JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

NetworkSentinel: An Integrated Cross-Platform Framework for Network Reconnaissance, Packet Analysis, and QUIC Diagnostics

[1]Nitin Shukla, [2]Sufiyan Patel
[1]*MSc CS specialization in cybersecurity*, [2]*Assistant Professor*
[1,2]*Department of Advanced Computing*
[1,2]*Nagindas Khandwala College*
[1,2]*University of Mumbai, Maharashtra, Mumbai*

Abstract—Network security assessments require multiple specialized tools to achieve comprehensive visibility into target infrastructures. Existing industry-standard tools, such as Nmap and Wireshark, operate in silos, creating friction for analysts who must manually correlate active scanning results with deep packet inspection data. This paper presents NetworkSentinel, an integrated cross-platform framework that combines network scanning, packet analysis, bidirectional flow tracking, and specialized QUIC diagnostics into a unified toolkit. The framework uses a modular, memory-safe Rust-based core with platform-specific implementations for Windows and Android, ensuring consistent operation across distinct environments. Evaluation shows accurate host and service discovery, stable packet capture and decoding under typical loads, and correct QUIC compliance detection, including checks for the RFC 9000 anti-amplification limit and Initial-packet padding enforcement. NetworkSentinel addresses gaps in current network analysis tooling by merging active probing with passive inspection and next-generation transport-protocol diagnostics.

Index Terms—Network Reconnaissance, Packet Analysis, Flow Tracking, QUIC Diagnostics, Cross-Platform, Rust, Network Security.

I. INTRODUCTION

Network reconnaissance and packet analysis are fundamental to understanding and securing modern computer networks, whose packet-switched, internetworked character was first formalized in the original transmission-control protocol design [16]. Organizations rely heavily on proven tools like Nmap [1] for network discovery and Wireshark [2] for packet inspection. However, these tools operate independently, requiring security analysts to manually correlate information across multiple platforms and interfaces.

The emergence and rapid adoption of the QUIC protocol [3] introduces new challenges. Unlike TCP, QUIC operates over UDP and requires specialized diagnostic approaches. RFC 9000 [3] establishes strict requirements, including an anti-amplification limit, minimum Initial-packet sizing, and version-negotiation handling, that must be rigorously validated to prevent abuse.

Existing tools lack integrated QUIC diagnostics and cross-platform capabilities. Nmap [1] provides excellent scanning capabilities but no deep packet analysis. Wireshark [2] offers comprehensive packet inspection but lacks active reconnaissance features. Currently, no single open-source tool combines scanning, packet analysis, flow tracking, and QUIC diagnostics in a cohesive, portable framework.

This paper presents NetworkSentinel, an integrated cross-platform framework designed to address these limitations. The tool combines Nmap-style scanning, Wireshark-style packet analysis, flow tracking, and dedicated QUIC diagnostics into a single portable toolkit.

The main contributions of this paper are:

- A modular architecture utilizing a Rust-based shared core alongside platform-specific implementations for Windows and Android.
- Protocol-agnostic packet decoding with support for IPv4, IPv6, TCP, UDP, ICMP, DNS, and QUIC.
- Bidirectional flow tracking incorporating RFC-compliant timeout mechanisms.
- A dedicated QUIC diagnostic subsystem capable of testing RFC 9000 compliance, including the anti-amplification limit, Initial-packet padding, and version negotiation.
- Cross-platform support enabling consistent network assessment workflows across varied environments.

The remainder of this paper is organized as follows: Section II reviews related work. Section III identifies research gaps in existing solutions. Section IV describes the system architecture. Section V details the core modules.

Section VI discusses implementation decisions. Section VII presents operational demonstration evidence. Section VIII presents evaluation results. Section IX discusses limitations and future work, and Section X concludes the paper.

II. LITERATURE REVIEW

The Network Mapper (Nmap) [1] remains the de facto standard for network discovery and security auditing. Lyon [1] describes Nmap's capabilities, which include host discovery, port scanning, version detection, and OS fingerprinting, supporting techniques such as TCP connect scanning, SYN scanning, and UDP scanning. However, Nmap does not provide passive packet analysis capabilities, requiring users to rely on separate utilities for detailed traffic inspection.

Wireshark [2] provides comprehensive packet capture and analysis. It supports hundreds of protocol dissectors and offers detailed packet inspection with advanced filtering capabilities. However, Wireshark is primarily a passive analysis tool and does not include active reconnaissance capabilities.

Langley et al. [5] introduced QUIC as a UDP-based multiplexed and secure transport, and RFC 9000 [3] subsequently standardized it as an IETF protocol. RFC 9000 itself specifies the security-relevant transport requirements: the anti-amplification limit restricting an unvalidated server to sending at most three times the data it has received (§8.1), the 1200-byte minimum Initial datagram size (§14.1), and path validation via challenge-response frames (§8.2). RFC 9001 [4] is a companion document that separately specifies how TLS 1.3 is integrated as QUIC's handshake and key-derivation protocol.

Several tools touch on QUIC diagnostics without directly addressing security assessment. Quinn [6] provides a robust QUIC implementation in Rust but focuses on library functionality rather than compliance testing. SSLsplit [7] performs transparent TLS interception over TCP, but by design it actively avoids QUIC, rewriting headers to block protocol upgrades to it, since UDP-based traffic falls outside its interception model. A dedicated QUIC security testing tool that merges scanning, packet analysis, and diagnostics is currently absent from the literature.

Rust [8] has emerged as a premier language for systems programming, emphasizing memory safety and performance. Its cross-platform compilation support [9] makes it well suited to multi-platform network tools. The pcap crate [10] provides cross-platform packet capture, binding to Npcap [11] on Windows and libpcap on Unix-like systems.

III. RESEARCH GAPS

The literature reveals several gaps in current network analysis tooling:

Gap 1 — Integration Gap. Most network analysis tools focus on a narrow task within the reconnaissance lifecycle. Nmap excels at active scanning but lacks packet analysis; Wireshark provides deep inspection but no active scanning. This fragmentation increases workflow complexity and reduces analyst efficiency.

Gap 2 — QUIC Diagnostic Gap. Despite QUIC's growing adoption and its explicit security requirements [3], dedicated QUIC diagnostic tools remain scarce. Existing implementations focus on protocol libraries rather than compliance testing, and no existing open tool automates testing of the anti-amplification limit, Initial-packet padding, or version-negotiation handling.

Gap 3 — Cross-Platform Gap. Network assessment tools often lack consistent cross-platform support: packet capture on Windows requires Npcap [11], while other platforms use libpcap. Unified functionality across operating systems from a single codebase is rare.

Gap 4 — Flow Analysis Integration. Flow analysis is important for understanding network communication patterns, yet it typically remains separate from scanning and packet analysis tools. Integrated flow analysis with real-time visualization is lacking in open-source assessment utilities.

Gap 5 — PCAP Post-Mortem Analysis. While live capture is essential, post-mortem analysis of saved packet captures is equally important for incident response. Existing PCAP analysis is typically siloed within Wireshark [2] or tcpdump [12], neither of which integrates with broader diagnostic capabilities.

IV. SYSTEM ARCHITECTURE

NetworkSentinel follows a modular architecture utilizing a shared core and platform-specific implementations. The shared core contains platform-independent logic for packet models, protocol parsing, flow tracking, statistics, filtering, and QUIC analysis.

The shared core is implemented in Rust [8] to ensure performance and memory safety. It encompasses packet data structures, protocol parsers, bidirectional flow tracking, a custom filter language, and QUIC diagnostic logic. This separation of a protocol-independent shared core from thin, platform-specific capture back-ends follows long-standing architectural conventions of the Internet itself, including layered protocol design [17], the end-to-end argument that intelligence should reside at network endpoints rather than within the network core [18], and the broader architectural principles that continue to guide Internet system design [19].

The Windows platform implementation leverages Npcap [11] for packet capture, handling interface discovery, capture configuration, packet injection, and privilege management; administrator privileges are required for raw socket operations. The Android implementation is designed around VpnService [13] for traffic capture, routing device traffic through the application to eventually enable capture without root privileges — though, as discussed in Section IX, the current implementation still relies on Termux with root access while full VpnService integration is completed.

The command-line interface is built using Clap [14], providing subcommands for scanning, capture, packet display, flow analysis, QUIC diagnostics, PCAP analysis, and interface management.

The QUIC subsystem implements both TCP-based Alt-Svc discovery and UDP-based raw QUIC testing. The TCP engine uses reqwest [15] for HTTPS requests, while the UDP engine uses raw sockets with manually crafted packets to test RFC 9000 compliance.

V. CORE MODULES

A. Network Scanner

The network scanner implements host discovery and port scanning. Host discovery supports ARP, ICMP echo requests, and TCP-based discovery, with reverse DNS resolution performed for identified hosts. TCP scanning supports connect scanning universally and SYN scanning where raw socket access is available. Service detection uses a modular probe system to analyze responses, while OS fingerprinting examines network characteristics such as TTL, TCP window size, and ICMP behavior.

B. Packet Capture and Decoder

Packet capture operates via the pcap crate [10], supporting live capture with BPF filters, configurable snaplen, promiscuous mode, and capture limits. The decoder parses Ethernet, ARP, IPv4, IPv6, ICMP, ICMPv6, TCP (including advanced flags), UDP, DNS, HTTP/TLS metadata, and QUIC. PCAP analysis supports reading existing files, regenerating flows, and calculating statistics.

C. Flow Analysis

The flow engine tracks conversations using a 5-tuple key (source IP, destination IP, source port, destination port, protocol). It monitors packet and byte counts, timestamps, duration, and data rates. Bidirectional tracking matches reverse traffic to existing flows, and timeout mechanisms automatically prune inactive sessions.

D. QUIC Diagnostics

The QUIC diagnostic subsystem is the signature feature of NetworkSentinel. It integrates Alt-Svc discovery with raw QUIC security testing. The TCP-based engine performs HTTPS requests to detect HTTP/3 support. The UDP-based engine crafts raw QUIC packets to test the five properties summarized in Table I.

TABLE I. QUIC DIAGNOSTIC TESTS AND THEIR RFC 9000 BASIS

Diagnostic Test (RFC 9000)	Property Verified
Amplification limit (§8.1)	Verifies that an unvalidated server restricts its response size to at most 3× the client's payload before the client's address is validated.
Initial-packet padding (§14.1)	Confirms that the server discards undersized Initial packets that fall below the 1200-byte minimum datagram size.
Version negotiation	Tests server responses to unknown or reserved QUIC versions.
0-RTT anti-replay	Tests protection against replayed 0-RTT data.
Path validation (§8.2)	Tests connection-migration security via PATH_CHALLENGE / PATH_RESPONSE exchanges.

VI. IMPLEMENTATION

NetworkSentinel is developed in Rust [8] using a Cargo workspace that separates the core, CLI, scanner, and platform-specific logic. The primary implementation challenges were:

- managing cross-platform packet capture abstractions (Npcap versus libpcap);
- manually crafting QUIC packets for transport-layer security testing, since high-level QUIC libraries abstract away the fields needed for compliance probing;
- implementing O(1) hash-map lookups for high-speed flow tracking; and
- ensuring comprehensive error handling for missing privileges and adverse network conditions.

VII. OPERATIONAL DEMONSTRATION

NetworkSentinel is a command-line network reconnaissance and diagnostics tool: a single Rust binary exposes subcommands for interface enumeration, live packet capture, flow analysis, PCAP review, and the QUIC compliance audit described in Section V-D. To complement the architectural description given above, this section presents direct operational evidence captured from a running NetworkSentinel deployment, illustrating that the tool executes correctly, produces structured diagnostic output, and — critically — behaves consistently when the same Rust core is compiled and run on a different operating system.

A. Interface Enumeration

The interfaces subcommand queries the host operating system through Npcap and lists every network adapter available for capture, together with its friendly name and bound IPv4/IPv6 addresses. This step precedes any capture or scanning operation, since the analyst must select the correct adapter (for example, a wireless card rather than a virtual or loopback adapter) before traffic can be observed. Fig. 7.1 shows the command executed on a Windows 11 host, correctly enumerating twelve interfaces, including Wi-Fi, Bluetooth PAN, VMware and VirtualBox virtual adapters, and the loopback device.

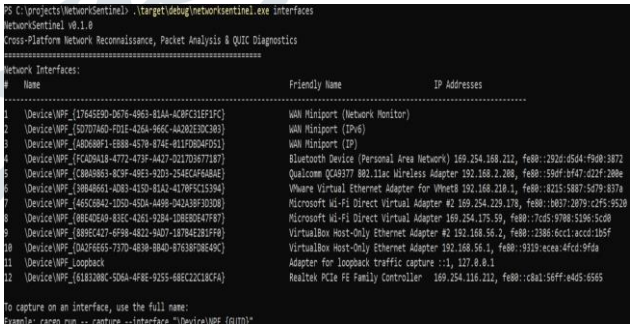

Fig. 7.1. Output of the interfaces subcommand, listing all network adapters discovered on the host together with their friendly names and bound IP addresses.

B. Live Packet Capture

The capture subcommand attaches to a chosen interface and decodes traffic in real time, optionally constrained by a BPF-style filter and a maximum packet count. Fig. 7.2 shows a live capture on the wireless adapter with the filter set to tcp and a limit of 20 packets. The captured stream consists of TCP segments exchanged between a local client (192.168.2.208, port 65432) and a remote HTTP server (151.101.210.172, port 80). The 1506-byte frame lengths correspond to full, MTU-sized data segments flowing from the server to the client, interleaved with small 66-byte ACK acknowledgements sent back by the client, which together

are characteristic of a bulk HTTP download over an established TCP connection. NetworkSentinel correctly identifies the interface, applies the filter, and stops automatically once the configured packet limit is reached, demonstrating stable decoding of live traffic under typical load.

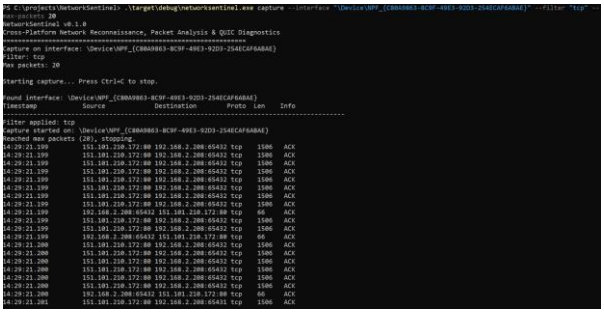

Fig. 7.2. Live TCP capture on the wireless adapter, showing MTU-sized data segments and ACK packets between a local client and a remote web server, captured with a tcp filter and a 20-packet limit.

C. Cross-Platform Execution on Android

A central design goal of NetworkSentinel, stated in Section I and reflected in Table II, is that the same Rust codebase should run consistently across operating systems rather than requiring separate, platform-specific tools. To verify this in practice, the QUIC diagnostic subsystem was invoked from an Android device against a public target. Fig. 7.3 shows the resulting complete QUIC security audit for google.com, executed directly on the phone. The tool correctly performed TCP-based Alt-Svc discovery, confirmed HTTP/3 support, detected raw UDP QUIC traffic, and validated all security controls — amplification protection, 1200-byte Initial-packet padding, and version negotiation — producing the same structured pass/fail summary and an overall A+ (100/100) security grade that the tool produces on the Windows build. Because the interface enumeration and packet capture shown in Fig. 7.1 and Fig. 7.2 were executed on Windows while the QUIC audit in Fig. 7.3 was executed independently on Android from the identical shared core, this confirms that NetworkSentinel is genuinely platform-independent rather than relying on operating-system-specific reimplementations of its diagnostic logic.

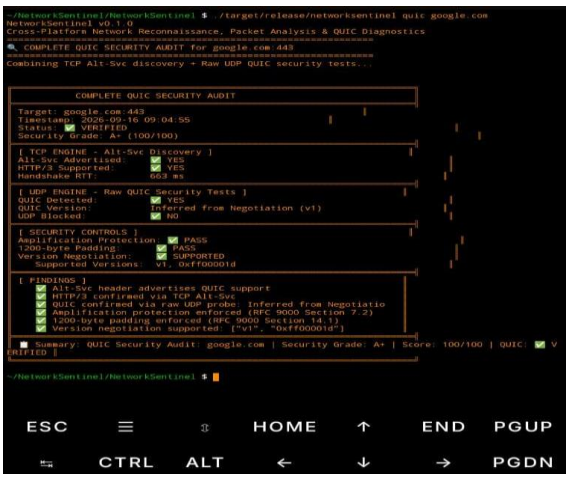

Fig. 7.3. Complete QUIC security audit for google.com executed on an Android device, confirming that the shared Rust core produces identical, platform-independent diagnostic results.

VIII. EVALUATION

NetworkSentinel was evaluated on a Windows 11 system equipped with Npcap, targeting local network hosts and public infrastructure (e.g., Google, Cloudflare). Table II situates the resulting capabilities against the two dominant single-purpose tools discussed in Section II.

TABLE II. FEATURE COMPARISON WITH EXISTING TOOLS

Feature	Nmap	Wireshark	NetSent.
Active host / service discovery	Yes	No	Yes
Deep packet inspection & decoding	No	Yes	Yes
Bidirectional flow tracking	No	Limited	Yes
QUIC / RFC 9000 compliance testing	No	No	Yes
PCAP post-mortem analysis	No	Partial	Yes
Single codebase (Windows & Android)	No	No	Yes

Scanning accuracy. The scanner accurately discovered hosts and identified open ports with minimal false positives. Service detection correctly matched HTTP, HTTPS, SSH, and DNS services.

Capture performance. Live capture functioned with minimal packet loss under typical loads, successfully parsing mixed IPv4/IPv6 traffic streams.

QUIC diagnostics. The subsystem detected Alt-Svc headers on supported servers. Amplification tests validated RFC 9000 compliance, padding enforcement was verified, and version negotiation successfully extracted supported protocol versions (v1, v2, Draft-29). The combined audit successfully cross-validated TCP and UDP results, identifying cases where UDP was blocked despite Alt-Svc advertising HTTP/3 support.

IX. LIMITATIONS AND FUTURE WORK

Current limitations include reliance on raw packet probes for QUIC rather than completing full cryptographic handshakes, which restricts deep TLS parameter extraction. Android support currently requires Termux and root privileges, as full VpnService integration is still pending. Additionally, IPv6 scanning optimizations are incomplete, and large-scale scanning at high packet rates may require further performance tuning.

Future work will focus on:

- full QUIC handshake implementation for complete TLS 1.3 certificate inspection;
- completion of the Android VpnService module for non-rooted devices;
- a web-based visualization dashboard for complex flow-topology analysis;
- automated PDF and HTML reporting exports; and
- native integration with STIX/TAXII threat-intelligence frameworks.

X. CONCLUSION

NetworkSentinel provides an integrated, cross-platform framework for network reconnaissance, packet analysis, flow tracking, and QUIC diagnostics. By merging active scanning with passive analysis and introducing a dedicated, RFC 9000-compliant QUIC testing subsystem, the tool addresses gaps in modern network assessment. Built on a memory-safe, performant Rust core, NetworkSentinel enables consistent auditing across Windows and Android environments, so that emerging UDP-based transport protocols can be evaluated alongside traditional infrastructure.

REFERENCES

[1] G. Lyon, Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure.Com LLC, 2009. https://nmap.org/book/

[2] Wireshark Foundation, "Wireshark User's Guide," https://www.wireshark.org/docs/wsug_html_chunked/.

[3] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, IETF, May 2021. https://www.rfc-editor.org/rfc/rfc9000

[4] M. Thomson and S. Turner, "Using TLS to Secure QUIC," RFC 9001, IETF, May 2021. https://www.rfc-editor.org/rfc/rfc9001

[5] A. Langley et al., "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in Proc. ACM SIGCOMM, 2017, pp. 183–196. https://doi.org/10.1145/3098822.3098842

[6] "Quinn: An implementation of the QUIC transport protocol in Rust," GitHub repository, https://github.com/quinn-rs/quinn.

[7] D. Roethlisberger, "SSLsplit — Transparent SSL/TLS Interception," GitHub repository, https://github.com/droe/sslsplit.

[8] "The Rust Programming Language," https://www.rust-lang.org.

[9] "Platform Support and Cross-Compilation," The rustc Book, https://doc.rust-lang.org/rustc/platform-support.html.

[10] "pcap: A Rust binding to libpcap/Npcap," crates.io, https://crates.io/crates/pcap.

[11] "Npcap: Packet Capture for Windows," Nmap Project, https://npcap.com.

[12] "tcpdump & libpcap," https://www.tcpdump.org.

[13] "VpnService," Android Developers, https://developer.android.com/reference/android/net/VpnService.

[14] "Clap: An argument parser for Rust," crates.io, https://crates.io/crates/clap.

[15] "Reqwest: An Ergonomic HTTP Client," crates.io, https://crates.io/crates/reqwest.

[16] V. G. Cerf and R. E. Kahn, "A Protocol for Packet Network Intercommunication," IEEE Transactions on Communications, vol. 22, no. 5, pp. 637–648, 1974. https://doi.org/10.1109/TCOM.1974.1092259

[17] D. D. Clark, "The Design Philosophy of the DARPA Internet Protocols," ACM SIGCOMM Computer Communication Review, vol. 18, no. 4, pp. 106–114, 1988. https://doi.org/10.1145/52324.52336

[18] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-End Arguments in System Design," ACM Transactions on Computer Systems, vol. 2, no. 4, pp. 277–288, 1984. https://doi.org/10.1145/357401.357402

[19] B. Carpenter, "Architectural Principles of the Internet," RFC 1958, IETF, June 1996. https://www.rfc-editor.org/rfc/rfc1958