TLSVerifier: An Automated TLS software Verification Tool Using Model-Based Testing*

Jaedeok Lim† and Minsuk Choi
Cybersecurity Research Division
Electronic and Telecommunication Research Institute, South Korea.
{jdscol92, yerocker}@etri.re.kr

Abstract

This paper presents TLSVerifier, an automated verification tool for the TLS protocol, which is widely employed in IoT services. As mission-critical IoT applications continue to expand across various social infrastructures, ensuring both the security and trustworthiness of data transmission and remote control has become indispensable. Among existing approaches, formal-method-based verification - which employs logical or mathematical reasoning – is regarded as the only viable solution to guarantee high assurance. However, its practical adoption has been limited due to the steep learning curve and the need for specialized expertise, and its application has been largely confined to the model(design) level rather than actual implementations. To address these limitations, this paper proposes an automated verification tool that enables users to easily perform high-assurance, formal-method-based verification of TLS implementations without prior knowledge of formal methods. The proposed tool provides both specification conformance verification based on RFC 5246 and RFC 8446, and security verification under the Dolev-Yao adversarial model.

Keyword: TLS formal verification, Automated verification, Model-based testing

1 Development of TLSVerifier

The Transport Layer Security (TLS) protocol (RFC 5246, RFC 8446) is the foundation of secure communication in mission-critical IoT systems such as connected vehicles, factories, and smart grids. Ensuring the correctness and robustness of TLS implementations is therefore vital for system-level trust. Conventional testing methods - including fuzzing and differential black-box testing - are effective in discovering runtime faults but cannot guarantee logical soundness across all protocol paths. In contrast, formal verification offers mathematical assurance but remains difficult to apply to real implementations because of the expertise and modeling effort it requires[1].

TLSVerifier was designed to bridge this gap by combining the rigor of formal reasoning with the accessibility of automated software testing. It integrates Model-Based Testing (MBT) with the Maude rewriting-logic engine to enable automatic, executable verification of TLS implementations. Users select a target library (e.g., OpenSSL, wolfSSL) and choose verification objectives - standard-based conformance or security robustness - without needing prior formal-method knowledge.

The tool employs Maude strategies that encode controlled deviations from the TLS handshake. These strategies intentionally modify, delete, or reorder handshake messages to generate meaningful test cases exposing protocol flaws. The resulting traces are automatically transformed into runnable

* Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. P-2, December 16-18, 2025, Sapporo, Japan.
† Corresponding author

scripts that drive real TLS endpoints and record behavior. This process mitigates the state-space explosion typical in symbolic analysis and delivers software-level verification coverage comparable to formal proof systems[2].

A dedicated graphical interface manages configuration, execution, and visualization. Results are displayed as sequence diagrams, coverage statistics, and violation logs, improving interpretability for engineers and auditors. In addition, a security validation module based on the Dolev–Yao adversarial model evaluates resilience against message interception and modification attacks. The verification framework supports both RFC conformance and security assurance within the same automated pipeline. illustrates the concept of TLSVerifier and a portion of its user interface.

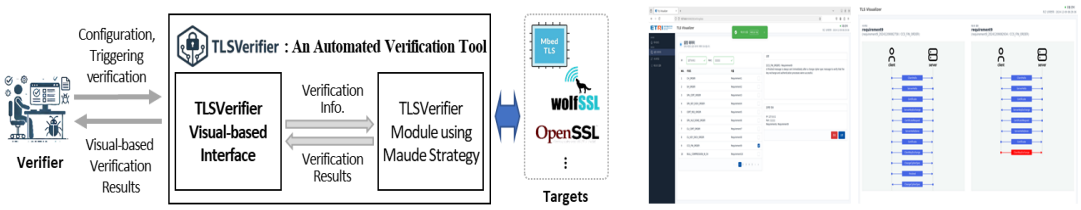

Figure 1: Concept of TLSVerifier(left) and a portion of its user interface(right)

In practical experiments, TLSVerifier encoded ten representative RFC-violation cases, including unsupported cipher-suite negotiation, omission of mandatory extensions, and illegal compression use[3]. Each violation scenario was automatically executed and validated against multiple TLS stacks. The results confirmed the correctness of standard-compliant libraries and revealed weaknesses in modified implementations. These outcomes demonstrate the feasibility of applying formal reasoning directly to deployed security software.

Using TLSVerifier, we also developed the high assurance TLS library, named HASP(High Assurance IoT Security Protocol), for IoT environments. The library was designed and verified through a formal state-model-based approach and successfully validated for interoperability with major TLS stacks, including OpenSSL, wolfSSL, and mbedTLS.

2 Conclusion

TLSVerifier illustrates that formal assurance and industrial automation can be combined in practice. By hiding the complexity of formal methods behind an intuitive workflow, it enables high-confidence verification of real TLS code without expert intervention. The approach significantly reduces verification time, ensures reproducibility, and supports continuous evaluation of security posture.

Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-0-00103, Development of security verification technology against vulnerabilities to assure IoT/IIoT device safety).

Reference

[1] Vaidehi Shah, A Systematic Review of Formal Methods for Reliable Network Testing and Verification. *Technix International Journal for Engineering Research (TIJER)*, Vol. 8, Issue 9, Sept. 2021.

[2] Jeahun Lee and Kyungmin Bae, Formal Specification and Model Checking of TLS Software Security Requirement Using Maude. *Korean Institute of Information Scientists and Engineers*, Vol. 51, No.5, pp. 426-437, May 2024.

[3] Jeahun Lee and Kyungmin Bae, A Scenario Generation for Model-based Testing of TLS using Maude Strategy. Formal Specification and Model Checking of TLS Software Security Requirement Using Maude, *The 9th International Conference on Mobile Internet Security(MobiSec2025)*, submitted.