Detecting Hidden Multiprotocol Label Switching Tunnels (MPLS) In Networks

Eyimofe Ajagunna
Mathematics,Computer, and Informnation Sciences
Mississippi Valley State University
Itta Bena, MS, USA
eyimofe.ajagunna@mvsu.edu

Blessed Kutyauripo
Mathematics,Computer, and Informnation Sciences
Mississippi Valley State University
Itta Bena, MS, USA
blessed.kutyauripo@mvsu.edu

LeDarius Robinson
Mathematics,Computer, and Informnation Sciences
Mississippi Valley State University
Itta Bena, MS, USA
ledarius.robinson@mvsu.edu

Khaled Sabahein
Mathematics,Computer, and Informnation Sciences
Mississippi Valley State University
Itta Bena, MS, USA
khaled.sabahein@mvsu.edu

Latonya Garner-Jackson
Mathematics,Computer, and Informnation Sciences
Mississippi Valley State University
Itta Bena, MS, USA
lgarner@mvsu.edu

***Abstract*—This research presents a method for detecting hidden Multiprotocol Label Switching (MPLS) tunnels using active network probing. Our system improves on existing traceroute-based approaches by combining TTL transparency analysis, quoted-TTL deficit measurement, MPLS label stack extraction (RFC 4950), and temporal correlation. It also distinguishes between routers that are part of tunnels and those that are simply rate-limiting responses. The proposed pipeline operates externally, requires no network access, and uses a weighted scoring model to identify tunnels and estimate their boundaries. This work contributes a reliable and reproducible framework for improving network visibility and measurement accuracy.**

Keywords—MPLS, hidden tunnel, Label Switched Path, TTL transparency, traceroute, pipe mode, ICMP rate-limiting, tunnel detection, quoted-TTL deficit, network security

I. INTRODUCTION

Multiprotocol Label Switching (MPLS) is a data-forwarding technique used to make large networks faster and more efficient. Instead of examining a packet's destination at every step, MPLS assigns a short label at the network entry point. Routers then forward packets based on this label along a pre-defined route called a Label Switched Path (LSP), reducing processing overhead and improving scalability.

A hidden MPLS tunnel is an LSP that exists outside normal visibility. It may be created without authorization, missing from network records, or configured to bypass inspection points. These tunnels can result from misconfigurations or misuse of protocols such as the Label Distribution Protocol (LDP), allowing traffic to quietly avoid monitoring systems.

It poses significant security risks as tools such as firewalls, intrusion detection systems (IDS), and deep packet inspection (DPI) rely on seeing network traffic. Hidden tunnels bypass these controls, creating unmonitored paths that can carry malicious activity without detection. Also, undocumented paths disrupt traffic planning, reduce reliability, and make troubleshooting difficult, since engineers cannot fix problems they cannot see.

This paper presents an external detection system that addresses key limitations of existing methodologies. Our approach combines multiple probing techniques and temporal analysis to identify hidden tunnels more reliably. We evaluate the system in a controlled emulated environment and demonstrate improved accuracy. The remainder of this paper is organized as follows: Section II reviews background concepts and related work, Section III explains our methodology, Section IV presents implementation and results, and Section V discusses implications and future work.

II. BACKGROUND & RELATED WORK

A. Traceroute and the Role of TTL

Traceroute maps the path packets take through a network using the Time to Live (TTL) field in IP packets. TTL starts at a fixed value and decreases by one at each router. When it reaches zero, the router drops the packet and returns an ICMP Time Exceeded reply, revealing its identity.

By sending probes with increasing TTL values (1, 2, 3, ...), traceroute discovers each hop along the path. Because every

router must reduce TTL by exactly one, the path is predictable: each TTL corresponds to a specific hop.

Hidden MPLS tunnels break this rule, making TTL behavior inconsistent. This inconsistency forms the basis for their detection.

B. MPLS Tunnel Behavior Under Probing

In Multiprotocol Label Switching (MPLS), packets are assigned labels at the network entry point and forwarded by Label Switching Routers (LSRs) without inspecting the IP header. The label is removed at the exit point, where normal routing resumes.

Two behaviors affect how tunnels appear during probing:

- **Uniform mode (TTL-transparent):** The IP TTL is preserved across the tunnel. Multiple TTL values may return the same router address, which is impossible in normal routing and indicates a tunnel.
- **Pipe mode:** The IP TTL is not updated inside the tunnel. As a result, the TTL observed in ICMP replies is lower than expected. This difference, called the *quoted-TTL deficit*, reveals hidden hops.

Both modes may also create sequences of non-responding hops. However, such gaps alone are not reliable indicators, since rate-limited routers can produce the same effect.

C. Scamper

Scamper is a network measurement tool designed for large-scale active probing. It supports techniques such as traceroute and includes Paris traceroute, which improves measurement accuracy.

Standard traceroute can produce inconsistent paths in load-balanced networks because probe packets vary slightly. Paris traceroute avoids this by keeping packet fields constant, ensuring all probes follow the same path.

Scamper provides accurate data collection but does not analyze that data for MPLS tunnels.

D. TNT

TNT builds on tools like Scamper to detect hidden MPLS tunnels. It analyzes traceroute output rather than collecting data itself.

For uniform mode, TNT detects repeated router responses across consecutive TTL values. For pipe mode, it uses Forward Route Path Length Analysis (FRPLA) to compare expected and observed TTL values. A difference indicates hidden hops.

TNT significantly improved detection, especially for pipe-mode tunnels, which were previously difficult to identify.

E. Limitations of Existing Approaches

Despite these advances, two key problems remain.

- **Rate-limiting ambiguity:** Silent hops may indicate either a hidden tunnel or a router limiting ICMP responses. Existing tools cannot reliably distinguish between these cases, leading to false positives.
- **Lack of temporal verification:** Real tunnels are persistent, while network anomalies are temporary. Current tools do not check whether detected patterns repeat over time, making the two difficult to distinguish.

III. METHODOLOGY

A. System Overview

Our system detects hidden MPLS tunnels entirely from outside the network. It requires only a destination IP address and no access to routers or topology information.

The pipeline consists of six steps:

1. Probe the path using traceroute-style packets.
2. Analyze the replies for signs of a hidden MPLS tunnel.
3. Repeat steps 1 and 2 five times and check whether the same pattern appears every time.
4. Assign each hop on the path a score based on how suspicious it looks, then identify where a tunnel begins and ends.
5. Optionally re-probe any silent hops at a slower rate to determine whether the silence is caused by a tunnel or simply by a router ignoring ICMP traffic.
6. Optionally compare the result against a previous detection run to confirm the tunnel has persisted over time.

Each step is described in detail below.

B. Flow-Consistent Multi-Run Probing

We send UDP probes with increasing TTL values (default: 30 hops), probing each hop three times.

To ensure all probes follow the same path, we use Paris-style probing, keeping packet fields constant. This avoids errors caused by load balancing.

From each ICMP response, we extract:

- **Quoted TTL:** Used to compute the *quoted-TTL deficit*, which indicates hidden hops.

- **MPLS label stack (RFC 4950):** Direct evidence of MPLS when present.

Each measurement is repeated multiple times (default: 5 runs) to enable consistency checks.

C. MPLS Signature Detection

Each run is analyzed using four checks:

1. **TTL transparency violation:**
   The same router responding to consecutive TTL values indicates a uniform-mode tunnel.
2. **Silent block detection:**
   Consecutive non-responding hops suggest hidden routers but are not conclusive alone.
3. **Quoted-TTL deficit:**
   A lower-than-expected quoted TTL indicates hidden hops (pipe mode).
4. **Compound signal:**
   A silent block combined with a TTL deficit strongly indicates a hidden tunnel.

D. Multi-Run Consistency

Results are compared across runs to determine if patterns repeat.

- Persistent patterns indicate real tunnels.
- Changing patterns suggest transient effects or load balancing.

This check is important because a single unusual measurement can look like a tunnel without being one. A real hidden MPLS tunnel produces the same signature every time we probe, because the label-forwarding tables in the routers do not change between runs.

A consistency threshold (e.g., ≥80%) is required for confirmation. This step helps us distinguish MPLS tunnels from ECMP behavior.

E. Scoring and Classification

Each hop is assigned a score (0.0–1.0) based on detected signals. **Signals are not simply added together.** If they were, a collection of weak signals could accumulate into a falsely high score. Instead, each signal raises the hop's score only if that signal's weight is higher than the current score. This means a sequence of weak signals can never exceed the weight of the weakest signal in the sequence — only a genuinely strong signal can push the score into the high-confidence range.

Each hop's final score maps to one of four confidence labels:

- **CONFIRMED** (≥0.9)
- **PROBABLE** (≥0.6)
- **POSSIBLE** (≥0.3)
- **UNLIKELY** (<0.3)

Using these scores, the system identifies tunnel entry, interior, and exit points. Table I lists all signals and their weight.

Table I — Detection Signal Weights

Signal	Weight	Why this weight
TTL transparency violation	1.0	Mathematically impossible in pure IP
RFC 4950 label stack confirmed	1.0	Direct router-reported evidence
Compound: deficit + silent block	1.0	Both signals together, near-certain
Quoted-TTL deficit ≥ 2	0.95	Multiple hidden hops confirmed
Silent block (3+ hops)	0.90	Very strong with consistency
RTT stable + near silent block	0.80	Compound condition
Silent block (2 hops)	0.70	Strong
Quoted-TTL deficit = 1	0.70	Treated with caution without FRPLA baseline
ASN continuity	0.50	Supporting signal only
Backbone hostname keyword	0.30	Fragile heuristic
Generic interface hostname	0.20	Fragile heuristic
RTT stability alone	0.10	Never sufficient alone.

F. ICMP Rate-Limit Discrimination

Silent hops are re-probed at a slower rate to distinguish tunneling from ICMP rate-limiting.

- If the hop **responds** to any slow probe — it was rate-limiting. The burst of fast probes drained its bucket, but the slow probes arrived after it had refilled. We reduce that hop's MPLS score by 0.5 to prevent it from contributing to a false positive.
- If the hop **stays silent** across all slow probes — rate limiting is ruled out. The silence is consistent with a genuine tunnel, and the hop's score is unchanged.

Because this step adds roughly six seconds per silent hop, it does not run automatically. It is invoked when the user wants higher confidence before acting on a detection result.

G. Temporal Correlation

A single measurement session cannot tell the difference between a real tunnel and a one-time anomaly that happened to look like one.

Our temporal correlation module addresses this by comparing the current detection result against one saved from an earlier session. For each hop position that appears in both results, we check whether the same signals fired, the same IP addresses responded, and the same silent block and deficit sizes were observed. We calculate a **temporal consistency score** — the fraction of hop positions where the pattern matched between sessions.

- Persistent patterns strengthen confidence in a tunnel.
- Non-repeating patterns suggest temporary anomalies.

H. Pseudocode/Algorithm

```
ALGORITHM: DetectHiddenMPLSTunnel(destination, runs=5)

PHASE 1: PROBING

flow_id ← random integer (fixed for all runs on this destination)

FOR run = 1 TO runs:
    FOR ttl = 1 TO MAX_HOPS:
        Send 3 UDP probes to destination with this TTL and flow_id
        IF ICMP Time Exceeded received:
            Record responding IP, RTT values
            Extract quoted_ttl from inner IP header of ICMP reply
            deficit ← (ttl - 1) - quoted_ttl
            Parse RFC 4950 extension objects for MPLS label stack
        ELSE:
            Record hop as non-responding (silent)
    END FOR
    Store complete hop sequence as trace[run]
END FOR

PHASE 2: PER-RUN ANALYSIS

FOR each trace[run]:

    PASS 1 — TTL Transparency Violation:
        FOR each hop in trace:
            IF same IP responds to 2+ consecutive TTL values:
                Record violation (HIGH if ≥3, MEDIUM if =2)

    PASS 2 — Silent Block Detection:
        Find longest sequence of consecutive non-responding hops
        IF length ≥ 2:
            Record silent_block (start_ttl, end_ttl, size, egress_ip)

    PASS 3 — Quoted-TTL Deficit:
        FOR each responding hop:
            IF deficit > 0:
                Record finding (HIGH if deficit ≥ 2, MEDIUM if = 1)

    PASS 4 — Compound Flag:
        IF silent_block AND deficit_found on same path:
            Raise compound_flag

END FOR

PHASE 3: MULTI-RUN CONSISTENCY

silent_sizes ← [silent_block.size for each run]
deficits ← [max_deficit for each run]

consistency_score ← fraction of runs with matching silent_block size
deficit_consistency ← fraction of runs with matching deficit value

IF consistency_score ≥ 0.8 AND deficit_consistency ≥ 0.8:
    compound_confirmed ← TRUE ← strongest possible finding
```

```
    ELSE IF consistency_score ≥ 0.8:
        silent_block_confirmed ← TRUE
    ELSE IF varying patterns across runs:
        ECMP_likely ← TRUE          ← not a tunnel


PHASE 4: SCORING

    FOR each hop in representative trace (run 1):

        score ← 0.0
        signals ← []

        IF hop falls within a TTL transparency violation:
            score ← max(score, 1.0)

        IF hop has RFC 4950 MPLS label stack:
            score ← max(score, 1.0)

        IF compound_confirmed AND hop has deficit > 0:
            score ← max(score, 1.0)
        ELSE IF hop.deficit ≥ 2:
            score ← max(score, 0.95)
        ELSE IF hop.deficit = 1:
            score ← max(score, 0.70)

        IF hop is non-responding AND silent_block.size ≥ 3:
            score ← max(score, 0.90)
        ELSE IF hop is non-responding AND silent_block.size = 2:
            score ← max(score, 0.70)

        IF RTT is stable AND hop is near silent block:
            score ← max(score, 0.80)

        IF hostname contains backbone keywords AND other signals fired:
            score ← min(1.0, score + 0.3)

        label ← CONFIRMED  if score ≥ 0.9
                PROBABLE   if score ≥ 0.6
                POSSIBLE   if score ≥ 0.3
                UNLIKELY   otherwise

        Append (hop, score, label, signals) to scored_hops

    Classify hop types:
        First PROBABLE or above → ENTRY
        Subsequent silent or suspicious hops → IN-TUNNEL
        First normal hop after tunnel → EXIT

    tunnel_detected ← any hop classified as ENTRY or IN-TUNNEL


PHASE 5: RATE-LIMIT DISCRIMINATION

    FOR each silent hop:
        FOR probe = 1 TO 3 (with 2-second wait between each):
            Send single UDP probe at this TTL
            IF response received:
                verdict ← RATE_LIMITED
                Reduce hop score by 0.5
                BREAK
        IF no response across all slow probes:
            verdict ← CONSISTENT WITH TUNNEL


PHASE 6: TEMPORAL CORRELATION

    Load previous_result from saved JSON file

    FOR each TTL position in both current and previous result:
        Compare: responding IP, signals fired, score label
        IF both match → consistent_hop_count += 1

    temporal_score ← consistent_hops / total_comparable_hops

    IF tunnel in both runs AND silent_block consistent AND deficit consistent:
        temporal_confidence ← CONFIRMED
    ELSE IF tunnel in one run only:
        temporal_confidence ← POSSIBLE (may be transient)


OUTPUT

    RETURN {
        tunnel_detected,
        overall_confidence (CONFIRMED / PROBABLE / POSSIBLE / UNLIKELY),
        tunnel_entry_ttl, tunnel_exit_ttl,
        per-hop scores and labels,
        temporal_confidence (if run),
        rate_check_verdicts (if run)
    }
```

IV. Implementation & Results

A. Architecture

The detection system is implemented in Python 3 using only the standard library. It relies on raw sockets to send

UDP probes and receive ICMP replies, so it must be run with root privileges.

The system is organized into separate modules, each handling one stage of the detection pipeline. These include probing, signature analysis, scoring, rate-limit checking, temporal comparison, and validation. Results from each run are saved automatically as timestamped JSON files, making it easy to compare outputs across scenarios or over time. Table II provides a more detailed summary on each module and its role.

Table II — Module Summary

Module	Responsibility
probe.py	Sends probes, extracts quoted TTL, parses RFC 4950 label stacks
ttl_transparency.py	Runs the four detection passes, multi-run consistency check
scoring.py	Assigns per-hop scores, classifies ENTRY / IN-TUNNEL / EXIT
detector.py	Orchestrates all modules, supports multiple destinations and IPv6
rate_check.py	Re-probes silent hops at slow rate for ICMP rate-limit discrimination
temporal.py	Compares current result against a saved previous run
validator.py	Measures TPR, FPR, FNR against known ground truth scenarios

B. Testbed Design

Evaluation was performed using Kathara, an open-source network emulation platform that runs each device as a Linux container. This allowed full control over topology and router behavior in a repeatable environment.

Three scenarios were created:

- **Scenario A: Hidden pipe-mode tunnel.** Two intermediate routers were configured to suppress ICMP Time Exceeded replies, simulating hidden MPLS forwarding.
- **Scenario B: ECMP load balancing.** Two equal-cost paths were configured without MPLS to test whether the system avoids false positives under path variation.
- **Scenario C: Clean IP path.** A simple visible path with no tunneling or load balancing served as the baseline case.

Each scenario was tested with five runs, three UDP probes per TTL, and a maximum path length of 30 hops.

C. Testing Results

Scenario A: Hidden Tunnel

The system correctly detected the hidden tunnel. Two consecutive silent hops appeared consistently across all five runs, followed by an egress router with a quoted-TTL deficit of 2. This matched the two hidden routers configured in the topology.

The silent block and deficit were both stable across runs, producing the strongest system verdict: **MPLS CONFIRMED**. The deficit signal also showed the number of hidden hops, which the silent block alone could not provide.

Scenario B: ECMP Load Balancing

The system correctly did not classify this scenario as a tunnel. Hop patterns changed across runs, causing the consistency score to fall below the detection threshold. This behavior matched ECMP load balancing rather than a persistent hidden tunnel.

Scenario C: Clean IP Path

The system correctly reported no tunnel. All hops responded normally, with no silent block and no quoted-TTL deficit. No false positive was produced.

D. Validation Metrics

Across the three scenarios, the system produced one true positive and two true negatives, with no false positives or false negatives. This corresponds to:

- **True Positive Rate:** 100%
- **False Positive Rate:** 0%
- **False Negative Rate:** 0%

These results show that the system correctly detected the hidden tunnel while avoiding false alarms in non-tunnel cases.

E. Contribution of Novel Signals

This work introduces two important signals beyond silent-hop detection: the quoted-TTL deficit and RFC 4950 MPLS label reporting.

In the emulated testbed, RFC 4950 label stacks were not observed because the simulated environment does not generate them the way real MPLS routers can. The quoted-TTL deficit, however, was clearly visible in the hidden tunnel scenario. It strengthened the result from **PROBABLE** to **CONFIRMED** and revealed the number of hidden hops.

This is important because silent hops alone suggest a tunnel, but they do not prove one. The deficit provides stronger evidence and improves confidence. In real networks, RFC 4950 support would provide direct confirmation wherever available.

V. DISCUSSIONS

This work addresses a central question: can hidden MPLS tunnels be reliably detected from outside the network? Results from controlled experiments suggest that they can, and that two key limitations in existing approaches can be resolved with targeted improvements to measurement and analysis.

Contributions

The most significant contribution is **ICMP rate-limit discrimination**. Silent hops are ambiguous: they may indicate either a tunnel or a router limiting ICMP responses. Existing tools do not distinguish between these cases, leading to false positives. Our approach resolves this by re-probing at a slower rate. Because rate limiting is temporary while tunnel silence is persistent, the system can classify the cause of silence directly.

The second contribution is **temporal correlation**. Detection is not a one-time task—defenders must know whether a pattern persists. By comparing results across independent runs, the system distinguishes stable tunnel signatures from transient network anomalies.

The **compound scoring model** further improves reliability. Instead of combining many weak signals, strong signals dominate the final decision. This prevents weak evidence from producing high-confidence false positives and better reflects the true strength of each signal.

Finally, the **quoted-TTL deficit** extends pipe-mode detection. Unlike prior approaches that rely on calibration datasets, this method extracts the signal directly from ICMP responses. It enables detection in environments where baseline data is unavailable and provides insight into the number of hidden hops.

Position in the Detection Landscape

This system is designed to complement existing tools such as Scamper and TNT. Scamper provides reliable large-scale measurement, while TNT performs established MPLS analysis. Our approach adds two missing capabilities: distinguishing rate-limited routers from tunneled ones, and verifying that detected patterns persist over time.

In practice, these components can be used as a second-stage analysis layer. After initial detection, the system can confirm results, reduce false positives, and track tunnel behavior across repeated measurements.

VI. CONCLUSION

This paper presented an external detection system for hidden Multiprotocol Label Switching (MPLS) tunnels that addresses key limitations in existing approaches. By introducing ICMP rate-limit discrimination and temporal correlation, the system distinguishes tunnel behavior from normal network effects and verifies whether detected patterns persist over time. Combined with a compound scoring model, this produces more reliable and interpretable detection outcomes.

Evaluation in a controlled emulated environment demonstrated correct detection of a hidden tunnel and no false positives in non-tunnel scenarios, achieving a True Positive Rate of 100% and a False Positive Rate of 0%. Beyond detection, the system provides insight into tunnel structure by identifying affected hops and quantifying hidden segments.

Overall, this work contributes a practical and reproducible approach for improving network visibility and strengthening measurement-based MPLS tunnel detection.

VII. FUTURE WORK

A. Validation on real MPLS infrastructure.
The most important next step is evaluation on physical MPLS routers or high-fidelity software routers such as **FRRouting**. This would allow validation of RFC 4950 label stack behavior and confirm detection accuracy under real network conditions.

B. Integration with existing tools.
The system can be integrated with tools such as Scamper and TNT as a second-stage analysis layer. In this role, it would refine initial detections,

reduce false positives through rate-limit discrimination, and track tunnel persistence over time.

C. **Expanded test coverage.**
Future evaluations should include more complex scenarios, such as deeper tunnels, partial tunnels affecting only some flows, and IPv6-based MPLS deployments. This would improve confidence in the system's robustness across diverse network conditions.

VIII. ACKNOWLEDGEMENT

The author would like to thank Dr. Khaled Sabehein for their guidance and supervision throughout this project.

We would also like to thank the Mathematics, Computer, and Information Sciences Department at Mississippi Valley State University for funding this research.

IX. REFERENCES

[1] Eric Rosen, Arun Viswanathan, and Ross Callon, "Multiprotocol Label Switching Architecture," RFC 3031, Internet Engineering Task Force, Jan. 2001. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3031

[2] Eric Rosen, Daniel Tappan, George Fedorkow, Yakov Rekhter, Dino Farinacci, Tony Li, and Adrian Conta, "MPLS Label Stack Encoding," RFC 3032, Internet Engineering Task Force, Jan. 2001. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3032

[3] Lars Andersson, Iain Minei, and Bob Thomas, "LDP Specification," RFC 5036, Internet Engineering Task Force, Oct. 2007. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5036

[4] Jon Postel, "Internet Protocol," RFC 791, Internet Engineering Task Force, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/rfc/rfc791

[5] Jon Postel, "Internet Control Message Protocol," RFC 792, Internet Engineering Task Force, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/rfc/rfc792

[6] Ron Bonica, David Gan, Daniel Tappan, and Carlos Pignataro, "ICMP Extensions for Multiprotocol Label Switching," RFC 4950, Internet Engineering Task Force, Aug. 2007. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4950

[7] Brice Augustin, Xavier Cuvellier, Benoit Orgogozo, Franck Viger, Timothy Friedman, Matthieu Latapy, Celine Magnien, and Ruben Teixido, "Avoiding Traceroute Anomalies with Paris Traceroute," in *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC)*, Rio de Janeiro, Brazil, Oct. 2006, pp. 153–158.

[8] Matthew Luckie, "Scamper: A Scalable and Extensible Packet Prober for Active Measurement of the Internet," in *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC)*, Melbourne, Australia, Nov. 2010, pp. 239–245.