Type	Description
packet_dump	bytes	Sample of malicious packets (pcap format)
flow_stats	struct	Flow statistics (PPS, BPS, duration)
heuristics	string	ID of the heuristic rule triggered

Table 2

5. Operational Considerations

5.1. Resource Constraints & Performance

The protocol is designed for resource-constrained edge environments (e.g., 512MB RAM). Implementation validation (v3.3.6) has demonstrated:

* *Latency*: The T0 local heuristic limiter (eBPF) achieves a query latency of 0.001ms (40x better than the 0.04ms target).

* *Throughput*: Capable of mitigating 40M PPS (SYN-Flood) on standard edge hardware while maintaining 100% availability of management channels (SSH).

* *Footprint*: The complete agent operates within < 50MB (Hybrid Mode) or < 5MB (Native Mode) of memory.

Implementers MUST prioritize kernel-space offloading (e.g., Netfilter/eBPF) to meet these latency requirements.

5.2. Deployment Modes

DTSP supports three deployment modes:

1. *T0-Only (Standalone)*: Edge client operates independently without blockchain connectivity. Suitable for air-gapped environments or maximum stability requirements.

2. *T0+T2 (Hybrid)*: Edge client with optional blockchain queries for enhanced threat intelligence. Recommended for most deployments.

3. *T0+T2+T3 (Full)*: Complete decentralized consensus with threat reporting capabilities.

Default configuration SHOULD disable T2/T3 modules to ensure maximum stability on resource-constrained devices.

5.3. Performance Validation

Reference implementation testing (2025-12-19):

* *Test Environment*: Linux 5.14.0, 4 CPU cores, 4.1Gi RAM.

* *API Latency*: 19-24ms (P95 < 50ms).

* *eBPF Query*: 0.001ms average.

* *Stress Test*: 38,766 requests, 0 failures (100% success rate).

* *Memory Usage*: 25.45 MB for 1516 threat entries.

6. Security Considerations

6.1. Sybil Attack Defense

To prevent Sybil attacks where an adversary creates multiple identities to flood the network with false reports, DTSP utilizes a Reputation System.

* Each client MUST stake tokens to participate in reporting.
* The weight of a report is proportional to the client’s ReputationScore.
* ReputationScore increases with valid reports and decreases with false positives.

6.2. Commit-Reveal Mechanism

To prevent "free-riding" (copying others’ reports) or front-running:

1. Clients MUST NOT broadcast raw evidence immediately.
2. Clients MUST broadcast Commit = Hash(Evidence | Nonce).
3. After T_reveal blocks, clients MUST broadcast Reveal = (Evidence, Nonce).
4. The network verifies Hash(Reveal) == Commit.

7. IANA Considerations

This document has no IANA actions.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9. Informative References

[OraSRS-Paper] Luo, Z., "OraSRS: A Compliant and Lightweight Decentralized Threat Intelligence Protocol", 2025, <https://doi.org/10.31224/5985>.

Author’s Address

Ziqian Luo
OraSRS Protocol
Email: luo.zi.qian@orasrs.net